AI6122 Text Data Management & Analysis

Topic: Efficient Scoring

Computing cosine scores

```
CosineScore($q$)
 1  float $Scores[N] = 0$
 2  float $Length[N]$
 3  for each query term $t$
 4  do calculate $\mathrm{w}_{t,q}$ and fetch postings list for $t$
 5      for each pair($d$, $\mathrm{tf}_{t,d}$) in postings list
 6      do $Scores[d] += \mathrm{w}_{t,d} \times \mathrm{w}_{t,q}$
 7  Read the array $Length$
 8  for each $d$
 9  do $Scores[d] = Scores[d]/Length[d]$
10  return Top $K$ components of $Scores[]$
```

Efficient cosine ranking

- Find the K docs in the collection "nearest" to the query
  - K largest query-doc cosines.

- Efficient ranking:
  - Computing a single cosine efficiently.
  - Choosing the K largest cosine values efficiently from all cosine values computed.

- Can we do this without computing cosines of all documents?

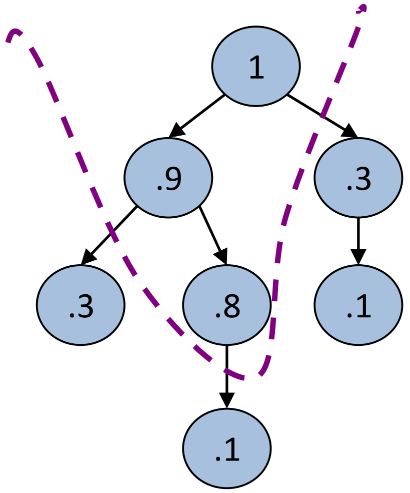
The K largest cosines: selection vs. sorting

- Typically we want to retrieve the top K docs with the highest cosine values to queries
  - We do not need to **totally order all docs** in the collection by their cosine values
- Let J = number of docs with nonzero cosines
  - We seek the K best of these J
- How to pick off docs with K highest cosines?

Use heap (data structure) for selecting top K

- Binary tree in which each node's value $>$ the values of children

- Takes $2J$ operations to construct, then each of K "winners" read off in $2\log J$ steps.
  - Assuming we have computed the cosine values for all these J documents.
  - For $J = 1M, K = 100$, this is about 10% of the cost of sorting.



http://en.wikipedia.org/wiki/Heap_(data_structure)

Bottlenecks: Cosine computation

- Primary computational bottleneck in scoring:
  - **cosine computation**

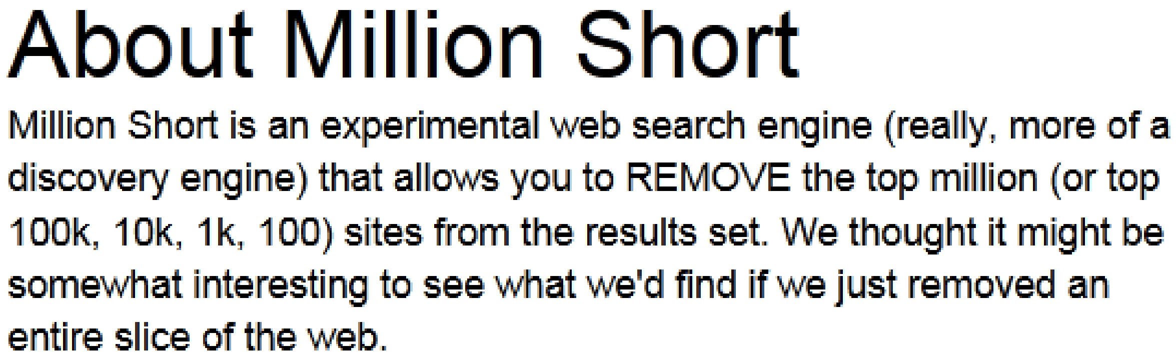
- Can we do this **without** computing cosines of all documents?
  - Yes, but may sometimes get it wrong.
  - A doc not in the top K may creep into the list of K output docs

- Is this such a bad thing?
  - What is the ideal top K?

Cosine similarity is only a proxy

- User has a task and a query formulation
  - Cosine matches docs to query
  - Thus cosine is anyway **a proxy** for user happiness

- If we get a list of K docs "close" to the top K by cosine measure, should be ok



About Million Short

Million Short is an experimental web search engine (really, more of a discovery engine) that allows you to REMOVE the top million (or top 100k, 10k, 1k, 100) sites from the results set. We thought it might be somewhat interesting to see what we'd find if we just removed an entire slice of the web.

Generic approach

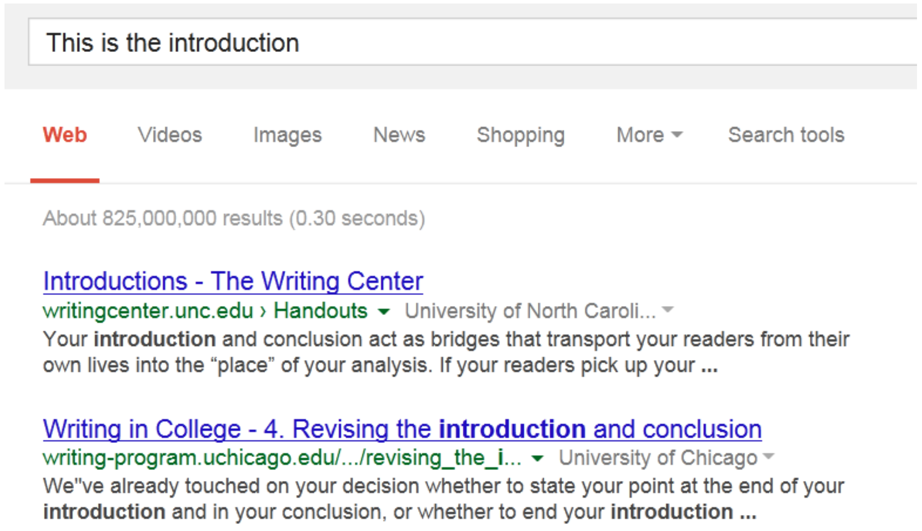
- Find a set A of contenders, with $K < |A| << N$ (all documents)
  - A may not contain the top K, but has many docs from among the top K
  - Return the top K docs in A
  - Think of A as pruning non-contenders

- The same approach is also used for other (non-cosine) scoring functions

- Will look at several schemes following this approach (in a simplified settings)
  - Assume each query term occurs only once
  - For ranking, don’t need to normalize query vector

Index elimination

- Basic algorithm cosine computation algorithm only considers docs containing at least one query term
- Take this further:
  - Only consider **high-idf** query terms
  - Only consider docs containing **many** query terms


This is the introduction

Web Videos Images News Shopping More Search tools

About 825,000,000 results (0.30 seconds)

Introductions - The Writing Center
writingcenter.unc.edu › Handouts University of North Caroli...
Your **introduction** and conclusion act as bridges that transport your readers from their own lives into the "place" of your analysis. If your readers pick up your **...**

Writing in College - 4. Revising the **introduction** and conclusion
writing-program.uchicago.edu/.../revising_the_**i**... University of Chicago
We"ve already touched on your decision whether to state your point at the end of your **introduction** and in your conclusion, or whether to end your **introduction ...**

Index elimination: high-idf query terms only

- For a query such as "*catcher in the rye*"
  - Only accumulate scores from *catcher* and *rye*
  - Intuition: ***in*** and ***the*** contribute little to the scores and so don't alter rank-ordering much

- Benefit:
  - Postings of low-idf terms have **many docs**
  - These (many) docs get eliminated from set *A* of contenders

$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

Index elimination: docs containing many query terms

- Any doc with at least one query term is a candidate for the top *K* output list
- For multi-term queries, only compute scores for docs containing several of the query terms
  - Say, at least 3 out of 4
- Easy to implement in postings traversal

Champion lists

- Pre-compute for each dictionary term t, the r docs of highest weight in t's postings
  - Call this the champion list for t
  - (aka fancy list or top docs for t)

- Note that r has to be chosen at index build time
  - Thus, it's possible that $r < K$

- At query time, only compute scores for docs in the champion list of some query term
  - Pick the K top-scoring docs from amongst these

Static quality scores

- We want top-ranking documents to be both *relevant* and ***authoritative***
  - *Relevance* is being modeled by cosine scores
  - *Authority* is typically a **query-independent** property of a document

- Examples of authority signals
  - Wikipedia among websites
  - Articles in certain newspapers
  - A paper with many citations
  - Many bitly's, diggs or del.icio.us marks
  - **PageRank**

Incorporating authority

- Assign to each document a query-independent quality score in [0,1] to each document d
  - Denote this by $g(d)$

- Consider a simple total score combining cosine relevance and authority
  - net-score(*q*,*d*) = $g(d) + cosine(q, d)$
  - Can use some other linear combination

- Now we seek the top K docs by <u>net score</u>

From query perspective: Query term proximity

- Free text queries: a set of terms typed into the query box – common on the web
  - Users prefer docs in which query terms occur within close proximity of each other

- Let w be the smallest window in a doc containing all query terms, e.g.,
  - For the query "*strained mercy*" the smallest window in the doc
  - *"The quality of <u>mercy</u> is not <u>strained</u>"* is 4 (words)

- Would like scoring function to take this into account – how?
  - Hand coded rules or learned from machine learning algorithms

Query parsers

- Free text query from user may in fact spawn one or more queries to the indexes, e.g., query *rising interest rates*
  - Run the query as a phrase query
  - If $< K$ docs contain the phrase "*rising interest rates*", run the two phrase queries "*rising interest*" and "*interest rates*"
  - If we still have $< K$ docs, run the vector space query "*rising*" "*interest*" "*rates*"
  - Rank matching docs by vector space scoring

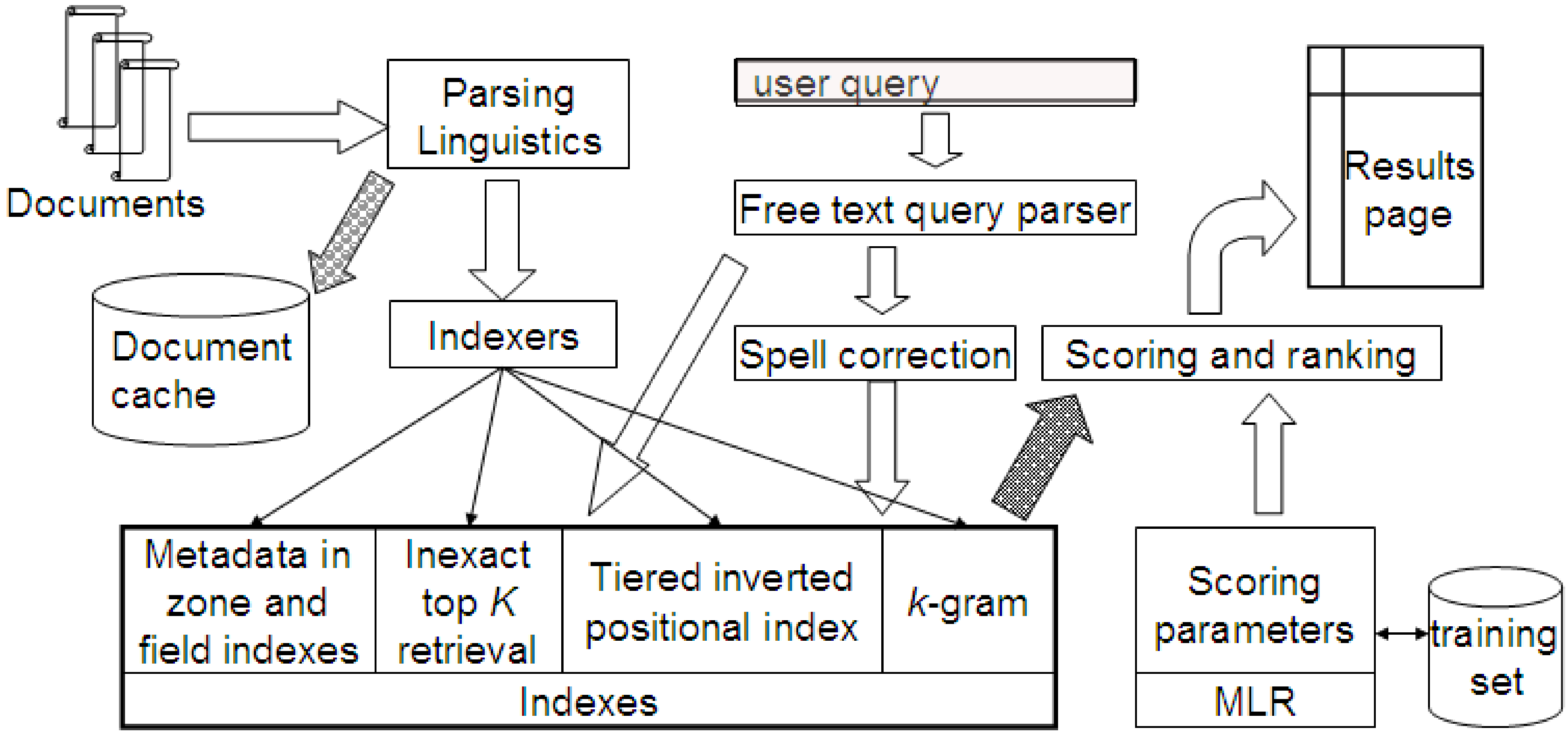
- This sequence is issued by a query parser

Aggregate scores

- We’ve seen that score functions can combine cosine, static quality, proximity, etc.

- How do we know the best combination?
  - Some applications – expert-tuned
  - Increasingly common: machine-learned

Putting it all together



Reading: IIR 7

Recall: Static quality scores

- We want top-ranking documents to be both *relevant* and ***authoritative***
  - *Relevance* is being modeled by cosine scores
  - *Authority* is typically a **query-independent** property of a document

- Examples of authority signals
  - Wikipedia among websites
  - Articles in certain newspapers
  - A paper with many citations
  - Many bitly’s, diggs or del.icio.us marks
  - (**Pagerank**)

- Next: PageRank

Origins of PageRank: Citation analysis

- Citation analysis: analysis of citations in the scientific literature.
  - Example citation: “Miller (2001) has shown that physical activity alters the metabolism of estrogens.”
  - We can view “Miller (2001)” as a hyperlink linking two scientific articles.

- One application of these “hyperlinks” in the scientific literature:
  - Measure the similarity of two articles by the overlap of other articles citing them.
  - This is called co-citation similarity.

Origins of PageRank: Citation analysis

- Another application
  - Citation frequency can be used to measure the **impact** of an article .
  - Simplest measure: Each article gets one vote – not very accurate.
- On the web
  - Citation frequency = **inlink count**
  - A high inlink count does not necessarily mean high quality, mainly because of link spam.
- Better measure
  - Weighted citation frequency or citation rank
  - An article's vote is weighted according to its citation impact.
  - This is basically PageRank.

Origins of PageRank: Summary

- We can use the same formal representation for
  - citations in the scientific literature
  - hyperlinks on the web

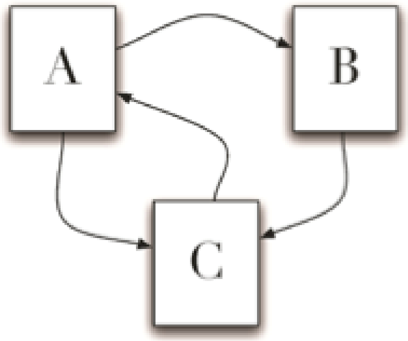
- Appropriately weighted citation frequency is an excellent measure of quality
  - Both for web pages and for scientific publications.

- Next: PageRank algorithm for computing weighted citation frequency on the web.

PageRank

- PageRank (PR) of page C:

$$PR(C) = \frac{PR(A)}{2} + \frac{PR(B)}{1}$$

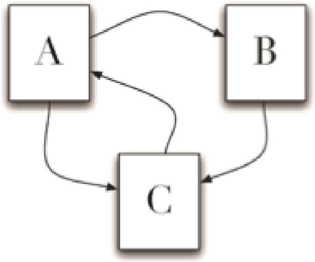


- More generally,

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L_v}$$

  – where B_u is the set of pages that point to u,
  L_v is the number of outgoing links from page v (not counting duplicate links)

PageRank



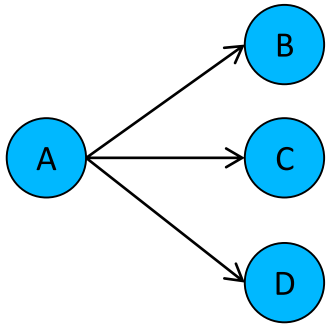
- We don't know PageRank values at start
- Assume equal values (1/3 in this case), then iterate:
  - First iteration:
    - $PR(C) = \frac{0.33}{2} + 0.33 = 0.5$, $PR(A) = 0.33$, and $PR(B) = 0.17$
  - Second iteration:
    - $PR(C) = \frac{0.33}{2} + 0.17 = 0.33$, $PR(A) = 0.5$, $PR(B) = 0.17$
  - Third iteration:
    - $PR(C) = 0.42$, $PR(A) = 0.33$, $PR(B) = 0.25$

$$\boxed{PR(C) = \frac{PR(A)}{2} + \frac{PR(B)}{1}}$$

- Converges to $PR(C) = 0.4$, $PR(A) = 0.4$, and $PR(B) = 0.2$

Model behind PageRank: Random walk

- Imagine a web surfer doing a random walk on the web
  - Start at a random page
  - At each step, go out of the current page along one of the links on that page, equiprobably



- If she visits some nodes more often than others, these are nodes with many links coming in from other frequently visited nodes
  - long-term visit rate

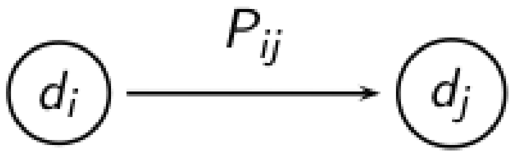
- **PageRank = long-term visit rate**
  - **Pages visited more often in the random walk are more important**

Formalization of random walk: Markov chains

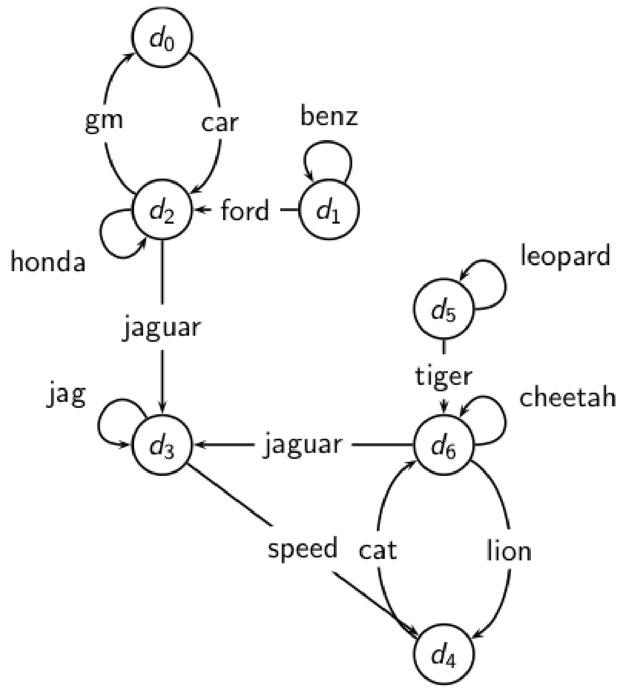
- A Markov chain consists of N states, plus an $N \times N$ transition probability matrix P. (a state = a page)
  - At each step, we are on exactly one of the pages.
  - For $1 \le i, j \le N$, the matrix entry P_{ij} tells us the probability of j being the next page, given we are currently on page i.

- Clearly, for all i,

$$\sum_{j=1}^{N} P_{ij} = 1$$



Example web graph and its link matrix



	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0	0	1	0	0	0	0
d_1	0	1	1	0	0	0	0
d_2	1	0	1	1	0	0	0
d_3	0	0	0	1	1	0	0
d_4	0	0	0	0	0	0	1
d_5	0	0	0	0	0	1	1
d_6	0	0	0	1	1	0	1

Transition probability matrix P for example

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.00	0.00	1.00	0.00	0.00	0.00	0.00
d_1	0.00	0.50	0.50	0.00	0.00	0.00	0.00
d_2	0.33	0.00	0.33	0.33	0.00	0.00	0.00
d_3	0.00	0.00	0.00	0.50	0.50	0.00	0.00
d_4	0.00	0.00	0.00	0.00	0.00	0.00	1.00
d_5	0.00	0.00	0.00	0.00	0.00	0.50	0.50
d_6	0.00	0.00	0.00	0.33	0.33	0.00	0.33

$$\sum_{j=1}^{N} P_{ij} = 1$$

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0	0	1	0	0	0	0
d_1	0	1	1	0	0	0	0
d_2	1	0	1	1	0	0	0
d_3	0	0	0	1	1	0	0
d_4	0	0	0	0	0	0	1
d_5	0	0	0	0	0	1	1
d_6	0	0	0	1	1	0	1

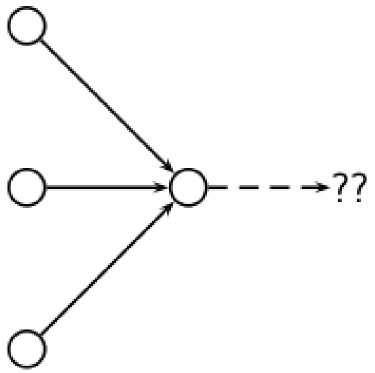
Long-term visit rate

- **PageRank = long-term visit rate**.
  - Long-term visit rate of page d is the probability that a web surfer is at page d at a given point in time.
  - What properties must hold of the web graph for the long-term visit rate to be well defined?

- The web graph **must not** contain dead ends.

Dead ends

- **The reality**: The web is full of dead ends.
- Random walk can get stuck in dead ends.



- If there are dead ends, long-term visit rates are not well-defined.

Teleporting – to get us out of dead ends

- Teleporting:
  - At a dead end, jump to a random web page with probability: $\frac{1}{N}$.
  - At a non-dead end, with probability 10%, jump to a random web page
    - to each with a probability of $\frac{0.1}{N}$.

    - 10% is a parameter, the **teleportation rate**.
    - With remaining probability (90%), go out on a random hyperlink.

- Example: if the page has 4 outgoing links:
  - randomly choose one with probability $\frac{1-0.10}{4} = 0.225$

- Note: "jumping" from dead end is independent of teleportation rate.

Result of teleporting

- With teleporting, we cannot get stuck in a dead end.

- More generally, we require that the Markov chain be ergodic.
  - Irreducibility**:** there is a path from any page to any other page.
  - Aperiodicity: the pages cannot be partitioned into sets such that all state transitions occur cyclically from one set to another.
  - For any ergodic Markov chain, there is a unique long-term visit rate for each state. This is the steady-state probability distribution

- Teleporting makes the web graph ergodic.
  - **WebGraph+teleporting** has a steady-state probability distribution.
  - Each page in the WebGraph+teleporting has a PageRank.

Formalization of "visit": Probability vector

- A probability vector $\vec{x} = (x_1, \ldots, x_N)$ tells us where the random walk is at any time point. In this example, we are now at state i:

(	0	0	0	...	1	...	0	0	0	)
	1	2	3	...	i	...	$N-2$	$N-1$	N	

- More generally: the random walk is on the page i with probability x_i. $\sum x_i = 1$

(	0.05	0.01	0.0	...	0.2	...	0.01	0.05	0.03	)
	1	2	3	...	i	...	$N-2$	$N-1$	N	

Steady state in vector notation

- The steady state in vector notation is simply a vector $\vec{\pi} = (\pi_1, \ldots, \pi_N)$ of probabilities.
  - We use $\vec{\pi}$ to distinguish it from the notation for the probability vector $\vec{x}$.

- $\vec{\pi}_i$ is the long-term visit rate (or PageRank) of page i.

- So we can think of PageRank as a very long vector
  - One entry per page.

How do we compute PageRank?

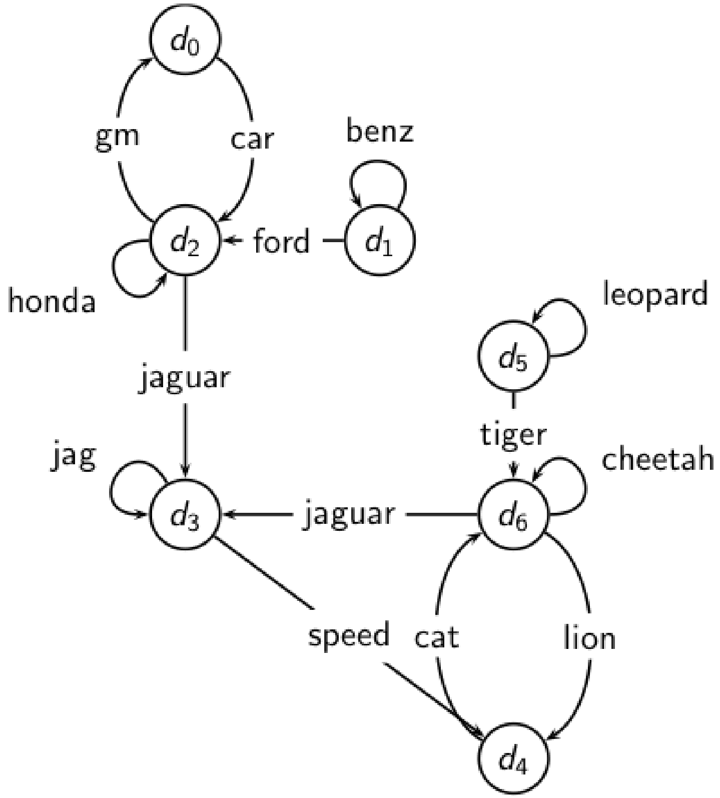
- Recall: $\vec{\pi} = (\pi_1, \ldots, \pi_N)$ is the PageRank vector, the vector of steady-state probabilities ...
  - If the distribution in this step is $\vec{x}$, then the distribution in the next step is $\vec{x}P$.
  - But $\vec{\pi}$ is the steady state!
  - So: $\vec{\pi} = \vec{\pi}P$
  - Solving this matrix equation gives us $\vec{\pi}$.

- $\vec{\pi}$ is the principal left eigenvector for P
  - i.e., the left eigenvector with the largest eigenvalue.

One way of computing the PageRank p

- Start with any distribution $\vec{x}$, e.g., uniform distribution
  - After one step, we're at $\vec{x}P$.
  - After two steps, we're at $\vec{x}P^2$.
  - After k steps, we're at $\vec{x}P^k$.
- Algorithm: multiply x by increasing powers of P until convergence.
- This is called the power method.
- Regardless of where we start, we eventually reach the steady state $\vec{\pi}$.
  - Thus: we will eventually reach the steady state.

Example web graph and transition probability matrix



	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.00	0.00	1.00	0.00	0.00	0.00	0.00
d_1	0.00	0.50	0.50	0.00	0.00	0.00	0.00
d_2	0.33	0.00	0.33	0.33	0.00	0.00	0.00
d_3	0.00	0.00	0.00	0.50	0.50	0.00	0.00
d_4	0.00	0.00	0.00	0.00	0.00	0.00	1.00
d_5	0.00	0.00	0.00	0.00	0.00	0.50	0.50
d_6	0.00	0.00	0.00	0.33	0.33	0.00	0.33

Not considering teleportation yet

Transition probability matrix, and transition matrix with teleporting

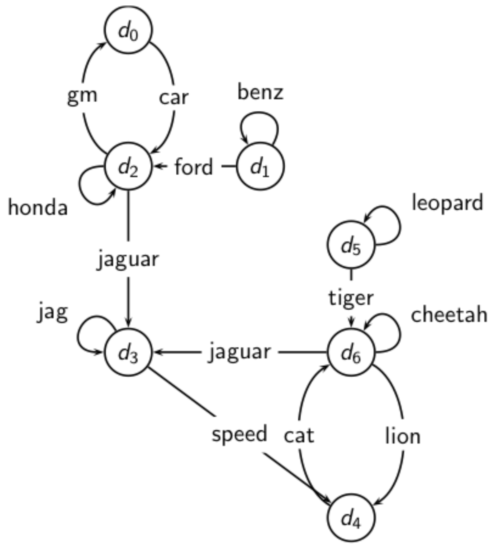
	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.00	0.00	1.00	0.00	0.00	0.00	0.00
d_1	0.00	0.50	0.50	0.00	0.00	0.00	0.00
d_2	0.33	0.00	0.33	0.33	0.00	0.00	0.00
d_3	0.00	0.00	0.00	0.50	0.50	0.00	0.00
d_4	0.00	0.00	0.00	0.00	0.00	0.00	1.00
d_5	0.00	0.00	0.00	0.00	0.00	0.50	0.50
d_6	0.00	0.00	0.00	0.33	0.33	0.00	0.33

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.02	0.02	0.88	0.02	0.02	0.02	0.02
d_1	0.02	0.45	0.45	0.02	0.02	0.02	0.02
d_2	0.31	0.02	0.31	0.31	0.02	0.02	0.02
d_3	0.02	0.02	0.02	0.45	0.45	0.02	0.02
d_4	0.02	0.02	0.02	0.02	0.02	0.02	0.88
d_5	0.02	0.02	0.02	0.02	0.02	0.45	0.45
d_6	0.02	0.02	0.02	0.31	0.31	0.02	0.31

teleportation rate = 0.14

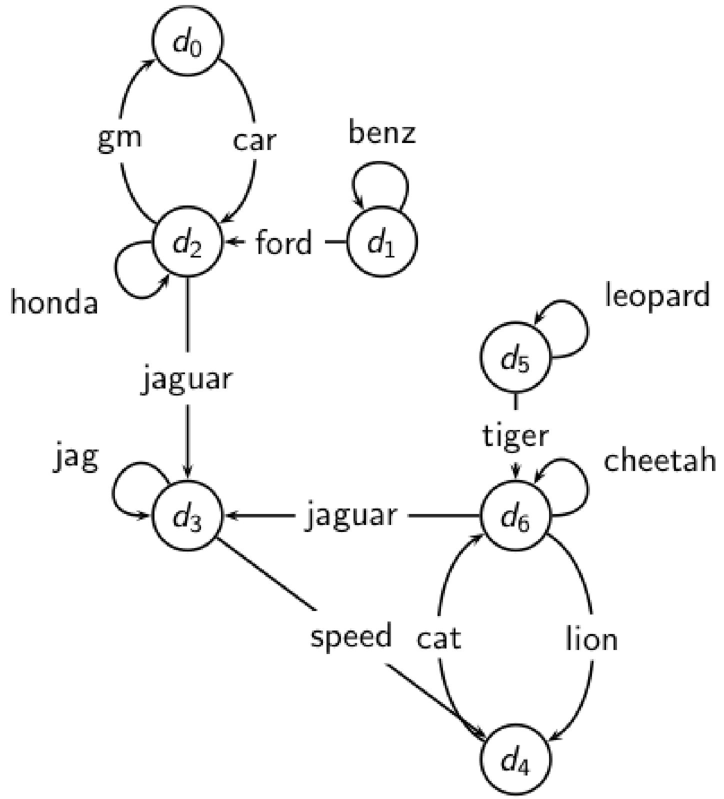
Power method vectors $\vec{x}P^k$

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.02	0.02	0.88	0.02	0.02	0.02	0.02
d_1	0.02	0.45	0.45	0.02	0.02	0.02	0.02
d_2	0.31	0.02	0.31	0.31	0.02	0.02	0.02
d_3	0.02	0.02	0.02	0.45	0.45	0.02	0.02
d_4	0.02	0.02	0.02	0.02	0.02	0.02	0.88
d_5	0.02	0.02	0.02	0.02	0.02	0.45	0.45
d_6	0.02	0.02	0.02	0.31	0.31	0.02	0.31



	$\vec{x}$	$\vec{x}P^1$	$\vec{x}P^2$	$\vec{x}P^3$	$\vec{x}P^4$	$\vec{x}P^5$	$\vec{x}P^6$	$\vec{x}P^7$	$\vec{x}P^8$	$\vec{x}P^9$	$\vec{x}P^{10}$	$\vec{x}P^{11}$	$\vec{x}P^{12}$	$\vec{x}P^{13}$
d_0	0.14	0.06	0.09	0.07	0.07	0.06	0.06	0.06	0.06	0.05	0.05	0.05	0.05	0.05
d_1	0.14	0.08	0.06	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
d_2	0.14	0.25	0.18	0.17	0.15	0.14	0.13	0.12	0.12	0.12	0.12	0.11	0.11	0.11
d_3	0.14	0.16	0.23	0.24	0.24	0.24	0.24	0.25	0.25	0.25	0.25	0.25	0.25	0.25
d_4	0.14	0.12	0.16	0.19	0.19	0.20	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21
d_5	0.14	0.08	0.06	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
d_6	0.14	0.25	0.23	0.25	0.27	0.28	0.29	0.29	0.30	0.30	0.30	0.30	0.31	0.31

Example web graph



PageRank	
d_0	0.05
d_1	0.04
d_2	0.11
d_3	0.25
d_4	0.21
d_5	0.04
d_6	0.31

PageRank summary

- Preprocessing
  - Given graph of links, build matrix P
  - Apply teleportation
  - From modified matrix, compute $\vec{\pi}$
  - πi is the PageRank of page i.

- Query processing
  - Retrieve pages satisfying the query
  - Rank them by their PageRank
  - Return re-ranked list to the user

PageRank issues

- Real surfers are not random surfers.
  - Examples of non-random surfing: back button, bookmarks, directories – and search!
  - Markov model is not the best model of surfing.
- Simple PageRank ranking produces bad results for many pages.
  - Consider the query [video service].
  - The Yahoo home page (i) has a very high PageRank and (ii) contains both video and service.
  - If we rank all Boolean hits according to PageRank, then the Yahoo home page would be top-ranked.
- In practice:
  - Rank according to weighted combination of raw text match, anchor text match, PageRank & other factors.

How important is PageRank?

- Frequent claim:
  - PageRank is the most important component of web ranking.

- The reality:
  - There are several components that are at least as important: e.g., anchor text, phrases, proximity, tiered indexes ...
  - Rumor has it that PageRank in his original form (as presented here) now has a negligible impact on ranking!
  - However, variants of a page's PageRank are still an essential part of ranking.

- However: PageRank or RandomWalk remains a key algorithm in many areas