Large-Scale Similarity Joins With Guarantees

Rasmus Pagh
IT University of Copenhagen

EDBT/ICDT
March 25, 2015
Similarity join example 1
(record linkage)

<table>
<thead>
<tr>
<th>Country</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>IBM</td>
</tr>
<tr>
<td>USA</td>
<td>Microsoft</td>
</tr>
<tr>
<td>Germany</td>
<td>SAP</td>
</tr>
<tr>
<td>China</td>
<td>Baidu</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Token</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft</td>
<td>1</td>
</tr>
<tr>
<td>SAP SE</td>
<td>2</td>
</tr>
<tr>
<td>I.B.M.</td>
<td>3</td>
</tr>
<tr>
<td>baidu.com</td>
<td>4</td>
</tr>
</tbody>
</table>
#### Similarity join example 1
(record linkage)

<table>
<thead>
<tr>
<th>Country</th>
<th>Name</th>
<th>Token</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>IBM</td>
<td>Mircosoft</td>
<td>1</td>
</tr>
<tr>
<td>USA</td>
<td>Microsoft</td>
<td>SAP SE</td>
<td>2</td>
</tr>
<tr>
<td>Germany</td>
<td>SAP</td>
<td>I.B.M.</td>
<td>3</td>
</tr>
<tr>
<td>China</td>
<td>Baidu</td>
<td>baidu.com</td>
<td>4</td>
</tr>
</tbody>
</table>
Similarity join example 2 (classification)

<table>
<thead>
<tr>
<th>Class</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mammalia</td>
<td>Cat</td>
</tr>
<tr>
<td>Mammalia</td>
<td>Dog</td>
</tr>
<tr>
<td>Reptilia</td>
<td>Snake</td>
</tr>
<tr>
<td>Aves</td>
<td>Parrot</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Image</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image](Images by Bodlina and Marek Szczepanek)</td>
<td>1</td>
</tr>
<tr>
<td>![Image](Images by Bodlina and Marek Szczepanek)</td>
<td>2</td>
</tr>
<tr>
<td>![Image](Images by Bodlina and Marek Szczepanek)</td>
<td>3</td>
</tr>
<tr>
<td>![Image](Images by Bodlina and Marek Szczepanek)</td>
<td>4</td>
</tr>
</tbody>
</table>
### Similarity join example 2
(classification)

<table>
<thead>
<tr>
<th>Class</th>
<th>Name</th>
<th>Features</th>
<th>Image</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mammalia</td>
<td>Cat</td>
<td>0101101</td>
<td><img src="image1.png" alt="Cat" /></td>
<td>1</td>
</tr>
<tr>
<td>Mammalia</td>
<td>Dog</td>
<td>1001000</td>
<td><img src="image2.png" alt="Dog" /></td>
<td>2</td>
</tr>
<tr>
<td>Reptilia</td>
<td>Snake</td>
<td>1101101</td>
<td><img src="image3.png" alt="Snake" /></td>
<td>3</td>
</tr>
<tr>
<td>Aves</td>
<td>Parrot</td>
<td>1101110</td>
<td><img src="image4.png" alt="Parrot" /></td>
<td>4</td>
</tr>
</tbody>
</table>

Images by Bodilina and Marek Szczepanek
Similarity join example 2
(classification)

<table>
<thead>
<tr>
<th>Class</th>
<th>Name</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mammalia</td>
<td>Cat</td>
<td>1100101</td>
</tr>
<tr>
<td></td>
<td>Dog</td>
<td>1101101</td>
</tr>
<tr>
<td>Reptilia</td>
<td>Snake</td>
<td>1011000</td>
</tr>
<tr>
<td>Aves</td>
<td>Parrot</td>
<td>1010111</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Features</th>
<th>Image</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0101101</td>
<td><img src="image1" alt="Dog" /></td>
<td>1</td>
</tr>
<tr>
<td>1001000</td>
<td><img src="image2" alt="Snake" /></td>
<td>2</td>
</tr>
<tr>
<td>1101101</td>
<td><img src="image3" alt="Parrot" /></td>
<td>3</td>
</tr>
<tr>
<td>1101110</td>
<td><img src="image4" alt="Girl" /></td>
<td>4</td>
</tr>
</tbody>
</table>
## Similarity join example 2 (classification)

<table>
<thead>
<tr>
<th>Class</th>
<th>Name</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mammalia</td>
<td>Cat</td>
<td>1100101</td>
</tr>
<tr>
<td>Mammalia</td>
<td>Dog</td>
<td>1101101</td>
</tr>
<tr>
<td>Reptilia</td>
<td>Snake</td>
<td>1011000</td>
</tr>
<tr>
<td>Aves</td>
<td>Parrot</td>
<td>1010111</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Features</th>
<th>Image</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0101101</td>
<td><img src="image1.png" alt="Cat Image" /></td>
<td>1</td>
</tr>
<tr>
<td>1001000</td>
<td><img src="image2.png" alt="Dog Image" /></td>
<td>2</td>
</tr>
<tr>
<td>1101101</td>
<td><img src="image3.png" alt="Snake Image" /></td>
<td>3</td>
</tr>
<tr>
<td>1101110</td>
<td><img src="image4.png" alt="Parrot Image" /></td>
<td>4</td>
</tr>
</tbody>
</table>

Images by Bodlina and Marek Szczepanek
More examples
Similarity Search in High Dimensions via Hashing

Aristides Gionis*, Piotr Indyk†, Rajeev Motwani‡

Department of Computer Science
Stanford University
Stanford, CA 94305

{gionis,indyk,rajeev}@cs.stanford.edu

Abstract

The nearest- or near-neighbor query problems arise in a large variety of database applications, usually in the context of similarity searching. Of late, there has been increasing interest in building search/index structures for performing similarity search over high-dimensional data e.g., image databases, document collections, time-series databases, and genome databases. Unfortunately, all known techniques for solving this problem fall prey to the "curse of dimensionality." That is, the data structures scale poorly with data dimensionality; in fact, if the number of dimensions exceeds 10 to 20, searching in k-d trees and related structures involves the inspection of a large fraction of the database, thereby doing no better than brute-force linear search. It has been suggested that since the selection of features and the choice of a distance metric in typical applications is rather heuristic, determining an approximate nearest neighbor should suffice for most practical purposes. In this paper, we examine a novel scheme for approximate similarity search based on hashing. The basic idea is to hash the points from the database so as to ensure that the probability of collision is much higher for objects that are close to each other than for those that are far apart. We provide experimental evidence that our method gives significant improvement in running time over other methods for searching in high-dimensional spaces based on hierarchical tree decomposition. Experimental results also indicate that our scheme scales well even for a relatively large number of dimensions (more than 50).

1 Introduction

A similarity search problem involves a collection of objects (e.g., documents, images) that are characterized by a collection of relevant features and represented as points in a high-dimensional attribute space; given queries in the form of points in this space, we are required to find the nearest (most similar) object to the query. The particularly interesting and well-studied case is the d-dimensional Euclidean space. The problem is of major importance to a variety of applications; some examples are: data compression [20], databases and data mining [21], information retrieval [11, 16, 38], image and video databases [15, 17, 37, 42], machine learning [7], pattern recognition [9, 13], and statistics and data analysis [12, 27]. Typically, the features of the objects of interest are represented as points in Rd and a distance metric is used to measure similarity of objects. The basic problem then is to perform indexing or similarity searching for query objects. The number of features (i.e., the dimensionality) ranges anywhere from tens to thousands. For example, in multimedia applications such as IBM’s QBIC (Query by Image Content), the number of features could be several hundreds [15, 17]. In information retrieval for text documents, vector-space representations involve several thousands of dimensions, and it is considered to be a dramatic improvement that dimension reduction techniques, such as the Karhunen–Loève transform [26, 20] (also known as principal components analysis [22] or latent semantic indexing [11]), can reduce the dimensionality to a mere few hundreds.
Similarity Search in High Dimensions via Hashing

Aristides Gionis

Department of Computer Science

Stanford University

Stanford, CA 94305

{gionis,indyk,rajeev}@cs.stanford.edu

Abstract

The nearest-neighbor near-neighbor query problems arise in a large variety of database applications, usually in the context of similarity searching. Of late, there has been increasing interest in building search index structures for performing similarity search over high-...
Beyond databases

- Association rule mining
- Automation
- Bio-chemistry (finding motifs)
- Bio-informatics (homology search)
- Clustering
- Computer vision and pattern recognition
- Data cleaning
- Data stream computation
- Data privacy
- First story detection (with application to twitter)
- Identifying trends in time series
- Linear algebra
- Motion planning for robots
- Near-duplicate detection
- News personalization (collaborative filtering)
- Privacy preserving data mining
- Search engines for 3D models
- Sensor networks
- …
Talk outline

• The algorithmic problem.
• Curse of dimensionality and approximation.
• Techniques for candidate set generation
Talk outline

• The algorithmic problem.
• Curse of dimensionality and approximation.
• Techniques for candidate set generation
  A. Locality-sensitive hashing
Talk outline

• The algorithmic problem.

• Curse of dimensionality and approximation.

• Techniques for candidate set generation
  A. Locality-sensitive hashing
  B. Cache-efficiency via recursion
Talk outline

• The algorithmic problem.

• Curse of dimensionality and approximation.

• Techniques for candidate set generation
  A. Locality-sensitive hashing
  B. Cache-efficiency via recursion
  C. Exact similarity join: Achieving 100% recall
Algorithmic problem

- Given distance function $d$ and tolerance $r$ compute:

$$Q \bowtie_r S = \{(q, x) \in Q \times S \mid d(q, x) \leq r\}$$
Algorithmic problem

• Given distance function $d$ and tolerance $r$ compute:

$$Q \otimes_r S = \{ (q, x) \in Q \times S \mid d(q, x) \leq r \}$$

• This talk:

Consider $n$ vectors in $\{0,1\}^D$ and Hamming distance.

$$q = 1100101$$

$$x = 1101101$$

$$d(q, x) = 1$$
Similarity join in a picture

$R = Q \bowtie_r S$
Similarity join in a picture

\[ R = Q \bowtie_r S \]
Similarity join in a picture

\[ R = Q \bowtie_r S \]
Similarity join in a picture

in high dimensional space ($D=80$)
$c$-approximate similarity join

$C \supseteq Q \bowtie_r S$
c-approximate similarity join

\[ C \supseteq Q \bowtie_r S \]
\textit{c}-approximate similarity join

\[ C \supseteq Q \bowtie_r S \]

False positive pairs, filter away in time

\[ |Q \bowtie_{cr} S| \]
Why approximation?
Why approximation?

Because of the CURSE of dimensionality!
Why approximation?

- **Silvestri et al. 2015:**
  \[ \exists \text{ similarity join algorithm using time } n^{1+o(1)} 2^{o(D)} \]
  \[ \Rightarrow \]
  \[ k\text{-SAT w. } n \text{ variables can be solved in time } 2^{n/2+o(1)} \]
Why approximation?

• Silvestri et al. 2015:

\[ \exists \text{ similarity join algorithm using time } n^{1+o(1)} 2^{o(D)} \]

\[ \Rightarrow \]

\[ k\text{-SAT w. } n \text{ variables can be solved in time } 2^{n/2+o(1)} \]

Huge research effort has been devoted to k-SAT, but so far failed to even reach $1.99^n$ for general $k$. 
Why approximation?

• Silvestri et al. 2015:
  ∃ similarity join algorithm using time $n^{1+o(1)} 2^{o(D)}$
  ⇒
  $k$-SAT w. $n$ variables can be solved in time $2^{n/2+o(1)}$

Conclusion: A highly efficient algorithm for similarity join with no approximation of distances would be very surprising.

Huge research effort has been devoted to $k$-SAT, but so far failed to even reach $1.99^n$ for general $k$. 
Effect of approximation

MNIST data set (unary encoding)

Fraction of all pairs within Hamming distance
Effect of approximation
MNIST data set (unary encoding)

Fraction of all pairs within Hamming distance
Effect of approximation

MNIST data set (unary encoding)

Fraction of all pairs within Hamming distance

\[ Q \otimes_{cr} S \]

\[ Q \otimes_r S \]
Effect of approximation

MNIST data set (unary encoding)

Fraction of all pairs within Hamming distance
Effect of approximation
MNIST data set (unary encoding)

In rest of the talk:
Assume $|Q \bowtie_{cr} S|$ is not much larger than $|Q \bowtie_r S|$
Part A.
Locality-sensitive hashing (LSH)
Locality-sensitive hashing

Idea: Consider projection onto a random subset of dimensions, each chosen with probability $p$
Locality-sensitive hashing

Idea: Consider projection onto a random subset of dimensions, each chosen with probability $p$
### Locality-sensitive hashing

**Idea**: Consider projection onto a random subset of dimensions, each chosen with probability $p$

$$h(x) = x \land a$$

```c
int lsh(int x) { return x & a; }
```

Gionis et al. VLDB 1999
Locality-sensitive hashing

Idea: Consider projection onto a random subset of dimensions, each chosen with probability $p$

$h(x) = x \land a$

```
int lsh(int x) { return x & a; }
```
Locality-sensitive hashing

\[ h(x) = x \land a \]

```
int lsh(int x) {return x & a}
```

Idea: Consider projection onto a random subset of dimensions, each chosen with probability \( p \)

Repeat enough times that pairs at distance \( r \) produce at least one collision
Locality-sensitive hashing

Idea: Consider projection onto a random subset of dimensions, each chosen with probability $p$

Candidate pair

Repeat enough times that pairs at distance $r$ produce at least one collision

$h_i(x) = x \land a_i$

int lsh(int x, int i) {return x & a[i]}
Chance of collision?

\[
x \land a_1 \quad ? \quad q \land a_1
\]

\[
x \land a_2 \quad ? \quad q \land a_2
\]

\[
x \land a_3 \quad ? \quad q \land a_3
\]

\[
\vdots \quad \vdots
\]

\[
x \land a_t \quad ? \quad q \land a_t
\]
Chance of collision?

\[ x \land a_1 \quad ? \quad q \land a_1 \quad \text{prob. } (1-p)^{d(q,x)} \]

\[ x \land a_2 \quad ? \quad q \land a_2 \quad \text{prob. } (1-p)^{d(q,x)} \]

\[ x \land a_3 \quad ? \quad q \land a_3 \quad \text{prob. } (1-p)^{d(q,x)} \]

\[ \vdots \quad \vdots \]

\[ x \land a_t \quad ? \quad q \land a_t \quad \text{prob. } (1-p)^{d(q,x)} \]
### Chance of Collision?

| $x \land a_1$ | $= ?$ | $q \land a_1$ | prob. $(1-p)^{d(q,x)}$ |
| $x \land a_2$ | $= ?$ | $q \land a_2$ | Probability of no collision: $(1-(1-p)^{d(q,x)})^t$ |
| $x \land a_3$ | $= ?$ | $q \land a_3$ |
| $\vdots$ | $\vdots$ |
| $x \land a_t$ | $= ?$ | $q \land a_t$ | prob. $(1-p)^{d(q,x)}$ |
Chance of collision?

\[ x \land a_1 = q \land a_1 \quad \text{prob.} \ (1-p)^{d(q,x)} \]

\[ x \land a_2 = q \land a_2 \]

\[ x \land a_3 = q \land a_3 \]

\[ \vdots \quad \vdots \]

Probability of no collision:

\[ (1-(1-p)^{d(q,x)})^t \]

From math book: \( (1-x)^{1/x} \rightarrow 1/e \) as \( x \rightarrow 0 \)
Summary of analysis

Analysis (GIM ’99): Each bit of $a_i$ is 1 with probability $p$.

- Expect $c_p \leq n^2 (1-p)^{cr}$ candidate pairs outside $Q \bowtie_{cr} S$
- Probability of $h(q) = h(x)$ when $d(q, x) = r$ is $(1-p)^r$, so need to hash $t_p = (1-p)^{-r}$ times to achieve constant recall.
- Choose $p$ to balance cost of hashing $t_p$ times and checking $t_p c_p$ candidate pairs not in $Q \bowtie_{cr} S$
Summary of analysis

Analysis (GIM ’99): Each bit of $a_i$ is 1 with probability $p$.

- Expect $c_p \leq n^2 (1-p)^{cr}$ candidate pairs outside $Q \ncong_{cr} S$
- Probability of $h(q)=h(x)$ when $d(q,x)=r$ is $(1-p)^r$, so need to hash $t_p=(1-p)^{-r}$ times to achieve constant recall.
- Choose $p$ to balance cost of hashing $t_p$ times and checking $t_p c_p$ candidate pairs not in $Q \ncong_{cr} S$
Summary of analysis

Analysis (GIM ’99): Each bit of $a_i$ is 1 with probability $p$.

- Expect $c_p \leq n^2 (1-p)^{cr}$ candidate pairs outside $Q \bowtie_{cr} S$
- Probability of $h(q)=h(x)$ when $d(q,x) = r$ is $(1-p)^r$, so need to hash $t_p=(1-p)^{-r}$ times to achieve constant recall.
- Choose $p$ to balance cost of hashing $t_p$ times and checking $t_p c_p$ candidate pairs not in $Q \bowtie_{cr} S$
Locality-sensitive hashing

Analysis (GIM ’99): Each bit of $a_i$ is 1 with probability $p$.

- Expect $c_p \leq n^2 (1-p)^{cr}$ candidate pairs outside $Q \bowtie_{cr} S$
- Probability of $h(q)=h(x)$ when $d(q,x) = r$ is $(1-p)^r$, so need to hash $t_p=(1-p)^{-r}$ times to achieve constant recall.
- Choose $p$ to balance cost of hashing $t_p$ times and checking $t_p c_p$ candidate pairs not in $Q \bowtie_{cr} S$
Locality-sensitive hashing

Analysis (GIM ’99): Each bit of $a_i$ is 1 with probability $p$.

- Expect $c_p \leq n^2 (1-p)^{cr}$ candidate pairs outside $Q \asymp_{cr} S$
- Probability of $h(q)=h(x)$ when $d(q,x)=r$ is $(1-p)^r$, so need to hash $t_p=(1-p)^r$ times to achieve constant recall.
- Choose $p$ to balance cost of hashing $t_p$ times and checking $t_p c_p$ candidate pairs not in $Q \asymp_{cr} S$

Number of operations is $O(n^{1+1/cD})$, expected; Can be implemented in $O(n^{1+1/cD/B})$ I/Os.
Can LSH work for large data?

• Quote from an ICDT paper: “LSH needs large memory space and long processing time to achieve good performance when searching a massive dataset”.
Can LSH work for large data?

• Quote from an ICDT paper: "LSH needs large memory space and long processing time to achieve good performance when searching a massive dataset".

For similarity join, space is linear: Process one hash function at a time.
Can LSH work for large data?

- Quote from an ICDT paper: "LSH needs large memory space and long processing time to achieve good performance when searching a massive dataset".

- Other issues:
  - LSH parameters are pessimistic; chosen to work for worst-case data set.

For similarity join, space is linear: Process one hash function at a time.
Can LSH work for large data?

• Quote from an ICDT paper: “LSH needs large memory space and long processing time to achieve good performance when searching a massive dataset”.

• Other issues:
  - LSH parameters are pessimistic; chosen to work for worst-case data set.
  - Poor use of internal memory: A simple nested loop join that uses $O(D^2n^2/(MB))$ I/Os is often better.
On pessimism

- LSH chosen to ensure few collisions at distance $cr$, even in worst-case scenarios:
On pessimism

- LSH chosen to ensure few collisions at distance $cr$, even in worst-case scenarios:

  **Price paid:** Also small collision probability at distance $r$, so need many repetitions.
Part B. Cache-efficiency via recursion
Cautious LSH
Joint work with Pham, Silvestri, and Stöckel

Idea: Apply a weak LSH with constant collision probability.
Cautious LSH
Joint work with Pham, Silvestri, and Stöckel

Idea: Apply a weak LSH with constant collision probability.
Cautious LSH

Joint work with Pham, Silvestri, and Stöckel
Cautious LSH
Joint work with Pham, Silvestri, and Stöckel

\[ Q_0 \lor_r S_0 \]

\[ Q_1 \lor_r S_1 \]
Cautious LSH
Joint work with Pham, Silvestri, and Stöckel

Subproblems of size $M$ or less require no further I/Os
Example, cautious LSH

• Assume weak LSH collision prob. 0.9.
• Prob. that \( q \) and \( x \) collide \( i \) times is \( 0.9^i \).
• If depth is \( \log n \), success probability is \( \approx n^{-0.152} \).
Example, cautious LSH

- Assume weak LSH collision prob. 0.9.
- Prob. that \( q \) and \( x \) collide \( i \) times is \( 0.9^i \).
- If depth is \( \log n \), success probability is \( \approx n^{-0.152} \).
Example 2, cautious LSH

- Assume weak LSH collision prob. 0.5.
- Prob. that $q$ and $x$ collide $i$ times is $0.5^i$. 
Example 2, cautious LSH

- Assume weak LSH collision prob. 0.5.
- Prob. that \( q \) and \( x \) collide \( i \) times is \( 0.5^i \).
- Idea: Recurse \textit{twice} at each node.
Example 2, cautious LSH

- Assume weak LSH collision prob. 0.5.
- Prob. that $q$ and $x$ collide $i$ times is $0.5^i$.
- Idea: Recurse *twice* at each node.

#subproblems containing $q$ and $x$:
1 at each level, expected!
Aside:
Don’t be fooled by great expectations
Aside:

Don’t be fooled by great expectations

*What is the expected TNT equivalent of asteroid impacts during this talk?*
Aside:
Don’t be fooled by great expectations

What is the expected TNT equivalent of asteroid impacts during this talk?

WolframAlpha

100 Teratons / $66 \times 10^6$ years

48.05 kg/s (kilograms per second)

$\approx 0.3 \times$ rate of trash production by New York City ($\approx 1 \times 10^7$ kg/day)
Branching processes

- **Abstract setting:**
  - One pair \((q,x)\) at root problem.
  - Generate \(t\) subproblems such that \((q,x)\) is “reproduced” in each with probability \(\geq 1/t\).
Branching processes

• Abstract setting:
  - One pair \((q, x)\) at root problem.
  - Generate \(t\) subproblems such that \((q, x)\) is “reproduced” in each with probability \(\geq 1/t\).

• What is the probability that \((q, x)\) is extinct at recursive level \(i\)?
Branching processes

• **Abstract setting:**
  - One pair \((q, x)\) at root problem.
  - Generate \(t\) subproblems such that \((q, x)\) is “reproduced” in each with probability \(\geq 1/t\).

• What is the probability that \((q, x)\) is *extinct* at recursive level \(i\)?
  - From theory of branching processes: \(\Omega(1/\sqrt{i})\)
I/O complexity, sketch

recursion

tree
I/O complexity, sketch

Size of all subproblems

n

recursion tree
I/O complexity, sketch

Size of all subproblems

n

nt

recursion
tree
I/O complexity, sketch

Size of all subproblems

$\vdots$

$nt^3$

$nt^2$

$nt$

$n$

recursion tree
I/O complexity, sketch

Size of all subproblems

\[ n \quad nt \quad nt^2 \quad nt^3 \quad \ldots \]

Expected average

#points at distance > cr.

Recursion tree
I/O complexity, sketch

Size of all subproblems

n
nt
nt^2
nt^3

Expected average

#points at distance > cr.

n
n/tn^c

recursion tree
I/O complexity, sketch

Size of all subproblems

\[ n \]
\[ nt \]
\[ nt^2 \]
\[ nt^3 \]
\[ \vdots \]

Expected average

\[ n \]
\[ nt^c \]
\[ nt^{2c} \]
\[ nt^{3c} \]
\[ \vdots \]

#points at distance > cr.
I/O complexity, sketch

Size of all subproblems

n
nt
nt^2
nt^3

Expected average

#points at distance > cr.

n
n/t^c
n/t^{2c}
n/t^{3c}

In-memory computation

recursion

tree
Size of all subproblems

\[ n \]
\[ nt \]
\[ nt^2 \]
\[ nt^3 \]

\[ \log_{tc} \left( \frac{n}{M} \right) \]

Expected average #points at distance > cr.

\[ n \]
\[ n/t^c \]
\[ n/t^{2c} \]
\[ n/t^{3c} \]

In-memory computation

recursion tree
I/O complexity

- Simplified:

\[ O \left( \left( \frac{n}{M} \right)^{1/c} \left( \frac{n}{B} + \frac{|Q \bowtie_{cr} S|}{MB} \right) \right) \text{ I/Os} \]

Assuming \(|Q \bowtie_{cr} S|\) is not much larger than \(|Q \bowtie_r S|\)
**I/O complexity**

- Simplified:

\[
O \left( \left( \frac{n}{M} \right)^{1/c} \left( \frac{n}{B} + \frac{|Q \bowtie_{r} S|}{MB} \right) \right) \text{ I/Os}
\]

Assuming \(|Q \bowtie_{cr} S|\) is not much larger than \(|Q \bowtie_{r} S|\)
I/O complexity

- Simplified:

\[
O \left( \left( \frac{n}{M} \right)^{1/c} \left( \frac{n}{B} + \frac{|Q \bowtie_{cr} S|}{MB} \right) \right) \quad \text{I/Os}
\]

Assuming \( |Q \bowtie_{cr} S| \) is not much larger than \( |Q \bowtie_r S| \)
I/O complexity

- Simplified:

\[ O\left(\left(\frac{n}{M}\right)^{1/c} \left(\frac{n}{B} + \frac{|Q \bowtie r S|}{MB}\right)\right) \text{ I/Os} \]

Assuming \(|Q \bowtie_{cr} S|\) is not much larger than \(|Q \bowtie_r S|\)
I/O complexity

• Simplified:

\[ O\left(\left(\frac{n}{M}\right)^{1/c} \left(\frac{n}{B} + \frac{|Q \bowtie S|_{\leq r}}{MB}\right)\right) \text{ I/Os} \]

• In general:

\[ \tilde{O}\left(\left(\frac{n}{M}\right)^{1/c} \left(\frac{n}{B} + \frac{|Q \bowtie S|_{\leq r}}{MB}\right) + \frac{|Q \bowtie S|_{\leq cr}}{MB}\right) \text{ I/Os} \]
Part C: Exact similarity join
Total recall?

- False negative problem of classical LSH: No collision for a close pair with prob. $P > 0$. 

30
Total recall?

- False negative problem of classical LSH: No collision for a close pair with prob. $P > 0$.
- Recall can be made $1 - P^k$ by repeating $k$ times.
Total recall?

• False negative problem of classical LSH: No collision for a close pair with prob. \( P > 0 \).

• Recall can be made \( 1 - P^k \) by repeating \( k \) times.

• Many recent papers deal with deterministic, exact similarity joins that achieve 100% recall.
Total recall?

- False negative problem of classical LSH: No collision for a close pair with prob. $P > 0$.
- Recall can be made $1 - P^k$ by repeating $k$ times.
- Many recent papers deal with deterministic, exact similarity joins that achieve 100% recall.
Correlated LSH

- **Idea**: Choose hash functions $h_1, h_2, \ldots$ to ensure at least one collision if distance is at most $r$. 

Efficient Exact Set-Similarity Joins

Arvind Arasu  
Microsoft Research  
One Microsoft Way  
Redmond, WA 98052  
arvinda@microsoft.com

Venkatesh Ganti  
Microsoft Research  
One Microsoft Way  
Redmond, WA 98052  
vganti@microsoft.com

Raghav Kaushik  
Microsoft Research  
One Microsoft Way  
Redmond, WA 98052  
skaushi@microsoft.com

ABSTRACT

Given two input collections of sets, a *set-similarity join* (SSJoin) identifies all pairs of sets, one from each collection, that have high similarity. Recent work has identified SSJoin as a useful primitive operator in data cleaning. In this paper, we propose new algorithms for SSJoin. Our algorithms have two important features: They are exact, i.e., they always produce the correct answer, and they carry...
Basic correlated LSH partitioning
Basic correlated LSH partitioning

Arasu et al. VLDB 2006
Basic correlated LSH partitioning

$\text{VLDB 2006}$

$h_2$
Basic correlated LSH

partitioning

$h_3$
Basic correlated LSH partitioning

$h_4$
Basic correlated LSH partitioning

For Hamming distance $\leq 3$, a collision is guaranteed!
Basic correlated LSH partitioning

For Hamming distance \( \leq 3 \), a collision is guaranteed!

To bound probability of collision for distance \( > 3 \), randomly permute the dimensions
Basic correlated LSH

equation

\[ h_{12} \]
Basic correlated LSH

\[ h_{13} \]
Basic correlated LSH

Arasu et al.
VLDB 2006

$h_{14}$
Basic correlated LSH

For Hamming distance \( \leq 2 \), a collision is guaranteed in \( h_{12}, h_{13}, h_{14}, h_{23}, h_{24}, h_{34} \).
Result in LSH framework

- The Gionis et al. LSH achieves:

\( \Pr[h(q) = h(x) \mid d(q, x) = cr] \leq \Pr[h(q) = h(x) \mid d(q, x) = r]^c \)
Result in LSH framework

• The Gionis et al. LSH achieves:

\[ \Pr[h(q) = h(x) \mid d(q, x) = cr] \leq \Pr[h(q) = h(x) \mid d(q, x) = r]^c \]

• Arasu et al. construction combining partitioning, enumeration, and random permutation:

\[ \Pr[h(q) = h(x) \mid d(q, x) = cr] \leq \Pr[h(q) = h(x) \mid d(q, x) = r]^{0.36c} \]
Conclusions

• Recursive application of LSH is a promising approach to large-scale similarity joins with theoretical guarantees.
Conclusions

• Recursive application of LSH is a promising approach to large-scale similarity joins with theoretical guarantees.

• Guarantees can include “total recall”, though with present constructions this requires a constant $c > 3$. 
Some open questions

• Practicality of recursive approach
  - Many choices made to enable theoretical reasoning should be reconsidered.
  - How about parallelism?
Some open questions

• Practicality of recursive approach
  - Many choices made to enable theoretical reasoning should be reconsidered.
  - How about parallelism?

• Better correlated LSH
  - Ideally matching the best performance possible without 100% recall guarantee.
Thank you!

IT UNIVERSITY OF COPENHAGEN

DET FRIE FORSKNINGSråD

European Research Council
Supporting top researchers from anywhere in the world
“There's nothing an agnostic can't do if he doesn't know whether he believes in anything or not.”

–Monty Python