2 Efficient Personalized Authority Ranking

2.1 Memory-efficient Incremental Page-Importance Computation 2.2 Personalized Page-Rank for Many Users

2.1 Memory-efficient Incremental Page Importance

Goals:

- Compute Page-Rank-style authority measure online
- without having to store the complete link graphRecompute authority incrementally as the graph changes

Key idea:

- Each page holds some "cash" that reflects its importance
- When a page is visited, it distributes its cash among its successors
- When a page is not visited, it can still accumulate cash
 This random process has a stationary limit
- that captures importance of pages

OPIC Algorithm (Online Page Importance Computation)

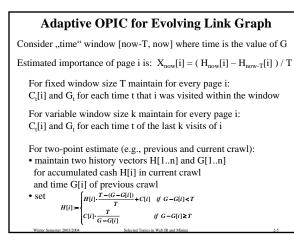
Maintain for each page i (out of n pages): C[i] – cash that page i currently has and distributes H[i] – history of how much cash page has ever had in total plus global counter G – total amount of cash that has ever been distributed for each i do { C[i] := 1/n; H[i] := 0 }; G := 0;

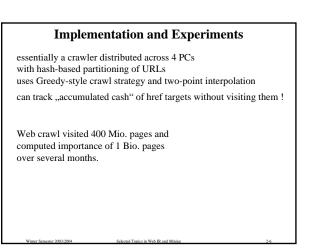
do forever {
 choose page i (e.g., randomly);
 H[i] := H[i] + C[i];
 for each successor j of i do C[j] := C[j] + C[i] / outdegree(i);
 G := G + C[i];
 C[i] := 0;
};

Note: 1) every page needs to be visited infinitely often (fairness) 2) the link graph L' is assumed to be strongly connected

selected Topics in Web IR and M

$\begin{array}{l} \textbf{OPIC Importance Measure} \\ \text{At each step t an estimate of the importance of page i is: (H_{t}[i] + C_{t}[i]) / (G_{t} + 1) (or alternatively: H_{t}[i] / G_{t}) \\ \hline \textbf{Theorem:} \\ \text{Let } X_{t} = H_{t} / G_{t} \text{ denote the vector of cash fractions accumulated by pages until step t. The limit X = lim _{t \rightarrow \infty} X_{t} exists with |X|_{I} = \Sigma_{t} X[i] = 1. \\ \hline \textbf{with crawl strategies such as: } \\ \text{ i random } \\ \text{ igreedy: read page i with highest cash C[i] (fair because non-visited pages accumulate cash until eventually read) \\ \text{ ocyclic (round-robin)} \end{array}$





2.2 Personalized Page-Rank for Many Users

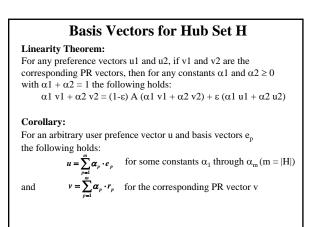
Efficiently compute solution of personalized PR vector $\mathbf{x}_{(n\times 1)}$ with user preference vector \mathbf{u} with $|\mathbf{u}|_1=1$ and $\mathbf{u}_i\neq 0$ only for $\mathbf{i}\in \mathbf{H}$, with $|\mathbf{H}|<<\mathbf{n}$, and $\mathbf{A}_{ii} = 1$ /outdegree(i) for edge $\mathbf{i}\rightarrow \mathbf{j}_i 0$ else: $\mathbf{x} = (\mathbf{1}-\boldsymbol{\varepsilon})\mathbf{A}\mathbf{x} + \boldsymbol{\varepsilon}\mathbf{u}$

Key ideas:

- 1) consider only basis vectors u with $u_p=1$ and $u_i=0$ for $i\neq p$ for hub p and represent full user preference as linear combination
- 2) represent p-specific Page-Rank vectors in the form of
- a hub skeleton and a set of partial vectors3) factor out common parts of different random walks

Notation:

hub set H, preference set $P \subseteq H \subset V = \{1, 2, ..., n\}$ basis vector e_p with single non-zero entry at p p-specific PR vector r_p partial vector $r_p - r_p^H$



Hubs Skeleton and Partial Vectors (2)

Partial vectors: For many q: $r_p(q) - r^H_p(q) = 0$, and the number of such q increases with |H|. \rightarrow store only sparse vectors $r_p(q) - r^H_p(q)$ *Note: partial vectors become smaller when pages in H have high PR* **Hubs skeleton:** Compute r^H_p vector from partial vectors and a "skeleton" **Hubs theorem:** For any page p and hub set H: $r_p^H = \frac{1}{\varepsilon} \sum_{h \in H} (r_p(h) - \varepsilon e_p(h)) (r_h - r_h^H - \varepsilon e_h)$ and thus $r_p = (r_p - r_p^H) + \frac{1}{\varepsilon} \sum_{h \in H} (r_p(h) - \varepsilon e_p(h)) ((r_h - r_h^H) - \varepsilon e_h)$

In addition to the partial vectors, we thus need to compute and store the skeleton $S = \{ r_p(h) | h \in H \}$ *Note:* $r_p(h)$ *for all* $h \in H$ *is much smaller than* $r_p(q)$ *for all pages* q

Hubs Skeleton and Web Skeleton

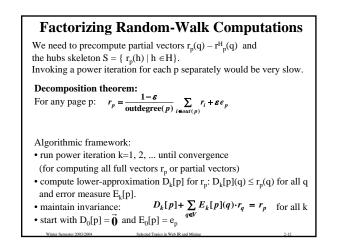
Intuition behind Hub Theorem:

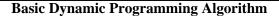
Distance from p to q through H is distance $r_p(h)$ from p to each $h \in H$ times the distance from $r_h(q)$ from h to q.

The hubs skeleton captures distances from hub to hub; partial vectors capture distances from hub to arbitrary node (without traveling through another hub).

Web skeleton:

In the hubs skeleton $r_p(h)$ is computed only for $p \in H$. This can be generalized to compute $r_p(h)$ for all $p \in V$ and $h \in H$. With this kind of Web skeleton, $r_p^H(q)$ would yield an approximation of personalized Page-Rank distances for arbitrary nodes $p, q \in V$.





In round k+1:

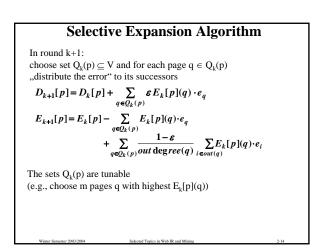
compute approximation of r_p from the round-k approximations for the successors of p

 \rightarrow substitute $D_k[p]+\hat{E_k}[p]$ invariance equation into decomposition theorem:

$$D_{k+1}[p] = \frac{1 - \varepsilon}{\text{outdegree}(p)} \sum_{i \in out(p)} D_k[i] + \varepsilon e_{i}$$

$$E_{k+1}[p] = \frac{1-\varepsilon}{\text{outdegree}(p)} \sum_{i \in out(p)} E_k[i]$$

reduces the error by factor 1- ε in each round



Repeated Squaring Algorithm

Compute iteration 2k results from iteration k results (based on Selective Expansion equations):

$$\begin{split} D_{2k}[p] = D_k[p] + &\sum_{q \in \mathcal{Q}_k(p)} E_k[p](q) \cdot D_k[q] \\ E_{2k}[p] = E_k[p] - &\sum_{q \in \mathcal{Q}_k(p)} E_k[p](q) \cdot e_q \\ &+ &\sum_{i \in out(q)} E_k[p](q) \cdot E_k[q] \end{split}$$

The sets $Q_k(p)$ are again tunable.

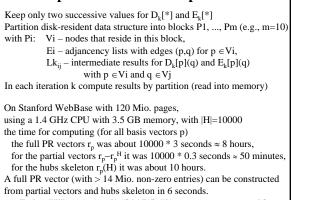
Computing Partial Vectors instead of Full Vectors

Partial vectors: specialized selected expansion algorithm by choosing $Q_0(p)=V$ and $Q_k(p)=V\text{-}H$ for $k{\geq}1$

 $\rightarrow D_k[p] + \epsilon E_k[p]$ converges to $r_p - r_p^H$

Hubs skeleton: specialized repeated squaring algorithm use results $D_k[p]$, $E_k[p]$ from partial-vector computation apply repeated squaring step using $Q_k(p) = H$

Implementation and Experiments



Literature

- Serge Abiteboul, Mihai Preda, Gregory Cobena: Adaptive on-line page importance computation, WWW Conference, 2003.
- Glen Jeh, Jennifer Widom: Scaling personalized web search, WWW Conference, 2003. (see also Technical Report 2002-12, CS Dept., Stanford University)