RECO – EXPERIMENTAL PERSONALIZED RECOMMENDATION FRAMEWORK

Jakub Ševcech
Faculty of Informatics and Information Technologies, Slovak University of Technology
Ilkovičova 3, 842 16 Bratislava, Slovakia, xsevcechj@is.stuba.sk

Michal Kompan
Faculty of Informatics and Information Technologies, Slovak University of Technology
Ilkovičova 3, 842 16 Bratislava, Slovakia, kompan@fiit.stuba.sk

Mária Bieliková
Faculty of Informatics and Information Technologies, Slovak University of Technology
Ilkovičova 3, 842 16 Bratislava, Slovakia, bielik@fiit.stuba.sk

ABSTRACT

The intensive research in the personalized recommendation area results into the need for the automatizing routine processes within the recommenders' design or evaluation. In this paper we propose a novel framework for evaluation and experimentation with recommenders. Proposed approach supports basic recommenders' types – content based and collaborative approaches and allows researchers to add new own recommenders, evaluation metrics or similarity computation algorithms. Moreover, proposed framework allows creating hybrid recommenders by combining various approaches or settings of one approach, while the traffic control or AB testing is supported as well. Proposed solution supports various feedback types, which are collected on the target websites or application. The REST API and jQuery extension allow researchers to integrate generated recommendation into various sites easily ant thus brings significant time savings.

KEYWORDS

Personalized recommendation, experimentation, web service, evaluation.

1. INTRODUCTION

The "on-line revolution" results into huge information overload of every Web' user. Various techniques as personalized search, text summarization or the personalized recommendation are trying to help to solve this problem. The area of recommenders is deeply studied in the last years. Intensive research in the field of recommendation brings researchers to the routine tasks, which have to be performed in order to design, experiment or find optimal setting for specific new approaches and domains. Moreover, in order to compare new approaches there is need to implement reference approaches over various domains and datasets again. This brings us to the problem of platform dependency - various domains often mean various languages or settings.

Two basic techniques for personalized recommendation have been proposed and are used nowadays. The collaborative recommendation focuses on the user to user similarity [11], while the assumption that similar users tend to prefer similar items is used. On the other hand, the content based recommendation uses the item to item similarity [6]. The recommendation is then generated based on the similar items to the previous liked by the user. Often these principles are mixed in order to obtain better results and to solve some problems of each approach (e.g. cold start problem) [2]. Such an ideal combination has to be found in the every application domain. Therefore, the process of such "ideal combination" search is often crucial, time and effort consummating. The collaborative recommendation brings better results in dynamic domains, where there are enough people interacting with the items. On the contrary, content based recommenders are suitable

where there is structured information (metadata) about the content available. Several shortcomings of these approaches are known. The well-known cold start problem refers to the situation where a new item is added to the system and thus cannot be included into the recommendation process while the similarity in the content based recommendation is computed, or minimal set of users rate this item in the collaborative recommendation. This effect is visible not only when a new item is added into the system, but similarly when a new user desires recommendations and his/her user model is empty. In such a situation no user's preferences are known, thus no similar users can be found respectively. Often top visited items, random items, or highest rated items approaches are used to minimize cold start effect. There are plenty of known shortcomings for various approaches as overspecialization, gray sheep effect or the trust of users, which offer a great area for the research.

For this purposes we propose a recommendation framework RECO, which is designed to provide one unified environment for the personalized recommendation experiments. It supports the design of the new recommenders and also provides tools for evaluation of such new approaches. While proposed framework is fully extensible, it allows researchers to add own recommenders, similarity metrics, collect various type of feedback and also create own evaluation metrics. On the other hand, set of predefined algorithm as similarity computation, or basic recommenders are included into the framework, which allow researchers to focus on designing new approaches instead of implementation of the required infrastructure (necessary for recommenders setting, deployment and recommender evaluation and comparison).

The paper is organized as follows. Section 2 provides short overview in the existing solutions for recommendation frameworks. In section 3 we analyze desired features of evaluation framework for recommender systems. Proposed solution is described in section 4. Results of evaluation performed over the proposed framework are presented in section 5.

2. RELATED WORK

Various approaches for the recommendation generation have been proposed, while there are several widely-used evaluation techniques. The basic and most visible difference is the purpose - why such a framework is designed. First, the need for the evaluation and experimentation environments results into the evaluation framework which are often extendable. Second, because of plenty of websites which desire the recommendation platforms. Various web services offering recommendations have been proposed respectively, while these are often not extendable and provide recommendations based on one technique but for various applications (multi-platform solutions).

Recmnd.com is an example of a commerce web service [10], which provides content based recommendations. Service offers simple API, and the extension to the well-known Wordpress content management system. As we can see, described approach offers only one type of the recommendation approach. This is unsuitable not only for the recommendation experimentation, but for various (dynamic) domains is the content-based recommendation unusable [9].

Sugestio.com from University of Ghent offers again content-based recommendation [12]. Interesting feature of Sugestio is the filtering of recommendations based on the actual website context. In the other words, if user browses e-shop in order to buy some CD, approach recommends only CD and no DVD is recommended. Because the service provides only the content-based recommendation, similarly to the previous service, Sugestio is unsuitable for various recommendation tasks in the field of experimentation with recommenders.

Recommenders Benchmark Framework [4] refers to the second approach, which represents the evaluation focused framework. Solution provides simple graphical interface to set various recommender approaches, benchmark metrics or datasets. Framework offers various graphs as the evaluation of compared approaches. The simple interface allows users to easily change settings of compared approaches. The main shortcoming of such an approach is that it does not provide advanced features of evaluation and comparison as the AB testing or there is no possibility to control the users' set in the time of recommendation (recommending only to the subset of real users). Similarly, the support for plugins of routine activities within the recommender approaches is missing (e.g. similarity metrics). Moreover the question of security issues is not discussed.

As we can see, frameworks for personalized recommendation can be divided into two main groups. While the first, commercially oriented, does not supports extending by own approaches of recommendation, the

second group is focused into the evaluation possibilities, while the aspects of providing real recommendation or evaluation within real web application are missing.

3. SETTING UP AND EVALUATION OF RECOMMENDERS

The increase of the recommenders' research results into the need for an evaluation framework, which is suitable for both - online and offline experiments. This is extremely important, while especially in online experiments there is a huge problem to find optimal settings, while it can take a long time to evaluate various settings of proposed approaches, thus such a framework has to support detailed settings adjustment. Moreover, the performance aspects are often crucial, while some approaches may not be designed for the large users' sets. Thus some mechanisms for traffic control and for various approaches evaluation are desired (e.g. AB-testing). The online evaluation brings another complication. In order to display generated recommendation to the users on the target website some generic approach is needed. On the other hand, feedback of users and their activity have to be collected.

Not only the need for online and offline evaluation is important. There are several routine actions which are performed in most of the proposed approaches (similarity computation). Such a framework should relieve users - researches from implementing these routine approaches, and should allow focusing in designing new approaches. Similarly, there are generic solutions for various recommenders' shortcomings. The well-known "cold-start" problem is often solved by introducing random, most visited or most popular items' recommendation approaches [1].

In addition, the trend for involving several simple approaches or algorithms into the one solution, which solve the particular problem based on the voting of these simple algorithms can be observed (e.g. random forest in the information retrieval). We can look on the task of the personalized recommendation in the similar way. The recommender system is constructed from various partial simple solutions (e.g. various similarity metrics). Thus, the possibility of combination of such approaches is desired. Not only from the point of view of recommenders types (hybrid recommenders), but also these particular simple approaches. While the power of combination of multiple recommendation approaches into hybrid recommender systems have been intensively studied and supported [3]. Approaches for assembly of various approaches can speed up the process of experimentation with recommenders, while the researches do not have to create new approaches only to combine existing solutions.

While the extendibility of various algorithms and metric is desired, the question of security and stability is crucial. Various accidental mistakes and errors can overload system resources and stop the process of recommendation, while this is visible especially when online evaluation is performed.

To perform online evaluation and even more when creating system is providing recommendation as a service, easy integration of provided recommendations into the website is required. To evaluate created recommender systems, interface for collecting user feedback is a necessity.

To sum up, several aspects should be fulfilled in order to obtain robust recommendation framework:

- High flexibility allowing to extend framework with own recommender approaches, similarity metrics and benchmark metrics
- Support for collaborative, content based and hybrid recommender approaches
- Various datasets support
- Various feedback type support
- Support for real-time - online evaluation, including intelligent traffic control and detailed settings adjustment
- Easy integration into the website (user activity source)

4. RECO – EXPERIMENTAL FRAMEWORK

Based on the desired requirements described above, we propose RECO - the recommendation framework in form of a web service that provides interface for composition of recommenders from the set of provided components. In addition, the default set of components can be easily extended by own implementations.

To configure a recommender system, built-in components can be used, but users have the possibility to create custom implementations of these components. The basis of the configuration is the module for computation of user-to-user or document-to-document similarities and recommender module. Most commonly used recommendation algorithm types such as content-based and collaborative recommendation are supported. By assembling recommender system from multiple modules and by configuring their parameters, user can easily build a great variety of recommender systems and he can configure one that perfectly fits his particular needs. This basic configuration can be extended by other functionality in form of modules e.g. by the module for evaluation using multiple metrics, by built-in support for AB-testing and traffic control, or by combining multiple recommendation approaches using modules for combining multiple recommender implementation into single list of recommendations (Figure 1). The configuration can be changed during experiments and the resulting change in performance is visualized in changing results of metrics in evaluation modules.

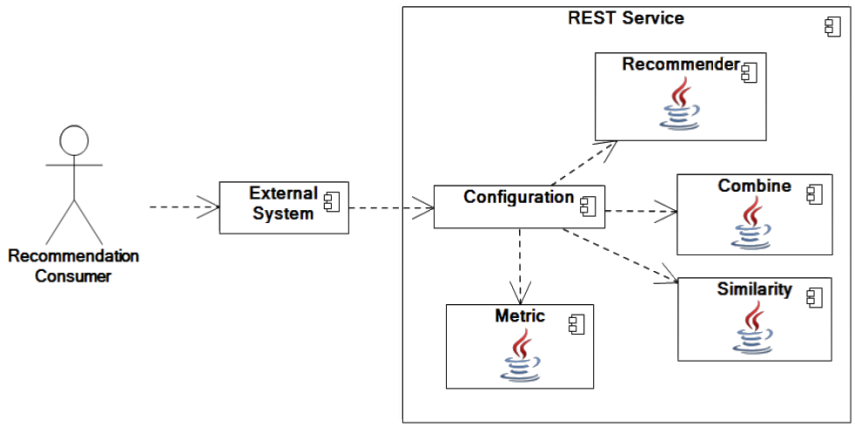

Figure 1. Architecture of proposed recommendation framework.

Moreover, the idea of recommender assembly allows RECO to use a set of predefined approaches for solving standard recommender problems e.g. well known cold-start problem. The set of predefined approaches for naive recommendation (e.g. random items or most visited items) is used in a standard way to bypass the cold-start problem caused by insufficient amount of information necessary to create recommendations.

For the online evaluation are recommendations provided by means of REST web service in JSON or XML format. Easier integration to various web pages is provided using extension of jQuery library. Such an approach allows researchers to use proposed framework in the evaluation process within various applications or web sites (generate recommendations, collect users' feedback), thus RECO can be used as the powerful tool for real-user experiments. The simplicity of integrating recommendation provided by RECO demonstrates this example:

```
<script type="text/javascript">
    jQuery(document).ready(function(){
      jQuery("#reco").reco({
        apikey: 'apikey',
        user: 'user_identifiction',
        config: 'configuration_name'
      });
    });
</script>
```

The basic sequence of steps necessary to create a recommender system starts with the composition of the recommender system from provided set of modules. These basic modules (recommender algorithm implementation, module for similarity calculation) may be enriched by modules for the evaluation of the configuration quality or by modules for the traffic control. After importing initial data about users and documents, the system is ready to generate recommendations. The import of initial data is supported by uploading XML formatted file via web interface or API.

RECO allows researchers to use various types of user's feedback as the user – document pairs, where the type of such pair is defined by the researcher. Provided feedback is taken into account in the process of recommendation generation and in the process of the user-to-user similarity calculation. Proposed framework allows users to implement their own modules in Java language and use them among others provided modules in recommender systems' configurations. The possibility to integrate user-created implementations into such a framework may cause multiple security issues. We resolved these concerns on various levels:

- The simplest one is to catch every error that may be produced in users' implementations.
- We defined timeouts for execution of user code, to prevent unresponsiveness or complete breakdown of recommendation service.
- We defined layer for data access, to prevent user implementations to read or change data that does not belong to that user.
- We allowed access only to essential system resources necessary for various types of implementations to fulfill their purpose.

Thanks to this support for hosting custom implementations in the connection with the possibility to configure recommender from existing modules and the possibility to evaluate success rate of different configurations and approaches, this framework can be used to serve both as a platform for evaluation of recommender algorithms and as a service for easy configuration and integration of recommender systems.

The evaluation of an adaptive system can be viewed through different layers [8], where the adaptive system is broken into its components (layers) and they are evaluated separately. The proposed framework can be used in evaluation of several of these layers. Through the possibility of fast configuration of recommender systems from premade modules, it is possible to find the best approach for recommendation in specific domain (Deciding upon Adaptation). Through the support for AB-testing, it is possible to look for optimal settings of selected recommendation approach (Applying Adaptation Decisions). Finally, the whole system incorporating created recommendations can be evaluated through the possibility to collect and evaluate user feedback (Evaluating Adaptation as a Whole).

5. EVALUATION

The evaluation of proposed recommendation framework is composed of three parts. Comparison of implemented recommendation algorithms shows the possibility to evaluate various approaches in the same domain or on the same dataset. The second part of the evaluation tend to show improvement in the speed of implementation of custom recommender system and the process of obtaining optimal settings of configuration for real use. The last part of performed experiments, evaluates the performance of created framework when used in online evaluation or as a system providing recommendation as a service.

The evaluation of two common recommendation approaches is performed in the domain of an education system ALEF [13], in the course of programming language LISP. 62 students were interacting with more than three hundred different learning objects (explanations, exercises, questions). We evaluated precision and recall of recommendations generated in an offline experiment using collaborative and content-based recommenders and using built-in modules for evaluation. Achieved results are summarized in the Table 1.

Table 1. Comparison of two built-in recommender algorithms in the domain of an education system.

	Precision	Recall	F-measure
Collaborative	35.64%	9.87%	15.46%
Content-based	38.53%	6.58%	11.25%

Basic content and collaborative recommenders were evaluated in an offline experiment, thus users did not see generated recommendations during their work in the system. We compared created recommendations to documents that users visited after generating recommendations. Obtained values of precision and recall in the domain of education system are promising, even when the synthetic evaluation was performed.

In order to investigate the domain dependency of provided default algorithms and modules, we performed similar tests in the news domain. While the domain of e-learning system is smaller and characterized by small amount of students frequently visiting similar documents, the news domain is highly dynamic with

large amount of users and documents. During the offline experiment in the domain or news portal, we created recommendations for 500 visitors of news portal SME.SK. We obtained better results as in another similar study [5], that suggest that proposed framework fits not only for evaluation of different recommendation algorithms, but it is also suitable for real world applications.

Obtained results showed, that proposed evaluation framework is able to provide the environment for designing, optimal settings search and evaluation of recommenders respectively. Moreover, application on various domains and platforms support our hypothesis - easy and wide-range application of our framework.

The second part of the evaluation of proposed framework, we performed, is the experiment to determine the amount of time spared during the implementation of custom recommender system (using RECO) compared to the time necessary to implement similar recommender system from scratch. A domain expert with the experience with recommender systems was asked to implement a simple collaborative recommender using proposed recommendation framework and compare time of implementation and configuration to the time necessary to create similar recommender system on his own (Figure 2). This volunteer never used our framework before.

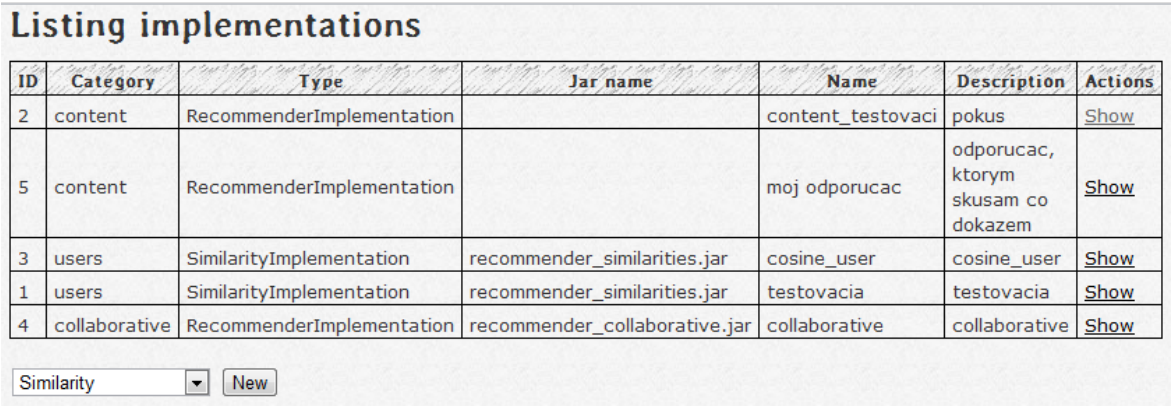

Figure 2. Example of framework environment - list of created recommenders' implementations.

The expert was able to implement new recommender system, insert it into service for recommender hosting, and use it in the new configuration and request recommendations from the system in about three and half hours (Table 2). By his words similar system will take him to implement without this framework at least seven hours.

Proposed estimation covers only a simple collaborative recommender, when more complex recommender is designed (e.g. hybrid recommenders); we expect to be savings increasing. When including the process of evaluation (comparison to other approaches), our approach will bring higher savings respectively.

Table 2. Comparison of time effort for implementation and experimentation of recommenders.

Activity	Proposed framework [h:mm]	Standard approach [h:mm]
Documentation study (first time only)	1:50	0:00
Recommender implementation	1:15	5:15
Data integration	0:25	0:10
Evaluation metrics	0:01	0:45
Experimentation	0:05	1:30
Sum	~ 3:40	~ 7:40

The most time consuming activity, during the experiment, was the study of provided documentation, to find out a way to implement recommender system for the first time. The learning curve of proposed framework is very steep and for the second and the following times the amount of the time necessary to implement custom recommender system will decrease with improving knowledge of proposed framework. During the implementation of custom recommender system the volunteer used multiple built in functions for interaction with data layer and he created new ones as well. For the calculation of user similarity he used one of provided similarity metrics. After implementing the recommender system, he experimented with different settings to find the best configuration for his domain.

The main aspect of proposed framework he appreciated is the reusability of functions and modules in the process of implementation of custom recommender system and reusability of whole parts of recommender systems in the process of the recommender configuration. Using these parts user is able to configure recommender system that fits to his needs in minutes. The fast configuration of recommender systems from existing parts is the most useful when comparing custom implementations to other common algorithms. The evaluation of different system configurations is even more emphasized by the built-in support for AB-testing and evaluation using various metrics.

For the online experiments (real users involved) the performance of proposed approach is important. In order to investigate ability of real-time recommendation generation we experimented with the simulated user traffic. For this purpose we used the Apache JMeter, which allows requesting for the recommendations repeatedly, while the response time is measured (Table 3).

Obtained times are the times which is experiencing the end-user; they do not represent times of computation (waiting time, transfer time and client-server communication). The computation time in this experiment did not exceed 50 milliseconds. Results clearly show, that even when 50 users are requesting for the recommendation continuously, proposed framework is able to provide and to display generated recommendations in no more than 2 seconds. The most time is wasted in the waiting time. Because of this, we included a support for the load balance in the mean of the proxy server, which balances the users' requests between several serves. Obtained results suggest, that our framework can be used in the online evaluation, and with the support of load balancing even as a system providing recommendations as a service.

Table 3. Performance characteristics.

Number of parallel requests	Average response time (ms)	Minimal response time (ms)	Maximum response time (ms)	Standard deviation (ms)
1	147	94	414	51.416
5	199	109	279	27.689
10	385	110	681	54.437
20	856	110	1781	124.417
50	2089	116	5096	250.825

To sum up, provided experiments were focused on the three basic areas:

- Evaluation of the extendibility idea and the domain independency of proposed framework - implementing several recommendation approaches, similarity or evaluation metrics
- Evaluation of the recommender framework concept – domain expert compared time of designing new recommender in proposed framework and from the scratch
- Evaluation of the performance in order to provide online experiments – simulating extreme user traffic

Proposed approach brings positive results in every performed experiment. Promising results were obtained in the comparison to the standard design approach, when the domain expert estimated the difference approximately 4 hours (first time documentation study included), therefore the described framework saved roughly a half of time necessary to create similar recommender system with the standard approach. We can expect more savings, where the design or evaluation process is repeated.

6. CONCLUSIONS

In order to reflect the increasing demand for the evaluation tool for the recommender systems, we proposed a recommender framework, which allows researchers to evaluate their approaches. Proposed framework supports the most important life-cycle phases of designing, optimal settings search and evaluating of new recommenders. Our approach allows researchers to create new approaches, while content and collaborative recommenders are already supported. Moreover, our solution includes set of predefined and extendable routine tasks, which are in the recommendation context often performed (e.g. similarity metrics computation). Similarly, the combination tool, allows creating hybrid recommenders, based on various recommenders or settings.

The second important task after designing new approach is the evaluation. Here, proposed framework provides a set of evaluation metrics. We support both online and offline evaluation, while AB-testing, or the traffic control are included in order to provide one complex tool, which generates recommendation for real users and allows evaluation in realistic conditions.

In the online evaluation, generated recommendations have to be presented to the users. Proposed solution based on web-service provides easy, platform independent approach to generate recommendations for every evaluated website, while minimal effort is needed (jQuery extension).

Thanks to the universal user model, proposed framework can be extended to include specific information about actual user state and use this information in the process of recommendation (e.g. time spent for learning [7], part of the day, when the content is consumed [14]).

Our experiments support the hypothesis that proposed novel approach is suitable to provide the unified, platform independent environment for designing and evaluation of new approaches – designing various approaches within various domains and platforms. Moreover, the performance evaluation shows, that our framework can be used for the online evaluation tasks and it is able to save a lot of time and effort when designing and evaluating recommender approaches (less than a half of time for standard design approach even for first time user).

ACKNOWLEDGEMENT

This work was partially supported by the Scientific Grant Agency of the Ministry of Education of Slovak Republic, grants VG1/0675/11, VG1/0971/11 and by the Slovak Research and Development Agency under the contract No. APVV-0208-10.

REFERENCES

1. Adomavicius G, Tuzhilin A. 2005. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, pp. 734-749.
2. Burke R. 2012. Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, vol 12, pp. 331–370.
3. Burke R. 2007. Hybrid web recommender systems. In: The adaptive web. Springer-Verlag, pp. 377–408.
4. Dayan A, Katz G, Biasdi N, et al. 2011. Recommenders benchmark framework. In: Proceedings of the fifth ACM conference on Recommender systems. ACM, pp. 353–354.
5. Kompan, M., Bieliková, M. 2010. Content-based news recommendation. In EC-Web, volume 61 of Lecture Notes in Business Information Processing. Spriner, pp. 61-72.
6. Lops, P., de Gemmis, M., Semeraro, G. 2010. Content-based Recommender Systems: State of the Art and Trends. In: Recommender Systems Handbook: A Complete Guide for Research Scientists & Practitioners, Springer, pp. 73-105.
7. Michlík, P., Bieliková, M. 2010. Exercises Recommending for Limited Time Learning. Elsevier. In: Procedia Computer Science. ISSN 1877-0509. Vol. 1, Issue 2, pp. 2821-2828.
8. Paramythis, A., Weibelzahl, S., Masthoff, J. 2010. Layered Evaluation of Interactive Adaptive Systems: Framework and Formative Methods. User Modeling and User-Adapted Interaction, 20(5), 383-453.
9. Pazzani MJ, Billsus D. 2007. Content-based recommendation systems. In: The adaptive web, pp. 325–341. Springer.
10. Recmnd - Intelligent Automated Content Reocmmendations. Available at: http://www.recmnd.com/ [20.5.2012]
11. Schafer J, Frankowski D, Herlocker J. 2007. Collaborative filtering recommender systems. The adaptive web, Springer pp. 291-324.
12. Sugestio Recommendation Engine. Available at: https://www.sugestio.com/ [20.5.2012]
13. Šimko, M., Barla, M., & Bieliková, M. 2010. ALEF: A Framework for Adaptive Web-Based Learning 2.0. Proc. of IFIP Advances in Information and Communication Technology. Springer, pp. 367-378.
14. Zelenk, D. 2011. An approach to context aware event reminding. In Information Sciences and Technologies Bulletin of the ACM Slovakia, volume 3, ACM Slovakia, pp. 126-130.