Identifying and Filtering Near-Duplicate Documents

Andrei Z. Broder*

AltaVista Company, San Mateo, CA 94402, USA
andrei.broder@av.com

Abstract. The mathematical concept of document resemblance captures well the informal notion of syntactic similarity. The resemblance can be estimated using a fixed size "sketch" for each document. For a large collection of documents (say hundreds of millions) the size of this sketch is of the order of a few hundred bytes per document.
However, for efficient large scale web indexing it is not necessary to determine the actual resemblance value: it suffices to determine whether newly encountered documents are duplicates or near-duplicates of documents already indexed. In other words, it suffices to determine whether the resemblance is above a certain threshold. In this talk we show how this determination can be made using a "sample" of less than 50 bytes per document.
The basic approach for computing resemblance has two aspects: first, resemblance is expressed as a set (of strings) intersection problem, and second, the relative size of intersections is evaluated by a process of random sampling that can be done independently for each document. The process of estimating the relative size of intersection of sets and the threshold test discussed above can be applied to arbitrary sets, and thus might be of independent interest.
The algorithm for filtering near-duplicate documents discussed here has been successfully implemented and has been used for the last three years in the context of the AltaVista search engine.

1 Introduction

A Communist era joke in Russia goes like this: Leonid Brezhnev (the Party leader) wanted to get rid of the Premier, Aleksey Kosygin. (In fact he did, in 1980.) So Brezhnev, went to Kosygin and said: "My dear friend and war comrade Aleksey, I had very disturbing news: I just found out that you are Jewish: I have no choice, I must ask you to resign." Kosygin, in total shock says: "But Leonid, as you know very well I am not Jewish!"; then Brezhnev says: "Well, Aleksey, then think about it..."

* Most of this work was done while the author was at Compaq's System Research Center in Palo Alto. A preliminary version of this work was presented (but not published) at the "Fun with Algorithms" conference, Isola d'Elba, 1998.

R. Giancarlo and D. Sankoff (Eds.): CPM 2000, LNCS 1848, pp. 1–10, 2000.

What this has to do with near-duplicate documents? In mid 1995, the AltaVista web search engine was built at the Digital research labs in Palo Alto (see [10]). Soon after the first internal prototype was deployed, a colleague, Chuck Thacker, came to me and said: "I really like AltaVista, but it is very annoying that often half the first page of answers is just the same document in many variants." "I know" said I. "Well," said Chuck, "you did a lot of work on fingerprinting documents; can you make up a fingerprinting scheme such that two documents that are near-duplicate get the same fingerprint?" I was of course indignant: "No way!! You miss the idea of fingerprints completely: fingerprints are such that with high probability two distinct documents will have different fingerprints, no matter how little they differ! Similar documents getting the same fingerprint is entirely against their purpose." So, of course, Chuck said: "Well, then think about it..." ...and as usual, Chuck was right.

Eventually I found found a solution to this problem, based on a mathematical notion called *resemblance* [4]. Surprisingly, fingerprints play an essential role in it.

The resemblance measures whether two (web) documents are roughly the same, that is, they have the same content except for modifications such as formatting, minor corrections, capitalization, web-master signature, logo, etc. The resemblance is a number between 0 and 1, defined precisely below, such that when the resemblance is close to 1 it is likely that the documents are roughly the same. To compute the resemblance of two documents it suffices to keep for each document a "sketch" of a few (three to eight) hundred bytes consisting of a collection of fingerprints of "shingles" (contiguous subsequences of words, sometimes called "q-grams"). The sketches can be computed fairly fast (linear in the size of the documents) and given two sketches the resemblance of the corresponding documents can be computed in linear time in the size of the sketches. Furthermore, clustering a collection of m documents into sets of closely resembling documents can be done in time proportional to $m \log m$ rather than m^2.

This first use of this idea was in a joint work with Steve Glassman, Mark Manasse, and Geoffrey Zweig to cluster a collection of over 30,000,000 documents into sets of closely resembling documents (above 50% resemblance). The documents were retrieved from a month long "full" crawl of the World Wide Web performed by AltaVista in April 96. (See [7].) (It is amusing to note that three years later, by mid 1999, AltaVista was crawling well over 20 million pages *daily*.)

Besides fingerprints, another essential ingredient in the computation of resemblance is a pseudo-random permutation of a large set, typically the set $[0, \ldots, 2^{64} - 1]$. In turns out that to achieve the desired result, the permutation must be drawn from a *min-wise independent* family of permutations. The concept of min-wise independence is in the same vein as the well known concept of pair-wise independence, and has many interesting properties. Moses Charikar, Alan Frieze, Michael Mitzenmacher, and I studied this concept in a paper [5].

The World Wide Web continues to expand at a tremendous rate. It is estimated that the number of pages doubles roughly every nine moths to year [2,14].

Hence the problem of eliminating duplicates and near-duplicates from the index is extremely important. The fraction of the total WWW collection consisting of duplicates and near-duplicates has been estimated at 30 to 45%. (See [7] and [13].) These documents arise innocently (e.g. local copies of popular documents, mirroring), maliciously (e.g., "spammers" and "robot traps"), and erroneously (crawler or server mistakes). In any case they represent a serious problem for indexing software for two main reasons: first, indexing of duplicates wastes expensive resources and second, users are seldom interested in seeing documents that are "roughly the same" in response to their queries.

However, when applying the sketch computation algorithm to the entire corpus indexed by AltaVista then even the modest storage costs described above become prohibitive. On the other hand, we are interested only whether the resemblance is above a very high threshold; the actual value of the resemblance does not matter.

This paper describes how to apply further processing to the sketches mentioned above to construct for each document a short vector of "features." With high probability, two documents share more than a certain number of features if and only if their resemblance is very high. For instance, using 6 features of 8 bytes, that is, 48 bytes/document, for a set of 200,000,000 documents:

- The probability that two documents that have resemblance greater than 97.5% *do not share* at least two features is less than 0.01. The probability that two documents that have resemblance greater than 99% *do not share* at least two features is less than 0.00022.
- The probability that two documents that have resemblance less than 77% *do share* two or more features is less than 0.01 The probability that two documents that have resemblance less than 50% share two or more features is less than 0.6×10^{-7}.

Thus the feature based mechanism for near-duplicate detection has excellent filtering properties. The probability of acceptance for this example (that is more than 2 common features) as a function of resemblance is graphed in Figure 1 on a linear scale and in Figure 2 on a logarithmic scale.

2 Preliminaries

We start by reviewing some concepts and algorithms described in more detail in [4] and [7].

The basic approach for computing resemblance has two aspects: First, resemblance is expressed as a set intersection problem, and second, the relative size of intersections is evaluated by a process of random sampling that can be done independently for each document. (The process of estimating the relative size of intersection of sets can be applied to arbitrary sets.)

The reduction to a set intersection problem is done via a process called *shingling*. Via shingling each document D gets an associated set S_D. This is done as follows:

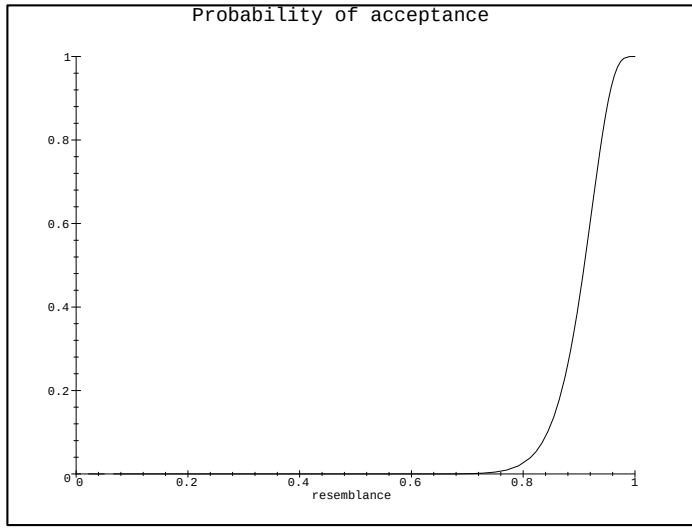

Fig. 1. The graph of $P_{6,14,2}(x)$ – linear scale

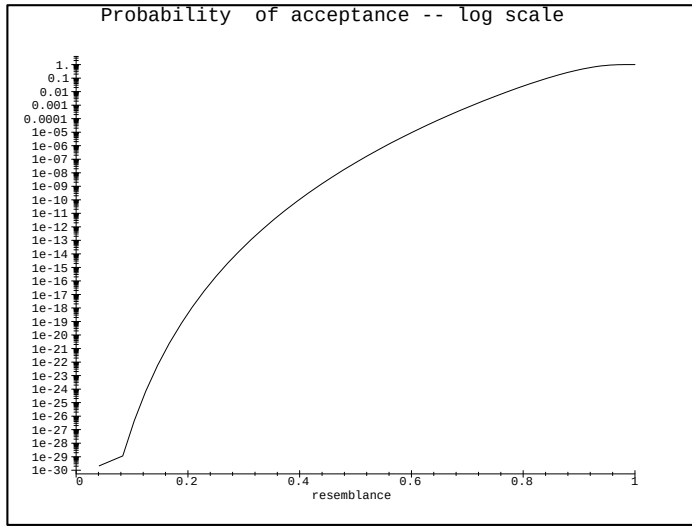

Fig. 2. The graph of $P_{6,14,2}(x)$ – logarithmic scale

We view each document as a sequence of tokens. We can take tokens to be letters, or words, or lines. We assume that we have a parser program that takes an arbitrary document and reduces it to a canonical sequence of tokens. (Here "canonical" means that any two documents that differ only in formatting or other information that we chose to ignore, for instance punctuation, formatting commands, capitalization, and so on, will be reduced to the same sequence.) So from now on a document means a canonical sequence of tokens.

A contiguous subsequence of w tokens contained in D is called a *shingle*. A shingle of length q is also known as a *q-gram*, particularly when the tokens are alphabet letters. Given a document D we can associate to it its *w-shingling* defined as the set of all shingles of size w contained in D. So for instance the 4-shingling of

$$(\texttt{a,rose,is,a,rose,is,a,rose})$$

is the set

$$\{(\texttt{a,rose,is,a}), (\texttt{rose,is,a,rose}), (\texttt{is,a,rose,is})\}$$

(It is possible to use alternative definitions, based on multisets. See [4] for details.)

Rather than deal with shingles directly, it is more convenient to associate to each shingle a numeric uid (unique id). This done by *fingerprinting* the shingle. (Fingerprints are short tags for larger objects. They have the property that if two fingerprints are different then the corresponding objects are certainly different and there is only a small probability that two different objects have the same fingerprint. This probability is typically exponentially small in the length of the fingerprint.)

For reasons explained in [4] it is particularly advantageous to use Rabin fingerprints [15] that have a very fast software implementation [3]. Rabin fingerprints are based on polynomial arithmetic and can be constructed in any length. It is important to choose the length of the fingerprints so that the probability of collisions (two distinct shingles getting the same fingerprint) is sufficiently low. (More about this below.) In practice 64 bits Rabin fingerprints are sufficient.

Hence from now on we associate to each document D a set of numbers S_D that is the result of fingerprinting the set of shingles in D. Note that the size of S_D is about equal to the number of words in D and thus storing S_D on-line for every document in a large collection is infeasible.

The *resemblance* $r(A, B)$ of two documents, A and B, is defined as

$$r(A, B) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|}.$$

Experiments seem to indicate that high resemblance (that is, close to 1) captures well the informal notion of "near-duplicate" or "roughly the same". (There are analyses that relate the "q-gram distance" to the edit-distance – see [16].)

Our approach to determining syntactic similarity is related to the sampling approach developed independently by Heintze [8], though there are differences

in detail and in the precise definition of the measures used. Related sampling mechanisms for determining similarity were also developed by Manber [9] and within the Stanford SCAM project [1,11,12].

To compute the resemblance of two documents it suffices to keep for each document a relatively small, fixed size *sketch*. The sketches can be computed fairly fast (linear in the size of the documents) and given two sketches the resemblance of the corresponding documents can be computed in linear time in the size of the sketches.

This is done as follows. Assume that for all documents of interest $S_D \subseteq \{0, \ldots, n-1\} \stackrel{def}{=} [n]$. (As noted, in practice $n = 2^{64}$.) Let π be chosen uniformly at random over S_n, the set of permutations of $[n]$. Then

$$\Pr\big(\min\{\pi(S_A)\} = \min\{\pi(S_B)\}\big) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|} = r(A, B). \tag{1}$$

Proof. Since π is chosen uniformly at random, for any set $X \subseteq [n]$ and any $x \in X$, we have

$$\Pr\big(\min\{\pi(X)\} = \pi(x)\big) = \frac{1}{|X|}. \tag{2}$$

In other words 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 α be the smallest image in $\pi(S_A \cup S_B)$. Then $\min\{\pi(S_A)\} = \min\{\pi(S_B)\}$, if and only if α is the image of an element in $S_A \cap S_B$. Hence

$$\begin{aligned}\Pr\big(\min\{\pi(S_A)\} = \min\{\pi(S_B)\}\big) &= \Pr(\pi^{-1}(\alpha) \in S_A \cap S_B) \\ &= \frac{|S_A \cap S_B|}{|S_A \cup S_B|} = r_w(A, B).\end{aligned}$$

Hence, we can choose, once and for all, a set of t independent random permutations $\pi_1, \ldots, \pi_t$. (For instance we can take $t = 100$.) For each document D, we store a *sketch*, which is the list

$$\bar{S}_A = (\min\{\pi_1(S_A)\}, \min\{\pi_2(S_A)\}, \ldots, \min\{\pi_t(S_A)\}).$$

Then we can readily estimate the resemblance of A and B by computing how many corresponding elements in $\bar{S}_A$ and $\bar{S}_B$ are equal. (In [4] it is shown that in fact we can use a single random permutation, store the t smallest elements of its image, and then merge-sort the sketches. However for the purposes of this paper independent permutations are necessary.)

In practice, we have to deal with the fact it is impossible to choose and represent π uniformly at random in S_n for large n. We are thus led to consider smaller families of permutations that still satisfy the *min-wise independence condition* given by equation (2), since min-wise independence is necessary and sufficient for equation (1) to hold. This is further explored in [5] where it is shown that random linear transformations are likely to suffice in practice. See also [6] for an alternative implementation. We will ignore this issue in this paper.

So far we have seen how to estimate the resemblance of a pair of documents. For this purpose the shingle fingerprints can be quite short since collisions have only a modest influence on our estimate if we first apply a random permutation to the shingles and then fingerprint the minimum value.

However sketches allow us to group a collection of m documents into sets of closely resembling documents in time proportional to $m \log m$ rather than m^2, assuming that the clusters are well separated which is the practical case.

We perform the clustering algorithm in four phases. In the first phase, we calculate a sketch for every document as explained. This step is linear in the total length of the documents.

To simplify the exposition of the next three phases we'll say temporarily that each sketch is composed of shingles, rather than images of the fingerprint of shingles under random permutations of $[n]$.

In the second phase, we produce a list of all the shingles and the documents they appear in, sorted by shingle value. To do this, the sketch for each document is expanded into a list of ⟨shingle value, document ID⟩ pairs. We simply sort this list. This step takes time $O(m \log m)$ where m is the number of documents.

In the third phase, we generate a list of all the pairs of documents that share any shingles, along with the number of shingles they have in common. To do this, we take the file of sorted ⟨shingle, ID⟩ pairs and expand it into a list of ⟨ID, ID, count of common shingles⟩ triplets by taking each shingle that appears in multiple documents and generating the complete set of ⟨ID, ID, 1⟩ triplets for that shingle. We then apply a merge-sort procedure (adding the counts for matching ID - ID pairs) to produce a single file of all ⟨ID, ID, count⟩ triplets sorted by the first document ID. This phase requires the greatest amount of disk space because the initial expansion of the document ID triplets is quadratic in the number of documents sharing a shingle, and initially produces many triplets with a count of 1. Because of this fact we must choose the length of the shingle fingerprints so that the number of collisions is small. To ensure this we can take it to be say $2 \log_2 m + 20$. In practice 64 bits fingerprints suffice.

In the final phase, we produce the complete clustering. We examine each ⟨ID, ID, count⟩ triplet and decide if the document pair exceeds our threshold for resemblance. If it does, we add a link between the two documents in a union-find algorithm. The connected components output by the union-find algorithm form the final clusters.

3 Filtering Near-Duplicates

Consider two documents, A and B, that have resemblance ρ. If ρ is close to 1, then almost all the elements of $\bar{S}_A$ and $\bar{S}_B$ will be pairwise equal. The idea of duplicate filtering is to divide every sketch into k groups of s elements each. The probability that all the elements of a group are pair-wise equal is simply ρ^s and the probability that two sketches have r or more equal groups is

$$P_{k,s,r} = \sum_{r \le i \le k} \binom{k}{i} \rho^{s \cdot i} (1 - \rho^s)^{k-i}.$$

The remarkable fact is that for suitable choices of $[k, s, r]$ the polynomial $P_{k,s,r}$ behaves as a very sharp high-band pass filter even for small values of k. For instance Figure 1 graphs $P_{6,14,2}(x)$ on a linear scale and Figure 2 graphs it on a logarithmic scale. The sharp drop-off is obvious.

To use this fact, we first compute for each document D the sketch $\bar{S}_D$ as before, using $k \cdot s$ independent permutations. (We can now be arbitrarily generous with the length of the fingerprints used to create shingle uid's; however 64 bits are plenty for our situation.) We then split $\bar{S}_D$ into k groups of s elements and fingerprint each group. (To avoid dependencies, we use a different irreducible polynomial for these fingerprints.) We can also concatenate to each group a group id number before fingerprinting.

Now all we need to store for each document is these k fingerprints, called "features". Because fingerprints could collide the probability that two features are equal is

$$\rho^s + p_f,$$

where p_f is the collision probability. This would indicate that it suffices to use fingerprints long enough to so that p_f is less than say 10^{-6}. However, when applying the filtering mechanism to a large collection of documents, we again use the clustering process described above, and hence we must avoid spurious sharing of features. Nevertheless, for our problem 64 bits fingerprints are again sufficient.

It is particularly convenient, if possible, to choose the threshold r to be 1 or 2. If $r = 2$ then the third phase of the merging process becomes much simpler since we don't need to keep track of how many features are shared by various pairs of documents: we simply keep a list of pairs known to share at least one feature. As soon as we discover that one of these pairs shares a second feature, we know that with high probability the two documents are near-duplicates, and thus one of them can be removed from further consideration. If $r = 1$ the third phase becomes moot. In general it is possible to avoid the third phase if we again group every r features into a super-feature, but this forces the number of features per document to become $\binom{k}{r}$.

4 Choosing the Parameters

As often the case in filter design, choosing the parameters is half science, half black magic. It is useful to start from a target threshold resemblance ρ_0. Ideally

$$P_{k,s,r}(\rho) = \begin{cases} 1, \text{ for } \rho \geq \rho_0; \\ 0, \text{ otherwise.} \end{cases}$$

Clearly, once s is chosen, r should be approximately $k \cdot \rho_0^s$ and the larger k (and r) the sharper the filter. (Of course, we are restricted to integral values for k, s, and r.)

If we make the (unrealistic) assumption that resemblance is uniformly distributed between 0 and 1 within the set of pairs of documents to be checked,

then the total error is proportional to

$$\int_0^{\rho_0} P_{k,s,r}(x)\,dx + \int_{\rho_0}^1 \big(1 - P_{k,s,r}(x)\big)\,dx$$

Differentiating with respect to ρ_0 we obtain that this is minimized when $P(\rho_0) = 1/2$. To continue with our example we have $P_{6,14,2}(x) = 1/2$ for $x = 0.909\ldots$.

A different approach is to chose s so that the slope of x^s at $x = \rho_0$ is maximized. This happens when

$$\frac{\partial}{\partial s}\left(s\rho_0^{s-1}\right) = 0 \tag{3}$$

or $s = 1/\ln(1/\rho_0)$. For $s = 14$ the value that satisfies (3) is $\rho_0 = 0.931\ldots$.

In practice these ideas give only a starting point for the search for a filter that provides the required trade-offs between error bounds, time, and space. It is necessary to graph the filter and do experimental determinations.

5 Conclusion

We have presented a method that can eliminate near-duplicate documents from a collection of hundreds of millions of documents by computing independently for each document a vector of features less than 50 bytes long and comparing only these vectors rather than entire documents. The entire processing takes time $O(m \log m)$ where m is the size of the collection. The algorithm described here has been successfully implemented and is in current use in the context of the AltaVista search engine.

Acknowledgments

I wish to thank Chuck Thacker who challenged me to find an efficient algorithm for filtering near-duplicates. Some essential ideas behind the resemblance definition and computation were developed in conversations with Greg Nelson. The prototype implementation for AltaVista was done in collaboration with Mike Burrows and Mark Manasse.

References

1. S. Brin, J. Davis, H. García-Molina. Copy Detection Mechanisms for Digital Documents. *Proceedings of the ACM SIGMOD Annual Conference*, May 1995.
2. K. Bharat and A. Z. Broder. A Technique for Measuring the Relative Size and Overlap of Public Web Search Engines. In *Proceedings of Seventh International World Wide Web Conference*, pages 379–388, 1998.
3. A. Z. Broder. Some applications of Rabin's fingerprinting method. In R. Capocelli, A. De Santis, and U. Vaccaro, editors, *Sequences II: Methods in Communications, Security, and Computer Science*, pages 143–152. Springer-Verlag, 1993.

4. A. Z. Broder. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of Sequences 1997*, pages 21–29. IEEE Computer Society, 1997.
5. A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-Wise Independent Permutations. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 327–336, 1998.
6. A. Z. Broder and U. Feige. Min-Wise versus Linear Independence. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 147–154, 2000.
7. A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the Web. In *Proceedings of the Sixth International World Wide Web Conference*, pages 391–404, 1997.
8. N. Heintze. Scalable Document Fingerprinting. *Proceedings of the Second USENIX Workshop on Electronic Commerce*, pages 191–200, 1996.
9. U. Manber. Finding similar files in a large file system. *Proceedings of the Winter 1994 USENIX Conference*,pages 1–10, 1994.
10. R. Seltzer, E. J. Ray, and D. S. Ray. *The AltaVista Search Revolution: How to Find Anything on the Internet.* McGraw-Hill, 1996.
11. N. Shivakumar, H. García-Molina. SCAM: A Copy Detection Mechanism for Digital Documents. *Proceedings of the 2nd International Conference on Theory and Practice of Digital Libraries*, 1995.
12. N. Shivakumar and H. García-Molina. Building a Scalable and Accurate Copy Detection Mechanism. *Proceedings of the 3nd International Conference on Theory and Practice of Digital Libraries*, 1996.
13. N. Shivakumar and H. García-Molina. Finding near-replicas of documents on the web. In *Proceedings of Workshop on Web Databases (WebDB'98)*, March 1998.
14. Z. Smith. The Truth About the Web: Crawling Towards Eternity, *Web Techniques Magazine*, May 1997.
15. M. O. Rabin. Fingerprinting by random polynomials. Center for Research in Computing Technology, Harvard University, Report TR-15-81, 1981.
16. E. Ukkonen. Approximate string-matching distance and the q-gram distance. In R. Capocelli, A. De Santis, and U. Vaccaro, editors, *Sequences II: Methods in Communications, Security, and Computer Science*, pages 300–312. Springer-Verlag, 1993.