Log-based e-commerce recommendation system: Project proposal

Katarina Smolikova, 1528686

***Abstract*—The Log-based e-commerce recommendation system is a service designed especially for e-shops but also for any website where there is some content recommendation required. That could also be newspapers, magazines, media, job boards, social networks etc. The service uses the data about user interaction which is saved for analytic and safety purposes to train a machine-learning recommendation engine. This way, the data from the website is only sent into one service which can be then used for both analysis and recommendations. That is not only more efficient, but also more convenient for the website owner. In this paper we describe the design of the service, specify the tools which will be used for the implementation and outline the data concerns in our service.**

***Index Terms*—proposal, model, design, project**

I. Introduction

THE main design principle of Log-based e-commerce recommendation system is separation of concerns. The components used in the service are loosely coupled and therefore they are not dependent on a specific technology. In this project, the service is based on Elasticsearch and PredictionIO. However, by design, it would be fairly easy to change the used tools. In this paper firstly, we will present the conceptual design of the services used in the project and we will describe their interactions. Further, we will specify the services and tools which will be used in the project. Last but not least, we will describe how the application-specific services will support data-concerns and the trafe-offs between selected concerns.

II. Conceptual design of services

In our design we will join two wastly used services in an e-commerce business: loggging service and recommendation service. The whole system consists of number of smaller services. The data source are the events happening on the website. The system can be used for keeping logs about all the events, even the ones not relevant for the recommendation system. The data from the web application is sent to the *Analytics service*. It processes the data and creates an object describing event which happened. Furhter, it sends this information to *Data service*. The *Data service* is responsible for storing the information. When the information has been sucessfully stored, the *Data service* sends the confirmation back to the *Analytics service* which notifies the *Notification service* that new data has been saved. The *Notification service* modifies the information and if it is considered relevant it is sent to the *Recommendation service*. Not all the events which happen are relevant for the *Recommendation service*. There might be events which are important for analytics and therefore need to be saved in the *Data service*, however, they might not be significant for the *Recommendation service*. The *Recommendation service* is based on machine-learning and according to previous interactions, it is able to provide information on user or item-specific recommendations. It provides this information through REST API to *Human service*. The *Human Service* is based on simple interface where the website administrator can manually insert or retrieva data. Moreover, he will be able to perform analysis on the data stored in the *Data service* and possibly adjust the parameters for the *Recommendation service*.

III. Used tools

To simulate the data-stream we will use the open data from Expedia.com. The data contains the real data about user interaction from expedia.com over few weeks. The data will be sent in fast random intervals. That is why in our mini-project, actually at the very beginning of the flow, there will be a *Simulation service* imitating the real event flow which will be sent in the Log-based e-commerce recommendation system. The main programming language used in this project will be Java. The data from the *Simulation service* will be sent to the *Analytics service* through REST interface. The *Data service* will be based on Elasticsearch. The *Analytics service* will send the data through Elasticsearch Java API. After the *Analytics service* receives the confirmation on saving the data, it will sent the data to the *Notification service* which will be accessible through REST API. Moreover, the *Recommendation service* will based on PredictionIO and the events will be sent there through Java SDK. Last but not least the *Human service* will be a simple web interface. The interface will communicate with the *Recommendation service* through raw HTTP requests. The analysis of the data will be done through Kibana visualisation tool for Elastic.

IV. Data concerns

One of the main concerns in our project is the quality of data. The service needs high number of events to be able to provide a precise recommendation. Therefore, we can say that the data quality depends heavily on the amount of data sent. However, the elasticity of data resources in our project is high. The data can be sent to our service thorugh REST interface and therefore data from various sources can be used. Moreover, the service is prepared to handle all kinds of events which can be useful especially for customers with low traffic. For example, if there is not enough data for purchases, the system can also consider other events, such as user browsing the pictures of the product, user spending more than a minute on the product detail, user rated the product etc. Since the *Data service* is based on Elasticsearch, it is perfectly suitable for large amounts of data. In case, there needs to be another machine added or the data needs to be stored in different location for legal purposes, the architecture of the service enables the data stores to be easily modified. Furthermore, for the users with low traffic numbers, it might be more suitable to use item-based recommendation instead of user-based. By changing the algorithm parameters in *Recommendation service* the recommendations can be more adjusted to such cases. To sum up, the service offers many ways of how to deal with the data quality concern.

The quality of the service itself will be measured based on the actions taken by human. The main objective of the service is to provide user with recommended items in which they are interested and which they will eventually buy. That is why the two main key performance indicators will be click-through rate (how many users clicked on a recommended item) and the conversion rate (how many users bought a recommended item). In case these two statistics are not achieving satisfiable results, the parameters of the *Recommendation service* need to be tuned. The KPIs can be checked through the service user interface by the data analyst. Moreover, there is a possibility to tune the algorithm parameters to achieve better precision. The elasticity of the quality of analytics is very high, since there is a possibility to deal with changes in quality of the data and performance at runtime.

Another concerns are which are directly connected are the speed of data processing and quality of results. The results of the service need to be available in milliseconds. Except of cases where recommendations are loaded asynchronously, the service response time directly influences the page load time. As can be seen in figure 1 the *Data service* sends the confirmation of saved information back to the *Analytics Service* which then sends the information to the *Notification service*. For quicker response time it would be more suitable that the *Data service* would send the information directly to the *Notification service*. This could be done using Elasticsearch X-Pack Watchers [1]. However, the watchers only offer time-based triggers. Therefore, we could for example set the watcher to watch the *Data service* every second if there were any new events added and if so, resend them to the *Notification service*. This way, we could eliminate the communication between the *Analytics service* and *Notification service* and therefore increase the performance of the service. However, the quality of the results is also essential. The data needs to be processed very quickly for the service to be able to use the latest information about the user interaction. For example, if user buys an item, the page after the purchase often contains more recommended items. It is desired that these recommendations already consider the purchase that the user just made. If the *Data service* would be checked periodically for changes, it could easily happen, that the interactions done in previous seconds would not be taken into account by the *Recommendation service*. That is why, in this case we considered the quality of the results to be more important than the speed of the data processing.

[1]https://www.elastic.co/guide/en/x-pack/current/how-watcher-works.html

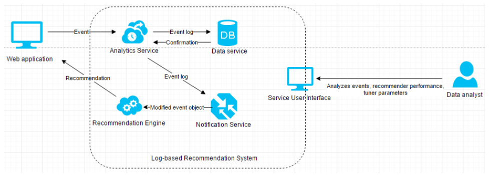

Fig. 1. Log-based e-commerce recommendation system : Project design

V. Conclusion

The Log-based e-commerce recommendation system creates a compact service by joining multiple smaller services. The whole service is provided together with an installation manual which makes it possible to install the service for anybody either on their local machine or on a cloud. Moreover, for the demonstration of the functinality there is a sample dataset together with the *Simulation service* provided. The main data concerns are the data quality and the quality of the results. Both of these concerns can be addressed at runtime directly by the data analyst through the user interface. Moreover, the user interface provides an overview on the key performance indicators. This enabled the customer to be able to control, analyze and modify the service performance all in one place.