L21 -- Page Rank
[Jeff Phillips - Utah - Data Mining]

Search Engine == inverted index

web page = {topics, words} = {terms}
index {terms} -> <webpages>

Pre-Google:

spider = program that randomly visited webpages
(it "crawled the web")
on each page it compiled important "terms"
and scored how relevant to each "term"

index = ranks webpages for each term
"magic"
(fast forward to now, still "magic")

search [____term____]
-> top 10 webpages

**term spam
- repeat the work "movie" 1000 times
- find high-ranked pages, copy entire page into html
"trick, do in same color as background, and very small"

PageRank:
IDEA 1:
pages are only important if **linked to** from other pages

p1 has {terms1}
p1 links to p2
p2 has {terms2}
p2 gets high score for term t if
t in terms1 intersect terms2

--> even better if hyper-text has "t"

Easy for spammer to put terms on his page
Hard for spammer to put terms on page linking to his page
(Well not that hard: spam farm = many pages w/ {terms} linking to page)

IDEA 2:
"Random Surfer Model"
and how to defeat "spam farms"

Internet is big (directed) graph G=(V,E)
V = webpages
E = (directed) links from one page to another

random surfer:
+ starts at one page
+ clicks random link on that page

defines Markov chain (P,q)
where converged-to distribution q_* = P^* q
gives importance q_*[v] of page v in V

INDEX (term) = top(k, f(page,term))
f(page,term) = MAGIC(q_*[page]*term(page) + SUM {q_*[link-to-page]*term(link-to-page)})

How to compute q_*
** don't compute P^n (why next lecture)

```
 compute q_1 = P q
         q_2 = P q_1
           ...
         q_n = P q_{n+1}
```

for n = between 50 and 75

Are we done?

Web graph is not ergodic
+ may not be connected
+ has transient nodes
(might be cyclic, but thats not as big a deal)

Structure of Web:

```
 Big SCC = Strongly Connected Component
 IN      = in components to SCC
 OUT     = out components of SCC (cannot link back to SCC)
 T-OUT   = tendrils out of IN
 T-IN    = tendrils into OUT
 TUBE    = paths from IN to OUT
 DISC    = disconnected components
```

what happens to OUT: all probability accumulates
"spider traps"

Solution:
"taxation" : each random web-surfer has a chance of going to a TOTALLY random page
1-beta = fraction of random restarts (about beta = 0.85)
--> graph totally connected
--> no transient nodes
--> not cyclic

--> no spider traps
--> mixes faster

SPAM FARMS:
spammers control some large number of pages
(how can these pages trick PageRank?)
1: own pages
2: corrupted pages
e.g. "blog comments"

target page
corrupted pages -> target
own pages <--> target

own pages accumulate "taxation moves"
own pages keep rank of target, goes to own pages, and comes back

----

HOW DO WE DEFEAT SPAM FARMS?
Search for spam farm structure, and eliminate/black-ball it
- but structure can be changed + modified...

TrustRank:
+certain pages are more trust-worthy
YES: wikipedia, .edu .mil .gov pages, main Amazon pages, VERY high PageRank
NO: blogs, pages with many comments
--> high-trust pages get more weight in PageRank (more random restarts?)

Spam Mass:
page has PageRank r, TrustRank t
s = (r-t)/r
IF s small, negative, then NOT Spam
IF s large, then likely Spam