A Two Stage Similarity-aware Indexing for Large Scale Real-time Entity Resolution

Shouheng Li  
Supervisor: Huizhi (Elly) Liang  
_u4713006@anu.edu.au_  
The Australian National University
Outline

- Introduction
- Related work
- Proposed approach
- Experimental Results
- Conclusion
Introduction

➢ Entity Resolution
  ➢ Data integration: linking different data sources
  ➢ Find data records refer to the same real world entity
  ➢ Example: data records from hospital and Facebook

➢ Real-time challenges
  ➢ Large scale datasets
  ➢ Quick response on user behaviours
    ➢ Online credit card inquiry
Related work

1. Locality Sensitive Hashing (LSH)
2. Indexing/Blocking
Related work: locality sensitive hashing (LSH)

- Probability based dimension reduction
- Locality sensitive hashing
  - Minhash
  - Jaccard similarity
  - Simulated using hash functions

\[ J(r_1, r_2) = \frac{|r_1 \cap r_2|}{|r_1 \cup r_2|} \]

<table>
<thead>
<tr>
<th>id</th>
<th>first name</th>
<th>last name</th>
<th>suburb</th>
<th>postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>Christy</td>
<td>Smith</td>
<td>Turner</td>
<td>2612</td>
</tr>
<tr>
<td>r2</td>
<td>Kristine</td>
<td>Smyth</td>
<td>Turner</td>
<td>2612</td>
</tr>
</tbody>
</table>

Problem: a rough filter, need to do pairwise comparison afterwards
Related work: Indexing/blocking

- **Purpose**
  - Reduce number of comparison

- **How?**
  - Only compare items in the same block

- **Examples**
  - Encoding based indexing
  - Sorted neighbourhood indexing
  - Similarity-aware indexing*
    - Pre-calculate similarities

<table>
<thead>
<tr>
<th>id</th>
<th>first name</th>
<th>last name</th>
<th>suburb</th>
<th>postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>Christy</td>
<td>Smith</td>
<td>Turner</td>
<td>2612</td>
</tr>
<tr>
<td>r2</td>
<td>Kristine</td>
<td>Smyth</td>
<td>Turner</td>
<td>2612</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>First name</th>
<th>Double-metaphone</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>smith</td>
<td>sm0</td>
</tr>
<tr>
<td>r2</td>
<td>smyth</td>
<td>sm0</td>
</tr>
<tr>
<td>r3</td>
<td>robert</td>
<td>rpt</td>
</tr>
</tbody>
</table>

Problem: if a block is huge, comparison can still be expensive

*Dynamic Similarity-Aware Inverted Indexing for Real-Time Entity Resolution
Banda Ramadan, Peter Christen, Huizhi Liang, Ross Gayler, and David Hawking.
DMAapps 2013, PAKDD'13
Proposed method

➢ Research gap
  ➢ LSH+Indexing?

➢ A two-stage approach
  ➢ Stage One: Locality sensitive hashing (LSH)
  ➢ Stage two: Similarity-aware indexing (SAI)
Stage 1: Locality sensitive hashing

- Locality Sensitive Hashing (LSH)
  - Min-hashing
    - Jaccard similarity
      \[ P_r(\min(\pi(r_i)) = \min(\pi(r_j))) = J(r_i, r_j) = \frac{|r_i \cap r_j|}{|r_i \cup r_j|} \]
  - Signature
    - \( \text{sig}(r) = (h_1(r), h_2(r), \ldots, h_n(r)) \)

**Example:**

<table>
<thead>
<tr>
<th>id</th>
<th>first name</th>
<th>last name</th>
<th>suburb</th>
<th>postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>Christy</td>
<td>Smith</td>
<td>Turner</td>
<td>2612</td>
</tr>
<tr>
<td>r2</td>
<td>Kristine</td>
<td>Smyth</td>
<td>Turner</td>
<td>2612</td>
</tr>
</tbody>
</table>

**LSH Index**

**Hash values:**
- r1: h1, h2, h3, h4
- r2: h3, h4, h5, h6

**Signature banding:**
- s1: h1, h2
- s2: h3, h4
- s3: h5, h6

**Same:**
- r1 and r2 have the same hash values in s1 and s2.
Stage 2: Similarity-aware indexing

- Fast pair-wise comparison using pre-calculated similarities
- Three indexes: Similarity Index (SI), Block Index (BI), LSH Index (LI)

<table>
<thead>
<tr>
<th>id</th>
<th>first name</th>
<th>last name</th>
<th>suburb</th>
<th>postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>Christy</td>
<td>Smith</td>
<td>Turner</td>
<td>2612</td>
</tr>
<tr>
<td>r2</td>
<td>Kristine</td>
<td>Smyth</td>
<td>Turner</td>
<td>2612</td>
</tr>
</tbody>
</table>
Experimental Evaluation

**Dataset 1**: North Carolina voter registration dataset from Oct 2011 to Dec 2012, 2,567,644 records

**Dataset 2**: Australian telephone directory dataset from 2002, modified, 25% duplications, 3,458,758 records

**Attributes**: first name, last name, suburb, postcode

**Setting**: 50% used for build, 50% used for query, 64 hash functions with 4 bits for each band, returns top 100 as query results

**Measurements**:
- 1. Processing time:
  - average time spent on queries, time used to build indexes
- 2. Memory usage
- 3. Recall
  - \[ \text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \]
  - Measures the capability of finding true matches
Experimental Results

Dataset 1: North Carolina voter registration dataset

Dataset 2: Australian telephone directory dataset
Conclusion

- Real-time large scale entity resolution
- A two-stage approach
  - Combination of LSH and SAI
- Future work
  - Combination with sorted neighborhood indexing
  - Application on real-time recommender systems
Any questions?