General Framework for Context-Aware Recommendation of Social Events

Wolfgang Beer, Walter Hargassner
Software Competence Center Hagenberg GmbH
Softwarepark 21, Austria
Hagenberg, Austria
{wolfgang.beer; walter.hargassner}@scch.at

Christian Derwein, Sandor Herramhof
Evntogram Labs GmbH
Leonfeldner Strasse 328
Linz, Austria
{sandor; chris}@evntogram.com

***Abstract*—Modern e-commerce systems offer a multitude of products and services in global marketplaces. The modern consumer is therefore overwhelmed by millions of options, variants and choices of products and services. With the rise of global marketplaces with their huge amount of items, recommendation systems became the basis for modern e-commerce systems. The traditional approaches for implementing recommendation engines, such as content and collaborative filtering, solve the challenge of calculating a recommendation set of items for a given user. While these traditional approaches cope well with large sets of static user and item information, they lack a general approach for including highly dynamic context-information. As the e-commerce market swiftly changes to mobile computing platforms, such as smartphones and tablets, the use of context-information for generating item recommendations is of great interest. In this work, we propose a concept for a general framework for the implementation of such context-aware recommendation engines, specifically for mobile platforms.**

Keywords-context awareness; context aware recommendation; decision support; recommendation system;

I. Introduction

The modern consumer is overwhelmed with options and choices. Global marketplaces, such as ebay, Amazon, Apple iTunes or Google Play, offer millions of different products and services in hundreds of different categories. These categories span a wide spectrum of product families from traditional hardware to software and mobile apps, ebooks, electronics, video and music streaming or even food. Today global marketplaces offer unlimited possibilities to publish and instantly deliver all kinds of products and services. While the publishing and delivery of products and services is getting easier for companies, it is difficult to uniquely present a specific product to customers. According to the huge number of products available in global marketplaces and the consumers limited time and motivation to check all similar products, recommendation engines are of crucial interest for modern consumers. Recommendation systems, such as the product recommendation at Amazon or ebay, are already present for several years. Without traditional recommendation systems, the consumer soon gets lost within the huge amount of available products. To counter that, global marketplaces soon recognized the need for transparent product recommendation within their systems. In 2006 the Netflix Prize competition was initiated with a 1 million dollar prize for achieving a ten percent or more improvement of Netflix's video recommendation algorithm. The training set that Netflix published for the price competition contained around 100 million ratings from about 500.000 anonymous customers on 17.000 videos. The contest attracted 48.000 competing teams from 182 different countries. The winning team (BellKor) from AT&T Research Labs (made up of Bob Bell and Chris Volinsky, from the Statistics Research group in AT&T Labs, and Yehuda Koren) was able to improve the performance of Netflix's recommendation algorithm by 8.43 percent. So it is obvious that traditional recommendation systems play an important role in modern consumer markets. On the other side many interesting aspects of recommendation systems have not been fully addressed by the research community so far. Bell et al. identified several such research aspects during their work on the Netflix prize competition [1]. One of these aspects, which we will also address in this work, is the relation of temporal effects and the realization that parameters describing the data are not static but dynamic functions. So the client-centric view on recommendation systems much depends on the consumer's actual context. Client-centric recommendation system approaches, such as implementations on smartphones and mobile devices, should focus on the user's demands in a tight relation to the users actual situation. So for a user the context-related dimensions time, location, weather, activity and companions play a major role in any decision. Bell et al. also identified that a combination and blending of several quite simple recommendation approaches often result in excellent recommendations. In this work, we also argue that building a framework for blending of multiple dimensions, containing context-related information such as time and distance and linked data information, can lead to reliable and customizable client-centric recommendation models.

Section II gives a short overview on state-of-the-art in recommendation systems and related work on how to introduce context-awareness in recommendations. Section III focuses on the requirements a general framework for context-aware recommendation systems has to fulfill. Section IV gives an abstract overview on our approach for introducing context-information in traditional recommendation methods and section V defines a practical software architecture to implement our approach. Section VI concludes with an application case study that introduces context-aware recommendation to the domain of social events.

II. Related Work

The importance of context-awareness in software systems has been discussed by research communities in many different application domains, including ubiquitous and pervasive computing, mobile computing, e-commerce, information retrieval, marketing and management as well as in several engineering disciplines. The term context-aware software was first used in the Xerox PARC research project PARCTAB in 1994 [2]. In this work, the term was defined and used for software that is able to adapt according to its actual location, the collection of nearby people, hosts and accessible devices. Also the possibility to track the changes of context information over time, in other words to store historic context information, was mentioned. Over the years, different research groups enriched this basic definition of context and context-aware software. Brown et al. [3], for example, widened the scope of context information to temperature, time, season and many other factors. Due to the fact that the number of context information factors is nearly unlimited, the definition of context by Anhind K. Dey is one of the most commonly used:

> Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves. [4]

This definition of context specifies that context contains any kind of information about an entity in order to understand its situation. So context information is not limited to location information, but could also mean information about the social situation of a person or the persons mood. Usually, such a sort of context information is hard to collect, but there are a reasonable number of research projects that try to collect even this kind of information. An interesting fact about the above definition of context is that Dey identifies three base classes that classify all objects: person, place and object. This kind of classification has practical reasons but is also fixed to a location-dependent view of context information. Over the last ten years several architectures and implementations of software middleware frameworks were published that emphasised the aggregation and interpretation of context-information [5].

In addition to context-acquisition, processing and interpretation, traditional algorithms for designing recommendation systems need to be considered. Traditional recommendation systems take a set U of users and a set of products (items) P, which should be recommended to a user. A recommendation system then provides an utility function f that measures the relevance of a product out of set P to a given user. This utility function f ($f : U \times P \rightarrow R$, where R is an ordered set of numbers) assigns a number to each item (or even to a compound set of items) in a way that captures the relevance or preference of an user. The objective of recommendation systems is to find or learn this utility function f. Function f is used to predict the relevance of items out of P and of new appearing items with similar attributes. In the literature different approaches exist for finding a function f by using an available dataset. Traditional recommendation approaches are distinguised into two major strategies: content filtering and collaborative filtering.

A. Content Filtering

The content filtering approach creates profiles for each item and user, in order to characterize and compare its nature [6]. Each profile contains a specific set of attributes, which can be used to compare objects. For example, a restaurant could have a cuisine attribute, describing the type of food it offers, a location attribute, a vegetarian tag, and so on. A recommendation function f chooses items that are similar to items the user has already chosen or rated before. The utility function compares the user's profile and calculates the similarity of a user profile with the available items. Therefore, the user profile allows the recommendation engine to create a list of items that could fit to a given user profile. Many implementations of this approach additionally refer to Linked Data information, such as RDF stores and Semantic Web repositories, to classify and search systematically for related information.

B. Collaborative Filtering

In collaborative filtering approaches, the recommendation function chooses items that were preferred by other users with similar attributes. So collaborative filtering approaches depend on either explicit or implicit user ratings of items. By rating different items a user can feed explicit ratings into the recommendation engine, while implicit feedback is collected by the system through the analysis of the users behaviour (previous purchases, navigation path, search terms, ...). Collaborative filtering is domain-free, which means that it can be applied to any application area and to different data aspects, which could be hard to formulate into an explicit profile. Collaborative filtering is more accurate than content filtering [6], but has the challenge of starting without any initial data sets (cold start problem). It is not directly possible to address new users or objects for which the system has no initial data set available. Popular collaborative filtering methods are neighborhood methods and latent factor models. The Pearson's correlation coefficient $sim(u, v)$ is often used to calculate the popular neighborhood method kNearest Neighbor, in order to measure the similarity between the

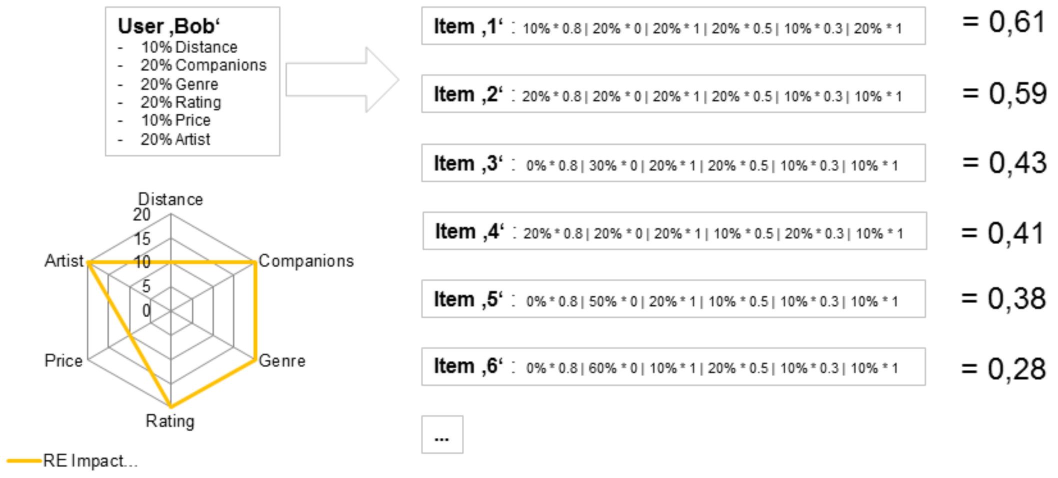
User ‚Bob‘
- 10% Distance
- 20% Companions
- 20% Genre
- 20% Rating
- 10% Price
- 20% Artist
Distance
Companions
Genre
Rating
Price
Artist
20
15
10
5
0
RE Impact...
Item ‚1‘ : 10% * 0.8 | 20% * 0 | 20% * 1 | 20% * 0.5 | 10% * 0.3 | 20% * 1
= 0,61
Item ‚2‘ : 20% * 0.8 | 20% * 0 | 20% * 1 | 20% * 0.5 | 10% * 0.3 | 10% * 1
= 0,59
Item ‚3‘ : 0% * 0.8 | 30% * 0 | 20% * 1 | 20% * 0.5 | 10% * 0.3 | 10% * 1
= 0,43
Item ‚4‘ : 20% * 0.8 | 20% * 0 | 20% * 1 | 10% * 0.5 | 20% * 0.3 | 10% * 1
= 0,41
Item ‚5‘ : 0% * 0.8 | 50% * 0 | 20% * 1 | 10% * 0.5 | 10% * 0.3 | 10% * 1
= 0,38
Item ‚6‘ : 0% * 0.8 | 60% * 0 | 10% * 1 | 20% * 0.5 | 10% * 0.3 | 10% * 1
= 0,28
...

Figure 1. General approach for estimating the overall relevance of items for a given user in multiple dimensions

target user u, and a neighbor v. The symbol $\overline{r}_u$ corresponds to an average rating of user u and P denotes the set of products or items.

$$sim(u,v) = \frac{\sum_{i \in P}(r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{i \in P}(r_{u,i} - \overline{r}_u)^2}\sqrt{(r_{v,i} - \overline{r}_v)^2}} \qquad (1)$$

Another method uses association rules to explicitly model the dependency and similarity of items. So a rule could e.g. be: when a customer buys ItemA and buys ItemB, then the rule recommends to buy ItemC. One of the most widespread methods for calculating latent factors is matrix factorization, which is described in detail in [6]. Most of the modern recommendation systems use a combination (hybrid approach) of content filtering and collaborative filtering approaches to further improve the accuracy of recommendations. Beside these traditional approaches for implementing recommendation algorithms, several groups are working on the challenge of customizing recommendations and to build flexible recommendation queries. REQUEST: a query language for customizing recommendations was published by Adomavicius et. al. in 2011 [7], which promotes a custom query language to build flexible and customized recommendation queries based on multidimensional OLAP-cubes. Several contributions have been made by research groups that built various application scenarios for context-aware recommendation systems, ranging from tourism [8], restaurants [9], or even people (e.g. glancee.com).

III. Framework Requirements

Within this section we would like to discuss requirements a general framework for implementing context-aware recommendation systems has to fulfill. To discuss each requirement in detail would exceed the scope of our work, so we focus on several requirements that had a high priority for our use-case in VI.

A. Flexible and dynamic customization

A client-centric view on the recommendation process demands for a flexible user interface to enable the customization and fine tuning of recommendation impact factors for non-technical users. So the users should be able to control the learning and recommendation process at a most fine grain level, while the configuration and presentation should be on an abstract and understandable level. The user should be able to specify a variable number of impact factor dimensions and even to add custom defined impact factors. The framework should normalize all the chosen impact factors and automatically provide a list of recommended items that is sorted according to the weighted sum of normalized impact factors.

B. Temporal aspect

Temporal aspects [10] deal with the change of the context and with the change of the content profiles over a timeline. A recommendation framework has to consider the fact that the importance of specific datasets may change over time. It makes a big difference, if a person has bought an item yesterday or 10 years ago. A general framework has to cope with this varying impact.

C. Transparency

To raise the users confidence in recommendations, it is of crucial importance to give immediate and transparent feedback on recommendations. The recommendation framework has to provide a human understandable explanation for a given recommendation set. Sundaresan, from ebay research, published a great article about the 6 questions you have to

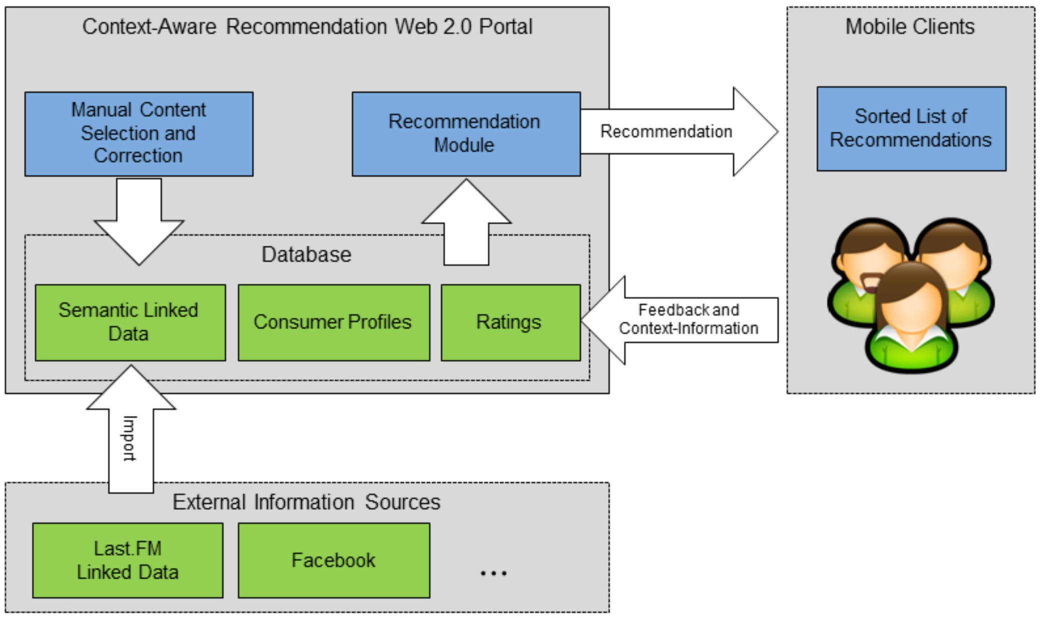
Context-Aware Recommendation Web 2.0 Portal
Manual Content Selection and Correction
Recommendation Module
Recommendation
Mobile Clients
Sorted List of Recommendations
Database
Semantic Linked Data
Consumer Profiles
Ratings
Feedback and Context-Information
Import
External Information Sources
Last.FM Linked Data
Facebook
...

Figure 2. General software architecture for the implementation of a context-aware recommendation system

address during the design and implementation of recommendation engines [11] (What, Where, When, Why, Who and How). He also points out that recommendation engines that address the transparency aspect (the Why question), offer a better conversion rate in e-commerce applications. There are several user studies that clearly show that addressing the transparency aspect improves the performance of recommendation engines [12].

D. Performance

In order to provide recommendations for mobile applications that consider the actual context of a mobile user, it is necessary to deliver immediate results. Recommendations that consider the location and activity of an user, have to react in time to provide recommendations in the specific situation, when a user needs them. As actual recommendation approaches harvest and analyse a huge amount of data, this requirement is critical for every implementation.

E. Quality

As users are implicitely benchmarking recommendation engines according to the quality of recommendations they are able to provide, it is necessary for a general framework to provide a standard approach for evaluating the quality of recommendation engines. A framework has to provide implicit and explicit quality evaluations, which means that the framework constantly evaluates the quality of results by using test data sets, as well as to explicitly ask the users for quality feedback.

IV. Approach

To implement a general framework for context-aware recommendation systems for mobile applications it is necessary to define the impact of context related, dynamic information on the recommendation of items for users. Compared to the traditional recommendation approaches, which were already discussed in Section II, we combine these traditional collaborative filtering approaches with user related context-information. For each application scenario there exists a collection of context aspects that are relevant for a recommendation in a certain situation. While the location information might not be relevant for recommending books in an online bookshop, it is of crucial importance for the recommendation of nearby restaurants. Each dimension of a given context, such as location, weather or even companions is represented through an impact function. An impact function defines the influence of one dimension of a given context on the overall relevance for a given user. All impact functions are of the given form

$$f_i : U \times P \times C \to R$$

where U represents the set of users, P the set of products and C a dimension of a given context (e.g. location, number of nearby friends, ...). The weighted sum of all normalized impact functions results in an overall relevance for a product p, a given user u and context c, where w_i represents a weight, that the end user defined for a specific dimension of the context:

$$f(u, p, c) = \sum_{i<P} f_i(u, p, c) w_i(c) \qquad (2)$$

A general framework for context-aware recommendation systems has to offer the basis for customizable recommendation engines, that consist of a variable and dynamic set of impact functions that can either be predefined by the framework (e.g. aspects such as distance, user ratings, history, friends, ...), or explicitly defined by users. Furthermore, the users are able to dynamically define and adapt the weight of different impact functions according to their preferences, which is shown in Fig. 1.

As Fig. 1 shows, the customized recommendation system within this example contains six different impact functions. Each of these impact functions calculates the relevance of an item for a given user for a specific dimension of a given context, such as distance, friends (companions), genre, ratings, prices or artist. The general framework is not limited to these six impact functions. The designers of recommendation systems should provide a domain-specific set of additional functions, in order to improve the quality of recommendations for the users in different application domains. The radar chart in the lower left corner of Fig. 1 visualizes the weight an individual user defined for a given set of impact functions. Each user specifies the personal weight of each dimension of the context. Another important feature of this approach is, that each of the impact functions can be defined and calculated by using completely different strategies. While the distance impact function could be a simple spatial query, the rating impact function could be implemented as traditional collaborative filtering approach.

V. System Architecture

The general software architecture for context-aware recommendation systems, that implements the approach in section IV is devided into a client-server model, that implements several subsystems. As it is shown in Fig. 2, the server defines all necessary subsystems for data access and third-party information retrieval, user interfaces for manual content selection and correction, as well as the recommendation module. The server database contains a matrix of given ratings, user and product profiles as well as additional semantic information. All additional semantic data can be accessed by using semantic web standards and query languages, such as RDF and SPARQL. The purpose of accessing these sources of semantic information is to receive additional item-based similarity measurements that are used in combination with the traditional collaborative filtering result. External sources of semantic information, such as Facebook or Last.FM, are either directly imported and duplicated, or directly accessed through a defined service interface. The decision if an external information source is either imported or directly accessed depends on the third-parties' service level agreements.

The recommendation module is responsible for calculating the recommendation approach in section IV and to communicate the resulting product ratings to the clients. The client-server communication is implemented as a lightweight REST (Representational state transfer) service approach. On the client-side, a local application is visualizing the resulting list of recommendations and is collecting the necessary context information in combination with the user's feedback on the given recommendations.

VI. Use-Case: Evntogram

The following use-case was selected out of a running project in cooperation with EVNTOGRAM, which is a platform operator for personalized and context-sensitive recommendation of social events. The philosophy of EVNTOGRAM is to analyze the users' habbits and activities, as well as their social interaction, in order to offer personalized and context-aware recommendations for social events, specifically in the domain of music events, such as concerts and music festivals. The general framework, explained in section IV and section V, helps to include various context-dimensions into the calculation of the relevance of an event for a given user. EVENTOGRAM records these context-dimensions, such as the users' activities, social interaction, music listening habbits and individual ratings, in order to sort a list of music events according to the calculated relevance, as it is shown in Fig. 3. In a first prototype EVENTOGRAM is trying to find out which subset of context-dimensions is providing good recommendations for the users. In that sense 'good' means the feedback the user is providing for a given ordering of items.

VII. Conclusion

In this work, we propose a general approach as well as a general software architecture for the implementation of context-aware recommendation systems was presented. The approach as well as the framework offers high flexibility according to the definition and configuration of new impact functions, which influence the recommendation of items for given users. The framework is domain-free, which means that this approach can be implemented and adapted for different application domains. The context-aware recommendation of items of all kind, ranging from products in e-commerce to activities and services in sport and fun will get much attention in future software development. We think that a general framework for designing and implementing such recommendation systems for different application domains is of great importance. The next steps within our work will be to gather empirical feedback from the community within the given use-case of recommending music related events.

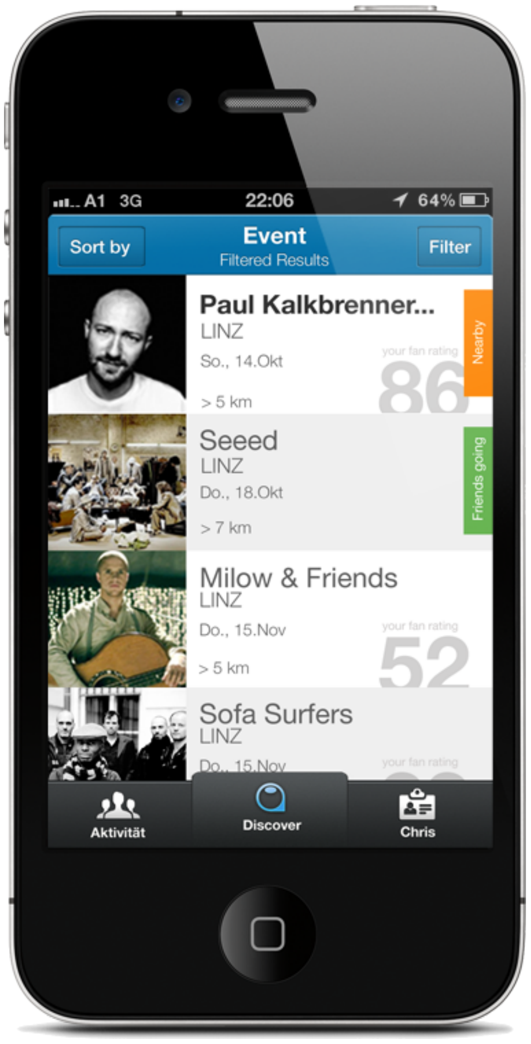

Figure 3. EVNTOGRAM Music Event Recommendation App

REFERENCES

[1] R. M. Bell, Y. Koren, and C. Volinsky, "The bellkor solution to the netflix prize," [retrieved: jan. 2013]. Available: http://www2.research.att.com/ volinsky/netflix/ProgressPrize2007BellKorSolution.pdf

[2] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on Mobile Computing Systems and Applications. IEEE, 1994, pp. 85–90.

[3] P. J. Brown, J. D. Bovey, and X. Chen, "Context-aware applications: From the laboratory to the marketplace," IEEE Personal Communication, vol. 4, no. 5, Oct. 1997, pp. 58–64.

[4] A. Dey and G. Abowd, "Towards a better understanding of context and context-awareness," in CHI 2000 Workshop on The What, Who, Where, When, and How of Context-Awareness, 2000.

[5] W. Beer, V. Christian, A. Ferscha, and L. Mehrmann, "Modeling context-aware behavior by interpreted eca rules," Euro-Par 2003 Parallel Processing, 2003, pp. 1064–1073.

[6] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," IEEE Computer, vol. 42, no. 8, Aug. 2009, pp. 30–37.

[7] G. Adomavicius, A. Tuzhilin, and R. Zheng, "Request: A query language for customizing recommendations," Info. Sys. Research, vol. 22, no. 1, Mar. 2011, pp. 99–117.

[8] W. Beer and A. Wagner, "Smart books: adding context-awareness and interaction to electronic books," in Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia, MoMM '11. New York, NY, USA: ACM, 2011, pp. 218–222.

[9] V.-G. Blanca, G.-S. Gabriel, and P.-M. Rafael, "Effects of relevant contextual features in the performance of a restaurant recommender system," in In RecSys11: Workshop on Context Aware Recommender Systems (CARS-2011), 2011.

[10] Y. Koren, "Collaborative filtering with temporal dynamics," in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '09, New York, NY, USA: ACM, 2009, pp. 447–456.

[11] N. Sundaresan, "Recommender systems at the long tail," in Proceedings of the fifth ACM conference on Recommender systems, ser. RecSys '11, New York, NY, USA: ACM, 2011, pp. 1–6.

[12] R. Sinha and K. Swearingen, "The role of transparency in recommender systems," in CHI '02 extended abstracts on Human factors in computing systems, ser. CHI EA '02. New York, NY, USA: ACM, 2002, pp. 830–831.