A platform for the recommendation of points of interest in brazilian cities: architecture and case study

Brazilian Symposium on Multimedia and the Web, 21th, 2015, Manaus.
http://www.producao.usp.br/handle/BDPI/49503

A Platform for the Recommendation of Points of Interest in Brazilian Cities: Architecture and Case Study

Marcos Aurélio Domingues, Thais Emanuele dos Santos, Raíza Hanada,
Bruna C. R. Cunha, Solange Oliveira Rezende, Maria da Graça Campos Pimentel
Institute of Mathematics and Computer Science
University of São Paulo, São Carlos, SP, Brazil
{mad,rhanada,brunaru,solange,mgp}@icmc.usp.br
thais.emanuele.santos@usp.br

ABSTRACT

The tourism sector in Brazil has grown considerably in recent years. Despite this growth, the sector still presents several problems such as the lack of information in Portuguese and in other languages for Brazilian and foreign tourists. The absence of information about tourist sites and ordinary services also affects individuals when settling in a new city, as it is the case when freshman students move to a new city to start their studies in a college or university. In this work, we propose an innovative vision of a context-aware platform for recommending points of interest in Brazilian cities, designed with mechanisms for collecting data from the web, for extracting points of interest and background information, and for learning context-aware recommendation models. The platform is accessed by a mobile application. To validate our proposal, we ran a case study where freshman students used the platform during their first months in a new city.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Design, Experimentation

Keywords

Personalization; Smartphones; Mobile Recommender Systems; Context-Aware Recommendation; Points of Interest

1. INTRODUCTION

The tourism sector in Brazil has grown considerably in recent years. In 2014, this sector generated R$ 492 billion in Brazil and put the country in the ninth place among the world tourism economies [22]. Events like the World Cup 2014 and the Olympic Games in 2016 have contributed greatly to this growth. Despite the growth of the tourism sector, it still presents several problems, such as the absence of information in Portuguese and other languages for Brazilian and foreign tourists. Another scenario where the deficiency of information about tourist sites and ordinary services receives great focus is the moving and the settling in a new city. For example, freshman students moving to a new city to start their studies in a college or university. During the first months, the freshmen will need information about bank, food, lodging, transport, and so on; since they do not know the new city where they will live.

In both scenarios, it is quite important to have ways to provide information about tourist sites and ordinary services for people going to a new and unknown city. A possible solution to the lack of information with respect to points of interest (tourist sites and ordinary services) is the design and development of a platform for a context-aware system to recommend points of interest in Brazilian cities. This work proposes an innovative vision of a context-aware platform for recommending points of interest, which will be designed with mechanisms for collecting data from the web, for extracting points of interest and background information, and for learning context-aware recommendation models. To access the platform, we propose a mobile application for smartphones.

The proposed platform is intended to be extensible, in the sense that it should not have a fixed set of recommendation strategies. In this way, recommendation is performed on the server side by a collection of pluggable and reusable components that act as recommendation suppliers. In the platform, communication has a central role. A central broker mediates between two types of actors – recommendation consumers and recommendation providers – that exchange messages using a language designed for this purpose.

This paper is organized as follows. In the next section, we discuss some platforms proposed in the literature for recommending points of interest (Section 2). We describe the architecture for our proposed platform in Section 3. In Section 4, we present an overview about the implementation of a prototype for our proposed platform. In Section 5, we validate our proposal with a case study, where freshman students use the platform during their first months in a new city. Finally, we present some final remarks and future work in Section 6.

WebMedia'15, October 27–30, 2015, Manaus, Brazil.

DOI: http://dx.doi.org/10.1145/2820426.2820448.

2. RELATED WORK

In recent years there has been an increasing amount of context-aware tourist recommendation works. Partly of this amount is because of the availability of diversified resources on the web and mobile phones. Although they have similar goals, these works differ mainly in their architecture, in the information used as context, how they treat such information and their recommendation strategy. In general, the works founded use a predefined set of strategies to achieve their recommendation. None of them uses an extensible approach which enables expansion of its platform and reuse of recommendation strategies how we propose in this work. The idea of a recommender system with pluggable modules is first presented in [5], however the scenario used is quite different since this type of system is used for web sites.

CATIS [16] and COMPASS [23] are among the first context-aware tourist recommendation works in the literature. The CATIS system suggests restaurants to users according to their dynamic context (i.e. location and time-based adaption) and static context (i.e. user profile). The recommendation retrieves the relevant places in a given area according to the user's distance. As this is a less exhaustive recommendation, the server itself is responsible for determining restaurants to be suggested. COMPASS combines the context information with a prediction strategy to estimate how interesting each point of interest (POI) is for the user. The prediction strategy combines or selects prediction techniques according to each POI class and user information. Prediction techniques include social filtering, case-based reasoning, item-item filtering and category learning. Since, their processing is more robust, the recommendation is made by a platform that contains generic parts of recommendation services and it is independent of the COMPASS application. CATIS and COMPASS showed that communication between components using XML can be efficiently. In both systems, user profile contains less information than we are collecting (e.g. users do not provide ratings). From all works founded, COMPASS recommendation is more related to ours, since the recommendation tool chooses which method to adopt. However, they do not permit extension of the recommendation, whereas they use predefined algorithms.

Some related works have more focus on obtaining contextual data and user profiles than on the recommendation task itself. An example is MyMap [3] that uses data of the user's current situation (e.g. weather), the situational preferences of users and locations descriptive metadata. The recommendation is made through common sense rules, which allows the system to deal with exceptional event or temporal reasoning. MyMap architecture consists of two parts: the Mobile User Profile (MUP) Manager and the Information Selection and Presentation Module. The first one is responsible for the management of user preferences and inference of situational information, while the second one is responsible for the selection of sites according to common sense rules. MoreTourism [19] combines collaborative filtering and content-based techniques along with social recommendation. The authors use information from social networks to form a collaborative tagging. The recommendation is made according to the similarity of the tag cloud generated in accordance with the user interests and the tag cloud generated for the targets places. It also supports groups cloud. MOPSI [24] focus on how to mine knowledge from user generated collections without any data cleansing. In this system, the user profile is entirely inferred based on user's activity on the system. The location-based data recommendation is done according to trusted services, geotagged photos and GPS routes. The recommender combines collaborative filtering (ratings of photos and services) with information about user profile and context. In all these systems, the architecture used is less robust than we are implementing as they have greater emphasis on getting and inferring information to be used by the recommendation tool. So far, our framework does not consider social information or infer situational data of users, restricting to data obtained by the sensors, the data provided by users (preferences and ratings) and its interactions with the system. However, these are options that we can consider in the future.

The usability of mobile systems is also a recurring concern in the works from literature. In [12], the authors discuss some design and implementation issues for their tourism mobile prototype, while in [1] the authors present how applications that consider context information are more pleasant than those that do not offer this type of service, focusing his work on the interaction of user and explanation of recommendations. Therefore we also use an heuristic evaluation process to obtain a better usability for the interface of our platform.

3. ARCHITECTURE OF THE PLATFORM

In this section, we propose an architecture for our recommendation platform. Our concern in designing an architecture for a recommendation platform is to support a wide range of pluggable components to implement different recommendation approaches. For this purpose, the architecture must define a set of interfaces to specify what will be requested to the platform and what it should provide by the platform. In this way, the platform expects two types of extension points: recommendation consumers and recommendation providers. Recommendation consumers are mobile clients (smartphones). Recommendation providers, or simply *Recommenders* are pluggable components that implement a particular strategy to recommend points of interest.

The architecture for the proposed platform is illustrated in Figure 1. Based on the work from Domingues [5], a central broker mediates the communication between mobile clients and the recommenders. The client side recommendation is based on recommendation requests that are channeled throw the broker to one or more recommenders connected to the platform. Place requests are forward to recommenders and the recommended places are loaded on the client side (smartphones). The mobile client is also responsible for collecting user interaction data and reporting them to the broker. All communication with the mobile client – recommendation and notification (interaction) messages – is recorded in a database and made available to recommenders. Using data from the database the recommenders can improve their response to recommendation requests.

The platform is inspired in Service Oriented Architectures (SOA) and follows some of its main guidelines [7]:

1. component interaction is based on implementation independent messages;

2. there is an prearranged agreement on communication content;

3. components are autonomous, i.e., each component is a fully working piece of software;

4. components hide implementation details and the platform focus on communicated data.

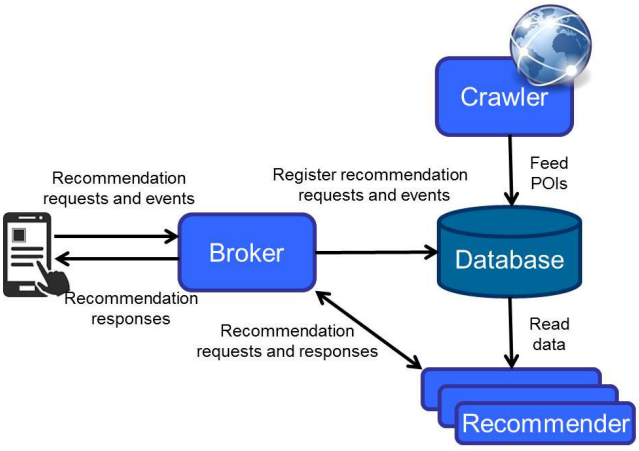

Figure 1: Platform for recommending points of interest.

The next subsections detail the essential parts of this platform, starting with the messages exchanged between components. Then, it is analyzed the role of each individual component: broker, crawler, database, recommender and mobile client.

3.1 The POI Message Language

To communicate, components exchange messages in our POI language, an XML [2] dialect that acts as recommendation language. The name of the language is a shorthand of Points Of Interest. Although there exist some POI languages proposed in the literature [25, 17, 11], these ones are quite complex and present much more elements than we need for our platform. Therefore, for this work, we proposed our own simple POI language. Our POI language was designed to encode all data communication in our recommendation platform, with emphasis on recommendation of place and notification of user activity.

The **recommend** message allows the identification of the geographic position of the user's smartphone and its preference, as well as the type (category) of place that should be recommend to the user. The XML message is presented in Figure 2.

The platform return a list of places (recommendations) that is insert into a map in the mobile application. The XML message which returns the list of places is presented in Figure 3. For each place, the **reply** message keeps its id, name, address, phone, geographic coordinates (latitude and longitude), representative icon, website, rating, and category. Additionally, the message can also contain some reviews about the places.

After the **recommend** and **reply** messages, **notify** is the most important message in our POI language. This message is issued by mobile client to report user activity/interaction. The message is processed by the broker and recorded in the database. The information in this message can be used later by recommenders. A **notify** message may report on several events. Examples of events are: 1) *click* (when the user clicks on a recommended place), 2) *direction* (when the user asks for directions from its current position to the position of the recommended place), 3) *check-in* (when the user reaches the recommended place), and 4) *review* (when the user makes its evaluation about the recommended place). The **notify** message is illustrated in Figure 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
<recommend>
        <user id="22055087052285671414611104171" date="14-
        10-29 17:32:50" lat="-22.0178" long="-47.8908">
                <profile>
                        <preference>loren ipsum</preference>
                        <preference>loren ipsum</preference>
                        <preference>loren ipsum</preference>

                        <!-- ... other preferences ... -->

                </profile>
        </user>
        <place category="food_drink"/>
</recommend>
</message>
```

Figure 2: The structure of the recommend message.

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
<notify>
        <event type="click|direction|check-in|review"/>
        <user id="22055087052285671414611104171" date="14-10
        -29 17:32:50" lat="-22.0178" long="-47.8908"/>
        <place id="d2a2bed0a1b5d134fe14ba88d1ffe3600e93dfae"
        category="food_drink">
                <review language="pt" rating="4.0">
                        Lugar agradavel.
                </review>
        </place>
</notify>
</message>
```

Figure 4: The structure of the notify message.

We also define our POI language in XML Schema [8]. Thus, messages in the POI language can be validated by all components of the platform. In particular, the broker should support message validation, but this feature should be activated by configuration since the overhead may be inadequate in a production scenario.

3.2 Broker Component

The broker has a central role in the platform. It is responsible for registering other components, managing communications and recording data to support recommendation.

Since it is a central component, the broker must be very efficient since it will process all requests coming from clients. It should be noted that several requests will hit the broker: one recommendation request for each recommender involved, and several notification requests, depending on the user interaction on the mobile client.

Using some administration commands, recommenders can register themselves in the broker, so that they can latter be invoked when a request is made for their type of recommendation. On registration, recommenders specify their IP address and their type. Several recommenders with the same type may be registered in the same broker. In this case, the broker will balance load among the recommenders with the same type.

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
<reply>
        <recommendations>
                <place id="d2a2bed0a1b5d134fe14ba88d1ffe3600e93dfae" name="Agua Doce Cachacaria" address="Rua 9 de Julho, 1625,
                Sao Carlos" phone="+55 16 3376-2077" lat="-22.0154" long="-47.8929" icon="bar.png" website="http://
                www.aguadoce.com.br/nossas-casas-agua-doce/sao-carlos/" rating="4.3" category="food_drink">
                        <reviews>
                                <review id="1" time="1339483415" language="pt" overall_rating="5.0">
                                        Excelente muito bom
                                </review>

                                <!-- ... other reviews ... -->

                        </reviews>
                </place>

                <!-- ... other places ... -->

        </recommendations>
</reply>
</message>
```

Figure 3: The structure of the reply message.

A recommender type is encode as an Uniform Resource Name (URN) [20], which is a kind of Uniform Resource Identifier, just as the popular URL used for referring web pages, that does not require the availability of the resource that is identified. URN must follow the syntax `urn:`*`NID`*`:`*`NSS`*, where *NID* is a namespace identifier and *NSS* is a namespace specific string. The namespace identifier of recommenders is the string "`poi-recommender`".

Recommendation requests are simply forwarded to a registered recommender with a compatible type. To process requests, the broker must parse the POI messages described in the previous Subsection 3.1. Then, the response produced by the recommender is sent back to mobile client passing through the broker.

3.3 Crawler Component

The crawler is a component to collect information from places of a city. Based on the types supported by the Google Places API [10], we have compiled a set of categories and types of places which we consider necessary for the platform (Table 1). Based on that list, we have developed a crawler that combines information retrieval techniques and some public APIs to collect such information [18].

Table 1: Categories of places for the crawler.

Category	Type of Place
Car Services	gas station, car dealer, car rental, car repair, car wash, parking
Bank	atm, bank
Education	school, university
Medical Care	dentist, doctor, pharmacy, physiotherapist, health
Emergency Services	fire station, hospital, police
Culture & Entertainment	aquarium, art gallery, casino, movie theater, museum, night club
Food & Drink	bakery, bar, cafe, convenience store, food, grocery or supermarket, liquor store, meal delivery, meal takeaway, restaurant

Table 1: (Continued) Categories of places for the crawler.

Category	Type of Place
Government	city hall, courthouse, embassy, local government office
Lodging	campground, lodging, real estate agency, rv park
Recreation	amusement park, bowling alley, gym, park, spa, stadium, zoo
Public Services	library, post office
Service Shops / Stores	bicycle store, book store, clothing store, department store, electronics store, florist, funeral home, furniture store, hardware store, home goods store, jewelry store, movie rental, shoe store, shopping mall, store, travel agency
Public Transport	airport, bus station, subway station, taxi stand, train station
Places of Worship	cemetery, church, hindu temple, mosque, place of worship, synagogue
Basic Home Services	electrician, general contractor, laundry, locksmith, moving company, painter, roofing contractor, plumber
Beauty Care	beauty salon, hair care
Administrative Services	accounting, finance, insurance agency, lawyer
Animal Care	pet store, veterinary care

3.4 Database Component

The database is an important part of the platform. It will record all interaction with the mobile client (i.e., recommendation and notification messages) and make it available to recommenders. The entity-relationship model for the database is presented in Figure 5 and discussed in the rest of this subsection.

Table ***Events****.* This table records all event requests, i.e., all the interaction between the user and the mobile application. It stores the identification (field *user_id*) and geographic position (fields *latitude* and *longitude*) of the user, the identification (field *place_id*) and category (field

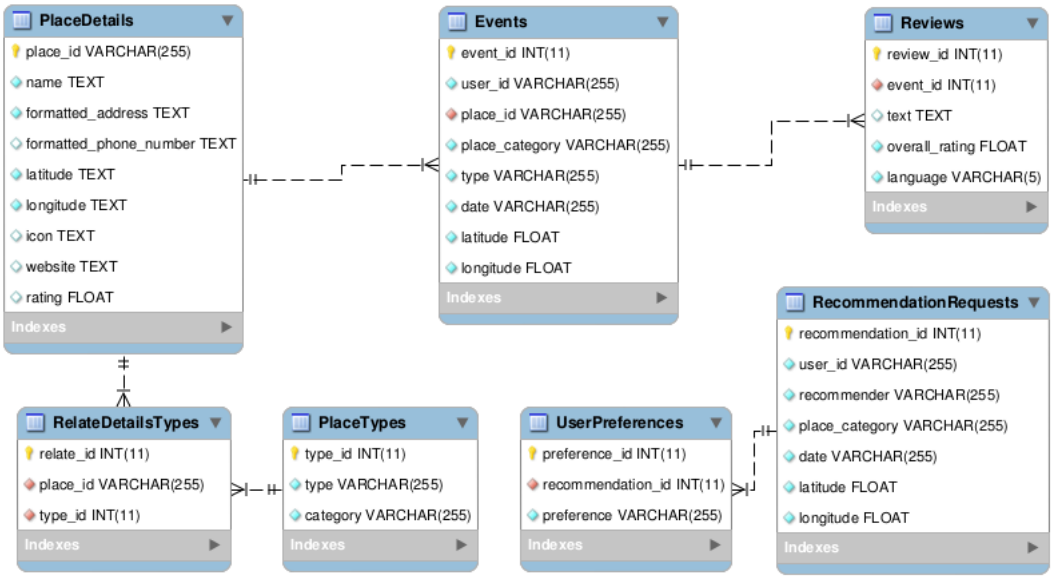

Figure 5: Entity-relationship model for the database.

place_category) of the place which the user is interacting with, the type of interaction (field *type*), and the moment which the event was performed (field *date*).

Table ***Reviews***. The table records the evaluations/reviews made by a user for a given recommended place. In this table, the platform records the rating evaluation (field *overall_rating*), the textual review (field *text*) and the language which the review was written (field *language*).

Table ***RecommendationRequests***. All recommendation requests are recorded in this table. It records the identification (field *user_id*) and geographic position (fields *latitude* and *longitude*) of the user who requested the recommendation, the category of place (field *place_category*), recommender system (field *recommender*) that will be used to generate the recommendations, and the moment of the recommendation request (field *date*).

Table ***UserPreferences***. In the field *preference* of this table we record some preferences from the user who requested the recommendation.

Table ***PlaceDetails***. Currently, this table records the information for all places in a city, and that can be used to generate a recommendation model. In this table, we record the name (field *name*), the address (field *formatted_address*), the phone number (field *formatted_phone_number*), the geographic position (fields *latitude* and *longitude*), the icon (field *icon*), the web site (field *website*) and the rating (field *rating*) for each place.

Table ***PlaceTypes***. This table is filled with information that categorize the places. In this table we have, for each place, a more specific categorization and a more general categorization which are recorded in the fields *type* and *category*, respectively.

Table ***RelateDetailsTypes***. This table relates each place recorded in the table ***PlaceDetails*** with its categorization in the table ***PlaceTypes***.

3.5 Recommenders

Recommenders are pluggable components that process a recommendation request and provide a recommendation response that is delivered to the mobile client component. This platform specifies only the role of recommenders, not their design. An implementation of this platform must specify only the interfaces of communication for the recommenders. Therefore, recommenders may run on different hosts. Some recommenders may be small programs experimenting a new algorithm while other may be a full fledged components optimized for performance.

For the current version of the platform, we proposed and implemented a context-aware recommender based on the Weight Post-Filtering (PoF) approach [6]. For this recommender, we first compute the distance from a user u to each point of interest p in a city by using the spherical law of cosines [9], since the distance is based on the latitude and longitude coordinates:

$$D(u,p) = acos(sin(u_{lat}) * sin(p_{lat}) + cos(u_{lon}) * cos(p_{lon}) * cos(u_{lon} - p_{lon})) * R, \quad (1)$$

where u_{lat} and u_{long} are the latitude and the longitude coordinates for the user u, p_{lat} and p_{long} are the latitude and the longitude coordinates for the point of interest p, and R is the earth's radius.

Then, we select n points of interest with the smaller distance and normalize the rank into standardized values from zero to one $[0, 1]$. In order to contextualize the n points of interest, before recommending them to the user u, we compute the probability $P_c(u,p)$ of a user u to access a point p under a given context c (in our case, the context is the hour of the access). We define the probability $P_c(u,p)$ as

$$P_c(u,p) = \frac{Num_c(k,p)}{Num_c(k)}, \quad (2)$$

where $Num_c(k,p)$ is the number of users k that, similarly to user u, also accessed the point p under the context c; and $Num_c(k)$ is the total number of users that accessed any point under the context c.

Figure 6: Screens for the mobile client component.

Finally, for each point of interest p, the distance D is multiplied by the probability P_c. This final rank is reordered and the top-n points of interest are recommended to the user u.

The idea behind the contextualization process is that points of interest more far away from the user's coordinates can get more emphasis (i.e., increase their score) if they are frequently accessed by other users in a given hour that is the same when the user requests recommendations.

3.6 Mobile Client Component

The mobile client has a very important role in the platform since it is responsible for requesting recommendation to the broker and reporting user activity/interaction for recording, and subsequently for processing the recommendations sent by recommenders.

As a design goal, the client component must be able to communicate with the broker independently from the HTTP request-response cycle and modify its interface, in accordance with the recommendation response. This component was developed and improved iteratively by following a heuristic evaluation of its interface [15, 14]. In Figure 6 we can see the interface of the mobile client as well as a simple navigation diagram.

In the first screen, the user selects the type of places that he/she would like to receive recommendations. The recommendations are sent to mobile application and inserted into a map (screen number 2). Then, the user can click on a recommended place and the screen number 3 appears. At this point, the user can evaluate the place in advance, if he/she already visited that place (screen number 4) or he/she can ask for directions to that place (screen number 5). In screen number 6, the user can evaluate the place after visiting it.

4. IMPLEMENTATION OVERVIEW

The platform was implemented based on the J2EE architecture. To implement it we used the Tomcat servlet container[1], and to implement the web services (i.e., the broker and the recommender) we used the SOAP engine Axis[2] on top of Tomcat.

To exchange a POI message, we inject it directly on the SOAP message. With this approach we avoid unmarshelling XML to an internal representation and marshelling again to XML, reducing to minimum the processing time for this type of message.

The broker must register the data from the recommendation and notification messages in the database. This makes the database a critical point of the platform from an optimization point of view. For this reason the broker uses a pool of connections to the database. By design, the database is independent from the relational database management system it uses and the connection pool respects this independence. Currently, we have been implementing the database by using the MySQL database management system[3].

[1]http://tomcat.apache.org
[2]http://axis.apache.org/axis
[3]http://www.mysql.com

Regarding the mobile client, the implementation of this component was carried out for Android smartphones[4]. The main requirement for this component is that it be asynchronous in the sense that it allows communication with the server simultaneously with user interaction and a response from the server does not imply an update of the interface. With a message being sent asynchronously, the user may continue interacting with the mobile application. When the response is returned by the broker, the POI message is parsed and the recommendation is inserted in the application.

Recommenders are pluggable components of the platform, not part of it. Thus, the platform is responsible only for defining the interface that the recommender must provide. In this implementation, since communication with recommender uses web services, this interface is as a WSDL document [4]. As in the broker, our proposed context-aware recommender was implemented by using the SOAP engine Axis[2].

5. CASE STUDY

The case study to validate our proposal covers six months of usage of the platform by freshmen of the Institute of Mathematics and Computer Sciences (ICMC) of the University of São Paulo (USP) in the city of São Carlos, SP, Brazil.

We implemented our platform and deployed it for the freshman students since they moved from their cities to São Carlos and needed some help (i.e., information about ordinary services) during their first months in the city. The mobile application, which gives access to the platform, could be downloaded from the Google Play Store[5]. The mobile application is called "Onde Tem", which means "Where do I get it?" in English.

The platform was used by 83 freshmen from 01-31-2015 to 06-11-2015. During this period, we recorded 594 requests of recommendation and 477 events of interaction with the mobile application. In Figure 7, we can see the evolution of recommendations and events along the months. The highest values in the second month (February) are explained by the advertisement that we carried out for the freshmen in their first week in the ICMC/USP.

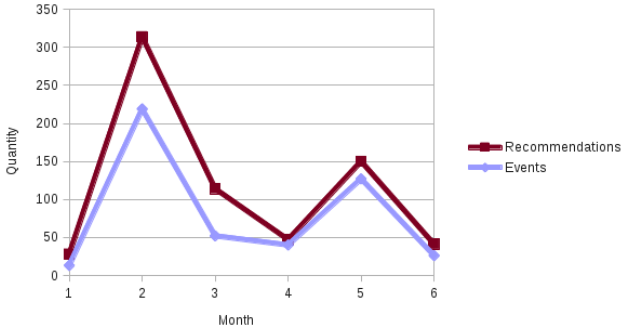

Figure 7: Evolution of recommendations and events.

In Figure 8 we can see the evolution of each type of interaction (i.e., Clicks, Reviews, Check-ins and Directions) carried out by the freshmen using the mobile application.

[4]http://www.android.com
[5]http://goo.gl/VvIYKo

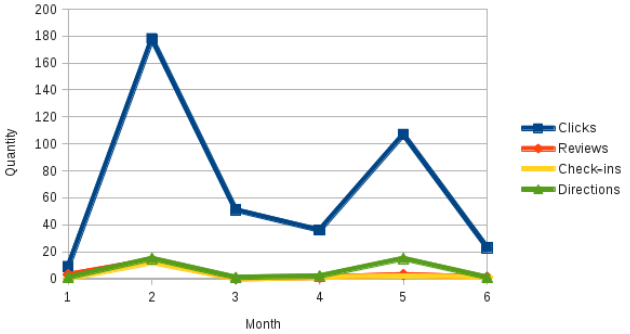

Figure 8: Evolution of each type of event.

In summary, the platform was widely used by the freshmen in their first weeks in the city of São Carlos. However, as the city is not so big, after a few weeks the freshmen reduced the interaction with the platform. An explanation for this fact is that they became more familiar with the city of São Carlos. During the case study, our platform also showed a good performance, serving the freshmen with recommendations in a good response time.

6. FINAL REMARKS

In this paper, we described the architecture of our platform for recommending points of interest in Brazilian cities. The platform is designed to connect recommendation consumers (mobile clients) with recommendation providers (recommenders), and to provide basic functionalities to support recommendation. Therefore, the major contributions of this work is the architecture for the platform based on service oriented concepts, the XML language for exchanging messages between the several types of components, and the context-aware recommender system.

In addition, we validated our platform with freshmen going to the Institute of Mathematics and Computer Sciences (ICMC) of the University of São Paulo (USP) in the city of São Carlos, SP, Brazil. The case study showed that the platform is useful for people going to a new city for the first time.

As future work, we intend to improve our platform before running a new case study during the 2016 Olympic Games in Rio de Janeiro, RJ, Brazil. To do that, we plan to improve our crawler by using more advanced techniques, as the ones presented in [18]. We also plan to propose a more efficient recommender system to our platform, following the ideas presented in [13, 21].

Acknowledgments

Thanks to the support from grants 2014/08996-0, São Paulo Research Foundation (FAPESP); 005/2012, Amazonas Research Foundation (FAPEAM); CNPq (311659/2011-0) and CAPES/Brazil.

7. REFERENCES

[1] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. Context-aware places of interest recommendations for mobile users. In A. Marcus, editor, *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, volume 6769 of *Lecture Notes in Computer*

Science, pages 531–540. Springer Berlin Heidelberg, 2011.

[2] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible markup language (xml) 1.0 (fourth edition). technical report, world wide web consortium, http://www.w3.org/tr/xml, 2006. Access date: 23/06/2015.

[3] B. D. Carolis, I. Mazzotta, N. Novielli, and V. Silvestri. Using common sense in providing personalized recommendations in the tourism domain. In *Proceedings of the Workshop on Context-Aware Recommender Systems*, CARS '09, 2009.

[4] E. Christensen, F. Curbera, G. Meredith, and G. Meredith. Web services description language (wsdl) 1.1. technical report, world wide web consortium, http://www.w3.org/tr/wsdl, 2001. Access date: 23/06/2015.

[5] M. A. Domingues. An independent platform for the monitoring, analysis and adaptation of web sites. In *Proceedings of the ACM Conference on Recommender Systems*, RecSys '08, pages 299–302, 2008.

[6] M. A. Domingues, M. G. Manzato, R. M. Marcacini, C. V. Sundermann, and S. O. Rezende. Using contextual information from topic hierarchies to improve context-aware recommender systems. In *Proceedings of the 22nd International Conference on Pattern Recognition*, ICPR '14, pages 3606–3611, 2014.

[7] T. Earl. *Service-Oriented Architecture - Concepts, Technology and Design*. Prentice Hall, 2005.

[8] D. C. Fallside and P. Walmsley. Xml schema part 0: Primer second edition. technical report, world wide web consortium, http://www.w3.org/tr/xmlschema-0, 2004. Access date: 23/06/2015.

[9] W. Gellert and V. N. R. Company, editors. *The VNR concise encyclopedia of mathematics*. Van Nostrand Reinhold Co. 1977, New York, 1975.

[10] Google Developers. Google places api - place types, https://developers.google.com/places/supported_types, 2015. Access date: 23/06/2015.

[11] Google Developers. Keyhole markup language, https://developers.google.com/kml, 2015. Access date: 23/06/2015.

[12] M. Kenteris, D. Gavalas, and D. Economou. An innovative mobile electronic tourist guide application. *Personal and Ubiquitous Computing*, 13(2):103–118, 2009.

[13] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel. Lars: A location-aware recommender system. In *Proceedings of the IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 450–461, 2012.

[14] J. Nielsen. Usability inspection methods. chapter Heuristic Evaluation, pages 25–62. John Wiley & Sons, Inc., 1994.

[15] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, pages 249–256, 1990.

[16] A. Pashtan, R. Blattler, Andi, A. Heusser, and P. Scheuermann. Catis: A context-aware tourist information system. In *Proceedings of the 4th International Workshop on Mobile Computing*, 2003.

[17] C. Portele. Geography markup language (gml), version 3.3. open geospatial consortium (ogc), http://www.opengeospatial.org/standards/gml, 2012. Access date: 23/06/2015.

[18] A. Rae, V. Murdock, A. Popescu, and H. Bouchard. Mining the web for points of interest. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 711–720, 2012.

[19] M. Rey-Lopez, A. Barragans-Martinez, A. Peleteiro, F. Mikic-Fonte, and J. Burguillo. moretourism: Mobile recommendations for tourism. In *Proceedings of the IEEE International Conference on Consumer Electronics*, ICCE '11, pages 347–348, 2011.

[20] K. Sollins and L. Masinter. Rfc 1737: Functional requirements for uniform resource names. technical report, the internet engineering task force, http://tools.ietf.org/html/rfc1737, 1994. Access date: 23/06/2015.

[21] K. Stefanidis, E. Ntoutsi, M. Petropoulos, K. Nørvåg, and H. Kriegel. A framework for modeling, computing and presenting time-aware recommendations. *Transactions on Large-Scale Data- and Knowledge-Centered Systems X - Special Issue on Database and Expert Systems Applications*, 10:146–172, 2013.

[22] Turismo.gov.br. 500 dias para a olimpíada: conheça os desafios do turismo, http://www.turismo.gov.br/ultimas-noticias/979-500-dias-para-a-olimpiada–conheca-os-desafios-do-turismo.html, 2015. Access date: 23/06/2015.

[23] M. van Setten, S. Pokraev, and J. Koolwaaij. Context-aware recommendations in the mobile tourist application compass. In P. De Bra and W. Nejdl, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 3137 of *Lecture Notes in Computer Science*, pages 235–244. Springer Berlin Heidelberg, 2004.

[24] K. Waga, A. Tabarcea, and P. Franti. Recommendation of points of interest from user generated data collection. In *Proceedings of the 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, CollaborateCom '12, pages 550–555, 2012.

[25] M. Womer. Points of interest core. w3c working draft, http://www.w3.org/tr/poi-core, 2011. Access date: 23/06/2015.