List of Twin Clusters: a Data Structure for Similarity Joins in Metric Spaces

Rodrigo Paredes

Nora Reyes

Supported in part by Millennium Nucleus Center for Web Research, Grant P04-067-F, Mideplan, Chile; and AECI A/8065/07.
Introduction

We focus on a particular case of the similarity join primitive:

Given two sets $A = \{a_1, a_2, \ldots, a_{|A|}\} \subseteq X$ and $B = \{b_1, b_2, \ldots, b_{|B|}\} \subseteq X$

$$A \bowtie_r B = \{(a_i, b_j), a_i \in A, b_j \in B, d(a_i, b_j) \leq r \}$$

Find all the object pairs at distance at most $r$ (if $A=B$, is called similarity self join).
Introduction

- Some applications: data mining, data cleaning, and data integration.
- This version of similarity join translates into solving several range queries.
Similarity Joins

Range queries with threshold \( r \) for all element in \( A \)
List of Clusters (LC)

- The LC splits the space into zones.

- Each zone has a center $c$ and stores both its radius $r_p$ and the bucket $I$ of internal objects.

- The center ball $(c, r_p) = \{ x \in X, \ d(c, x) \leq r_p \}.$
List of Clusters (LC)

\( (c_1, r_{p1}, l_1) \) \( E_1 \) \( (c_2, r_{p2}, l_2) \) \( E_2 \) \( (c_3, r_{p3}, l_3) \)
We search exhaustively in \( I \), and continue searching in \( E \).

We do not search in \( I \), but we continue searching in \( E \).

We search exhaustively in \( I \), but we do not continue searching in \( E \).
Similarity Joins

- Given $A, B \subseteq X$, the naive approach to compute the similarity join uses $|A| \cdot |B|$ distance computation.

- This is usually called the *Nested Loop*.

- A natural approach consists in indexing one or both sets independently, and then solving range queries for the involved elements.
We propose to index both sets jointly: solving the similarity join by indexing the datasets $A$ and $B$ in a single data structure.

- We do not perform distance computations between objects of the same set.

- The LTC is based on the *List of Clusters*. 
List of Twin Clusters (LTC)

- We have chosen to use clusters with fixed radius.

- LTC considers a list of overlapping clusters, which we call *twin clusters*.

- Most of relevant objects would belong to the twin cluster of the object we are considering.
List of Twin Clusters (LTC)

The twin clusters with centers $c_A$ and $c_B$
List of Twin Clusters (LTC)

- We solve range queries for objects from one set retrieving relevant objects from the other.

- We suppose:
  - We are computing range queries for elements in $A$,
  - $|A| \geq |B|$.
Range Queries with LTC

- We have to process three kinds of objects: cluster centers, regular objects (inside clusters), and non-indexed objects (the rest).

- We use the *triangle inequality* and all the *distances in the index* (the list of twin clusters, the distances among centers, and the distances to closest and furthest centers) to avoid distance computations.
Computing the LTC-join

- Given the datasets $A$ and $B$, and a radius $R$, we compute the LTC index.

- Then, with the join threshold $r$ we actually compute range queries:
  - cluster centers: previous clusters of the list.
  - regular objects: the list and distances among centers.
  - non-indexed objects: distances among centers and distances to closest and furthest centers.
Experimental Evaluation

- We compare our proposal against:
  - The *Nested loop*.
  - The simple join algorithm having a LC built for one database: LC-join.
  - Indexing both databases with LC with a join algorithm that uses all the information from the indices to improve the join cost: LC2-join.
Experimental Evaluation

- Three different pairs of real world databases from two metric spaces:
  - Face images: 1,016 761-dimensional feature vectors from a face image database.
  - Strings: a dictionary of words.
    - A subset of English words with a subset of Spanish words.
    - The same English subset with a subset of the vocabulary of terms from a document collection.
Experimental Evaluation

- We need to fix the radius before building the LC and LTC.

- We choose the radius $R$ which obtains better join cost for each alternative.

- $R$ should be greater than or equal to the largest $r$ used in the similarity join: $A \bowtie_r B$
Experimental Evaluation
Experimental Evaluation

Join Costs between Spanish and English dictionaries

- LTC, radius 3
- LTC, radius 4
- LTC, radius 5
- LTC, radius 6

Distance evaluations x 1,000,000

Threshold used for join
Experimental Evaluation

Join Costs between Vocabulary and English dictionary

- LTC join, radius 3
- LTC join, radius 4
- LTC join, radius 5
- LTC join, radius 6

Distance evaluations x 1,000,000

Threshold used for join
Experimental Evaluation

- The better results are obtained with the building radius $R$ closest to the greatest value of $r$ considered.

- The construction costs of the LTC and the LC over one of the databases are similar.
Experimental Evaluation

Join Costs bet. FACES762/254 feature vectors

Distance evaluations

Threshold used for join

LC-pin, radius 0.38
LC2-pin, radius 0.38
LTC-pin, radius 0.38
Experimental Evaluation

Join Costs between Spanish and English dictionaries

- LC-join, radius 4
- LC2-join, radius 3
- LTC-join, radius 3

Distance evaluations x 1,000,000

Threshold used for join
Experimental Evaluation

Join Costs between English dictionary and Vocabulary

Distance evaluations x 1,000,000

LC-join, radius 4 (green)
LC2-join, radius 3 (blue)
LTC-join, radius 3 (red)

Threshold used for join

1  2  3
Experimental Evaluation

- The LTC-join outperforms largely the LC-join and LC2-join in two of the database pairs.

- For the other pair LTC largely improves over nested loop, but LC and LC2 beat us.

- This non-intuitive behavior can be explained by taking into account the amount of non-indexed elements.
Conclusions

- We show a new approach, LTC-join, for similarity join by indexing both datasets jointly.

- Our results show speedups over LC-join and LC2-join.

- LTC index stands out as a practical and efficient data structure to solve a particular case of similarity join.
Work in Progress

- The similarity self join.
- Improve the LTC by exploiting internal distances.
- The center selection.
- A version of the LTC similar to the recursive list of clusters.
- Researching on alternatives to manage the non-indexed objects.