BDA Programming: assignment 2 report

Mauro Paradela del Río

February 19, 2018

1 Implementation discussion

The main class of my implementation is LSHSearch.java. The SimilarPair class has been modified, and the BruteForce class has been removed.

When LSHSearch is invoked, the first step is calculating the signature matrix. This is done online: when a document arrives, it is shingled, and this shingles are hashed and added to the signature matrix right away. This saves memory, because no shingles-docs matrix is created. Also, instead of hashing each shingle each time it appears, they are hashed just the first time, and the hash is stored on a lookup table. This increases speed: reading the tweets takes less than 5 mins, and does not sacrifice much memory (the look up table is 4 bytes * functions * nShingles big).

Then, the LSH is performed: the signature matrix is iterated by rows and bands, which can be configured through the command line. For each band, the signature of each document is computed and hashed into a HashMap, that stores in each entry the bucket and the documents mapped to it. Each time a document is added to a bucket, all candidates on it are checked for similarity, and if the value is over the threshold, then they are added to the set of similar pairs, or rejected otherwise. To improve speed, the SimPairs *equals* and *hashcode* functions have been modified, so the id of each element is used to distinguish each pair (ignoring the similarity). That way, similarity of pairs already classified as similar must not be recalculated, which saves time.

2 Experiments

The parameters of LSH can be tuned to achieve better performance depending on the data. There are two main parameters: rows and bands. Shingling also affects the results, so the number of shingles and its length will also be tested. Finally, Signature matrix rows (hashing functions) and similarity threshold can also be configured.

All these parameters are tested over two results: the one produced by the brute force algorithm, and LSH under the same settings. A subset of 30000 tweets was used instead of the whole set.

Table 1: Shingle length evaluation

Length	TPrate	TP	FP	Time
5	0.85	554	292	1 min 10 s
4	0.83	779	405	3 min 14 s
3	0.83	1227	573	10 min 46 s

A configuration of 100 bands and one row was used for the first experiments. The goal is analysing only the performance of MinHash, and preventing False Negatives on LSH from affecting the result. Later rows and bands are evaluated.

2.1 MinHash

2.1.1 Shingles Length

According to the book chapter provided in the documentation, 5 shingles suffices for emails. Since tweets are usually shorter, experiments range between 3 and 5. In this experiment, the brute force algorithm is also rerun for each possible length.

Table 1: More singles length means that tweets have less shingles in common. Thus, less computation time is needed: since similarity is lower, tweets fall easily under the threshold (0.5 here), so they are not even candidates. Overall, TPrate stays constant, and there are around twice more TP than FP. The FPrate is not used because there 10^6 more TN than FP, so it is better to just look at the total number of FP. I believe 3 shingles are good, since the FP will we later lowered with rows in the LSH.

2.1.2 Number of shingles

For a length of 5, there are 3.2 Million different shingles, considering 20 characters (see documentation). For 3, just 8000. But, twitter has a rich character setting (emojis,capital letters), therefore my first estimation was that the amount of necessary shingles is 40 to the power of k. For k=3, there would be $40^3 = 64000$ shingles. However, the results in Table 2 suggest that until 100k shingles there are collisions that rise the TP and FP, and at least 200k shingles are required to avoid them. The best trade off seems to be at 300k (Low FP, high enough TP), that will be the chosen parameter. I believe even more could be used: after all performance is not affected by shingle number.

2.1.3 Hashing functions

They determine the rows in the signature matrix: the more functions the higher memory requirement. The following experiment uses one row on the LSH to minimize TN.

Table 2: Number of shingles

Length	TPrate	TP	FP	Time
10k	0.77	1296	974	12 min 47 s
50k	0.79	1324	1130	11 min 01 s
100k	0.87	1465	1120	12 min 42 s
200k	0.62	1038	456	10 min 39 s
300k	0.83	1227	573	10 min 46 s
400k	0.82	1190	522	11 min 44 s

There results in Table 3are interesting: the number of permutations considered of the matrix does not have a significant impact on the TP: more or less, the same pairs are classified as similar. Only the FP change: the more permutations the less (except for the 80 interval, where it increases). However, having more bands means spending more time iterating them. Furthermore, memory usage increases. For the next exercise, I would take the 60 hash functions as a starting point, since it yields low enough FP and high TP under less time, and see if I can reduce the FP.

Table 3: Rows of the signature matrix

Length	TPrate	TP	FP	Time
40	0.74	1235	10224	1 min 12 s
60	0.84	1410	2149	3 min 12 s
80	0.89	1493	3330	6 min 40 s
100	0.80	1350	922	11 min 36 s

2.2 LSH Rows and columns

Two tables are shown: one with a threshold of 0.7, in Table 4 and another with 0.8 in Table 5. 0.9 was not used since 15 TP were produced at maximum, and that is not representative.

The data is a bit sparse, with no more than 100 TP. Anyway, there are some trends: more rows means less TP and FP overall. Furthermore, beyond 10 rows the FP is too low. Also, more rows means less FP, so there are less candidate pairs per bucket, which reduces computations. Remarkably, a higher threshold also impacts performance: the higher the faster, and more rows are admissible. My intuition is that on the full dataset,since the threshold is 0.9, more rows are admissible to reduce computation. I would start tuning with ten

The summary of all preceding parameters is in Table 6

Table 4: Rows and columns evaluation. Threshold = 0.7

Rows	Bands	TPrate	TP	FP	Time
2	30	0.82	69	10	6 s
4	15	0.90	76	4	2.3 s
5	12	0.89	75	18	2.2 s
10	6	0.59	50	0	2.2 s
15	4	0.28	24	1	2.2 s
20	3	0.21	18	0	2.2 s
30	2	0.17	15	0	2.2 s

Table 5: Rows and columns evaluation. Threshold = 0.8

Rows	Bands	TPrate	TP	FP	Time
2	30	0.92	37	10	5.8 s
4	15	0.82	33	8	2.1 s
5	12	0.82	33	6	2.1 s
10	6	0.79	32	8	2.1 s
15	4	0.54	22	1	2.1 s
20	3	0.44	18	0	2.1 s
30	2	0.37	15	0	2.0 s

Table 6: Results of all experiments

Shingles length	3
Max shingles	300k
Sig. rows	Min. 60
Rows	Around 10
Columns	4 to 6