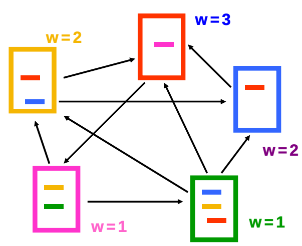
More on Rankings

Query-independent LAR

- Have an a-priori ordering of the web pages
- **Q**: Set of pages that contain the keywords in the query **q**
- Present the pages in **Q** ordered according to order **π**
- **What are the advantages of such an approach?**

InDegree algorithm

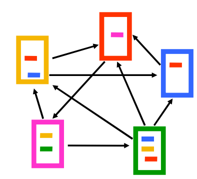
- Rank pages according to in-degree
  - $w_i = |B(i)|$



1. **Red Page**
2. **Yellow Page**
3. **Blue Page**
4. **Purple Page**
5. **Green Page**

PageRank algorithm [BP98]

- Good authorities should be pointed by good authorities
- Random walk on the web graph
  - pick a page at random
  - with probability 1- α jump to a random page
  - with probability α follow a random outgoing link
- Rank according to the stationary distribution
- $$PR(p) = \alpha \sum_{q \to p} \frac{PR(q)}{|F(q)|} + (1-\alpha)\frac{1}{n}$$



1. **Red Page**
2. **Purple Page**
3. **Yellow Page**
4. **Blue Page**
5. **Green Page**

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
    - $\sum_j P_{ij} = 1$ (stochastic matrix)

- Memorylessness property: The next state of the chain depends only at the current state and not on the past of the process (first order MC)
  - higher order MCs are also possible

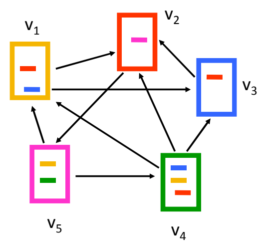
Random walks

- Random walks on graphs correspond to Markov Chains
  - The set of states S is the set of nodes of the graph G
  - The transition probability matrix is the probability that we follow an edge from one node to another

An example

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

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



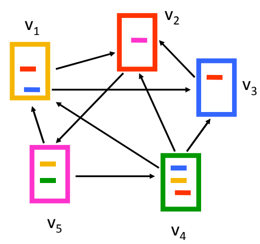
State probability vector

- The vector $q^t = (q^t_1, q^t_2, \dots, q^t_n)$ that stores the probability of being at state i at time t
  - q^0_i = the probability of starting from state i

$$q^t = q^{t-1} P$$

An example

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



$q^{t+1}_1 = 1/3\, q^t_4 + 1/2\, q^t_5$

$q^{t+1}_2 = 1/2\, q^t_1 + q^t_3 + 1/3\, q^t_4$

$q^{t+1}_3 = 1/2\, q^t_1 + 1/3\, q^t_4$

$q^{t+1}_4 = 1/2\, q^t_5$

$q^{t+1}_5 = q^t_2$

Stationary distribution

- A stationary distribution for a MC with transition matrix P, is a probability distribution π, such that $\pi = \pi P$
- A MC has a unique stationary distribution if
  - it is irreducible
    - the underlying graph is strongly connected
  - it is aperiodic
    - for random walks, the underlying graph is not bipartite
- The probability π_i is the fraction of times that we visited state i as $t \rightarrow \infty$
- The stationary distribution is an eigenvector of matrix P
  - the principal left eigenvector of P – stochastic matrices have maximum eigenvalue 1

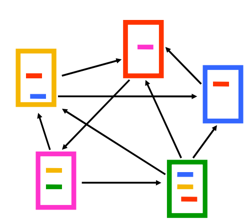
Computing the stationary distribution

- The Power Method
  - Initialize to some distribution q^0
  - Iteratively compute $q^t = q^{t-1}P$
  - After enough iterations $q^t \approx \pi$
  - Power method because it computes $q^t = q^0 P^t$
- Rate of convergence
  - determined by λ_2

The PageRank random walk

- Vanilla random walk
  - make the adjacency matrix stochastic and run a random walk

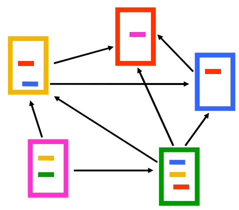
$$P = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$



The PageRank random walk

- What about sink nodes?
  - what happens when the random walk moves to a node without any outgoing inks?

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

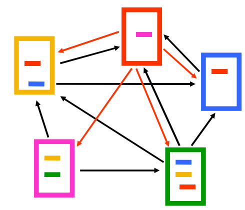


The PageRank random walk

- Replace these row vectors with a vector v
  - typically, the uniform vector

$$P' = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

$P' = P + dv^T$ $\quad d = \begin{cases} 1 & \text{if i is sink} \\ 0 & \text{otherwise} \end{cases}$



The PageRank random walk

- How do we guarantee irreducibility?
  - add a random jump to vector v with prob α
    - typically, to a uniform vector

$$P'' = \alpha \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix} + (1-\alpha) \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix}$$

$P'' = \alpha P' + (1-\alpha)uv^T$, where u is the vector of all 1s

Effects of random jump

- Guarantees irreducibility
- Motivated by the concept of random surfer
- Offers additional flexibility
  - personalization
  - anti-spam
- Controls the rate of convergence
  - the second eigenvalue of matrix P'' is α

A PageRank algorithm

- Performing vanilla power method is now too expensive – the matrix is not sparse

$q^0 = v$
$t = 1$
repeat
$\quad q^t = (P'')^T q^{t-1}$
$\quad \delta = \|q^t - q^{t-1}\|$
$\quad t = t + 1$
until $\delta < \varepsilon$

Efficient computation of $y = (P'')^T x$

$y = \alpha P^T x$
$\beta = \|x\|_1 - \|y\|_1$
$y = y + \beta v$

Random walks on undirected graphs

- In the stationary distribution of a random walk on an undirected graph, the probability of being at node i is proportional to the (weighted) degree of the vertex
- Random walks on undirected graphs are not "interesting"

Research on PageRank

- Specialized PageRank
  - personalization [BP98]
    - instead of picking a node uniformly at random favor specific nodes that are related to the user
  - topic sensitive PageRank [H02]
    - compute many PageRank vectors, one for each topic
    - estimate relevance of query with each topic
    - produce final PageRank as a weighted combination
- Updating PageRank [Chien et al 2002]
- Fast computation of PageRank
  - numerical analysis tricks
  - node aggregation techniques
  - dealing with the "Web frontier"

Topic-sensitive pagerank

- HITS-based scores are very inefficient to compute
- PageRank scores are independent of the queries
- Can we bias PageRank rankings to take into account query keywords?

Topic-sensitive PageRank

Topic-sensitive PageRank

- Conventional PageRank computation:
- $r^{(t+1)}(v)=\Sigma_{u \in N(v)} r^{(t)}(u)/d(v)$
- $N(v)$: neighbors of v
- $d(v)$: degree of v
- $r = Mxr$
- $M' = (1-\alpha)P + \alpha[1/n]_{nxn}$
- $r = (1-\alpha)Pr + \alpha[1/n]_{nxn}r = (1-\alpha)Pr + \alpha p$
- $p = [1/n]_{nx1}$

Topic-sensitive PageRank

- $r = (1-\alpha)Pr + \alpha p$
- **Conventional PageRank:** p is a uniform vector with values $1/n$
- Topic-sensitive PageRank uses a **non-uniform** personalization vector p
- Not simply a post-processing step of the PageRank computation
- Personalization vector p introduces bias in all iterations of the iterative computation of the PageRank vector

Personalization vector

- In the random-walk model, the personalization vector represents the addition of a set of transition edges, where the probability of an artificial edge (u,v) is αp_v
- Given a graph the result of the PageRank computation only depends on α and p : $PR(\alpha,p)$

Topic-sensitive PageRank: Overall approach

- Preprocessing
  - Fix a set of k topics
  - For each topic c_j compute the PageRank scores of page u wrt to the j-th topic: $r(u,j)$
- Query-time processing:
  - For query q compute the total score of page u wrt q as $score(u,q) = \Sigma_{j=1 \ldots k} Pr(c_j|q)\ r(u,j)$

Topic-sensitive PageRank: Preprocessing

- Create **k** different biased PageRank vectors using some pre-defined set of k categories **($c_1,\ldots,c_k$)**
- **T_j**: set of URLs in the **j**-th category
- Use non-uniform personalization vector **$p=w_j$** such that:

$$w_j(v) = \begin{cases} \frac{1}{T_j}, v \in T_j \\ 0, \text{o/w} \end{cases}$$

Topic-sensitive PageRank: Query-time processing

- **D_j**: class term vectors consisting of all the terms appearing in the **k** pre-selected categories

$$\Pr(c_j \mid q) = \frac{\Pr(c_j)\Pr(q \mid c_j)}{\Pr(q)} \propto \Pr(c_j)\prod_i \Pr(q_i \mid c_j)$$

- How can we compute **$P(c_j)$**?
- How can we compute **$Pr(q_i|c_j)$**?

- Comparing results of Link Analysis Ranking algorithms
- Comparing and aggregating rankings

Comparing LAR vectors

w_1 = [1 0.8 0.5 0.3 0]

w_2 = [0.9 1 0.7 0.6 0.8]

- How close are the LAR vectors w_1, w_2?

Distance between LAR vectors

- Geometric distance: how close are the numerical weights of vectors w_1, w_2?

$$d_1(w_1, w_2) = \sum |w_1[i] - w_2[i]|$$

w_1 = [1.0 0.8 0.5 0.3 0.0]

w_2 = [0.9 1.0 0.7 0.6 0.8]

$d_1(w_1,w_2)$ = 0.1+0.2+0.2+0.3+0.8 = 1.6

Distance between LAR vectors

- Rank distance: how close are the ordinal rankings induced by the vectors w_1, w_2?
  - Kendal's τ distance

$$d_r(w_1, w_2) = \frac{\text{pairs ranked in a different order}}{\text{total number of distinct pairs}}$$