Web Crawling

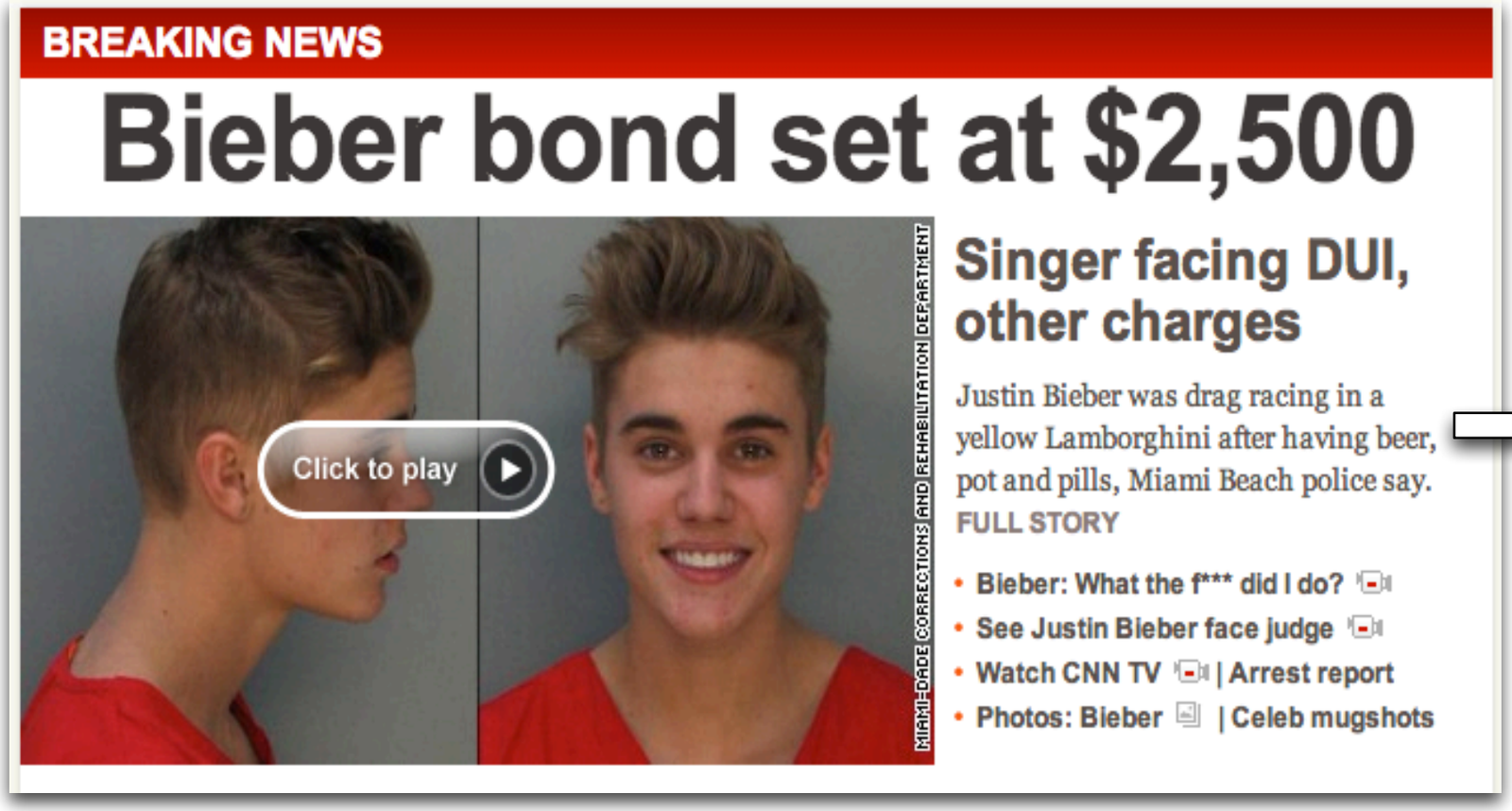
Introduction to Information Retrieval
INF 141/ CS 121
Donald J. Patterson

Content adapted from Hinrich Schütze
http://www.informationretrieval.org

The index

- Why does the crawling architecture exists?
  - To gather information from web pages (aka documents).
- What information are we collecting?
  - Keywords
    - Mapping documents to a “bags of words” (aka vector space model)
  - Links
    - Where does a document link to?
    - Who links to a document?

The index has a list of vector space models

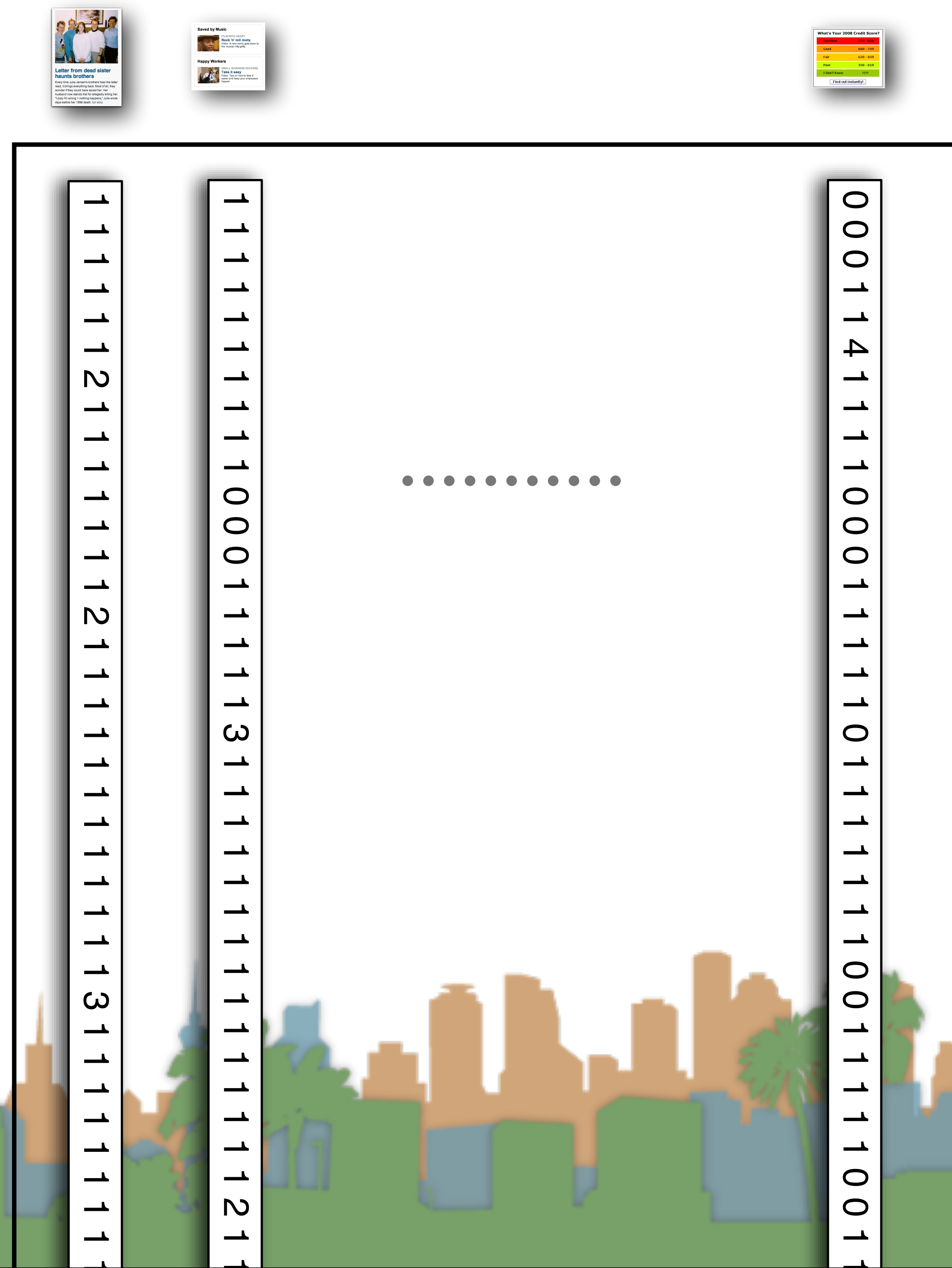


1 2500	2 justin
1 I	1 lamborghini
1 a	1 miami
1 after	1 mugshots
1 and	1 news
1 arrest	1 other
1 at	2 photos
1 beach	1 pills
1 beer	1 police
6 bieber	1 pot
1 bond	1 racing
1 breaking	1 report
1 celeb	1 say.
1 charges	1 see
1 cnn	1 set
1 did	1 singer
1 do	1 story
1 drag	1 the
1 dui	1 tv
1 f	1 was
1 face	1 watch
1 facing	1 what
1 full	1 yellow
1 having	
1 in	
1 judge	

1 1 1 1 1 1 1 1 1 1 6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Our index is a 2-D array or Matrix

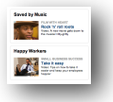
A Column for Each Web Page (or "Document")

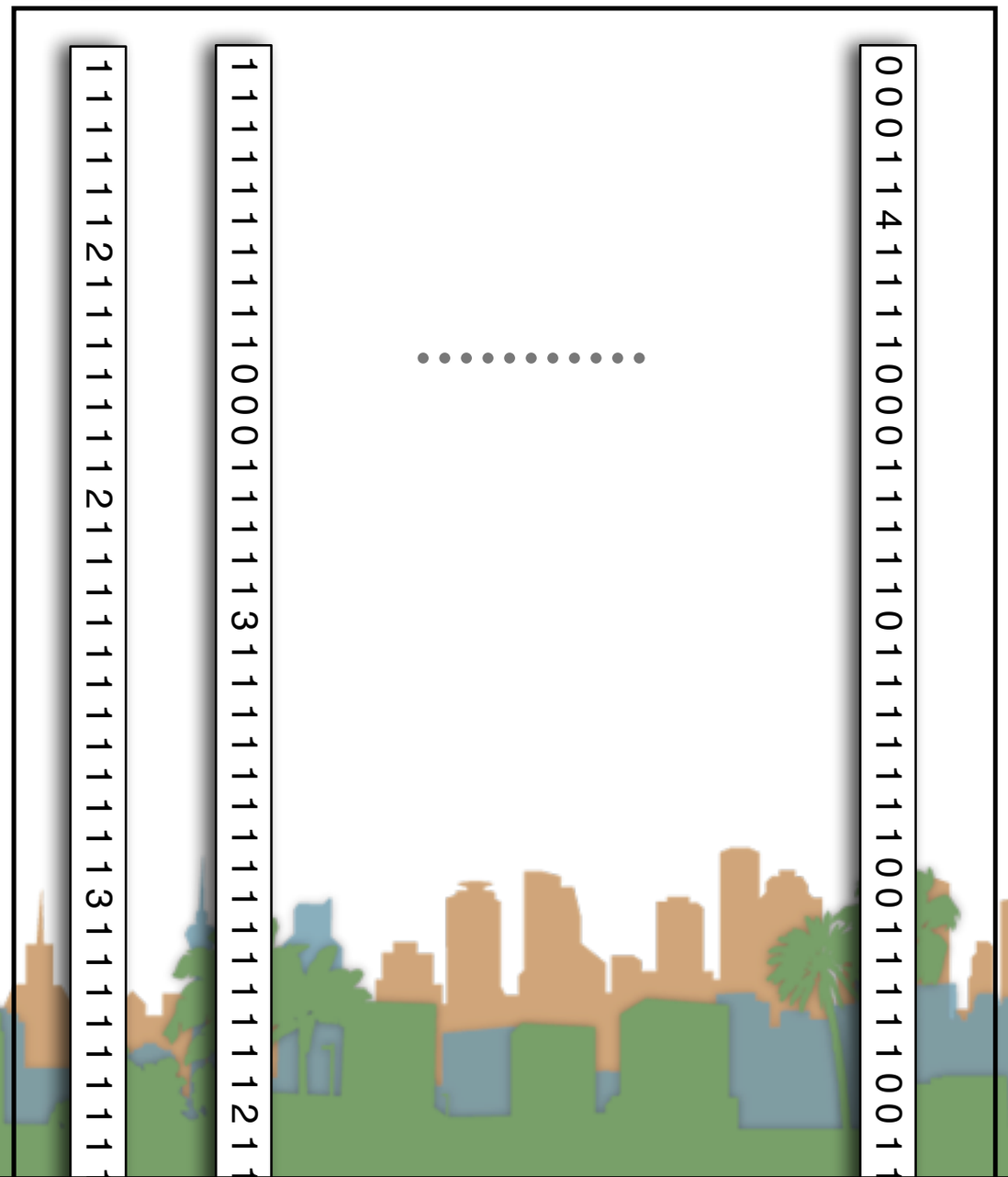


A Row For Each Word (or "Term")

"Term-Document Matrix" Capture Keywords

A Column for Each Web Page (or "Document")





The Term-Document Matrix

- Is really big at a web scale
- It must be split up into pieces
- An effect way to split it up is to split up the same way as the crawling
  - Equivalent to taking vertical slices of the T-D Matrix
  - Helps with cache hits during crawl
- Later we will see that it needs to be rejoined for calculations across all documents

Connectivity Server

- Other part of reason for crawling
- Supports fast queries on the web graph
  - Which URLS point to a given URL (in-links)?
  - Which URLS does a given URL point to (out-links)?
- Applications
  - Crawl control
  - Web Graph Analysis
  - Link Analysis (aka PageRank)
    - Provides input to "quality" for URL frontier

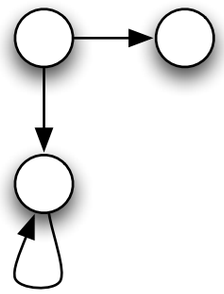
Indices - Connectivity Server

Adjacency Matrix - Conceptual Idea

A

B

C



	A	B	C
A	0	1	1
B	0	0	0
C	0	0	1

Connectivity Server in practice

- What about Adjacency Lists instead?
  - Set of neighbors of a node
  - Assume each URL represented by an integer
  - i.e. 4 billion web pages need 32 bits per URL
  - Naive implementation requires 64 bits per link
    - 32 bits to 32 bits

Connectivity Server in practice

- What about Adjacency Lists instead?
  - Non-naive approach is to exploit compression
    - Similarity between lists of links
    - Locality (many links go to “nearby” links)
    - Use gap encodings in sorted lists
    - Leverage the distribution of gap values

Connectivity Server in practice

- Current state of the art is Boldi and Vigna
  - http://www2004.org/proceedings/docs/1p595.pdf
  - They are able to reduce a URL to URL edge
    - From 64 bits to an average of 3 bits
    - For a 118 million node web graph
- How?

Connectivity Server in practice

- Consider a lexicographically ordered list of all URLS, e.g:
  - http://www.ics.uci.edu/computerscience/index.php
  - http://www.ics.uci.edu/dept/index.php
  - http://www.ics.uci.edu/index.php
  - http://www.ics.uci.edu/informatics/index.php
  - http://www.ics.uci.edu/statistics/index.php

Connectivity Server in practice

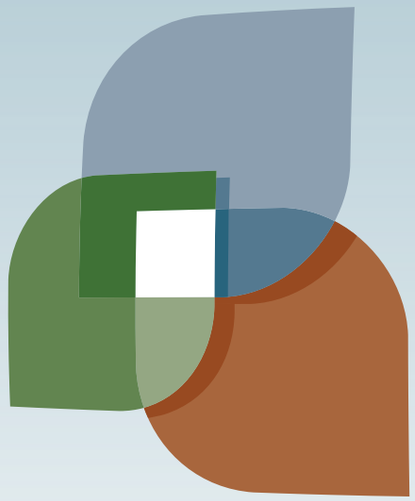
- Each of these URLs has an adjacency list
- Main idea: because of templates, the adjacency list of a node is similar to one of the 7 preceding URLs in the lexicographic ordering.
- So, express adjacency list in terms of a template

Connectivity Server in practice

- Consider these adjacency lists
  - 1, 2, 4, 8, 16, 32, 64
  - 1, 4, 9, 16, 25, 36, 49, 64
  - 1, 2, 3, 5, 6, 13, 21, 34, 55, 89, 144
  - 1, 4, 8, 16, 25, 36, 49, 64
    - Encode this as row(-2), -URL(9), +URL(8)
- Very similar to tricks done in assembly code

Connectivity Server in practice summary

- The web is enormous
- A naive adjacency matrix would be several billion URLS on a side
- Overall goal is to keep the adjacency matrix in memory
- Webgraph is a set of algorithms and a java implementation for examining the web graph
  - It exploits the power law distribution to compress the adjacency matrix very tightly
  - http://webgraph.dsi.unimi.it/



LUCI