Multi-label Graph Analysis and Computations Using GraphX

Qingbo Hu[1], Qiang Zhu[2]

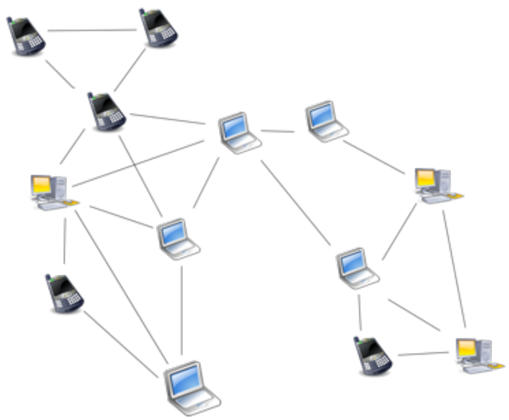
1. Qingbo Hu is a Senior Business Analytics Associate at LinkedIn
2. Qiang Zhu currently works for Airbnb. The work introduced in this talk was done when he was a manager at LinkedIn

Overview

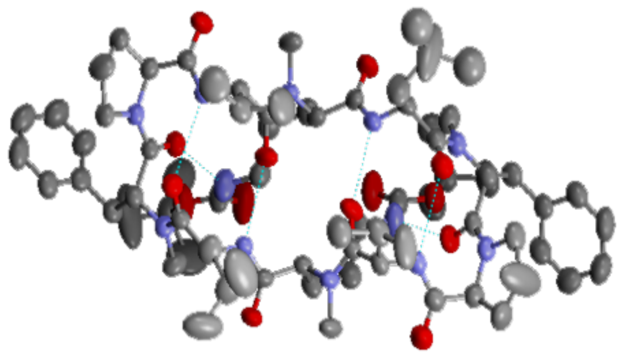
- Background
- Motivation and Goal
- Constructing Multi-label Graphs
- Multi-label PageRank
- Experiments
- Conclusion

Background

- Network Analysis
  - Applications:



Telecommunication Network



Bioinformatics

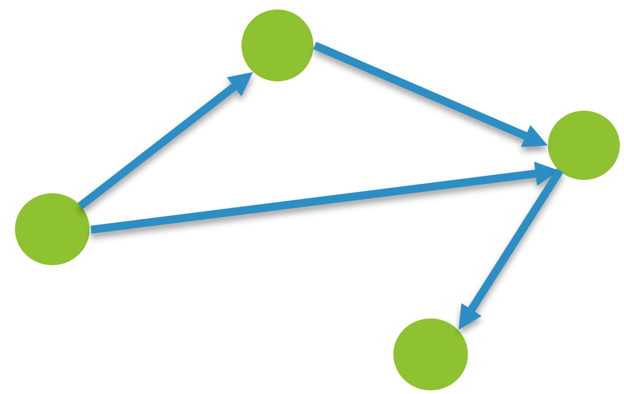
Social Network

Background

- Network Analysis (cont'd)
  - Features of interest:
    - (In/Out) degrees
    - # triangles
    - (Strongly) connected component
    - Etc.
  - Graph-based algorithms
    - PageRank [1]
    - Label Propagation [2]
    - HITS [3]
    - etc.

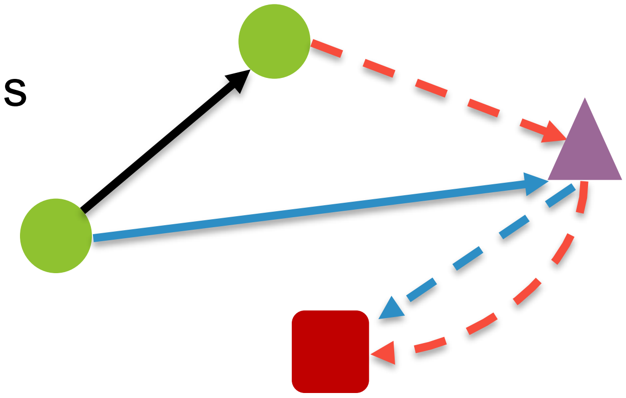
Motivation and Goal

- Homogeneous Network
  - Single type of nodes and single type of edges
  - Example:
    Citation networks: author, citation
    Friendship networks: user, friendship
  - Not enough to depict complicated real-life networks
  - Supported by GraphX



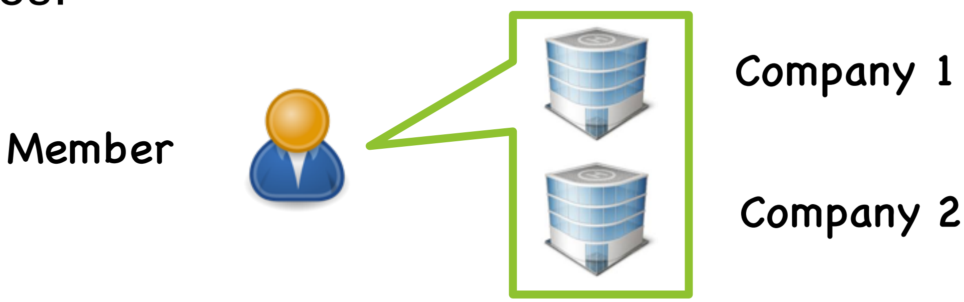
Motivation and Goal

- Heterogeneous Networks
  - Nodes of multiple types and edges of multiple types
  - Example:
    Social Network User Activity Graph: user, reply, comment, like etc.
    LinkedIn Economic Graph: member, company, employment, connection etc.
  - Better resembles real-life networks
  - Can be represented by **labels** on nodes and edges
    ➡ **Multi-label graphs**
  - Not directly supported by GraphX

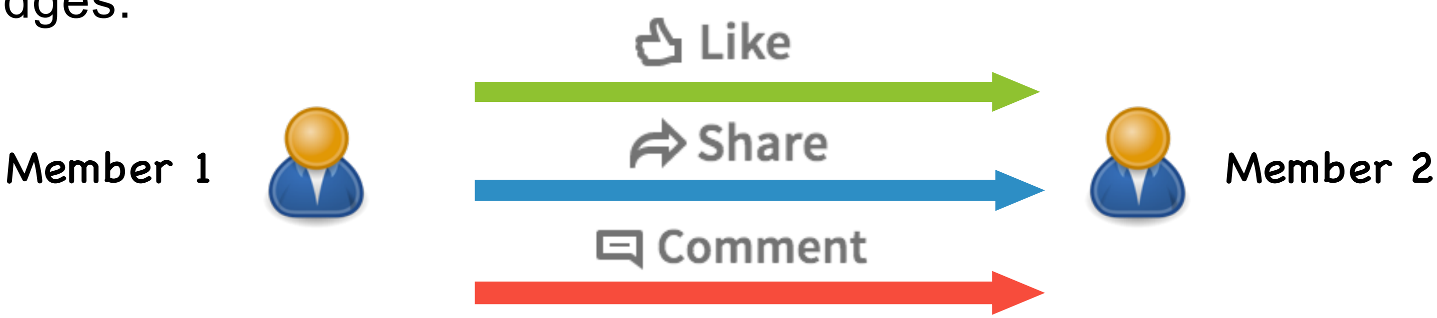


Motivation and Goal

- Social activity graph on LinkedIn
  - Nodes:



  - Edges:



Motivation and Goal

- Social activity graph on LinkedIn (cont'd)
  - Questions:
    - How many times a member likes/comments/shares other people's posts?
    - Who has the highest PageRank score in each company with respect to like/comment/share behavior?
    - Etc.

Network features with respect to labels

Graph-based algorithm on label level

Spark + GraphX

- No direct support
- Multiple subgraphs for different labels => waste of time and resource
- A unified solution is preferred

Motivation and Goal

- Solutions based on GraphX to provide Multi-label graph analysis
- Short-term goals
  - Construction of multi-label graphs
  - Efficient computation of PageRank score with respect to all labels
- Long-term goals
  - A general API library supports the following additional operations:
    - Multi-label Graph transformation
    - Network features on the label level
  - Implementations for additional common graph-based algorithms
    - Label Propagation
    - HITS
    - Etc.

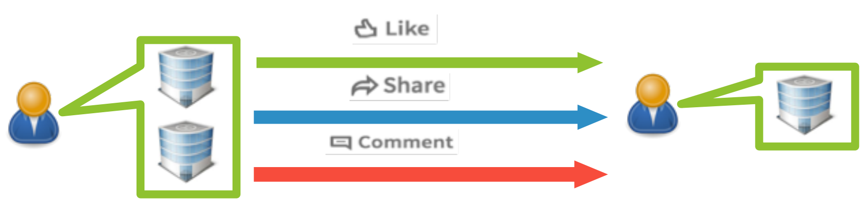
Constructing Multi-label Graphs

- Node
  - (ID, labels, nodeFeatures)
    - ID: a unique long associated with the node
    - labels: A set contains node labels
    - nodeFeatures: Other node dependent features
- Edge
  - (fromID, toID, label, edgeFeatures)
    - fromID: the ID of the edge’s source node
    - toID: the ID of the edge’s target node
    - label: A label associated with the edge
    - edgeFeatures: Other edge dependent features

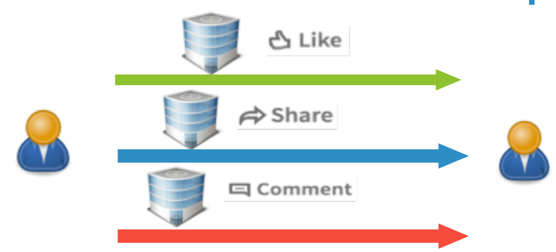
Constructing Multi-label Graphs

- Node labels vs. edge labels
  - Edge label is more important in many network features
    - PageRank score, (in/out) degrees, strongly connected component etc.
  - Node labels are used to filter nodes
  - Why?
    - Edge labels are usually used to form meaningful subgraphs
      - Random walk follows edges, degrees are respect to edge labels etc.
    - Node labels can be absorbed in edges if necessary
      - a graph transform operation

Top influencers for each company



Top influencers within each company

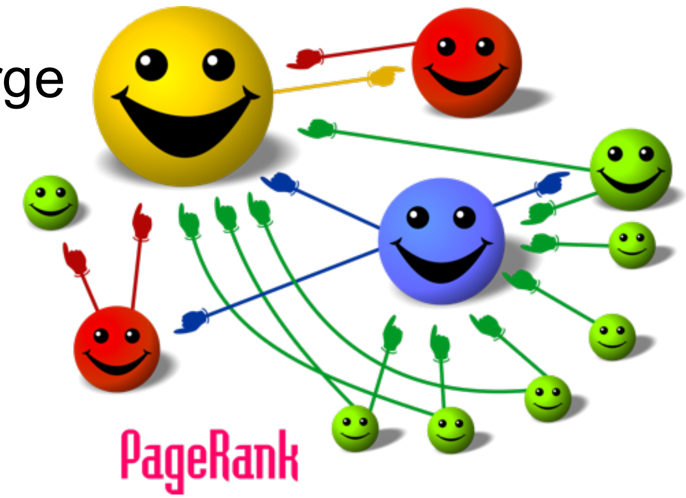


Constructing Multi-label Graphs

- Methods to create a multi-label graph
  - NodeRDDs + EdgeRDDs
  - EdgeRDDs (no node labels)
  - Load directly from file:
    A list of edges: (source, target, label)
    A list of nodes: (ID, label_1, label_2, …, label_n) => optional
  - Transformation from other multi-label graphs

Multi-label PageRank

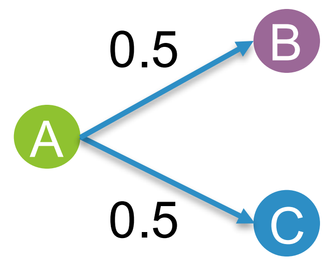
- PageRank
  - Developed by Larry Page and Sergey Brin
  - Used to rank web pages
  - Important pages are always linked by other important pages
  - Iteratively updating scores until they converge
  - The obtained score: PageRank score



Multi-label PageRank

- PageRank (cont'd)
  - For an edge (p_j, p_i), the edge weight is defined by 1/$L(p_j)$,where $L(p_j)$ is the out degree of p_j
  - Initial score for every node: 1.0 or 1.0 / N
  - Later iteration:

$$PR(p_i) = \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$



  - In order to ensure convergence, we allow a small probability to be "teleported" to any node (reset probability)

$$PR(p_i) = 1 - d + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

or

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

Multi-label PageRank

- PageRank (cont'd)
  - Power iteration through matrix manipulation
    - Vector: scores
    - Matrix: transitional matrix
    - Each iteration: vector * matrix
    - Waste resource if the transitional matrix is sparse
  - Directly simulate the computation process
    - Easier for parallel implementation
    - Pregel

Multi-label PageRank

- Pregel
  - A general programming interface for graph-based algorithms
  - Proposed by Google
  - Supported by GraphX
  - Iterative algorithm until convergence conditions are met
  - For each iteration, we need to consider:
    1. How to construct the message passed along edges?
       => Message sender
    2. How to combine received messages on a node?
       => Message combiner
    3. How to use the combined message to update the info on a node?
       => Vertex Program

Multi-label PageRank

- Construct a graph used for PageRank computation
  - PageRankNodeType: **Map[Int, (Double, Double)]**
    - label: the label associated with the PageRank score
    - score: the value of PageRank score
    - score_diff: the difference of scores between two iterations
  - PageRankEdgeType: **[Int, Double]**
    - label: the label associated with the message
    - weight: the transitional probability on the edge
  - PageRankMsgType: **Map[Int, Double]**
    - label: the label associated with the message
    - message: a double valued score used to update PageRank score

Why do we use Map[Int, Double] instead of (Int, Double)?

Multi-label PageRank

- Message Sender

```
def sendMessage(edge: EdgeTriplet[PageRankNodeType, (Short, Double)]) = {
  // Label on the current edge
  val label = edge.attr._1
  if (edge.srcAttr(label)._2 > tol) {
    val msg = mutable.Map[Short, Double]()
    msg += label -> edge.srcAttr(label)._2 * edge.attr._2
    Iterator((edge.dstId, msg))
  }
  else {
    Iterator.empty
  }
}
```

Create the message to be passed on the edge as a map

Multi-label PageRank

- Message Combiner

```
def messageCombiner(a : PageRankMsgType, b : PageRankMsgType) :
PageRankMsgType = {
 a ++ b.map{ case (k,v) => k -> (v + a.getOrElse(k, 0.0))}
}
```

Combine received maps into a single one

Multi-label PageRank

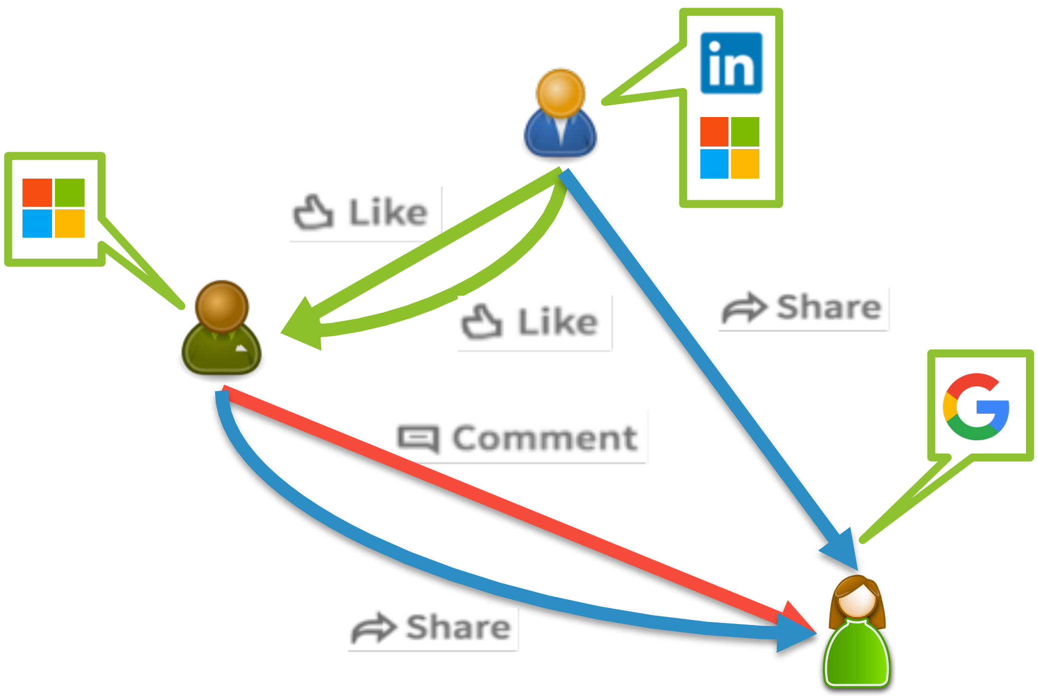
- Vertex Program

```
def vertexProgram(id: VertexId, attr: PageRankNodeType,  msgSum:
PageRankMsgType): PageRankNodeType  = {
  …
  attr.map{
    case (label, (oldPR, lastDelta)) => {
      val newPR = oldPR + (1.0 - resetProb) * msgSum.getOrElse(label, 0.0)
      val newDelta = newPR - oldPR
      (label -> (newPR, newDelta))
    }
  }
}
```

Using combined message to update PageRank score

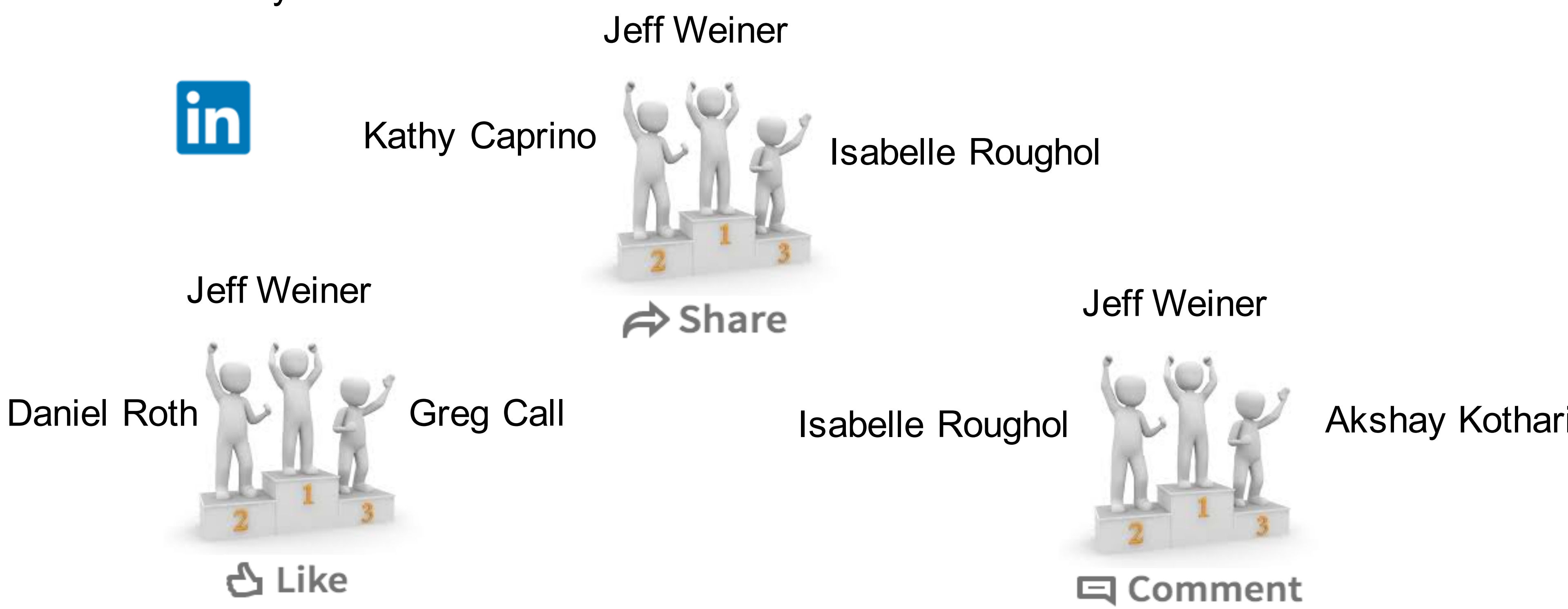
Experiments

- LinkedIn social activity graph
  - Sampled from all social activities in Nov. 2016
  - Nodes: ~2 million users
  - Node labels: companies
  - Edge labels:
    - Like
    - Share
    - Comment
  - Edges: ~76 million
  - Rest probability: 0.15
  - Convergence granularity: 1e-3
  - Number of executor: 50
  - Executor cores: 3
  - Executor Memory: 12G



Experiments

- Convergence around 100 iterations
- Total running time: 30~40 mins
- A case study for LinkedIn:



Experiments

- Further discussions and lessons learned
  - For edge type in multi-label graphs
    (fromID, toID, label, edgeFeatures) => (fromID, toID, Map(label, edgeFeatures))
    - Reduce duplication and save space
    - Slower process time
  - Standard Pregel interface in GraphX
    - Although data from the last iteration is unpersisted, DAG will keep grow
    - Might cause out of memory error
    - Pregel interface with (local) checkpoint to cut off the DAG after several iteration
  - Test on larger data sets and various data sources

Conclusion

- Network Analysis
  - Graph features
  - Graph-based algorithms
- Homogeneous vs. Heterogeneous Networks
- Multi-label Graphs
  - Node & Node labels
  - Edge & Edge labels
  - Constructing a multi-label graph
- Multi-label PageRank
  - PageRank
  - Pregel-based implementation
- Experiments

References

[1] Page, Lawrence, et al. *The PageRank citation ranking: Bringing order to the web*. Stanford InfoLab, 1999.

[2] Zhu, Xiaojin, and Zoubin Ghahramani. "Learning from labeled and unlabeled data with label propagation." (2002): 1.

[3] Kleinberg, Jon M. "Hubs, authorities, and communities." *ACM computing surveys (CSUR)* 31.4es (1999): 5.

Thank You!

Qingbo Hu (qihu@linkedin.com)