SpotSigs
Robust & Efficient Near Duplicate Detection in Large Web Collections

Martin Theobald
Jonathan Siddharth
Andreas Paepcke

Stanford University
Sigir 2008, Singapore

Near-Duplicate News Articles (I)

Near-Duplicate News Articles (II)

Our Setting

What is SpotSigs?

- Robust signature extraction
  - Stopword-based signatures favor natural-language contents of web pages over navigational banners and advertisements

- Efficient near-duplicate matching
  - Self-tuning, highly parallelizable clustering algorithm
  - Threshold-based collection partitioning and inverted index pruning

... but

- Many different news sites get their core articles delivered by the same sources (e.g., Associated Press)

- Even within a news site, often more than 30% of articles are near duplicates (dynamically created content, navigational pages, advertisements, etc.)
Case (I): What’s different about the core contents?

Stopword occurrences: the, that, {be}, {have}

Case (II): Do not consider for deduplication!

no occurrences of: the, that, {be}, {have}

Spot Signature Extraction

• “Localized” signatures: n-grams close to a stopword antecedent
  
  – E.g.: that:presidential:campaign:hit
  
  antecedent nearby n-gram

Spot Signature Extraction Example

• Consider the text snippet:
  
  “At a rally to kick off a weeklong campaign for the South Carolina primary, Obama tried to set the record straight from an attack circulating widely on the Internet that is designed to play into prejudices against Muslims and fears of terrorism.”

  \( S = \{ \text{a:rally:kick, a:weeklong:campaign, the:south:carolina, the:record:straight, an:attack:circulating, the:internet:designed, is:designed:play} \} \)

  [for antecedents \{a, the, {be}\}, uniform spot distance \(d=1\), chain length \(c=2\)]

Signature Extraction Algorithm

• Simple & efficient sliding window technique

\[
O(|\text{tokens}|) \text{ runtime}
\]

\[
\text{Largely independent of input format (maybe remove markup)}
\]

\[
\text{No expensive and error-prone layout analysis required}
\]
Choice of Antecedents

F1 measure for different antecedents over a “Gold Set” of 2,160 manually selected near-duplicate news articles, 68 clusters

How to deduplicate a large collection efficiently?

- Given \( \{S_1, ..., S_n\} \) Spot Signature sets
  - For each \( S_i \), find all similar signature sets \( S_1, ..., S_k \) with similarity \( \text{sim}(S_i, S_j) \geq \tau \)
- Common similarity measures:
  - Jaccard, Cosine, Kullback-Leibler, ...
- Common matching algorithms:
  - Various clustering techniques, similarity hashing, ...

Which documents (not) to compare?

- Given 3 Spot Signature sets:
  - \( A \) with \( |A| = 345 \)
  - \( B \) with \( |B| = 1045 \)
  - \( C \) with \( |C| = 323 \)

Which pairs would you compare first? Which pairs could you spare?

\( \text{idea:} \) Two signature sets \( A, B \) can only have high (Jaccard) similarity if they are of similar cardinality!

Upper bound for Jaccard

- Consider Jaccard similarity
  \[ \text{sim}(A, B) = \frac{|A \cap B|}{|A \cup B|} \]
- Upper bound
  \[ \text{sim}(A, B) \leq \frac{|A|}{|B|} \leq \min\left(\frac{|A|}{|B|}, \frac{|B|}{|A|}\right) \]
  \[ |A \cap B| \leq \min(|A|, |B|) \]
  \[ |A \cup B| \geq \max(|A|, |B|) \]

\( \text{idea:} \) Never compare signature sets \( A, B \) with \( |A|/|B| < \tau \) i.e. \( |B| - |A| > (1-\tau) \cdot |B| \)

Multi-set Generalization

- Consider weighted Jaccard similarity
  \[ \text{sim}(A, B) = \frac{\min\left(\sum_{x \in A} \text{freq}(x), \sum_{x \in B} \text{freq}(x)\right)}{\max\left(\sum_{x \in A} \text{freq}(x), \sum_{x \in B} \text{freq}(x)\right)} \]
- Upper bound
  \[ \text{sim}(A, B) \leq \frac{\min\left(\sum_{x \in A} \text{freq}(x), \sum_{x \in B} \text{freq}(x)\right)}{|A|} \cdot \frac{|B|}{\max\left(\sum_{x \in A} \text{freq}(x), \sum_{x \in B} \text{freq}(x)\right)} \]

\( \text{idea:} \) Still skip pairs \( A, B \) with \( |B| - |A| > (1-\tau) \cdot |B| \)
Partitioning the Collection

- Given a similarity threshold $\tau$, there is no contiguous partitioning (based on signature set lengths), s.t.
  (A) any potentially similar pair is within the same partition, and
  (B) any non-similar pair cannot be within the same partition.

... but: there are many possible partitionings, s.t.
- (A) any similar pair is (at most) mapped into two neighboring partitions.

Optimal Partitioning

- Given $\tau$, find partition boundaries $p_0,\ldots,p_k$ s.t.
  (A) all similar pairs (based on length) are mapped into
  at most two neighboring partitions (no false negatives)
  (B) no non-similar pair (based on length) is
  mapped into the same partition (no false positives)
  (C) all partitions’ widths are minimized w.r.t. (A) & (B)

$\Rightarrow$ But expensive to solve exactly ...

Approximate Solution

"Starting with $p_0 = 1$, for any given $p_k$, choose $p_{k+1}$ as the smallest integer $p_{k+1} > p_k$
 s.t. $p_{k+1} - p_k > (1-\tau)p_k$".

For $\tau=0.7$: $p_0=1, p_1=3, p_2=6, p_3=10,\ldots, p_7=43, p_8=59,\ldots$

$\Rightarrow$ Converges to optimal partitioning when distribution is dense
$\Rightarrow$ Web collections typically skewed towards shorter document lengths
$\Rightarrow$ Progressively increasing bucket widths are even beneficial for more
uniform bucket sizes (next slide!)

Partitioning Effects

$\Rightarrow$ Optimal partitioning approach even smoothes
skewed bucket sizes

(plot for 1,274,812 TREC WT10g docs with at least 1 Spot Signature)

$\Rightarrow$ Can do better:
- Create auxiliary inverted indexes within partitions
- Prune inverted index traversals using the very same
  threshold-based pruning condition as for partitioning

... but
Inverted Index Pruning

Pass 1:
- For each partition, create an inverted index:
  - For each spot signature \( s_i \):
    - Create inverted list \( L_i \), with pointers to documents \( d_j \) containing \( s_i \)
    - Sort inverted list in descending order of \( f_{freq}(s_i) \) in \( d_j \)

Partition \( k \):
- \( d_{2,8} \), \( d_{3,4} \), \( d_{1,5} \)

Pass 2:
- For each document \( d_i \), find its partition, then:
  - Process lists in descending order of \( (s_i) \)
  - Maintain two thresholds:
    - \( \delta_1 \): Minimum length distance to any document in the next list
    - \( \delta_2 \): Minimum length distance to next document within the current list
  - Break if \( \delta_1 + \delta_2 > |d_i| \)

SpotSigs Deduplication Algorithm

- Still \( O(n^2 m) \) worst case runtime
- Empirically much better, may outperform hashing
- Tuning parameters: none
- See paper for more details!

Deduplication Example

Given:
- \( D_1 = \{ s_1, s_2, s_3, s_4 \} \)
- \( D_2 = \{ s_5, s_6, s_7 \} \)
- \( D_3 = \{ s_8, s_9 \} \)

Threshold:
- \( \tau = 0.8 \)
- Break if:
  1) \( \delta_1 + \delta_2 > |d_i| \)
  2) \( \delta_1 \rightarrow \delta_2 \), continue
  3) \( \delta_1 \rightarrow \delta_2 \), stop

Experiments

- Collections
  - "Gold Set" of 2,160 manually selected near-duplicate news articles from various news sites, 68 clusters
  - TREC WT10g reference collection (1.6 Mio docs)
- Hardware
  - Dual Xeon Quad-Core @ 3GHz, 32 GB RAM
  - 8 threads for sorting, hashing & deduplication
- For all approaches
  - Remove HTML markup
  - Simple ID filter for signatures, remove most frequent & infrequent signatures

Competitors

- Shingling [Broder, Glassman, Manasse & Zweig '97]
  - In-gram sets/vectors compared w/ Jaccard/Cosine similarity
  - \( O(n^2 m) \) runtime (using LSH for matching)
- i-Match [Chaudhary, Broder, Grossman & Mitra '12]
  - Employs a single SHA-1 hash function
  - Highly tunable
  - \( O(n \log m) \) runtime
- Locality Sensitive Hashing (LSH) [Chen, Glass & Manasse '06; [Broder et al. '03]
  - Employs random hash functions, each concatenates \( \delta \) MinHash signatures
  - Highly tunable
  - \( O(n \log m) \) runtime
- Hybrids of i-Match and LSH with Spot Signatures (i-Match-5 & LSH-5)

“Gold Set” of News Articles

- Manually selected set of 2,160 near-duplicate news articles (LA Times, SF Chronicle, Huston Chronicle, etc.), manually clustered into 68 topic directories
- Huge variations in layout and ads added by different sites
**SpotSigs vs. Shingling – Gold Set**

- Using (weighted) Jaccard similarity
- Using Cosine similarity (no pruning)

**Runtime Results – TREC WT10g**

- SpotSigs vs. LSH using I-Match-S as recall base

**Summary – Gold Set**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Memory (MB)</th>
<th>Runtime (ms)</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpotSigs</td>
<td>D=0.4, 0.75</td>
<td>69</td>
<td>1.05</td>
</tr>
<tr>
<td>1-Shingles</td>
<td>D=0.4, 0.75</td>
<td>339</td>
<td>14,157</td>
</tr>
<tr>
<td>3-Shingles</td>
<td>D=0.4, 0.75</td>
<td>339</td>
<td>17,136</td>
</tr>
<tr>
<td>LSH-S</td>
<td>D=0.4, 0.75</td>
<td>180</td>
<td>40,236</td>
</tr>
<tr>
<td>I-Match-S</td>
<td>D=0.4, 0.75</td>
<td>180</td>
<td>44,514</td>
</tr>
<tr>
<td>No-Partitions</td>
<td>D=0.4, 0.75</td>
<td>339</td>
<td>196,749</td>
</tr>
<tr>
<td>No-Pruning/No-Partitions</td>
<td>D=0.4, 0.75</td>
<td>339</td>
<td>10,091,013</td>
</tr>
</tbody>
</table>

**Summary – TREC WT10g**

<table>
<thead>
<tr>
<th>τ</th>
<th>Parameters</th>
<th>Memory (MB)</th>
<th>Runtime (ms)</th>
<th>Relative Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SpotSigs</td>
<td>D=0.4, 0.75</td>
<td>69</td>
<td>1.00</td>
</tr>
<tr>
<td>0.9</td>
<td>339</td>
<td>14,157</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>339</td>
<td>17,136</td>
<td>0.96</td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td>180</td>
<td>40,236</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>180</td>
<td>44,514</td>
<td>0.97</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>339</td>
<td>196,749</td>
<td>0.96</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>339</td>
<td>10,091,013</td>
<td>0.96</td>
<td></td>
</tr>
</tbody>
</table>

**Conclusions & Outlook**

- **Robust Spot Signatures** favor natural-language page components
- **Full-fledged clustering** algorithm, returns complete graph of all near-duplicate pairs
- **Efficient & self-tuning** collection partitioning and inverted index pruning, highly parallelizable deduplication step
- **Surprising:** May outperform linear-time similarity hashing approaches for reasonably high similarity thresholds
- **Future Work:**
  - Efficient (sequential) index structures for disk-based storage
  - Tight bounds for more similarity metrics, e.g., Cosine measure
Related Work

- **Shingling**
  [Broder, Glassman, Manasse & Zweig '97], [Broder '00], [Hod & Zobel '03]

- **Random Projection**
  [Charikar '02], [Henzinger '06]

- **Signatures & Fingerprinting**
  [Manber '94], [Birn, Dori & Garcia-Molina '95], [Shivakumar '95], [Manku '06]

- **Constraint-based Clustering**
  [Klein, Kamvar & Manning '02], [Tang & Callan '06]

- **Similarity Hashing**
  - i-Match: [Chowdhury, Frieder, Grossman & McCabe '02], [Chowdhury '04]
  - LSH: [Indyk & Motwani '98], [Indyk, Gionis & Motwani '99]
  - MinHashing: [Indyk '01], [Broder, Charikar & Mitzenmacher '03]

- **Various filtering techniques**
  - Entropy-based: [Büttcher & Clarke '06]
  - IDF, rules & constraints, ...