Recommendation System For Sightseeing Tours

Ricardo Valadas
Instituto Superior Técnico - Universidade Aberta
Lisboa, Portugal
ricardojcvaladas@gmail.com

ABSTRACT

The tourism sector has been launched a continuous improvement mission to its stakeholders. Inside of a bunch of knowned challenges to explore, there is the “time” factor, that is directly related to the tourists who intend to participate in a touristic itinerary through a specific city, according to their preferences at that time.

The study proposes to demonstrate a model of a recommendation system, able to present suggested tips of what to visit in a city, based on the available time, personal preferences, current geolocalization and the user’s context awareness. These suggestions are calculated based on the treatment of collected data in real time by externals Application Programming Interfaces, through a list of Points of Interest located within a radius that can be reached by the user.

With the results obtained, it is possible to prove and validate the value of the model and its capacity to recommend a route to whom is looking for this kind of touristic offer and have a good time knowing a city or place according to his needs in a specific moment.

Author Keywords

Smart tourism; Recommendation system; Route planner; Context awareness; Point-of-interest.

1. INTRODUCTION

Nowadays, the ever increasing importance role of the tourism in the global economy is evident. From 2020, it is expected that this sector will increase an average annual rate of about 4,3% [1]. International tourism represents 7% of the world’s exports in goods and services. Tourism has grown faster than world trade for the past five years [2].

However not everything is perfect in this sector. One of the biggest issues that tourists are facing when visiting new places is decide what to do, visit and how to get a geographic point, according to his preferences. The working hours of the establishments around is also an important information, there is no reason to give a suggestion about visiting a certain place that is currently closed. This concept is called context awareness, that basically means the detection of environment changes that may arise during the system usage [3].

1.1 Goals

This thesis aims to propose a recommendation system model for sightseeing tours based on the available time, personal preferences, current geolocalization and context awareness variables of an user. The results will be presented with the aid of convenient graphical tools.

In the end, the main goals that are expected to achieve are:

- Presenting a state of the art to discover and analyse others recommendation system models in the tourism sector.
- Improve skills in the recommendation system models area.
- Identify the strengths and weaknesses of each related model.
- Test, evaluate and validate the model with the representation of a recommendation system prototype.

1.2 Research Methodology

Design Science Research Methodology (DSRM) was chosen for approaching this thesis.

This methodology encompasses the creation and evaluation of innovative products, which provides feedback about the problem and design process to be improved [4].

Figure 1. DSRM Methodology applied to this research

2. STATE OF THE ART

This chapter supports all the main objectives definition and, if we look at Figure 1, the identification and motivation step of the DSRM methodology.

In order to understand what is being done in this area, the following list of similar projects and scientific articles were analysed comparing the strengths and weaknesses of each one and, in the end, with this project.
2.1 Similar projects

2.1.1 PAT-Planner

This project aims to personalize a route based on points of interest near by, available time and budget of the user. [5]

Strengths:
• With the last arguments, the proof of concept of PAT-Planner is directly related of what is wanted to develop in this project.
• Several user evaluations were made.
• The conclusion reveals that the main goals were achieved and also the intention of implement geolocation for user tracking.

Weaknesses:
• What went wrong? It was not answered by the authors of this article. So there was a lack of failed tests during the evaluation with users.

2.1.2 Multi Request Route Planning (MRRP)

Based on the Dijkstra algorithm [6], the authors of this article wanted to suggest to the users a walking route through a different types of places in urban environments. [7]

Strengths:
• Project's architecture presented by perceptible diagrams.
• Webservices culture that can be useful to this project.
• Usage of Google Places API to get places according to user's location.

Weaknesses:
• The Context awareness term is not mentioned by the authors of this article and it is a very important area to explore in that context.

2.1.3 Trip planning route

The main goal is to suggest tourist routes according to the working hours and the popular time at each place to visit. [8]

Strengths:
• The time factor is the biggest challenge of this article.
• The results of the route generator is the combination of two methods: distance between two points and another one using Google Maps API LUT.

Weaknesses:
• No tests were applied to proof this model concept.

2.1.4 Xenia

Xenia is a recommendation system model that receive information about POIs from Flickr, a social network. [9]

Strengths:
• Context awareness it is the biggest challenge of this article. They receive that kind of information from Flickr, which sending to them the user's activities around a specific location.
• Project's architecture presented by perceptible diagrams.
• Webservices culture that can be useful to this project.

Weaknesses:
• No tests were applied to proof this model concept.

2.1.5 Trip-Mine

Trip-Mine wants to improve the recommendations in the tourism sector according to the user's location. The biggest challenge is to deal with the user's available time.

Strengths:
• The authors developed optimization mechanisms to improve the efficiency of the place search.
• They analyzed and adopted the Travelling Salesman Problem Algorithm to start the logic of their own algorithm.

Weaknesses:
• Due the great theory around this article, authors forgot about give examples of their concept in real life.

2.1.6 Touch Map

It is a framework developed with two modules: 1) A cloud system dedicated to search for points of interest. 2) A cloud system to create a touristic route based on user inputs, like available time and budget limit.

Strengths:
• Problems were well defined. The authors also developed a very well defined framework to solve their problems.
• This project was developed based on Trip-Mine that was also analyzed in this state of the art.
• The evaluations were made in several places around the world, like EUA and Asia.

2.1.7 Personalized Sightseeing Planning System (PSiS)

PSiS is a recommendation system to create and suggest touristic routes based on user parameters such as preferences, budget limit and available time. [10]

Strengths:
• The authors analyzed and adopted the Travelling Salesman Problem Algorithm to start the logic of their own algorithm.
• They save user data that can be useful in the future.

Weaknesses:
• There is no future work vision that can explain to which way they want to grow.
2.1.8 Google Trips

Google Trips is a mobile application that has a lot of features for its users. One of the main features is the possibility to create custom touristic walking routes around the world, according to the user preferences. [11]

Strengths:

- Google Trips uses Google Places API. The biggest advantage is to use this in-house API.
- This mobile application is very known around the world and has million of active users. This is a positive point because it turns information more fiable.

Weaknesses:

- The quantity of the available features can make the user experience worst.
- The process to create a new touristic walking route is very long and the user has to select a lot of options to finish.

2.2 Goals definition

After completing the study about similar projects, the following goals were clearly defined:

- Development of a recommendation system for sightseeing tours, according to user's current location, available time, personal preferences and availability in real time of touristic attractions (context awareness).
- Retrieve points of interest from external API's that holds this kind of information.
- Develop a mobile application to allow the user insert his inputs and obtain suggestions in real time, supported by a convinient graphical tool.

3. CONCEPTUAL MODEL

This chapter aims to understand which tasks are being done currently by the users, which kind of architecture should be implemented and know which type of application flow should be followed.

3.1 Requirements gathering

Before the prototype's development phase, it is necessary understand which tasks are being done currently by the users. One way to get this information is ask them what tasks they prefer or avoid to do while using similar recommendation systems and also, what tasks they want to see available in the near future.

To obtain all answers, a survey with a total of 12 questions was prepared and sent to potential users of this type of systems. With this survey could be possible to know some personal information about the user like age and gender, current tasks and ideal tasks in the future.

As a result of the survey, the following list represents the most suggested user's ideas during the interview process:

- List only sightseeing tours with touristic attractions open now (context awareness)
- List only sightseeing tours based on user's preferences
- List only sightseeing tours based on user's available time

3.2 Architecture

This recommendation system model is based on an Online Architecture, which means that all included features must be accessed through a device connected to the Internet. This is an important requirement since it is necessary to know, in real time, information about places around the user. Figure 2 represents the architecture proposed to solve the problem.

![Architecture diagram](image)

Figure 2. Architecture diagram.

3.3 Algorithm demonstration

After analysing the architecture diagram represented by Figure 2, it is possible to understand that the most critical and challenging part is the connection between internal and external API's. One of the user's inputs is his available time at the moment of the sightseeing tour creating process, so it is important to get places from the external API according to this input. The longer the available time, the more number of places should be returned.

It is extremely important to pay attention to the execution time and complexity that this problem involves. Figures 3 and 4 represents algorithm demonstrations to solve this complexity.
As Figure 3 shows, the repetition statement is only executed while total time is less than the limit time. After that condition, the filling of the POIsList should be finished and will represent every suggested stop during the sightseeing tour.

Sometimes a duplicated POI should be returned by the external API. To avoid this situation, this algorithm demonstration has also a condition to do not let POIsList be filled by duplicated data.

Finally we have Figure 4 that demonstrates how to end with the repetition statement. The total time must be always incremented by the time between current and POI's locations.

4. PROTOTYPE IMPLEMENTATION

In this chapter are explained the used methods for the development of the prototype. The first decisions made was related to the technologies to apply in every component. The main goal of this project is the study of a recommendation system model and, only in second instance, the development of the prototype, which has as main focus the DSRM methodology represented in chapter 1: the design and development, demonstration and evaluation phases.

4.1 Internal API

Symfony 3.4 framework [12] was the chosen technology to implement the Internal API. The following list presents the main reasons of this decision:

- Professional and academic experience of the author using these technologies.
- Excellent official documentation of Symfony available online.
- Symfony 3.4 version is more minimalistic than older, which means that becomes a cleaner solution.
- Use Doctrine ORM [13] for data layer abstraction it is easier.
- The selected database management system was MySQL. It is easier to map with Doctrine ORM and grants reliability and data security.

Figure 5 represents the Symfony framework architecture based on this project.

![Symfony architecture](image)

![Doctrine ORM](image)

Looking at Figure 6, it is possible to verify how Doctrine ORM deals with database layer. In this case the table contains data that will be transformed to a JSON Object.
4.2 External API

For the external API it was decided to use Google API's because:

- One of the most important requirements was to get updated data from POI's in real time and with no geographical restrictions.
- It was necessary to obtain quality and trusted data, since the main goal of this recommendation system for sightseeing tours is to turn the user experience as best as possible.
- The most projects studied in chapter 2, every time they need to access data about location and POI's, they used Google external API's.

Google has several API's with different proposes. In this case we opted for the following ones:

- Google Places API.
- Google Distance Matrix API.

Google Places API [13] is a service with information about places, be they establishments, geographic locations or points of interest. With this API is possible to get places near by specific coordinates, types (categories) and, according to a boolean parameter, if they are open now or not (context awareness). These 3 parameters are directly related with the base of the present project, which means that it will be possible to get all POIs according to them.

On the other hand there is the Google Distance Matrix API [14]. Thanks to this service is possible to calculate how long will take to get from Point A to Point B when hiking. The distance between these two points is also possible to get in the same service response object.

4.3 Database

As defined above, the database that is supporting the internal API was developed using the database management system MySQL. Considering all features to be implemented in the prototype of this project, it was only needed to create 4 tables represented in Figure 7.

The first decision made was create a mapping between Category and GoogleType entities. The first one holds information about the three category types to be selected by the user and it is only used internally, while the second one has several sub types defined by Google Places API. See Figure 8 and Figure 9 to take a look at the content of each described table. In short:

- For Category entity "Sabores", the three sub types are mapped in GoogleType entity: "bakery", "cafe " and "restaurant".

Thanks to this mapping, it is possible to suggest a more varied tour to the user.

![Figure 7. E-R Diagram.](image)

![Figure 8. Static data from Category table.](image)

![Figure 9. Static data from GoogleType table.](image)

4.4 Mobile Application

To evaluate the conceptual model of the present recommendation system, it was developed a mobile application, since it will be here the user can insert all required inputs parameters.
The development of the mobile application required the following decisions:

- Developed using the NativeScript framework because of the author's professional experience.
- This framework turns the development faster and easy to connect with the internal API.

In Figure 10 it is represented the developed mobile application running.

![Figure 10. Tour category selector.](image)

After user insert his inputs, category and available time, the mobile application will ask user to accept location permissions and finally generate the sightseeing tour, as finally represented in Figure 11.

![Figure 11. Sightseeing tour created.](image)

User location is updated real time to help him found the next POI. Also, these POIs are ordered by proximity and numbered when user clicks on any of them.

5. EVALUATION

This chapter presents the experimental results of the prototype. It was performed two types of tests: performance and functionality. With the first ones is expected to obtain results about the performance of the API and the Server where it was hosted, to understand if the development described in chapter 4 was correctly applied. The functionality tests wants to validate what users think about this recommendation system.

5.1 Performance tests

About the test environment is important refer that the internal API is located at DigitalOcean, a hosting provider, with the following technical characteristics:

- 512MB RAM
- 20GB disk space
- Operating system Linux Ubuntu 16.04
- Located at Amsterdam, The Netherlands

To run the performance tests, it was used a tool called JMeter [15]. This work tool allows run tests simulating different number of simultaneous users using different endpoints of the API.

5.1.1 GET Category test

To test many requests to the GET category endpoint it was first defined 5 simultaneous users. After that, the same endpoint was requested by 10 users. Figure 12 represents a graph with the final results of this test.

![Figure 12. GET Category (requisitions x milliseconds).](image)

5.1.2 POST Create tour (2 hours)

Figure 13 represents the results considering requests to POST create tour. This service is the most complex of this system because will creates a new tour calling external API's and taking into account all the requirements defined previously, and 2 hours inserted by user as available time to spend.
5.1.3 POST Create tour (8 hours)

There is only a difference between this test and the last one, the available time to spend by user. Considering 8 hours means much more complexity for the system. Figure 14 represents the results of this test.

5.2 Functionality tests

To validate the prototype functionality, were performed tests with 16 users: 10 males and 6 females. They are between 18 and 40 years old. The sample of the results was chosen considering people living in Great Lisbon, because is where the Author is currently located and where is more probably to get good POI's results.

To test all functionalities, it was installed the mobile application on every user's device. They follow a test script to know what to do during the test process, with the following steps:

1. Go to the recommendation system application.
2. Create a sightseeing tour answering all questions.
3. If you are satisfied with the suggest tour, start the journey using the help provided by the application.
4. At the end, before turn off the application, call the author of this project.
5. Turn off the application.

After the 16 tests done, it was defined 4 requirements to evaluate:

- R1: User was able to select the tour category.
- R2: User was able to select the available time wanted.
- R3 (optional): User does not like the sightseeing tour suggested and was able to return back to create a new one.
- R4: User was able to follow the tour indications supported by a real time map.

Next, in Table 1, there are described all functionality tests results:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Success (%)</th>
<th>Failure (%)</th>
<th>Not done (%)</th>
<th>Usage Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>R2</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>R3</td>
<td>50</td>
<td>18.75</td>
<td>31.25</td>
<td>62.5</td>
</tr>
<tr>
<td>R4</td>
<td>93.75</td>
<td>6.25</td>
<td>0</td>
<td>87.5</td>
</tr>
</tbody>
</table>

Table 1. Functionality test benchmark.

6. CONCLUSIONS AND FUTURE WORK

In this project was proposed a recommendation system for sightseeing tours based on 4 essentials requirements: personal preferences, available time, current location and context awareness variables.

The chapter 2 defined similar important projects where it was possible to finish with a comparation between them and this project. This helped to understand what these projects are doing in this area and found a way to assimilate the best of each one.

In chapter 3 it was possible to structure the conceptual model and design the solution to be implemented. It became clear which needs the architecture and algorithm demonstration showed about the system complexity.

The chapter 4 demonstrate how internal API was developed, considering its connection to the Google API. It was also described the entity relation database diagram and the development of the mobile application, that will be useful during the tests with real users.

Finally, chapter 5 evaluates the prototype in 2 different test types: performance and functionality. Both revealed positive results. In terms of performance, it was concluded that, the more complexity requested to the different endpoints, the more time the server needs to take processing the data, which is a normal behaviour, but the fact that is a proportional result proves that the server and the internal API responded with success.

In functionality tests it is important to refer that the success rate for all requirements was expressly positive, considering percentages of usage experience between 62.5% and 100%, which means that this model can be very useful in this
scientific area. R3 is the requirement that must have improved in the future, because has a considerable percentage of "Not done" and "Negative" usage experience.

As future work there are some interesting features that can improve this recommendation system model, such as:

- Average visit times to spend in each POI: With this feature it is possible to give to users a more accurate estimated time of a sightseeing tour. This feature can offer a considerable algorithmic complexity that must be studied before the development phase.
- Meteorology: This suggestion was made by 21.2% of the survey's respondents in chapter 3. With this feature the context awareness will be improved in this recommendation system.
- Sightseeing tours list: At this moment the only graphic visualization method is to show the suggest tour in a map. But there is no option to select a tour from a list of several tours. Some users that performed the functionality tests referred about the lack of this feature.

REFERENCES