Proximity in Graphs by using Random Walks

CS345a: Data Mining
Jure Leskovec and Anand Rajaraman
Stanford University
Q: what is most related conference to ICDM

A: Random walks!

[Sun, ICDM2005]
Neighborhood Search on Graphs

- ICDM
  - PKDD
    - SDM
  - PAKDD
    - KDD
    - ICML
    - ICDE
    - SIGMOD
- CIKM
- ECML
- DMKD
Automatic Image Captioning

A: Proximity on graphs!

[Pan, KDD ‘04]

Sea Sun Sky Wave

Cat Forest Grass Tiger

{?, ?, ?,}
Automatic Image Captioning

Region

Image

Keyword

Sea Sun Sky Wave Cat Forest Tiger Grass
Automatic Image Captioning

[Pan, KDD '04]

Image

Region

Test Image

{Grass, Forest, Tiger}

Keyword

Sea  Sun  Sky  Wave  Cat  Forest  Tiger  Grass

Connection subgraphs:

What is the most likely connection between Andrew McCallum and Yiming Yang:
Center-Piece Subgraph (CEPS)

- **Given** Q query nodes
- **Find** Center-piece subgraph on $b$ nodes

**Input of CEPS:**
- Q Query nodes
- Budget $b$
- $k$-softAND coefficient
CEPS: 3 steps

- **Individual Score Calculation:**
  - Measure proximity of each node with respect to individual query node

\[
 r(i, j) = \begin{pmatrix} n 	imes Q \end{pmatrix}
\]

- **Combine Individual Scores:**
  - Measure importance of a node to the whole query set

\[
 r(Q, j) = \begin{pmatrix} n 	imes 1 \end{pmatrix}
\]

- **“Extract” the connection subgraph:**

\[
 \arg \max_H g(H)
\]
Goal:
- Calculate importance score $r(i,j) = r_{i,j}$
- for each node $j$ and each query node $i$

How to do that?
Proximity on Graphs

A - I - J - H - B

a.k.a.: Relevance, Closeness, ‘Similarity’...
Shortest path is not good:

No influence for degree-1 nodes (E, F)!
- known as ‘pizza delivery guy’ problem in undirected graph
- Multi-faceted relationships
**Good proximity measure?**

- Max-flow is not good:

![Graph 1](A -> D -> B)

- Does not punish long paths

![Graph 2](A -> D -> E -> B)
What is good notion of proximity?

- Multiple Connections
- Quality of connection
  - Direct & In-direct connections
  - Length, Degree, Weight...
Random Walk with Restarts
Random Walks with Restarts

Nearby nodes, higher scores
More red, more relevant

<table>
<thead>
<tr>
<th></th>
<th>Node 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 1</td>
<td>0.13</td>
</tr>
<tr>
<td>Node 2</td>
<td>0.10</td>
</tr>
<tr>
<td>Node 3</td>
<td>0.13</td>
</tr>
<tr>
<td>Node 4</td>
<td>0.22</td>
</tr>
<tr>
<td>Node 5</td>
<td>0.13</td>
</tr>
<tr>
<td>Node 6</td>
<td>0.05</td>
</tr>
<tr>
<td>Node 7</td>
<td>0.05</td>
</tr>
<tr>
<td>Node 8</td>
<td>0.08</td>
</tr>
<tr>
<td>Node 9</td>
<td>0.04</td>
</tr>
<tr>
<td>Node 10</td>
<td>0.03</td>
</tr>
<tr>
<td>Node 11</td>
<td>0.04</td>
</tr>
<tr>
<td>Node 12</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Ranking vector \( \vec{r}_4 \)
Why is RWR a good score?

\[ Q = (I - c\tilde{W})^{-1} = \]

\[ Q(i, j) \propto r_{i,j} \]

\[ \tilde{W} : \text{adjacency matrix.} \]
\[ c : \text{damping factor} \]

\[ Q = I + c\tilde{W} + c^2\tilde{W}^2 + c^3\tilde{W}^3 + \ldots \]

all paths from \( i \) to \( j \) with length 1
all paths from \( i \) to \( j \) with length 2
all paths from \( i \) to \( j \) with length 3
Computing the score

\[ \vec{r}_i = c \vec{W} \vec{r}_i + (1 - c) \vec{e}_i \]

- Ranking vector
- Transition matrix
- Restart prob.
- Starting vector

\[
\begin{pmatrix}
0.13 \\
0.10 \\
0.13 \\
0.22 \\
0.13 \\
0.05 \\
0.05 \\
0.08 \\
0.04 \\
0.03 \\
0.04 \\
0.02
\end{pmatrix} = 0.9 \times
\begin{pmatrix}
1/3 & 1/3 & 1/3 \\
1/3 & 1/3 & 1/3 \\
1/3 & 1/3 & 1/3 \\
1/3 & 1/2 & 1/2 & 1/4 \\
1/4 & 1/2 \\
1/4 & 1/2 \\
1/3 & 1/4 \\
1/4 & 1/3 \\
1/2 & 1/3 & 1/2 \\
1/4 & 1/3 & 1/2 \\
1/3 & 1/3 & 1/3
\end{pmatrix}
\begin{pmatrix}
0.13 \\
0.10 \\
0.13 \\
0.22 \\
0.13 \\
0.05 \\
0.05 \\
0.08 \\
0.04 \\
0.03 \\
0.04 \\
0.02
\end{pmatrix} + 0.1 \times
\begin{pmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{pmatrix}
\]
Computing the score

\[
\begin{pmatrix}
\vdots \\
\end{pmatrix}
\begin{pmatrix}
0.9 \\
\vdots \\
\end{pmatrix}
\begin{pmatrix}
0 \\
\vdots \\
\end{pmatrix}
= \begin{pmatrix}
\vdots \\
\end{pmatrix}
\begin{pmatrix}
1/3 & 1/3 & 1/3 \\
1/3 & 1/3 & 1/4 \\
1/3 & 1/3 & 1/3 \\
1/3 & 1/3 & 1/4 \\
1/3 & 1/2 & 1/2 & 1/4 \\
1/4 & 1/2 \\
1/4 & 1/2 \\
1/3 & 1/4 & 1/2 & 1/3 \\
1/4 & 1/3 \\
1/2 & 1/3 & 1/2 \\
1/4 & 1/3 & 1/3 \\
1/3 & 1/3 \\
\end{pmatrix}
\begin{pmatrix}
0 \\
\vdots \\
\end{pmatrix}

+ 0.1 \times
\begin{pmatrix}
1 \\
0 \\
0 \\
\vdots \\
0 \\
0 \\
0 \\
0 \\
0 \\
\end{pmatrix}
\]

Query

Starting vector

Ranking vector

Adjacency matrix

Ranking vector

2/4/2010

Jure Leskovec & Anand Rajaraman, Stanford CS345a: Data Mining
Computing $r_i$

$$r_i[t+1] = c \vec{W} r_i[t] + (1 - c) \vec{e}_i$$

No pre-computation / light storage

Slow on-line response \[O(mE)\]
Calculating Score $r(i,j)$
Combining the scores

- The RWR score $r(i,j)$
  - $i$ ... query node, $i \in Q$
  - $j$ ... node in the network

- Combine $r(i,j)$ to get the score $r(Q,j)$
  - 1) AND query: Meeting probability
    - prob. that $|Q|$ random particles coincide on node $j$
    
    $$r(Q,j) \triangleq r(Q, j, Q) = \prod_{i=1}^{Q} r(i, j)$$
Combining the scores: k-SoftAND

- OR query:
  - At least one particle arrives to node $j$:
    (i.e., node $j$ is important to at least 1 query node)
    
    $$ r(Q, j) \triangleq r(Q, j, 1) = 1 - \prod_{i=1}^{Q} (1 - r(i, j)) $$

- k-SoftAND query:
  - Node $j$ is important to at least $k$ query nodes
  - Can be computed recursively:
    
    $$ r(Q, j, k) = r(\hat{Q}, j, k-1) \cdot r(Q, j) + r(\hat{Q}, j, k) \cdot (1 - r(Q, j)) $$
Goal:

- Extract a sub-graph $S$ on $b$ nodes that maximizes the $\sum_{u \in S} r(Q, u)$

Idea:

- Iterate until budget is reached
  - Pick not-yet-selected node $j = \text{arg max}_j r(Q, j)$
  - Find good paths to all query nodes
  - Add the paths to $S$
AND vs. k-SoftAND

AND

2-SoftAND
1 - SoftAND = OR
Example:
Solving the random walk for each query separately is time consuming
- Not appropriate for real-time

Can we do better?

Idea 1) Pre-compute scores for each possible query node i
Pre-compute all the scores

\[ Q^{-1} = (I - c\tilde{W})^{-1} \]

Fast on-line response
Heavy pre-computation/storage cost

\[ O(n^3) \quad O(n^2) \]
Idea 2: Approximation

Find Communities

Fix the remaining

Combine

[Tong-Faloutsos-Pan, '06]
Step 1: Partition the graph

\[ \tilde{\mathbf{W}} = \tilde{\mathbf{W}}_1 + \tilde{\mathbf{W}}_2 \]

Within-partition links  
Cross-partition links
Step 2: Make $W_1$ block-diagonal

\[
\tilde{W}_1 = \begin{pmatrix}
\tilde{W}_{11} & \cdots & \tilde{W}_{12} \\
\cdots & \ddots & \cdots \\
\tilde{W}_{13} & \cdots & \tilde{W}_{33}
\end{pmatrix}
\]

\[
Q_{1,i}^{-1} = (I - \tilde{W}_{1,i})^{-1}
\]
Step 3: Cross community links

\[ \tilde{W}_2 \approx USV = \]

\[ |S| \ll |\tilde{W}_2| \]
Final algorithm: Offline stage

- Offline stage:

\[ Q_1^{-1} = \begin{pmatrix} Q_{1,1} & \cdot & \cdots & \cdot \\ \cdot & Q_{1,2} & \cdot & \cdots & \cdot \\ \cdot & \cdot & \ddots & \cdot & \cdots \\ \cdot & \cdot & \cdot & Q_{1,k} & \cdot \\ \cdot & \cdot & \cdot & \cdot & Q_{1,n} \end{pmatrix} \]

\[ \tilde{\Lambda} = (S^{-1} - cVQ_1^{-1}U)^{-1} \]

\[ \tilde{W} = \tilde{W}_1 + \tilde{W}_2 \]
One more thing:

- Sherman-Morrison-Woodbury matrix identity:
  - Inverse of a rank-$k$ correction of matrix $X$ can be computed by doing a rank-$k$ correction to $X^{-1}$:
    \[(X - USV)^{-1} = X^{-1} + X^{-1} U \tilde{\Lambda} V X^{-1}\]
    where $\tilde{\Lambda} = (S^{-1} - VX^{-1} U)^{-1}$

- In our case:
  \[Q^{-1} = c(I-W)^{-1} = c(I-W_1-W_2)^{-1} =\]
  \[Q^{-1} \approx Q_1^{-1} + cQ_1^{-1} U \tilde{\Lambda} V Q_1^{-1}\]
Final algorithm: Online stage

- Online stage:
  - Compute column $i$ of:
    \[
    Q^{-1} \approx Q_1^{-1} + cQ_1^{-1}U\tilde{\Lambda}VQ_1^{-1}
    \]
  - Using:
    \[
    \vec{r}_i = (1 - c)(Q_1^{-1}\vec{e}_i + cQ_1^{-1}U\tilde{\Lambda}VQ_1^{-1}\vec{e}_i)
    \]
Acknowledgment

- Most slides are borrowed from the tutorial by Hanghang Tong and Christos Faloutsos from CMU