Data Mining: Learning from Large Data Sets - Fall Semester 2015

mwurm@student.ethz.ch
merkim@student.ethz.ch
lwoodtli@student.ethz.ch

October 11, 2015

Approximate near-duplicate search using Locality Sensitive Hashing

Problem Description

The goal of the project was to develop a program that finds duplicate videos. The videos were provided as shingles in a text file. There was also a training set for the video data. The program needed to run on a Map Reduce environment and should be developed with a Locality Sensitive Hashing algorithm (LSH).

Approach of the Team

The team decided that each member will develop the algorithm independently in a first step. Afterwards the implementations were compared. So the details could be discussed and errors were eliminated. At the end the team decided which implementation should be improved for submission.

Environment

The running environment with a Map Reduce framework was provided. So the Mapper and the Reducer were developed independently and then submitted to the running environment.

Map

The Mapper does the main part of the work. It analyses the shingles of the videos and submits potential duplicates.

For that it uses a Locality Sensitive Hashing algorithm.

In a first step the provided shingles of each video are min-hashed.

```
# each line contains data for a video
for line in sys.stdin:
        # ...
# iterate over the shingles of a video
for shingle in shingles:
        # iterate over the hash functions
        for i,hash in enumerate(minHashes):
                hashed = (hash[0]*shingle + hash[1]) % 20000
                # keep hashed value only if it is smaller
                        if hashed < signature[i]:
                        signature[i] = hashed
```

As a second step the Min-hashes are divided in bands by a Locality Sensitive Hash algorithm.

Resulting potential video pairs are sent to the Reducer. To improve the precision of the Reducer the shingles of the videos are also provided. So the reducer can do a real comparisons of the videos.

But it's still important to have an optimal Mapper step. Missed potential duplicates (false negatives) and wrong potential duplicates (false positives) can weaken the result.

false negatives: They are never compared by the reducer and are missed totally.

false positives: The reducer needs to compare to much potential duplicates and is not efficient anymore.

Reduce

The original shingles are sent to the Reducer together with the hashes. So the Reduces compares all potential pairs for 90% equality. This reduces the risk of false positives. But it introduces more work to the Reducer.

Improvement

To improve the the result there are two main parameters that can be tuned. One of them is the number of hash function to use for calculating the Min-hashes. The other one is the number of rows in a band for the LSH algorithm.

For this project the parameters were tuned by hand. But it would be possible to tune them automatically for some given test data.

Possible Improvement

To reduce the risk of outputting a video pair more than once there could be a final step that compares the output of all Reducers and eliminates multiple same results.

Conclusion

This project shows that LSH is a nice approach to find near duplicates in a large set of videos. It can be simply developed in a Map Reduce environment.