Achieving both High Precision and High Recall in Near-duplicate Detection (with High Performance)

Given a set of 400 million web pages, can you partition it into subsets of similar web pages with high precision and high recall in less than one week, with a cluster of 10 computers ?

Lian'en Huang, Lei Wang and Xiaoming Li
Computer Networks and Distributed Systems Laboratory
Peking University

An alternative way to describe the work: A LCS Based Algorithm for Partitioning Large Set of Web Pages into Subsets of Similar Pages

As described this way, this work **is not about**:

Given two web pages, how to tell if they are similar accurately and effectively.

(This is a kind of micro-problem)

But is about:

Given a large set of web pages, how to partition it into subsets of similar web pages efficiently with high quality.

(This is a macro-problem, though it will use some solutions to the micro-problem. But because our objective is at a higher level, we have a larger "design space" to work with. In this sense, what our paper is about is really a **framework** of algorithm)

A=ab**cab**b**a** B=**c**b**aba**c

LCS=**caba**

$$r(A,B)=\frac{|LCS(A,B)|}{|A|+|B|-|LCS(A,B)|}$$

News or blog articles, etc.
not navigational pages

We have 400 million pages to process

And for better precision and recall, we decided to use LCS as the measure

The core challenge: the efficiency of the whole process

- Given two pages A and B, the brute-force algorithm to compute LCS of A and B is time consuming

$$\longrightarrow O(|A| \times |B|)$$

- And in theory, we need to do this for each pair of the 400 million pages !

Two natural lines of thinking to make the task easier:

1. Find a more efficient algorithm than brute-force
2. Divide & conquer - employ a two phase process: (1) pre-partition the original set into some kinds of smaller subsets, (2) let the pair-wise comparison only occurs within each subset.

The primary decisions for the two aspects

- Use Myer's difference algorithm to compute LCS(A,B).

$$\longrightarrow O\left(\left(|A|+|B|\right)\times D\right)$$

  - ✓ D: the length of the shortest edit script between A and B. The more similar A and B, the smaller D.
- Let the pre-partitioned subsets be subsets of "perhaps-similar" web pages.
  - ✓ As such, D is presumably small within the subsets, which is helpful to reduce the time to execute Myer's algorithm

For higher performance of the process

- Further reduce D and |A|+|B| before applying Myer's algorithm
  - ✓ Filter out portions of a page which unlikely exist in the LCS of two pages
- Avoid "complete" pair-wise comparison in perhaps-similar subsets
  - ✓ Sort pages in each subset by some criterion. Each page only compare with pages after it and not similar with pages before it. Example:
    - Suppose the order of sorted pages is A B C D E F.
    - Compare A with BCDEF. If C and D turn out to be similar with A, ACD form a set of similar pages.
    - B will be compared with E and F, but not be compared with C and D.

For higher quality of the result

- To improve recall – allow perhaps-similar subsets be reasonably large, but avoid too large for performance
  - ✓ Compute a set of fingerprints for each page
  - ✓ Common fingerprints define a collection of initial equivalent sets of pages, the *initial perhaps-similar* page sets
  - ✓ Employ some partial transitivity to obtain larger equivalent sets – the *perhaps-similar* page sets, and make sure they form a partition of original set.
- To improve precision – filter out portions of a page which are not relevant to the topical content of the page
  - ✓ e.g., uniform templates used in creating pages within a website.

Experiments

- Data set
  - ✓ From Chinese web archive: Web Infomall
  - ✓ 400 millions topical-type web pages
- Evaluation objectives
  - ✓ Precision
  - ✓ Recall
  - ✓ Efficiency
- Comparison
  - ✓ simhash
  - ✓ cosine

After phase 1, 46 million perhaps-similar sets were obtained.

Phase 2 refined them and generated 68 million similar sets.

Measurement for quality

- We sampled 1000 sets from the 68 million sets obtained by our algorithm
- For each set, via human inspection, we chose a representative page (RP) which has the most true similar pages in this set.
- Precision and recall are computed as follows:

$$precision = \frac{\sum_{\text{the sampled sets}} \frac{\#\text{ true similar pages of RP in this set}}{\#\text{ all pages in this set}}}{\#\text{ the sampled sets}}$$

$$recall = \frac{\sum_{\text{the sampled sets}} \frac{\#\text{ true similar pages of RP detected by LCS (Cosine)}}{\#\text{ true similar pages of RP detected by Simhash}}}{\#\text{ the sampled sets}}$$

The result

The recall used simhash as a baseline

Table 1 Precision, Recall and Efficiency

Algorithms	Precision	Recall	Precision(same site)	Recall(same site)	Efficiency
Simhash	0.72	1	0.52	1	94hours
Cosine	0.82	1.19	0.63	1	68+534hours
LCS	0.95	1.86	0.91	1.03	118hours

Table 2 Evolvement of the efficiency and precision

Stage	Efficiency	Precision
Without stage one	Several years	
After stage one	5700 hours	0.67
After stage two	118 hours	
After stage three		0.95

Perhaps similar sets

After LCS comparison

Summary

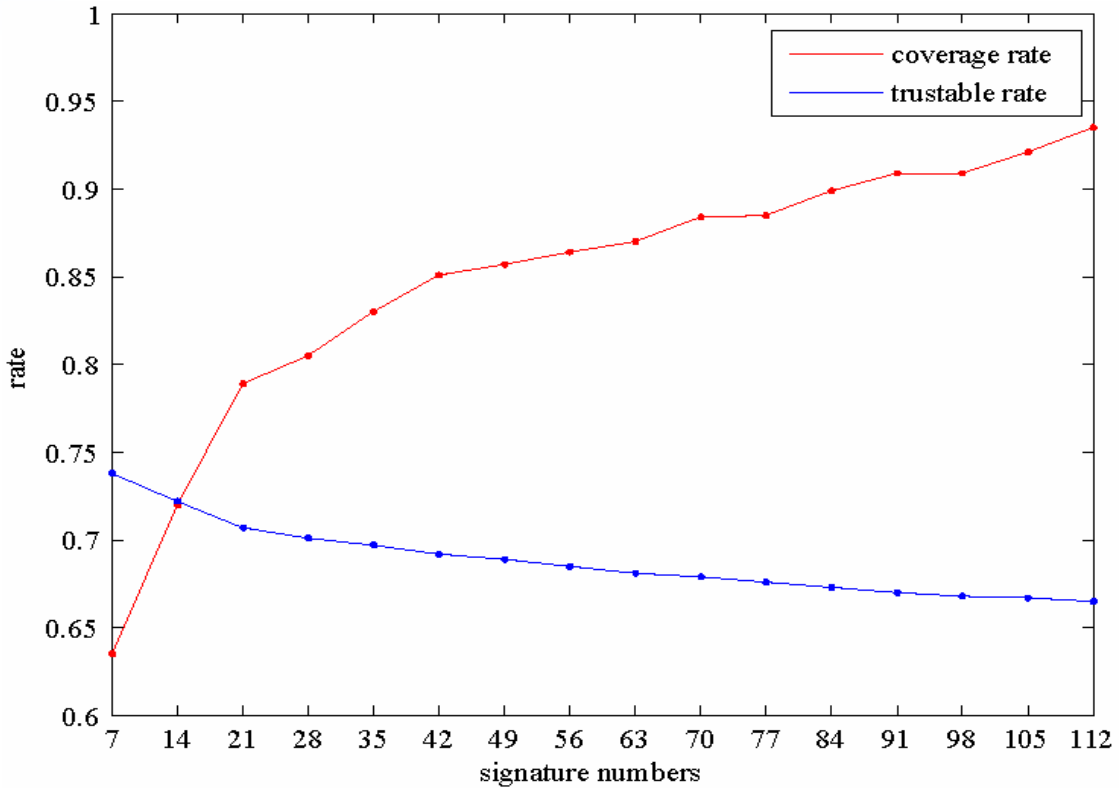
- Using six PC servers, we spent 120 hours in partitioning 430 millions web pages into 68 millions subsets of similar pages, with much better precision and recall than the mainstream algorithms. And the performance is close.
- Why could the precision be improved?
  - ✓ We used LCS as similarity measurement. Intuitively it reflects the true similarity of pages.
  - ✓ Other algorithms such as simhash and shingling conceptually all use “probability of being similar” as the measurement. It is indirect.
- Why can we achieve acceptable performance for the task ? Divide & conquer with fine tuning of each key point.

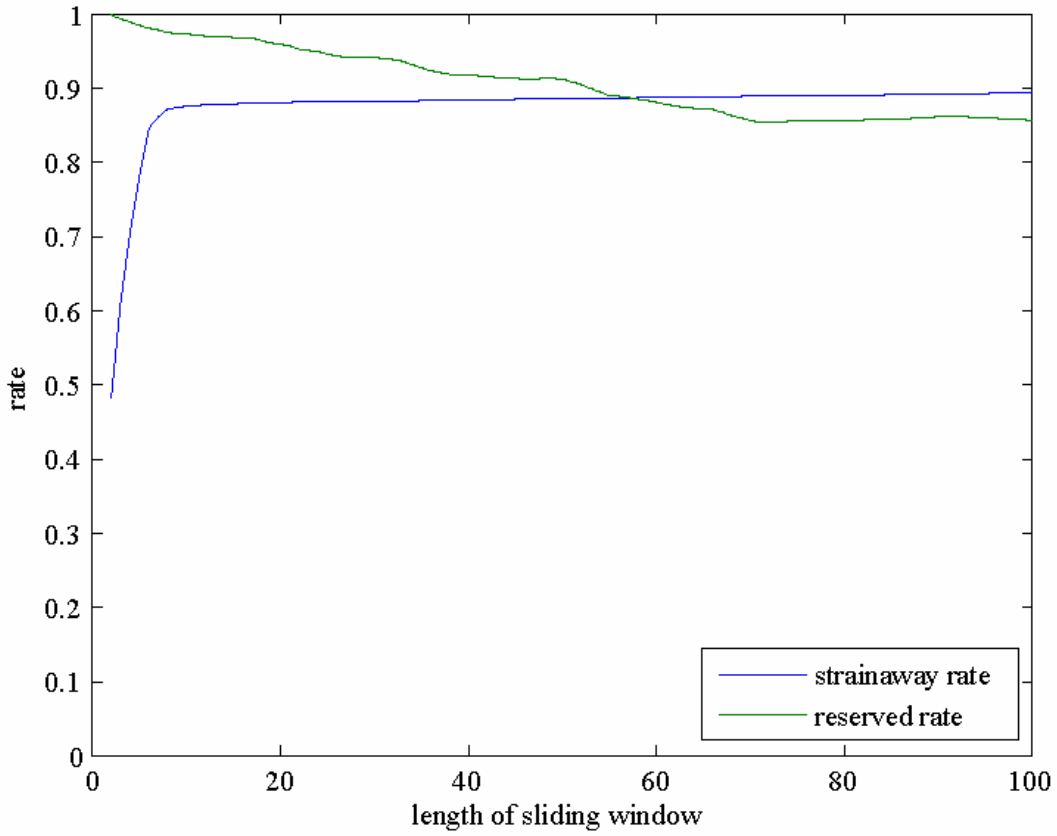
Thank you!

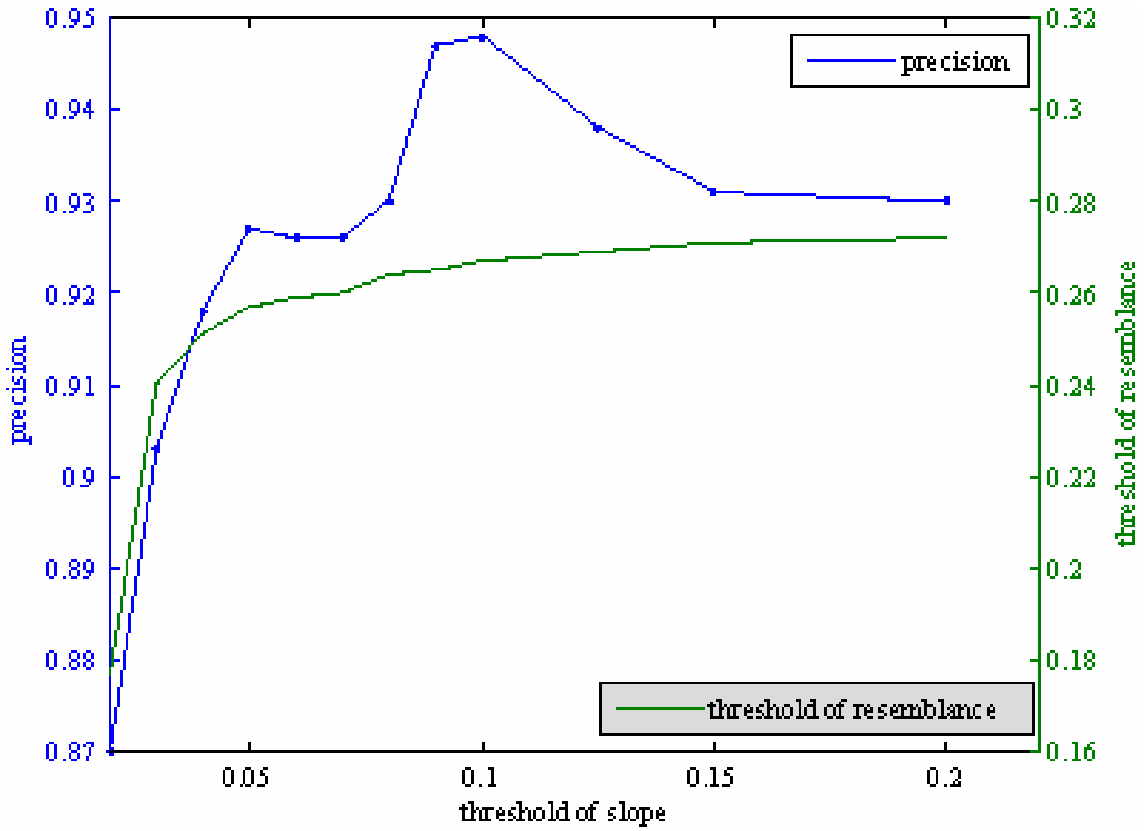
wanglei@net.pku.edu.cn


NC&IS


NC&IS


coverage rate
trustable rate
rate
signature numbers
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
7
14
21
28
35
42
49
56
63
70
77
84
91
98
105
112

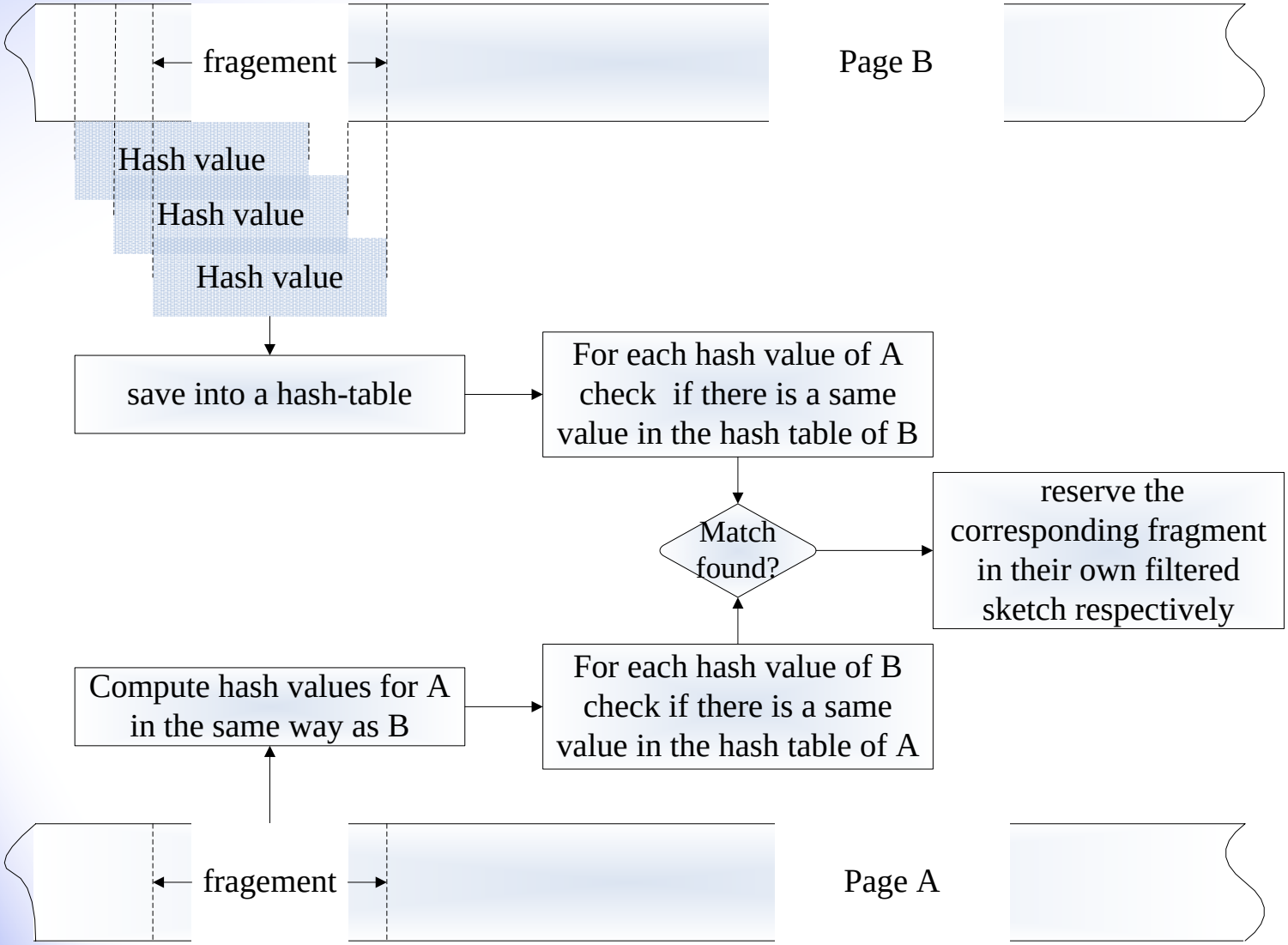

1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
rate
0
20
40
60
80
100
length of sliding window
strainaway rate
reserved rate


precision
threshold of resemblance
precision
threshold of resemblance
threshold of slope
0.95
0.94
0.93
0.92
0.91
0.9
0.89
0.88
0.87
0.32
0.3
0.28
0.26
0.24
0.22
0.2
0.18
0.16
0.05
0.1
0.15
0.2

Details: divide into perhaps-similar sets

- Transform each original page into sets of sentences.
  - Remove all HTML markups and formatting instructions etc.
- Compute an MD5 value for each sentence.
- Distribute each MD5 value into one of m (such as m=7) buckets according to the result of mod m of this MD5 value.
- For each bucket, select n (such as n = 3) smallest MD5 values to compute a new hash value, called a fingerprint.
- Use these fingerprints to index and cluster pages to form sets of perhaps-similar pages.

Details: compute filtered sketches



Details: computing LCS

- We could use the LCS computed by Myer's algorithm directly.
- Templates on same web sites also exist in LCS, and make bad influence on precision.
- Templates often exist on the head and tail portions of a page with other noise information.
- It made the distribution of LCS incoherent in the head and tail portions and coherent in the main text of the original pages.
- Based on these observation, we chose a trustable portion of LCS which was coherent and existed in the central portion in the original page.