Winnowing: Local Algorithms for Document Fingerprinting

http://theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf

If you wanted to compare two whole files to see if the are a clone then the obvious method would be to hash the files and compare the hash values. This algorithm applies this to finding partial clones. In short it works by removing irrelevant features from the text, splitting the text into parts of length *k* called *k-grams* and then hashing each *k-gram*. A small subset of these hashes is the derived and this is called the called the fingerprints of the document. The idea is that if two documents share one or more fingerprints then they likely share the same text.

The problem comes in finding the best way to decide which hash to use as one of the fingerprints of the file. This paper purports to give an efficient algorithm to select the fingerprints and also guarantees that at least part of any sufficiently long match is detected.

The paper compares their solution, called *winnowing*, to other algorithms. It uses the density of the algorithm and it defines the density of a fingerprinting algorithm to be the expected fraction of hashes that are selected from all the hash values computer when given a random input. It defines a *local* algorithm, which captures certain properties of a fingerprinting algorithm. It must define a window of size *w* to be *w* consecutive hashes of *k-grams* in a document and selects at least one fingerprint from each window. The algorithm is considered *local* iff the choice of fingerprint of each window only depends on the hashes in that window.

Winnowing algorithm:

2 thresholds are chosen by the user, noise threshold k is the lower bound (no matching strings shorter than k), and the guarantee threshold t (match all substrings at least as long).

Noise threshold k is used to divide the text into k-grams. Bigger k reduces noise, but also reduces sensitivity to the reordering of contents.

Window size w = t – k +1

For each window select the minimum hash value to be the fingerprint of the document. If there is a tie use the rightmost minimal value.

The winnowing algorithm is first compared to a *0 mod p* algorithm, which is where the hash is selected if it is a divisor of p. It is then compared to other local algorithms, to see if there is another algorithm that performs better than winnowing. They prove that the winnowing algorithm has a lower bound on the density as good as the optimum local fingerprinting algorithm and therefore there does not exist a local fingerprinting algorithm with lower density.

Finally the paper shows the real world results with web data, and with plagiarism detection in the *MOSS* system. They find a problem due to the fact that there are large passages of repetitive low-entropy strings. Here winnowing encounters many identical hash values and therefore many ties for the minimum hash in the window. This causes poor behaviour; to solve this they define Robust Winnowing, which is not a local algorithm. Robust winnowing breaks ties by selecting the same hash as the previous window if possible. Otherwise it reverts back to normal winnowing behaviour. This is used to reduce the density on low-entropy strings. Moss uses this algorithm for its plagiarism detection. They discuss the implementation of *Moss* and how it stores the fingerprint data, how the comparisons are done against all the entries in the database, as well as how it presents its results.