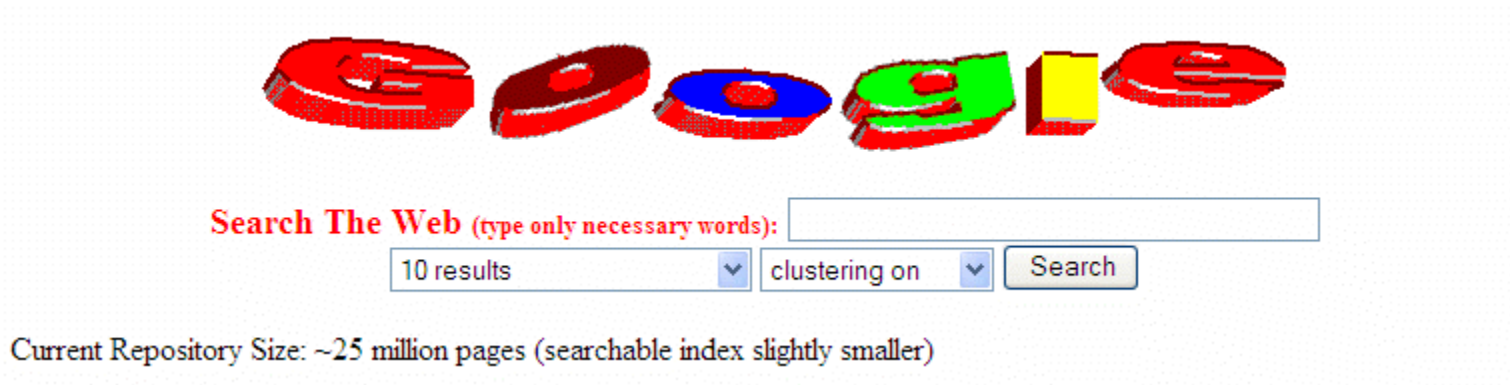
The Science of Search Engines

John MacCormick
Microsoft Research Silicon Valley

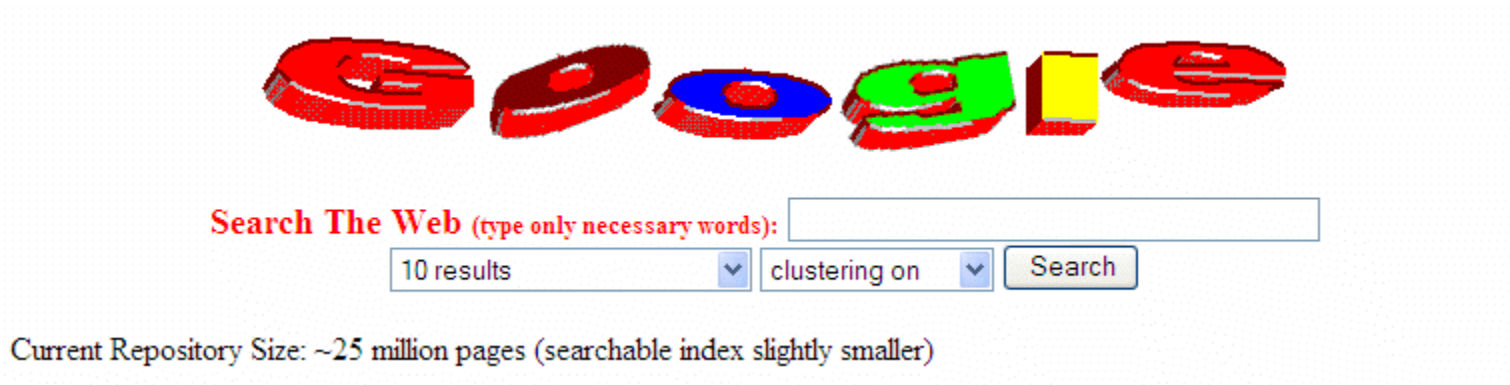
Search engines have profoundly changed the way ordinary people use computers

- huge amount of information available
  - “most” of the world’s “useful” information is out there on the Web??
- search engines are incredibly easy to use
  - no fancy query language needed

Google in 1998

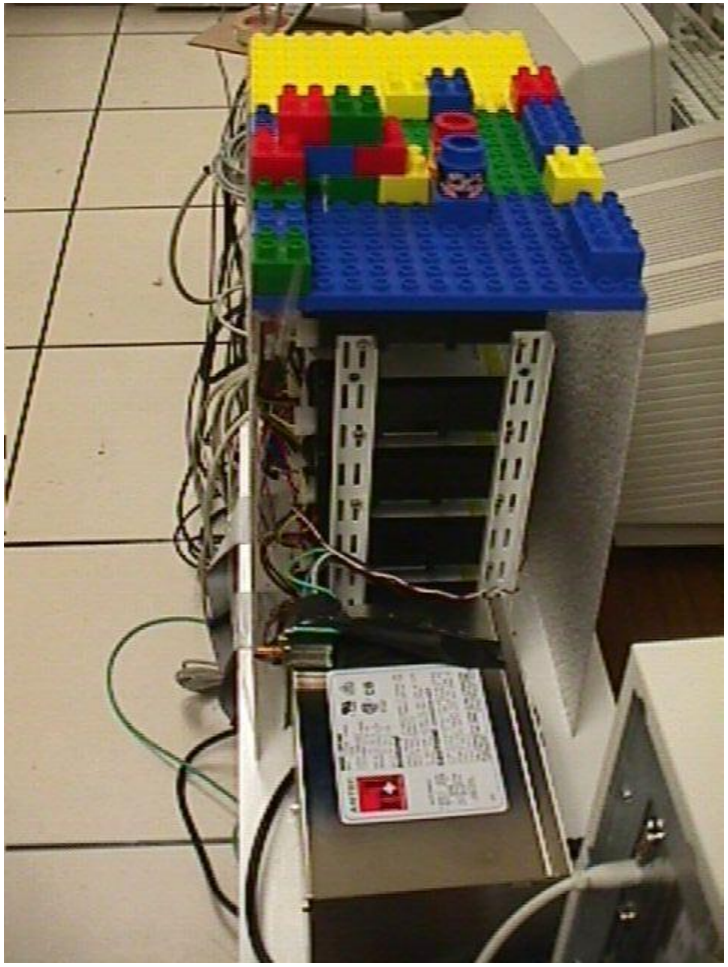


Google in 1998



Google in 1998

Google’s storage system in 1998



(ten 9-gigabyte hard drives)

how do search engines do it?

1. web crawling
2. indexing
3. searching
    a) retrieval
    b) ranking

how do search engines do it?

1. web crawling
2. indexing
3. searching
    a) retrieval
    b) ranking

what is the ranking problem?

- “scrambled eggs” gets nearly one million hits
- user only has patience to look at about 10 results
- need to *rank* the one million hits, and present the top 10 on the first page of results

which page is probably more “useful” or “authoritative”?

links **to** a page
confer authority
on that page

which page is probably more “useful” or “authoritative”?

links from a more
authoritative
page confer
greater authority

PageRank computes the authority of a page rigorously, using matrix algebra

- create “hyperlink matrix”

- transform slightly (details omitted)
- compute principal eigenvector
  - nth coordinate of the eigenvector is the PageRank of the nth page

how do search engines do it?

1. web crawling
2. indexing
3. searching
    a) retrieval
    b) ranking

standard indexing uses document IDs

1 the cat sat on the mat

2 the dog stood on the mat

3 the cat stood while a dog sat

a	3		
cat	1	3	
dog	2	3	
mat	1	2	
the	1	2	3
sat	1	3	
stood	2	3	
on	1	2	
while	3		

example queries:

while

dog

cat dog

“cat sat”

standard indexing is not powerful enough

brilliant idea number 1: index word *locations* within documents

1 the cat sat on the mat

2 the dog stood on the mat

3 the cat stood while a dog sat

a	3.5
cat	1.2 3.2
dog	2.2 3.6
mat	1.6 2.6
the	1.1 1.5 2.1 2.5 3.1
sat	1.3 3.7
stood	2.3 3.3
on	1.4 2.4
while	3.4

phrase queries are easy using location-based indexing

...	...
cat	... 5.9 6.1 8.3 ...
sat	... 4.2 6.3 6.9 9.5
...	...

query: “cat sat”

result: no documents match

phrase queries are easy using location-based indexing

...	...
cat	... 5.9 6.8 8.3 ...
sat	... 4.2 6.3 6.9 9.5
...	...

query: “cat sat”

result: document 6 matches

NEAR queries are also easy using location-based indexing

...	...				
cat	...	5.9	6.1	8.3	...
sat	...	4.2	6.5	6.9	9.5
...	...				

query: cat NEAR sat

result: no matches

NEAR queries are also easy using location-based indexing

...	...				
cat	...	5.9	6.9	8.3	...
sat	...	4.2	6.5	6.7	9.5
...	...				

query: cat NEAR sat

result: document 6 matches

knowing NEARness is also important for ranking

- example query: departed movie
- document 1:
  - "...The Departed is an great movie starring Jack Nicholson..."
- document 2:
  - "blog blog blog ... went to see a movie ... blog blog blog ... more blog ... had to fly to New York ... flight was late ... it finally departed at 10 PM"

document 1 should be ranked higher; location-based indexing lets you do that

brilliant idea number 2: use *metawords* to permit queries that reflect the *structure* of documents

1 **My Cat**

the cat sat on the mat

2 **My Dog**

the dog stood on the mat

3 **My Pets**

the cat stood while a dog sat

brilliant idea number 2: use *metawords* to permit queries that reflect the *structure* of documents

1 <title>My Cat</title> <body>the cat sat on the mat</body>

2 <title>My Dog</title> <body>the dog stood on the mat</body>

3 <title>My Pets</title> <body>the cat stood while a dog sat</body>

brilliant idea number 2: use *metawords* to permit queries that reflect the *structure* of documents

1 <title>My Cat</title> <body>the cat sat on the mat</body>

2 <title>My Dog</title> <body>the dog stood on the mat</body>

3 <title>My Pets</title> <body>the cat stood while a dog sat</body>

cat	1.3	1.7	3.7
sat	1.8	3.12	
...	...		
<title>	1.1	2.1	3.1
</title>	1.4	2.4	3.4
<body>	1.5	2.5	3.5
</body>	1.12	2.12	3.13

queries on document structure are easy

...	...
cat	... 5.9 6.8 7.3 ...
...	...
<title>	... 5.2 6.3 7.1
</title>	... 5.4 6.5 7.4

query: cat IN <title>

result: document 7 matches

queries on document structure are easy

...	...
cat	... 5.7 6.4 8.3 ...
...	...
<title>	... 5.2 6.3 7.1
</title>	... 5.8 6.5 7.4

query: cat IN <title>

result: documents 5 and 6

IN queries also help with ranking

- example query: cat
- document 1: “<title>The Cat Page</title>...”
- document 2: “<title>John’s blog</title><body>blog blog blog...more blog blog...I dressed up as a black cat for Halloween...blog blog blog</body>

document 1 should be ranked higher; location-based indexing lets you do that

location-based indexing can be implemented in an elegant object-oriented framework

- use *index stream reader* (ISR) objects
- ISR methods are:
  - get_loc()
  - get_next_loc()
  - get_loc_limit()
  - get_previous_loc()
- subclasses include:
  - ISR_and
  - ISR_or
  - ISR_not

how do search engines do it?

1. web crawling
2. indexing
3. searching
    a) retrieval
    b) ranking

some more science behind search engines:

- GFS (Google file system)
- MapReduce, Dryad (parallel computation)
- shingling (efficient similarity detection)
- ad pricing (real-time auctions)
- Mercator (web crawling)

thank you very much!

questions?