Detecting near-duplicates for web crawling

Authors: Gurmeet Singh Manku, Arvind Jain, Anish Das Sarma

Presented by YuLong Pei

Intro/Background

- Many near duplicate web documents
- Exact duplicates: mirroring, plagiarism
  - Easy to find with checksum
- Near duplicates: identical content but differ in small portions like ads, counters, timestamps
- Elimination saves network bandwidth, reduces storage cost, improves quality of search indexes, reduces load on remote hosts

Related Works

- Various techniques developed to deal with different:
- Corpus: web documents, files in a file system, e-mails, domain-specific corpora
- End goals: web mirrors, clustering for “related documents” query, data extraction, plagiarism, spam detection, duplicated in domain-specific corpora
- Feature-set per document: shingles from page content, document vector from page content, connectivity information, anchor text, anchor window, phrases
- Signature schemes: mod-p shingles, min-hash for Jaccard similarity of sets, signatures/fingerprints over IR-based document vectors, checksums

This paper focus on large sized web document, improve web-crawling, using document vector, and simhash for small-sized fingerprints.

Why we care

- Knowing about existing techniques is good start to finding our own algorithm
- Find near duplicates between files and cluster the files
- Extract data and information from the files
- Google's example is good because it experimented with large amounts of files, so Big Data
  - 8Billion files vs ten of thousands

Problem to solve

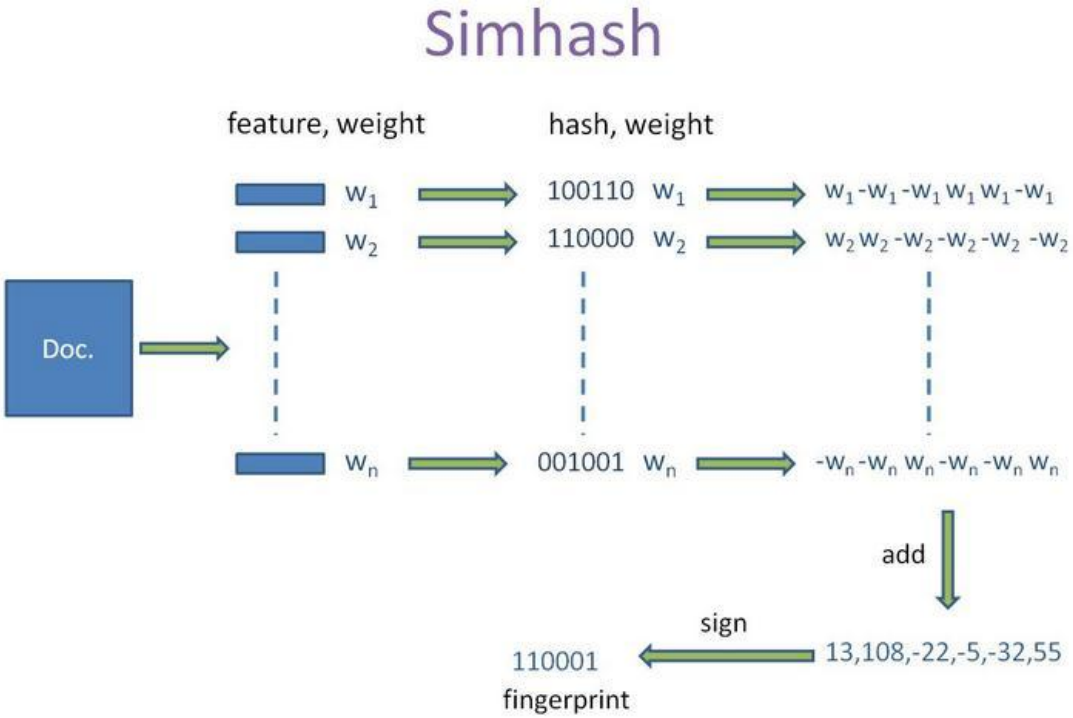
Hamming Distance Problem: In collection of f-bit fingerprints, quickly find all fingerprints that differ from given query fingerprint in at most k bit positions, where k is small integer

- technique needs to be useful for both online queries (single fingerprints) and batch queries (multiple fingerprints)

Fingerprints: Charikar's simhash

- convert web-page into set of features, each feature has weight

Charikar's simhash

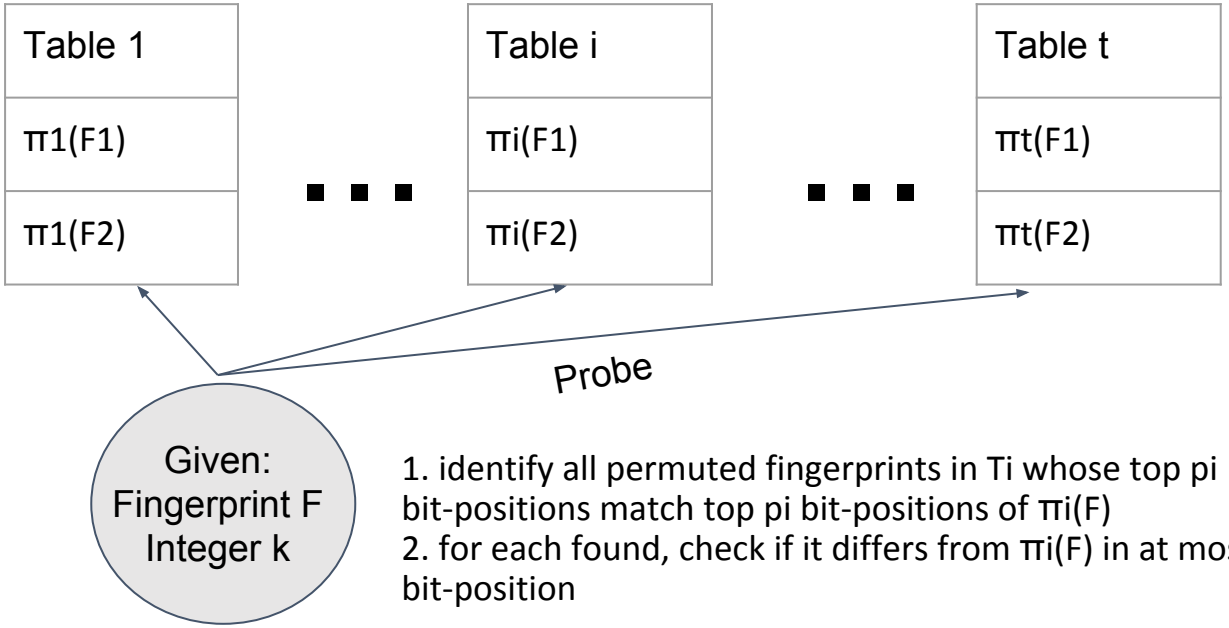


Properties of simhash

A. fingerprint of document is a "hash" of its features
B. similar documents have similar hash values (uncommon with hash functions)

Assumption: Property A holds, experimentally measure impact of B
→ fingerprints are distributed uniformly at random, with some non-uniformity

Hamming Distance: Single Query



1. identify all permuted fingerprints in Ti whose top pi bit-positions match top pi bit-positions of πi(F)
2. for each found, check if it differs from πi(F) in at most k bit-position

- Permuted f-bit sorted
- table compressed and sorted

Runtime:
- O(pi) steps by binary search
- O(log pi) interpolation search shrinks

Hamming Distance: Batch Queries

- 8B existing fingerprints (F) and batch of 1M query fingerprints (Q) are stored in files
- in Google, GFS breaks the files into smaller files
- MapReduce

    Mapper: solves Hamming Distance Problem over each chunk of F and entire file Q

    Reducer: collects output of list of near-duplicate fingerprints found, removes duplicates and produce single sorted file

Different from single queries
- build tables corresponding to file Q (for single F table is built for existing fingerprint file)

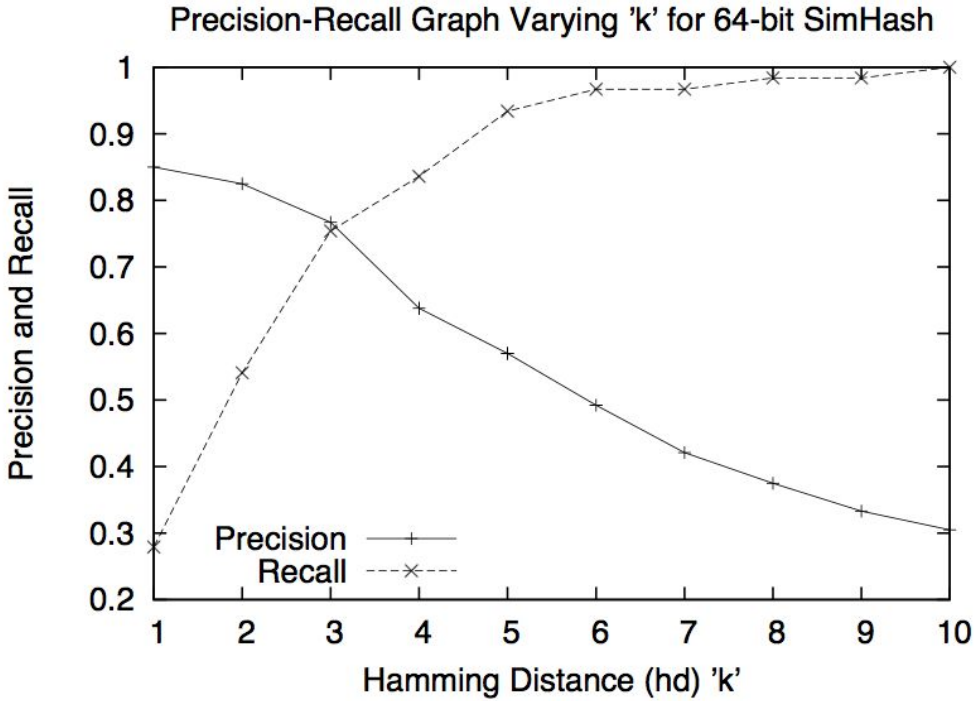
Compression of fingerprints

- successive fingerprints share top d bits in expectation
- have h denote position of most significant 1-bit of XOR of two successive fingerprints
- find distribution of h values and find Huffman code
- first fingerprint in block remembered entirely; subsequent once are XOR with previous fingerprint and appended with Huffman code and rest of bits; until block is full

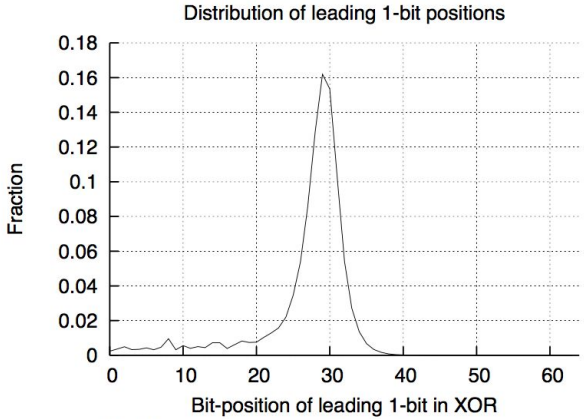
Scalability: 200 mappers, 64 GB → 32GB compressed, < 100 seconds

Experimental Results

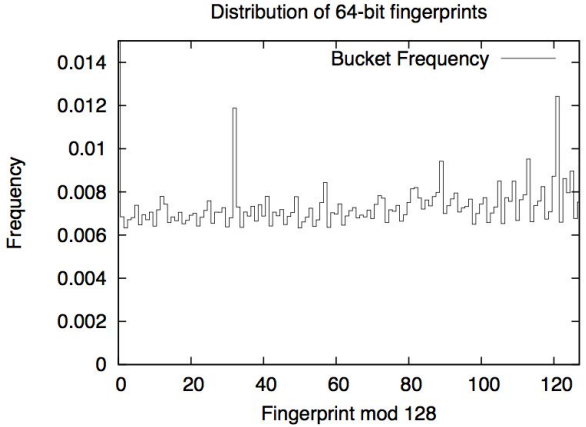
Parameter Choice:
K = 3, f = 64



Experimental Results



(a) Distribution of leading 1-bit positions of exclusive-OR of successive fingerprints.



(b) Bucketization of fingerprints.

- true randomness, entire graph should exhibit right side
- Dense on left side due to clustering

- spikes from empty page, page not found, same bulletin board software

Future works

- Categorize web-pages into different categories, search only from within
- Does simhash-based technique work for focused crawlers
- Can near duplicate detection develop to facilitate clustering

Conclusion

- Algorithm for near-duplicate detection using simhash
- Simhash create small-sized fingerprint, great for the scale of the problem
- Need to work with online mode and batch mode
- Experiment to find good results with f = 64, k = 3

Pro/Con of the Paper

Pro:

- Nice structure
- Many examples for readers to understand why detect near-duplicate

Con:

- Inconsistency that led to error in description
- Poor explanation for some technical concepts

References

Manku, Gurmeet Singh, Arvind Jain, and Anish Das Sarma. "Detecting near-duplicates for web crawling." Proceedings of the 16th international conference on World Wide Web. ACM, 2007.

Massive Algorithms, SimHash: Hash-based Similarity Detection; http://massivealgorithms.blogspot.com/2014/12/simhash-hash-based-similarity-detection.html