Overview

(I) Similarity Learning

- Algorithmic Work
- Given Pair of Objects
- Return Similarity Values

- Specific Data Type
- Effectiveness in Applications
- Privacy Issue

Outline

- Graph Data Management and Analysis
  - 1) Inter-Graph Node Metric Learning
  - 2) Temporal Graph Engine - Temporal Subgraph Retrieval

- Time-Series Data Management and Analysis
  - 3) Privacy Preserving Time-Series Similarity Learning
  - 4) Generic Time-Series Engine - Subsequence Retrieval

Node Similarity
Node Similarity Applications

- Transfer Learning
  - Communication Networks

- Graph De-anonymization
  - Social Networks

Existing Node Similarity Measures

<table>
<thead>
<tr>
<th>Similarity</th>
<th>Inter-G</th>
<th>Extra-Info</th>
<th>Tri-Jacc</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringRank [4]</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>P-Rank [14]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PathRank [20]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>StrMat + [4]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Matchify [75]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>StrMat* [10]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Roughly [46]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>HTS-based Similarity [12, 15, 99]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Personalized PageRank [48]</td>
<td>No</td>
<td>No</td>
<td>NO</td>
</tr>
<tr>
<td>Similarity Flooding [9]</td>
<td>Yes</td>
<td>Edge Label</td>
<td>NO</td>
</tr>
<tr>
<td>Sampling-based Similarity [13, 82]</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Neighbor Set Similarity [79, 109, 120, 148]</td>
<td>Yes</td>
<td>Node Identifier</td>
<td>Yes</td>
</tr>
<tr>
<td>Neighbor Vector Similarity [80, 49, 50]</td>
<td>Yes</td>
<td>Node Label</td>
<td>NO</td>
</tr>
<tr>
<td>Oddlink [2]</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>NetSimi [8]</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Geometric Feature-based Similarity [4]</td>
<td>Yes</td>
<td>Edge Length</td>
<td>Yes</td>
</tr>
<tr>
<td>Role - based Similarity [61, 62, 104, 19]</td>
<td>Yes</td>
<td>Node Role</td>
<td>Yes</td>
</tr>
<tr>
<td>Graphlets-based Similarity [20]</td>
<td>Yes</td>
<td>Graphlets</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Our Node Metric: NED

- Use the neighborhood topological structure to describe a node.

- Use Neighborhood Trees rather than Sub-Graphs
  - Computability - Let $d(u, v)$ be a distance function between a pair of nodes $u$ and $v$. The neighborhood subgraphs of $u$ and $v$ are isomorphic, iff $d(u, v) = 0$. Then $d$ is not polynomial-computable. (As hard as graph isomorphism)
  - Effectiveness - Non-identity is not good for top-k similar nodes ranking and k-nearest-neighbor similar nodes finding.
Tree Edit Distance

- The similarity between two nodes is measured by the edit distance between neighborhood adjacent-trees
  \[ \delta(u, v) = \delta_{TED}(T(u, k), T(v, k)) \]
- The computation of tree edit distance on unordered unlabeled trees belongs to NP-Complete, even MaxSNP-Hard.

Modified Tree Edit Distance: TED*

- Modified Tree Edit Distance:
  - No operation can change the depth of any existing node in the tree
- Edit Operation in TED*:
  - Insert a leaf node
  - Delete a leaf node
  - Move a node at the same level
- Advantage of TED*?
  - Metricicty: Also Metric as TED
  - Efficiency: Polynomial-Time Computable

Computation of TED*

- Iterative Bottom-Up Algorithm

Padding Cost @ Lv 3 = 1

Computation of TED*

Canonization

Bipartite Graph Construction
Computation of TED*
Computation of TED*

Bipartite Graph Matching

Matching Cost @ Lv 2 = 0

Re-Canonization

Computation of TED*

Padding

Padding Cost @ Lv 1 = 0

Canonization

Computation of TED*

Bipartite Graph Construction

Computation of TED*

Bipartite Graph Matching
Computation of TED*

Bipartite Graph Matching

- Matching Cost @ Lv 1 = 1

Computation of TED*

Re-Canonization

- δTED*(T(u,k), T(v,k)) = \sum_{i=0}^{k}(P_i + M_i)
  - δTED*(T(A,3), T(α,3)) = 2

Computation of TED*

Padding

- Padding Cost @ Lv 0 = 0

Computation of TED*

Canonization

- Efficiency
  - GED ≤ 2TED*

 TED* vs TED & GED

- Efficiency
**TED* vs TED & GED**

- Small Difference
- Equivalency

**Metricity**

- TED* is a metric for trees
  - $\delta_{TED}(T(u, k), T(v, k)) = \sum_{i=1}^{n} p_i + \sum_{j=m}^{n} |G_i^2|/2$
  - $\delta_{TED}(T(u, k), T(v, k)) \geq 0$
  - $\delta_{TED}(T(u, k), T(v, k)) = \delta_{TED}(T(v, k), T(u, k))$
  - $\delta_{TED}(T(u, k), T(v, k)) = 0$ if $T(u, k) = T(v, k)$
- Node Edit Distance (NED) can be a metric for inter-graph nodes

**NED for Nearest Neighbor Query**

- Ties Issue
- Collaborate with Metric Index

**Graph De-Anonymization**

- Higher Precision
- More Robust to Permutation

**Outline**

- **Graph Data Management and Analysis**
  - 1) Inter-Graph Node Metric Learning
  - 2) Temporal Graph Engine - Temporal Subgraph Retrieval

- **Time-Series Data Management and Analysis**
  - 3) Privacy Preserving Time-Series Similarity Learning
  - 4) Generic Time-Series Engine - Subsequence Retrieval

**Temporal Graph Visualization (I)**

Air Traffic Map from NATS
Temporal Graph Visualization (II)

Dynamic Graph from Graph.

Temporal Graph Instances

- **Temporal Graphs** contain more valuable information than static graphs in real-world settings.
- Temporal Graphs cover versatile applications:
  - Communication Networks
  - Social Networks
  - Growing Social Networks

Temporal Graph Definition

- A **Temporal Graph** is a sequence of graph snapshots
  
  \[ G(t) = < G_1, G_2, G_3, ..., G_n > \]

- \( G_i \) is a snapshot \( G_i = (V_i, E_i) \) on timestamp \( i \)

- A **Temporal Subgraph** is used for time-sensitive analytics on temporal graphs.
  
  - What is time-sensitive analytics ?
  - What are temporal subgraphs ?
  - How to efficiently query temporal subgraphs ?
Temporal Edges and T-Subgraph

- **Temporal Edge List** is the most space-efficient format to store temporal graphs.
- **Covering Subgraph**: the whole temporal edges covered by a time period: \(<\text{plan a schedule}>\)
- **Persisting Subgraph**: the temporal edges covering a whole time period: \(<\text{find stable status}>\)

Edge Sequence Query

- **Edge Sequence Query Statement**: 
  For a given set of tokens \(Q_s, Q_e, t_x\) and \(t_y\), retrieve all temporal edges \(e^t(u, v, t^s, t^f) \in E(t)\) (A Temporal Graph) that satisfy the temporal predicates as \(t^s Q_s t_x\) and \(t^e Q_e t_y\), where \(Q_s\) and \(Q_e\) are relation symbols such as: \(<, \leq, >, \geq\), \(t_x\) and \(t_y\) are the temporal predicates.

Temporal Hashing

- **Non-redundant Temporal Hashing**: 
  \(e^t(u, v, t_i, t_j) \in b_j \iff i = \mathcal{H}(t^i) = [t^i/B]\)
- **Redundant Temporal Hashing**: 
  \(e^t(u, v, t_i, t_j) \in b_j \iff i \in [\mathcal{H}(t^i), \mathcal{H}(t^j)]\)

Redundancy Analysis

- The probability density function for edge duration is 
  \(f_x(x) = \begin{cases} \alpha x^{-(a+1)} & 0 < x < \mu \\ 0 & x \geq \mu \end{cases}\)
- The space for non-redundant temporal hashing 
  \(F_x(n) = F_x(mB) = 1 - \frac{1}{mB}\)
- The space for redundant temporal hashing 
  \(F'_x(n) = F'_x(mB) = \sum_{k=1}^{m} \int_{\mu}^{\mu + 1} (k + 1)x^{-(a+1)} dx = B^{-a} \sum_{k=1}^{m} (k + 1)(k - 1 + \frac{1}{B})^{-a} - k^{-a}\)

Distribution

- **Edge Duration Distribution**

LiTe-Graph Architecture

- **Temporal Path Finding**
- **Temporal Subgraph Query**
- **Redundant Temporal Hashing Index**
- **In-mem-caching**
- **Persistent storage**
- **Partial Order Edge List**
- **Data Importer**
- **Index Builder**
3D Distribution

3D Distribution for Edge Duration and Starting Time

Storage Space / Index Space

- Storage space: $O(n)$ 16 GB
- Index space: $O(n/B) \rightarrow O(2n/B)$ 20 MB

Query Performance

Parameter Effects

Time-Sensitive Analytics

Temporal Path Finding

- From the single-scan algorithm to the multi-passes algorithm

- Lock-free parallel computing solves the multi-passes issue
In-Memory Database on Single PC

- For in-memory LiTe-Graph engine on single PCs, 32 GB Memory ($300) is enough to manage a temporal graph with 1 billion edges.

Outline

- Graph Data Management and Analysis
  - 1) Inter-Graph Node Metric Learning
  - 2) Temporal Graph Engine - Temporal Subgraph Retrieval

- Time-Series Data Management and Analysis
  - 3) Privacy Preserving Time-Series Similarity Learning
  - 4) Generic Time-Series Engine - Subsequence Retrieval

Time-Series Data

- Sensitive Time-series Data
  - Physiological Medical Data
  - Personnel Trajectory Data
  - Signature Data
  - Financial Data

Privacy Preserving Similarity Evaluation

- Two-Party Computation

Distance Evaluation of Time Series Data

- Distance Functions
  - Euclidean Distance
  - LCSS Distance
  - ERP Distance
  - Fréchet Distance
  - Dynamic Time Warping (DTW)

- Dynamic Time Warping
  \[ d(i,j) = d(i,j) + \min[D(i-1,j),D(i,i-1),D(i-1,j-1)] \]
  Where \( d(i,j) = \| x_i - y_j \| \)

What we need to hide?

- The Original Time Series
- The Matrix
  - Entries
  - Optimal Path
- Potential Risk during Computation
  - Filling the Matrix
### How to Fill One Entry of the Matrix

- **Dynamic Time Warping**
  
  \[ D(i,j) = d(i,j) + \min(D(i-1,j), D(i,j-1), D(i-1,j-1)) \]

  Where \( d(i,j) = \|x_i - y_j\| \)

  - Secure Euclidean Distance Computation (Part 1)
  - Privacy Preserving Comparison of Ciphertexts (Part 2)

- **Secure Euclidean Distance**
  - Paillier Encryption - One Way communication protocol

- **Privacy Preserving Comparison**
  - Random Offsets - One Round communication protocol

### Secure Euclidean Distance Computation

- **PaillierEncryption (Partial Homomorphic Encryption)**
  - \( Enc(m_1 + m_2 \mod n) = Enc(m_1) \cdot Enc(m_2) \mod n^2 \)
  - \( Enc(m_1 \cdot k \mod n) = Enc(m_1)^k \mod n^2 \)

- **Square of Euclidean Distance Computation**
  - \( Enc((D^2(i,j)) = Enc(\sum_{i=1}^{n} p_i^2 + \sum_{i=1}^{n} q_i^2 - 2 \sum_{i=1}^{n} p_i q_i) \)
  - \( Enc(\sum_{i=1}^{n} p_i^2) \) can be computed by the owner of \( p_i \) (The Server)
  - \( Enc(\sum_{i=1}^{n} q_i^2) \) can be computed by the owner of \( q_i \) (The Client)
  - \( Enc(-2 \sum_{i=1}^{n} p_i q_i) = \prod_i^\infty Enc(p_i)^{-2 q_i} \) can be evaluated by the owner of \( q_i \) (The Client) with information of \( Enc(p_i) \)
  - the owner of \( q_i \) (The Client) is the owner of the matrix who cannot decrypt ciphertexts. The Client is also the evaluator of part 1 of the protocol
Privacy Preserving Comparison

- The Client Generates Candidates:
  - The client generates $k$ random values: $r_1, r_2, r_3, ... , r_k$ ($r_1$ is the minimum).
  - The client creates $k+2$ candidates: $\text{Enc}(A+r_1), \text{Enc}(B+r_1), \text{Enc}(C+r_1), \text{Enc}(X+r_1), ... , \text{Enc}(X+r_k)$, ($X$ is one of $A$, $B$, and $C$) by using homomorphic addition of Paillier encryption.
  - The client permutes and sends the $k+2$ candidates to the server.

- The Server Decrypts and Compares Candidates
  - The server decrypts $k+2$ candidates with the secret key and finds the minimal plaintext $M$ among the $k+2$ plaintexts.
  - The server re-encrypts the minimal plaintext $M$ and returns $\text{Enc}(M)$ to the client.

- The Client Computes the Result
  - The client computes $\text{Enc}(\min(A, B, C)) = \text{Enc}(M - r_1)$ by using homomorphic addition of Paillier encryption as the result.

Performance Analysis

- Communication Cost: $mn(d + k + 4)$
- Computation Cost
  - Server: $mn(d + 2)$ encryptions and $mn(k + 2)$ decryptions
  - Client: $mn(k + 1)$ encryptions

- $m$: length of time series data of server
- $n$: length of time series data of client
- $d$: dimensionality of single data point
- $k$: size of random set

For low dimensions, the server has more computational cost than the client, because encryptions and decryptions cost more than other operations and decryptions cost more than encryptions.

Security Analysis

- The client learns nothing during the computation since the client cannot decrypt any ciphertext.
- The server learns nothing during the first phase of the protocol, however.... The server decrypts $k+2$ candidates.

- The server may create a permuted linear equation system. To solve this system is NP-hard.
- The Client needs to select the range for random values close to the actual values and needs to make sure that there is no wrap around.
- We show that our protocol can preserve at least half of the information entropy of the plaintexts.

Experiments

- Performance vs Sequence Size (Different Phases)
- Performance vs Dimensionality (Different Phases)
Experiments

- Performance for Discrete Fréchet Distance (DFD)

![Graph showing performance](image1)

Outline

- **Graph Data Management and Analysis**
  1) Inter-Graph Node Metric Learning
  2) Temporal Graph Engine - Temporal Subgraph Retrieval

- **Time-Series Data Management and Analysis**
  3) Privacy Preserving Time-Series Similarity Learning
  4) Generic Time-Series Engine - Subsequence Retrieval

Outline

- **Graph Data Management and Analysis**
  1) Inter-Graph Node Metric Learning
  2) Temporal Graph Engine - Temporal Subgraph Retrieval

- **Time-Series Data Management and Analysis**
  3) Privacy Preserving Time-Series Similarity Learning
  4) Generic Time-Series Engine - Subsequence Retrieval

Content-Based Searching

![Image of content-based searching](image2)

Time-Series Similarity

- **Distance Functions for time-series data (sequence)**
  - Euclidean Distance
  - Hamming Distance
  - Dynamic Time Warping
  - Levenshtein Distance
  - Fréchet distance
  - ERP
  - ...

- **Discrete Fréchet distance**
  \[ F(A, B) = \inf_{\alpha, \beta} \max \{ d(\alpha(t), B(t)) \} \]
  \[ d(\alpha(t), B(t)) \] is a distance function for metric space of objects

Subsequence Query

- **Subsequence Query Statement:**
  Let \( S \) be the time-series in \( db \) and \( Q \) be the query series.
  For a distance function \( d \) and a threshold \( \epsilon \), retrieve all subsequences in \( S \) that have the distances to any subsequence in \( Q \) smaller than \( \epsilon \). Meanwhile, the subsequences cannot be shorter than \( \lambda \)

  - Is that possible to have a time-series data management engine working for different distance functions \( d \)?

Another Challenge

- **Number of Candidates**
  - There are totally \( O(m^2n^2) \) pairs of subsequences could be results.
  - \( m^2 \) candidates from database
    - \( m \) is the total length of sequences in the database
  - \( n^2 \) candidates from query
    - \( n \) is the length of sequence for the query object
    - The searching space is too large to be practical even efficient

- **Objective:**
  - Reduce number of candidates to \( O(mn) \)
  - How to do it ? (segmentation and consistency property)
Segmentation

- Segmentation on Time-series Databases
  - Fixed windows
  - Length of $I \leq \lambda/2$

- Segmentation on Query Series
  - Sliding windows
  - Length in $I - \lambda$

Only $O(mn/I)$ pairs of segments need to be checked.
- How to make sure there is no missing result? (no false negative, exact query)

Consistency Property

- Guarantee every pair of similar subsequences must include a pair of similar segments.

- Definition: Let $X$ and $Q$ be two sequences, then for every subsequence $SX$ of $X$, there exists a subsequence $SQ$ of $Q$, such that $\delta(SQ,SX) \leq \delta(Q,X)$. Then $\delta$ is a consistent distance.

- Consistency property is for time-series measurements.

Consistent Distances

- Consistent Distances
  - Euclidean Distance
  - Hamming Distance
  - Dynamic Time Warping
  - Levenshtein Distance
  - Fréchet distance
  - ERP
  - ...

- There are a few distances that do not satisfy the consistency property, such as: longest common subsequence (LCS), edit distance with real penalty (EDR)

Query Workflow

- Fixed Window
  - Build
  - Index
  - Reference Net

- Validation
  - Candidates
  - Index

- Query
  - Sliding Window

Reference Net

- Reference Net
  - Each node $R(i,j)$ is a reference. The range of reference $R(i,j)$ is $2^i$

- Reference-based Index
  - If $\delta(Q_1, X) + r < r_2$, we can claim that all distances between $Q_1$ and $X$ are definitely smaller than $r_2$.
  - If $\delta(Q_1, X) - r > r_1$, we can claim that all distances between $Q_1$ and every $X_i$ are definitely larger than $r_1$. 
Reference Net

Net, not Tree

- Net is a multi-parents hierarchical structure
  - Assume $|R_1, X_j| \leq \epsilon$ and $|R_2, X_j| \leq \epsilon$, but $X_j$ are only in the list of $R_2$. If $\delta(Q, R_1) + \epsilon > r$ we do not know whether $\delta(Q, X_j) \leq r$ or not.
  - However, if we maintain $X_j$ also in $R_1$, and $\delta(Q, R_1) + \epsilon \leq r$, we can make sure that $\delta(Q, X_j) \leq r$

4 Pruning Rules:

- Single list removal
- Single list commit
- Generated list removal
- Generated list commit

Index Space

Query on Proteins

Query on Songs

Query on Tracks
Future Work

- Practical Solutions for Database Systems:
  - Comprehensive Temporal Graph Analytic Platform
  - Real-Time Dynamic Temporal Graph Management Engine
  - Content-based Searching Engine on Time-series Databases
  - Comprehensive Time-series Analytic Platform
  - Real-Time Dynamic Time-series Data Management Engine

- Theoretical Contributions for Similarity Learning:
  - Node Sequence Similarity in Temporal Graph
  - Node Behaviors and Node Dynamic Roles Learning
  - Privacy Preserving Node Distance Computation
  - Graph Similarity - Graph Searching

Thank You!