CSCI1520

Eric Cho

February 2025

Idea

I decided to use the min hash, and lsh approach to figure out the q most similar document pairs.

Approach

For this, it was most convenient to use the lecture approach

1. Pick b and r
2. Run (br) minHash $\quad h_i(s_j) \forall i = 1, \cdots, (br) \forall j = 1, \cdots, n$
   (a) Note: early stopping is possible
3. For group $i = 1, \cdots, b$
   (a) For $j = 1, \cdots, n$, throw S_j to bucket $\begin{pmatrix} h_{(i-1)r+1}(s_j) \\ \vdots \\ h_{i-r}(s_j) \end{pmatrix}$
   (b) For every non-empty bucket b, add all pairs in B into C.
   $C \leftarrow C \cup \{(j,k)\} \forall j, k \in B$
4. Return $C \leftarrow$ final candidate pairs

Picking b and r, (and t)

I chose b and r based on the fact that

$$Pr[(S,T) \in C] = 1 - (1 - x^r)^b$$

In other words, if a pair has $x = 0.5$ similarity, then with minHash + LSH, the pair would have probability $1 - (1 - 0.5^r)^b$ of being in the same bucket.

From trial and error, $r = 1, 2, 3, 4$ would make it so that too many document pairs would be in the same bucket. Thus, I decided on $r = 5$. To get the maximum amount of document pairs, using a threshold of 0.5

$$1 - (1 - x^r)^b = 0.5 \approx 1 - \frac{1}{e}$$

$$\Leftrightarrow x^r = \frac{1}{b}$$

for $x = 0.5$ and $r = 5$, $b = 32$.

However, $1 - \frac{1}{e} \approx 0.6$ more than 0.5. So, I decided to use $b = 20$ as

$$1 - (1 - 0.5^5)^{20} \approx 0.47$$

Using this logic, I decided to use $r = 5$ and $b = 20$. Finally, to choose a threshold t for the number of pairs, given that the best threshold is at 0.5, I decided to stop LSH when we found $t = n/2$ document pairs, where t is the threshold number of document pairs and n is the number of documents

Optimizations

Optimization 1

In my first appraoch, I found all the k-shingles per document, and then computed multiple minHash based on my hash functions on the k-shingles. However, it kept crashing because it meant that, if each k-shingle length was approximately $10,000$ and there were $28,000$ documents, I would be storing $28,000 \cdot 10,000 \cdot 6$ characters. This is roughly 2 GB of data.

So, I decided, for each document, to first find the k-shingle, and then compute all the minHash per hash functions. That way, I would store the list of minHashes and not the k-shingles. Thus, I would only be storing $28,000 \cdot b \cdot r$ minhashes.

Optimization 2

I also stopped my lsh after t document pairs were found. This is because, probabilistic, most of the documents would have a similarity score above 0.5 and would definitely be above 0.18. Thus, my code would be faster

Optimization 3

To compute the buckets, I took each band $[h_{(i-1)r+1(s_j)}, \cdots h_{(i-r)}(s_j)]$ for $i = 1, \cdots, b$ and converted them to tuples. Then, I used a dictionary that hashed the tuples into keys and sorted them into buckets. Thus, it would take $O(1)$ time to sort the bands into buckets, and also $O(1)$ to see how many bands were in each bucket.