ALARM: A Light Application for Recommendation and Monitoring

Arthur Rodrigues Batista
State University of Maringá
Maringá, Brazil
ra105422@uem.br

Luiz Fellipe Machi Pereira
State University of Maringá
Maringá, Brazil
ra103491@uem.br

Marcos Aurélio Domingues
State University of Maringá
Maringá, Brazil
madomingues@uem.br

ABSTRACT

Big companies usually have human and financial resources to personalize their websites. On the other hand, small and medium-sized companies usually do not have such resources. In this paper we propose *ALARM*: A Light Application for Recommendation and Monitoring. This free platform enables automatic recommendations and monitoring in small and medium-sized websites. The platform is independent of the site structure, as well as monitoring and recommendation methods which may be used in it. We illustrate the features of the platform in a case study, where we show how it can be used to provide recommendations as well as to analyze them.

KEYWORDS

Web Personalization, Recommender System, Monitoring, Platform

1 INTRODUCTION

Commercial and non-commercial companies have common goals in relation to the organization of their websites: users should be able to easily access the information contained in the site, that is relevant to them, and the site should have on each page the most useful links to pursue the search for information [5, 6]. Additionally, it is also important to have data to support the management of the website [2, 4]. However, meeting these goals permanently means that companies must have human and financial resources available; and small and medium-sized companies usually do not have such resources.

In this paper we propose *ALARM*: A Light Application for Recommendation and Monitoring. This platform enables automatic recommendations and monitoring in small and medium-sized websites. The platform is independent of the site structure, as well as monitoring and recommendations methods which may be used in it. The platform is simple to install and easy to be used, and it can be downloaded for free.

The remaining of this paper is organized as follows. In Section 2, we present the architecture of our platform. The case study used to validate our proposal is presented in Section 3. Finally, we discuss the conclusion and directions for future work in Section 4.

2 ARCHITECTURE OF THE PLATFORM

The architecture for the proposed platform is illustrated in Figure 1. There, a central broker mediates the communication between the website and the recommenders. The recommendation requests are channeled throw the broker to one or more recommenders connected to the platform. Recommendation requests are forward to recommenders and the list of recommendations (i.e. web pages) is loaded on the web side in a carousel. By using an AJAX script in the website, the platform is able to collect recommendation requests and user interactions, and reporting them to the broker. The AJAX script is also in charge of loading the recommendations in the carousel. All communication with the website – recommendation and interaction messages – is recorded in a database and made available to recommenders. By using data from the database, the recommenders can provide their response to recommendation requests. The platform also has a monitoring tool that uses data from the database to analyze the usage of the recommendations. Additionally, the monitoring tool also allows the inclusion and the management of new recommenders. In the platform, the components exchange asynchronous messages by using the JSON notation[1].

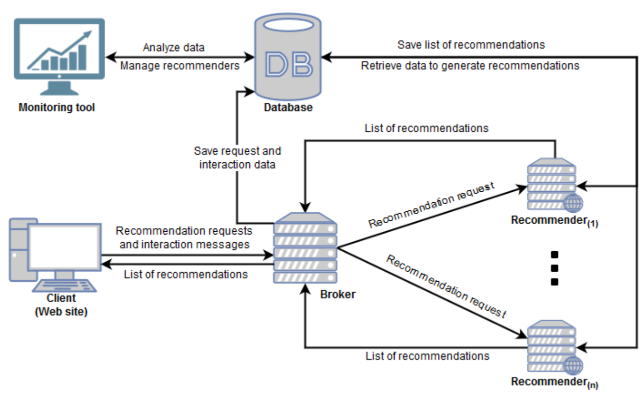

Figure 1: Architecture of the platform *ALARM*.

The platform is based on the Model-View-Template[2] (MVT) framework and the Web Server Gateway Interface[3] (WSGI). To implement the platform we used the Pyhton[4] programming language and the Django framework[5]. As it uses relative addresses, the platform and the website can be located in different servers.

Implemented by using the previous technologies, our proposed platform is simple of installing and using, and enables automatic

In: XVII Workshop de Trabalhos de Iniciação Científica (WTIC 2020), São Luís, Brasil. Anais Estendidos do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia). Porto Alegre: Sociedade Brasileira de Computação, 2020.

ISSN 2596-1683

[1] https://www.json.org
[2] https://djangobook.com/mdj2-django-structure
[3] https://www.fullstackpython.com/wsgi-servers.html
[4] https://www.python.org
[5] https://www.djangoproject.com

recommendations and monitoring in small and medium-sized websites. Each individual component of the platform is described in details as follows.

2.1 Broker Component

It should be noted that several requests will hit the broker: one recommendation request for each recommender involved, and several interaction requests, depending on the user interaction on the website. Thus, the broker must be very efficient since it will process all requests coming from a website. Here, the efficiency is reached by implementing JSON messages and pooling of data to be stored in the database.

2.2 Database Component

We have designed a database with four tables that store all data of a website that the platform needs (Figure 2). In the table **ClickStream** we store interaction data, such as the *IP* of the client computer, identification of the user (*idUser*), *current* web page, previously accessed web page (*href*), *class* in the tag of the web page link, textual content of the web page (*text*), *timestamp* of the current access, date and hour in the timestamp format (*dateTimeStamp*) and date in the format ISO 8601 (*dateR*). The field *idClick* is the primary key of the table. In the table **Recommenders** we have information about the recommender systems available on the platform. In this table, we store the name of the recommender system, as the primary key (*rid*), and its status *active* (i.e. value "on" or "off"). In the third table, called **GeneratedRecommendation**, we store the *url* of the recommendations generated by the recommender. The interaction and recommender used to generate the recommendations are represented as foreign keys (fields *rid* and *idClick*). The field *idRow* is the primary key of the table. In the last table, called **AccessedRecommendation**, we store (through the foreign keys *rid* and *idClick*) the recommendations accessed/consumed by the users. The field *idRow* is the primary key of the table.

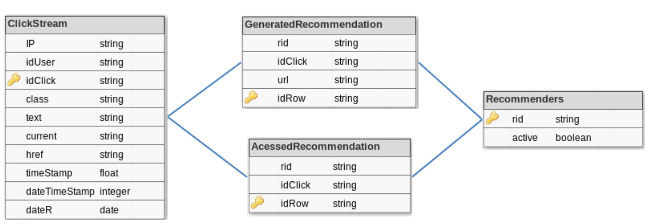

Figure 2: Schema of the database.

In order to develop a light platform, the database was implemented by using the SQLite[6] as database management system.

2.3 Recommender Component

Recommenders are pluggable components that process a recommendation request and provide a recommendation response that is delivered to the website and inserted in a carousel. For the current version of the platform, we have implemented two different algorithms well-known in the literature: 1) the Most Popular recommendation [1], and 2) the Item-Based Collaborative Filtering (IBCF) [3].

The Most Popular algorithm simply recommends the top-N most accessed pages on the website. On the other hand, the IBCF algorithm analyses historical information to identify relations among items. The recommendation model is a matrix representing the similarities between all the pairs of items, according to a similarity measure. In our case, an item is an accessed web page and the similarity measure is the cosine angle, defined by

$$sim(i,j) = cos(\vec{i}, \vec{j}) = \frac{\vec{i} . \vec{j}}{||\vec{i}|| * ||\vec{j}||},$$

where $\vec{i}$ and $\vec{j}$ are binary vectors representing the users that accessed the pages i and j, and "." denotes the dot-product of the two vectors.

If we want the N best recommendations, we use the recommendation model to output a list with the top-N pages more similar to the one being accessed by the user at the moment.

2.4 Client (Website) Component

The client component in the website is responsible for requesting recommendation to the broker, reporting user interaction to be recorded in the database, and subsequently for processing the recommendations received from the recommenders, by putting them in the carousel of the website. As already stated, this component was developed as an AJAX script that uses the JQuery framework[7] to interact with the HTML and CSS elements. In Figure 3 we illustrate the recommendations in the carousel of a website.

Figure 3: The recommendation carousel in a website.

2.5 Monitoring Component

This component allows us to assess the recommendations on a website. It provides different types of reports and their evolution in time (i.e. hour, day, month and year), including:

- Statistics about the number of different pages, users, recommendations, accesses/clicks on the web pages, etc;
- Percentage of accesses/clicks to the website that follow from recommendations generated by the recommenders (recommendation adhesion);
- Percentage of recommendations that are accessed/clicked by the users (recommendation efficacy).

[6]https://www.sqlite.org

[7]https://jquery.co

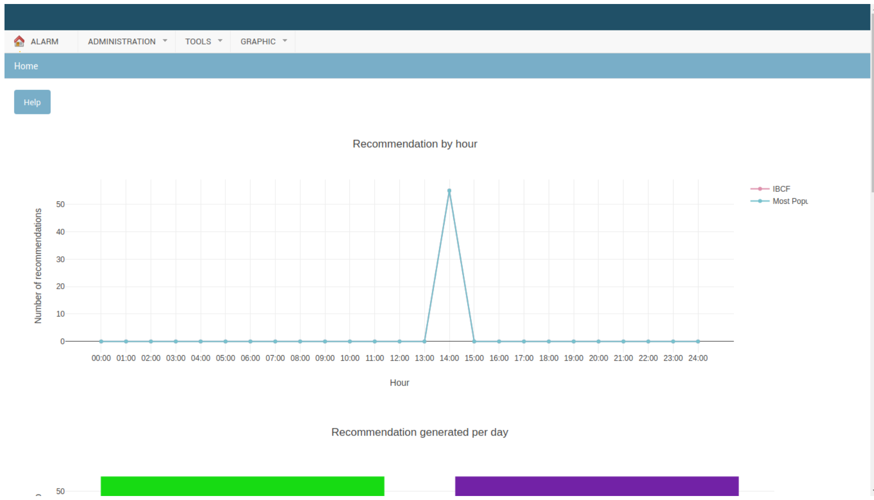

Figure 4: Screen of the monitoring component of the platform *ALARM*.

With these reports, we can measure the impact of the recommendations from different perspectives. We can assess what users actually do when interacting with the site and, in particular, how they react to recommendations. This enables us to better understand the behavior of users, identify usage patterns, problems with the content and structure of the site and unmet needs. In Figure 4, we can see a screen of the monitoring component.

In addition, this component also provides a mechanism to include and manage new recommendation algorithms in the platform, as can be see in Figures 5 and 6. After being inserted, the algorithms are mapped and made available automatically at startup of the platform and can be activated or deactivated at any time on the monitoring component.

```
"""
Note:
The parameter "data" contains information about a click/access to a web page, such as:

data.ip, -> IP of the client computer
data.idUser, -> identification of the user
data.idClick, -> identification of the click
data.classe, -> Class presented in the tag of the clicked web page
data.texto, -> Textual content of the clicked web page
data.current, -> Url of the clicked web page
data.href, -> Url of the previous clicked web page
data.timestamp, -> Timestamp of the click
data.dateTimestamp, -> Date and hour in the timestamp format
data.dateR -> Date in the format ISO 8601

The vector "recomendacoes" will contain the urls of the recommended web pages

Example:
recomendacoes=[link_1.com,link_2.com ... link_n.com]
"""

from webapp.models import Adapters, Post, RecomendacaoAcessada, RecomendacaoGerada
def NameOfRecommendator(request,data):
    recomendacoes = []
    """
        Insert code here!
    """
    return recomendacoes
```

Figure 5: Monitoring component showing instructions to include new recommendation algorithms.

The monitoring tool was made available following the django-admin-tools[8] framework, in which the graphs were added with the use of the Ploty.js[9] library.

[8]https://github.com/django-admin-tools/django-admin-tools
[9]https://plot.ly/javascript

ID	Status	Delete	Vizualize code
Most Popular		Delete	Visualize
IBCF		Delete	Visualize

Figure 6: Monitoring component to manage the recommendation algorithms.

3 CASE STUDY

To validate our proposal we deployed the platform in the *PET-Informática* website[10]. We ran the case study during April, 2019; and the main goal was to analyze the performance of platform in a small/medium-size website. To do that, we added two recommendation algorithms in the platform, the Most Popular and the Item-Based Collaborative Filtering (IBCF) algorithms. The platform showed itself adequate to provide recommendations by responding to recommendation requests just in time. Additionally, we also monitored the recommendations with the monitoring component. Following, we discuss some graphics obtained with monitoring component. In Figure 7 we can see the number of recommendations generated along the month. There, we can see a similar number of recommendations generated by both recommendation algorithms.

Although there are many ways to measure the efficiency of the recommendation algorithms, in Figure 8 we analyze the number of recommendations accessed/consumed along the month. Here, we are assuming that the higher the number of recommendations consumed by the users the better is the recommendation algorithm.

In Figure 8 we can see that the recommendations generated by the IBCF algorithm were the most consumed by the users. This fact gave us evidences that the IBCF is a better option to the website.

Finally, in Figure 9 we see the value of the recommendation efficacy measure for both recommendation algorithms. Again, we can see that the IBCF algorithm is a better option to the website, since its percentage of recommendations that are accessed/clicked by the users is higher than for the Most Popular algorithm.

[10]http://www.din.uem.br/pet

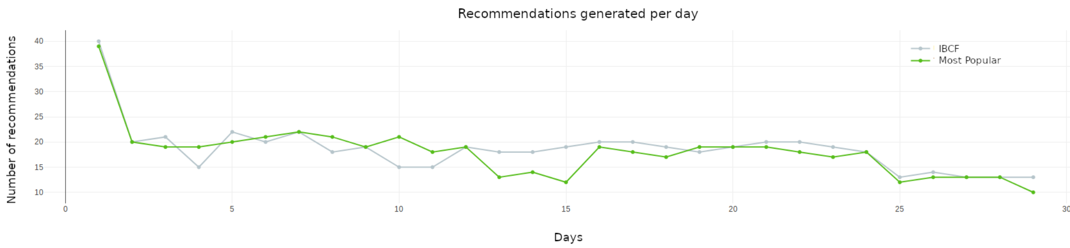

Figure 7: Number of recommendations generated along the month.

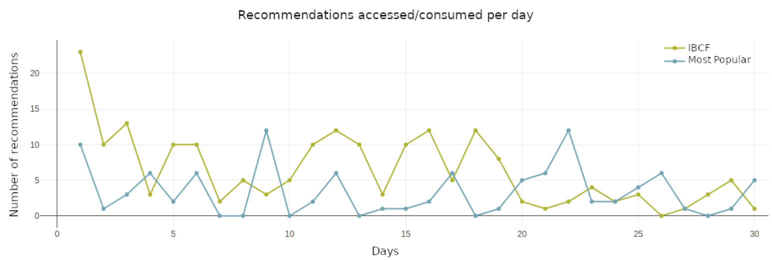

Figure 8: Number of recommendations accessed/consumed by the users along the month.

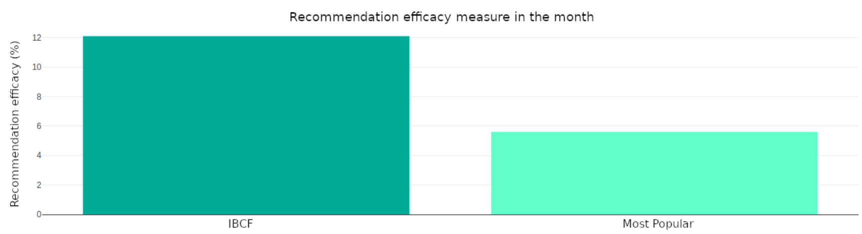

Figure 9: Recommendation efficacy measure for April, 2019.

4 CONCLUSION AND FUTURE WORK

In this work we proposed *ALARM*: A Light Application for Recommendation and Monitoring. The platform is independent of the site structure, as well as monitoring and recommendations methods which may be used in it. The platform is simple to install and easy to be used. We validated our proposal through a case study, which demonstrated that the platform is able to provide automatic recommendations and monitoring in small and medium-sized websites.

As future work, we intend to add new recommendation algorithms and evaluation measures in the platform. Additionally, we also plan to validate continuously our platform on other websites.

The platform is currently available for download for free in the Github[11].

ACKNOWLEDGMENTS

This work was financed by Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq) - grant #403648/2016-5; and the Tutorial Education Program, developed by Ministry of Education, in particular the PET-Informática of the State University of Maringá.

REFERENCES

[1] Marco Bressan, Stefano Leucci, Alessandro Panconesi, Prabhakar Raghavan, and Erisa Terolli. 2016. The Limits of Popularity-Based Recommendations, and the Role of Social Ties. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, 745–754.

[2] Ana Ribeiro Carneiro, Alípio M. Jorge, Pedro Quelhas Brito, and Marcos Aurélio Domingues. 2014. Measuring the Effectiveness of an E-Commerce Site Through Web and Sales Activity. Proceedings in Mathematics Statistics. 1ed.: Springer International Publishing, 149–162.

[3] M. Deshpande and G. Karypis. 2004. Item-based top-N Recommendation Algorithms. *ACM Transactions on Information Systems* 22, 1 (2004), 143–177.

[4] Marcos Aurélio Domingues, Carlos Soares, and Alípio M. Jorge. 2012. Using statistics, visualization and data mining for monitoring the quality of meta-data in web portals. Information Systems and e-Business Management, 569–595.

[5] Alípio M. Jorge, João Vinagre, Marcos Aurélio Domingues, João Gama, Carlos Soares, Pawel Matuszyk, and Myra Spiliopoulou. 2016. Scalable Online Top-N Recommender Systems. In *E-Commerce and Web Technologies - 17th International Conference, EC-Web 2016, Porto, Portugal, Revised Selected Papers.* 3–20.

[6] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul Kantor. 2010. Recommender Systems Handbook. Springer-Verlag New York, New York, NY, USA.

[11] https://github.com/LFMP/ALARM