Algorithms in the Real World: Nov 25, 2002

Lecturer: G. Blelloch Scribes: A. Xhafa

In today's lecture we will talk about syntactic clustering of the web and min-wise independent permutations.

I. Problem Statement

The Web has undergone exponential growth since its birth, and this expansion has generated a number of problems [1], however, in today's lecture we will concentrate in two of them:

1) Many document pages are the same

2) Many document pages are almost the same

The first problem occurs if for example, my web-site is accessed from different links. Two or more document pages have the same content except for some minor modification [1]. The second problem is more common and can occur in news stories where one page is contained inside another page; people have different web-sites with some little extra changes so that their web-site can not be removed; plagiarism where we want to know if a student copy a paper from the web and just handed in; mirror sites where the links within the sites might be different, etc. (see [1]).

The objective is to remove the identical or almost identical documents. To remove identical documents the idea is simple: use a hash function for the document; hence, if you have seen the document before, you also have seen its hash function. Therefore, you do not post the document. Removing almost identical documents is a little more complex and the hash function idea that we mentioned for the removal of identical documents will not quite work.

II. Goals of the Algorithm

To take care of the above problems, the algorithm that we want to come up with has to achieve the following goals:

- Find pages for which either most of the text is the same one or the document is contained in another: this is called Syntactic Clustering

- Constraints:
  - Keep a small amount of information per document (like MD5), so that it is easy for one to search (e.g., 1000 bytes per document independent of the size of the document).
  - We want the comparison be linear in summary size
- It is not required to be robust against omniscient adversary (in fact, this is very difficult one and it is related to water-marking which theoretically is very difficult to do). Usually, if you know how the algorithm works, you can get around it, and this is the reason why Google does not publish details how ranking works.
  - approach: use vocabulary sensitive hashing
  - cheating code: this code was designed to check whether the undergraduate student were cheating in their homeworks. The code was written by Dr. Sleator and it works as follows: it collects statistics of ifs, less signs, etc., and does the summary. The closest Hamming distance will show whether two students have cheated or not (one way to get around it is to flip the statements). In general, this approach would not work on the web.
  - fingerprinting of documents where one picks some random words from the document and use as them as fingerprints (after hashing them). It can happen that you end up with the same fingerprints. To circumvent this, use Blum filters where each term corresponds to a set of bits and you take the XOR of them.

III. PROPOSED SOLUTION

In order to find pages for which either most of the text is the same one or the document is contained in another, one can use *shingling* or *tiling* approach[1]. The idea is as follows: view each document page as a sequence of words (summary words). Then associate with every document page D a set of subsequences of words $S(D, w)$.Thus, a shingle is a contiguous subsequence contained in D. Let us use a more rigorous way by defining the concept of almost the same via the mathematical concepts of *resemblance* and *containment* as in[1].

The *resemblance* of two documents A and B is a number between 0 and 1, such that when the resemblance is close to 1 it is likely that the documents are almost the same. In the same way, the *containment* of A and B is a number between 0 and 1 such that, when close to 1, indicates that A is almost contained within B. Thus, one needs to keep a few hundred of bytes for each documents, and therefore, the resemblance or containment of the corresponding documents can be computed in linear time in the summary size [1].

Consider pages A and B that have c summary words in common, while the union of their summary words is d (see Figure 1). Then, the resemblance of A and B, denoted as $R(A, B)$ is given as:

$$R(A, B) = \frac{|S(A) \bigcap S(B)|}{|S(A) \bigcup S(B)|} = \frac{c}{d} \tag{1}$$

Thus, if the resemblance is 1, that means exact match and 0 means no resemblance at all.

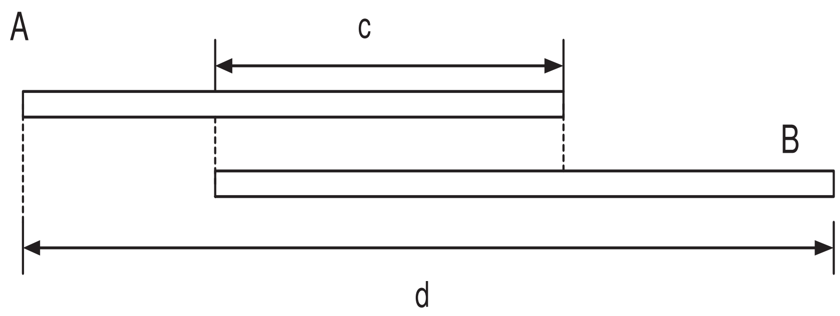

Fig. 1. Resemblance example.

Consider the following idea: Think that each shingle is an index/integer number. Assume that we have summarized 100 shingles and we want to associate them with the smallest 100 integers and store their hash functions. If two documents are the same, that means then the hashed integers will be the same. However, one could use a slightly different approach: take each of the shingles and randomly map it to a fixed integer and take the minimum of this mapping. Perform a hashing operation on this minimum. Then, this will differentiate between the documents. Since in this approach we mentioned hashing and random mapping, in the next sections we are going to talk briefly about universal class of hash functions and min-wise independent permutations.

IV. UNIVERSAL CLASS OF HASH FUNCTIONS

In various adaptive hashing schemes it is assumed that a hash function with certain predescribed properties can be found in constant expected time [2]. However, this might not be the case if the search is limited to a smaller set of functions. This takes us to universal hashing which is defined below.

Let H be a family class of hash functions; h a particular element of it, and the range of hash functions is $1...m$. We say that the function is universal (*weakly universal*) if h is chosen uniformly at random from H then one has:

$$Pr(h(x_1) \approx h(x_2)) \leq \frac{1}{m} \tag{2}$$

where x_1 and x_2 are distinct elements. This classes are extremely practical. Consider for example:

$$h_{ab} = ((ax + b) mod\, p) mod\, m$$
$$a = 1...p - 1$$
$$b = 0...p - 1$$

The family of hash functions H is said to be (strongly) universal or pair-wise independent if for any pair of distinct elements x_1, x_2 and arbitrary y_1, y_2, the following holds [2]:

$$Pr(h(x_1) = y_1\; and\; h(x_2) = y_2) = \frac{1}{m^2} \tag{3}$$

V. MIN-WISE INDEPENDENT CLASS OF PERMUTATIONS

Let us now define min-wise independent class of permutations. Let S_n be the set of all permutations of $[n]$, Π a family of permutation functions, X be the set such that $X \subseteq \{1...n\}$, and $x \in X$. Then, Π is min-wise independent if for any set X and any $x \in X$, when π is chosen uniformly at random in Π one has the following [2]:

$$Pr(min\{\pi(X)\} = \pi(x)) = \frac{1}{|X|} \tag{4}$$

What this is means is that all the elements of any fixed set X have an equal chance to become the minimum element of the image of X under π.

Let us now go back to our goal of finding document pages for which most of the text is the same or the page is contained in another one. Eqn(4) can be re-written as follows:

$$Pr(min\{\pi(S(A))\} = \pi(S(B))) = R(A,B) \tag{5}$$

Thus, one can choose 100 independent random permutations and for each page store the list of the minimums; e.g., list of the minimums for $S(A)$: (min$\{\pi_1$(S(A))$\}$, min$\{\pi_2(S(A))\}$,...). Then, one can estimate $R(A,B)$ by finding how many common elements are in the list of the minimums in A and B (see Figure 2). For the minimum to be on the c range, its probability is c/n, which is the resemblance R(A,B). This probability in fact is very small.

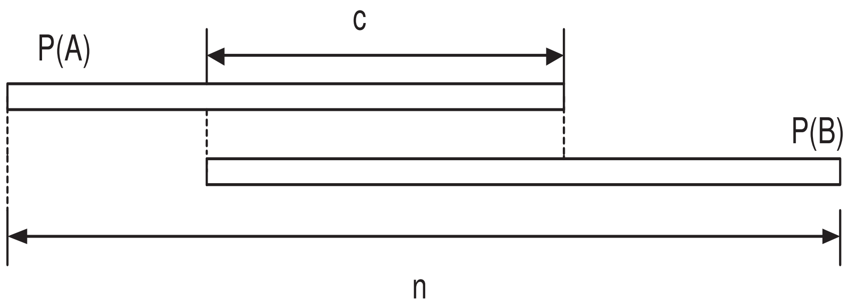

Fig. 2. Permutations example (P(A) and P(B) means permutations).

However, in reality it is impossible to choose the permutations uniformly at random in S_n. Therefore, we deal with approximately min-wise independent class of permutations, which satisfies the following:

$$|Pr(min\{\pi(X)\} = \pi(x)) - \frac{1}{|X|}| \leq \frac{\epsilon}{|X|} \tag{6}$$

where X is any set such that $X \subseteq$[n], $x \in X$, π is chosen at random, and ϵ is the relative error.

REFERENCES

[1] A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher, "Syntactic clustering of the web", SRC Technical Note 1997-015, July25, 1997, available at http://ftp.digital.com/pub/DEC/SRC/technical-notes/SRC-1997-015-html.

[2] A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher, "Min-wise independent permutations", in Proc. of 30th Annual ACM Symp. Theo. Comp., pp. 327-336, May 1998.