Jaccard Coefficients
What is the Goal of Computing Jaccard?

• Compute the similarity between the neighborhoods of two nodes

Which 2 entities share >1 addresses?
Jaccard Use Cases: Community Detection

- Originally introduced the context of geographical location of botanical species
- Has since been studied extensively for community detection (with many variations)
Jaccard Use Cases: Computing Similarity Between Wikipedia Pages

- Computes similarity between pages
- Algorithm uses MapReduce (Hadoop)
- Demonstrates that Jaccard is highly parallelizable, and scales well
Jaccard: Potential Graph Benchmark

• Compared to BFS:
  – Jaccard focuses computation towards dense neighborhoods
  – Jaccard Larger computation $O(N^3)$ work (worst case)
  – Jaccard can be adapted towards streaming variants
What is a Jaccard Coefficient?

- Similarity between neighborhoods of two nodes \((V, U)\):
  - \(\Gamma(u,v) = |N(V) \cup N(U)|\)
  - \(\varepsilon(u, v) = |N(V) \cap N(U)|\)
  - \(Jaccard(V, U) = \frac{\Gamma(u, v)}{\varepsilon(u, v)}\)
  - \(\varepsilon(A, C) = 1\)
  - \(\Gamma(A, C) = 3\)
  - \(Jaccard(A, C) = 1/3\)
Metrics for Jaccard

- Standard wall-clock time
- Jaccards per second (JACS)
  - Useful for scalability and comparing across machines
Jaccard Naïve Sequential Algorithm

• Comes down to being able to compute intersection of neighborhoods ($\gamma(u, v)$)
  
  $\gamma(u, v) = |N(V) \cap N(U)|$
  
  $\Gamma(u,v) = |N(V)| + |N(U)| - \gamma(u, v)$

• Intersection algorithm:
  
  • For each vertex $Y$ in $N(V)$:
    
    ▸ If $Y$ is in $N(U)$
      
      » IntersectCounter++
Complexity of Computing Jaccard

- To compute Jaccard(U, V)
  - If lists of neighbors are sorted:
    - $O(M)$ – $M$ is max of outdegree of U or V
  - If lists of neighbors have to be sorted first
    - $O(M\log(M))$
  - Otherwise perform repeated searches:
    - $O(M^2)$
Compute Jaccard With GraphBLAS

• GraphBLAS
  – Linear Algebra package to perform graph operations
  – Can be used to compute Jaccard efficiently
  – Represent graph G as matrix A, compute A\*A=C
  – Values in C correspond to the intersection size
  – Complexity: O(nnz(A))
## Simple Example

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

![Diagram of network connections]
Jaccard with MapReduce

• 1 MapReduce ‘step’ has 3 phases:
  1. **Map** some function over the data
  2. **Group** pairs by key
  3. **Reduce** Each group to solve

• Two different implementations exist, they 3 and 5 steps
Next Algorithm

• Adapt idea from Triangle Counting algorithm

  – Partition graph into overlapping subsets so that each triangle is in at least one of the subsets
  – Use sequential Jaccard algorithm as black box
  – Combine results