Sampling 2: Random Walks

Lecture 20

CSCI 4974/6971

10 Nov 2016

Today's Biz

1. **Reminders**
2. Review
3. Random Walks

Reminders

- Assignment 5: due date November 22nd
  - Distributed triangle counting
- Assignment 6: due date TBD (early December)
- Tentative: **No class November 14 and/or 17**
- Final Project Presentation: December 8th
- Project Report: December 11th
- Office hours: Tuesday & Wednesday 14:00-16:00 Lally 317
  - Or email me for other availability

Today's Biz

1. Reminders
2. **Review**
3. Random Walks

Quick Review

Graph Sampling:

- Vertex sampling methods
  - Uniform random
  - Degree-biased
  - Centrality-biased (PageRank)
- Edge sampling methods
  - Uniform random
  - Vertex-edge (select vertex, then random edge)
  - Induced edge (select edge, include all edges of attached vertices)

Today's Biz

1. Reminders
2. Review
3. **Random Walks**

Random Walks on Graphs - Classification, Clustering, and Ranking

Ahmed Hassan, University of Michigan

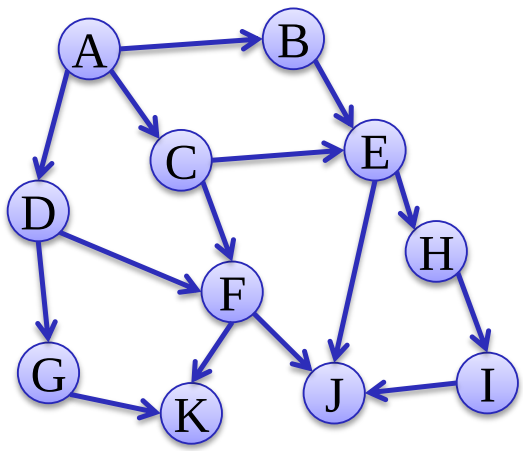
Random Walks on Graphs
Classification, Clustering, and Ranking

Ahmed Hassan
Ph.D. Candidate
Computer Science and Engineering Dept.
The University of Michigan Ann Arbor
hassanam@umich.edu

Random Walks on Graphs
Why Graphs?

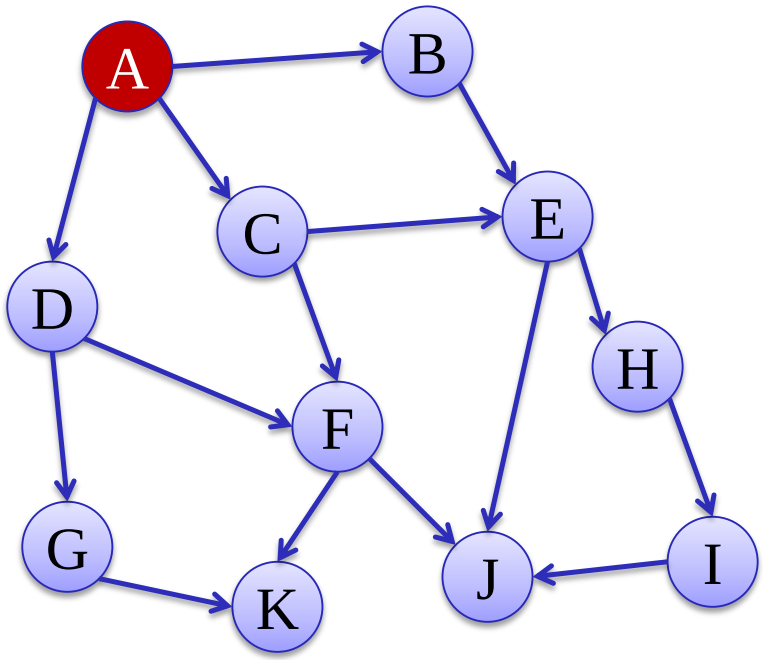
The underlying data is naturally a graph

- Papers linked by citation
- Authors linked by co-authorship
- Bipartite graph of customers and products
- Web-graph
- Friendship networks: who knows whom



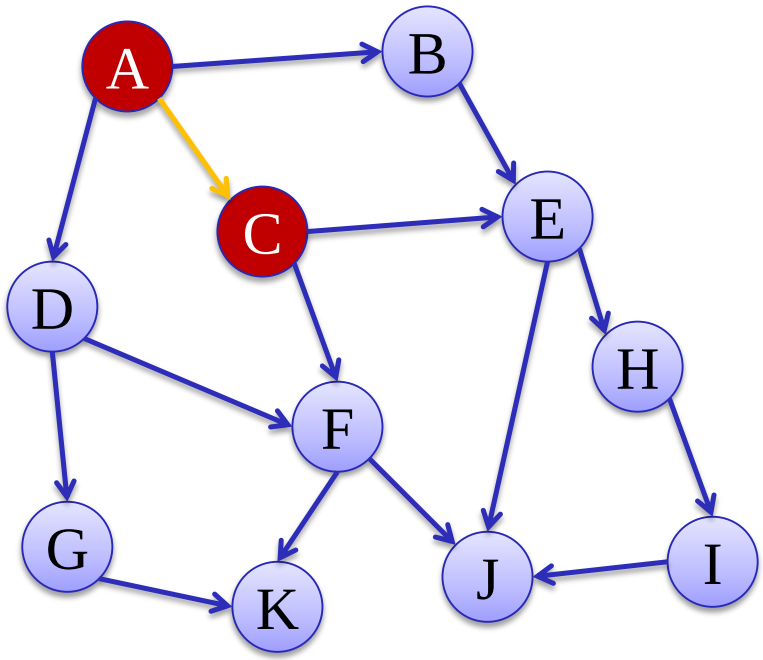
What is a Random Walk

- Given a graph and a starting node, we select a neighbor of it at random, and move to this neighbor



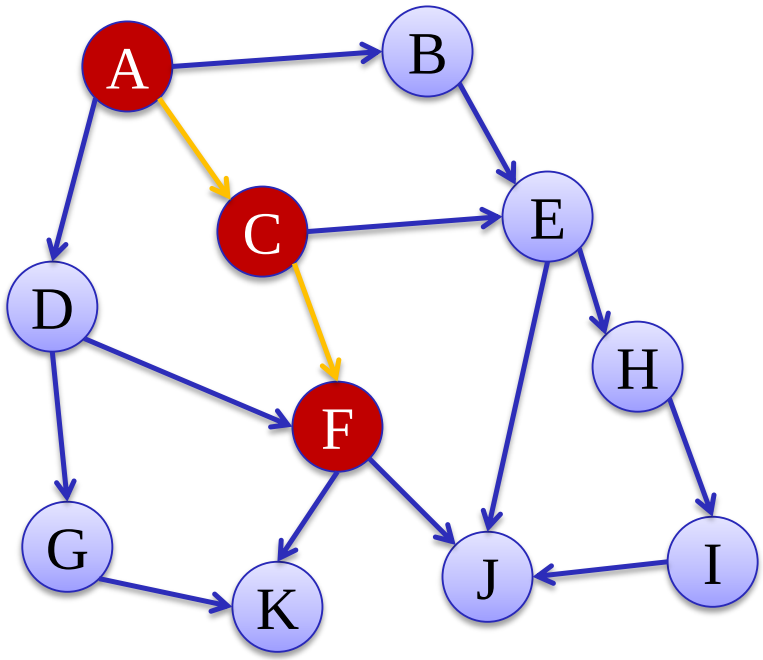
What is a Random Walk

- We select a neighbor of it at random, and move to this neighbor



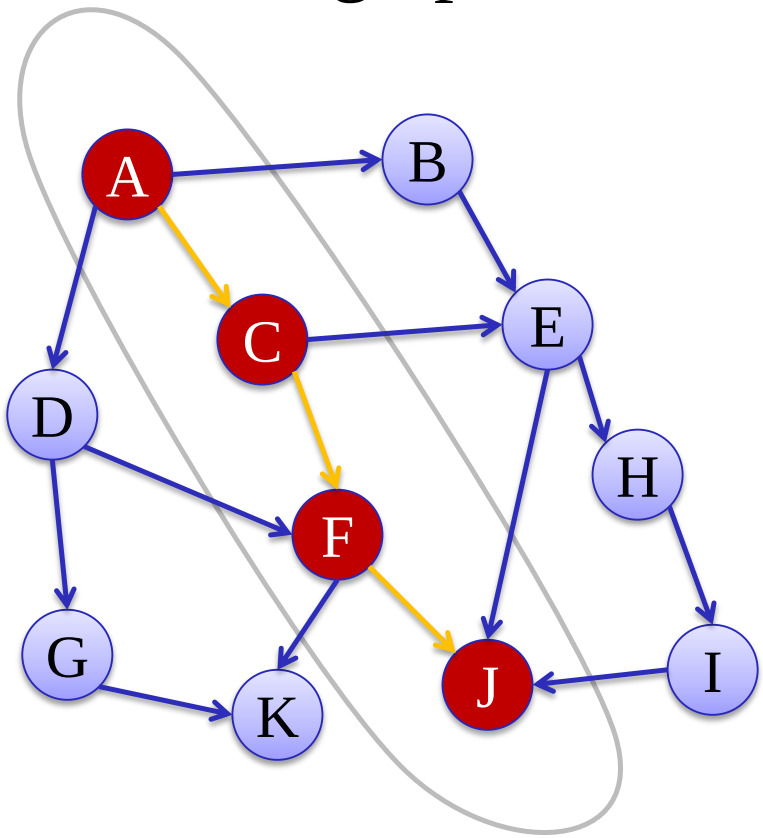
What is a Random Walk

- Then we select a neighbor of this node and move to it, and so on.



What is a Random Walk

•The (random) sequence of nodes selected this way is a random walk on the graph

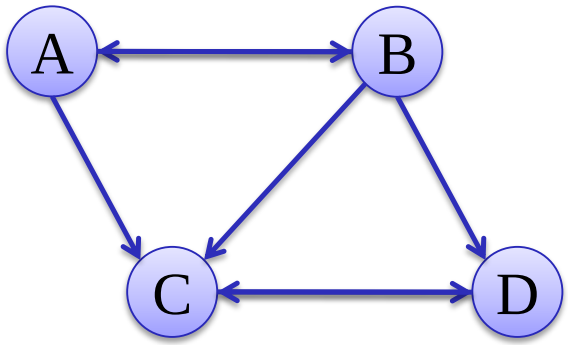


Adjacency Matrix vs. Transition Matrix

- A transition matrix is a stochastic matrix where each element a_{ij} represents the probability of moving from i to j, with each row summing to 1.

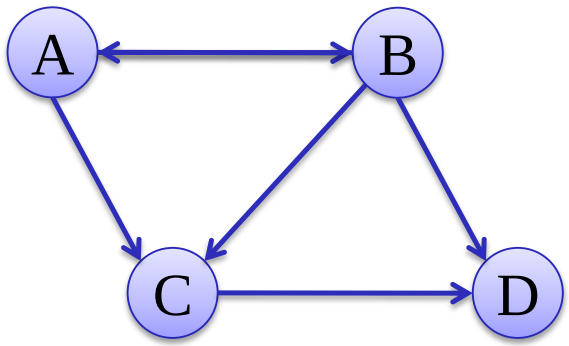
Adjacency Matrix

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



Transition Matrix

$$\begin{bmatrix} 0 & 1/2 & 1/2 & 0 \\ 1/3 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



Markov chains

•A Markov chain describes a discrete time stochastic process over a set of states

$$S = \{s_1, s_2, \ldots s_n\}$$

according to a transition probability matrix

$$P = \{P_{ij}\}$$

P_{ij} = probability of moving to state j when at state i

• Markov Chains are memoryless: The next state of the chain depends only at the current state

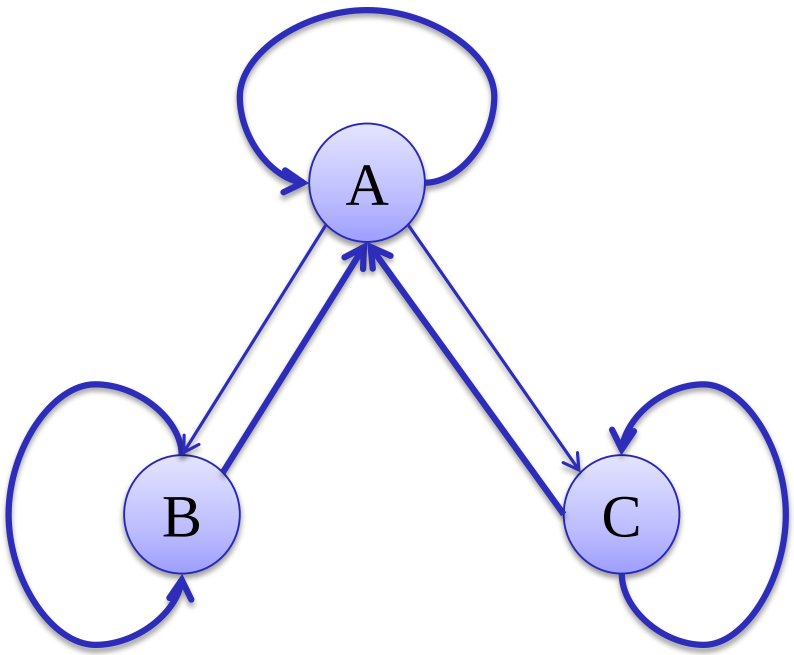
Random Walks & Markov chains

• Random walks on graphs correspond to Markov Chains

- The set of states S is the set of nodes of the graph
- The transition probability matrix is the probability that we follow an edge from one node to another

Random Walks & Markov chains

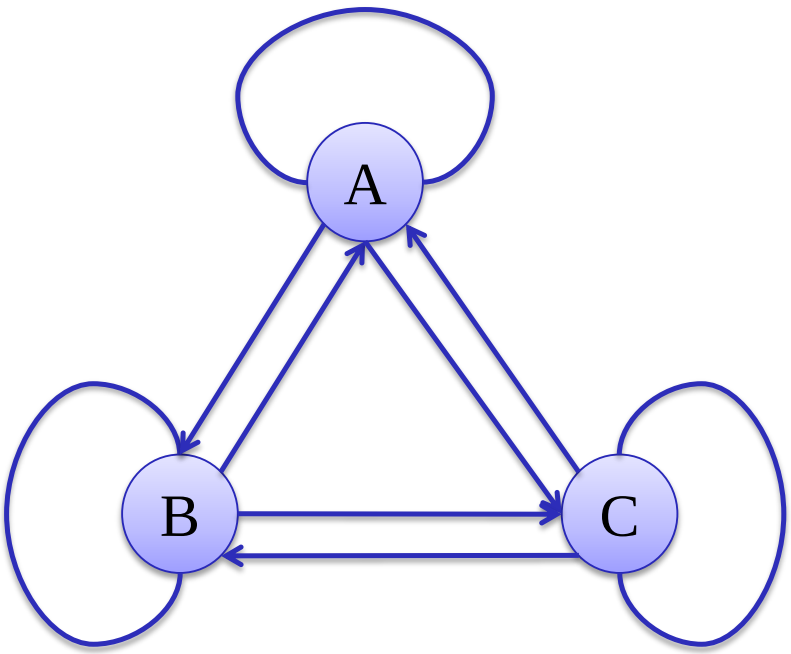
P^1_{ij} is the probability that the random walk starting in node *i*, will be in node *j* after 1 step



$$p^1 = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \end{bmatrix}$$

Random Walks & Markov chains

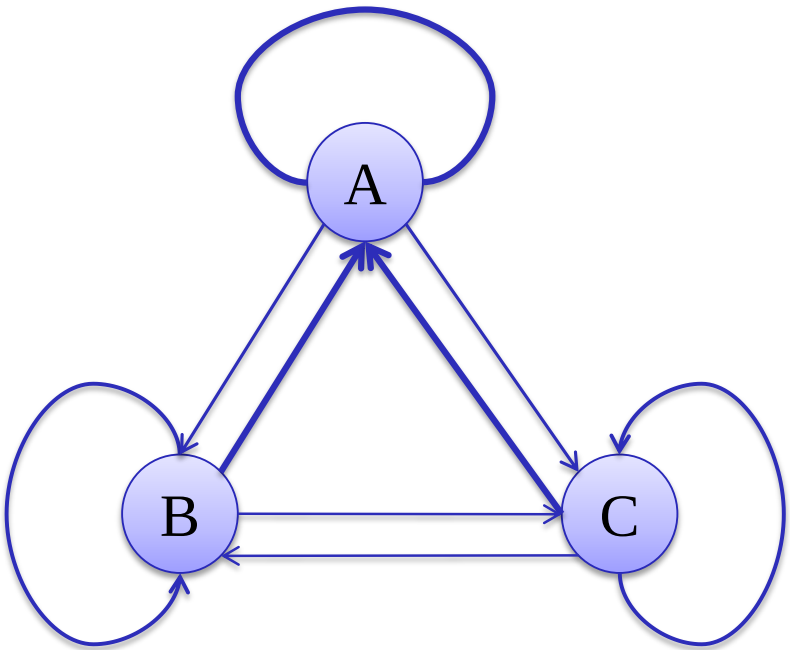
P^2_{ij} is the probability that the random walk starting in node *i*, will be in node *j* after 2 steps



$$p^2 = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.5 & 0.375 & 0.125 \\ 0.25 & 0.125 & 0.375 \end{bmatrix}$$

Random Walks & Markov chains

P^3_{ij} is the probability that the random walk starting in node *i*, will be in node *j* after 2 steps



$$p^3 = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.5 & 0.3125 & 0.1875 \\ 0.5 & 0.1875 & 0.3125 \end{bmatrix}$$

Stationary Distribution

- $x_t(i)$ = probability that the surfer is at node i at time t

- $x_{t+1}(j) = \sum_i x_t(i) \cdot P_{ij}$
- $x_{t+1} = x_t P = x_{t-1} P P = x_0 P^t$

- What happens when the surfer keeps walking for a long time?
  - We get a stationary distribution

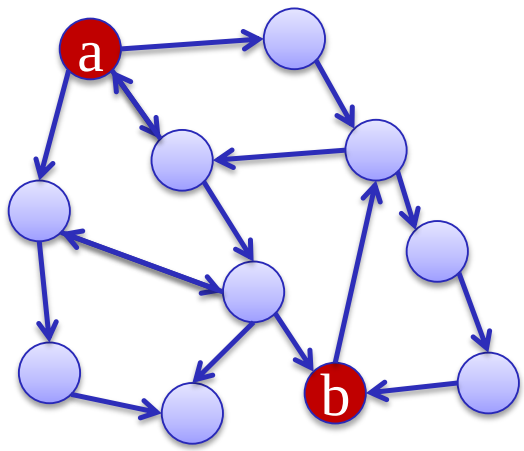
Stationary Distribution

- The stationary distribution at a node is related to the amount of time a random walker spends visiting that node
- When the surfer keeps walking for a long time, the distribution does not change any more: $x_{t+1}(i) = x_t(i)$
- For "well-behaved" graphs this does not depend on the start distribution

Hitting Time

- How long does it take to hit node *b* in a random walk starting at node *a* ?

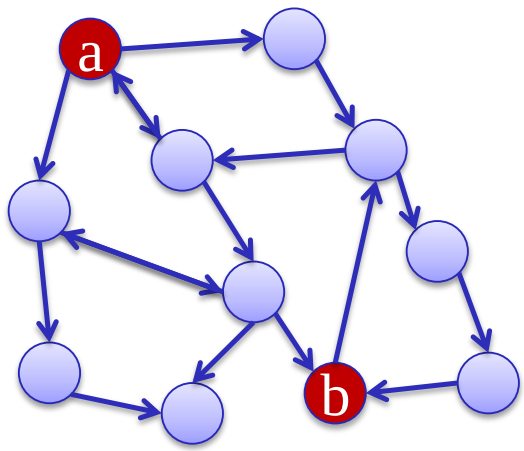
- Hitting time from node i to node j
  - Expected number of hops to hit node *j* starting at node *i*.
  - Not symmetric
  - $h(i,j) = 1 + \Sigma_{k \in adj(i)} P(i,k)\, h(k,j)$



Commute Time

- How long does it take to hit node *b* in a random walk starting at node *a* and come back to *a*?

- Commute time from node *i* to node *j*
  - Expected number of hops to hit node *j* starting at node *i* and come back to *i*.
  - Symmetric
  - $c(i,j) = h(i,j) + h(j,i)$



Ranking using Random Walks

Ranking Web Pages

- Problem Defenition:
  - Given:
    - a search query, and
    - A large number of web pages relevant to that query
  - Rank web pages based on the hyperlink structure

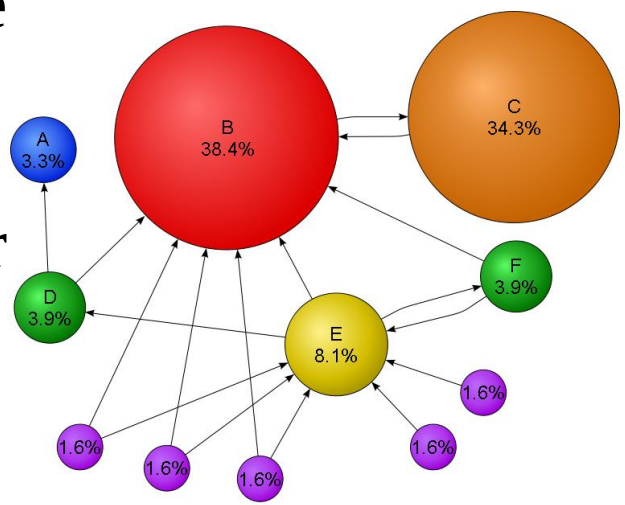
- Algorithm
  - Pagerank (Page et al. 1999)
    - *PageRank Citation Ranking: Bringing Order to the Web*
  - HITS (Kleinberg 1998)
    - *Authoritative sources in a hyperlinked environment*

Pagerank (Page et al. 1999)

- Simulate a random surfer on the Web graph
- The surfer jumps to an arbitrary page with non-zero probability
- A webpage is important if other important pages point to it

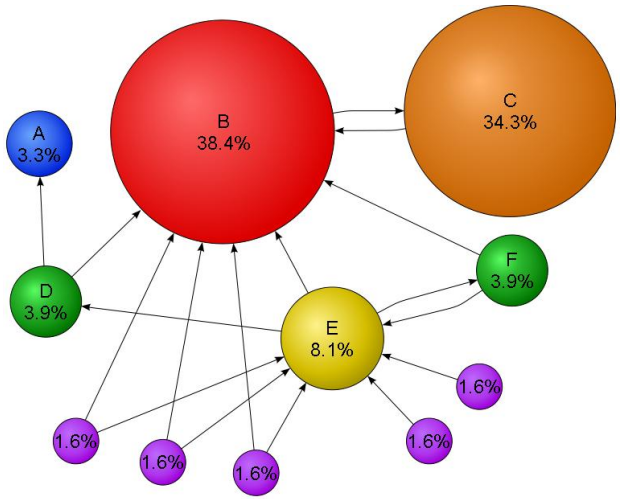
$$s(i) = \sum_{j \in adj(i)} \frac{s(j)}{\deg(j)}$$

- s works out to be the stationary distribution of the random walk on the Web graph

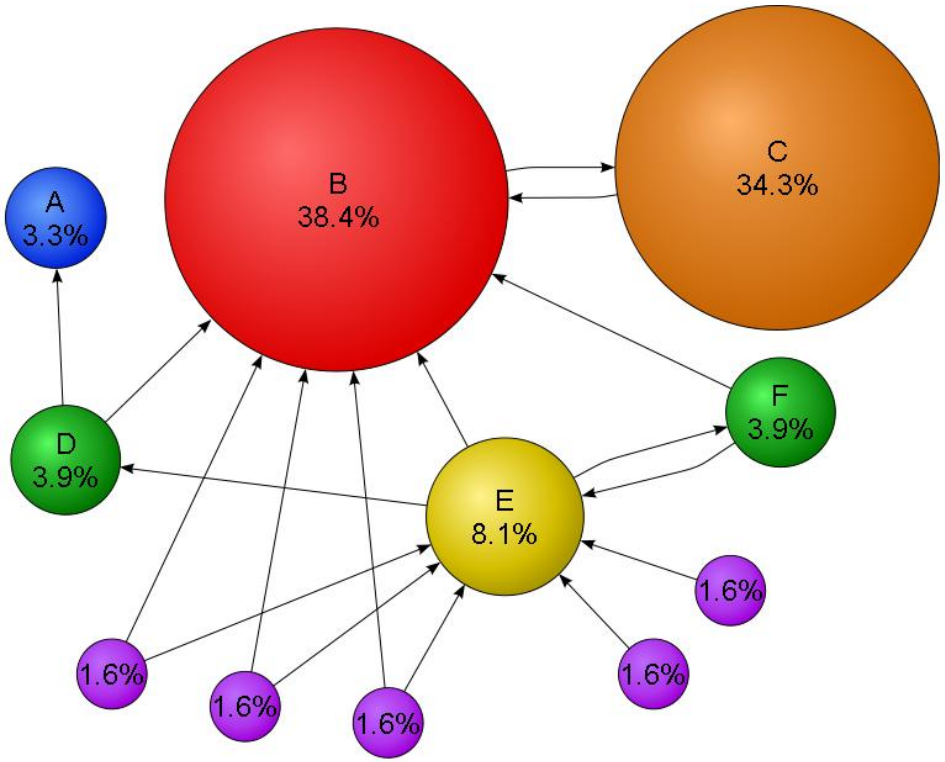


Power Iteration

- Power iteration is an algorithm for computing the stationary distribution
  - Start with any distribution x_0
  - Let $x_{t+1} = x_t P$
  - Iterate
  - Stop when x_{t+1} and x_t are almost the same



Pagerank Demo



Ranking Sentences for Extractive Summarization

- Problem Defenition:
  - Given:
    - document
    - A similarity measure between sentences in the document
  - Rank sentences based on the similarity structure

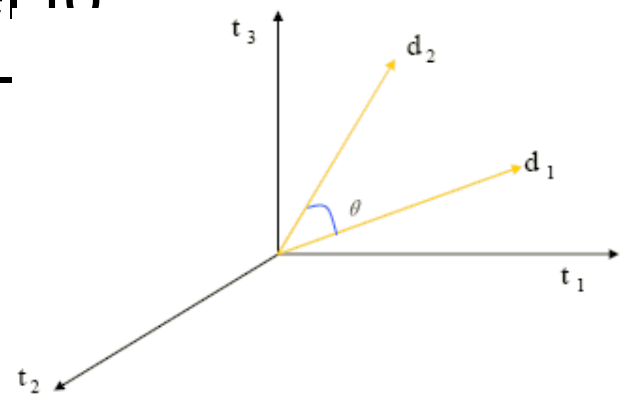
- Algorithm
  - Lexrank (Erkan et al. 2004)
    - *Graph-based centrality as salience in text summarization.*

Lexrank (Erkan et al. 2004)

- Perform a random walk on a sentence similarity graph
- Rank sentences according to node probabilities in the stationary distribution

Graph Construction

- They use the bag-of-words model to represent each sentence as an n-dimensional vector



- tf-idf representation

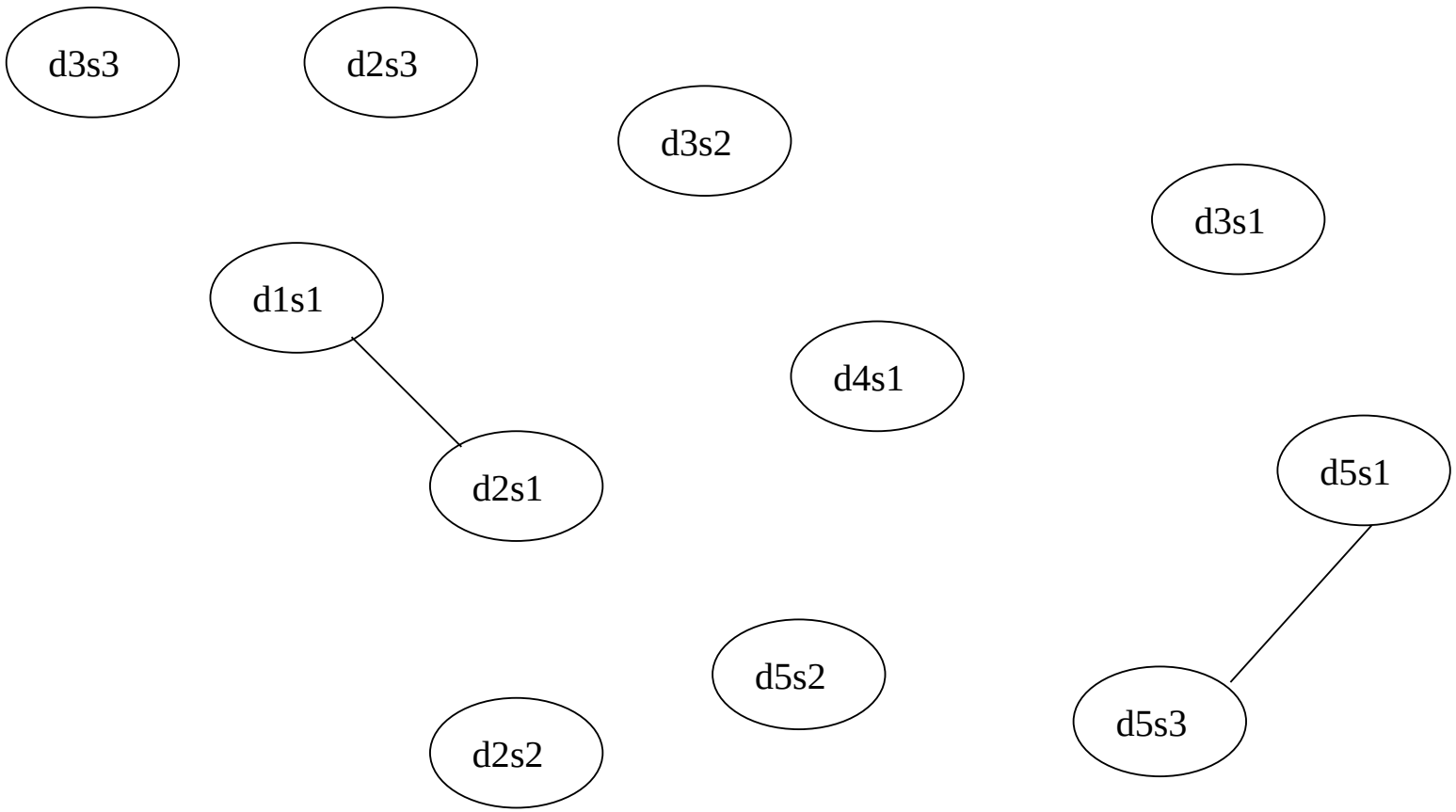
- The similarity between two sentences is then defined by the cosine between two corresponding vectors

Cosine Similarity

	1	2	3	4	5	6	7	8	9	10	11
1	1.00	0.45	0.02	0.17	0.03	0.22	0.03	0.28	0.06	0.06	0.00
2	0.45	1.00	0.16	0.27	0.03	0.19	0.03	0.21	0.03	0.15	0.00
3	0.02	0.16	1.00	0.03	0.00	0.01	0.03	0.04	0.00	0.01	0.00
4	0.17	0.27	0.03	1.00	0.01	0.16	0.28	0.17	0.00	0.09	0.01
5	0.03	0.03	0.00	0.01	1.00	0.29	0.05	0.15	0.20	0.04	0.18
6	0.22	0.19	0.01	0.16	0.29	1.00	0.05	0.29	0.04	0.20	0.03
7	0.03	0.03	0.03	0.28	0.05	0.05	1.00	0.06	0.00	0.00	0.01
8	0.28	0.21	0.04	0.17	0.15	0.29	0.06	1.00	0.25	0.20	0.17
9	0.06	0.03	0.00	0.00	0.20	0.04	0.00	0.25	1.00	0.26	0.38
10	0.06	0.15	0.01	0.09	0.04	0.20	0.00	0.20	0.26	1.00	0.12
11	0.00	0.00	0.00	0.01	0.18	0.03	0.01	0.17	0.38	0.12	1.00

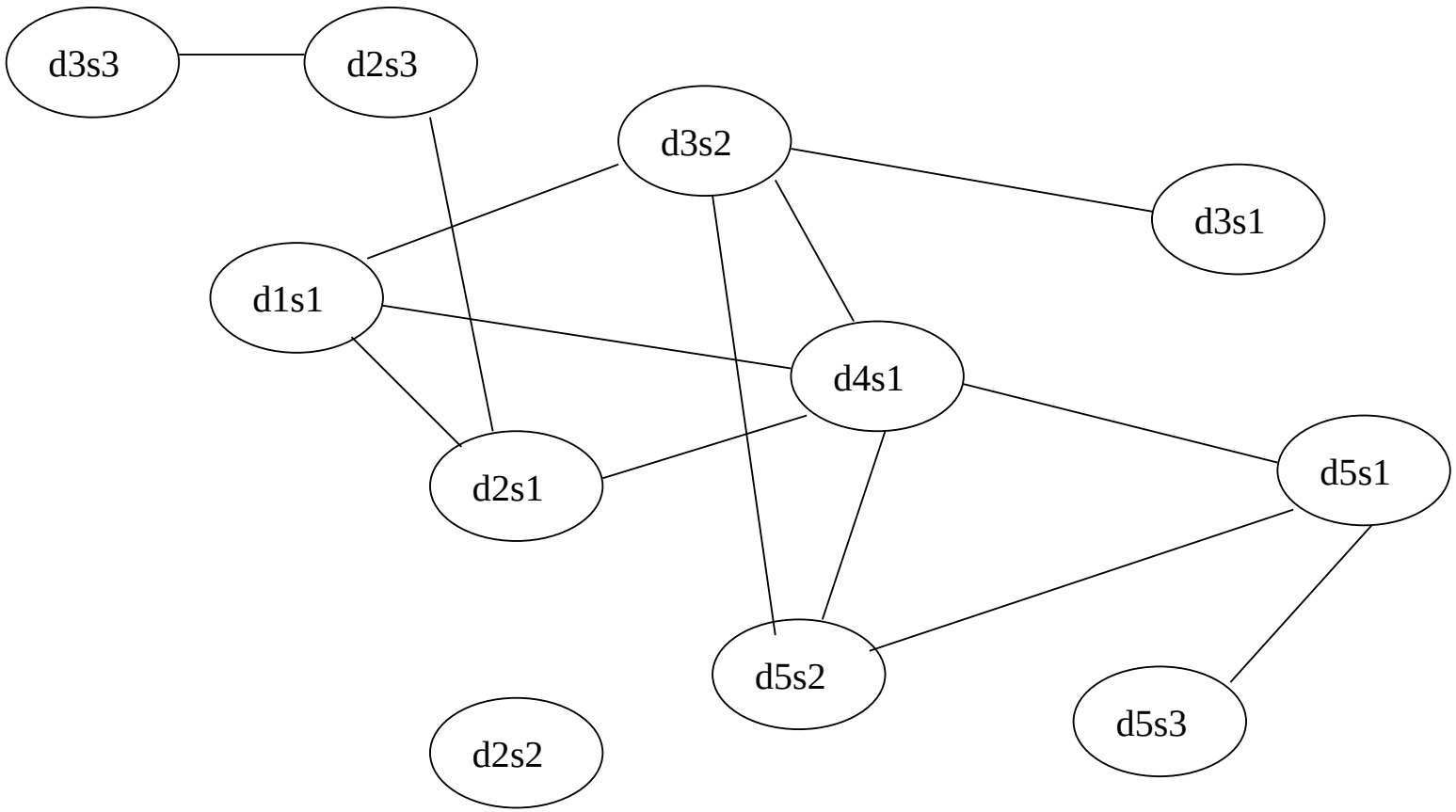
Slide from “Random walks, eigenvectors, and their applications to Information Retrieval, Natural Language Processing, and Machine Learning”. Dragomir Radev.

Lexical centrality (t=0.3)



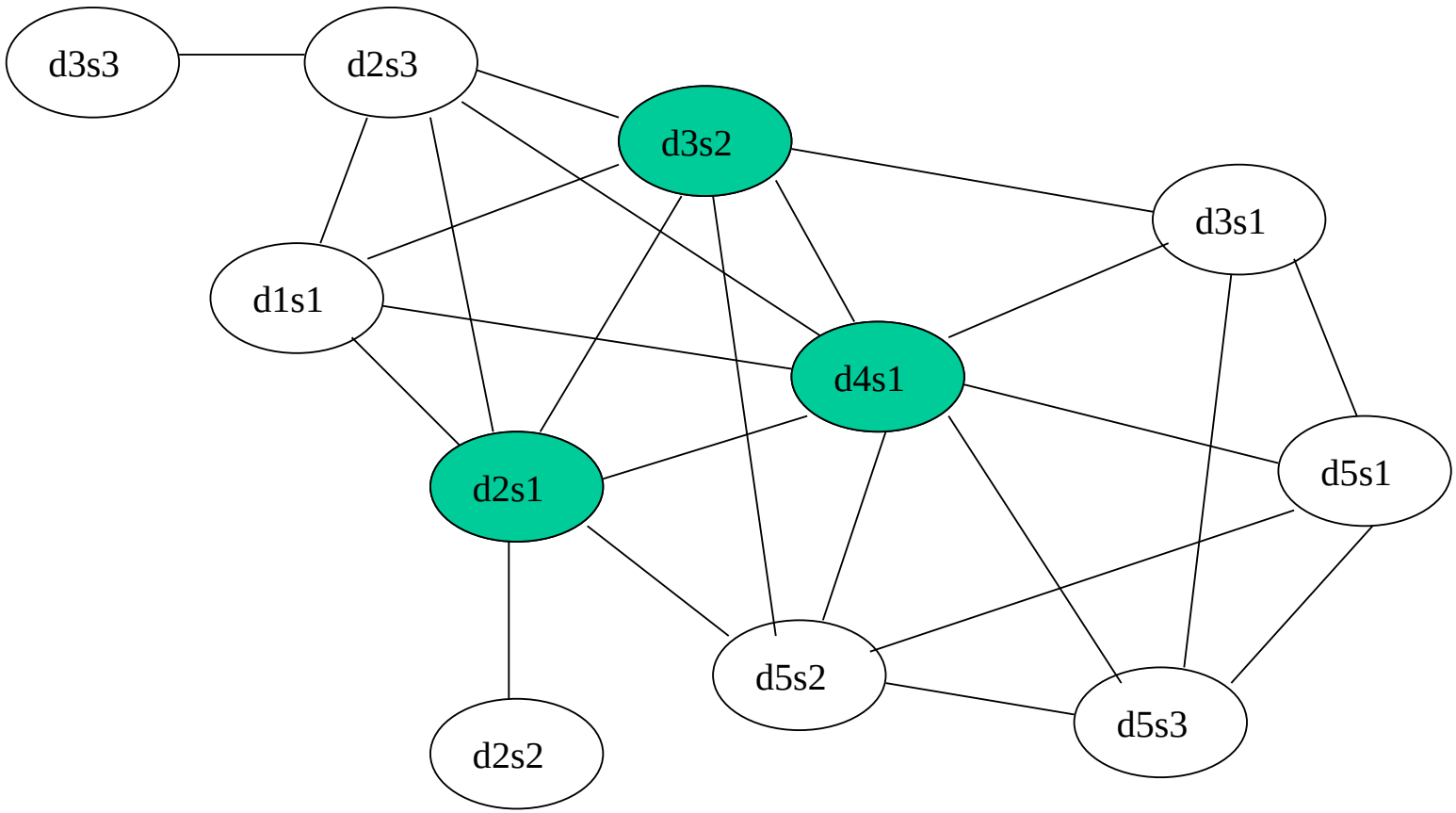
Slide from “Random walks, eigenvectors, and their applications to Information Retrieval, Natural Language Processing, and Machine Learning”. Dragomir Radev.

Lexical centrality (t=0.2)



Slide from “Random walks, eigenvectors, and their applications to Information Retrieval, Natural Language Processing, and Machine Learning”. Dragomir Radev.

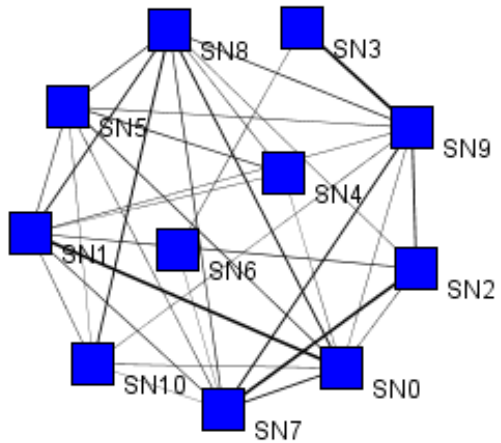
Lexical centrality (t=0.1)



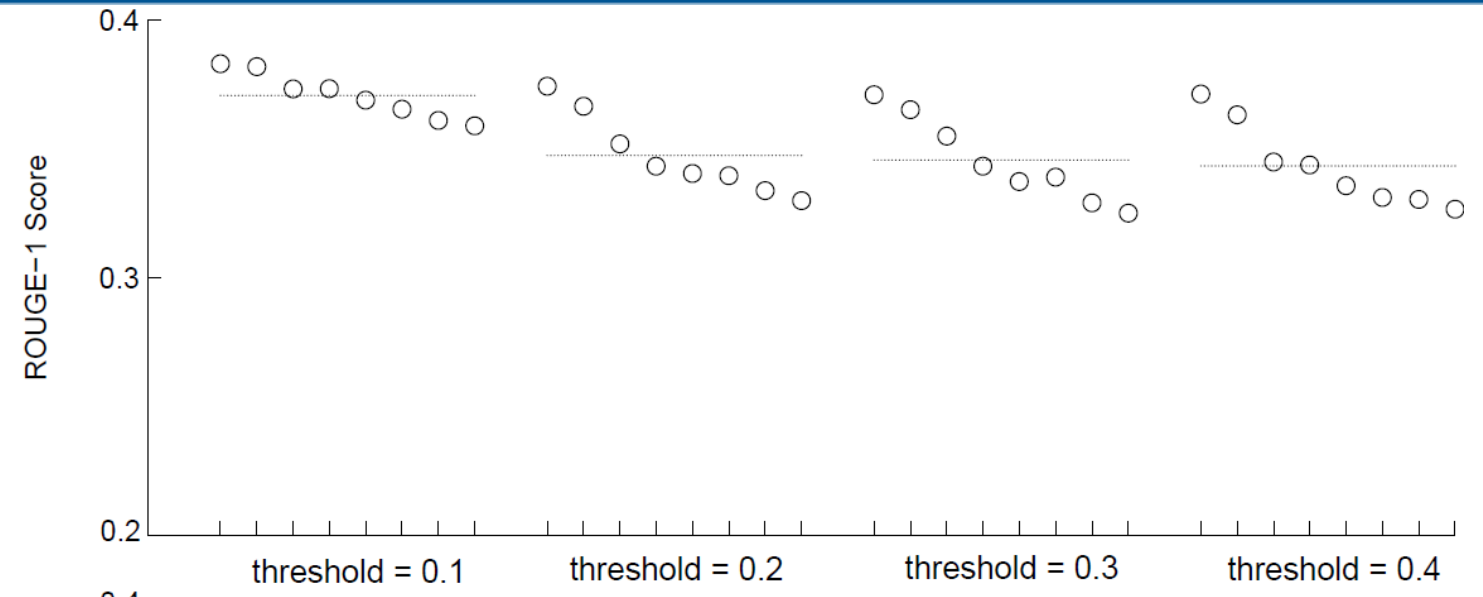
Slide from “Random walks, eigenvectors, and their applications to Information Retrieval, Natural Language Processing, and Machine Learning”. Dragomir Radev.

Sentence Ranking

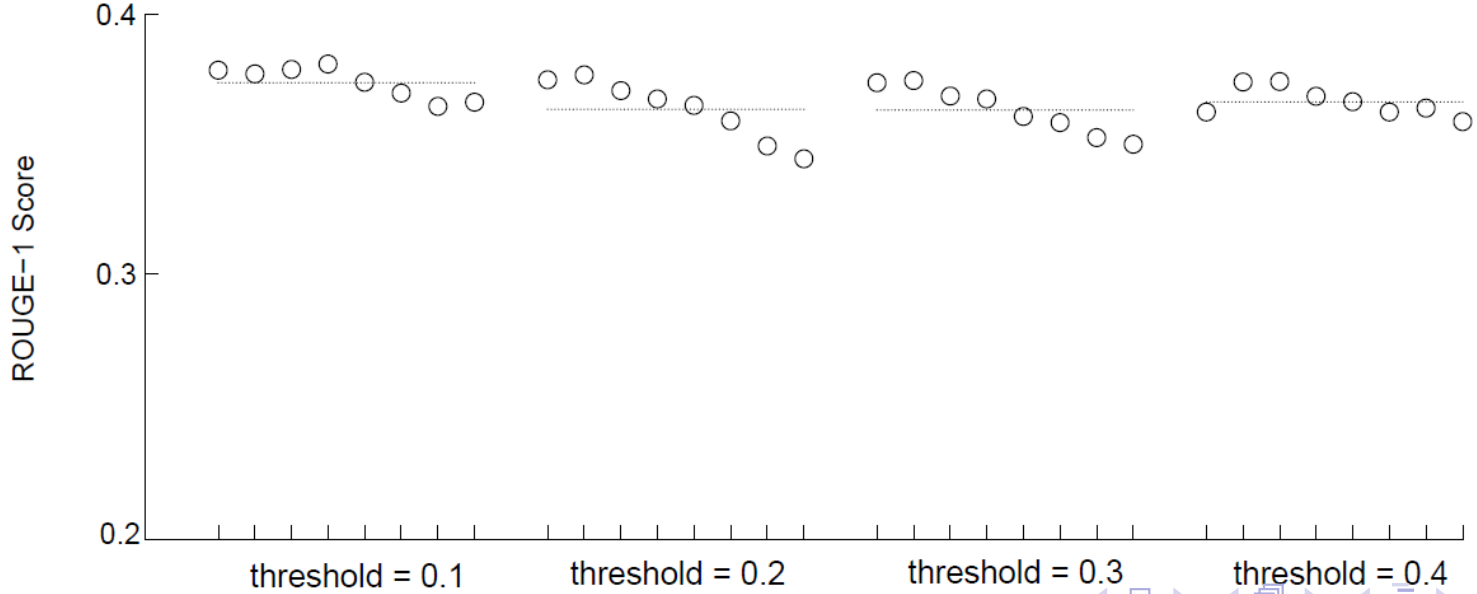
- Simulate a random surfer on the sentence similarity graph
- A sentence is important if other important sentences are similar to it
- Rank sentences according to the stationary distribution of the random walk on the sentence graph



Results

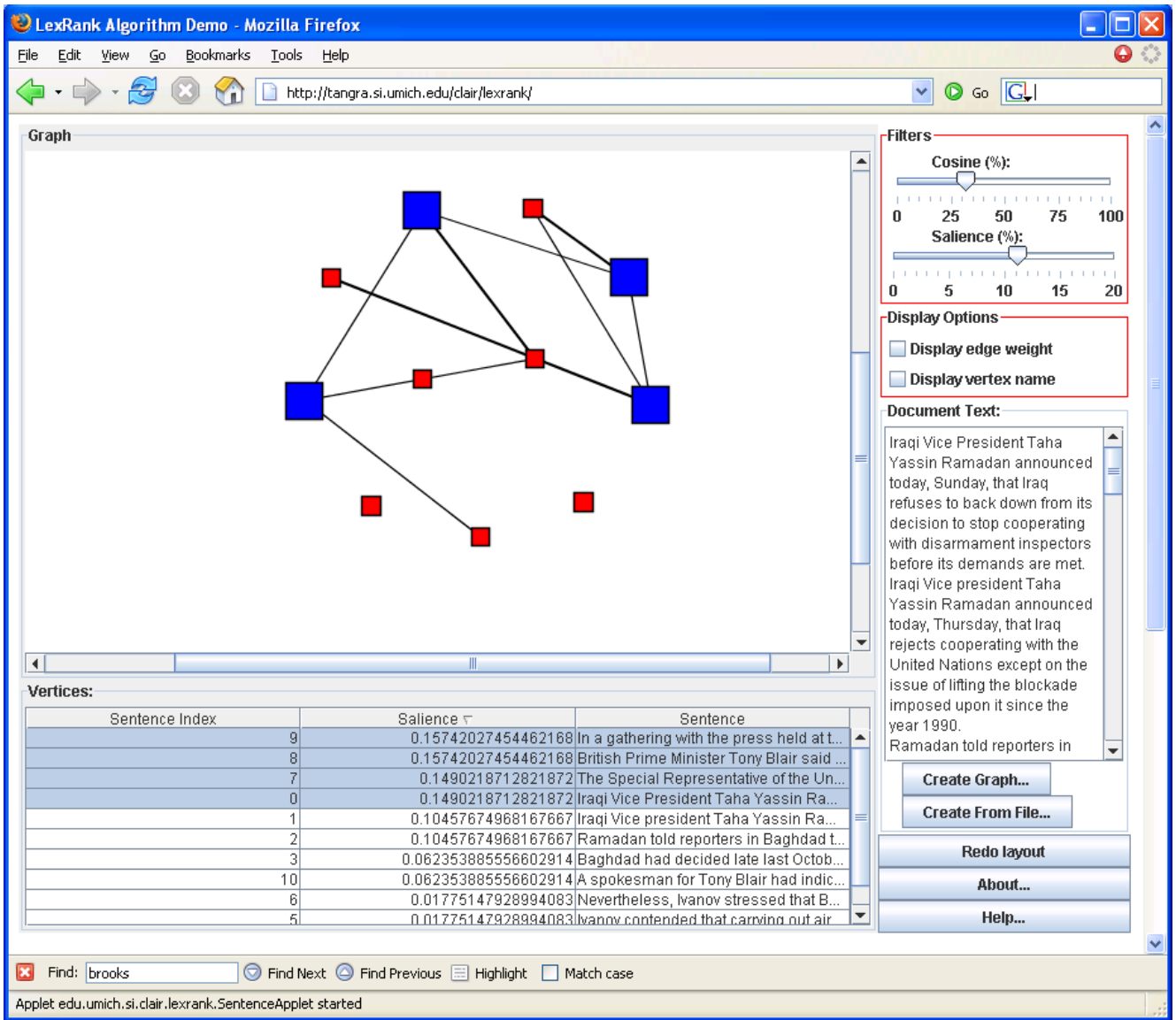


Degree Centrality
DUC 2004



Lexrank
DUC 2004

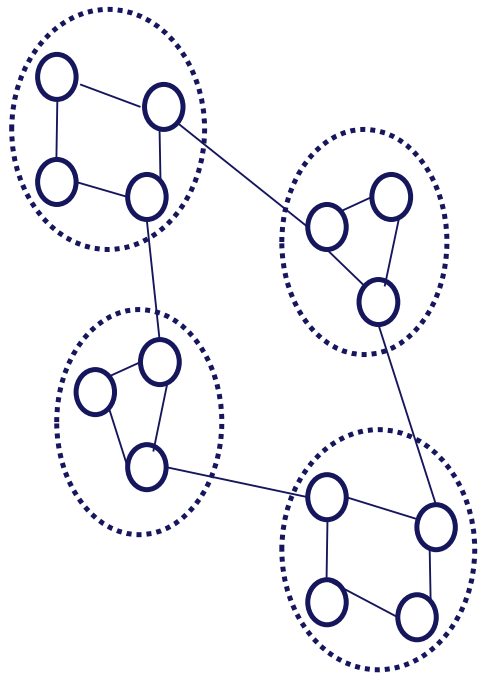
Lexrank Demo



Graph Clustering using Random Walks

Graph Clustering

- Problem Defenition:
  - Given:
    - a graph
  - Assign nodes to subsets (clusters) such that intra-cluster links are minimized and inter-cluster links are maximized
- Algorithm
  - (Yen et al. 2005)
    - *Clustering using a random walk based distance measure*
  - *MCL (van Dongen 2000)*
    - *A cluster algorithm for graphs*



Clustering using a random-walk based distance measure (Yen et al. 2005)

- The Euclidean Commute Time distance (ECT)
- A random walk based distance measure between nodes in a graph
- Clustering using K-means on the new distance measure

Euclidean Commute Time distance

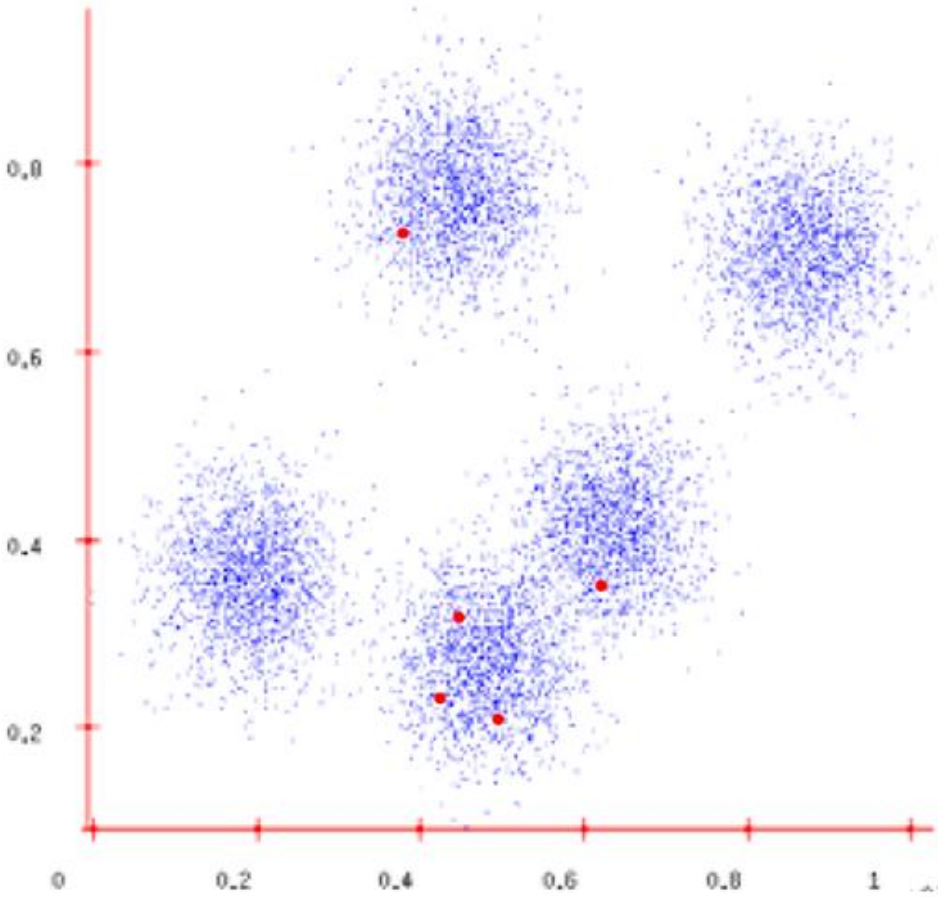
- Average hitting time $m(k|i)$: average number of steps a random walker starting at node *i* will take to reach node *k*

- Average commute time $c(k|i)$: average number of steps a random walker starting at node *i* will take to reach node k and go back to *i*

- Use the average commute time as a distance measure between any nodes in the graph

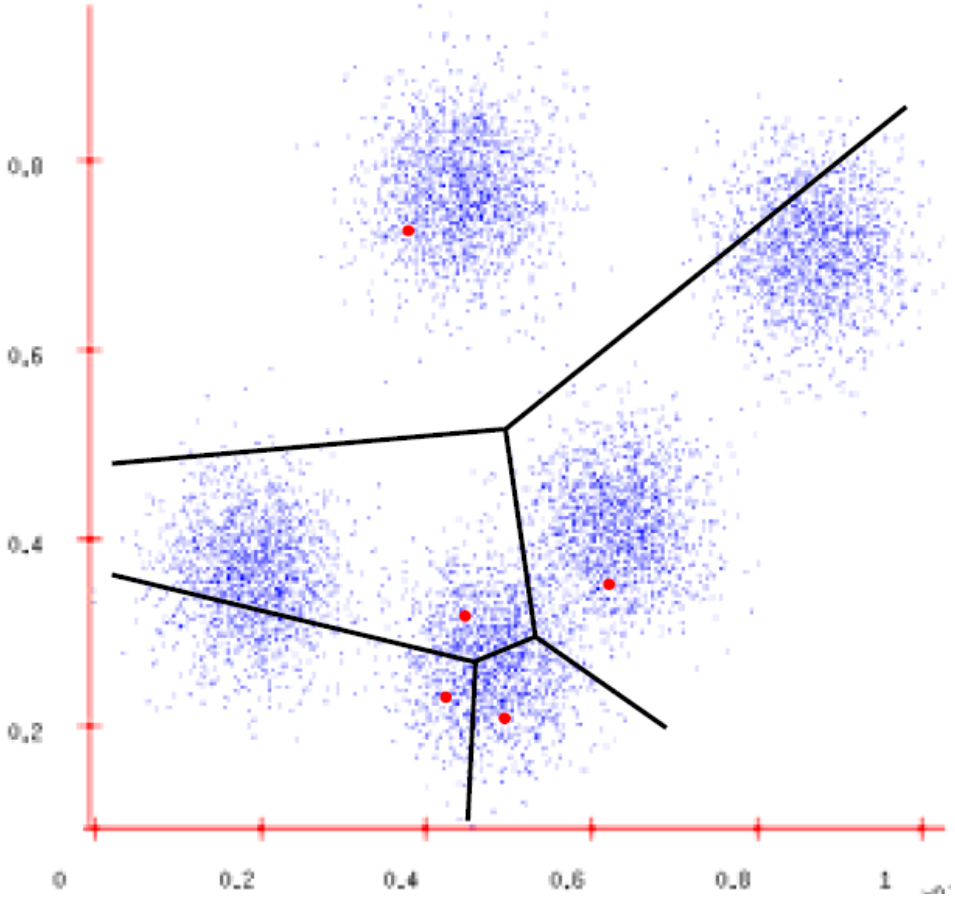
Kmeans + ECT

- Randomly guess *k* cluster prototypes



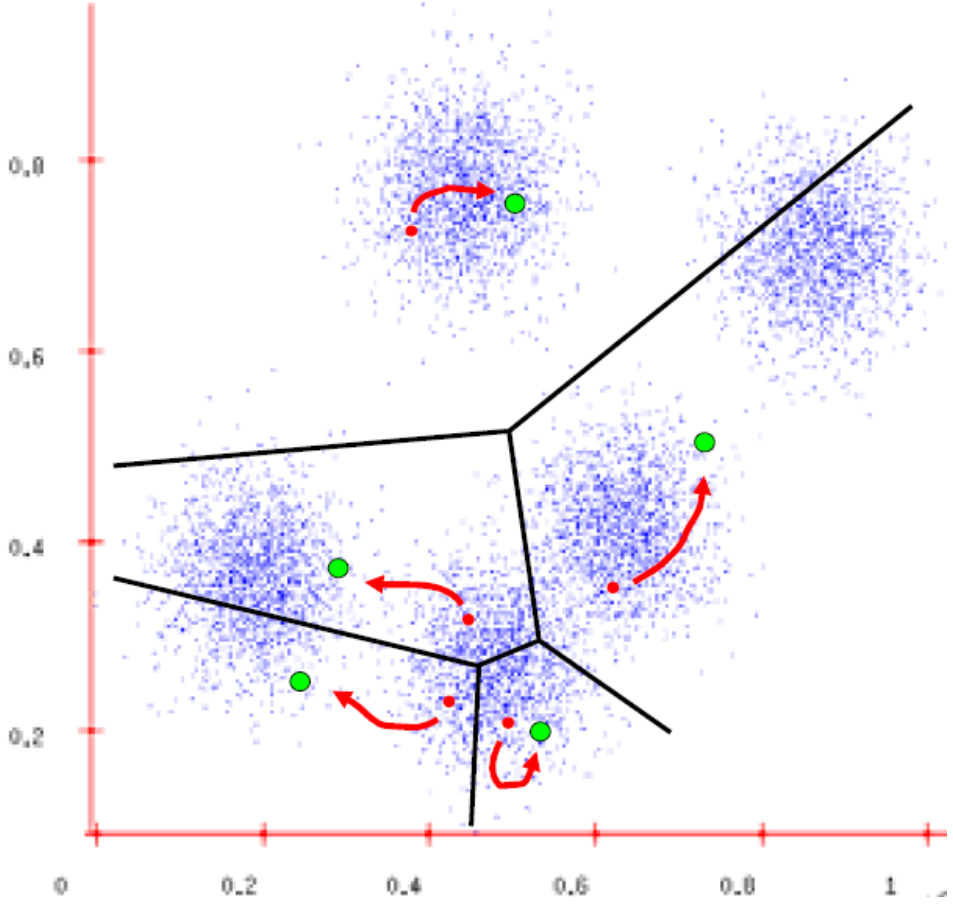
Kmeans + ECT

- Find the prototype with the least ECT distance to each data point and assign it to that cluster



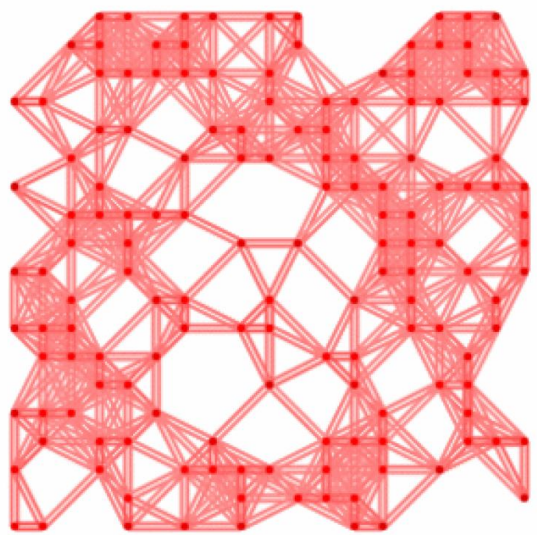
Kmeans + ECT

- Calculate new cluster prototypes (minimize the within cluster variance w.r.t. ECT) and repeat …..



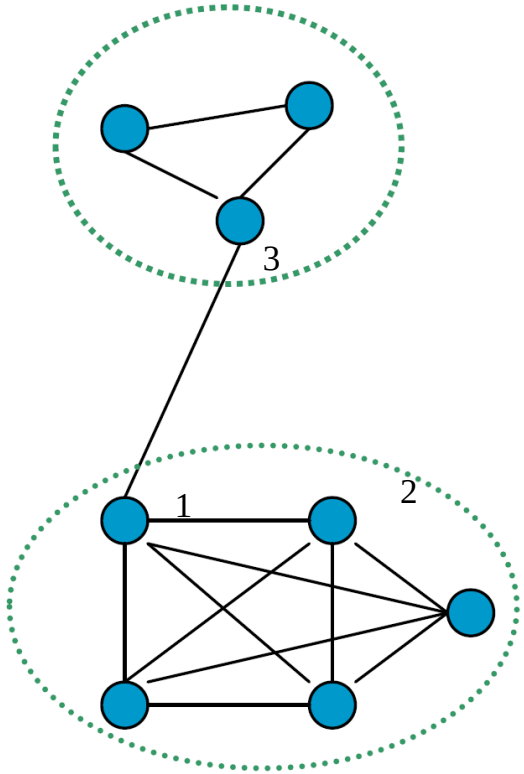
MCL (van Dongen 2000)

- Many links within cluster and fewer links between clusters
- A random walk starting at a node is more likely to stay within a cluster than travel between clusters
- This is the key idea behind MCL



MCL (van Dongen 2000)

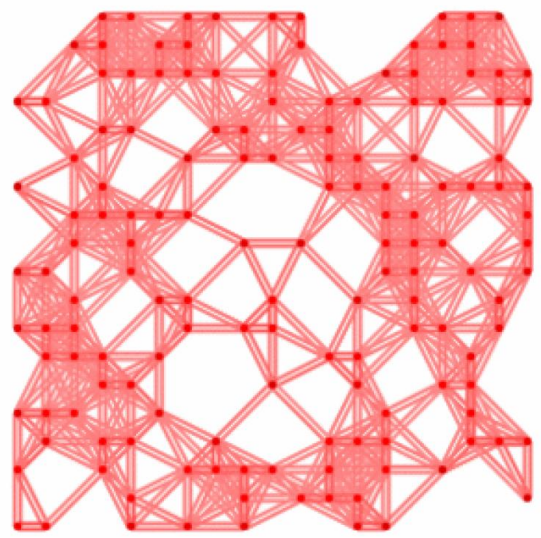
Node	Prob. Next Step within cluster	Prob. Next Step between clusters
1	80%	20%
2	100%	0%
3	67%	33%



Random walks on a graph reveal where the flow tends gather in a graph.

Stochastic Flow

- Flow is easier within clusters than across clusters
- To simulate flow:
  - Raise the transition matrix to integer powers (In each step of the random walk, we do one matrix multiplication)
- During the earlier powers of the transition matrix, edge weights will be higher in links within clusters
- However, in the long run this effect disappears



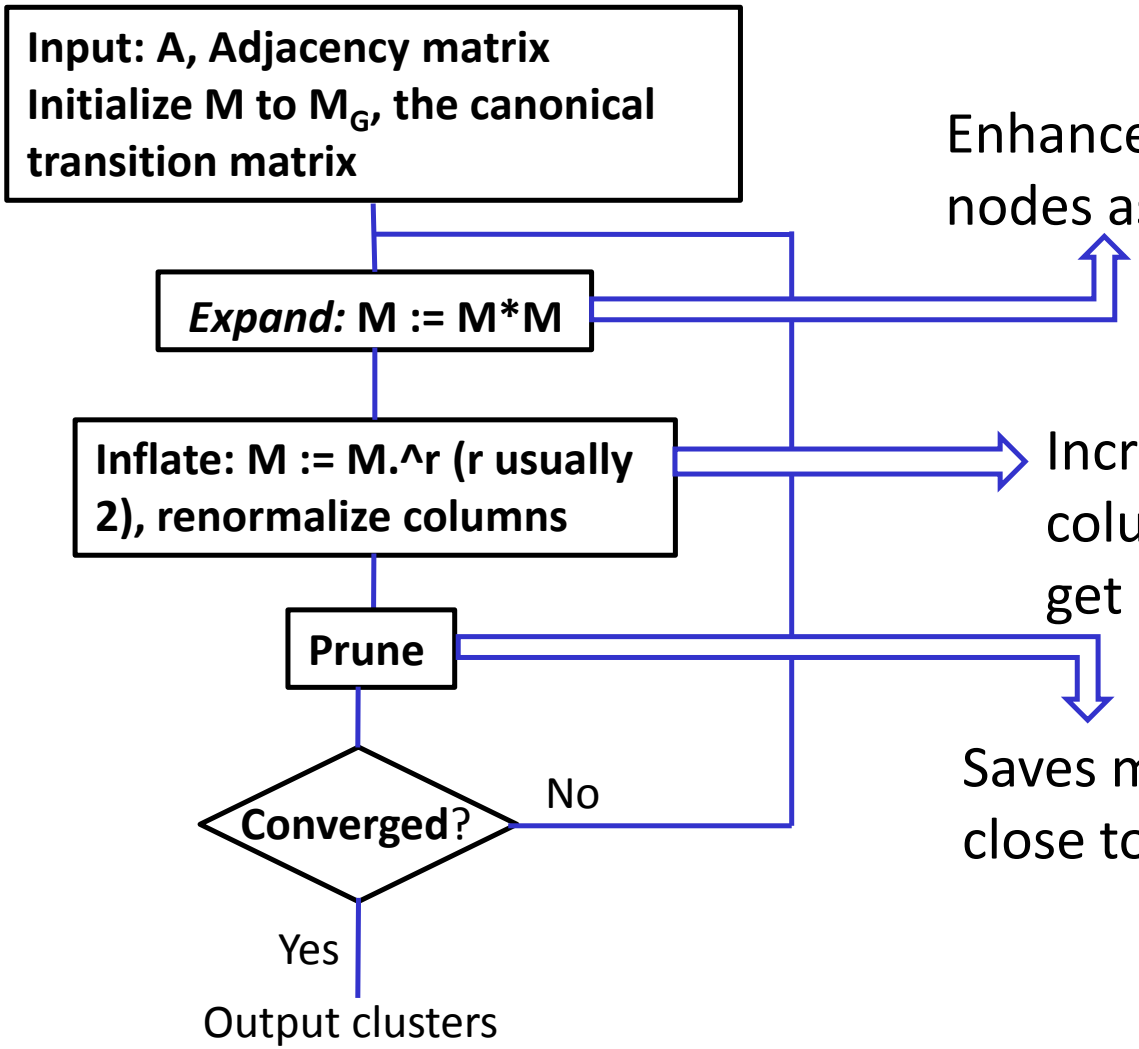
Stochastic Flow

- MCL boosts this effect by stopping the random walk and adjusting weights

- Weights are adjusted such that:
  - Strong neighbors are further strengthened
  - Weak neighbors are further weakened
  - This process is called inflation

$$\begin{bmatrix} 0 \\ 1/2 \\ 0 \\ 1/6 \\ 1/3 \end{bmatrix} \xrightarrow{\text{Squaring}} \begin{bmatrix} 0 \\ 1/4 \\ 0 \\ 1/36 \\ 1/9 \end{bmatrix} \xrightarrow{\text{Normalization}} \begin{bmatrix} 0 \\ 9/14 \\ 0 \\ 1/14 \\ 4/14 \end{bmatrix}$$

MCL Overview



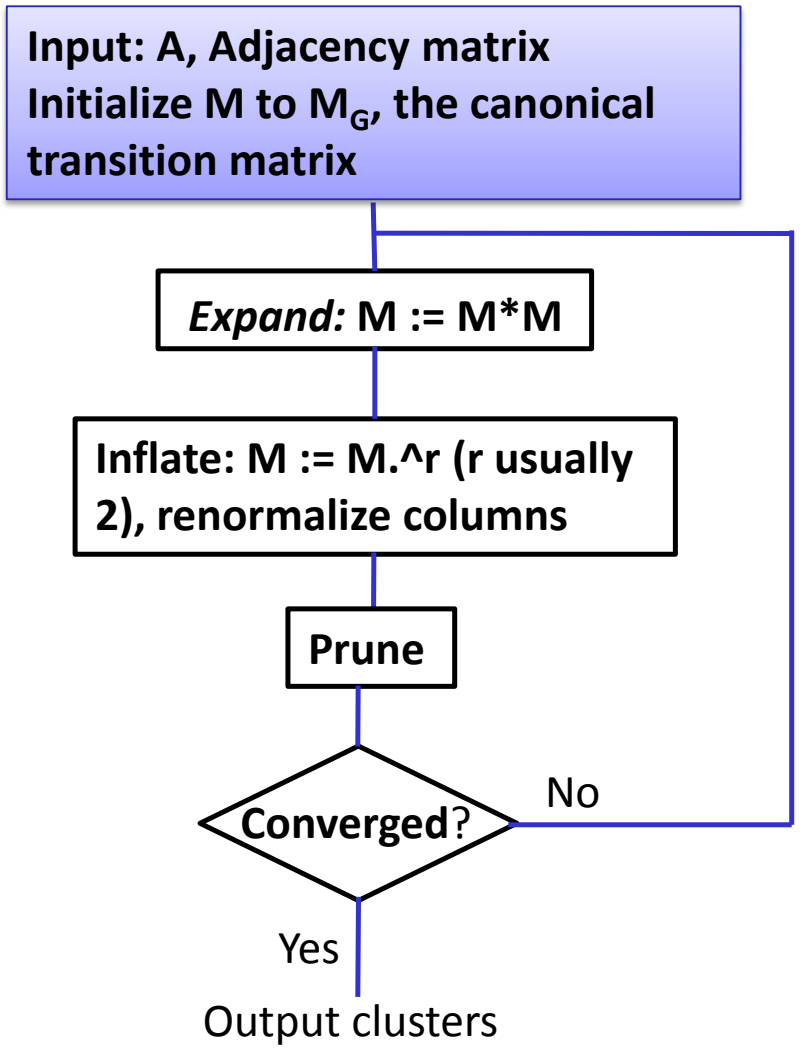
Enhances flow to well-connected nodes as well as to new nodes.

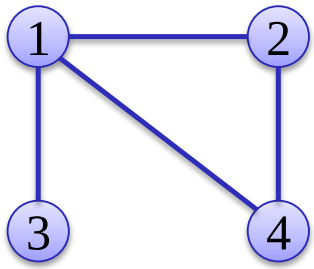
Increases inequality in each column. “Rich get richer, poor get poorer.”

Saves memory by removing entries close to zero.

Slide from ”Scalable Graph Clustering using Stochastic Flow” Venu Satuluri and Srinivasan Parthasarathy

MCL Overview

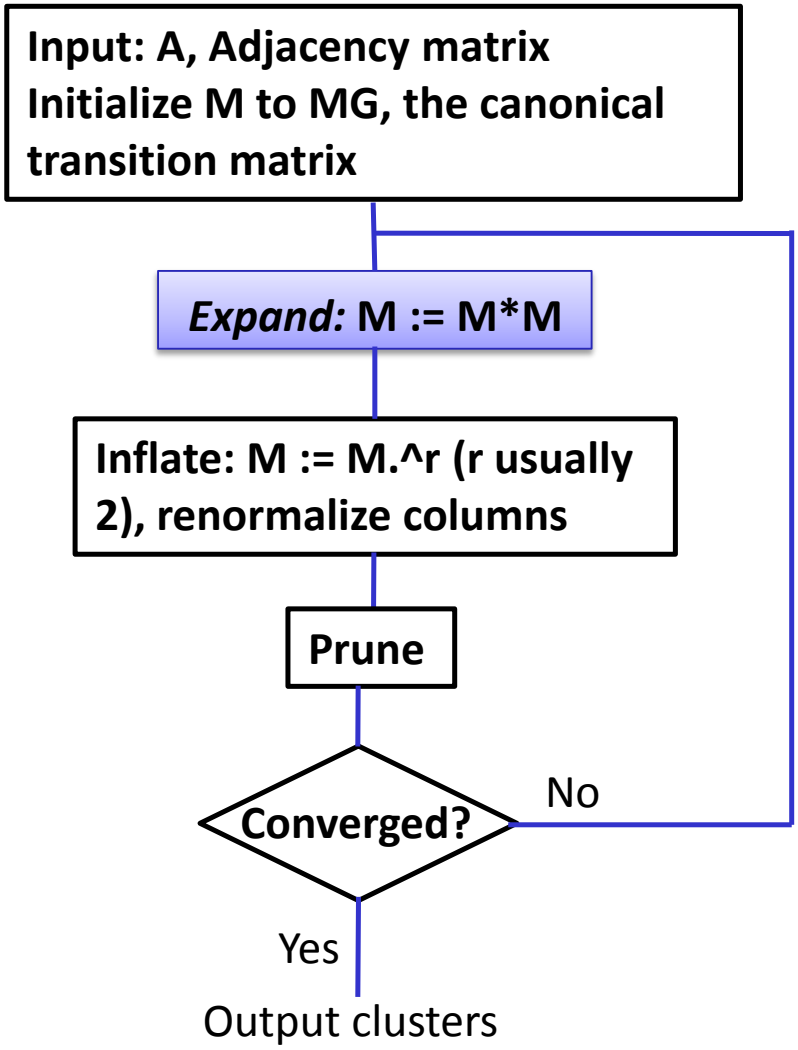


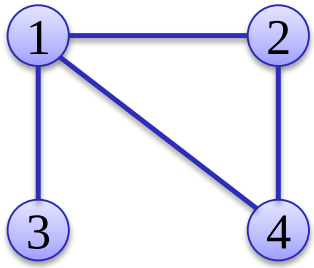


$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1/4 & 1/3 & 1/2 & 1/3 \\ 1/4 & 1/3 & 0 & 1/3 \\ 1/4 & 0 & 1/2 & 0 \\ 1/4 & 1/3 & 0 & 1/3 \end{bmatrix}$$

MCL Overview

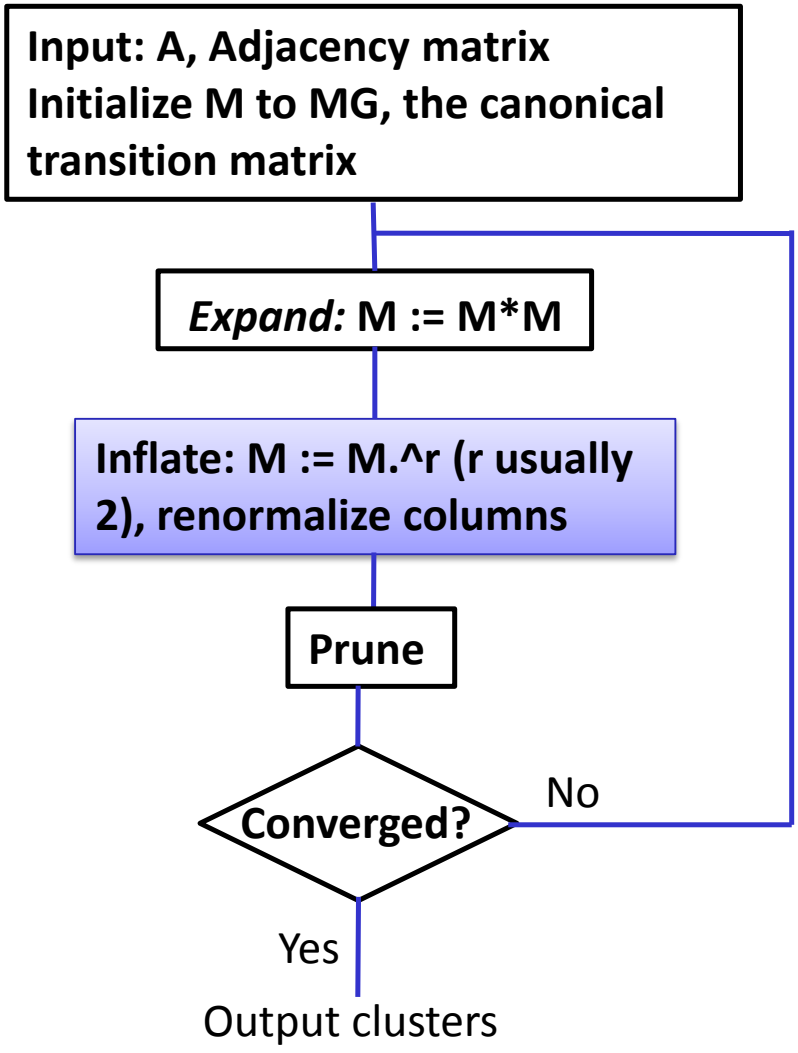


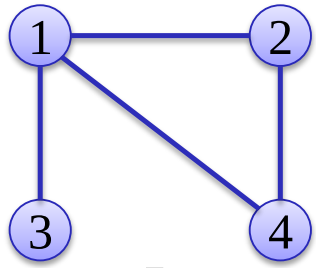


$$\begin{bmatrix} 1/4 & 1/3 & 1/2 & 1/3 \\ 1/4 & 1/3 & 0 & 1/3 \\ 1/4 & 0 & 1/2 & 0 \\ 1/4 & 1/3 & 0 & 1/3 \end{bmatrix} * \begin{bmatrix} 1/4 & 1/3 & 1/2 & 1/3 \\ 1/4 & 1/3 & 0 & 1/3 \\ 1/4 & 0 & 1/2 & 0 \\ 1/4 & 1/3 & 0 & 1/3 \end{bmatrix}$$

$$= \begin{bmatrix} 0.35 & 0.31 & 0.38 & 0.31 \\ 0.23 & 0.31 & 0.13 & 0.31 \\ 0.19 & 0.08 & 0.38 & 0.08 \\ 0.23 & 0.31 & 0.13 & 0.31 \end{bmatrix}$$

MCL Overview





$$\begin{bmatrix} 0.35 & 0.31 & 0.38 & 0.31 \\ 0.23 & 0.31 & 0.13 & 0.31 \\ 0.19 & 0.08 & 0.38 & 0.08 \\ 0.23 & 0.31 & 0.13 & 0.31 \end{bmatrix}$$

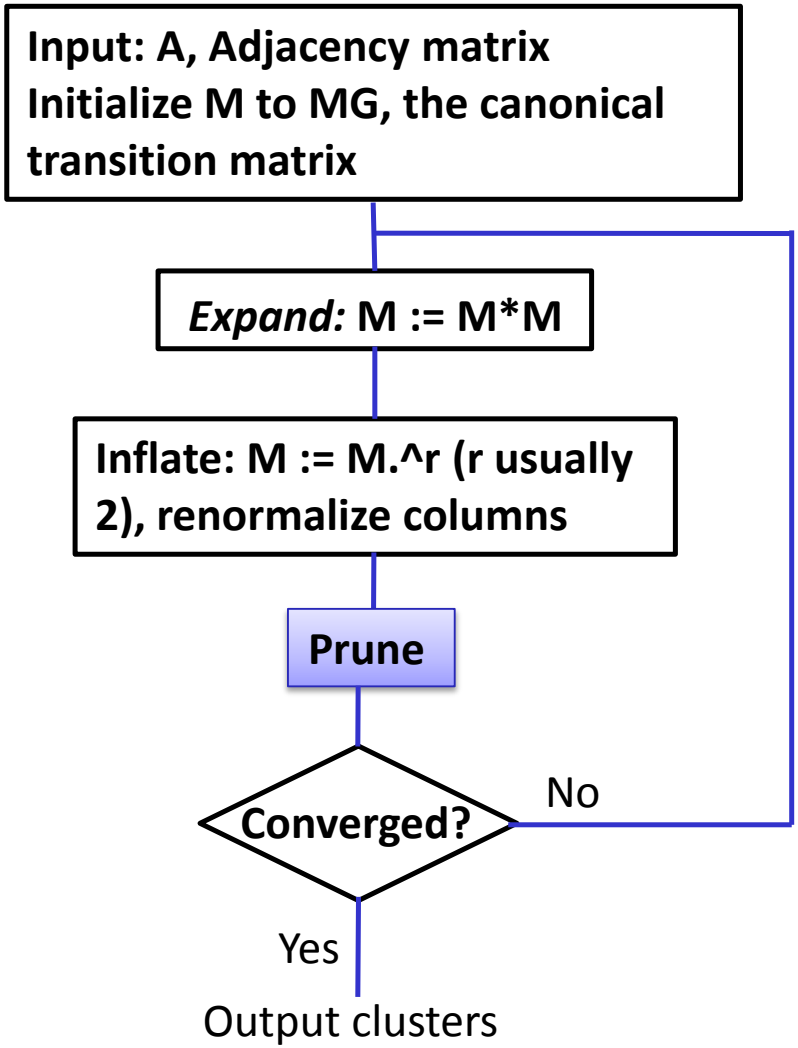
inflation

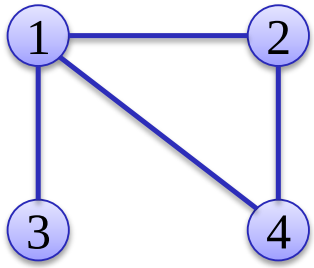
$$\begin{bmatrix} 0.13 & 0.09 & 0.14 & 0.09 \\ 0.05 & 0.09 & 0.02 & 0.09 \\ 0.04 & 0.01 & 0.14 & 0.01 \\ 0.05 & 0.09 & 0.02 & 0.09 \end{bmatrix}$$

normalization

$$\begin{bmatrix} 0.47 & 0.33 & 0.45 & 0.33 \\ 0.20 & 0.33 & 0.05 & 0.33 \\ 0.13 & 0.02 & 0.45 & 0.02 \\ 0.20 & 0.33 & 0.05 & 0.33 \end{bmatrix}$$

MCL Overview

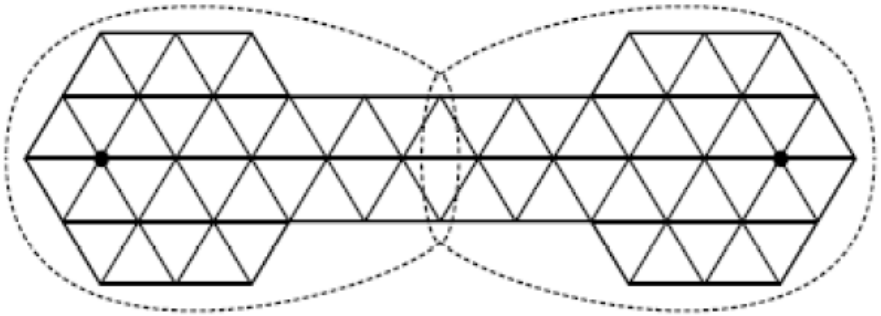




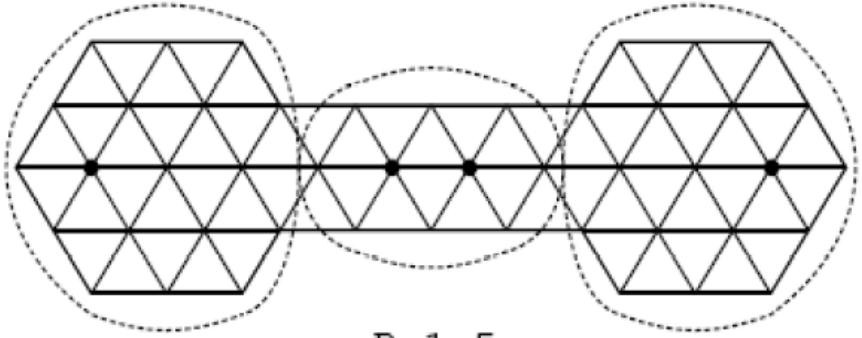
$$\begin{bmatrix} 0.47 & 0.33 & 0.45 & 0.33 \\ 0.20 & 0.33 & 0.05 & 0.33 \\ 0.13 & 0.02 & 0.45 & 0.02 \\ 0.20 & 0.33 & 0.05 & 0.33 \end{bmatrix}$$

$$\begin{bmatrix} 0.47 & 0.33 & 0.45 & 0.33 \\ 0.20 & 0.33 & 0.05 & 0.33 \\ 0.13 & 0 & 0.45 & 0 \\ 0.20 & 0.33 & 0.05 & 0.33 \end{bmatrix}$$

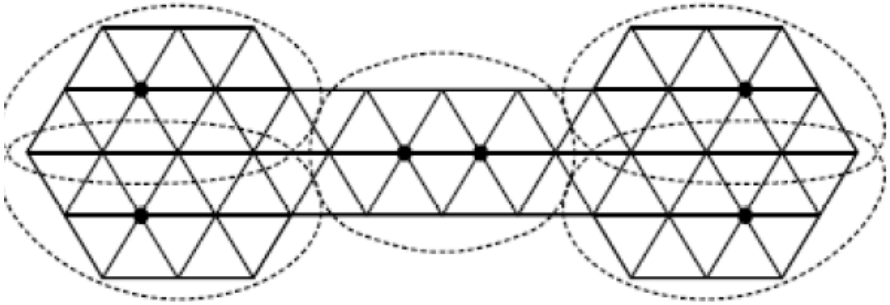
MCL Inflation Parameter



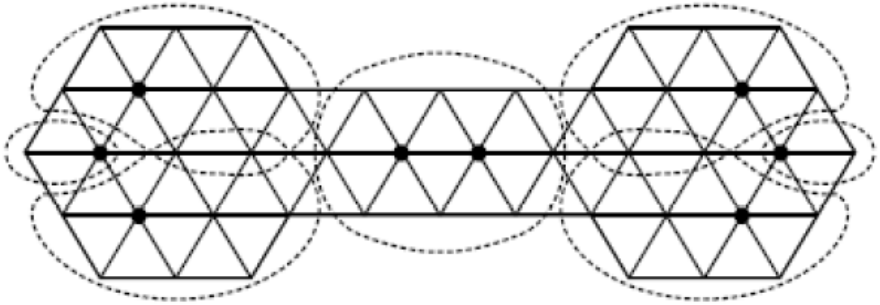
R 1.4



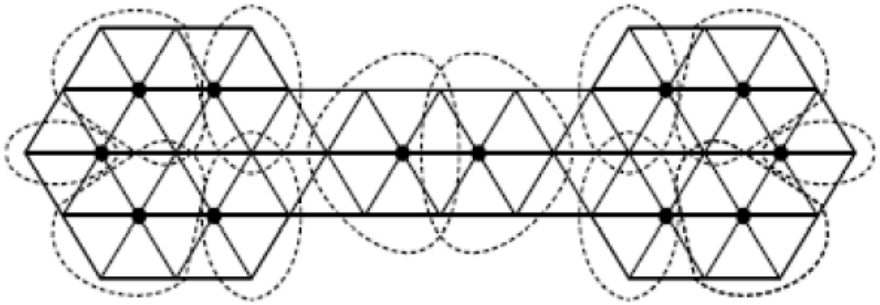
R 1.5



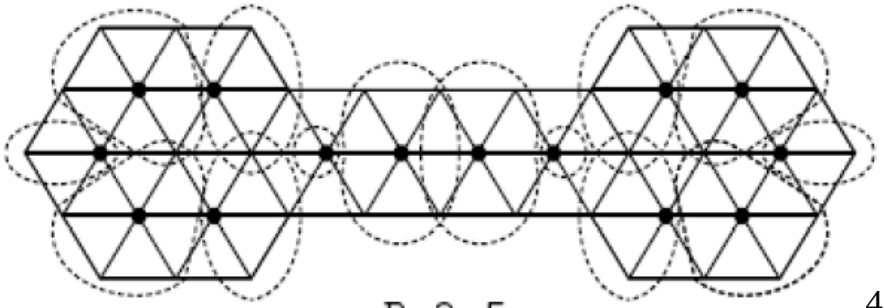
R 1.7



R 2.0



R 2.1

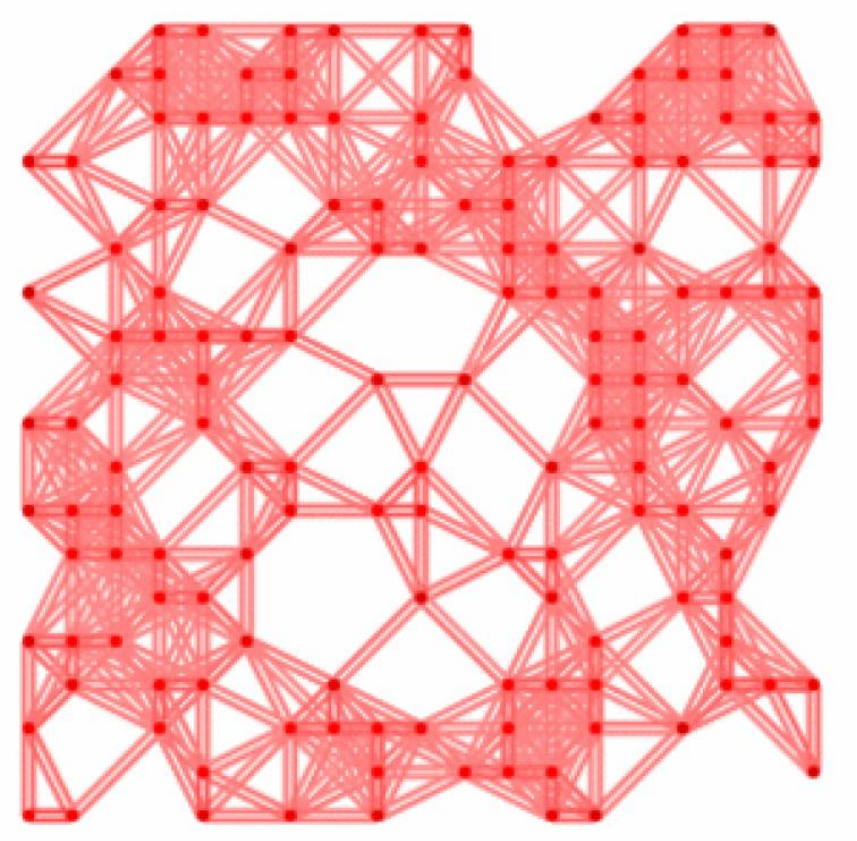


R 2.5

MCL Summary

- Time $O(N^3)$
- Input: Undirected weighted/unweighted graph
- Number of clusters not specified ahead of time
- Parameters: inflation parameter
- Evaluation: Random graphs (10000 nodes)
- Convergence: 10 ~ 100 steps

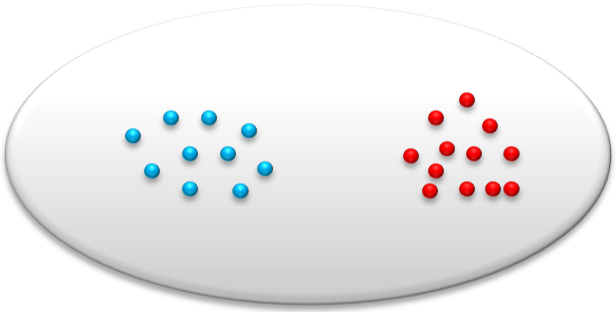
MCL Demo



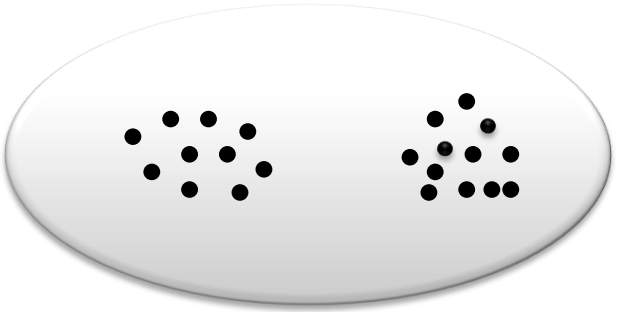
Classification using Random Walks

Semi-Supervised Learning

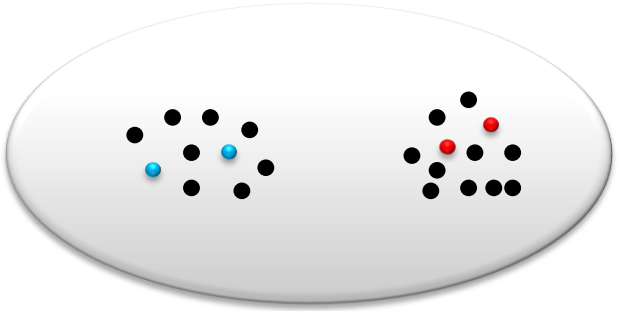
Supervised Learning



Unsupervised Learning



Semi-Supervised Learning



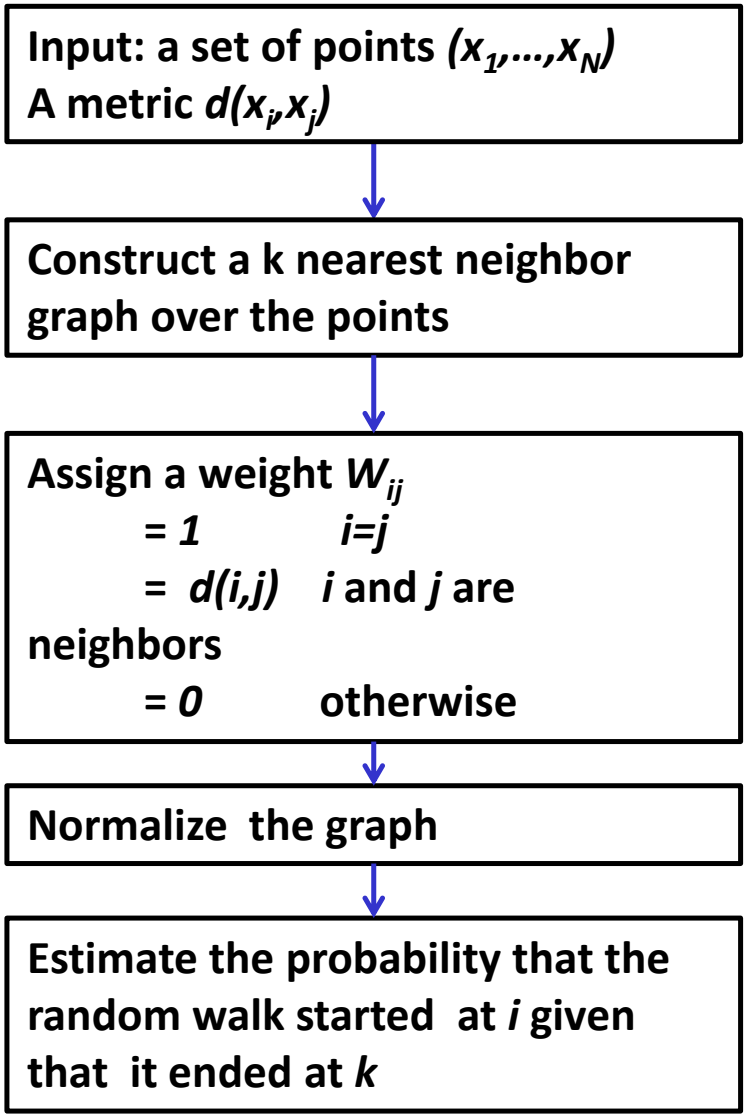
Why Semi-Supervised Learning?

- Labeled data:
  - Expensive
  - Hard to obtain
- Unlabeled data:
  - Cheap
  - Easy to obtain

Partially labeled classification with Markov random walks (Szummer 2000)

- Represent data points through a Markov random walk
- Advantages:
  - Data points in the same high density clusters have similar representation

Overview


Input: a set of points $(x_1,\ldots,x_N)$
A metric $d(x_i,x_j)$
Construct a k nearest neighbor graph over the points
Assign a weight W_{ij}
= 1 i=j
= d(i,j) i and j are neighbors
= 0 otherwise
Normalize the graph
Estimate the probability that the random walk started at i given that it ended at k

Representation

- Each node k is represented as a vector $[P_{0|t}(x_1|k), \ldots\ldots, P_{0|t}(x_n|k)]$
- $P_{0|t}(i|k)$ is the probability than the random walk ending at *k* started at *i*
- Two points are similar $\Leftrightarrow$ their random walks have indistinguishable starting points

Classification

$$P(y \mid k) = \sum_{i \in L \cup U} Q(y \mid i) P(i \mid k)$$

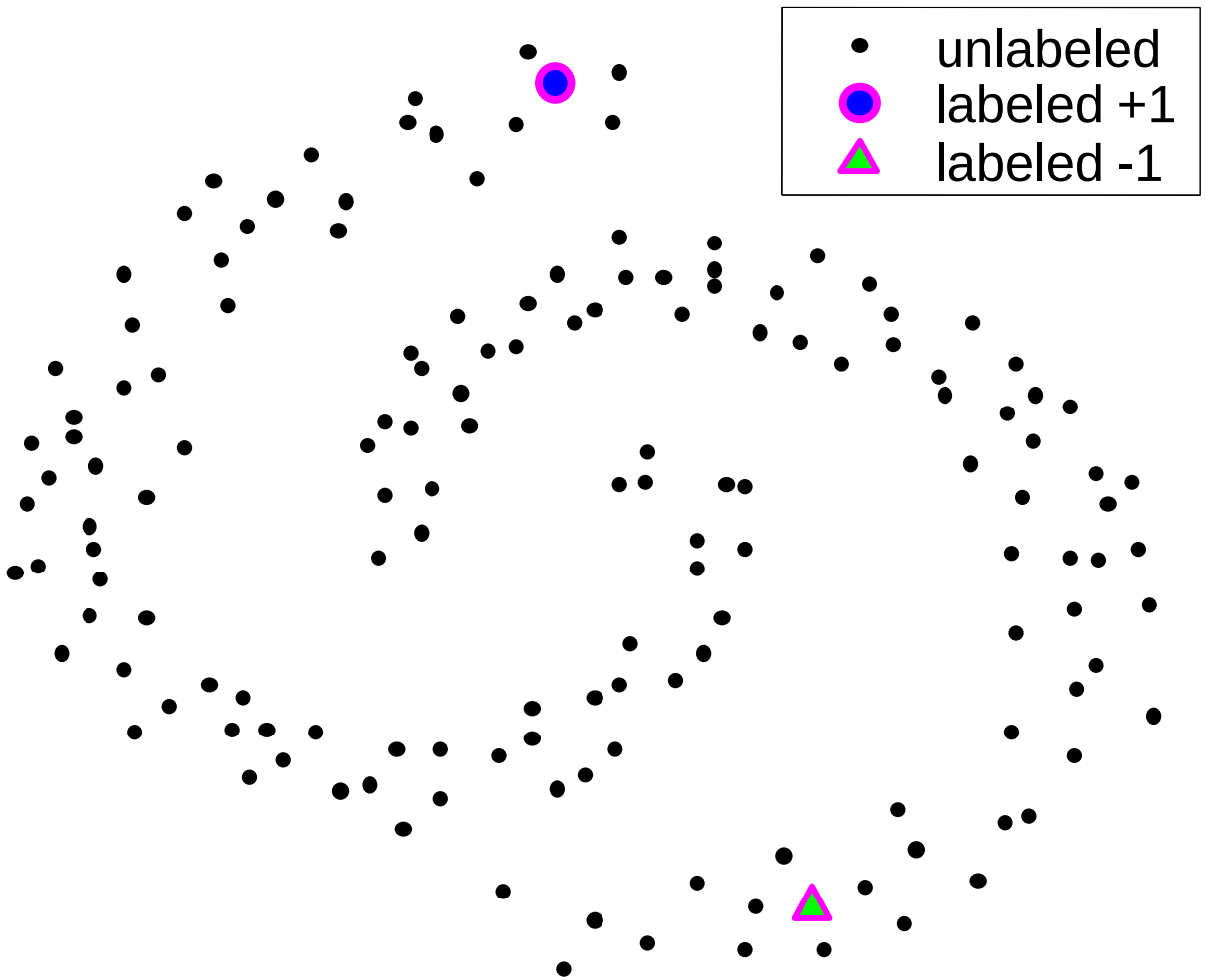
$Q(y \mid i)$ - parameters that are estimated for all points

$P(i \mid k)$ - Markov random walk representation

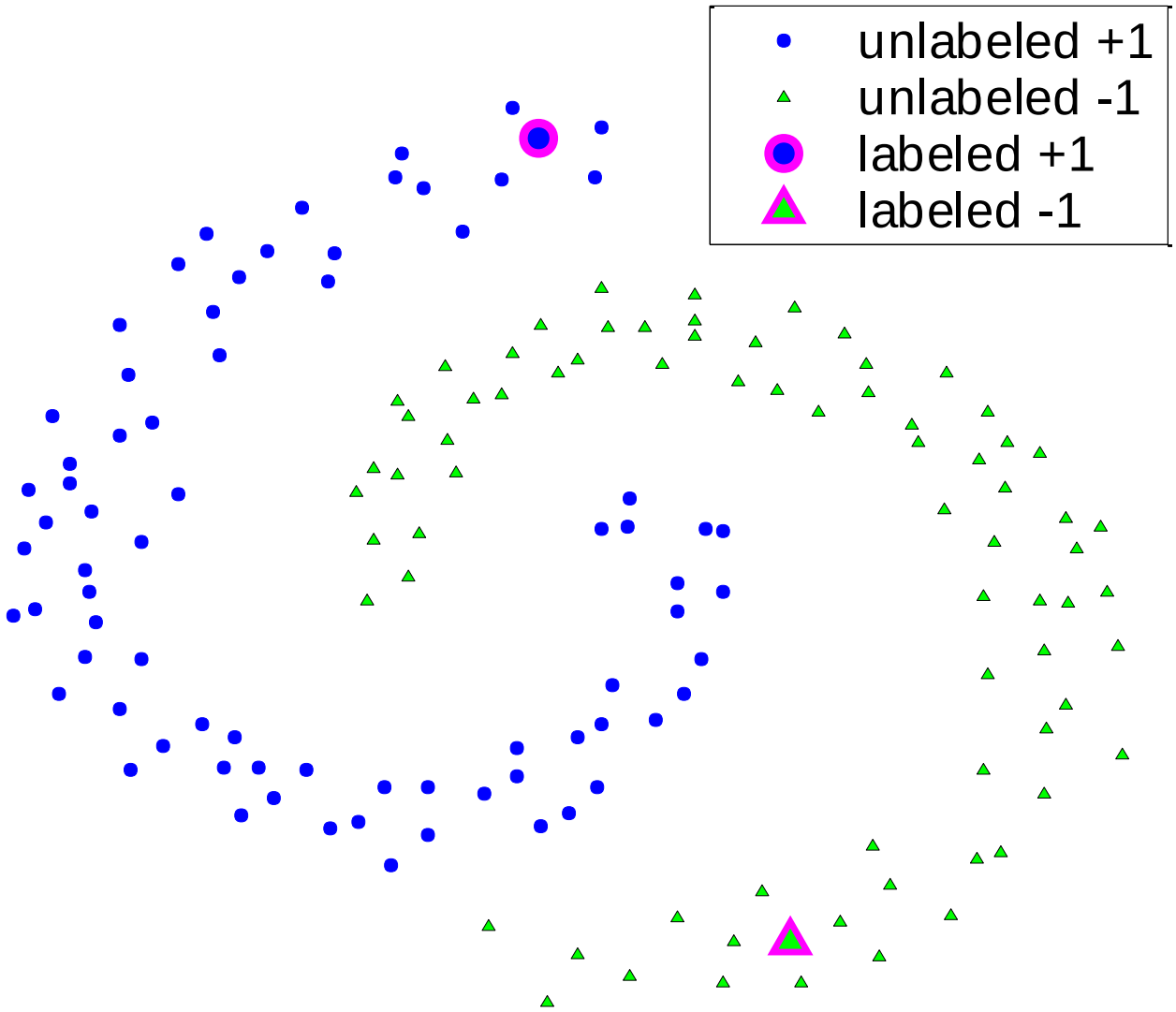
Question: how do we obtain *Q(y|i)*?

Maximize conditional log-likelihood over the labeled data using the EM algorithm

Swiss roll problem

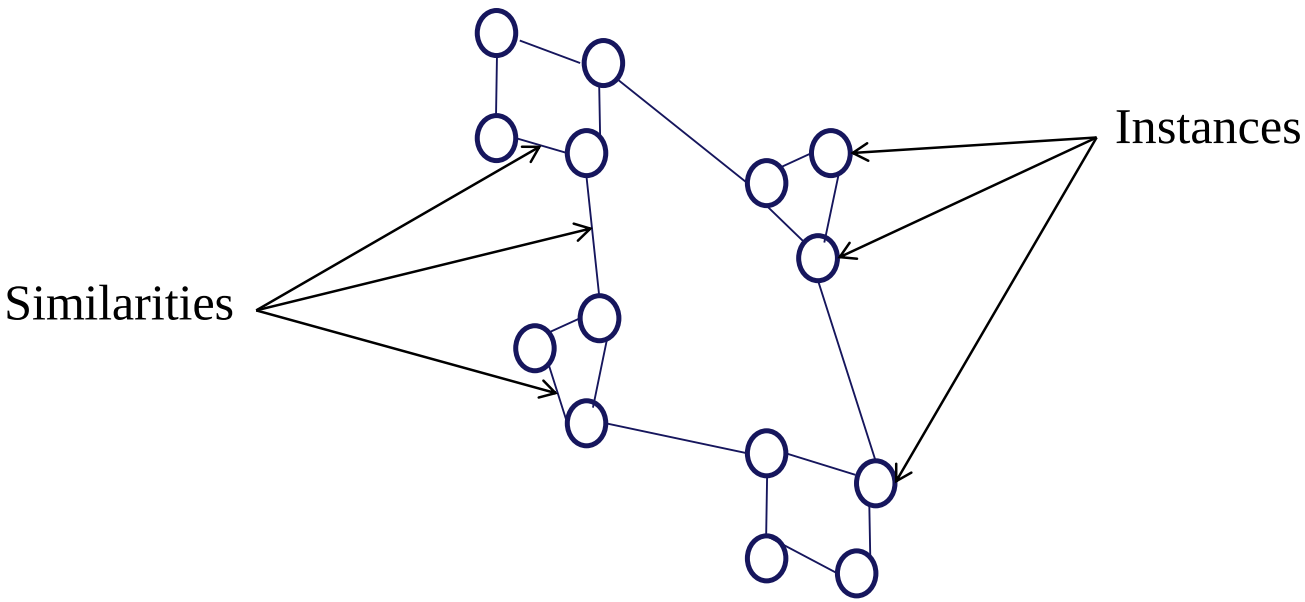


Swiss roll problem



Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions (Zhu et al. 2003)

- Labeled and Unlabeled data are represented as vertices in a weighted graph
- Edge weights encode similarity between instances



Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions (Zhu et al. 2003)

- The value of *f* at each unlabeled point is the average of *f* at neighboring points

$$f(i) = \frac{1}{d_i} \sum_{i \sim j} w_{ij} f(j) \qquad i \text{ is unlabeled}$$

- Edge weights encode similarity between instances

$$f(i) = y_i \qquad i \text{ is labeled}$$

- *f* is called a harmonic function

Partially labeled classification with Markov random walks (Szummer 2000)

- *f(i)* is the probability that a random surfer starting at node *i* hits a labeled node with label 1

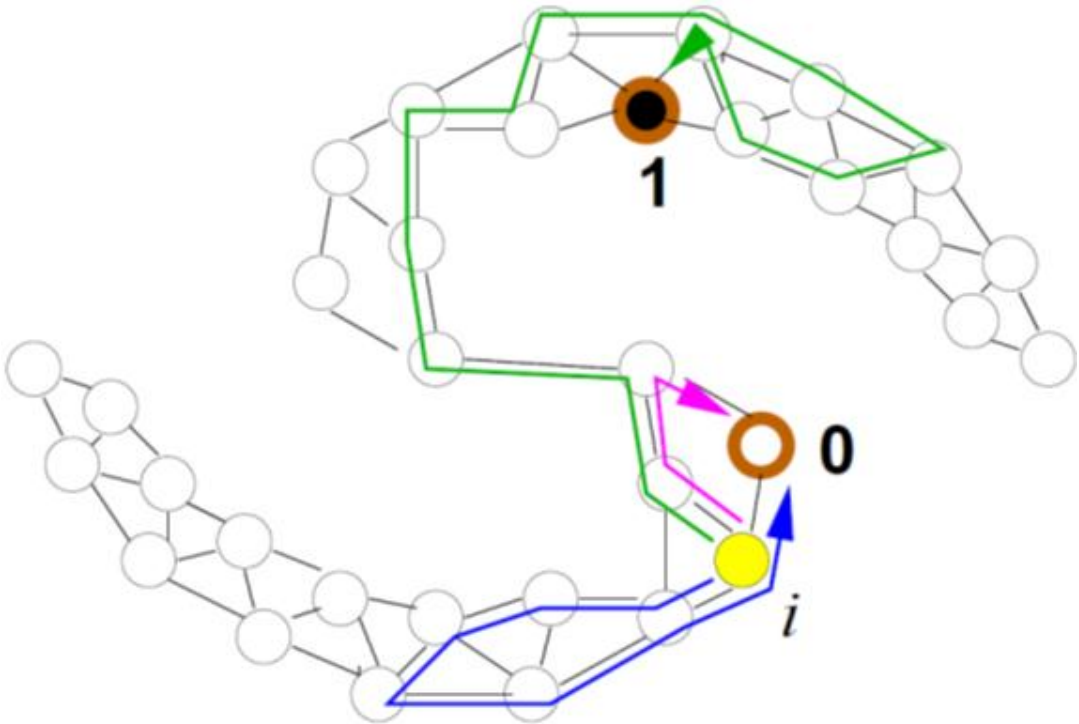


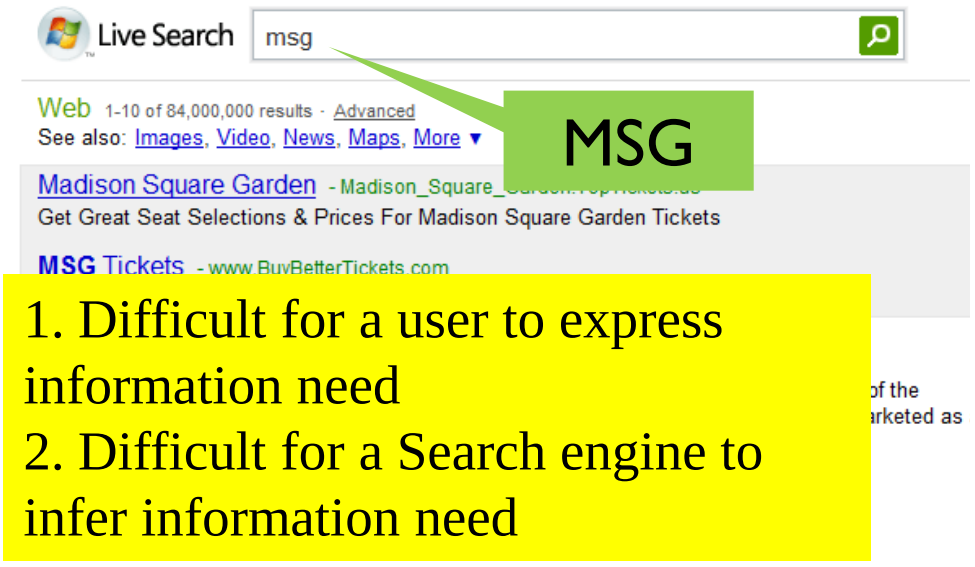
Figure from “Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions” (Zhu et al. 2003)

Other Applications using Random Walks

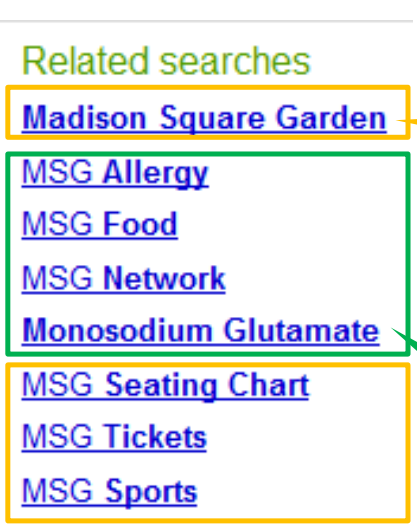
Query Suggestion Using Hitting Time (Mei et al. 2008)

- How can query suggestions be generated in a principled way?
- Construct a bipartite Graph of queries and url's
- Use Hitting Time to any given query to find related queries

Motivating Example



1. Difficult for a user to express information need
2. Difficult for a Search engine to infer information need



Sports center

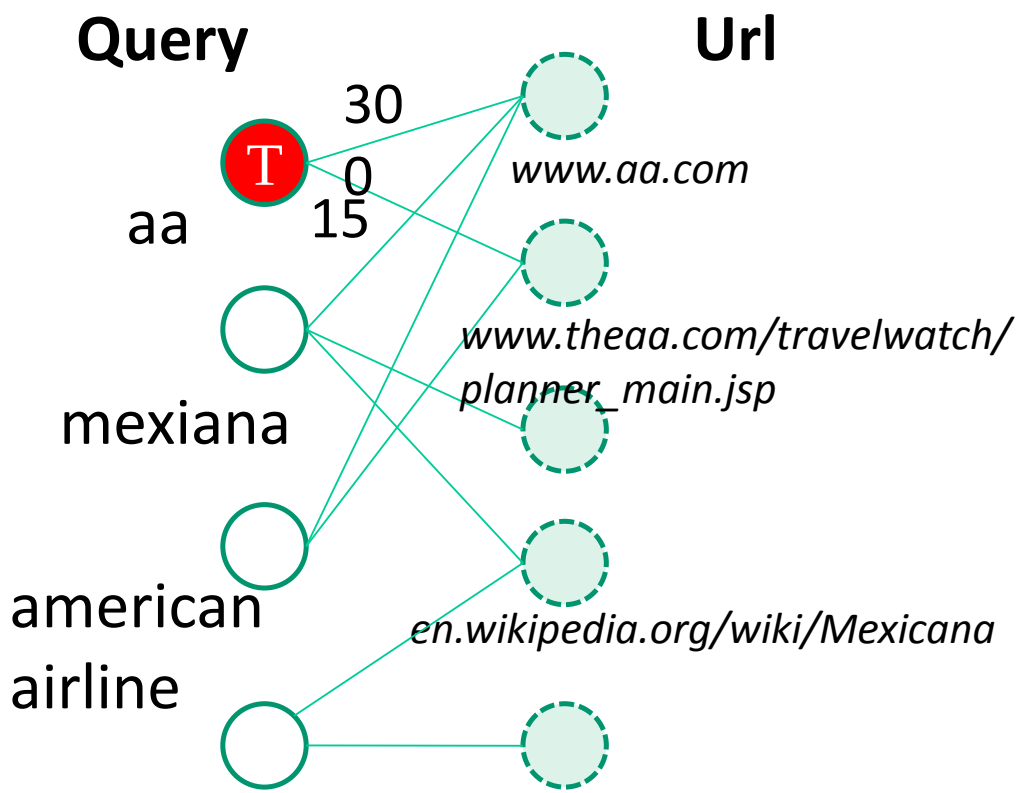
Food Additive

Query Suggestions: Accurate to express the information need;

Easy to infer information need

Slide from Query Suggestion Using Hitting Time (Mei et al. 2008)

Generate Query Suggestion



- Construct a (kNN) subgraph from the query log data (of a predefined number of queries/urls)
- Compute transition probabilities p(i → j)
- Compute hitting time h_i^A
- Rank candidate queries using h_i^A

Slide from Query Suggestion Using Hitting Time (Mei et al. 2008)

Result: Query Suggestion

Query = friends

Google
friendship
friends poem
friendster
friends episode guide
friends scripts
how to make friends
true friends

Yahoo
secret friends
friends reunited
hide friends
hi 5 friends
find friends
poems for friends
friends quotes

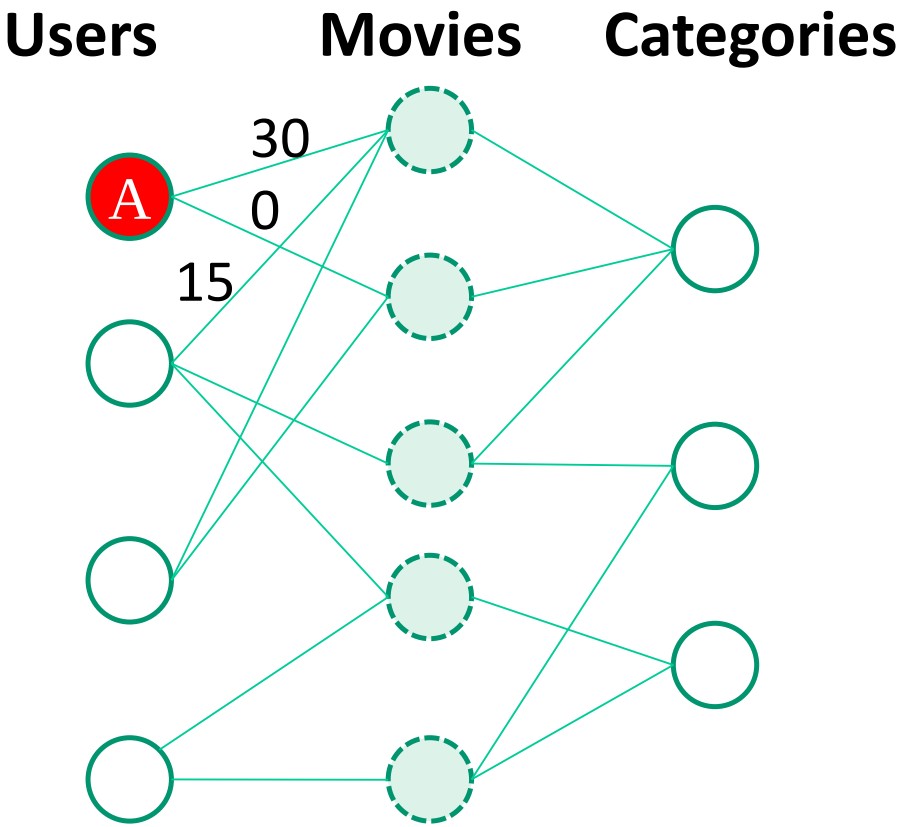
Hitting time
wikipedia friends
friends tv show wikipedia
friends home page
friends warner bros
the friends series
friends official site
friends(1994)

Slide from Query Suggestion Using Hitting Time (Mei et al. 2008)

Collaborative Recommendation (Fouss et al.)

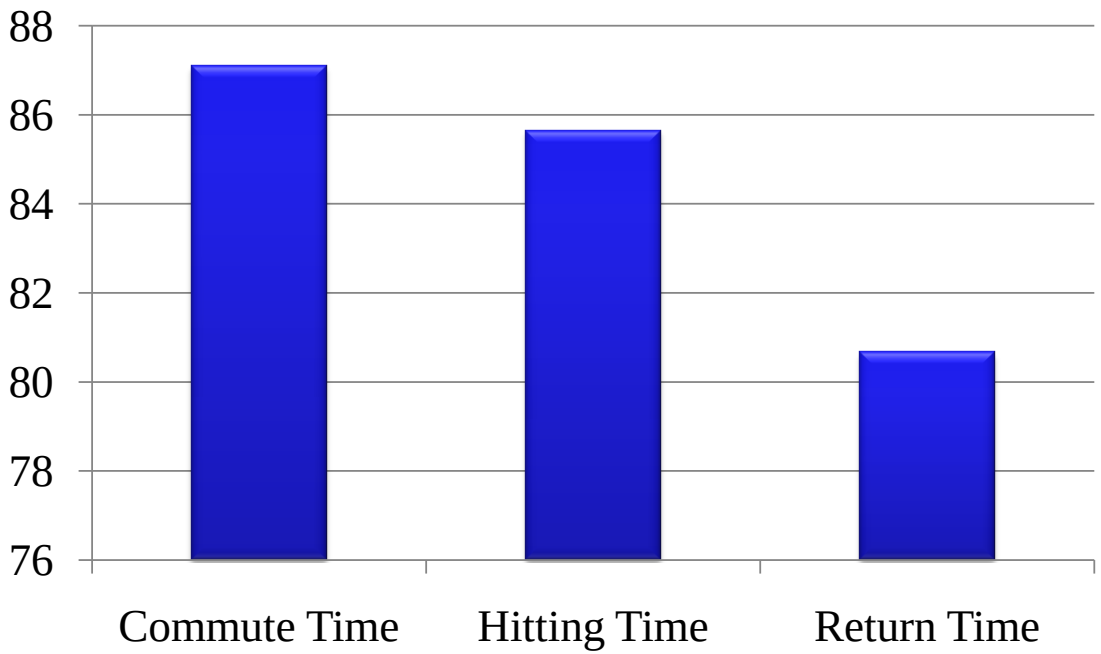
- How can query recommend movies to users?
- Construct a tripartite graph of users, movies, and movie categories
- Use Hitting Time, Commute Time, or Return Time to any given user to find closes movies

Collaborative Recommendation



- Construct a tripartite graph of users, movies, and categories
- Compute hitting time, commute time and return time from each movie to user A
- Rank movies and recommend the closet one to A

Result: Collaborative Recommendation



Language Model-Based Document Clustering Using Random Walks (Erkan 2006)

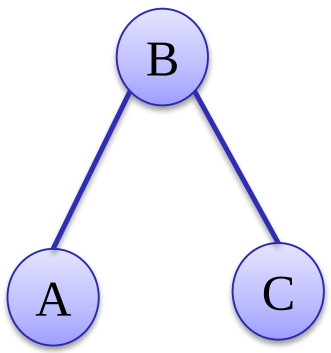
- A new document representation for clustering
- A document is represented as an n-dimensional vector
- The value at each dimension of the vector is closely related to the generation probability based on the language model of the corresponding document.
- Generation probabilities are reinforced by iterating random walks on the underlying graph

Language Model-Based Document Clustering Using Random Walks (Erkan 2006)

- For each ordered document pair (di, dj):
  - Build a language model from d_j (lm_j)
  - compute the generation probability of d_i from lm_j
- Build a generation graph where nodes are documents edge weights represent generation probabilities

Language Model-Based Document Clustering Using Random Walks (Erkan 2006)

- There are “strong” generation links from A to B and B to C, but no link from A to C.
- The intuition says that A must be semantically related to C
- This relation is approximated by considering the probabilities of t-step random walks from A to C



Sampling and Summarization for Social Networks

ShouDe Lin, MiYen Yeh, and ChengTe Li, National Taiwan University

Sampling by Exploration

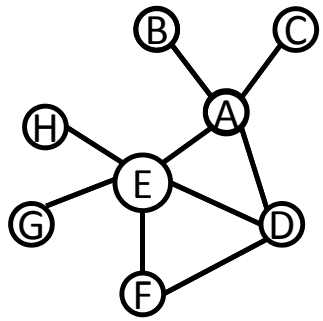
- Random Walk [Gjoka'10]
  - The next-hop node is chosen uniformly among the neighbors of the current node
- Random Walk with Restart [Leskovec'06]
  - Uniformly select a random node and perform a random walk with restarts
- Random Jump [Ribeiro'10]
  - Same as random walk but with a probability p we jump to any node in the network
- Forest Fire [Leskovec'06]
  - Choose a node *u* uniformly
  - Generate a random number *z* and select *z* out links of *u* that are not yet visited
  - Apply this step recursively for all newly added nodes

Sampling by Exploration (cont.)

- Ego-Centric Exploration (ECE) Sampling
  - Similar to random walk, but each neighbor has p probability to be selected
  - Multiple ECE (starting with multiple seeds)
- Depth-First / Breadth-First Search [Krishnamurthy'05]
  - Keep visiting neighbors of *earliest* / *most recently* visited nodes
- Sample Edge Count [Maiya'11]
  - Move to neighbor with the highest degree, and keep going
- Expansion Sampling [Maiya'11]
  - Construct a sample with the maximal expansion. Select the neighbor v based on $argmax_{v \in N(S)} |N(\{v\}) - (N(S) \cup S)|$

S: the set of sampled nodes, N(S): the 1st neighbor set of S

Example: Expansion Sampling

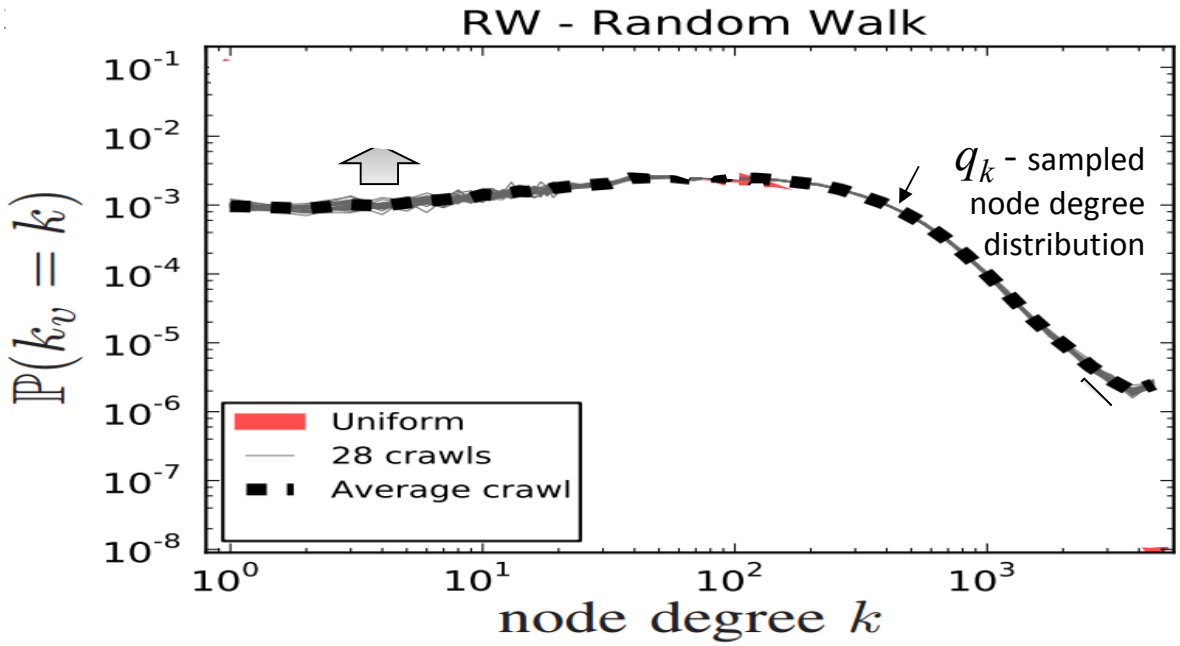


$|N(\{A\})|=4$

$|N(\{E\}) - N(\{A\}) \cup \{A\}|=|\{F,G,H\}|=3$

$|N(\{D\}) - N(\{A\}) \cup \{A\}|=|\{F\}|=1$

Drawback of Random Walk: Degree Bias!



- Real average node degree ~ 94, Sampled average node degree ~ 338
- Solution: modify the transition probability :

$$P_{v,w} = \begin{cases} \frac{1}{k_v} * \min(1, \frac{k_v}{k_w}) & \text{If w is a neighbor of v} \\ 1 - \sum_{y \neq v} P_{v,y} & \text{If w = v} \\ 0 & \text{otherwise} \end{cases}$$

Metropolis Graph Sampling [Hubler'08]

- Step 1: Initially pick one subgraph sample S with n' nodes randomly
- Step 2: Iterate the following steps until convergence
  - 2.1: Remove one node from S
  - 2.2: Randomly add a new node to S → S'
  - 2.3: Compute the likelihood ratio $a = \frac{\rho^*(S')}{\rho^*(S)}$

    $if\ a \geq 1: accept\ transition: S := S'$

    $if\ a < 1: accept\ transition: S := S'$ with probability a

    $reject\ transition: S := S'$ with probability $1 - a$
  - – $\rho^*(S)$ measures the similarity of a certain property between the sample S and the original network G
    - Be derived approximately using Simulated Annealing

Today: In class work

- Implement random walk sampling methods
- Compare their efficacy on various networks

Graph Sampling
Blank code and data available on website
(Lecture 20)
`www.cs.rpi.edu/~slotag/classes/FA16/index.html`