Recommenders Benchmark Framework

Aviram Dayan[1], Guy Katz[1], Karl-Heinz Lüke[3], Lior Rokach[1,2], Bracha Shapira[1,2], Roland Schwaiger[3], Aykan Aydin[3], Radmila Fishel[1,2], and Nassem Biadsy[1,2]

[1]Deutsche Telekom Laboratories at Ben-Gurion University of the Negev, Israel
{avdayan, nassem} @ cs.bgu.ac.il
[2]Department of Information System Engineering, Ben-Gurion University of the Negev, Israel
{guykat, liorrk, bshapira, fishelr} @ bgu.ac.il
[3]Deutsche Telekom AG, Laboratories, Berlin, Germany
{Karl-Heinz.Lueke, Roland.Schwaiger, Aykan.Aydin} @ telekom.de

Abstract: Recommender Systems are software tools and techniques providing suggestions for items to be of use to a user. Recommender systems have proven to be a valuable means for online users to cope with the virtual information overload and have become one of the most powerful and popular tools in electronic commerce. Correspondingly, various techniques for recommendation generation have been proposed during the last decade. In this paper we present a new benchmark framework. It allows researchers or practitioners to quickly try out and compare different recommendation methods on new data sets. Extending the framework is easy thanks to a simple and well-defined Application Programming Interface (API). It contains a plug-in mechanism allowing others to develop their own algorithms and incorporate them in the framework. An interactive graphical user interface is provided for setting new benchmarks, integrate new plug-ins with the framework, setting up configurations and exploring benchmark results.

Keywords: E-commerce, Information Retrieval, Recommender Systems, Machine Learning Software, Data Visualization.

1 Introduction

Recommender systems aim to guide users in a personalized way to interesting objects in a large space of possible options. In their simplest form, personalized recommendations are offered as ranked lists of items. For example, the online book retailer Amazon presents a list of recommended books to help the user decide which book to purchase. The core of each recommender system is a recommendation algorithm that tries to predict what the most suitable items are, based on the user's preferences and constraints. Practitioner who wishes to employ a recommendation system must choose between a set of candidate algorithms based on various criteria such as accuracy and robustness. According to the "no free lunch" theorem [WM97], [HP02], no single recommendation algorithm is superior on all domains. This raises the main dilemma, how to choose the right algorithm for the domain at hand. Moreover, each algorithm can be usually tuned using various controlling parameters. Finding the optimal setting is a tedious task. As stated by [FFSt07], *"Successfully developing and maintaining recommender knowledge bases requires intelligent testing environments that can guarantee recommendations'*

correctness.". Still, there is no comprehensive workbench for recommender systems available for researchers and practitioners alike. However, there is an increasing trend in comparing and examining different recommendation systems. [H02] Propose a framework which explores the user satisfaction from the recommender system, including ease of use by users, presentation and other HCI (Human Computer Interaction) factors, and chooses an algorithm for the recommender systems based on user satisfaction. However, a real application that should test the proposed framework was not developed yet. [CP10] Seek to compare recommender systems by learning the cognitive acceptance of recommendations from users' point of view and how to make a user to trust in the recommender system.

The proposed benchmark framework aims to address the challenges of comparing recommender algorithms, allowing researchers and practitioners to compare and evaluate recommenders' performances over a variety of datasets, using a wide selection of statistical and quality measures. The framework provides a maximum of flexibility when trying recommendation algorithms on new domains. By providing a diverse set of methods that are available through a common interface, the proposed framework makes it easy to compare different solution strategies based on the same evaluation method and to identify the one that is most appropriate for the problem at hand. It is implemented in Java and runs on almost any computing platform.

The benchmark framework may serve two goals:

1. A decision support tool for selecting the best suitable recommender engine for a certain problem and data. Thus, the framework provides a recommendation-system administrator with the tools for evaluating multiple recommenders with different datasets, metrics, and benchmark-policies to assist with her decision;
2. An infrastructure to perform research related to recommender systems.

In addition to a rich set of powerful recommendation algorithms, the framework includes basic statistics and visualization tools all available through an easy to use graphical user interface. To achieve this goal we incorporated several standard Recommender Systems (RS) techniques into the benchmark, so that researchers and practitioners can use it with their own datasets for evaluation. From an experimental point of view, the proposed framework allows other users to develop new algorithms, install new datasets as plug-ins and explore the benchmark results thanks to its built-in plug-in mechanism.

The framework was designed to support the following features: (1) Support of a variety of RS techniques (e.g., collaborative filtering (CF), content based methods (CB), Social, etc.), (2) Support of a variety of evaluation measures, (3) Support of off-line and on-line evaluations protocols, (4) Including built-in statistical procedures and reporting tools, (5) Easy to plug new algorithms, datasets and measures using API, and (6) User-friendly.

In this paper we present a Recommenders Benchmark Framework that will be used as a decision support tool for selecting the best suitable recommender engine for a certain problem and data, and as an infrastructure to perform research related to recommender systems. In section 2 we present a detailed instruction for using the benchmark framework. In section 3 we briefly discuss about the implemented algorithms in the

framework. Section 4 and 5 introduces the metrics and dataset characteristics that we examine in the framework. Section 6 presents the system architecture. Finally, in section 7 we present the different applications of the benchmark.

2 Exploring the Data

The main graphical user interface (before installations of any plug-ins: recommenders/datasets) is shown in Figure 1. All actions are initiated from this screen. Here, one can install new recommenders and datasets, run new benchmarks, explore dataset characteristics and examine various metrics and evaluators, according to the execution matrix that is defined for a specific scenario.

Figure 1: Main Screen Overlook

In the "*Plugin*" tab, the user can choose the plug-in (dataset/recommender) library for installation, and change the configuration of the chosen plug-in.

When installing a new recommender, the user should click on the button next to the *"Plugin library"*, browse and select the recommender plug-in file (*.jar), click open and select the correct file, and finally install the plug-in. A successful installation of a recommender algorithm of SVD (See further explanation regarding the algorithm in chapter 3 – Recommendation Algorithms), e.g. named "*my_SVD*", is shown in Figure 2. One may notice that the new recommender was added on the left side under "*Recommenders*".

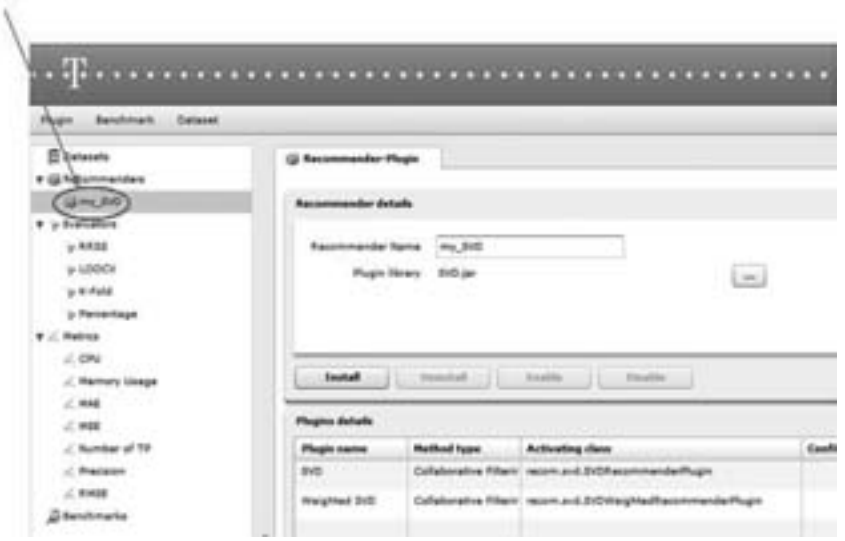

Figure 2: Successful Installation of a New Recommender

When installing a new dataset, the user should type a name for his dataset next to the "Dataset Name" label. Then, choose the relevant dataset type (DB, FILESYSTEM or REMOTE). In this case, the raw data for the "Movielens" dataset is stored in a file, so the type "FILESYSTEM" is selected. Then, the user should browse, locate the dataset plug-in file (*.jar) and press open.

If the installation process has finished successfully, the list of recommenders that the selected file contains will appear at the bottom of the screen. Since "Movielens" dataset contains three plug-ins and its type is FILESYSTEM, we need to associate the raw data file with each plug-in. Click and the button next to each plug-in, locate the file and press open (Figure 3).

If bootstrapping is needed, check the bootstrap box and finally press update (Figure 4). The user will notice that the "Movielens" dataset was added to the dataset group on the left side of the screen (Figure 5).

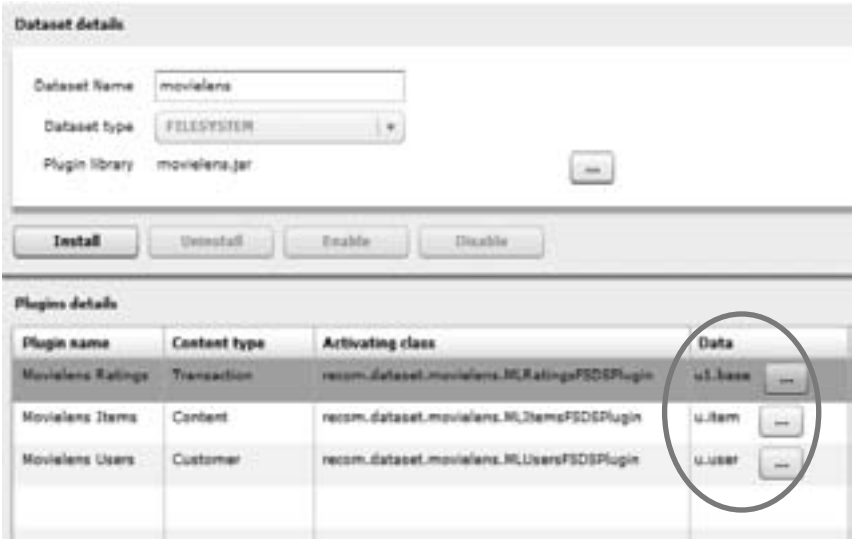

Figure 3: Associating Files with Raw Data

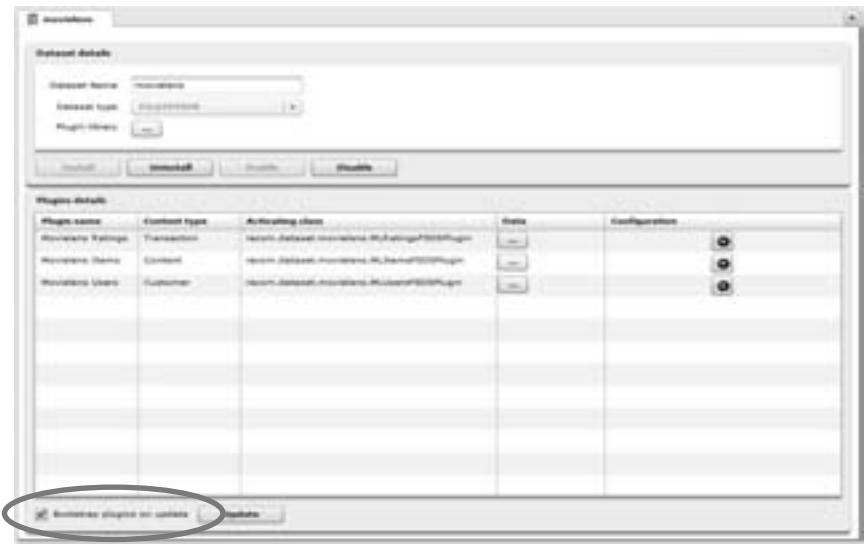

Figure 4: Bootstrapping

Now, the user can create a new benchmark using the "*Benchmark"* tab. The user should check which recommenders will run on which datasets. Also, he should check the metrics he would like to measure and the evaluators (the benchmark framework can deal with different kinds of evaluators such as RRSS, Leave one out, K-fold and Percentage) he would like to use (Figure 6).

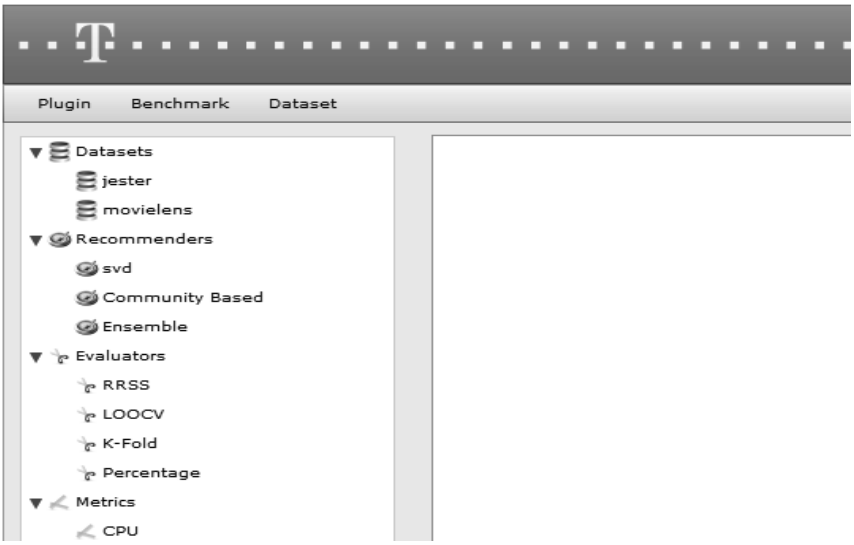

Figure 5: Successful Installation of a New Dataset

Figure 6: Benchmark Configuration

After executing the selected algorithms, the user can explore the benchmark results (for example: MAE, MSE, and RMSE. See further explanation regarding the measures in chapter 4 - Evaluation Metrics) using several views (Figure 7) to deduce conclusions and manipulate the dataset characteristics and get analysis of the datasets. For instance, the results of the SVD algorithm are better than the results of the Ensemble – Configuration algorithm, because the different measures of the errors (MAE, MSE, RMSE) are smaller in SVD.

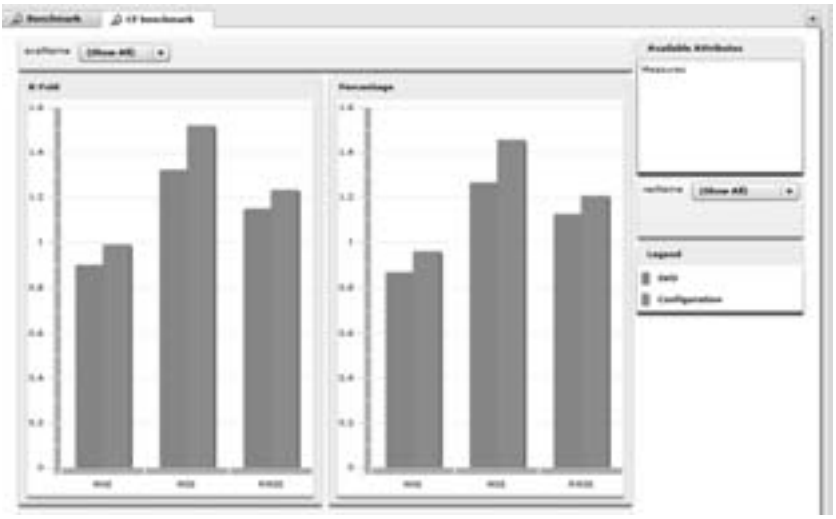

Figure 7: Benchmark Results

3 Recommendation Algorithms

As we mentioned before, the benchmark framework enables to evaluate multiple recommenders with different datasets, metrics and benchmark policies. Thus, the benchmark framework contains a diverse set of recommendation algorithms. In the following subsections we briefly present the algorithms that are provided within the framework. The users can also add their own algorithms.

Nearest Neighbours Collaborative Filtering

Collaborative Filtering (CF) [S00] is considered to be the most successful recommender system to date, and is used in many E-Commerce sites. The basic idea is to predict the user's opinion about the different items and recommend the "best" items. The prediction is based on the user's previous likings and the opinions of other like-minded users. It does not require the creation of explicit profiles, as opposed to techniques that need essential information about the user (like the demographic based [P99] approaches). In addition, CF techniques do not require any domain knowledge and avoid the need for extensive data collection, as opposed to the content based [PB07] and knowledge based [R06] approaches. Since CF systems use a learning algorithm, they are also adaptive i.e. the more data (ratings) the system uses, the more accurate the recommendation are. These characteristics give the CF technique the capability to be implemented in many domains without the need to change the algorithm or the data representation. For example, a movie CF based recommendation system, will also work for book recommendation system.

Community Based

Community Based [G09], [KSS97] is a novel approach which works under the assumption that a specific user has similar preferences as the community that the user belongs to. The intuition behind this approach is that a person will strive to be friends with people who resemble him the most. Thus, users will tend to receive advices from people they trust (i.e. from their friends). Those friends can be defined explicitly by the users or inferred from social networks they are registered to. Researches have shown great accuracy improvement when combining social ties to traditional CF algorithms.

Context Based

Context based recommender systems [AT08] take into consideration additional information about the context of the consumption of an item. For example, in case of a music CD, the time of day or the mood of the user may be used to improve the recommendation. Moreover, different search engines (e.g. Google) return different results during the week vs. the weekend for the same submitted query. This is because they assume that the intents of people vary according to their current situation. For example: A search for a restaurant during the day may be an indication for a business lunch, and the same search at night, may have a more romantic aspect.

Collaborative Filtering using Singular Value Decomposition

Singular Value Decomposition (SVD) is a matrix factorization technique. SVD takes a high dimensional dataset and reduces it to a lower dimensional space. Recent studies [S02] had shown that most of the time, SVD proposes a better result than the traditional CF. However, the main disadvantage of SVD is its computationally very expensive matrix factorization steps. Thus, SVD based recommender algorithm is not suitable for large-scale datasets.

Ensemble

Ensemble methods [P06] have been receiving special attention, mainly due to reported significant performance improvements over single models. The main idea is to use a combination of many models (and not necessarily the best ones) to create an improved composite model.

4 Evaluation Metrics

Our framework supports various evaluation metrics. The user should select the measure that suits the application on hand. Many applications attempt to recommend items to users that they may use. In this case we would like to measure if the system successfully predicts which items the user will choose or consume.

For this purpose we implement well-known measures [H04] such as:

1. *Precision* – The proportion of items in the recommended list which are considered to be relevant to the user (i.e. it represents the probability that a selected item is relevant).
2. *Recall* – defined as the ratio of relevant items presented by the system to total number of relevant items available in the product catalogue.

In certain cases, we wish to predict the rating a user would give to an item (e.g. 1-star through 5-stars in the movie domains). For this purpose we need to measure the accuracy of the system's predicted ratings.

The most popular measures [H04] in this category are:

1. *MAE (Mean Absolute Error)* which measures the average absolute deviation between a predicted rating (P_i) and the user's true rating (R_i), over all the known ratings (N).
2. *RMSE (Root Mean Square Error)* - A variation of MAE that puts more emphasis on large errors by squaring it.

In addition to accuracy metrics we implemented also quality-of-service measures such as CPU and Memory usage.

5 Dataset Characteristics

We believe that the dataset characteristics can have an influence on the decision of choosing a recommendation algorithm. Thus, before comparing the candidate algorithms we should define what properties should the dataset have in order to best model the tasks for which the recommender is being evaluated.

The proposed framework supports various characteristics, such as:

1. *Number of users and items* - Some data sets have more items than users, though most data sets have many more users than items.
2. *Sparsity* - Real E-Commerce sites offer thousands of items (e.g.: books or movies). Users on the other hand rate or purchase only a small number of items (less than 1%). Thus, it may be impossible to find a sufficient number of neighbours in order to produce good recommendations.

6 System Architecture

The benchmark framework architecture (Figure 8) follows the multi-tier paradigm. In addition, to allow maximum flexibility, recommendation algorithms and dataset are implemented as plug-ins to the framework. This plug-ins can be added and removed to and from the framework in runtime which allows development and integration with no dependency of the framework itself. Another key feature is the complete decoupling of the recommendation algorithms plug-ins and the dataset plug-ins implementation. An

algorithm can use the dataset as input as long as the data contained in this dataset is appropriate to the algorithm (for example, community-based recommendation algorithm needs community data).
The framework contains four core components:

1. The recommendations framework – contains the APIs' for recommender algorithms and data-sets.
2. The benchmark framework – contains the APIs' for benchmarking. In addition contains the notions of Evaluators and Metrics that are integral part of the benchmark.
3. The plug-in manager – contains the APIs' for wrapping recommenders and data-sets as plug-ins.
4. Recommenders and data-sets plug-ins – the plug-ins are developed separately from the framework and are following the plug-in APIs'. They are plugged-in to the system using the plug-in manager. Recommender and dataset plug-ins are completely decoupled. The benchmark framework allows using a recommender with a dataset according to the used recommendation method and the content-type of the data in the dataset. For example: collaborative filtering recommenders can be benchmarked only with datasets that contain rating data.

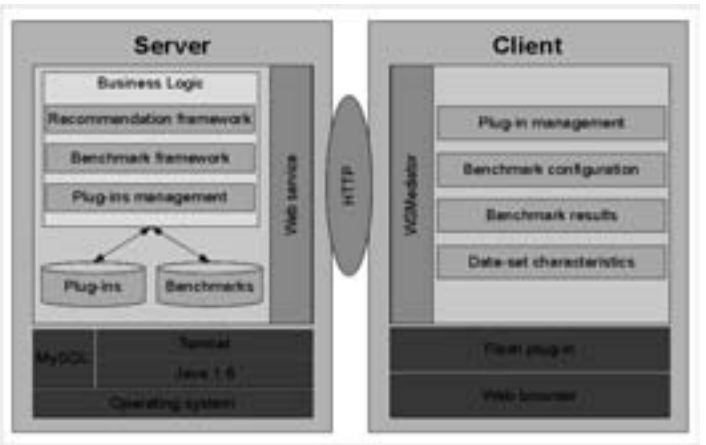

Figure 8: System Architecture

7 Applications

The benchmark framework was developed for recommender systems and its purpose is to take multiple recommending algorithms and different datasets, and benchmark them using specific evaluators, metrics and data characteristics. The benchmark framework allows other programmers to develop new recommenders and datasets according to a well defined API. Datasets and recommenders are concepts that are completely decoupled; that means that a recommender can be tested with different datasets and vice versa. It enables the users of the benchmark to test different algorithms on one (or

several) datasets and select the most suitable recommender algorithm for their needs. Also, the benchmark is used as an integral part of a research that is main objective is improving the prediction accuracy of CF-based recommendations systems by applying different ensemble methods in order to construct a set of combined predictors.

8 Summary

The benchmark framework that we developed presents a novel approach towards recommender systems and has some principal advantages. The most important, it is recommendation system oriented. Existing predictive analytics software such as R Project, Weka and RapidMiner do not support popular recommendation algorithms, unlike the benchmark framework, that in addition to its comparison features, contains algorithms that were developed specifically for the recommendation domain. Another important advantage is that using the benchmark's plug-in mechanism, allows others to develop new recommenders and datasets, plug them into the framework, run benchmark and explore the results. Furthermore, it is a web application which means it can be run and accessed anywhere, without any need to download anything or other any further requirements.

9 Future Work

In the future, we predict that the benchmark framework will be used as a uniformed platform for testing recommender algorithms. We are currently working on additional measures, dataset characteristics, algorithms and datasets that will be added to the benchmark.

Bibliography

[S00] Sarwar, B. et. al.: Analysis of Recommendation Algorithms for E-Commerce. Electronic Commerce Proceedings of the 2nd ACM conference on Electronic commerce, 158 – 167, 2000.

[PB07] Pazzani, M. J.; Billsus, D.: Content-Based Recommendation Systems. The AdaptiveWeb: Methods and Strategies of Web Personalization, Lecture Notes in Computer Science, Vol. 4321. Springer-Verlag , 2007.

[P99] Pazzani, M. J.: A Framework for Collaborative, Content-Based and Demographic Filtering. Artificial Intelligence Review, 13 (5/6), 393-408, 1999.

[R06] Ricci, F.; Cavada, D.; Mirzadeh, N.; Venturini A.: Case-based Travel Recommendations. Destination Recommendation Systems: Behavioural Foundations and Applications, pages 67–93, CABI, 2006.

[AT08] Adomavicius, G.; Tuzhilin, A.: Context-Aware Recommender Systems, 2nd ACM International Conference on Recommender Systems, 2008.

[S02] Sarwar, B. et. al.: Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems. 5th International Conference on Computer and Information Technology (ICCIT), 2002.

[FFSt07] Felfernig, A.; Friedrich, G.; Schmidt-Thieme L.: Guest Editors' Introduction: Recommender Systems. IEEE Intelligent Systems: IEEE Computer Society, vol. 22, pp. 18-21, 2007.

[HP02] Ho, Y. C.; Pepyne, D. L.: Simple Explanation of the No-Free-Lunch Theorem and Its Implications1. Journal of Optimization Theory and Applications, vol. 115, No. 3, pp. 549–570, December 2002.

[WM97] Wolpert, D. H.; Macready, W. G.: No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation: vol. 1, Issue 1, pp. 67-82, April 1997.

[P06] Polikarm, R.: Ensemble Based Systems in Decision Making. IEEE Systems Magazine, Issue 3, 2006, 21-45.

[H04] Herlocker, J. L. et. al.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information System (TOIS), vol. 22 Issue1, January 2004.

[KSS97] Kautz, H.; Selman, B.; Shah, M.: Referral Web: combining social networks and collaborative filtering, Communications of the ACM, v.40 n.3, p.63-65, March 1997.

[G09] Guy, I. et. al.: Personalized recommendation of social software items based on social relations. Third ACM Conference on Recommender Systems (RecSys'09), pages 53-60, New York, NY, USA, 2009.

[CP10] Chen L.; Pu, P.: User Evaluation Framework of Recommender Systems. Workshop SRS'10, February 7, 2010, Hong Kong, China.

[H02] Hayes, C. et. al.: An online evaluation framework for recommender systems. In Workshop on Personalization and Recommendation in E-Commerce (Malaga, Spain, May 2002), Springer-Verlag.