tagMeAR : Context-Aware Mobile Recommendation System

Jun-Cheng Chen*
Department of Computer Science
University of Maryland
College Park, MD 20742
pullpull@cs.umd.edu

Miguel Rios Berrios
Department of Computer Science
College Park, MD 20742
mrios@cs.umd.edu

Tak Yeon Lee
Department of Computer Science
College Park, MD 20742
tylee@umd.edu

Abstract

This paper presents *TagMeAR* a context-aware mobile recommendation system for improving quality of life on university campus. TagMeAR is a mashup service that combines existing campus information services. Its recommendation engine uses Markov Logic Network to provides users only relevant information according to current context and previous usage. With Augmented Reality technology running on Android mobile phone, TagMeAR's information browser shows information overlaid on live image of physical space by using built-in camera, compass and GPS module.

1 Introduction

Over 185,000 apps are being sold in the App Store only for mobile devices from Apple, April 2010. The commercial of iPhone 3G "there is an app for that" might be true, however, the skyrocketting number of apps brings a new set of problems. First, it gets harder and harder to find an appropriate app by using only keyword search, categorial browsing and users' rates. Users are often frustrated when realizing the app they just bought does not give what they expected, and the smartphone is full of icons rarely used after installing it. The second problem is increasing access time to the right information of a specific app. As frequently being used with just one hand while doing other tasks, one of important goals in smartphone user interface design is to minimize operational steps. Once it seemed to be achieved by making tiny apps with very specific goals, though, browsing more than a hundred apps made it bothersome. To address these problems, we put an additional layer of application called TagMeAR that shows, recommends and manages information retrieved from other apps.

TagMeAR is designed to support students and faculty members in the campus at University of Maryland. Thus, information from campus web applications such as parking lot information, shuttle bus schedule and event mailing list, are aggregated and delivered to users via TagMeAR system. In order to prevent information overburden, TagMeAR's recommendation engine rates each piece of information by its properties. As most information about university campus are encoded with time and location, TagMeAR can decide which information to deliver with its time and location properties, but how to set and manage rules for different services is not a straightforward problem. Instead

*The three authors equally contributed to the project.

of predefining rules for every services manually, TagMeAR's recommendation engine generates rules by learning from users previous behavior - *which services are used in what context* - by using Markov Logic Network (MLN).

In addition to the recommendation engine, TagMeAR utilizes Augmented Reality (AR) technology for intuitive representation of information. TagMeAR accomplishes it with built-in compass and GPS module by overlaying information at the exact location on the live image from its camera. AR user interface offers location-based query functionality which is useful when user is interested in a specific location but has no clue of search keywords or adequate apps. Another benefit of using AR user interface is that it is suitable for being used while walking because it doesn't require any accurate input from user's fingertip.

TagMeAR consists of three components - AR information browser, Recommendation engine and Service provider. First of all, AR information browser shows useful information to user, collect user's current context entities, and send it to the Recommendation engine. Second, the Recommendation engine running on the TagMeAR server receives user's current context from the Client application, and requests information from the Service wrappers. Finally, Service provider connects to all the services(either HTML page, RSS feeds or web API) and generate pieces of information formatted for the Recommendation engine.

The rest of the paper is organized as follows. In Section 2, we discuss related works. Section 3 explains TagMeAR system components that consists of AR information browser, recommendation engine and service provider. Section 4 explains implementation of TagMeAR prototype with screenshots. Section 5 contains our conclusion.

2 Related Work

TagMeAR project intersects with many research areas - context-awareness, campus information system, recommendation engine, and Augmented Reality user interface. To our knowledge, there is no research project that has exactly the same position.

MyeVyu[1] is an integrated campus information system at the University of Maryland running on mobile devices. Although emphasizing safety on campus, MyeVyu provides users context-sensitive, location-aware information such as current weather, campus events, location of vending machines, directions anywhere on campus, support for disables access and paths to follow. Not just providing static information to users, MyeVyu also offers shortcuts to services which means that user can contact a Police Dispatcher by simple pressing a button his/her mobile phone for example. TagMeAR shares the basic idea of MyeVyu, using user's context to customize information being delivered, however, TagMeAR's information browser shows the overview of currently available information instead of presenting many icons.

Magitti[2] is another context aware travel guide running mobile devices that suggests local venues for shopping and other activities based on location, time of day, preferences and past behavior. By combining these context entities of a user in its Personalized Activity-related Recommendation System, Magitti predicts what kind of activity (Shopping,Seeing, Doing and Reading) the user will do, and rates shops nearby for each activity. Although focusing on different kind of services, TagMeAR's recommendation system shares many aspects of Magitti's recommendation. However, we claim that TagMeAR has more extensibility than Magitti, because TagMeAR is designed to hold any kind location related information services while Magitti is restricted to recommend only shops relevant to four pre-defined activities. In terms of user interface, Magitti uses a map to provide locations of recommendation, while TagMeAR's information browser has AR view which is more intuitive when showing relative orientation.

A lot of smartphone apps using AR technology are announced last year(2009). For example, Acrossair's nearest tube[1] shows metro stations nearby, Peak.AR by Salzburg Research[2] provides the name and the height of the mountains and hills around user. While most mobile AR apps provide only information for a specific topic, Layar reality browser[3] mashes up any information with geolocation

[1] http://www.acrossair.com

[2] http://peakar.salzburgresearch.at/

[3] http://www.layar.com/

tags from multiple sources such as Wikipedia, Tweeter and flickr. A challenge that Layar would face is that the small screen on mobile device can be quickly filled up with icons, while TagMeAR's recommendation engine aims to resolve the problem.

3 TagMeAR System

3.1 Recommendation Engine

A plenty of various services are available around our living environment, but few of them suffice our needs indeed. This poses a serious problem for users finding the right application for the given situation in tolerable time. Thus, the recommendation engine of tagMeAR system aims to assist users through this problem. On the other hand, the recommendation results (e.g. ranks and recommendation scores of services) can be further used to improve the limited size problem of display screen of mobile devices by the filtering function when many superimposed service icons occupy or mess up the device's screen. In this project, we implement the recommendation engine using Markov Logic Network to model the relationships between given situations and services selected so as to generate recommendation scores for each service. More implementation details are described in the following sections.

$$\forall x \; Smokes(x) \Rightarrow Cancer(x)$$
$$\forall x, y \; Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$$

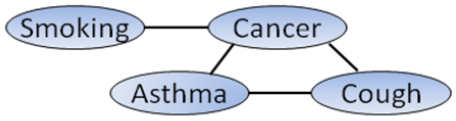

Figure 1: (above) an example of first-order logical formula. (below) an illustration of Markov network for modeling the relationship between smoking and cancer and symptoms (asthma and cough) where x, y are variables which could be substituted by human (e.g. Smokes(Anna) means Anna smokes, and Friends(Anna, Helen) means Anna and Helen are friends.)

Markov Logic Network

Nowadays, most recommendation systems adopt machine learning algorithms to learn user preference model so as to offer users better recommendations as more user data are available. However, those model-based systems seriously suffer from the cold start problem. That is at the very beginning stage no enough data available to build an accurate user model. Therefore, a set of simple heuristic rules from domain knowledge are needed as the initial model to collect enough user data. Fortunately, recent advancement on the research area of statistical relational learning introduces a novel tool, Markov Logic Network (MLN) [4][3], which harnesses logic rules (handling complexity) and statistical learning theory (handling uncertainty) at the same time. MLN allows us to define a set of rules using first-order logic formulas consisting of variables, and predicates, and we are able to specify weight (i.e. in the range [0, 1]) for each rule. More importantly, MLN framework enables us update weights of each rule and even learn new rules as more training data are provided. That is also the main reason why we use MLN for recommendation engine to leverage the power of domain knowledge and information from data. The following is an example of first-order logic formula and a Markov network.

Implementation

In our implementation, because of no data available initially, we have to generate fake data (5000 instances) instead based on heuristic rules in our prototype system for testing purpose. The data contexts we use to model the different situations are currently (1) location (i.e. GPS data with latitude and longitude), (2) time (i.e. date, and current hour/min/sec), and (3) commuting pattern

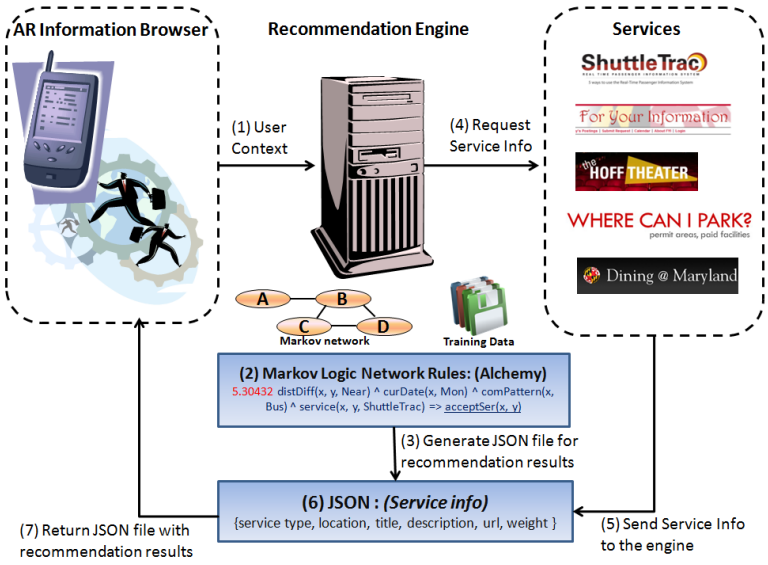

Figure 2: An overview of how our recommendation engine works

(i.e. walk, car, metro, shuttle, and none). Given current available services (i.e. shuttleTrac, fyi, dining, Hoff theater, and parking, more details in service section.) in the prototype system, these three data are enough to model situations at current stage. (i.e. more user centric context including user background, education, etc will be considered to be added when more services are included in our system.)

Then, using fake data, we can train a MLN model, and then attain a test data from our mobile device when users attempt to use tagMeAR for service information, the MLN will generate the probability for each service, our so called recommendation score, to denote how possible each service to be selected. The MLN implementation we used here is open source software, Alchemy[4], developed by the research team of University of Washington. Alchemy allows user to define the first-order formula in the .mln file and training/testing data in .db file. The example formula with soft weight is shown as follows.

```
5.30432 distDiff(x, y, Near) ^ curDate(x, Mon) ^ comPattern(x, Bus) ^
service(x, y, ShuttleTrac) => acceptSer(x, y)
```

where (a) distDiff, (b) curDate, (c) comPattern, (d) service, and (e) acceptSer are predicates and denotes (a) the distance difference between the service and the user, (b) current date, (c) commuting pattern, (d) service ID, (e) which service will be accepted, respectively. The formula mainly describes the scenario:

> "In situation x, where a person is **near to the bus stop y**, today is **Monday**, and his commuting pattern is **Bus**, then he probably (5.30432) wants **bus schedule**."

After finishing recommendation processing, based on recommendation scores, the engine could decide to services to be recommended and encode services information into JSON format and return it back to the mobile device for visualization. The detail of our data field used in JSON are shown below.

```
{ 'type':serType,
                  'location':{
                              'latitude':geoLocation[0],
                              'longitude':geoLocation[1]
                             },
                  'title':title,
                  'description':desc,
```

[4]http://alchemy.cs.washington.edu/

```
 'url':url,
 'weight':weight,
}
```

where type records service ID, location records latitude/longitude, title and description record service information, url records the link to service website, and weight records the recommendation score. 2 briefly give an overview of how our recommendation engine works: (1) Acquire user context (location, time, commuting pattern) from the mobile device, (2) Use pre-trained Markov Logic Network to generate recommendation scores for each service given situation information from user context, and step (3)(4)(5)(6)(7) Generate JSON file and return information (including recommendation score) of all of selected services satisfying our rules with recommendation scores.

3.2 Service Provider

TagMeAR is a mashup service that combines campus information from external sources on the internet. In this project we used five examplary sources - *ShuttleTrac, Parking Lot information, For Your Information(FYI), Hoff Theater movie schedule, and South Hall dining menu* - but adding more services is very easy. The Service Provider sends queries to those sources and parses the result to extract metadata (event title, description, location, time and other tags). With extracted location information which is a building name in general, the distance from user is added to the event's metadata.

3.3 Augmented Reality Information Browser

The Augmented Reality Information Browser is a mobile application designed for devices running the Google Android Operating System. The prerequisites to use the application are an device running Android 2.1 with an integrated camera, GPS, compass and a data connection. Almost any Android device in the stores right now comply with these requirements already. In the forthcoming section we will be giving you a walkthrough of the application. For further and technical details about the application please refer to section 4.

The application has a tab-based interface with three main components: the list view, the augmented reality view and the options view. All these options have different, but related functionality. As an overview: the list view contain the recommended services in an ordered way, the augmented reality view contain the same recommended services in a AR browser and the options offer different customizations to the user.

The List View

This view is the first screen that the users see when opening the application. It contains a list of items, or services, that were recommended by our backend recommendation service. Each service appear in a block with its icon in the left side and its title in the left side. 3(left) show an overview of the list view listing shuttle trac and FYI services that were recommended to the user. See 3(middle) for details of each service.

The user can interact with each service in the list view by touching it. A window with the full name of the service, its icon (which represents the service type) and the details are show in screen. The details block is scrollable, allowing the user to get extra information fro that specific service. This view also offers two options to the user, a like and a unlike button, which are currently unimplemented, but in a future work, can be attached to the recommendation service to help it "learn" the preferences of each user.

The Augmented Reality Browser

In this view, the user can browse the recommended services through an augmented reality interface. It uses the built-in camera to display the current view and overlap the icons of each service in their correspondent geographical location, relative to the user's location. 4(above) shows the interface with all the recommended services. The interface has a distance filter in the right, as a slider, to filter out services that are not close enough, as seen in 4(left) where only the three icons that are closer to the user are shown.

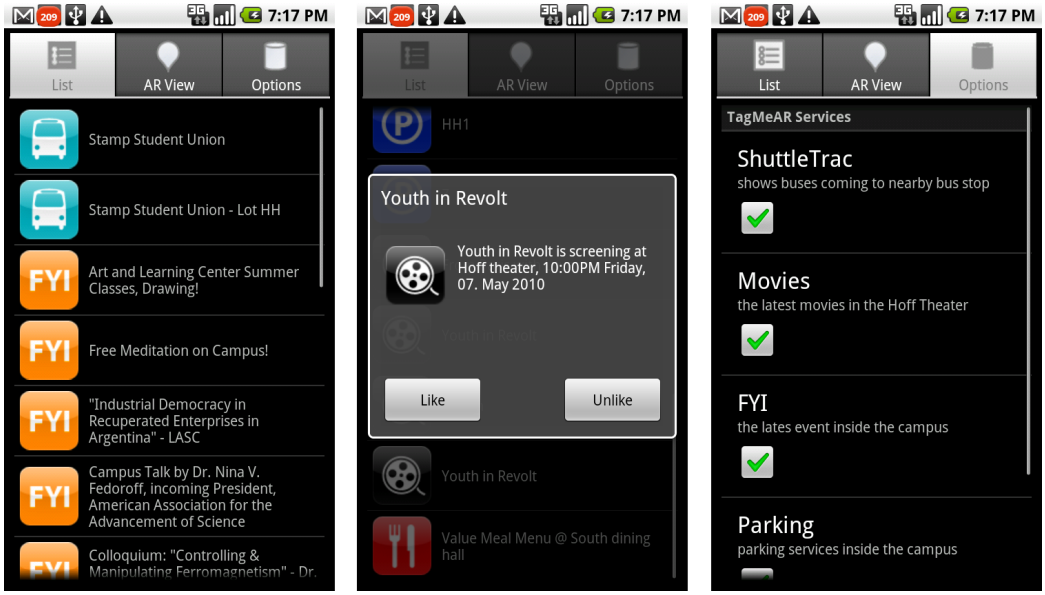

Figure 3: (left)List view, (middle)Detailed information, (right)Option

Figure 4: (above)Augment Reality Information Browser, (left)icons filtered by distance, (right) detail information

When pressing on a service, a window pops up in the screen with the full name of the service and its description. This window also offer two important elements: a button to access the url of the service in an embedded web browser and the capability to browse other services by swiping the finger into

the window, which helps the user to browse in an easier way when there are too many services in a small area. 4(right) display this scenario, where the user pressed in a FYI event to see its details.

The options view, seen in 3(Right), display to the user services-wide filters. The user has the ability to choose which services to display and which services to hide, depending on his or her tastes or current interests.

4 Implementation

Our system implementation was done in modules to improve the efficiency of the development without depending in other components of the system. The main components of our system are, as detailed above, the recommendation system, a provider of on-campus services and the augmented reality information browser. All those components communicate to each other by using web services implemented in JSON. This was done to maintain a consistent protocol and for simplicity of parsing in the Android end.

At first, these three components were implemented individually. The recommendation system, written in python, was hosted in a local server and the input, location coordinates and time, was emulated manually. In the services provider end, the development was made ad-hoc too, before connecting it to the recommendation system and then to the Android application.

The mobile application development process was simple. Android OS provides a simple framework that makes the development a straightforward process. We used the Wikitude API to support our augmented reality information browser. In the AR end, the development was done simply by calling Wikitude methods and managing user inputs and other events. On the other hand, the connection to the web server was not so straighforward due to complications we had by migrating the cgi scripts written in python to the hosted server.

The mobile application makes a call to the web server via a HTTP request and send the coordinates of the GPS in the phone to the recommendation system. The system then connects to the services provider, get the events and then decide which events it will send back to the phone. The phone receives a JSON file with the services that were recommended and they it display it in the current view (either the list feed or the augmented reality feed). All this was implemented by using Android OS and Java libraries, without having to go to the underlaying connectivity or data management issues.

Testing of this implementation was limited due to time constrains, but building unit testing modules would be simple due to the modularity of our project. The problems faced during our limited testing session were quicky solved by identifying the source (module) causing the problem, and by fixing it straight from either the web server (in the recommendation or service provider systems) or directly into the IDE used to develop the Android application.

5 conclusion

TagMeAR is a system designed to support the college community in the University of Maryland. It is intended to be able to find events and services based in the users' preferences. TagMeAR also gives the users the ability of look at those events in a different and more intuitive way by using novel methods to show and browse the information.

By using augmented reality technology, TagMeAR displays to the user these recommended services and events in their physical space, by using the built in camera, compass and GPS location of an Android device. The device has the support of a web server, which running a recommendation engine and a service provider, is able to recommend events and services to the user by using state-of-the-art methods and algorithms by using context, specifically previous preferences, geographical location and time of the request. All these components were designed for this project.

Acknowledgments

We deelply thank to Neha Gupta for helping us with her knowledge in recommendation system.

References

[1] Mind lab: Myevyu project(website, visited 5/13/2010). http://mindlab.umd.edu/myevyu/home.php.

[2] V. Bellotti, B. Begole, E. H. Chi, N. Ducheneaut, J. Fang, E. Isaacs, T. King, M. W. Newman, K. Partridge, B. Price, et al. Activity-based serendipitous recommendations with the magitti mobile leisure guide. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, page 1157?1166, 2008.

[3] P. Domingos. Whats missing in AI: the interface layer. *University of Washington, Washington, USA*, 2007.

[4] P. Domingos, S. Kok, H. Poon, M. Richardson, and P. Singla. Unifying logical and statistical AI. In *Proc. of AAAI*, volume 6, page 2?7, 2006.