Big Data and Analytics Programming: assignment 3

Toon Nolten
toon.nolten@student.kuleuven.be

April 14, 2016

1 Application 1: document similarity detection

I tuned the parameters on 300 files from the Reuters dataset, 100 as in the example was too few to see a significant difference in time between the *brute force search* and the *locality sensitive hashing* approaches. To tune the parameters I implemented another argument in *DocumentRunner*, `-rowsPerBand`, which made it easier to vary the relevant parameters. I then fixed the number of bands to 100 and experimented with the number of rows per band.

At four rows per band, the total amount of compared documents went below 100. I arbitrarily decided this was a good number of rows, this amounts to slightly less than 0.1% of all the documents being compared. At three rows per band about 0.4% of the documents were compared, the reason I decided against this is that it resulted in only one extra similar document pair to be found.

With the number of rows fixed I started decreasing the number of bands, to speed up the algorithm. This has two effects: since the number of rows is fixed, decreasing the number of bands means decreasing the number of hash functions and second, decreasing the number of bands also decreases the number of candidate pairs. At 29 bands, with four rows per band that's 116 hash functions, the performance seemed optimal. Compared to 100 bands we lost two true positives but we only look at 6 false positives. These decisions were obviously taken with speed of execution in mind first and foremost.

If the relative importance of false negatives and false positives was known we could make better decisions. Catching 14 out of 20 similar pairs while only looking at six false positives seems like a strong result. Equivalently: catching 70% of similar pairs while looking at only 0.05% of the possible pairs. The *brute force* solution took approximately 7.6 seconds while *locality sensitive hashing* took only 1.8 seconds.

The command with tuned parameters, as reproduced in the README: `java DocumentRunner -threshold 0.5 -dir ../data/reuters -maxFiles 300 -method lsh -shingleLength 10 -numHashes 116 -numBands 29`

2 Application 2: document similarity detection

I decided to use the same settings as in the previous application as a starting point: a threshold of 0.5, 29 bands and 116 hash functions. Then I started

implementing the rating prediction. First of all I started from the baseline, so the average rating over all ratings. This marks the 0% improvement point, obviously.

Next I added the user and the movie's deviation from the average rating. If the user rates movies higher on average, the prediction will be slightly higher and if a movie is rated higher on average the prediction will increase as well. This resulted in a 2.6% improvement over the baseline.

Lastly I implemented simple collaborative filtering. For every user I retrieve all the users that are similar with respect to the threshold. For each of those I retrieve the rating for the movie under consideration and calculate the deviation from the average rating. Then I calculate the sum of the deviations weighted by their similarity to the user under consideration and divided by the sum of those similarities. This is less than ideal: a better approach would be to look for weights that minimize the RMSE. Another improvement would be to make use of similarity between movies, however the provided data doesn't include the necessary information for this.

Because of horrible planning on my part I did not have sufficient time to tune the parameters. This is because I introduced a bug in the addition of the collaborative filtering which causes `NaN`'s to be produced for the RMSE and without an estimate of the improvement it is of course impossible to tune the parameters. I did find out that the number of rows per band was too small causing more than 400000 false positives.

The command with suboptimally tuned parameters, as reproduced in the README: `java MovieRunner -trainingFile ../data/movielens/ra.train -testFile ../data/movielens/ra.test -threshold 0.5 -method lsh -numHashes 116 -numBands 29`