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

caifa.zhou@geod.baug.ethz.ch
pungast@student.ethz.ch
llara@student.ethz.ch

October 14, 2015

Approximate near-duplicate search using Locality Sensitive Hashing

To detect duplicate videos a locality sensitive hashing solution using map-reduce was implemented. Following the steps are described that produce our solution.
First, a mapper was set up. The mapper reads each line and extracts the video ID and shingles, deletes recurring elements in the shingles and sorts the shingles. Then it calculates the signature matrix column for each video and partition it into b bands and hashes each band. Next the mapper emits a key-value pair, where the key is band IDd + hased band and the value is the video ID.

In detail, this is done by implementing a function that creates n hash functions by generating parameters a and b, where a and b are arrays of size n with random integers. The bitstring is hashed with a linear hash function with vector s and the randomly set parameter a and b. The band of the signature matrix is hashed also with a linear hash function. Furthermore the mapper initializes the signature matrix and then computes its column for each video. Candidate pairs are columns that then hash for the same bucket.

It was made sure that each machine used the same seed when generating random numbers for the hash functions.

The paramters where set, such that $br = n$, with $b = 128$ and $r = 8$, both can be changed manually

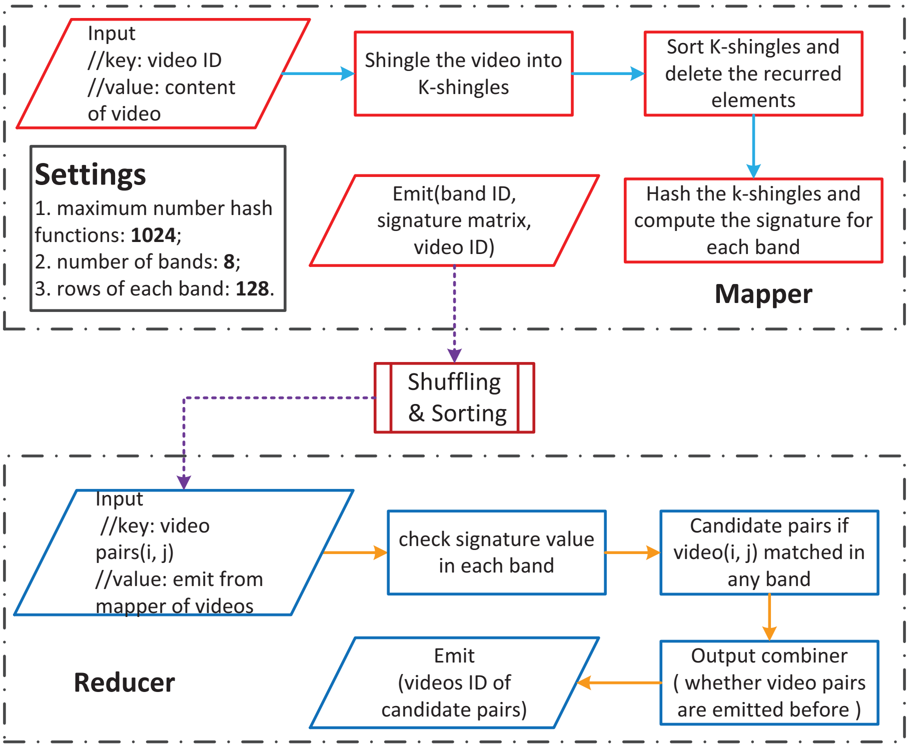
number of bands												
		number of hash functions per group	2	4	8	16	32	64	128	256	512	1024
groups	1	1024	0,999	0,995	0,984	0,958	0,897	0,771	0,545	0,250	0,044	0,00
	2	512	0,997	0,989	0,968	0,917	0,805	0,595	0,297	0,063		
	4	256	0,995	0,979	0,937	0,841	0,648	0,354	0,088			
	8	128	0,989	0,958	0,878	0,707	0,420	0,125				
	16	64	0,979	0,917	0,771	0,500	0,177					
	32	32	0,958	0,841	0,595	0,250						
	64	16	0,958	0,707	0,354							
	128	2	0,841	0,500								
	256	4	0,707									
	512	2	0,00									

Table 1: For finding the number of bands

in the mapper. The parameters b and r were selected in such a way that produces a lower threshold in order to reduce false negatives with t approximately being $\left(\frac{1}{b}\right)^{\frac{1}{r}}$ and thus to catch most similar pairs, but few nonsimilar pairs. The number of bands where based on the computation with the number of groups of the hash functions, which you can find in table 1 below.

In the second part, the reducer was build up. The reducer checks for each key whether the videos are actually equal, by calculating the similarity between the signature matrix columns.The similarity function following computes the proportion of matching elements. If the videos are equal, meaning the similarity is ¿= 0.9, the reducer outputs the video. This is done by filtering out the videos that are not equal.

The general worksflow of the mapper and reducer are illustrated in the Figure 1.



Input
//key: video ID
//value: content of video
Shingle the video into K-shingles
Sort K-shingles and delete the recurred elements
Settings
1. maximum number hash functions: 1024;
2. number of bands: 8;
3. rows of each band: 128.
Emit(band ID, signature matrix, video ID)
Hash the k-shingles and compute the signature for each band
Mapper
Shuffling & Sorting
Input
//key: video pairs(i, j)
//value: emit from mapper of videos
check signature value in each band
Candidate pairs if video(i, j) matched in any band
Reducer
Emit (videos ID of candidate pairs)
Output combiner (whether video pairs are emitted before)

Figure 1: Workflow of the locality sensitive hashing solution using map-reduce.

The groupwork was done according to each member's background. First we sat together and discussed the assignment and overall structure, then Caifa Zhou took over the implementation of the mapper, Taivo Pungas created the reducer as well as revised the mapper and Lara Lingelbach generated the report.