19.6 Near-duplicates and shingling

One aspect we have ignored in the discussion of index size in Section 19.5 is *duplication*: the Web contains multiple copies of the same content. By some estimates, as many as 40% of the pages on the Web are duplicates of other pages. Many of these are legitimate copies; for instance, certain information repositories are mirrored simply to provide redundancy and access reliability. Search engines try to avoid indexing multiple copies of the same content, to keep down storage and processing overheads.

The simplest approach to detecting duplicates is to compute, for each web page, a *fingerprint* that is a succinct (say 64-bit) digest of the characters on that page. Then, whenever the fingerprints of two web pages are equal, we test whether the pages themselves are equal and if so declare one of them to be a duplicate copy of the other. This simplistic approach fails to capture a crucial and widespread phenomenon on the Web: *near duplication*. In many cases, the contents of one web page are identical to those of another except for a few characters – say, a notation showing the date and time at which the page was last modified. Even in such cases, we want to be able to declare the two pages to be close enough that we only index one copy. Short of exhaustively comparing all pairs of web pages, an infeasible task at the scale of billions of pages, how can we detect and filter out such near duplicates?

SHINGLING

We now describe a solution to the problem of detecting near-duplicate web pages. The answer lies in a technique known as *shingling*. Given a positive integer k and a sequence of terms in a document d, define the k-shingles of d to be the set of all consecutive sequences of k terms in d. As an example, consider the following text: a rose is a rose is a rose. The 4-shingles for this text ($k = 4$ is a typical value used in the detection of near-duplicate web pages) are a rose is a, rose is a rose and is a rose is. The first two of these shingles each occur twice in the text. Intuitively, two documents are near duplicates if the sets of shingles generated from them are nearly the same. We now make this intuition precise, then develop a method for efficiently computing and comparing the sets of shingles for all web pages.

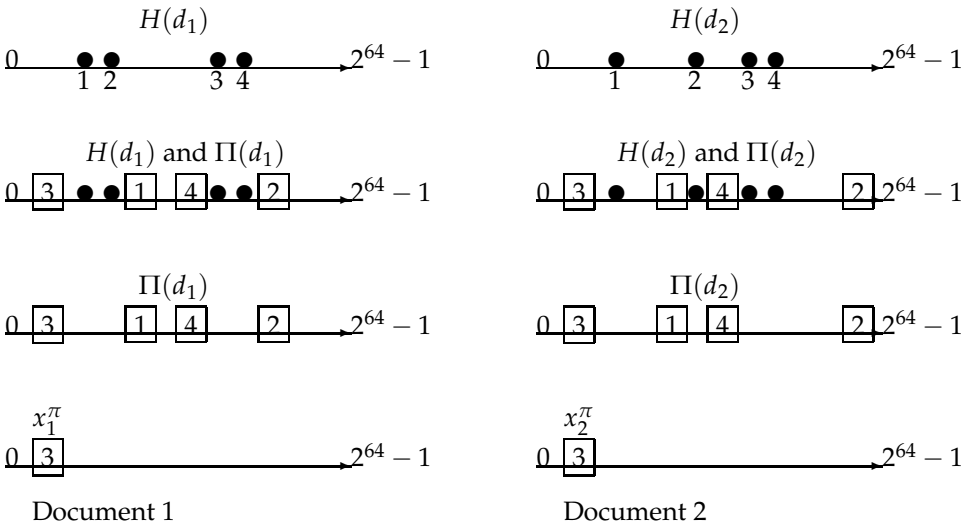
Let $S(d_j)$ denote the set of shingles of document d_j. Recall the Jaccard coefficient from page 61, which measures the degree of overlap between the sets $S(d_1)$ and $S(d_2)$ as $|S(d_1) \cap S(d_2)|/|S(d_1) \cup S(d_2)|$; denote this by $J(S(d_1), S(d_2))$. Our test for near duplication between d_1 and d_2 is to compute this Jaccard coefficient; if it exceeds a preset threshold (say, 0.9), we declare them near duplicates and eliminate one from indexing. However, this does not appear to have simplified matters: we still have to compute Jaccard coefficients pairwise.

To avoid this, we use a form of hashing. First, we map every shingle into a hash value over a large space, say 64 bits. For $j = 1, 2$, let $H(d_j)$ be the corresponding set of 64-bit hash values derived from $S(d_j)$. We now invoke the following trick to detect document pairs whose sets $H()$ have large Jaccard overlaps. Let π be a random permutation from the 64-bit integers to the 64-bit integers. Denote by $\Pi(d_j)$ the set of permuted hash values in $H(d_j)$; thus for each $h \in H(d_j)$, there is a corresponding value $\pi(h) \in \Pi(d_j)$.

Let x_j^π be the smallest integer in $\Pi(d_j)$. Then

Theorem 19.1.

$$J(S(d_1), S(d_2)) = P(x_1^\pi = x_2^\pi).$$



► **Figure 19.8** Illustration of shingle sketches. We see two documents going through four stages of shingle sketch computation. In the first step (top row), we apply a 64-bit hash to each shingle from each document to obtain $H(d_1)$ and $H(d_2)$ (circles). Next, we apply a random permutation Π to permute $H(d_1)$ and $H(d_2)$, obtaining $\Pi(d_1)$ and $\Pi(d_2)$ (squares). The third row shows only $\Pi(d_1)$ and $\Pi(d_2)$, while the bottom row shows the minimum values x_1^π and x_2^π for each document.

Proof. We give the proof in a slightly more general setting: consider a family of sets whose elements are drawn from a common universe. View the sets as columns of a matrix A, with one row for each element in the universe. The element $a_{ij} = 1$ if element i is present in the set S_j that the jth column represents.

Let Π be a random permutation of the rows of A; denote by $\Pi(S_j)$ the column that results from applying Π to the jth column. Finally, let x_j^π be the index of the first row in which the column $\Pi(S_j)$ has a 1. We then prove that for any two columns j_1, j_2,

$$P(x_{j_1}^\pi = x_{j_2}^\pi) = J(S_{j_1}, S_{j_2}).$$

If we can prove this, the theorem follows.

Consider two columns j_1, j_2 as shown in Figure 19.9. The ordered pairs of entries of S_{j_1} and S_{j_2} partition the rows into four types: those with 0's in both of these columns, those with a 0 in S_{j_1} and a 1 in S_{j_2}, those with a 1 in S_{j_1} and a 0 in S_{j_2}, and finally those with 1's in both of these columns. Indeed, the first four rows of Figure 19.9 exemplify all of these four types of rows.

S_{j_1}	S_{j_2}
0	1
1	0
1	1
0	0
1	1
0	1

▶ **Figure 19.9** Two sets S_{j_1} and S_{j_2}; their Jaccard coefficient is 2/5.

Denote by C_{00} the number of rows with 0's in both columns, C_{01} the second, C_{10} the third and C_{11} the fourth. Then,

$$J(S_{j_1}, S_{j_2}) = \frac{C_{11}}{C_{01} + C_{10} + C_{11}}. \tag{19.2}$$

To complete the proof by showing that the right-hand side of Equation (19.2) equals $P(x_{j_1}^{\pi} = x_{j_2}^{\pi})$, consider scanning columns j_1, j_2 in increasing row index until the first non-zero entry is found in either column. Because Π is a random permutation, the probability that this smallest row has a 1 in both columns is exactly the right-hand side of Equation (19.2). □

Thus, our test for the Jaccard coefficient of the shingle sets is probabilistic: we compare the computed values x_i^{π} from different documents. If a pair coincides, we have candidate near duplicates. Repeat the process independently for 200 random permutations π (a choice suggested in the literature). Call the set of the 200 resulting values of x_i^{π} the *sketch* $\psi(d_i)$ of d_i. We can then estimate the Jaccard coefficient for any pair of documents d_i, d_j to be $|\psi_i \cap \psi_j|/200$; if this exceeds a preset threshold, we declare that d_i and d_j are similar.

How can we quickly compute $|\psi_i \cap \psi_j|/200$ for all pairs i, j? Indeed, how do we represent all pairs of documents that are similar, without incurring a blowup that is quadratic in the number of documents? First, we use fingerprints to remove all but one copy of *identical* documents. We may also remove common HTML tags and integers from the shingle computation, to eliminate shingles that occur very commonly in documents without telling us anything about duplication. Next we use a *union-find* algorithm to create clusters that contain documents that are similar. To do this, we must accomplish a crucial step: going from the set of sketches to the set of pairs i, j such that d_i and d_j are similar.

To this end, we compute the number of shingles in common for any pair of documents whose sketches have any members in common. We begin with the list $< x_i^{\pi}, d_i >$ sorted by x_i^{π} pairs. For each x_i^{π}, we can now generate

all pairs i, j for which x_i^π is present in both their sketches. From these we can compute, for each pair i, j with non-zero sketch overlap, a count of the number of x_i^π values they have in common. By applying a preset threshold, we know which pairs i, j have heavily overlapping sketches. For instance, if the threshold were 80%, we would need the count to be at least 160 for any i, j. As we identify such pairs, we run the union-find to group documents into near-duplicate "syntactic clusters". This is essentially a variant of the single-link clustering algorithm introduced in Section 17.2 (page 382).

One final trick cuts down the space needed in the computation of $|\psi_i \cap \psi_j|/200$ for pairs i, j, which in principle could still demand space quadratic in the number of documents. To remove from consideration those pairs i, j whose sketches have few shingles in common, we preprocess the sketch for each document as follows: sort the x_i^π in the sketch, then shingle this sorted sequence to generate a set of *super-shingles* for each document. If two documents have a super-shingle in common, we proceed to compute the precise value of $|\psi_i \cap \psi_j|/200$. This again is a heuristic but can be highly effective in cutting down the number of i, j pairs for which we accumulate the sketch overlap counts.

?

Exercise 19.8

Web search engines A and B each crawl a random subset of the same size of the Web. Some of the pages crawled are duplicates – exact textual copies of each other at different URLs. Assume that duplicates are distributed uniformly amongst the pages crawled by A and B. Further, assume that a duplicate is a page that has exactly two copies – no pages have more than two copies. A indexes pages without duplicate elimination whereas B indexes only one copy of each duplicate page. The two random subsets have the same size before duplicate elimination. If, 45% of A's indexed URLs are present in B's index, while 50% of B's indexed URLs are present in A's index, what fraction of the Web consists of pages that do not have a duplicate?

Exercise 19.9

Instead of using the process depicted in Figure 19.8, consider instead the following process for estimating the Jaccard coefficient of the overlap between two sets S_1 and S_2. We pick a random subset of the elements of the universe from which S_1 and S_2 are drawn; this corresponds to picking a random subset of the rows of the matrix A in the proof. We exhaustively compute the Jaccard coefficient of these random subsets. Why is this estimate an unbiased estimator of the Jaccard coefficient for S_1 and S_2?

Exercise 19.10

Explain why this estimator would be very difficult to use in practice.