Text Retrieval and Web Search
IIR 1: Boolean Retrieval

Mihai Surdeanu
(Based on slides by Hinrich Schütze at informationretrieval.org)

Spring 2017

Take-away

- Why you should take this course
- Admin issues
- Boolean Retrieval: Design and data structures of a simple information retrieval system
- What topics will be covered in this class?

Why you should take this course (take 1)

THE BEST PLACE TO HIDE A DEAD BODY
IS PAGE TWO OF THE GOOGLE SEARCH RESULTS

Why you should take this course (take 2)

"Making the world's problem solvers 10% more efficient"

Why you should take this course (take 2)

- Anurag Acharya is the key inventor of Google Scholar.
- `https://medium.com/backchannel/the-gentleman-who-made-scholar-d71289d9a82d`

Outline

Instructor information

Instructor: Mihai Surdeanu
Email: `msurdeanu@email.arizona.edu`
Office: Gould-Simpson 746
Office Hours: Mon/Wed 11 – noon

TA: Enrique Noriega
Email: `enoriega@email.arizona.edu`
Office: Gould-Simpson 931
Office Hours: Tue/Thu 11 – 12:30

Website and syllabus

- Website:
  `http://surdeanu.info/mihai/teaching/csc483583-spring17/`
- Syllabus:
  `http://surdeanu.info/mihai/teaching/csc483583-spring17/IR-syllabus.pdf`
- Piazza:
  `https://piazza.com/arizona/spring2017/csc483583/home`
- See website and syllabus for: code of conduct, classroom electronics, DRC accommodations, non-discrimination policy, code of academic integrity, etc.
- But most materials will actually be in D2L

Prerequisites

- Know how to program and have a decent understanding of data structures such as hash maps and trees: CSC 345
- Ideally, Math 129 (Calc 2). However, we will cover the necessary math in class.

Prerequisites: does this look scary?

```
$\textsc{IntersectWithSkips}(p_1, p_2)$
answer $\leftarrow \langle\ \rangle$
while $p_1 \neq$ NIL and $p_2 \neq$ NIL
do if $docID(p_1) = docID(p_2)$
      then Add(answer, $docID(p_1)$)
           $p_1 \leftarrow next(p_1)$
           $p_2 \leftarrow next(p_2)$
      else if $docID(p_1) < docID(p_2)$
              then if $hasSkip(p_1)$ and $(docID(skip(p_1)) \leq docID(p_2))$
                      then while $hasSkip(p_1)$ and $(docID(skip(p_1)) \leq docID(p_2))$
                              do $p_1 \leftarrow skip(p_1)$
                      else $p_1 \leftarrow next(p_1)$
              else if $hasSkip(p_2)$ and $(docID(skip(p_2)) \leq docID(p_1))$
                      then while $hasSkip(p_2)$ and $(docID(skip(p_2)) \leq docID(p_1))$
                              do $p_2 \leftarrow skip(p_2)$
                      else $p_2 \leftarrow next(p_2)$
return answer
```

Prerequisites: does this look scary?

$$||x||_2 = \sqrt{\sum_i x_i^2}$$

$$\cos(\vec{q}, \vec{d}) = \textsc{sim}(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

Dot product, matrix multiplication, Bayes rule

Choosing a programming language
My recommendations

- Scala
- Java
- Python
- C/C++

Java

- Pros
  - Pretty fast
  - Probably the most common language for IR and NLP
  - Clean exception handling
  - Statically typed
  - Garbage collection
  - Several great IDEs
- Cons
  - Syntax too verbose
  - Inconsistent semantics due to enforced backwards compatibility (primitive types vs. objects, equality, etc.)

Scala

- Pros
  - Pretty fast
  - “Hot” language for IR, NLP, ML, distributed computing, web development
  - Clean, transparent exception handling
  - Clean, minimalist syntax
  - Consistent semantics
  - Statically typed
  - Garbage collection
  - At least one great IDE (IntelliJ)
  - Fully compatible with Java (use all Java libraries)
- Cons
  - It has some “dark corners”
  - Backwards compatibility not guaranteed

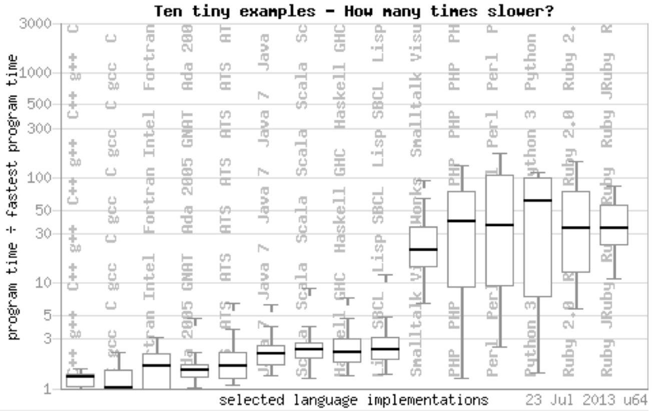
Python

- Pros
  - Clean syntax
  - Popular: many NLP/ML libraries exist
  - Clean exception handling
  - Easy access to GPUs
- Cons
  - **Slow**
  - **Dynamically typed**
  - No great IDE

C/C++

- Pros
    - As fast as it gets
- Cons
    - Too many to list

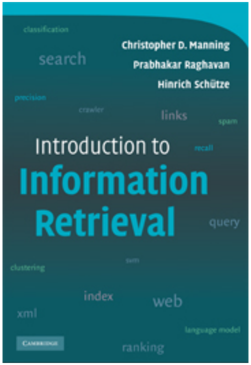
Comparison



More benchmarks:

`http://benchmarksgame.alioth.debian.org/u64/benchmark.php?test=all&lang=all&data=u64`

Textbook



- Introduction to Information Retrieval, by Manning et al.
- `http://nlp.stanford.edu/IR-book/`

Grading

- Final grade = 4 assignments + 2 exams + 1 project + in-class participation
- First assignment has been posted. Due January 29! Let's take a quick look at the assignment.
- Undergraduate vs. graduate requirements

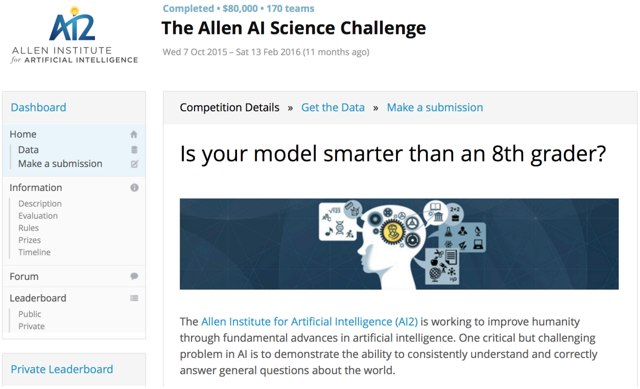
Grading scheme

Component	Weight
Written assignments	300 pts
Midterm exam	200 pts
Final exam	275 pts
Programming project	200 pts
In-class participation	25 pts
Total	1000 pts

Grade	Point Range
A	900 – 1000
B	800 – 899
C	700 – 799
D	600 – 699
E	0 – 599

Final Project: Option 1

Final Project: Option 2



Completed • $80,000 • 170 teams

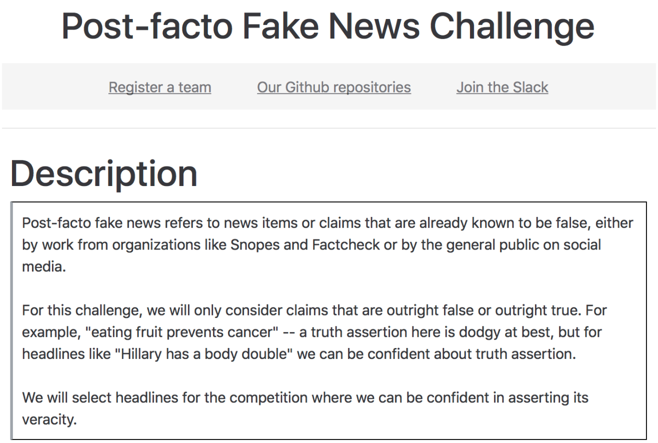
The Allen AI Science Challenge

Wed 7 Oct 2015 – Sat 13 Feb 2016 (11 months ago)

Is your model smarter than an 8th grader?

The Allen Institute for Artificial Intelligence (AI2) is working to improve humanity through fundamental advances in artificial intelligence. One critical but challenging problem in AI is to demonstrate the ability to consistently understand and correctly answer general questions about the world.

Final Project: Option 3



Post-facto Fake News Challenge

Register a team   Our Github repositories   Join the Slack

Description

Post-facto fake news refers to news items or claims that are already known to be false, either by work from organizations like Snopes and Factcheck or by the general public on social media.

For this challenge, we will only consider claims that are outright false or outright true. For example, "eating fruit prevents cancer" -- a truth assertion here is dodgy at best, but for headlines like "Hillary has a body double" we can be confident about truth assertion.

We will select headlines for the competition where we can be confident in asserting its veracity.

Late work + attendance policy

- Late work is not accepted, except in case of documented emergency approved by the instructor

- Attendance is required
- Students who miss class due to illness or emergency are required to bring documentation

Dates

What	**When**
HW 1	January 29
HW 2	February 19
Midterm	March 1
HW 3	March 12
Spring recess	March 13 - 17
HW 4	April 9
Project	May 7
Final	After last week of class

Written assignments will be due approximately every three weeks, as announced by the instructor. All assignments are due in the D2L dropbox by 11:59 P.M. on the indicated day.

Cooperation & Cheating

- Students may not discuss individual homework with anybody other than the instructors and teaching assistants.
- Students may not share individual homework solutions with anybody.
- Students may post questions to Piazza, but should refrain from posting solutions or partial solutions.
- Students may not share test cases with anybody.
- Students may share class notes with anybody.
- Students may not seek individual homework help from anybody other than the instructors, teaching assistants, or departmental tutors.
- If permitted, the use of open source or third party materials in student submissions must be clearly identified and credited. Assignment and project submissions must be substantially the work of the student who submits the work.

Cooperation & Cheating

- Students who violate the Code should expect a penalty that is **greater than the value of the work in question up to and including failing the course**.
- A record of the incident **will** be sent to the Dean of Students office. If you have been involved in other Code violations, the Dean of Students may impose additional sanctions.

Outline

Definition of *information retrieval*

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

Definition of *information retrieval*

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

Definition of *information retrieval*

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

Definition of *information retrieval*

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

Definition of *information retrieval*

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

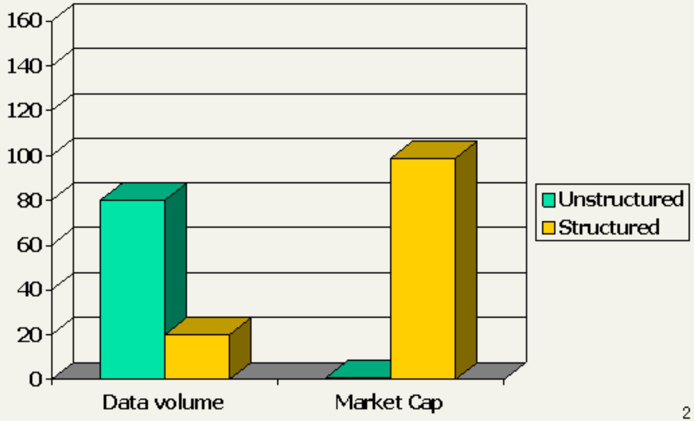
Definition of *information retrieval*

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

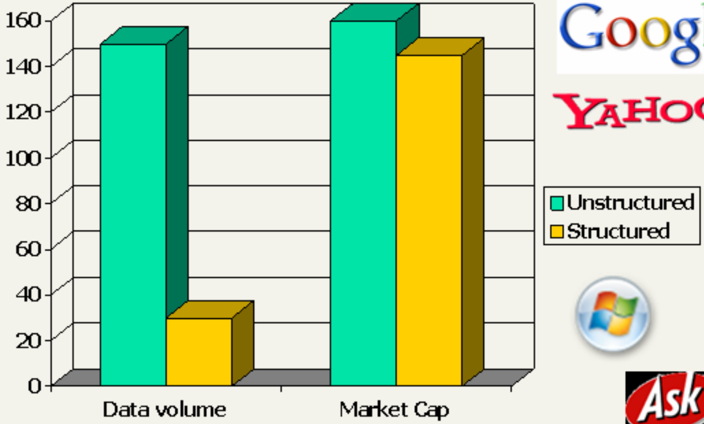
Definition of *information retrieval*

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

Unstructured (text) vs. structured (database) data in 1996



Unstructured (text) vs. structured (database) data in 2006



Companies using IR

- Google, Yahoo, Microsoft: search web, email, choose ads
- Facebook: search friends' posts, choose wall
- Twitter: search tweets
- HP Autonomy: enterprise search
- Pandora: music (!) search

Why you should take this course

- Take 3: IR is important
- Take 4:
  - This is a gateway course for data science and machine learning
  - Real-world applications of programming + data structures + probability theory

Boolean retrieval

- The Boolean model is arguably the simplest model to base an information retrieval system on.
- Queries are Boolean expressions, e.g., CAESAR AND BRUTUS
- The search engine returns all documents that satisfy the Boolean expression.

Boolean retrieval

- The Boolean model is arguably the simplest model to base an information retrieval system on.
- Queries are Boolean expressions, e.g., CAESAR AND BRUTUS
- The search engine returns all documents that satisfy the Boolean expression.

Does Google use the Boolean model?

Does Google use the Boolean model?

- On Google, the default interpretation of a query [w_1 w_2 $\ldots w_n$] is w_1 AND w_2 AND $\ldots$ AND w_n
- Cases where you get hits that do not contain one of the w_i:
  - anchor text
  - page contains variant of w_i (morphology, spelling correction, synonym)
  - long queries (n large)
  - boolean expression generates very few hits
- Simple Boolean vs. Ranking of result set
  - Simple Boolean retrieval returns matching documents in no particular order.
  - Google (and most well designed Boolean engines) rank the result set – they rank good hits (according to some estimator of relevance) higher than bad hits.

Outline

Unstructured data in 1650: Shakespeare

Unstructured data in 1650

- Which plays of Shakespeare contain the words BRUTUS AND CAESAR, but NOT CALPURNIA?
- One could grep all of Shakespeare's plays for BRUTUS and CAESAR, then strip out lines containing CALPURNIA.
- Why is grep not the solution?

Unstructured data in 1650

- Which plays of Shakespeare contain the words BRUTUS AND CAESAR, but NOT CALPURNIA?
- One could grep all of Shakespeare's plays for BRUTUS and CAESAR, then strip out lines containing CALPURNIA.
- Why is grep not the solution?
    - Slow (for large collections)
    - grep is line-oriented, IR is document-oriented
    - "NOT CALPURNIA" is non-trivial
    - Other operations (e.g., find the word ROMANS near COUNTRYMAN) not feasible

Term-document incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	1	1	0	0	0	1	
BRUTUS	1	1	0	1	0	0	
CAESAR	1	1	0	1	1	1	
CALPURNIA	0	1	0	0	0	0	
CLEOPATRA	1	0	0	0	0	0	
MERCY	1	0	1	1	1	1	
WORSER	1	0	1	1	1	0	
...							

Entry is 1 if term occurs. Example: CALPURNIA occurs in *Julius Caesar*.
Entry is 0 if term doesn't occur. Example: CALPURNIA doesn't occur in *The tempest*.

Term-document incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	. . .
ANTHONY	1	1	0	0	0	1	
BRUTUS	1	1	0	1	0	0	
CAESAR	1	1	0	1	1	1	
CALPURNIA	0	1	0	0	0	0	
CLEOPATRA	1	0	0	0	0	0	
MERCY	1	0	1	1	1	1	
WORSER	1	0	1	1	1	0	
. . .							

Entry is 1 if term occurs. Example: CALPURNIA occurs in *Julius Caesar*.

Entry is 0 if term doesn't occur. Example: CALPURNIA doesn't occur in *The tempest*.

Term-document incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	. . .
ANTHONY	1	1	0	0	0	1	
BRUTUS	1	1	0	1	0	0	
CAESAR	1	1	0	1	1	1	
CALPURNIA	0	1	0	0	0	0	
CLEOPATRA	1	0	0	0	0	0	
MERCY	1	0	1	1	1	1	
WORSER	1	0	1	1	1	0	
. . .							

Entry is 1 if term occurs. Example: CALPURNIA occurs in *Julius Caesar*.
Entry is 0 if term doesn't occur. Example: CALPURNIA doesn't occur in *The tempest*.

Incidence vectors

- So we have a 0/1 vector for each term.
- To answer the query BRUTUS AND CAESAR AND NOT CALPURNIA:

Incidence vectors

- So we have a 0/1 vector for each term.
- To answer the query BRUTUS AND CAESAR AND NOT CALPURNIA:
  - Take the vectors for BRUTUS, CAESAR, and CALPURNIA
  - Complement the vector of CALPURNIA
  - Do a (bitwise) AND on the three vectors
  - 110100 AND 110111 AND 101111 = 100100

0/1 vector for BRUTUS

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	1	1	0	0	0	1	
BRUTUS	1	1	0	1	0	0	
CAESAR	1	1	0	1	1	1	
CALPURNIA	0	1	0	0	0	0	
CLEOPATRA	1	0	0	0	0	0	
MERCY	1	0	1	1	1	1	
WORSER	1	0	1	1	1	0	
...							
result:	1	0	0	1	0	0	

Answers to query

Anthony and Cleopatra, Act III, Scene ii

Agrippa [Aside to Domitius Enobarbus]: Why, Enobarbus,
When Antony found Julius Caesar dead,
He cried almost to roaring; and he wept
When at Philippi he found Brutus slain.

Hamlet, Act III, Scene ii

Lord Polonius: I did enact Julius Caesar: I was killed i' the
Capitol; Brutus killed me.

Bigger collections

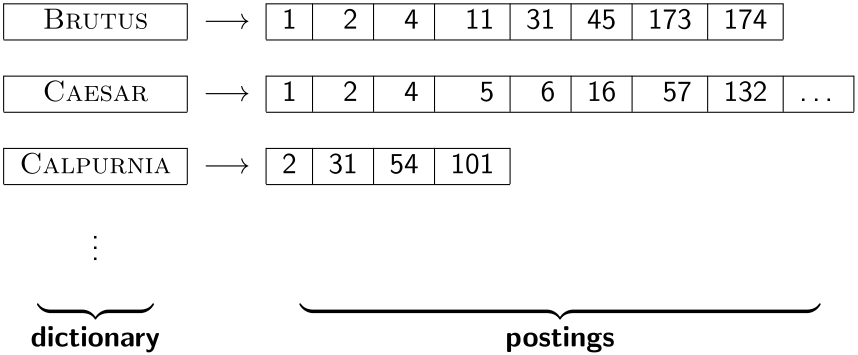
- Consider $N = 10^6$ documents, each with about 1000 tokens
- $\Rightarrow$ total of 10^9 tokens
- On average 6 bytes per token, including spaces and punctuation $\Rightarrow$ size of document collection is about $6 \cdot 10^9 =$ 6 GB
- Assume there are $M = 500{,}000$ distinct terms in the collection
- (Notice that we are making a term/token distinction.)

Can’t build the incidence matrix

- $M = 500{,}000 \times 10^6 =$ half a trillion 0s and 1s.
- But the matrix has no more than one billion 1s.
  - Matrix is extremely sparse.
- What is a better representations?
  - We only record the 1s.

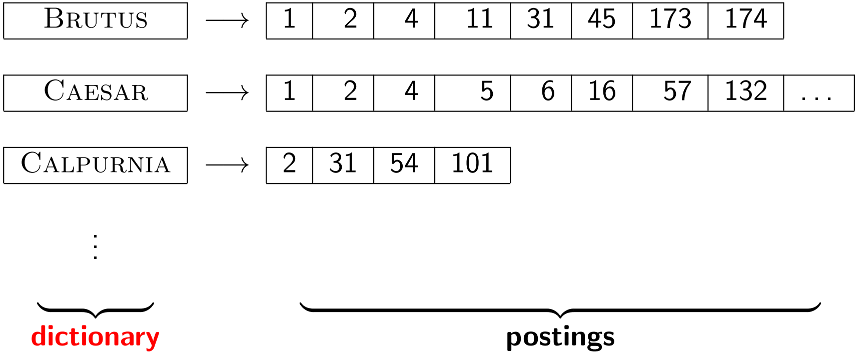
Inverted Index

For each term t, we store a list of all documents that contain t.



Inverted Index

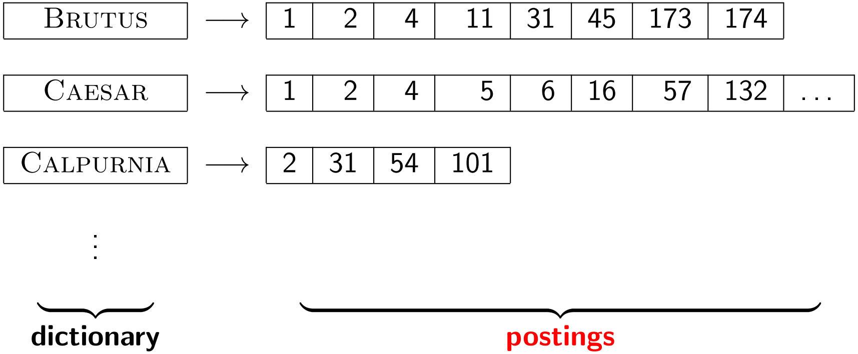
For each term *t*, we store a list of all documents that contain *t*.



BRUTUS → 1 2 4 11 31 45 173 174
CAESAR → 1 2 4 5 6 16 57 132 ...
CALPURNIA → 2 31 54 101
⋮
dictionary
postings

Inverted Index

For each term t, we store a list of all documents that contain t.


BRUTUS → 1 2 4 11 31 45 173 174
CAESAR → 1 2 4 5 6 16 57 132 ...
CALPURNIA → 2 31 54 101
⋮
dictionary
postings

Inverted index construction

1. Collect the documents to be indexed:
   $\boxed{\text{Friends, Romans, countrymen.}}$ $\boxed{\text{So let it be with Caesar}}$ ...
2. Tokenize the text, turning each document into a list of tokens:
   $\boxed{\text{Friends}}$ $\boxed{\text{Romans}}$ $\boxed{\text{countrymen}}$ $\boxed{\text{So}}$ ...
3. Do linguistic preprocessing, producing a list of normalized tokens, which are the indexing terms: $\boxed{\text{friend}}$ $\boxed{\text{roman}}$ $\boxed{\text{countryman}}$ $\boxed{\text{so}}$ ...
4. Index the documents that each term occurs in by creating an inverted index, consisting of a dictionary and postings.

Tokenization and preprocessing

Doc 1. I did enact Julius Caesar: I was killed i' the Capitol; Brutus killed me.
Doc 2. So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious:

$\Longrightarrow$

Doc 1. i did enact julius caesar i was killed i' the capitol brutus killed me
Doc 2. so let it be with caesar the noble brutus hath told you caesar was ambitious

Generate postings

Doc 1. i did enact julius caesar i was killed i' the capitol brutus killed me
Doc 2. so let it be with caesar the noble brutus hath told you caesar was ambitious

$\Longrightarrow$

term	**docID**
i	1
did	1
enact	1
julius	1
caesar	1
i	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Sort postings

term	**docID**
i	1
did	1
enact	1
julius	1
caesar	1
i	1
was	1
killed	1
i’	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

$\Longrightarrow$

term	**docID**
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
i	1
i	1
i’	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

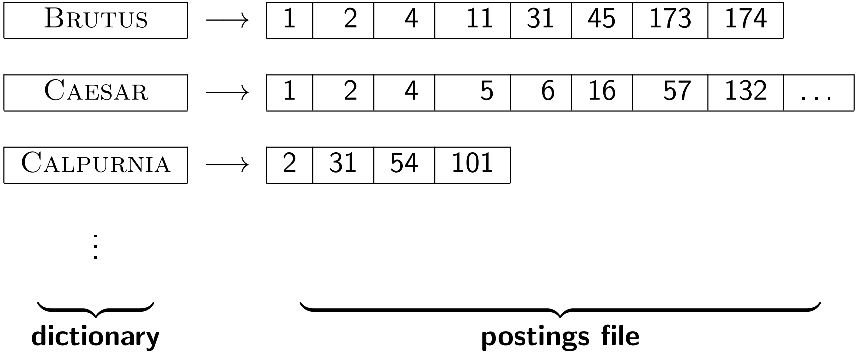
Create postings lists, determine document frequency

term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
i	1
i	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

$\Longrightarrow$

term	doc. freq.	→	postings lists
ambitious	1	→	$\boxed{2}$
be	1	→	$\boxed{2}$
brutus	2	→	$\boxed{1} \rightarrow \boxed{2}$
capitol	1	→	$\boxed{1}$
caesar	2	→	$\boxed{1} \rightarrow \boxed{2}$
did	1	→	$\boxed{1}$
enact	1	→	$\boxed{1}$
hath	1	→	$\boxed{2}$
i	1	→	$\boxed{1}$
i'	1	→	$\boxed{1}$
it	1	→	$\boxed{2}$
julius	1	→	$\boxed{1}$
killed	1	→	$\boxed{1}$
let	1	→	$\boxed{2}$
me	1	→	$\boxed{1}$
noble	1	→	$\boxed{2}$
so	1	→	$\boxed{2}$
the	2	→	$\boxed{1} \rightarrow \boxed{2}$
told	1	→	$\boxed{2}$
you	1	→	$\boxed{2}$
was	2	→	$\boxed{1} \rightarrow \boxed{2}$
with	1	→	$\boxed{2}$

Split the result into dictionary and postings file


BRUTUS → 1 2 4 11 31 45 173 174
CAESAR → 1 2 4 5 6 16 57 132 ...
CALPURNIA → 2 31 54 101
⋮
dictionary
postings file

Later in this course

- Index construction: how can we create inverted indexes for large collections?
- How much space do we need for dictionary and index?
- Index compression: how can we efficiently store and process indexes for large collections?
- Ranked retrieval: what does the inverted index look like when we want the “best” answer?

Outline

Simple conjunctive query (two terms)

- Consider the query: BRUTUS AND CALPURNIA
- To find all matching documents using inverted index:
  1. Locate BRUTUS in the dictionary
  2. Retrieve its postings list from the postings file
  3. Locate CALPURNIA in the dictionary
  4. Retrieve its postings list from the postings file
  5. Intersect the two postings lists
  6. Return intersection to user

Intersecting two postings lists

$$\begin{array}{lcl}
\textsc{Brutus} & \longrightarrow & \boxed{1}\rightarrow\boxed{2}\rightarrow\boxed{4}\rightarrow\boxed{11}\rightarrow\boxed{31}\rightarrow\boxed{45}\rightarrow\boxed{173}\rightarrow\boxed{174} \\
\textsc{Calpurnia} & \longrightarrow & \boxed{2}\rightarrow\boxed{31}\rightarrow\boxed{54}\rightarrow\boxed{101} \\
\text{Intersection} & \Longrightarrow &
\end{array}$$

Intersecting two postings lists

$$\text{BRUTUS} \longrightarrow \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$$

$$\text{CALPURNIA} \longrightarrow \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$$

$$\text{Intersection} \Longrightarrow$$

Intersecting two postings lists

$$\begin{tikzcd}
\textsc{Brutus} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{174} \\
\textsc{Calpurnia} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101} \\
\text{Intersection} \arrow[r, Rightarrow] & {}
\end{tikzcd}$$

Intersecting two postings lists

$$\begin{tikzcd}
\textsc{Brutus} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{174} \\
\textsc{Calpurnia} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101} \\
\text{Intersection} \arrow[r, Rightarrow] & \boxed{2}
\end{tikzcd}$$

Intersecting two postings lists

$$\begin{tikzcd}
\textsc{Brutus} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{174} \\
\textsc{Calpurnia} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101} \\
\text{Intersection} \arrow[r, Rightarrow] & \boxed{2}
\end{tikzcd}$$

Intersecting two postings lists

$$\textsc{Brutus} \longrightarrow \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$$

$$\textsc{Calpurnia} \longrightarrow \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$$

$$\text{Intersection} \Longrightarrow \boxed{2}$$

Intersecting two postings lists

$$\begin{tikzcd}[column sep=small]
\textsc{Brutus} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{174} \\
\textsc{Calpurnia} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101} \\
\text{Intersection} \arrow[r, Rightarrow] & \boxed{2}
\end{tikzcd}$$

Intersecting two postings lists

$$\begin{tikzcd}
\textsc{Brutus} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{\color{red}31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{174} \\
\textsc{Calpurnia} \arrow[r] & \boxed{2} \arrow[r] & \boxed{\color{red}31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101} \\
\text{Intersection} \arrow[r, Rightarrow] & \boxed{2} \arrow[r] & \boxed{31}
\end{tikzcd}$$

Intersecting two postings lists

$\textsc{Brutus} \longrightarrow \boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

$\textsc{Calpurnia} \longrightarrow \boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow \boxed{2} \rightarrow \boxed{31}$

Intersecting two postings lists

$$\begin{tikzcd}
\text{BRUTUS} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{174} \\
\text{CALPURNIA} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101} \\
\text{Intersection} \arrow[r, Rightarrow] & \boxed{2} \arrow[r] & \boxed{31}
\end{tikzcd}$$

Intersecting two postings lists

$$\begin{tikzcd}
\textsc{Brutus} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{174} \\
\textsc{Calpurnia} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101} \\
\text{Intersection} \arrow[r, Rightarrow] & \boxed{2} \arrow[r] & \boxed{31}
\end{tikzcd}$$

Intersecting two postings lists

$$\begin{tikzcd}
\textsc{Brutus} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{174} \\
\textsc{Calpurnia} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101} \\
\text{Intersection} \arrow[r, Rightarrow] & \boxed{2} \arrow[r] & \boxed{31}
\end{tikzcd}$$

Intersecting two postings lists

$$\text{B\scriptsize RUTUS} \longrightarrow \boxed{1}\rightarrow\boxed{2}\rightarrow\boxed{4}\rightarrow\boxed{11}\rightarrow\boxed{31}\rightarrow\boxed{45}\rightarrow\boxed{173}\rightarrow\boxed{174}$$

$$\text{C\scriptsize ALPURNIA} \longrightarrow \boxed{2}\rightarrow\boxed{31}\rightarrow\boxed{54}\rightarrow\boxed{101}$$

$$\text{Intersection} \Longrightarrow \boxed{2}\rightarrow\boxed{31}$$

- This is linear in the length of the postings lists.

Intersecting two postings lists

$$\begin{array}{lll}
\textsc{Brutus} & \longrightarrow & \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174} \\
\textsc{Calpurnia} & \longrightarrow & \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101} \\
\text{Intersection} & \Longrightarrow & \boxed{2} \to \boxed{31}
\end{array}$$

- This is linear in the length of the postings lists.
- Note: This only works if postings lists are sorted.

Intersecting two postings lists

```
INTERSECT($p_1, p_2$)
 1  $answer \leftarrow \langle\ \rangle$
 2  while $p_1 \neq$ NIL and $p_2 \neq$ NIL
 3  do if $docID(p_1) = docID(p_2)$
 4        then ADD($answer, docID(p_1)$)
 5             $p_1 \leftarrow next(p_1)$
 6             $p_2 \leftarrow next(p_2)$
 7        else if $docID(p_1) < docID(p_2)$
 8               then $p_1 \leftarrow next(p_1)$
 9               else $p_2 \leftarrow next(p_2)$
10  return $answer$
```

Query processing: Exercise

$$\begin{tikzcd}
\text{FRANCE} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{3} \arrow[r] & \boxed{4} \arrow[r] & \boxed{5} \arrow[r] & \boxed{7} \arrow[r] & \boxed{8} \arrow[r] & \boxed{9} \arrow[r] & \boxed{11} \arrow[r] & \boxed{12} \arrow[r] & \boxed{13} \arrow[r] & \boxed{14} \arrow[r] & \boxed{15} \\
\text{PARIS} \arrow[r] & \boxed{2} \arrow[r] & \boxed{6} \arrow[r] & \boxed{10} \arrow[r] & \boxed{12} \arrow[r] & \boxed{14} \\
\text{LEAR} \arrow[r] & \boxed{12} \arrow[r] & \boxed{15}
\end{tikzcd}$$

Compute hit list for ((paris AND NOT france) OR lear)

Boolean queries

- The Boolean retrieval model can answer any query that is a Boolean expression.
  - Boolean queries are queries that use AND, OR and NOT to join query terms.
  - Views each document as a set of terms.
  - Is precise: Document matches condition or not.
- Primary commercial retrieval tool for 3 decades
- Many professional searchers (e.g., lawyers) still like Boolean queries.
  - You know exactly what you are getting.
- Many search systems you use are also Boolean: spotlight, email, intranet etc.

Commercially successful Boolean retrieval: Westlaw

- Largest commercial legal search service in terms of the number of paying subscribers
- Over half a million subscribers performing millions of searches a day over tens of terabytes of text data
- The service was started in 1975.
- In 2005, Boolean search (called "Terms and Connectors" by Westlaw) was still the default, and used by a large percentage of users . . .
- . . . although ranked retrieval has been available since 1992. Why do you think this is the case?

Westlaw: Example queries

Information need: Information on the legal theories involved in preventing the disclosure of trade secrets by employees formerly employed by a competing company

Query: "trade secret" /s disclos! /s prevent /s employe!

Westlaw: Example queries

Information need: Requirements for disabled people to be able to access a workplace

Query: disab! /p access! /s work-site work-place (employment /3 place)

Westlaw: Example queries

Information need: Cases about a host's responsibility for drunk guests

Query: host! /p (responsib! liab!) /p (intoxicat! drunk!) /p guest

Westlaw: Comments

- Proximity operators: /3 = within 3 words, /s = within a sentence, /p = within a paragraph
- Space is disjunction, not conjunction! (This was the default in search pre-Google.) Why do you think this was the case?
- Long, precise queries: incrementally developed, not like web search
- Why professional searchers often like Boolean search: precision, transparency, control
- When are Boolean queries the best way of searching? Depends on: information need, searcher, document collection, . . .

Outline

Query optimization

- Consider a query that is an AND of n terms, $n > 2$
- For each of the terms, get its postings list, then AND them together
- Example query: BRUTUS AND CALPURNIA AND CAESAR
- What is the best order for processing this query?

Query optimization

- Example query: BRUTUS AND CALPURNIA AND CAESAR

Query optimization

- Example query: BRUTUS AND CALPURNIA AND CAESAR
- Simple and effective optimization: Process in order of increasing frequency. Why?

Query optimization

- Example query: BRUTUS AND CALPURNIA AND CAESAR
- Simple and effective optimization: Process in order of increasing frequency. Why?
- Start with the shortest postings list, then keep cutting further

Query optimization

- Example query: BRUTUS AND CALPURNIA AND CAESAR
- Simple and effective optimization: Process in order of increasing frequency. Why?
- Start with the shortest postings list, then keep cutting further

$$\begin{tikzcd}
\text{BRUTUS} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{174} \\
\text{CALPURNIA} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101} \\
\text{CAESAR} \arrow[r] & \boxed{5} \arrow[r] & \boxed{31}
\end{tikzcd}$$

Query optimization

- Example query: BRUTUS AND CALPURNIA AND CAESAR
- Simple and effective optimization: Process in order of increasing frequency. Why?
- Start with the shortest postings list, then keep cutting further
- In this example, first CAESAR, then CALPURNIA, then BRUTUS

$$\text{BRUTUS} \longrightarrow \boxed{1}\to\boxed{2}\to\boxed{4}\to\boxed{11}\to\boxed{31}\to\boxed{45}\to\boxed{173}\to\boxed{174}$$

$$\text{CALPURNIA} \longrightarrow \boxed{2}\to\boxed{31}\to\boxed{54}\to\boxed{101}$$

$$\text{CAESAR} \longrightarrow \boxed{5}\to\boxed{31}$$

Optimized intersection algorithm for conjunctive queries

```
$\text{INTERSECT}(\langle t_1, \ldots, t_n \rangle)$
1  $terms \leftarrow \text{SORTBYINCREASINGFREQUENCY}(\langle t_1, \ldots, t_n \rangle)$
2  $result \leftarrow postings(first(terms))$
3  $terms \leftarrow rest(terms)$
4  while $terms \neq$ NIL and $result \neq$ NIL
5  do $result \leftarrow \text{INTERSECT}(result, postings(first(terms)))$
6     $terms \leftarrow rest(terms)$
7  return $result$
```

More general optimization

- Example query: (MADDING OR CROWD) AND (IGNOBLE OR STRIFE)
- Get frequencies for all terms
- Estimate the size of each OR by the sum of its frequencies (conservative)
- Process in increasing order of OR sizes

Outline

Course overview

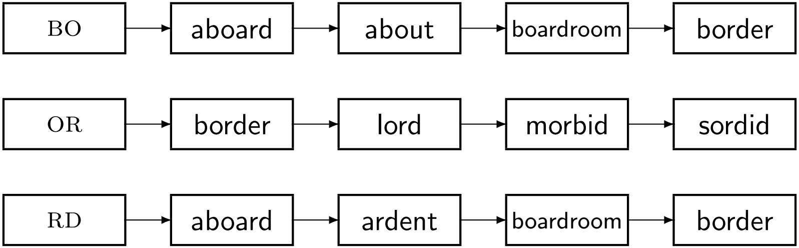
- We are done with Chapter 1 of IIR (IIR 01).
- Plan for the rest of the semester: 18–20 of the 21 chapters of IIR
- In what follows: teasers for most chapters – to give you a sense of what will be covered.

IIR 02: The term vocabulary and postings lists

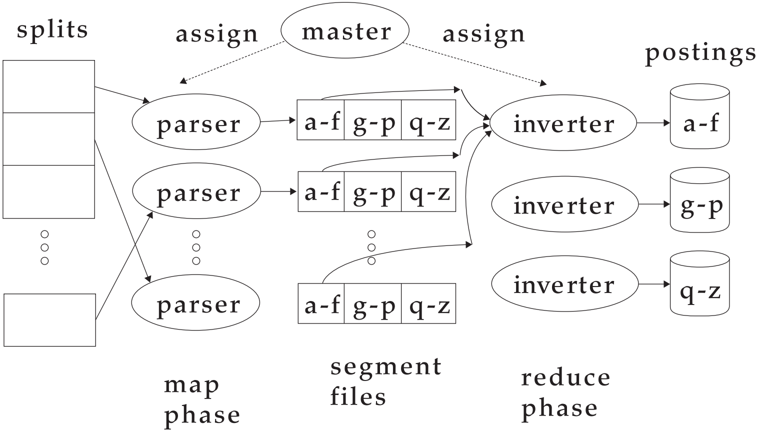
- Phrase queries: "STANFORD UNIVERSITY"
- Proximity queries: GATES NEAR MICROSOFT
- We need an index that captures position information for phrase queries and proximity queries.

IIR 03: Dictionaries and tolerant retrieval

Wildcard queries, spelling correction

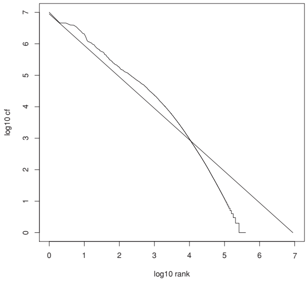


IIR 04: Index construction



IIR 05: Index compression

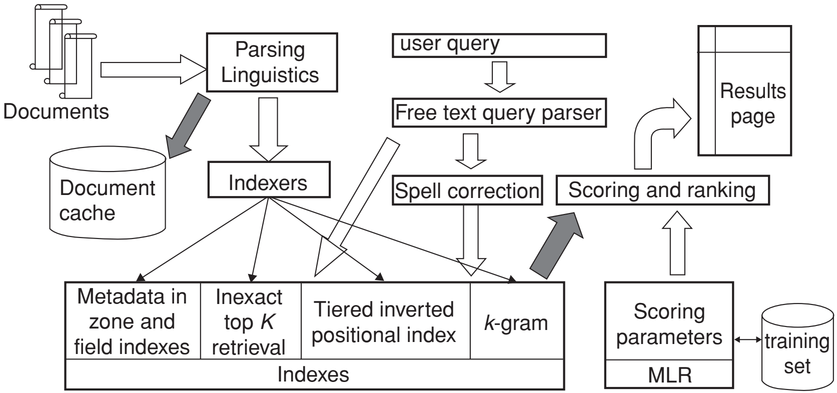
Zipf's law:



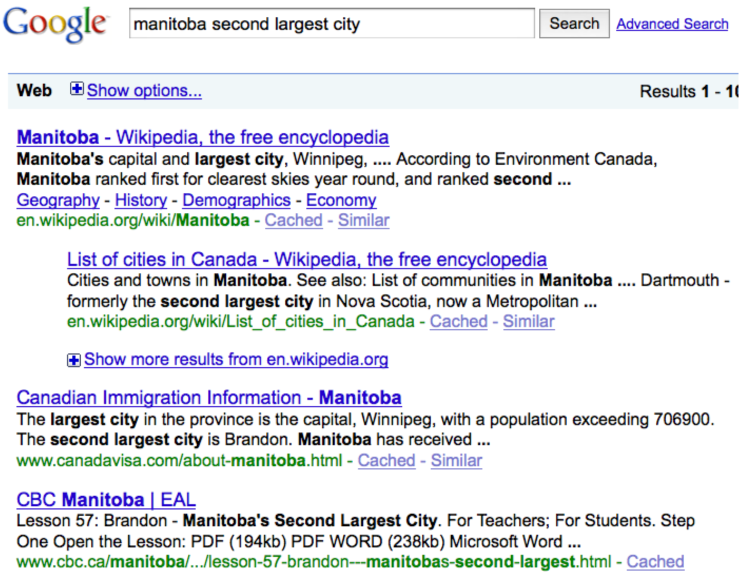
IIR 06: The vector space model

- Ranking search results
  - Boolean queries only give inclusion or exclusion of documents.
  - For ranked retrieval, we measure the proximity between the query and each document.
  - One formalism for doing this: the vector space model
- Key challenge in ranked retrieval: evidence accumulation for a term in a document
  - 1 vs. 0 occurence of a query term in the document
  - 3 vs. 2 occurences of a query term in the document
  - Usually: more is better
  - But by how much?
  - Need a scoring function that translates frequency into score or weight

IIR 07: Scoring in a complete search system



IIR 08: Evaluation and dynamic summaries



Manitoba - Wikipedia, the free encyclopedia
Manitoba's capital and **largest city**, Winnipeg, ... According to Environment Canada, **Manitoba** ranked first for clearest skies year round, and ranked **second** ...
Geography - History - Demographics - Economy
en.wikipedia.org/wiki/**Manitoba** - Cached - Similar

List of cities in Canada - Wikipedia, the free encyclopedia
Cities and towns in **Manitoba**. See also: List of communities in **Manitoba** .... Dartmouth - formerly the **second largest city** in Nova Scotia, now a Metropolitan ...
en.wikipedia.org/wiki/List_of_cities_in_Canada - Cached - Similar

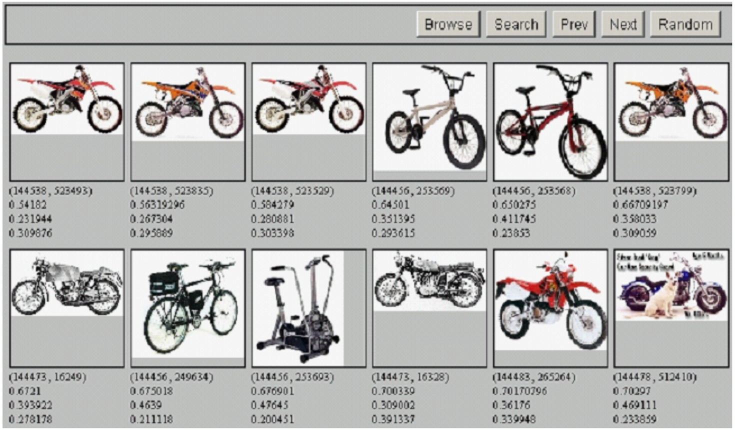
Show more results from en.wikipedia.org

Canadian Immigration Information - **Manitoba**
The **largest city** in the province is the capital, Winnipeg, with a population exceeding 706900. The **second largest city** is Brandon. **Manitoba** has received ...
www.canadavisa.com/about-**manitoba**.html - Cached - Similar

CBC **Manitoba** | EAL
Lesson 57: Brandon - **Manitoba's Second Largest City**. For Teachers; For Students. Step One Open the Lesson: PDF (194kb) PDF WORD (238kb) Microsoft Word ...
www.cbc.ca/**manitoba**/.../lesson-57-brandon---**manitobas-second-largest**.html - Cached

IIR 09: Relevance feedback & query expansion

Keeping the human in the loop:



(144538 , 523493)	(144538 , 523835)	(144538 , 523529)	(144456 , 253569)	(144456 , 253568)	(144538 , 523799)
0.54182	0.56319296	0.584279	0.64501	0.650275	0.66709197
0.231944	0.267304	0.280881	0.351395	0.411745	0.358033
0.309876	0.295889	0.303398	0.293615	0.23853	0.309059
(144473 , 16249)	(144456 , 249634)	(144456 , 253693)	(144473 , 16328)	(144483 , 265264)	(144478 , 512410)
0.6721	0.675018	0.676901	0.700339	0.70170796	0.70297
0.393922	0.4639	0.47645	0.309002	0.36176	0.469111
0.278178	0.211118	0.200451	0.391337	0.339948	0.233859

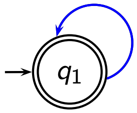
IIR 11: Probabilistic information retrieval

$P(R|d, q)$ is modeled using term incidence vectors as $P(R|\vec{x}, \vec{q})$

$$\begin{aligned} P(R = 1|\vec{x}, \vec{q}) &= \frac{P(\vec{x}|R = 1, \vec{q})P(R = 1|\vec{q})}{P(\vec{x}|\vec{q})} \\ P(R = 0|\vec{x}, \vec{q}) &= \frac{P(\vec{x}|R = 0, \vec{q})P(R = 0|\vec{q})}{P(\vec{x}|\vec{q})} \end{aligned}$$

- $P(\vec{x}|R = 1, \vec{q})$ and $P(\vec{x}|R = 0, \vec{q})$: probability that if a relevant or nonrelevant document is retrieved, then that document's representation is $\vec{x}$
- Use statistics about the document collection to estimate these probabilities

IIR 12: Language models



w	$P(w\|q_1)$	w	$P(w\|q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
		...	...

This is a one-state probabilistic finite-state automaton – a unigram language model – and the state emission distribution for its one state q_1.

STOP is not a word, but a special symbol indicating that the automaton stops.

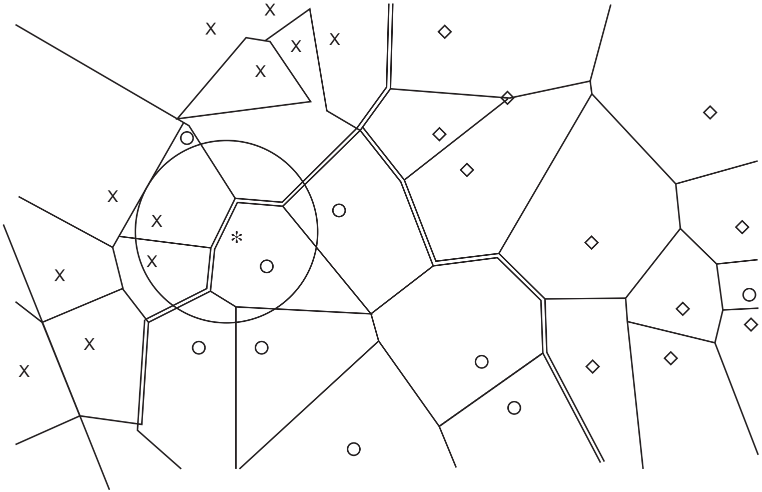
"frog said that toad likes frog STOP"

$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2 = 0.0000000000048$

IIR 13: Text classification & Naive Bayes

- Text classification = assigning documents automatically to predefined classes
- Examples:
    - Language (English vs. French)
    - Adult content
    - Region
    - Fake news
    - Jeopardy challenge
    - Multiple choice questions

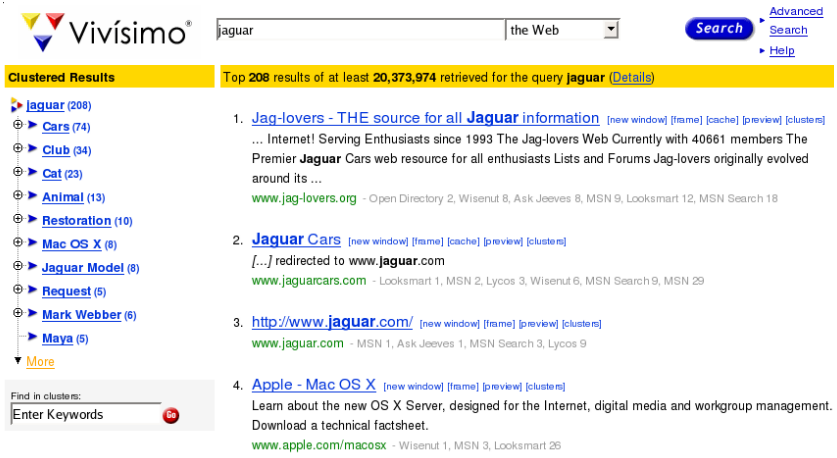
IIR 14: Vector classification



IIR 15: Learning to rank

- How to rank documents/pages when they are described by multiple pieces of information (features).
- Use machine learning to quantify the importance of each feature.
- Use this for your project if you are a grad student!

IIR 16: Flat clustering



Clustered Results

- jaguar (208)
  - Cars (74)
  - Club (34)
  - Cat (23)
  - Animal (13)
  - Restoration (10)
  - Mac OS X (8)
  - Jaguar Model (8)
  - Request (5)
  - Mark Webber (6)
  - Maya (5)
- More

Top **208** results of at least **20,373,974** retrieved for the query **jaguar** (Details)

1. Jag-lovers - THE source for all **Jaguar** information [new window] [frame] [cache] [preview] [clusters]
   ... Internet! Serving Enthusiasts since 1993 The Jag-lovers Web Currently with 40661 members The Premier **Jaguar** Cars web resource for all enthusiasts Lists and Forums Jag-lovers originally evolved around its ...
   www.jag-lovers.org - Open Directory 2, Wisenut 8, Ask Jeeves 8, MSN 9, Looksmart 12, MSN Search 18

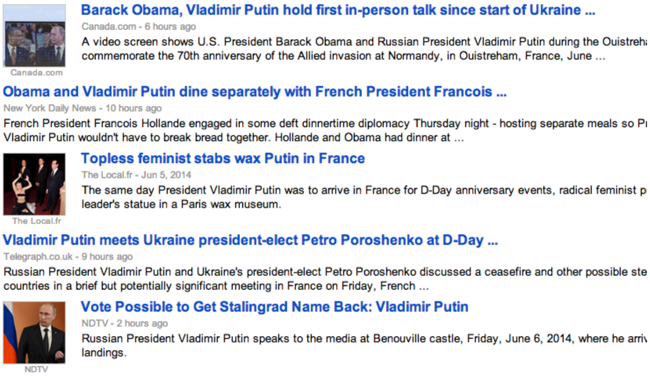
2. **Jaguar** Cars [new window] [frame] [cache] [preview] [clusters]
   [...] redirected to www.**jaguar**.com
   www.jaguarcars.com - Looksmart 1, MSN 2, Lycos 3, Wisenut 6, MSN Search 9, MSN 29

3. http://www.**jaguar**.com/ [new window] [frame] [preview] [clusters]
   www.jaguar.com - MSN 1, Ask Jeeves 1, MSN Search 3, Lycos 9

4. Apple - Mac OS X [new window] [frame] [preview] [clusters]
   Learn about the new OS X Server, designed for the Internet, digital media and workgroup management. Download a technical factsheet.
   www.apple.com/macosx - Wisenut 1, MSN 3, Looksmart 26

IIR 17: Hierarchical clustering

http://news.google.com



Barack Obama, Vladimir Putin hold first in-person talk since start of Ukraine ...
Canada.com - 6 hours ago
A video screen shows U.S. President Barack Obama and Russian President Vladimir Putin during the Ouistreh
commemorate the 70th anniversary of the Allied invasion at Normandy, in Ouistreham, France, June ...

Obama and Vladimir Putin dine separately with French President Francois ...
New York Daily News - 10 hours ago
French President Francois Hollande engaged in some deft dinnertime diplomacy Thursday night - hosting separate meals so P
Vladimir Putin wouldn't have to break bread together. Hollande and Obama had dinner at ...

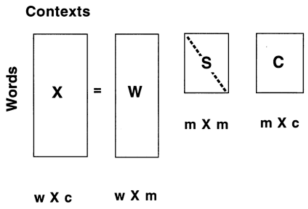
Topless feminist stabs wax Putin in France
The Local.fr - Jun 5, 2014
The same day President Vladimir Putin was to arrive in France for D-Day anniversary events, radical feminist p
leader's statue in a Paris wax museum.

Vladimir Putin meets Ukraine president-elect Petro Poroshenko at D-Day ...
Telegraph.co.uk - 9 hours ago
Russian President Vladimir Putin and Ukraine's president-elect Petro Poroshenko discussed a ceasefire and other possible ste
countries in a brief but potentially significant meeting in France on Friday, French ...

Vote Possible to Get Stalingrad Name Back: Vladimir Putin
NDTV - 2 hours ago
Russian President Vladimir Putin speaks to the media at Benouville castle, Friday, June 6, 2014, where he arriv
landings.

IIR 18: Latent Semantic Indexing

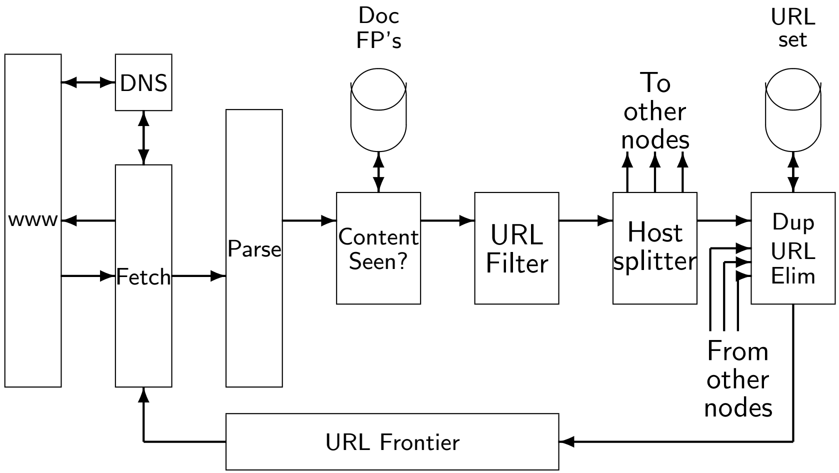
“You shall know a word by the company it keeps.”



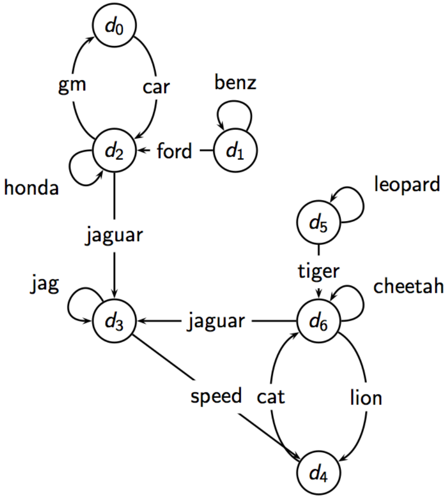
IIR 19: The web and its challenges

- Unusual and diverse documents
- Unusual and diverse users and information needs
- Beyond terms and text: exploit link analysis, user data
- How do web search engines work?
- How can we make them better?

IIR 20: Crawling



IIR 21: Link analysis / PageRank



Take-away

- Why you should take this course
- Admin issues
- Boolean Retrieval: Design and data structures of a simple information retrieval system
- What topics will be covered in this class?