Pattern Recognition Letters 105 (2018) 121–134

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

EventAware: A mobile recommender system for events

Daniel Horowitz [a], David Contreras [b,a,*], Maria Salamó [a]

[a] Dept. Mathematics and Informatics, University of Barcelona, Gran Via de les Corts Catalanes, 585,-08007, Barcelona, Spain
[b] Facultad de Ingeniería y Arquitectura, Universidad Arturo Prat, Avenida Arturo Prat, 2120, Iquique, Chile

ARTICLE INFO

Article history:
Available online 8 July 2017

Keywords:
Recommender systems
Natural language processing
Mobile technologies

ABSTRACT

Developing a recommender system for events raises several issues that are different from other domains. Events rapidly disappear, users' preferences quickly change over time, and direct feedback does not exist for events that have not taken place. As the recommendations will not be further available, user's context become a key factor for providing accurate recommendations. In this paper we introduce EventAware, a context-aware mobile recommender system to personalize the agenda of users attending to a congress. In particular, we first introduce the EventAware system, which includes an intuitive user interface with an attractive design to enhance user experience. EventAware incorporates some implicit contextual information, automatically initializes both the user's profiles with minimal user interaction and the properties of the items and it uses a context-aware tag-based recommender algorithm. We demonstrate its usability through a live-user case-study in one of the biggest events of mobile technology in the world, held in Barcelona.

1. Introduction

Recommender Systems (RSs) are software tools and techniques that allow users to identify the items that are likely to be more attractive and useful [39]. The event recommendation problem is intrinsically cold-start as Events are so called *one-and-only* items [12], which makes them harder to recommend. Events, by definition, are always in the future and are short-lived as they take place at a specific moment in time and place to become irrelevant very quickly afterwards. For this reason, the development of a recommender system [2] for events raises several issues that are different from other domains. In addition to the ephemeral nature of events, the users' preferences quickly change over time, and direct feedback does not exist for events that have not taken place.

Traditional collaborative filtering [22] techniques cannot cope at all with time-specific items, like events, which typically receive their ratings once they have finished. As a result, in the literature, most of the proposals for event recommendation are based on hybrid algorithms [4], which combine content-based (CB) [26] and collaborative filtering (CF) [21] techniques. A special kind of recommender systems are Context-Aware Recommender Systems (CARSs) [1], which take into consideration information about the *context*[1] in which recommendations occurs in addition to the information that traditional recommender systems deal with: information about the items to be recommended and information about the users asking for recommendations. Although recent empirical evaluations indicate that no context-aware approach is universally dominant in the RSs literature [33], with the advent of mobile devices and ubiquitous computing RSs have begun to incorporate Location Based Services[2] into mobile applications to provide users with interesting items according to their contextual information [1].

In this paper we describe in depth *EventAware*, a context-aware tag-based mobile recommender system for events that personalizes the agenda of users attending to a congress. EventAware has been specifically crafted to assist attending users to a congress by providing them with smart and personalized sessions and exhibitors during the congress. EventAware includes an intuitive user interface with an attractive design to enhance user experience. Instead of generating recommendations based on past events rated by the users in the past as in collaborative filtering, which are in many cases difficult to obtain because the event may be completely different each year, we propose to integrate a context-aware and a tag-based approach for generating recommendations. Addi-

* Corresponding author at : Facultad de Ingeniería y Arquitectura, Universidad Arturo Prat, Avenida Arturo Prat, 2120, Iquique, Chile.
E-mail address: dcontreras@maia.ub.es (D. Contreras).

[1] *Context* is defined as any information that can be used to characterize the situation of an entity [9]. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application.

[2] A Location Based Service (LBS) is a software-level service that uses location data to control features.

http://dx.doi.org/10.1016/j.patrec.2017.07.003
0167-8655/

tionally, EventAware incorporates some implicit contextual information, automatically initializes both the user's profiles with minimal user interaction and the properties (i.e., tags) that describe the items. We demonstrate its usability through a live-user case-study in one of the biggest events of mobile technology in the world, held in Barcelona.

The rest of this article is organized as follows: the next section reviews related work on event recommender systems; Section 3 describes the proposed event recommender; Next, Section 4 discusses the usability of the proposal; the final section gives conclusions and future work.

2. Related work

Event recommendation is an area of research relatively new. Researchers have focused event recommendation on two main research areas: (1) Recommenders Systems [39] and (2) Event-based Social Networks [25].

Firstly, a large amount of the research on event recommendation investigated the core algorithms used for *recommendation generation* [13]. Many of the proposals for generating event recommendations are based on hybrid algorithms that combine content-based and collaborative filtering techniques. One of the first proposals towards event recommendation, that has not been evaluated with experiments, uses a hybrid content and collaborative filtering approach within a fuzzy relational framework [7]. In particular, the underlying goal of this fuzzy relational approach is to recommend future events if they are similar to past events that similar users have liked.

Alternatively, it has also been proposed a cultural event recommender [23], based on collaborative filtering, that provides a way for users to rate the trustworthiness of other users. Then, according to these ratings, a recommendation is generated. A similar principle follows a recommender system for academic events [20], which focuses on social network analysis in combination with collaborative filtering. It is worth noting an approach [10] that compares event recommendation algorithms (i.e., CB, CF, CB+CF, among other combinations) with the aim of analyzing the user satisfaction in the end, by performing an online user-centric based evaluation experiment.

More recently, it has been also proposed a collaborative ranking of future events [31]. In this case, the user study conducted concerned to the specific application of event recommendation to scientific talks. The event recommendations were presented on a web-based environment, from which they collected the users' preferences. Similarly to us, they identify topics from the seminar announcements but they do not model implicitly the users' areas of interest (i.e., they elicited explicitly feedback from users) and do not use contextual information for recommendation generation.

Secondly, with the advent of Event-Based Social Networks (EBSNs), event recommendation has recently garnered increased attention. EBSNs are online social networks where users can create, promote and share upcoming events of any kind with other users. For instance, Meetup.com, one of the largest EBSNs available today.

Several studies on EBSNs have utilized content information for event recommendation. Qiao et al. [38] proposed a bayesian matrix factorization approach and employ social regularization factors inspired by user interactions in an EBSN. Khrouf and Troncy [18] proposed a hybrid event recommender for recommending music-related events that combines linked open data, social information and content features. Another approach [40] focuses on a collective Bayesian Poisson factorization to jointly model user response to events, social relation, and content text.

In addition to the content modeling, some recent work found that multiple data modalities and contextual information are very useful in event recommendation. For example, Lu et al. [27] presented a system that extracts events from multiple data modalities and recommends events related to the user's ongoing search based on previously selected attribute values and dimensions of events being viewed. In [15] on Meetup investigated how social network, user profiles and geo-locations affect user participation when the social event is held by a single organizer. Another approach is a content-based recommender where cultural events metadata are enriched with open linked data available on the web [34]. In [37] it has been proposed a standard matrix factorization approach, which jointly models event, location, and social relation but they ignore content and organizer information of events. Another approach has considered spatial and temporal context to predict events [11]. A large-scale analysis of several factors that impact a user's propensity to reply positively future events in an EBSN has been conducted in [28]. Considering this study, later, it has been proposed a context-aware approach [29] that exploits various contextual signals available from EBSNs, including social signals based on group memberships, location signals based on the users' geographical preferences, and temporal signals derived from the users' time preferences. In the same direction, it has been proposed a social event recommendation method [5] that exploits a user's social interaction relations and collaborative friendships. In a large majority of EBSNs users join groups unified by a common interest, and events are organized by groups. Jhamb and Fang [14] have investigated the effect of group information on event recommendation. Outlife [35] is a mobile EBSN that offers users personalized suggestions for events, as well as suggestions for inviting a group of friends to attend the recommended event together, based on the individual preferences and the users' Facebook profiles. The event recommendation is addressed by selecting the most appropriate algorithm for each situation (with a decision tree) out of a set of recommender algorithms. If no ratings are available a content-based algorithm is used.

Although the latter approach seems similar to EventAware, it is important to remark that our proposal is not an EBSN because it was conceived to be a mobile recommender to personalize the agenda of users attending to a congress, where the users have not social interactions among them. In consequence, it is not possible to use in the event recommendation process the social interactions that exists among users. However, similarly to some of the existing proposals in EBSNs, EventAware exploits contextual information and uses multiple sources of information for defining items and users' profiles with tags. As part of our future work, we plan to include in EventAware an ephemeral social network[3] [6,17] of attendees to the congress.

As far as we know, our proposal is the first mobile tag-based recommender proposed for events. Most of the previous approaches use past events rated by the users in the past to decide which are the best recommendations and few of them consider contextual information. It is worth noting that we do not use past experience of the users in past events and the proposal lacks of a social network. We only generate recommendations based on the similarity of the tags that describe the items to the areas of interests described in the user's profile. Moreover, we take into account contextual information to improve the proposed recommendations. In addition, our approach implicitly elicits the user's preferences (i.e., areas of interests) by using their LinkedIn account.

We consider that currently the most similar approach to our work is the proposal of Kaminskas et al. [16]. Although the two approaches differ in the domain (i.e., they recommend music and we recommend events), both take into account tags and contextual information for generating recommendations. Mainly, some of

[3] An ephemeral social network is a social network temporarily created ad-hoc at a specific location for a specific purpose and lasting for a short period of time.

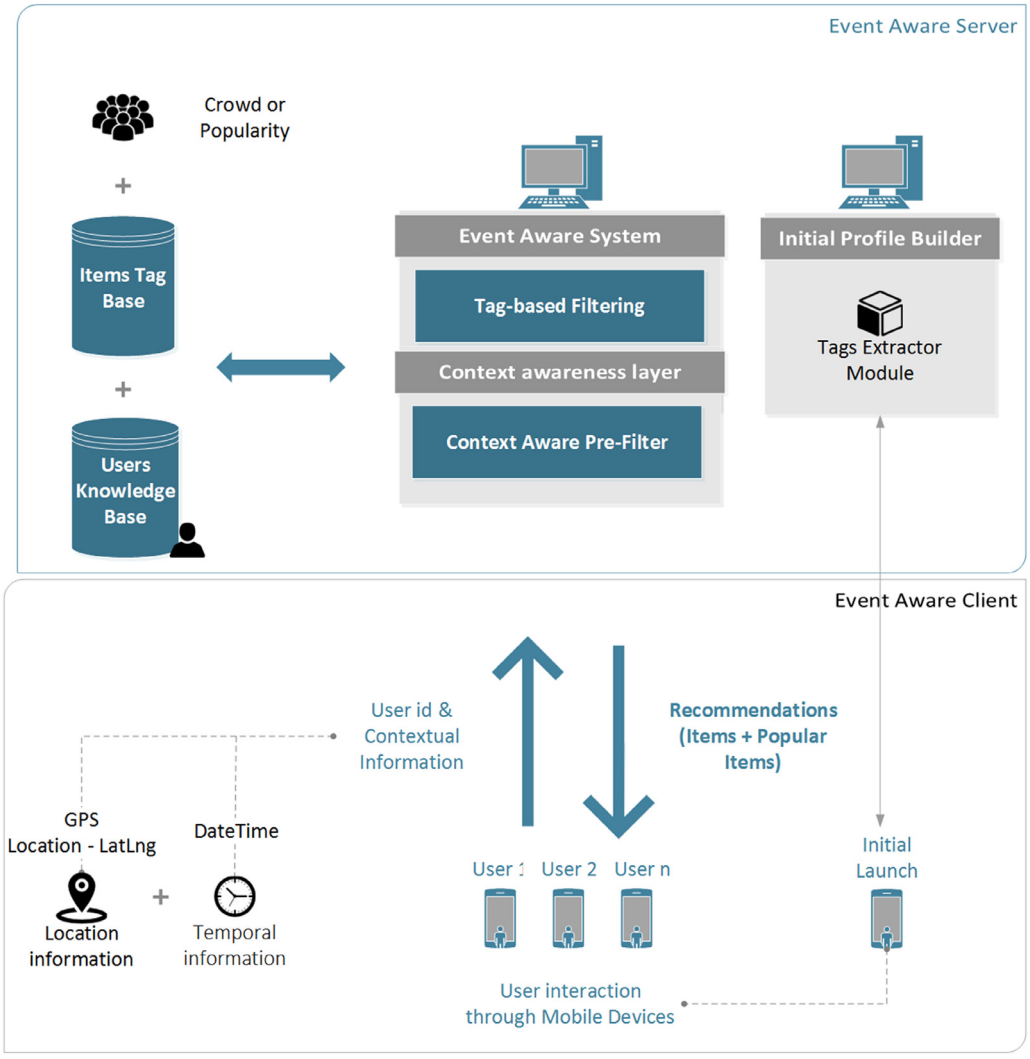


Fig. 1. Conceptual architecture of Event Aware framework.

the differences are: (1) we use Linkedin or Wikipedia for automatically extracting tags and they use human-taggers for the content tagging process; (2) they evaluated their proposal in a web-based environment and we focus on a mobile environment. As a result, the initial purpose of the recommender in both cases has been the use of tags and context for generating recommendations but the final recommenders are completely different from each other and the tested environment too.

3. Description of the EventAware recommender

This section describes the EventAware architecture and its mobile interface, how to model both users' profile and items, and the process for generating recommendations.

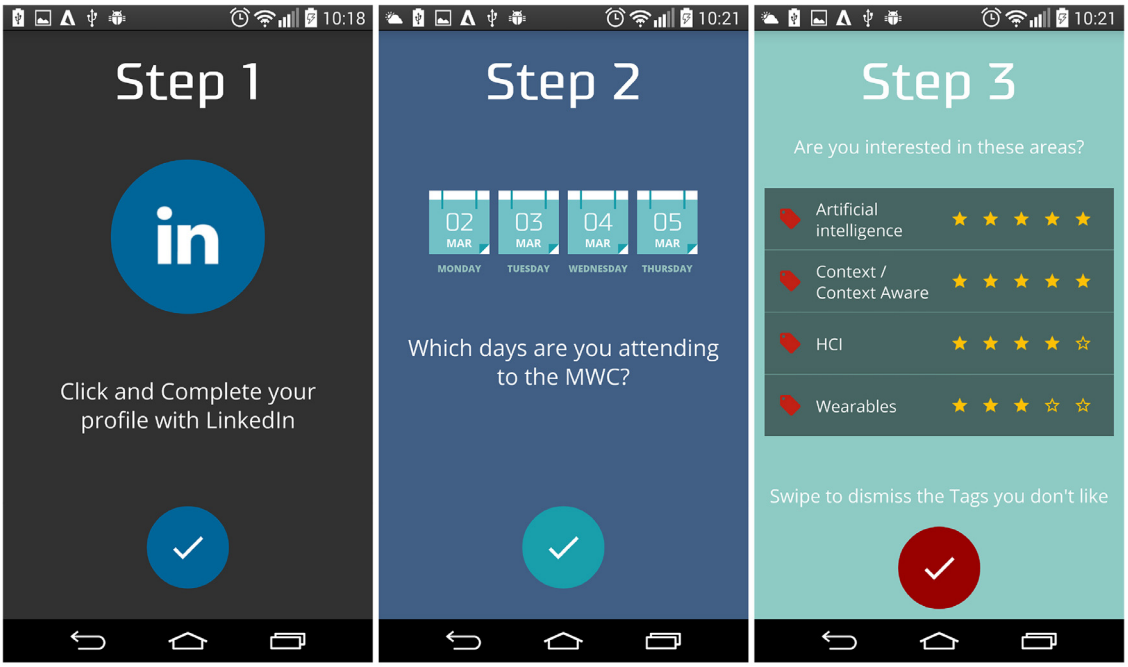
Before proceeding it is worth highlighting one important point: the proposal is general enough to be adapted to any event domain, however, in this paper, we have focused the deployed mobile application on personalizing the agenda of users attending to a congress. In particular, the items to be recommended are sessions and exhibitors. Sessions include conferences, seminars, sponsored events, and other several different programs. Exhibitors are companies who display their products and projects at the event.

3.1. EventAware conceptual architecture

Fig. 1 illustrates the client-server architecture of the proposal, which consists of two main components: the *Event Aware Server* and the *Event Aware Client*.

First, the **Event Aware Server**, which includes the items tag base (ITB), the users knowledge base (UKB), the Event Aware System for generating recommendations, and the initial profile builder. The ITB stores the items to be recommended to the user. The UKB stores personal information of the users as well as their preferences (i.e., they are also called in this paper areas of interest). The *Initial Profile Builder* is used for modeling both the items with tags and the initial users' preferences, as described **in** Section 3.3. The *Event Aware System* contains the context awareness layer to collect contextual information, the context-aware pre-filter, and the tag-based filtering algorithm, which are described in depth **in** Section 3.4.

Second, the **Event Aware Client** is responsible for gathering both contextual information and user's information, and communicating with the *Event Aware Server*. The *Event Aware Client* supports the building of the initial user profile, tracks user's location based on GPS, and requests and renders the recommendations provided by the server.



(a) (b) (c)

Fig. 2. Initial profile building process in the Event Aware recommender system.

3.2. Event Aware mobile interface

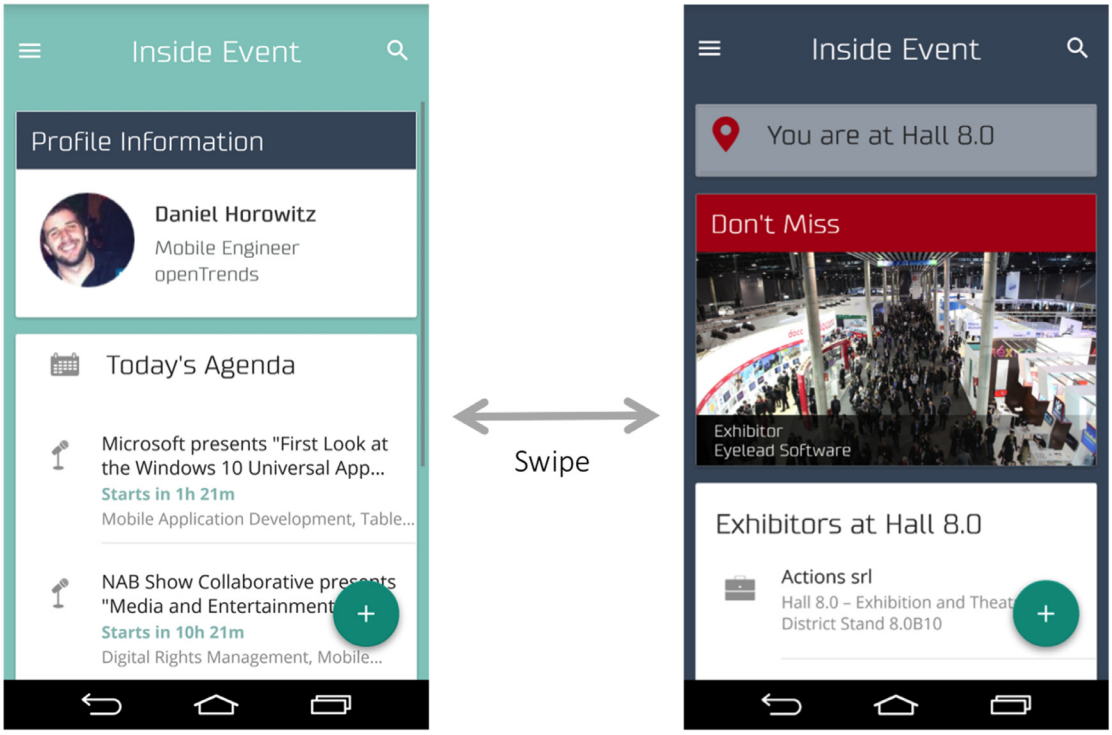
The conceptual architecture of the Event Aware described above has been deployed with an *Event Aware Client* based on a mobile environment. In this section, we describe how the user interacts with the mobile application.

Briefly, the basic scenario that a new user faces when interacting is as follows. First, the user is asked to sign in with his LinkedIn account in order to retrieve his areas of interest and to automatically model his initial profile (see Fig. 2a). A user receives the information of his LinkedIn account and confirms it. Once the sign in is complete, the user provides information about the days that he is willing to attend to the event (see Fig. 2b). Later, the system presents to the user a set of possible areas of interest, which are the user's preferences represented by tags (see Fig. 2c). Finally, the user evaluates this set by editing these areas of interest (i.e., modifying the rating proposed) or deleting them if he finds necessary. After the user confirms their areas of interests, the application retrieves the user's current context (e.g., location and time).

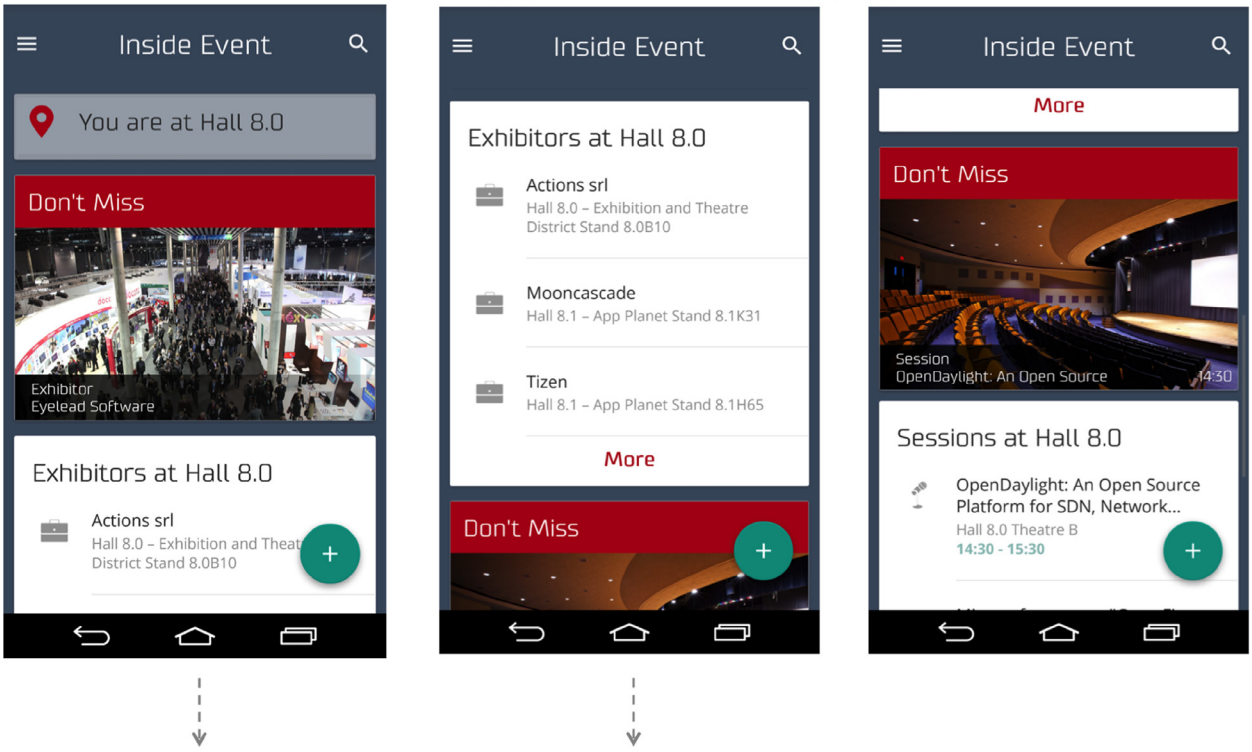
With the initial building process finished, the contextual information along with the user's profile are sent to the *Event Aware Server* in order to generate a set of recommendations. First, the context pre-filtering module (see Fig. 1) filters the recommended items (e.g., exhibitors and sessions) by discarding the items that do not match the current user context. The filtered set of items are sent to the tag-based filter, which is the recommender algorithm that uses the filtered set as input, as described in Section 3.4. The result is a set of recommended items that contains both exhibitors and sessions that matches both user's preferences and the user's current context. Furthermore, the Event Aware also selects two popular items from the items tag base, one from each category (i.e., exhibitors and sessions). The popularity is defined by items with the highest "likes" count from the initial filtered set. In addition, popular items are labeled with a flag in order to highlight them when presenting the recommendations to the user. Both items are later inserted into the final recommendations set. These recommendations are sent to the *Event Aware Client* and are presented to the user. With this first display of recommendations, the sign in process is finished.

The mobile interface displays the information in two different concept pages (see Fig. 3a) that the user can navigate through using the swipe gesture. On the left side is the Event Aware page, which includes static information about the event such as the user's basic information and the current day agenda. Note that this page displays a small set of recommended sessions that contains information of the Session's name, the time remaining to the beginning of the session, and the set of tags that matched in the recommendation process. It is important to note that the user may also visualize the full agenda as a calendar, see Fig. 4a. On the right side of Fig. 3a is the Inside Event page that presents dynamic recommendations that change according to the user's current context (i.e., location and time). The Inside Event page is also shown in Fig. 3b with the complete scroll down. This page contains a *Don't miss* section that displays the most popular items. Concretely, the *Don't miss* section presents a picture illustrating the item's category, the category name, and the name of the item.

Furthermore, the user is able to see the set of recommended exhibitors and sessions grouped by Hall, as shown in Fig. 4b. By clicking on the picture the user is able to visualize the item's details, see Fig. 5. At the top of the item's details screen, the mobile interface shows the item's score (represented as stars) in relation to the user profile. Moreover, it displays the actions that the user is able to perform. One of these actions is adding to favorites, which will be later stored in the users knowledge base enabling the computation of popular items. By scrolling down the user also sees the tags that describe the items that matched with his areas of interest. This information is illustrated on the right side of Fig. 5. The Sessions details contains almost the same information as the Exhibitors details, but it also displays the speakers that are involved in the Session and the Session's start time and end time.



(a) Event Aware page (left) and Inside Event page (right)

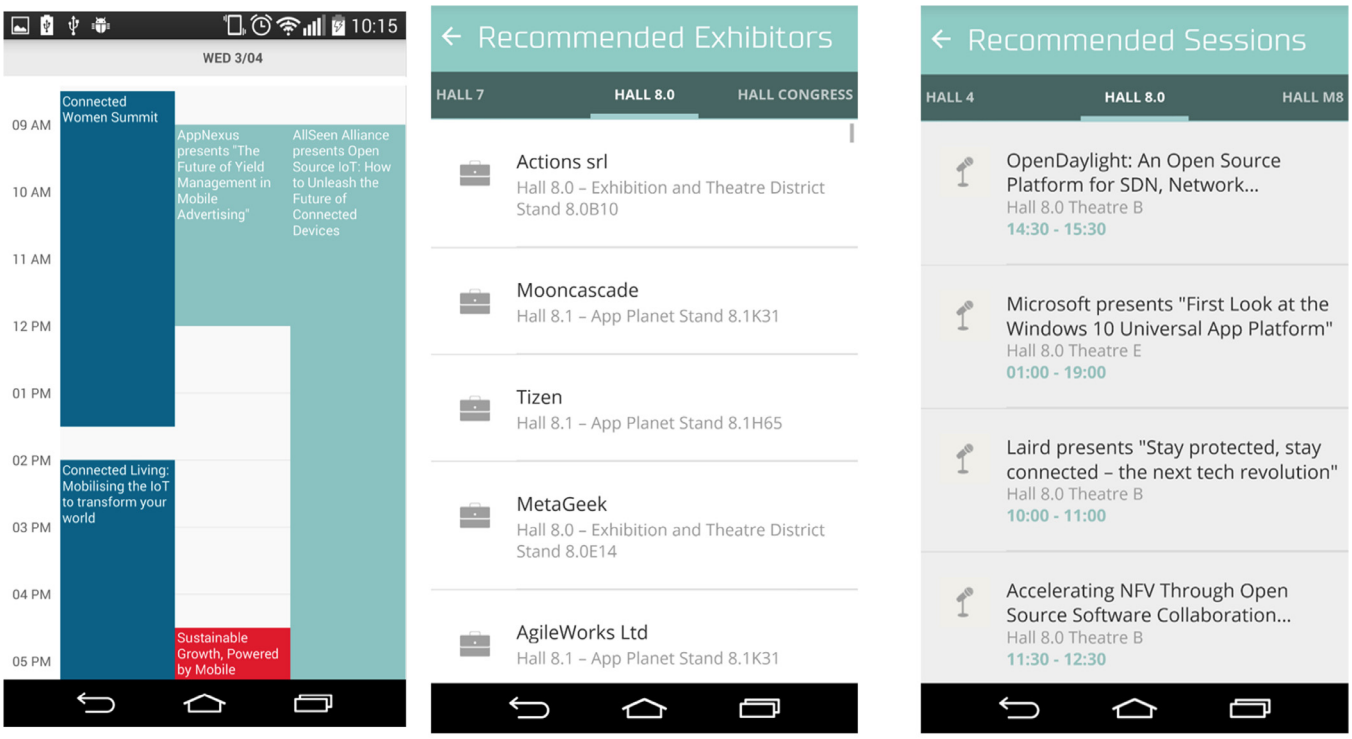


(b) Inside Event page (scrolling down)

Fig. 3. Description of the initial interaction pages at the Event Aware interface.

3.3. Modeling items and user profiles

EventAware includes a context-aware tag-based recommender that represents preferences of the users as well as the properties of the items with tags. In our proposal, the recommender does not rely on past user experiences or a complete description of the items' properties. Initially, the available information only includes: (1) a small set of general tags that describe the main topics of the event; (2) a set of items' properties described in natural language in which tags are not included; and (3) the basic information of each user described in natural language and extracted from his LinkedIn profile. Thus, it is necessary to model items and



(a) Agenda

(b) Items grouped by Hall

Fig. 4. Description of the different interaction pages at the Event Aware interface.

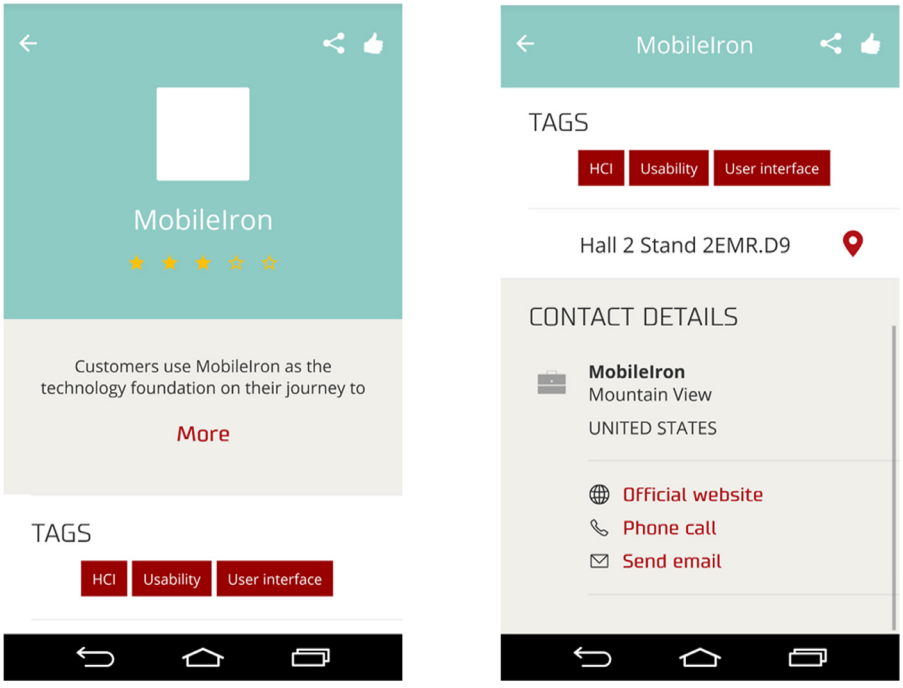


Fig. 5. Exhibitor details page.

user profiles with the same set of tags. These tags used for describing the items, the preferences of the users, and for generating recommendations are automatically inferred from several external sources (i.e., Wikipedia and LinkedIn). In order to describe in depth the automatic tagging process of both the items' descriptions and the user profiles, we need to define first the representation of the tag base, items tag base and the users knowledge base.

3.3.1. Defining the representation of the bases used in the recommendation process

Let $ITB = \{i_1, \ldots, i_n\}$ be the **items tag base**, depicted in Fig. 6, which contains the set of items for recommendation, where i_j is the jth item. Each item i_j is described as a tuple $i_j = <properties, TD>$ where *properties* includes general information about the item, such as name, description, location (i.e., hall and stand), Timetables (e.g., a session's start time), and contact information. In particular,

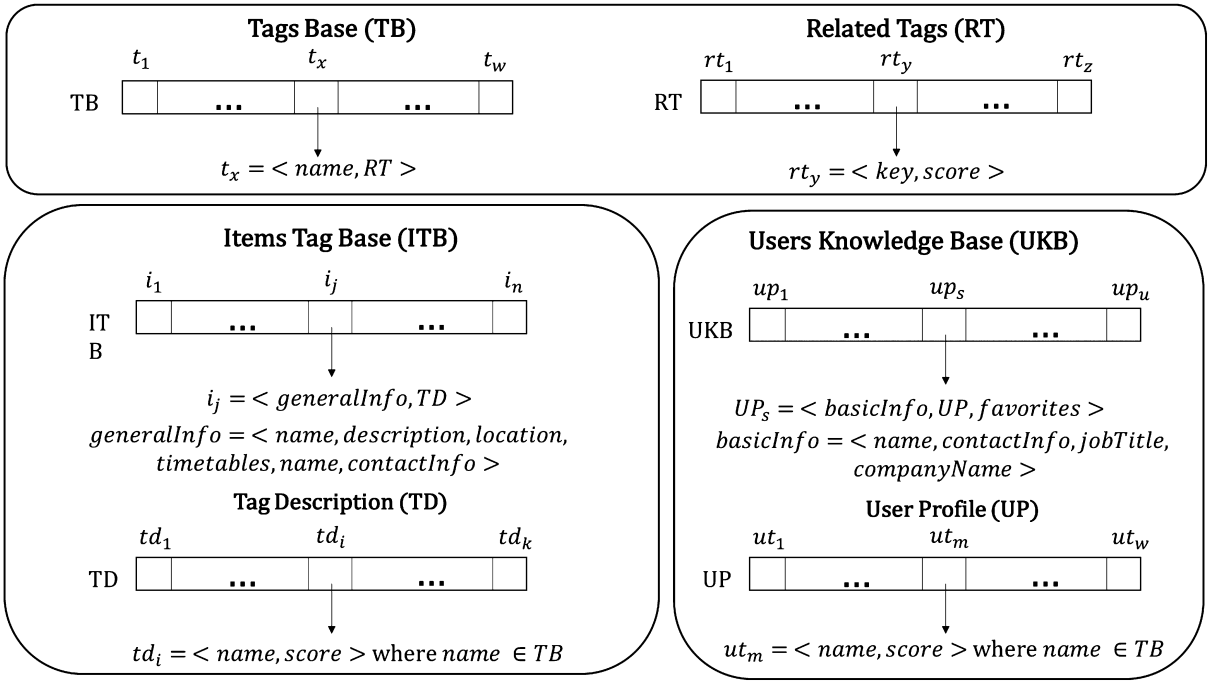


Fig. 6. Description of the tag base with its related tags, items tag base, and users knowledge base.

Event Aware recommends different items: exhibitors and sessions. Exhibitors are companies who display their products and projects at the event. Sessions are conferences, seminars, sponsored events, and other several different programs. Typically, a session involves one or more speakers that are divided into two different categories: key speakers and chairs. $TD = \{td_1, ...td_k\}$ is the tag description set of the item where each td_i is described as a tuple $td_i = < name, score >$, where $name \in TB$ is the name of the tag and the score is computed using the related tags, as described below. TD is the structure that stores the relation between tags and items. It is important to notice that the tag description set, TD, is structured and generated by the system, as shown in Section 3.3.3.

Finally, let $UKB = \{up_1, \ldots, up_u\}$ be the **users knowledge base**, depicted in Fig. 6, which stores a set of user profiles, where up_s is the sth user profile. It is described as a tuple $up_s = < basicInfo, UP >$, where *basicInfo* includes general information about the user, such as full name, contact information, and job title. Let $UP = \{ut_1, ...ut_w\}$ be the user profile, where ut_m is described as a tuple $ut_m = < name, priority\_factor >$ in which $name \in TB$ is the name of the area of interest and the priority factor is computed using the related tags. *UP* is the structure that stores the preferences of the user in terms of tags. We infer the user's interests by retrieving information from the Professional Network LinkedIn,[4] see Section 3.3.4.

The contextual information (i.e., location and time) of each user, is obtained in real-time whenever the user makes a request for recommendations. This information is not stored in the UKB due to its ephemeral nature.

3.3.2. Content-tagging process

Since our recommender departs from an initial and small set of general tags (see *TB* in Fig. 6) that describe the main topics of the event, one of the main challenges in our recommender has been to represent appropriately both the items with tags and the users' preferences with tags in order to be able to make recommendations that are of interest for the users. Albeit the set of tags was already stored in TB, the set of related tags for each tag was empty.

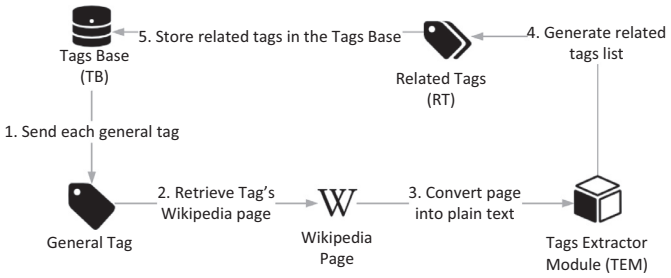


Fig. 7. Extracting related tags.

The content-tagging process consists of extracting automatically for each general tag a set of related tags (*RT*), as illustrated in Fig. 7. This process is performed in several steps:

- First step sends each general tag to identify its Wikipedia page.
- Second step automatically identifies the Wikipedia page for each general tag and converts it into plain text. For example, for the tag "Data Analysis" the Wikipedia page is Data_Analysis.[5] If a page is not found, the *RT* for this general tag will be empty.
- Third step uses the Tags Extractor Module (TEM) depicted in Fig. 1 for extracting related tags along with their scores, by using text plain as input. In this case, the input has been the Wikipedia page that has been converted into plain text by removing unnecessary information, such as HTML tags, URL's, or Wikipedia tags.
- Fourth and fifth steps use the generated related tag list from TEM to store them into *RT* and it also links the related tags in the corresponding general tag in *TB* (see Fig. 6).

Concretely, as shown in Fig. 8, TEM carries out the following process:

[4] www.linkedin.com.

[5] http://en.wikipedia.org/wiki/Data_analysis.

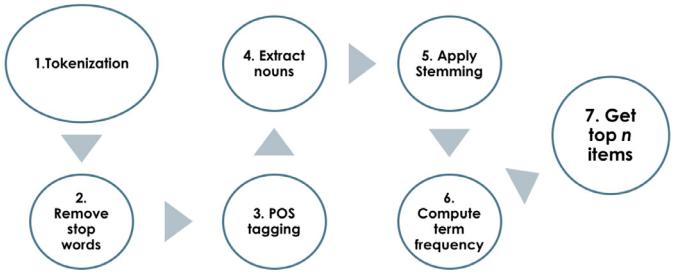


Fig. 8. Process of the Tags Extractor Module.

First, TEM tokenizes the text [30]. It is a process in which every character sequence is chopped up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation. **Second**, it removes stop words (i.e., words that are very commonly used in a given language). For example, in the context of a search engine, if the search query is "how to develop information retrieval applications", the search engine is going to find more pages that contain the term "how" and "to" than pages with information about "develop", "information", or "applications" as the term "how" and "to" are commonly used in the English language. Disregarding these two common terms, the search engine focus on retrieving pages that are of interest to the purpose of the search. In our case, the stop words removal helps to mitigate the noise inflicted by these words and keeps only the words that could be relevant to identify the text's tags.

Third, TEM applies a Part-Of-Speech Tagger in order to assign part of speech to each word such as noun, verb, or adjective. The POS Tagger implemented for this purpose used the Penn Treebank Project[6] tags. **Fourth**, we only extracted the tags that correspond to nouns (i.e., noun singular (NN), noun plural (NNS), proper noun singular (NNP), and proper noun plural (NNPS)).

Fifth, TEM focuses on stemming. For grammatical reasons, documents (e.g., Wikipedia pages) use different forms of a word, such as "organize", "organizes", and "organizing". The goal of stemming is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For example, considering the words "mobile" and "mobility", after applying stemming they are both converted to the stem "mobile". By applying stemming to words, we are able to perform the term frequency computation more effectively. Next, as **sixth** step, TEM computes the term frequency in order to identify the main tags of the preprocessed plain text. The scoring mechanism computes a score that is the sum, over the set of terms identified as possible tags, of the match scores between each term and the document. Towards this end, we assign to each term in a document a weight for that term, that depends on the number of occurrences of the term in the document. The simplest approach is to assign the weight to be equal to the number of occurrences of term t in document d. This weighting scheme is referred to as term frequency and is denoted $\mathrm{tf}_{t,d}$.

Finally, **seventh** step is devoted to select and return the top n scoring tags along with their scores. In particular, we have opted for defining empirically $n = 20$. The scores are useful in order to identify which related tag is more relevant to each tag. For example, the "Internet of things" tag contains the related tag "mobil" with a score of 20 and the "Mobile Application development" tag also contains the related tag "mobil" with a score of 100, in this case the related tag "mobil" is clearly more relevant to the "Mobile Application development" tag than it is to the "Internet of things" tag.

[6] https://www.cis.upenn.edu/treebank/.

From TEM, the content-tagging process receives the top n scoring tags along with their scores, as depicted in stage fourth and fifth in Fig. 7. These related tags and scores are then inserted into *RT* and linked to *TB*.

Furthermore, the related tags score will be later used to compute the score of the user's areas of interests and the highest scoring items will be recommended to the user as his initial profile. When the process of establishing the relation between the tags and related tags is finished, that is, *TB* is completely defined, the system is ready to assign tags to items.

3.3.3. Assigning tags to items

Initially the available information in EventAware only includes a set of item's properties described in natural language in which tags are not included. Each item stored in *ITB*, includes the field *description* (see Fig. 6). The process for assigning tags to items is devoted to complete the *ITB* with a set of tags that describes the items and which will be stored in *TD*, as shown in Fig. 6. Fig. 9 describes the process for automatically assigning tags to items.

First step uses the *description* field of the item to serve as the input text in TEM to retrieve the item's set of related tags (*IRT*). Concretely, in the **second** step, TEM performs the same process described in Fig. 8: it receives as input text the item's description, tokenizes it, removes stop words, applies a Part-of-Speach Tagger, stems the words, computes the score of each one, and generates an item's set of related tags named *irt* that contains the top n scoring tags for the provided item's description.

Third and **fourth** steps consist of matching *irt* to the tags stored in the tags base (*TB*). In the **fifth** step, the Tags Matcher generates the item's set of tags descriptors, (*TD*), that matches the item's related tags (*irt*) with the tags stored in the *TB*. Initially, we define that *TD* contains all the tags described in *TB* but with a zero value on its score. Then, the matching is performed by searching for each related tag $irt_y \in irt$ whether exists an equal related tag in one of the tags $t_x \in TB$. If the match exists, the score of $td_i \in TD$ is updated with the score of the related tag in t_x. The matching is performed by the following rule:

If t_x contains irt_y we add the score of irt_y to the total score of t_x and update it on TD

Note that *TD* contains the matched tags and their corresponding score. We repeat this matching rule for all extracted related tags in *irt*. The scores are useful in order to identify which related tag is more relevant to each tag. For example, the "Internet of things" tag contains the related tag "mobil" with a score of 20 and the "Mobile Application development" tag also contains the related tag "mobil" with a score of 100, in this case the related tag "mobil" is clearly more relevant to the "Mobile Application development" tag than it is to the "Internet of things" tag. Finally, in the **sixth** step, we select the top $n = 20$ tags from *TD* and add them to *ITB*.

3.3.4. Building user's initial profile

One of the most difficult tasks in terms of usability and user experience is to construct an initial profile to the user. The difficulty lies in retrieving the information about the user without hindering the user experience. We have opted for inferring implicitly the user's profile from an external source of information (i.e., the Professional Network LinkedIn[7]) and we refine the user's profile with the help of the user.

In order to build the user's initial profile, we have taken the LinkedIn's profile page of the user. This page stores a wide array of information regarding the user professional, social, and personal

[7] LinkedIn (www.linkedin.com) is a business-oriented social network service mainly used for professional networking and it has more than 259 million users in more than 200 countries and territories [32].

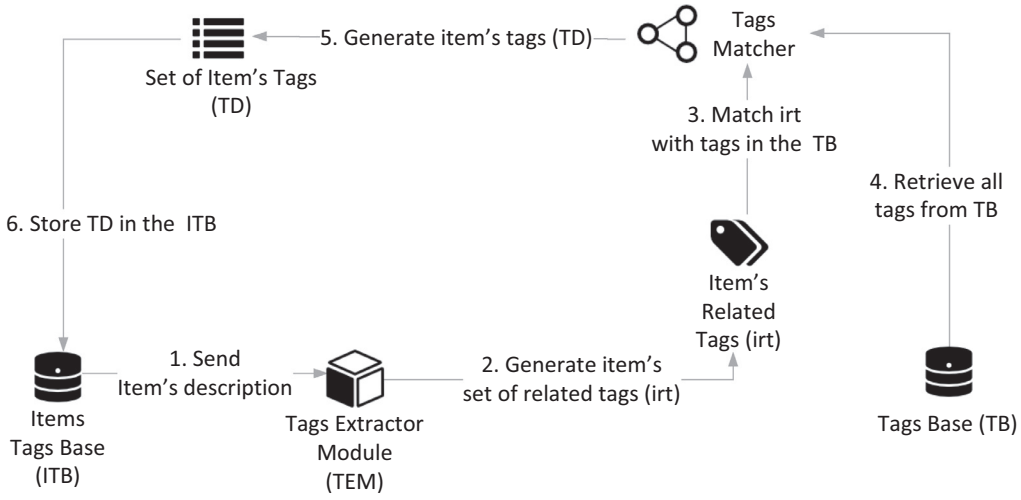


Fig. 9. Process of assigning tags to items.

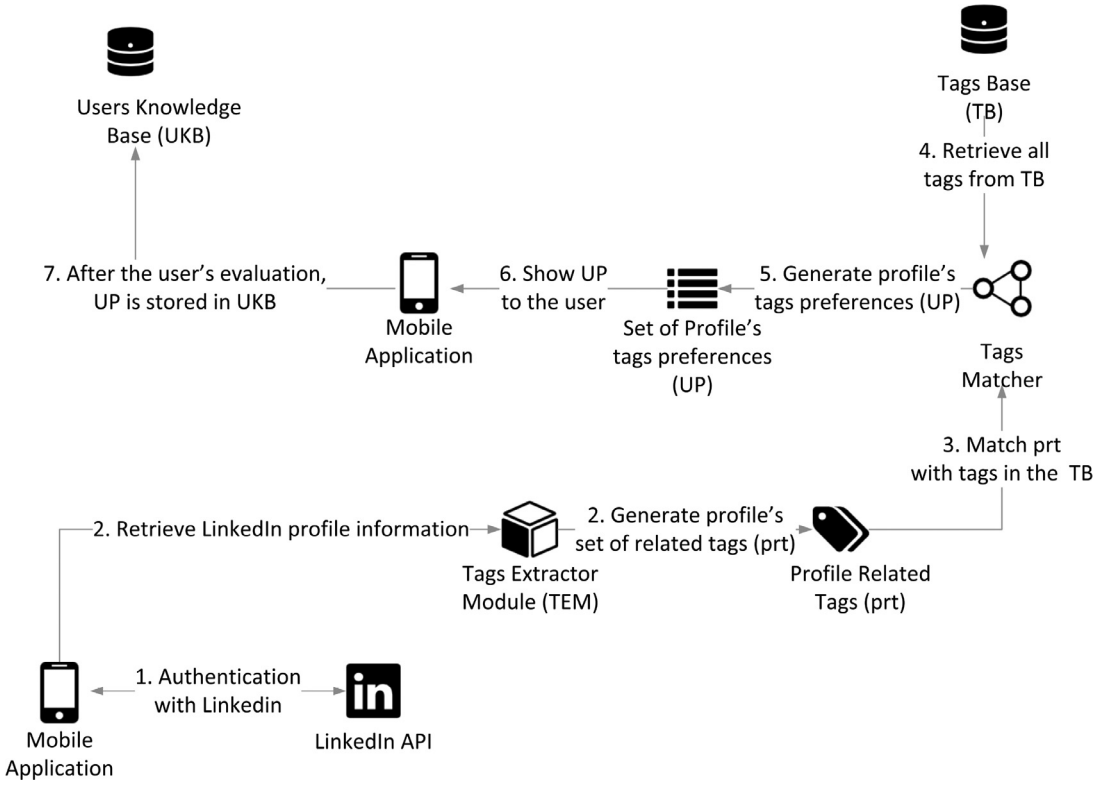


Fig. 10. Process of building the user profile.

life. Concretely, we have selected several fields[8] from this array of information and have introduced weights to each relevant field involved in the LinkedIn user profile modeling. The fields we have included in the user modeling and their corresponding relevancy -in parentheses- are the following: *Interests* (4), *Skills* (2), *Summary* (1), *Positions* (0.5), and *Groups* (0.1). For each type of information included in the model, we assigned priorities according to the field empirically observed relevancy. The relevancy takes into account the profile fields that better define the user's preferences.

The main goal of the process of building the initial user's profile is to infer the most relevant areas of interest of the user from his LinkedIn profile and add them to *UP* (see Fig. 6). Initially, we define that *UP* contains all the tags described in *TB* but with a zero value on its score. Next, we perform the process depicted in Fig. 10.

The process for building the initial user profile resemblance in some respects to the process for extracting tags from Wikipedia but are also different in others. Both processes use TEM and the Tags Matcher. They differ in the source of information used to obtain the related tags and that, in this case, the user is able to evaluate and update the result of this process before storing it to their personal user's profile.

In **first** step, the process asks for the user's authentication on LinkedIn (see step 1 and 2 in Figs. 2a and b). Since security and privacy are one of the main concerns nowadays for online services, the user is asked for his allowance that a LinkedIn application is

[8] Most of the profile fields are accessible using the LinkedIn API [24].

Algorithm 1 Context-Aware and Tag-Based Filtering Algorithm.

```
Input  : ITB: set of items, contextLoc: contextual location, contextTime: current time, UP: user profile
Output: RI: set of recommended items
1  define contextTagRecSys (ITB, contextLoc, contextTime, UP)
2  begin
3      ITB_1 ← locationPreFiltering(contextLoc, ITB) // collected with the mobile's GPS sensor
4      ITB_2 ← timePreFiltering(contextTime, ITB_1) // obtained from the time of mobile operational system
5      RI ← tabBasedRecommendItems(UP, ITB_2)// recommends items that satisfy the user's interests
6      RI ← addPopularItems(ITB_2, ITB_1, ITB, RI) // recommends popular items
7      return RI
8  end
9  define tabBasedRecommendItems(UP, ITB_2)
10 begin
11     score ← 0
12     RI ← ∅
13     foreach I in ITB_2 do
14         TM ← tagsThatMatch(I_TD, UP)
15         if TM! = ∅ then
16             ri ← ∅
17             score ← computeScore(UP,TM)
18             ri.i = I
19             ri.u_s = score
20             RI ← RI + ri
21         end
22     end
23     if RI! = ∅ then
24         RI ← sortByScore(RI)
25     end
26     return RI
27 end
28 define addPopularItems(ITB_2, ITB_1, ITB, RI)
29 begin
30     popularItems ← ∅
31     if ITB_2! = ∅ then
32         popularItems ← mostPopular(ITB_2, RI)
33     else if ITB_1 ≠ ∅ then
34         popularItems ← mostPopular(ITB_1, RI)
35     else
36         popularItems ← mostPopular(ITB, RI)
37     end
38     RI ← RI + popularItems
39     return RI
40 end
```

asking for his profile information. Next, as a **second** step, we transform the LinkedIn profile information into plain text and pass it to TEM, which generates the set of related tags, with its corresponding score, called *prt*. **Third and fourth** steps are devoted to match *prt* to the related tags stored in *TB*. The Tags Matcher performs the same process described in Section 3.3.3. It generates in **fifth** step a set of profile's tags preferences (i.e., areas of interest of the user). Each area of interest contains a priority factor that is the normalized score to the range 1 to 5. The normalization helps to represent later this information as stars at the user's interface. Finally, as **sixth** step, *UP* is presented to user (see step 3 in Fig. 2c), who evaluates the areas of interest. Note at the interface that we represent, as mentioned above, the score obtained in the matching process by stars (from 1 to 5). Afterwards, to finish the process in the **seventh** step, the evaluated set is added to *UKB* as the current user profile, *UP*.

It is important to note that the set of areas of interest generated is evaluated by the user before storing it on *UKB* and, in this way, the user is able to delete them using the swipe gesture or to change the scoring to his real preferences. Note that the tags from *TB* have been used for defining both item's properties and the user's profile.

The contextual information of each user, is obtained in real-time whenever the user makes a request for recommendations and it is not stored in the UKB due to it's ephemeral nature.

3.4. Context-aware tag-based recommender

Until now, we have been describing how the EventAware system automatically retrieves and updates with the same set of tags both the items and the users' profiles, which are the main sources of information for the context-aware tag-based recommender. With all the information, context, items, and users' profiles, the recommender algorithm generates the list of items that is recommended to the user.

Algorithm 1 depicts the process for generating recommendations. The goal of the recommendation algorithm is to match the tags that represent the areas of interest of the user, defined in *UP*, with the tags that describe the items *TD*. For each item that contains one or more of the areas of interest of the user, we compute it's score taking into account the number of tags that matched and the priority factor of the area of interest. which was defined in the number of stars assigned to it (see Section 3.3.4).

In Algorithm 1, the **first two** steps (lines 3 and 4) are devoted to the contextual pre-filtering, by taking the full set of items *ITB* and retrieving only those that match the user's current context: location and time. The **first** one only retrieves the items that matches the user's current location and inserted them into ITB_1. The location is obtained using the mobile's GPS sensor. Then, we also apply an algorithm using geofencing in order to determine in which hall is located the attendee. If he is outside the Geofence perimeter of the event venue, we use the GPS information. The **second** step takes ITB_1 and filters out the items that match the user's current time information obtained using the time of the mobile operational system and the result is ITB_2. In summary, the pre-filtering steps reduce the set of possible recommendations according to the user's current context.

The **third** step, see lines 5 and 9–27 in Algorithm 1, consists of generating the item recommendations, *IR*, based on the compatibility of the areas of interests of the user (*UP*) and the score of each item in ITB_2. Formally, let $RI = \{ri_1,..., ri_z\}$ be the set of recommended items to the user. Each recommended item x is represented by the tuple $ri_x = <i, u_s>$ where i is the item, (an Exhibitor or a Session), $i \in ITB$, and u_s is a computed score that reflects the compatibility of an item i for a user profile, *UP*. For each item *I*, in the ITB_2, first of all we verify in line 14 if its tag descriptor set, I_{TD}, contains one or more tags from the user profile, *UP*, and store the matching tags in *TM*. If *TM* is not an empty set, we compute the score of the matching tags of this item, using Eq. (1), and we add this item as a new recommendation item to *RI* (see lines 18–20).

$$computeScore(UP, TM) = \frac{\sum_{i=1}^{|TM|} Relevancy(TM_i, UP)}{|UP|}, \quad (1)$$

where *TM* is the set of matching tags and $Relevancy(TM_i, UP)$ retrieves the relevancy of the matched tag, TM_i, at the user profile, *UP*. That is, the priority factor of the area of interest in *UP* that coincides with the matched tag. After computing the score of *I*, we add the item and its score to *RI* (see lines 18–20). Once added all items that matched with the user preferences, the *RI* is sorted in decreasing order (see lines 23–25).

Table 1
Post test questionnaire.

ID Question	Statement
Q1	The items recommended to me matched my interests
Q2	The items recommended to me took contextual factors into consideration
Q3	I found easy to tell the system about my preferences
Q4	Finding an item with the help of the recommender is easy
Q5	I feel supported to find what i like with the help of the recommender
Q6	I understood why the items were recommended to me
Q7	Overall, I am satisfied with the recommender
Q8	If a recommender such as this exists, I will use it to find items
Q9	I will use this recommender again
Q10	I will tell my friends about this recommender

Finally, as **fourth** step, we add the two most popular items, which are not already included, to the set of recommended items *RI* (see lines 6 and 28–40). As our recommendation process applies several filters and, in consequence, we have several item knowledge bases (ITB_2, ITB_1, and ITB), we select the first non-empty item knowledge base that best satisfies the user context and the user's preferences. From the selected item knowledge base we extract the two most popular items that have not been already included in *RI*, see lines 32, 34, and 36. It is worth mentioning that the popular items are displayed separately from the other items in *RI*, for this reason, they are not included in the sorting process. Once the *RI* is complete, we send it to the mobile application (see Fig. 1). Note that, in this way, we present to the user not only items that matched the user's preferences but also items which are "out of the box" from the user's perspective. The client application is responsible for displaying a subset of *RI*, with the maximum length of 20 items per hall ordered by their score in descending.

4. Evaluating the usability of the mobile application

We evaluate the usability of EventAware with real users. Given the high fidelity of our application, we used the summative usability evaluation method [3].

4.1. Setup and methodology

In our experiment, we recruited 15 participants, diverse in features such as age, gender, computer skills, and experience in mobile environments. The test was performed during a mobile technology congress that took place at a conventions pavilion in the city of Barcelona. The test was conducted by a moderator and an observer. The moderator welcomed the users and briefly explained test objectives. Users were requested to use the mobile application to locate the events they are willing to attend. Each participant owned an Android device with Internet and Location services enabled. The test protocol consisted of four stages:

1. **Pretest interview**: In this stage the moderator welcomed the user, briefly explained test objectives and asked about their previous experience with mobile applications and recommender systems.
2. **Training**: During this stage users were asked to create their initial profile and then they were freely navigating on the mobile application. They were asked to locate a predefined event.
3. **Test**: In this stage users performed, without guidance, a test task that consists of using freely the mobile application. They roam freely through the event venue in order to receive context aware recommendations for both exhibitors and sessions the days they are attending to the congress. During the task, we recorded the test session.
4. **Posttest questionnaire**: After the test, users were asked to fill out a web form that contains a satisfaction questionnaire consisting of 10 questions (see Table 1). The questionnaire is based on well known usability evaluation models [36], including TAM [8] and SUMI [19]. The questionnaire was answered using a five-point Likert scale where 1 corresponds to *strongly disagree* and 5 to *strongly agree*.

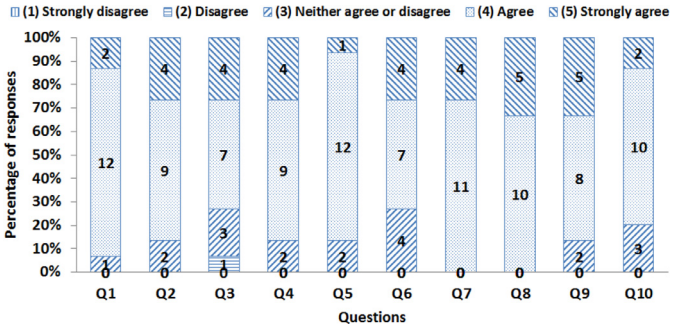


Fig. 11. Usability analysis of the context-aware tag-based mobile application interface.

After the test, we analyzed the posttest questionnaire to extract relevant data concerning usability. The next section describes this evaluation analysis.

4.2. Analysis of usability

Figs. 11 and 12 depict the results obtained from the posttest questionnaire (shown in Table 1). Note that these results are related to the subjective perception of users but are quantitative data, which gives us valuable information about users' perceptions in terms of usefulness and usability of EventAware mobile application. Overall, the quantitative results obtained are very satisfactory (see Fig. 11), as 99.3% of the responses were ranked with 3 or more points, 0.7% relied to a question with value 2, and 0% of responses correspond to a minimal score (1 value).

Considering the context compatibility (i.e., question Q2), Fig. 11 shows that 86.7% of participants' responses show that the users perception is that the system takes into consideration contextual factors and evaluated this aspect with 4 or more points and none of participants replied to this question with a minimal score (i.e., 2 or 1 values). The details of question Q2 are in Fig. 12b, which shows that 13.3% of the participants ranked with 3 (*Neither agree or disagree*) points, 60% ranked with 4 (*Agree*) points, and 26.7% ranked with 5 (*Strongly agree*) points. Moreover, the obtained result for Q3 is satisfying, too, as 14 participants ranked over 3 points (i.e., 93.3% of participants) our initial profile builder, 1 participant respond with value 2, and none of them replied to this question with value 1 (see Fig. 11). This result is positive considering that we build automatically the initial user profile with the information stored at LinkedIn. In fact, as shown in Fig. 12c, 6.7% of participants ranked it with 2 (*Disagree*) points, 20.0% ranked with 3 (*Neither*

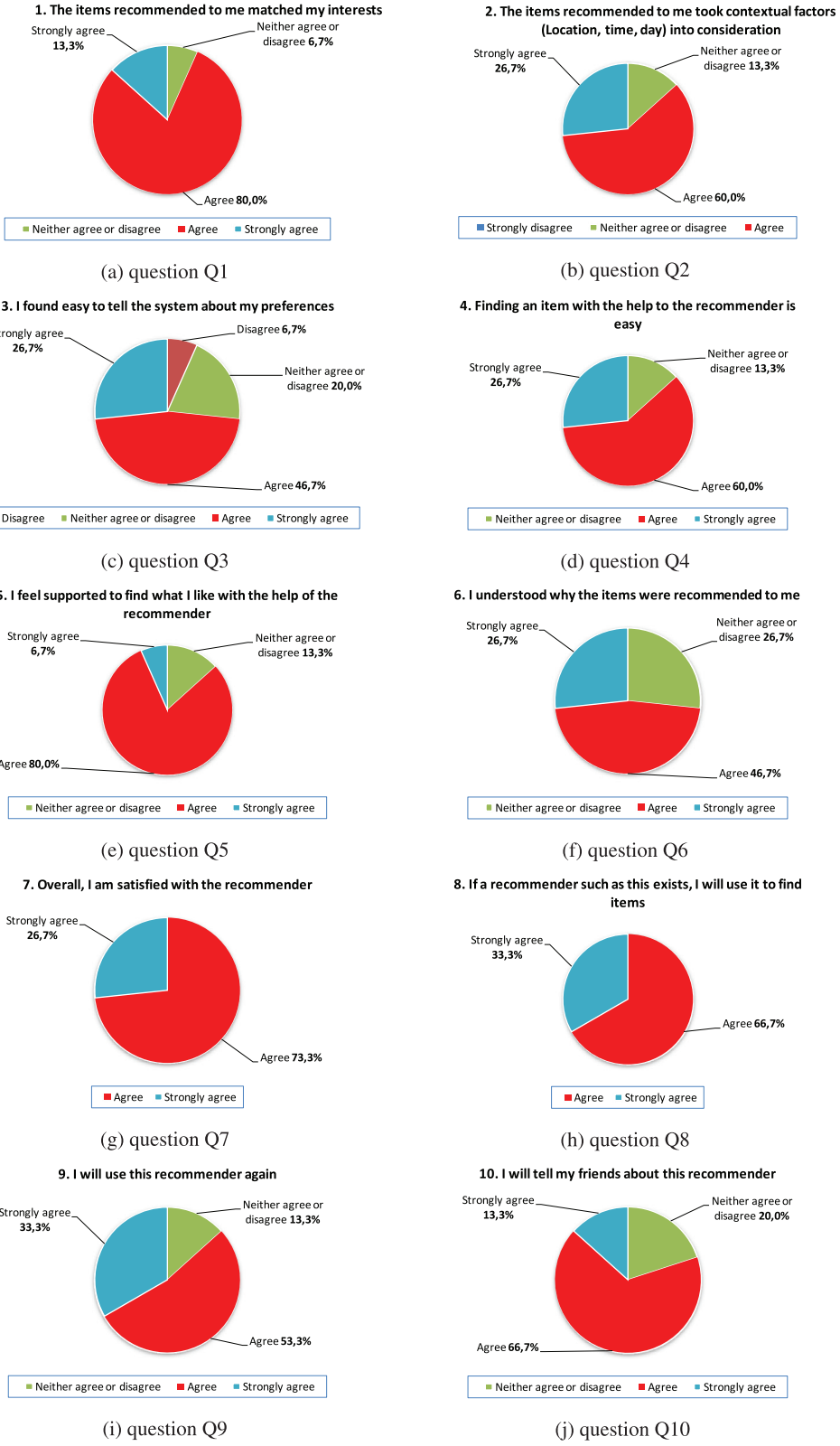


Fig. 12. Detailed responses to the usability questionnaire of EventAwar.

agree or disagree) points, 46.7% ranked it with 4 (*Agree*) points, and 26.7% ranked with 5 (*Strongly agree*) points.

With regard to the easy of decision making in question Q4, Fig. 11 shows that 86.7% of the participants positively evaluated this aspect with 4 or more points. Moreover, Fig. 11 shows that the remaining percentage corresponds to 3 points (i.e., 13.3% of participants in Fig. 12d), and no one respond with 2 or 1 points. That is, 26.7% ranked Q4 with 5 (*Strongly agree*) points, 60% of participants ranked it with 4 (*Agree*) points, and 13.3% ranked it with 3 points (see Fig. 12d).

In this way, the result in the perceived usefulness (Q5) shows that 100% of participants perceive that the recommender supports

them to find what they like (i.e., all participants evaluate this question with over 3 points, see Fig. 11). In particular, as shown in Fig. 12e, 13.3% of participants ranked Q5 with 3 (*Neither agree or disagree*) points, 80% ranked it with 4 (*Agree*) points, and 6.7% ranked it with 5 (*Strongly agree*) points.

In addition, another very important aspect of the recommendation process, is to build a certain amount of trust between the user and the system. In this way, a 73.3% of participants replied to the question Q6 with over 4 points, see Fig. 11. Thus denoting that they mostly understand the items that have been recommended to them. Specifically, 26.7% of participants ranked Q6 with 3 (*Neither agree or disagree*) points, 46.7% ranked it with 4 (*Agree*) points, and 26.7% ranked it with 5 (*Strongly agree*) points (see Fig. 12f).

Moreover, the results in Q7 indicate that the majority of users were satisfied with the system (i.e., 100% ranked Q7 with 3 or more points, see Fig. 11. Specifically, Fig. 12g shows that 73.3% of participants ranked Q7 with (*Agree*) points and 26.7% ranked it with 5 (*Strongly agree*) points.

We were also very interested in users' opinions regarding whether they will use in the future a recommender system as the one tested; responses to Q8 depict that 100% of the participants evaluated it with 4 and 5 points (see Fig. 11). Concretely, Fig. 12h shows that 66.7% of participants ranked Q8 with 4 (*Agree*) points and 33.3% ranked it with 5 (*Strongly agree*) points.

It is also worth noting that in the case of Q9 all of participants ranked this question with 3 or more points, which means that users have a good perception about the usefulness of the mobile application (see Fig. 11). Moreover, Fig. 12i shows that 13.3% of participants ranked Q9 with 3 (*Neither agree or disagree*) points, 53.3% ranked it with 4 (*Agree*) points, and 33.3% ranked it with 5 (*Strongly agree*) points.

Additionally, when users answer about their intention to tell their friends about this recommender (question Q10) only 3 participants ranked Q10 with 3 points, the remaining ones replied to this question with 4 or more points. As detailed in Fig. 12j, 20.0% of participants ranked Q10 with 3 (*Neither agree or disagree*) points, 66.7% ranked it with 4 (*Agree*) points, and 13.3% ranked it with 5 (*Strongly agree*) points.

Finally, with regard to the perceived effectiveness by users during the recommendation process, results of question Q1 show that 100% of participants evaluated positively this aspect with three or more points (see Fig. 11). In fact, participants have expressed that they have approved the recommended items, 6.7% of the testers ranked Q1 with 3 (*Neither agree or disagree*) points in the Likert scale, 80% ranked it with 4 (*Agree*) points, and 13.3% ranked it with 5 (*Strongly agree*) points, as shown in Fig. 12a.

5. Conclusions

In this article we have described in depth EventAware, a mobile recommender system for events. Our proposal integrates a context-aware and a tag-based recommendation algorithm into a mobile application for recommending events. To the best of our knowledge, this is the first mobile recommender system for events. Specifically, we have described the conceptual architecture and the intuitive user interface with an attractive design that enhances user experience. Moreover, EventAware initializes automatically both the user's profiles with minimal user interaction and the properties of the items and generates recommendations using a context-aware tag-based recommender algorithm. The usability of this novel mobile application for recommending events has been evaluated by life users. Overall, the results obtained are very satisfactory, as 99.3% of the responses were ranked with 3 or more points and 0% of responses correspond to a minimal score (1 value). Concretely, our results show that 100.0% of participants evaluated positively (i.e., with 3 or more points) the effectiveness of the EventAware. Moreover, 100% of participants also had a good perception (with 3 or more points) about the usefulness of EventAware.

Acknowledgments

D. Contreras is supported by a doctoral fellowship "Becas Chile". This work has been supported by projects: SGR-623-2014 and TIN2015-71147-C2-2 from the Spanish Ministry of Science and Innovation.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.patrec.2017.07.003.

References

[1] G. Adomavicius, B. Mobasher, F. Ricci, A. Tuzhilin, Context-aware recommender systems, AI Mag. 32 (2011) 67–80.
[2] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state of the art and possible extensions, 2005.
[3] D. Bowman, J. Gabbard, D. Hix, A survey of usability evaluation in virtual environments : classification and comparison of methods, Presence: Teleoperators and Virtual Environonments. 11 (2002) 404–424.
[4] R. Burke, Hybrid recommender systems: survey and experiments, User Model. User-Adapted Interact. 12 (2002) 331–370.
[5] C.C. Chen, Y. Sun, Exploring acquaintances of social network site users for effective social event recommendations, Inf. Process. Lett. 116 (2016) 227–236.
[6] A. Chin, H. Wang, B. Xu, K. Zhang, H. Wang, L. Zhu, Connecting people in the workplace through ephemeral social networks, in: Privacy, Security, Risk and Trust (PASSAT), IEEE Third Int. Conf. on Social Computing, 2011, pp. 527–530.
[7] C. Cornelis, X. Guo, J. Lu, G. Zhang, A fuzzy relational approach to event recommendation, in: Proc. 2nd Indian Int. Conf. on Artificial Intelligence, 2005, pp. 2231–2242.
[8] F.D. Davis, Perceived usefulness, perceived ease of use, and user accep, MIS Q. 13 (1989) 319.
[9] A.K. Dey, Understanding and using context, Pers. Ubiquitous Comput. 5 (2001) 4–7.
[10] S. Dooms, T. De Pessemier, L. Martens, A user-centric evaluation of recommender algorithms for an event recommendation system, in: Workshop on User-Centric Evaluation of Recommender Systems and Their Interfaces, 2011, pp. 67–73.
[11] R. Du, Z. Yu, T. Mei, Z. Wang, Z. Wang, B. Guo, Predicting activity attendance in event-based social networks: content, context and social influence, in: The 2014 ACM Conference on Ubiquitous Computing, UbiComp '14, Seattle, WA, USA, September 13–17, 2014, 2014, pp. 425–434.
[12] X. Guo, G. Zhang, E. Chew, S. Burdon, A hybrid recommendation approach for one-and-only items, in: AI 2005: Advances in Artificial Intelligence, in: LNCS, volume 3809, Springer, 2005, pp. 457–466.
[13] D. Jannach, M. Zanker, A. Felfernig, G. Friedrich, Recommender Systems: An Introduction, 1st, Cambridge University Press, New York, NY, USA, 2010.
[14] Y. Jhamb, Y. Fang, A dual-perspective latent factor model for group-aware social event recommendation, Inf. Process. Manage. 53 (2017) 559–576.
[15] J. Jiang, C. Li, Analyzing social event participants for a single organizer, in: Proc. of the Tenth Int. Conf. on Web and Social Media, 2016, pp. 599–602.
[16] M. Kaminskas, F. Ricci, M. Schedl, Location-aware music recommendation using auto-tagging and hybrid matching, in: Proc. of the 7th ACM Conf. on Rec. Systems, ACM, 2013, pp. 17–24.
[17] A. Kermarrec, E.L. Merrer, Offline social networks: stepping away from the internet, in: Proc. of the Fifth Workshop on Social Network Systems, 2012, pp. 14–15.
[18] H. Khrouf, R. Troncy, Hybrid event recommendation using linked data and user diversity, in: 7th ACM Conf. on Recommender Systems, 2013, pp. 185–192.
[19] J. Kirakowski, M. Corbett, SUMI: the software usability measurement inventory, Br. J. Educ. Technol. (1993) 10–12.
[20] R. Klamma, P. Cuong, Y. Cao, You never walk alone: Recommending academic events based on social network analysis, in: Complex Sciences, in: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, volume 4, Springer, 2009, pp. 657–670.
[21] J. Konstan, J. Riedl, Recommender systems: from algorithms to user experience, User Model. User-adapt Interact. 22 (2012) 101–123.
[22] Y. Koren, R. Bell, Advances in collaborative filtering, in: Recommender Systems Handbook, Springer, 2011, pp. 145–186.
[23] D.H. Lee, Pittcult: Trust-based cultural event recommender, in: Proc. of the 2nd Conf. on Recommender Systems, ACM, 2008, pp. 311–314.
[24] LinkedIn Corporation, L., Linkedin API documentation, 2015, (https://developer.linkedin.com/docs).
[25] X. Liu, Q. He, Y. Tian, W. Lee, J. McPherson, J. Han, Event-based social networks: linking the online and offline social worlds, in: The 18th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2012, pp. 1032–1040.

[26] P. Lops, M. Gemmis, G. Semeraro, Content-based recommender systems: State of the art and trends, in: Recommender Systems Handbook, Springer, 2011, pp. 73–105.
[27] D. Lu, C.R. Voss, F. Tao, X. Ren, R. Guan, R. Korolov, T. Zhang, D. Wang, H. Li, T. Cassidy, H. Ji, S. Chang, J. Han, W.A. Wallace, J.A. Hendler, M. Si, L.M. Kaplan, Cross-media event extraction and recommendation, in: Proc. of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, 2016, pp. 72–76.
[28] A.Q. Macedo, L.B. Marinho, Event recommendation in event-based social networks, in: Proc. of the 1st Int. Workshop on Social Personalization, CEUR-WS.org, 2014.
[29] A.Q. Macedo, L.B. Marinho, R.L. Santos, Context-aware event recommendation in event-based social networks, in: Proc. of the 9th ACM Conf. on Recommender Systems, ACM, 2015, pp. 123–130.
[30] C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.
[31] E. Minkov, B. Charrow, J. Ledlie, S. Teller, T. Jaakkola, Collaborative future event recommendation, in: Proc. of the 19th Int. Conf. on Information and Knowledge Management, ACM, 2010, pp. 819–828.
[32] D. Nishar, 200 million members, Official LinkedIn Blog at http://blog.linkedin.com/2013/01/09/linkedin-200-million/.
[33] U. Panniello, A. Tuzhilin, M. Gorgoglione, Comparing context-aware recommender systems in terms of accuracy and diversity, User Model. User-adapt. Interact. 24 (2014) 35–65.
[34] T.D. Pessemier, S. Coppens, E. Mannens, S. Dooms, L. Martens, K. Geebelen, An event distribution platform for recommending cultural activities, in: Proc. of the 7th Int. Conf. on Web Information Systems and Technologies, 2011, pp. 231–236.
[35] T.D. Pessemier, J. Minnaert, K. Vanhecke, S. Dooms, L. Martens, Social recommendations for events, in: Proc. of the Fifth ACM Workshop on Recommender Systems and the Social Web co-located with the 7th ACM Conf. on Rec. Systems, 2013.
[36] P. Pu, L. Chen, A user - centric evaluation framework for recommender systems, in: Proc. of the 5th Conf. on Recommender Systems, 2011, pp. 157–164.
[37] Z. Qiao, P. Zhang, Y. Cao, C. Zhou, L. Guo, B. Fang, Combining heterogenous social and geographical information for event recommendation, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014a, pp. 145–151.
[38] Z. Qiao, P. Zhang, C. Zhou, Y. Cao, L. Guo, Y. Zhang, Event recommendation in event-based social networks, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014b, pp. 3130–3131.
[39] F. Ricci, L. Rokach, B. Shapira, P. Kantor (Eds.), Recommender Systems Handbook, Springer US, Boston, MA, 2011.
[40] W. Zhang, J. Wang, A collective bayesian poisson factorization model for cold-start local event recommendation, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1455–1464.