Efficient Parallel Partition based Algorithms for Similarity Search and Join with Edit Distance Constraints

Yu Jiang, Dong Deng, Jiannan Wang, Guoliang Li, and Jianhua Feng

Tsinghua University

Similarity Search&Join Competition on EDBT/ICDT 2013
Outline

1. Motivation
   - Problem Definition
   - Application

2. Our Approach
   - Pass Join Algorithm
   - Additional Filters
   - Parallel

3. Experiment
   - Evaluating Pruning Techniques
   - Evaluating Parallelism
   - Evaluating Scalability
Given a set of strings $S$, the task is to find all pairs of $\tau$-similar strings from $S$. A program must output all matches with both string identifiers and distance $\tau$. (Track II)
Consider the string dataset in Table 1. Suppose $\tau = 3$. $\langle s_4, s_6 \rangle$ is a similar pair as $ED(s_4, s_6) \leq \tau$.
Application

- Data cleaning
- Information Extraction
- Comparison of biological sequences
- ...

Motivation

Our Approach

Experiment

Problem Definition

Application

Dong Deng

Parallel PassJoin
Basic Idea

Lemma

Given a string $r$ with $\tau + 1$ segments and a string $s$, if $s$ is similar to $r$ within threshold $\tau$, $s$ must contain a segment of $r$.

Example

$\tau = 1$, $r =$ “EDBT” has two segments “ED” and “BT”. $s =$ “ICDT” cannot similar to $r$ as $s$ contains none of the two segments.
Even Partition Scheme

**Definition**

In even partition scheme, each segment has almost the same length. \(\left\lfloor \frac{|s|}{\tau+1} \right\rfloor\) or \(\left\lceil \frac{|s|}{\tau+1} \right\rceil\).

**Example**

\(\tau = 3\), we partition \(s_1 = \text{“vankatesh”}\) into four segments “va”, “nk”, “at”, “esh”.
Substring Selection
Basic Methods

- Enumeration:
  Enumerate all substrings for each of the segment.

- Length-based:
  For each segment, only select substrings with same length.

- Shift-based:
  For segment with start position $p_i$, select substrings with start position in $[p_i - \tau, p_i + \tau]$
Substring Selection
Position-aware Substring Selection

Observation

Theorem (Position-aware Substring Selection)

For segment with start position $p_i$, select substrings with start position in $[p_i - \lfloor \frac{\tau - \Delta}{2} \rfloor, p_i + \lceil \frac{\tau + \Delta}{2} \rceil]$ where $\Delta = |s| - |r|$. 

Dong Deng
Parallel PassJoin
Substring Selection
Position-aware Substring Selection

Observation

![Diagram showing substring selection process]

Theorem (Position-aware Substring Selection)

For segment with start position $p_i$, select substrings with start position in $[p_i - \left\lfloor \frac{\tau - \Delta}{2} \right\rfloor, p_i + \left\lceil \frac{\tau + \Delta}{2} \right\rceil]$ where $\Delta = |s| - |r|$.
## Substring Selection
### Position-aware Substring Selection

### Example

$$r = \text{“vankatesh”} \quad s = \text{“avataresha”}$$

- $p_1=1$, $va \quad \rightarrow \quad [1,3]: \text{av va at}$
- $p_2=3$, $nk \quad \rightarrow \quad [2,5]: \text{va at ta ar}$
- $p_3=5$, $at \quad \rightarrow \quad [4,7]: \text{ta ar re es}$
- $p_4=7$, $esh \quad \rightarrow \quad [6,8]: \text{res esh sha}$

$$\tau = 3, \ \Delta = 1, \ \lfloor p_i - \frac{\tau - \Delta}{2} \rfloor, \ p_i + \lfloor \frac{\tau + \Delta}{2} \rfloor = [p_i - 1, p_i + 2]$$
Observation

\[ r_l = "" \quad r_r \]
\[ r = \text{“vankatesh”} \rightarrow \{\text{va, nk, at, esh}\} \]
\[ s = \text{“avataresha”} \quad |s_l|-|r_l| = 1 \]

\( r_r \) has 3 segments to detect, 2 errors allowed

There must be another matching between \( r_r \) and \( s_r \).

Theorem (Multi-match-aware Substring Selection)

For the \( i \)-th segment with start position \( p_i \), select substrings within \([p_i - i, p_i + i] \cap [p_i + \Delta - (\tau + 1 - i), p_i + \Delta + (\tau + 1 - i)]\).
**Observation**

Let \( r_l = "" \) and \( r_r \).

\[
\begin{align*}
rs &= "vankatesh" \quad \rightarrow \quad \{va, nk, at, esh\} \\
ss &= "avataresha" \quad \Rightarrow \quad |s_l - |r_l|| = 1
\end{align*}
\]

\( r_r \) has 3 segments to detect, 2 errors allowed.

There must be another matching between \( r_r \) and \( s_r \).

**Theorem (Multi-match-aware Substring Selection)**

For the \( i \)-th segment with start position \( p_i \), select substrings within \([p_i - i, p_i + i]\) \( \cap \) \([p_i + \Delta - (\tau + 1 - i), p_i + \Delta + (\tau + 1 - i)]\).
**Example**

\[ r = \text{“vankatesh”} \quad s = \text{“avataresha”} \]

- \( p_1 = 1, \text{ va } \) \( \rightarrow \) \( 1,1: \text{ av} \)
- \( p_2 = 3, \text{ nk } \) \( \rightarrow \) \( 2,4: \text{ va at ta} \)
- \( p_3 = 5, \text{ at } \) \( \rightarrow \) \( 5,7: \text{ ar re es} \)
- \( p_4 = 7, \text{ esh } \) \( \rightarrow \) \( 8,8: \text{ sha} \)
Theoretical Results

1. The number of selected substrings by the multi-match-aware method is minimum.

2. For strings longer than $2 \times (\tau + 1)$, our selection method is the only way to select minimum number of substrings.
## Motivation

## Our Approach

## Experiment

### Pass Join Algorithm

### Additional Filters

### Parallel Filters

### Substring Selection

## Experimental Results

**Figure:** Numbers of selected substrings

(a) Author Name (Avg Len = 15)

(b) Query Log (Avg Len = 45)

(c) Author+Title (Avg Len = 105)
Substring Selection
Experimental Results

(a) Author Name  
(Avg Len = 15)
(b) Query Log  
(Avg Len = 45)
(c) Author+Title  
(Avg Len = 105)

Figure: Elapsed time for generating substrings
Inspired by the position-aware substring selection.
- Save at least half computation than traditional dynamic method.
- Save even more using improved early termination.
Inspired by the position-aware substring selection.

Save at least half computation than traditional dynamic method.

Save even more using improved early termination.
Inspired by the position-aware substring selection.
- Save at least half computation than traditional dynamic method.
- Save even more using improved early termination.
Inspired by the multi-match-aware substring selection.
- Using tighter thresholds to verify the candidate pairs.
- Verify if $ED(r_r, s_r) \leq \tau + 1 - i$ and $ED(r_l, s_l) \leq i - 1$. 

Dong Deng
Parallel PassJoin
Verification
Extension-based Verification

- Inspired by the multi-match-aware substring selection.
- Using tighter thresholds to verify the candidate pairs.
- Verify if $ED(r_r, s_r) \leq \tau + 1 - i$ and $ED(r_l, s_l) \leq i - 1$. 
Inspired by the multi-match-aware substring selection.
Using tighter thresholds to verify the candidate pairs.
Verify if $ED(r_r, s_r) \leq \tau + 1 - i$ and $ED(r_l, s_l) \leq i - 1$. 
Verification
Experimental Results

Figure: Elapsed time for verification

(a) Author Name
(Avg Len 15)

(b) Query Log
(Avg Len 45)

(c) Author+Title
(Avg Len 105)
Partition longer strings into segments.
Select substrings from shorter strings.
Longer segments decrease the possibility of matching.
Thus decrease the number of candidates.
**Effective Indexing Strategy**

- Partition longer strings into segments.
- **Select substrings from shorter strings.**
- Longer segments decrease the possibility of matching.
- Thus decrease the number of candidates.
Additional Filters
Effective Indexing Strategy

- Partition longer strings into segments.
- Select substrings from shorter strings.
- Longer segments decrease the possibility of matching.
- Thus decrease the number of candidates.
Motivation
Our Approach
Experiment
Pass Join Algorithm
Additional Filters
Parallel

Additional Filters
Effective Indexing Strategy

- Partition longer strings into segments.
- Select substrings from shorter strings.
- Longer segments decrease the possibility of matching.
- Thus decrease the number of candidates.
Observation

- Let $\mathcal{H}_r$ denote the character frequency vector of $r$.
- $r = “abyyyy”, \ s = “axxyyxy$.
  $\mathcal{H}_r = \{\{a, 1\}, \{b, 1\}, \{y, 4\}\}$, $\mathcal{H}_s = \{\{a, 1\}, \{x, 3\}, \{y, 4\}\}$
- Let $\mathcal{H}_\triangle = |\mathcal{H}_r - \mathcal{H}_s|$.
- $\mathcal{H}_\triangle = |\mathcal{H}_r - \mathcal{H}_s| = |1| + | - 3| = 4$.
- A deletion or insertion changes $\mathcal{H}_\triangle$ by 1 at most.
- An substitution changes $\mathcal{H}_\triangle$ by 2 at most.
Observation

- Let $\mathcal{H}_r$ denote the character frequency vector of $r$.
- $r = \text{"abyyyy"}$, $s = \text{"axxyyyxy"}$.
  $\mathcal{H}_r = \{\{a, 1\}, \{b, 1\}, \{y, 4\}\}$, $\mathcal{H}_s = \{\{a, 1\}, \{x, 3\}, \{y, 4\}\}$
- Let $\mathcal{H}_\triangle = |\mathcal{H}_r - \mathcal{H}_s|$.
- $\mathcal{H}_\triangle = |\mathcal{H}_r - \mathcal{H}_s| = |1| + | - 3| = 4$.
- A deletion or insertion changes $\mathcal{H}_\triangle$ by 1 at most.
- An substitution changes $\mathcal{H}_\triangle$ by 2 at most.
Observation

- Let $H_r$ denote the character frequency vector of $r$.
- $r = "abyyyy", s = "axxyyyxy"$.
  - $H_r = \{\{a, 1\}, \{b, 1\}, \{y, 4\}\}$, $H_s = \{\{a, 1\}, \{x, 3\}, \{y, 4\}\}$
- Let $H_\triangle = |H_r - H_s|$.
- $H_\triangle = |H_r - H_s| = ||1| + | - 3|| = 4$.
- A deletion or insertion changes $H_\triangle$ by 1 at most.
- An substitution changes $H_\triangle$ by 2 at most.
Let $H_r$ denote the character frequency vector of $r$.

\[ r = \text{"abyyyy"}, \quad s = \text{"axxyyyxy"}. \]

\[ H_r = \{\{a, 1\}, \{b, 1\}, \{y, 4\}\}, \quad H_s = \{\{a, 1\}, \{x, 3\}, \{y, 4\}\} \]

Let $H_\triangle = |H_r - H_s|$.

\[ H_\triangle = |H_r - H_s| = |1| + | - 3| = 4. \]

A deletion or insertion changes $H_\triangle$ by 1 at most.

An substitution changes $H_\triangle$ by 2 at most.
Observation

- At most $\tau$ edit operations, $H_\triangle \leq 2\tau$.
- At most $\tau - |r| - |s|$ substitutions, $H_\triangle \leq 2\tau - |r| - |s|$.
- Group symbols to improve the content-filter running time.
- Integrate the content filter with the extension-based verification.
Observation

- At most $\tau$ edit operations, $\mathcal{H}_\Delta \leq 2\tau$.
- At most $\tau - |r| - |s|$ substitutions, $\mathcal{H}_\Delta \leq 2\tau - |r| - |s|$.
- Group symbols to improve the content-filter running time.
- Integrate the content filter with the extension-based verification.
Observation

- At most $\tau$ edit operations, $\mathcal{H}_\triangle \leq 2\tau$.
- At most $\tau - |r| - |s|$ substitutions, $\mathcal{H}_\triangle \leq 2\tau - |r| - |s|$.
- Group symbols to improve the content-filter running time.
- Integrate the content filter with the extension-based verification.
**Observation**

- At most $\tau$ edit operations, $H_\triangle \leq 2\tau$.
- At most $\tau - |r| - |s|$ substitutions, $H_\triangle \leq 2\tau - |r| - |s|$.
- Group symbols to improve the content-filter running time.
- *Integrate the content filter with the extension-based verification.*
2. Parallel Building Indexes. Parallel building indexes for each group.
3. Parallel Joins. Parallel perform similarity joins on each group.
Experiment Setup

Table: Datasets

<table>
<thead>
<tr>
<th>Datasets</th>
<th>cardinality</th>
<th>average len</th>
<th>max len</th>
<th>min len</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeoNames</td>
<td>400,000</td>
<td>11.106</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>GeoNames Query</td>
<td>100,000</td>
<td>10.7</td>
<td>2</td>
<td>43</td>
</tr>
<tr>
<td>Reads</td>
<td>750,000</td>
<td>101.388</td>
<td>86</td>
<td>106</td>
</tr>
<tr>
<td>Reads Query</td>
<td>100,000</td>
<td>101.2</td>
<td>88</td>
<td>116</td>
</tr>
</tbody>
</table>
Experiment Setup

(a) GeoNames
(b) Reads

Figure: Length Distribution.
Evaluating Pruning Techniques

**Figure:** Evaluating pruning techniques for similarity joins (8 threads).

(a) GeoNames

(b) Reads
Evaluating Pruning Techniques

Figure: Evaluating pruning techniques for similarity search (8 threads).
Evaluating Parallelism

Figure: Evaluating running time of similarity join by varying number of threads.
Evaluating Speedup

(Evaluating Pruning Techniques)
(Evaluating Parallelism)
(Evaluating Scalability)

Motivation
Our Approach
Experiment

---

Figure: Evaluating speedup of similarity join.

(a) GeoNames

(b) Reads
Evaluating Parallelism

(a) GeoNames

(b) Reads

Figure: Evaluating running time of similarity search by varying number of threads.
Evaluating Speedup

Figure: Evaluating speedup of similarity search.
Evaluating Scalability

Figure: Evaluating the scalability of the similarity join algorithm (8 threads).
Evaluating Scalability

Figure: Evaluating the scalability of the similarity search algorithm (8 threads).
About our team

- We are from Tsinghua University, Beijing, China.
- Yu Jiang, Jiannan Wang, Guoliang Li, Jianhua Feng and Dong Deng.
About our team II

Dong Deng

Parallel PassJoin
Thank You
Q & A

http://dbgroup.cs.tsinghua.edu.cn/dd