Configuring and Monitoring Recommender System as a Service

David Ben-Shimon &
Alexander Tsikinovsky
YooChoose Labs Ltd
Hagoren 6, Omer 84965, Israel
{david.ben-shimon, alexander.tskinovsky} @yoochoose.com

Michael Friedmann &
Johannes Hörle
YooChoose GmbH, Bonner 484, Cologne 50968, Germany
{michael.friedmann, johannes.hoerle} @yoochoose.com

ABSTRACT
Many small and medium e-commerce retailers and publishers use recommender systems (RS) to personalize the website content. Many of them do not have an on premise solution for doing that, but rather contact a company that delivers the RS as a service to their website. The service is then responsible for collecting and storing the data, building recommendation models, and answering recommendation requests.
Once the integration to such a service is done, the e-commerce retailer still wish to have some control on the service. Control that allows him to configure the recommendation models, turn off/on the service, apply filters on recommendations, define fallback models and more. In this demo we provide an overview of a real backend system which enables to a typical website owner exactly these capabilities. Capabilities for controlling the RS service in terms of configuration, management and monitoring.

Categories and Subject Descriptors
H.5 **Information interfaces and presentation: Miscellaneous**

General Terms
Recommender System as a Service, Configuration, API

Keywords
Recommender System as a Service; Configuration; Management; Graphic User Interface

1. INTRODUCTION
Many e-commerce businesses record activities of users in order to enable personalization of the content and recommend items to view or to purchase. Such personalization is expected to improve the user experience and satisfaction and to increase the revenue of the e-business.

The decision of an e-business to apply personalization and integrate recommendations into their existing systems can be challenging and even risky, as the expected return on investment is difficult to predict. The required knowledge, expertise, and resources needed for implementing a personalized recommender system may be beyond the reach of medium and small e-businesses. To minimize the investment and risk, many small or medium e-businesses prefer to purchase the recommender system as a service [1] [4]. The providers of such services are called RS providers. Such services allow the e-businesses to request recommendations for the active user through a remote server, which is not owned or operated by the e-business owner. These services can be purchased and activated rapidly. They can also be turned off without any major cost in case of an unsuccessful deployment that does not withstands the goals of the e-business owner. Thus, such services reduce the required initial investment and represent very little risk to the e-business. The Integration to such a services is usually done via a REST API [3] which is provided by the RS provider, or could be done automatically if the required RS services are limited[1]. Once the integration is done the website is sending user activities to the service, the service is building recommendation models based on that, and then is able to respond to online recommendation requests out from the models.

In order to be able to serve many types of potential customers the RS provider builds the service so it could be configured and adapted for many type of customers i.e. retailers, publishers etc. This includes the ability to support the building of several type of models i.e. Collaborative Filtering (CF), Content Based (CB), Popularity etc., supporting recommendations for different item types such as products, images, movies, article and more. Additionally there is an option to define fallback models. Fallback models are models that deliver recommendations when the preferred primary model cannot deliver recommendations from whatever reason. Additional aspects related to possible configurations exist but are not discussed here. Such a service is also working with default values so it can work smoothly, tough not ideally, in case no configuration has been provided.

Many customers that are connected to the service wish to have the ability to control the service generally and these aspects specifically. In this demo we provide an insight on a real productive application that enables each customer, which is connected to service of the RS provider, to configure and manage the service via a graphic user interface according to his needs and understanding.

2. CONFIGURING AND MANAGING THE SERVICE
As mentioned above the website is integrating the service via REST API. This integration then enables the service to do three main things. First, to collect and store the desired activities of the end users; the storage is in the RS provider servers. Second, once enough data has been collected, the service is building recommendation models using the model-based [2] approach. Third, the service is ready to answer recommendation requests very fast, so given a user, item or any preconfigured type of input, the service provides in return a list of recommendations using JSON or XML format. For more details on how to build such a service see the Netflix tutorial [5].

RecSys'14, October 6–10, 2014, Foster City, Silicon Valley, CA, USA.
ACM 978-1-4503-2668-1/14/10.
http://dx.doi.org/10.1145/2645710.2645713

We now wish to define a new concept that we use in our system and is called *scenario*. Scenario is an abstract phrase that encapsulates within it all the information that the system needs in order to deliver recommendations according to some specific customers' expectations. From the customer perspective it could be viewed as a component that is responsible to populate a certain box/area in the GUI with recommendations.

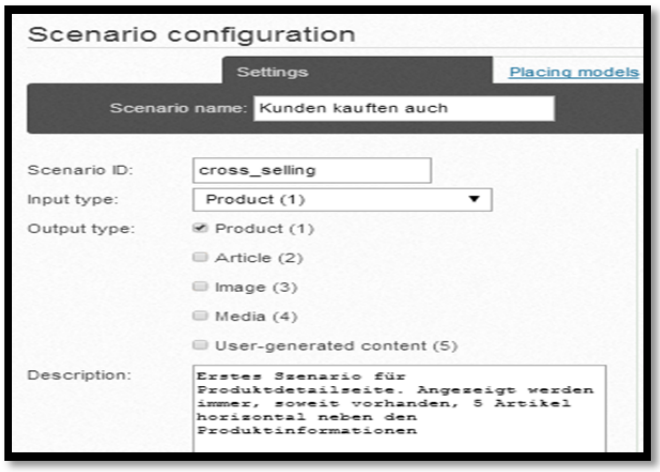

Figure 1. Snapshot of scenario configuration

Figure 1 provides a snapshot from the system when a new scenario is about to be generated by the customer. The customer needs to give a name to the scenario and to define the item type of the input and the output. I.e. the model is based on buying events of products and also recommends products. Description is not mandatory. Customers can apply filters on the recommendations. Filters are rules which work on the scenario and restrict the scenario from delivering certain items. I.e. do not recommend to the user items he already purchased in the past. This filter for instance may be valid in many stores but in an online grocery it is not. Once the customer provides these details for the scenario, he now has to define the recommendation models that has to be built for this scenario.

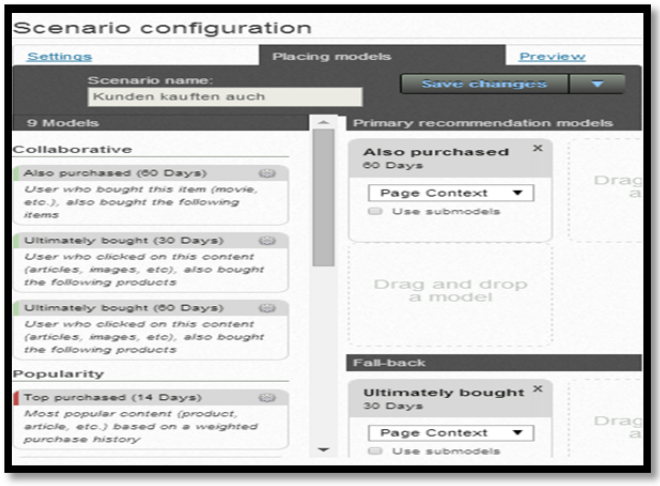

Figure 2. Placing models in the scenario

Figure 2 provides a snapshot of placing models inside a scenario. Here the customers decides what will be the type of the models, i.e. whether the recommendations will come from memory-based CF model, popularity model and the like. The customer also defines here the ranking of the models; meaning what is the primary model and which models serve as fallback and in what order. There is also the ability to configure recommendation list which is generated from more than one model. All this information is required for building the desired models and serving the requests from this scenario properly. Functionalities like defining AB tests, previewing recommendations, applying sophisticated filters and more, are also available but are not presented here. Finally whenever the customer enters the system he gets statistics of the service. Figure 3 provides a snapshot of this information. For instance the customer can view the statistics on the clicks, purchases, recommendation requests, AB test results and more.

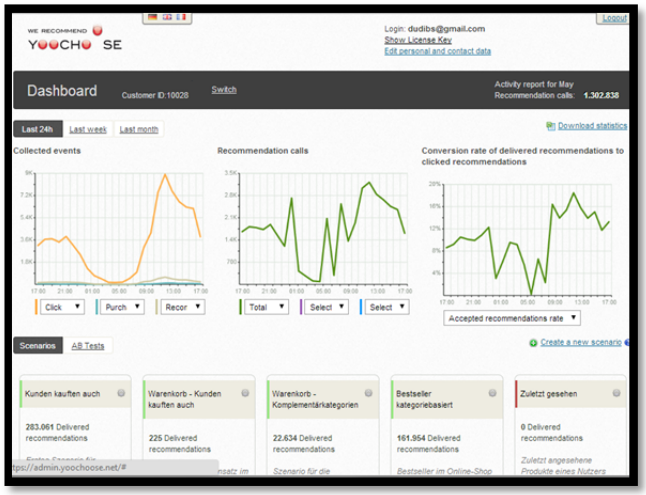

Figure 3. Overview on the entire service

3. SUMMARY

In this demo we provide an insight on how customers of a recommender system as a service can configure and manage the service. Part of the customers do not want to be involved in this configuration but some of them are eager for such a dashboard application to manage and configure the service for various reasons.

4. REFFERNCES

[1] Ben Shimon, D., Friedman, M., Hoerle, J., Tsikinovsky, A., Gude, R., & Alukhanov, R. (2014, February). Deploying recommender system for the masses. In Proceedings of the companion publication of the 19th international conference on Intelligent User Interfaces (pp. 1-4). ACM.

[2] Breese, J. S., Heckerman, D., & Kadie, C. (1998, July). Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (pp. 43-52). Morgan Kaufmann Publishers Inc.

[3] Masse, M. (2011). REST API design rulebook. " O'Reilly Media, Inc.".

[4] Ronen, R., Koenigstein, N., Ziklik, E., Sitruk, M., Yaari, R., & Haiby-Weiss, N. (2013, October). Sage: recommender engine as a cloud service. In Proceedings of the 7th ACM conference on Recommender systems (pp. 475-476). ACM.

[5] http://recsys.acm.org/2012/tutorials.html#building