Recommender System 2015 Challenge Polimi

Jacques Peeters
Polytech Lille
Avenue Paul Langevin, 59655
Villeneuve d Ascq cedex, France
jpeeters@polytech-lille.net

Antoine Vastel
Polytech Lille
Avenue Paul Langevin, 59655
Villeneuve d Ascq cedex, France
avastel@polytech-lille.net

1. INTRODUCTION

During the competition we experimented different algorithms. Some on them were not really successful (UBCF, IBCF …). Two models kept our attention. Firstly a matrix decomposition thanks to a SVD. Secondly a Content Based one.

2. LOCAL TESTING

In order to train and tune our models locally before uploading it. The procedure is the following:

- Compute the popularity for each item (number of times the items was rated)
- Select users which have at least 2 ratings superior or equal to 8 (9615 users),
- Select randomly n test users between those (we choosed 500 in order to not alterate to much the sparsity of the URM and therefore the performance of trained algorithme),
- Delete the 50% more popular ratings of those users from the training dataset and put them in the test dataset,
- Train the choosen model and evaluate its MAP@5 performance.
- Plot the results and tune the model according to the performance.

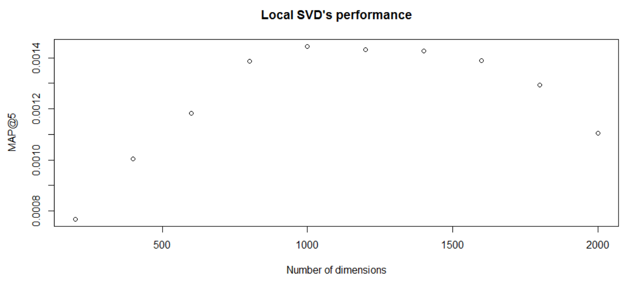
In order to provide more stability to the local results several simulation were done and results were aggregated (kind of bootstrap technique).

3. PRINCIPAL ALGORITHMS

3.1 SVD

No particular library have been used. All the code have been realized personally realized with R language.

There is nothing in particular. Here is the result of MAP@5 performance according to the number of Eigen values kept. The number of randomly selected users is 500 and number of test dataset samples generated is 20. Below is the result:



The best number of dimension seems to be around 1000 and it was confirmed by online testing. We have really good MAP@5 results, it is probably due to the fact that, even if we put the less popular items in our local test dataset, they are still way more popular than the ones in the online test dataset.

3.2 Content Based

Due to the extreme sparsity of the data, we tried a Content Based model. In order to give a bigger weight to the less popular features, we applied Inverse Document Frenquency. There is more parameters to be tuned. Therefore local testing was only used in order to check if the performance was good. This model provided us a result of 0.00502 on the public leaderboard, which is our best performance.

For computing the 33k*33k similarity matrix, we computed blocks of 33k*1k at once, in order not to have memory problem. Therefore we did not needed to store it on the disk, it was faster to just compute it whenever it was needed.

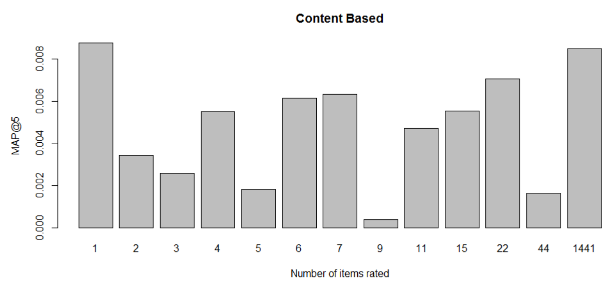
3.3 Hybrid

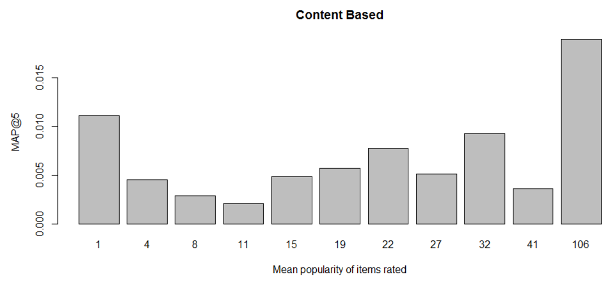
We wanted to see if one of our two models were performing better than the other on a part of users. For this purpose we created User-Features. For example we wanted to see the impact on the performance based on:

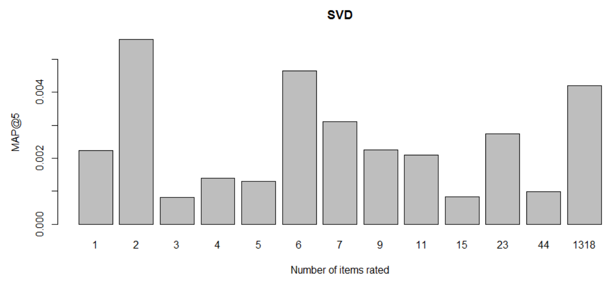
- The number of items seen by the user,
- The mean popularity of the items seen.

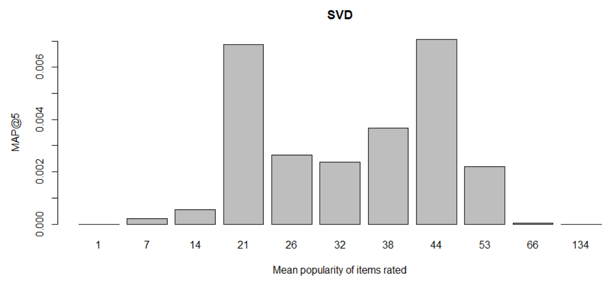
Each bar plot represents an interval of 5% (or 10% if the interval was too small) of the users tested on the feature.

Here are the results plotted:









Despite our work, we don't see any real trend. The single small one seems that SVD algorithm perform better on users with a mean popularity of items rated between 21 and 66. Therefore we tried to recommend to those users the SVD's recommendation and Content Based ones otherwise. Unfortunately, after trying the solution the result wasn't convincing. We could have also try to test compute the mean similarity between items seen for an user but the complexity is n! (with n the number of items seen).

4. CONCLUSION

We really think that higher numbers of models and features tested might be able to provide better hybrid models thanks to using the right model for the right user. Ideally we would have several User-features then a Neural Network for deciding which model to give to each user. As each model is optimized alone and therefore avoid overfitting as much as possible, combining models shouldn't give an higher overfitting.

We worked on our personal laptop, therefore the computation and memory capacity weren't the best. We tried to optimize it in order to keep a good rapidity (less than 10mins for any model or bootstrap step) in order to analyze our results as soon as possible. I would have liked to push the problem a bit forward but during the semester we also had two find our 6 month hiring internship for March and a master thesis for our French school. However the class was a real pleasure and deeply interesting. It did allow us to greatly improve our coding skills which is extremely useful for the future.