

CC5212-1

PROCESAMIENTO MASIVO DE DATOS

OTOÑO 2019

Lecture 7

Information Retrieval: Crawling & Indexing

Aidan Hogan

aidhog@gmail.com

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

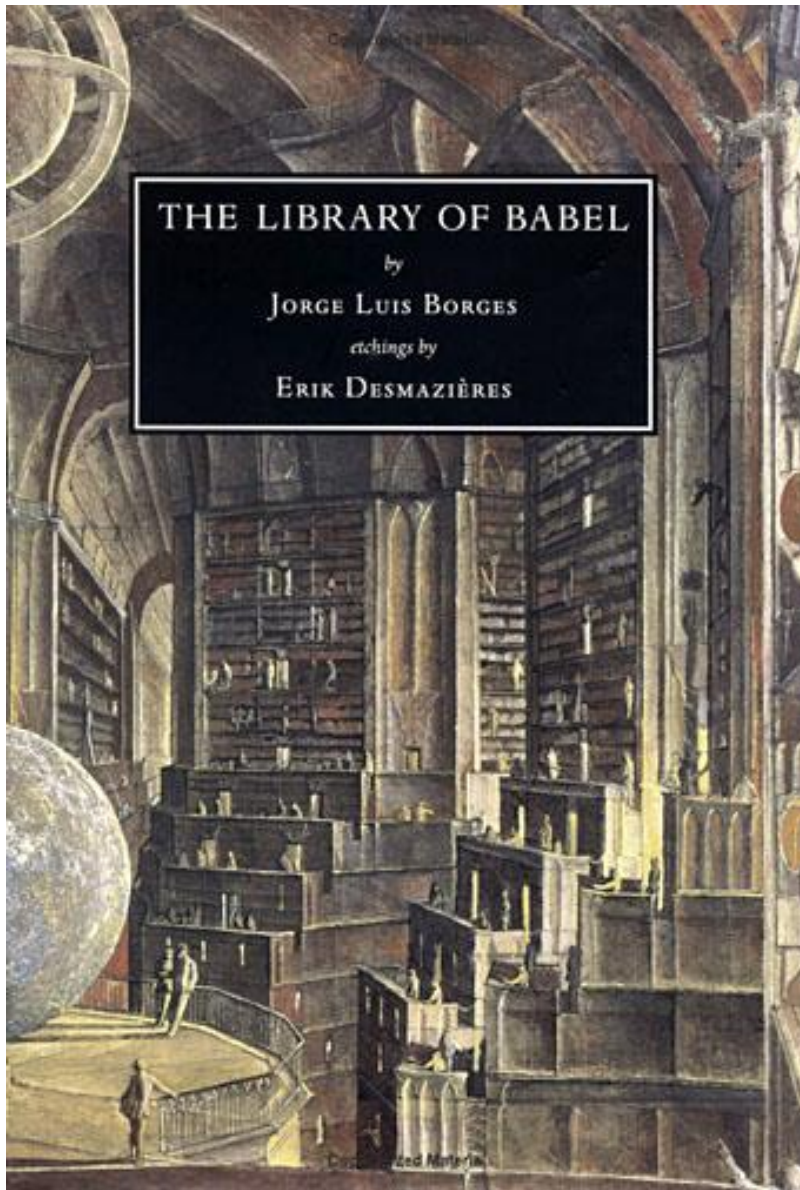


MANAGING TEXT DATA

Information Overload



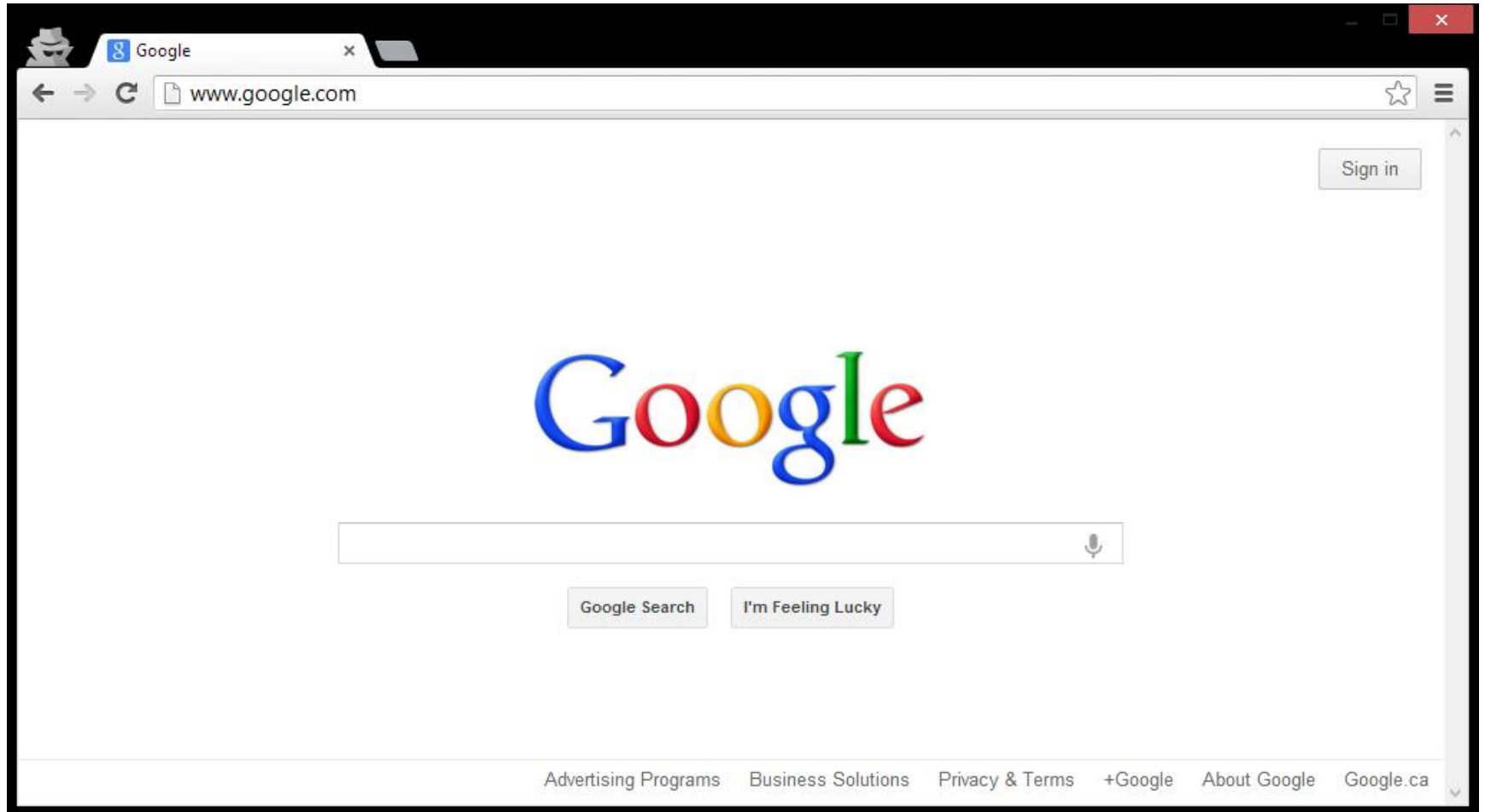
If we didn't have search ...



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

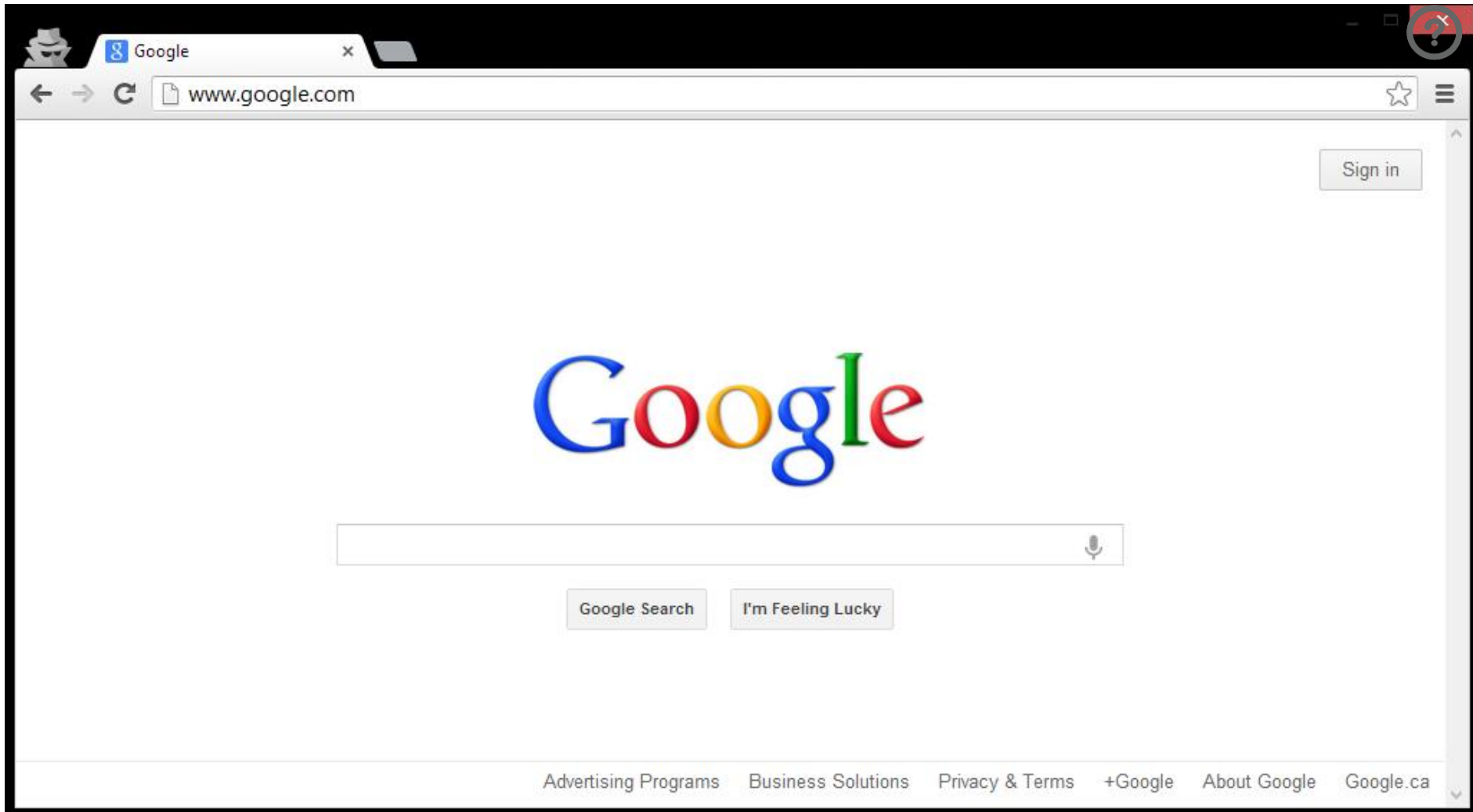
All information = Zero information

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



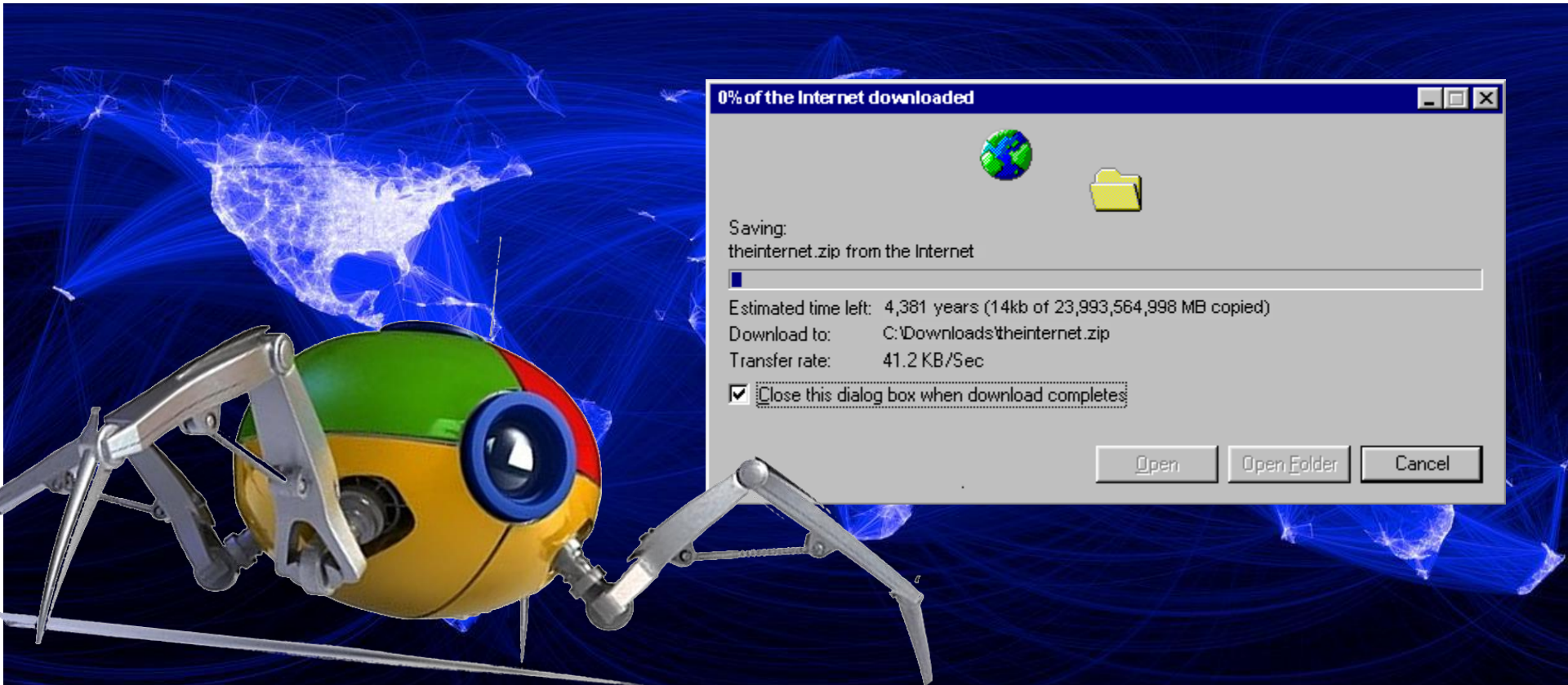
1. How relevant is a page? (TF-IDF)
2. How important is it? (PageRank)
3. How many users clicked it?

...

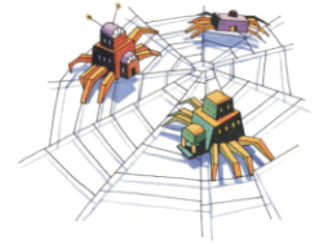


INFORMATION RETRIEVAL: CRAWLING

How does Google know about the Web?



Crawling



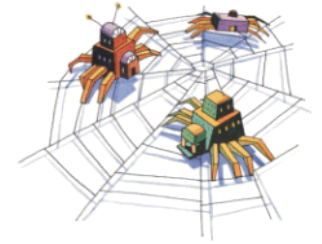
Download the Web. 😊

```
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. 😊

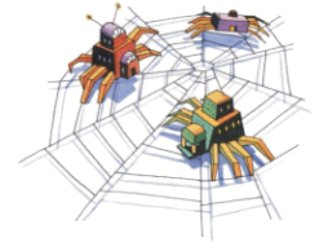
```
crawl(list seedUrls)
  frontier_i = seedUrls
  new set urlsSeen
  while(!frontier_i .isEmpty())
    new list frontier_i+1
    for url : frontier_i
      page = downloadPage(url)
      urlsSeen.add(url)
      frontier_i+1.addAll(extractUrls(page).removeAll(urlsSeen))
      store(page)

  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?



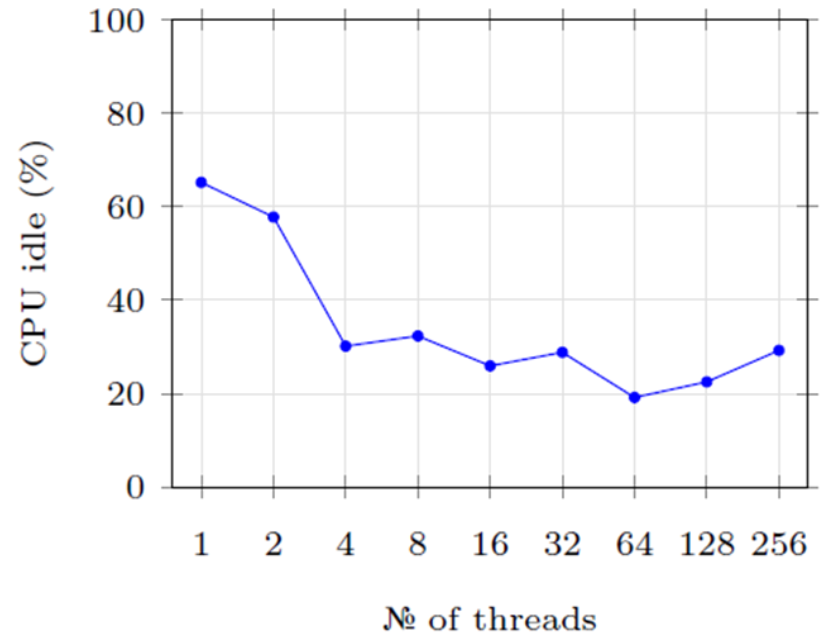
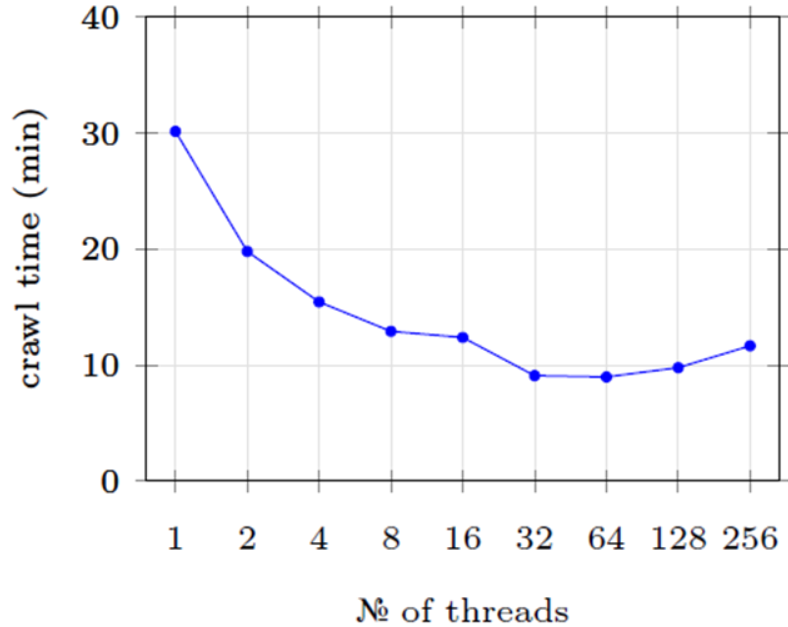
Crawling: Multi-threading Important

```
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 Channel
6667 #loic Disconnected.

1. Select your target

2. Attack options

3. Ready?

IMMA CHARGIN MAH LAZER

Selected target

NONE!

Attack status

Idle Connecting Requesting Downloading Downloaded Requested Failed

github.com/NewEraCracker/LOIC

Crawling: Avoid (D)DoSing



- Christopher Weatherhead
- 18 months prison



... 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,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)
```

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 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

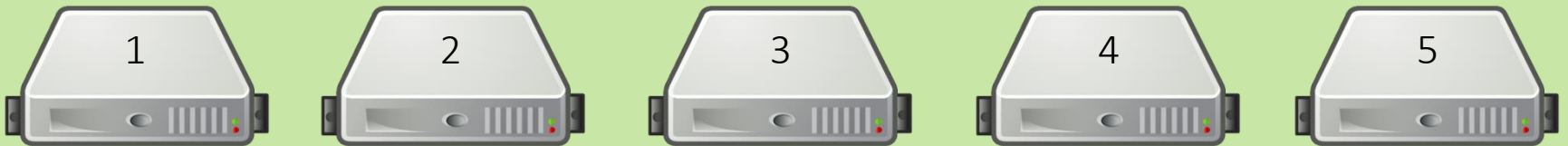
- **Seed-list**: Entry point for crawling
- **Frontier**: Extract links from current pages for next round
- **Seen-list**: Avoid cycles
- **Threading**: Keep machines busy
- **Politeness**: Don't annoy web-sites
 - Set delay between crawling pages on the same web-site
 - Stick to what's stated in the 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?

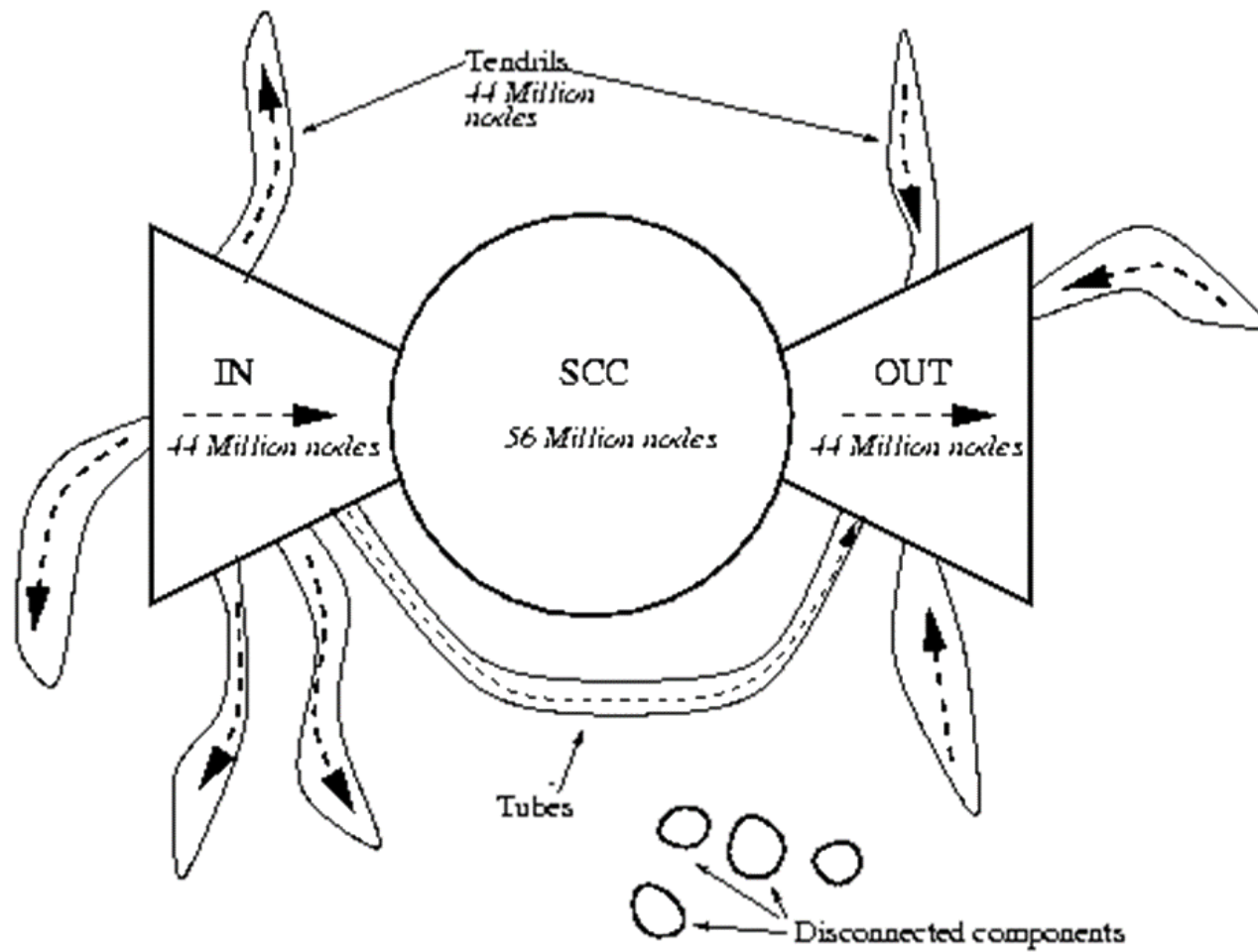
Can ~~we~~ crawl all the Web?



Can Google crawl all the Web?



Crawling: Inaccessible (Bow-Tie)



Crawling: Inaccessible (Deep Web)

What is the Deep Web?

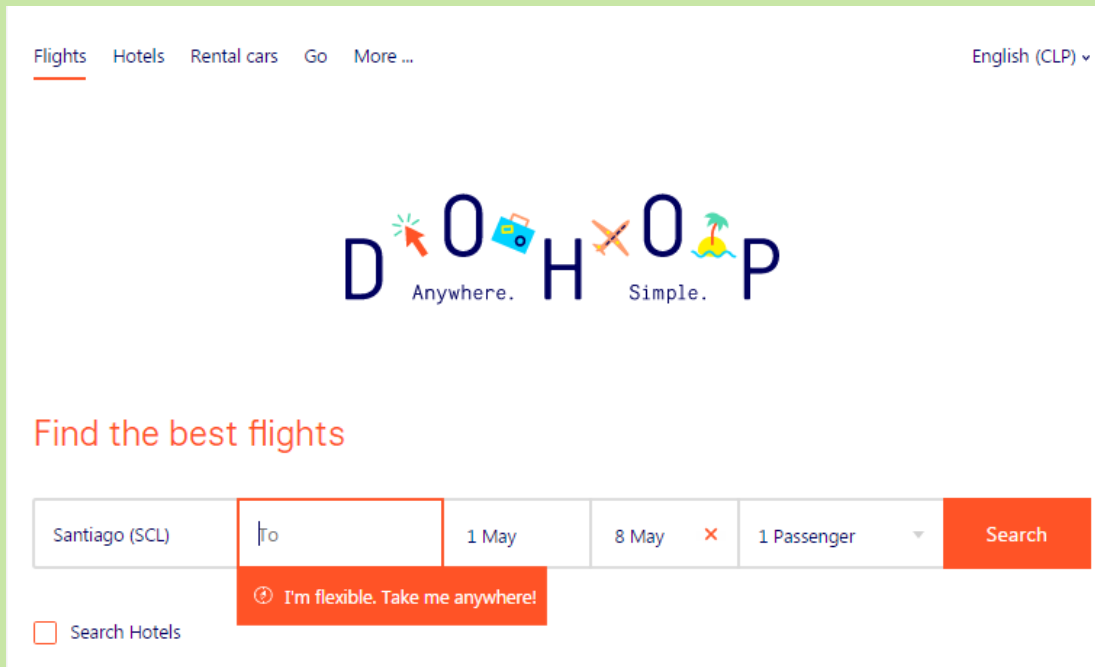


Crawling: Inaccessible (Deep Web)

What is the Deep Web?



- Dynamically-generated content



The screenshot shows a flight search interface with the following elements:

- Navigation menu: Flights (underlined), Hotels, Rental cars, Go, More ...
- Language: English (CLP) with a dropdown arrow.
- Logo: Stylized letters D, O, H, O, P with icons (arrow, suitcase, airplane, palm tree) and the text "Anywhere." and "Simple." below them.
- Section header: "Find the best flights" in orange.
- Search form: A row of input fields containing "Santiago (SCL)", "to", "1 May", "8 May" with a red 'x' icon, and "1 Passenger" with a dropdown arrow. An orange "Search" button is on the right.
- Flexibility button: An orange button with a circular icon and the text "I'm flexible. Take me anywhere!".
- Search Hotels: A checkbox labeled "Search Hotels".

Crawling: Inaccessible (Deep Web)

What is the Deep Web?



- Dynamically-generated content
- Password-protected

The screenshot shows a web interface for a university course. At the top, there are navigation links for "Flights", "Hotels", and "Rental cars". The main header includes the course name "CC5212-1 Procesamiento Masivo de Datos 2017, Otoño" and the logo for "fcfm" (Facultad de Cs. Físicas y Matemáticas). A user profile for "AIDAN HOGAN" is visible on the left, with a list of menu items: "Mi Inicio", "Mis Canales", "Mis Datos", "Todos Mis Cursos", "Mi Horario", "Mis Estrellas", and "CURSOS ACTUALES". The "CURSOS ACTUALES" section lists several courses, with "CC5212-1 Procesamiento Masivo de Datos" highlighted. Below this, there is a "Historial" section with a search filter for "Por Fecha" selected. The "Historial" entry shows a forum post from "Matilde Rivas L." with the subject "Re (3): Sobre los controles".

Find the best flight

Santiago (SCL) To

Search Hotels

Salir Contacto Buscar...

U-Cursos

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

Administrar Calendario Correo Datos del Curso Encuestas Enlaces Estadísticas Favorito Inicio

Foro Historial Horario Integrantes Material Alumnos Material Docente Notas Parciales Tareas

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

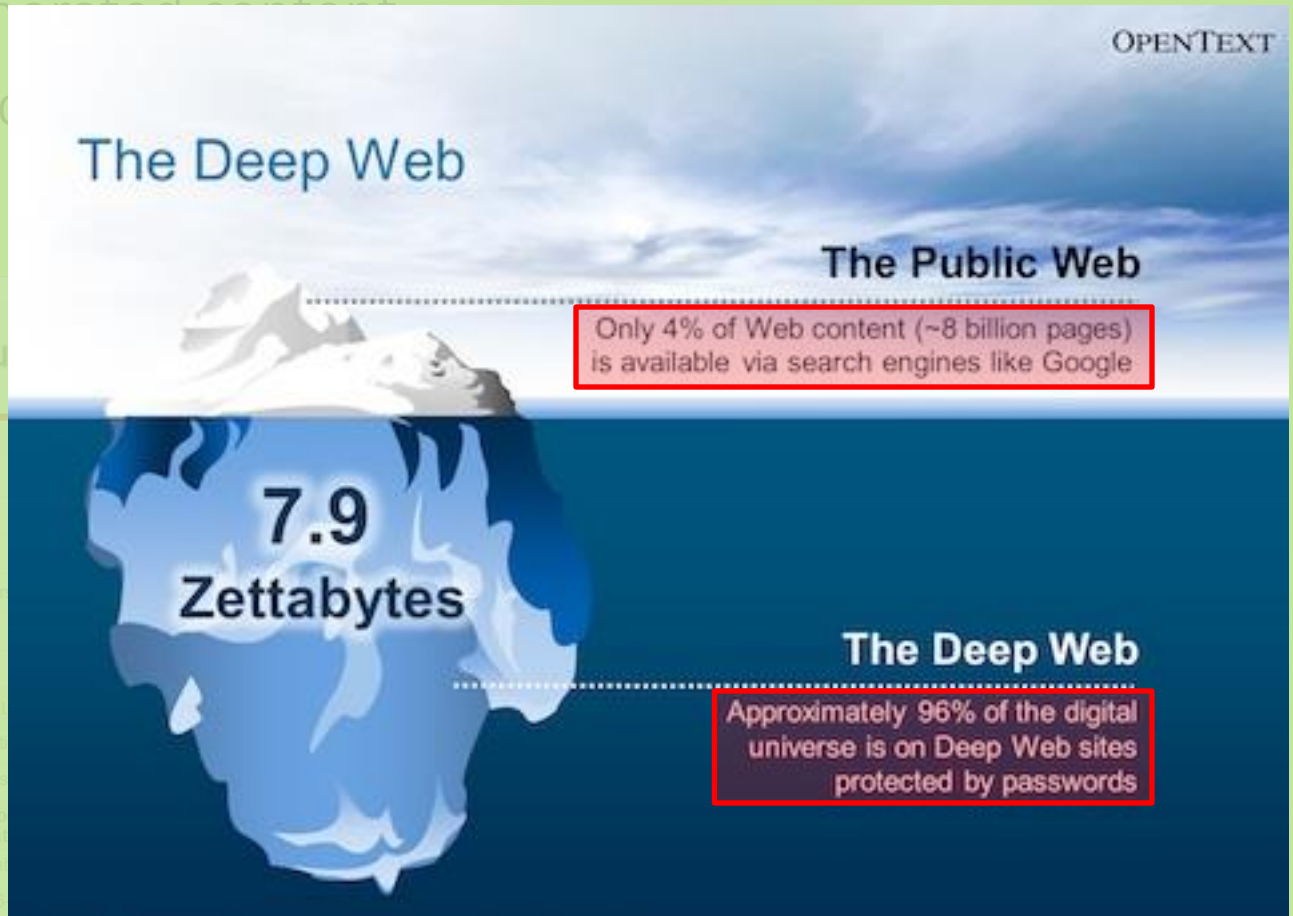
The image shows two side-by-side screenshots. The left screenshot is from a travel website with a navigation menu including 'Flights', 'Hotels', and 'Rental cars'. A search box is visible with 'Santiago (SCL)' entered. The right screenshot is from the Silk Road anonymous market, featuring a camel logo and a list of categories such as 'Drugs 7,052', 'Apparel 259', and 'Art 114'. Below the categories, there are product listings with images and prices, including '1g crack pure!!', '1 oz White Rhino', '100 Restoran 30mg (Novartis)', 'ICE / 1 POINT (0.1G)', '20x 1MG Alprazolam', and '50x MDMA / 1gr pure'.

Crawling: Inaccessible (Deep Web)

What is the Deep Web?



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



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?



Google Search

I'm Feeling Lucky

Buscar

Show all Only English Only from Chile

IMDb Find Movies, TV shows, Celebrities and more... All [Search]

Movies, TV & Showtimes Celebs, Events & Photos News & Community Watchlist

WIKIPEDIA

English <i>The Free Encyclopedia</i> 4 501 000+ articles	Español <i>La enciclopedia libre</i> 1 096 000+ artículos
日本語 フリー百科事典 906 000+ 記事	Deutsch <i>Die freie Enzyklopädie</i> 1 712 000+ Artikel
Русский <i>Свободная энциклопедия</i> 1 108 000+ статей	Français <i>L'encyclopédie libre</i> 1 499 000+ articles
Italiano <i>L'enciclopedia libera</i> 1 117 000+ voci	Português <i>A enciclopédia livre</i> 825 000+ artigos
Polski <i>Wolna encyklopedia</i> 1 042 000+ hasel	中文 自由的百科全书 764 000+ 條目

 English [Search]

Inverted Index: Example

1



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



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** }

Term List

Posting Lists

a Synonym expansion

{ **film** , **movie** } → { **movie** }

Inverted index:

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** }

Term List

Posting Lists

a Synonym expansion

{ **film** , **movie** } → { **movie** }

➤ Language specific!

➤ Use same normalisation on query and document!



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

Word-level inverted index:

Additionally maps words with positional information

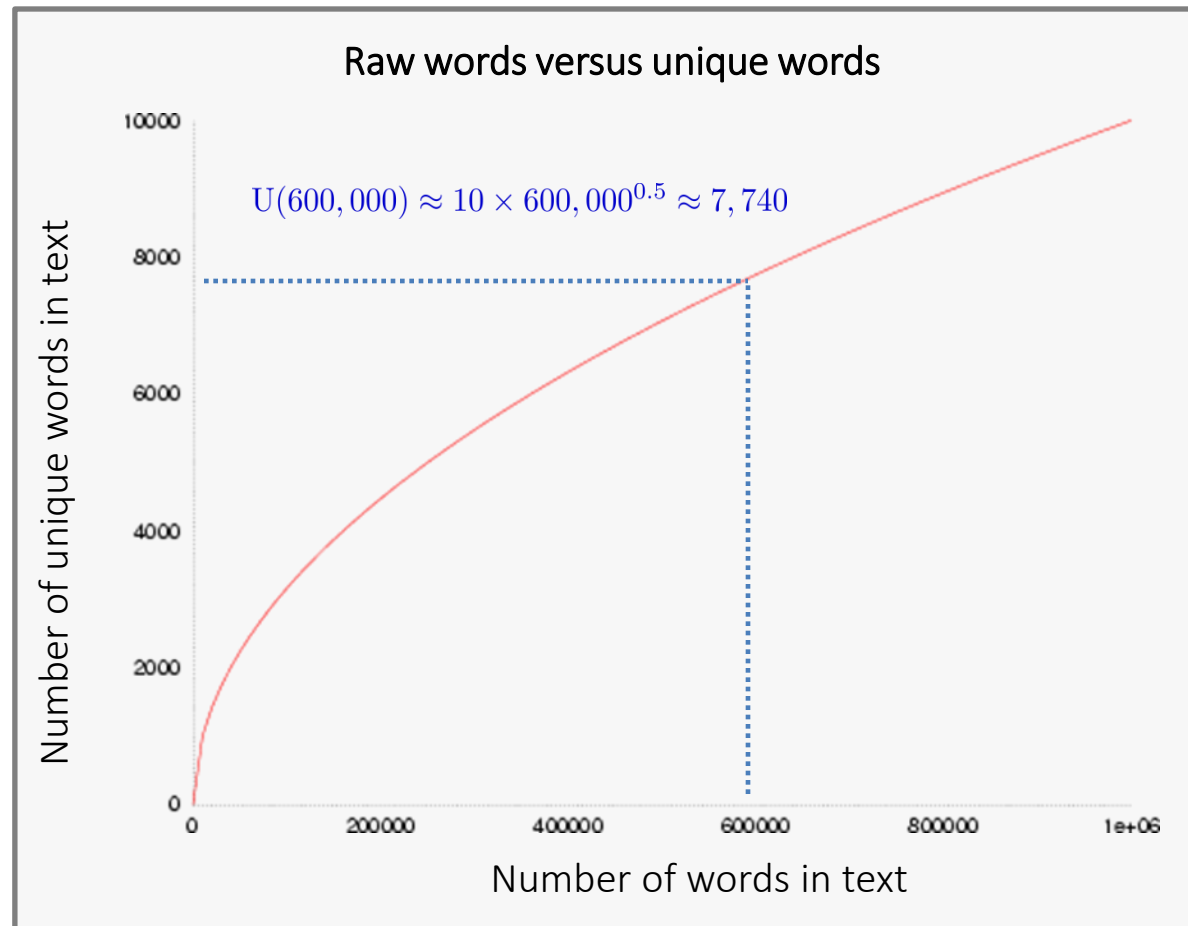
Term List	Posting List
a	(1,[21,96,103,...]), (2,[...]), ...
american	(1,[28,123]), (5,[...]), ...
and	(1,[57,139,...]), (2,[...]), ...
by	(1,[70,157,...]), (2,[...]), ...
directed	(1,[61,212,...]), (4,[...]), ...
drama	(1,[38,87,...]), (16,[...]), ...
...	...

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

Inverted Index: Unique Words

Not so many unique words ...

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



Inverted Index: Space

$$U(d) \approx K \times W(d)^\beta$$



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?



$\sum_{d \in D} U(d)$ (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?



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

Inverted Index: Common Words

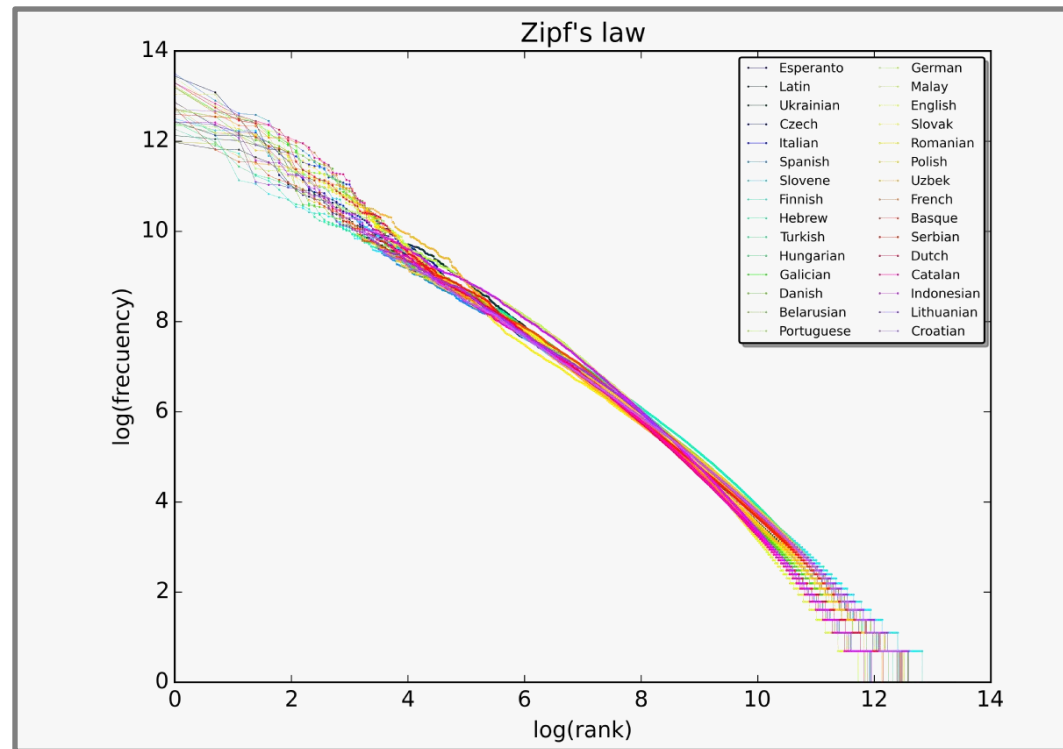
Many occurrences of few words

/ 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

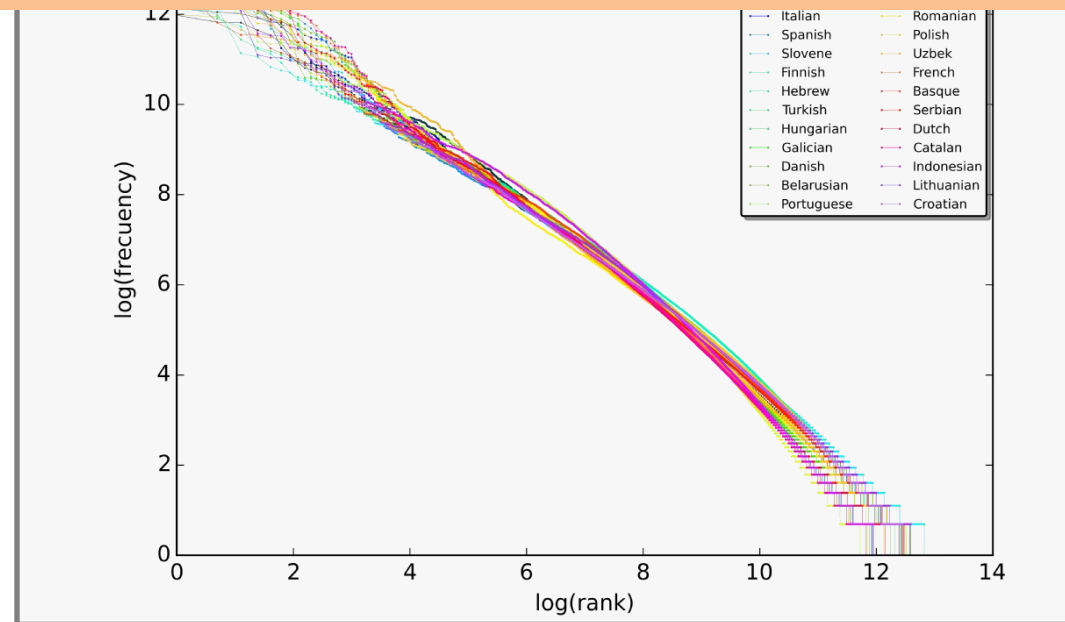
/ Few occurrences of many words

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

What is the effect on phrase search?



Small blocks ~ **okay**
Big blocks ~ **not okay**

Block 2

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

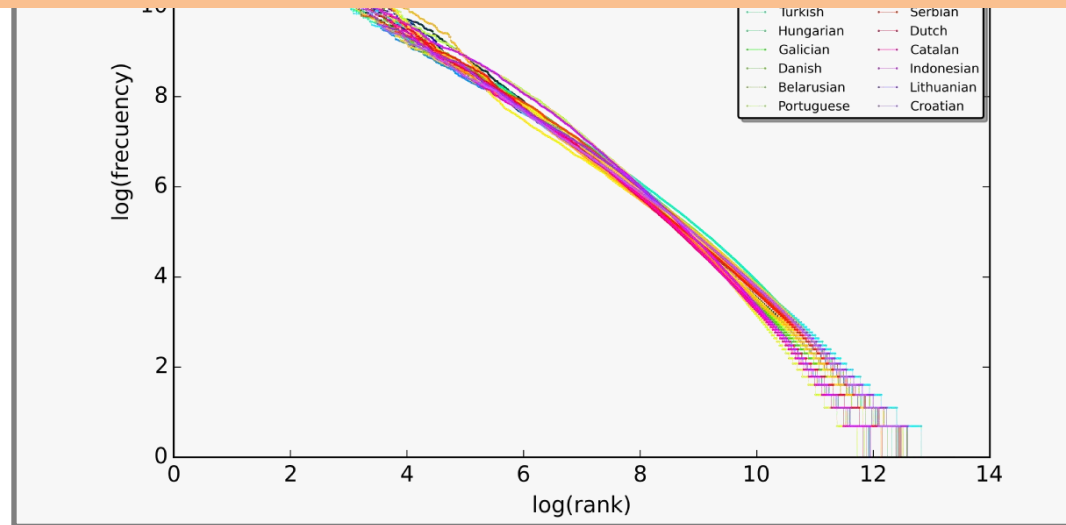
/ Few occurrences of many words

Expect **long posting lists** for common words



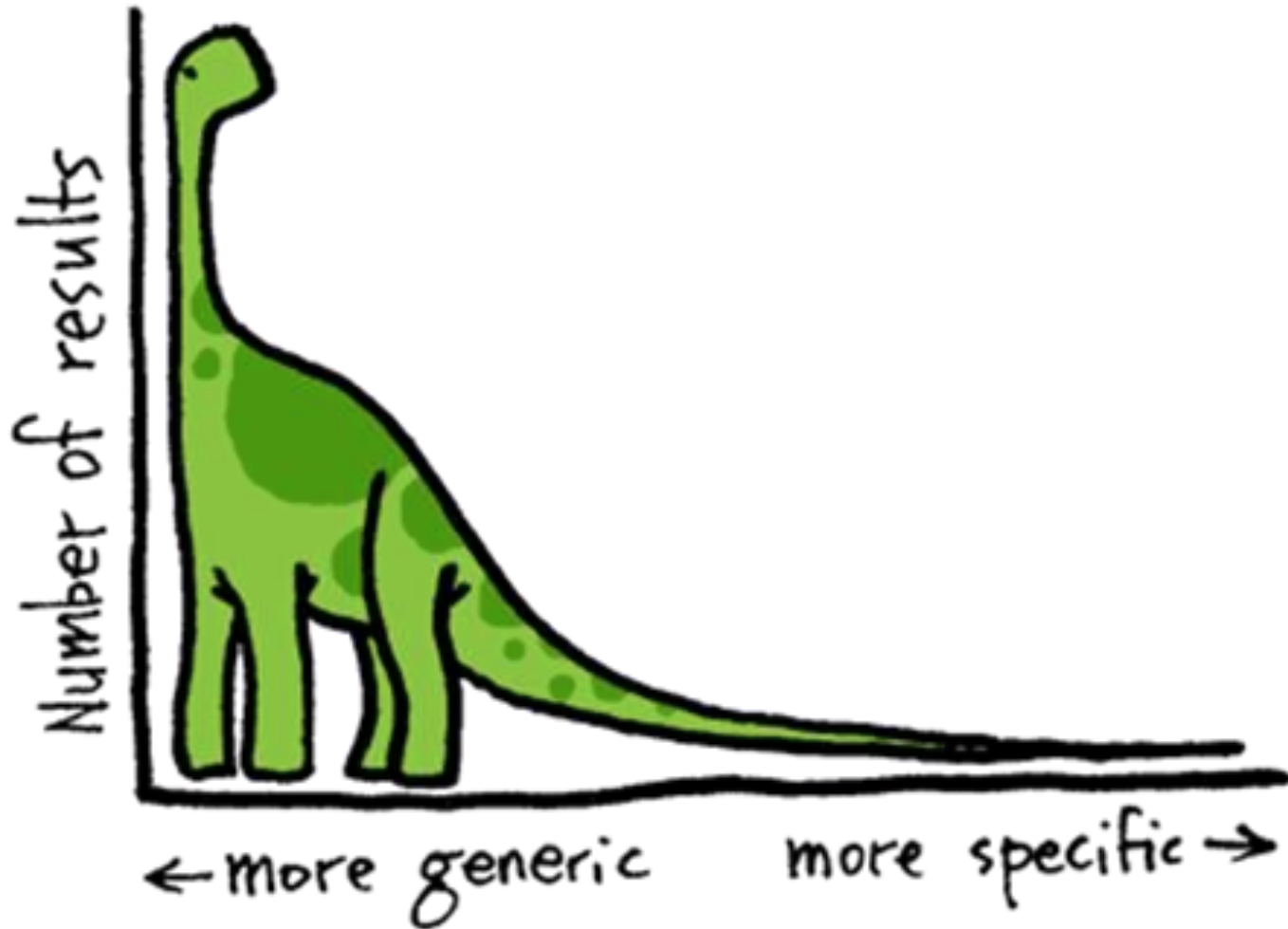
Expect **more queries** with common words

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

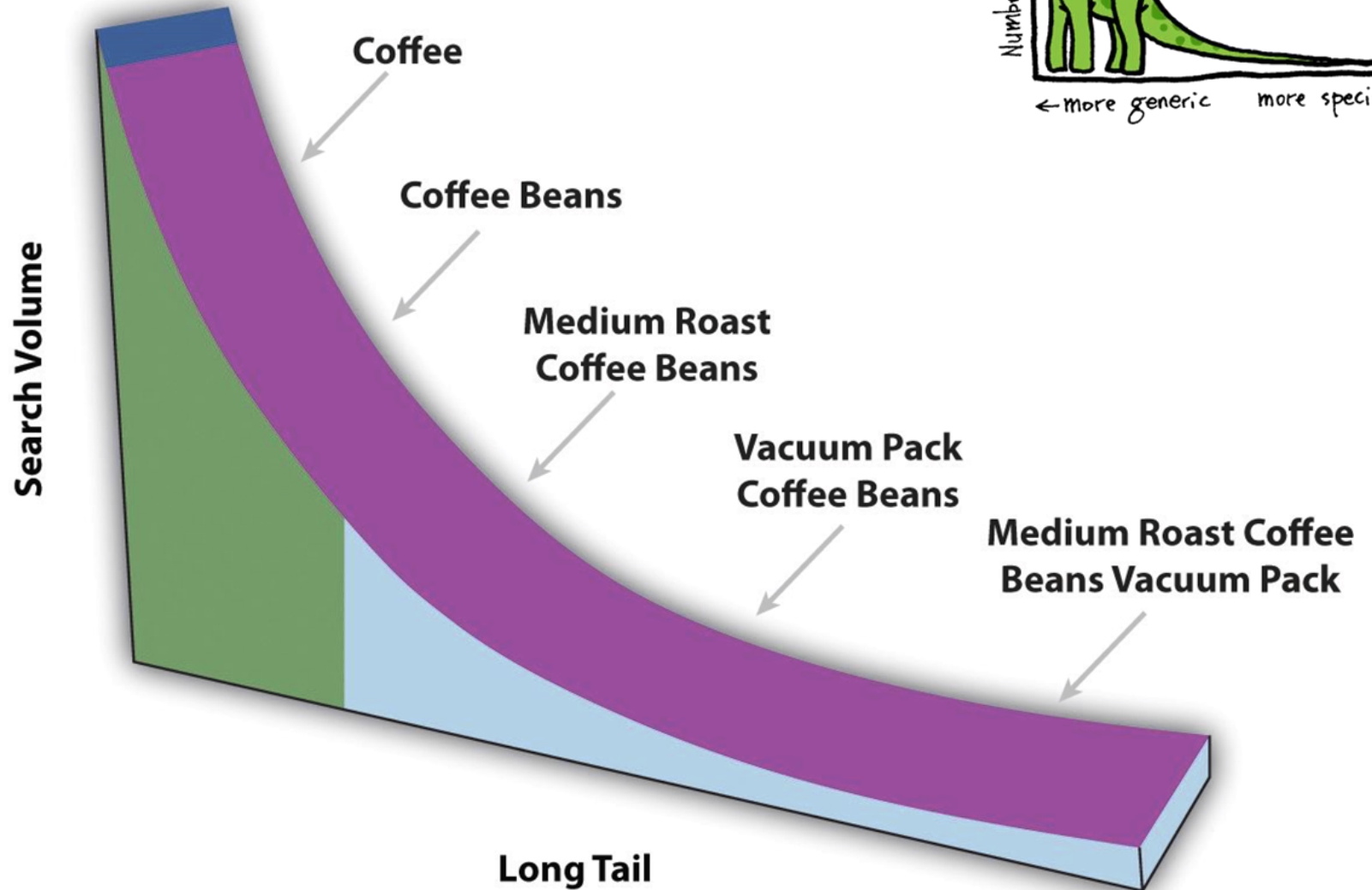


Zipf's law: *the most popular word will occur twice as often as the second most popular word, thrice as often as the third most popular word, n times as often as the n-most popular word.*

The Long Tail of Search



The Long Tail of Search

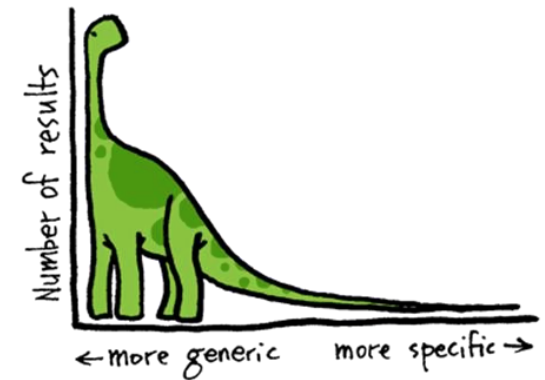


How to optimise for this?



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)



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

$2\lceil \log_2(z) \rceil + 1$ bits

z: integer to encode	$n = \lceil \log_2(z) \rceil$ 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

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

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
...

Comparison

- Small values

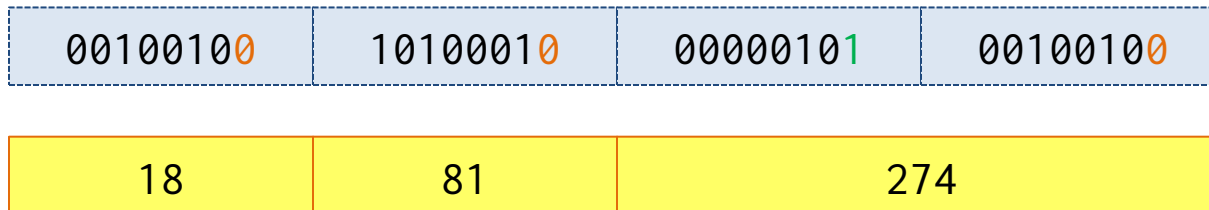
z: integer de entrada	código Elias γ	código Elias δ	código Golomb-3
1	0	0	00
2	100	1000	010
3	101	1001	011
4	11000	10100	100
5	11001	10101	1010
6	11010	10110	1011
7	11011	10111	1100
8	1110000	11000000	11010

- Larger values

z: integer de entrada	código Elias γ	código Elias δ	código Golomb-3
100	1111110100100	10110100100	1111111...101
...			...

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 😊



Google Search

I'm Feeling Lucky

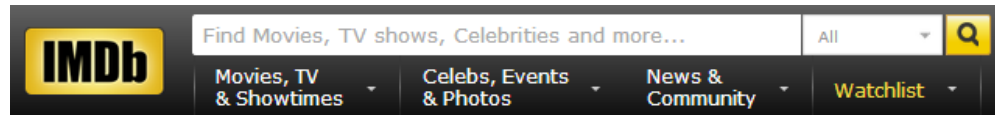
YAH!
CHILE

Buscar

bing

Beta

Show all Only English Only from Chile



IMDb Find Movies, TV shows, Celebrities and more... All

Movies, TV & Showtimes Celebs, Events & Photos News & Community Watchlist

WIKIPEDIA

English

The Free Encyclopedia
4 501 000+ articles

Español

La enciclopedia libre
1 096 000+ artículos

日本語

フリー百科事典
306 000+ 記事

Deutsch

Die freie Enzyklopädie
1 712 000+ Artikel

Русский

Свободная энциклопедия
1 108 000+ статей

Français

L'encyclopédie libre
1 499 000+ articles

Italiano

L'enciclopedia libera
1 117 000+ voci

Português

A enciclopédia livre
825 000+ artigos

Polski

Wolna encyklopedia
1 042 000+ haseł

中文

自由的百科全书
764 000+ 條目



English



LUCENE: TEXT INDEXING

Apache Lucene



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



Apache Lucene

- Inverted Index
 - Re-used in other well-known projects



Doug Cutting (above) & Mike Cafarella (below)



CLASS PROJECTS

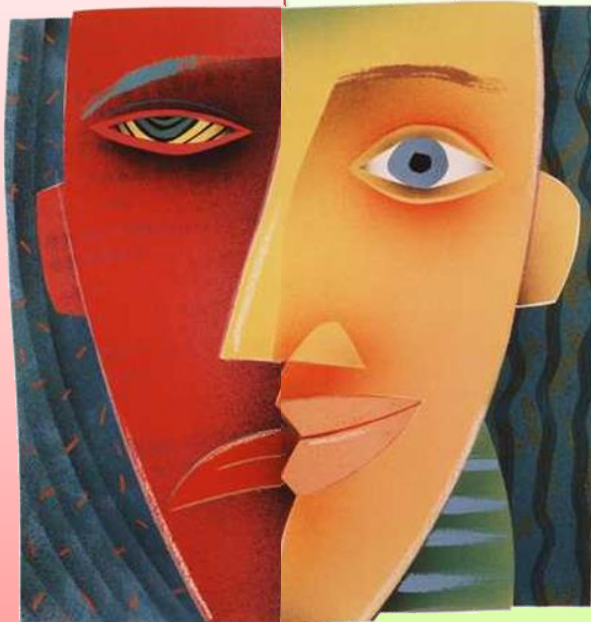
Course Marking

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

Assignments each week

Controls

Working in groups



Hands-on each week!

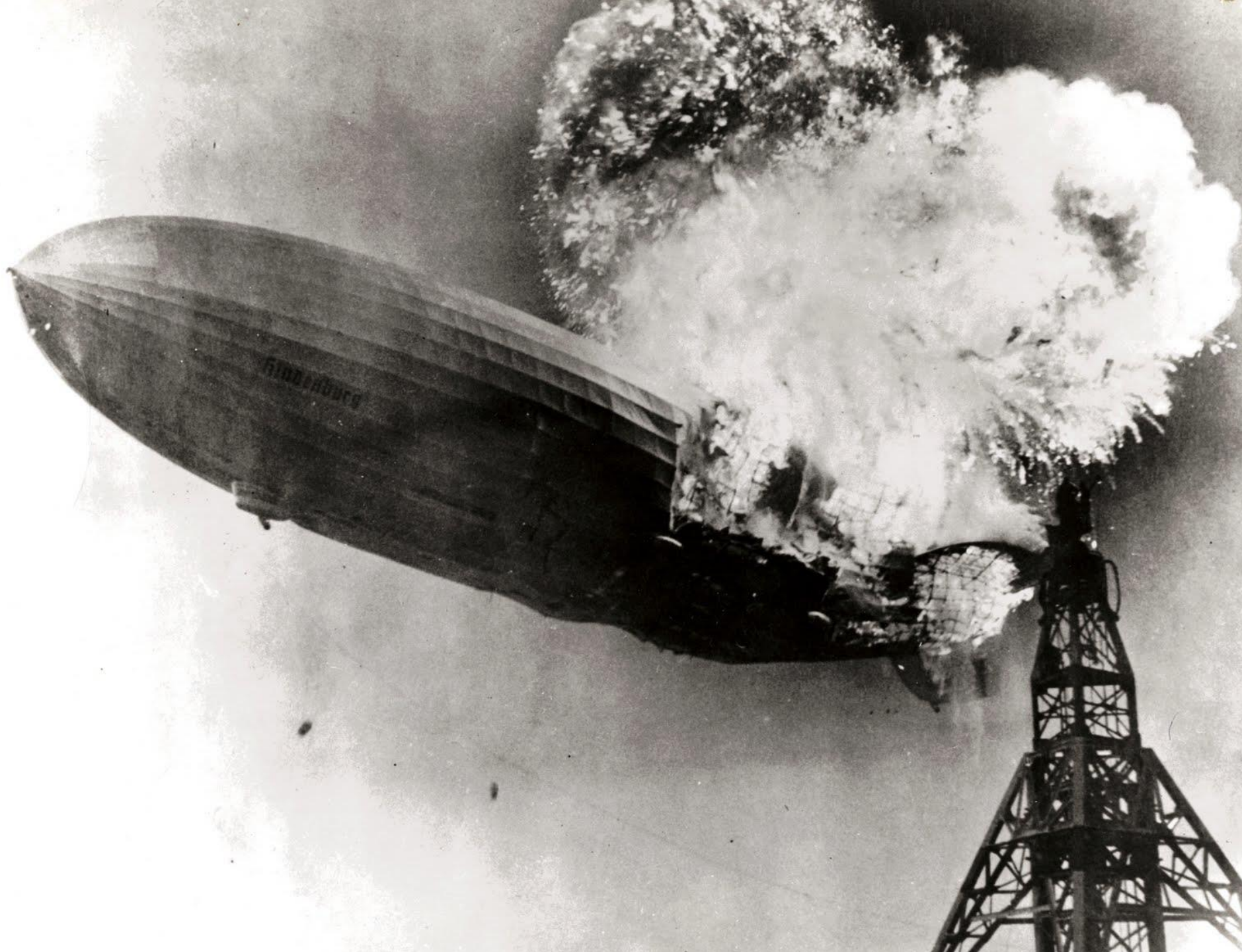
No final exam!

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





Questions?