CS 324 Data Mining **Problem Set 2**
Il Shan Ng **April 12, 2017**

In this report, I describe how the computed average Jaccard similarity and runtime for both the brute force and LSH algorithms depend on the size of the dataset (d), the number of nearest neighbors for the k-nn portion of the algorithms (k), the number of rows in each band (r), and the number of rows in the signature matrix (n). Unless explicitly varied, the default values used to obtain the following series of results are

$$d = 1000, \quad k = 3, \quad r = 3, \quad n = 1000.$$

Figure 1 shows the effect of the dataset size, measured by the number of documents d, on the average Jaccard similarity and runtime. The left panel shows that the LSH algorithm consistently underestimates the true Jaccard similarity. This error, however, is small. The right panel shows that although both algorithms have about the same runtime for smaller d, beyond $d = 7000$, the brute force algorithm becomes far slower than LSH. In fact, the right panel suggests that the runtime for brute force increases quadratically, whereas that for LSH seems to increase linearly, and so the difference in runtime will be very large for large d. The two panels combined provides strong evidence for the assertion that LSH is able to give a good estimate of the actual Jaccard similarity at much lower computational cost for large datasets.

Figure 1: Effect of number of documents on Jaccard similarity and runtime

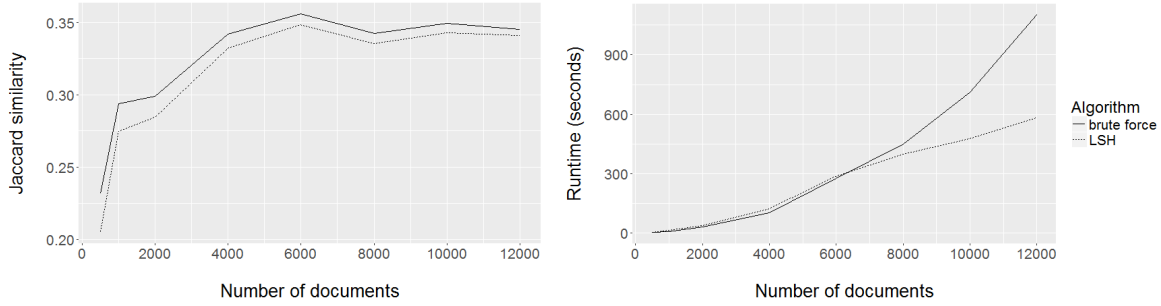

Figure 2 shows the effect of the number of nearest neighbors k on the average Jaccard similarity and runtime. The left panel shows that the Jaccard similarity computed by both algorithms decreases with k. This makes sense, because as k increases, for each given document, both algorithms are picking documents that are less and less similar to that document, resulting in lower average Jaccard similarity. LSH also underestimates the true average Jaccard similarity more as k increases. This happens because it is harder for LSH to find k candidates for each document, and so pads the list of candidates with random documents more frequently for larger k, thus reducing the average similarity across the neighbors. The right panel shows that the runtime for brute force is relatively unaffected by k, whereas that for LSH increases slightly with k. Because this increase is small, I suspect that it arises due to the need to pad the candidate lists of more documents with random draws from the data set. Note that brute force is faster than LSH in this case because I had used only 1000 documents for this simulation.

Figure 2: Effect of number of nearest neighbors on Jaccard similarity and runtime

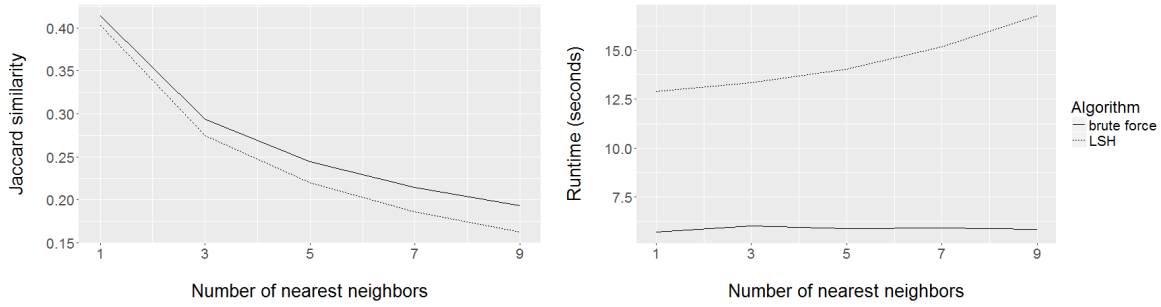

Figure 3 shows the effect of the number of rows r per band in the LSH algorithm on the average Jaccard similarity and runtime. Because the value of r does not affect the brute force algorithm in any way, the Jaccard similarity and runtime for brute force are both constant, and serve as a baseline against which to check the performance of LSH. The left panel shows that the estimate produced by LSH gets worse with increasing r. This occurs because the larger the bands in the signature matrix, the fewer the number of document pairs that agree in at least one band, and so the fewer the candidate pairs that have their Jaccard similarities estimated to be considered for inclusion in the k closest neighbors. Essentially, larger bands filter out more document pairs, sacrificing accuracy for improved runtime. The right panel shows this, with runtime dramatically decreasing as r is increased to 4. For large r, LSH runs as quickly as brute force, but produces estimates so poor that we might as well use brute force.

Figure 3: Effect of number of rows per band on Jaccard similarity and runtime

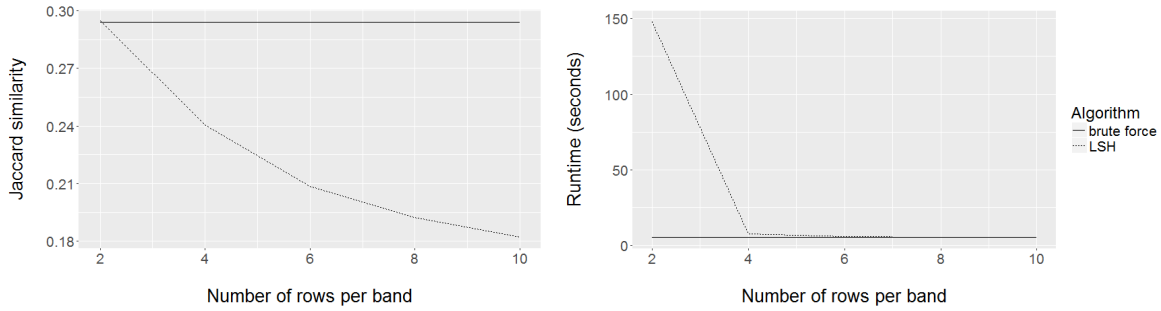

Figure 4 shows the effect of the number of rows n in the signature matrix on the average Jaccard similarity and runtime. Like r, n does not affect the brute force algorithm in any way, and so the results for the brute force can be used as a baseline with which to assess the performance of LSH. The left panel shows that LSH produces better estimates as n increases. This makes sense because using more hash functions to construct the signature matrix effectively simulates a larger sample size, allowing the estimated Jaccard similarity (which is just a proportion) to better estimate the true similarity. The downside to this is shown in the right panel; runtime increases quadratically with n. This happens because the number of element-wise comparisons between two documents increases as n increases.

Figure 4: Effect of number of rows of signature matrix on Jaccard similarity and runtime