

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Web Search

COMP90049 Knowledge Technologies

Lea Frermann and Justin Zobel and Karin Verspoor, CIS

Semester 2, 2019



THE UNIVERSITY OF

MELBOURNE

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Mid-semester Test

- **(slight) Time change!**

Friday Aug 30, 8.20-9.20am (exam start: 8.30 sharp)

- on the desk: pens, studentID

- write your student ID on the exam sheet

- may use blank pages of the exam, but **clearly indicate which Q you're answering!**

- directions (Kwong Lee Dow):

<https://maps.unimelb.edu.au/parkville/building/263>

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Assignment 1

- Another small glitch, see updated candidates.txt (no more dictionary words)

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Assignment 1

- Another small glitch, see updated candidates.txt (no more dictionary words)

Workshops

- please only attend workshops you're registered for
- if problem persists, we'll go through name lists!

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Last week(s): Information retrieval methods and metrics

- ...general view – information (type) and resource can be diverse
- how to measure relevance (similarity)?
- how to evaluate performance?
- **models** of IR

This week: Web Search

- IR specific to the web (huge!)
- IR specific to text-based information (messy!)
- How to store lots of messy information in a useful way?
- How to retrieve efficiently?
- Later on (probably) some (more) recent developments (knowledge graphs, ...)

Elements of a web search engine

Web Search

COMP90049
Knowledge
Technologies

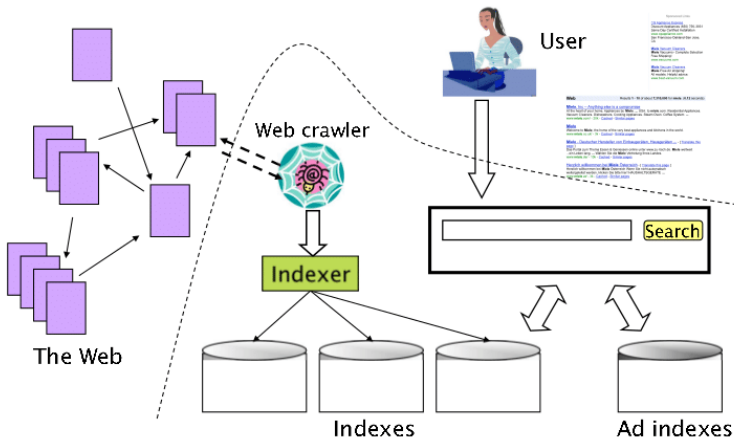
Overview Elements

Crawling Basics Challenges

Parsing Page analysis Tokenisation Stemming Zoning

Indexing Concepts Inverted indices

Querying Boolean queries Ranked querying



Manning et al. (2008) *Introduction to Information Retrieval*. Cambridge University Press (p. 434)

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Web search involves four main technological components.

Crawling: the data to be searched needs to be gathered from the web.

Parsing: the data then needs to be translated into a canonical form.

Indexing: data structures must be built to allow search to take place efficiently.

Querying: the data structures must be processed in response to queries.

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Web search involves four main technological components.

Crawling: the data to be searched needs to be gathered from the web.

Parsing: the data then needs to be translated into a canonical form.

Indexing: data structures must be built to allow search to take place efficiently.

Querying: the data structures must be processed in response to queries.

Next lecture: overview of advanced topics

- link structure (page rank)
- knowledge graphs
- advertising

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Before a document can be queried, the search engine must know that it exists.

- On the web, this is achieved by crawling.
(Web crawlers are also known as spiders, robots, and bots.)

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Before a document can be queried, the search engine must know that it exists.

- On the web, this is achieved by crawling.
(Web crawlers are also known as spiders, robots, and bots.)
- Crawlers attempt to visit every page of interest and retrieve them for processing and indexing.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Before a document can be queried, the search engine must know that it exists.

- On the web, this is achieved by crawling.
(Web crawlers are also known as spiders, robots, and bots.)
- Crawlers attempt to visit every page of interest and retrieve them for processing and indexing.
- Basic challenge: there is no central index of URLs of interest.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Before a document can be queried, the search engine must know that it exists.

- On the web, this is achieved by crawling.
(Web crawlers are also known as spiders, robots, and bots.)
- Crawlers attempt to visit every page of interest and retrieve them for processing and indexing.
- Basic challenge: there is no central index of URLs of interest.
- Secondary challenges:
 - same content as a new URL
 - never return status 'done' on access
 - websites not intended to be crawled
 - content generated on-the-fly from databases → costly for the content provider → excessive visits unwelcome
 - Some content has a short lifespan
 - Some regions and content providers have low bandwidth.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

The web is a highly linked graph → effective harvesting

Assumption: if a web page is of interest, there will be a link to it from another page.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

The web is a highly linked graph → effective harvesting

Assumption: if a web page is of interest, there will be a link to it from another page.

Corollary: given a sufficiently rich set of starting points, every interesting site on the web will be reached eventually.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

The web is a highly linked graph → effective harvesting

Assumption: if a web page is of interest, there will be a link to it from another page.

In principle:

Create a prioritised list L of URLs to visit

Create a list V of URLs that have been visited and when.

Repeat forever:

- 1 Choose a URL u from L and fetch the page $p(u)$ at location u .
- 2 Parse and index $p(u)$
Extract URLs $\{u'\}$ from $p(u)$.
- 3 Add u to V and remove it from L
Add $\{u'\} - V$ to L .
- 4 Process V to move expired or 'old' URLs to L .

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

The web is a highly linked graph → effective harvesting

Assumption: if a web page is of interest, there will be a link to it from another page.

In principle:

Create a **prioritised** list L of URLs to visit

Create a list V of URLs that have been visited and when.

Repeat forever:

- 1 Choose a URL u from L and
- 2 Parse and index $p(u)$
Extract URLs $\{u'\}$ from $p(u)$
- 3 Add u to V and remove it from L
Add $\{u'\} - V$ to L .
- 4 Process V to move expired or 'old' URLs to L .

- Every page is visited eventually.
- Synonym URLs are disregarded.
- Significant or dynamic pages are visited sufficiently frequently.
- The crawler isn't cycling indefinitely in a single web site (caught in a crawler trap).

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

The web is a highly linked graph → effective harvesting

Assumption: if a web page is of interest, there will be a link to it from another page.

In principle:

Create a **prioritised** list L of URLs to visit

Create a list V of URLs that have been visited and when.

Repeat forever:

- 1 Choose a URL u from L and fetch the page $p(u)$ at location u .
- 2 Parse and index $p(u)$
Extract URLs $\{u'\}$ from $p(u)$
- 3 Add u to V and remove it from L .
Add $\{u'\} - V$ to L .
- 4 Process V to move expired or 'old' URLs to L .

page processing is much faster than URL resolution, so numerous streams of pages should be processed simultaneously

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Crawler traps are surprisingly common. For example, a ‘next month’ link on a calendar can potentially be followed until the end of time.

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Crawler traps are surprisingly common. For example, a ‘next month’ link on a calendar can potentially be followed until the end of time.

The **Robots Exclusion Standard**: protocol that all crawlers are supposed to observe. It allows website managers to restrict access to crawlers while allowing web browsing.

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Crawler traps are surprisingly common. For example, a ‘next month’ link on a calendar can potentially be followed until the end of time.

The **Robots Exclusion Standard**: protocol that all crawlers are supposed to observe. It allows website managers to restrict access to crawlers while allowing web browsing.

Simple crawlers are now part of programming languages, for example Perl’s `LibWWW`, and good crawlers are available as part of systems such as `Nutch`.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Once a document has been fetched, it must be **parsed**.

That is, the ~~words~~ tokens in the document are **extracted**, then added to a data structure that records which documents contain which ~~words~~ tokens.

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Once a document has been fetched, it must be **parsed**.

That is, the ~~words~~ tokens in the document are **extracted**, then added to a data structure that records which documents contain which ~~words~~ tokens.

At the same time, information such as **links and anchors** can be analysed, **formats** such as PDF or Postscript or Word can be translated, the **language** of the documents can be identified, and so on.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Once a document has been fetched, it must be **parsed**.

First step: determining the format of the page.

The most basic element is the **character encoding**, which has to be captured in the page's metadata.

- For the first decade or so of the web, most pages were in ASCII.
(Want to travel in time? Try the **Wayback Machine**.
http://web.archive.org/web/19970501*/https://www.unimelb.edu.au/)
- HTML markup was used to provide an extended character set.
- ISO-8859 and ISO-8859-* now provide extended Latin character sets (Cyrillic, Thai, Greek, ...)
- UTF-8 is the dominant character set covering the large-alphabet languages, with codes from 8 to 32 bits. The first 128 of the 8-bit codes are ASCII.

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Web pages are supposed to be in HTML or XML (or sometimes in other formats, hence `ftp://` and so on).

The format separates user-visible content from metadata.

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Web pages are supposed to be in HTML or XML (or sometimes in other formats, hence `ftp://` and so on).

The format separates user-visible content from metadata.

Many, many websites are not.

- accidental errors
- deliberate attempt to take advantage of known browser behaviour

(improved slightly due to the prevalence of Web publishing software; this also produces non-conformant HTML surprisingly often.)

Parsers therefore need to be robust and flexible.

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Scraping

- only some components of the page are considered
- e.g., ignore ads or comments (on a news website)
- only article content is retained

Invisible content

- e.g., white text on white background, non-displayed HTML content
- designers actively seek to avoid indexing invisible content
- misleading for users
- allows spoofing

HotAIR - Rare and well-dane tidbits from the Annals of Improbable Research - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.improb.com/

☐ Customize Links ☐ Free Hotmail ☐ Windows Media ☐ Windows

[About AIR](#) /
 [Subscribe](#) /
 [Our blog](#) /
 [Events calendar](#) /
 [Contact us](#) /
 [Search](#)



Annals of IMPROBABLE RESEARCH

NOTE THIS: JoAnn O'Linger-Luscusk and Alasdair Skelton have [joined](#) the Hair Club

Our blog: [Something New and Improbable](#), every day M-F



Download a sample issue!

You are Arkham City, January 1, 2015

[Le Nobel Prizes](#) (the 2005 winners, and video of the ceremony)

Our annual awards for achievements that make people laugh, then think

[AIRchive:](#)

- [Magazine \(AIR\) / Newsletter \(news-AIR\)](#)
- [Newspaper Column / Blog](#)
- [Classics / Press Clips](#)

[Projects & Surveys](#)

- [Hair Club / Bureaucracy Club / Broken News / Teachers / Universal History / Feline Reactions](#) and more

[Improbable Research Shows](#)

Schedules and more

[ShaveAIR](#)

Other improbable websites (submit yours!)

[Bookstore](#)

Improbable books and whatnot

This entire web page, including all files under the domains www.improbable.com and www.improb.com, is copyright The Annals of Improbable Research. HotAIR is made possible through the admirable assistance of the folks at [Cybercon](#).

Flowers, Flower Arrangements, & Flower Delivery	Powered by Cyber Flowers
Ace Avenue Florists Adriana's Florists Aloha Aloha Florists Alto Florists Annapolis Florists Antique Florists Arizona Florists Atlanta Florists Austin Florists Baltimore Florists Bay Area Florists Beach Florists Bellevue Florists Berkeley Florists Boston Florists Butte Florists Cape Fear Florists Carroll Florists Charlotte Florists Chicago Florists Dallas Florists Denver Florists Detroit Florists Florida Florists Fort Worth Florists Fresno Florists Gainesville Florists Hawaii Florists Houston Florists Idaho Florists Illinois Florists Indiana Florists Iowa Florists Kansas Florists Kentucky Florists Louisiana Florists Maine Florists Maryland Florists Massachusetts Florists Michigan Florists Minnesota Florists Mississippi Florists Missouri Florists Montana Florists Nebraska Florists Nevada Florists New Hampshire Florists New Jersey Florists New Mexico Florists New York Florists North Carolina Florists North Dakota Florists Ohio Florists Oklahoma Florists Oregon Florists Pennsylvania Florists Rhode Island Florists South Carolina Florists South Dakota Florists Tennessee Florists Texas Florists Utah Florists Vermont Florists Virginia Florists Washington Florists West Virginia Florists Wisconsin Florists Wyoming Florists	Ace Avenue Florists Adriana's Florists Aloha Aloha Florists Alto Florists Annapolis Florists Antique Florists Arizona Florists Atlanta Florists Austin Florists Baltimore Florists Bay Area Florists Beach Florists Bellevue Florists Berkeley Florists Boston Florists Butte Florists Cape Fear Florists Carroll Florists Charlotte Florists Chicago Florists Dallas Florists Denver Florists Detroit Florists Florida Florists Fort Worth Florists Fresno Florists Gainesville Florists Hawaii Florists Houston Florists Idaho Florists Illinois Florists Indiana Florists Iowa Florists Kansas Florists Kentucky Florists Louisiana Florists Maine Florists Maryland Florists Massachusetts Florists Michigan Florists Minnesota Florists Mississippi Florists Missouri Florists Montana Florists Nebraska Florists Nevada Florists New Hampshire Florists New Jersey Florists New Mexico Florists New York Florists North Carolina Florists North Dakota Florists Ohio Florists Oklahoma Florists Oregon Florists Pennsylvania Florists Rhode Island Florists South Carolina Florists South Dakota Florists Tennessee Florists Texas Florists Utah Florists Vermont Florists Virginia Florists Washington Florists West Virginia Florists Wisconsin Florists Wyoming Florists

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying



Improbable Research - Mozilla Firefox

Improbable Research x +

https://www.improbable.com

Melbourne Forecast COMP90049 Calendar Email Files - OneDrive Thems polleverywhere yoga Get Ready to Work - H... electronics skeds Tim's Calendar Amazon Research Aw...

IMPROBABLE RESEARCH Research that makes people LAUGH and then THINK

Subscribe Google Custom Search Search

Home (Improbable Blog)

What is Improbable Research?

Ig Nobel Prizes
2019 Ceremony
About the Igs
The Winners
247
Archive

Publications
Magazine (Annals of Improbable Research)
Newsletter (mini-AIR)
Classics

Podcast & videos
Podcast
Improbable TV

Events Schedule
Upcoming Events
Past Events

Press Clips
Luxurious Hair Clubs for Scientists
Store
Info / Contact Us

Wine bottle bottom dimple enigma
August 25th, 2019

Why do wine bottles (usually) have a raised-up internal dome-like structure [its technical name is a "punt"] at the bottom of the bottle?

If you have a theory about such things, yours can be compared to a substantial list of possible explanations – bearing in mind that experts in the field are as yet undecided as to which one(s) might be correct. They include (but are not limited to) :

- Having the function of making the bottle less likely to topple over.
- Allowing bottles to be more easily stacked end to end.
- Taking up some volume of the bottle, causing it to appear larger for the same amount of wine.

See: Wikipedia

Photo Credit: Aurelien Mole @ Wikipedia

posted by **Martin Gardiner** in [Arts and Science](#) | [Comments Off](#)

Boom! "Science that's hard to take seriously and even harder to ignore"
August 23rd, 2019

Scott Lafee's [Wellness](#) syndicated column often presents tidbits about things that have won Ig Nobel Prizes. Here's [the most recent](#) (August 19, 2019):

Ig Nobel Appraised
The Ig Nobel Prizes celebrate achievements that make people laugh and

Sign up for mini-AIR, our monthly e-mail newsletter. It's free. Just type in your e-mail address...

...and press **HERE**

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

```
<head>
<META NAME="keywords" CONTENT="science humor, science humour, science,
humor, humour, ig-nobel, ig nobel, ignobel, hotair, hot-air, hot air,
improbable research">
<META HTTP-EQUIV="expires" CONTENT="0">
<title>HotAIR - Rare and well-done tidbits from the Annals of
Improbable Research</title>
</head>

<a href="/navstrip/about.html">About AIR</a>
| <a href="/navstrip/subscribe.html"><font color="red">Subscribe</font></a>
| <a href="http://improbable.typepad.com/">Our blog </a>
| <a href="/navstrip/schedule.shtml">Events calendar</a>
| <a href="/navstrip/contact.html">Contact us</a>
| <a href="/navstrip/google-search.html">Search</a>
<hr>



<tr>
<td colspan=2 align="center">
<b><br><b>NOTE THIS:  JoAnn O'Linger-Luscusk and Alasdair
Skelton have <a href="/projects/hair/hair-club-top.html#newest">joined</a>
the Hair Club</b>
</td> </tr>
```

hotair rare and well done tidbits from the annals of improbable research
note this joann o linger luscusk and alasdair skelton have joined the hair club

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis

Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

reduce a web page, or a query, to a sequence of tokens.

- ideally: query tokens will match parsed website tokens → query evaluation without approximate matching.

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

reduce a web page, or a query, to a sequence of tokens.

- ideally: query tokens will match parsed website tokens → query evaluation without approximate matching.
- Websites typically have reasonably well-formed sentences; should make parsing straightforward. **But** (in English):
 - **Hyphenation**. Is 'Miller-Zadek' one word or two? Is 'under-coating'? 'Re-initialize'? 'Under-standing'?
 - **Compounding**. Is 'football' one word or two? 'Footballgame'?
 - **Possessives**. Is 'Zadek's' meant to be 'Zadek' or 'Zadeks'? 'Smiths'?
 - Other languages have different issues.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

reduce a web page, or a query, to a sequence of tokens.

- ideally: query tokens will match parsed website tokens → query evaluation without approximate matching.
- Websites typically have reasonably well-formed sentences; should make parsing straightforward. **But** (in English):
 - **Hyphenation.** Is 'Miller-Zadek' one word or two? Is 'under-coating'? 'Re-initialize'? 'Under-standing'?
 - **Compounding.** Is 'football' one word or two? 'Footballgame'?
 - **Possessives.** Is 'Zadek's' meant to be 'Zadek' or 'Zadeks'? 'Smiths'?
 - Other languages have different issues.
- **ambiguities:** 'listen to the wind' vs. 'wind up the clock'; 'apple (fruit)' vs. 'apple (computer)'.
- particularly difficult in queries (context!)

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis

Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Indexing process that relies on fact extraction needs information in a **canonical form**

Tokenisation (usually) entails canonicalising the underlying **wordform**.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Indexing process that relies on fact extraction needs information in a **canonical form**

But **other information** may also need to be canonicalised, including:

- Dates. Consider 5/4/2011, 4/5/2011, April 5 2011, first Tuesday in April 2001.
- Numbers. 18.230,47 versus 18,230.47. Or 18 million versus 18,000,000.
- Variant spelling. Color versus colour.
- Variant usage. Dr versus Doctor. (What is the top match for Dr Who under Google?)
- Variant punctuation. 'e.g.' versus 'eg'.
- removing stop words ('content-free' terms such as the, or, and so on) (controversial!)
- delete 'weird tokens' from queries (e.g., > 64 characters)

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Arguably the most significant form of canonicalisation (for English words)

inexpensive → in+expense+ive

- attempt to undo the processes that lead to word formation
- no guarantee that the resulting stem looks like a “word”
- words in English are derived from a root or stem
- challenge: every word has a different set of legal suffixes
- result does not necessarily look like a proper ‘word’ (cf., **lemmatization**)

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics
Challenges

Parsing

Page analysis
Tokenisation

Stemming

Zoning

Indexing

Concepts
Inverted indices

Querying

Boolean queries
Ranked querying

Arguably the most significant form of canonicalisation (for English words)

inexpensive → in+expense+ive

- attempt to undo the processes that lead to word formation
- no guarantee that the resulting stem looks like a “word”
- words in English are derived from a root or **stem**
- challenge: every word has a different set of legal suffixes
- result does not necessarily look like a proper ‘word’ (cf., **lemmatization**)

suffix stripping (Porter Stemmer)

- sses → ss
- ies → i
- tional → tion
- tion → t

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

NLTK Stemming Demo!

<https://text-processing.com/demo/stem/>

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Web documents can usually be segmented into discrete zones such as **title**, **anchor text**, **headings**, and so on.

Parsers also consider issues such as **font size**, to determine which text is most prominent on the page and thus generate further zones.

Web search engines typically calculate **weights** for each of these zones, and compute similarities for documents by combining these results on the fly.

Web search engines tend to favour pages with the **query terms in titles**.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Index

- for fast query evaluation
- data structure that maps terms to the documents that contain them
- e.g., index of a book maps term → page numbers
- allows to restrict processing to documents which contain query term(s)
- many different types of index have been suggested

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

The only practical index structure for text query evaluation is the **inverted index**

- collection of lists (one per term)
- each containing IDs of documents containing that term
- aka “term-document matrix” (but more efficient representation)

(There’s nothing fundamentally different between a ‘forward’ and an ‘inverted’ index.)

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Conceptually,

- we want a structure which can process query terms
- find document(s) that contain term
- access the resulting documents

We require three elements

- the search structure (vocabulary)
- an inverted list for each term in the vocabulary
- a mapping table of document identifiers to documents

(By convention, the “inverted index” refers to just the inverted lists.)

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Search structure

For each distinct word t , the search structure contains:

- A pointer to the start of the corresponding inverted list.
- A count f_t of the documents containing t .

That is, the search structure contains the vocabulary.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Inverted lists

For each distinct word t , the inverted list contains:

- The identifiers d of documents containing t , as ordinal numbers.
- The associated frequency $f_{d,t}$ of t in d .
(We could instead store $w_{d,t}$ or $w_{d,t}/W_d$.)

Example inverted index

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

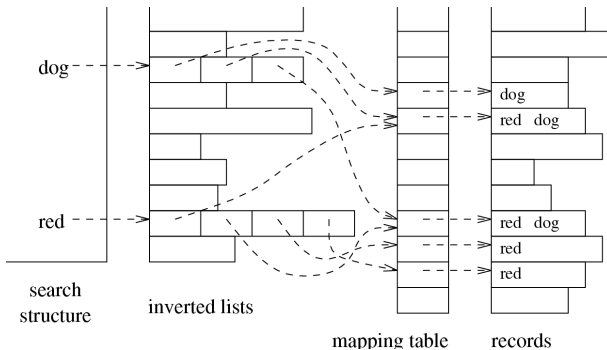
Inverted indices

Querying

Boolean queries

Ranked querying

Together with an array of W_d values (stored separately), the search structure and inverted index provide all the information required for Boolean and ranked query evaluation.



Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

For example:

We few, we happy few, we band of brothers

$\langle a, \text{aardvark}, \dots, \text{band}, \dots, \text{brothers}, \dots, \text{few}, \dots, \text{happy}, \dots \rangle$

$\langle 0, 0, \dots, 1, \dots, 1, \dots, 2, \dots, 1, \dots \rangle$

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Inverted index (one document):

band	→	(1,1)
brothers	→	(1,1)
few	→	(1,2)
happy	→	(1,1)
of	→	(1,1)
we	→	(1,3)

Example inverted index

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Inverted index (multiple documents):

...					
band	→ ...	→	$(d, f_{d,\text{band}})$	→	...
...					
brothers	→ ...	→	$(d, f_{d,\text{brothers}})$	→	...
...					
few	→ ...	→	$(d, f_{d,\text{few}})$	→	...
...					
happy	→ ...	→	$(d, f_{d,\text{happy}})$	→	...
...					
of	→ ...	→	$(d, f_{d,\text{of}})$	→	...
...					
we	→ ...	→	$(d, f_{d,\text{we}})$	→	...
...					

Example inverted index

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

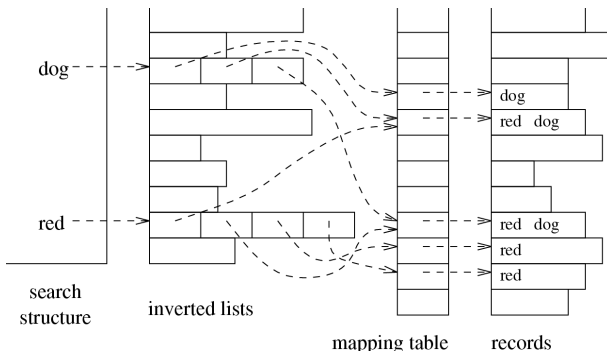
Querying

Boolean queries

Ranked querying

An inverted index allows for fast querying because:

- (1) the terms in the query correspond to the search structure
- (2) the index only indicates documents where the term is present



Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

In a simple representation, for (say) a gigabyte of newswire data,

- 12 MB (say) for 400,000 words, pointers, counts.
- 280 MB for 70,000,000 document identifiers (4 bytes each).
- 140 MB for 70,000,000 document frequencies (2 bytes each).

The total size is 432 MB, or just over 40% of the original data.

For 100 GB of web data, the total size is about 21 GB, or just over 20% of the original text. (Many web pages contain large volumes of unindexed data such as markup.)

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Using a term-document matrix (TDM)

- TDM is compact to store (1b per term per document)
- bitwise comparisons are fast
- desirable for modest document collections
- memory issue: consider hundreds of millions of documents
- space waste: most values in the matrix are 0 → never used in any comparison

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Using an inverted index

- Fetch the inverted list for each query term.
- Use intersection of lists to resolve AND.
- Use union of lists to resolve OR.
- Take the complement of a list to resolve NOT (how?).
- Ignore within-document frequencies.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Using an inverted index

- Fetch the inverted list for each query term.
- Use intersection of lists to resolve AND.
- Use union of lists to resolve OR.
- Take the complement of a list to resolve NOT (how?).
- Ignore within-document frequencies.

For strictly conjunctive queries

- start with the query term with lowest f_t
- shortest list as a set of candidates
- then eliminate documents that do not appear in the other lists (working from shortest to longest)

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Goal: Document ranking for a typical TF-IDF model

- Cosine as a similarity measure
- $w_{d,t}$: The frequency of each query term in each document (TF)
- f_t : The number of documents where each query term occurs (DF)
- W_d : The length of each document

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Goal: Document ranking for a typical TF-IDF model

- Cosine as a similarity measure
- $w_{d,t}$: The frequency of each query term in each document (TF)
- f_t : The number of documents where each query term occurs (DF)
- W_d : The length of each document

Typical cosine:

$$S(q, d) = \frac{q \cdot d}{|q||d|} = \frac{\sum_i q_i \cdot d_i}{|q||d|}$$

- calculate the dot product
- divide by the vector lengths
- length of the query q is often ignored (constant)

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Goal: Document ranking for a typical TF-IDF model

- Cosine as a similarity measure
- $w_{d,t}$: The frequency of each query term in each document (TF)
- f_t : The number of documents where each query term occurs (DF)
- W_d : The length of each document

Typical cosine:

$$S(q, d) = \frac{q \cdot d}{|q||d|} = \frac{\sum_i q_i \cdot d_i}{|q||d|}$$

Non-boolean TDM (32 bits per term per document): is too large to contemplate for a typical document collection on the WWW

Inverted index is not designed to allow us to compare documents one at a time.

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Inverted index to evaluate a query under the **cosine measure**,

- 1 Allocate an accumulator A_d for each document d
initialize $A_d \leftarrow 0$.
- 2 For each query term t ,
 - 1 Calculate $w_{q,t}$, and fetch the inverted list for t .
 - 2 For each pair $\langle d_t, f_{d,t} \rangle$ in the inverted list
Calculate $w_{d,t}$, and
Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$
- 3 Read the array of W_d values
for each $A_d > 0$, set $A_d \leftarrow A_d / W_d$.
- 4 Identify the r greatest A_d values and return the corresponding documents.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Inverted index to evaluate a query under the **cosine measure**,

- 1 Allocate an **accumulator** A_d for each document d
initialize $A_d \leftarrow 0$.
- 2 For each query term t ,
 - 1 Calculate $w_{q,t}$, and fetch the inverted list for t .
 - 2 For each pair $\langle d_t, f_{d,t} \rangle$ in the inverted list
Calculate $w_{d,t}$, and
Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$
- 3 Read the array of W_d values
for each $A_d > 0$, set $A_d \leftarrow A_d / W_d$.
- 4 Identify the r greatest A_d values and return the corresponding documents.

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Inverted index to evaluate a query under the **cosine measure**,

- 1 Allocate an accumulator A_d for each document d
initialize $A_d \leftarrow 0$.
- 2 For each **query term** t ,
 - 1 Calculate $w_{q,t}$, and **fetch the inverted list** for t .
 - 2 For each pair $\langle d_t, f_{d,t} \rangle$ in the inverted list
Calculate $w_{d,t}$, and
Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$
- 3 Read the array of W_d values
for each $A_d > 0$, set $A_d \leftarrow A_d / W_d$.
- 4 Identify the r greatest A_d values and return the corresponding documents.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Inverted index to evaluate a query under the **cosine measure**,

- 1 Allocate an accumulator A_d for each document d
initialize $A_d \leftarrow 0$.
- 2 For each query term t ,
 - 1 Calculate $w_{q,t}$, and fetch the inverted list for t .
 - 2 For each **pair** $\langle d_t, f_{d,t} \rangle$ **in the inverted list**
Calculate $w_{d,t}$, and
Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$ (**increment the accumulator**)
- 3 Read the array of W_d values
for each $A_d > 0$, set $A_d \leftarrow A_d / W_d$
- 4 Identify the r greatest A_d values and return the corresponding documents.

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

Inverted index to evaluate a query under the **cosine measure**,

- 1 Allocate an accumulator A_d for each document d
initialize $A_d \leftarrow 0$.
- 2 For each query term t ,
 - 1 Calculate $w_{q,t}$, and fetch the inverted list for t .
 - 2 For each pair $\langle d_t, f_{d,t} \rangle$ in the inverted list
Calculate $w_{d,t}$, and
Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$
- 3 Read the array of W_d values
for each $A_d > 0$, set $A_d \leftarrow A_d / W_d$ (**normalize by doc length**)
- 4 Identify the r greatest A_d values and return the corresponding documents.

Ranked Querying using an inverted index

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Inverted index to evaluate a query under the **cosine measure**,

- 1 Allocate an accumulator A_d for each document d
initialize $A_d \leftarrow 0$.
- 2 For each query term t ,
 - 1 Calculate $w_{q,t}$, and fetch the inverted list for t .
 - 2 For each pair $\langle d_t, f_{d,t} \rangle$ in the inverted list
Calculate $w_{d,t}$, and
Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$
- 3 Read the array of W_d values
for each $A_d > 0$, set $A_d \leftarrow A_d / W_d$
- 4 Identify the r **greatest** A_d **values** and return the corresponding documents (**ranking!**)

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

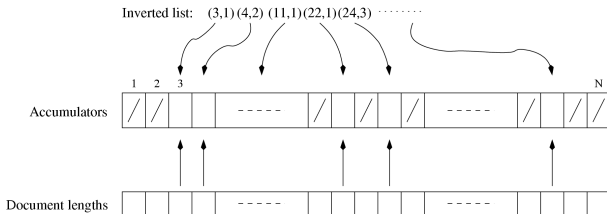
Inverted indices

Querying

Boolean queries

Ranked querying

In a nutshell,



- start with a set of N zero'ed accumulators
- use the inverted lists to update the accumulators term by term
- use the document lengths to normalize each non-zero accumulator

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

With the standard query evaluation algorithm and long queries, **most accumulators are non-zero** and an **array** is the **most space- and time-efficient** structure.

But the majority of those accumulator values are **trivially small**, with the only matching terms being **one or more common words**. And note that the accumulators are required on a per-query basis.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

With the standard query evaluation algorithm and long queries, **most accumulators are non-zero** and an **array** is the **most space- and time-efficient** structure.

But the majority of those accumulator values are **trivially small**, with the only matching terms being **one or more common words**. And note that the accumulators are required on a per-query basis.

If only low f_t (that is, **rare**) **terms are allowed to create accumulators**, the number of accumulators is greatly reduced.

A simple mechanism is to impose a **limit L on the number of accumulators**. This is another example of an efficiency-driven **compromise** that **alters the set of documents** returned, and may therefore impact on effectiveness.

The “limiting” approach

Web Search

COMP90049
Knowledge
Technologies

Overview
Elements

Crawling
Basics
Challenges

Parsing
Page analysis
Tokenisation
Stemming
Zoning

Indexing
Concepts
Inverted indices

Querying
Boolean queries
Ranked querying

- 1 Create an empty set A of accumulators and **set a length limit L**
- 2 For each query term t , **ordered by decreasing $w_{q,t}$**
 - 1 Calculate $w_{q,t}$, and fetch the inverted list for t .
 - 2 For each pair $\langle d_t, f_{d,t} \rangle$ in the inverted list
 - If there is no Accumulator for d **and** $|A| < L$
Create an accumulator A_d for d
 - If d has an accumulator
Calculate $w_{d,t}$
Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$
- 3 For each accumulator set $A_d \leftarrow A_d / W_d$.
- 4 Identify the r greatest A_d values and return these documents.

There are many variations on these algorithms.

The “thresholding” approach

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

- 1 Create an empty set A of accumulators, and **set a threshold** S
- 2 For each query term t , **ordered by decreasing** $w_{q,t}$
 - 1 Calculate $w_{q,t}$, and fetch the inverted list for t .
 - 2 For each pair $\langle d_t, f_{d,t} \rangle$ in the inverted list
Calculate $w_{d,t}$
If there is no Accumulator for d **and** $w_{q,t} \times w_{d,t} > S$
Create an accumulator A_d for d
If d has an accumulator
Set $A_d \leftarrow A_d + w_{q,t} \times w_{d,t}$
- 3 For each accumulator set $A_d \leftarrow A_d / W_d$.
- 4 Identify the r greatest A_d values and return these documents.

There are many variations on these algorithms.

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Several resources must be considered.

Disk space: for the index, at 40% of the size of the data. (With unstemmed terms, the index can be around 80% of the size of the data.)

Memory space: for accumulators, for the vocabulary, and for caching of previous results.

CPU time: for processing inverted lists and updating accumulators.

Disk traffic: to fetch inverted lists.

By judicious use of **compression** and careful **pruning**, all of these costs can be dramatically reduced compared to this first implementation. (Any practical implementation must make some use of compression.)

Web Search

COMP90049
Knowledge
Technologies

Overview

Elements

Crawling

Basics

Challenges

Parsing

Page analysis

Tokenisation

Stemming

Zoning

Indexing

Concepts

Inverted indices

Querying

Boolean queries

Ranked querying

Brin, Sergey and Lawrence Page (1998). “The Anatomy of a Large-Scale Hypertextual Web Search Engine”. *Computer Networks* 30: 107–117.

Barroso, Luiz André, Jeffrey Dean, and Urs Hötze (2003) “Web Search for a Planet: The Google Cluster Architecture”. *IEEE Micro* 23 (2): 22–28. doi:10.1109/MM.2003.1196112

Zobel, Justin and Alistair Moffatt (2006). “Inverted Files for Text Search Engines”. *ACM Computing Surveys* 38 (2): 1–56. doi:10.1145/1132956.1132959

Manning, Christopher D., Prabhakar Raghavan, Heinrich Schütze (2008). “Introduction to Information Retrieval”. Chapters 1–2, 20–21. Cambridge University Press.