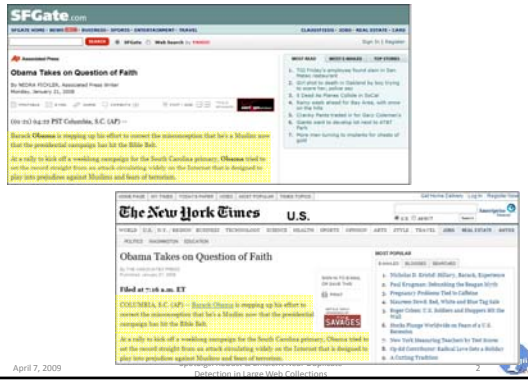
SpotSigs

Robust & Efficient Near Duplicate Detection in Large Web Collections

Martin Theobald
Jonathan Siddharth
Andreas Paepcke

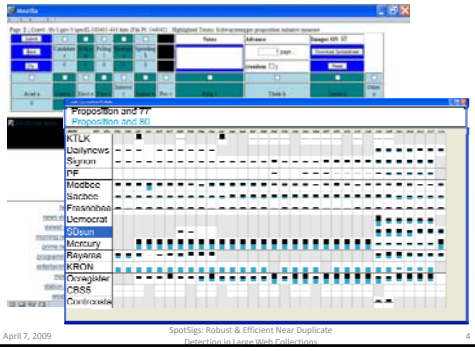
Stanford University
Sigir 2008, Singapore

Near-Duplicate News Articles (I)



Near-Duplicate News Articles (II)

Our Setting



… 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.)

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**

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** } **Spot Signature *s***
    antecedent **nearby n-gram**

Spot Signature Extraction

- **"Localized" signatures:**
  n-grams close to a stopword antecedent
  - E.g.: **that:presidential:campaign:hit**
  - Parameters:
    - Predefined list of (stopword) **antecedents**
    - Spot distance ***d***, chain length ***c***

→ Spot Signatures occur **uniformly** and **frequently** throughout any piece of natural-language text
→ **Hardly** occur in navigational web page components or ads

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*** = *{a:rally:kick, a:weeklong:campain, the:south:carolina, the:record:straight, an:attack:circulating, the:internet:designed, is:designed:play}*

(for antecedents ***{a, the, is}***, uniform spot distance ***d=1***, chain length ***c=2***)

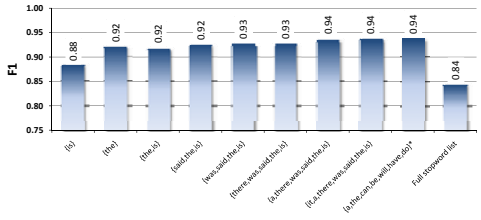
Signature Extraction Algorithm

- **Simple & efficient sliding window technique**

```
input: token stream tokens, chain length c, spot distance d
spots ← ∅
for i ← 0 to len(tokens) do
  if tokens[i] ∈ antecedents then
    chain ← ∅
    k ← i + d
    for j ← 0 to c do
      while k < len(tokens) and tokens[k] ∈ stopwords do
        k ← k + 1
      end while
      if not tokens[k] ∈ stopwords then
        chain ← chain ∪ {tokens[k]}
      end if
      k ← k + d
    end for
    spots ← spots ∪ {chain}
  end if
end for
return spots
```

→ **O(|tokens|) runtime**
→ Largely independent of input format (maybe remove markup)
→ 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

Done?

How to deduplicate a large collection *efficiently*?

- Given $\{S_1,\ldots,S_N\}$ Spot Signature sets
  - For each S_i, find all similar signature sets $S_{i1},\ldots,S_{ik}$ with similarity $\boldsymbol{sim(S_i, S_{ij}) \geq \tau}$
- 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?

→ **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

$$sim(A,B) = |A \cap B| / |A \cup B|$$

- Upper bound

$$sim(A,B) = \frac{|A \cap B|}{|A \cup B|} \leq \frac{\min(|A|,|B|)}{\max(|A|,|B|)}$$

$$|A \cap B| \leq \min(|A|,|B|)$$

$$|A \cup B| \geq \max(|A|,|B|)$$

→ $sim(A,B) \leq \frac{|A|}{|B|}$ *(for $|B| \geq |A|$, w.l.o.g.)*

→ Never compare signature sets *A*, *B* with $\boldsymbol{|A|/|B| < \tau}$ *i.e.* $\boldsymbol{|B|-|A| > (1-\tau)\,|B|}$

Multi-set Generalization

- Consider *weighted* Jaccard similarity

$$\widetilde{sim}(A,B) = \frac{\sum_{s_j \in A \cap B} \min\left(freq_A(s_j), freq_B(s_j)\right)}{\sum_{s_j \in A \cup B} \max\left(freq_A(s_j), freq_B(s_j)\right)}$$

- Upper bound

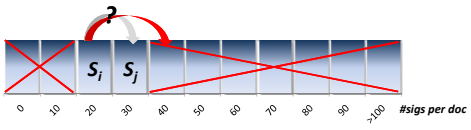
$$\widetilde{sim}(A,B) \leq \frac{\min\left(\sum_{s_j \in A} freq_A(s_j), \sum_{s_j \in B} freq_B(s_j)\right)}{\max\left(\sum_{s_j \in A} freq_A(s_j), \sum_{s_j \in B} freq_B(s_j)\right)}$$

$|A|$ $|B|$

→ Still skip pairs *A*, *B* with $\boldsymbol{|B|-|A| > (1-\tau)\,|B|}$

Partitioning the Collection

- Given a similarity threshold τ, 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

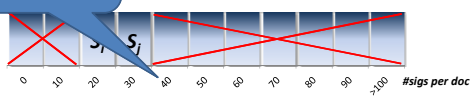


… **but:** there are many possible partitionings, s.t.
- **(A)** any similar pair is (at most) mapped into two neighboring partitions

Partitioning the Collection

- Given a similarity threshold τ, 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

Also: Partition widths should be a function of τ



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

Optimal Partitioning

- Given τ, 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) ***(minimality)***

→ But expensive to solve *exactly* …

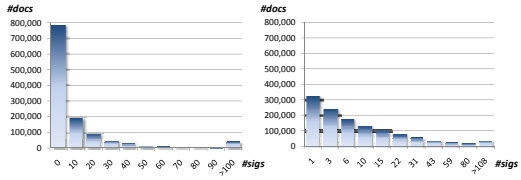
Approximate Solution

"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. $\boldsymbol{p_{k+1} - p_k > (1 - \tau)p_{k+1}}$*"*

E.g. *(for* $\boldsymbol{\tau=0.7}$*)*: p_0=1, p_1=3, p_2=6, p_3=10,…, p_7=43, p_8=59,…

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

Partitioning Effects



→ Optimal partitioning approach even smoothes skewed bucket sizes

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

… but

- **Comparisons within partitions still quadratic!**

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

Inverted Index Pruning

Pass 1:

- For each partition, create an inverted index:
  - For each Spot Signature s_j
    - Create inverted list L_j with pointers to documents d_i containing s_j
    - Sort inverted list in descending order of $freq_i(s_j)$ in d_i

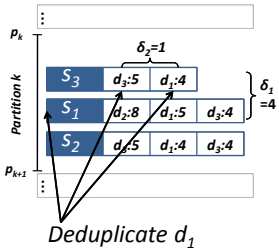
the:campaign	d_7:8	d_1:5	d_5:4	...		
an:attack	d_6:6	d_2:6	d_7:4	d_5:3	d_1:3	...

Partition k

Pass 2:

- For each document d_i, find its partition, then:
  - Process lists in descending order of $|L_j|$
  - ***Maintain two thresholds:***
    - δ_1 – Minimum length distance to any document in the next list
    - δ_2 – Minimum length distance to next document within the current list
  - ***Break if $\delta_1 + \delta_2 > (1-\tau)|d_i|$, also iterate into right neighbor partition***

Deduplication Example



Given:

$d_1 = \{s_1{:}5, s_2{:}4, s_3{:}4\}$, $|d_1|=13$
$d_2 = \{s_1{:}8, s_2{:}4\}$, $|d_2|=12$
$d_3 = \{s_1{:}4, s_2{:}5, s_3{:}5\}$, $|d_3|=13$

Threshold: $\tau = 0.8$
Break if: $\delta_1 + \delta_2 > (1-\tau)|d_i|$

S_3: 1) $\delta_1=0$, $\delta_2=1 \rightarrow$ ***sim(d_1,d_3) = 0.8***
2) $d_1=d_1 \rightarrow$ *continue*
3) $\delta_1=4$, $\delta_2=0 \rightarrow$ ***break!***

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!**

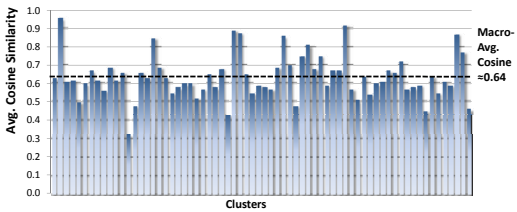
Experiments

- **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 IDF filter for signatures, remove most frequent & infrequent signatures

Competitors

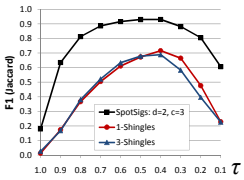
- **Shingling** [Broder, Glassman, Manasse & Zweig '97]
  - N-gram sets/vectors compared with Jaccard/Cosine similarity
    → in between **O (n^2 m)** and **O (n m)** runtime (using LSH for matching)
- **I-Match** [Chowdhury, Frieder, Grossman & McCabe '02]
  - Employs a single SHA-1 hash function
  - Hardly tunable
    → **O (n m)** runtime
- **Locality Sensitive Hashing (LSH)** [Indyk, Gionis & Motwani '99], [Broder et al. '03]
  - Employs *l* (random) hash functions, each concatenating *k* **MinHash** signatures
  - Highly tunable
    → **O (k *l* n m)** runtime
- **Hybrids** of I-Match and LSH with Spot Signatures (**I-Match-S** & **LSH-S**)

"Gold Set" of News Articles

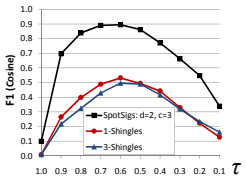


- Manually selected set of 2,160 near-duplicate news articles (**LA Times, SF Chronicle, Huston Chronicle**, etc.), manually clustered into **68 topic directories**
- **Huge variations in layout and ads** added by different sites

SpotSigs vs. Shingling – Gold Set

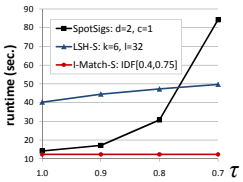


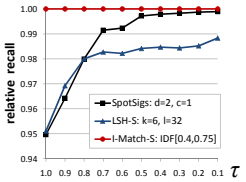
Using (weighted) Jaccard similarity



Using Cosine similarity (no pruning!)

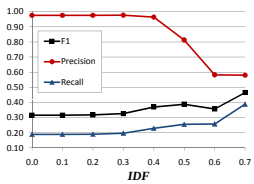
Runtime Results – TREC WT10g



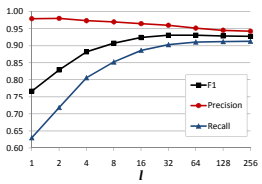


SpotSigs vs. LSH-S using I-Match-S as recall base

Tuning I-Match & LSH on the Gold Set



Tuning I-Match-S: varying IDF thresholds



Tuning LSH-S: varying #hash functions *l* for fixed #MinHashes *k = 6*

→ SpotSigs does not need this tuning step!

Summary – Gold Set

	Parameters	Memory (MB)	Runtime (ms.)	Macro-Avg. F1
SpotSigs	IDF [0.2,0.85]	**2.5**	**1,748**	**0.94**
1-Shingles	IDF [0.2,0.85]	**24.8**	**9,451**	**0.71**
3-Shingles	IDF [0.2,0.85]	**18.6**	**9,202**	**0.69**
LSH-S	IDF [0.2,0.85] k = 6, l = 32	**2.0**	**710**	**0.93**
I-Match	IDF [0.4,0.75]	**0.1**	**581**	**0.05**
I-Match-S	IDF [0.4,0.75]	**0.1**	**284**	**0.37**

Summary of algorithms at their best F1 spots (τ = 0.44 for SpotSigs & LSH)

Summary – TREC WT10g

	τ	Parameters	Memory (MB)	Runtime (ms.)	Relative Recall
I-Match-S	n/a	IDF [0.4,0.75]	**49**	**12,295**	**1.00**
SpotSigs	1.0 0.9	IDF [0.4,0.75]	**339** **339**	**14,157** **17,136**	**0.95** **0.96**
LSH-S	1.0 0.9	IDF [0.4,0.75] k = 6, l = 32	**180** **180**	**40,226** **44,514**	**0.95** **0.97**
No-Partitions **No-Pruning/** **No-Partitions**	0.9 0.9	IDF [0.4,0.75] IDF [0.4,0.75]	**339** **339**	**196,749** **10,090,013**	**0.96** **0.96**

Relative recall of SpotSigs & LSH using I-Match-S as recall base at τ = 1.0 and τ = 0.9

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**
  [Manbar '94], [Brin, Davis & Garcia-Molina '95], [Shivakumar '95], [Manku '06]
- **Constraint-based Clustering**
  [Klein, Kamvar & Manning '02], [Yang & 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, …