doi:10.20944/preprints201709.0074.v1

*Article*

# Design and Implementation of a Context-aware Personalized Recommendation System: Mobile and Web Application

**Sara Saeedi [1,*], Xueyang Zou [2], Mariel Gonzales [1], Steve Liang [1]**

[1] Department of Geomatics Engineering, University of Calgary, Canada; (ssaeedi,gonzales.mariel, liangsteve)@ucalgary.ca
[2] Department of Geography, University of Calgary, Canada; xueyang.zou@ucalgary.ca
* Correspondence: ssaeedi@ucalgary.ca; Tel.: +1-403-926-4030

**Abstract:** The ubiquity of mobile sensors (such as GPS, accelerometer and gyroscope) together with increasing computational power have enabled an easier access to contextual information, which proved its value in next generation of the recommender applications. The importance of contextual information has been recognized by researchers in many disciplines, such as ubiquitous and mobile computing, to filter the query results and provide recommendations based on different user status. A context-aware recommendation system (CoARS) provides a personalized service to each individual user, driven by his or her particular needs and interests at any location and anytime. Therefore, a contextual recommendation system changes in real time as a user's circumstances changes. CoARS is one of the major applications that has been refined over the years due to the evolving geospatial techniques and big data management practices. In this paper, a CoARS is designed and implemented to combine the context information from smartphones' sensors and user preferences to improve efficiency and usability of the recommendation. The proposed approach combines user's context information (such as location, time, and transportation mode), personalized preferences (using individuals past behavior), and item-based recommendations (such as item's ranking and type) to personally filter the item list. The context-aware methodology is based on preprocessing and filtering of raw data, context extraction and context reasoning. This study examined the application of such a system in recommending a suitable restaurant using both web-based and android platforms. The implemented system uses CoARS techniques to provide beneficial and accurate recommendations to the users. The capabilities of the system is evaluated successfully with recommendation experiment and usability test.

**Keywords:** recommendation system; context awareness; location based services; mobile computing, cloud-based computing

## 1. Introduction

Traditionally, recommender systems deal with applications having only two types of entities, users and items, and initially based on collaborative filtering of the content of them [1]. These applications have looked the same, and worked in the same way, for every customer. However, the advances in sensor technology and deploying them on portable devices together with ubiquity of internet-accessing smartphones and ever-increasing computing power, ignited the idea of context-aware and personalized recommendation system. Context-aware systems take into account contextual information (such as such as time, location, activity, social information and etc.) in order to adapt their operations to the current context without explicit user intervention [2]. The users carry the mobile devices with internet-access and various sensors (e.g. GPS) almost anywhere and at any time; therefore, collecting a vast amount of information about the user is feasible in an automatic way [2]. However, there is still a gap between raw data collected from mobile sensors and suitable context information. The context-aware algorithm uses semantic modelling and machine learning techniques to automatically recognize the contexts from heterogeneous and noisy sensor data [3]. The current challenge in recommendation systems is to design a pervasive, real-time service which is adaptable to user's modes and context. A well-designed recommendation system can provide users with options that are more relevant to their situation, interests and preferences.

The objective of this study is to design and develop an application for personalized and context-aware restaurants recommendation, targeting individuals with willingness to plan out dining activities in the Calgary area. This study focuses on developing a context-aware recommendation system (CoARS) for local restaurants in Calgary based on the "travel locality" principle, which will be stated in the paper. The context information (restaurant's location and ratings, user's location, mode of the transportation, and preference on different types of restaurants), has been used to calculate a recommendation index for each restaurant in the data base, and the priority of the recommended restaurants can be determined by order of the indexes. Context acquisition follows a feature-level recognition approach which includes preprocessing, feature detection, feature selection and classification step. The appropriate set of sensors and features is carefully selected to perform real-time and accurate activity recognition. The details of a mobile application and performance analysis of different classification techniques is already stated in the previous publications [2-4]. The restaurant of interest can be chosen by the user from the recommended list, followed by an illustration of the optimal route and travel mode via the launching of the Google Maps app. This paper contributes to the CoARS in the following three aspects:

- Design and developing a context-aware recommender mobile and web application for Calgary restaurants.
- Context-aware algorithm using multi-sensor and multi-source data integration for smartphones. Since the context information may be acquired from heterogeneous sources, defining an appropriate strategy to integrate various sources of information is necessary. Therefore, a hybrid multi-level (sensor-, feature-, decision-level fusion) context detection algorithm is developed to apply smartphone sensors. Moreover, fuzzy inference engine is used for uncertainty modeling of the hybrid method. This algorithm provides a reliable and readable method which is less sensitive to the noise of the data.
- Using a context-sensitive distance measurement based on the user's mode of transportation.
- Recommending the best places list based on restaurant ranking, location and type and examining context information (i.e. number of visits for users, user's preference on type of the restaurant, weather and time of the day in user's location) if available.

This paper is divided as follows: Section 2 contains summarization of related work; Section 3 contains steps of how the system runs; Section 4 discusses experimental results and discussions. Finally, Section 5 concludes the paper and presents future works.

## 2. Literature Review

Recommendation systems carry out a logical data collection, rating, and filtering procedure while handling user-specified queries. During this procedure, community opinions are utilized and the user-valued information is sorted out efficiently [5-8]. Common recommender systems can be categorized into two types: spatial-related and non-spatial related query filtering. Currently, there exists many spatial-sensitive recommendation systems allow users to check in and rate their visit on nearby services (e.g. TripAdvisor). However, traditional recommendation systems are mostly non-spatial related, as the locational information from neither the users nor the locations is involved into the sorting algorithm. For items insensitive to their locational properties, the related recommendation systems tend to fall into the non-spatial related category which recommends items purely based on personal preferences (e.g. Netflix). However, previous studies have discovered that there is a strong dependency between the location and user preference in non-spatial items, meaning that users from the same region tend to prefer similar items which are different than those preferred from another region [9]. Therefore, this spatial dependency suggests that at least users' locational information should be involved. Furthermore, studies have also revealed the property of "travel locality" in most spatial-sensitive recommendation systems, which refers to people's preference to travel less yet still achieve satisfying searching/recommended results [2, 9]. State of the research in recommender system has designed algorithms to explore the various applications of the location-aware recommender system, where some of the findings have accordingly affirmed the nature of the spatial dependency for recommendation items [8]. For example, the collaborative technique conducts similarity analysis based on similarity indices between users and/or items in a certain spatial domain, which in turn validates the existence of spatial autocorrelation in the items of interest [10-12].

Spatial dependency can be considered as a location context in a CoARS. Furthermore, other context information can be measured about the user or item status. According to Charu Aggrawal, "Context-sensitive recommender systems tailor their recommendations to additional information that defines the specific situation under which recommendations are made. This additional information is referred to as the context."[6] Defined by Saeedi et al. [2], a context-aware model consists of four parts: data preprocessing, context extraction and validation, association rule mining and context reasoning. The first procedure, data preprocessing, aims at filtering the acquired real-life data to be suitable for machine learning in further steps. Data preprocessing operations include but not limited to similarity measurement, resampling, dimension reduction, and denoising. The "similarity measure" using the collaborative filtering algorithm is most commonly adopted by classic CoARS design. As Park et al. [10] summarized, the general idea behind the collaborative filtering algorithm is to reflect 'personal locality', that users within a regional domain vote/rate on items more similarly. It predicts an item's rating by aggregating its ratings from users with similar profiles [13]. Ekstrand et al. [14] concluded the collaborative filtering as one of the 'core concepts' in CoARS, as it provides the first filtering process for the upcoming steps. The collaborative filtering conducts similarity measures between users and/or items using similarity indices, such as Pearson's correlation index [15], cosine similarity measure, random similarity measure [16] and the Bayesian similarity measure [17]. According to Lathia et al. [16] comparison of the performances of different similarity measures, the prediction accuracy is irrelevant to the choice of collaborative filtering techniques. However, Park et al. [10] and Guo et al. [17] noticed the inefficiency of the traditional Pearson correlation and cosine similarity measures; and adopted a novel Bayesian similarity measure for better prediction results. Under the circumstances where huge computational power is required, sampling and dimension reduction techniques are preferred to perform on the original dataset to achieve better performances [3]. For data containing various noise, the denoising step is required to remove unwanted effect while preserving its information content.

The context extraction procedure aims at labeling the items into categories which are meaningful for the specific application of the CoARS under design. For example, a CoARS for the local restaurant can use human preference context recognition to identify which type of restaurants (North American, Indian, South Asian, European and etc.) is more popular for a user using a series of observations from the history of user's location. Common classifiers include nearest neighbors, decision trees, artificial neural networks and etc., and they have been well adopted for both supervised and unsupervised context extraction and validation. However, apart from the machine-learning based approaches of recognition, the direct acquisition of a single class out of a complex profile of the recommendation items via data mining can also be considered. The cluster analysis is widely used in scaling down the scope of computation in generating the recommendation index. It discovers the patterns within the dataset and identifies items that are inherently more similar. For studies concerning higher quality recommending results with better efficiency, the partitional and hierarchical clustering algorithms have been adopted [18, 19].

In a CoARS model, context reasoning techniques attempt to integrate different sources of information such as user's personal data, environmental sensors, and inferred contexts according to logical operation [2]. Therefore, the aim of context reasoning is matching and deducing useful context using association rules or the axiomatic semantics-based inference and domain specific rules [15, 18, 20, 21]. Moreover, by reasoning context classification, system is able to detect more complex context and resolve the conflict. For example, Madadipouya [15] included time, price and mood information of items and users into his CoARS model, aiming at providing real-time emotion-based recommendation results.

This study focuses on developing a CoARS for local restaurants in Calgary based on the "travel locality" principle stated above. It aims at overcoming the drawbacks of the traditional non-spatial recommendation system by providing user-specific searching results with thorough concerns on the user location, the item location and the user's preference. The main objective of this study is to create two platforms for the CoARS of local Calgary restaurants in order to suit for different users with different devices. The two platforms are: 1) A web-based platform suitable for users with desktop, laptop, cellphone and tablet. 2) A smartphone-based Android App for users prefer faster access and clearer interface.

## 3. CoARS Design and Methodology

Our study adopted a simplified version of the classic CoARS design (data preprocessing, context extraction and validation, and context reasoning) [2]. Time, location, user preference and modes of transportation are main context information in the proposed CoARS design which incorporate in restaurant ranking to filter the available items. The user's location and time can be automatically calculated using the embedded GPS sensor in a smartphone or the location/time of the browser. First the restaurant list is refined by adding users' time, so that restaurants that don't operate at the preferred time could be eliminated before filtering procedure. The distance and best route between user and restaurant is computed based on a context-aware method using user's transportation mode. The recommendation index is constructed from three influential factors: 1) the context-aware distance between the user and the candidate restaurants; 2) the ratings of the candidate restaurants and 3) the user's browsing history which reflects his/her personal preference in a specific type of restaurants. A database is used for storing the restaurants' profile information. The user's location and the browsing history information (real-time clicking on specific restaurants) is dynamically retrieved and stored in another local SQLite database, and the information can be exploited by the recommendation generator. The similarities between local users were not considered in our study, therefore only the property of 'travel locality' has been taken into consideration in our system design. The following equation expresses the equation calculating the recommendation index for each restaurant:

$$\text{Rating } \times \text{Prefernce Index}/\text{Context\_Aware Distance} \tag{1}$$

The "*Rating*" property is stored in the static database along with all the other restaurants' context information. The "*Preference*" index is reflected by the user's browsing history in terms of the restaurant types. The proportion of the records containing each type of restaurant among the user's overall browsing records is considered as the weight "Preference" index, which is assigned to every restaurants belonging to the same category. The "*CA Distance*" is a context-aware (CA) distance calculated as the Network distance between the user's current location and the restaurant's location considering the mode of transportation. Our recommendation algorithm utilizes this constraint for delimiting the list of suggested places: only places "near" the user are analyzed and nearness is defined by the mode of transportation. For example, if an individual is riding a bicycle and requesting a restaurant recommendation, the system will not suggest places, which are an hour biking distance away. However, if the person is driving a car, a restaurant that is an hour away by bike could still be recommended. Our study also considers that requiring the user to constantly update their current form of transportation is uncomfortable. Therefore a person's mode of transportation is automatically detected. The detection is done on the user's smartphone. The method for automatically detecting a user's form of transportation is similar to the process proposed by Saeedi et al. [2]. The mobile system discriminated between a person who is stationary, walking, biking, or driving. The classification is realized by a Bayesian Network (BN) to detect the transportation mode [4]. BN is a probabilistic graphical model that encodes probabilistic dependencies among the corresponding variables of interest by using training dataset. As it is shown in Figure 1, BN is used to learn relationships between transportation mode classes and feature space to predict the class labels for a new sample.

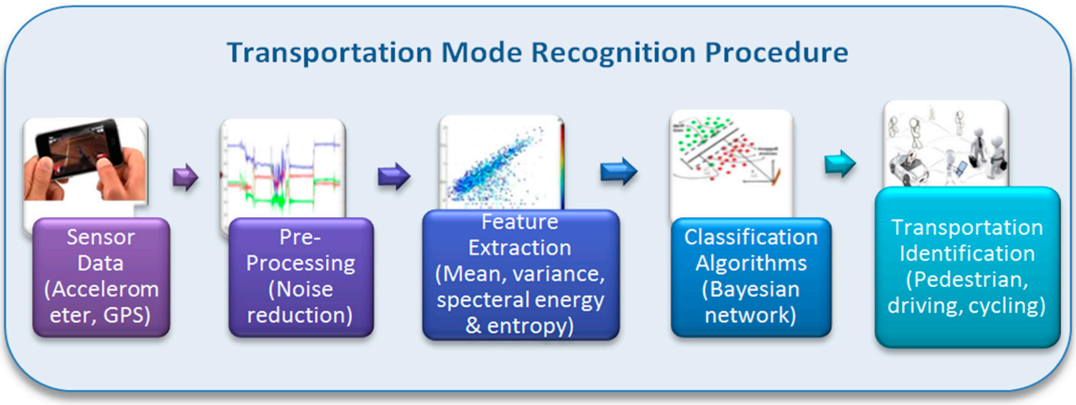

Transportation Mode Recognition Procedure
Sensor Data (Accelerom eter, GPS)
Pre-Processing (Noise reduction)
Feature Extraction (Mean, variance, specteral energy & entropy)
Classification Algorithms (Bayesian network)
Transportation Identification (Pedestrian, driving, cycling)

**Figure 1.** Overview of the transportation mode context extraction from the smartphone's accelerometer and GPS sensors

The accelerometer signal and the GPS speed data from a smartphone is used to detect mode of transportation. Signal variance, spectral energy and entropy are used as feature information extracted from the sensor data (Figure 1). These features have been selected based on the highest classification rate for transportation mode detection. These features include basic waveform characteristics and signal statistics and they are directly derived from the sensor data. The time- and frequency-domain features are mentioned here:

**Mean**: The mean value of the signal over a window segment is considered as a feature according to below equation.

$$\bar{y} = \frac{\sum_{j=1}^{N} y_i}{N} \tag{2}$$

**Root Mean Square (RMS):** RMS of a signal can be considered over a defined widow using following formula.

$$RMSE = \sqrt{\frac{1}{N-1}\sum_{j=i}^{N}(y_j - \bar{y})^2} \tag{3}$$

**Spectral Energy:** Spectral energy density describes how the energy of a signal is distributed over the different frequencies. Energy features can be used to capture periodicity of the data in the frequency domain and it can be used to distinguish inactive activities from dynamic activities. The energy feature is calculated as the sum of the squared discrete FFT component magnitudes of the signal. For the signal in discrete form, energy can be calculated using the following equations:

$$S_{xx}(\omega) = |\hat{x}(\omega)|^2 \tag{4}$$

where $\omega$ is the angular frequency and $\hat{x}(\omega)$ is Fourier Transform of the signal. When, the feature is computed over a window, the sum of the above equation over the window is divided by the window length for normalization.

**Spectral Entropy:** To discriminate the user modes with similar energy values. The frequency entropy is calculated according to the following formula:

$$Entropy = -P(x_i)\log(P(x_i)) \tag{5}$$

where $x_i$ are the frequency components of the signal for a given frequency band and $P(x_i)$ the probability of $x_i$. This feature is a measure of the distribution of the frequency components in the frequency band.

Figure 2 shows the recognition rate for each transportation mode. By investigating each transportation mode, it can be inferred that the user activities such as: driving, walking, running, taking stairs and elevator modes have an accuracy of 95% [2]. Also the user can change the mode of transportation at any time by manually selecting it.

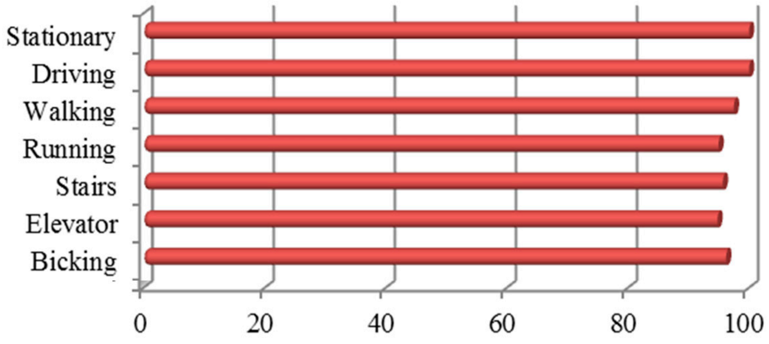


**Figure 2.** Recognition rates for different transportation mode context

*3.1. System Architecture*

The CoARS application can be embedded in both web- and mobile-based environments. However, most existing literature reviews focus on developing mobile-based recommendation system due to the better portability of mobile devices. Park et al. [10] also pointed out that mobile-based environment can excel web-based environment by providing more direct service with 'push' service using SMS or other interaction channels to users. Nevertheless, speaking from a system-design point of view, the conceptual models of the CoARS application share similarities regardless of the operating environment. Two databases (one for storing user's location information and the other for storing the restaurant profile information) create the theoretic foundation of the CoARS, upon which the various filtering, extraction and reasoning techniques are performed. System architecture of the application designed in this study is shown in the Figure 3. The system architecture can be simply divided into two different implementations: web-based and android-based.

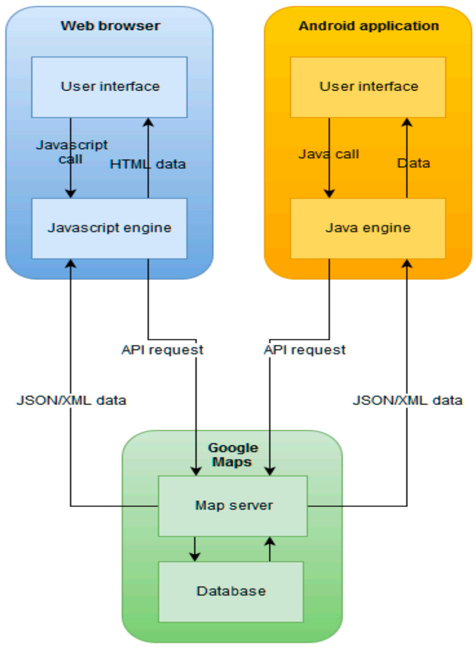


**Figure 3.** CoARS system architecture for the two cases of web-based and android-based implementation

The device's sensors capture the user's current location and transit mode and return it to the database. The recommendation index is then calculated for each restaurant, and then, a list of high-indexed restaurants is returned to the user interface. When the user is in motion, i.e., his position changes notably, the system retrieves nearby restaurants and ranking them, based on their properties, according to their scores.

## 4. CoARS Implementation & Results

### *4.1. Data and pre-processing*

Available APIs for achieving the CoARS functionality include TripAdvisor, Google Map, Foursquare [22], Yelp and Whrrl [23], and the content can be manipulated mainly by Java and Javascript to suit for both web and mobile environment. These APIs contain functions that allow for crawling the check-in history of the users (e.g. Foursquare and Yelp) and dynamically retrieving users' locations (e.g. Google Map). Collecting huge amount of information using the listed APIs can be time-consuming according to Ye et al. [23], as the retrieval of profile (location, check-in history) information of 153,557 users lasted for one month in their study. The proposed CoARS in this paper is developed based on the data collected from the restaurants in the city of Calgary. The website of City of Calgary provides some basic information of them, such as name, address and coordinates, but there is no rating information, which is critically important for recommending, in this dataset. As a result, we decided to extract data from yelp.com, a well-known website that provides crowd-sourced review for local businesses. Python scripts have been written to "scrape" the restaurant data, including name, pictures, address and customers' ratings. An open source python framework, Scrapy, is used when extracting data from website. 100 items for categories of Cafe, Canadian food (North American fast food), Italian Food, Asian Food, and Pubs containing the above information were used in this paper, since extracting large volumes of data from yelp.com against the Terms of Service of the website. In addition, the geocoding of the restaurants from "address" to "longitude and latitude" is acquired by ArcGIS.

### *4.2. Implementation and Results of Android-based CoARS application*

The solution has been implemented and tested in Android Studio. The application consists of several components, which are reflected visually in Figure 4. The implementation of the software was split into two sections. The GUI, which the users will be interacting with and the backend which searches and returns the queries that the users will request. The front end uses the Google Maps API to display an interactive map which the user can pick a location on the map, or enter in a word query. When the user enters a query of some kind, it then sends the query to the backend server which will then search the database. For easy implementation and testing of the software, the publically available data was stored on the user side as opposed to a server side. This allowed the client side query to be tested and verified. Figure 4 below describes the structure of the Android version of CoARS. Three major activities were included in the Android application: Main activity (the home page), List activity and Map activity. The map activity was linked to the Google Map Android app, which could be triggered by the user for real-time traffic information.

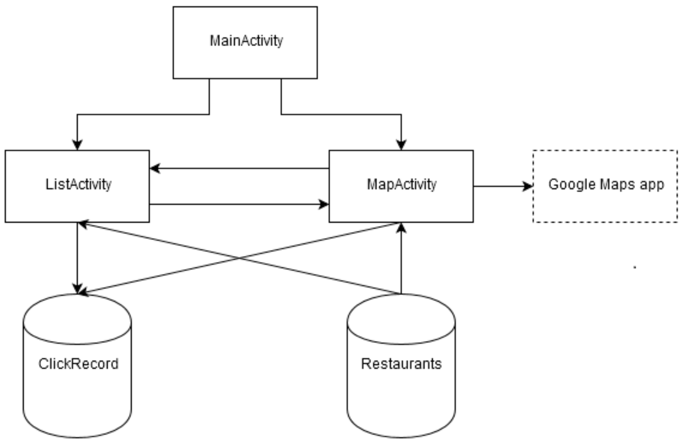
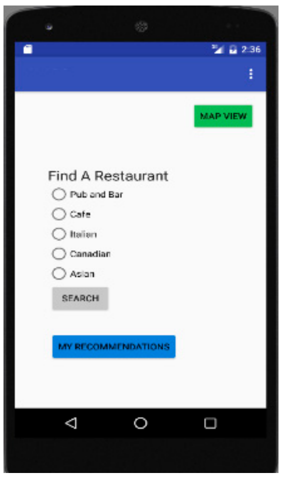
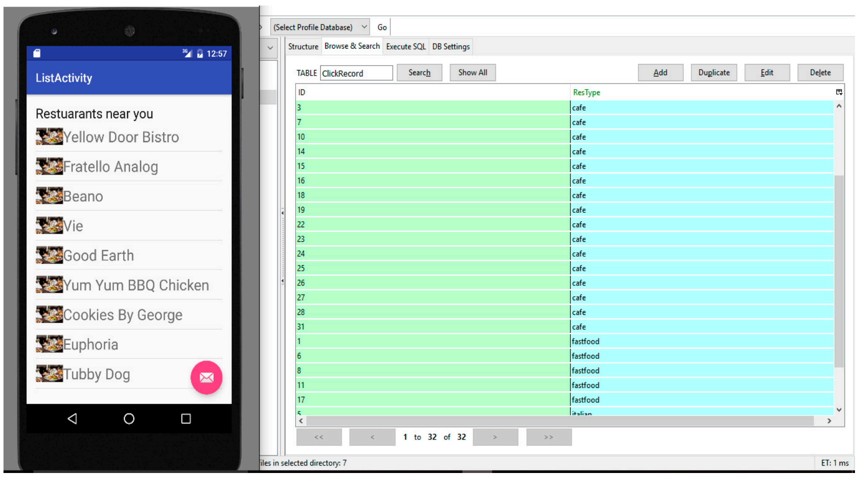


**Figure 4.** Detailed structure of the android-based Co-ARS designed in this study

**Main activity:** The main activity includes the following components: a search bar, a map button (labeled as "MAP VIEW") and a recommendation button ("labeled as "MY RECOMMENDATIONS") which is presented in the Figure 5-a. The search bar provides the functionality of searching the nearby restaurants based on a specific "type" selected by the user. The "search" button leads to a List activity which displays the nearby restaurants of the specific type selected by the user. When the restaurant type was constrained, the recommendation index was calculated simply by: rating/distance. The "MAP VIEW" button leads to a Map activity based on the user's current location as well as all the nearby restaurants. This functionality is useful when the user prefers a random, unbiased glance at the potential choices on the dining activities nearby. The recommend button leads to a List activity which displays the top 20 restaurants with the highest recommendation index.

**List activity:** In this activity, the database-related actions were completed using SQLite. A local database storing the restaurants' context information (name, rating, type, longitude, latitude) was created for convenient record fetching and index calculating. The user's preference component in the recommendation index (Equation 1) was calculated as the percentage of each type of restaurant being checked out by the user in the browsing history. A local SQLite database was used to store user's browsing history in terms of restaurant types. After clicking on one item in the list view, the type of the specific restaurant would be uploaded into the SQLite database, which was created in the user's device at the first time the app was started (Figure 5-b). The recommendation List activity can be updated when clicking on a specific item, which would re-fetch the user's location and user's browsing history to calculate new indexes for each restaurant item. Restaurants with new indexes would then be sorted and the new orders would be reflected in the List activity.





Restuarants near you
Yellow Door Bistro
Fratello Analog
Beano
Vie
Good Earth
Yum Yum BBQ Chicken
Cookies By George
Euphoria
Tubby Dog

TABLE ClickRecord

| ID | ResType |
|---|---|
| 3 | cafe |
| 7 | cafe |
| 10 | cafe |
| 14 | cafe |
| 15 | cafe |
| 16 | cafe |
| 18 | cafe |
| 19 | cafe |
| 22 | cafe |
| 23 | cafe |
| 24 | cafe |
| 25 | cafe |
| 26 | cafe |
| 27 | cafe |
| 28 | cafe |
| 31 | cafe |
| 1 | fastfood |
| 6 | fastfood |
| 8 | fastfood |
| 11 | fastfood |
| 17 | fastfood |
| 5 | italian |

1 to 32 of 32

iles in selected directory: 7

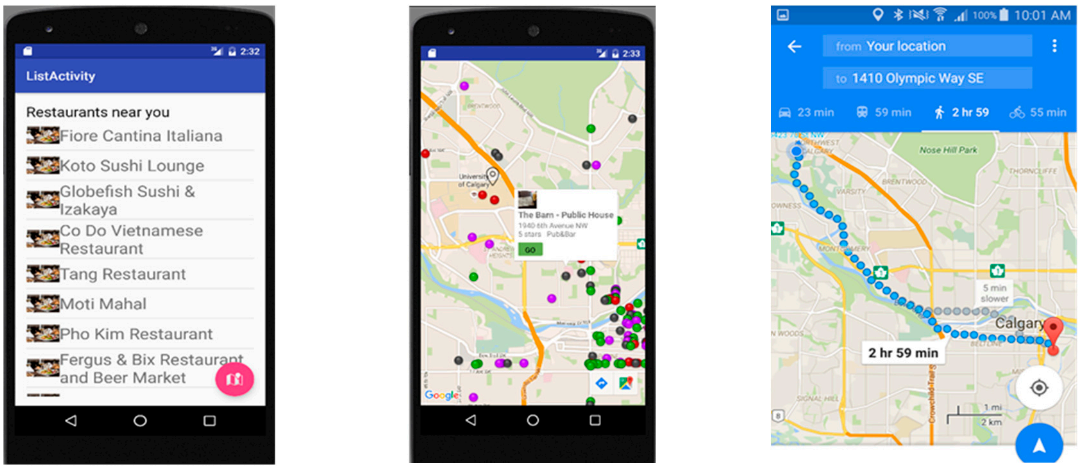
ET: 1 ms

(a) (b)

**Figure 5.** Detailed structure of the android-based Co-ARS designed in this study: (a) The search page of the CoARS application, (b) Main activity interface in the Co-ARS. The "Find a restaurant" radio group search function selects only the required restaurant type, and the "MY RECOMMENDATIONS" function returns restaurants with high synthetic recommendation indexes. Both functions lead to a list activity.

**Map activity:** The Map activity used the Android GoogleMap library to display a base map of the area around the user. Markers have been placed for each restaurant location. Data such as restaurant rating, type, and address, as well as its thumbnail, are shown in the information window. Each information window can be clicked, and the user will be taken to the Google Maps application. The Google Maps application is automatically configured to zoom in to the restaurant. The user may then choose to view the surrounding area or get directions to the restaurant.

Figure 6-a shows the recommended restaurants sorted by the synthetic recommendation index (Equation 1). The Map activity can be reached either from the List activity via clicking on a specific item, or through the "view map" button in the Main activity. In the map activity, when the user clicks on a specific marker on the map, a pop-up window shows up with the basic information of the restaurant, including name, address, rating and thumbnail (Figure 6-b). The user can also trigger the Google Map App from the map activity, where detailed transportation plans will be provided based on the user's selection of time of departure, preferred routes and preferred transportation mode (Figure 6-c).


ListActivity

Restaurants near you

Fiore Cantina Italiana

Koto Sushi Lounge

Globefish Sushi & Izakaya

Co Do Vietnamese Restaurant

Tang Restaurant

Moti Mahal

Pho Kim Restaurant

Fergus & Bix Restaurant and Beer Market

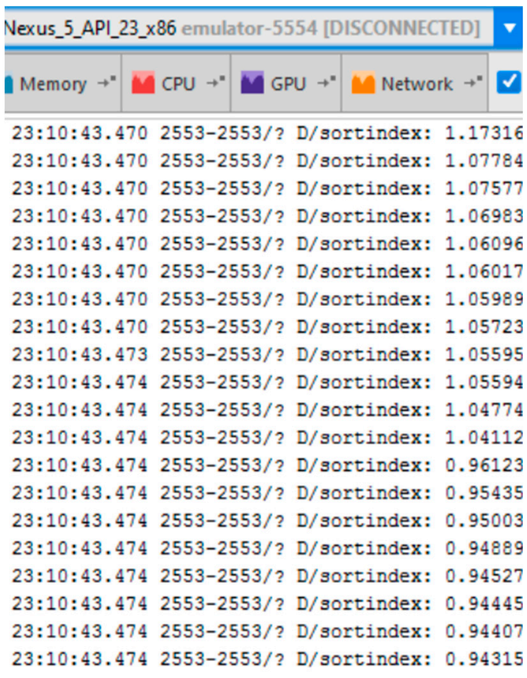
The Barn - Public House
1940 6th Avenue NW
5 stars Pub&Bar
GO

from Your location
to 1410 Olympic Way SE
23 min
59 min
2 hr 59
55 min
2 hr 59 min
5 min slower

(a) (b) (c)

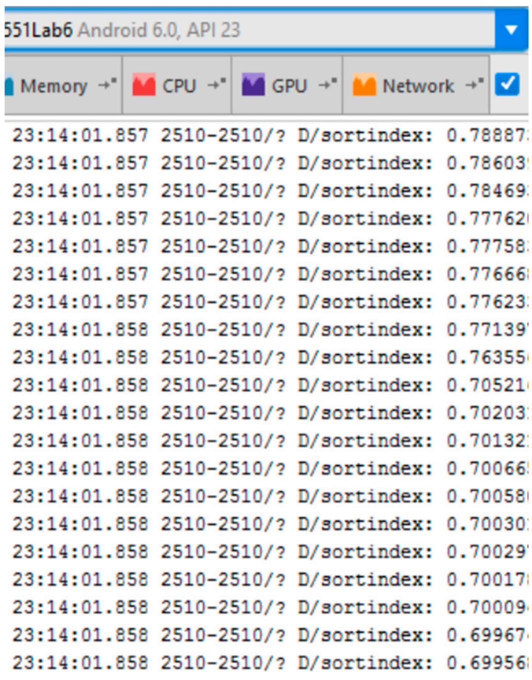
**Figure 6.** The results for a recommendation from CoARS: (a) User Interface: List activity of recommended results sorted by the synthetic index, (b) User Interface: Map activity of the recommended restaurants sorted by the synthetic index. The pop-up window lists the details of the selected restaurant, (c) Direction from the user's location to the chosen restaurant using Google Maps Integration.

In the control panel, user can choose the number of recommendations, or choose recommending something else with the button "I might interest in other". In addition, a historical table records user's choices. Based on the use's selection of the restaurant the database gets updated. In Figure 7, the results of the CoARS is presented before and after considering the context information index.



```
23:10:43.470 2553-2553/? D/sortindex: 1.17316
23:10:43.470 2553-2553/? D/sortindex: 1.07784
23:10:43.470 2553-2553/? D/sortindex: 1.07577
23:10:43.470 2553-2553/? D/sortindex: 1.06983
23:10:43.470 2553-2553/? D/sortindex: 1.06096
23:10:43.470 2553-2553/? D/sortindex: 1.06017
23:10:43.470 2553-2553/? D/sortindex: 1.05989
23:10:43.470 2553-2553/? D/sortindex: 1.05723
23:10:43.473 2553-2553/? D/sortindex: 1.05595
23:10:43.474 2553-2553/? D/sortindex: 1.05594
23:10:43.474 2553-2553/? D/sortindex: 1.04774
23:10:43.474 2553-2553/? D/sortindex: 1.04112
23:10:43.474 2553-2553/? D/sortindex: 0.96123
23:10:43.474 2553-2553/? D/sortindex: 0.95435
23:10:43.474 2553-2553/? D/sortindex: 0.95003
23:10:43.474 2553-2553/? D/sortindex: 0.94889
23:10:43.474 2553-2553/? D/sortindex: 0.94527
23:10:43.474 2553-2553/? D/sortindex: 0.94445
23:10:43.474 2553-2553/? D/sortindex: 0.94407
23:10:43.474 2553-2553/? D/sortindex: 0.94315
```

**(a)**



```
23:14:01.857 2510-2510/? D/sortindex: 0.78887
23:14:01.857 2510-2510/? D/sortindex: 0.78603
23:14:01.857 2510-2510/? D/sortindex: 0.78469
23:14:01.857 2510-2510/? D/sortindex: 0.77762
23:14:01.857 2510-2510/? D/sortindex: 0.77758
23:14:01.857 2510-2510/? D/sortindex: 0.77666
23:14:01.857 2510-2510/? D/sortindex: 0.77623
23:14:01.858 2510-2510/? D/sortindex: 0.77139
23:14:01.858 2510-2510/? D/sortindex: 0.76355
23:14:01.858 2510-2510/? D/sortindex: 0.70521
23:14:01.858 2510-2510/? D/sortindex: 0.70203
23:14:01.858 2510-2510/? D/sortindex: 0.70132
23:14:01.858 2510-2510/? D/sortindex: 0.70066
23:14:01.858 2510-2510/? D/sortindex: 0.70058
23:14:01.858 2510-2510/? D/sortindex: 0.70030
23:14:01.858 2510-2510/? D/sortindex: 0.70029
23:14:01.858 2510-2510/? D/sortindex: 0.70017
23:14:01.858 2510-2510/? D/sortindex: 0.70009
23:14:01.858 2510-2510/? D/sortindex: 0.69967
23:14:01.858 2510-2510/? D/sortindex: 0.69956
```

**(b)**

**Figure 7.** Android monitor of recommendation index: (a) before refreshing and (b) after refreshing

*4.3. Implementation and Results of the web-based CoARS application*

The solution has been implemented and tested in Visual Studio using C#, JavaScript, HTML5 and CSS3 programing languages. The web version of CoARS was built with the simplest MVC (model-view-controller) framework where no professional MVC framework is used. "Model" was used to manage data and transfer data from the database into view. "View" was responsible for rendering the transferred data into the browser as a map and recommended result. The connection between the model and view, and various requests from users is handled by the controller. Figure 8 below indicates a complete life cycle of the application.

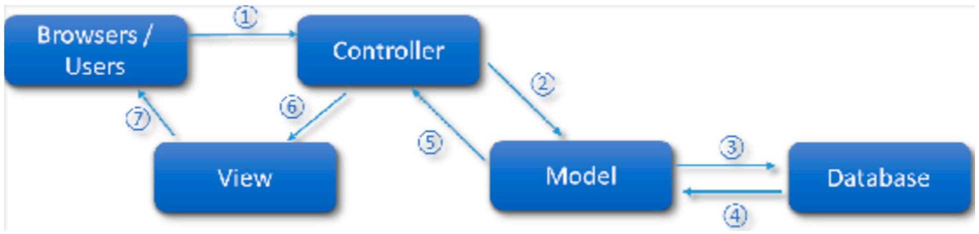


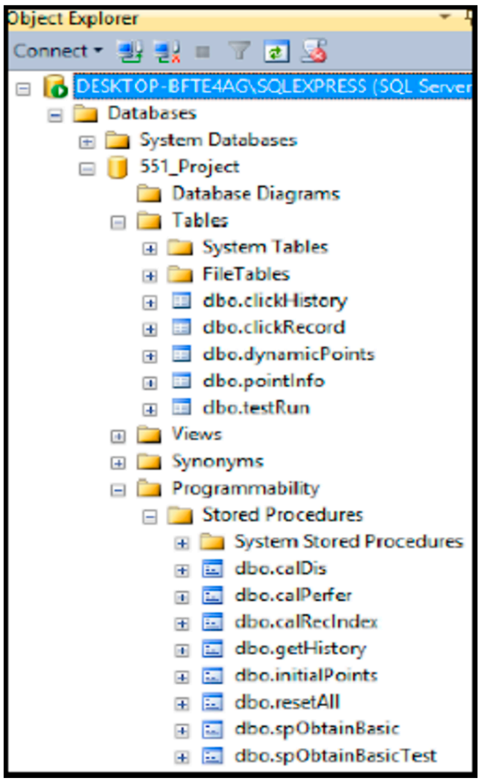
**Figure 8.** Complete life cycle of web-based CoARS implementation

According to Figure 8, user (or browser) sends requests to controller first, then controller connects the model to retrieve data. Model connects the database and returns obtained data back to the controller. Then, the controller passes the retrieved data to view, which is able to present those data as a map and recommended points into the browsers. Each component is illustrated as follows:

**Browsers:** Generally, the web version CoARS fits most mainstream browsers. The Google Chrome and IE 11 was used in debugging.

**Controller**: In the designed CoARS, the controller specifically refers to two JavaScript files called "MainJS.js" and "base.js" respectively. "base.js" contains some basic functions, such as DOM operations, and event responses. "MainJS.js" uses functions from "base.js" to connect the model and view.

**Model & Database:** "Model" and "Database" are the most important sections in this application as they implement the recommendation and provide data to the front end. Microsoft SQL Server 2012 was used as the database in this application. The model consists of two files: "model.js" and "WebService1.asmx". "model.js" includes all the jQuery.Ajax requests, which could ask data from web service page. Web service page is an essential component of the .NET Framework which is used as "WebService1.asmx" in this application. "WebService1.asmx" is responsible for connecting the database through ADO.NET technology.



```
DESKTOP-BFTE4AG\SQLEXPRESS (SQL Server
  Databases
    System Databases
    551_Project
      Database Diagrams
      Tables
        System Tables
        FileTables
        dbo.clickHistory
        dbo.clickRecord
        dbo.dynamicPoints
        dbo.pointInfo
        dbo.testRun
      Views
      Synonyms
      Programmability
        Stored Procedures
          System Stored Procedures
          dbo.calDis
          dbo.calPerfer
          dbo.calRecIndex
          dbo.getHistory
          dbo.initialPoints
          dbo.resetAll
          dbo.spObtainBasic
          dbo.spObtainBasicTest
```

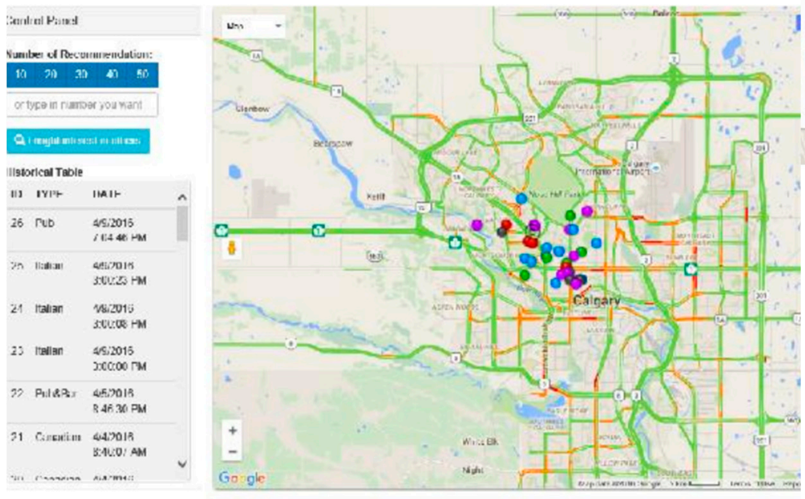
**Figure 9.** Screenshot of the database structure

In the Microsoft SQL Server, all the data used in this application is stored in a database, see Figure 9. Table "dbo.pointInfo" contains all the information about restaurants, including name, customers' rating, type, picture URL, address, and coordinates. Table "dbo.clickRecord" and "dbo.clickHistory" store the current user's choice and the time when the user chose it. Also, some procedures were created in this database. The procedure is used to execute certain SQL commands. In this application, those procedures are used to calculate the context aware distance between the user's current location and each restaurant, calculate the user's preference and the final recommend index. In addition, the procedure "resetAll" is used to clean the history of user's choice and preference.

Generally, when the controller sends a data request to model, "model.js" will identify the request and sends this request to "WebService1.asmx" through jQuery.Ajax technology. Then the "WebService1.asmx" opens the connection with the database, calls certain procedures in the database and passes the corresponding parameters. The database receives those requests, executes corresponding procedures, and returns requested data to "WebService1.asmx". At last, "WebService1.asmx" passes the retrieved data back to "model.js" so that controller can use those requested data.

**View:** View is responsible for rendering all the data onto the screen (or browser). The most basic components in view are HTML elements and CSS style. In this application, the Google Map API for JavaScript and Bootstrap CSS Library are also used. Google Map JavaScript API can show a base map on the webpage, and convert data into points on the map. In addition, the pop-up information window, service of current traffic condition, and service of direction are provided by Google Map API. Once controller get the requested data from model, it will pass the data to view, which can dynamically change the contents of browser.

Web-based CoARS implementation result is shown in Figure 10-a. This figure displays the main page of the web version application. The map and recommended restaurant are listed on the right side, and a control panel is located on the left side. In the control panel, the user can choose the number of recommendations, or choose to recommend something else with the button "I might interest in other". In addition, a historical table recorded user's choices are below those buttons. User can click each point on the map to see more information about corresponding restaurant in the pop-up information window, like name, user's rating, location and picture, see what showing in Figure 10 below. The button says "Go There" could be clicked if the user decides to go to this restaurant, and the corresponding direction would be displayed on the map, see Figure 10-c.


Control Panel

Number of Recommendation:

10 20 30 40 50

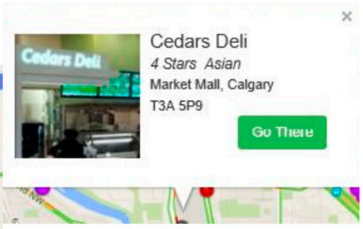
or type in number you want

I might interest in others

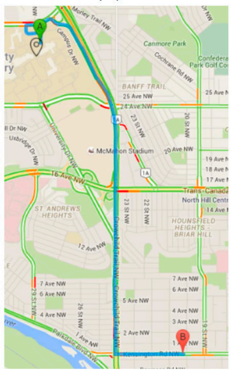
Historical Table

| ID | TYPE | DATE |
|---|---|---|
| 26 | Pub | 4/9/2016 7:04:46 PM |
| 25 | Italian | 4/8/2016 3:00:23 PM |
| 24 | Italian | 4/8/2016 3:00:08 PM |
| 23 | Italian | 4/9/2016 3:00:00 PM |
| 22 | Pub&Bar | 4/5/2016 8:46:30 PM |
| 21 | Canadian | 4/4/2016 8:40:07 AM |

(a)


Cedars Deli
4 Stars Asian
Market Mall, Calgary
T3A 5P9

(a)



(c)

**Figure 10.** Screenshots of the results of using web-based CoARS: (a) Main page of the Web-based CoARS implementation, (b) b) Pop-up information window, (c) Direction to the chosen restaurant.

## 4. Conclusions

In this study, the context-aware recommendation system was designed for Calgary restaurants, in the form of a mobile and web application to help citizens and tourists. The app incorporates various APIs and databases (Google Maps Distance Matrix API, SQLite and etc.), and uses the context information of the users and the restaurants' ranking to generate the optimal recommendation results. Personalized context information utilized by this app includes: user's browsing history, transportation mode, the network distance between the user's current location and the destination and the overall rating of the restaurant. The user's browsing history was stored in a local database, which could be dynamically refreshed after the recommended items being checked out.

The evaluation of the CoARS application has been conducted successfully with recommendation experiment and usability test. However, this app still can be improved regarding the inclusion of more context information of both users and the restaurants. For future work, we would like to retrieve users' group preference information from users' social media (e.g. Facebook, twitter etc.), and incorporate that information to recommend restaurants to a group of friends. Furthermore, data mining techniques for classification (e.g. Bayesian Believe Network, Support Vector Machine etc.) could be utilized to create more effective recommendations. Last but not least, instead of using a static database for restaurants' context information, we would like to connect our app dynamically to Yelp, so that any changes in the restaurants' rating and locational information will be observed by the recommendation index generator which will provide the users with up to date results.

**Acknowledgments:** This research was supported in part by research funds to Dr. Steve Liang from the Natural Science and Engineering Research Council of Canada (NSERC).

**Author Contributions:** This paper is part of a research work done by Dr. Sara Saeedi and other authors under the supervision of Dr. Steve Liang. Dr. Sara Saeedi conceived, designed and implemented the proposed CoARS and analyzed the data; Xueyang Zou and Mariel Gonzales has predominantly participated in the recommendation system algorithm, performing the experiments using mobile and web app. Sara Saeedi wrote the paper and other authors contributed regarding final paper structure and reviewing it.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Adomavicius G., Tuzhilin A. Context-Aware recommender Systems, in: F. Ricci, et al. (Ed.), Recommender Systems Handbook, 2011, pp. 217–253.
2. Saeedi, S.; Moussa, A.; El-Sheimy, N. Context-Aware Personal Navigation Using Embedded Sensor Fusion in Smartphones. *Sensors* 2014, 14, 5742–5767
3. Saeedi, S.; El-Sheimy, N. Activity Recognition Using Fusion of Low-Cost Sensors on a Smartphone for Mobile Navigation Application. Micromachines, 2015, Vol 6, no.8, pp. 1100-1134.
4. Zou X., Gonzales M., Saeedi S. 2016. A Context-aware Recommendation System using smartphone sensors, 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference
5. Carmen M.; Llarri S.; Trillo-Lado R.; Hermoso R. Location-Aware Recommendation Systems: Where we are and where we recommend to go, LocalRec'15, Vienna, Austria, 2015.
6. Aggarwal, Charu C. *Recommender systems*. Springer International Publishing, 2016.
7. Zhou, Dequan, et al. "A Study of Recommending Locations on Location-Based Social Network by Collaborative Filtering." *Canadian Conference on AI*. 2012.
8. Cui G.; Luo J.; Wang X. Personalized travel route recommendation using collaborative filtering based on GPS trajectories, *International Journal of Digital Earth*, 2017, Pages 1-24
9. Levandoski J.J.; Sarwat M.; Eldawy A.; Modbel M.F. LARS: A Location-Aware Recommendation System, 2012.
10. M.H. Park, J.H. Hong, S.B. Cho, Location-based recommendation system using Bayesian user's preference model in model devices, UIC 2007, LNCS 4611, 2007, pp. 1130-1139.
11. Ricci F.; Rokach L.; Shapira B.; Kantor P.B. *Recommender systems handbook*, 2011.
12. M. Sarwt, J.J. Levandoski, A. Eldawy, M.F. Mokebel, "LARS*: an efficient and scalable location-aware recommender system", Transactions on Knowledge and Data Engineering, vol. 6(1), 2011.
13. Guo,G., Zhang,J., Yorke-Smith, N. 2013. A novel Bayesian similarity measure for recommender systems.
14. Michael D. Ekstrand , Praveen Kannan , James A. Stemper , John T. Butler , Joseph A. Konstan , John T. Riedl, Automatically building research reading lists, Proceedings of the fourth ACM conference on Recommender systems, September 26-30, 2010, Barcelona, Spain [doi>10.1145/1864708.1864740]
15. Madadipouya, Kasra, and Sivananthan Chelliah. "A Literature Review on Recommender Systems Algorithms, Techniques and Evaluations." *BRAIN. Broad Research in Artificial Intelligence and Neuroscience* 8.2 (2017): 109-124.
16. Lathia, Neal, Stephen Hailes, and Licia Capra. "Trust-based collaborative filtering." *IFIP International Conference on Trust Management*. Springer, Boston, MA, 2008.
17. Guo, Guibing, Jie Zhang, and Neil Yorke-Smith. "A Novel Bayesian Similarity Measure for Recommender Systems." *IJCAI*. 2013.
18. Levandoski, Justin J., et al. "Lars: A location-aware recommender system." *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE, 2012.
19. Levandoski, Justin J., et al. "Lars: A location-aware recommender system." *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE, 2012.
20. Yang W.S.; Cheng H.C.; Dia J.B. A location-aware recommender system for mobile shopping environments, *Expert Systems with Applications*, vol.34, 2008.

21. Khoshnood F.; Mahdavi M.; Sarkaleh M.K. Designing a Recommender System Based on Social Networks and Location Based Services, *International Journal of Managing Information Technology*, 2012, ISSN 0975-5926,Vol. 4;Issue: 4; Start page: 41.
22. Savage N.S.; Baranski M.; Chavez N.E.; and Höllerer T. I'm feeling LoCo: a location based context aware recommendation system," Advances in Location-Based Services, pp. 37–54, 2012.

Ye, Mao, et al. "Exploiting geographical influence for collaborative point-of-interest recommendation." *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 2011.