Flash reactivity: adaptive models in recommender systems

Julien Gaillard, Marc El Bèze, Eitan Altman, Emmanuel Ethis

Flash reactivity: adaptive models in recommender systems

J. Gaillard[1], **M. El-Beze**[1], **E. Altman**[2] **and E. Ethis**[3]
[1] SFR Agorantic, University of Avignon, France
[2] Maestro, INRIA Sophia-Antipolis, France
[3] Norbert Elias Center, University of Avignon, France

Abstract— *Recommendation systems take advantage of products and users information in order to propose items to targeted consumers. Collaborative recommendation systems, content-based recommendation systems and a few hybrid systems have been developed. We propose a dynamic and adaptive framework to overcome the usual issues of nowadays systems. We present a method based on adaptation in time in order to provide recommendations in phase with the present instant. The system includes a dynamic adaptation to enhance the accuracy of rating predictions by applying a new similarity measure. We did several experiments on films data from Vodkaster, showing that systems incorporating dynamic adaptation improve significantly the quality of recommendations compared to static ones.*

Keywords: recommender systems, adaptive model, instantaneity, similarity measure, evaluation protocol

1. Introduction

This work has been carried out in partnership with the website Vodkaster[1], often considered as the Cinema social network in France. Users post *micro-reviews* (MR) to express their opinion on a movie and rate it. These reviews should not exceed 140 characters like on Twitter. More details on the corpus are given in section 5.

Though for the moment, we use only the users ratings, note that as future work, we intend to include the semantic level derivable from the reviews. In this perspective we will take into account for each pair movie-user both a rating and the textual argument associated to it. However, this is not the only reason we are working on the dataset provided by Vodkaster. In fact, we are interested in building a recommendation system relying on opinions expressed by a cinephilic community, exactly what Vodkaster offers.

Nowadays, classical Recommender Systems (RS) are able to suggest appropriate items to users from a large catalog of products. Those systems are individually adapted by using a profile for each user, itself made upon an analysis of past ratings. The most common techniques used in RS are Content-Based Filtering (CBF) and Collaborative Filtering (CF). Hybrid systems combine collaborative and content-based techniques, thus taking advantages from both methods.

[1]www.vodkaster.com

However, whatever the technique used, one of the biggest issues remains reactivity [2]. The last decade has shown a historical change in the way we purchase and/or consume products. Nowadays, society demands having everything instantaneously. The needs have to be satisfied and change more and more quickly. This is mostly due to the growth of the Internet use and it is Internet itself that allows us to meet this legitimate expectation. It is therefore necessary to design RS adapting themselves instantaneously [5].

In this paper, we propose a new method that makes the system very fitted to dynamic behavior, reactivity and swift adaptivity. From this point of view we have to avoid the complete recalculation of models at each new incoming data or update, but only update a few relevant variables. In this way the system will be able to react promptly on the fly.

In the next section, we present the state of the art in recommendation systems and introduce our improvements. Then, we present our approach and define the methods corresponding to it. We describe the evaluation protocol and perform experiments. Finally we report our results and compare them to a baseline.

2. Related work and choice of a baseline

In this section, we present the methods used in most of classical recommender systems. [6] CF system uses logs of users, mainly user ratings on items, with dates. In these systems, the following hypothesis is made : if user a and user b rate n items similarly, they will rate other items in the same way [4]. This technique has many well-known issues such as the cold start problem, i.e when a new element (item or user) is created, it is impossible to make a recommendation, due to the absence of rating data. Other limitations of recommendation systems are the data sparsity problem, the scalability problem, overspecialization and domain-dependency.

In CBF systems it is supposed that users are independent [3]. Hence for a given user, recommendations will be made by taking into account items he previously liked. Metadata are compared to explicit or implicit user preferences. Unlike CBF, CF systems do not need a description of items to be recommended, a simple identifier number is enough.

2.1 Similarity measures

The similarity measures between two entities (items or users) is a cornerstone of systems based on neighborhood methods and one well worth noting [11]. The Pearson (eq.1) correlation coefficient was one of the first similarity measure proposed by Resnick [1]. There exist other similarity measures such as Jaccard [12] [14], Cosine, similarity based on the Euclidian distance, etc.

Let T_i be the set of users who have rated item i, S_u the set of items rated by u, $r_{u,i}$ the rating of user u for item i and $\overline{r_x}$ the mean of x (user or item).

$$Pearson(i,j) = \frac{\sum_{u \in T_i \cap T_j}(r_{u,i} - \overline{r_i})(r_{u,j} - \overline{r_j})}{\sqrt{\sum_{u \in T_i \cap T_j}(r_{u,i} - \overline{r_i})^2 \sum_{u \in T_i \cap T_j}(r_{u,j} - \overline{r_j})^2}} \quad (1)$$

$$Cosine(i,j) = \frac{\sum_{u \in T_i \cap T_j} r_{u,i} \times r_{u,j}}{\sqrt{\sum_{u \in T_i \cap T_j} r_{u,i}^2 \sum_{u \in T_i \cap T_j} r_{u,j}^2}} \quad (2)$$

We choose the Pearson similarity measure as a baseline.

2.2 Rating prediction

Consider a given user u and a given item i. We assume the pair (u,i) is unique since generally, social networks may not allow one user to give multiple ratings for one item, and this rule is applied by Vodkaster. We define two rating prediction methods : one *user oriented* and the other *item oriented*. In the following, *Sim* will denote a similarity function.

$$\begin{aligned} rating(u,i) &= \frac{\sum_{v \in T_i} Sim(u,v) \times r_{v,i}}{\sum_{v \in T_i} |Sim(u,v)|} \quad (3) \\ rating(i,u) &= \frac{\sum_{j \in S_u} Sim(i,j) \times r_{u,j}}{\sum_{j \in S_u} |Sim(i,j)|} \end{aligned}$$

Finally, we do a linear combination of $rating(u,i)$ and $rating(i,u)$.

$$\hat{r}_{u,i} = \beta \times rating(u,i) + (1-\beta) \times rating(i,u) \quad (4)$$

We add two components in order to balance and correct the prediction by taking into account $\overline{r_u}$ and $\overline{r_i}$. We combine these two averages ratings with two coefficients, m_i for $\overline{r_i}$ and m_u for $\overline{r_u}$., with $m_i + m_u = 1$. We call this new rating function weighted rating ($\hat{r}w$)

$$\hat{r}W_{u,i} = \gamma \hat{r}_{u,i} + (1-\gamma)(m_i \overline{r_i} + m_u \overline{r_u}) \quad (5)$$

In the case where $\hat{r}_{u,i}$ is not computable, we apply a backing off like strategy relying on $m_i\overline{r_i} + m_u\overline{r_u}$. It is possible that i (or u) has no ratings, hence $\overline{r_i}$ (or $\overline{r_u}$) does not exist. In the absence of either one of these two averages, we only rely on the other one.

3. Methods

In this section we present the methods we use and propose some of the improvements we have implemented in our system.

3.1 Distance of Manhattan

To derive a similarity measure from the distance of Manhattan, also known as the taxicab distance [13], we take the complement to one. Hence, the more ratings are close, the more the similarity tends to 1, and therefore the more the users or items are considered as alike. We normalize the results by dividing the sum by the maximum difference between two ratings ($MaxD$) times the number of elements in the intersection. In the remainder, this similarity function is used for both users and items. In the following k is either an item or user, x, y is a pair of items or users depending on the case.

$$Manhattan(x,y) = 1 - \frac{\sum_{k \in T_x \cap T_y} |r_{k,x} - r_{k,y}|}{|T_x \cap T_y| MaxD} \quad (6)$$

We add another component that takes into account the difference of the means $\overline{r_x} - \overline{r_y}$, with a coefficient F. This new component can be useful in some cases. For instance, let us look at the similarity between user a and user b. User a has a certain tendency to be very severe on items he rates. On the contrary, b is more indulgent. This difference of behaviors between a and b is somehow related to the difference of average ratings. In the end, this heterogeneity is taken into account in the similarity by a coefficient F.

$$Manhattan2(x,y) = 1 - \frac{\sum_{k \in T_x \cap T_y} |r_{k,x} - r_{k,y}| + F|\overline{r_x} - \overline{r_y}|}{(|T_x \cap T_y| + F) MaxD} \quad (7)$$

We use a coefficient proportional to the cardinality of the intersection $T_x \cap T_y$ as a confidence measure. Therefore we are giving more weight to items sharing a greater number of users. We call this similarity measure the Manhattan Weighted Corrected similarity (MWC).

$$MWC(x,y) = Manhattan2(x,y) \times \left(1 - \frac{1}{|T_x \cap T_y|^\alpha}\right) \quad (8)$$

3.2 Metadata

Metadata allows our system to overcome the cold start problem whenever a new item is added to the database and thus has not been yet rated. It is therefore impossible to compute the similarity with another item based on ratings of common users. The use of metadata can fix this problem, since we can now compare two items according to their metadata. In our case, we are dealing with movies. Metadata are for instance the director's name, main actors or

genre. Such data can be found on IMDB[2] (Internet Movie DataBase) and can be downloaded.

3.3 Dynamic adaptation with Manhattan

In this section we present the process used to attain a dynamic adaptation along time. The key idea follows the simple principle that each update or new pair (u, i) has to be taken into account instantaneously by the system. It cannot be delayed for some days since everything changes so fast. It could already be too late and thus mislead the following recommendations, especially the ones based on a small number of ratings. One log of rating can make the difference.

The similarity measure named Manhattan Weighted Corrected (eq. 8) is designed to allow us to update items-to-items and users-to-users similarities in a very efficient way. Indeed, unlike $Pearson$ or $Cosine$, this method does not lead to a complete re-calculation of the pre-calculated models.

For instance, we look at the similarity between item a and item b. User $sarah$ has previously rated item a and now rates item b, that she has never rated before. $\overline{r}_{sarah}$ and $\overline{r_b}$ are updated very easily. Indeed, if we look at the details of the MWC function, we clearly see that in the numerator sum :

$$\sum_{u \in T_a \cap T_b} |r_{u,a} - r_{u,b}|$$

since $sarah$ is now belonging to $T_a \cap T_b$, we just need to add $|r_{sarah,a} - r_{sarah,b}|$ to the pre-calculated sum. We also have to increment the cardinality of the intersection by one. And we are done. With only four simple additions, we have updated the database. The same holds for items.

Taking advantage of this property, we ran the updating algorithm on the training corpus. The results are outstanding. The complexity has been reduced from $o(n^2)$ to $o(n)$ (square to linear). If we consider the whole set of updates, we reduced from $o(n^3)$ to $o(n)$.

Suppose now a new item is created and consequently has not been rated yet. It is thus impossible to predict a rating for this item, unless we take into account metadata. We define a new similarity measure based on metadata, namely Metadata Based similarity (MBS). Let M_x be the set of metadata of x (user or item). We then have the following :

$$MBS(x, y) = \frac{|M_x \cap M_y|}{|M_x \cup M_y|} \tag{9}$$

This ratio is also known as the Jaccard similarity coefficient, used to measure similarity between two sets.

In case the classical similarities cannot be calculated, MBS allows the system to make a prediction and therefore it increases the coverage. In other cases where an item has already been rated, the use of MBS enhances the prediction precision (see results).

[2]www.imdb.com

4. Evaluation criteria

In this section we present our evaluation protocol. Since we cannot make online experiments with real users, we are not able to measure the impact of our recommendations, that should lead in the best case to an act of consumption with a good feedback (good rating). However, the key point in a recommender system is the rating prediction accuracy [10]. Hence, one could say that a recommender system could be evaluated on his ability to predict the rating of a given user u for an item i. From this perspective, we can test the system on predicting a rating for which we know the actual real rating. In other words, we compare $r_{u,i}$ and $\hat{r}_{u,i}$.

If a prediction $\hat{r}_{u,i}$ is less than a certain threshold r_{min}, we assume item i is not recommendable for user u. Therefore, any prediction below this threshold is not taken into account in the evaluation. On the contrary, when $\hat{r}_{u,i} > r_{min}$ and $r_{u,i} > r_{min}$, we consider our recommendation as successful.

Ideally we should be able to measure the quality and the performance of two functionalities expected from a recommender system. The first one is the ability to promote an item appreciated by a small number of amateurs to a maximum number of users themselves likely to appreciate it. The second one consists in recommending to an user the maximum number of items he is likely to appreciate. In this paper, we did a hybrid evaluation. We evaluate the system globally on each pair (u, i). Hence we are in between the promotion of items for users and the promotion of users for items.

4.1 Root Mean Squared Error

This measure namely *Root Mean Square Error* is often used to evaluate different methods applied in RS. It also has become popular with the Netflix Challenge [7]. Let R be the set of the predicted ratings, the RMSE is defined as follows :

$$RMSE = \sqrt{\frac{1}{|R|} \sum_{(u,i,r) \in R} (\hat{r}_{u,i} - r_{u,i})^2} \tag{10}$$

It is widely assumed that reducing the RMSE amounts to increasing the relevance and precision of the recommendations.

4.2 Mean Absolute Error

We also use the Mean Absolute Error (MAE) to evaluate how close our predictions are to the real ratings. The mean absolute error is given by :

$$MAE = \frac{1}{|R|} \sum_{(u,i,r) \in R} |\hat{r}_{u,i} - r_{u,i}| \quad (11)$$

5. Experiments

5.1 Corpus

The corpus contains over 50,000 MR. For each MR, we have : *an unique ID, the author's name, the text, the rating (0.5 up to 5), the date of the post, the film title, its country*, and its *release year*. We have split the corpus into three sub-corpuses : training, development and test. The date makes the corpus chronologically sortable. It is very important to note that in our experiments, we take into account the date since we work on dynamic adaptation. The chronological order, from old to recent is : training, development, test.

	Training	Development	Training+Development	Test
Size	57631	9999	68502	9999
Films	8680	3298	9428	3951
Users	1824	730	2080	737

	Development	Test
User unseen	2858	3274
Film unseen	2452	1895
User and Film unseen	446	375
User unseen different	244	235
Film unseen different	675	849
User and Film unseen different	442	375

Table 1: Statistics on the corpus

The second part of Table 1 shows how important the adaptation is. Indeed, the number of unseen users and unseen films is quite large (first two rows). Note that these users and films are re-appearing at least twice in the development (or test) corpus. In this case, adaptation makes even more sense. This is not the case when an unseen user or film appears only once (last 3 rows of the table). The worst case is when we have a two sided cold start, from the user's side and the film's side (user and film unseen).

5.2 Evaluating

To evaluate the system, we first create our item-users database from the training sub-corpus. (8680 items, 1824 users) Then, we go through the development sub-corpus. Each element in it is a pair user-item (u, i), containing the rating of user u on item i. In the remainder, an item (or user) that has been rated by users (resp. rated items) is considered has *seen*. Otherwise it is *unseen*. For each of the following cases, the system reacts differently.

a) u and i are seen: This is the simplest case in which we have rating data for i and u. We just use the weighted rating method (eq. 5).

b) u seen, i unseen: Item i is a new item and has not been rated yet. But u is seen. This means u has already rated at least one item. We can thus use the user oriented rating function with the metadata based similarity function. This rating function is based on similarities between i and the items already rated by u. It is hence not necessary for i to be seen (*i.e* rated).

c) u unseen, i seen: User u is new and has not rated any item yet. But i is seen. This means i has already been rated once. In that case, we use the item oriented rating function again with the metadata based similarity function. This rating function is based on similarities between u and the users that have already rated i. It is hence not necessary for u to be seen.

d) u unseen, i unseen: In this extreme case, we generally have access only to the metadata for items nor for users. It is then difficult to make a prediction.

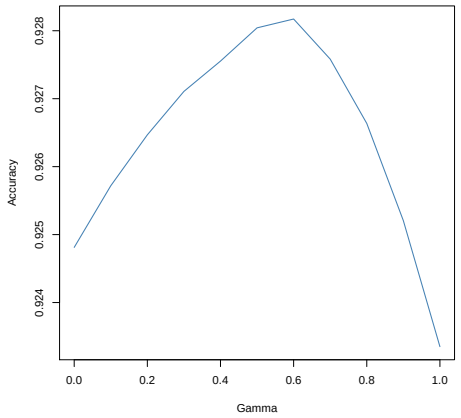

Fig. 1: Accuracy in function of γ at constant coverage (2200 predictions) on the development corpus. Optimal value is 0.6

We did several tests on the development corpus in order to determine the optimal γ. Recall that this coefficient weights the average in the weighted rating prediction formula (eq. 5). The results of these tests are shown on figure 1.

Figure 2 shows the effect of adaptation. We recall that the development corpus (and others too) is sorted chronologically, from older to newer ratings. Therefore in this graph, the closer we are to time zero, the closer we are to the training corpus, time speaking. The first observation we can make is the global tendency for accuracy to decrease as time goes by, that is as we go away from the training corpus in time. However, we also observe that the adaptation slows down this tendency and in some time ranges even reverse

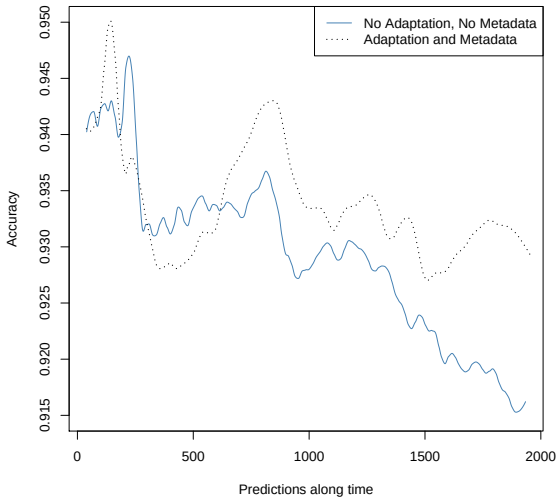

Fig. 2: Evolution in time at constant coverage (2100 predictions) with adaptation and metadata, and none of them (development corpus)

the trend (between 1500 and 1800).

5.3 Results

To be able to compare different methods, we take a constant coverage (2200 predictions).

5.3.1 Pearson

We present here the results obtained with our baseline.

Corpus	Method	Score	RMSE	MAE
Development	No adaptation	84.36	0.93	0.73
Test	No adaptation	86.77	0.94	0.71

Table 2: Results with Pearson

We observe that the results are better on the test corpus than on the development corpus. This can be explained by the fact that the training corpus used for predicting the test is larger than the one used for predicting the development.

5.3.2 Manhattan Weighed Corrected

We present here the results with our method.

	Score	RMSE	MAE
No adaptation No metadata	91.09	0.90	0.70
Adaptation only	92.76	0.88	0.69
Metadata only	91.50	0.89	0.70
Metadata and Adaptation	92.93	0.87	0.68

Table 3: Results with MWC on the development corpus

Table III and Table IV show the effect of adaptation and metadata, together and separately. We can see that using

	Score	RMSE	MAE
No adaptation No metadata	89.6	0.98	0.75
Adaptation only	90.6	0.94	0.72
Metadata only	89.3	0.99	0.75
Metadata and Adaptation	90.7	0.94	0.73

Table 4: Results with MWC on the test corpus

metadata only is not as useful as expected. However, the adaptation combined with metadata allows a gain in accuracy greater than 1.5%.

5.4 Analysis

We present some examples of good recommendations and mistakes too.

- *The Hobbit : An Unexpected Journey* (2011, USA) recommended to user *Zarai*.

	#ratings in training	#ratings in test	Average
The Hobbit	0	7	3.76
Zarai	0	87	4.4

Predicted rating	5	Real rating	5

We are able to recommend a film that has not been rated yet to an user unseen in the training. Thanks to adaptation, this becomes possible and the prediction is a very good one.

- *Le Père Noël est une ordure* (1982, France) recommended to user *Fernand*.

	#ratings in training	#ratings in test	Average
Le Père Noël...	14	2	4.1
Fernand	0	45	4.2

Predicted rating	5	Real rating	5

We recommend this film seen 14 times in the training corpus to an user named Fernand unseen in the training corpus (does not include any movie rated by him). However, at the moment we recommend this movie, we can take into account all of his ratings found in the test so far. This example is a good proof of the interest of a short-term adaptation.

- *The Nightmare Before Christmas 3D* (2006, USA) recommended to user *Bart*.

	#ratings in training	#ratings in test	Average
The Nightmare...	2 ($r = 1, 3$)	2 ($r = 4.5, 5$)	3.4
Bart	0	31	4.68

Predicted rating	4.7	Real rating	2.5

This is the first error observed when the predictions values are sorted in decreasing order. The user has rated this movie 2.5. However, it has been well rated in the test and it's probably the main reason we have recommended it. Before the recommendation has been

done, Bart's average rating was 4.68, which is very high. In this case, adaptation misleads the system.

5.4.1 Accuracy in function of coverage

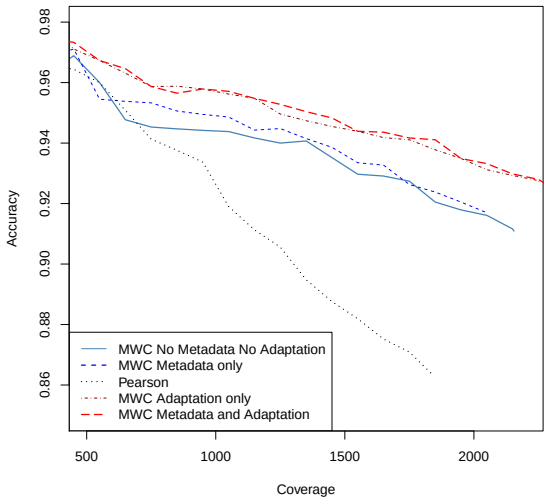

Fig. 3: Accuracy for Pearson and MWC in function of the coverage on the development corpus

Figure 3 depicts the difference of accuracy between a classical Pearson similarity measure and the Manhattan Weighted Corrected similarity at several levels of coverage. We can see that in the case of the MWC method, the accuracy stays very high (over 92.5%) even for large coverages (over 2000). On the contrary, the Pearson similarity leads to a very fast decrease in accuracy. Indeed, the accuracy is only 85.1% at a coverage of 2000, whereas the MWC gives 93.3%. Our method outperforms the baseline.

5.4.2 Robustness

As we can see in Fig. 4, the results obtained on the test and development corpus can be considered as similar since the confidence interval has been estimated to be 0.011 (i.e 1.1%).
Fig. 4 also depicts the fact that the knee of the curve, for both development and test experiments is around a prediction value of 3.75 (vertical line). Below this threshold, the accuracy level drops very quickly. However, predictions above the same threshold can be considered as trustworthy (see Table V).

	Threshold	Coverage	Accuracy	RMSE	MAE
Development	3.75	1537	94.5	0.84	0.65
Test	3.75	1455	92.4	0.92	0.69

Table 5: Accuracy at a preset confidence level

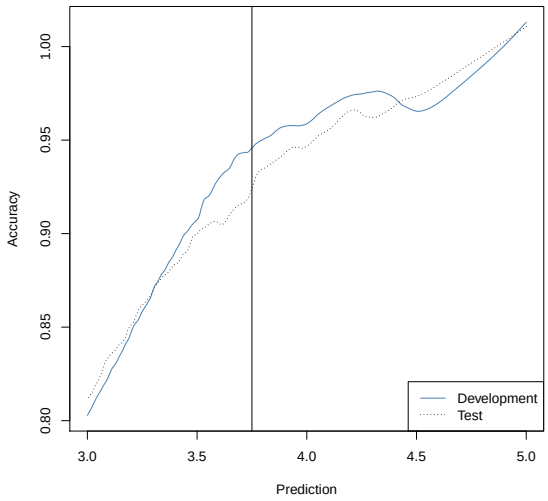

Fig. 4: Accuracy in function of the predicted ratings on development and test corpora

6. Conclusions and perspectives

In order to obtain a flash reactivity, we have proposed a new similarity measure based on the distance of Manhattan. This new measure named *Manhattan Weighted Corrected* similarity leads to a significant decrease in complexity and allows an instantaneous adaptation. Hence we are able to update the parameters of the recommender system step by step, whenever a new rating occurs. Thanks to this new method, we obtained results outperforming the one's obtained with a classical Pearson. Moreover, by applying the same algorithm during the training phase, we have dramatically reduced its complexity. We have also shown that this method allows us to perform a detailed analysis of the prediction errors (bad recommendations). This analysis could be used for future improvements of our system.
We are currently working on adding text content. The idea is to extract information about users movies taste (horror film, thriller, this actor, interested in soundtrack...) and films characteristics (good scenario, too long, great special effects...) as it is expressed in natural language in micro-reviews. Therefore, it will be possible to take into account the aesthetics tastes of users and not only their ratings.
We are also developing a new adaption method that adapts itself according to the users taste at a given moment in time. We will check whether it is worth or not to take into account the entire rating history. This is coherent with our conception of recommendation. We believe that nowadays recommender systems have to be instantaneous, giving the right recommendation at the right time, learning from their mistakes, and adapting the model not to repeat again and again the same errors.

Acknowledgment

The authors would like to thank Vodkaster for providing the data.

References

[1] P Resnick and R. Varian Hal, Recommender systems (introduction to special section). Communications of the ACM 40, 1997

[2] G. Adomavicius and A. Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State of the Art and Possible Extensions, IEEE Trans. Knowl. Data Eng, 17 (6), 2005, pp. 734-749.

[3] B. Mehta, T. Hofmann, and W. Nejdl. Robust collaborative filtering. In RecSys, 2007

[4] M. Deshpande and G. Karypis. Item based top-N recommendation algorithms. ACM Trans. Inf. Syst., 2004.

[5] N. Lathia. Evaluating Collaborative Filtering Over Time. PhD thesis, University College London, 2010.

[6] R. Burke, Hybrid Web Recommender Systems. The Adaptive Web, 2007, pp. 377-408

[7] R. Bell, Y. Koren and C. Volinsky. The BellKor 2008 Solution to the Netflix Prize The Netflix Prize, 2007.

[8] B.M Sarwar, Konstan J.A., Borchers, A., Herlocker, J., Miller, B., Riedl, J. Using filtering agents to improve prediction quality in the groupLens research collaborative filtering system. Proceedings of the ACM Conference on Computer Supported Cooperative Work, 1998.

[9] Schein, A.I., A. Popescul and L.H Ungar. Methods and metrics for cold-start recommendations. Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2002.

[10] J. Herlocker, J.A Konstan, L. Terveen and J. Riedl. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS) 22 (1), 2004.

[11] C Ziegler, S.M McNee, J.A Konstan and G. Lausen. Improving recommendation lists through topic diversification Fourteenth International World Wide Web Conference, 2005.

[12] F. Meyer. Recommender systems in industrial contexts, PhD thesis, University of Grenoble, France, 2012.

[13] Eugene F. Krause. Taxicab Geometry. Dover, 1987.

[14] F. Meyer, F. Fessant. Reperio: A Generic and Flexible Industrial Recommender System, Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011, pp. 502-505