On the resemblance and containment of documents

Andrei Z. Broder
DIGITAL Systems Research Center
130 Lytton Avenue, Palo Alto, CA 94301, USA
broder@pa.dec.com

Abstract

Given two documents A and B we define two mathematical notions: their *resemblance* $r(A, B)$ and their *containment* $c(A, B)$ that seem to capture well the informal notions of "roughly the same" and "roughly contained."

The basic idea is to reduce these issues to set intersection problems that can be easily evaluated by a process of random sampling that can be done independently for each document. Furthermore, the resemblance can be evaluated using a fixed size sample for each document.

This paper discusses the mathematical properties of these measures and the efficient implementation of the sampling process using Rabin fingerprints.

1 Introduction

The on-line information explosion, and in particular the World Wide Web, has led a proliferation of documents that are identical or almost identical. In many situations it is necessary to determine whether two documents are "roughly the same" or "roughly contained" in each other.

These informal concepts do not seem to be well captured by any of the standard distances defined on strings (Hamming, Levenshtein, etc.). Furthermore the computation of these distances usually requires the pairwise comparison of entire documents. For a very large collection of documents this is not feasible, and a sampling mechanism per document is necessary.

To attack this problem we use two mathematical concepts *resemblance* and *containment* defined precisely below.

The *resemblance* $r(A, B)$ 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 roughly the same. Similarly, the *containment* $c(A, B)$ of A in B is a number between 0 and 1 that, when close to 1, indicates that A is roughly contained within B. The resemblance has the additional property that $d(A, B) = 1 - r(A, B)$, is a metric (obeys the triangle inequality), which is useful for the design of algorithms intended to cluster a collection of documents into sets of closely resembling documents.

To compute the resemblance and/or the containment of two documents it suffices to keep for each document a relatively small *sketch*. The sketches can be computed fairly fast (linear in the size of the documents) and given two sketches the resemblance or the containment of the corresponding documents can be computed in linear time in the size of the sketches. For computing resemblance, it suffices to keep a fixed size sketch. For computing containment, we need a sketch proportional to the size of the underlying document; however as it will be explained this problem can be finessed at the cost of a loss of precision.

Our approach to determining syntactic similarity is related to the sampling approach developed independently by Heintze [6], 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 [7] and within the Stanford SCAM project [2, 8, 9].

We tested the ideas discussed above by building a clustering of a collection of over 30,000,000 documents into sets of closely resembling document. The documents were retrieved from a walk of the World Wide Web performed by the Alta Vista search engine. The total input data was over 150 Gbytes. We calculated our clusters based on a 50% resemblance. We found 3.6 million clusters containing a total of 12.3 million documents. Of these, 2.1 million clusters contained only identical documents (5.3 million documents). The remaining 1.5 million clusters contained 7 million documents (a mixture of exact duplicates and similar). For further details see [5].

The basic approach has two aspects: First, resemblance and containment are expressed as set intersection problems (this is explained in Section 2) and second, the relative size of these intersections is evaluated by a process of random sampling that can be done independently for each document (this is explained in Section 3). This process of estimating the relative size of intersection of sets can be applied to arbitrary sets, and thus might be of independent interest.

2 Definitions as set intersection problems

We view each document as a sequence of tokens. We can take tokens to be letters, or words, or lines. From a mathematical point of view all what we need is for the set of tokens to be countable.

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 reduces to the same sequence.) So from now on a document means a canonical sequence of tokens.

Our immediate goal is to associate to every document D a *set* of subsequences of tokens $S(D, w)$ where w is a parameter defined below.

A contiguous subsequence contained in D is called a *shingle*. Given a document D we can associate to it its w-*shingling* defined as the bag (multiset) 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 bag

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

There are now two ways to proceed. Option A keeps more information about the document; option B is more efficient.

Option A. By labelling each element of a w-shingling with its occurrence number we obtain its *labelled w-shingling*. Option A is to define $S(D, w)$ as this set. Continuing the example above this would be the *set*

$$\{(\texttt{a,rose,is,a,1}),(\texttt{rose,is,a,rose,1}),(\texttt{is,a,rose,is,1}),\\ (\texttt{a,rose,is,a,2}),(\texttt{rose,is,a,rose,2})\}.$$

Option B. This is to take $S(D, w)$ to be the set of shingles in D. For the example above this is the set

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

Once we fixed a shingle size and one of the options above, the resemblance r of two documents A and B is defined as

$$r_w(A, B) = \frac{|S(A, w) \cap S(B, w)|}{|S(A, w) \cup S(B, w)|},$$

and the containment of A in B is defined as

$$c_w(A, B) = \frac{|S(A, w) \cap S(B, w)|}{|S(A, w)|}.$$

Hence the resemblance is some number between 0 and 1, and $r(A, A) = 1$, in another words A resembles itself 100%, for any size. Similarly, the containment is some number between 0 and 1 and if A is a contiguous subsequence of B then $c(A, B) = 1$. Furthermore computing the resemblance and the containment boils down to evaluating the relative size of set intersections.

As an example if

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

and

$$B = (\texttt{a,rose,is,a,flower,which,is,a,rose})$$

then under the first option A resembles B 70% for shingle size 1, 50% for size 2, 30% for size 3, etc. Under the second option, A resembles B 60% for size 1, 50% for size 2, 42.85% for size 3, etc.

Notice however that even under the first option, if A resembles B 100% for shingle size 1, it only means that B is an arbitrary permutation of A; for larger sizes if A resembles B 100% it is still the case that B could be a permutation of A but only certain permutations are possible: for instance (`a, c, a, b, a`) resembles (`a, b, a, c, a`) 100% for size 2. To make resemblance more sensitive to permutation changes we have to take a larger size; on the other hand a large size is possibly over-sensitive to small alterations since the change in one token affects w shingles.

Notice also that resemblance is not transitive (a well-known fact bemoaned by grandparents all over), but neither is our informal idea of "roughly the same;" for instance consecutive versions of a paper might well be "roughly the same," but version 100 is probably quite far from version 1.

Another fact to note is that the resemblance distance defined as

$$d_w(A, B) = 1 - r_w(A, B)$$

is a metric. (The tedious proof is omitted in this preliminary version.) This might be useful in choosing a clustering algorithm. (See Section 4.4 below.)

Experiments show that strong resemblance and strong containment (that is, close to 1) capture our informal notion of "roughly the same" and "roughly contained."

3 Estimating the resemblance and the containment

Fix a shingle size w, and let Ω be the set of all labelled (for option A) or unlabelled (for option B) shingles of size w. Without loss of generality we can assume that Ω is totally ordered. Now fix a parameter s. For a set $W \subseteq \Omega$ define $\mathrm{MIN}_s(W)$ as

$$\mathrm{MIN}_s(W) = \begin{cases} \text{the set of the smallest } s \text{ elements in } W, & \text{if } |W| \geq s; \\ W, & \text{otherwise.} \end{cases}$$

where "smallest" refers to the order defined on Ω and and for a set $I \subseteq \mathcal{N}$ define

$$\mathrm{MOD}_m(I) = \text{the set of elements of } W \text{ that are 0 mod } m.$$

Theorem 1 *Let $g : \Omega \to \mathcal{N}$ be an arbitrary injection, let $\pi : \Omega \to \Omega$ be a permutation of Ω chosen uniformly at random and let $M(A) = \mathrm{MIN}_s(\pi(S(A, w)))$ and $L(A) = \mathrm{MOD}_m(g(\pi(S(A, w))))$. Define $M(B)$ and $L(B)$ analogously.*

- *The value*
$$\frac{|\mathrm{MIN}_s(M(A) \cup M(B)) \cap M(A) \cap M(B)|}{|\mathrm{MIN}_s(M(A) \cup M(B))|}$$
is an unbiased estimate of the resemblance of A and B.

- *The value*
$$\frac{|L(A) \cap L(B)|}{|L(A) \cup L(B)|}$$
is an unbiased estimate of the resemblance of A and B.

- *The value*
$$\frac{|L(A) \cap L(B)|}{|L(A)|}$$
is an unbiased estimate of the containment of A in B.

Proof: Clearly

$$\begin{aligned}\mathrm{MIN}_s(M(A) \cup M(B)) &= \mathrm{MIN}_s\Big(\pi\big(S(A,w)\big) \cup \pi\big(S(B,w)\big)\Big)\\ &= \mathrm{MIN}_s\Big(\pi\big(S(A,w) \cup S(B,w)\big)\Big)\end{aligned}$$

Let α be the smallest element in $\pi(S(A,w) \cup S(B,w))$. Then

$$\begin{aligned}\mathbf{Pr}(\alpha \in M(A) \cap M(B)) &= \mathbf{Pr}(\pi^{-1}(\alpha) \in S(A,w) \cap S(B,w))\\ &= \frac{|S(A,w) \cap S(B,w)|}{|S(A,w) \cup S(B,w)|} = r_w(A,B).\end{aligned}$$

Since we can repeat this argument for every element of $\mathrm{MIN}_s(\pi(S(A,w) \cup S(B,w)))$ this proves the first claim. The proof of the other two claims is straightforward. $\Box$

In view of the above, we can choose a random permutation and afterwards keep for each document D a *sketch* consisting only of the set $M(D)$ and/or $L(D)$. The sketches suffice to estimate the resemblance or the containment of any pair of documents without any need for the original files.

The set $M(D)$ has the advantage that it has a fixed size, but it allows only the estimation of resemblance. The size of $L(D)$ grows as D grows, but allows the estimation of both resemblance and containment.

To limit the size of $L(D)$ we can proceed as follows: for documents that have size between (say) $100 * 2^i$ and $100 * 2^{i+1}$, we store the set $L_i(D) = \mathrm{MOD}_{2^i}(g(\pi(S(D))))$. The expected size of $L_i(D)$ is always between 50 and 100. On the other hand, we can easily compute $L_{i+1}(D)$ from $L_i(D)$. (We simply keep only those elements divisible by 2^{i+1}.) Thus, if we are given two documents, A and B, and 2^i was the modulus used by the longer document, we use $L_i(A)$ and $L_i(B)$ for our estimates. The disadvantage of this approach is that the estimation of the containment of very short documents into subtantially larger ones is rather error prone due to the paucity of samples.

4 Implementation issues

4.1 Choosing a random permutation and a sample

The total size of a shingle is relatively large: for instance if shingles are made of 7 (English) words each, a shingle will contain about 40-50 bytes on average. Hence to reduce storage we first associate to each shingle a (shorter) id of ℓ bits, and then use a random permutation π of the set $\{0, 1, \ldots, 2^\ell\}$. (Here we can commute g and π in Theorem 1.) There is a trade-off here: if we take ℓ large then we can ensure that most/all id's will be unique, but we will have to pay a storage penalty. On the other hand, a large number of collisions will degrade our estimate as explained below.

Let $f : \Omega \to \{0, \ldots, 2^\ell - 1\}$ be the function that produces this id. Once f fixed, what we are estimating is

$$r_{w,f}(A, B) = \frac{|f(S(A, w)) \cap f(S(B, w))|}{|f(S(A, w)) \cup f(S(B, w))|}$$

rather than

$$r_w(A, B) = \frac{|S(A, w) \cap S(B, w)|}{|S(A, w) \cup S(B, w)|}.$$

Fix an arbitrary set $S \subseteq \Omega$ of size n. Then if f is chosen uniformly at random

$$\mathbf{E}(|f(S)|) = 2^l \left(1 - \left(1 - \frac{1}{2^\ell}\right)^n\right) = n - \binom{n}{2}\frac{1}{2^\ell} + \binom{n}{3}\frac{1}{2^{2\ell}} + \cdots$$

If ℓ is substantially larger than $\log n$ then

$$\mathbf{E}(|f(S)|) \approx n - \binom{n}{2}\frac{1}{2^\ell},$$

in other words in this case we can ignore the effect of multiple collisions, that is, three or more distinct elements of S having the same image under f. Furthermore, it can be argued that the size of $f(S)$ is fairly well concentrated. By Azuma's inequality [1] we have

$$\mathbf{Pr}\left(\Big||f(S)| - \mathbf{E}(|f(S)|)\Big| > \lambda\sqrt{n}\right) < e^{-\lambda^2/2}.$$

However, for practical situations, ℓ is sufficiently large so that it is simpler to use Markov inequality. (E.g.: the probability that the numbe of collisions exceeds $\binom{n}{2}\frac{1}{2^{\ell-10}}$ is less than 1/1000.)

Omitting many details this leads to the fact that with probability better than 99.9% (over the choices of f)

$$|r_{w,f}(A, B) - r_w(A, B)| < \frac{|S(A, w) \cup S(B, w)|}{2^{\ell-11}}.$$

In an actual implementation, f is not totally random, and the probability of collision might be higher. A good choice is to take f to be Rabin's fingerprinting function [10] in which case the probability of collision of two strings s_1 and s_2 can be bounded (in a adversarial model for s_1 and s_2) by $\max(|s_1|, |s_2|)/2^{l-1}$ where $|s_1|$ is the length of the string s_1 in bits.

The advantage of choosing Rabin fingerprints (which are based on random irreducible polynomials) rather than some arbitrary hash functions is that their probability of collision is well understood. Furthermore Rabin fingerprints can be computed very efficiently in software (see [3]) and we can take advantage of their algebraic properties when we compute the fingerprints of "sliding windows." (See section 4.3.)

For the clustering experiment discussed in the introduction the size of a typical set of shingles is about 1000 and the length in bits of a shingle is about 400. We used Rabin fingerprints with $l = 40$.

To produce our sample we need a random permutation $\pi : \{0, 1, \ldots, 2^\ell\} \to \{0, 1, \ldots, 2^\ell\}$. In practice we can imagine that the fingerprints implement $\pi(f(\cdot))$ rather than $f(\cdot)$ or use random linear transformations. (See [4] for an in-depth discussion of this topic.)

Using Rabin fingerprints, in which strings are viewed as polynomials over $\mathbf{Z}_2$, we can imagine that the underlying polynomial is first multiplied by a suitable power or, equivalently, we can imagine that a suitable number of nulls are appended to the underlying string. This changes slightly the way the fingerprints are computed but there is no computing time extra cost. It has the effect that the fingerprints will appear "more random." (Without this trick two strings that differ only in the last letter will differ only in the last eight bits of their fingerprint.)

For further efficiency, rather than keep the set $\mathrm{MIN}_s(\pi(S(A, w)))$ we can keep the MIN_s of the set of fingerprints that are 0 mod m for a suitable chosen m. (That is, we construct the set $\mathrm{MIN}_s(\mathrm{MOD}_m(\pi(S(A, w))))$.)

For option A the set MIN_s should be kept in a heap, with the maximum at the root. A new fingerprint should replace the current root whenever it is smaller than it and then we should re-heapify. The expected number of times this happens is $O(s \log(n/m))$ where n is the number of tokens in the document and m is the modulus discussed above, because the probability that the k'th element of a random permutation has to go into the heap is s/k. The cost per heap operation is $O(\log s)$ and thus the expected total cost for the heap operations is $O(s \log s \log(n/m))$. For option B we need to keep a balanced binary search tree. Some possibilities are red-black trees, randomized search trees, and skip lists. The cost is still $O(s \log s \log(n/m))$ but the constant factor is probably larger. Yet another possibility at the same cost is a heap as before, plus a hash table to check for duplication.

4.2 The size of the sample

The larger the sample, the more accurate the results are likely to be. But we pay for a larger sample in storage costs. The number of common shingles in the sample has a hypergeometric distribution. Since the size of the sample is usually much smaller than the size of the document we can approximate the hypergeometric distribution by the binomial distribution. Under this approximation, if r is the resemblance, then the probability that our estimate is within $[r - \epsilon, r + \epsilon]$ is given by

$$p(s, r, \epsilon) = \sum_{s(r-\epsilon) \leq k \leq s(r-\epsilon)} \binom{s}{k} r^k (1 - r)^{s-k}.$$

For a fixed s our estimate is likely to be worse if r is close to 0.5, but often we are interested only in the case where r is larger. For practical applications, 100 samples seems reasonable and 200 seems more than enough. Depending on the length chosen for the fingerprints this means that the sketches should be between 300 to 800 bytes long. For the entire web the number of pairs where the estimate is relatively far off can be of course substantial. However even for the entire web it is unlikely (probability less than 0.1%) that any pair of documents that resembles less than 50% will be estimated as resembling more than 90%.

4.3 Computing the fingerprints

Computing the fingerprints is fairly fast even if we do each shingle from scratch. The total cost in this case is $O(wn)$ where the O notation hides the width of the token. However, for Rabin fingerprints, we can gain a factor of w if we take advantage of the fact that we are computing the fingerprint of a "sliding window," particularly if the window's width in bytes is fixed or within a narrow range. This is automatic if the tokens have fixed width. Otherwise we can define the tokens to have a small maximum width and pad them as necessary to get this effect.

For instance we can define a word to have at most 8 bytes. Longer words will be viewed as the catenation of several words. When we compute fingerprints we can pad shorter words to 8 bytes.

If we adopt option A then we have to check for each shingle how many time it was encountered before. This can be done by fingerprinting it and then searching a hash table. Once the proper label is decided, a second fingerprint needs to be computed. Some care is needed to avoid too much dependence among the fingerprints of the same shingle with different labels. Option B avoids these computations altogether, but as explained above we must keep a binary search tree rather than a heap to make sure we don't insert the same value twice. Nevertheless option B seems faster in practice.

4.4 Evaluating resemblance

We store each sketch as a list sorted in increasing order. Then all we have to do is to merge-sort removing duplicates, and count how many duplicates were encountered within the first s outputs. This is $O(s)$.

It is likely that we are interested in more than two documents. For r documents evaluating all resemblances takes $O(r^2 s)$ time. However we can try to do a "greedy clustering" as follows: keep a set of current clusters (initially empty) and process the sketches in turn. For each cluster keep a representative sketch. If a new sketch sufficiently resembles a current cluster then add the sketch to it (and possibly recompute the representative); otherwise start a new cluster. It is likely that in practice every fingerprint belongs only to a few distinct clusters. Under this assumption, if for each fingerprint we encounter we remember to which clusters it belongs and store the fingerprints in a hash table, the entire procedure can be implemented in $O(rs)$ time. As a representative sketch, we can take the s most popular fingerprints in a cluster, or simply, just the first member of the cluster.

For very large collections of documents where r is such that external storage is needed, different approaches become necessary. For further details see [5].

5 Acknowledgments

Some essential ideas behind the resemblance definition and computation were developed in conversations with Greg Nelson. The clustering of the entire Web was done in collaboration with Steve Glassman, Mark Manasse, and Geoffrey Zweig.

References

[1] N. Alon and J. H. Spencer. *The Probabilistic Method*. John Wiley and Sons, 1992.

[2] S. Brin, J. Davis, H. Garcia-Molina. Copy Detection Mechanisms for Digital Documents. *Proceedings of the ACM SIGMOD Annual Conference*, May 1995.

[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, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-Wise Independent Permutations. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, 1998, to appear.

[5] A. Z. Broder, S. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. In *Proceedings of the Sixth International WWW Conference*, April 1997.

[6] Nevin Heintze. Scalable Document Fingerprinting. *Proceedings of the Second USENIX Workshop on Electronic Commerce*, November 1996.

[7] U. Manber. Finding similar files in a large file system. *Proceedings of the 1994 USENIX Conference*, January 1994.

[8] N. Shivakumar, H. Garcia-Molina. SCAM: A Copy Detection Mechanism for Digital Documents. *Proceedings of the 2nd International Conference on Theory and Practice of Digital Libraries*, 1995.

[9] N. Shivakumar and H. Garcia-Molina. Building a Scalable and Accurate Copy Detection Mechanism. *Proceedings of the 3nd International Conference on Theory and Practice of Digital Libraries*, 1996.

[10] M. O. Rabin. Fingerprinting by random polynomials. Center for Research in Computing Technology, Harvard University, Report TR-15-81, 1981.