A New Cross-Validation Technique to Evaluate Quality of Recommender Systems

Dmitry I. Ignatov[1], Jonas Poelmans[2,⋆], Guido Dedene[2,4], and Stijn Viaene[2,3]

[1] National Research University Higher School of Economics, Russia
dignatov@hse.ru
[2] Katholieke Universiteit Leuven, Belgium
{jonas.poelmans,guido.dedene}@econ.kuleuven.be
[3] Vlerick Leuven Management School, Belgium
stijn.viaene@vlerick.be
[4] Amsterdam Business School, Netherlands

Abstract. The topic of recommender systems is rapidly gaining interest in the user-behaviour modeling research domain. Over the years, various recommender algorithms based on different mathematical models have been introduced in the literature. Researchers interested in proposing a new recommender model or modifying an existing algorithm should take into account a variety of key performance indicators, such as execution time, recall and precision. Till date and to the best of our knowledge, no general cross-validation scheme to evaluate the performance of recommender algorithms has been developed. To fill this gap we propose an extension of conventional cross-validation. Besides splitting the initial data into training and test subsets, we also split the attribute description of the dataset into a hidden and visible part. We then discuss how such a splitting scheme can be applied in practice. Empirical validation is performed on traditional user-based and item-based recommender algorithms which were applied to the MovieLens dataset.

Keywords: recommender systems, quality of recommendations, user-behavior modeling, applied combinatorics.

1 Introduction

A modern Internet user rather frequently faces recommender systems. For example, if a user buys a book X in an online book shop she also gets recommendations in the form "other customers who bought book X also bought books Y and Z". There are a lot of web systems which can recommend potentially interesting web sites to a particular user, they are called social bookmarking systems (e.g. http://del.ici.ou.us). Besides consulting the Internet the most popular and non-technological way to get recommendations is still friends' suggestions. However, if a user wants to buy multiple items (to watch, to read etc.) the task

⋆ Jonas Poelmans is an aspirant of the Fonds voor Wetenschappelijk Onderzoek Vlaanderen or Research Foundation Flanders.

becomes harder, because there may be a lot of different choice options and her friends may not be informed about latest items in the field or just have a different taste. To cope with these difficulties so-called collaborative filtering can be used [3]. Recommender algorithms based on collaborative filtering techniques utilise a fairly simple scheme. They first find users of the system who have similar tastes or preferences as the current user, then compose the list of items the users selected and rank these items. The results she gets are the Top-N items of the list. Another less evident but demanded application is recommending key phrases in web advertising systems, where firms buy advertising phrases from web search engines to show advertisement by a user's request [4,5]. In this paper we propose a novel scheme for evaluating the quality of recommendations made. We empirically show the usefulness of this scheme by applying it to a traditional user-based and item-based recommender algorithm. These algorithms were using the well-known MovieLens dataset.The remainder of this paper is composed as follows. In section 2 we describe in detail the two major groups of recommender algorithms. In section 3 we zoom in on some popular similarity measures. In section 4 we present our recommendation quality evaluation scheme. Section 5 details the results of some experiments made. Finally section 6 concludes this paper.

2 Recommender Models

In this paper without loss of generality we consider only two groups of recommender techniques, which can be called the classical ones, mainly user-based and item-based approaches [2,1]. A key notion for these techniques is similarity, which can be expressed as Jacquard measure, Pearson correlation coefficient, cosine similarity etc. Initial data are usually represented by an object-attribute matrix, where the rows describe objects (users) and the columns represent attributes (items). A particular cell of the matrix can be either 1 or 0, which stands for the fact that the item was purchased or not respectively. Also the values can be rates or marks of items, for example, movie ratings given by users.

User-Based Recommendations. User-based methods find similarity between a target user u_0 and other users of the recommender system. As a result the target user has n most frequently bought items by k most similar to u_0 users (customers). Let u_0 be a target user, u_0^I be items that she evaluated, $sim(u_0, u)$ be a similarity between the target user u_0 and another user u. In this research we use Pearson correlation coefficient as a similarity measure. Define the set of nearest neighbors (neighborhood) for the target user by the formula:

$$N(u_0) = \{u | sim(u_0, u) \leq \Theta\}.$$

However, it is appropriate to obtain $Top - k$ nearest neighbors, that is $Top - k$ defines the threshold Θ. Hence the set of nearest neighbors includes k users which have the similarity with u_0 higher than a certain threshold. After ordering the users by decreasing similarity, one should select not only $Top - k$ of them, but also check the similarity value of $(k+1)$-th user in the list. If this similarity value is equal to the preceding one than one should add $(k+1)$-th user to the neighborhood $N(u_0)$. One should repeat the procedure until the next similarity value changes. Since we predict the rate of an item i by a specific target user u_0 we are interesting only those users from the neighborhood who have evaluated i:

$$N(u_0|i) = \{u|i \in u^I \& \ u \in N(u_0)\}.$$

Denote by r_{ui} the rate (mark) of an item i by a user u we obtain the formula for the predicting rate

$$\hat{r}_{u_0 i} = \frac{\sum\limits_{u \in N(u_0|i)} sim(u_0, u) \times r_{ui}}{\sum\limits_{u \in N(u_0|i)} sim(u_0, u)}.$$

Item-based recommendations. The idea of the item-based technique is similar to the described user-based method, but similarity is calculated between items. Denote by u_0 again the target user, by u_0^I the items she evaluated, by $sim(i,j)$ the similarity between items i and j. Define the neighborhood for an item i analogously the neighborhood for a target user by $N(i) = \{j|sim(i,j) \geq \Theta\}$. By doing so we have top-k nearest items to i, that is top-k defines Θ. To predict the rate for a target user u_0 one has to compare the items which u_0 evaluated with those that she didn't rate. Therefore we refine the formula for item neighborhood taking into account the target user as follows $N(i|u_0) = \{j|j \notin u_0^I, i \in u_0^I, j \in N(i)\}$.

Denote by r_{ui} the rate of an item i by a user u and by doing so we get

$$\hat{r}_{u_0 i} = \frac{\sum\limits_{j \in u_0^I} sim(i,j) \times r_{u_0 j}}{\sum\limits_{j \in u_0^I)} sim(i,j)}.$$

Then we rank marks in decreasing order and return the first n of them as a recommendation. The main computational advantage of this method is based on the following fact: the number of e-commerce web-site users is usually increasing over time, but new items are added not so frequent. That is why pairwise users' similarity computation while forming a new recommendation may take much time, but items' similarity can be calculated offline in advance and the obtained similarity matrix can be reused many times later. These techniques have some shortcomings, for instance, in case of the so-called cold start problem we don't know user's history and these methods are not able to make recommendations, but they can do it better than some more sophisticated algorithms in case the user's history is known.

3 Similarity Measures

To define similarity between two objects or attributes different similarity measures (or even metrics) are used. Usually, such a measure has the value between 0 and 1 (for absolute similarity). Let us consider some of these measures.

Distance-Based Similarity. To calculate similarity we should find the distance between compared objects or attributes. There are some frequently used ways to calculate the distance. Each initial object is represented by a vector in attribute space (dually for distance between two attributes). Then *Euclidean distance* between two objects x and y is defined as $d(x, y) = \sqrt{\sum\limits_i (x_i - y_i)^2}$. *Hamming distance* is usually used for binary data, which is defined by the formula $d(x, y) = \sum\limits_{x_i \neq y_i} 1$. The simplest way to calculate the similarity then is to apply the following formula $s(x, y) = \frac{1}{1+d(x,y)}$ (see, e.g. [7]).

Let us take a closer look at Hamming distance. In this case d takes only natural values and 0 and the maximal value of s is equal to 1 for $d = 0$. And for the next value $d = 1$ the similarity s is equal to 1/2; it's a clear drawback of the similarity calculation formula. For example, let x and y be two binary vectors which differ only in one component, according to the previous formula they are only one half similar. This rough character of s values can be easily seen in figure 1.

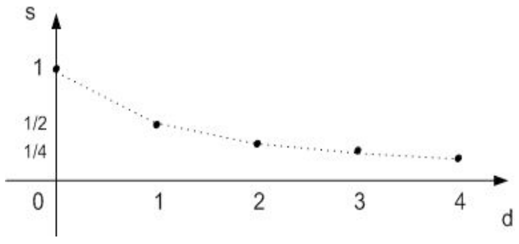

Fig. 1. Similarity versus Hamming distance

Correlation as similarity. In the formula below similarity between two vectors is calculated by means of the well-known Pearson correlation:

$$Pearson(x, y) = \frac{\sum\limits_i (x_i - \overline{x})(y_i - \overline{y})}{\sum\limits_i (x_i - \overline{x})^2 \cdot \sum\limits_i (y_i - \overline{y})^2},$$

where $-1 \leq Pearson \leq 1$.

The main drawback of Pearson correlation as a similarity measure is its undefined value for vectors with constant components. By the way, we have zero denominator for the vector $x = (4, 4, \cdots, 4)$. This is why we may loose some potentially recommended items. For example, let us consider two vectors $a = (0, 5, 5, 4)$ and

$b = (0, 4, 5, 0)$. If one would think on them like tuples of two users' rates then it is intuitively clear that these users are quite similar to each other. However, the correlation will not be calculated because a following constraint: the initial vectors are trimmed to their non-zero components [8]. In our case one should calculate the correlation between (5,5) and (4,5). How it was shown above the correlation is undefined. Some authors proposed to suppose the correlation value equal to 0 [7], but in our opinion it is not correct.

Other similarity measures. There are dozens different measures to find the similarity, for example cosine similarity (very close to Pearson), Jacquard and Tanimoto coefficients, etc.

4 Recommendation Quality Evaluation

In this section we propose the scheme for quality evaluation of an arbitrary recommender system. Let initial data be represented by an object-attribute table (binary relation) $T \subseteq U \times I$, which shows that a user $u \in U$ purchased $i \in I$, i.e. uTi. To evaluate quality of recommendations in terms of precision and recall we can split the initial user set U into training $U_{training}$ and U_{test} test subsets. The size of the test set as a rule should be less than the size of the training set, e.g. 20% and 80% respectively. Recommendation precision and recall is evaluated on the test set. This part of the algorithm looks like one step of the conventional cross-validation. Then each user vector u from U_{test} is divided into two parts which consist of evaluated items $I_{visible}$ and non evaluated items I_{hidden}. I_{hidden} are the items which we intentionally hided. Note that in existing literature the proportion between size of $I_{visible}$ and I_{hidden} is not discussed even in similar schemes [8]. Then, for example, a user-based algorithm makes recommendations according to similarity between users from the test and training sets. Each user from U_{test} gets the recommendations as a set of fixed size $r_n(u) = \{i_1, i_2, \cdots, i_n\}$. Precision and recall defined by

$$recall = \frac{|r_n(u) \cap u^I \cap I_{hidden}|}{|u^I \cap I_{hidden}|},$$

$$precision = \frac{|r_n(u) \cap u^I \cap I_{hidden}|}{|r_n(u) \cap I_{hidden}|},$$

where u^I is a set of all items from I bought by the user u. The values of these measures are calculated for each user and then averaged. The experiment is performed several times, e.g. 100, for different test and training set splits. Then the values are averaged again. In addition there is a possibility to select I_{hidden} set, which can be done at random, but we have to specify the proportion, e.g. 20%. The idea behind the method comes from machine learning where it is called cross-validation, but in case of recommender systems some modifications are needed. We modified m-fold cross-validation, which is performed by splitting the initial set into m disjoint subsets, where each subset is used as a test set and the other subsets are considered as training ones. In addition to standard ways of precision and recall computation we propose original modification of their formulas in case of division by zero. In particular, if $|u^I \cap I_{hidden}| = 0$ then $recall = 1$. If $|r_n(u) \cap I_{hidden}| = 0$ and $u^I = 0$, then $precision = 1$, otherwise $precision = 0$. Experiments have been done on MovieLens datasets about films' rates and synthetical ones which are generated by us.

5 Experiment Results

We have carried out a series of experiments on the movie dataset which contains 1682 movies rated by 943 users and each of the users has evaluated at least 20 movies. All experiments were done on a laptop with Intel Core 2 Duo 2 GHz processor and 3Gb RAM under the operating system Windows Vista. All algorithms were implemented in Python 2.6. We show results of the experiment concerning precision and recall behavior for different number of hidden items (10-fold cross-validation with a neighborhood size of 10).

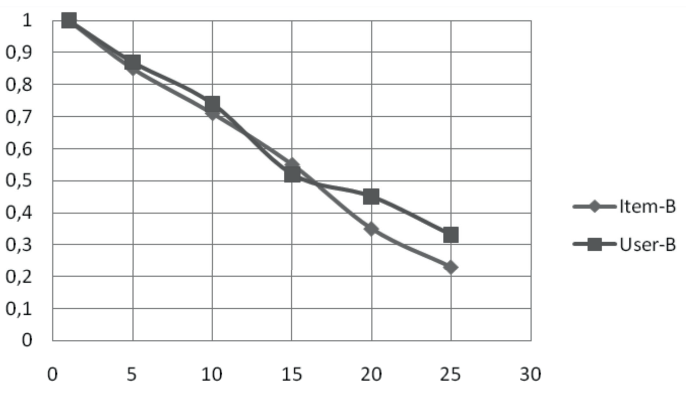

Fig. 2. Recall versus number of hidden items

As we can see from 3 and 4 these methods have almost identical behavior, but the recall of the user-based method is a bit higher in the range from 1 to 10 hidden attributes. In the same way, we conduct experiments on movie rates data where the fraction of hidden attributes was ranging from 1% to 20%. Our diagrams show that the item-based method works better with a rather small number of hidden attributes (≈1%). For higher values of $|I_{hidden}|$ the item-based method's quality drastically decreases, while the performance of the user-based method is still quite stable, moreover in our experiments at 6-7% for $|I_{hidden}|$ the recall slightly increases.

We show how the quality of the results is influenced by the number of neighbors and the test set size for our synthetic data set of size 20 $users \times 20$ $items$ with four rectangles 5×5 full of ones (see 4 and 5). We can conclude that precision and recall increase while the number of neighbors grows and the user-based

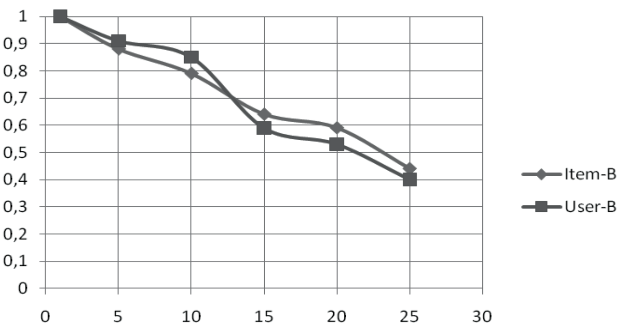

Fig. 3. Precision versus number of hidden items

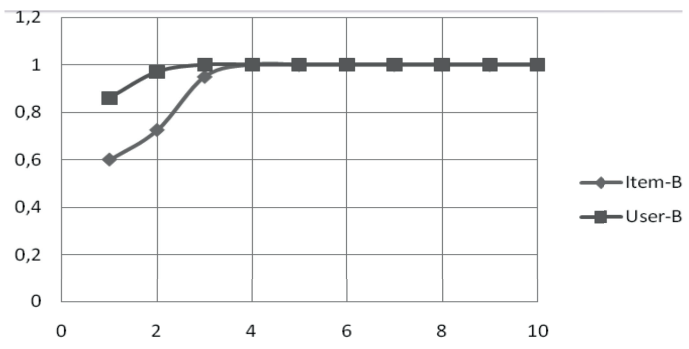

Fig. 4. Recall versus number of nearest neighbors

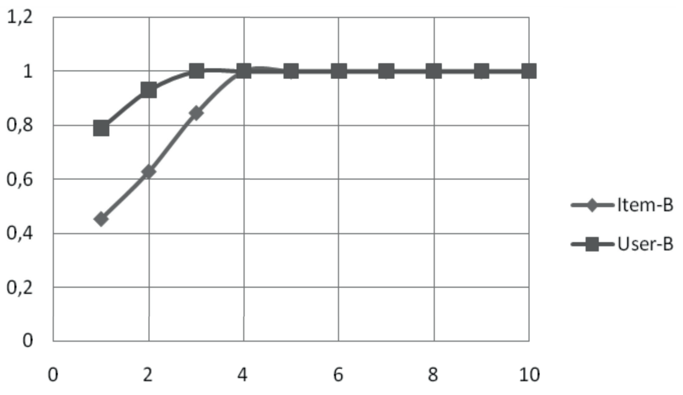

Fig. 5. Precision versus number of nearest neighbors

method needs less number of neighbors than item-based for gaining the same quality. Results for varying test set's size have similar nature, the larger test size the better quality of prediction, and again user-based method shows better quality.

6 Conclusions

The proposed method for recommender systems evaluation makes it possible to compare recommender algorithms and tune their parameters. In the experimentation on real data sets we have found that classical user-based methods are better than item-based in terms of recall and precision for 10 hidden items (Top-10, one of the most typical sizes of recommender lists). Our approach enables the comparison of any other recommender algorithms, e.g. biclustering based algorithms [4,5]. Our future research on recommender systems' quality assessment include a comparative study of biclustering based algorithms and investigation of statistical properties of the approach also taking into account the time parameter. Another potentially interesting avenue is analysing the performance of a Formal Concept Analysis inspired recommender algorithm [9].

References

1. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. J. ACM Transactions on Information Systems 22(1), 143–177 (2004)
2. Sarwar, B.M., Karypis, J., Konstan, J.A., Riedl, J.: Analysis of recommendation algorithms for e-commerce. In: ACM Conference on Electronic Commerce, pp. 158–167 (2001)
3. Goldberg, D., Nichols, D.A., Oki, B.M., Terry, D.B.: Using Collaborative Filtering to Weave an Information Tapestry. Communications of the ACM 35(12), 61–70 (1992)
4. Ignatov, D.I., Kuznetsov, S.O.: Concept-based Recommendations for Internet Advertisement. In: Sixth International Conference on Concept Lattices and Their Applications, pp. 157–166. Palacky University, Olomouc (2008)
5. Ignatov, D.I., Kuznetsov, S.O.: Data Mining techniques for Internet Advertisement Recommender System. In: Proc. of 11th National Conference on Artificial Intelligence, Lenand, Moscow, vol. 2, pp. 34–42 (2008) (in Russian)
6. del Olmo, F.H., Gaudioso, E.: Evaluation of recommender systems: A new approach. J. Expert Systems with Applications 35, 790–804 (2008)
7. Segaran, T.: Programming Collective Intelligence. O'Reilly Media, Sebastopol (2007)
8. Symeonidis, P., Nanopoulos, A., Papadopoulos, A.N., Manolopoulos, Y.: Nearest-biclusters collaborative filtering based on constant and coherent values. J. Information Retrieval 11, 51–75 (2007)
9. Poelmans, J., Elzinga, P., Viaene, S., Dedene, G.: Curbing domestic violence: instantiating C-K theory with formal concept analysis and emergent self-organizing maps. Int. Syst. in Accounting, Finance and Management 17(3-4), 167–191 (2010)