3.4 Locality-Sensitive Hashing for Documents

Even though we can use minhashing to compress large documents into small signatures and preserve the expected similarity of any pair of documents, it still may be impossible to find the pairs with greatest similarity efficiently. The reason is that the number of pairs of documents may be too large, even if there are not too many documents.

EXAMPLE 3.9 Suppose we have a million documents, and we use signatures of length 250. Then we use 1000 bytes per document for the signatures, and the entire data fits in a gigabyte – less than a typical main memory of a laptop. However, there are $\binom{1,000,000}{2}$ or half a trillion pairs of documents. If it takes a microsecond to compute the similarity of two signatures, then it takes almost six days to compute all the similarities on that laptop.

If our goal is to compute the similarity of every pair, there is nothing we can do to reduce the work, although parallelism can reduce the elapsed time. However, often we want only the most similar pairs or all pairs that are above some lower bound in similarity. If so, then we need to focus our attention only on pairs that are likely to be similar, without investigating every pair. There is a general theory of how to provide such focus, called *locality-sensitive hashing* (LSH) or *near-neighbor search.* In this section we shall consider a specific form of LSH, designed for the particular problem we have been studying: documents, represented by shingle-sets, then minhashed to short signatures. In Section 3.6 we present the general theory of locality-sensitive hashing and a number of applications and related techniques.

3.4.1 LSH for Minhash Signatures

One general approach to LSH is to "hash" items several times, in such a way that similar items are more likely to be hashed to the same bucket than dissimilar items are. We then consider any pair that hashed to the same bucket for any of the hashings to be a *candidate pair.* We check only the candidate pairs for similarity. The hope is that most of the dissimilar pairs will never hash to the same bucket, and therefore will never be checked. Those dissimilar pairs that do hash to the same bucket are *false positives*; we hope these will be only a small fraction of all pairs. We also hope that most of the truly similar pairs will hash to the same bucket under at least one of the hash functions. Those that do not are *false negatives;* we hope these will be only a small fraction of the truly similar pairs.

If we have minhash signatures for the items, an effective way to choose the hashings is to divide the signature matrix into *b* bands consisting of *r* rows each. For each band, there is a hash function that takes vectors of *r* integers (the portion of one column within that band) and hashes them to some large number of buckets. We can use the same hash function for all the bands, but we use a separate bucket array for each band, so columns with the same vector in different bands will not hash to the same bucket.

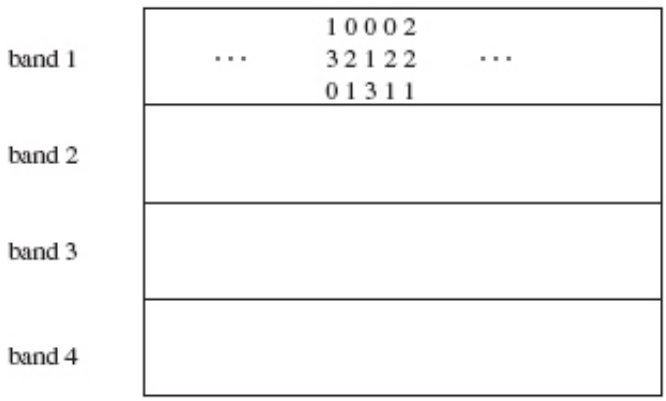

Figure 3.6 Dividing a signature matrix into four bands of three rows per band

EXAMPLE 3.10 Figure 3.6 shows part of a signature matrix of 12 rows divided into four bands of three rows each. The second and fourth of the explicitly shown columns each have the column vector [0, 2,1] in the first band, so they will definitely hash to the same bucket in the hashing for the first band. Thus, regardless of what those columns look like in the other three bands, this pair of columns will be a candidate pair. It is possible that other columns, such as the first two shown explicitly, will also hash to the same bucket according to the hashing of the first band. However, since their column vectors are different, [1, 3, 0] and [0, 2,1], and there are many buckets for each hashing, we expect the chances of an accidental collision to be very small. We shall normally assume that two vectors hash to the same bucket if and only if they are identical.

Two columns that do not agree in band 1 have three other chances to become a candidate pair; they might be identical in any one of these other bands. However, observe that the more similar two columns are, the more likely it is that they will be identical in some band. Thus, intuitively the banding strategy makes similar columns much more likely to be candidate pairs than dissimilar pairs.

3.4.2 Analysis of the Banding Technique

Suppose we use *b* bands of *r* rows each, and suppose that a particular pair of documents have Jaccard similarity *s*. Recall from Section 3.3.3 that the probability the minhash signatures for these documents agree in any one particular row of the signature matrix is *s*. We can calculate the probability that these documents (or rather their signatures) become a candidate pair as follows:

(1) The probability that the signatures agree in all rows of one particular band is s^r.

(2) The probability that the signatures do not agree in at least one row of a particular band is $1 - s^r$.

(3) The probability that the signatures do not agree in all rows of any of the bands is $(1 - s^r)^b$.

(4) The probability that the signatures agree in all the rows of at least one band, and therefore become a candidate pair, is $1 - (1 - s^r)^b$.

It may not be obvious, but regardless of the chosen constants *b* and *r*, this function has the form of an *S-curve,* as suggested in Fig. 3.7. The *threshold,* that is, the value of similarity *s* at which the rise becomes steepest, is a function of *b* and *r*. An approximation to the threshold is $(1/b)^{1/r}$. For example, if $b = 16$ and $r = 4$, then the threshold is approximately 1/2, since the 4th root of 16 is 2.

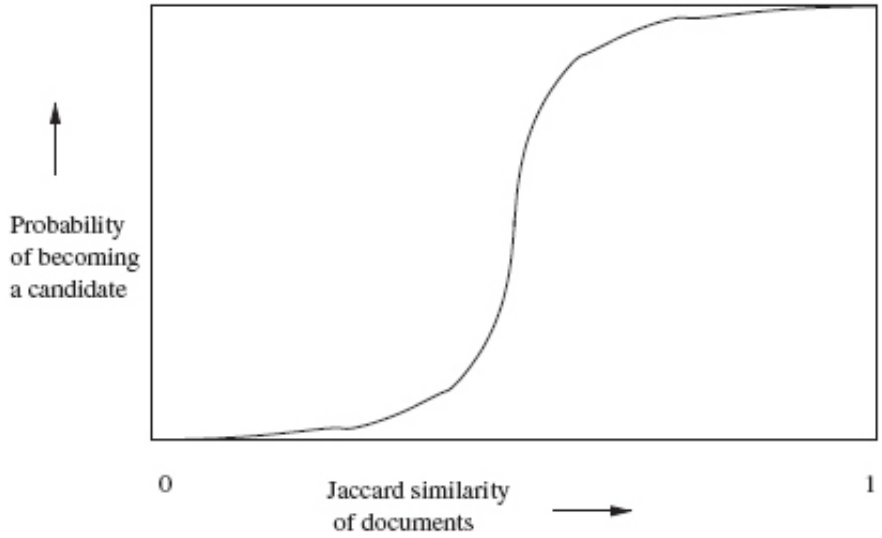

Figure 3.7 The S-curve

EXAMPLE 3.11 Let us consider the case $b = 20$ and $r = 5$. That is, we suppose we have signatures of length 100, divided into twenty bands of five rows each. Figure 3.8 tabulates some of the values of the function $1 - (1 - s^5)^{20}$. Notice that the threshold, the value of s at which the curve has risen halfway, is just slightly more than 0.5. Also notice that the curve is not exactly the ideal step function that jumps from 0 to 1 at the threshold, but the slope of the curve in the middle is significant. For example, it rises by more than 0.6 going from $s = 0.4$ to $s = 0.6$, so the slope in the middle is greater than 3.

For example, at $s = 0.8$, $1 - (0.8)^5$ is about 0.328. If you raise this number to the 20th power, you get about 0.00035. Subtracting this fraction from 1 yields 0.99965. That is, if we consider two documents with 80% similarity, then in any one band, they have only about a 33% chance of agreeing in all five rows and thus becoming a candidate pair. However, there are 20 bands and thus 20 chances to become a candidate. Only roughly one in 3000 pairs that are as high as 80% similar will fail to become a candidate pair and thus be a false negative.

s	$1-(1-s^r)^b$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

Figure 3.8 Values of the S-curve for $b = 20$ and $r = 5$

3.4.3 Combining the Techniques

We can now give an approach to finding the set of candidate pairs for similar documents and then discovering the truly similar documents among them. It must be emphasized that this approach can produce false negatives – pairs of similar documents that are not identified as such because they never become a candidate pair. There will also be false positives – candidate pairs that are evaluated, but are found not to be sufficiently similar.

(1) Pick a value of k and construct from each document the set of k-shingles. Optionally, hash the k-shingles to shorter bucket numbers.

(2) Sort the document-shingle pairs to order them by shingle.

(3) Pick a length n for the minhash signatures. Feed the sorted list to the algorithm of Section 3.3.5 to compute the minhash signatures for all the documents.

(4) Choose a threshold t that defines how similar documents have to be in order for them to be regarded as a desired "similar pair." Pick a number of bands b and a number of rows r such that $br = n$, and the threshold t is approximately $(1/b)^{1/r}$. If avoidance of false negatives is important, you may wish to select b and r to produce a threshold lower than t; if speed is important and you wish to limit false positives, select b and r to produce a higher threshold.

(5) Construct candidate pairs by applying the LSH technique of Section 3.4.1.

(6) Examine each candidate pair's signatures and determine whether the fraction of components in which they agree is at least t.

(7) Optionally, if the signatures are sufficiently similar, go to the documents themselves and check that they are truly similar, rather than documents that, by luck, had similar signatures.

3.4.4 Exercises for Section 3.4

EXERCISE 3.4.1 Evaluate the S-curve $1 - (1 - s^r)^b$ for $s = 0.1, 0.2, \ldots, 0.9$, for the following values of r and b:

- $r = 3$ and $b = 10$.
- $r = 6$ and $b = 20$.
- $r = 5$ and $b = 50$.

! EXERCISE 3.4.2 For each of the (r, b) pairs in Exercise 3.4.1, compute the threshold, that is, the value of s for which the value of $1 - (1 - s^r)^b$ is exactly 1/2. How does this value compare with the estimate of $(1/b)^{1/r}$ that was suggested in Section 3.4.2?

! EXERCISE 3.4.3 Use the techniques explained in Section 1.3.5 to approximate the S-curve $1 - (1 - s^r)^b$ when s^r is very small.

! EXERCISE 3.4.4 Suppose we wish to implement LSH by map-reduce. Specifically, assume chunks of the signature matrix consist of columns, and elements are key-value pairs where the key is the column number and the value is the signature itself (i.e., a vector of values).

(a) Show how to produce the buckets for all the bands as output of a single mapreduce process. *Hint*: Remember that a Map function can produce several key-value pairs from a single element.

(b) Show how another map-reduce process can convert the output of (a) to a list of pairs that need to be compared. Specifically, for each column i, there should be a list of those columns $j > i$ with which i needs to be compared.