A Comparison and Generalization of Blocking and Windowing Algorithms for Duplicate Detection

Uwe Draisbach
uwe.draisbach@fernuni-hagen.de

Felix Naumann
naumann@hpi.uni-potsdam.de
Introduction Duplicate Detection

- **Duplicate Detection**
  - Find all pairs of tuples that represent the same real-world object
  - Number of comparisons = \( \frac{n^2 - n}{2} \)
  - 10,000 tuples result in approx. 50 Mio. comparisons

- **Challenge**
  - *Efficiency*: reduce the number of comparisons
  - *Effectiveness*: find a similarity measure that classifies pairs of tuples correctly as duplicates or non-duplicates
Blocking and Windowing Algorithms

- **Blocking:**

  - Sorting:
  
  - Building disjoint blocks:

  - Duplicate detection within blocks:

- **Sorted Neighborhood Method [HS98]:**

  - Sorting:
  
  - Slide window over sorted tuples:

  - Search for duplicates within the windows:

Sorted Blocks | Draisbach, Naumann | 24. August 2009
Comparing Blocking and Windowing

Window size: 3
Block size: 5

Sorted tuples

Tuples 1 & 5 are only compared using Blocking

Tuples 16 & 14 are only compared using SNM
A Comparison and Generalization of Blocking and Windowing Algorithms for Duplicate Detection

Increasing window size to approximate Blocking

Window size: 5
Block size: 5

Sorted tuples

SNM
Blocking
A Comparison and Generalization of Blocking and Windowing Algorithms for Duplicate Detection

Overlapping blocks to approximate Windowing

Window size: 3
Block size: 5

Sorted tuples
Sorted Blocks Method

- **Approach**
  1. Sort tuples
  2. Build disjoint partitions
  3. Perform complete comparison within partitions
  4. Overlap partitions
  5. Slide fixed size window across sorted tuples within overlap

- **Overlap**
  - Parameter $o =$ number of tuples from one partition that are part of the overlap
  - Overlap size = $2o$
  - Size of window = $o+1$
Sorted Blocks Method

Complete comparison within partitions

Quadratic complexity
Sorted Blocks Method

Complete comparison within partitions

$P_1$  

$P_2$  

$P_3$  

$P_4$

$O_{P_1,P_2} = 2o$

$w = o + 1 = 3$

Linear complexity

Comparisons within overlap
Sorted Blocks Method

Complete comparison within partitions

$\omega = 2$

$w = \omega + 1 = 3$

$|O_{P_1, P_2}| = 2\omega$

$|O_{P_2, P_3}| = 2\omega$

$|O_{P_3, P_4}| = 2\omega$

Comparisons within overlap
### Complexity Analysis

<table>
<thead>
<tr>
<th>Method</th>
<th>Blocking</th>
<th>Windowing</th>
<th>Sorted Blocks (fixed partition size)</th>
<th>Full enumeration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key generation</strong></td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>---</td>
</tr>
<tr>
<td><strong>Sorting</strong></td>
<td>$O(n \log n)$</td>
<td>$O(n \log n)$</td>
<td>$O(n \log n)$</td>
<td>---</td>
</tr>
<tr>
<td><strong>Detection</strong></td>
<td>$O(n^2/2b)$</td>
<td>$O(wn)$</td>
<td>$O(nm/2)$</td>
<td>$O(n^2/2)$</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>$O(n (n^2/2b + \log n))$</td>
<td>$O(n(w + \log n))$</td>
<td>$O(n(m/2 + \log n))$</td>
<td>$O(n^2/2)$</td>
</tr>
</tbody>
</table>

- $n = \text{number of tuples}$
- $b = \text{number of blocks}$
- $w = \text{window size}$
- $m = \text{partition size}$
Experiment

- Partitioning methods implemented in Java/MySQL environment
- Data Set
  - 9,763 records with audio CD information from freeDB
  - 298 real duplicates detected manually

- Sorting key
  - Concatenation of the first three letters of attributes `artist1`, `title1` and `track01`
Experiment

- Similarity Measure
  - Average similarity of attributes *artist1*, *title1* and *track01*
  - Not complex, but sufficient for comparing partition methods

\[
f(t_1, t_2) = \frac{s(t_1[\text{Artist1}], t_2[\text{Artist1}]) + s(t_1[\text{Title1}], t_2[\text{Title1}]) + s(t_1[\text{Track01}], t_2[\text{Track01}])}{3}
\]

with:

\[
s(x, y) = \begin{cases} 
  1, & \text{if } x=\text{SubstringOf}(y) \text{ or } y=\text{SubstringOf}(x) \\
  \text{threshold of similarity function, if } \text{IsNull}(x) \text{ or } \text{IsNull}(y) \\
  1 - \frac{\text{edit\_distance}(x, y)}{\max\{|x|, |y|\}}, & \text{otherwise}
\end{cases}
\]

- Fixed size blocks/partitions have been used for simplicity
- \( o=2 \) delivers the best results for this data set
**Experiment - Precision**

- **Precision**: proportion of correctly identified duplicates
  - No. correctly identified duplicates / No. all identified duplicates

> Partitioning methods have similar precision

> Partition methods perform better than exhaustive comparison
Experiment - Recall

- Recall: proportion of identified real-world duplicates
  - No. correctly identified duplicates / No. all duplicates

Exhaustive comparison is upper bound

Sorted Blocks performs better for small partitions, but is not monotonically increasing

Blocking delivers worst results
Conclusion

- **Results**
  - Sorted Neighborhood outperforms Blocking
  - Sorted Blocks parameterizes the degree of overlap from none (Blocking) to $w-1$ (Sorted Neighborhood)
  - Sorted Blocks outperforms Sorted Neighborhood slightly

- **Open issues for future research**
  - Confirm results by additional experiments with other data sets
  - Use variable partition sizes for experiments
  - Allow dynamic adaption of overlap
  - Multipass
Experiment – F-Measure

- F-Measure = harmonic mean of precision and recall

Curve shapes are similar to those for recall

SNM & Sorted Blocks perform better than Blocking