2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology

UPComp - a PHP component for recommendation based on user behaviour

Ladislav Peška, Alan Eckhardt, Peter Vojtáš
Department of Software Engineering
Charles University in Prague
Prague, Czech Republic
Email: lpeska@seznam.cz, eckhardt@ksi.mff.cuni.cz, vojtas@ksi.mff.cuni.cz

***Abstract*—In this paper, we investigate the possibilities of interpreting user behaviour in order to learn his/her preferences. UPComp, a PHP component enabling use of user preferences for recommendation, is described. UPComp is a standalone component that can be integrated into any PHP web with only basic knowledge of PHP, HTML and SQL.**

The methods of user behaviour interpretation are evaluated on a real web shop with tourist trips using UPComp.

***Keywords*-Recommender systems; Web design; User preferences**

I. Introduction

Recommender systems has got a lot of attention in a last few years mainly because of enormous expansion of internet and, from the scientific point of view, also thanks to the NetFlix prize challenge [1]. Most of the efforts were in developing algorithms that match most closely the preferences of the user. Especially in the final stages of NetFlix, the methods get very complex with little actual knowledge about the user preferences - what and why the user prefers. Implementing such methods for an average webmaster of a small webshop is very difficult.

That was the motivation for developing a component for recommendation that would be simple enough to deploy for a ordinary webmaster, yet extendible and flexible. The main focus was on the ease of use, leaving the complex algorithms as future work and possible enhancements. Also the component shouldn't require modification of existing user interface.

In this paper, we propose a new PHP recommendation component UPComp. UPComp is a flexible and easy to embed. We do not want to propose a new recommendation strategy, but we want to allow average webmasters to implement of recommendation to their websites. We also present experiments on a real e-shop with considerable traffic, where naive recommendation was previously implemented.

The paper is structured into five sections. The second section covers some of the related work, UPComp is described in the third section. Experiments on the real web shop are described in the fourth section and finally the paper concludes in the fifth section.

II. Related work

Our motivation is an unknown user, which is just a new session in web server. For being able to recommend preferred products to the user, we have to have some sort of feedback from the user on his/her preferences. Two main types of user feedback are distinguished in [2] - explicit and implicit.

A. Explicit feedback

Explicit feedback is given by the user in order to express his/her preferences. A typical example is giving an object a rating, often in form of stars (1-5 stars). We can be sure that the object is preferred to the degree specified by the user.

Having explicit feedback, recommendation is much simpler. There are many methods learning preferences from the ratings that are able to order the whole database according to the learnt preferences [3], [4], [5], [6].

The main drawback of explicit feedback is the possible unwillingness of the users to give it and that rating interrupts typical "workflow" of the user on the website. The user often does not know what is the benefit of giving the rating, so he/she is unwilling to do it.

B. Implicit feedback

Implicit feedback is actions that the user does while pursuing his/her goal - most often searching for the best product and buying it. The main difference to explicit feedback is that it does not require any special effort from the user, the user interface does not need to change and the user behaves in his/her usual pattern.

There are many types of actions that can be captured. Viewing time and scrolling are used most often [7]; but the results are ambiguous. Some works state that there is a correlation between interest and viewing time, while others disagree. For example, in [8], the same amount of time was spent viewing relevant and irrelevant documents.

The price we pay when using implicit feedback is that the user actions are very difficult to interpret. Does the long viewing time correspond to the high preference? There is also the difficulty in expressing negative preference - is not clicking on a product detail expression of negative preference or expression of not noticing the link?

978-0-7695-4513-4/11 $26.00 © 2011 IEEE
DOI 10.1109/WI-IAT.2011.180

```
class NewMethod extends AbstractMethod
 implements ObjectSimilarity{
 public function getSimilarObjects(
     $objectID, $noOfObjects, $objectList="");
 public function getSimilarObjectsFrom(
     $from,$objectID,$noOfObjects, $objectList="");
 }
```

Figure 1. New recommendation method stub

C. Other possibilities for recommendation

There are open source systems that already contain the explicit recommendation possibility [9]. The description of various approaches is in [10] but there is no possibility to download source codes. To our best knowledge, UPComp is a first attempt to provide a standalone component that is able to be included in an existing system.

III. UPComp

Our main objective was to develop a recommender system called UPComp suitable for e-shops. UPComp is easy to implement and it should increase the effectiveness of the e-shop. The most important function of recommender systems in e-shops is to provide a set top-k "best" objects. The recommended set of objects may differ at different contexts, e.g. recommended items in shop cart would be related to the items in the cart, but recommended items in a category listing would be related to the selected category. Recommended objects are typically displayed in a small area, in addition to the main information on the page. This fact results into a small number of objects presented.

According to the preliminary work, we have formulated the key features of our recommender system:

- domain independence
- focus on calculating the top-k objects (not to give an exact recommendation for a single object)
- simple developer interface
- capability to use different computational methods + possible extensions in the future. New methods may look similar to the code in Figure 1
- simple, developer friendly user preferences mining, important difference against the most of the other recommender system
- easy switch from an existing trivial recommendation - only replacing the SQL query with a few lines of PHP

UPComp requires following assumptions:

- query → response communication style based on the same principles as querying the database. The items are recommended with respect to the current context of the page
- using SQL for specifying satisfactory objects
- UPComp results are lists of top-k objects

A. UPComp architecture

A web shop is typically divided into three components - web pages for visual presentation, PHP for business logic and database for storing data. The UPComp implementation (Figure 2) can be divided into two main parts: preference mining and top-k objects evaluation. UPComp is spread over all three components (web, PHP and database)- on the web part is represented by Javascript that monitors users behaviour and by PHP scripts that create the queries to the UPComp. In the middle is the core PHP script that processes preferences and transforms the data loaded from the database. In Figure 3 is an example how to load UPComp in the beginning of the PHP script. The data about users is stored in the database, too.

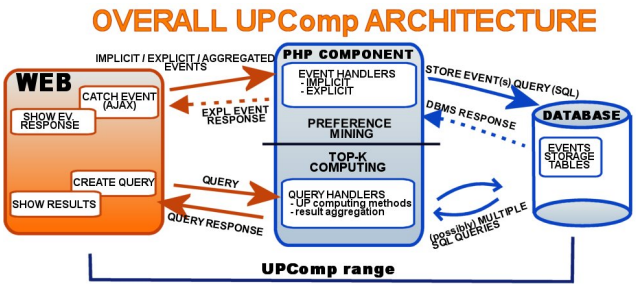

Figure 2. UPComp architecture

```
require_once("UPCompCore.php");
UPCompCore::loadCore();
UPCompCore::loadJavaScripts($object_id);
```

Figure 3. UPComp loading in PHP

We used JavaScript events to catch user actions as the preference collecting (e.g. `<body onload="objectOpen();">`). Each combination of a user and a type of action produces a single user-object preference (e.g. 10 views of a product detail, 50 onmouseover events over a product detail). Then we use AJAX with PHP script on the server side to store the preferences into MySQL database. The set of user actions is extensible, as long as the actions are catch-able via JavaScript. Although the system allows catching both implicit and explicit user actions, we mainly focused on the implicit user actions, because webmasters do not often want to use ratings as they require a change in the user interface. Implicit actions are aggregated to one value for each action type, e.g. 50 onmouseover events over a product detail produce the deep-pageview preference in the degree 0.8.

The observed user actions were:

- *page-view* - opening of the object detail page
- *deep-pageview* - "using" the object detail page (counting the onmouseover events)
- *object-opened-from-list* - opening a detail page of an object shown in the retrieved top-k list
- *buy* - buying of an object

When deploying UPComp in the web shop, the webmaster has to choose some of the above actions as "user preference" and set weight to each action. Weighted average of the

selected actions is then interpreted as the preference.

B. Methods for recommendation

We implemented several basic methods for evaluating top-k objects.

Random method

Random method simply picks k random objects from the set of satisfactory ones. As this is very simple and preferenceless method, all other methods should be better than Random. If they fail to be better, there is no reason to use them.

Object rating

Object rating computes weighted sum of user preference of the given object. Type of the user actions used F and its weight w_F can be specified by the method parameters. Methods result is the k best rated objects.

$$ObjectRating(o) = \sum_{F,U} w_F * F(U, o)$$

Collaborative

Collaborative method first computes m nearest neighbours of the current user. Method uses Pearson's Correlation of user preferences as the user similarity function. Then it computes Object rating restricted to the users from m-NN set. Method's result is the k best rated objects among the users similar to the current one. Bellow is the formula for computing the similarity among users.

$$Sim(U_i, U) = \underset{\forall o}{cor}(Pref(U_i, o), Pref(U, o))$$

where $Pref(U, o)$ is the preference of object o by user U.

Bellow is the formula for estimating the preference of an object using Collaborative filtering. $m - NN$ is the set of m nearest neighbours to the current user.

$$Coll.(o) = \frac{\sum_{U_i \in m-NN} Pref(U_i, o) * Sim(U_i, U)}{\sum_{U_i \in m-NN} Sim(U_i, U)}$$

C. Top-k list building

For the top-k evaluation process, we have adjusted the Combined() method from [11]. It allows using more than one method and using them together.

$$Combined(o) = \frac{w_{RA} * RA(o) + w_{OR} * OR(o) + w_{CF} * CF(o)}{w_{RA} + w_{OR} + w_{CF}}$$

In the formula, w_X corresponds to the weight of method X and $X(o)$ means the preference estimated by method X for object o. Contrary to the original Combined() method, we have allowed web developer to specify (as a part of the UPComp query) which method to use and what are their weights.

Only objects that satisfy given context are taken into account. E.g. objects from a category, or related to the objects in the shop cart.

In experiments, we used only one of the above mentioned method for preference estimation (Random, Object rating, Collaborative) in order to be able to compare the methods among them.

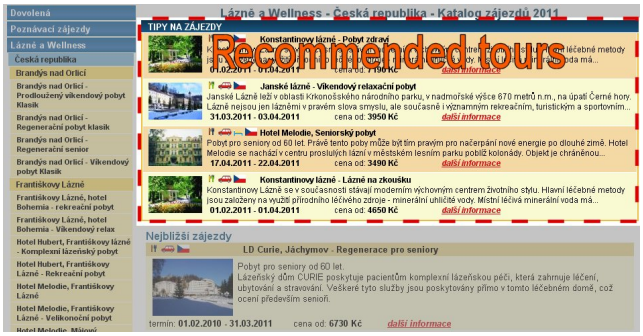

Figure 4. Screenshot of Slantour web page

IV. Experiments

We conducted testing of the UPComp recommender on the real users of the SLAN tour travel agency (www.slantour.cz). The website contains a catalogue of tours divided into several categories and subcategories. Each tour has several attributes such as the name, the destination, the tour type, the departure and the arrival dates etc. During the period of testing, the website showed the following average characteristics:

approx. 530 unique users/day (9000 total)
1800 tour detail views/day
13 bookings/day (218 total)
3min. average visit duration

Tours recommended by the UPComp were then shown on each category and subcategory page instead of the original SQL query (originally a random pick from the objects of the given category was used). The example of such a page is in Figure 4.

The website does not contain any controls to specify explicit user feedback, which is usually used as the correct user preference [7], [12]. Instead of using explicitly given preferences, we assume that booking an object (a tour) represents maximal positive preference of the user.

Also opening object detail from the list of recommended objects represents positive preference, though not as high as the booking. Accordingly, we introduced 3 functions for measuring recommendation methods performance:

1) Open/Shown = #(object detail opened from top-k list of objects) / #(object shown in top-k list of objects)
2) OrderShown/Shown = #(object booked by user whom it was shown in the top-k list) / #(object shown in top-k list)
3) OrderOpen/Shown = #(object booked by user who opened it from the top-k list) / #(object shown in top-k list)

In Table I is the source data collected during the experiment. There are differences between the amounts of users for each method because of variation of traffic in time. Each method was deployed on the server for one week.

Column Shown corresponds to the total number of recommended tours shown. Next column Open is the number

Table I
DATA OBTAINED DURING THE EXPERIMENT

Method	Shown	Open	Order	OrderShown	OrderOpen
Random	40997	836	66	16	6
ObjectRating	45128	1008	94	53	21
Collaborative	40176	754	58	44	14

Table II
RESULTS OF METHODS EVALUATION

Method	Open/ Shown	OrderShown/ Shown	OrderOpen/ Shown
Random	0,020392	0,000390	0,000146
ObjectRating	0,022336	0,001174	0,000465
Collaborative	0,018767	0,001095	0,000348

of tour detail pages opened from the recommendation by clicking on the link. The next column is the number of ordered tours. The fourth column is the number of ordered tours that were shown in the recommendation box. Note that the user does not necessarily click on the recommendation - he/she could have followed other click stream to order the tour. The best possibility is the last column, which is the number of tours that were ordered from the recommendation box. The user clicked on the recommendation and ordered the tour.

Note that OrderOpen is a subset of OrderShown and OrderShown is a subset of Order.

In Table II are processed results for the three methods and the above defined measures.

The test results proved that both Collaborative and Object rating methods are significantly better then the Random method in the most of the observed performance measures. However, the absolute results are yet not very impressive, so further work should focus on the methods parameters adjustment, methods combination or the new methods implementation. Quite surprising was the success of simple Object rating method which overtook both other methods in all measured functions. On the other hand, collaborative method had the best performance in showing the objects that was later booked by the user.

V. CONCLUSION

We have described UPComp - a component for user preference integration into PHP web shops. UPComp should allow web shop owner to add personalized recommendations to their webs, without having to cope with recommendation theory and difficult implementation. Only basic knowledge of PHP and JavaScript is required.

UPComp was also extensively evaluated on a real system with considerable traffic. The test results proved that both Collaborative and Object rating methods are significantly better then the Random method in the most of the observed performance measures. The results are quite promising, but they also show need of future examination and implementation of more complex methods.

UPComp is available at http://code.google.com/p/ksipreferencesoftware as SVN checkout or zipped PHP source codes.

ACKNOWLEDGMENT

The work on this paper was supported by Czech projects SVV-2010-261312, MSM 0021620838 and GACR 202-10-0761 and GACR 202-11-0968

REFERENCES

[1] Netflix dataset, http://www.netflixprize.com.

[2] D. Kelly and J. Teevan, "Implicit feedback for inferring user preference: a bibliography," *SIGIR Forum*, vol. 37, no. 2, pp. 18–28, 2003.

[3] A. Eckhardt, "Prefwork - a framework for user preference learning methods testing," in *In proceedings of ITAT 2009 Information Technologies - Applications and Theory, Slovakia*, P. Vojtas, Ed. CEUR-WS.org, 2009, pp. 7–13.

[4] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. San Francisco: Morgan Kaufmann, 2005.

[5] T. Kliegr, "Uta - nm: Explaining stated preferences with additive non-monotonic utility functions," in *Proceedings of Workshop Preference Learning in ECML/PKDD'09*, September 2009.

[6] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[7] M. Claypool, P. Le, M. Wased, and D. Brown, "Implicit interest indicators," in *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*. New York, NY, USA: ACM, 2001, pp. 33–40.

[8] D. Kelly and N. J. Belkin, "Reading time, scrolling and interaction: exploring implicit sources of user preferences for relevance feedback," in *SIGIR '01*. New York, NY, USA: ACM, 2001, pp. 408–409.

[9] B. Václav, A. Eckhardt, and P. Vojtáš, "Prefshop - a web shop with user preference search capabilities," in *In Proceedings of 2010 International Workshop on Web Information Retrieval Support Systems*, IEEE Computer Society. IEEE Computer Society, 2010, pp. 330–333.

[10] J. B. Schafer, J. A. Konstan, and J. Riedl, "E-commerce recommendation applications," *Data Min. Knowl. Discov.*, vol. 5, pp. 115–153, January 2001.

[11] A. Eckhardt and P. Vojtáš, "Combining various methods of automated user decision and preferences modelling," in *MDAI 2009 - The 6th International Conference on Modeling Decisions for Artificial Intelligence*, 2009, pp. 172–181.

[12] R. White, J. M. Jose, and I. Ruthven, "Comparing explicit and implicit feedback techniques for web retrieval: Trec-10 interactive track report," in *TREC*, 2001.