BBM Team Report

Alessandro Baggio
alessandro1.baggio@mail.polimi.it

Davide Bianchi
davide9.bianchi@mail.polimi.it

Camilo Martinez
camiloandres.martinez@mail.polimi.it

Categories and Subject Descriptors
Recommender System

General Terms
Algorithm

Keywords
Collaborative filtering, Association Rules, KNN, Hybrid recommendation.

1. INTRODUCTION

The following paper represent the work done by the BBM group for the AUI: Technology 2014 Competition on Kaggle.

In the following section are described our main algorithms used for this competition. To understand our final algorithm that is a hybrid one, we will explain also the algorithms that take part into it. For this reason we are going to introduce:

- Associations Rules
- User-similarity Knn
- Hybrid version (involving the last two algorithms)

We also have tried to use item-item similarity and Pure SVD without achieving better result than the ones that we had at the time we implement them.

2. OUR WORK

2.1 Associations Rules

The basic idea for applying association rules to recommender systems is to treat the items rated by each user as a single itemset. We considered whether to include only the items the users rated as relevant, however better results were obtained by including all the rated items.

Once these itemsets were defined we use the a priori algorithm to mine association rules, which requires to set two parameters: minimum support and confidence.

The support is defined on an itemset X. In our case it means the proportion of users who have rated a given set of movies. From this it can be said that a set of movies with high support is popular among the users.

The minimum support was set to 2% and its main effect is to speed up the computation of the rules.

The second parameter of the algorithm is associated to a given rule X => Y, X being the antecedent of the rule (i.e. a set of movies rated by the user) and Y the consequent (i.e. a set of movies recommended for the user by this rule). The confidence is defined as the proportion of user who have rated both X and Y over the proportion of users who have rated only X. The confidence can be interpreted as the probability that a set of movies Y would be rated by the user given that he already rated the movies in X.

The minimum confidence was kept at 30% as suggested by default by the implementation used. It's mainly used to control the number of rules generated.

Another significant parameter of the a priori implementation used was the length of the rule defined as the number of items on X plus the number of items on Y. The usual convention in association rules is followed in the implementation being used such that only rules with one item on Y are considered. We avoided single item rules as they would only replicate top popular recommendations. Instead we focused on a minimum of two items (i.e. one on X, one on Y) and tried longer rules without improving the results.

Once the rules have been learned the procedure to generate recommendations is to order them by some metric and match items on the left side of the rule (i.e. items on X) with items on the user itemset (i.e. rated by the user), then recommend the first n unique items on the right side of the ordered rules.

From this is clear that the metric used to order the recommendations has an important impact on their quality. We tried several but the one with best results was confidence. Intuitively this can be explained by the fact that it measures how probable is that the items being recommended would be rated by the user.

Results obtained with association rules were our best for some time but were not enhancing. In fact we discarded this approach when we got good results with the next method.

However it is worth mentioning one of the algorithms used in the final submission leveraged rules with high confidence in order to boost the first position of the recommendation which allowed us to increase a bit our score, as will be explained in the Hybrid section.

The implementation used to mine association rules for recommender systems was the one available in the framework RecommenderLab for R.

2.2 User-similarity KNN

The user similarity is based on calculating the distance (or similarity) between users in order to find the closest (or most similar) to a given user. The prediction is calculated using the top-popular over this subset of users.

At first we looked at the SVD reduced matrix to find the closest users. This method is particularly fast (almost 10 time faster w.r.t. the full rating matrix) but less accurate if the goal is to improve MAP@5 metric (10% less accurate).

The distance (or similarity) is calculated with the spearman formula, which is slightly faster (10-15%) than cosine distance and produced the same experimental results.

The spearman short formula:

$$\rho_s = 1 - \frac{6\sum_i D_i^2}{N(N^2-1)}$$

$$D_i = r_i - s_i$$

And the spearman extended formula:

$$\rho_s = \frac{\sum_i (r_i - \overline{r})(s_i - \overline{s})}{\sqrt{\sum_i (r_i - \overline{r})^2}\sqrt{\sum_i (s_i - \overline{s})^2}}$$

Experimental results shown that even if only ratings grater or equal than 4 are accepted as correct, all the ratings down to 1 must be considerate not only for the formation of the similarity matrix but also for the calculation of the weighted top-popular.

2.3 Hybrid

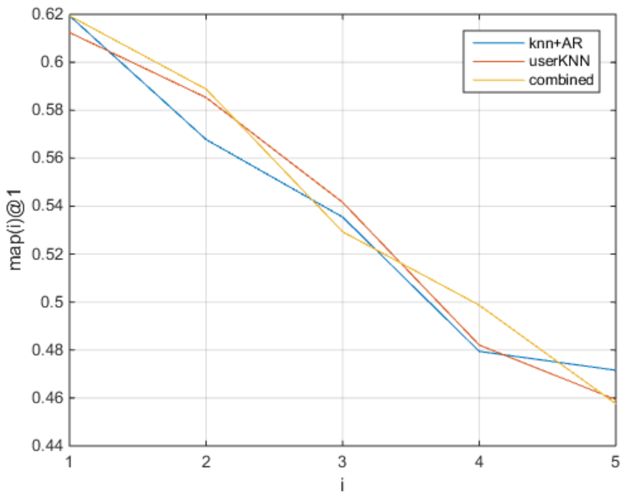
An input of our final algorithm was a hybrid algorithm obtained by extracting some very specific association rules using high confidence level (87%) and high support (12%) and then filling the missing recommendations with the ones coming from our user-similarity KNN algorithm. This increased a bit our score (we gain about 0.5% on MAP@5) basically because we selected very few but effective rules that were almost certain. Unfortunately the number of association rules produced in this way and also the number of recommendations coming with a selection of rules were really few and for this reason we didn't improve more. Probably a larger dataset would generate more rules, so this technique could lead to a better improvement.

Our final submission was based on a cross validation optimization over the possible combination orders of our best performing algorithms by tuning different parameters on them. We also kept in mind to use the minimum number of algorithms possible if we arrived to the same score. In fact we finally settled on a combination of two algorithms we took position one and five of the final recommendation from the hybrid between high-confidence AR rules and user KNN mentioned above. We filled positions 2 to 4 with the best performing spearman similarity user KNN algorithm mentioned last section. As it could be the case that some recommendations were repeated we actually fallback to pick missing recommendations from AR rules hybrid first then from remaining items recommended from user KNN. This was all driven by CV on the data and optimization over the private leaderboard.

By using this combination between different algorithms we were able to improve around 1% more compared to our best performing algorithm working alone. However, we recognized this should be done as a last resort as the improvements are marginal as well as we think is possible that the combined recommendations didn't performed event better as the basis for the combined algorithms were rather similar (i.e. differently tuned versions of user KNN).

A visual tool we relied upon to assess the different well performing algorithms was plotting the MAP@1 for the different positions of the recommendations being made. Keep in mind that this will be obviously higher than the total MAP@5. Figure 1 shows the results for the algorithms combined in our best performing submission. From the plot is possible to justify why we picked the hybrid including AR for the first as last positions as its single position map is over the one for the pure user KNN approach. It is also interesting to see that although we could expect obtaining at least the best result on each position while combining, this doesn't happen as the recommendation algorithms are not as diverse as it will be desirable so that there is some inevitable overlapping specially affecting position 3 which is lower than any of the individual algorithms probably because some of the best recommendations available are being moved to higher positions in the combination.

Figure 1. MAP@1 over each position for combined algorithms



3. EVALUATION/ RESULT

Our evaluation mechanism was basically based on cross validation, using the code provided during the course.

Table 1. In the table are shown the score the algorithms used

Algorithm	**Score on CV (approx.)**	**Score in the competition (private)**
Top Popular (baseline)	30,00%	30,67%
Associations Rules	32,50%	Missing
User-similarity KNN	44,50%	Missing
User-similarity Knn + Associations Rules (1st hybrid)	45,00%	45,37%
Final Hybrid	45,50%	45,75

4. ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed and for the tutors for the code provided as part of the MATLAB practice lessons.

5. REFERENCES

[1] Kluver, Daniel; Konstan, Joseph (2014): *Evaluating recommender behavior for new users*. In: Proceedings of the 8th ACM Conference on Recommender systems (RecSys '14).

[2] Cremonesi, Paolo; Koren, Yehuda; Turrin, Roberto (2010): *Performance of Recommender Algorithms on Top-N Recommendation Tasks*. (RecSys 2010)

[3] Özseyhan, Civan; Badur, Bertan; Darcan, Osman (2012): *An Association Rule-Based Recommendation Engine for an Online Dating Site*. In: Communications of the IBIMA (Vol. 2012)

[4] Hahsler, Michael *recommenderlab: A Framework for Developing and Testing Recommendation Algorithms*.