Introducing Community Awareness to Location-Based Social Networks

Pavlos Kosmides(✉), Chara Remoundou, Ioannis Loumiotis, Evgenia Adamopoulou, and Konstantinos Demestichas

Institute of Communication and Computer Systems - ICCS, National Technical University of Athens, Zografou, Athens, Greece
{pkosmidis, chremoundou, i_loumiotis, eadam, cdemest}@cn.ntua.gr

Abstract. During the last years, Social Networks have been in the spotlight of many researchers, trying to enhance them with pervasive features that will simplify and facilitate users' experience. One of the most innovative additions to social networks has been the introduction of communities in users' lifecycle. However, there are still a lot of issues regarding the automation of this feature in order to minimize user's effort to discover new communities and as a result, to improve his experience. In this paper, we introduce the use of communities in location-based social networks. We also present the proposed systems architecture including Processes and Services.

Keywords: Communities · Social networks · Location-Based · MVC

1 Introduction

The concept of "Virtual Communities" (VC) based on social networks is now widely accepted by millions of users worldwide. Many networking platforms have been created, more or less differentiated from classical established networks, like Facebook [1] and Twitter [2] attracting people with common interests and pursuits, thus building virtual communities of users.

The MVC (Mobile Virtual Community) is the natural evolution of these communities, combining the features of a VC with the services offered by a smart mobile device (smartphone) or tablet. As the usage of smartphones/tablets has increased and the offered services have been improved, the MVC has ceased to be simply the mobile version of VC. Instead, it has evolved and gained its own momentum.

There has been numerous works that deal with the existence of communities in social networks and how we can take advantage of them. In [3], the authors view community members relationships as cliques in a graph that represents the social network. They also study members behaviour in communities and how their behaviour depends on the communities that they belong to. Similarly, in [4] the authors identify communities in social networks, based on interests and activities of users. Finally the authors in [5] propose a model based on social theory, in order to find communities in dynamic social networks. They present a social cost model and formulate an optimization problem in order to find the community structure from the sequence of arbitrary graphs.

DOI: 10.1007/978-3-319-19743-2_19

A survey has been conducted in [6] where the authors recognize the importance of communities as a feature of social networks. They also state the need for discovering communities in social networks and present some proposed approaches. They also introduce various types of social network and suggest a classification for community detection methods based on the type and nature of social networks.

In this paper, we combine the location-based social networks with the evolutionary concept of MVCs. Specifically we propose an innovative system architecture that takes advantage of the existence of MVCs, and adapts the popularity of nearby locations according to the communities that a user belongs. In addition, user preferences for specific locations, including the number of checkins as well as user's rating, are being processed in order to suggest MVCs to each user.

The following sections present in detail the designed architecture for the implementation of the abovementioned service, including foreseen components and the specified interactions among them. The architecture is designed using ArchiMate® [7] showing the application layer entities (application functions) and their relationship.

2 Popularity Options

The *Popularity options* service is responsible for collecting users' preferences, regarding the results that users expect to receive. Users can define the average age in order to display the results of popularity that relate to specific age groups. They may also choose to show results depending on the daytime period (morning, afternoon, evening) or choose a time window in which to limit the search results to the application. Finally, users are able to choose whether the results of the application will depend on a list of communities they have selected and they are registered to.

Figure 1 states the main application functions and components together with relevant interfaces and information flows for the *Popularity options* service. As shown, the following functions have been identified for the application layer:

- *Retrieve Avg Age:* this application function retrieves the average age that the user is interested in, in order to adapt the popularity list of the nearby locations.
- *Retrieve Time Criteria:* this application function retrieves the time criteria the user is interested in, in order to show the popularity list according to the daytime period (morning, evening, night).
- *Retrieve MVCs:* this application function retrieves the list of MVCs that the user is part of and desires to include them to the computation of the popularity statistics. In this way, the popularity list of the nearby locations will be adjusted according to other users' rates that belong in the same MVCs.
- *Retrieve Time Interval:* the time interval that the user wants to adjust the popularity statistics is retrieved with this function. For example, the user might be interested in producing a popularity list with ratings that took place during the last week or month.
- *Send to platform:* this function is responsible to communicate with the remote platform and to send users' preference in order to be elaborated.

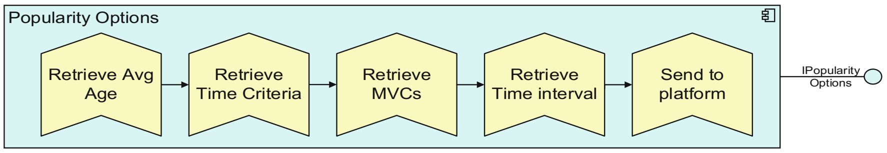

Fig. 1. Popularity options – main application components, functions and interfaces.

3 MVC Aware Machine-Learning Engine Training

The *MVC aware machine-learning engines training* service is responsible for the centralized training of machine-learning engines which will be used by the *MVC Suggestions* service (described in Sect. 4). The centralized training is made on the platform side.

Figure 2 summarizes the main application functions and components together with relevant interfaces, main data objects and information flows for the *MVC aware machine-learning engines training* service.

As can be seen from the diagram presented, the application functions can be divided into two categories. The following application functions have been identified for the first category:

- *Retrieve CheckIns per MVC:* this application function retrieves the checkIns that have been made to one location from members that belong to a specific MVC. This is made for all available MVCs and data are retrieved from the MVCDB (MVC Database).
- *Retrieve Rating per MVC:* this application function retrieves the ratings that have been made to one location from members that belong to a specific MVC. This is made for all available MVCs and data are retrieved from the MVCDB.
- *Generate MLE:* this application function is responsible for the creation of the Machine-Learning Engines that will be used for the training process.
- *Retrieve Relevant Training Datasets:* this application function retrieves the relevant dataset that will be used for the training process from the MLTDB (Machine Learning Training Database).
- *Select training options:* this application function selects and defines the training options for training the datasets.
- *Perform MLE Training:* MLE training is performed on this application function with regard to the retrieved training datasets and the options that were defined in the previous application functions.
- *Store MLE:* the resulting MLEs are stored in the MLEDB (Machine Learning Engine Database) through this application function.

The above mentioned application functions are implemented by the application component named *MLE Training Execution*.

The application functions of the second category are:

- *Periodic Triggering:* this application function periodically sends messages to trigger the Machine-Learning Engine Training.

- *Retrieve Manual Trigger:* this application function allows manual triggering of the Machine-Learning Engine Training.
- *Retrieve events:* this application function triggers the Machine-Learning Engine Training based on events.
- *Trigger MLE Training Execution:* this application function is responsible for triggering the MLE Training Execution component, after it receives the appropriate message from the above functions.

These application functions are implemented by the application component named *MLE Training Scheduler*. The main purpose of this component is to initiate the generation of new MLEs.

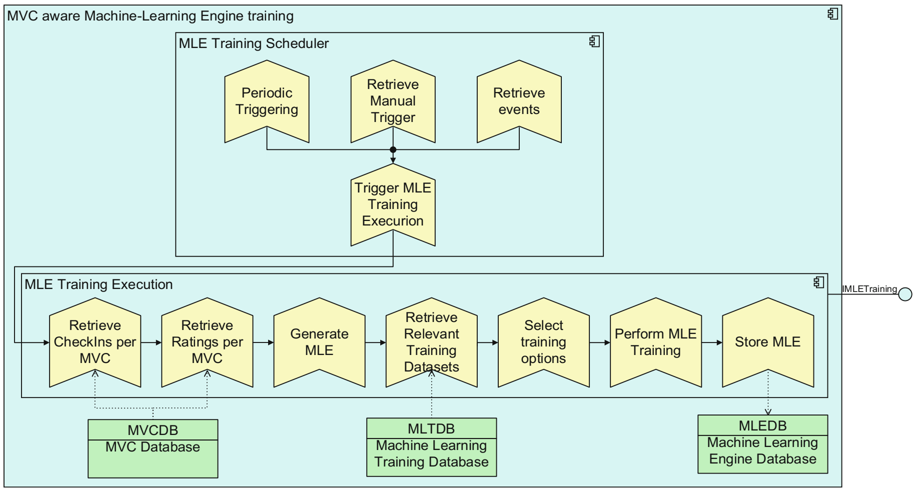

Fig. 2. MVC aware Machine-Learning Engine training – main application components, functions, interfaces and data objects.

4 MVC Suggestions

The *MVC Suggestions* service is responsible for performing estimation of the MVCs that a user may be interested in, according to his preferences.

Figure 3 states the main application functions and components together with relevant interfaces, main data objects and information flows for the *MVC Suggestions* service. The applications that can be identified from the diagram are the following:

- *Retrieve Popularity Options:* this function retrieves the popularity options that the user has defined from the *Popularity options* service.
- *Retrieve Related MLE:* the MLE that has been trained from the *MVC aware Machine-Learning Engine training* service is retrieved from MLEDB (Machine Learning Engine Database).

- *Use alternative means for MVC suggestions:* in case there is any problem retrieving data required for successfully estimating the suggestions list, such as no relevant MLE having been created yet, this function uses a fallback mechanism to provide estimations with alternative means.
- *Estimate MVCs for Suggestion:* This function finally provides the estimated list with suggested MVCs.

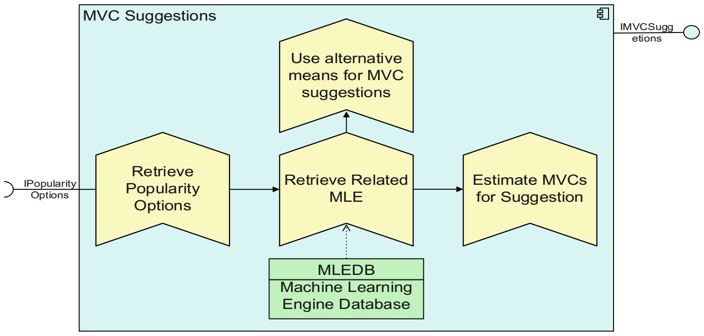
MVC Suggestions
IMVCSuggetions
Use alternative means for MVC suggestions
IPopularity Options
Retrieve Popularity Options
Retrieve Related MLE
Estimate MVCs for Suggestion
MLEDB
Machine Learning Engine Database

Fig. 3. MVC Suggestions – main application components, functions, interfaces and data objects.

5 Device Triggering

The *Device triggering* application component, which runs on the mobile device, is responsible for triggering the *MVC aware Machine-Learning Engine training* service after specific events that take place on user's mobile device (Fig. 4).

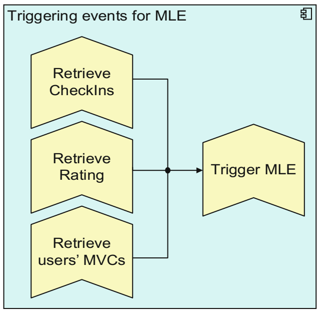

Fig. 4. Triggering events for MLE – main application components and functions.

For the *Triggering events for MLE* component, the application functions that have been identified are the following:

- *Retrieve CheckIns:* this application function retrieves user's checkIns that were made on a specific location.

- *Retrieve Rating:* this application function retrieves the ratings that the user has made for a specific location.
- *Retrieve users' MVCs:* this application function retrieves the MVCs that the user selects to include in his popularity options.
- *Trigger MLE:* this application function triggers the MLE training mechanism.

The main purpose of this component is to initiate the generation of new MLEs and its application functions are not mapped to any business layer processes.

6 Conclusions

In this paper, we have discussed on a novel approach for robust and accurate estimation of Mobile Virtual Communities that match users' preferences through the deployment of machine learning techniques which render the mobile device capable of learning over time to predict and provide suggestions to users. The application and business layers of the identified architecture, including foreseen components and their interactions, were presented in detail. Further research activities include the implementation of the discussed approach, proving the concept's wide degree of feasibility.

Acknowledgments. This work has been performed under the Greek National project WikiZen (11ΣYN_10_1808), which has received research funding from the Operational Programme "Competitiveness & Entrepreneurship" of the National Strategic Reference Framework NSRF 2007–2013. This paper reflects only the authors' views, and the Operational Programme is not liable for any use that may be made of the information contained therein.

References

1. Facebook: https://www.facebook.com
2. Twitter: https://twitter.com
3. Modani, N., Dey, K., Mukherjea, S., Nanavati, A.A.: Discovery and analysis of tightly knit communities in telecom social networks. IBM J. Res. Dev. **56**(3), 618–630 (2010)
4. Moosavi, S.A., Jalali, M.: Community detection in online social networks using actions of users. In: 2014 IEEE Iranian Conference on Intelligent Systems (ICIS), pp. 1–7 (2014)
5. Tantipathananandh, C., Berger-Wolf, T.Y.: Finding communities in dynamic social networks. In: IEEE 11th International Conference on Data Mining, pp. 1236–1241 (2011)
6. Pourkazemi, M., Keyvanpour, M.: A survey on community detection methods based on the nature of social networks. In: 3rd International Conference on Computer and Knowledge Engineering, (ICCKE 2013), pp. 114–120 (2013)
7. ArchiMate® 2.1 Specification. http://pubs.opengroup.org/architecture/archimate2-doc/. Accessed May 2014