CC5212-1

PROCESAMIENTO MASIVO DE DATOS
OTOÑO 2018

Lecture 6

Information Retrieval: Crawling & Indexing

Aidan Hogan
aidhog@gmail.com

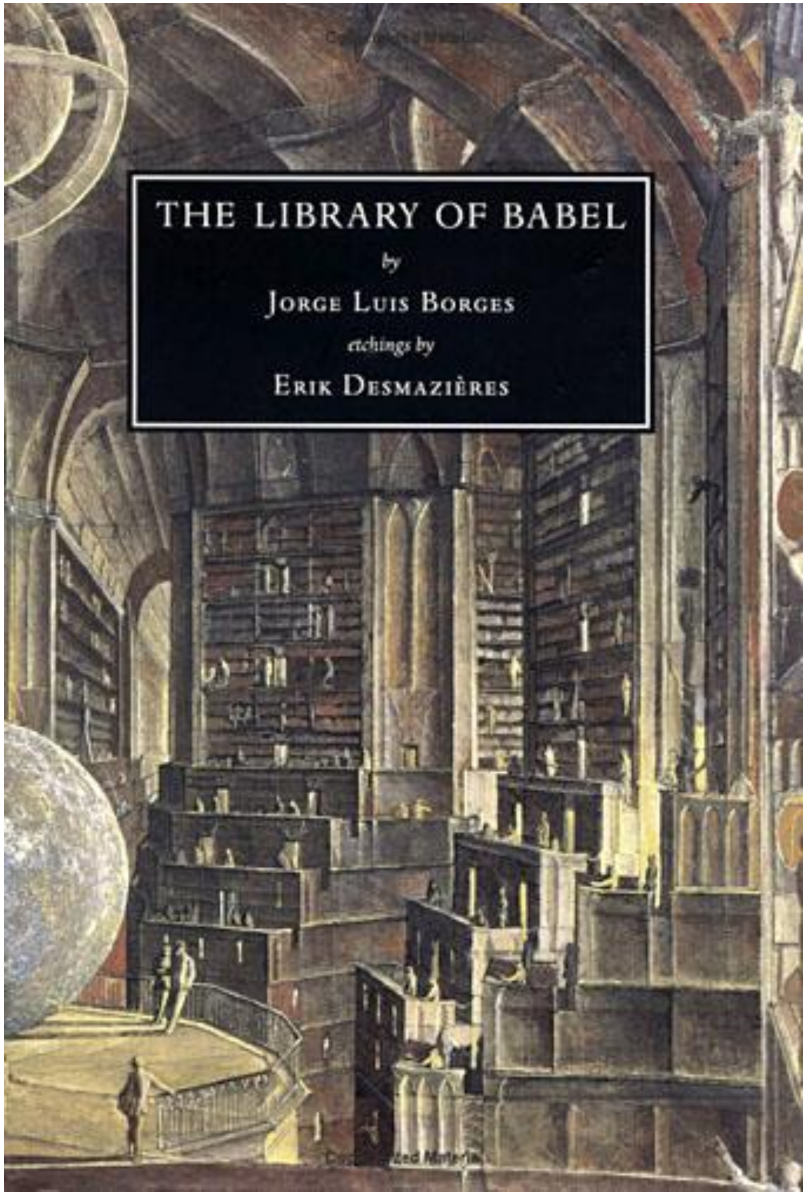
MapReduceBase
HDFS
grunt
Pig
replication
Sort
replicas
Hive
Rack-awareness
MapReduce
Partitioner
JobNode
JobTracker
ChunkServer
GFS
chunks
Hadoop
Reporter
Mapper
Writable
Shuffle
NameNode
Pipelined-reads
Reducer
Combiner
WritableComparable
DataNode
SecondaryNameNode

MANAGING TEXT DATA

Information Overload

Getting information off the Internet is like taking a drink from a fire hydrant.
Mitchell Kapor

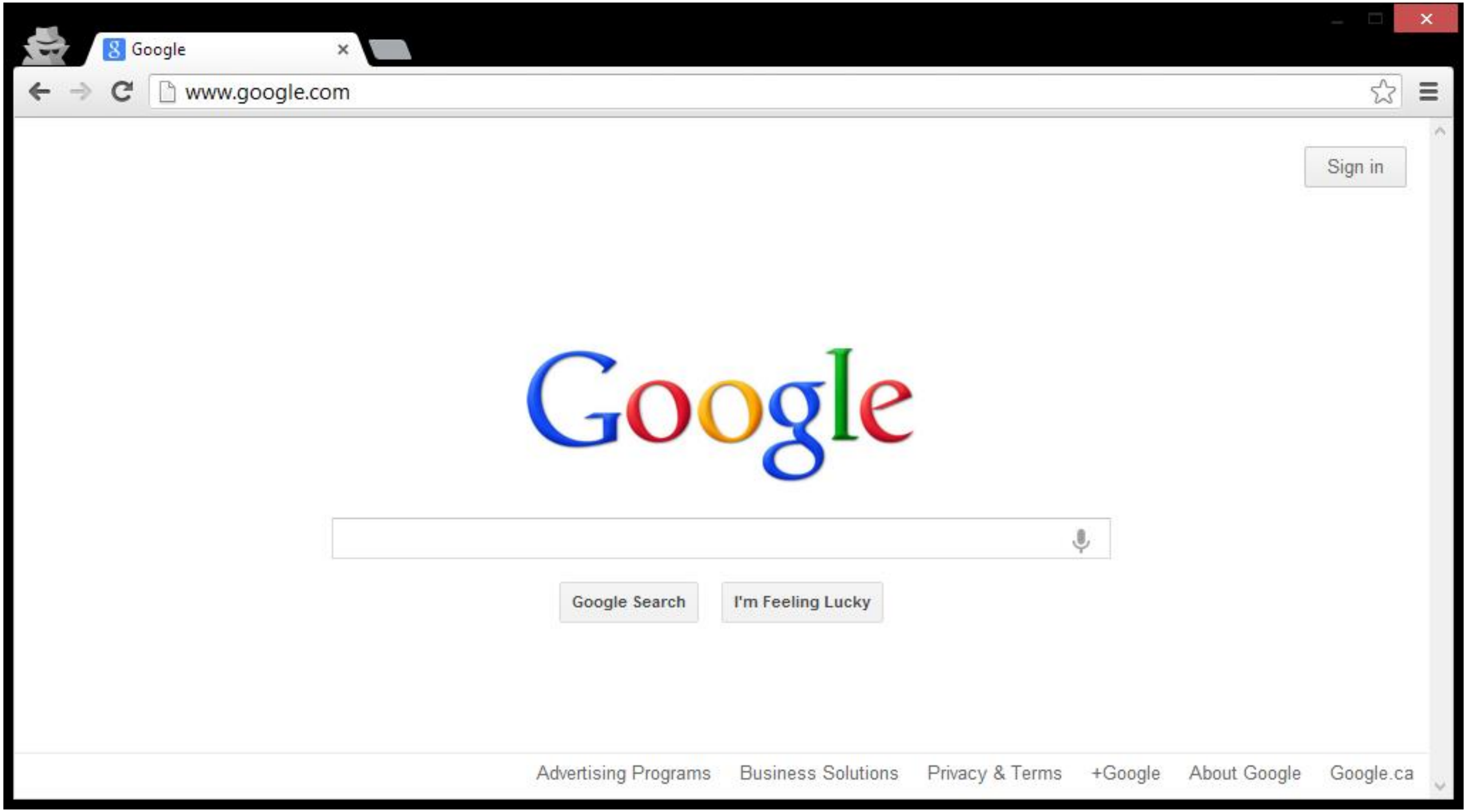
If we didn’t have search …



- Contains all books with
  - 25 unique characters
  - 80 characters per line
  - 40 lines per page
  - 410 pages
  - 410 x 40 x 80 = 1,312,000 chars
  - $25^{1,312,000}$ books
- Would contain any book imaginable
  - Including a book with the location of useful books ;)

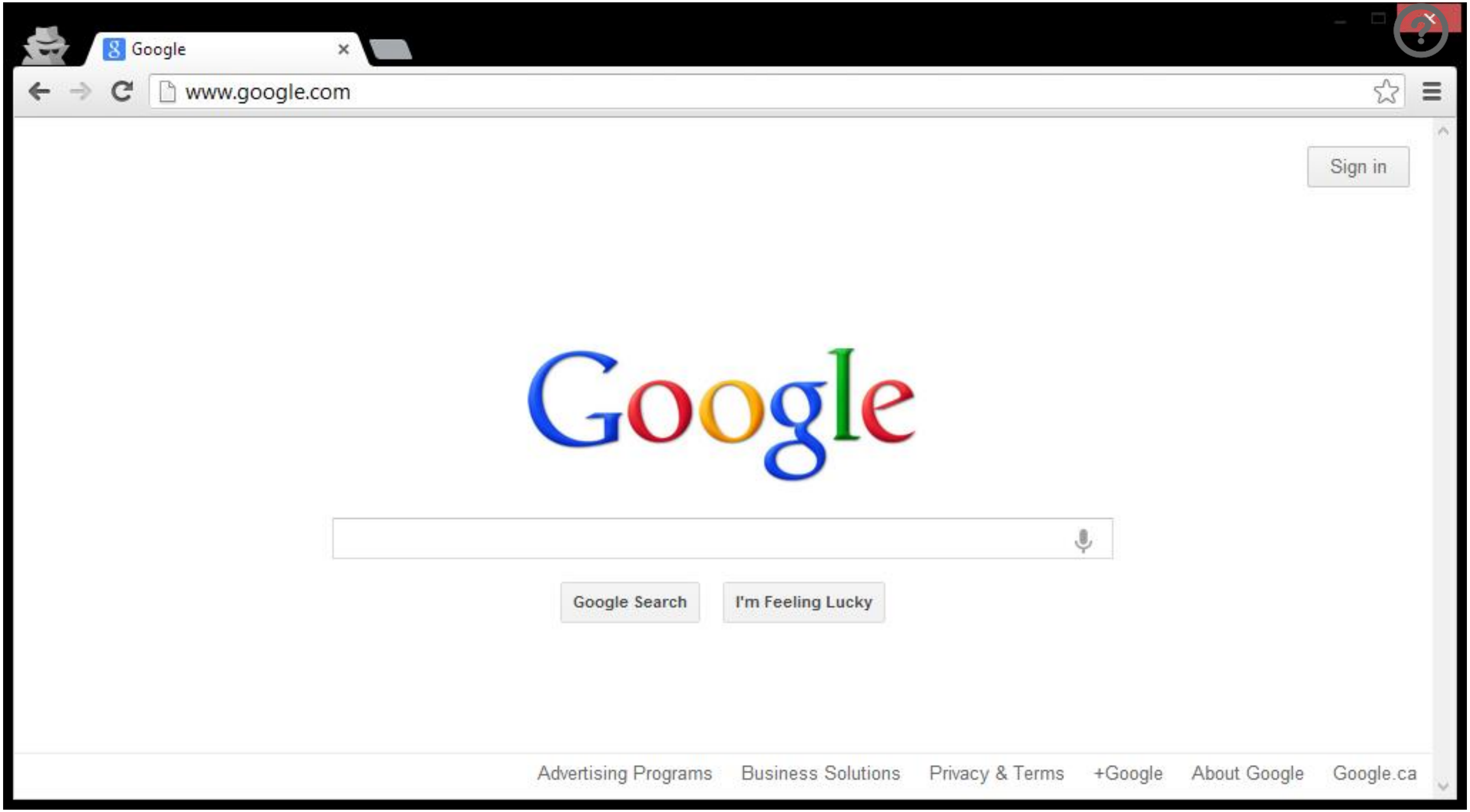
All information = Zero information

The book that indexes the library

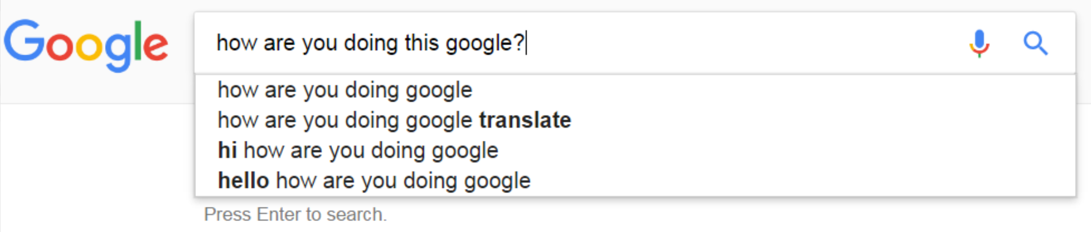


Web Search/Retrieval

Building Google Web-search



Building Google Web-search



how are you doing this google?

how are you doing google
how are you doing google **translate**
hi how are you doing google
hello how are you doing google

Press Enter to search.

What processes/algorithms does Google need to implement Web search?

Crawling

1. Parse links from webpages
2. Schedule links for crawling
3. Download pages, GOTO 1

Indexing

1. Parse keywords from webpages
2. Index keywords to webpages
3. Manage updates

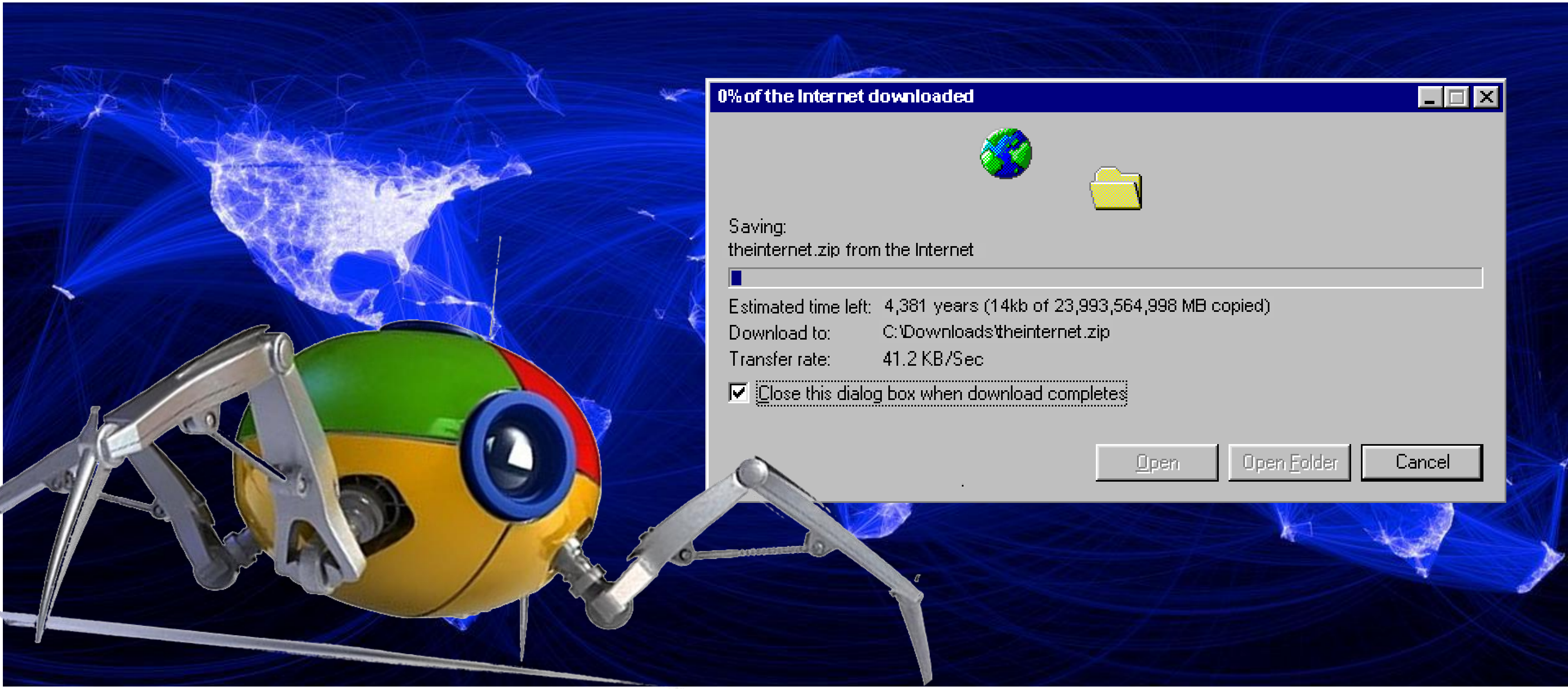
Ranking

1. How relevant is a page? (TF-IDF)
2. How important is it? (PageRank)
3. How many users clicked it?

...

Information Retrieval: Crawling

How does Google know about the Web?



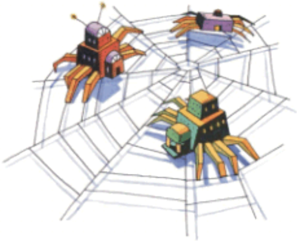
0% of the Internet downloaded

Saving:
theinternet.zip from the Internet

Estimated time left: 4,381 years (14kb of 23,993,564,998 MB copied)
Download to: C:\Downloads\theinternet.zip
Transfer rate: 41.2 KB/Sec

☑ Close this dialog box when download completes

Crawling

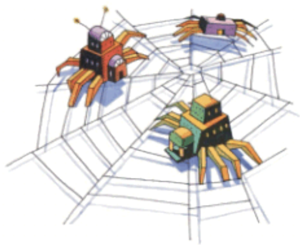


Download the Web. ☺

```
crawl(list seedUrls)
    frontier_i = seedUrls
    while(!frontier_i .isEmpty())
        new list frontier_i+1
        for url : frontier_i
                page = downloadPage(url)
                frontier_i+1.addAll(extractUrls(page))
                store(page)
        i++
```

What’s missing? ?

Crawling: Avoid Cycles

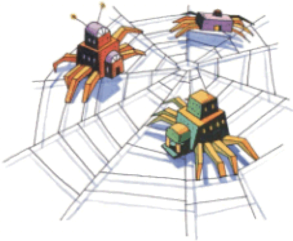


Download the Web. ☺

```
crawl(list seedUrls)
    frontier_i = seedUrls
    new set urlsSeen
    while(!frontier_i .isEmpty())
        new list frontier_i+1
        for url : frontier_i
                page = downloadPage(url)
                urlsSeen.add(url)
                frontier_i+1.addAll(extractUrls(page).removeAll(urlsSeen))
                store(page)
        i++
```

Performance?

Crawling: Avoid Cycles

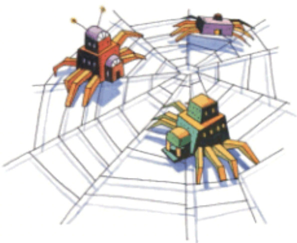


Download the Web. ☺

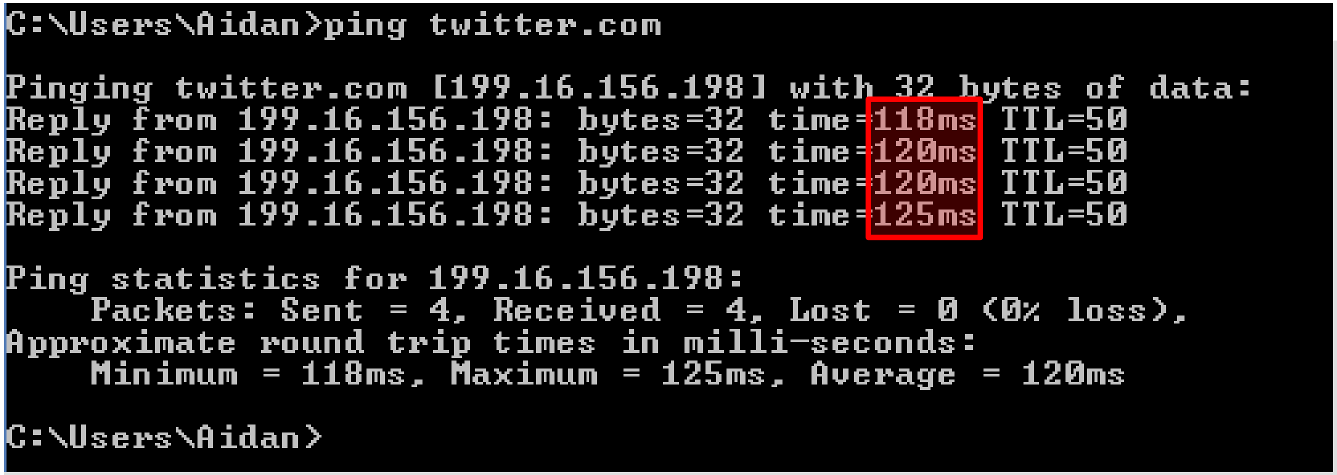
```
crawl(list seedUrls)
    frontier_i = seedUrls
    new set urlsSeen
    while(!frontier_i .isEmpty())
        new list frontier_i+1
        for url : frontier_i
                page = downloadPage(url)
                urlsSeen.add(url)
                frontier_i+1.addAll(extractUrls(page).removeAll(urlsSeen))
                store(page)
        i++
```

Performance? ?

Crawling: Avoid Cycles



Download the Web. ☺



```
C:\Users\Aidan>ping twitter.com

Pinging twitter.com [199.16.156.198] with 32 bytes of data:
Reply from 199.16.156.198: bytes=32 time=118ms TTL=50
Reply from 199.16.156.198: bytes=32 time=120ms TTL=50
Reply from 199.16.156.198: bytes=32 time=120ms TTL=50
Reply from 199.16.156.198: bytes=32 time=125ms TTL=50

Ping statistics for 199.16.156.198:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 118ms, Maximum = 125ms, Average = 120ms

C:\Users\Aidan>
```

page = downloadPage(url)

- Majority of time spent waiting for connection ⚠
- Disk/CPU usage will be near 0
- Bandwidth will not be maximised

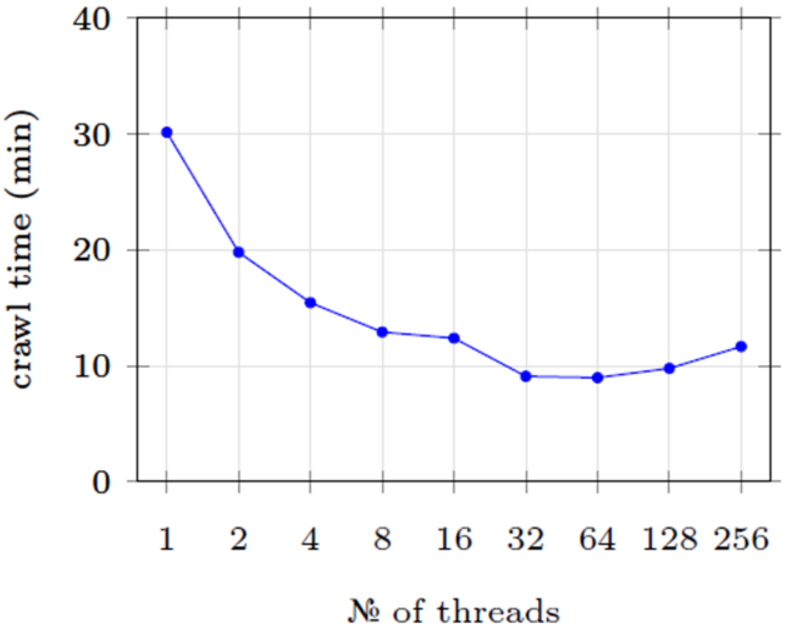
Performance?

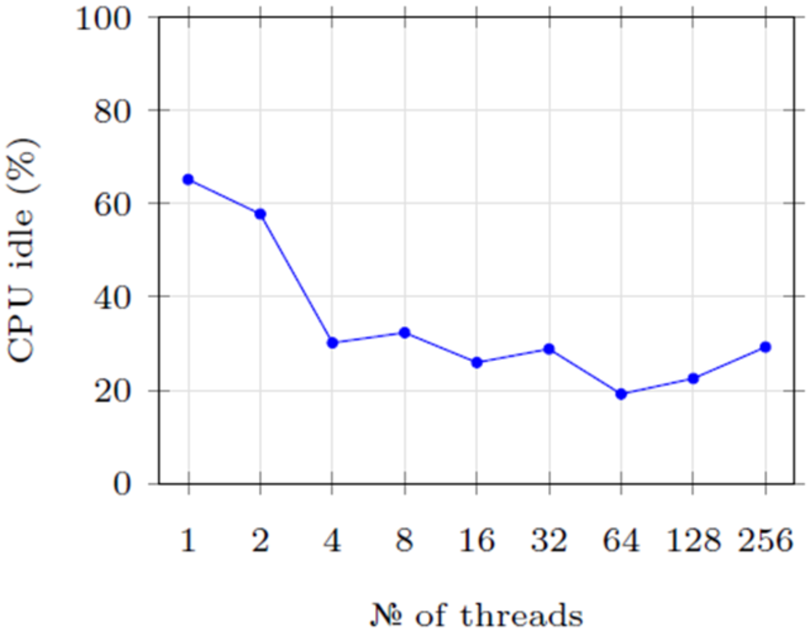
Crawling: Multi-threading Important

```
crawl(list seedUrls)
    frontier_i = seedUrls
    new set urlsSeen
    while(!frontier_i .isEmpty())
        new list frontier_i+1
        new list threads
        for url : frontier_i
                thread = new DownloadPageThread.run(url,urlsSeen,frontier_i+1)
                threads.add(thread)
        threads.poll()
        i++
 DownloadPageThread: run(url,urlsSeen,frontier_i+1)
     page = downloadPage(url)
     synchronised: urlsSeen.add(url)
     synchronised: frontier_i+1.addAll(extractUrls(page).removeAll(urlsSeen))
     synchronised: store(page)
```

Crawling: Multi-threading Important

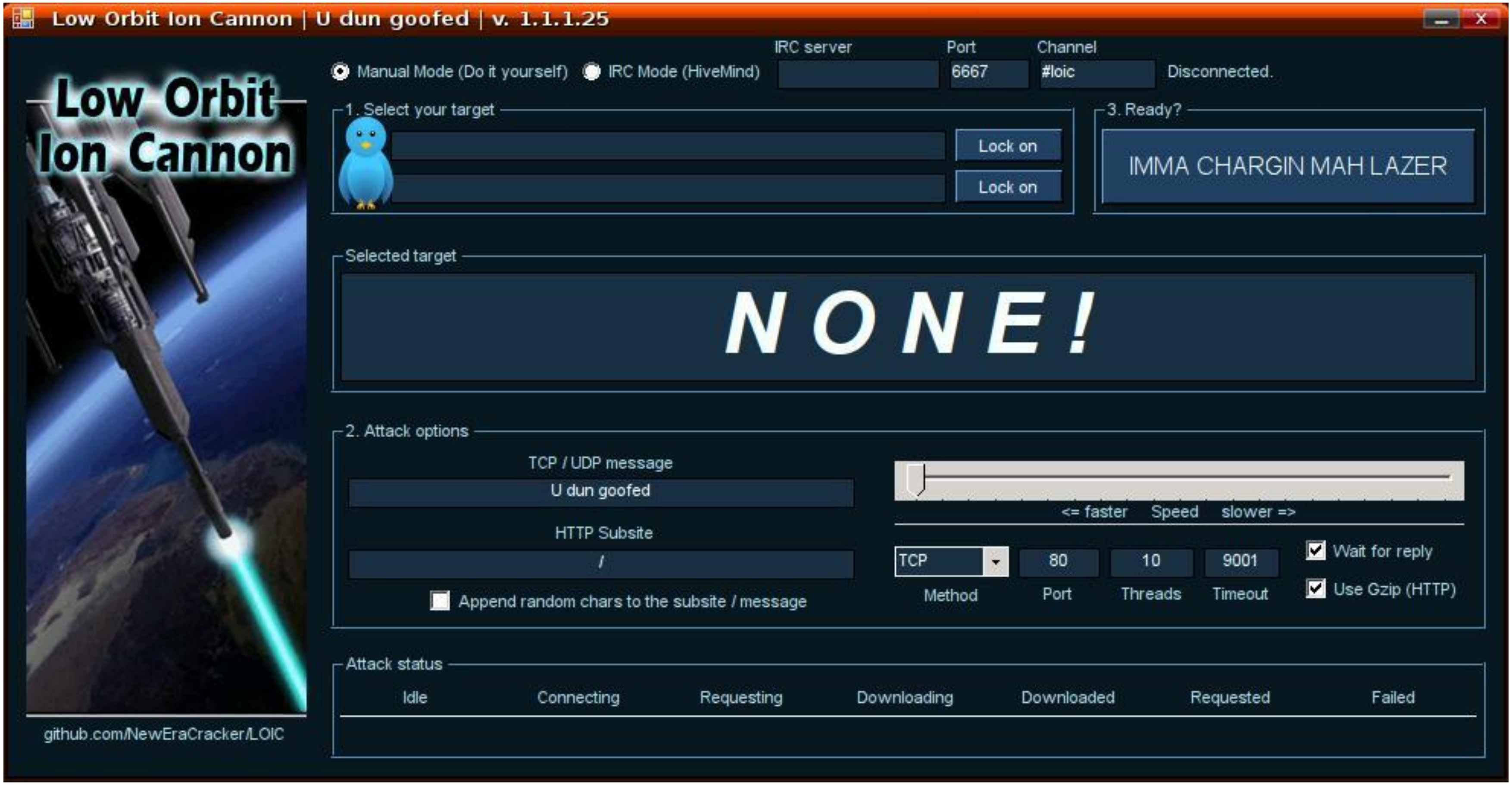
Crawl 1,000 URLs …





Crawling: Important to be Polite!

(Distributed) Denial of Server Attack: (D)DoS



Low Orbit Ion Cannon | U dun goofed | v. 1.1.1.25

● Manual Mode (Do it yourself) ○ IRC Mode (HiveMind)

IRC server:
Port: 6667
Channel: #loic
Disconnected.

1. Select your target

3. Ready?

Selected target

NONE!

2. Attack options

TCP / UDP message: U dun goofed

HTTP Subsite: /

□ Append random chars to the subsite / message

<= faster Speed slower =>

Method: TCP
Port: 80
Threads: 10
Timeout: 9001

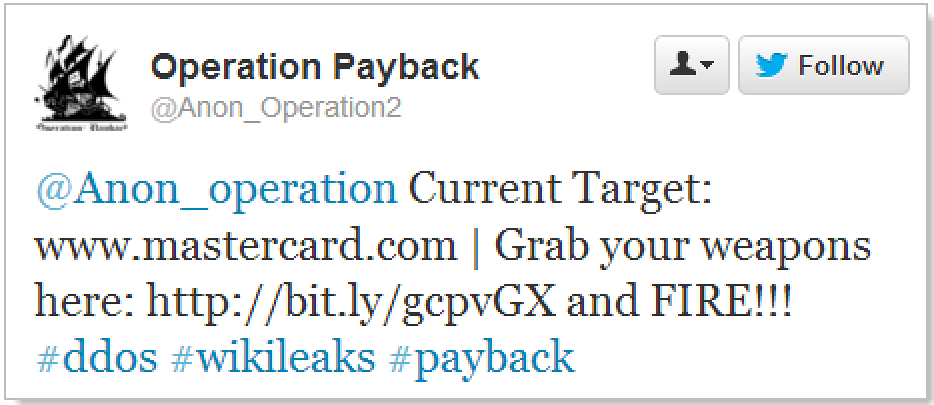
☑ Wait for reply
☑ Use Gzip (HTTP)

Attack status

Idle	Connecting	Requesting	Downloading	Downloaded	Requested	Failed

Crawling: Avoid (D)DoSing

- Christopher Weatherhead ⚠
- 18 months prison



Operation Payback
@Anon_Operation2

@Anon_operation Current Target: www.mastercard.com | Grab your weapons here: http://bit.ly/gcpvGX and FIRE!!! #ddos #wikileaks #payback

… more likely your IP range will be banned

Crawling: Web-site Scheduler

```
crawl(list seedUrls)
    frontier_i = seedUrls
    new set urlsSeen
    while(!frontier_i .isEmpty())
        new list frontier_i+1
        new list threads
        for url : schedule(frontier_i) #maximise time between two pages on one site
                thread = new DownloadPageThread.run(url,urlsSeen,fronter_i+1)
                threads.add(thread)
        threads.poll()
        i++
 DownloadPageThread: run(url,urlsSeen,frontier_i+1)
     page = downloadPage(url)
     synchronised: urlsSeen.add(url)
     synchronised: frontier_i+1.addAll(extractUrls(page) .removeAll(urlsSeen))
     synchronised: store(page)
```

Robots Exclusion Protocol

http://website.com/robots.txt

```
User-agent: *
Disallow: /
```

No bots allowed on the website.

```
User-agent: *
Disallow: /user/
Disallow: /main/login.html
```

No bots allowed in /user/ sub-folder or login page.

```
User-agent: googlebot
Disallow: /
```

Ban only the bot with “user-agent” googlebot.

Robots Exclusion Protocol (non-standard)

```
User-agent: googlebot
Crawl-delay: 10
```

Tell the googlebot to only crawl a page from this host no more than once every 10 seconds.

```
User-agent: *
Disallow: /
Allow: /public/
```

Ban everything but the /public/ folder for all agents

```
User-agent: *
Sitemap: http://example.com/main/sitemap.xml
```

Tell user-agents about your *site-map*

Site-Map: Additional crawler information

```xml
<?xml version="1.0" encoding="utf-8"?>
<urlset>
    <url>
        <loc>http://aidanhogan.com/</loc>
        <lastmod>2017-04-17</lastmod>
        <changefreq>weekly</changefreq>
        <priority>0.8</priority>
    </url>
    <url>
        <loc>http://aidanhogan.com/teaching/</loc>
        <lastmod>2017-04-04</lastmod>
        <changefreq>monthly</changefreq>
        <priority>0.5</priority>
    </url>
</urlset>
```

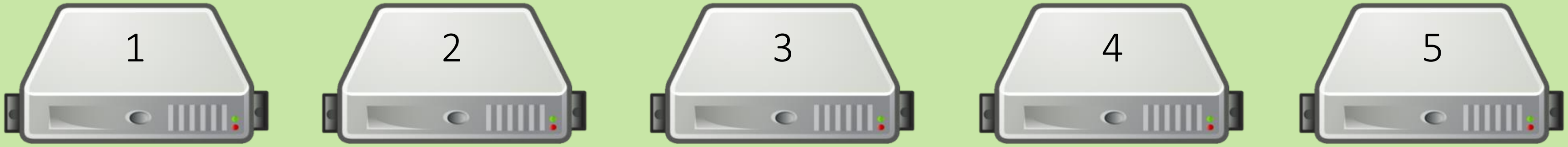
Crawling: Important Points

- Seed-list: Entry point for crawling
- Frontier: Extract links from current pages for next round
- Seen-list: Avoid cycles
- Threading: Keep machines busy
- Politeness: Don’t annoy web-sites
  - Set delay between crawling pages on the same web-site
  - Stick to what’s stated in the robots.txt file
  - Check for a site-map

Crawling: Distribution

How might we implement a distributed crawler?

```
for url : frontier_i-1
        map(url,count)
```



Similar benefits to multi-threading

What will be the bottleneck as machines increase?

Bandwidth or politeness delays

Crawling: All the Web?

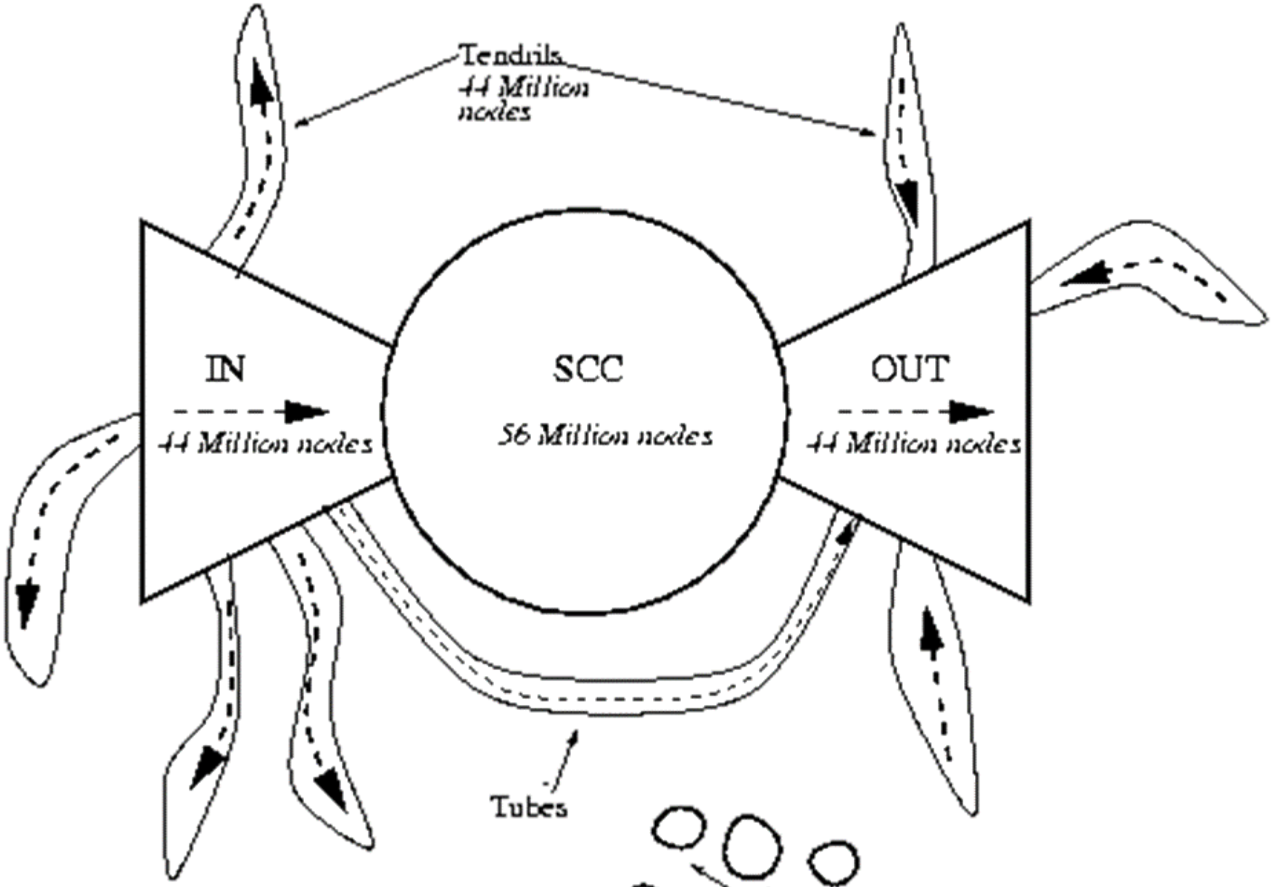
Can we crawl all the Web? ?

Crawling: All the Web?

Can ~~we~~ crawl all the Web?

Can Google crawl all the Web?

Crawling: Inaccessible (Bow-Tie)



Broder et al. “Graph structure in the web,” Comput. Networks, vol. 33, no. 1-6, pp. 309–320, 2000

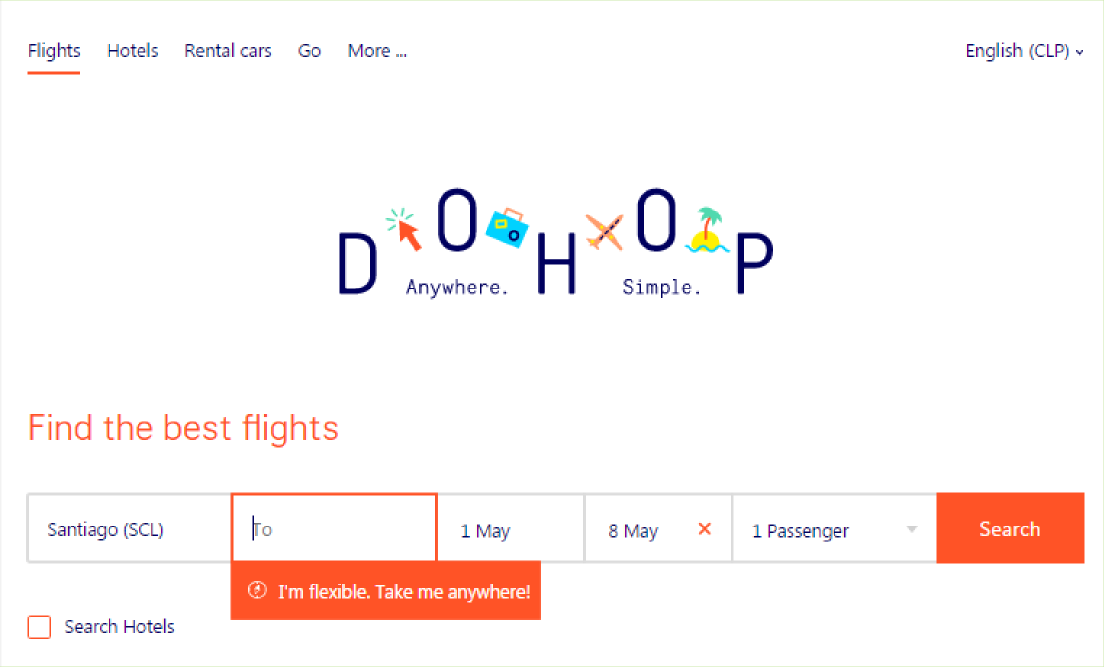
Crawling: Inaccessible (Deep Web)

What is the Deep Web?

Crawling: Inaccessible (Deep Web)

What is the Deep Web?

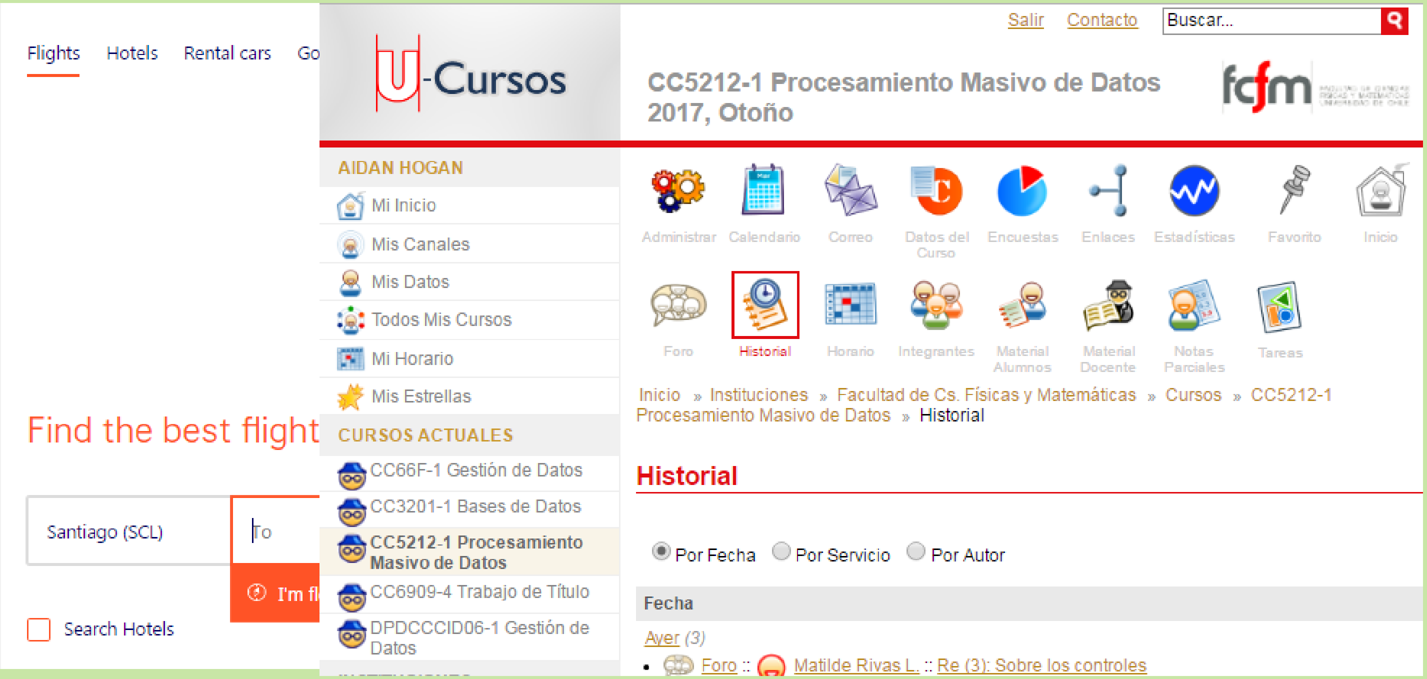
- Dynamically-generated content



Crawling: Inaccessible (Deep Web)

What is the Deep Web?

- Dynamically-generated content
- Password-protected



Flights Hotels Rental cars Go

Find the best flight

Santiago (SCL) | To

I'm fl

□ Search Hotels

U-Cursos

Salir Contacto Buscar...

CC5212-1 Procesamiento Masivo de Datos
2017, Otoño

fcfm

AIDAN HOGAN

- Mi Inicio
- Mis Canales
- Mis Datos
- Todos Mis Cursos
- Mi Horario
- Mis Estrellas

CURSOS ACTUALES

- CC66F-1 Gestión de Datos
- CC3201-1 Bases de Datos
- **CC5212-1 Procesamiento Masivo de Datos**
- CC6909-4 Trabajo de Título
- DPDCCCID06-1 Gestión de Datos

Inicio » Instituciones » Facultad de Cs. Físicas y Matemáticas » Cursos » CC5212-1 Procesamiento Masivo de Datos » Historial

Historial

● Por Fecha ○ Por Servicio ○ Por Autor

Fecha

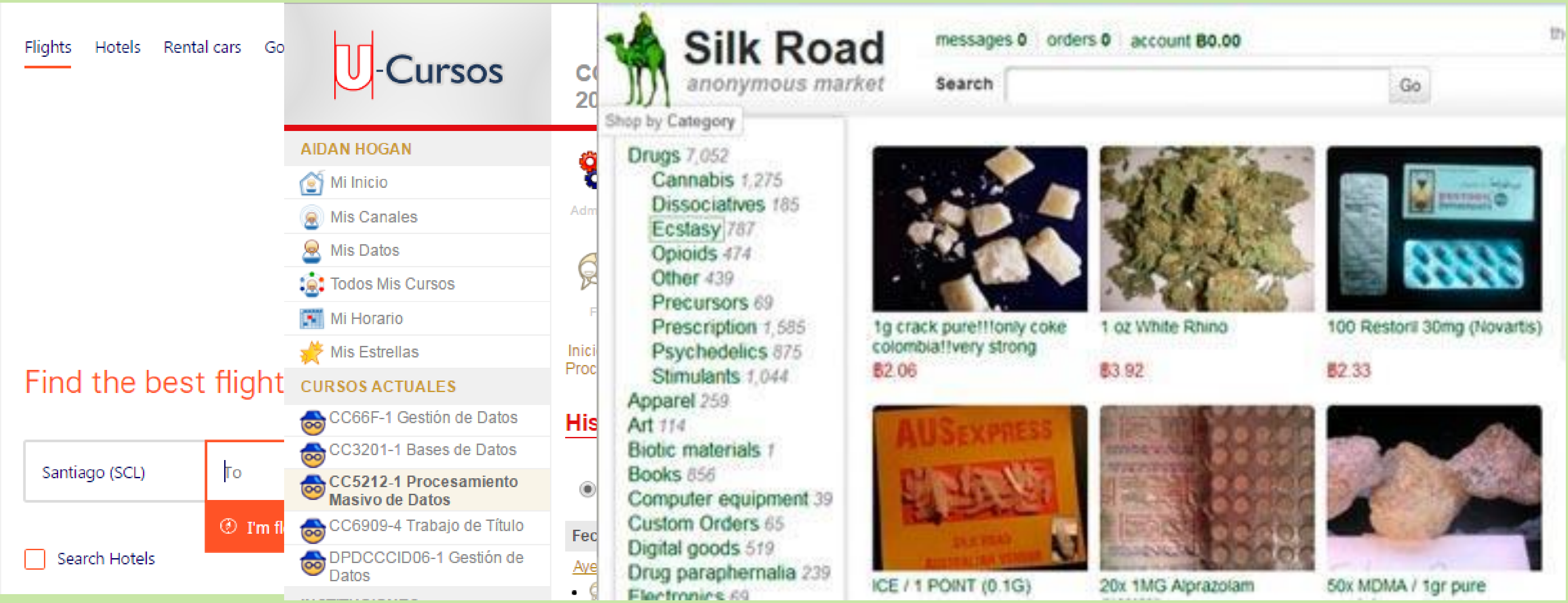
Ayer (3)

- Foro :: Matilde Rivas L. :: Re (3): Sobre los controles

Crawling: Inaccessible (Deep Web)

What is the Deep Web?

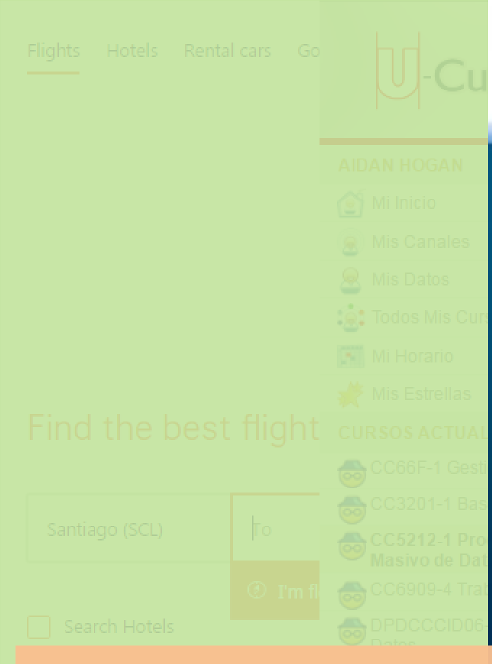
- Dynamically-generated content
- Password-protected
- Dark Web

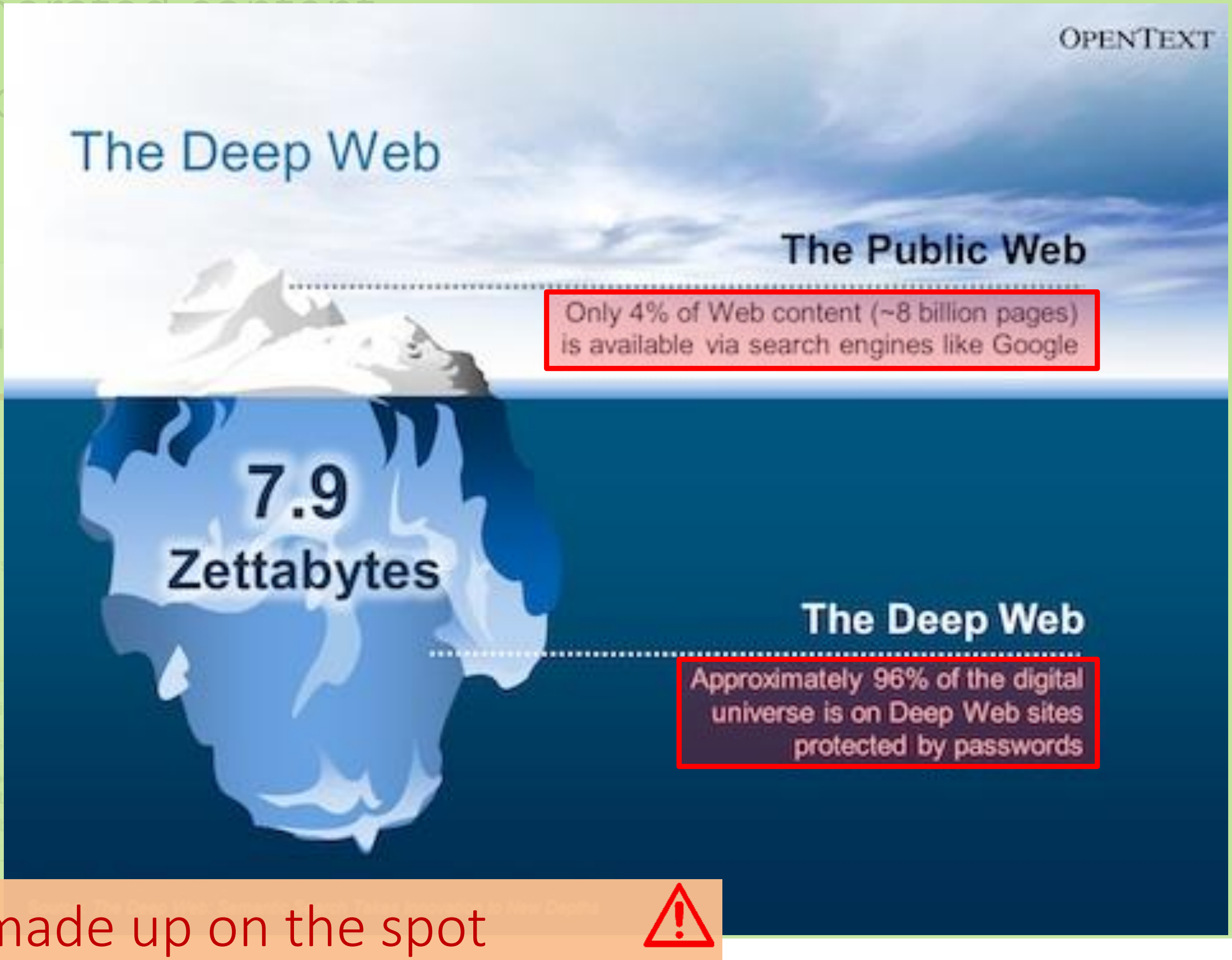


Crawling: Inaccessible (Deep Web)

What is the Deep Web?

- Dynamically-generated content
- Password-protec
- Dark Web




OPENTEXT
The Deep Web
The Public Web
Only 4% of Web content (~8 billion pages) is available via search engines like Google
7.9 Zettabytes
The Deep Web
Approximately 96% of the digital universe is on Deep Web sites protected by passwords

46% of statistics made up on the spot ⚠

Crawling: All the Web?

Can ~~we~~ crawl all the Web? ?

Can Google crawl all the Web? ?

Can Google crawl itself? ?

Apache Nutch

- Open-source crawling framework!
- Compatible with Hadoop!

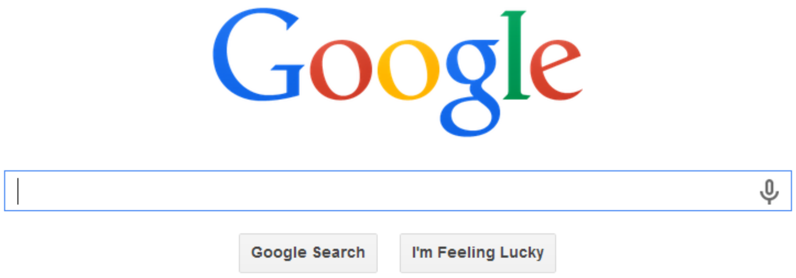
https://nutch.apache.org/

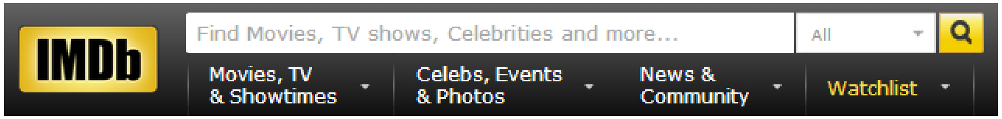
Information Retrieval: Inverted Indexing

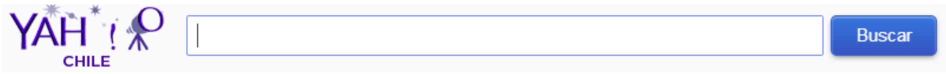
Inverted Index

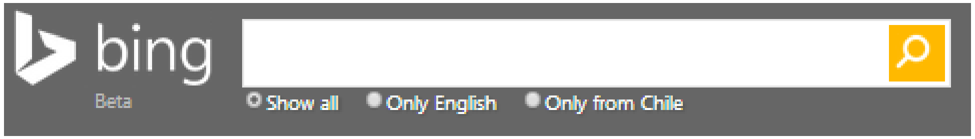
- Inverted Index: A map from words to documents
  - "Inverted" because usually documents map to words

Examples of applications?









Inverted Index: Example

1

en.wikipedia.org/wiki/Fruitvale_Station

Fruitvale Station

From Wikipedia, the free encyclopedia

Fruitvale Station is a 2013 American drama film written and directed by Ryan Coogler.

Inverted index:

Term List	Posting List
a	(1,2,…)
american	(1,5,…)
and	(1,2,…)
by	(1,2,…)
directed	(1,2,…)
drama	(1,16,…)
…	…

Inverted Index: Example Search

american drama

- AND: Intersect posting lists
- OR: Union posting lists
- PHRASE: ???

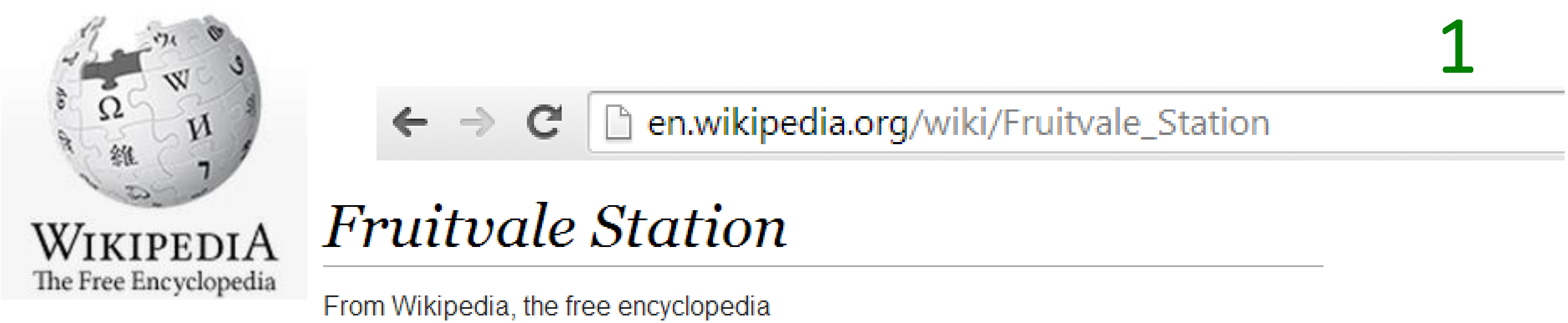
How should we implement PHRASE?

Inverted index:

Term List	Posting List
a	(1,2,…)
american	(1,5,…)
and	(1,2,…)
by	(1,2,…)
directed	(1,2,…)
drama	(1,16,…)
…	…

Inverted Index: Example

1



en.wikipedia.org/wiki/Fruitvale_Station

Fruitvale Station

From Wikipedia, the free encyclopedia

1 10 18 21 23 28 37 43 47 55 59 68 71 76

Fruitvale Station is a 2013 American drama film written and directed by Ryan Coogler.

Inverted index:

Term List	Posting List
a	(1,[21,96,103,…]), (2,[…]), …
american	(1,[28,123]), (5,[…]), …
and	(1,[57,139,…]), (2,[…]), …
by	(1,[70,157,…]), (2,[…]), …
directed	(1,[61,212,…]), (4,[…]), …
drama	(1,[38,87,…]), (16,[…]), …
…	…

Inverted Index: Flavours

Record-level inverted index:

Maps words to documents without positional information

Term List	Posting List
a	(1,2,…)
american	(1,5,…)
and	(1,2,…)
by	(1,2,…)
directed	(1,2,…)
drama	(1,16,…)
…	…

Word-level inverted index:

Additionally maps words with positional information

Term List	Posting List
a	(1,[21,96,103,…]), (2,[…]), …
american	(1,[28,123]), (5,[…]), …
and	(1,[57,139,…]), (2,[…]), …
by	(1,[70,157,…]), (2,[…]), …
directed	(1,[61,212,…]), (4,[…]), …
drama	(1,[38,87,…]), (16,[…]), …
…	…

Inverted Index: Word Normalisation

drama america

How can we solve this problem?

Inverted index:

Term List	Posting List
a	(1,[21,96,103,…]), (2,[…]), …
american	(1,[28,123]), (5,[…]), …
and	(1,[57,139,…]), (2,[…]), …
by	(1,[70,157,…]), (2,[…]), …
directed	(1,[61,212,…]), (4,[…]), …
drama	(1,[38,87,…]), (16,[…]), …
…	…

Inverted Index: Word Normalisation

drama america

How can we solve this problem?

Normalise words:

Stemming cuts the ends off of words using generic rules:

{ America , American , americas , americanise } → { america }

Inverted index:

Term List	Posting List
a	(1,[21,96,103,…]), (2,[…]), …
american	(1,[28,123]), (5,[…]), …
and	(1,[57,139,…]), (2,[…]), …
by	(1,[70,157,…]), (2,[…]), …
directed	(1,[61,212,…]), (4,[…]), …
drama	(1,[38,87,…]), (16,[…]), …
…	…

Inverted Index: Word Normalisation

drama america

How can we solve this problem?

Normalise words:

Stemming cuts the ends off of words using generic rules:

{ America , American , americas , americanise } → { america }

Lemmatisation uses knowledge of the word to normalise:

{ better , goodly , best } → { good }

Inverted index:

Term List	Posting List
a	(1,[21,96,103,…]), (2,[…]), …
american	(1,[28,123]), (5,[…]), …
and	(1,[57,139,…]), (2,[…]), …
by	(1,[70,157,…]), (2,[…]), …
directed	(1,[61,212,…]), (4,[…]), …
drama	(1,[38,87,…]), (16,[…]), …
…	…

Inverted Index: Word Normalisation

drama america

How can we solve this problem?

Normalise words:

Stemming cuts the ends off of words using generic rules:

{ America , American , americas , americanise } → { america }

Lemmatisation uses knowledge of the word to normalise:

{ better , goodly , best } → { good }

Synonym expansion

{ film , movie } → { movie }

Inverted index:

Term List	Posting Lists
a	(1,[21,96,103,…]), (2,[…]), …
america	(1,[28,123]), (5,[…]), …
and	(1,[57,139,…]), (2,[…]), …
by	(1,[70,157,…]), (2,[…]), …
directed	(1,[61,212,…]), (4,[…]), …
drama	(1,[38,87,…]), (16,[…]), …
…	…

Inverted Index: Word Normalisation

drama america

How can we solve this problem?

Normalise words:

Stemming cuts the ends off of words using generic rules:

{ America , American , americas , americanise } → { america }

Lemmatisation uses knowledge of the word to normalise:

{ better , goodly , best } → { good }

Synonym expansion

{ film , movie } → { movie }

- ➢ Language specific!
- ➢ Use same normalisation on query and document!

Inverted index:

Term List	Posting Lists
a	(1,[21,96,103,…]), (2,[…]), …
america	(1,[28,123]), (5,[…]), …
and	(1,[57,139,…]), (2,[…]), …
by	(1,[70,157,…]), (2,[…]), …
directed	(1,[61,212,…]), (4,[…]), …
drama	(1,[38,87,…]), (16,[…]), …
…	…

Inverted Index: Space

Record-level inverted index:

Maps words to documents without positional information

Term List	Posting List
a	(1,2,…)
american	(1,5,…)
and	(1,2,…)
by	(1,2,…)
directed	(1,2,…)
drama	(1,16,…)
…	…

Space?

$\sum_{d \in D} \mathrm{U}(d)$ (sum of unique words in all docs)

Word-level inverted index:

Additionally maps words with positional information

Term List	Posting List
a	(1,[21,96,103,…]), (2,[…]), …
american	(1,[28,123]), (5,[…]), …
and	(1,[57,139,…]), (2,[…]), …
by	(1,[70,157,…]), (2,[…]), …
directed	(1,[61,212,…]), (4,[…]), …
drama	(1,[38,87,…]), (16,[…]), …
…	…

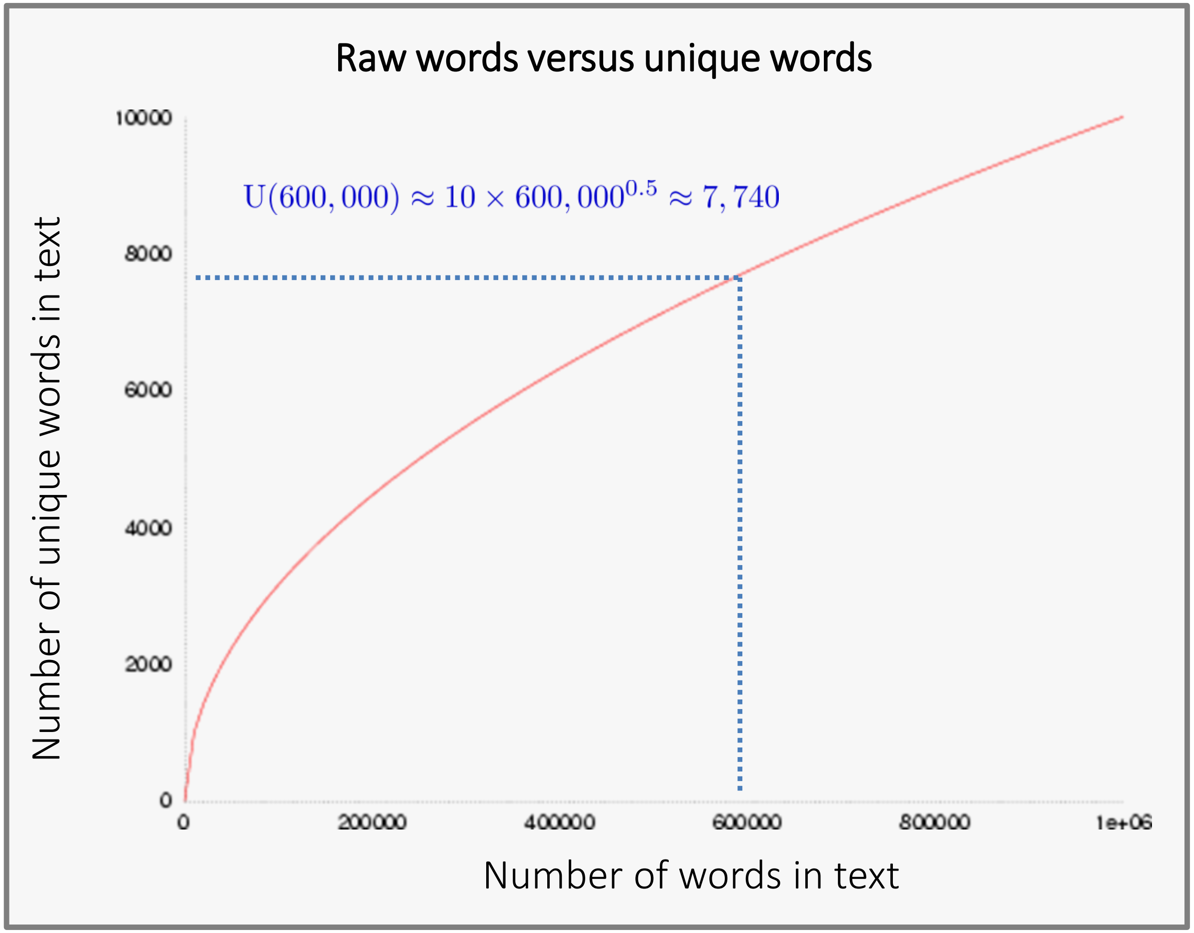
Space?

$\sum_{d \in D} \mathrm{W}(d)$ (sum of all word occurrences in all docs)

Inverted Index: Unique Words

Not so many unique words …

- Heap's law: $\mathrm{U}(n) \approx Kn^{\beta}$
- English text
  - K ∈ [10,100]
  - β ∈ [0.4,0.6]



Inverted Index: Space

$\mathrm{U}(d) \approx K \times \mathrm{W}(d)^{\beta}$ ⚠

Record-level inverted index:

Maps words to documents without positional information

Term List	Posting List
a	(1,2,…)
american	(1,5,…)
and	(1,2,…)
by	(1,2,…)
directed	(1,2,…)
drama	(1,16,…)
…	…

Space? ? $\sum_{d \in D} \mathrm{U}(d)$ (sum of unique words in all docs)

Word-level inverted index:

Additionally maps words with positional information

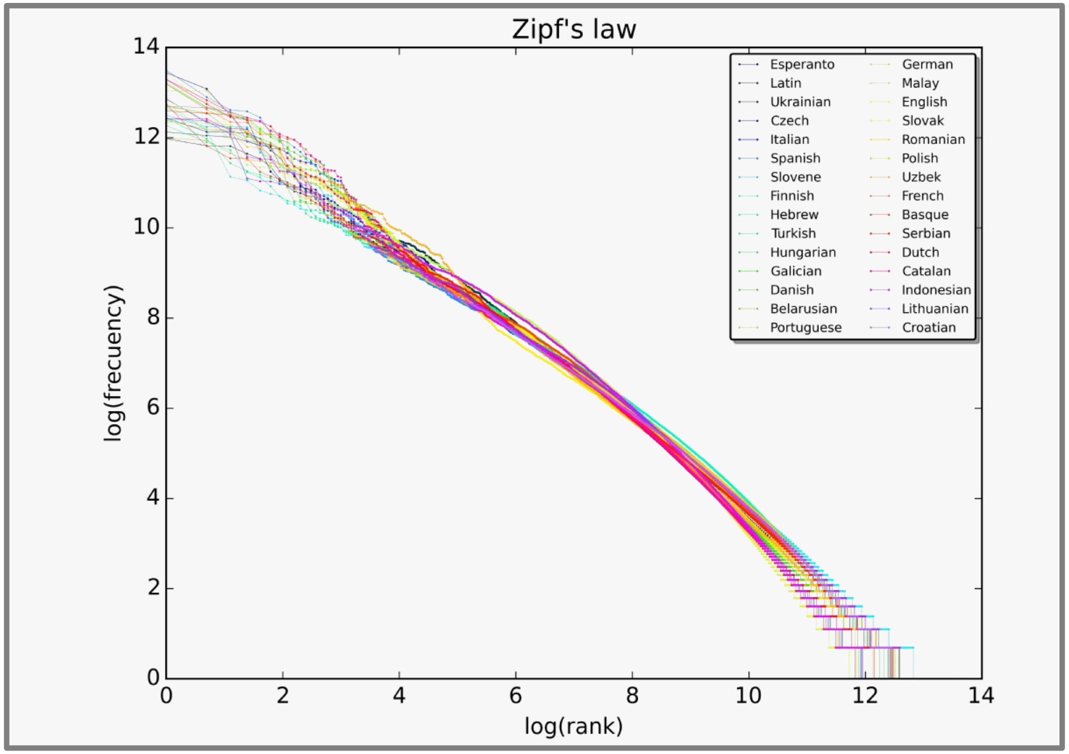
Term List	Posting List
a	(1,[21,96,103,…]), (2,[…]), …
american	(1,[28,123]), (5,[…]), …
and	(1,[57,139,…]), (2,[…]), …
by	(1,[70,157,…]), (2,[…]), …
directed	(1,[61,212,…]), (4,[…]), …
drama	(1,[38,87,…]), (16,[…]), …
…	…

Space? ? $\sum_{d \in D} \mathrm{W}(d)$ (sum of all word occurrences in all docs)

Inverted Index: Common Words

Many occurrences of few words / Few occurrences of many words

- Zipf’s law
- In English text:
  - “the” 7%
  - “of” 3.5%
  - “and” 2.7%
  - 135 words cover half of all occurrences



Zipf’s law: *the most popular word will occur twice as often as the second most popular word, thrice as often as the third most popular word, n times as often as the n-most popular word.*

Inverted Index: Common Words

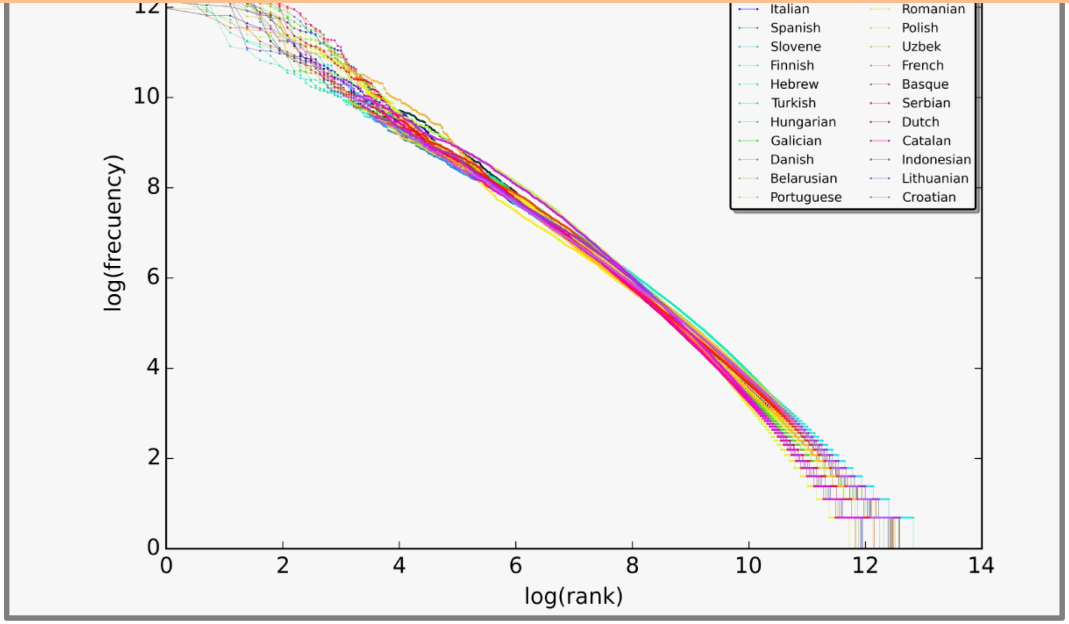
Many occurrences of few words

/ Few occurrences of many words

Expect long posting lists for common words ⚠

– In English text:

- “the” 7%
- “of” 3.5%
- “and” 2.7%
- 135 words cover half of all occurrences



Zipf’s law: *the most popular word will occur twice as often as the second most popular word, thrice as often as the third most popular word, n times as often as the n-most popular word.*

Inverted Index: Common Words

- Perhaps implement stop-words?
    - Most common words contain least information

```
the drama in america
```

Inverted Index: Common Words

- Perhaps implement stop-words?
- Perhaps implement block-addressing?

Fruitvale Station is a 2013 American drama film written and directed by Ryan Coogler.

Block 1

Block 2

What is the effect on phrase search?

Small blocks ~ okay
Big blocks ~ not okay

Term List	Posting List
a	(1,[1,…]), (2,[…]), …
american	(1,[1,…]), (5,[…]), …
and	(1,[2, …]), (2,[…]), …
by	(1,[2, …]), (2,[…]), …
…	…

Inverted Index: Common Words

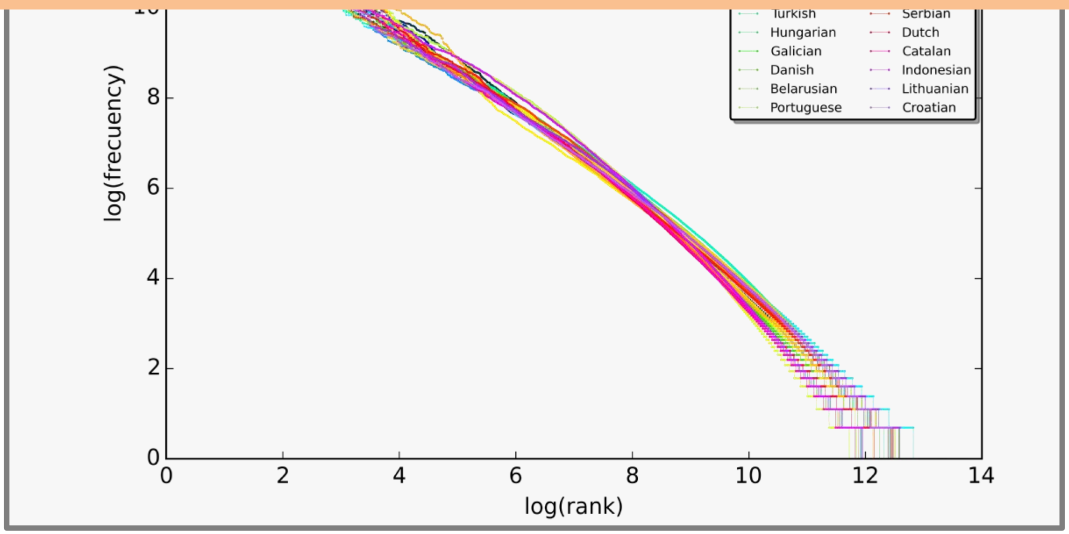
Many occurrences of few words

/ Few occurrences of many words

Expect long posting lists for common words ⚠

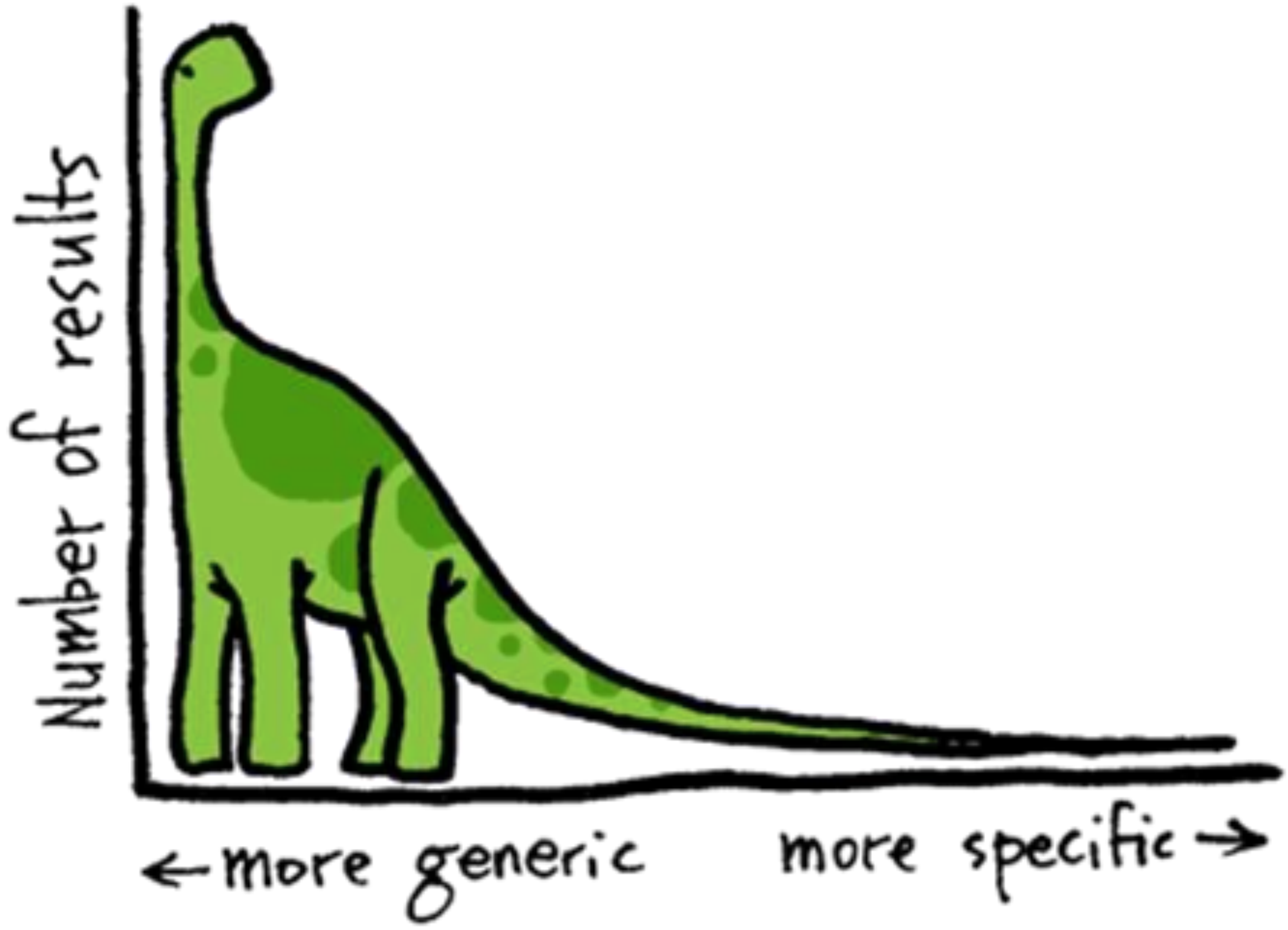
Expect more queries with common words

- “the” 7%
- “of” 3.5%
- “and” 2.7%
- 135 words cover half of all occurrences

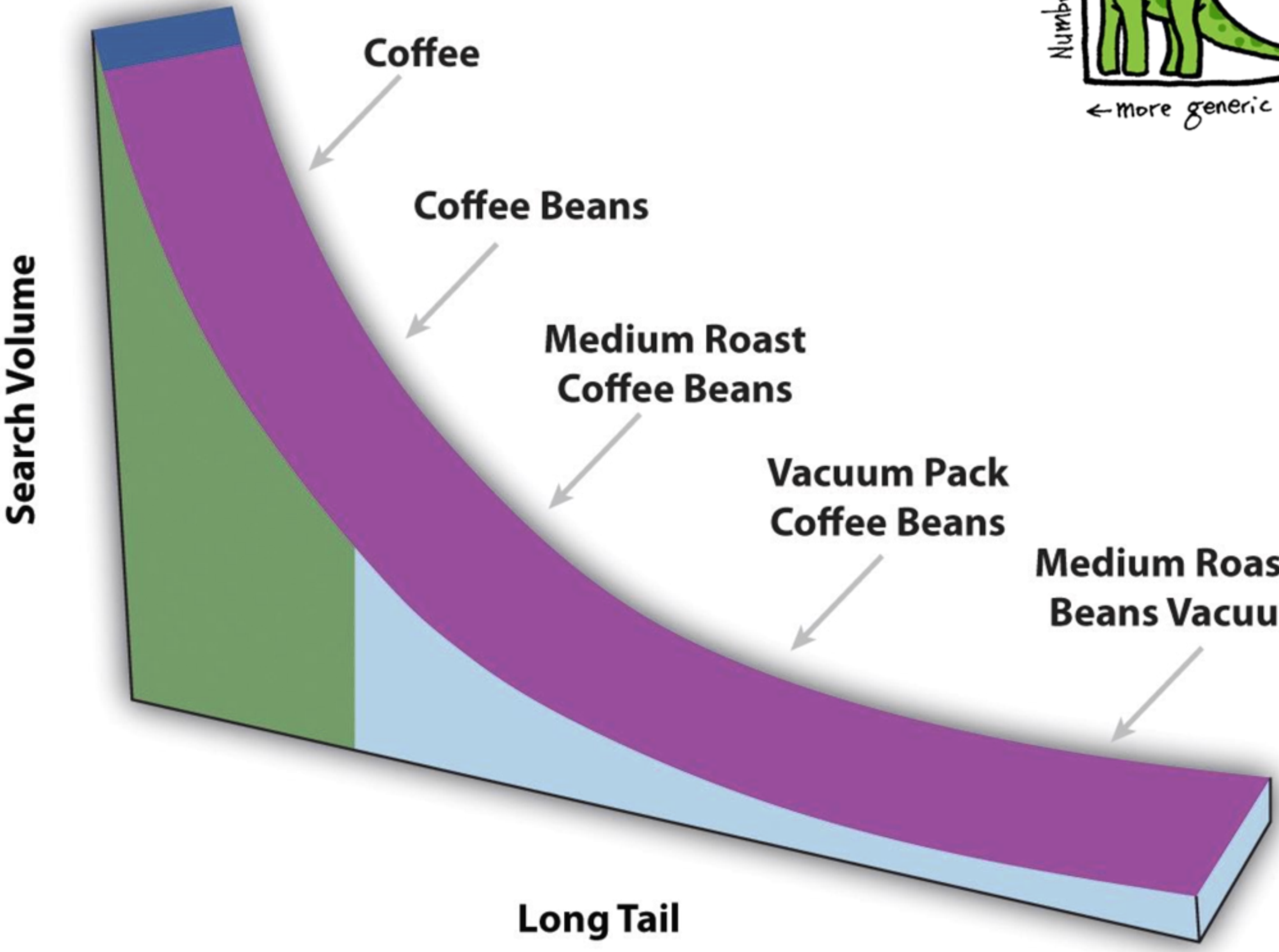


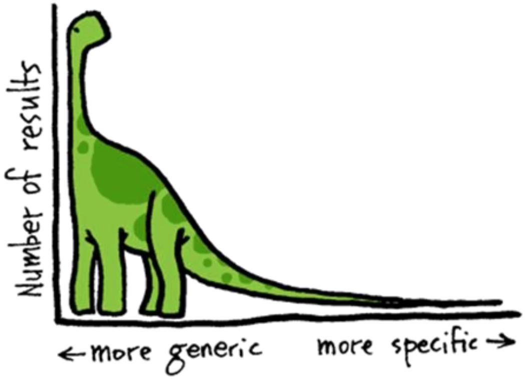
Zipf’s law: *the most popular word will occur twice as often as the second most popular word, thrice as often as the third most popular word, n times as often as the n-most popular word.*

The Long Tail of Search



The Long Tail of Search

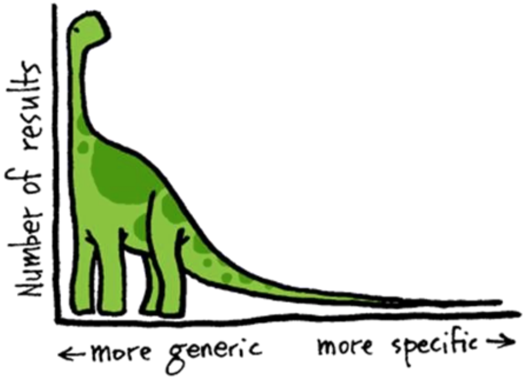


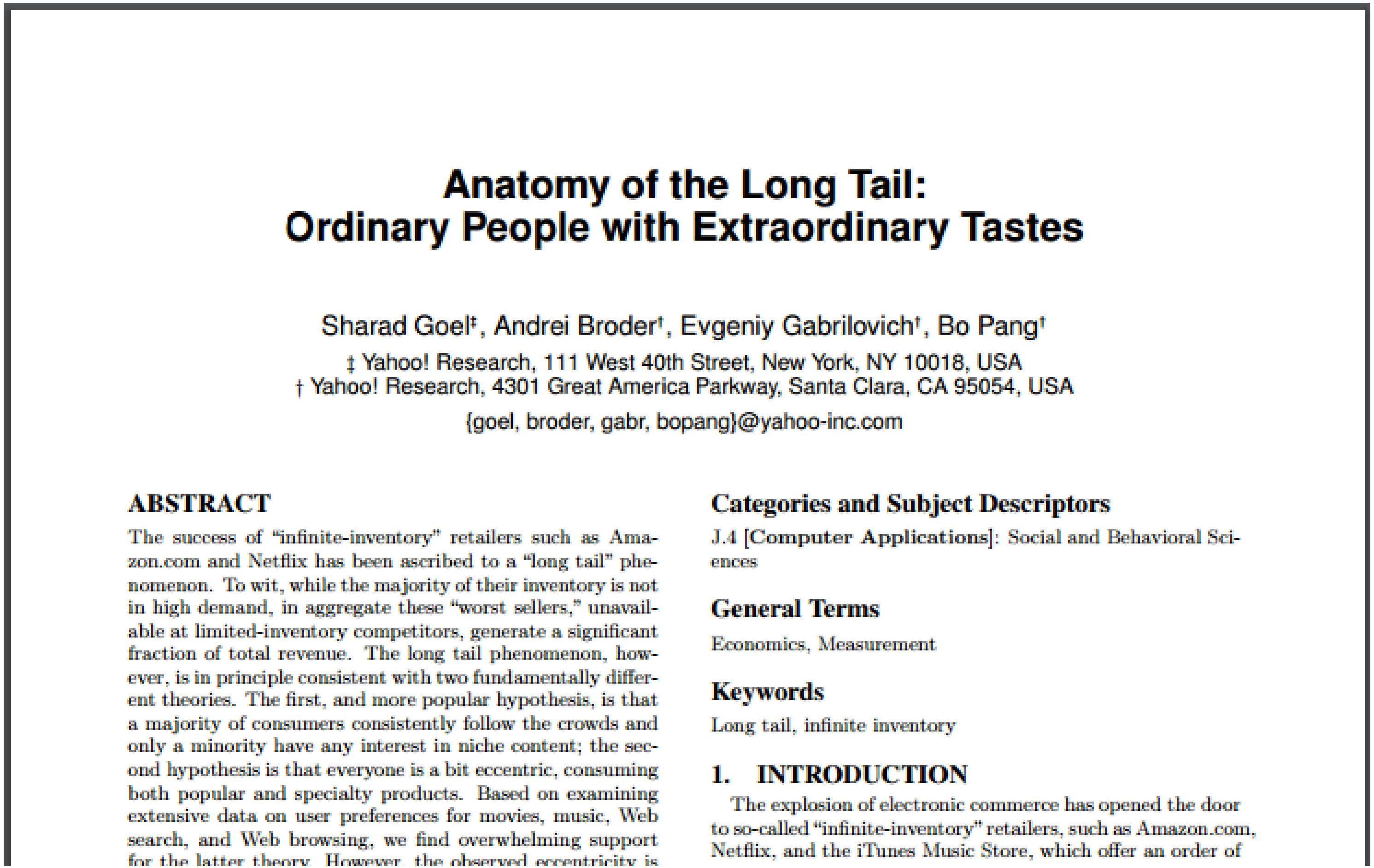


How to optimise for this?

? Caching for common queries like “coffee”

If interested …





Anatomy of the Long Tail: Ordinary People with Extraordinary Tastes

Sharad Goel[‡], Andrei Broder[†], Evgeniy Gabrilovich[†], Bo Pang[†]

‡ Yahoo! Research, 111 West 40th Street, New York, NY 10018, USA
† Yahoo! Research, 4301 Great America Parkway, Santa Clara, CA 95054, USA

{goel, broder, gabr, bopang}@yahoo-inc.com

ABSTRACT

The success of "infinite-inventory" retailers such as Amazon.com and Netflix has been ascribed to a "long tail" phenomenon. To wit, while the majority of their inventory is not in high demand, in aggregate these "worst sellers," unavailable at limited-inventory competitors, generate a significant fraction of total revenue. The long tail phenomenon, however, is in principle consistent with two fundamentally different theories. The first, and more popular hypothesis, is that a majority of consumers consistently follow the crowds and only a minority have any interest in niche content; the second hypothesis is that everyone is a bit eccentric, consuming both popular and specialty products. Based on examining extensive data on user preferences for movies, music, Web search, and Web browsing, we find overwhelming support for the latter theory. However, the observed eccentricity is

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences

General Terms

Economics, Measurement

Keywords

Long tail, infinite inventory

1. INTRODUCTION

The explosion of electronic commerce has opened the door to so-called "infinite-inventory" retailers, such as Amazon.com, Netflix, and the iTunes Music Store, which offer an order of

Search Implementation

- Vocabulary keys:
  - Hashing: O(1) lookups (assuming ideal hashing)
    - no range queries
    - relatively easy to update (though rehashing expensive!)
  - Sorting/B-Tree: O(log(u)) lookups, u unique words
    - range queries
    - tricky to update (standard methods for B-trees)
  - Tries: O(l) lookups, l length of the word
    - range queries, compressed, auto-completion!
    - referencing becomes tricky (on disk)

Tries? (in class) ?

Memory Sizes

- Term list (vocabulary keys) small:
  - Often will fit in memory!
- Posting lists larger:
  - On disk / Hot regions cached

Term List	Posting List
a	(1,[21,96,103,…]), (2,[…]), …
american	(1,[28,123]), (5,[…]), …
and	(1,[57,139,…]), (2,[…]), …
by	(1,[70,157,…]), (2,[…]), …
directed	(1,[61,212,…]), (4,[…]), …
drama	(1,[38,87,…]), (16,[…]), …
…	…

Compression techniques

- Numeric compression important

Term List	Posting List
country	(1), (2), (3), (4), (6), (7), …
…	…

Compression techniques: High Level

- Interval indexing
  - Example for record-level indexing
    - Could also be applied for block-level indexing, etc.

Term List	Posting List
country	(1), (2), (3), (4), (6), (7), …
…	…

Term List	Posting List
country	(1-4), (6-7),
…	…

Compression techniques: High Level

- Gap indexing
  - – Example for record-level indexing
    - Could also be applied for block-level indexing, etc.

Term List	Posting List
country	(1), (3), (4), (8), (9), …
…	…

Term List	Posting Lists
country	(1), 2, 1, 4, 1
…	…

Benefit?

Repeated small numbers easier to compress!

Compression techniques: Bit Level

- Variable length coding: bit-level techniques
- For example, Elias γ (gamma) encoding
  - Assumes many small numbers

$2\lfloor \log_2(z) \rfloor + 1$ bits

z: integer to encode	n = $\lfloor \log_2(z) \rfloor$ coded in unary	a zero marker	next n binary numbers	final Elias γ code
1	0			0
2	1	0	0	100
3	1	0	1	101
4	11	0	00	11000
5	11	0	01	11001
6	11	0	10	11010
7	11	0	11	11011
8	111	0	000	1110000
…	…	…	…	…

Can you decode "01000011000111000011001"? <1,2,1,1,4,8,5>

Compression techniques: Bit Level

- Variable length coding: bit-level techniques
- For example, Elias δ (delta) encoding
  - Better for some distributions

$\lfloor \log_2(z) \rfloor + 2\lfloor \log_2(\lfloor \log_2(z) \rfloor + 1) \rfloor + 1$ bits

z: integer to encode	$\lfloor \log_2(z) \rfloor + 1$ coded in **Elias γ**	next $\lfloor \log_2(z) \rfloor$ binary numbers	final Elias δ code
1	0		0
2	100	0	1000
3	100	1	1001
4	101	00	10100
5	101	01	10101
6	101	10	10110
7	101	11	10111
8	11000	000	11000000
…	…	…	…

Can you decode “0110000011001011001001”? ? <1,9,3,1,17>

Compression techniques: Bit Level

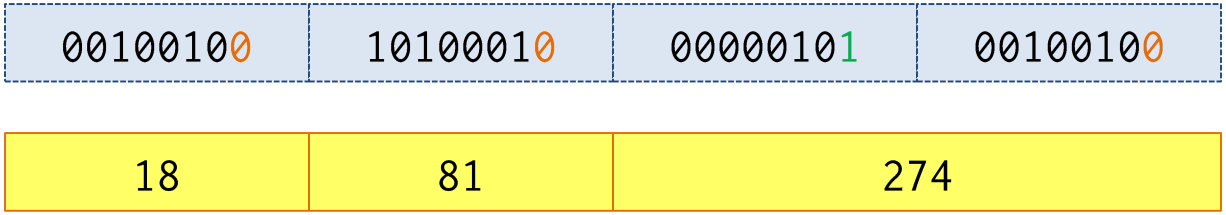
- Previous methods “non-parametric”
  - Don’t take an input value
- Other compression techniques parametric:
  - for example, Golomb-*3* code:

z: integer to encode	$n = \lfloor (z-1)/3 \rfloor$ coded in unary	zero separator	remainder	final Golomb-3 code
1	0		0	00
2	0		10	010
3	0		11	011
4	1	0	0	100
5	1	0	10	1010
6	1	0	11	1011
7	11	0	0	1100
8	11	0	10	11010
…	…		…	…

Compression techniques: Byte Level

- Use variable length byte codes
- Use last bit of byte to indicate if the number ends

- For example:



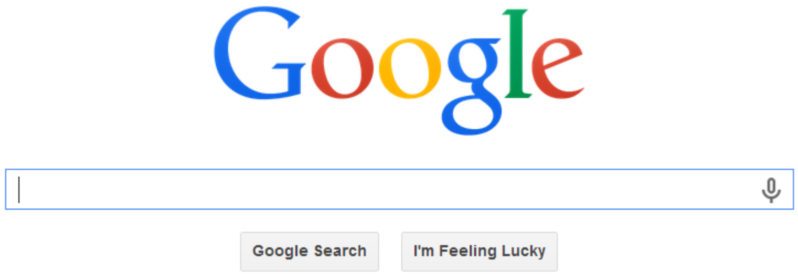
Other Optimisations

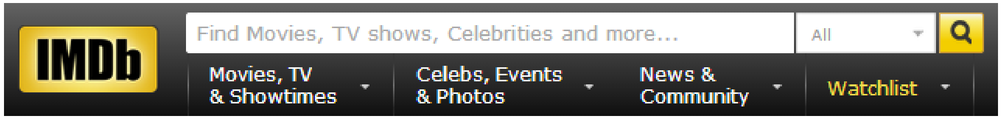
- Top-Doc: Order posting lists to give likely “top documents” first: good for top-*k* results
- Selectivity: Load the posting lists for the most rare keywords first; apply thresholds
- Sharding: Distribute over multiple machines

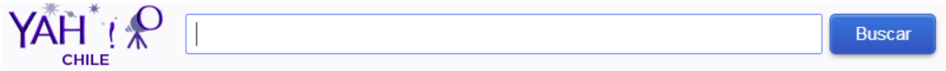
How to distribute? (in class) ?

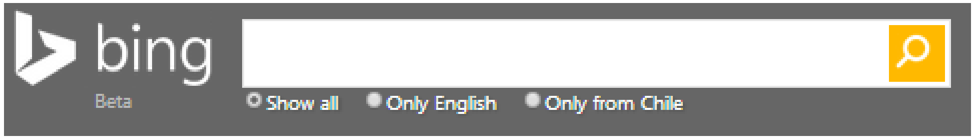
Extremely Scalable/Efficient

When engineered correctly ☺









Lucene: Text Indexing

Apache Lucene

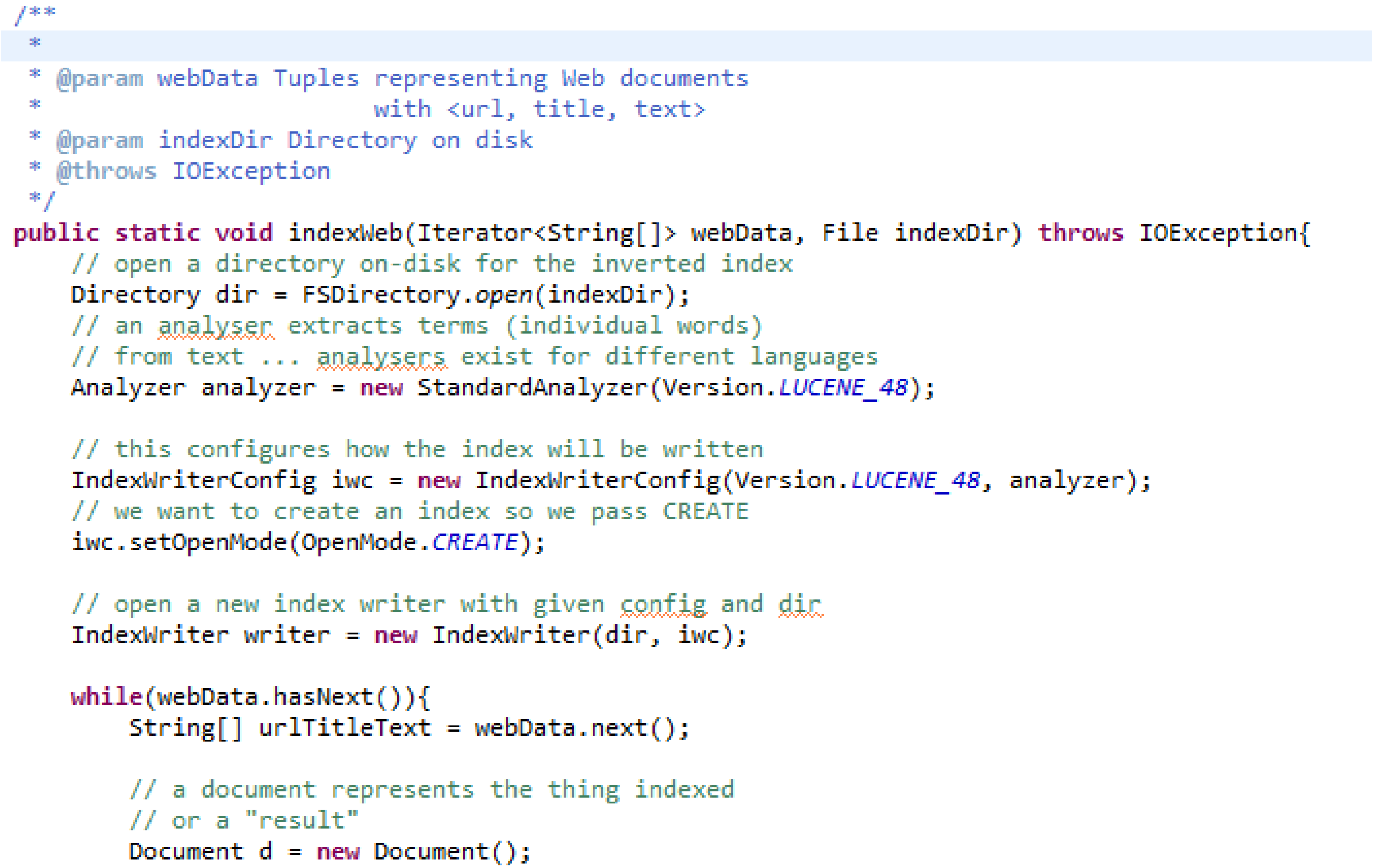
- Inverted Index
  - They built one so you don’t have to!
  - Open Source in Java

(Apache Solr)

- Built on top of Apache Lucene
- Lucene is the inverted index
- Solr is a distributed search platform, with distribution, fault tolerance, etc.

- *We will work with Lucene in the lab*

Apache Lucene: Indexing Documents

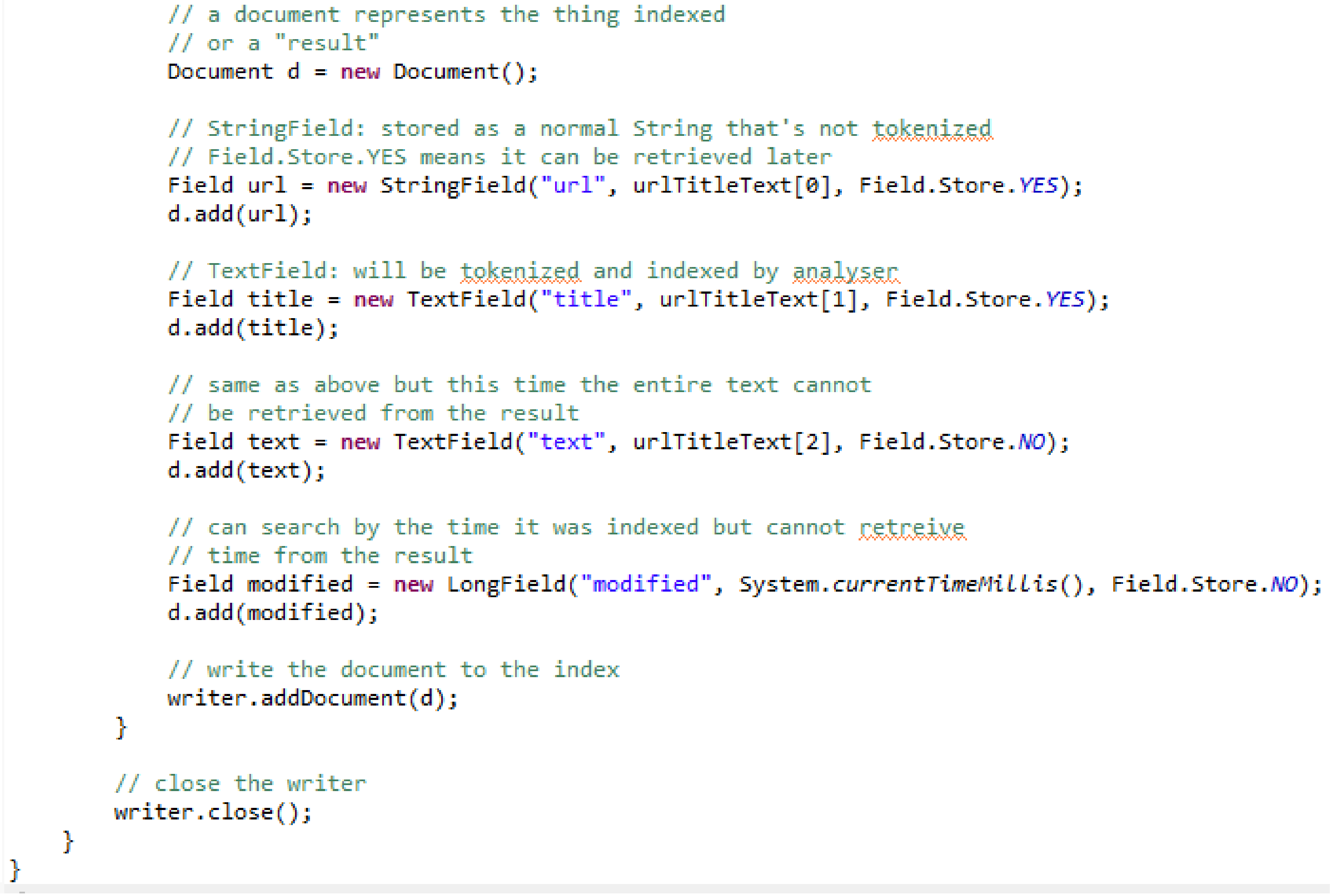


```
/**
 *
 * @param webData Tuples representing Web documents
 *                        with <url, title, text>
 * @param indexDir Directory on disk
 * @throws IOException
 */
public static void indexWeb(Iterator<String[]> webData, File indexDir) throws IOException{
    // open a directory on-disk for the inverted index
    Directory dir = FSDirectory.open(indexDir);
    // an analyser extracts terms (individual words)
    // from text ... analysers exist for different languages
    Analyzer analyzer = new StandardAnalyzer(Version.LUCENE_48);

    // this configures how the index will be written
    IndexWriterConfig iwc = new IndexWriterConfig(Version.LUCENE_48, analyzer);
    // we want to create an index so we pass CREATE
    iwc.setOpenMode(OpenMode.CREATE);

    // open a new index writer with given config and dir
    IndexWriter writer = new IndexWriter(dir, iwc);

    while(webData.hasNext()){
        String[] urlTitleText = webData.next();

        // a document represents the thing indexed
        // or a "result"
        Document d = new Document();
```

… continued …

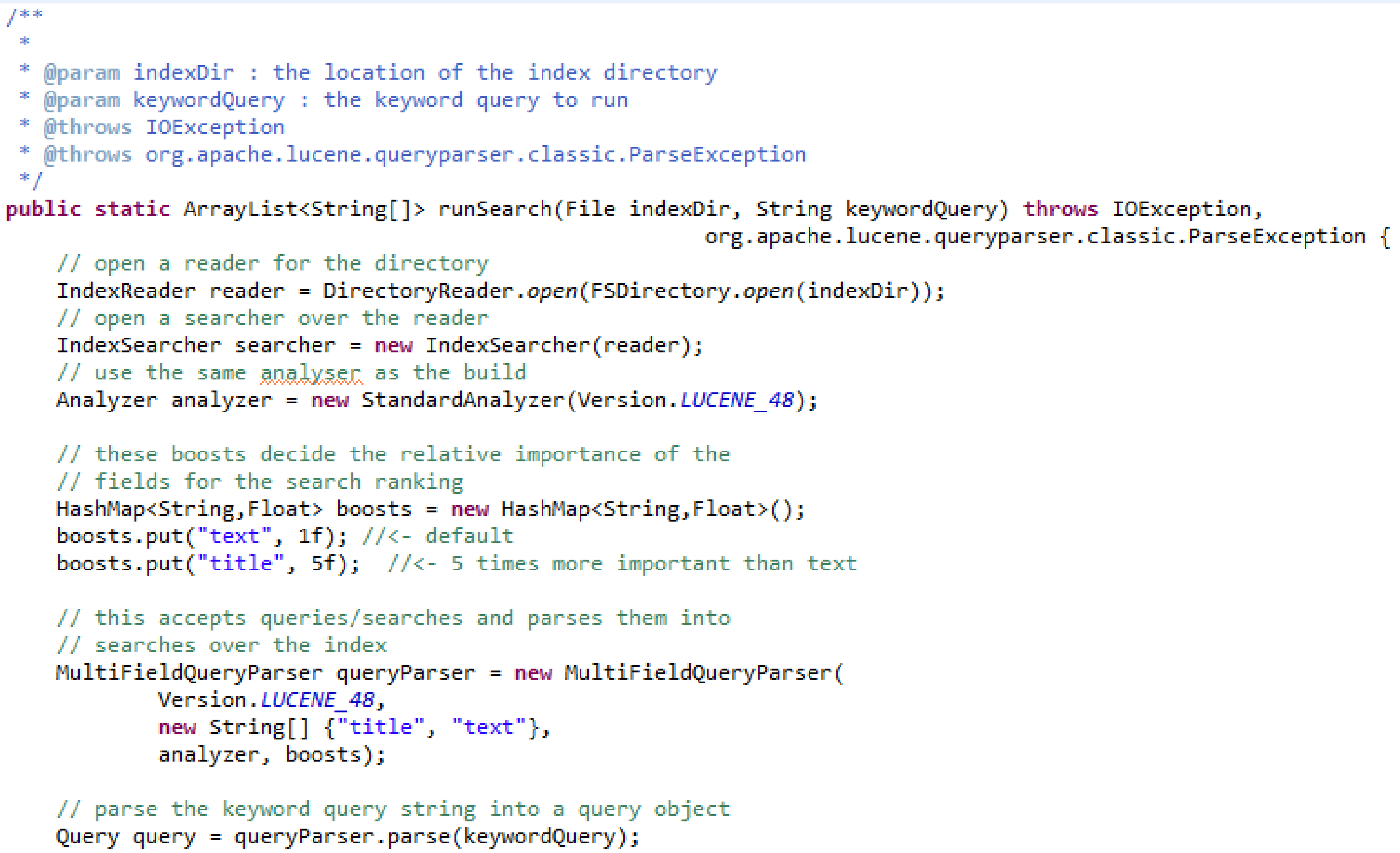
Apache Lucene: Indexing Documents

… continued …



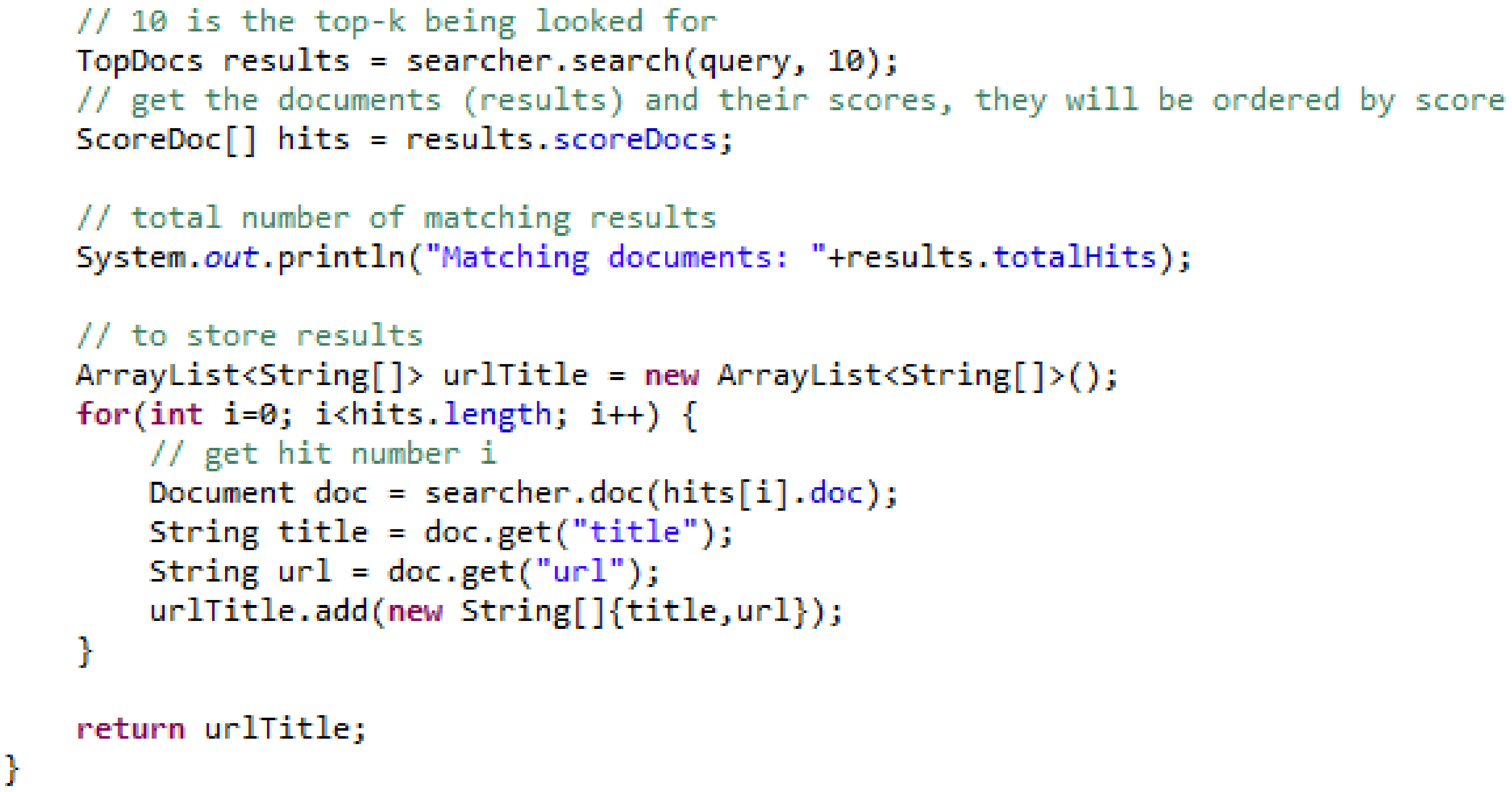
```
            // a document represents the thing indexed
            // or a "result"
            Document d = new Document();

            // StringField: stored as a normal String that's not tokenized
            // Field.Store.YES means it can be retrieved later
            Field url = new StringField("url", urlTitleText[0], Field.Store.YES);
            d.add(url);

            // TextField: will be tokenized and indexed by analyser
            Field title = new TextField("title", urlTitleText[1], Field.Store.YES);
            d.add(title);

            // same as above but this time the entire text cannot
            // be retrieved from the result
            Field text = new TextField("text", urlTitleText[2], Field.Store.NO);
            d.add(text);

            // can search by the time it was indexed but cannot retreive
            // time from the result
            Field modified = new LongField("modified", System.currentTimeMillis(), Field.Store.NO);
            d.add(modified);

            // write the document to the index
            writer.addDocument(d);
        }

        // close the writer
        writer.close();
    }
}
```

Apache Lucene: Searching Documents



```
/**
 *
 * @param indexDir : the location of the index directory
 * @param keywordQuery : the keyword query to run
 * @throws IOException
 * @throws org.apache.lucene.queryparser.classic.ParseException
 */
public static ArrayList<String[]> runSearch(File indexDir, String keywordQuery) throws IOException,
                                                       org.apache.lucene.queryparser.classic.ParseException {
    // open a reader for the directory
    IndexReader reader = DirectoryReader.open(FSDirectory.open(indexDir));
    // open a searcher over the reader
    IndexSearcher searcher = new IndexSearcher(reader);
    // use the same analyser as the build
    Analyzer analyzer = new StandardAnalyzer(Version.LUCENE_48);

    // these boosts decide the relative importance of the
    // fields for the search ranking
    HashMap<String,Float> boosts = new HashMap<String,Float>();
    boosts.put("text", 1f); //<- default
    boosts.put("title", 5f);  //<- 5 times more important than text

    // this accepts queries/searches and parses them into
    // searches over the index
    MultiFieldQueryParser queryParser = new MultiFieldQueryParser(
            Version.LUCENE_48,
            new String[] {"title", "text"},
            analyzer, boosts);

    // parse the keyword query string into a query object
    Query query = queryParser.parse(keywordQuery);
```

Apache Lucene: Searching Documents



```
    // 10 is the top-k being looked for
    TopDocs results = searcher.search(query, 10);
    // get the documents (results) and their scores, they will be ordered by score
    ScoreDoc[] hits = results.scoreDocs;

    // total number of matching results
    System.out.println("Matching documents: "+results.totalHits);

    // to store results
    ArrayList<String[]> urlTitle = new ArrayList<String[]>();
    for(int i=0; i<hits.length; i++) {
        // get hit number i
        Document doc = searcher.doc(hits[i].doc);
        String title = doc.get("title");
        String url = doc.get("url");
        urlTitle.add(new String[]{title,url});
    }

    return urlTitle;
}
```

Control: Friday

Friday, 18th April

- 2 hours
- Four questions, all mandatory
  1. Distributed systems/GFS
  2. MapReduce/Hadoop
  3. PIG
  4. Spark
- One page of notes (back and front)

Class Projects

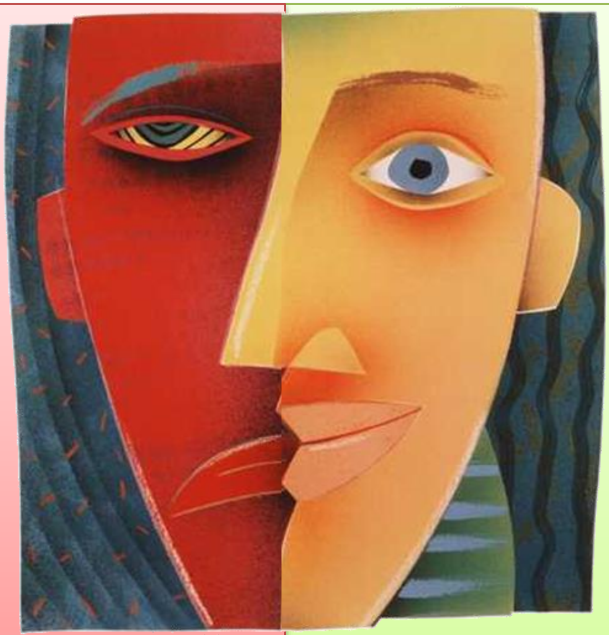
Course Marking

- 55% for Weekly Labs (~5% a lab!)
- 15% for Class Project
- 30% for 2x Controls

Assignments each week

Controls

Working in groups



Only need to pass overall!

No final exam!

Working in groups!

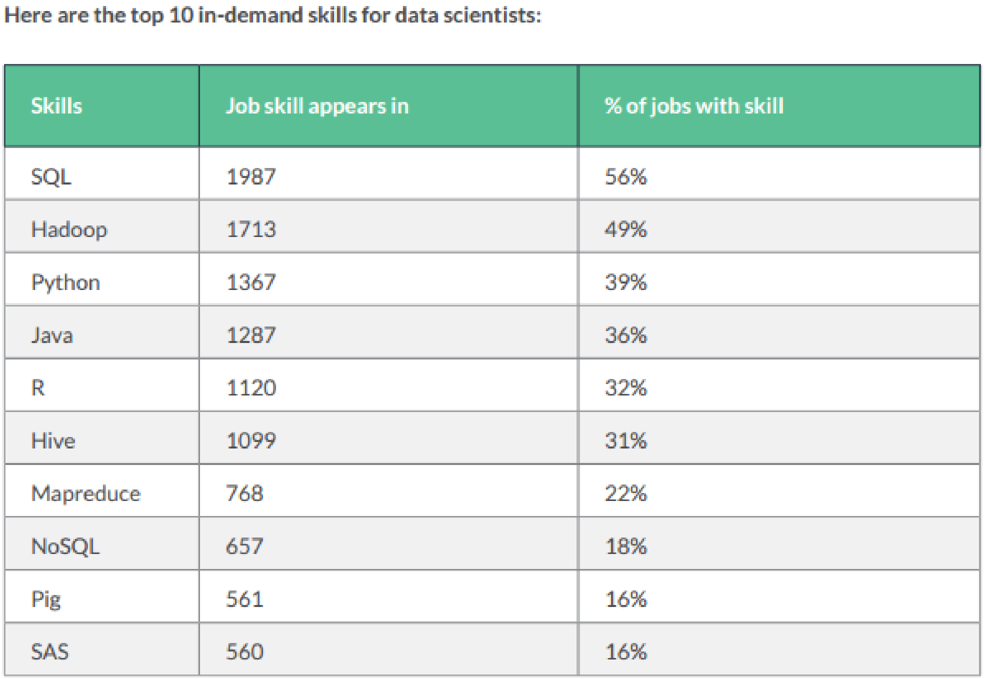
Class Project

- Done in threes
- Goal: Use what you’ve learned to do something cool/fun (hopefully)
- Expected difficulty: A bit more than a lab’s worth
  - But without guidance (can extend lab code)
- Marked on: Difficulty, appropriateness, scale, good use of techniques, presentation, coolness, creativity, value
  - Ambition is appreciated, even if you don’t succeed
- Process:
  - Start thinking up topics / find interesting datasets!
- Deliverables: 4 minute presentation & short report

Next week

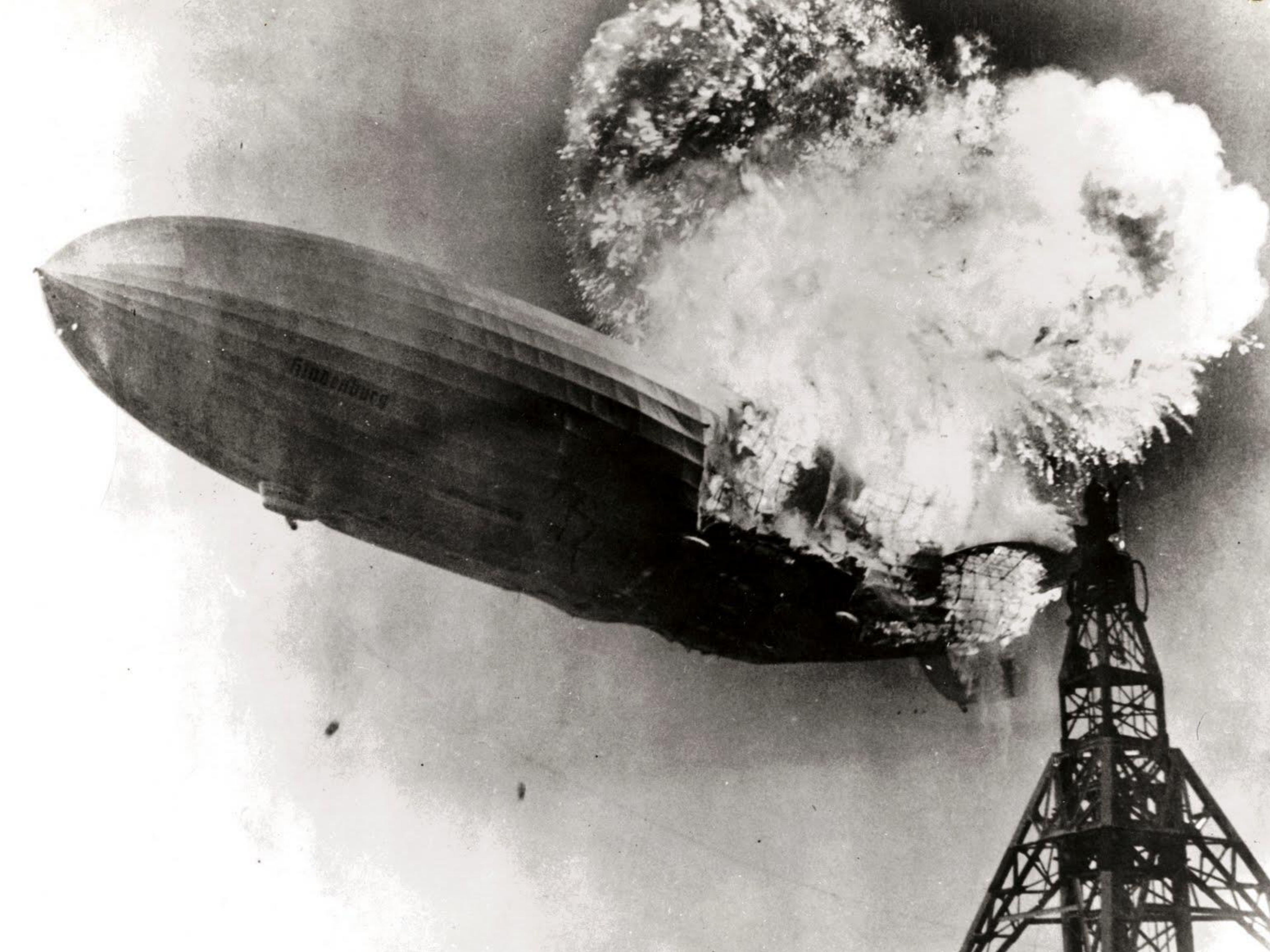
Exercise

- I will not be here next week
- Exercise (groups of two):
  - Find movies with rating greater than X, with number of votes greater than Y, where all actors are male|female
  - In MapReduce (Java), Pig, Spark!



Here are the top 10 in-demand skills for data scientists:

Skills	Job skill appears in	% of jobs with skill
SQL	1987	56%
Hadoop	1713	49%
Python	1367	39%
Java	1287	36%
R	1120	32%
Hive	1099	31%
Mapreduce	768	22%
NoSQL	657	18%
Pig	561	16%
SAS	560	16%


Hindenburg

Questions?