Large scale graph processing Random walks, PageRank,Personalized pagerank

James G. Shanahan [2]

Assistants: Liang Dai[1, 3]

[1]NativeX, [2]iSchool UC Berkeley, CA, [3]UC Santa Cruz

EMAIL: James_DOT_Shanahan_AT_gmail_DOT_com

Live Session #9

March 8, 2016

Conferences where you can learn more

- **Have a look at the following webpage for ideas:** http://www.kdnuggets.com/meetings/
- **Here is a short list to get this discussion underway:**
  - KDD (in San Francisco this year)
  - NIPS
  - Spark Summit Series
  - WWW
  - http://cvpr2016.thecvf.com/ (Computer vision conference)
  - MLConf

13 Lectures, 1 Midterm, End of term exam, plus weekly homework and projects

Semester Week	Week in 2016	Monday	Notes	
1	Week 3	1/11/16		
2	Week 4	1/18/16		
3	Week 5	1/25/16		
4	Week 6	2/1/16		
5	Week 7	2/8/16		
6	Week 8	2/15/16		
7	Week 9	2/22/16		
8	Week 10	2/29/16	Mid Term Exam	
9	Week 11	3/7/16		
10	Week 12	3/14/16		
Spring Break	Week 13	3/21/16	Spring Break	
11	Week 14	3/28/16		
12	Week 15	4/4/16		
13	Week 16	4/11/16		
14	Week 17	4/18/16		
15	Week 18	4/25/16	End of term	4/29/15
	Week 19	5/2/16		
		5/18/2016 (Wednesday)		Grades Due

Schedule and HW Schedule

- **Spring Break week (Mar 22-25). No Class!**
  - During this week there will be no class and there is no planned office hours. The academic calendar can be found the I School website here:
  - https://www.ischool.berkeley.edu/intranet/students/mids/calendar
- **Homework Schedule:**
  - Given the spring break timing, and the fact that some of you requested extra time for HW7, I propose the following homework schedule:
  - **HW7 is due this Saturday (March 12) at midday**
  - **HW9 is due Saturday (March 19) at midday**
  - **HW10 is due Tuesday (April 5) at 8AM**

Important Links for Week 9

- **Live session Slides**
  - https://www.dropbox.com/s/lt9ke46t7c11aiw/MIDS-Week-09-Live-Session-PageRank-2016-03-08.pdf?dl=0
- **Instructions for Peer grading of homework HW**
  - https://www.dropbox.com/s/97m31frthj4ac28/HOMEWORK%20GRADING%20INSTRUCTIONS%20for%20MIDS%20MLS?dl=0
- **Homework HW Folder Questions + Data**
  - HW is a group oriented homework (Check out the Data sub folder)
  - https://www.dropbox.com/s/wp4cz1e0bif1k76/HW9-Assignment.txt?dl=0
  - https://www.dropbox.com/sh/2c0k5adwz36lkcw/AAAAKsjQfF9uHfv-X9mCqr9wa?dl=0
- **Team assignments**
  - https://docs.google.com/spreadsheets/d/1ncFQl5Tovn-16slD8mYjP_nzMTPSfiGeLLzW8v_sMjg/edit?usp=sharing
- **Please submit your homeworks (one per team) going forward via this form (and not thru the ISVC):**
  - https://docs.google.com/forms/d/1ZOr9RnIe_A06AcZDB6K1mJN4vrLeSmS2PD6Xm3eOiis/viewform?usp=send_form

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Mid term; Feedback/Evaluation
  - Homework HW6, HW7, HW9
  - AWS: no access
  - Async lecture recap plus Q&A (PageRank)
  - Contextual advertising
  - Text as a graph: TextRank
    - Keyword extraction (from text/target pages)
    - Text Summarization
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

MidTerm Week 8

- ..

```
===Order inversion===

MT1. Suppose you wish to write a MapReduce job that creates
normalized word co-occurrence data form a large text.
To ensure that all (potentially many) reducers
receive appropriate normalization factors (denominators)
in the correct order in their input streams
(so as to minimize memory overhead),
the mapper should emit according to which pattern:

(a) emit (word,*) count
(b) emit (*,word) count
(c) There is no need to use  order inversion here
(d) None of the above

===Apriori principle===

MT2. When searching for frequent itemsets with the Apriori algorithm
(using a threshold, N), the Apriori principle allows us to avoid
tracking the occurrences of the itemset {A,B,C} provided

(a) all subsets of {A,B,C} occur less than N times.
(b) any subset of {A,B,C} occurs less than N times.
(c) any pair of {A,B,C} occurs less than N times.
(d) All of the above

===Bayesian document classification===

MT3. When building a Bayesian document classifier, Laplace smoothing serves what purpose?

(a) It allows you to use your training data as your validation data.
(b) It prevents zero-products in the posterior distribution.
(c) It accounts for words that were missed by regular expressions.
(d) None of the above

===Bias-variance tradeoff===

MT4. By increasing the complexity of a model regressed for on some samples of data,
it is likely that the ensemble will exhibit which of the following?

(a) Increased variance and bias
(b) Increased variance and decreased bias
(c) Decreased variance and bias
(d) Decreased variance and increased bias
```

Audience Participation

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Mid term; Feedback/Evaluation
  - Homework HW6, HW7, HW9
  - AWS: no access
  - Async lecture recap plus Q&A (PageRank)
  - Contextual advertising
  - Text as graph: TextRank
    - Keyword extraction (from text/target pages)
    - Text Summarization
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Mid term; Feedback/Evaluation
  - Homework HW6, HW7, HW9
  - AWS: no access
  - Async lecture recap plus Q&A (PageRank)
  - Contextual advertising
  - Text as graph: TextRank
    - Keyword extraction (from text/target pages)
    - Text Summarization
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

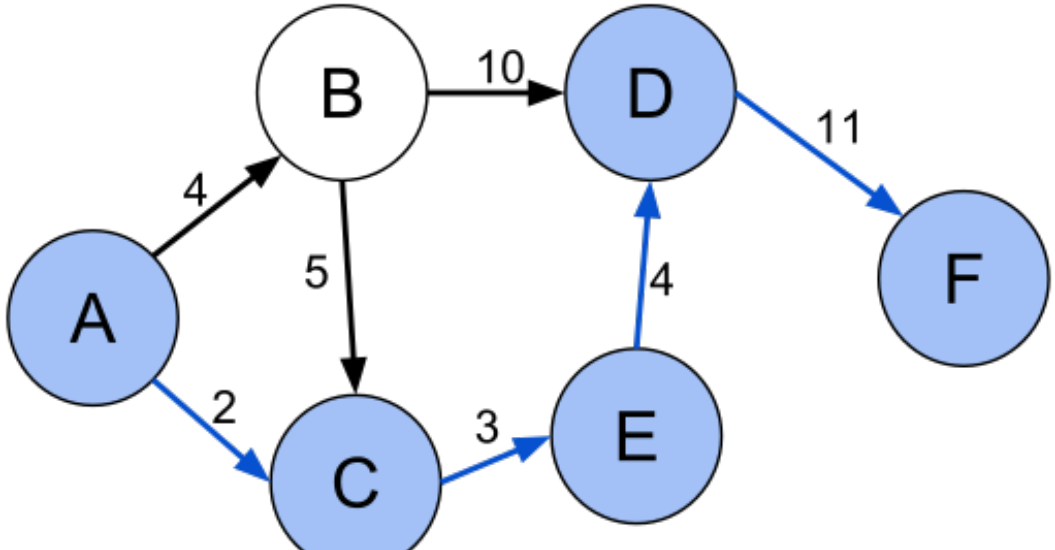
Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Mid term; Feedback/Evaluation
  - Homework HW6, HW7, HW9
  - AWS: no access
  - Async lecture recap plus Q&A (PageRank)
  - Contextual advertising
  - Text as graph: TextRank
    - Keyword extraction (from text/target pages)
    - Text Summarization
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

Week 7: Graphs and MapReduce

- **Graph algorithms typically involve:**
  - Performing computation at each node
  - Processing node-specific data, edge-specific data, and link structure
  - Traversing the graph in some manner
- **Key questions:**
  - How do you represent graph data in MapReduce?
  - How do you process a graph in stateless MapReduce?

For maximum parallelism, you need the Maps and Reduces to be stateless, to not depend on any data generated in the same MapReduce job.
You cannot control the order in which the maps run, or the reductions.



Single source: smallest number of edges that must be traversed in order to get to every vertex

- **Shortest is relative**
  - Unweighted
  - Weighted (sum of edge weights on path from source node to target node)

- **Graph Traversal**
  - Unweighted single-source shortest path
  - BFS, DFS
- **Dijkstra algorithm**
  - Weighted single-source shortest path
  - Modified BFS (with weights+priority queue)

Large scale graph processing Random walks, PageRank,Personalized pagerank

James G. Shanahan

EMAIL: James_DOT_Shanahan_AT_gmail_DOT_com

Largescale Machine Learning, MIDS

UC Berkeley

Lecture 9

Summary: from random walks to pagerank

- **View pages as states, and webGraph as a transition matrix→ A Markov process who steady state distribution tells about which pages we spend a lot of time on.**
- **This is a proxy for popularity and is a very discriminating feature for in search**
- **Making some minor adjustments to the transition matrix**
  - (stochasticity adjustment to deal with dangling nodes (sinks))
  - Primitivity adjustment via teleportation
- **We can then use the power iteration method**
  - (first eigenvector, all positive values; eigenvalue ==1)
- **Personalized pagerank**

- **Is the pageRank algorithm embarrassingly parallel?**

- **Is the pageRank algorithm embarrassingly parallel?**

- **Kinda only:**

Library of Congress: 200TB

- Assume 8MB per book at 26 million books, which is closer to 198 TB.

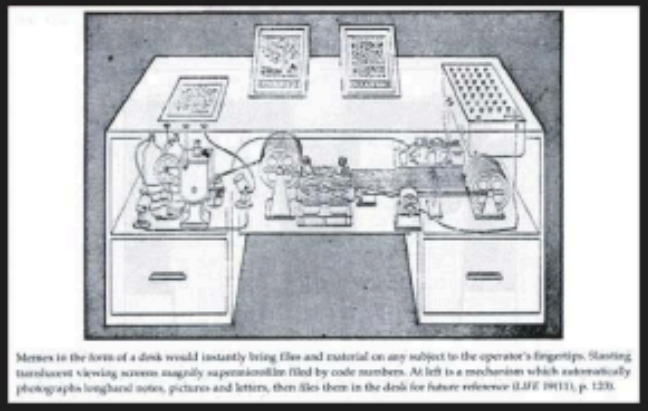
- 26,000,000 * 8 / 1,024 / 1,024 = 198.36 TB, not 208 TB, where lazy division was used (1,000 instead of 1,024).

Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?

- It's huge
  - 10^{12} pages with 10^6 (average webpage sizetoday) = 10^{18} Exabyte; 1M laptops
  - 20 times size of Library of Congress print collection
  - LOC 200TB (10^{12})
  - Deep Web - 550 billion pages
- It's dynamic.
  - Google crawls 20 billion sites a day
  - content changes: 40% of pages change in a week, 23% of .com change daily
  - size changes: billions of pages added each year
- It's self-organized.
  - no standards, review process, formats
  - errors, falsehoods, link rot, and spammers!
- Ah, but it's *hyperlinked* !
  - Vannevar Bush's 1945 memex



Memex in the form of a desk would instantly bring files and material on any subject to the operator's fingertips. Slanting translucent viewing screens magnify supermicrofilm filed by code numbers. At left is a mechanism which automatically photographs longhand notes, pictures and letters, then files them in the desk for future reference (LIFE 19(11), p. 123).

PageRank: how to visualize?

Pagerank

Google.com 10^{-9}

…

eBay.com 10^{-10}

…

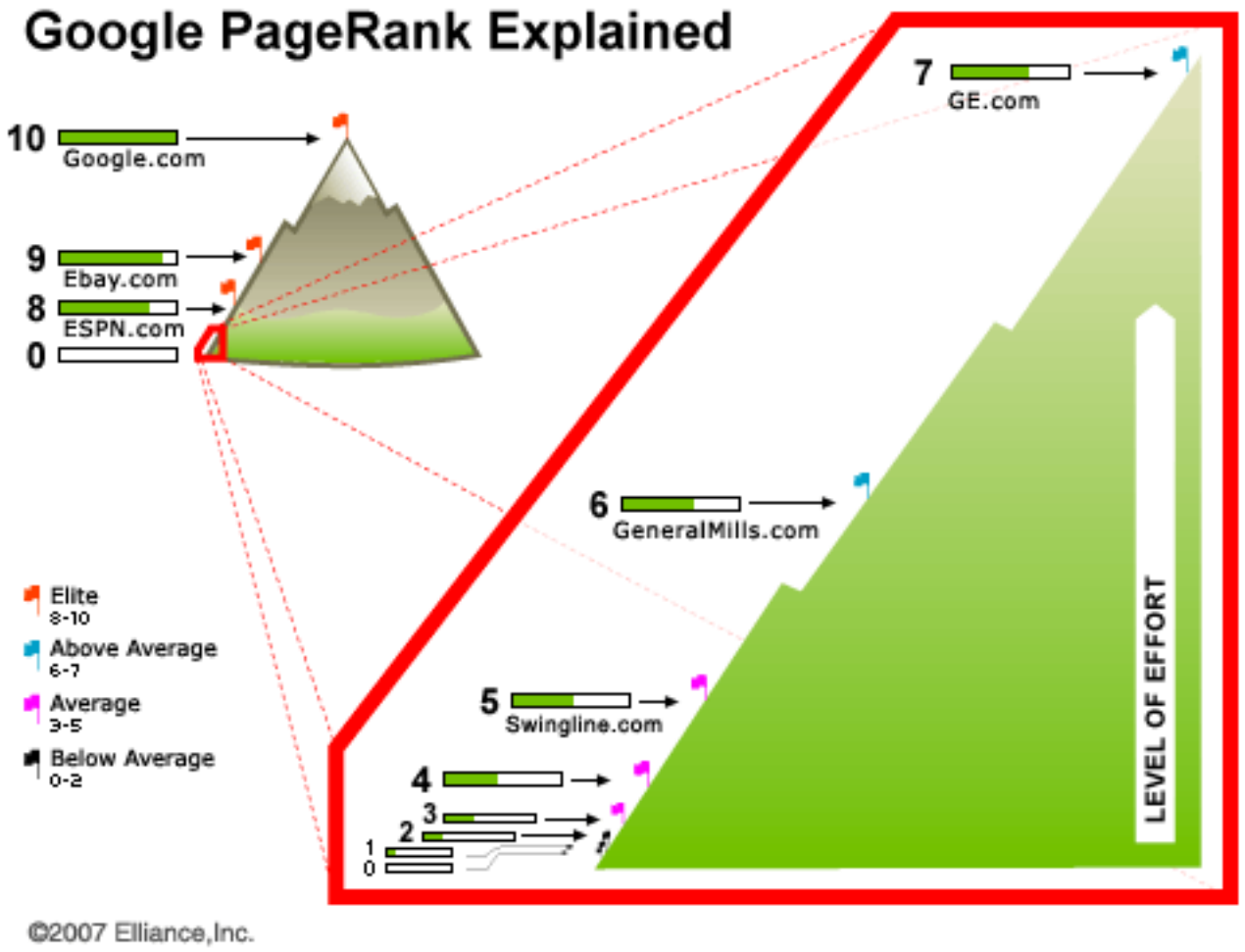
ESPN.com 10^{-11}

…

GE.com 10^{-12}

….

GeneralMills.com



http://www.hobo-web.co.uk/google-pr-update/

PageRank in percentile buckets?

Pagerank

Google.com 10^{-9}

…

eBay.com 10^{-10}

…

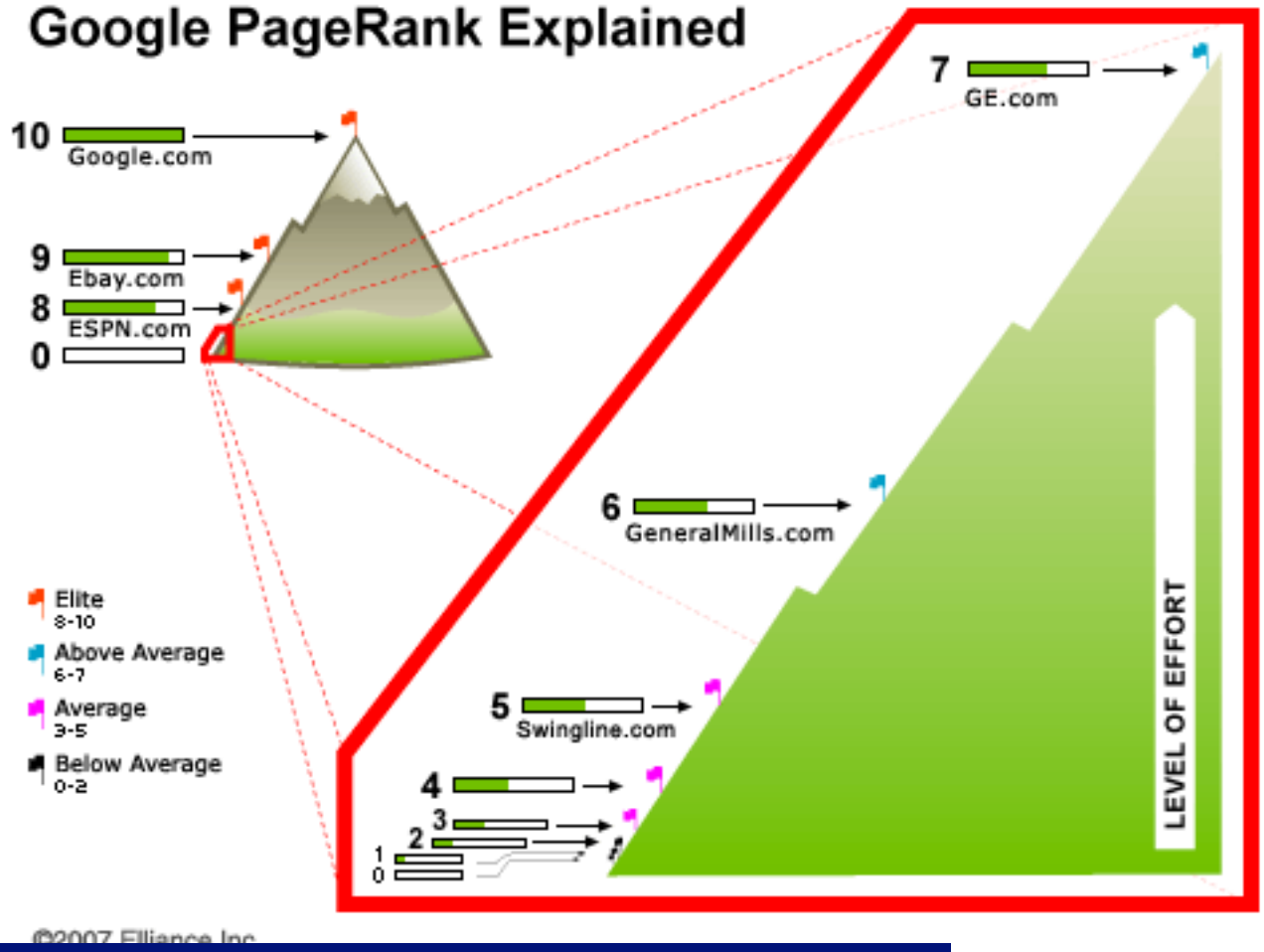
ESPN.com 10^{-11}

…

GE 10^{-12}

….

GeneralMills.com



http://www.hobo-web.co.uk/google-pr-update/

Library of Congress: 200TB

- Assume 8MB per book at 26 million books, which is closer to 198 TB.

- 26,000,000 * 8 / 1,024 / 1,024 = 198.36 TB, not 208 TB, where lazy division was used (1,000 instead of 1,024).

Inverted Index Questions

k-step transition matrix, 179

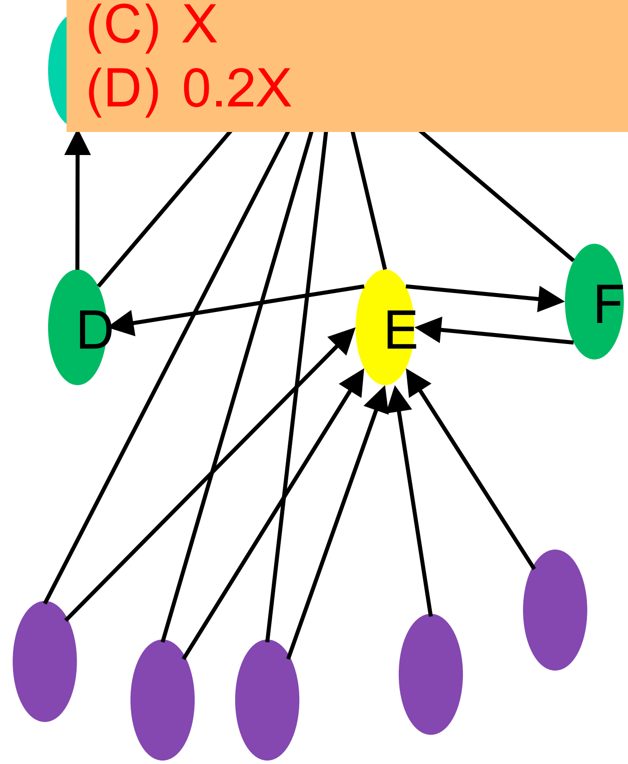
a vector, 37, 38, 75, 80
A9, 142
absolute error, 104
absorbing Markov chains, 185
absorbing states, 185
accuracy, 79–80
adaptive PageRank method, 89–90
Adar, Eytan, 146
adjacency list, 77
adjacency matrix, 33, 76, 116, 132, 169
advertising, 45
aggregated chain, 197
aggregated chains, 195
aggregated transition matrix, 105
aggregated transition probability, 197
aggregation, 94–97
  approximate, 102–104
  exact, 104–105
  exact vs. approximate, 105–107
  iterative, 107–109
    partition, 109–112
aggregation in Markov chains, 197
aggregation theorem, 105
Aitken extrapolation, 91
Alexa traffic ranking, 138
algebraic multiplicity, 157
algorithm
  PageRank, 40
  Aitken extrapolation, 92
  dangling node PageRank, 82, 83
  HITS, 116
  iterative aggregation updating, 108
  personalized PageRank power method, 49
  quadratic extrapolation, 93
  query-independent HITS, 124
α parameter, 37, 38, 41, 47–48
Amazon's traffic rank, 142
anchor text, 48, 54, 201
Ando, Albert, 110
aperiodic, 36, 133
aperiodic Markov chain, 176
Application Programming Interface (API), 65, 73, 97
approximate aggregation, 102–104
arc, 201
Arrow, Kenneth, 136
asymptotic convergence rate, 165

asymptotic rate of convergence, 41, 47, 101, 119, 125
Atlas of Cyberspace, 27
authority, 29, 201
authority Markov chain, 132
authority matrix, 117, 201
authority score, 115, 201
authority vector, 201

Babbage, Charles, 75
back button, 84–86
BadRank, 141
Barabasi, Albert-Laszlo, 30
Berry, Michael, 7
bibliometrics, 32, 123
bipartite undirected graph, 131
BlockRank, 94–97, 102
blog, 55, 144–146, 201
Boldi, Paolo, 79
Boolean model, 5–6, 201
bounce back, 84–86
bowtie structure, 134
Brezinski, Claude, 92
Brin, Sergey, 25, 205
Browne, Murray, 7
[illegible], 3, 10
[illegible]
canonical form, reducible matrix, 182
censored chain, 104
[illegible]ed chains, 194
[illegible]ed distribution, 104, 195
[illegible]ed Markov chain, 194
[illegible]ship, 146–147
Cesàro sequence, 162
Cesàro summability, stochastic matrix, 182
characteristic polynomial, 120, 156
Chebyshev extrapolation, 92
Chien, Steve, 102
cloaking, 44
clustering search results, 142–143
co-citation, 123, 201
co-reference, 123, 201
Collatz–Wielandt formula, 168, 172
complex networks, 30
compressed matrix storage, 76
condition number, 59, 71, 155
Condorcet, 136
connected components, 127, 133

1 Seconds CR!!

10^{11} webpages
10^{4} bytes per page
10^{15} bytes (1 Petabyte)
How big is the inverted index?
(A) 10X
(B) 2X
(C) X
(D) 0.2X



- **Synchronized carriage return**

Inverted Index Questions

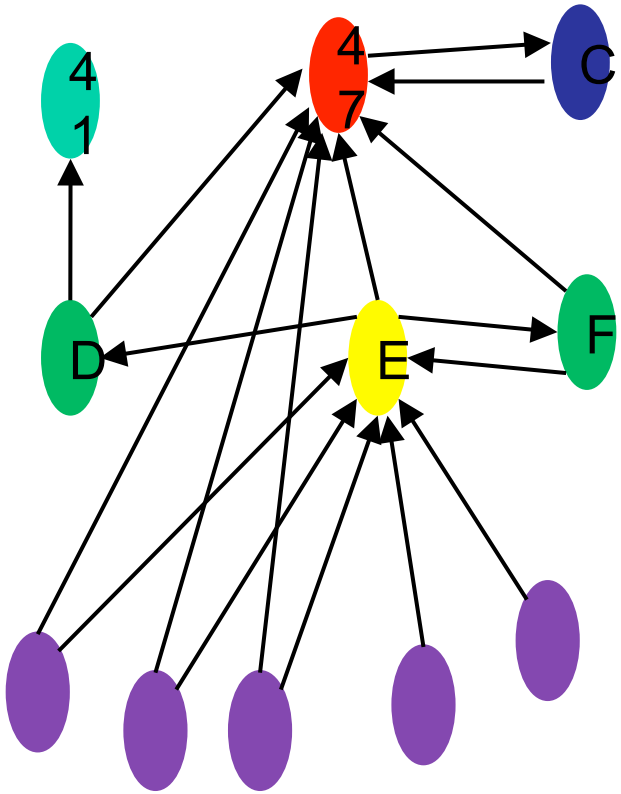
k-step transition matrix, 179

n vector, 37, 38, 75, 80
A9, 142
absolute error, 104
absorbing Markov chains, 185
absorbing states, 185
accuracy, 79–80
adaptive PageRank method, 89–90
Adar, Eytan, 146
adjacency list, 77
adjacency matrix, 33, 76, 116, 132, 169
advertising, 45
aggregated chain, 197
aggregated chains, 195
aggregated transition matrix, 105
aggregated transition probability, 197
aggregation, 94–97
  approximate, 102–104
  exact, 104–105
  exact vs. approximate, 105–107
  iterative, 107–109
    partition, 109–112
aggregation in Markov chains, 197
aggregation theorem, 105
Aitken extrapolation, 91
Alexa traffic ranking, 138
algebraic multiplicity, 157
algorithm
  PageRank, 40
  Aitken extrapolation, 92
  dangling node PageRank, 82, 83
  HITS, 116
  iterative aggregation updating, 108
  personalized PageRank power method, 49
  quadratic extrapolation, 93
  query-independent HITS, 124
α parameter, 37, 38, 41, 47–48
Amazon's traffic rank, 142
anchor text, 48, 54, 201
Ando, Albert, 110
aperiodic, 36, 133
aperiodic Markov chain, 176
Application Programming Interface (API), 65, 73, 97
approximate aggregation, 102–104
arc, 201
Arrow, Kenneth, 136
asymptotic convergence rate, 165
asymptotic rate of convergence, 41, 47, 101, 119, 125
Atlas of Cyberspace, 27
authority, 29, 201
authority Markov chain, 132
authority matrix, 117, 201
authority score, 115, 201
authority vector, 201

Babbage, Charles, 75
back button, 84–86
BadRank, 141
Barabasi, Albert-Laszlo, 30
Berry, Michael, 7
bibliometrics, 32, 123
bipartite undirected graph, 131
BlockRank, 94–97, 102
blog, 55, 144–146, 201
Boldi, Paolo, 79
Boolean model, 5–6, 201
bounce back, 84–86
bowtie structure, 134
Brezinski, Claude, 92
Brin, Sergey, 25, 205
Browne, Murray, 7
Bush, Vannevar, 3, 10

Campbell, Lord John, 23
canonical form, reducible matrix, 182
censored chain, 104
censored chains, 194
censored distribution, 104, 195
censored Markov chain, 194
censorship, 146–147
Cesàro sequence, 162
Cesàro summability, stochastic matrix, 182
characteristic polynomial, 120, 156
Chebyshev extrapolation, 92
Chien, Steve, 102
cloaking, 44
clustering search results, 142–143
co-citation, 123, 201
co-reference, 123, 201
Collatz–Wielandt formula, 168, 172
complex networks, 30
compressed matrix storage, 76
condition number, 59, 71, 155
Condorcet, 136
connected components, 127, 133

10^{11} webpages
10^{4} bytes per page
10^{15} bytes (1 Petabyte)
How big is the inverted index?
20-40% of the of the original corpus
~$2*10^{14}$



Inverted Index Questions

k-step transition matrix, 179
a vector, 37, 38, 75, 80
A9, 142
absolute error, 104
absorbing Markov chains, 185
absorbing states, 185
accuracy, 79–80
adaptive PageRank method, 89–
Adar, Eytan, 146
adjacency list, 77
adjacency matrix, 33, 76, 116, 1
advertising, 45
aggregated chain, 197
aggregated chains, 195
aggregated transition matrix, 105
aggregated transition probability, 197
aggregation, 94–97
  approximate, 102–104
  exact, 104–105
  exact vs. approximate, 105–107
  iterative, 107–109
    partition, 109–112
aggregation in Markov chains, 197
aggregation theorem, 105
Aitken extrapolation, 91
Alexa traffic ranking, 138
algebraic multiplicity, 157
algorithm
  PageRank, 40
  Aitken extrapolation, 92
  dangling node PageRank, 82, 83
  HITS, 116
  iterative aggregation updating, 108
  personalized PageRank power method, 49
  quadratic extrapolation, 93
  query-independent HITS, 124
α parameter, 37, 38, 41, 47–48
Amazon's traffic rank, 142
anchor text, 48, 54, 201
Ando, Albert, 110
aperiodic, 36, 133
aperiodic Markov chain, 176
Application Programming Interface (API), 65, 73, 97
approximate aggregation, 102–104
arc, 201
Arrow, Kenneth, 136
asymptotic convergence rate, 165
asymptotic rate of convergence, 41, 47, 101, 119, 125
Atlas of Cyberspace, 27
authority, 29, 201
authority Markov chain, 132
authority matrix, 117, 201
authority score, 115, 201
bipartite undirected graph, 131
BlockRank, 94–97, 102
blog, 55, 144–146, 201
Boldi, Paolo, 79
Boolean model, 5–6, 201
bounce back, 84–86
bowtie structure, 134
Brezinski, Claude, 92
Brin, Sergey, 25, 205
Browne, Murray, 7
Bush, Vannevar, 3, 10

Campbell, Lord John, 23
canonical form, reducible matrix, 182
censored chain, 104
censored chains, 194
censored distribution, 104, 195
censored Markov chain, 194
censorship, 146–147
Cesàro sequence, 162
Cesàro summability, stochastic matrix, 182
characteristic polynomial, 120, 156
Chebyshev extrapolation, 92
Chien, Steve, 102
cloaking, 44
clustering search results, 142–143
co-citation, 123, 201
co-reference, 123, 201
Collatz–Wielandt formula, 168, 172
complex networks, 30
compressed matrix storage, 76
condition number, 59, 71, 155
Condorcet, 136
connected components, 127, 133

Postings lists: delta encoding

A vector {37, 38, 75, 80}

{37 1, 37, 5} #delta

10^{11} webpages

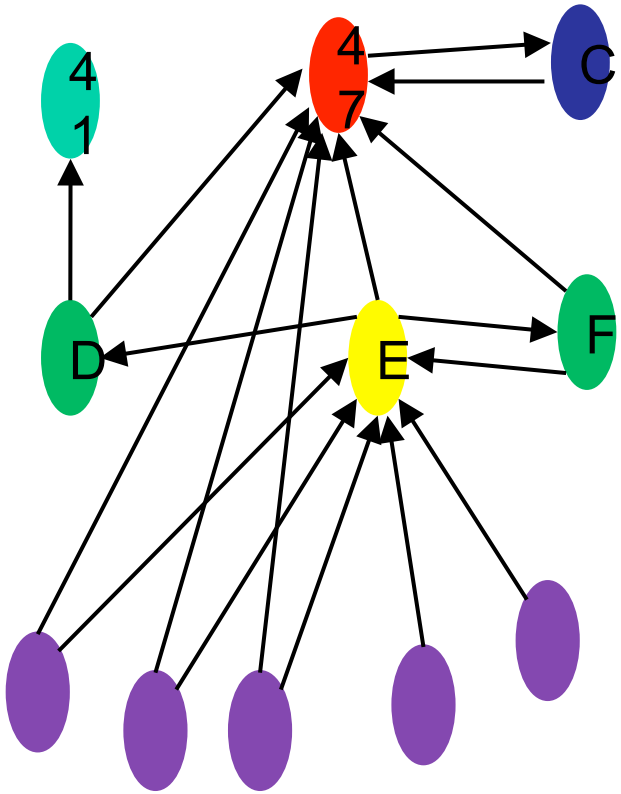
10^{4} bytes per page

10^{15} bytes (1 Petabyte)

How big is the inverted index?

20-40% of the of the original corpus

~$2*10^{14}$



Question

- **How could one organize the postings list to make it efficient to process at query time?**

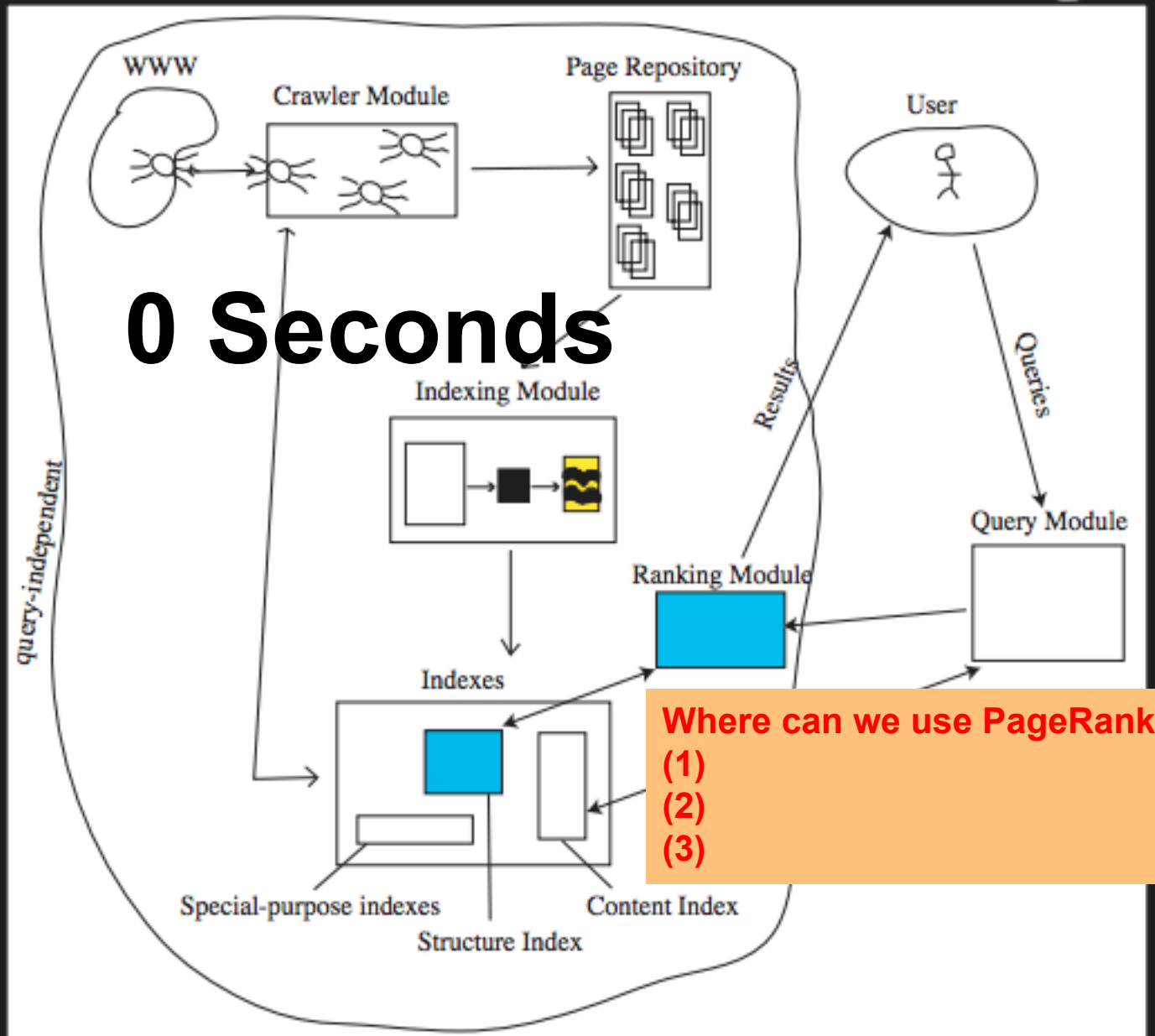
Query for Hadoop

- **A sample Query : "Hadoop", 100 Million entries in the it posting list.**
- **Load the postings list for Hadoop into memory**
  - Hadoop: 1, 79, 89, 100, 10000, 33333,….
  - How to assign page IDs?
    - Random?
    - Or??

Query for Hadoop

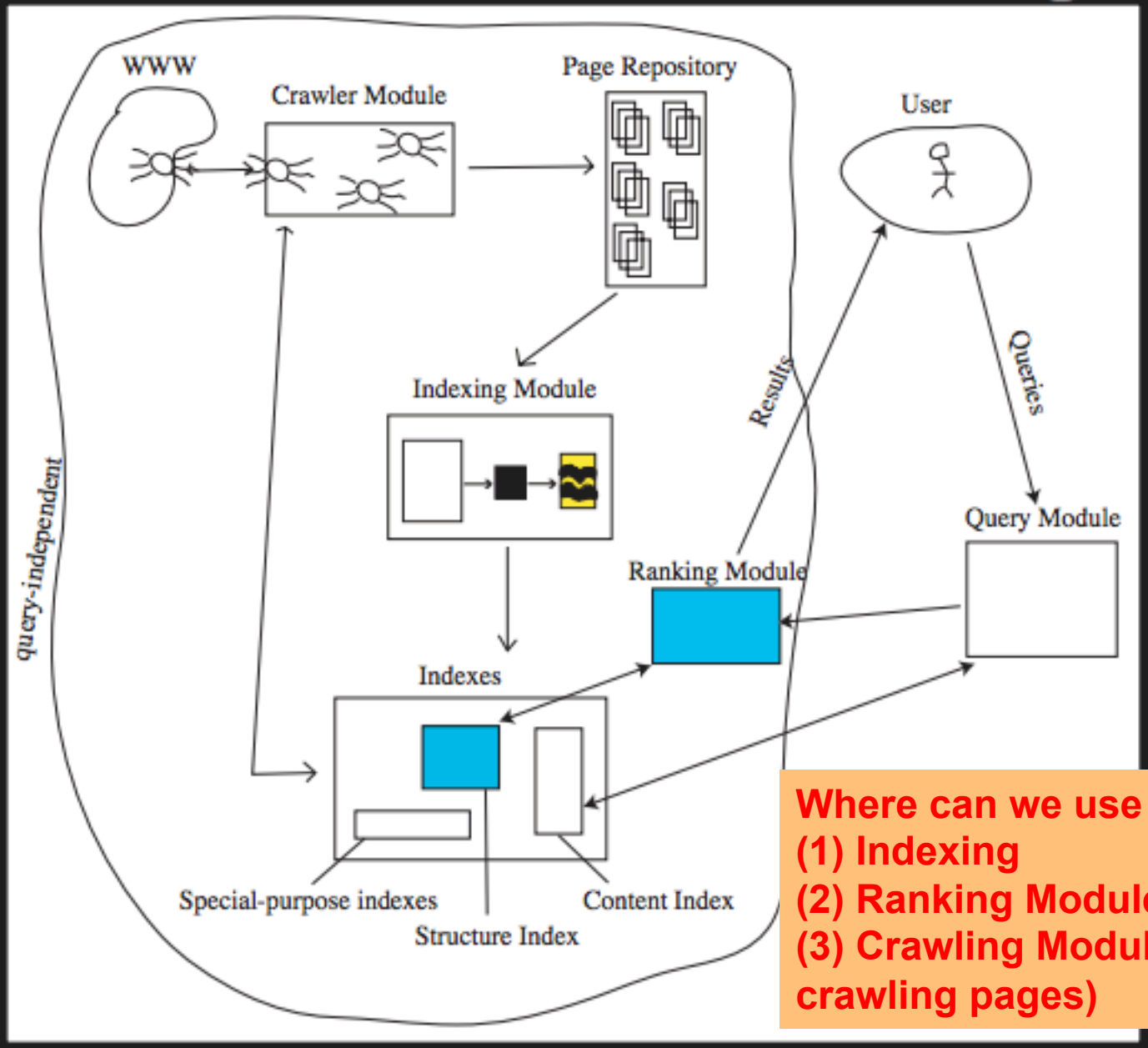
- **Query : Hadoop**
- **Load the postings list for Hadoop into memory**
  - Hadoop: 1, 79, 89, 100, 10000, 33333,….
  - How to assign page IDs?
    - Random and sort?
    - Assign IDs based on PageRank Order
      - BECAUSE: put important pages in the front of postings list!

Elements of a Web Search Engine



Where can we use PageRank value here?
(1)
(2)
(3)

Elements of a Web Search Engine



Where can we use PageRank he
(1) Indexing
(2) Ranking Module (LeToR)
(3) Crawling Module (priority for crawling pages)

- **Crawling challenges**
  - Dead end nodes

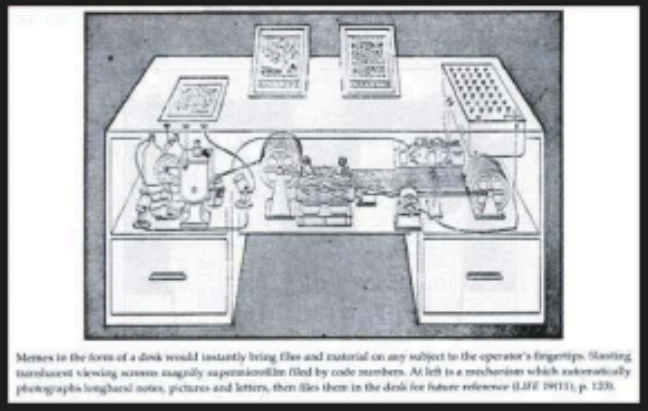
- **Use pageRank to prioritize my crawl**

Web Information Retrieval

IR before the Web = traditional IR
IR on the Web = web IR

How is the Web different from other document collections?

- It's huge
  - 10^{12} pages with 10^6 (average webpage sizetoday) = 10^{18} Exabyte; 1M laptops
  - 20 times size of Library of Congress print collection
  - LOC 200TB (10^{12})
  - Deep Web - 550 billion pages
- It's dynamic.
  - Google crawls 20 billion sites a day
  - content changes: 40% of pages change in a week, 23% of .com change daily
  - size changes: billions of pages added each year
- It's self-organized.
  - no standards, review process, formats
  - errors, falsehoods, link rot, and spammers!
- Ah, but it's *hyperlinked* !
  - Vannevar Bush's 1945 memex



Memex in the form of a desk would instantly bring files and material on any subject to the operator's fingertips. Slanting translucent viewing screens magnify supermicrofilm filed by code numbers. At left is a mechanism which automatically photographs longhand notes, pictures and letters, then files them in the desk for future reference (LIFE 19(11), p. 123).

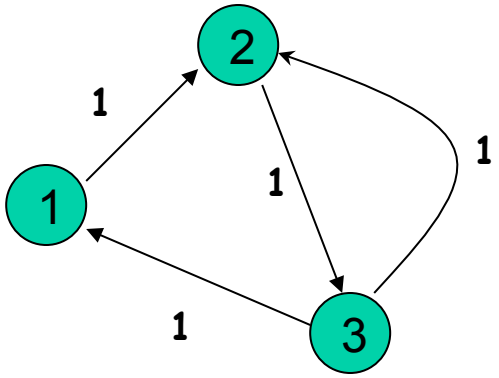
Adjacency Matrix → Transition Matrix

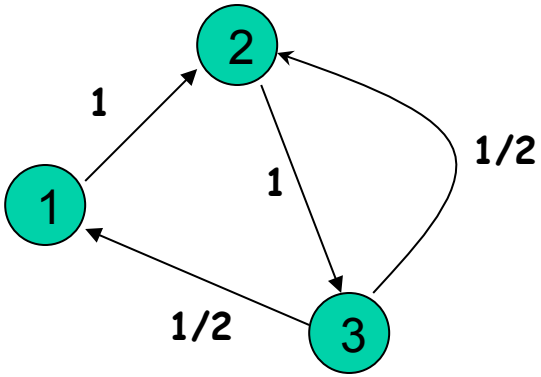
0	1	0
0	0	1
1	1	0

Adjacency matrix A

0	1	0
0	0	1
1/2	1/2	0

Transition matrix P

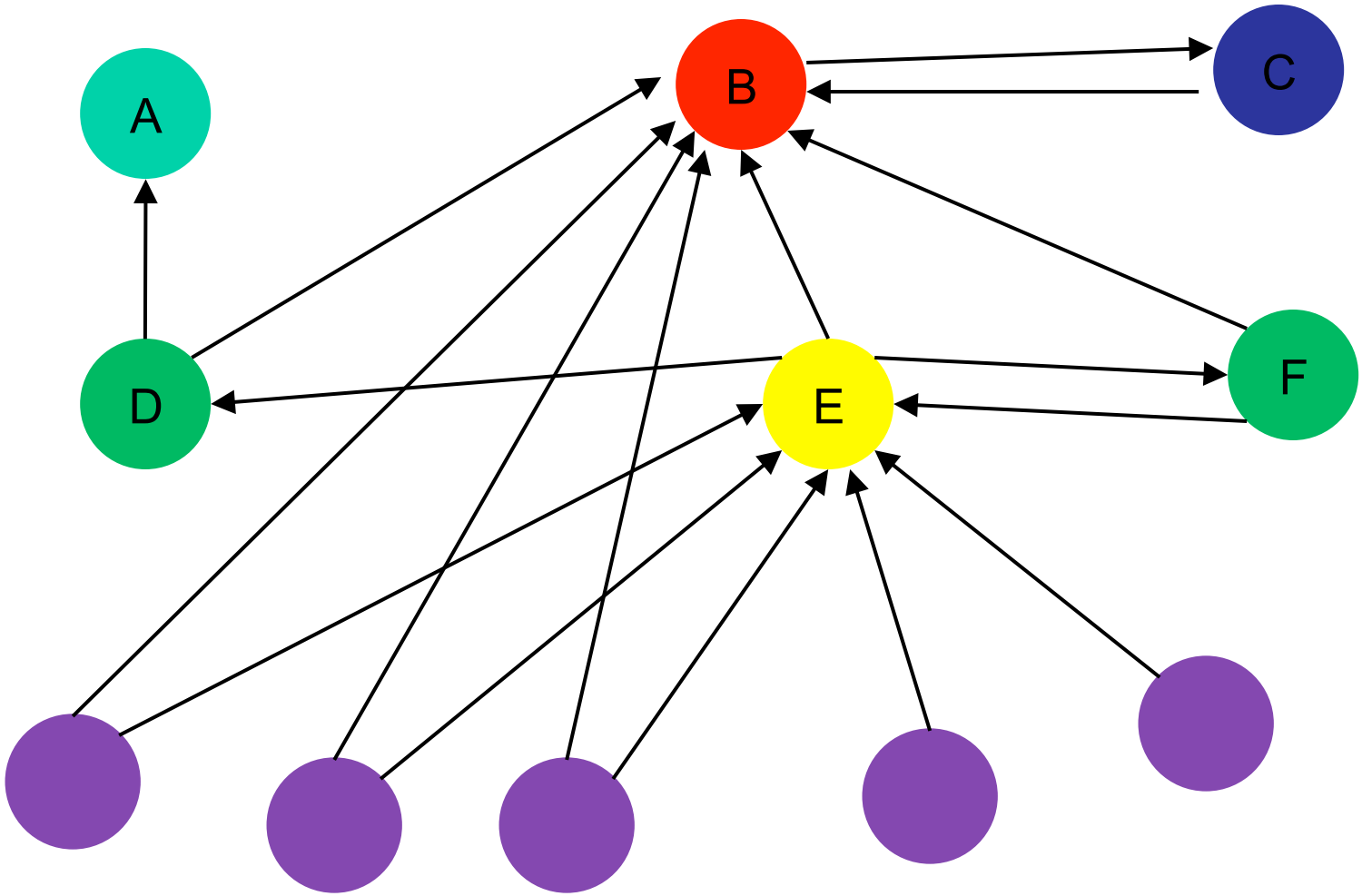




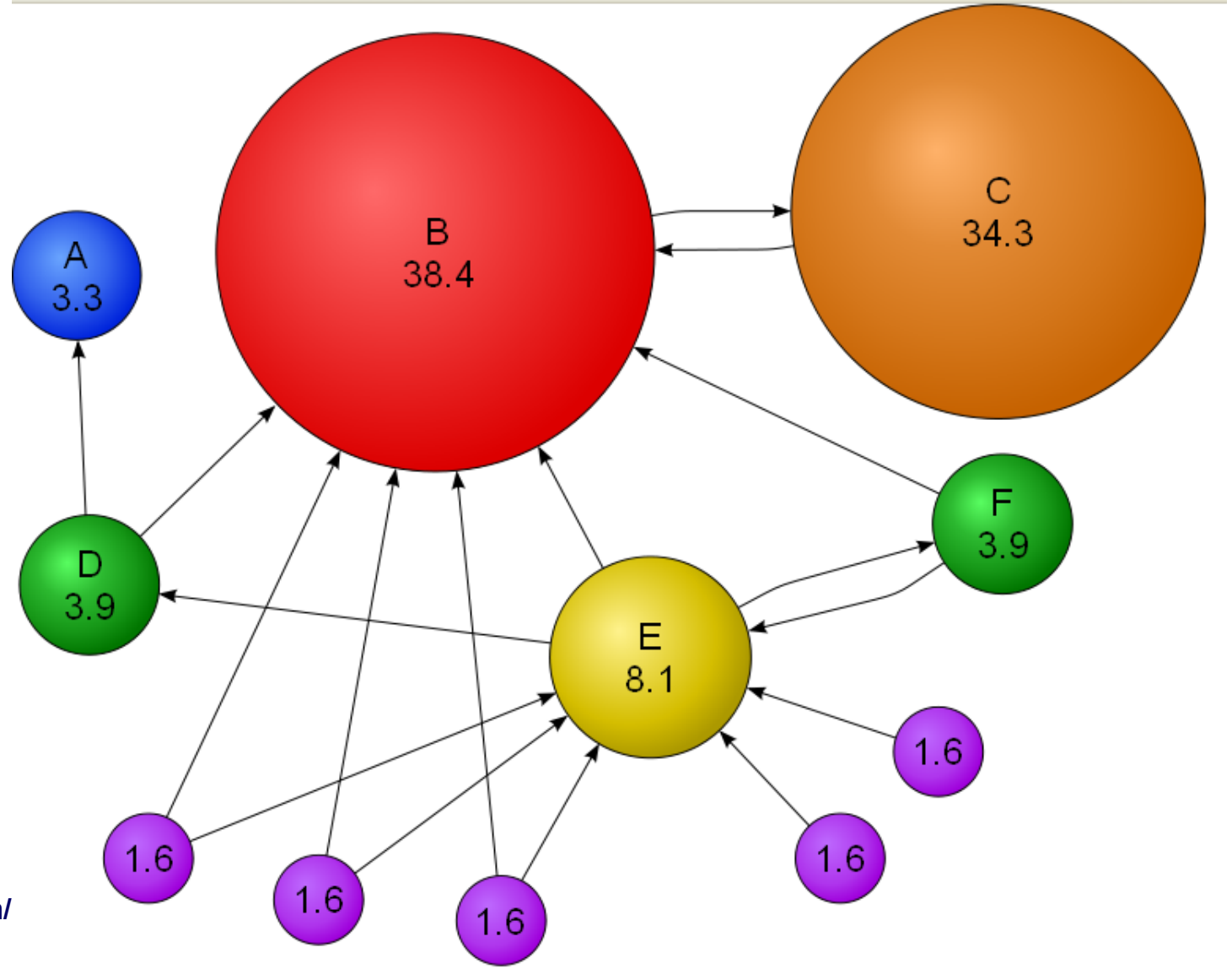
AKA: A right stochastic matrix is a real square matrix, with each row summing to 1.

PageRank example

15% probability of a random jump + dangling nodes



PageRank: Steady State Pr of being at a Page



Wikipedia.com/

PAGERANK: from random walks to pagerank

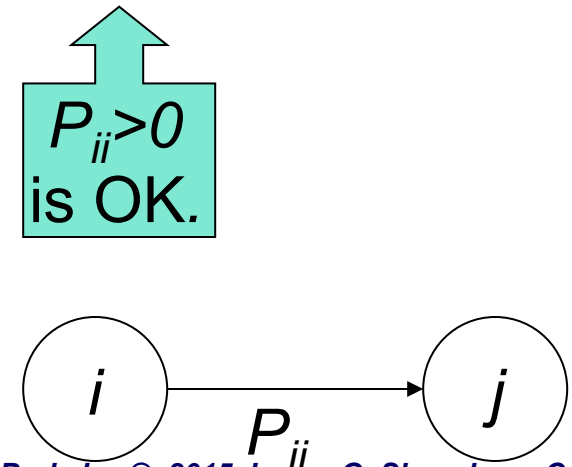
- **View pages as states, and webGraph as a transition matrix**
- **A Markov process who steady state distribution tells about which pages we spend a lot of time on.**
- **Making some minor adjustments to the transition matrix**
  - (stochasticity adjustment to deal with dangling nodes (sinks)); In this adjustment, the 0 rows of matrix H are replaced with 1/n e making H stochastic. This adjustment now allows the random surfer to hyperlink to any page at random after entering a dangling node.
  - Primitivity adjustment via teleportation
- **We can then use the power iteration method**
  - (first eigenvector, all positive values; eigenvalue ==1)
- **This is a proxy for popularity and is a very discriminating feature for in sebsearch context webpages**
- **Variations: Personalized pagerank**

Model random walk as a Markov Chain

- A Markov chain, named after Andrey Markov, is a mathematical system that undergoes transitions from one state to another, between a finite or countable number of possible states.
- It is a random process usually characterized as memoryless: the next state depends only on the current state and not on the sequence of events that preceded it.
- This specific kind of "memorylessness" is called the Markov property. Markov chains have many applications as statistical models of real-world processes.

Markov chain=States+Transition Matrix

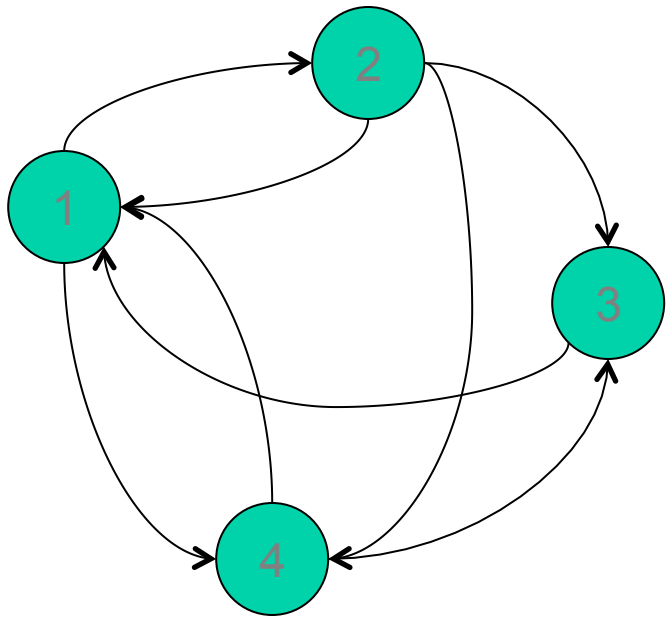
- A Markov chain consists of *n* states, plus an *n*×*n* transition probability matrix P.
- At each time step, we are in exactly one of the states.
- For $1 \le i,j \le n$, the matrix entry P_{ij} tells us the probability of *j* being the next state, given we are currently in state *i*.



Markov chains are abstractions of random walks

- **Clearly, for all each state/node i,** $\sum_{j=1}^{n} P_{ij} = 1.$
- **An example Transition Matrix without teleportation**

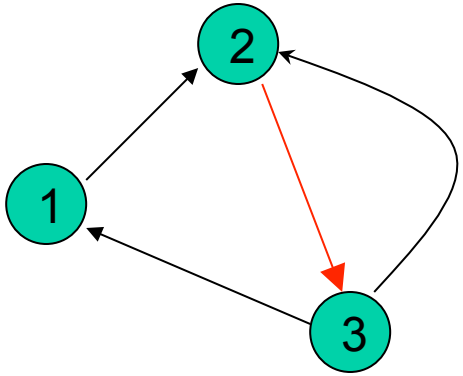
	1	2	3	4
1	0	0.5	0	0.5
2	0.33	0	0.33	0.33
3	1	0	0	0
4	0.5	0	0.5	0



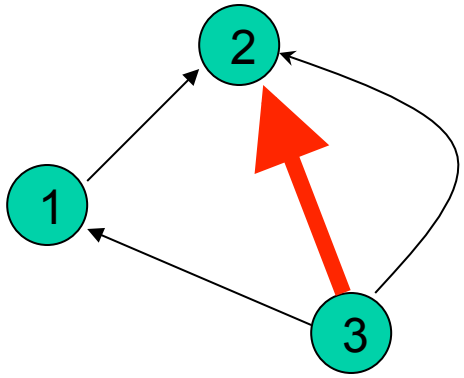
Well behaved graphs: Irreducible

- **Irreducible: There is a path from every node to every other node.**

Node 2 is a Dangling node



Irreducible

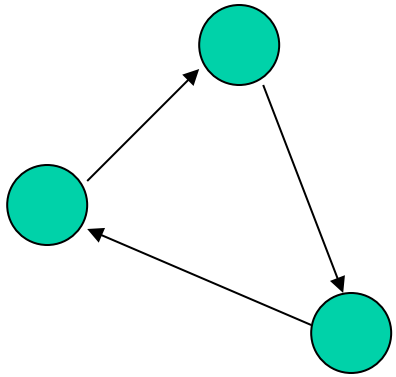


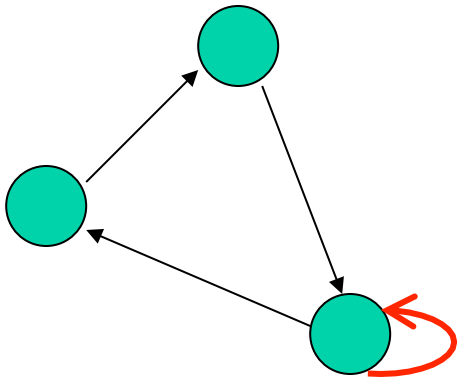
Not irreducible

Can not reach nodes 1 and 3 from node 2. (aka Dangling node, i.e., a node with no out links)

Well behaved graphs: Must be Aperiodic

- **Aperiodic: The GCD (great common divisor) of all cycle lengths is *1*. The GCD is also called period.**





A Markov chain is aperiodic if every state is aperiodic.
An irreducible Markov chain only needs one aperiodic state to imply all states are aperiodic.

Periodicity is fixed at 3 (can reach any other node in one step)

Aperiodic: The self-loops (G_ii > 0 for all i) create aperiodicity.

http://en.wikipedia.org/wiki/Aperiodic_graph

PageRank in Linear Algebra

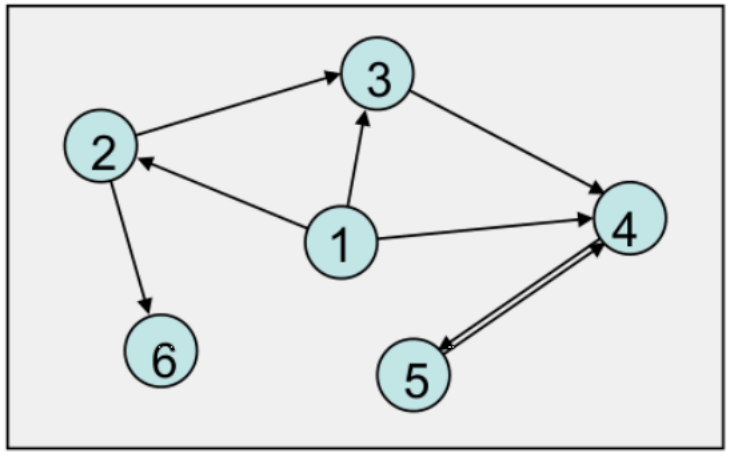


Figure 1: Example network with nodes representing websites and edges representing links

We can formalize this intuition via the process of a random walk. We would like to model a "random-surfer" who is traversing the web with uniform probability of following any link outgoing from the page the surfer is currently on. We are interested in the behavior of this random surfer in the limit as she takes an infinite number of jumps.

To express this in linear algebra, we will make a use of the adjacency matrix, A, and a out-degree matrix D, where:

$$A_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Q: transition matrix

To express this in linear algebra, we will make a use of the adjacency matrix, A, and a out-degree matrix D, where:

$$A_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$D = \begin{pmatrix} deg(v_1) & 0 & \dots & 0 \\ 0 & deg(v_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & deg(v_n) \end{pmatrix}$$

Let $Q = D^{-1}A$. This forms the transition matrix of the random-walker. Given the current state of the walker each row of the matrix gives the probability the walker will transition to each new state.

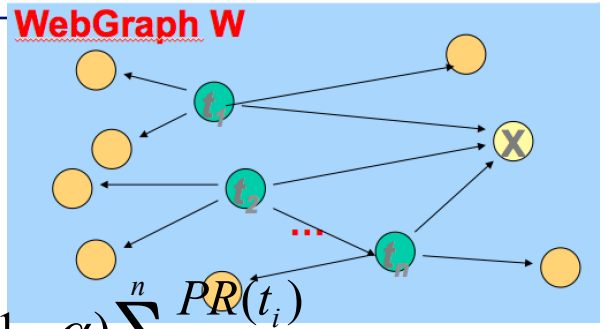
$$Q_{i,j} = \begin{cases} 1/deg(v_i) & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

It's interesting to note that Q^k_{ij} is equal to the probability of going from node i to node j in exactly k steps, in a random walk over graph G.

PageRank: Summary

Given page *x* with inbound links $t_1 \ldots t_n$, where

- $C(t)$ is the out-degree of t
- α is probability of random jump
- $|W|$ is the total number of nodes in the graph
- V is a biased **teleport vector**



$$PR(x \mid W) = \alpha\left(\frac{1}{|W|}\right) + (1-\alpha)\sum_{i=1}^{n}\frac{PR(t_i)}{C(t_i)}$$

Biased PageRank

$$PR(x \mid W, v) = \alpha v + (1-\alpha)\sum_{i=1}^{n}\frac{PR(t_i)}{C(t_i)}$$

P (stochastic) transition probability matrix: **P = (1-α)H + αI(1/n)**

$$P = 0.85 \times \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 1/3 & 1/2 & 1/5 \\ 1/2 & 0 & 0 & 1/2 & 1/5 \\ 1/2 & 1 & 0 & 0 & 1/5 \\ 0 & 0 & 1/3 & 0 & 1/5 \\ 0 & 0 & 1/3 & 0 & 1/5 \end{bmatrix} \end{array} + 0.15 \times \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix} \end{array}$$

Hyperlink Matrix — Teleporting Matrix

Display at 00:21

PageRank Algorithm: Apply Power iteration method: after k iterations $\pi = P^k\pi$
Where π is vector consisting of: π_i = pagerank of page *I*; *H* is the hyperlink graph

http://www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html

PageRank Computation

- **Target**
  - Solve the steady-state probability vector π, which is the PageRank of the corresponding Web page.
  - $\pi P=\lambda \pi$, λ is 1 for stochastic matrix.
- **Method**
  - Power iteration.
  - Given an initial probability distribution vector π^0
  - $\pi^0*P=\pi^1$, $\pi^1*P=\pi^2$ … Until the probability distribution converges. (Variation in the computed values are below some predetermined threshold.)

Display at 1:00

Matrix multiplication at scale
Better to view from a graph perspective

PageRank

This means we now will use the following matrix to parameterize our Markov chain describing our random walker.

$$P = \alpha\Lambda + (1-\alpha)Q$$

where $0 < \alpha < 1$ is the teleport probability, and Λ is a rank 1 matrix whose rows correspond to the distribution of where a teleporting surfer arrives.
We are looking for a row vector π of node probabilities such that:

$$\pi P = \pi$$

We normally solve problems like this via power iteration. That algorithm is very simple. We initialize $V^{(0)}$ to any initial distribution -for example to the uniform vector $V^{(0)} = [1/n, \ldots, 1/n]$. Then we follow a converging recurrence:

$$V^{(k+1)} = V^{(k)}P = V^{(0)}P^{k+1}$$

Skip to next section

- **Other background slides in this section**
  - (no need to cover; see async video); i.e., skip ahead to the next section in these slides
  - Pagerank
  - Distribute pagerank

Perron Frobenius Theorem

- **If the Markov chain is time-homogeneous, then the transition matrix P is the same after each step, so the *k*-step transition probability can be computed as the *k*-th power of the transition matrix, P^k.**
- **If a markov chain is irreducible and aperiodic then the largest eigenvalue of the transition matrix will be equal to 1 and all the other eigenvalues will be strictly less than 1.**

- **Implications of the Perron Frobenius Theorem**
  - Let the eigenvalues of P be $\{\sigma_i | \; i=0:n-1\}$ in non-increasing order of σ_i .
  - $\sigma_0 = 1 > \sigma_1 > \sigma_2 >= \ldots\ldots >= \sigma_n$

Implications of the Perron Frobenius Theorem

- **Primitive matrix is irreducible and aperiodic**
- **If a markov chain is irreducible and aperiodic then the largest eigenvalue of the transition matrix will be equal to 1 and all the other eigenvalues will be strictly less than 1.**
  - Let the eigenvalues of P be $\{\sigma_i | \; i=0:n-1\}$ in non-increasing order of σ_i .
  - $\sigma_0 = 1 > \sigma_1 > \sigma_2 >= \ldots\ldots >= \sigma_n$
- **These results imply that for a well behaved graph there exists an unique stationary distribution.**
- **If primitive transition matrix then can use power method**

- This random walk algorithm serves as the basis for many more graph-based algorithms
- The modifications we made also (for personalization) will be revisited in the context of social graphs under a very different guise.
- Some of which we will look at in later lectures
- Most web, mobile, social Graphs these days run in the order of trillions of edges…
-

Transition matrix; stationary vector

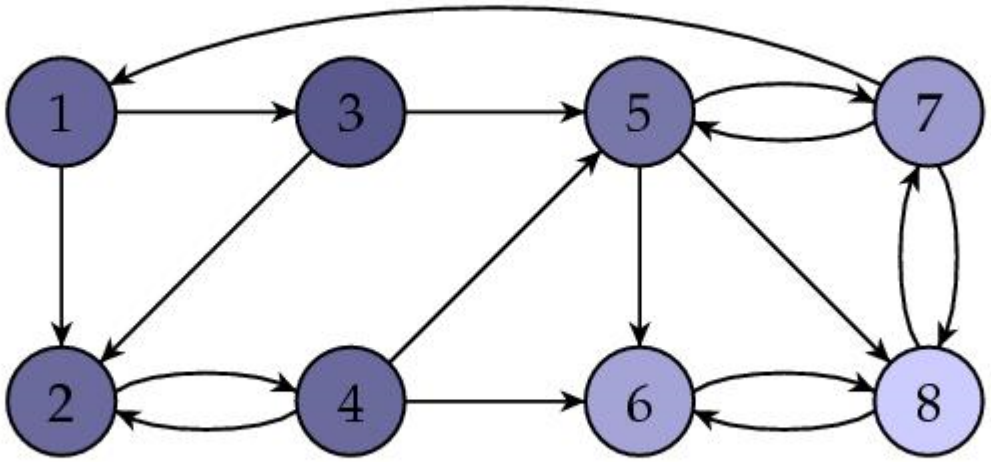
Graph in matrix form

$$\mathrm{H} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 1 & 1/3 & 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix}$$

with stationary vector

$$I = \begin{bmatrix} 0.0600 \\ 0.0675 \\ 0.0300 \\ 0.0675 \\ 0.0975 \\ 0.2025 \\ 0.1800 \\ 0.2950 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix}$$

This shows that page 8 wins the popularity contest. Here is the same figure with the web pages shaded in such a way that the pages with hig
PageRanks are lighter.



Computing PR Via Power Method

The method is founded on the following general principle that we will soon investigate.

> ***General principle:*** *The sequence I^k will converge to the stationary vector I.*

We will illustrate with the example above.

I^0	I^1	I^2	I^3	I^4	...	I^{60}	I^{61}
1	0	0	0	0.0278	...	0.06	0.06
0	0.5	0.25	0.1667	0.0833	...	0.0675	0.0675
0	0.5	0	0	0	...	0.03	0.03
0	0	0.5	0.25	0.1667	...	0.0675	0.0675
0	0	0.25	0.1667	0.1111	...	0.0975	0.0975
0	0	0	0.25	0.1806	...	0.2025	0.2025
0	0	0	0.0833	0.0972	...	0.18	0.18
0	0	0	0.0833	0.3333	...	0.295	0.295

It is natural to ask what these numbers mean. Of course, there can be no absolute measure of a page's importance, only relative measures for comparing the importance of two pages through statements such as "Page A is twice as important as Page B." For this reason, we may multiply all the importance rankings by some fixed quantity without affecting the information they tell us. In this way, we will always assume, for reasons to be explained shortly, that the sum of all the popularities is one.

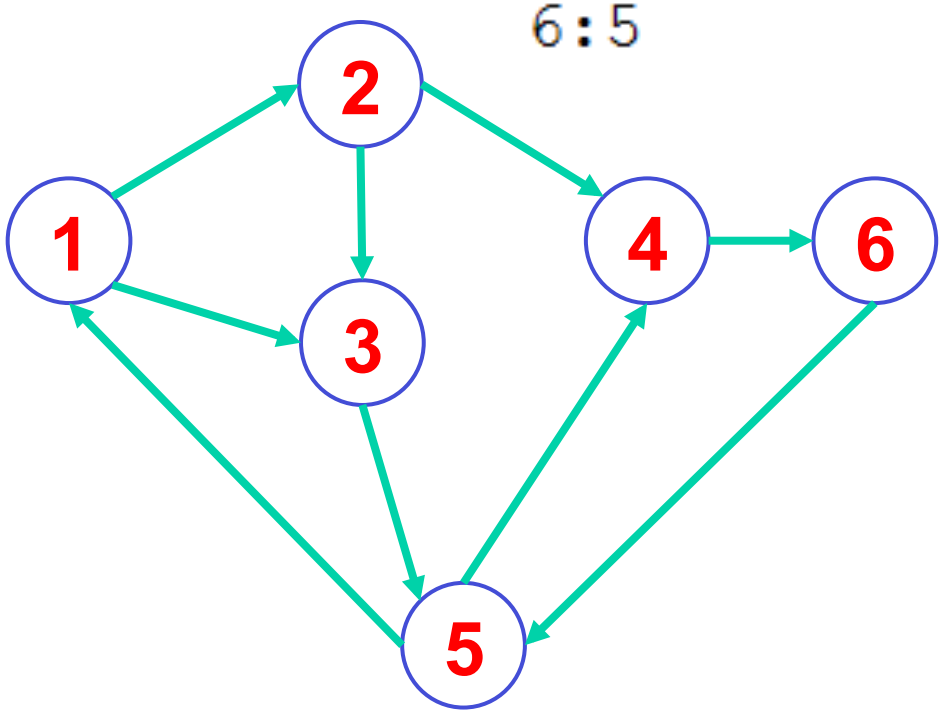
PageRank homegrown implementation

Adjacency matrix is fixed

```
1:2,3
2:3,4
3:5
4:6
5:1,4
6:5
```

```
%%writefile PageRank.txt
1:2,3
2:3,4
3:5
4:6
5:1,4
6:5
```

```
Overwriting PageRank.txt
```



Pagerank

- Connect all the dots: Random surfer→ markov process →

- Adapt the machinery of markov processes to give us a principled approach to calculate the pagerank of each webpage; it nothing more than the steady state probability distribution of the markov process underlying the random surfer model of web navigation
- PageRank is a link analysis algorithm that "measures" relative importance of each within the webgraph.

Adjustments to WebGraph Markov Process

- **Stochasticity adjustment**
  - to get over dangling edges
- **Primitivity adjustment**
  - guarantees convergernce (teleportation in Brin and Page paper)
  - Primitive matrix is irreducible and aperiodic

- **Use power iteration method to calculate the steady state distribution of the web stochastic Markov process → PageRank**

Using Pagerank

- **Preprocessing:**
  - Given graph of links, build matrix **P**.
  - From it compute **a**.
  - The entry a_i is a number between 0 and 1: the pagerank of page *i*.
- **Query processing:**
  - Retrieve pages matching the query.
  - Rank them by their pagerank.
  - Order is query-*independent*.
- **What is a good way to order Documents within an index server?**

- **From random walks to Markov processes to PageRank**
- **Seen how to potentially use PR in websearch for re-ranking**
- **Next**

CLEVE'S CORNER

THE WORLD'S LARGEST MATRIX COMPUTATION

Google's PageRank is an eigenvector of a matrix of order 2.7 billion.

BY CLEVE MOLER

One of the reasons why Google is such an effective search engine is the PageRank™ algorithm, developed by Google's founders, Larry Page and Sergey Brin, when they were graduate students at Stanford University. PageRank is determined entirely by the link structure of the Web. It is recomputed about once a month and does not involve any of the actual content of Web pages or of any individual query. Then, for any particular query, Google finds the pages on the Web that match that query and lists those pages in the order of their PageRank.

Imagine surfing the Web, going from page to page by randomly choosing an outgoing link from one page to get to the next. This can lead to dead ends at pages with no outgoing links, or cycles around cliques of interconnected pages. So, a certain fraction of the time, simply choose a random page from anywhere on the Web. This theoretical random walk of the Web is a *Markov chain* or *Markov process*. The limiting probability that a dedicated random surfer visits any particular page is its PageRank. A page has high rank if it has links to and from other pages with high rank.

Let W be the set of Web pages that can reached by following a chain of hyperlinks starting from a page at Google and let n be the number of pages in W. The set W actually varies with time, but in May 2002, n was about 2.7 billion. Let G be the n-by-n connectivity matrix of

It tells us that the largest eigenvalue of A is equal to one and that the corresponding eigenvector, which satisfies the equation

$$x = Ax,$$

exists and is unique to within a scaling factor. When this scaling factor is chosen so that

$$\sum_i x_i = 1$$

then x is the state vector of the Markov chain. The elements of x are Google's PageRank.

If the matrix were small enough to fit in MATLAB, one way to compute the eigenvector x would be to start with a good approximate solution, such as the PageRanks from the previous month, and simply repeat the assignment statement

```
x = Ax
```

until successive vectors agree to within specified tolerance. This is known as the power method and is about the only possible approach for very large n. I'm not sure how Google actually computes PageRank, but one step of the power method would require one pass over a database of Web pages, updating weighted reference counts generated by the hyperlinks between pages.

Section 9.9 Distributed PageRank

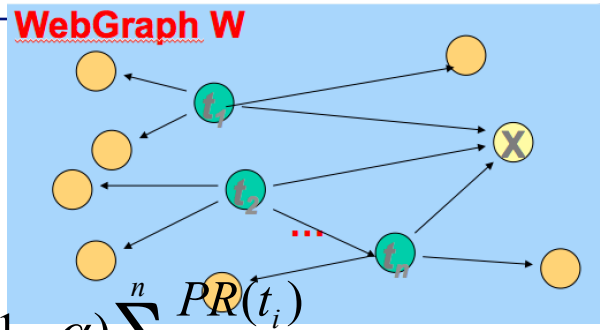
- How to distribute the PageRank calculation of a big graph across a cluster of computers using a MapReduce framework

Display at 00:01

PageRank: Summary

Given page *x* with inbound links $t_1 \ldots t_n$, where

- $C(t)$ is the out-degree of t
- α is probability of random jump
- $|W|$ is the total number of nodes in the graph
- V is a biased **teleport vector**



$$PR(x \mid W) = \alpha\left(\frac{1}{|W|}\right) + (1-\alpha)\sum_{i=1}^{n}\frac{PR(t_i)}{C(t_i)}$$

Biased PageRank

$$PR(x \mid W, v) = \alpha v + (1-\alpha)\sum_{i=1}^{n}\frac{PR(t_i)}{C(t_i)}$$

P (stochastic) transition probability matrix: **P = (1-α)H + αI(1/n)**

$$P = 0.85 \times \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 1/3 & 1/2 & 1/5 \\ 1/2 & 0 & 0 & 1/2 & 1/5 \\ 1/2 & 1 & 0 & 0 & 1/5 \\ 0 & 0 & 1/3 & 0 & 1/5 \\ 0 & 0 & 1/3 & 0 & 1/5 \end{bmatrix} \end{array} + 0.15 \times \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix} \end{array}$$

Hyperlink Matrix — Teleporting Matrix

Display at 00:21

PageRank Algorithm: Apply Power iteration method: after k iterations $\pi = P^k\pi$
Where π is vector consisting of: π_i = pagerank of page *I*; *H* is the hyperlink graph

http://www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html

PageRank Computation

- **Target**
  - Solve the steady-state probability vector π, which is the PageRank of the corresponding Web page.
  - $\pi P=\lambda \pi$, λ is 1 for stochastic matrix.
- **Method**
  - Power iteration.
  - Given an initial probability distribution vector π^0
  - $\pi^0*P=\pi^1$, $\pi^1*P=\pi^2$ … Until the probability distribution converges. (Variation in the computed values are below some predetermined threshold.)

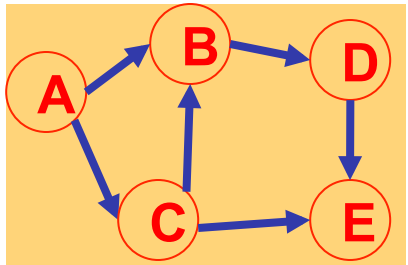
Display at 1:00

Matrix multiplication at scale
Better to view from a graph perspective

Web graph: Rep^t as Adjacency Lists

Graph →Adjacency Lists

Key	Value	
A	B,C	|payload
B	D	
C	B,E	
D	E	
E		



WebGraph

- **Series of power iterations**
- $\pi^{i+1} = P\pi^i$

Iterate until convergence

Value of node is graph structure (i.e., neighbors) and PageRank of that nod

Series of power iterations
- $\pi^{i+1} = P\pi^i$

Each input node will generate multiple records in the stream using the Mapper corresponding to the Reducer will group the generated records

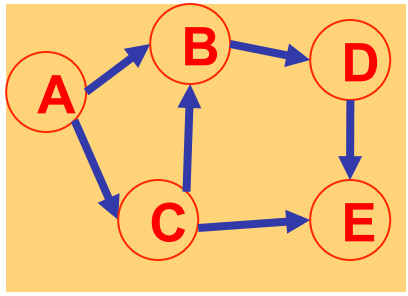
Key	Value	
A	A	$\|\pi^A$
B	AB	$\|\pi^B$
C	AC	$\|\pi^C$
D	ABD	$\|\pi^D$
E	ABDE	$\|\pi^E$

Display at 1:20

Web graph: Initialize

Graph →Adjacency Lists

Key	Value	
A	A	|π^A
B	AB	|π^B
C	AC	|π^C
D	ABD	|π^D
E	ABDE	|π^E



WebGraph

Initialize

Key	Value	
A	A	|1/N
B	AB	|1/N
C	AC	|1/N
D	ABD	|1/N
E	ABDE	|1/N

Uniform probability:
Where is N is the number of nodes in the graph

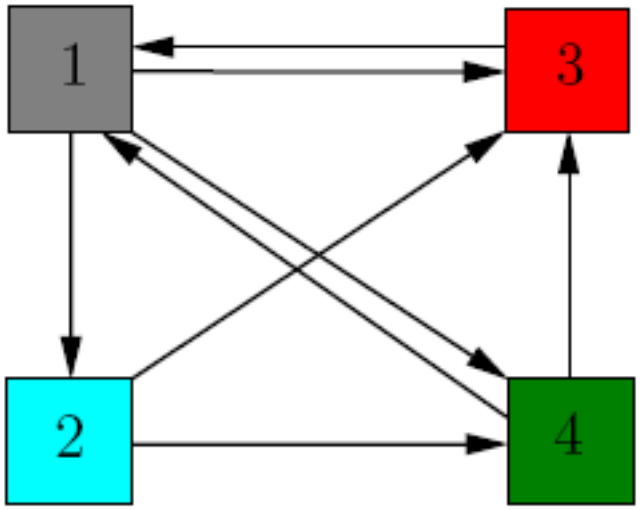
Display at 2:20

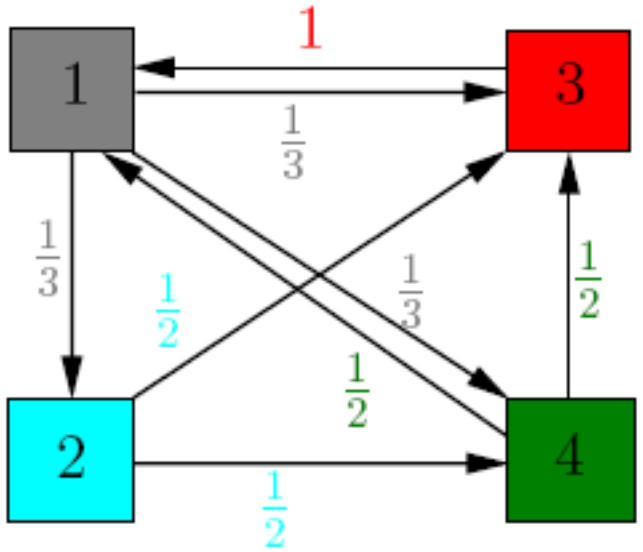
Divide and conquer around each node

- **Atomic level of operation for PageRank is the graph node**
- **Where the *Key* is node;**
- **And the *Value* is**
  - outlink neighbors (partial graph structure) and
  - PageRank of that node
- **Divide the node's pagerank calculation into pieces**
- **And synchronize around the node by gathering the component pieces and summing them up**

Display at 2:35

PageRank: Adjacency Matrix





Pr(3→1)=1
Pr(1|3)

A[1,3] = Pr(3→1) = 1

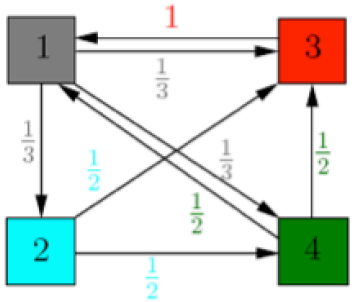
From

To

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

PageRank Calculation in a Slide

Suppose that initially the importance is uniformly distributed among the 4 nodes, each getting ¼. Denote by v the initial rank vector, having all entries equal to ¼. Each incoming link increases the importance of a web page, so at step 1, we update the rank of each page by adding to the current value the importance of the incoming links. This is the same as multiplying the matrix A with v . At step 1, the new importance vector is $v_1 = Av$. We can iterate the process, thus at step 2, the updated importance vector is $v_2 = A(Av) = A^2v$. Numeric computations give:



$$v = \begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{pmatrix}, \quad Av = \begin{pmatrix} 0.37 \\ 0.08 \\ 0.33 \\ 0.20 \end{pmatrix}, \quad A^2 v = A(Av) = A \begin{pmatrix} 0.37 \\ 0.08 \\ 0.33 \\ 0.20 \end{pmatrix} = \begin{pmatrix} 0.43 \\ 0.12 \\ 0.27 \\ 0.16 \end{pmatrix}$$

$$A^3 v = \begin{pmatrix} 0.35 \\ 0.14 \\ 0.29 \\ 0.20 \end{pmatrix}, \quad A^4 v = \begin{pmatrix} 0.39 \\ 0.11 \\ 0.29 \\ 0.19 \end{pmatrix}, \quad A^5 v = \begin{pmatrix} 0.39 \\ 0.13 \\ 0.28 \\ 0.19 \end{pmatrix}$$

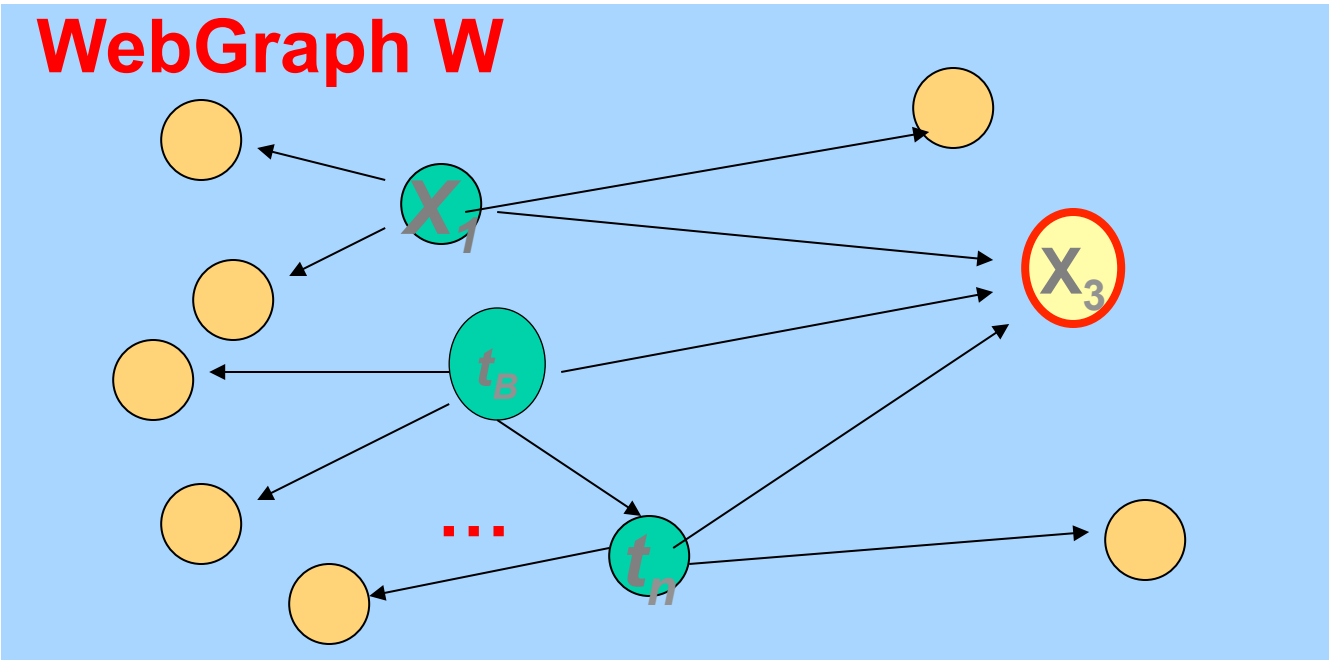
$$A^6 v = \begin{pmatrix} 0.38 \\ 0.13 \\ 0.29 \\ 0.19 \end{pmatrix}, \quad A^7 v = \begin{pmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{pmatrix}, \quad A^8 v = \begin{pmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{pmatrix}$$

$$\text{To} \begin{bmatrix} 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{bmatrix} \text{From}$$

We notice that the sequences of iterates v, Av, ..., $A^k v$ tends to the equilibrium value $v^* = \begin{pmatrix} 0.38 \\ 0.12 \\ 0.29 \\ 0.19 \end{pmatrix}$. We call this the PageRank vector of our web graph.

PageRank as a matrix multiplication: Graph → Matrix → Sparse Matrix

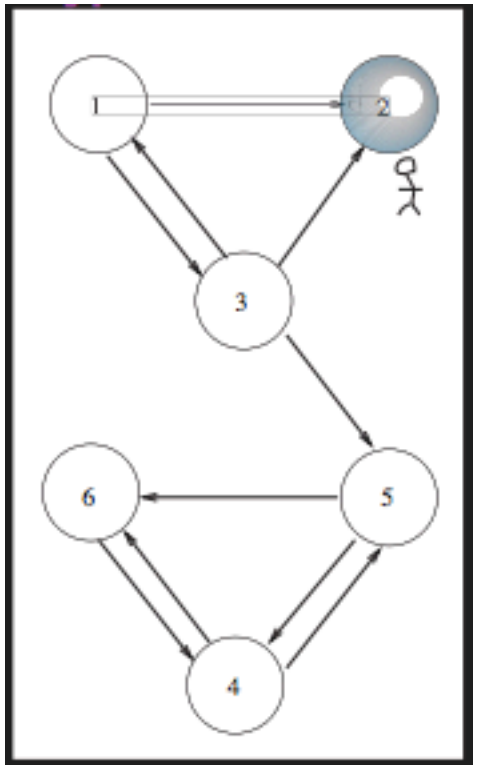
Fill sparse calculation in pagerank



- **Dangling nodes (deadend nodes)**

Dangling nodes (deadend nodes)

- **Node 2 is an example of a dangling node (it has no outlinks)**



More complete PageRank in MR

- **Random jump factor (teleportation)**
- **Dangling nodes:**
  - Dangling nodes are nodes in the graph that have no outgoing edges (dead end nodes)i.e., their adjacency lists are empty.
  - In the hyperlink graph of the web, these might correspond to pages in a crawl that have not been downloaded yet.
  - If we simply run the algorithm presented above on graphs with dangling nodes, the total PageRank mass will not be conserved, since no key-value pairs will be emitted when a dangling node is encountered in the mappers
  - (could end up with lot of missing mass from the initialization phase)

Get the mapper to track the PR Mass of dangling nodes in a Hadoop Counter and redistribute to all nodes

- **The proper treatment of PageRank mass “lost" at the dangling nodes is to redistribute it across all nodes in the graph evenly**
- **Instrument MR with counters**
  - One simple approach is by instrumenting the algorithm presented above with counters: whenever the mapper processes a node with an empty adjacency list, it keeps track of the node's PageRank value in the counter.
  - At the end of the iteration, we can access the counter to find out how much PageRank mass was lost at the dangling nodes.
  - NOTE In Hadoop, counters are 8-byte integers: a simple workaround is to multiply PageRank values by a large constant, and then cast as an integer.

Get the mapper to track the PR Mass of dangling nodes in a Hadoop Stream and redistribute to all nodes

- **The proper treatment of PageRank mass “lost" at the dangling nodes is to redistribute it across all nodes in the graph evenly**
- **Instrument MR with an instream key-value pair**
  - Another approach is to reserve a special key for storing PageRank mass from dangling nodes. When the mapper encounters a dangling node, its PageRank mass is emitted with the special key; the reducer must be modified to contain special logic for handling the missing PageRank mass.
  - Yet another approach is to write out the missing PageRank mass as side data" for each map task (using the in-mapper combining technique for aggregation); a final pass in the driver program is needed to sum the mass across all map tasks.

How to redistribute the dangling PR Mass?

- Either way, we arrive at the amount of PageRank mass lost at the dangling nodes. This then must be redistributed evenly across all nodes.
- This redistribution process can be accomplished by mapping over all nodes again.
- At the same time, we can take into account the random jump factor. For each node, its current PageRank value p is updated to the final PageRank value p0 according to the following formula:

$$p' = \alpha\left(\frac{1}{|G|}\right) + (1-\alpha)\left(\frac{m}{|G|} + p\right)$$

  - where m is the missing PageRank mass, and |G| is the number of nodes in the entire graph.
- Need a second MR job with a Mapper to do this

Each PageRank iteration requires 2 MR jobs

- **Distribute PageRank mass along graph edges,**
- **And the second to take care of dangling nodes and the random jump factor.**
- **NOTE:**
  - At end of each iteration, we end up with exactly the same data structure as the beginning, which is a requirement for the iterative algorithm to work.
  - Also, the PageRank values of all nodes sum up to one, which ensures a valid probability distribution.

How many Iterations for MR PageRank?

- Typically, PageRank is iterated until convergence, i.e., when the PageRank values of nodes no longer change (within some tolerance, to take into account, for example, floating point precision errors).

- Stopping Criteteria

Alternative stopping Criteria?

- **PageRank values of nodes**
  - Typically, PageRank is iterated until convergence, i.e., when the PageRank values of nodes no longer change (within some tolerance, to take into account, for example, floating point precision errors).
- **Fixed Number of iterations**
- **Ranks of PageRank values no longer change**
  - Alternative stopping criteria include running a fixed number of iterations (useful if one wishes to bound algorithm running time) or stopping when the ranks of PageRank values no longer change.
  - The latter is useful for some applications that only care about comparing the PageRank of two arbitrary pages and do not need the actual PageRank values.
- **Rank stability is obtained faster than the actual convergence of values.**

- **Topic Specific Pagerank**

Topic Specific Pagerank: One Map Reduce Job

- **Implementation**
  - **offline**:Compute pagerank distributions wrt to *individual* categories

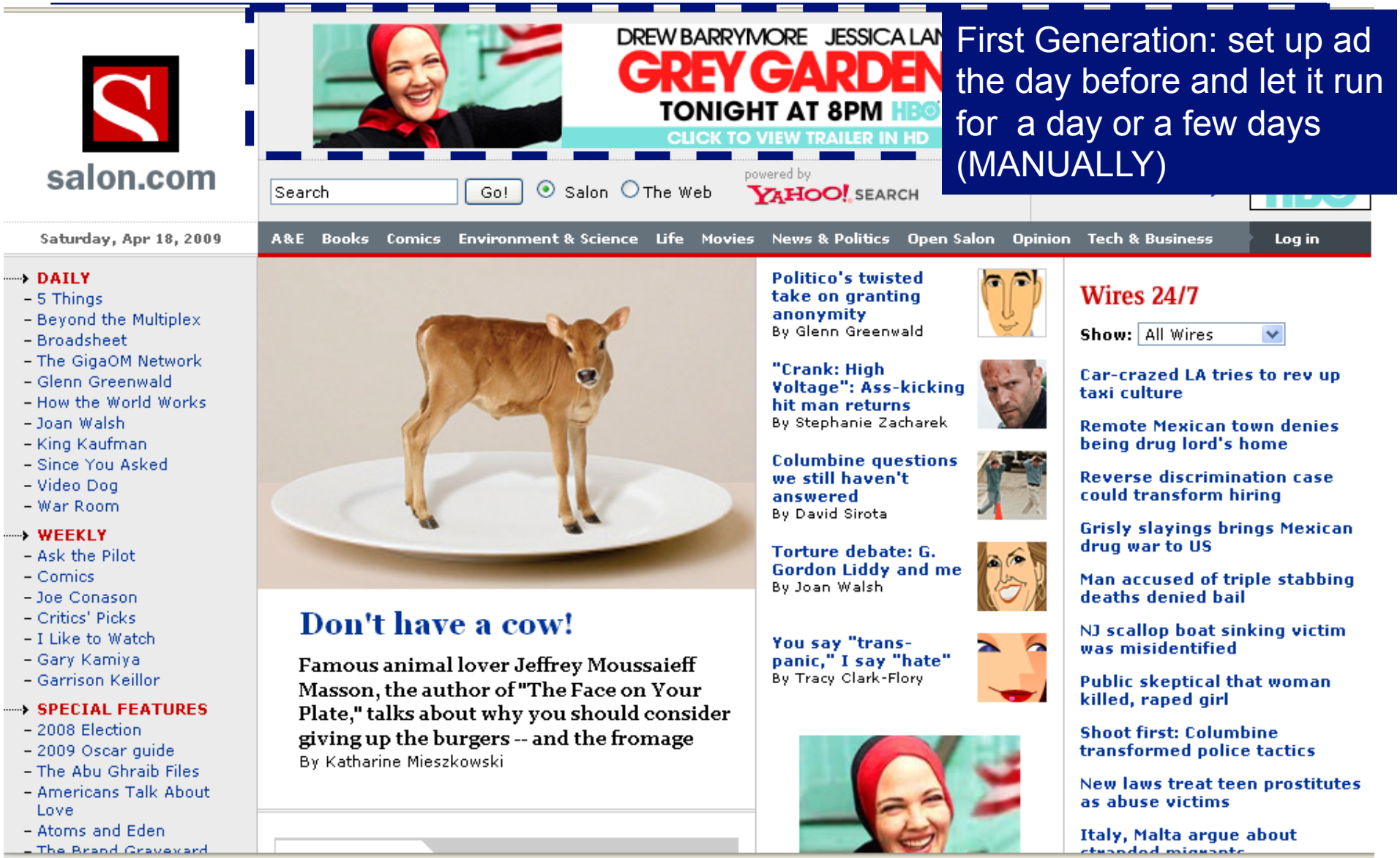
    Query independent model as before

    Each page has multiple pagerank scores – one for each ODP category, with teleportation only to that category
- **MapReduce (one set for 16+1 PR)**
  - **Page calculation**
  - **Dangling nodes plus teleportation**

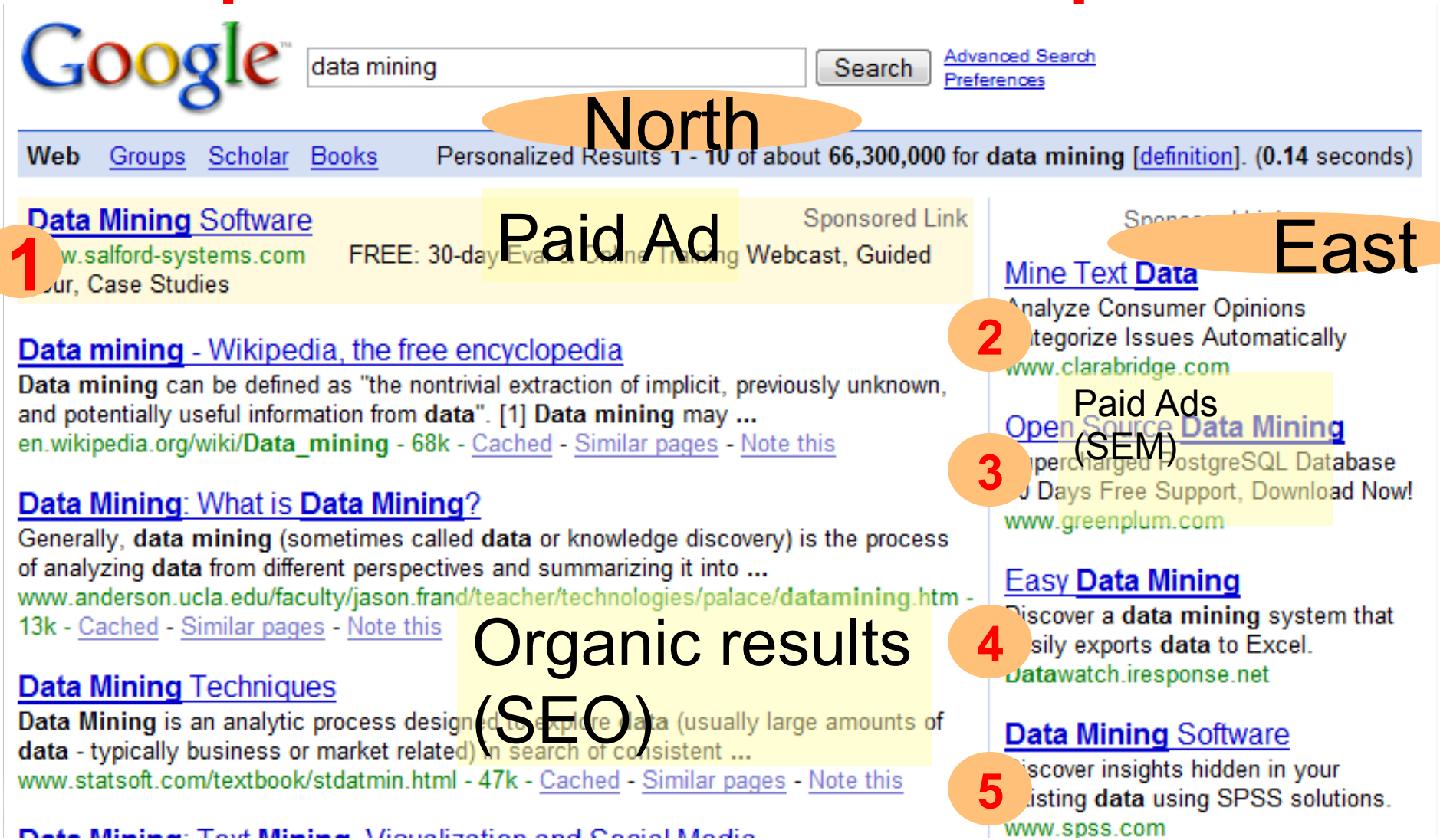
Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Mid term; Feedback/Evaluation
  - Homework HW6, HW7, HW9
  - AWS: no access
  - Async lecture recap plus Q&A (PageRank)
  - Contextual advertising
  - Text as graph: TextRank
    - Keyword extraction (from text/target pages)
    - Text Summarization
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

Ad Placement used to be simple circa mid to late 90s

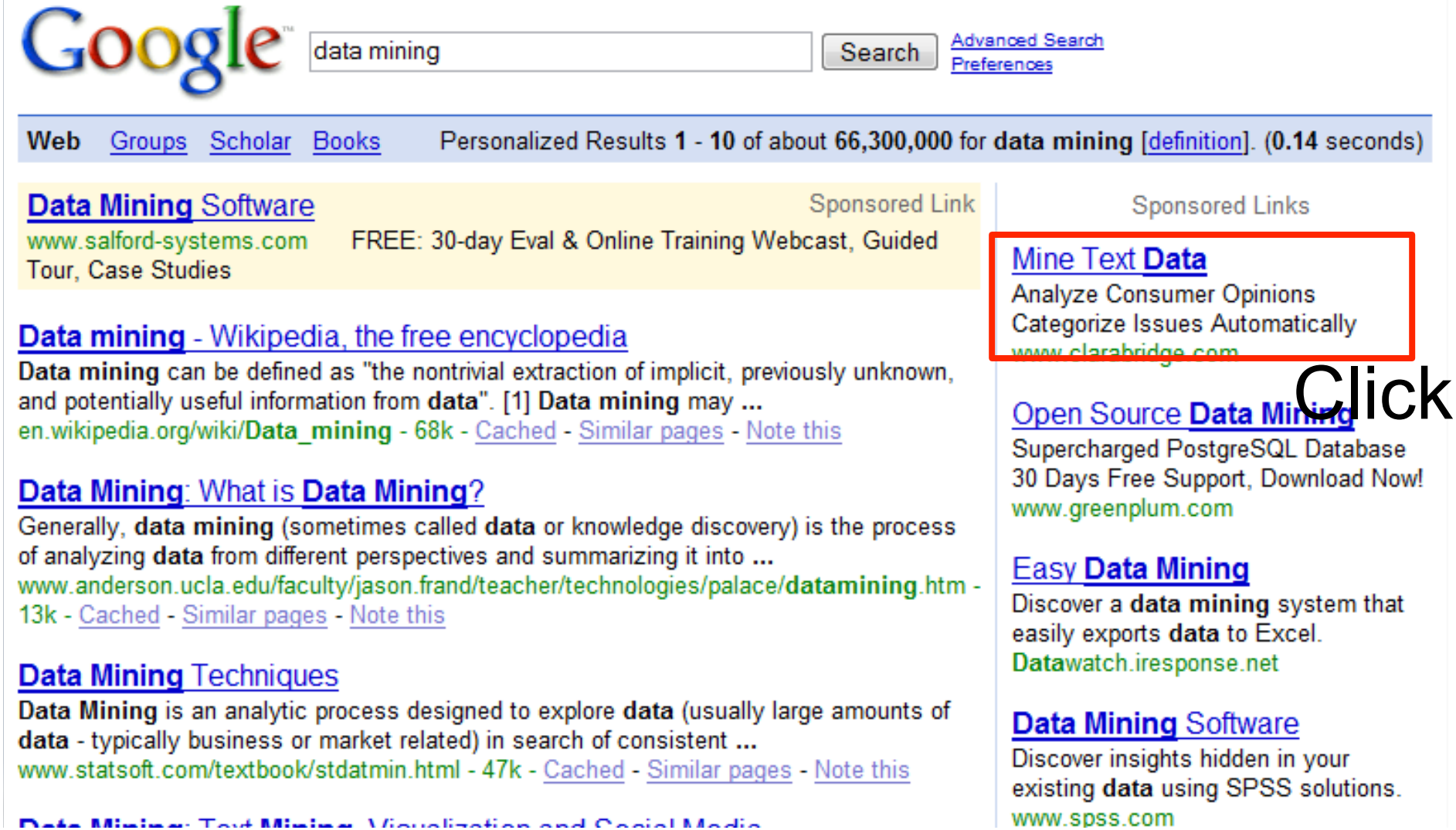


Sponsored Search: Ad Impression



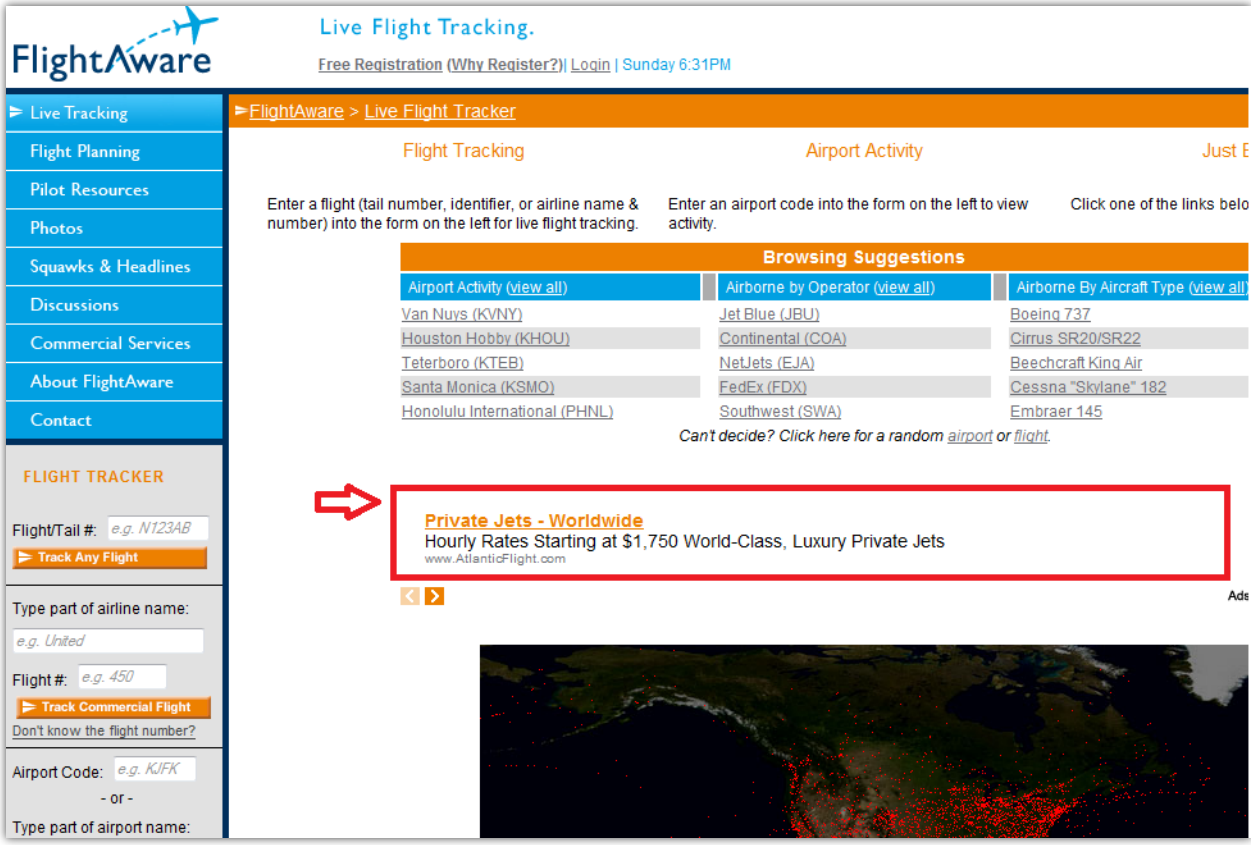
Ads are clearly distinguishable from the actual search results and they rotate

Sponsored Search: Click



Contextual Ads:
Target based on text of page

- Served by the web site to its visitors
- Ad network selects ads that are highly related to the content of the web page



Display Ads: Ads targeted based on the demographics of the visitors

- Also called banner ads
- Served by web sites
- Similar to 'Display' or 'Hoardings' on road side
- Ads are targeted based on the **demographics of the visitors** of the web site

Display Advertising: Mobile Web

- **How are the ads targeted on this page?**
- **First review (next slide)**
- **Challenge: Second next slide**

Contextual Advertising on Web Pages

How are the ads targeted on this page?

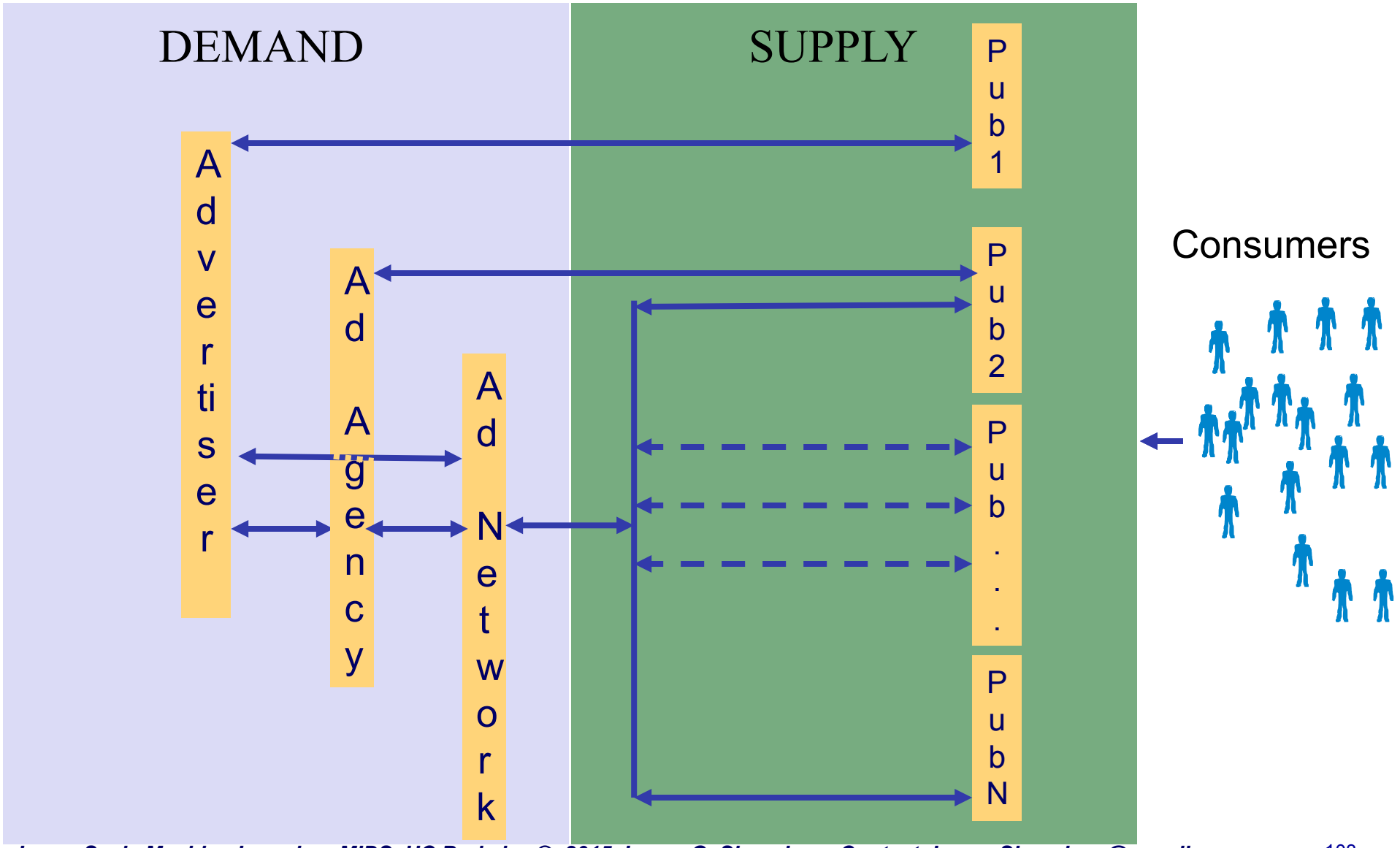
How are the ads targeted on this page?

Supply can be fragmented → Ad Networks

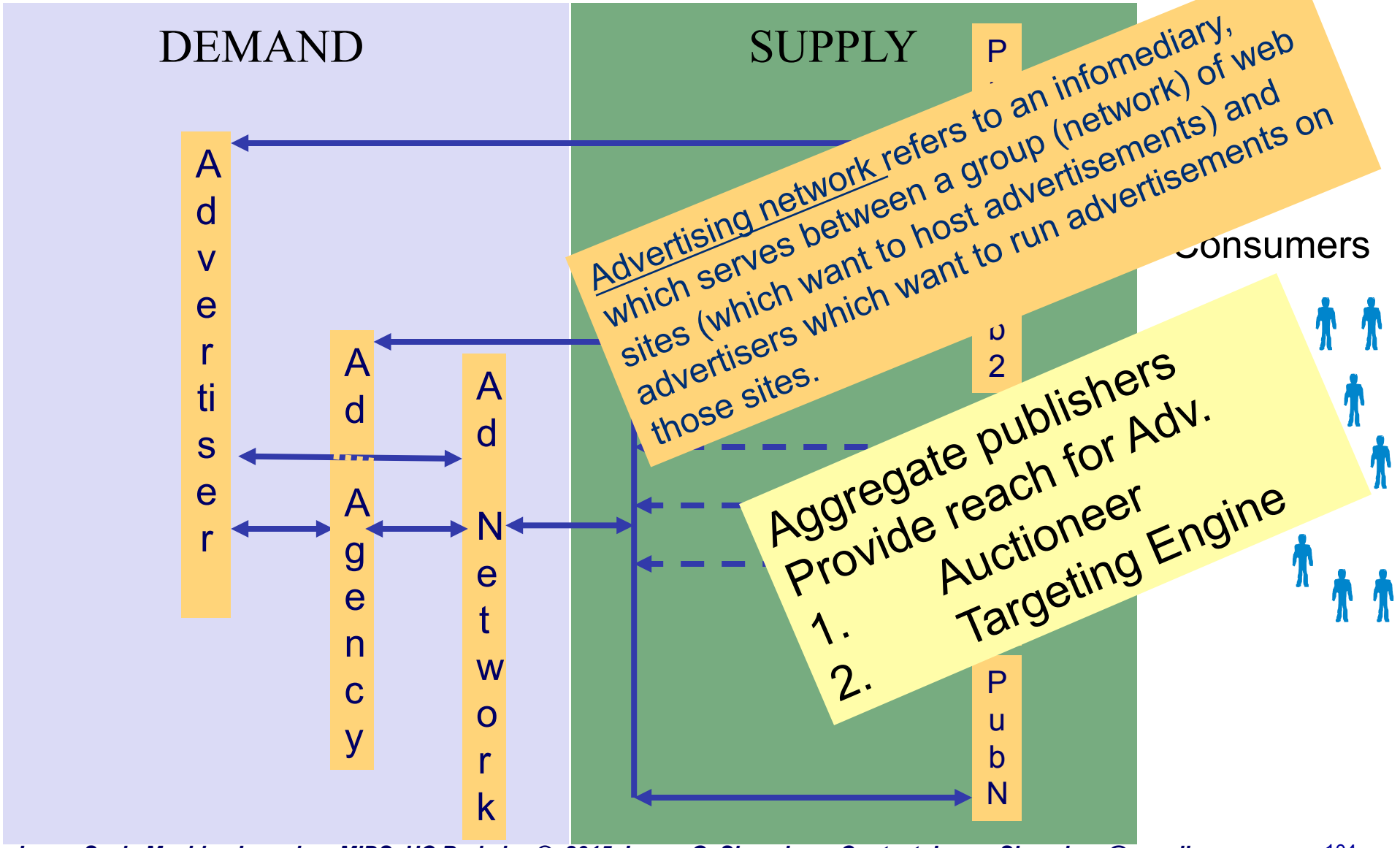
- **Supply can be fragmented outside of search**
- **Publishers maybe small and not have a sales team**

- **Led to the development of different types of marketplaces**
  - Publishers bring their produce for sale
  - Advertisers (or their representatives) come to buy
  - Ad Networks
  - …. and later
    - Ad Exchanges, Yield mgt and Demand side platforms

Advertising Network: Aggregates Publishers



Advertising Network: Aggregates Publishers



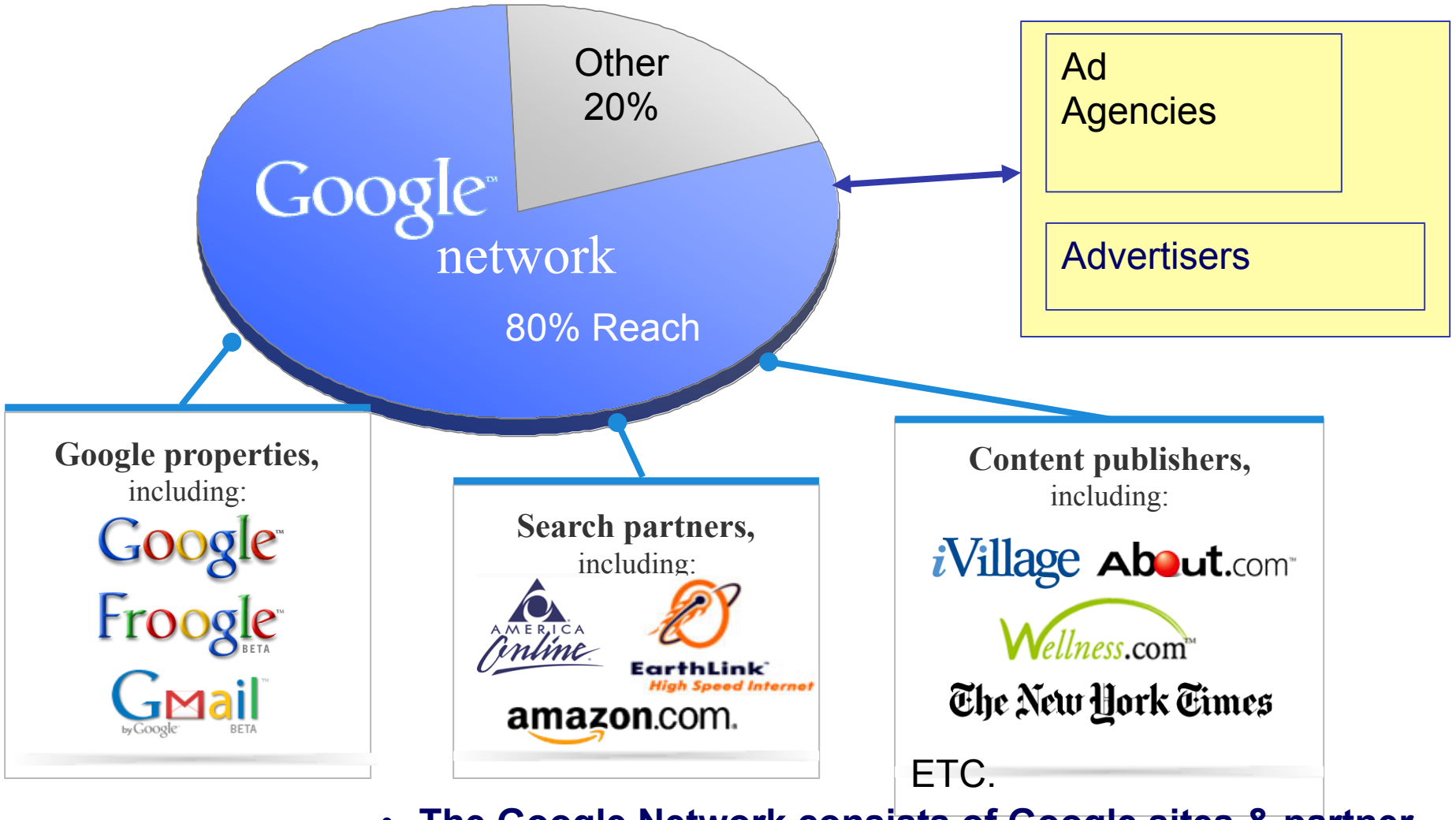
Ad Network

- **Key function is the aggregation of ad space *supply* from publishers and matching it with advertiser *demand.***
  - Aggregate webpage space where online ads can be embedded and, in addition, providing both targeting and auctioneering capabilities.
- **Ad networks were one of the first big directions taken to make this traditional market economy more efficient**
  - they took an economy of scale tack creating a huge marketplace that aggregated the supply of ad space from publishers
  - Advantage for advertisers
    - increased reach, reduced (economy of scale) pricing, reduced media-buyng effort, targeting and analytics.

Ad Network: Optimize ROI and Revenue

- **Their sole objective**
  - Optimize ROI for the advertiser and revenue for the ad network.
- **Today over 350 ad networks**
  - Some of the bigger ad networks include Google, Yahoo, and Microsoft.
- **DoubleClick (acquired by Google in 2007) was one of the first online ad networks, 1996-97 [http://en.wikipedia.org/wiki/DoubleClick].**

Example Ad Network: Google



- **The Google Network consists of Google sites & partner properties that use Google AdSense to serve AdWords ads**

* Adapted: comScore Media Metrix (September, 2004)

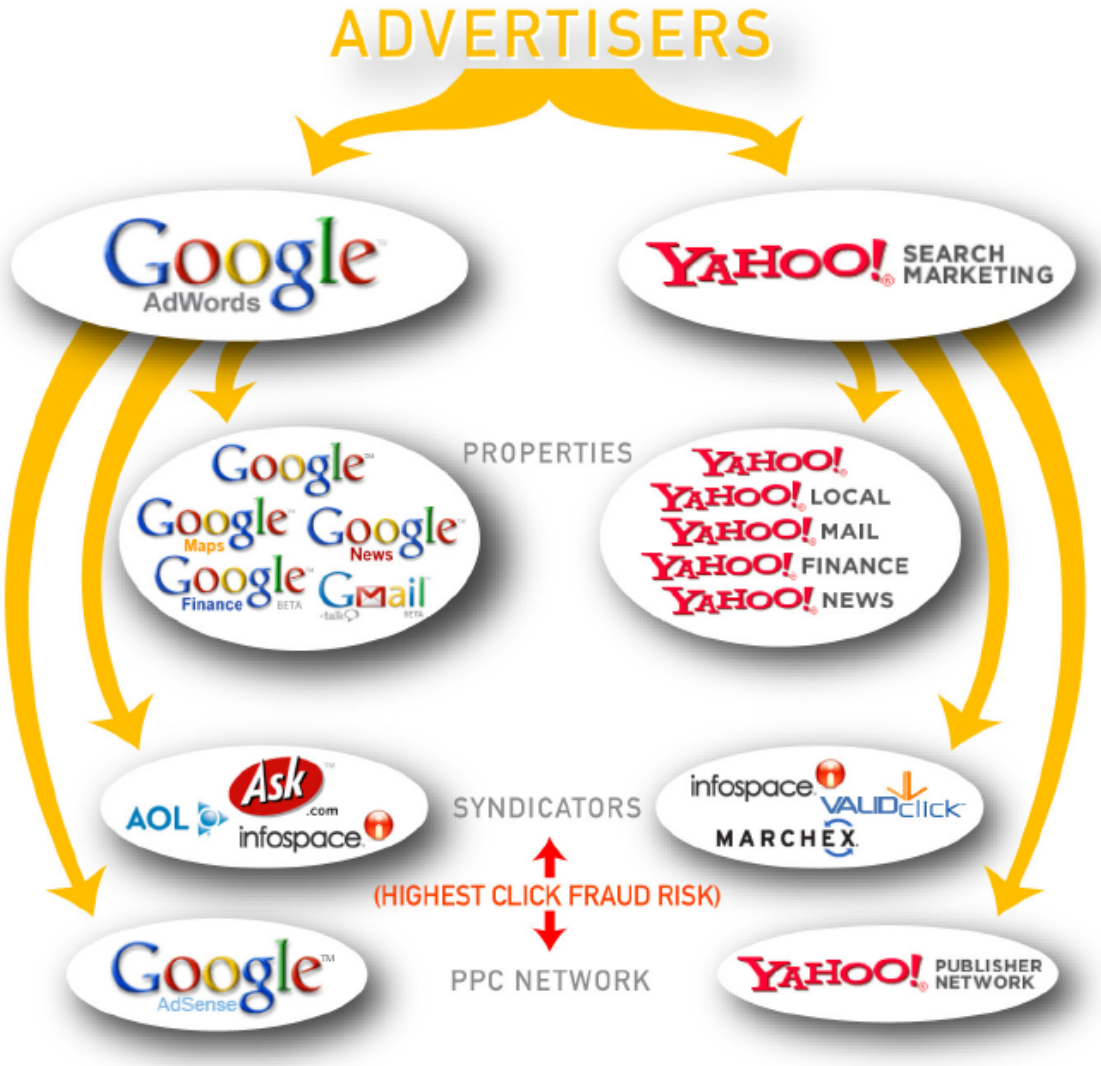
Yahoo and Google Ad Networks

Sponsored Search Advertising

Contextual Advertising

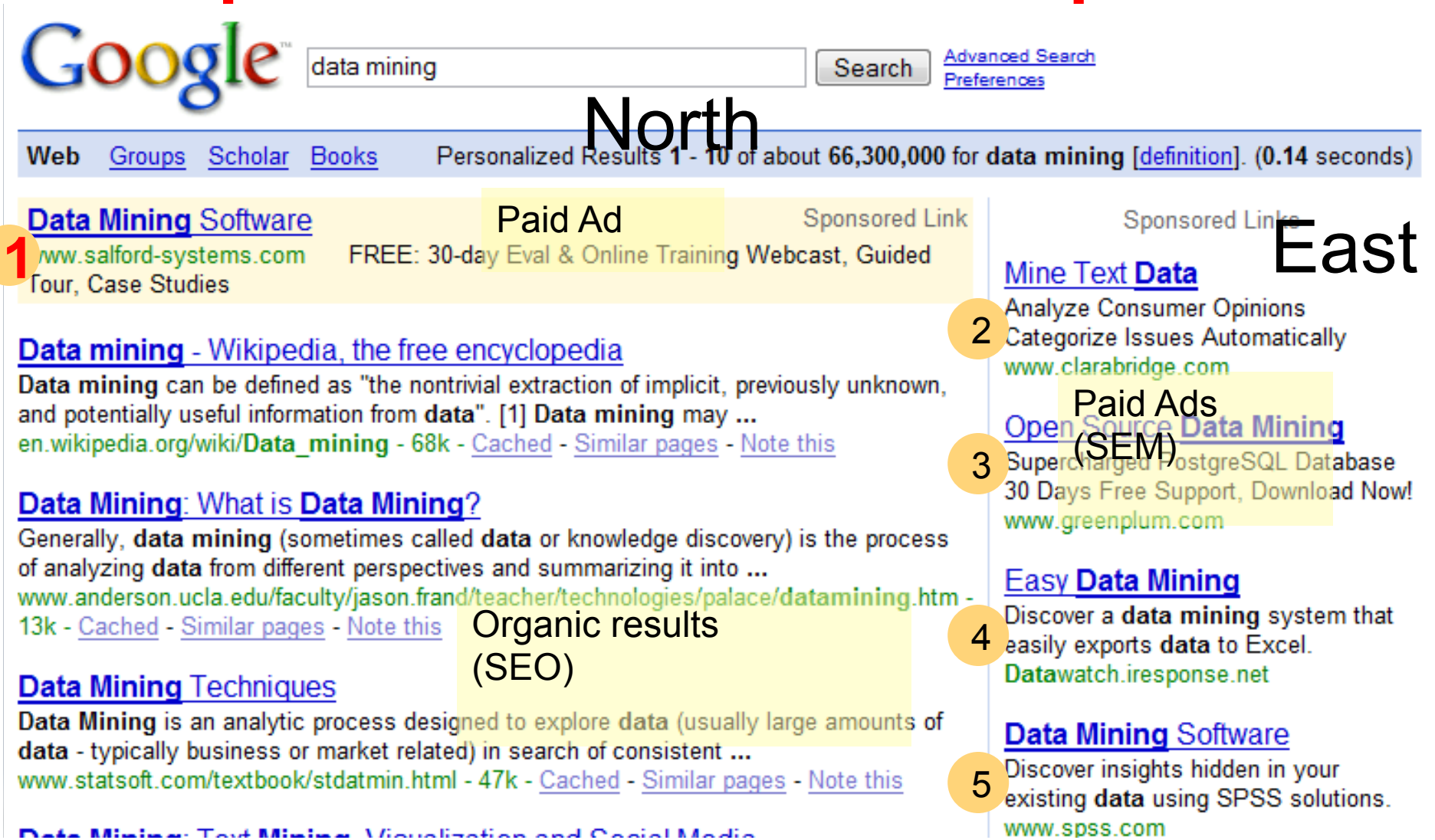
Sponsored Search Advertising

Contextual Advertising



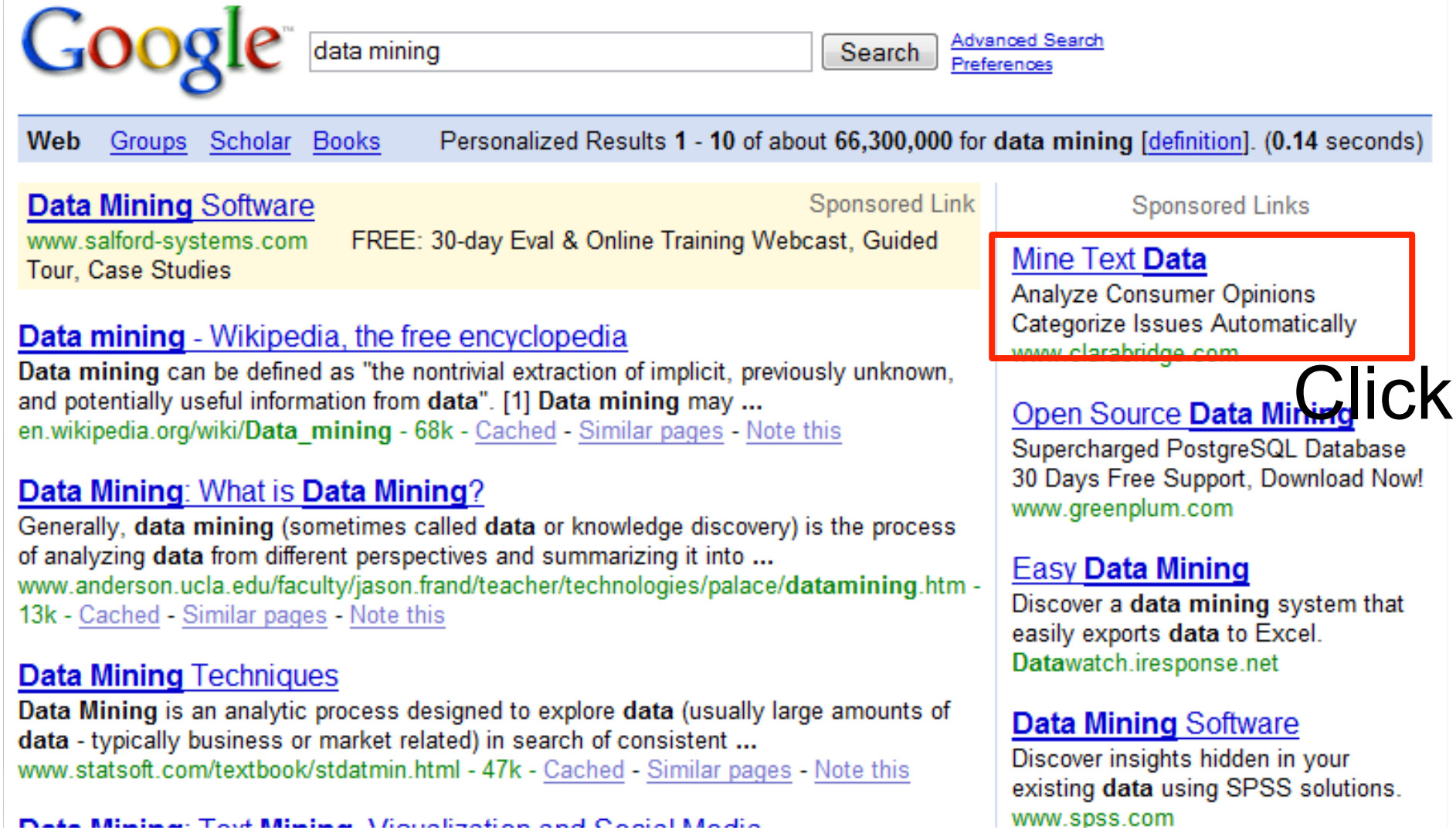
[Adapted from Think Partnership, 2007]

Sponsored Search: Ad Impression



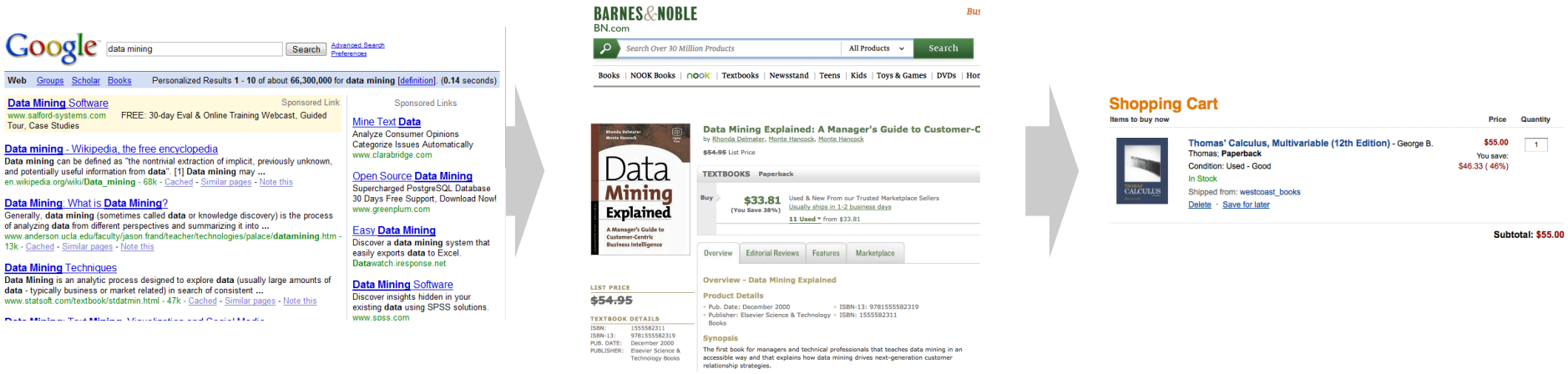
Ads are clearly distinguishable from the actual search results and they rotate

Sponsored Search: Click



Impressions→ Clicks → Transactions

Impressions → Clicks → Transactions



CPC ← RPC ← Revenue

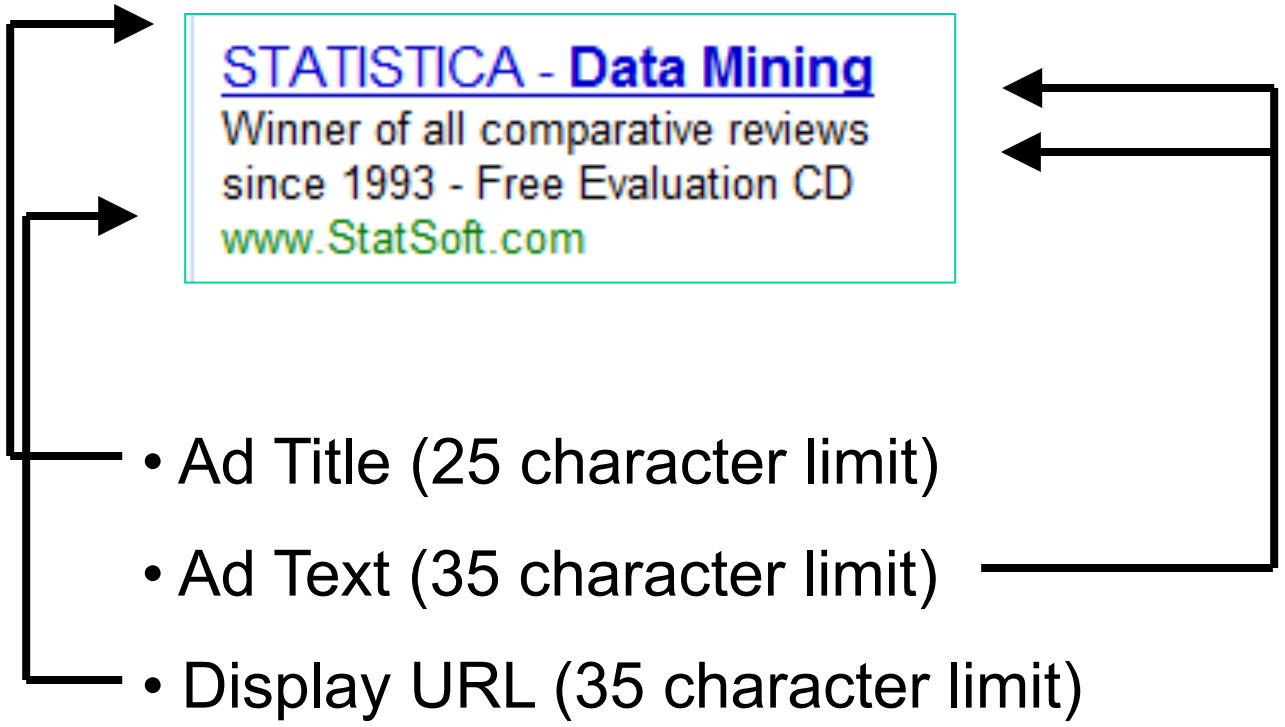
ctr = clicks / impressions
cost = clicks * cpc
cpa = cost / conversions

rpc = revenue /clicks
profit = revenue - cost
margin = profit / cost

Creating an online ad campaign

- **Typical workflow**
  1. Create advertiser account (name/address/Credit card details/etc.)
  2. Create ad creative
  3. Create an ad campaign
  4. Upload creative's
  5. Specify targeting constraints (e.g., keywords, categories, geo, dates)
  6. Specify bid price and budget
     - For a CPA network, just specify the bid price for an action; no need for keyword portfolio management
     - Deploy action beacon on landing page
  - Optimize ad creative/user-landing-experience, bid price: AB Test, DOE
- **Ad network/exchange**
  - Turn.com (CPA, CPC, CPM), Google (CPC, CPM), Yahoo (CPC, CPM), Right Media (CPM), Etc.
- **SEM: e.g., Efficient Frontier**

A Typical Text-based CPC Ad



1. Specify Start/End Dates of Campaign
2. Specify keywords+bids
3. Specify Budget
4. Specify other constraints (locality/publisher/etc)

E.g., Google AdWords

Advertise your business on Google

No matter what your budget, you can display your ads on Google and our advertising network. Pay only if people click your ads.

Sign up now »

Your ads appear beside related search results...

People click your ads...

...And connect to your business

Sign in to Google AdWords with your Google Account

Email:

Password:

Sign in

Forgot your password?

Learn about AdWords

- How it works
- Why it works
- Costs and payment
- For local businesses
- Assisted signup options
- Success stories

You create your ads
You create ads and choose keywords, which are words or phrases related to your business.
Get keyword ideas

Your ads appear on Google
When people search on Google using one of your keywords, your ad may appear next to the search results. Now you're advertising to an audience that's already interested in you.

You attract customers
People can simply click your ad to make a purchase or learn more about you. It's that easy!

Sign up now | Next topic »

Keywords are what people search for on Google.

Regional Targeting

Google AdWords: Regional and Local Targeting - Windows Internet Explorer

https://adwords.google.com/select/targeting.html

Google AdWords

Contact Us - Help

AdWords Home

AdWords Support

Overview
- AdWords Advantages
- Program Comparison
- Success Stories
- News and Updates
- Demos and Guides
- Industry Research
- Inside AdWords Blog

Getting Started
- Editorial Guidelines
- Step-by-Step
- Tips for Success
- Account Navigation
- Keyword Tools

Regional and Local Targeting: Sharpen Your Advertising Focus

With AdWords, you can target your ads to appear only in specific geographic locations. You can choose country-level targeting or narrow your focus to:

Region and city-level targeting: Show your ads to people searching for results in regional areas you choose. (Available in select countries.)

Region

City

Customized targeting: Show your ads to people searching for results in an area you define. (Available worldwide.)

Within your defined radius

Within your defined borders

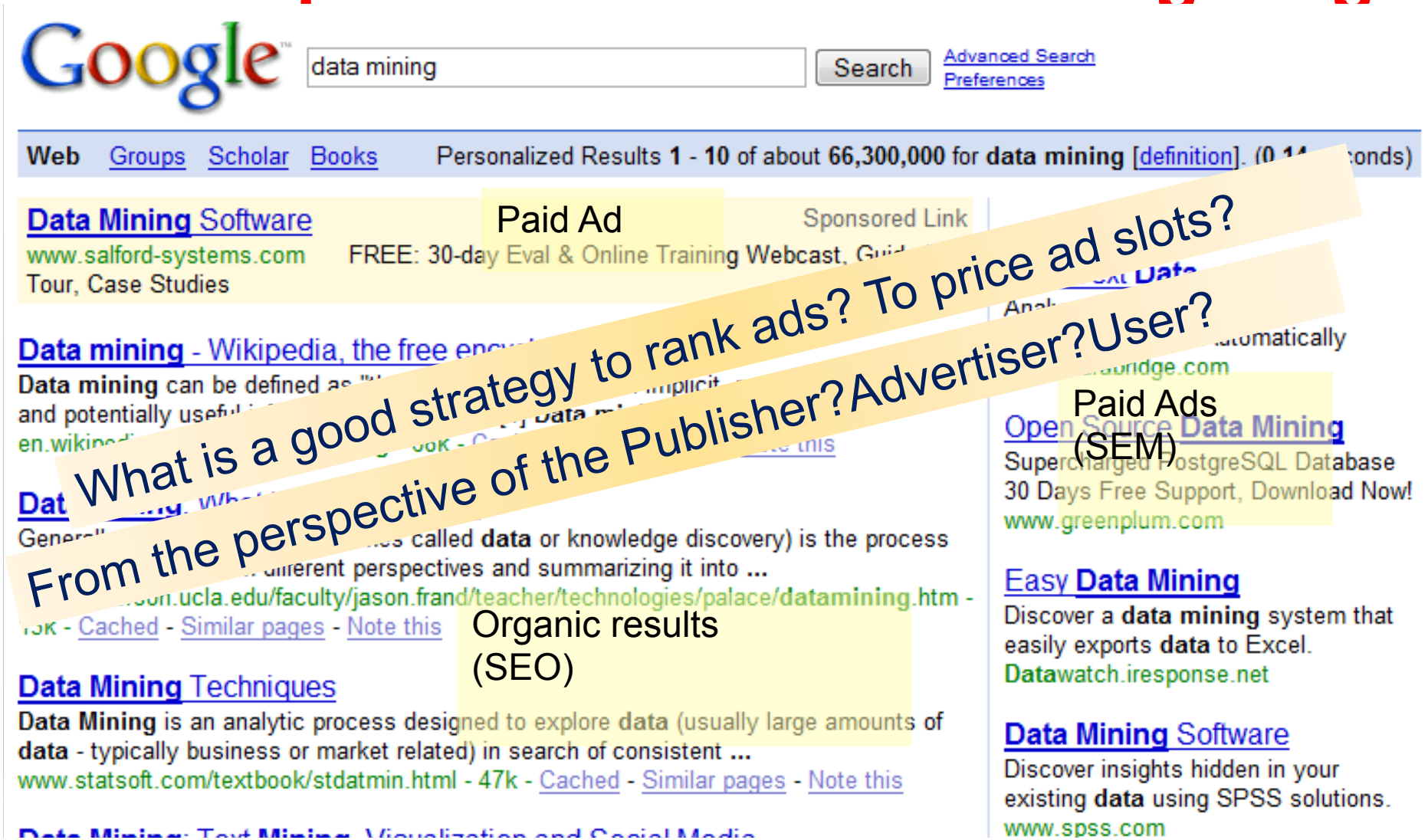
- **How does this benefit me?**
  When you target regional and local areas, you can reach the prospects who are most appropriate for your business and you can write ads that highlight special promotions or pricing based on geography.
- **Which ad targeting option is right for me?**
  Use regional and city-level targeting if you know which specific cities and regional areas are appropriate for your market. Choose customized targeting if you want to define your own target area. Indicate your area by choosing a point and a surrounding radius or by picking points to define a border.
- **How does this work?**
  The AdWords system may analyze a searcher's query (for example "London florist") to establish what location that person is searching for. The system may also take note of the person's Internet Protocol (IP) address to see where he or she is searching from.
- **I'm ready to reach new customers now. How do I get started?**
  All you need to do is create and activate an AdWords account.

See Local Targeting in Action

View our interactive demo to explore this targeting option and learn how to set up a local or regional AdWords campaign of your own. Start Demo

Done | Internet | Protected Mode: On | 100%

Sponsored Search: Ad targeting



Ads are clearly distinguishable from the actual search results and they rotate

Generalized 2nd Price (GSP) Auction

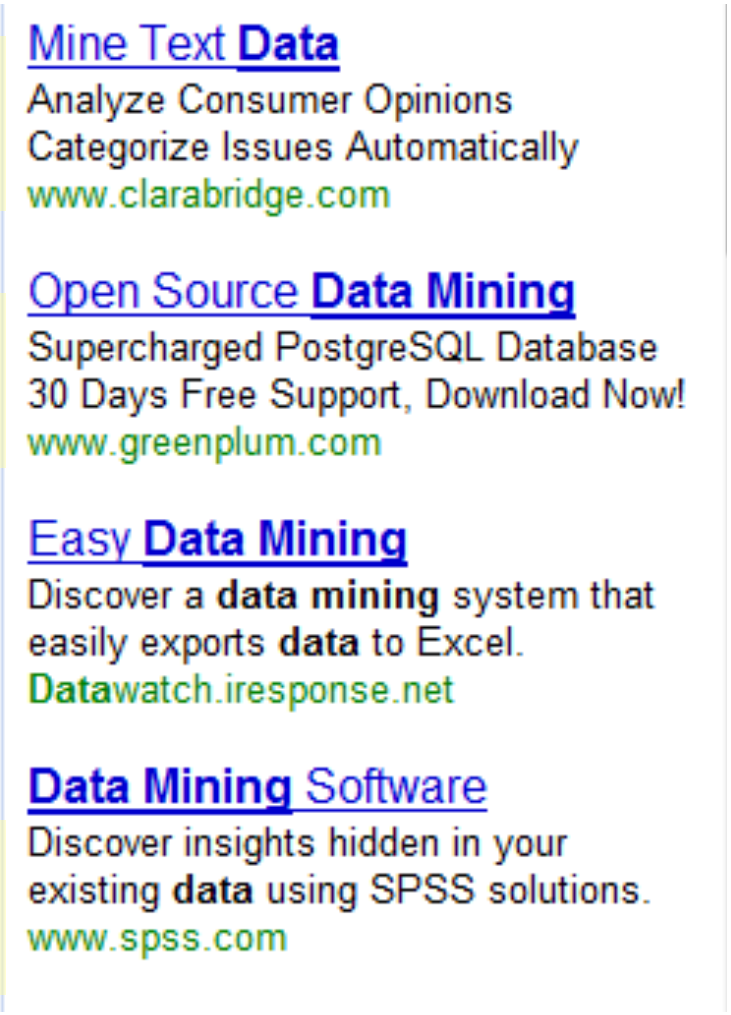
1. In a GSP, multiple items are up for auction;
2. The highest bidder wins the first item at the second price (+delta)
3. The second-highest bidder wins the second item at the third-highest price, and so on

Bid = $10
PPC = $5

Bid = $5
PPC = $2

Bid = $2
PPC = $1

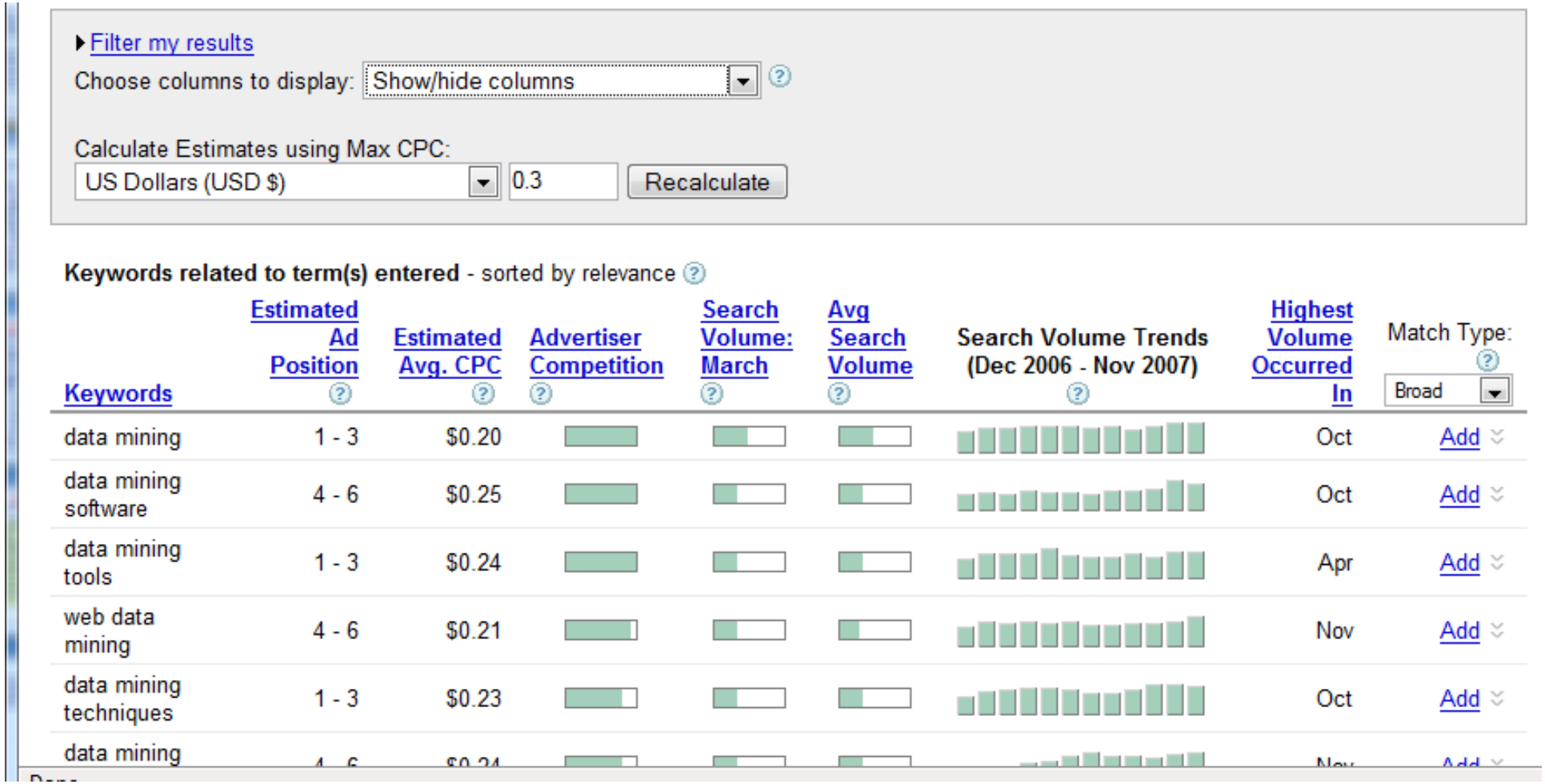
Bid = $1
PPC = $0.57



Introduced by Google in Feb 2002 (AdWords); overcomes the instability of GFP because by design the bidder is incentivized to pay the true value?!

Select Portfolio of Keywords and Bids

[https://adwords.google.com/select/KeywordToolExternal?defaultView=2]



Filter my results

Choose columns to display: Show/hide columns

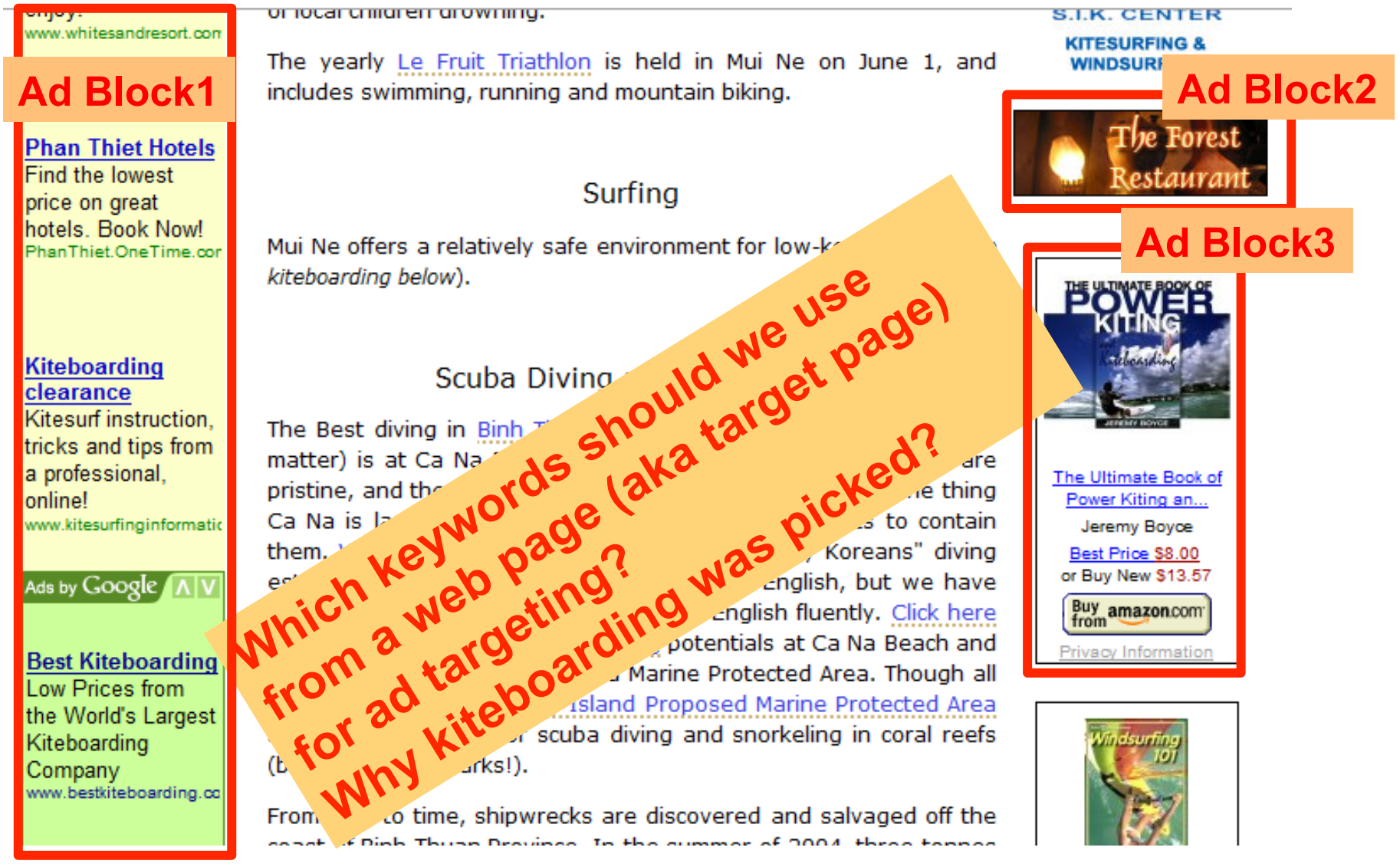
Calculate Estimates using Max CPC:

US Dollars (USD $) 0.3 Recalculate

Keywords related to term(s) entered - sorted by relevance

Keywords	Estimated Ad Position	Estimated Avg. CPC	Advertiser Competition	Search Volume: March	Avg Search Volume	Search Volume Trends (Dec 2006 - Nov 2007)	Highest Volume Occurred In	Match Type: Broad
data mining	1 - 3	$0.20					Oct	Add
data mining software	4 - 6	$0.25					Oct	Add
data mining tools	1 - 3	$0.24					Apr	Add
web data mining	4 - 6	$0.21					Nov	Add
data mining techniques	1 - 3	$0.23					Oct	Add
data mining	4 - 6	$0.24					Nov	Add

How are the ads targeted on this page?



- **Which keywords should we use from a web page (aka target page) for ad targeting?**

Keyword harvesting from TPs

- **Keyword harvesting from webpages**
  - Treat target page as a query but can be long. Suggest keywords. Which terms to extract? Train a LogReg classifier on hand-picked keywords and their features. Predict if a word is keyword or not. Study performed at Microsoft Research
  - [Yih et al., WWW 2006, "Finding advertising keywords on web pages"]

- **Overall the problems in traditional IR + more (see WWW, SIGIR CIKM)**
  - Synonym detection; entity extraction from queries; query disambiguation

Finding Advertising Keywords on Web Pages

Wen-tau Yih
Microsoft Research
1 Microsoft Way
Redmond, WA 98052
scottyih@microsoft.com

Joshua Goodman
Microsoft Research
1 Microsoft Way
Redmond, WA 98052
joshuago@microsoft.com

Vitor R. Carvalho
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
vitor@cs.cmu.edu

ABSTRACT

A large and growing number of web pages display contextual advertising based on keywords automatically extracted from the text of the page, and this is a substantial source of revenue supporting the web today. Despite the importance of this area, little formal, published research exists. We describe a system that learns how to extract keywords from web pages for advertisement targeting. The system uses a number of features, such as term frequency of each potential keyword, inverse document frequency, presence in meta-data, and how often the term occurs in search query logs. The system is trained with a set of example pages that have been hand-labeled with "relevant" keywords. Based on this training, it can then extract new keywords from previously unseen pages. Accuracy is substantially better than several baseline systems.

Categories and Subject Descriptors

H.3.1 [**Content Analysis and Indexing**]: Abstracting methods; H.4.m [**Information Systems**]: Miscellaneous

General Terms

Algorithms, experimentation

Keywords

keyword extraction, information extraction, advertising

of information, to find prominent keywords on that page. These keywords are then sent to an advertising system, which matches the keywords against a database of ads. Advertising appropriate to the keyword is displayed to the user. Typically, if a user clicks on the ad, the advertiser is charged a fee, most of which is given to the web page owner, with a portion kept by the advertising service.

Picking appropriate keywords helps users in at least two ways. First, choosing appropriate keywords can lead to users seeing ads for products or services they would be interested in purchasing. Second, the better targeted the advertising, the more revenue that is earned by the web page provider, and thus the more interesting the applications that can be supported. For instance, free blogging services and free email accounts with large amounts of storage are both enabled by good advertising systems

From the perspective of the advertiser, it is even more important to pick good keywords. For most areas of research, such as speech recognition, a 10% improvement leads to better products, but the increase in revenue is usually much smaller. For keyword selection, however, a 10% improvement might actually lead to nearly a 10% higher click-through-rate, directly increasing potential revenue and profit.

In this paper, we systematically investigated several different aspects of keyword extraction. First, we compared looking at each occurrence of a word or phrase in a document separately, versus combining all of our information about the word or phrase. We also compared approaches that look at the word or phrase monolithically to approaches

TextRank – Keyword Extraction

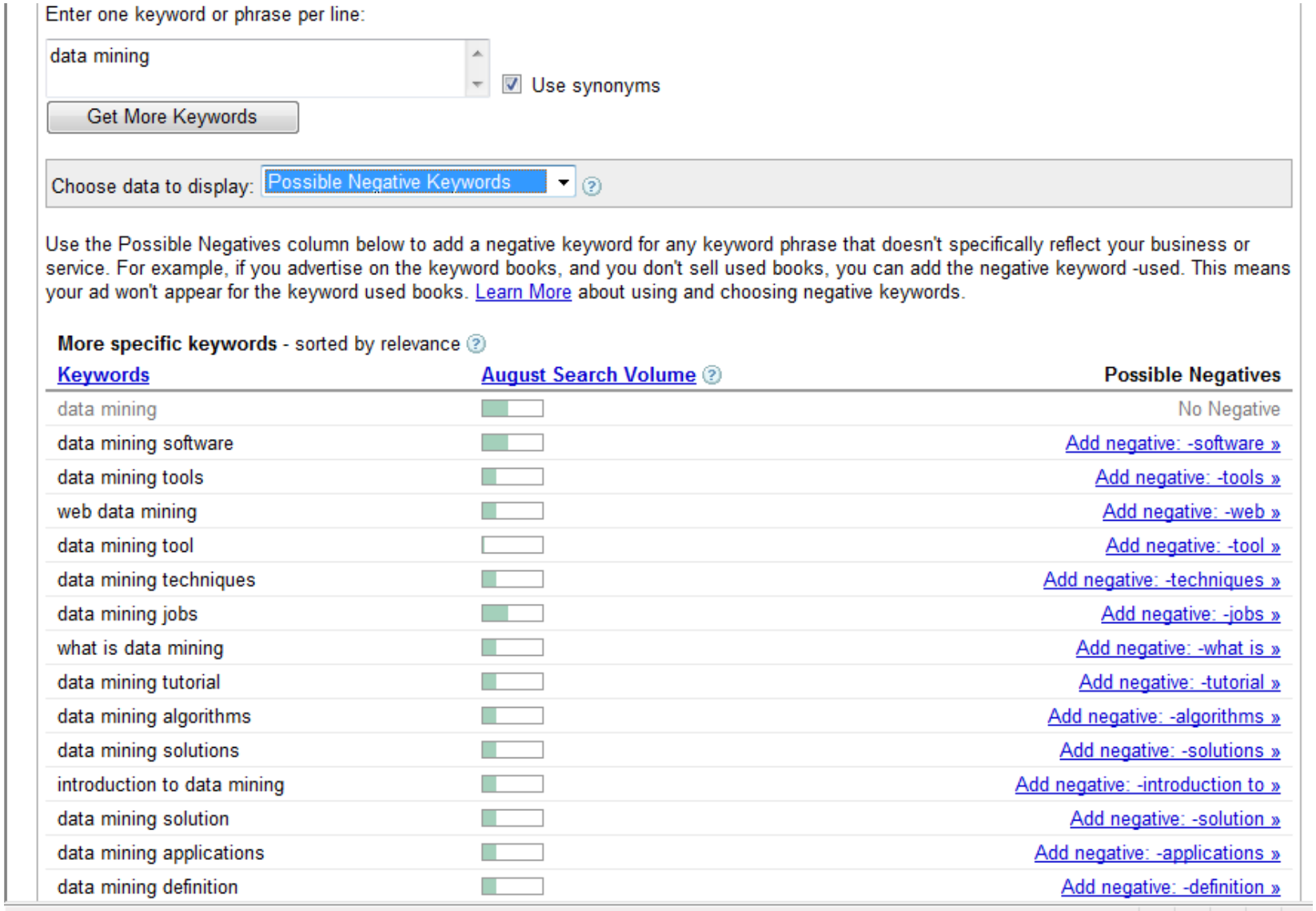
- **Identify important words in a text**
- **Keywords useful for**
  - Automatic indexing
  - Terminology extraction
  - Within other applications: Information Retrieval, Text Summarization, Word Sense Disambiguation
- **Previous work**
  - mostly supervised learning
  - genetic algorithms [Turney 1999], Naïve Bayes [Frank 1999], rule induction [Hulth 2003]
- **Extract keywords using TextRank:**
  - Bringing Order into Texts by treating text as a graph
  - Rada Mihalcea and Paul Tarau, EMNLP Conference 2004

PageRank Input: Markov Process

- **Nodes:**
  - words

- **Transition matrix**
  - Co-occurrence

Keyword Suggester



Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Mid term; Feedback/Evaluation
  - Homework HW6, HW7, HW9
  - AWS: no access
  - Async lecture recap plus Q&A (PageRank)
  - Contextual advertising
  - Text as graph: TextRank
    - Keyword extraction (from text/target pages)
    - Text Summarization
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

TextRank – Keyword Extraction

- **Identify important words in a text**
- **Keywords useful for**
  - Automatic indexing
  - Terminology extraction
  - Within other applications: Information Retrieval, Text Summarization, Word Sense Disambiguation
- **Previous work**
  - mostly supervised learning
  - genetic algorithms [Turney 1999], Naïve Bayes [Frank 1999], rule induction [Hulth 2003]

[TextRank: Bringing Order into Texts
Rada Mihalcea and Paul Tarau, EMNLP Conference 2004]

TextRank – Keyword Extraction

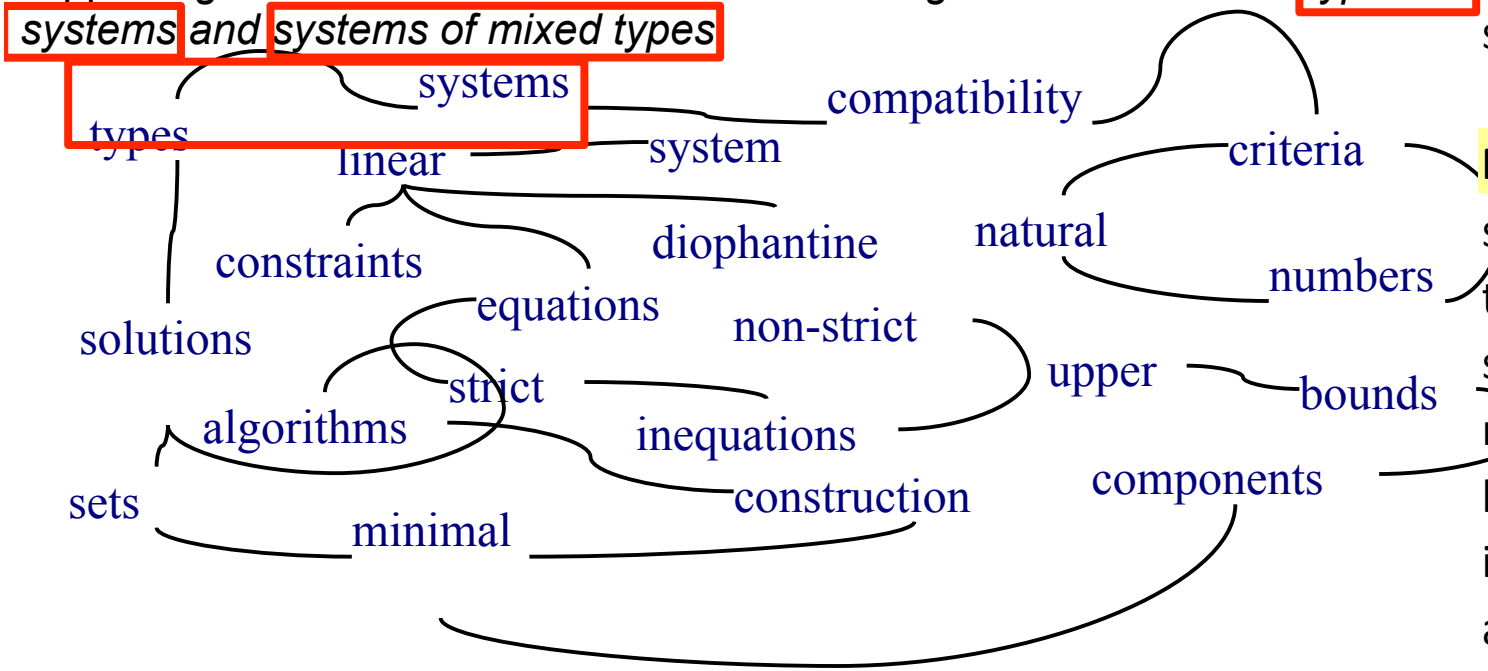
- **Identify important words in a text**
- **Keywords useful for**
  - Automatic indexing
  - Terminology extraction
  - Within other applications: Information Retrieval, Text Summarization, Word Sense Disambiguation
- **Previous work**
  - mostly supervised learning
  - genetic algorithms [Turney 1999], Naïve Bayes [Frank 1999], rule induction [Hulth 2003]
- **Extract keywords using TextRank:**
  - Bringing Order into Texts by treating text as a graph
  - Rada Mihalcea and Paul Tarau, EMNLP Conference 2004

Text a graph

- **Crowd source the importance of a webpage plus, say, PageRank**
  - Graph-based ranking algorithms like Kleinberg's HITS algorithm (Kleinberg, 1999) or Google's PageRank (Brin and Page, 1998) have been successfully used in citation analysis, social networks, and the analysis of the link-structure of the World Wide Web.
  - Arguably, these algorithms can be singled out as key elements of the paradigm-shift triggered in the field of Web search technology, by providing a Web page ranking mechanism that relies on the collective knowledge of Web architects rather than individual content analysis of Web pages
- **Graph-based ranking model for text processing: TextRank**
  - Lexical or semantic graphs extracted from natural language documents, results in a graph-based ranking model that can be applied to a variety of natural language processing applications

TextRank: An Example

Compatibility of systems of linear constraints over the set of natural numbers Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types of systems and systems of mixed types



TextRank

numbers (1.46)
inequations (1.45)
linear (1.29)
diophantine (1.28)
upper (0.99)
bounds (0.99)
strict (0.77)

Frequency

systems (4)
types (4)
solutions (3)
minimal (3)
linear (2)
inequations (2)
algorithms (2)

Keywords by TextRank: *linear constraints, linear diophantine equations, natural numbers, non-strict inequations, strict inequations, upper bounds*

Keywords by human annotators: *linear constraints, linear diophantine equations, non-strict inequations, set of natural numbers, strict inequations, upper bounds*

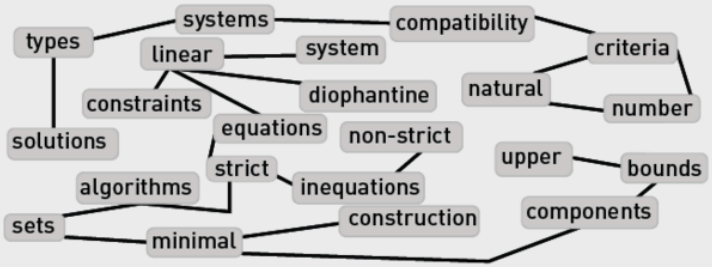
100% match

13.9 Spark SQL and TextRank

TextRank: An Example | Display | Print | 51 of 61

TextRank: An Example

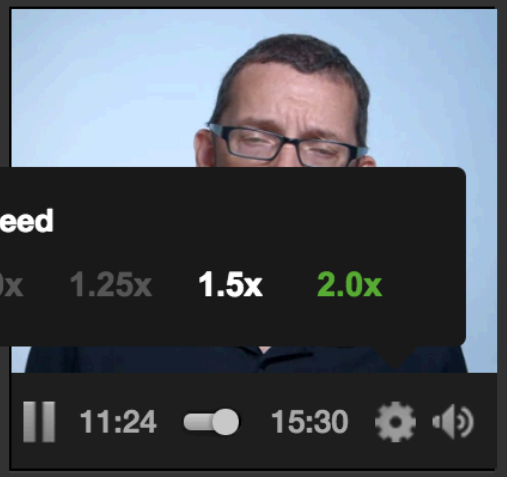
Compatibility of systems of linear constraints over the set of natural numbers Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generation sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types of systems and systems of mixed types.



TextRank
numbers (1.46)
inequations (1.45)
linear (1.29)
diophantine (1.28)
upper (0.99)
bounds (0.99)
strict (0.77)

Frequency
systems (4)
types (4)
solutions (3)
minimal (3)
linear (2)
inequations (2)
algorithms (2)

Keywords by TextRank: *linear constrains, linear diophantine equations natural numbers, non-strict inequations, strict inequations, upper bounds* ~ 100% match
Keywords by human annotators: *linear constraints, linear diophantine equations, non-strict inequations, set of natural numbers, strict inequations, upper bounds*



https://learn.datascience.berkeley.edu/mod/page/view.php?id=11114

TextRank: main steps

Build graph

PageRank

Sort and select

1. *Identify text units that best define the task at hand, and add them as vertices in the graph.*
2. *Identify relations that connect such text units, and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.*
3. *Iterate the graph-based ranking algorithm until convergence.*
4. *Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.*

TextRank: 2 possible applications

- **(1) A keyword extraction task, consisting of the selection of keyphrases representative for a given text; and**
  - Such keywords may constitute useful entries for building an automatic index for a document collection, can be used to classify a text, or may serve as a concise summary for a given document.
- **(2) A sentence extraction task, consisting of the identification of the most "important" sentences in a text, which can be used to build extractive summaries.**

TextRank Implementation

- **Have a look at the following CODE:**
  - http://www.davidadamojr.com/textrank-implementation-in-python-github-repo/
  - https://github.com/davidadamojr/TextRank
- **This is a python implementation of TextRank for automatic keyword and sentence extraction (summarization) as done in**
  - https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf .
- **However, this implementation uses Levenshtein Distance (simplifies the code) as the relation between text units.**
- **This implementation carries out automatic keyword and sentence extraction on 10 articles gotten from http://theonion.com**
  - 100 word summary
  - Number of keywords extracted is relative to the size of the text (a third of the number of nodes in the graph)
  - Adjacent keywords in the text are concatenated into keyphrases

Similarity is key in the adjacency matrix construction

- **Any relation that can be defined between two lexical units is a potentially useful connection (edge) that can be added between two such vertices. We are using a co-occurrence relation, controlled by the distance between word occurrences:**
  - two vertices are connected if their corresponding lexical units co-occur within a window of maximum words, where can be set anywhere from 2 to 10 words.
  - Co-occurrence links express relations between syntactic elements, and similar to the semantic links found useful for the task of word sense disambiguation (Mihalcea et al., 2004), they represent cohesion indicators for a given text

Filter out words with certain POS

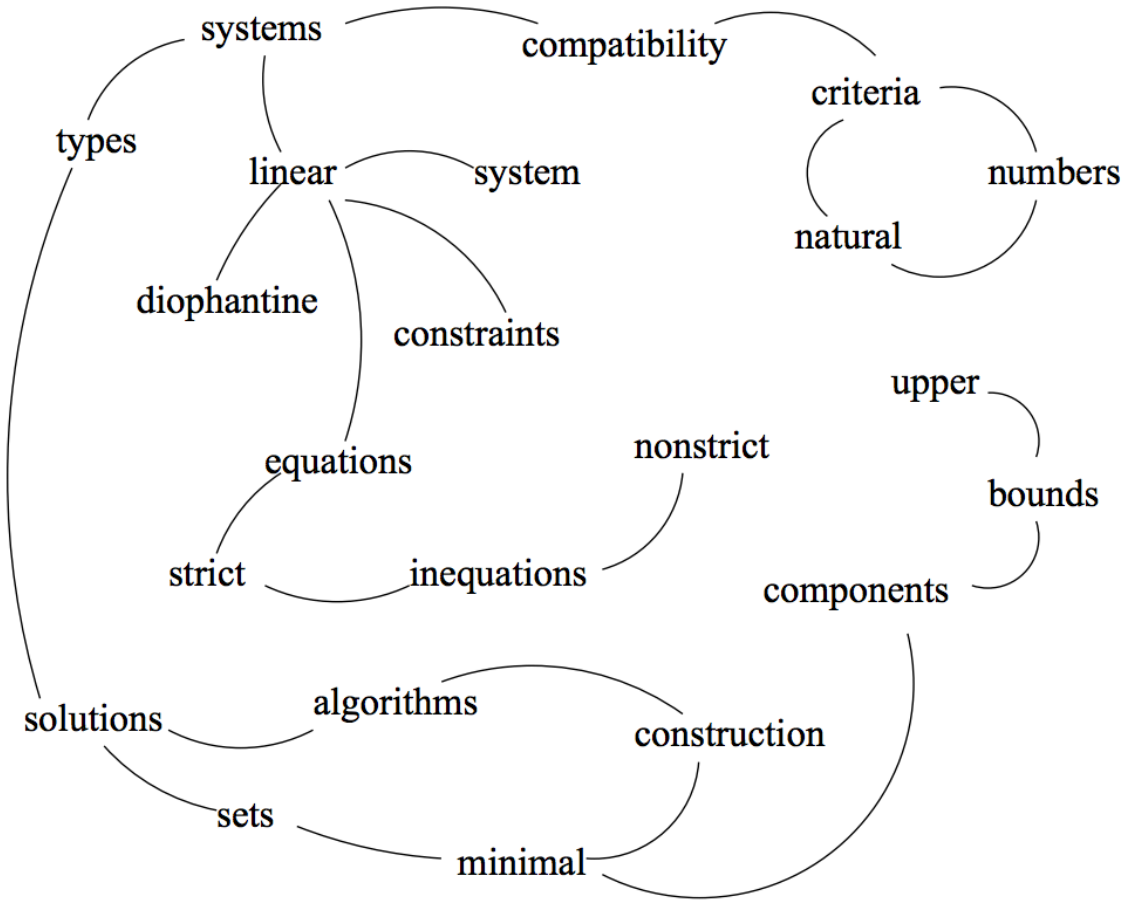
- The vertices added to the graph can be restricted with syntactic filters, which select only lexical units of a certain part of speech. One can for instance consider only nouns and verbs for addition to the graph, and consequently draw potential edges based only on relations that can be established between nouns and verbs.

Build a graph based upon co-occurrence and run PageRank

- **Similarity defined in terms of co-occurrence**
  - Next, all lexical units that pass the syntactic filter are added to the graph, and an edge is added between those lexical units that co-occur within a window of words.
- **Run pagerank**
  - After the graph is constructed (undirected unweighted graph), the score associated with each vertex is set to an initial value of 1, and the TEXTRank ranking algorithm described below is run on the graph for several iterations until it converges – usually for 20-30 iterations, at a threshold of 0.0001.

Compatibility of systems of linear constraints over the set of natural numbers. Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types systems and systems of mixed types.

Word co-occurrence based graph



Use uncontrolled vocab of keywords

Evaluation

- Test with 500 Abstracts
- Inspec abstracts are from journal papers from Computer Science and Information Technology.
- Each abstract comes with two sets of keywords assigned by professional indexers: controlled keywords, restricted to a given thesaurus, and uncontrolled keywords, freely assigned by the indexers. The authors follow the evaluation approach from (Hulth, 2003), and use the uncontrolled set of keywords.

The data set used in the experiments is a collection of 500 abstracts from the *Inspec* database, and the corresponding manually assigned keywords. This is the same test data set as used in the keyword extraction experiments reported in (Hulth, 2003). The *Inspec* abstracts are from journal papers from Computer Science and Information Technology. Each abstract comes with two sets of keywords assigned by professional indexers: controlled keywords, restricted to a given thesaurus, and uncontrolled keywords, freely assigned by the indexers. We follow the evaluation approach from (Hulth, 2003), and use the uncontrolled set of keywords.

In her experiments, Hulth is using a total of 2000 abstracts, divided into 1000 for training, 500 for development, and 500 for test[2]. Since our approach is completely unsupervised, no training/development data is required, and we are only using the test docu-

- **Is textRank supervised or unsupervised algorithm?**

500 Abstracts

Method	Assigned		Correct		Precision	Recall	F-measure
	Total	Mean	Total	Mean			
TextRank							
Undirected, Co-occ.window=2	6,784	13.7	2,116	4.2	**31.2**	43.1	**36.2**
Undirected, Co-occ.window=3	6,715	13.4	1,897	3.8	28.2	38.6	32.6
Undirected, Co-occ.window=5	6,558	13.1	1,851	3.7	28.2	37.7	32.2
Undirected, Co-occ.window=10	6,570	13.1	1,846	3.7	28.1	37.6	32.2
Directed, forward, Co-occ.window=2	6,662	13.3	2,081	4.1	31.2	42.3	35.9
Directed, backward, Co-occ.window=2	6,636	13.3	2,082	4.1	31.2	42.3	35.9
Hulth (2003)							
Ngram with tag	7,815	15.6	1,973	3.9	25.2	**51.7**	33.9
NP-chunks with tag	4,788	9.6	1,421	2.8	29.7	37.2	33.0
Pattern with tag	7,012	14.0	1,523	3.1	21.7	39.9	28.1

Table 1: Results for automatic keyword extraction using TextRank or supervised learning (Hulth, 2003)

- **TextRank Code on a single core machine in python**

https://www.dropbox.com/s/3clpo20ns3mmb1l/TextRank-with-trace-JGS-2016-03-08.ipynb?dl=0

TextRank Code and data

- https://www.dropbox.com/sh/3s70t7wgxte30kp/AABoMJrZY83pvx-F8VL_yiDra?dl=0

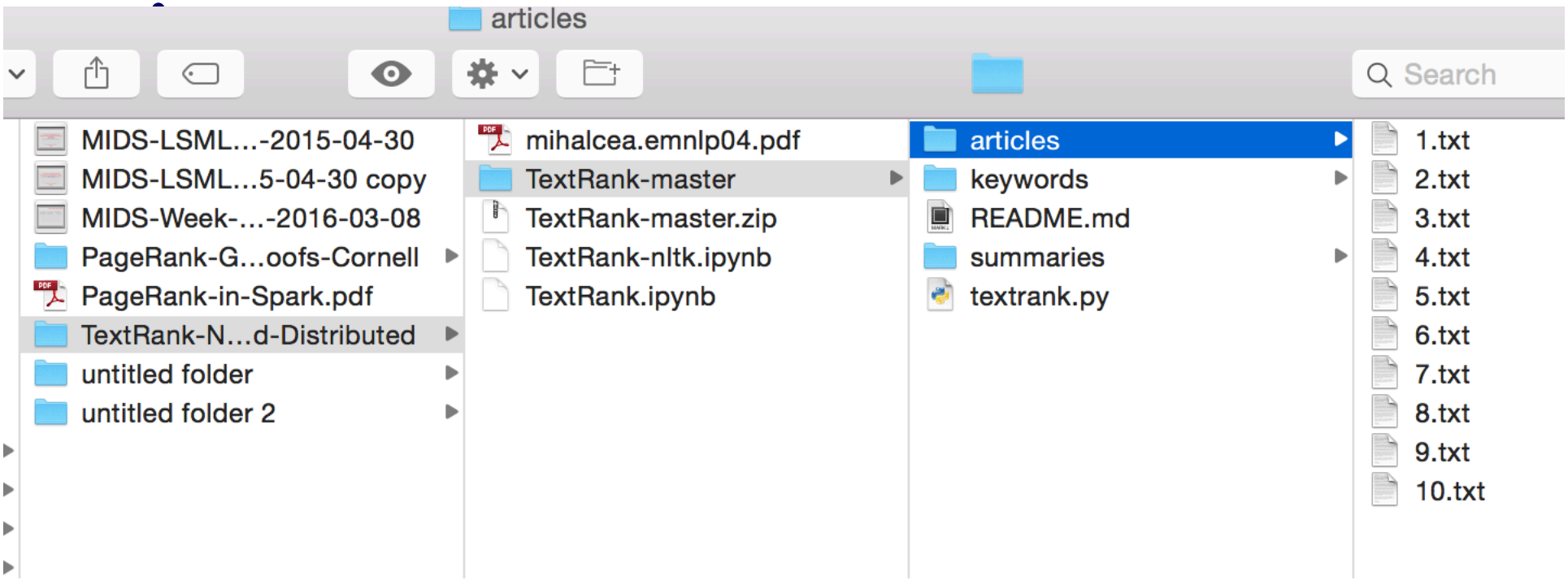
```
"""
From this paper: https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf

External dependencies: nltk, numpy, networkx

Based on https://gist.github.com/voidfiles/1646117
"""

import io
import nltk
import itertools
from operator import itemgetter
import networkx as nx
import os

#apply syntactic filters based on POS tags
def filter_for_tags(tagged, tags=['NN', 'JJ', 'NNP']):
    return [item for item in tagged if item[1] in tags]

def normalize(tagged):
    return [(item[0].replace('.', ''), item[1]) for item in tagged]

def unique_everseen(iterable, key=None):
    "List unique elements, preserving order. Remember all elements ever seen."
    # unique_everseen('AAAABBBCCDAABBB') --> A B C D
    # unique_everseen('ABBCcAD', str.lower) --> A B C D
    seen = set()
    seen_add = seen.add
    if key is None:
        for element in itertools.ifilterfalse(seen.__contains__, iterable):
            seen_add(element)
            yield element
    else:
        for element in iterable:
            k = key(element)
            if k not in seen:
                seen_add(k)
                yield element

```

```
def lDistance(firstString, secondString):
    "Function to find the Levenshtein distance between two words/sentences - gotten from http://rosettacode.org/w
    if len(firstString) > len(secondString):
        firstString, secondString = secondString, firstString
    distances = range(len(firstString) + 1)
    for index2, char2 in enumerate(secondString):
        newDistances = [index2 + 1]
        for index1, char1 in enumerate(firstString):
            if char1 == char2:
                newDistances.append(distances[index1])
            else:
                newDistances.append(1 + min((distances[index1], distances[index1+1], newDistances[-1])))
        distances = newDistances
    return distances[-1]

def buildGraph(nodes):
    "nodes - list of hashables that represents the nodes of the graph"
    gr = nx.Graph() #initialize an undirected graph
    gr.add_nodes_from(nodes)
    nodePairs = list(itertools.combinations(nodes, 2))

    #add edges to the graph (weighted by Levenshtein distance)
    for pair in nodePairs:
        firstString = pair[0]
        secondString = pair[1]
        levDistance = lDistance(firstString, secondString)
        gr.add_edge(firstString, secondString, weight=levDistance)

    return gr

```

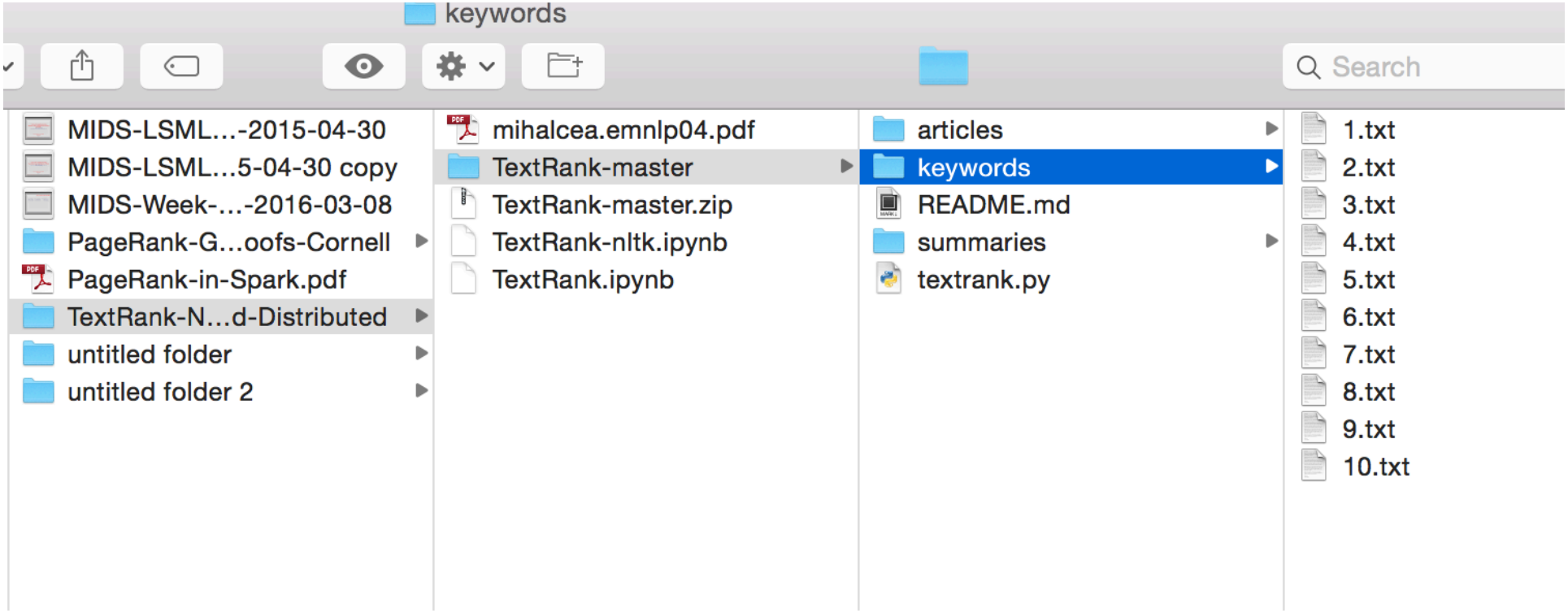
```python
def extractKeyphrases(text):
    # tokenize the text using nltk
    wordTokens = nltk.word_tokenize(text)

    # assign POS tags to the words in the text
    tagged = nltk.pos_tag(wordTokens)
    textlist = [x[0] for x in tagged]

    tagged = filter_for_tags(tagged)
    tagged = normalize(tagged)

    unique_word_set = unique_everseen([x[0] for x in tagged])
    word_set_list = list(unique_word_set)

    # this will be used to determine adjacent words in order to construct keyphrases with two words
    graph = buildGraph(word_set_list)
    # pageRank - initial value of 1.0, error tolerance of 0,0001,
    calculated_page_rank = nx.pagerank(graph, weight='weight')
    # most important words in ascending order of importance
    keyphrases = sorted(calculated_page_rank, key=calculated_page_rank.get, reverse=True)
    # the number of keyphrases returned will be relative to the size of the text (a third of the number of vertices)
    aThird = len(word_set_list) / 3
    keyphrases = keyphrases[0:aThird+1]
    # take keyphrases with multiple words into consideration as done in the paper - if two words are adjacent
    # in the text and are selected as keywords, join them together
    modifiedKeyphrases = set([])
    dealtWith = set([]) #keeps track of individual keywords that have been joined to form a keyphrase
    i = 0
    j = 1
    while j < len(textlist):
        firstWord = textlist[i]
        secondWord = textlist[j]
        if firstWord in keyphrases and secondWord in keyphrases:
            keyphrase = firstWord + ' ' + secondWord
            modifiedKeyphrases.add(keyphrase)
            dealtWith.add(firstWord)
            dealtWith.add(secondWord)
        else:
            if firstWord in keyphrases and firstWord not in dealtWith:
                modifiedKeyphrases.add(firstWord)
            #if this is the last word in the text, and it is a keyword,
            #it definitely has no chance of being a keyphrase at this point
            if j == len(textlist)-1 and secondWord in keyphrases and secondWord not in dealtWith:
                modifiedKeyphrases.add(secondWord)
        i = i + 1
        j = j + 1

    return modifiedKeyphrases
```

```
125
126 def extractSentences(text):
127     sent_detector = nltk.data.load('tokenizers/punkt/english.pickle')
128     sentenceTokens = sent_detector.tokenize(text.strip())
129     graph = buildGraph(sentenceTokens)
130
131     calculated_page_rank = nx.pagerank(graph, weight='weight')
132
133     #most important sentences in ascending order of importance
134     sentences = sorted(calculated_page_rank, key=calculated_page_rank.get, reverse=True)
135
136     #return a 100 word summary
137     summary = ' '.join(sentences)
138     summaryWords = summary.split()
139     summaryWords = summaryWords[0:101]
140     summary = ' '.join(summaryWords)
141
142     return summary
143
144 def writeFiles(summary, keyphrases, fileName):
145     "outputs the keyphrases and summaries to appropriate files"
146     print "Generating output to " + 'TextRank-master/keywords/' + fileName
147     keyphraseFile = io.open('TextRank-master/keywords/' + fileName, 'w')
148     for keyphrase in keyphrases:
149         keyphraseFile.write(keyphrase + '\n')
150     keyphraseFile.close()
151
152     print "Generating output to " + 'TextRank-master/summaries/' + fileName
153     summaryFile = io.open('TextRank-master/summaries/' + fileName, 'w')
154     summaryFile.write(summary)
155     summaryFile.close()
156
157     print "-"
158
159 def mainRunner():
160     nltk.download()
161     #retrieve each of the articles
162     articles = os.listdir("TextRank-master/articles")
163     for article in articles:
164         print 'Reading articles/' + article
165         articleFile = io.open('TextRank-master/articles/' + article, 'r')
166         text = articleFile.read()
167         keyphrases = extractKeyphrases(text)
168         summary = extractSentences(text)
169         writeFiles(summary, keyphrases, article)
```

Articles test dataset



Outputs: Keywords folder and Summaries folder



```
    articles = os.listdir("TextRank-master/articles")
    for article in articles:
        print 'Reading articles/' + article
        articleFile = io.open('TextRank-master/articles/' + article, 'r')
        text = articleFile.read()
        keyphrases = extractKeyphrases(text)
        summary = extractSentences(text)
        writeFiles(summary, keyphrases, article)

```

```
Reading articles/1.txt
Generating output to TextRank-master/keywords/1.txt
Generating output to TextRank-master/summaries/1.txt
-
Reading articles/10.txt
Generating output to TextRank-master/keywords/10.txt
Generating output to TextRank-master/summaries/10.txt
-
Reading articles/2.txt
Generating output to TextRank-master/keywords/2.txt
Generating output to TextRank-master/summaries/2.txt
-
Reading articles/3.txt
Generating output to TextRank-master/keywords/3.txt
Generating output to TextRank-master/summaries/3.txt
-
Reading articles/4.txt
Generating output to TextRank-master/keywords/4.txt
Generating output to TextRank-master/summaries/4.txt
-
Reading articles/5.txt
Generating output to TextRank-master/keywords/5.txt
Generating output to TextRank-master/summaries/5.txt
-
Reading articles/6.txt
Generating output to TextRank-master/keywords/6.txt
Generating output to TextRank-master/summaries/6.txt
-
Reading articles/7.txt
Generating output to TextRank-master/keywords/7.txt
Generating output to TextRank-master/summaries/7.txt
-
Reading articles/8.txt
Generating output to TextRank-master/keywords/8.txt
Generating output to TextRank-master/summaries/8.txt
-
Reading articles/9.txt
Generating output to TextRank-master/keywords/9.txt
```

Detailed TRACE

In [13]:
```
text="""LINCOLNSHIRE, IL With next-generation video game systems such as the Xbox One and the Playstation 4 hitting
stores later this month, the console wars got even hotter today as electronics manufacturer Zenith announced
the release of its own console, the Gamespace Pro, which arrives in stores Nov. 19. “With its sleek silver-and-gray box,
double-analog-stick controllers, ability to play CDs, and starting price of $374.99, the Gamespace Pro is our way of
saying, ‘Move over, Sony and Microsoft, Zenith is now officially a player in the console game,’” said
Zenith CEO Michael Ahn at a Gamespace Pro press event, showcasing the system’s launch titles MoonChaser:
Radiation, Cris Collinsworth’s Pigskin 2013, and survival-horror thriller InZomnia. “With over nine launch
titles, 3D graphics, and the ability to log on to the internet using our Z-Connect technology, Zenith is finally
poised to make some big waves in the video game world.” According to Zenith representatives, over 650 units have
already been preordered."""
```

In [17]:
```
articleFile = io.open('TextRank-master/articles/' + "1.txt", 'r')
text = articleFile.read()
wordTokens = nltk.word_tokenize(text)

#assign POS tags to the words in the text
tagged = nltk.pos_tag(wordTokens)
textlist = [x[0] for x in tagged]
```

In [18]:
```
tagged
```

Out[18]:
```
[(u'\ufeffLINCOLNSHIRE', 'NN'),
 (u',', ','),
 (u'IL', 'NNP'),
 (u'With', 'IN'),
 (u'next-generation', 'JJ'),
 (u'video', 'NN'),
 (u'game', 'NN'),
 (u'systems', 'NNS'),
 (u'such', 'JJ'),
 (u'as', 'IN'),
 (u'the', 'DT'),
 (u'Xbox', 'NNP'),
 (u'One', 'NNP'),
 (u'and', 'CC'),
 (u'the', 'DT'),
 (u'Playstation', 'NNP'),
 (u'4', 'CD'),
 (u'hitting', 'NN'),
 (u'stores', 'NNS'),
```

In [20]:
```
tagged = filter_for_tags(tagged)
tagged
```

Out[20]:
```
[(u'\ufeffLINCOLNSHIRE', 'NN'),
 (u'IL', 'NNP'),
 (u'next-generation', 'JJ'),
 (u'video', 'NN'),
 (u'game', 'NN'),
 (u'such', 'JJ'),
 (u'Xbox', 'NNP'),
 (u'One', 'NNP'),
 (u'Playstation', 'NNP'),
 (u'hitting', 'NN'),
 (u'month', 'NN'),
 (u'console', 'NN'),
 (u'today', 'NN'),
 (u'manufacturer', 'NN'),
 (u'Zenith', 'NNP'),
 (u'release', 'NN'),
 (u'own', 'JJ'),
 (u'console', 'NN'),
 (u'Gamespace', 'NNP'),
 (u'Pro', 'NNP'),
 (u'Nov.', 'NNP'),
 (u'sleek', 'JJ'),
 (u'silver-and-gray', 'NN'),
 (u'box', 'NN'),
 (u'double-analog-stick', 'JJ'),
 (u'ability', 'NN'),
 (u'price', 'NN'),
 (u'Gamespace', 'NNP'),
 (u'Pro', 'NNP')
```

In [21]:
```
tagged = normalize(tagged)
tagged
```

Out[21]:
```
[(u'\ufeffLINCOLNSHIRE', 'NN'),
 (u'IL', 'NNP'),
 (u'next-generation', 'JJ'),
 (u'video', 'NN'),
 (u'game', 'NN'),
 (u'such', 'JJ'),
 (u'Xbox', 'NNP'),
 (u'One', 'NNP'),
 (u'Playstation', 'NNP'),
 (u'hitting', 'NN'),
 (u'month', 'NN'),
 (u'console', 'NN'),
 (u'today', 'NN'),
 (u'manufacturer', 'NN'),
 (u'Zenith', 'NNP'),
 (u'release', 'NN'),
 (u'own', 'JJ'),
 (u'console', 'NN'),
 (u'Gamespace', 'NNP'),
 (u'Pro', 'NNP'),
 (u'Nov', 'NNP'),
 (u'sleek', 'JJ'),
 (u'silver-and-gray', 'NN'),
 (u'box', 'NN'),
 (u'double-analog-stick', 'JJ'),
 (u'ability', 'NN'),
 (u'price', 'NN'),
 (u'Gamespace', 'NNP'),
 (u'Pro', 'NNP')
```

In [22]:
```
unique_word_set = unique_everseen([x[0] for x in tagged])
word_set_list = list(unique_word_set)

#this will be used to determine adjacent words in order to construct keyphrases with two words
graph = buildGraph(word_set_list)

#pageRank - initial value of 1.0, error tolerance of 0,0001,
calculated_page_rank = nx.pagerank(graph, weight='weight')

#most important words in ascending order of importance
keyphrases = sorted(calculated_page_rank, key=calculated_page_rank.get, reverse=True)
```

In [26]:
```
word_set_list
```

Out[26]:
```
[u'\ufeffLINCOLNSHIRE',
 u'IL',
 u'next-generation',
 u'video',
 u'game',
 u'such',
 u'Xbox',
 u'One',
 u'Playstation',
 u'hitting',
 u'month',
 u'console',
 u'today',
 u'manufacturer',
 u'Zenith',
 u'release',
 u'own',
 u'Gamespace',
 u'Pro',
 u'Nov',
 u'sleek',
```

In [35]:
```
keyphrases
```

Out[35]:
```
[u'double-analog-stick',
 u'survival-horror',
 u'silver-and-gray',
 u'next-generation',
 u'\ufeffLINCOLNSHIRE',
 u'Collinsworth\u2019s',
 u'manufacturer',
 u'Playstation',
 u'technology',
 u'MoonChaser',
 u'Gamespace',
 u'Microsoft',
 u'Z-Connect',
 u'Radiation',
 u'system\u2019s',
 u'InZomnia',
 u'thriller',
 u'internet',
 u'Pigskin',
 u'ability',
 u'hitting',
```

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Mid term; Feedback/Evaluation
  - Homework HW6, HW7, HW9
  - AWS: no access
  - Async lecture recap plus Q&A (PageRank)
  - Contextual advertising
  - Text as graph: TextRank
    - Keyword extraction (from text/target pages)
    - Text Summarization
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

Pagerank expects as input

- **Nodes**
- **Matrix**

- **For text summarization what are the nodes and how do we create the adjacency matrix?**

Pagerank expects as input

- Nodes
- Matrix

- For text summarization what are the nodes and how do we create the adjacency matrix?

- Nodes are sentences
- Links are the blah blah between sentences?

Text Summarization

- To apply TextRank, we first need to build a graph associated with the text, where the graph vertices are representative for the units to be ranked. For the task of sentence extraction, the goal is to rank entire sentences, and therefore a vertex is added to the graph for each sentence in the text.

Sentence similarity

Formally, given two sentences S_i and S_j, with a sentence being represented by the set of N_i words that appear in the sentence: $S_i = w_1^i, w_2^i, ..., w_{N_i}^i$, the similarity of S_i and S_j is defined as:

$$Similarity(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \& w_k \in S_j\}|}{log(|S_i|) + log(|S_j|)}$$

Other sentence similarity measures, such as string kernels, cosine similarity, longest common subsequence, etc. are also possible, and we are currently evaluating their impact on the summarization performance.

Standard PageRank

- Formally, let $G = (V, E)$ be a directed graph with the set of vertices V and set of edges E, where E is a subset of $V \times V$. For a given vertex V_i, let $In(V_i)$ be the set of vertices that point to it (predecessors), and let $Out(V_i)$ be the set of vertices that vertex V_i points to (successors). The score of a vertex V_i is defined as follows (Brin and Page, 1998):

$$S(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

where d is a damping factor that can be set between 0 and 1, which has the role of integrating into the model the probability of jumping from a given vertex to another random vertex in the graph. In the context

Weighted-PageRank

- **PageRank** $S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$

Consequently, we introduce a new formula for graph-based ranking that takes into account edge weights when computing the score associated with a vertex in the graph. Notice that a similar formula can be defined to integrate vertex weights.

$$WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

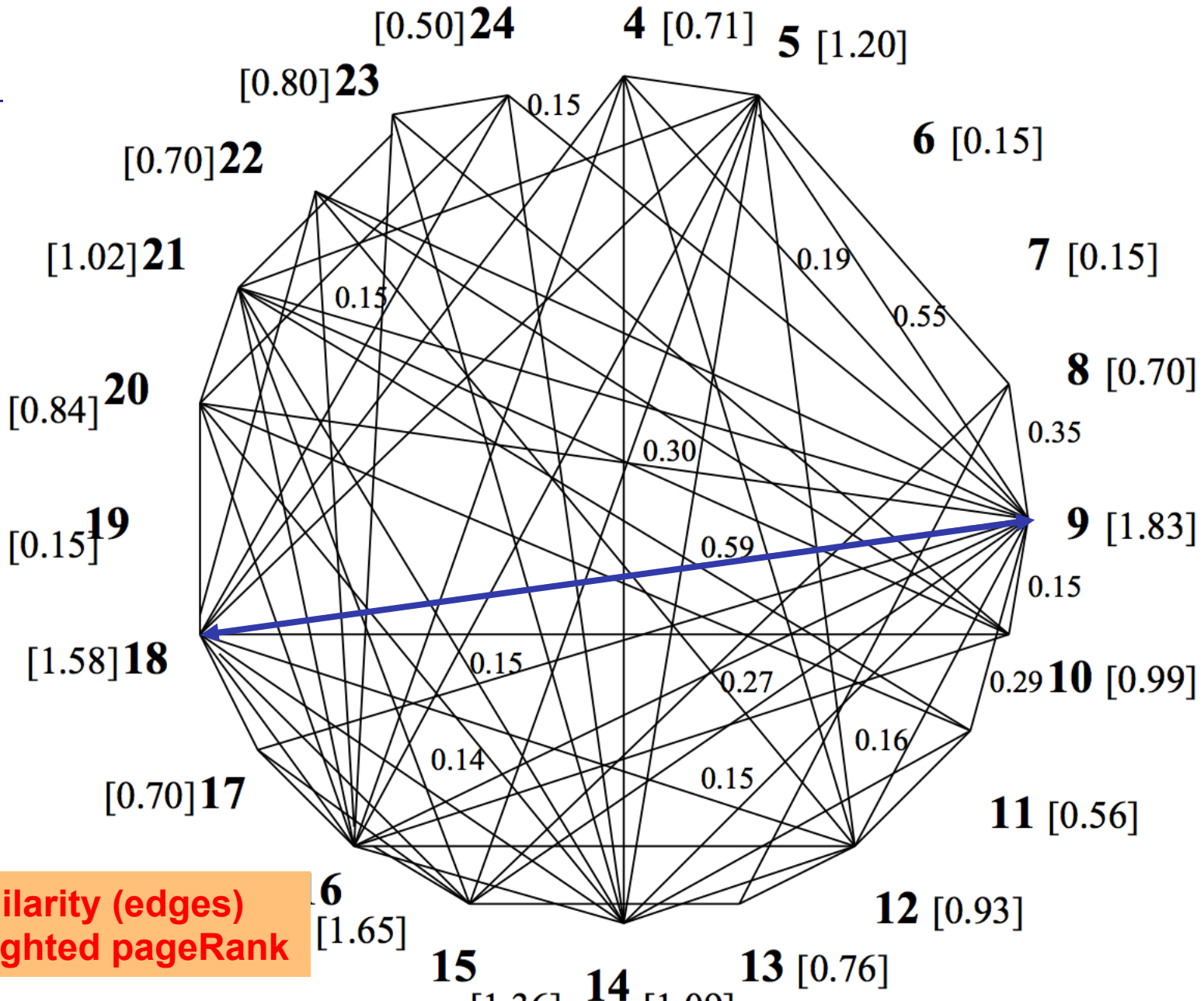
Similarity: Connection between two sentences

- Such a relation between two sentences can be seen as a process of “recommendation”: a sentence that addresses certain concepts in a text, gives the reader a “recommendation” to refer to other sentences in the text that address the same concepts, and therefore a link can be drawn between any two such sentences that share common content.

Text Summarization Example

- **Sample graph build for sentence extraction from a newspaper article. Manually assigned summaries and TextRank extractive summary are also shown.**

3: BC−HurricaineGilbert, 09−11 339
4: BC−Hurricaine Gilbert, 0348
5: Hurricaine Gilbert heads toward Dominican Coast
6: By Ruddy Gonzalez
7: Associated Press Writer
8: Santo Domingo, Dominican Republic (AP)
9: Hurricaine Gilbert Swept towrd the Dominican Republic Sunday, and the Civil Defense alerted its heavily populated south coast to prepare for high winds, heavy rains, and high seas.
10: The storm was approaching from the southeast with sustained winds of 75 mph gusting to 92 mph.
11: "There is no need for alarm," Civil Defense Director Eugenio Cabral said in a television alert shortly after midnight Saturday.
12: Cabral said residents of the province of Barahona should closely follow Gilbert's movement.
13: An estimated 100,000 people live in the province, including 70,000 in the city of Barahona, about 125 miles west of Santo Domingo.
14. Tropical storm Gilbert formed in the eastern Carribean and strenghtened into a hurricaine Saturday night.
15: The National Hurricaine Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo.
16: The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westard at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the storm.
17. The weather service issued a flash flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.
18: Strong winds associated with the Gilbert brought coastal flooding, strong southeast winds, and up to 12 feet to Puerto Rico's south coast.
19: There were no reports on casualties.
20: San Juan, on the north coast, had heavy rains and gusts Saturday, but they subsided during the night.
21: On Saturday, Hurricane Florence was downgraded to a tropical storm, and its remnants pushed inland from the U.S. Gulf Coast.
22: Residents returned home, happy to find little damage from 90 mph winds and sheets of rain.
23: Florence, the sixth named storm of the 1988 Atlantic storm season, was the second hurricane.
24: The first, Debby, reached minimal hurricane strength briefly before hitting the Mexican coast last month.



Similarity (edges)
Weighted pageRank

Summarize: Take top ranked sentences

- **After the ranking algorithm is run on the graph, sentences are sorted in reversed order of their score, and the top ranked sentences are selected for inclusion in the summary.**

TextRank extractive summary

Hurricane Gilbert swept toward the Dominican Republic Sunday, and the Civil De– fense alerted its heavily populated south coast to prepare for high winds, heavy rains and high seas. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday at latitude 16.1 north, longitude 67.5 west, about 140 miles south of Ponce, Puerto Rico, and 200 miles southeast of Santo Domingo. The National Weather Service in San Juan, Puerto Rico, said Gilbert was moving westward at 15 mph with a "broad area of cloudiness and heavy weather" rotating around the center of the storm. Strong winds associated with Gilbert brought coastal flooding, strong southeast winds and up to 12 feet to Puerto Rico's south coast.

Manual abstract I

Hurricane Gilbert is moving toward the Dominican Republic, where the residents of the south coast, especially the Barahona Province, have been alerted to prepare for heavy rains, and high wind and seas. Tropical storm Gilbert formed in the eastern Carribean and became a hurricane on Saturday night. By 2 a.m. Sunday it was about 200 miles southeast of Santo Domingo and moving westward at 15 mph with winds of 75 mph. Flooding is expected in Puerto Rico and in the Virgin Islands. The second hurricane of the season, Florence, is now over the southern United States and down– graded to a tropical storm.

Manual abstract II

Tropical storm Gilbert in the eastern Carribean strenghtened into a hurricane Saturday night. The National Hurricane Center in Miami reported its position at 2 a.m. Sunday to be about 140 miles south of Puerto Rico and 200 miles southeast of Santo Domingo. It is moving westward at 15 mph with a broad area of cloudiness and heavy weather with sustained winds of 75 mph gusting to 92 mph. The Dominican Republic's Civil Defense alerted that country's heavily populated south coast and the National Weather Service in San Juan, Puerto Rico issued a flood watch for Puerto Rico and the Virgin Islands until at least 6 p.m. Sunday.

Evalaution

- a single-document summarization task, using 567 news articles provided during the Document Understanding Evaluations 2002 (DUC, 2002). For each article, TextRank generates an 100-words summary — the task undertaken by other systems participating in this single document summarization task.

System	ROUGE score – Ngram(1,1)		
	basic (a)	stemmed (b)	stemmed no-stopwords (c)
S27	0.4814	0.5011	0.4405
S31	0.4715	0.4914	0.4160
TextRank	**0.4708**	**0.4904**	**0.4229**
S28	0.4703	0.4890	0.4346
S21	0.4683	0.4869	0.4222
Baseline	*0.4599*	*0.4779*	*0.4162*
S29	0.4502	0.4681	0.4019

Table 2: Results for single document summarization: TextRank, top 5 (out of 15) DUC 2002 systems, and baseline. Evaluation takes into account (a) all words; (b) stemmed words; (c) stemmed words, and no stop-words.

TextRank Code and data

- https://www.dropbox.com/sh/3s70t7wgxte30kp/AABoMJrZY83pvx-F8VL_yiDra?dl=0

Summarization via TextRank

In [31]:
```
sent_detector = nltk.data.load('tokenizers/punkt/english.pickle')
sentenceTokens = sent_detector.tokenize(text.strip())
graph = buildGraph(sentenceTokens)

calculated_page_rank = nx.pagerank(graph, weight='weight')
```

In [32]:
```
sentenceTokens
```

Out[32]:
```
[u'\ufeffLINCOLNSHIRE, IL With next-generation video game systems such as the Xbox One and the Playstation 4 hitting
stores later this month, the console wars got even hotter today as electronics manufacturer Zenith announced the rele
ase of its own console, the Gamespace Pro, which arrives in stores Nov. 19.',
 u'\u201cWith its sleek silver-and-gray box, double-analog-stick controllers, ability to play CDs, and starting price
of $374.99, the Gamespace Pro is our way of saying, \u2018Move over, Sony and Microsoft, Zenith is now officially a p
layer in the console game,\u2019\u201d said Zenith CEO Michael Ahn at a Gamespace Pro press event, showcasing the sys
tem\u2019s launch titles MoonChaser: Radiation, Cris Collinsworth\u2019s Pigskin 2013, and survival-horror thriller I
nZomnia.',
 u'\u201cWith over nine launch titles, 3D graphics, and the ability to log on to the internet using our Z-Connect tec
hnology, Zenith is finally poised to make some big waves in the video game world.\u201d According to Zenith represent
atives, over 650 units have already been preordered.']
```

In [33]:
```
text
```

Out[33]:
```
u'\ufeffLINCOLNSHIRE, IL With next-generation video game systems such as the Xbox One and the Playstation 4 hitting s
tores later this month, the console wars got even hotter today as electronics manufacturer Zenith announced the relea
se of its own console, the Gamespace Pro, which arrives in stores Nov. 19. \u201cWith its sleek silver-and-gray box,
double-analog-stick controllers, ability to play CDs, and starting price of $374.99, the Gamespace Pro is our way of
saying, \u2018Move over, Sony and Microsoft, Zenith is now officially a player in the console game,\u2019\u201d said
Zenith CEO Michael Ahn at a Gamespace Pro press event, showcasing the system\u2019s launch titles MoonChaser: Radiati
on, Cris Collinsworth\u2019s Pigskin 2013, and survival-horror thriller InZomnia. \u201cWith over nine launch titles,
3D graphics, and the ability to log on to the internet using our Z-Connect technology, Zenith is finally poised to ma
ke some big waves in the video game world.\u201d According to Zenith representatives, over 650 units have already bee
n preordered.'
```

In [34]:
```
calculated_page_rank
```

Out[34]:
```
{u'\u201cWith its sleek silver-and-gray box, double-analog-stick controllers, ability to play CDs, and starting price
of $374.99, the Gamespace Pro is our way of saying, \u2018Move over, Sony and Microsoft, Zenith is now officially a p
layer in the console game,\u2019\u201d said Zenith CEO Michael Ahn at a Gamespace Pro press event, showcasing the sys
tem\u2019s launch titles MoonChaser: Radiation, Cris Collinsworth\u2019s Pigskin 2013, and survival-horror thriller I
nZomnia.': 0.363650754848674,
 u'\u201cWith over nine launch titles, 3D graphics, and the ability to log on to the internet using our Z-Connect tec
hnology, Zenith is finally poised to make some big waves in the video game world.\u201d According to Zenith represent
atives, over 650 units have already been preordered.': 0.31264271392367987,
 u'\ufeffLINCOLNSHIRE, IL With next-generation video game systems such as the Xbox One and the Playstation 4 hitting
stores later this month, the console wars got even hotter today as electronics manufacturer Zenith announced the rele
ase of its own console, the Gamespace Pro, which arrives in stores Nov. 19.': 0.3237065312276459}
```

Rouge Metrics

- ROUGE, or Recall-Oriented Understudy for Gisting Evaluation,[1] is a set of metrics and a software package used for evaluating automatic summarization and machine translation software in natural language processing.
- The metrics compare an automatically produced summary or translation against a reference or a set of references (human-produced) summary or translation.
- ROUGE is available at
  - http://www.berouge.com/Pages/default.aspx

https://en.wikipedia.org/wiki/ROUGE_(metric)

https://www.dropbox.com/s/0bmtv8hzf87xbtf/Rouge-Bleu-Evaluation-Metrics.pdf?dl=0

Rouge Metrics Family

- ,..

The following five evaluation metrics[2] are available.

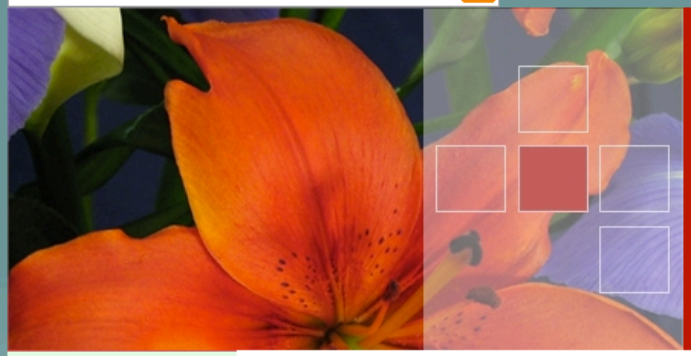
- ROUGE-N: N-gram[3] based co-occurrence statistics.
- ROUGE-L: Longest Common Subsequence (LCS)[4] based statistics. Longest common subsequence problem takes into account sentence level structure similarity naturally and identifies longest co-occurring in sequence n-grams automatically.
- ROUGE-W: Weighted LCS-based statistics that favors consecutive LCSes .
- ROUGE-S: Skip-bigram[5] based co-occurrence statistics. Skip-bigram is any pair of words in their sentence order.
- ROUGE-SU: Skip-bigram plus unigram-based co-occurrence statistics.

ROUGE can be downloaded from berouge download link.

www.berouge.com/Pages/default.aspx

oogle Docs | Word2Vec: an introd | nbviewer.ipython.org | Bookmarks | (2) MIDS-MLS-2015 | Stanford Machine Le | Getting Started | Statistical Analysis H | eBay/Google 2013: E | Inquiri

Type to search the Web



ROUGE: Recall-Oriented Understudy of Gisting Evaluation

A software package for automated evaluation of summaries

- Home
- Download ROUGE
- Member Login

What is ROUGE?

ROUGE is a software package for automated evaluation of summaries. It was developed by Chin-Yew Lin while he was at the Information Sciences Institute of University of Southern California (USC/ISI).

Automated text summarization has drawn a lot of interest in the natural language processing and information retrieval communities in the recent years. A series of workshops on automatic text summarization (WAS 2000, 2001, 2002), special topic sessions in ACL, COLING, and SIGIR, and government sponsored evaluation efforts in the United States (DUC 2002) and Japan (Fukusima and Okumura 2001) have advanced the technology and produced a couple of experimental online systems (Radev et al. 2001, McKeown et al. 2002). Despite these efforts, however, there are no common, convenient, and repeatable evaluation methods that can be easily applied to support system development and just-in-time comparison among different summarization methods.

Following the recent adoption by the machine translation community of automatic evaluation using the BLEU/NIST scoring process, we conduct an in-depth study of a similar idea for evaluating summaries. The results show that automatic evaluation using unigram co-occurrences, i.e. ROUGE, between summary pairs correlates surprising well with human evaluations, based on various statistical metrics; while direct application of the BLEU evaluation procedure does not always give good results. For the inception of ROUGE, please read Lin & Hovy's HLT-NAACL 2003 (Lin and Hovy 2003) paper. For more details, please read Lin's paper "ROUGE: a Package for Automatic Evaluation of Summaries" (Lin 2004a). For the effect of sample size, please see "Looking for a Few Good Metrics: Automatic Summarization Evaluation - How Many Samples Are Enough?" (Lin 2004b). For the application of ROUGE in automatic machine translation evaluation, please see "Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics" (Lin & Och 2004a) and "ORANGE: a Method for Evaluating Automatic Evaluation Metrics for Machine Translation" (Lin & Och 2004b).

For a brief review of ROUGE, please see my presentation at the Workshop on Machine Translation Evaluation - Towards Systematizing MT Evaluation, entitled "Cross-domain Study of N-gram Co-Occurrence Metrics - A Case in Summarization".

ROUGE has been used in DUC 2004 and will be used in DUC 2005 and the multilingual summarization evaluation to be held with the Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization.

If you would like to use ROUGE in your experiments, you can download the most recent version here. If you have any suggestions and comments please contact me at: rouge AT berouge.com.

Anatomy of BLEU Matching Score

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

Precision-based Metric!

Weighted geometric average favors longer N-gram matches

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$

Counts of N-gram overlaps between a candidate and reference translations

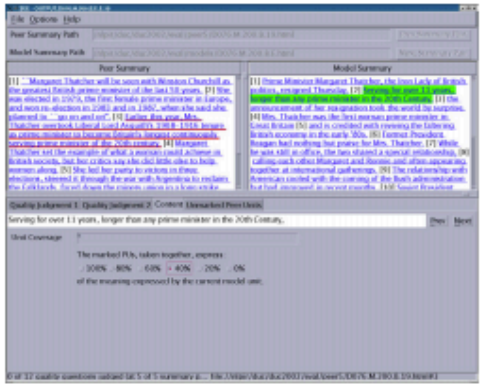
$$p_n = \frac{\sum_{\mathcal{C} \in \{Candidates\}} \sum_{n\text{-}gram \in \mathcal{C}} Count_{clip}(n\text{-}gram)}{\sum_{\mathcal{C} \in \{Candidates\}} \sum_{n\text{-}gram \in \mathcal{C}} Count(n\text{-}gram)}$$

Total number of n-gram in the candidate translation

Papineni et al. 2001, iBleu: a Method for Automatic Evaluation of Machine Translation i IBM Research Report RC22176(W0109-022)

https://www.dropbox.com/s/0bmtv8hzf87xbtf/Rouge-Bleu-Evaluation-Metrics.pdf?dl=0

ROUGE: Recall-Oriented Understudy for Gisting Evaluation



ROUGEs — N-gram co-occurrence metrics measuring content overlaps

Counts of N-gram overlaps between candidate and model summaries

$$ROUGE_n = \frac{\sum_{C \in \{Model\ Units\}} \sum_{n-gram \in C} Count_{match}(n-gram)}{\sum_{C \in \{Model\ Units\}} \sum_{n-gram \in C} Count(n-gram)}$$

Total number of n-grams in the model summary

Recall-based Metric!
(fixed-length summaries)

https://www.dropbox.com/s/0bmtv8hzf87xbtf/Rouge-Bleu-Evaluation-Metrics.pdf?dl=0

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Mid term; Feedback/Evaluation
  - Homework HW6, HW7, HW9
  - AWS: no access
  - Async lecture recap plus Q&A (PageRank)
  - Contextual advertising
  - Text as a graph: TextRank
    - Keyword extraction (from text/target pages)
    - Text Summarization
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

- **End of lecture**