Com S 435/535: Notes VI

1 Locality Sensitive Hashing

We can use the MinHash matrix to estimate similarity between any two documents. Suppose that for each document D_a, we would like to identify all pairs of documents that are "highly similar" to D_a. A naive way to accomplish is by considering all pairs of documents and estimate their similarity. Note that this approach takes $O(N^2)$ time, and if N is a Million, this is practically infeasible. Can we focus on pairs that are likely to similar instead of focusing on every pair? This can be achieved by a technique known as *locality-sensitive hashing*. This is a hashing technique that maps "similar items" into the same bucket.

Let M be the $k \times N$ MinHash matrix. Partition rows of M into b bands such that each band has r rows (thus $k = rb$). For a document D_i let its MinHash signature be

$$MH_i = \langle n_1, \cdots, n_k \rangle$$

Note that the entries of MH_i appear in the ith column of M; Let MH_i^1 be the first r entries of MH_i, MH_i^2 be the next r entries and so on. In general for $1 \le \ell \le b$

$$MH_i^\ell = \langle n_{\ell-1} * r + 1, n_{i-1} * \ell + 2, \cdots, n_\ell \rangle.$$

Thus MH_i^ℓ is a r-tuple.

Randomly pick a hash function h from set of r-tuples to $\{1, \cdots, T\}$ (where $T > N$). Let $T_1, T_2, \cdots T_b$ be b hash tables. More precisely each T_i is an array of size T and each cell of the array points to a list.

Use the following algorithm to identify candidate similar pairs.

1. For every $i \in \{1, \cdots, N\}$
   - compute $h(MH_i^1), h(MH_i^2), \cdots h(MH_i^b)$.
   - For every $\ell \in \{1, \cdots, b\}$ place D_i in the list at $T_\ell[h(M_i^\ell)]$

Suppose that two documents D_i and D_j are s-similar, i.e, $Jac(D_i, D_j) = s$. What is the probability that both D_i and D_j are mapped on to the same bucket in *some* hash table? Let us fix a table T_ℓ. Note that D_i and D_j are placed in the same bucket of T_ℓ if $h(MH_i^\ell)$ equals $h(MH_j^\ell)$. Since D_i and D_j are s-similar, the probability that any two corresponding terms of MH_i and MH_j are the same is s. Thus the probability that MH_i^ℓ and MH_j^ℓ are the same is s^r. Thus the probability that $h(MH_i^\ell)$ equals $h(MH_j^\ell)$ is at least s^r. This implies that the probability that D_i and D_j are placed in the same bucket of T_ℓ is at least s^r. Thus the probability that D_i and D_j are placed in different buckets of T_ℓ is at most $(1 - s^r)$. Thus the probability that D_i and D_j are placed in different buckets of every hash table is at most $(1 - s^r)^b$. Thus the probability that D_i and D_j are placed into the same bucket of some hash table is at least $1 - (1 - s^r)^b$.

For the moment, let us assume that $1 - (1 - s^r)^b$ is high when $s \geq 0.8$. This means that if any pair of documents D_i and D_j that are more than 0.8 similar are placed into the same bucket of some hash table. This means the following: For every hash table T_ℓ, for each bucket of the hash table, look at all documents that placed into the same bucket; All of these documents are similar to each other with high probability.

We can ensure that $1 - (1 - s^r)^b$ is high. Consider the plot of the function $f(s) = 1 - (1 - s^r)^b$ with s on x-axis. This function has S shape with sharp threshold (approximately) at $(1/b)^{1/r}$. This means the following: If $Jac(D_i, D_j)$ a little less than $(1/b)^{1/r}$, then the probability that D_i and D_j are mapped to the same bucket in some hash table is small; and if $Jac(D_I, D_j)$ is little greater than $(1/b)^{1/r}$, then the probability that they are mapped to the same bucket in some table in high. Thus our goal is to group documents that are s -similar (for a chosen value of s), then we choose b and r so that s approximately $(1/b)^{1/r}$.

To summarize, below is the algorithm to identify near duplicate documents in a collection of documents:

1. Input $D = \{D_1, \cdots, D_N\}$.
2. Collect all terms $T = \{t_1, \cdots t_M\}$.
3. By identifying each term with an integer, view T as $\{1, \cdots, M\}$.
4. Uniformly at random pick k permutations $\Pi_1, \cdots, \Pi_k$ from $\{1, \cdots, M\}$ to $\{1, \cdots, M\}$.
5. Compute the MinHash Matrix M.
6. Partition rows of M into b bands with each band having r-rows.
7. Pick a hash function h and b hash tables $T_1, \cdots, T_b$.
8. Hash the documents into hash tables $T_1, \cdots T_b$ by using the algorithm described above.
9. Identify documents that are mapped into the same bucket as a group of similar documents.
10. If you want to eliminate false positives in each group (documents that are not similar that are mapped into the same bucket), for every pair of documents in the group estimate their Jaccard similarity using the minhash signatures.