FINDING NEAR DUPLICATE WEB PAGES
- A LARGE-SCALE EVALUATION OF ALGORITHMS

- Monika Henzinger
  Published: 2006, Google Inc.

Presenter:
Ketan Akade
INTRODUCTION

• What are near duplicate web pages?

• What problems it causes?
  ➢ Increases the space needed to store the index
  ➢ Slows down and increases the cost of serving results

• Why detect near duplicate web pages?
  ➢ Smarter crawling
  ➢ Improved page rank, less storage, smaller indexing
**APPROACHES TO FIND NEAR DUPLICATES**

- Compare all pairs of documents (naive solution)
  - very expensive on large databases
- Manber and Heintze proposed algorithm based on sequences of adjacent characters
- Shivakumar and Garcia-Molina focused on scaling this algorithm up to multi-gigabyte databases.
- Broder et al. used word sequences to efficiently find near duplicate web pages
- Charikar developed an algorithm based on random projections of the words in a document.
ALGORITHM FOR DETECTING NEAR DUPLICATES

• Every HTML page is converted into a token sequence
  ➢ All HTML markups are replaced by white spaces
  ➢ All the formatting instructions are ignored
• Every alphanumeric sequence is considered as a term
• Term is hashed using Rabin’s fingerprinting scheme to generate tokens
• Every URL is broken at slashes and dots and is treated like a sequence of terms
• To distinguish between images the URL in image tag is considered as a term
• A bit string is generated from the token sequence of a page and used to determine near-duplicates for the pages
Broder et al.’s Algorithm (Algorithm B)

- Every subsequence of k tokens is fingerprinted using 64-bit Rabin fingerprints.
- It results in n-k+1 fingerprints, called shingels.
- % of unique shingles between page d and d’ = \[
\frac{|S(d) \cap S(d')|}{|S(d) \cup S(d')|}
\]
- Shingels is a good measure of similarity of d and d’.
- Every shingle is fingerprinted with m different fingerprinting functions \(f_i\) for \(1 \leq i \leq m\) that are same for all pages.
- For each i, the smallest i-th minvalue is stored creating m-dimensional vector for each page.
- This m-dimensional vector is reduced to \(m'\) dimensional vector of supershingles by fingerprinting non-overlapping sequence of minvalues.
Broder et al.’s Algorithm Contd...

• Let $l = \frac{m}{m'}$
• The concatenation of minvalue $j*l, \ldots, (j+1)*l-1$ for $0 \leq j < m'$ is fingerprinted with another function to form a **supershingle vector**
• The number of identical entries in supershingle vectors of two pages is their B-similarity
• Two pages are near duplicates of Alg.B iff their B-similarity is at least 2

<table>
<thead>
<tr>
<th>B-similarity</th>
<th>Number of near duplicates</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>958,899,366</td>
<td>52.4</td>
</tr>
<tr>
<td>3</td>
<td>383,076,019</td>
<td>20.9</td>
</tr>
<tr>
<td>4</td>
<td>225,454,277</td>
<td>12.3</td>
</tr>
<tr>
<td>5</td>
<td>158,989,276</td>
<td>8.7</td>
</tr>
<tr>
<td>6</td>
<td>104,628,248</td>
<td>5.7</td>
</tr>
</tbody>
</table>

Table: Number of near-duplicate pairs found for each B-similarity value for 1.6B pages
CHARIKAR’S ALGORITHM (ALGORITHM C)

- Let \( b \) be a constant
- Each token is projected into \( b \)-dimensional space by randomly choosing \( b \) entries from \{-1,1\}
- This projection is same for all pages
- For each page a \( b \)-dimensional vector is created by adding the projections of all the tokens in its token sequence.
- The final vector for the page is created by setting every positive entry in the vector to 1 and every non-positive entry to 0, resulting in random projection for each page
- \( C \)-similarity of two pages is the number of bits their projections agree on
COMPARISON OF ALGORITHM B AND C

• Both algorithms were executed on a set of 1.6B unique pages collected by Google crawler
• Algorithms are compared based on:
  - precision
  - the distribution of the number of term differences in the near duplicate pairs
• Each near duplicate pair is labeled as correct, incorrect, undecided
• Two web pages are correct near duplicates if
  - their text differs only by session id, timestamp, a visitor count or part of their url
  - the difference is invisible to the visitors of the page
• A near duplicate pair is incorrect if the main item of the page is different
• The remaining near duplicate pairs are called undecided.
• Reasons for undecided pairs are:
  - Prefilled forms with different but erasable values such that erasing the values results in same form
  - If a page gets refreshed, then the token sequence will be different from stored page sequence.
  - If human evaluator could not decide if the difference between two pages is major or minor (mostly in case of Chinese, Japanese pages)
RESULTS OF ALGORITHM B AND C

• Manual evaluation
  • Algorithm C achieves higher precision (0.5) than Algorithm B (0.38)
  • Algorithm B has higher recall while Algo. C has higher precision for pairs on the same site

<table>
<thead>
<tr>
<th>B-similarity</th>
<th>C-similarity average</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>331.5</td>
</tr>
<tr>
<td>3</td>
<td>342.4</td>
</tr>
<tr>
<td>4</td>
<td>352.4</td>
</tr>
<tr>
<td>5</td>
<td>358.9</td>
</tr>
<tr>
<td>6</td>
<td>370.9</td>
</tr>
</tbody>
</table>

Table: For a given B-similarity, average C-similarity

Therefore, combination of two algorithms can achieve higher precision and recall pairs

The C-similarity distribution for different fixed B-similarities.
THE COMBINED ALGORITHM

• First compute all B-similar pairs
• Then filter out those pairs whose C-similarity falls below a certain threshold
• The precision and recall of the combined algorithm can significantly improve if high value of C-similarity is used.

<table>
<thead>
<tr>
<th>Precision</th>
<th>C-similarity threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.754</td>
<td>351</td>
</tr>
<tr>
<td>0.77</td>
<td>355</td>
</tr>
<tr>
<td>0.789</td>
<td>359</td>
</tr>
</tbody>
</table>

Table: Precision values of Algo. C for different C-similarity threshold and B-similarity equal to 3
CONCLUSION

• Evaluation of Algorithm B and C shows that neither performs well on the pages from same site.

• Algorithm C has higher precision while Algorithm B has higher recall.

• Combined algorithm is superior in both precision and recall.