EVALUATION OF COLLABORATIVE FILTERING ALGORITHMS USING A SMALL DATASET

Fabio Roda, Leo Liberti
LIX, École Polytechnique, 91128 Palaiseau, France

Franco Raimondi
School of Engineering and Information Sciences, Middlesex University, London, U.K.

Keywords: Recommender systems, TMW, KNN.

Abstract: In this paper we report our experience in the implementation of three collaborative filtering algorithms (user-based k-nearest neighbour, Slope One and TMW, our original algorithm) to provide a recommendation service on an existing website. We carry out the comparison by means of a typical metric, namely the accuracy (RMSE). Usually, evaluations for these kinds of algorithms are carried out using off-line analysis, withholding values from a dataset, and trying to predict them again using the remaining portion of the dataset (the so-called "leave-n-out approach"). We adopt a "live" method on an existing website: when a user rates an item, we also store in parallel the predictions of the algorithms on the same item. We got some unexpected results. In the next sections we describe the algorithms, the benchmark, the testing method, and discuss the outcome of this exercise. Our contribution is a report of the initial phase of a Recommender Systems project with a focus on some possible difficulties on the interpretation of the initial results.

1 INTRODUCTION

Recommender systems have became an important research area in the field of information retrieval; many approaches have been developed in recent years and many empirical studies appeared. Evaluation of most works has been carried out using "artificial" dataset provided by well know research groups, such as GroupLens, or by the Netflix prize. This situation assures somehow a standard in the way results are evaluated but also can lead to tuning the algorithms to work better with dataset which have specific features. In general, from the practitioner's point of view, there is a lack of precise delimitation for the domain of applicability of algorithms and it is not totally clear which is the minimal size of a dataset for a certain algorithm. Small and poor datasets can "waste" good algorithms; in addition, most algorithms are likely to have very similar performance on these kinds of dataset, thus rendering initial design choices very difficult. The aim of this paper is to provide elements of discussion on this subject.

To ground our claims we make use of a concrete online application which suggests places in the real world (restaurants, bars, shops, etc) to registered users, trying to match users' tastes by digging into the network of friends and places of the community to get the best possible responses. Using an accepted terminology (Vozalis and Margaritis, 2003), we can classify the application as a Collaborative Filtering System. We have implemented our in-house algorithm for this application, called TMW, which tries to overcome the issue of fragmented information, and we have compared it to two classical algorithms: user-based k-nearest neighbour and Slope One. These three algorithms constitute the base for the experimental evaluation presented in this paper.

The reminder of this paper is organised as follows. In Section 2 we describe the three algorithms. In Section 3 we report the experimental results obtained when running these algorithms on a production environment. In Section 4 we discuss the results.

2 BACKGROUND

In this section we describe the algorithms used for the evaluation: *k*-nearest neighbour (KNN), Slope One,

Roda F., Liberti L. and Raimondi F..
EVALUATION OF COLLABORATIVE FILTERING ALGORITHMS USING A SMALL DATASET.
DOI: 10.5220/0003336506030606
In *Proceedings of the 7th International Conference on Web Information Systems and Technologies* (WEBIST-2011), pages 603-606
ISBN: 978-989-8425-51-5

and TMW. For a given user u and a given item (place) i we denote with $Pr(u,i)$ the outcome of the algorithm (the prediction).

2.1 KNN

In the well known KNN approach to collaborative filtering (Breese et al., 1998), input data are expressed by a matrix R, where each entry r_{ui} represents the rating of user u for item i. Rows and columns (representing users or items) are compared to identify "similarities". Especially in the so called "user based" approach we look for correlation between users. Indeed, users who rate the same items in a similar way form a *"Neighborhood"* (we call them *"mentors"*) that we can use to estimate a prediction. We use a to identify the "active user" (the one we want to anticipate) and sim_{au} the similarity between the active user a and a generic user u. The simplest way to use the mentors' ratings is a weighted average, so the prediction function is:

$$prediction(a,i) = \frac{\sum_u r_{ui} * sim_{au}}{\sum_u |sim_{au}|}$$

There are many ways to compute the similarity between users, for example, a well-established technique is *"Cosine Similarity"*. The similarity between two users, u, and a, is obtained by calculating the cosine of the angle formed between the corresponding vectors (rows in the matrix).

$$cosine\ similarity = \sum_i \frac{r_{ui} * r_{ai}}{\sqrt{\sum_i r_{ui}^2}\sqrt{\sum_i r_{ai}^2}}$$

Different similarity criteria can been used, for example *"Pearson's Correlation"* is a frequent alternative to cosine similarity. This form of KNN is quite primitive, and some authors proposed various improvements. However, as shown in the Netflix prize, these improvements cannot overturn extremely negative results. Therefore, we decided to implement these two basic versions of the KNN to have an idea of its initial performance. In the following KNN 1 is based on Cosine Similarity and KNN 2 on Pearson's Correlation.

2.2 Slope One

The Slope One algorithm compares similar items rather than similar users and determines the "popularity differential" between them. Essentially, it calculates the difference δ between the average rating of two items, j_1 and j_2, and then uses this value to predict the rating for j_2 of a given user u_i in the following way:

$$prediction(u_i, j_2) = r_{i_1} + \delta$$

If we look at this process in terms of linear regression we can say that, instead of applying a transformation like f(x) = ax + b it uses a reduced form f(x) = x + b. When we have more than one known r_i we can get several predictions and so the proposed best one is the average of all:

$$prediction = \frac{\sum_i r_{i_1} + \delta}{|R_i|}$$

where R_i is the set of all relevant items. It has been shown in (Lemire and Maclachlan, 2005) that this algorithm can provide a good accuracy.

2.3 TMW

The idea underlying TMW is to form and maintain an heterogeneous graph whose nodes are users and places. This idea has been proposed by (Aggarwal et al., 1999), and in (Huang et al., 2004) the authors propose using a *"two-layer graph"* (one layer for users and one for items). This approach is not mainstream in the recommender system community so we describe briefly the basic ideas of our TMW algorithm below (details can be found in our previous work (Roda et al., 2009)).

2.3.1 Formalization

Let U (for *users*) and P (for *places*) be two finite sets and $V = U \cup P$ the resulting vertex set. We introduce the following two graphs:

- $R = (V,A)$ is a bipartite digraph; $A \subseteq U \times P$ is weighted by a function $\rho : A \to [-1,1]$ (weights represent *ratings*).
- $S = (U,B)$ is a graph weighted by a function $\gamma : B \to [0,1]$ (it represents the known *social network* and weights represent the "trust" between users).

$G = R \cup S$ is the union of the two graphs which models a mixed network (ratings and social relationships). It is a base to infer new relations and to modify existing ones exploiting the fact that sometimes users like (almost) the same places in (almost) the same way. While the system gets more information (i.e. new ratings are expressed by old and new users) new relations can be established and the social network grows (formally, G evolves in a richer graph G'). In order to produce a prediction for a given user and a given place, we have to find the best "stream of information" in this augmented and dynamic network.

2.3.2 Identification of Maximum Confidence Paths

We treat this problem as an optimization problem and try to consider this situation as a stream of information which flows in the social network, constrained by the level of trust users share each other. Intuitively, we look for the best path between the active user and the best mentor. We make the assumption that the confidence for a path is defined by the lowest confidence arc in the path, thus trusting the *"word of mouth"* if this flows through trusted users only, and we discard a path if it contains an "unreliable" agent.

The problem reduces to finding the *maximum confidence path* between a user and the best mentor: this is the same as finding a path whose arc of minimum weight is maximum between all the path minima, which is precisely the so-called *maximum capacity path problem*. Algorithms are known to solve this problem in time linear in the number of arcs (Punnen, 1991). When the best path is found we take the last but one edge m_{best} as the best mentor and the weight of the very last arc (i.e., its rating) as the prediction.

In parallel, we carry out the computation of the value of the bottleneck which represents a trust level and which allows us to choose trusted recommendations. When the trust level is lower than a certain threshold, we replace the TMW prediction with a value computed as the linear regression between user's ratings and the average community rating.

3 EVALUATION OF THE ALGORITHMS

We use a simple metric to evaluate our system, in order to have clear results easy to communicate to non technical stakeholders. The choice was made because of the clamour of the Netflix competition and so we used the same metric: accuracy computed as the square root of the averaged squared difference between each prediction and the actual rating (the root mean squared error or "RMSE"). Let the r_{ui} denote the actual rating provided by a certain user u for an item i, with $i = 1, 2, \ldots, n_u$ ($n_u \leq n$, where n is the number of all available items) and let p_{ui} denote the prediction generated by a certain algorithm for the same user and the same item. RMSE, relating to user u, is defined by:

$$RMSE_u = \sqrt{\frac{\sum_{u=1}^{n_u} (r_{ui} - p_{ui})^2}{n_u}}$$

The total RMSE is obteined as the average of the RMSE of all users. As mentioned above, the evaluation of RMSE is typically performed using the "leave-n-out" approach , where a part of the dataset is hidden and the the rest is used as a training set for the recommender, which tries to predict properly the withheld ratings. Here we employ a different method, due to the possibility of recording ratings and predictions live. TMW is currently used as the main recommendation algorithm for a website to which we have authorized access; in this way we don't need to simulate the prediction process because we can compare predictions with real ratings from users. Using the website users enter ratings for places (such as restaurant, bars, etc.), on a scale from 0 to 5, and can receive recommendation based on their voting history.

For our experiment we proceed as follows: every time a user provides a rating, we calculate the predictions with all the three algorithms described above using the entire dataset (with the exception of the rating entered), and store all the prediction results. Basically, for every rating $R(u_i, p_k)$, we record a line in a log file with the predictions about that item and user of TMW, KNN 1 (using cosine similarity), KNN 2 (using Pearson correlation) and Slope One, respectively, as shown in Table 1.

Table 1: Log line for the experiment.

rating (real)	TMW	KNN 1	KNN 2	Slope 1
$R(u_i, p_k)$	Pr_{tmw}	Pr_{knn_1}	Pr_{knn_2}	Pr_{sl1}

As a benchmark to evaluate all the algorithms we employ the *community average* for a certain item, with the aim of measuring how much each algorithm can improve the simple *community recommendation*. Thus, we also compute the RMSE of the community recommendation with respect to the actual ratings provided by users.

The results reported in the following tables refer to the analysis we performed using the ratings we got over the first four months of activity of the website. We have 315,463 ratings, inserted by 69,794 users on 147,319 items (places). Table 2 reports the overall RMSE computed in this period.

Table 2: RMSE for all ratings.

comm.	TMW	KNN 1	KNN 2	Slope 1
1.0710	0.9865	0.9872	0.9884	0.9859

In Table 3 we express this result in relative terms by providing the *rate of improvement* with respect to the average of the ratings by the community: for instance, TMW improves community ratings by 7.89% on average.

Table 3: Improvement over community average.

Algorithm	Improvement over community avg.
TMW	7.89 %
KNN 1	7,82 %
KNN 2	7,71 %
Slope One	7,95 %

4 DISCUSSION AND CONCLUSIONS

A first consideration is that all the algorithms don't supply predictions dramatically more accurate than the community average. Even if it is well known now (see Netflix prize results) that an improvement of a few percentage points of accuracy is hard to get, still in absolute terms the RMSE seems a bit excessive (remember that votes range from 0 to 5). Indeed, in all cases the RMSE is always greater than 0.9, representing an average error in the order of 20% on the actual ratings.

A second observation is that our results do not clearly show which algorithm works better. TMW provides slightly better results than KNN but their outcomes are very similar. We find a bit surprising that KNN using cosine similarity performs better than the one based on Pearson Correlation and that Slope One performs better than both of them. Indeed, in line with previous works we expected KNN based on Pearson Correlation to perform the best. We identify two possible reasons. First, we considered a plain version of KNN and did not investigate possible improvements. Second, TMW only looks for the best mentor available, instead of carrying out a full neighbourhood formation, which could pose problems in a sparse dataset.

We suspect that the results obtained in our experiments are due to the structure and the dimension of the dataset. Indeed, the training data set of Netflix consists of 100 million ratings provided by over 480 thousand users, on nearly 18 thousand movie titles. Group Lens provides three datasets, one of 100,000 ratings by 943 users for 1,682 movies, another of 1 million ratings by 6,040 users for 3,900 movies, and a third of 10 million ratings and 100,000 tags by 71,567 users for 10,681 movies. Jester Joke dataset has 4.1 million by 73,496 users on 100 jokes. On the contrary, our dataset is sparser, and thus most of the RMSE analyses that can be found in the literature do not apply to our case.

The main issue, which arises from our experience, is that it is not clear which is the minimal dimension of a dataset to make it a reliable base to build a test bed. This a very important question in our opinion and our impression is that it has been underestimated in the literature. Of course, if a dataset is untrustworthy an alternative consists in using public datasets, but this may be audacious, because it may be not ideal to tune a system to recommend places on a dataset which was originated from a system to recommend movies. Additionally, even if the collaborative filtering approach works with generic items, new difficulties may occur later when the system is extended with content based capabilities.

In conclusion, this experiment has shown that from the practitioner's point of view, finding the best algorithm is equivalent to finding a reliable dataset and test bed, and that this issue has not been addressed adequately in the literature.

REFERENCES

Aggarwal, C., Wolf, J., Wu, K.-L., and Yu, P. (1999). Horting hatches an egg: a new graph-theoretic approach to collaborative filtering. In Fayyad, U., Chaudhuri, S., and Madigan, D., editors, *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 201–212, New York. ACM.

Breese, J., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. Technical Report MSR-TR-98-12, Microsoft Research.

Huang, Z., Chung, W., and Chen, H. (2004). A graph model for e-commerce recommender systems. *Journal of the American Society for Information Science and Technology*, 55(3):259–274.

Lemire, D. and Maclachlan, A. (2005). Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*.

Punnen, A. (1991). A linear time algorithm for the maximum capacity path problem. *European Journal of Operational Research*, 53:402–404.

Roda, F., Liberti, L., and Raimondi, F. (2009). Combinatorial optimization based recommender systems. In *Proceedings of the 8th Cologne-Twente workshop (CTW09) on Graphs and Combinatorial Optimization*, Paris.

Vozalis, E. and Margaritis, K. (2003). Analysis of recommender systems algorithms. In Lipitakis, E., editor, *The 6th Hellenic European Conference on Computer Mathematics & its Applications*, pages 732–745, Athens. Athens University of Economics and Business.