Graph Algorithms and Use Cases - A Survey

Xinyu Chang Director of Customer Solutions

Overview

Overview

Graph algorithms can bring unique insights and significant enhancement to your data analytic and machine learning tasks in many use cases of different industries. In this session you will learn

- Functionality of each graph algorithm
- Mechanism of graph algorithms and their use cases
- Some real life use cases.

Overview

- Well-designed commonly-used graph algorithms written in GSQL
- TigerGraph's Advantages:
  - **Native MPP design** means faster execution
  - **GSQL was designed for analytic functions**
  - **Open-source and user-extensible**
    - Users can see the GSQL, learn from it, and modify it, to customize their algorithms.
    - In other platforms, the algorithms are embedded functions which cannot be viewed or modified.
    - On Github: https://github.com/tigergraph/gsql-graph-algorithms/tree/master/algorithms

Algorithms and Their Use Cases

Algorithm Types

- **Centrality**
  Indicators of centrality assign numbers or rankings to nodes within a graph corresponding to their network position
- **Classification**
  Classify the graph into sets
- **Community**
  A network is said to have community structure if the nodes of the network can be easily grouped into sets of nodes such that each set of nodes is densely connected internally
- **GraphML/Embeddings**
  A graph embedding is an approach converting neighborhood topology into a fixed size vector of decimal values
- **Path**
  Identify path between vertexes and path analytics
- **Similarity**
  Computes similarity between pairs of items
- **Topological Link Prediction**
  Link prediction is the problem of predicting the existence of a link between two entities in a network
- **Frequent Pattern Mining**
  An analytical process that finds frequent patterns

Algorithms

- **Centrality**
  - PageRank
  - Article Rank
  - Betweenness
  - Closeness
  - Degree
  - Eigenvector
  - Harmonic
  - Influence Maximization
- **Community**
  - Connected Components
  - K Core
  - K Means
  - Label Propagation
  - Local Cluster Coefficient
  - Louvain
  - Speaker-Listener Label Propagation
  - Triangle Counting
- **GraphML/Embeddings**
  - FastRP
  - Node2Vec
- **Path**
  - Astar_shortest_path
  - BFS
  - Cycle_detection
  - Estimated_diameter
  - Maxflow
  - Minimum_spanning_forest
  - Minimum_spanning_tree
  - Shortest Path
- **Classification**
  - Greedy Graph Coloring
  - Maximal_independent_set
- **Similarity**
  - Cosine
  - Jaccard
  - K Nearest Neighbors
  - Approximate Nearest Neighbors
- **Topological Link Prediction**
  - Adamic Adar
  - Common Neighbors
  - Preferential Attachment
  - Resource Allocation
  - Same Community
  - Total Neighbors
- **Frequent Pattern Mining**
  - Apriori

Centrality Algorithms

Name	Mechanism	Use Case
Eigenvector	A vertex's influence is based on its indegree and the influence of the vertices which refer to it.	• Finding influencer from a social network • Ranking web pages for searching engine • Identify the marketing targets within a doctor referral network • Personalized version can be used for user user behavior prediction and recommender systems
PageRank	Variant of Eigenvector. Adding additional jump probability	
Article Rank	Variant of PageRank. Lowers the influence of low-degree nodes	
Closeness	Find vertexes that have shorter distance to all other nodes	• Recommend locations for a grocery store
Harmonic	Variant of Closeness Centrality. Reverses the sum and reciprocal operations in the definition of closeness centrality	• Same as Closeness, but works better when run on disconnected graph.

Centrality Algorithms (Cont.)

Name	Mechanism	Use Case
Degree	Returns number of indegree or outdegree	• Find hub nodes • Find the most purchased products • Find the most followed twitter accounts • Sanity check on the graph data
Betweenness	Number of shortest paths that pass through this vertex, divided by the total number of shortest paths.	• Recommend locations for transportation center • AML. Find the bank accounts with most of the money transfers flows through
Influence Maximization	Finding a small subset of vertices in a social network that could maximize the spread of influence.	• Outbreak minimization

Community

Name	Mechanism	Use Case
Weakly Connected Components (WCC)	Every vertex is reachable from every other vertex through an undirected path	• Entity Resolution, obtain a holistic picture of an given user entity • Fraud ring detection
Strongly Connected Components	Every vertex is reachable from every other vertex through an directed path	• AML, money laundering ring detection
K Core	Maximal connected subgraph in which every vertex is connected to at least k vertices in the subgraph	• Network analytical characterization of hierarchical layers (Hierarchical result based on different k values)
K-means	Partition vertexes into k clusters in which each vertex belongs to the cluster with the nearest mean	• Grouping transportation orders into clusters based on longitude and latitude • Grouping product into categories based on their embeddings

Community (Cont.)

Name	Mechanism	Use Case
Label Propagation	If the plurality of your neighbors all bear the label X, then you should label yourself as also a member of X.	• Anti-fraud, detect fraud rings that is a group of applications/accounts that share common PII info • Detect user communities in a social network, at later stage, one could extract features of each user communities for marketing purposes
Speaker-Listener Label Propagation	Variant of the Label Propagation algorithm that is able to detect overlapping communities.	
Louvain	Partitions the vertices in a graph by approximately maximizing the graph's modularity score	
Local Cluster Coefficient (LCC)	local clustering coefficient = Number_trangles/((n-1)n/2)	• Anti phone call fraud, how many pairs of your callees called each other
Triangle Counting	Count the number of triangles in the graph	• Evaluate the cohesiveness of communities

GraphML/Embeddings

Name	Mechanism	Use Case
FastRP	It generates node embeddings of low dimensionality through random projections from the graph's adjacency matrix to a low-dimensional matrix	• Use the embeddings as machine Learning features • Use the embeddings for similarity algorithms
Node2Vec	Uses random walks in the graph to create a vector representation of a node.	

Path

Name	Mechanism	Use Case
Astar Shortest Path	A shortest path algorithm guided by a heuristic function that estimates the cost of the cheapest path	• Route planning. Planning the best route for delivering an order based on train schedules • Flight search. Planning a series of flights that take least amount of travelling time • Anti-fraud. Analyse relationship between a new credit card application and a blacklisted application
Shortest Path	Return either the shortest or top k shortest unweighted/weighted paths connecting a set of source vertices to a set of target vertices	
Cycle Detection	Find all the cycles (loops) in a graph	• AML, find the transaction loops
Estimated Diameter	The worst-case length of a shortest path between any pair of vertices in a graph	• Understand the interconnectedness of the graph to support following analytics, e.g. to choose small world version of WCC or standard version

Path (cont.)

Name	Mechanism	Use Case
Maxflow	Finding a feasible flow through a flow network that obtains the maximum possible flow rate	• AML, find the maximum money flow between two accounts • Supply chain, find the maximum supply flow from one site to another
Minimum Spanning Tree	A minimum spanning tree is a set of edges that can connect all the vertices in the graph with the minimal sum of edge weights	• Travelling salesman problem
Minimum Spanning Forest	A set of minimum spanning trees, one for each WCC	

Classification

Name	Mechanism	Use Case
Greedy Graph Coloring	Assigns a unique integer value known as its color to the vertices of a graph such that no neighboring vertices share the same color	• Find the maximum tasks that can be performed together without conflicting with each other • Find the maximum of vertexes that can execute some logic in parallel without conflicting with each other
Maximal Independent Set	An independent set of vertices does not contain any pair of vertices that are neighbors	

Similarity

Name	Mechanism	Use Case
Cosine Similarity	Calculates cosine similarity based on topological common attributes	• Find similar users, patients • User purchased item A also purchased product B
Jaccard Similarity	Calculates jaccard similarity based on topological common attributes	
Approximate Nearest Neighbors	Calculate the approximate nearest neighbors with similarity score based on non-topological attribute values	
K Nearest Neighbors	Based on similarity scores makes a prediction for every vertex whose label is not known	• Anti-fraud, predict fraud labels

Topological Link Prediction

Name	Mechanism	Use Case
Adamic Adar	Number of shared links between two nodes	• Collaborative filtering for recommender systems
Common Neighbors	Number of common neighbors between two vertices	
Preferential Attachment	Product of the number of neighbors of the two vertices.	
Resource Allocation	Reciprocal of number of neighbors of common neighbors	

Topological Link Prediction

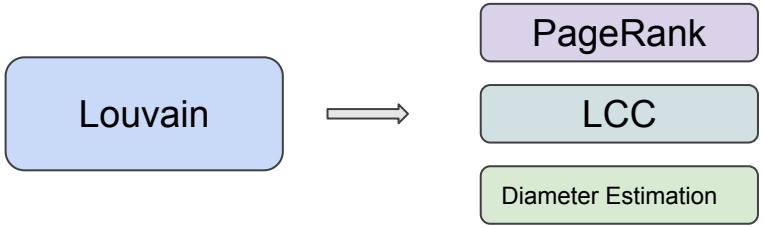
Name	Mechanism	Use Case
Same Community	Returns 1 if they two vertices are in the same community, and returns 0 if they are not in the same community.	• Collaborative filtering for recommender systems
Total Neighbors	Total number of neighbors of two vertices	

Frequent Pattern Mining

Name	Mechanism	Use Case
Apriori	Returns top k most frequent sequential patterns	• Application UX improvements • Customer churn prediction • Bad rating cause analysis

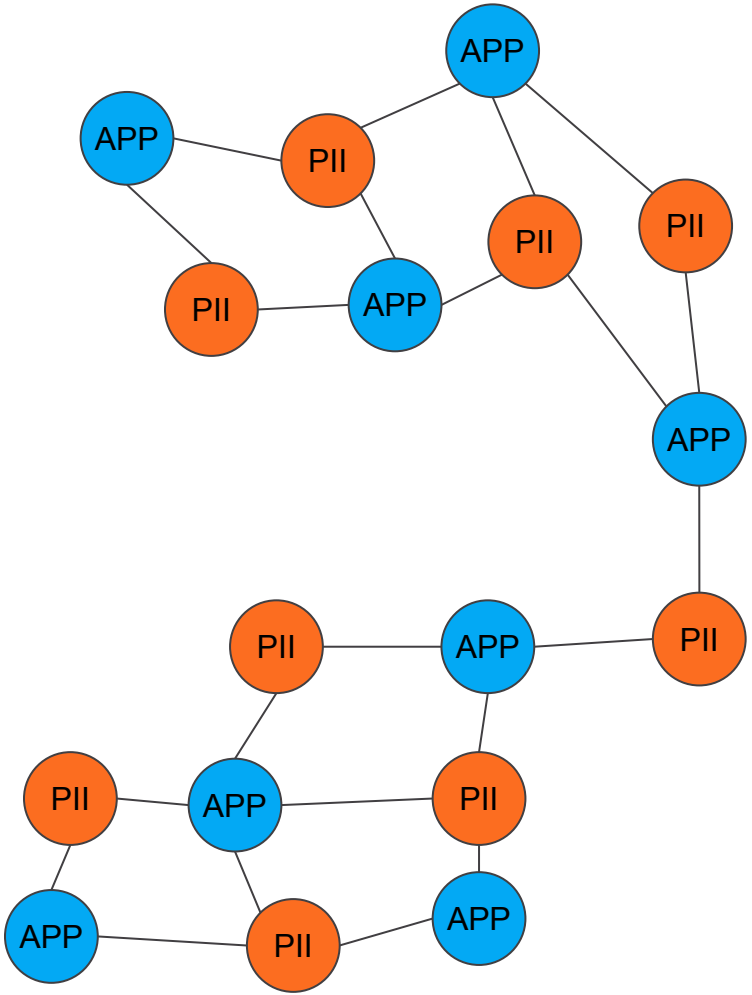
Real Life Use Cases

Use Case 1 Anti-Fraud



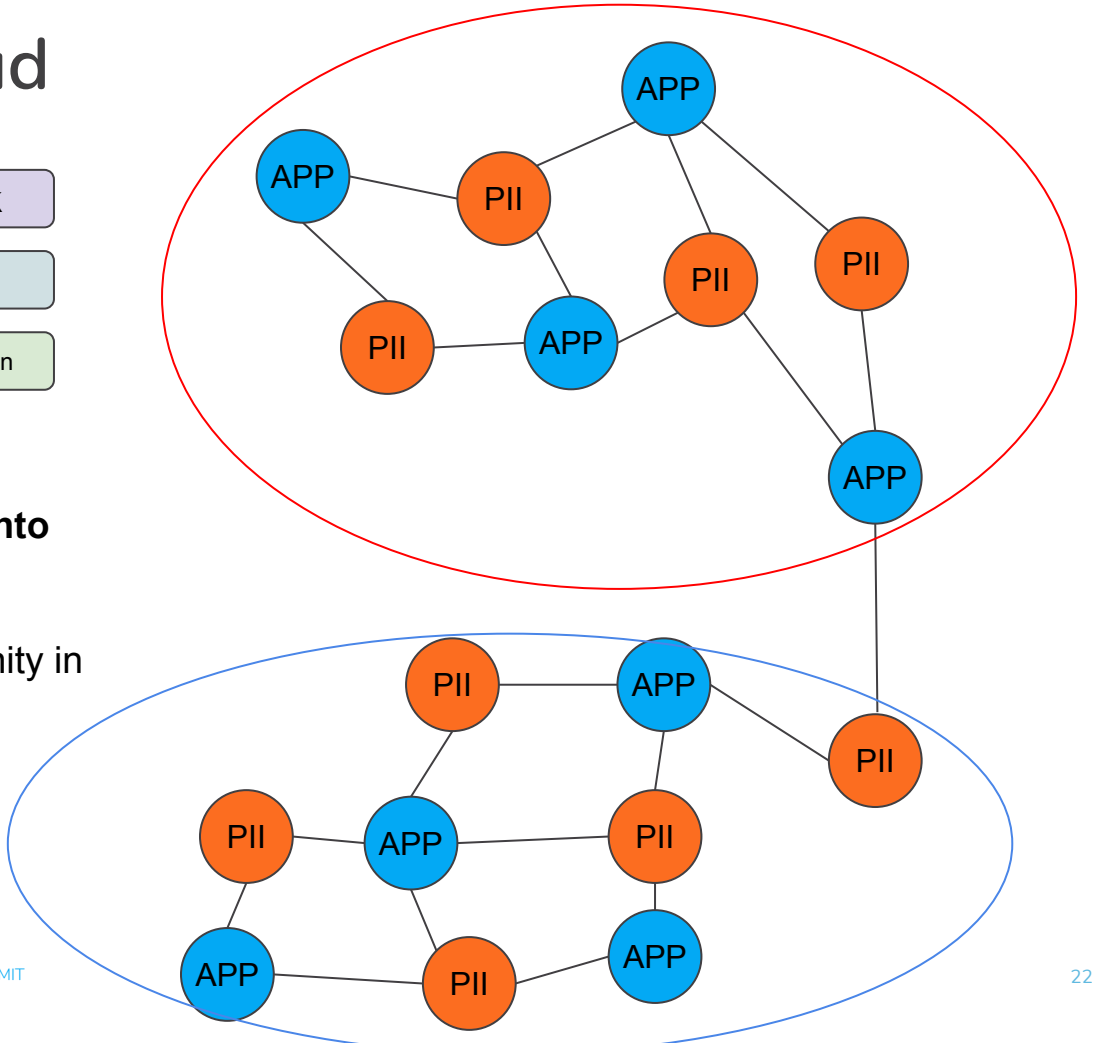
1. Divide the application-pii graph into communities

2. Extract features from each community in order to identify fraud rings



Use Case 1 Anti-Fraud

Louvain ⟹ PageRank, LCC, Diameter Estimation



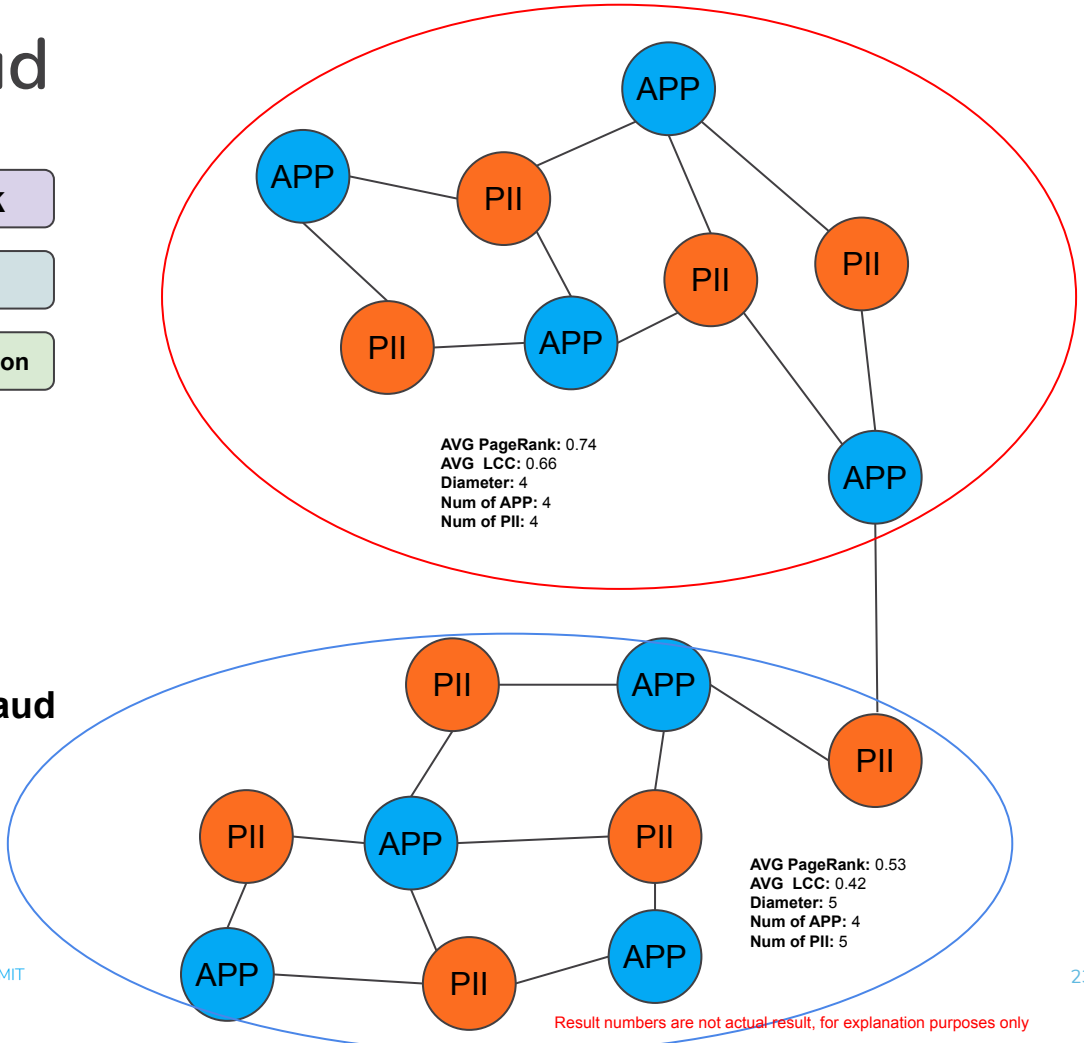
1. **Divide the application-pii graph into communities**

2. Extract features from each community in order to identify fraud rings

Use Case 1 Anti-Fraud

Louvain ⟹ **PageRank** | **LCC** | **Diameter Estimation**

1. Divide the application-pii graph into communities
2. **Extract features from each community in order to identify fraud rings**



Result numbers are not actual result, for explanation purposes only

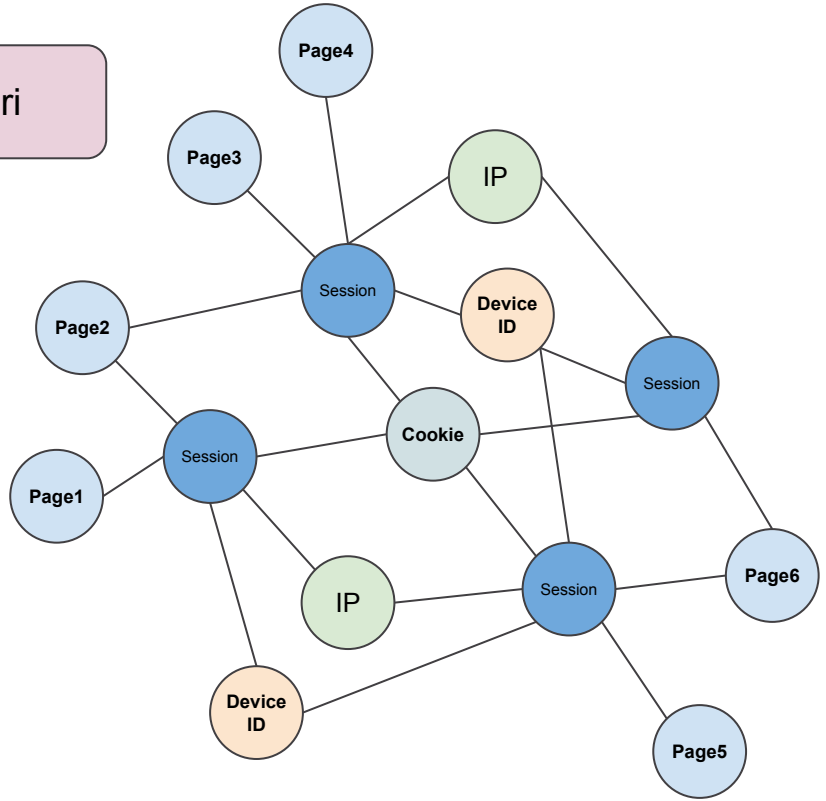
Use Case 2 Digital Fingerprint Entity Resolution

Cosine Similarity → WCC → Apriori

1. Connect the web sessions that share digital figureprints with similarity score beyond the threshold

2. Run WCC to connect the sessions that are connected to a unified entity vertex

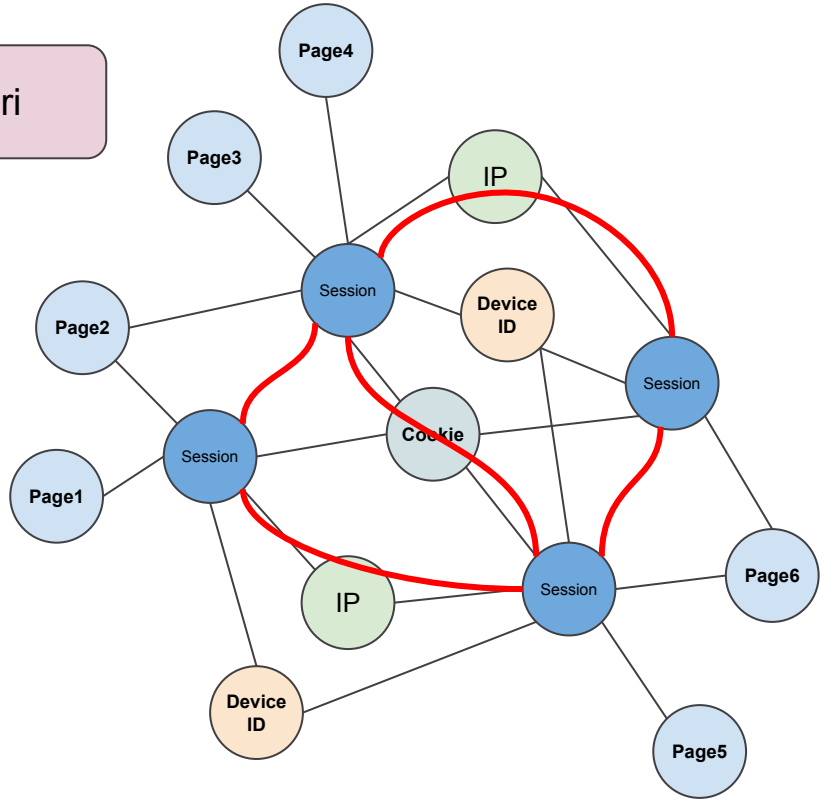
3. Run frequent sequential pattern mining algorithms to analyze user behaviors



Use Case 2 Digital Fingerprint Entity Resolution

Cosine Similarity → WCC → Apriori

1. **Connect the web sessions that share digital figureprints with similarity score beyond the threshold**
2. Run WCC to connect the sessions that are connected to a unified entity vertex
3. Run frequent sequential pattern mining algorithms to analyze user behaviors



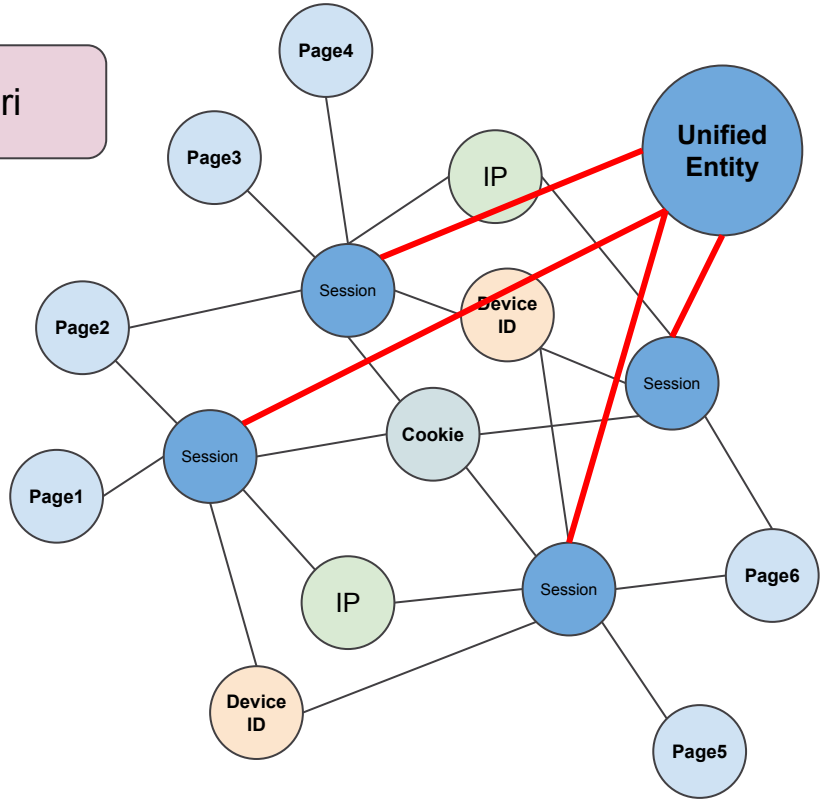
Use Case 2 Digital Fingerprint Entity Resolution

Cosine Similarity → **WCC** → Apriori

1. Connect the web sessions that share digital figureprints with similarity score beyond the threshold

2. **Run WCC to connect the sessions that are connected to a unified entity vertex**

3. Run frequent sequential pattern mining algorithms to analyze user behaviors



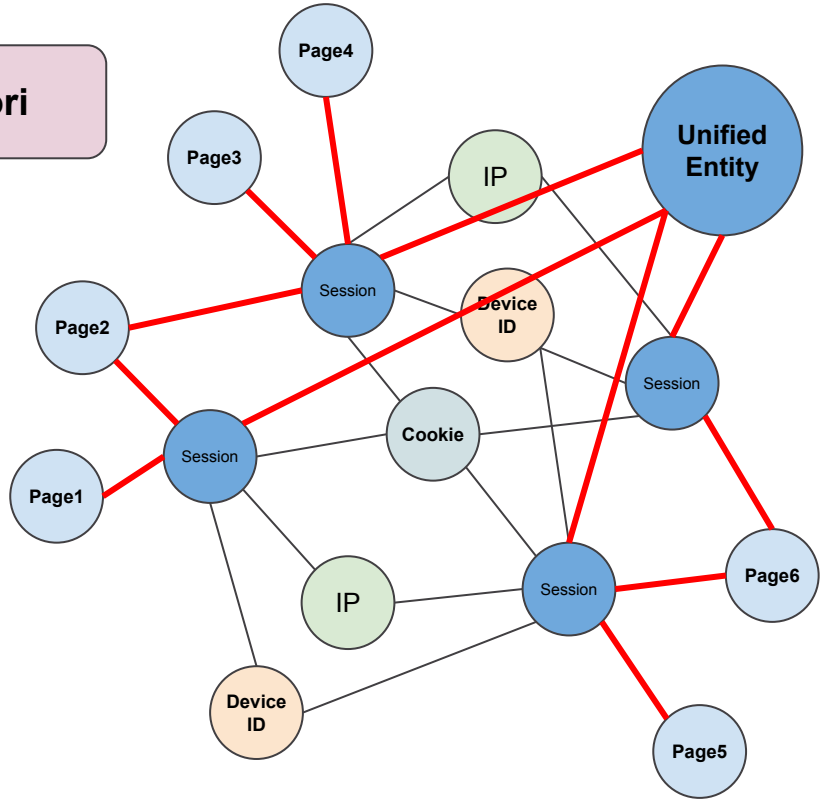
Use Case 2 Digital Fingerprint Entity Resolution

Cosine Similarity → WCC → **Apriori**

1. Connect the web sessions that share digital figureprints with similarity score beyond the threshold

2. Run WCC to connect the sessions that are connected to a unified entity vertex

3. **Run frequent sequential pattern mining algorithms to analyze user behaviors**



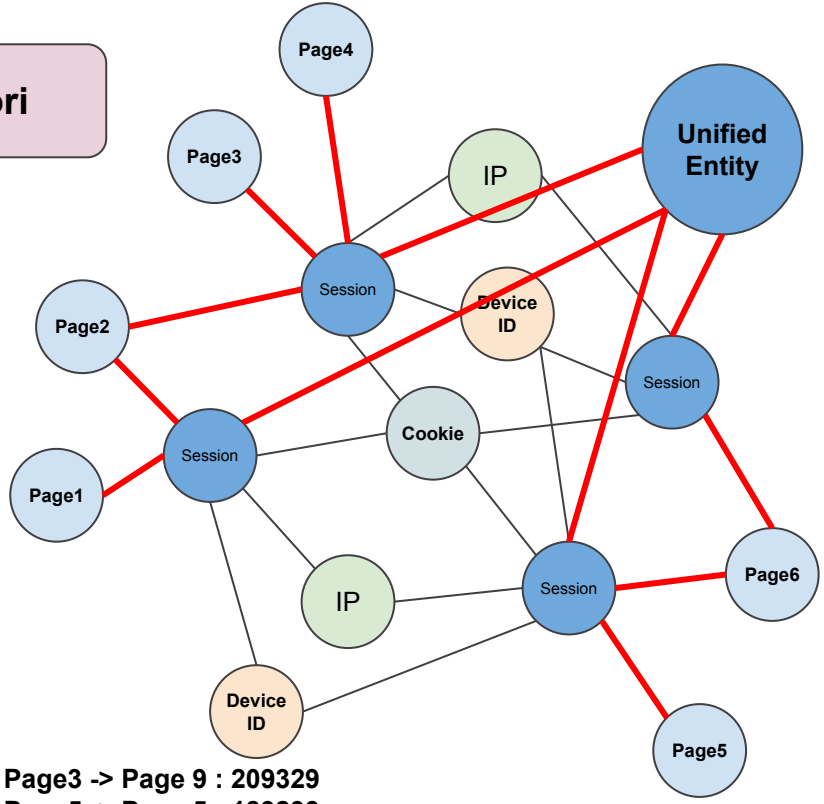
Use Case 2 Digital Fingerprint Entity Resolution

Cosine Similarity → WCC → **Apriori**

1. Connect the web sessions that share digital figureprints with similarity score beyond the threshold

2. Run WCC to connect the sessions that are connected to a unified entity vertex

3. **Run frequent sequential pattern mining algorithms to analyze user behaviors**

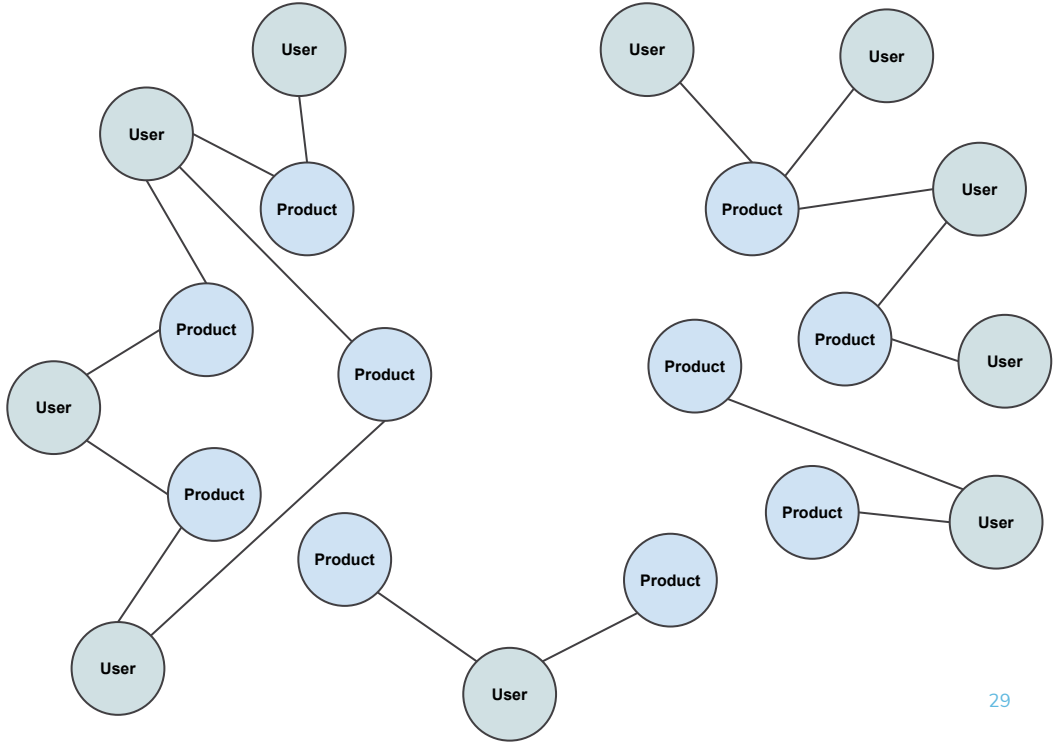


Page1 -> Page3 -> Page 9 : 209329
Page1 -> Page5 -> Page 5 : 189293
...

Result numbers are not actual result, for explanation purposes only

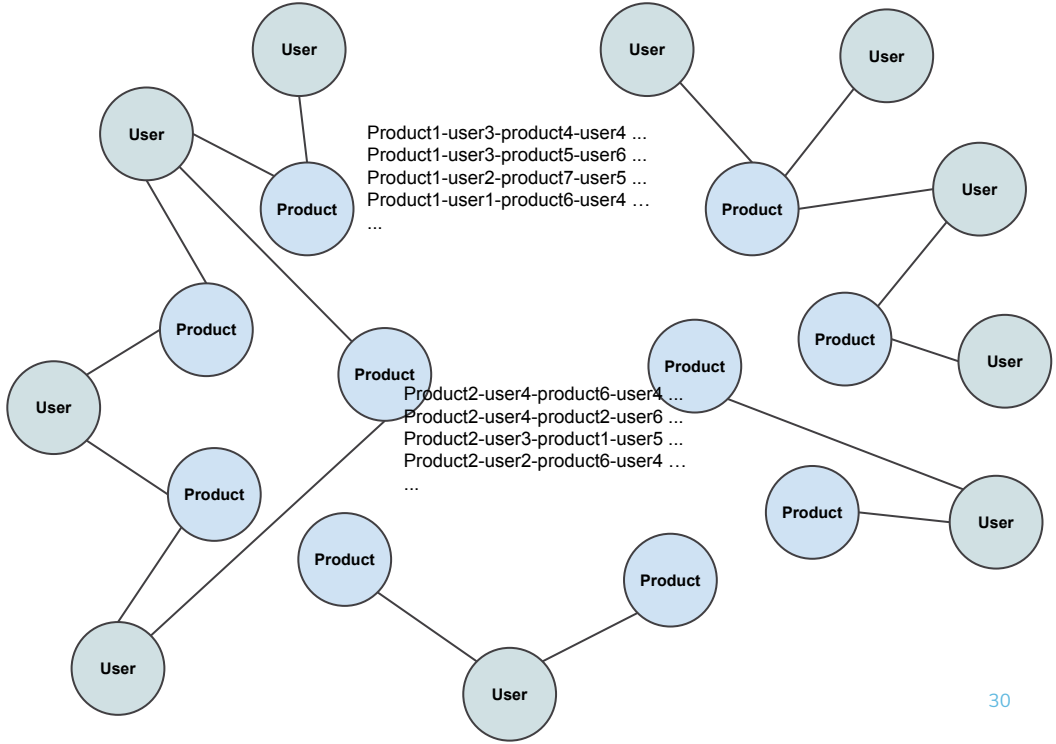
Use Case 3 Product Categorization

1. Run node2Vec to calculate the embeddings of the products
2. Run k-Means to divide the products into k clusters
3. Within each cluster Run ANN to connect the top k most similar products
4. Run SCC to divide the product into finer categories



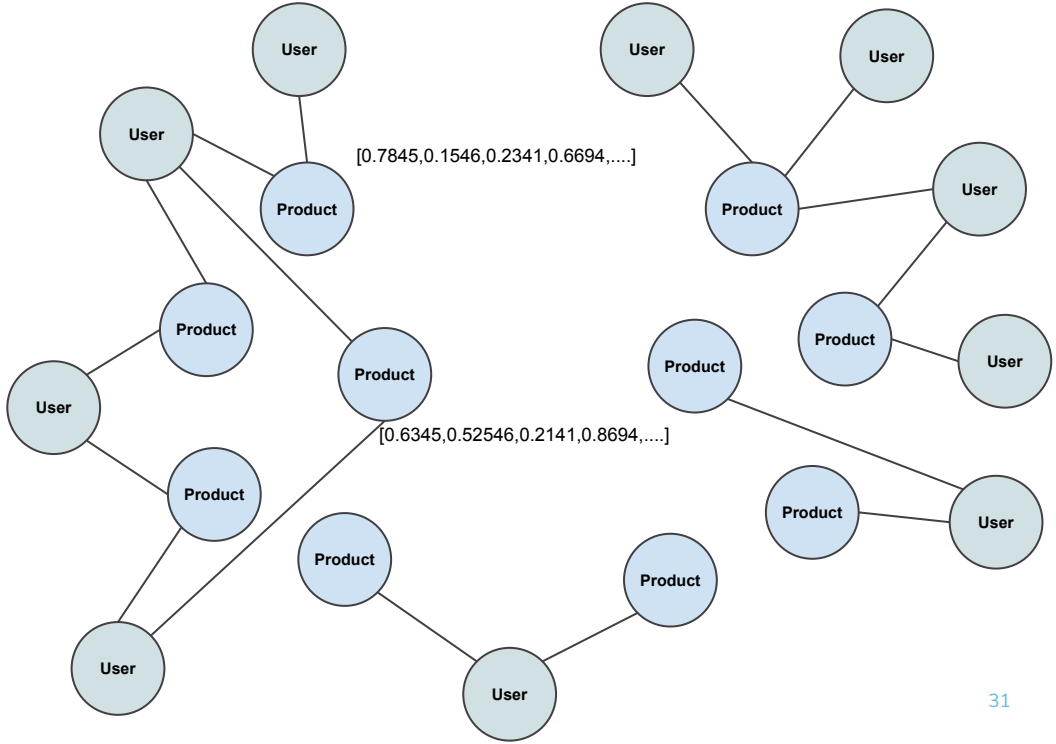
Use Case 3 Product Categorization

1. **Run node2Vec to calculate the embeddings of the products**
2. Run k-Means to divide the products into k clusters
3. Within each cluster Run ANN to connect the top k most similar products
4. Run SCC to divide the product into finer categories



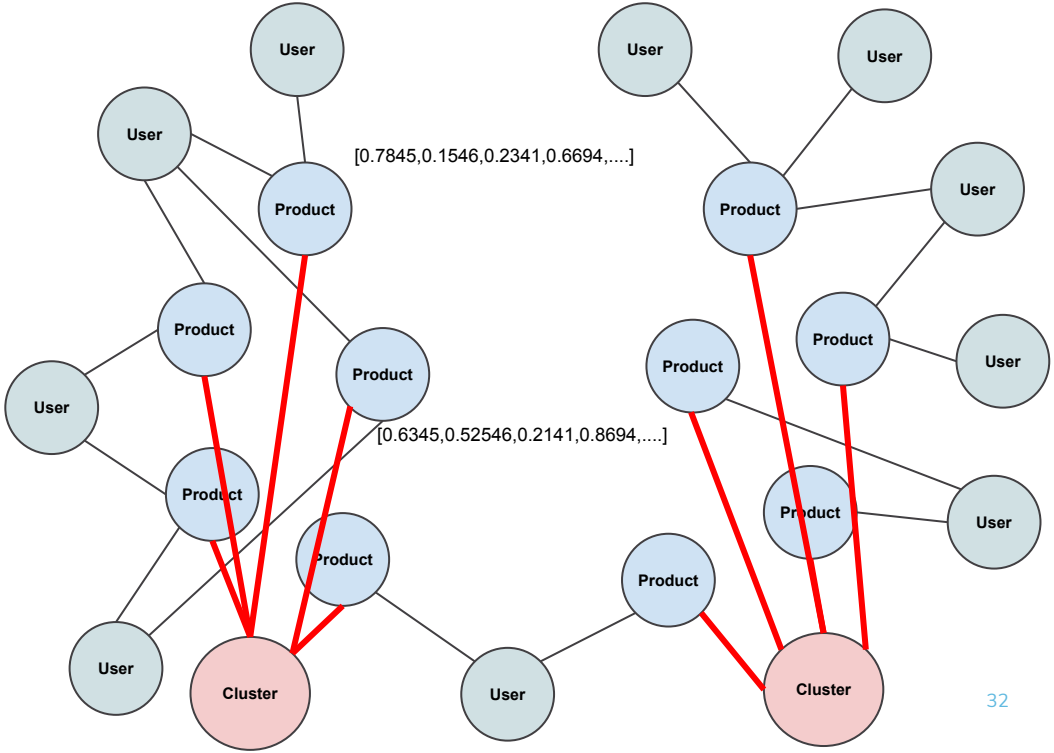
Use Case 3 Product Categorization

1. **Run node2Vec to calculate the embeddings of the products**
2. Run k-Means to divide the products into k clusters
3. Within each cluster Run ANN to connect the top k most similar products
4. Run SCC to divide the product into finer categories



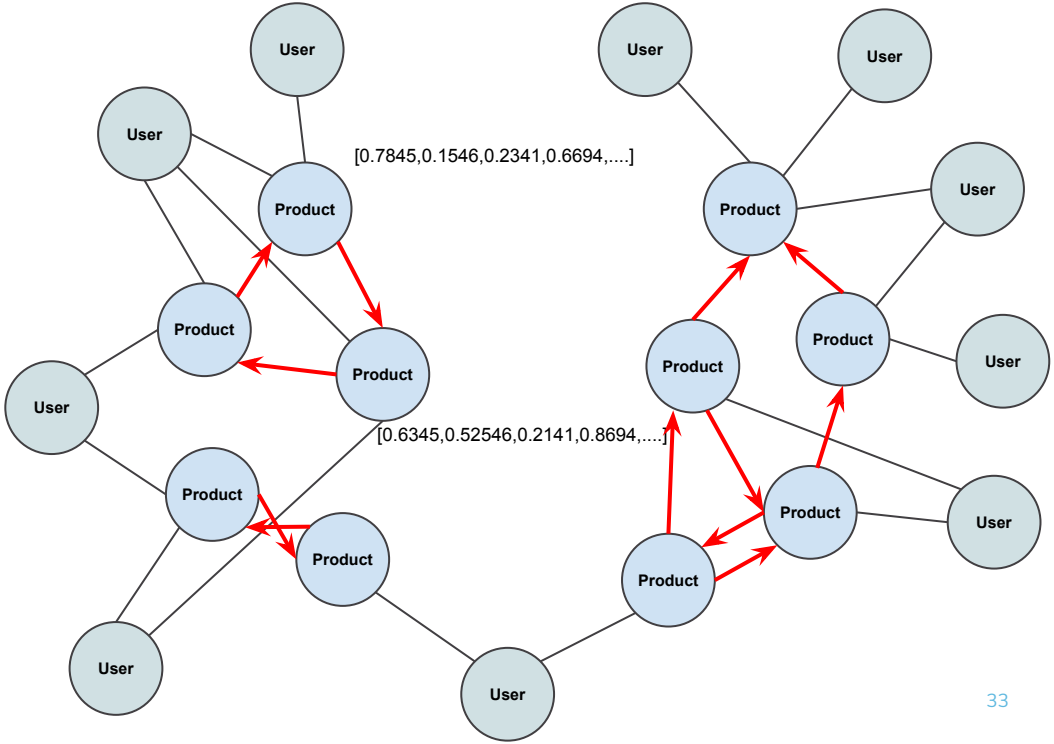
Use Case 3 Product Categorization

1. Run node2Vec to calculate the embeddings of the products
2. **Run k-Means to divide the products into k clusters**
3. Within each cluster Run ANN to connect the top k most similar products
4. Run SCC to divide the product into finer categories



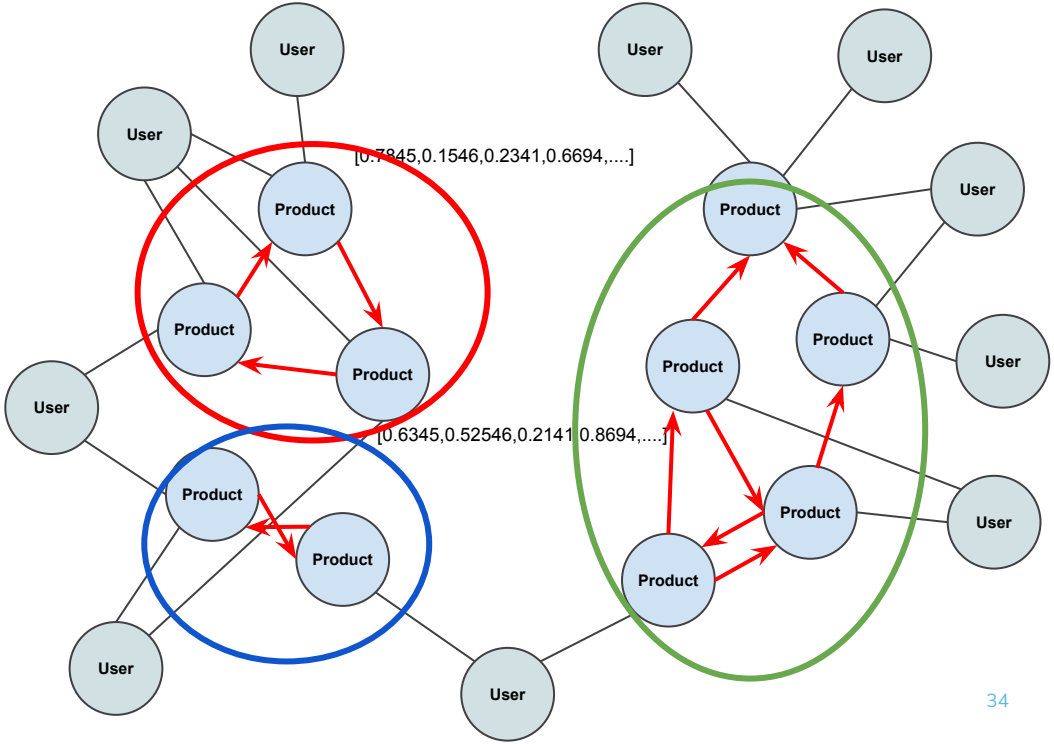
Use Case 3 Product Categorization

1. Run node2Vec to calculate the embeddings of the products
2. Run k-Means to divide the products into k clusters
3. **Within each cluster Run ANN to connect the top k most similar products**
4. Run SCC to divide the product into finer categories



Use Case 3 Product Categorization

1. Run node2Vec to calculate the embeddings of the products
2. Run k-Means to divide the products into k clusters
3. Within each cluster Run ANN to connect the top k most similar products
4. **Run SCC to divide the product into finer categories**



Q&A