Introduction to Information Retrieval and Text Mining
Lecture 01: Introduction and Boolean Retrieval

Roman Klinger

Institute for Natural Language Processing, University of Stuttgart
(this lecture builds on top of `http://informationretrieval.org/`)

2021-10-21

Notes

- This class is recorded (my voice and slide projection). Nobody is visible in the videos and you, the audience, is hardly audible in the recording.
- This class is in English. Feel free to ask questions in German.
- Please ask me to repeat something in German if something is unclear in English.

Take-away

- Boolean Retrieval: Design and data structures of a simple information retrieval system
- Formalities
- What topics will be covered in this class?

Outline

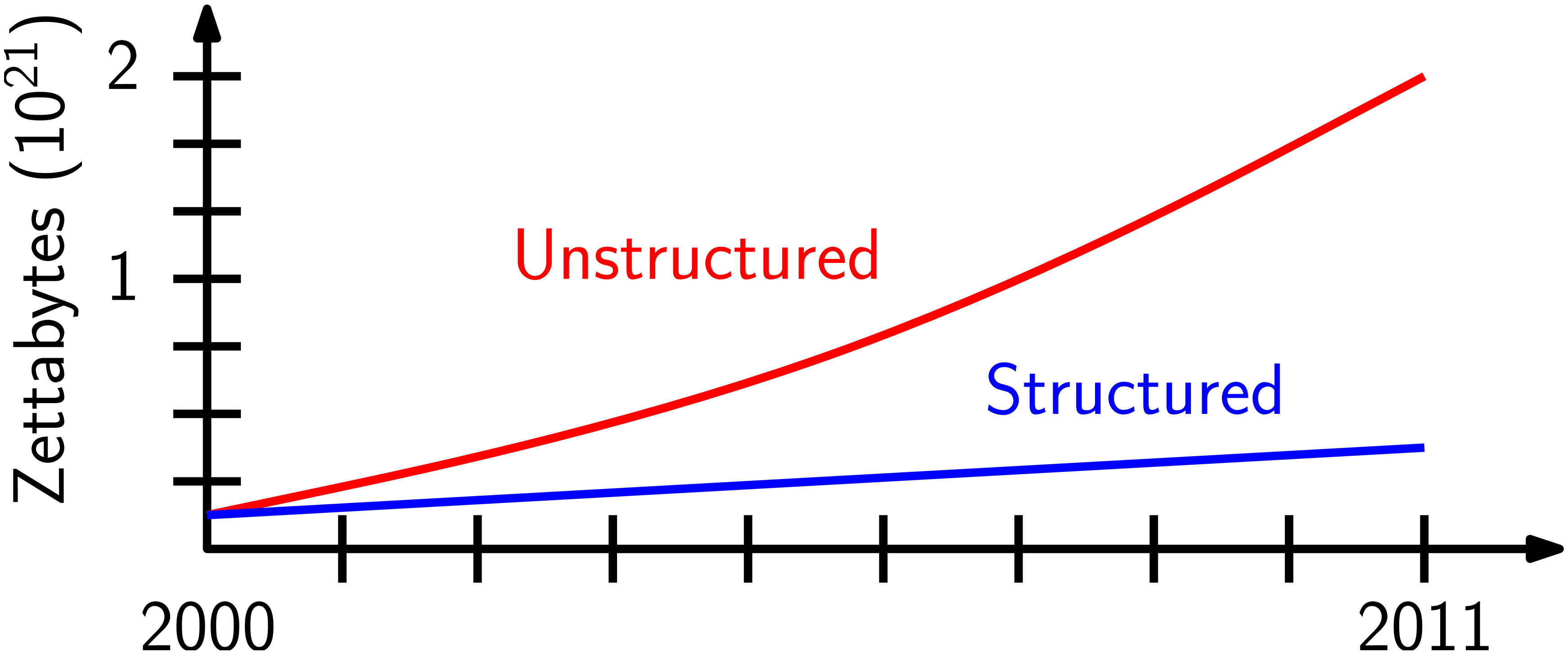
Definition of *information retrieval*

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 *text mining*

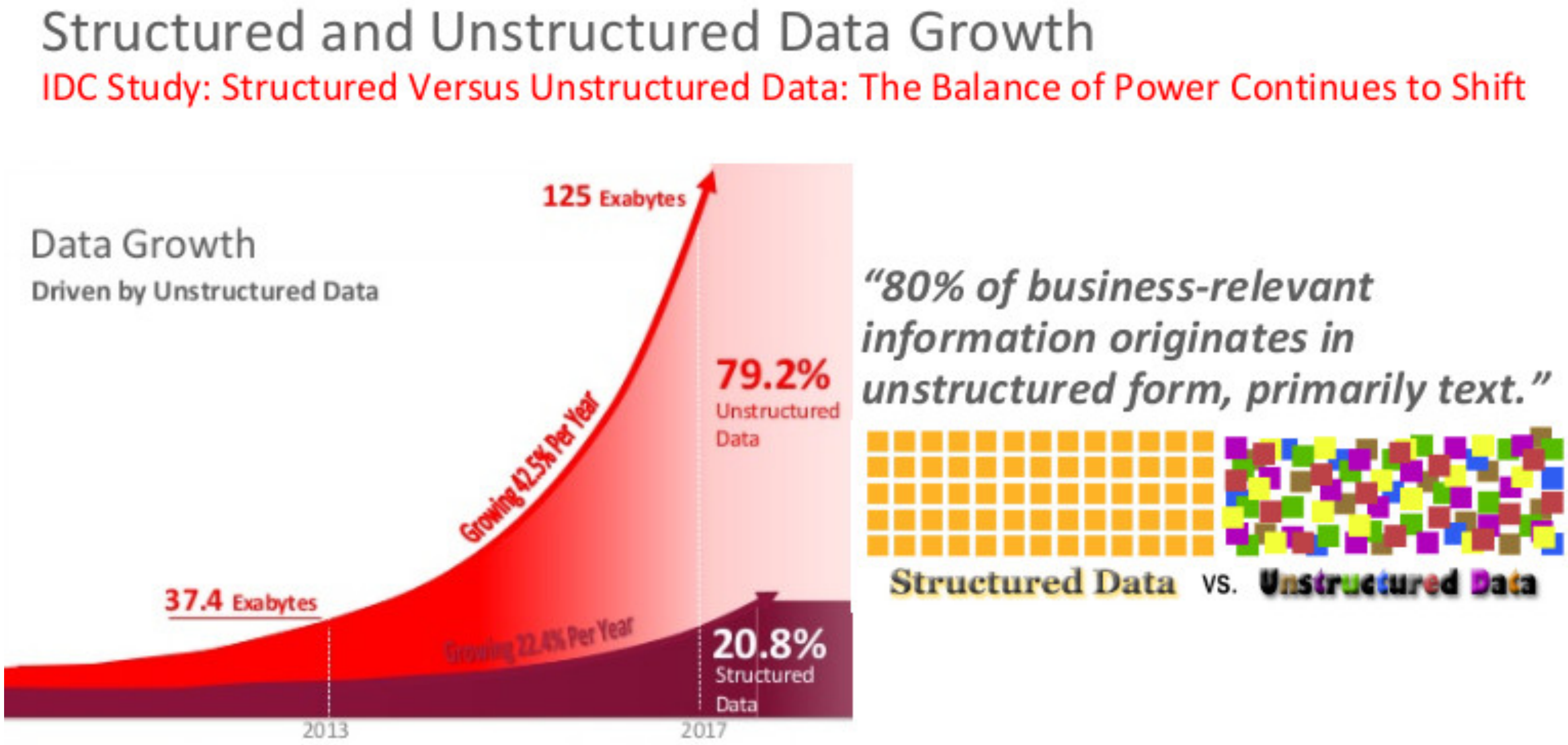
Text Mining is the derivation of information (usually in structured form) from unstructured text. In contrast, data mining is typically applied on structured data. Typically, methods from information retrieval are used.

Structured vs. Unstructured data (I)



data from `http://www.couchbase.com/why-nosql/nosql-database`

Use Case at Oracle



https://www.linkedin.com/pulse/got-analytics-machine-learning-cloud-yes-oracle-advanced-lefranc/

Why is there more unstructured (textual) data in contrast to structured databases?

What do you think? Why is this the case?

multiple sources of data

↳ no need to structure

text is a natural form of communication

text is more accessible

structuring data is costly

Structured vs. Unstructured data (II)

Structured data

- Complex queries possible
- Easier to understand
- Structure often makes assumptions of use case
- Examples: Relational databases
- ...

Unstructured data

- Most knowledge is only available in unstructured form
- Data often comes this way
  (scientific articles, blog posts, Facebook, Twitter, Images)

Boolean retrieval (from unstructured text)

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

anchor text

Does Google use the Boolean model?

synonyms

popular documents

Does Google use the Boolean model?

- On Google, the default interpretation of a query

  is w_1 AND w_2 AND . . . 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.
- Most Boolean engines rank the result set – they rank good hits (according to some estimator of relevance) higher than worse 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.

```
grep "Brutus" inputdata.txt \
| grep "Caesar" | grep -v "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

documents

terms

	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:
  - Take the vectors for BRUTUS, CAESAR, and CALPURNIA
  - Complement the vector of CALPURNIA
  - Do a (bitwise) AND

1	1	0	1	0	0
1	1	0	1	1	1
1	0	1	1	1	1
1	0	0	1	0	0

0/1 vector

	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
  $\Rightarrow$ 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 representation?
  - We only record the 1s.

Inverted Index

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

BRUTUS	$\longrightarrow$	1	2	4	11	31	45	173	174	
CAESAR	$\longrightarrow$	1	2	4	5	6	16	57	132	...
CALPURNIA	$\longrightarrow$	2	31	54	101					
⋮										

$\underbrace{\qquad\qquad}_{\textbf{dictionary}}$ $\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{\textbf{postings}}$

Inverted index construction

1. Collect the documents to be indexed:
   | Friends, Romans, countrymen. | So let it be with Caesar | ... |
2. Tokenize the text, turning each document into a list of tokens:
   | Friends | Romans | countrymen | So | ... |
3. Do linguistic preprocessing, producing a list of normalized tokens, which are the indexing terms:
   | friend | roman | countryman | 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		term	docID
i	1		ambitious	2
did	1		be	2
enact	1		brutus	1
julius	1		brutus	2
caesar	1		capitol	1
i	1		caesar	1
was	1		caesar	2
killed	1		caesar	2
i'	1		did	1
the	1		enact	1
capitol	1		hath	1
brutus	1		i	1
killed	1		i	1
me	1	$\Longrightarrow$	i'	1
so	2		it	2
let	2		julius	1
it	2		killed	1
be	2		killed	1
with	2		let	2
caesar	2		me	1
the	2		noble	2
noble	2		so	2
brutus	2		the	1
hath	2		the	2
told	2		told	2
you	2		you	2
caesar	2		was	1
was	2		was	2
ambitious	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

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

Split the result into dictionary and postings file

BRUTUS	8	$\longrightarrow$	1	2	4	11	31	45	173	174	
CAESAR	15	$\longrightarrow$	1	2	4	5	6	16	57	132	...
CALPURNIA	4	$\longrightarrow$	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

BRUTUS	$\longrightarrow$	$\boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$
CALPURNIA	$\longrightarrow$	$\boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$
Intersection	$\Longrightarrow$	

Intersecting two postings lists

BRUTUS	$\longrightarrow$	$\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$
CALPURNIA	$\longrightarrow$	$\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$
Intersection	$\Longrightarrow$	

Intersecting two postings lists

BRUTUS	$\longrightarrow$	$\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$
CALPURNIA	$\longrightarrow$	$\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$
Intersection	$\Longrightarrow$	

Intersecting two postings lists

BRUTUS	$\longrightarrow$	1 → 2 → 4 → 11 → 31 → 45 → 173 → 174
CALPURNIA	$\longrightarrow$	2 → 31 → 54 → 101
Intersection	$\Longrightarrow$	2

Intersecting two postings lists

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

CALPURNIA $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

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

Intersecting two postings lists

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

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

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

Intersecting two postings lists

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

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

Intersection $\Longrightarrow \boxed{2}$

Intersecting two postings lists

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

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

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

Intersecting two postings lists

BRUTUS ⟶ 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

CALPURNIA ⟶ 2 → 31 → 54 → 101

Intersection ⟹ 2 → 31

Intersecting two postings lists

BRUTUS ⟶ 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

CALPURNIA ⟶ 2 → 31 → 54 → 101

Intersection ⟹ 2 → 31

Intersecting two postings lists

BRUTUS ⟶ 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

CALPURNIA ⟶ 2 → 31 → 54 → 101

Intersection ⟹ 2 → 31

Intersecting two postings lists

BRUTUS	$\longrightarrow$	$\boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$
CALPURNIA	$\longrightarrow$	$\boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$
Intersection	$\Longrightarrow$	$\boxed{2} \to \boxed{31}$

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

Query processing: Exercise

FRANCE ⟶ 1 → 2 → 3 → 4 → 5 → 7 → 8 → 9 → 11 → 12 → 13 → 14 → 15

PARIS ⟶ 2 → 6 → 10 → 12 → 14

LEAR ⟶ 12 → 15

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 at least 3 decades
- Many professional searchers (e.g., lawyers, patent offices,. . .) prefer Boolean queries.
  - You know exactly what you are getting.
- Many search systems you use are also Boolean: spotlight, email, intranet etc.

Example for 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.

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!

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)

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.)
- 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, . . .

PubMed, Example

- PubMed is the main search engine for life science articles
- The database with 27 Million abstracts is called MEDLINE (Oct 2021)

Give me all articles about spinal cord injury

```
( "spinal cord injuries"[MeSH Terms]  OR ( "spinal"[All Fields] AND
"cord"[All Fields] AND "injuries"[All Fields] ) OR "spinal cord
injuries"[All Fields]  OR ( "spinal"[All Fields] AND "cord"[All Fields]
AND "injury"[All Fields] ) OR "spinal cord injury"[All Fields] ) OR ( (
"brain"[MeSH Terms] OR "brain"[All Fields] )  AND ( "wound healing"[MeSH
Terms] OR ( "wound"[All Fields] AND "healing"[All Fields] ) OR "wound
healing"[All Fields] OR "repair"[All Fields] ) ) OR ( "spinal cord
regeneration"[MeSH Terms] OR ("spinal"[All Fields] AND "cord"[All
Fields] AND "regeneration"[All Fields]) OR "spinal cord
regeneration"[All Fields] OR ("spinal"[All Fields] AND "cord"[All
Fields] AND "repair"[All Fields]) OR "spinal cord repair"[All Fields] )
OR "brain injuries"[MeSH Terms] OR ( "brain"[All Fields] AND
"injuries"[All Fields] ) OR "brain injuries"[All Fields] OR (
"brain"[All Fields] AND "injury"[All Fields] ) OR "brain injury"[All
Fields]
```

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

BRUTUS ⟶ 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

CALPURNIA ⟶ 2 → 31 → 54 → 101

CAESAR ⟶ 5 → 31

Optimized intersection algorithm for conjunctive queries

$\textsc{Intersect}(\langle t_1, \ldots, t_n \rangle)$

$\quad$ *terms* $\leftarrow \textsc{SortByIncreasingFrequency}(\langle t_1, \ldots, t_n \rangle)$

$\quad$ *result* $\leftarrow$ *postings*(*first*(*terms*))

$\quad$ *terms* $\leftarrow$ *rest*(*terms*)

$\quad$ **while** *terms* $\neq$ NIL and *result* $\neq$ NIL

$\quad$ **do** *result* $\leftarrow \textsc{Intersect}$(*result*, *postings*(*first*(*terms*)))

$\qquad$ *terms* $\leftarrow$ *rest*(*terms*)

$\quad$ **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)

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

Formalities

- Lecture takes place Tuesdays and Thursdays
  - In V55.01, videos will be uploaded in the evening, latest the morning of the next day.
  - There will be a written exam (date organized by examination office)
  - There will be 5 home work assignments.
  - (Nearly) All students need to register for two exams:
    - Written exam (graded)
    - Assignments (ungraded)
    - Otherwise: Module not achieved.
  - Prerequisite for exam are 80 points in assignments.
  - You can pass the assignments and fail the exam.
    - No need to do the assignments again.

Attendance on Campus

- You can (but don't need to) participate in the lecture hall when you are listed in the respective group on CAMPUS.
- Currently, each group can participate between 4 and 6 times on campus.
- Move yourself to another group, if you wish (and the capacity=50 allows)
- Please also move yourself from the groups "Ungerade/Gerade" into the standard group (capacity=∞) if you don't want to participate in the lecture hall. This will allow others to come to campus.
- When I see that there is typically a low number of people in the room, I will combine groups.

Assignments

- More on assignments:
  - You can reach 100 points in pen & paper exercises (points granted for correct/serious submissions, partial points granted).
  - You can reach 50 points in practical exercises (points granted for submission of correct and well documented code/result).
  - That means: You can choose between different tasks!
- Working in groups of up to 3 people is encouraged.
  - We accept submissions of groups with maximally 4 people.
  - We never accept larger groups. Don't ask.
  - Who joins the class too late for completing assignment 2 (with prove for not being responsible for that) will have the chance to do an extra task of 30 points.
- Changing groups during term is possible but not encouraged.
- Pen/paper assignment will be discussed in online exercise sessions.
- Programming assignments are not discussed.

Homepage and Contact

Lecturer: Roman Klinger

- roman.klinger@ims.uni-stuttgart.de
- Office: PWR5b, 01.007, office hours on appointment, short meetings when door is open, and on WebEx Teams when not in DND mode.

Teaching Assistants:
Valentino Sabbatino, Maximilian Kuhn, Patrick Bareiß

- Available for exercise discussions or questions on appointment
- Manage the exercises and offer discussion sessions

Question on the topic? Want a meeting?
Question on assignment correction?

- Contact us at irtm-teachers@ims.uni-stuttgart.de Please also use the forum for anything that you might want to know (we check this nearly daily).

Material

Main material:

- Please register with the Ilias class (via CAMPUS), slides and assignments will be shared there.
- Lectures are published on Ilias.
- Publication/distribution of content is not allowed.

Additional material:

- This class is mostly build on top of Manning/Raghavan/Schütze: Introduction to Information Retrieval (`http://informationretrieval.org`)
- Another nice book is from Baeza-Yates: Modern Information Retrieval (2010)
- Some lectures use additional material

Outline

Course overview

- This lecture follows the concept of Hinrich Schütze's class and his book "Introduction to Information Retrieval", with some changes
- 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 + a bit more text mining

Schedule 1 (preliminary)

IRTM 21/22 Schedule

	Date	Session	TOPIC	Assignments	Group
DO	21.10.2021	1	Introduction and Boolean Retrieval		Gerade 2
DI	26.10.2021	2	Term Vocabularies and Postings Lists		Ungerade 1
DO	28.10.2021	3	Dictionaries and Tolerant Retrieval		Ungerade 2
DI	02.11.2021	4	Spelling	Publication Assignment 1	Gerade 1
DO	04.11.2021	5	Index Construction		Gerade 2
DI	09.11.2021	6	Compression	Deadline Assignment 1	Ungerade 1
DO	11.11.2021	7	Scoring	Publication Assignment 2	Ungerade 2
DI	16.11.2021		DISCUSSION ASSIGMENT 1		Online Only
DO	18.11.2021	8	Ranking		Gerade 2
DI	23.11.2021	9	System, Summaries, Intro to Evaluation		Ungerade 1
DO	25.11.2021	10	Evaluation, IA Agreement	Deadline Assignment 2	Ungerade 2
DI	30.11.2021	11	Query Expansion, Probabilistic Retrieval, Lang Models	Publication Assignment 3	Gerade 1
DO	02.12.2021		DISCUSSION ASSIGNMENT 2		Online Only
DI	7.12.2021	12	LM, Text Classification		Ungerade 1
DO	09.12.2021	13	TC, NB		Ungerade 2
DI	14.12.2021	14	NB, Evaluation, MaxEnt	Deadline Assignment 3	Gerade 1
DO	16.12.2021	15	Feature Selection, Vector Space Classification, Perceptron	Publication Assignment 4	Gerade 2
DI	21.12.2021		DISCUSSION ASSIGNMENT 3		Online Only
DI	11.01.2022	16	Support Vector Machines, Learning to Rank		Gerade 1
DO	13.01.2022	17	Representation Learning and Deep Learning for TC		Gerade 2
DI	18.01.2022	18	Introduction to Clustering	Deadline Assignment 4	Ungerade 1
DO	20.01.2022	19	Evaluation of Clustering, Hierarchical Clustering	Publication Assignment 5	Ungerade 2
DI	25.01.2022		DISCUSSION ASSIGNMENT 4		Online Only
DO	27.01.2022	20	Hierarch. Clustering 2		Gerade 2
DI	01.02.2022	21	Clustering 3, Link Analysis	Deadline Assignment 5	Ungerade 1
DO	03.02.2022	22	Link Analysis, Web, Crawling		Ungerade 2
DI	08.02.2022		DISCUSSION ASSIGNMENT 5		Online Only
DO	10.02.2022		Question Session		Online Only

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.

Dictionaries and tolerant retrieval

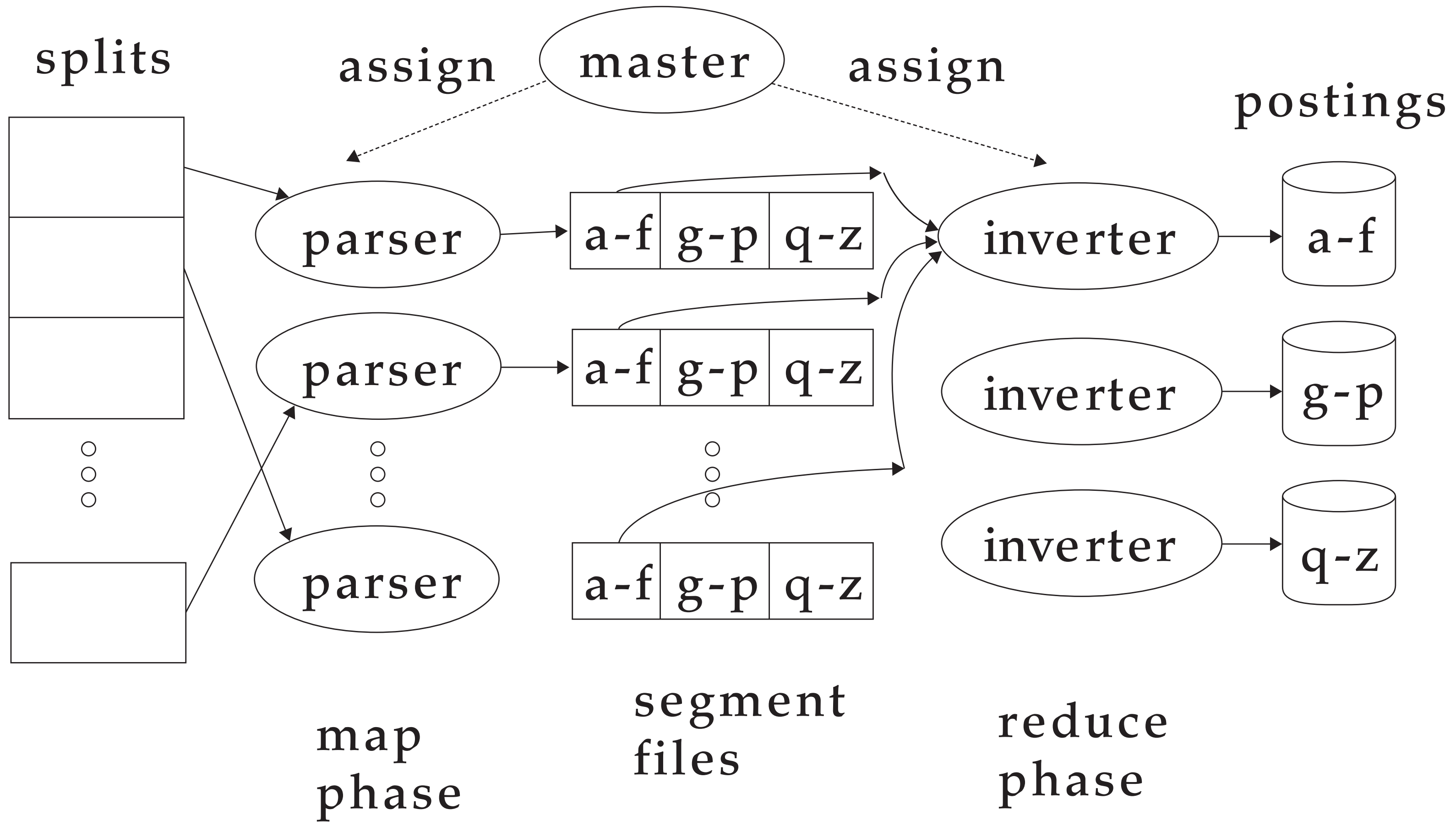
BORD

BO → aboard → about → boardroom → border

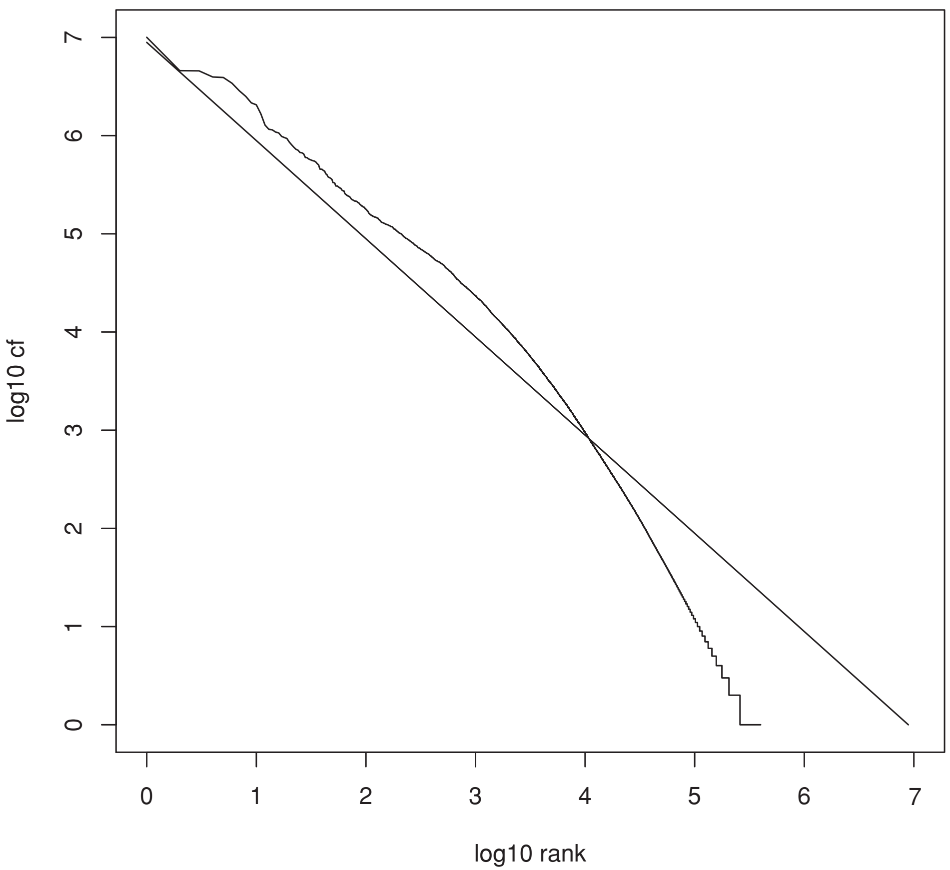
OR → border → lord → morbid → sordid

RD → aboard → ardent → boardroom → border

Index construction



Index compression

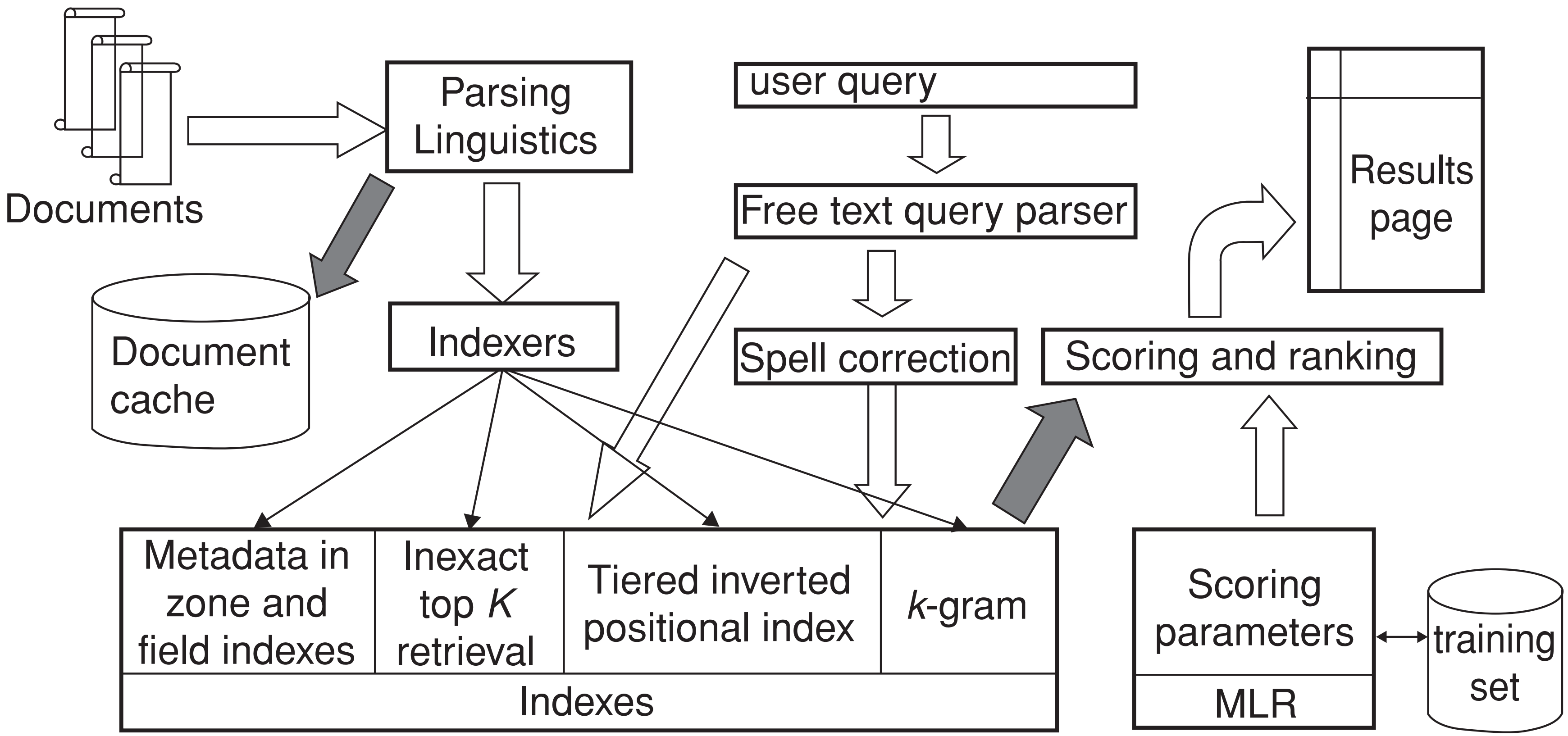


Zipf's law

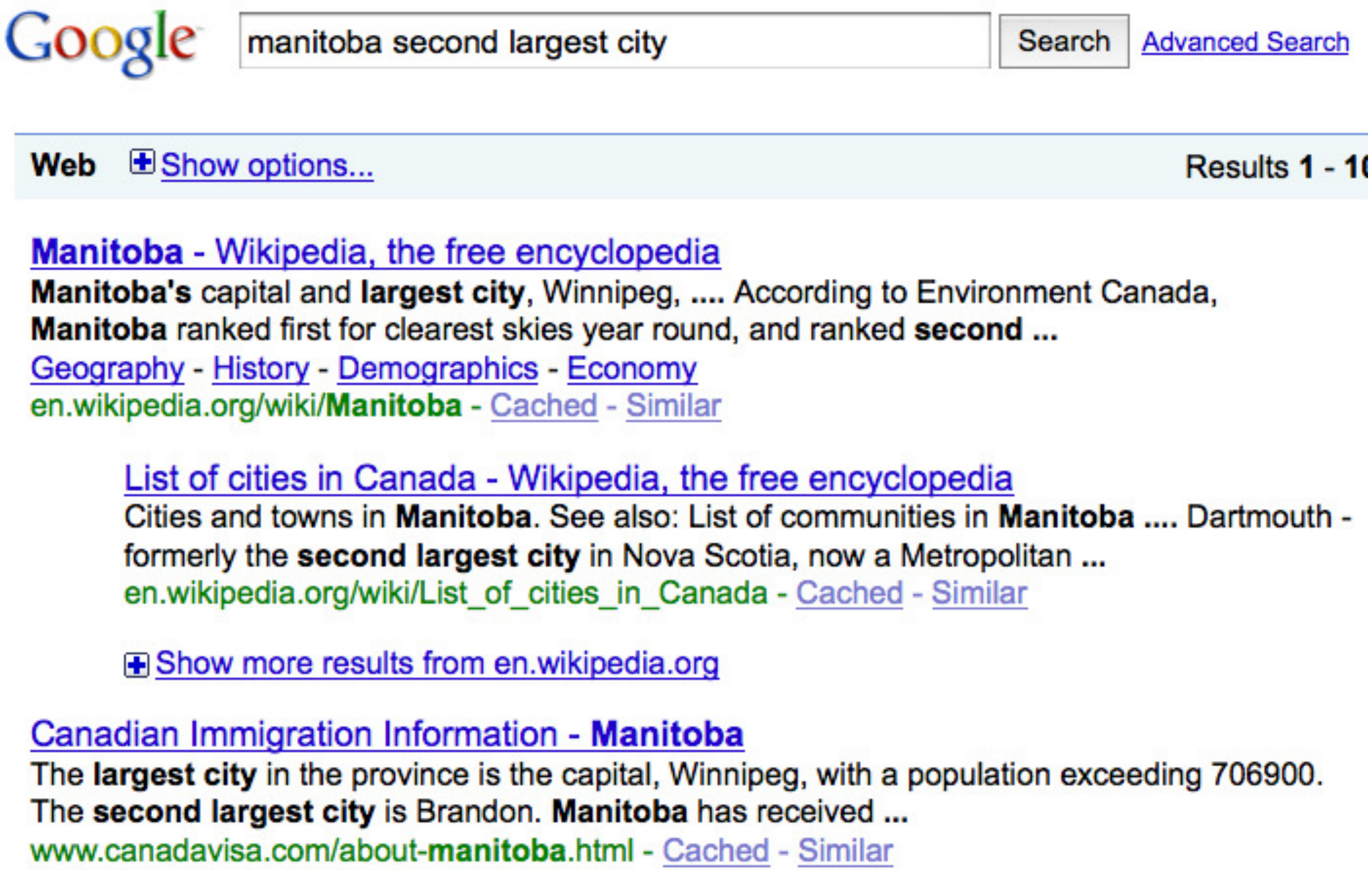
Scoring, term weighting and the vector space model

- 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

Ranking in a complete search system



Manual Corpus Annotation
Evaluation and dynamic summaries



Probabilistic information retrieval

	document	relevant ($R=1$)	nonrelevant ($R=0$)
Term present	$x_t = 1$	p_t	u_t
Term absent	$x_t = 0$	$1-p_t$	$1-u_t$

$$O(R|\vec{q},\vec{x}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0,q_t=1} \frac{1-p_t}{1-u_t} \tag{1}$$

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.

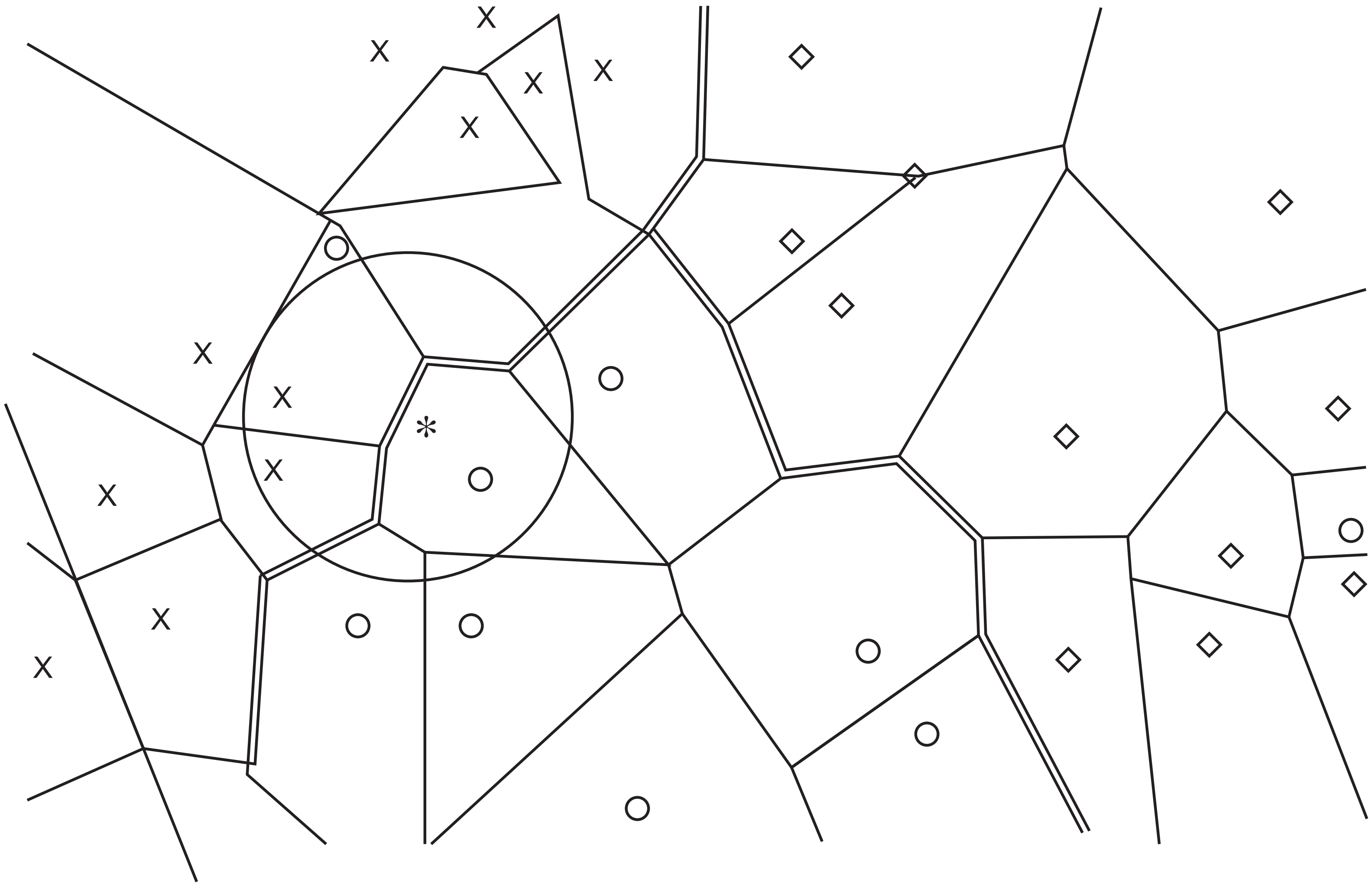
frog said that toad likes frog STOP

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

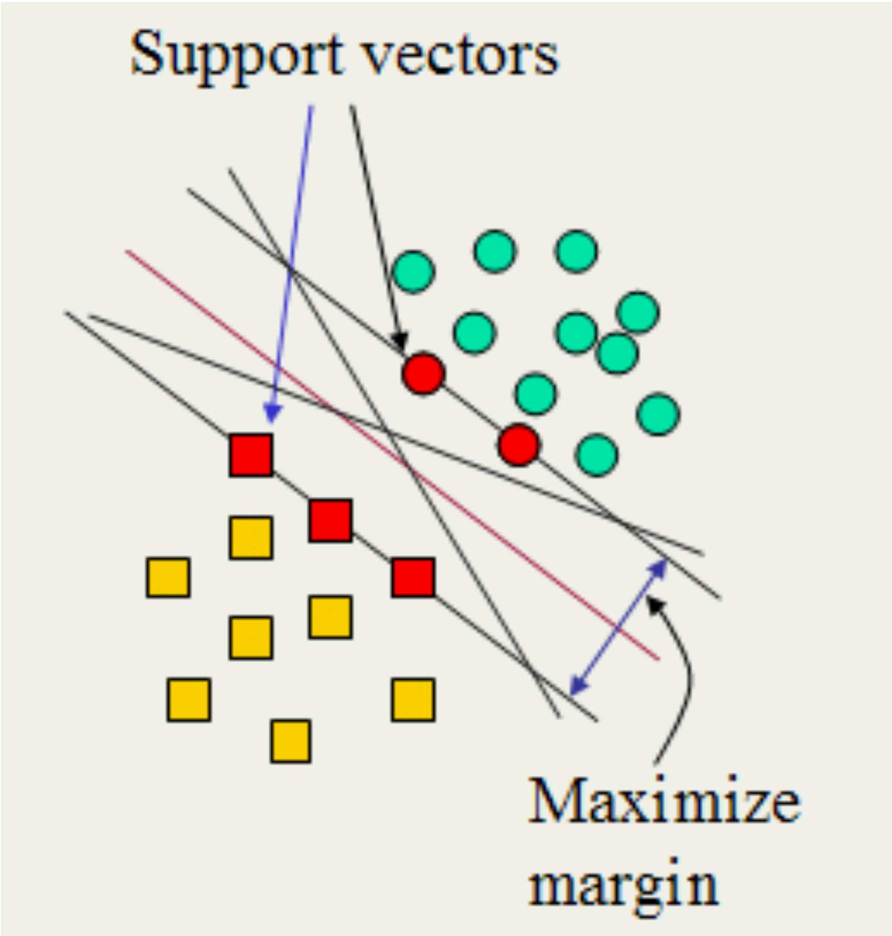
Text classification & Naive Bayes, MaxEnt

- Text classification = assigning documents automatically to predefined classes
- Examples:
  - Language (English vs. French)
  - Adult content
  - Region

Vector classification



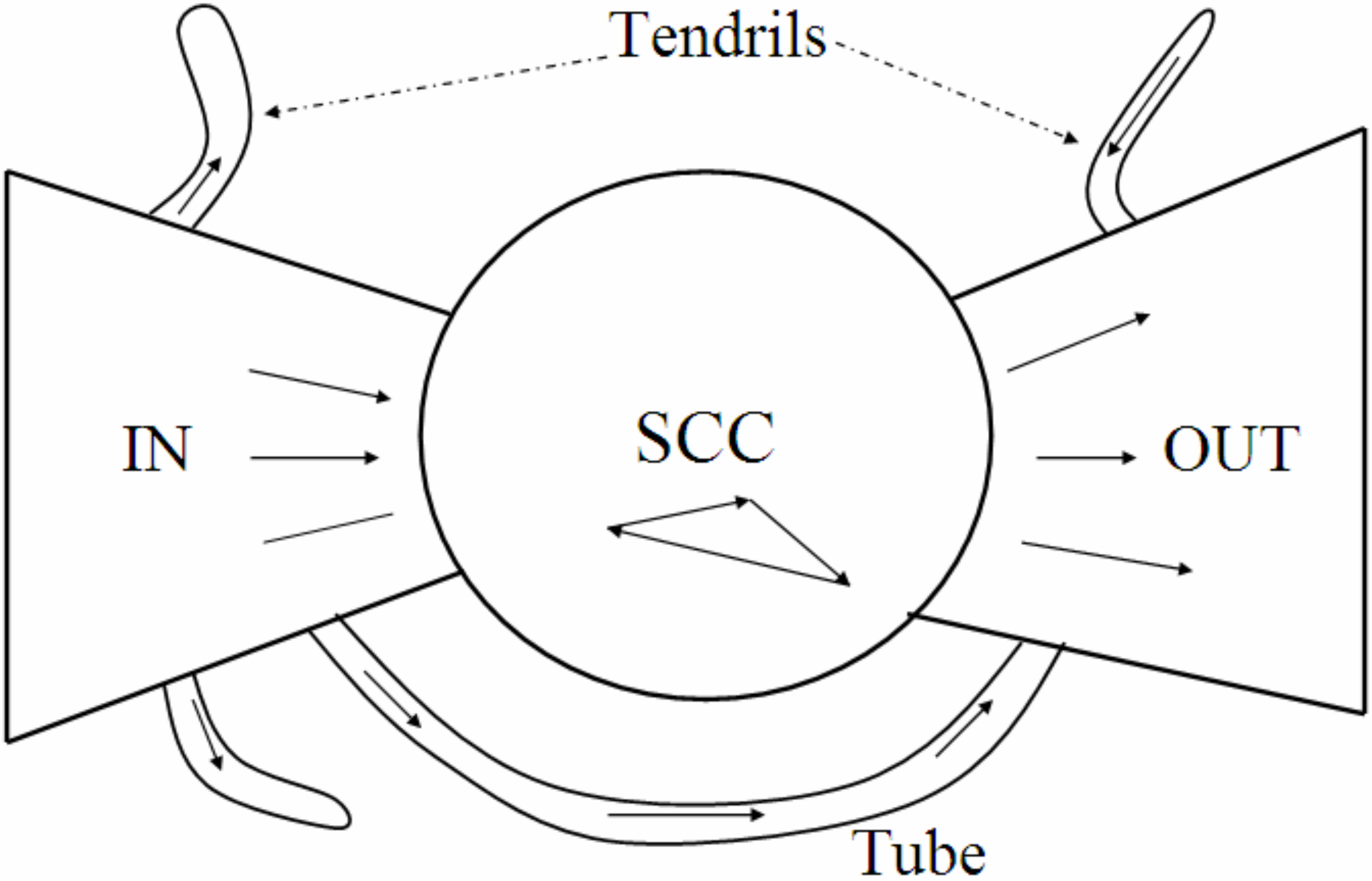
Support vector machines



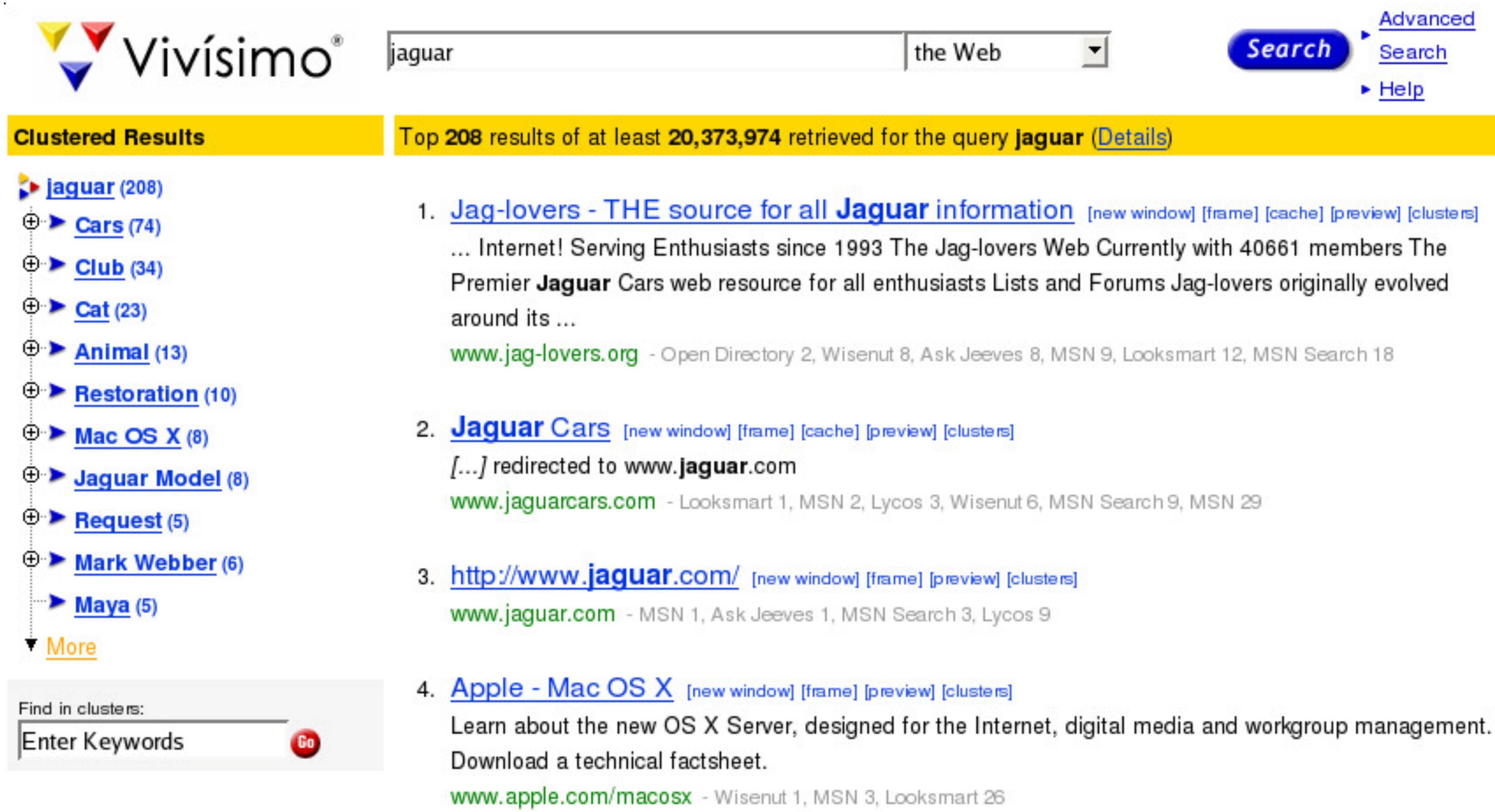
Feature Selection

- Which dimensions should really be taken into account?
- Which are confusing the classifier?

Link analysis / PageRank / Learning to Rank



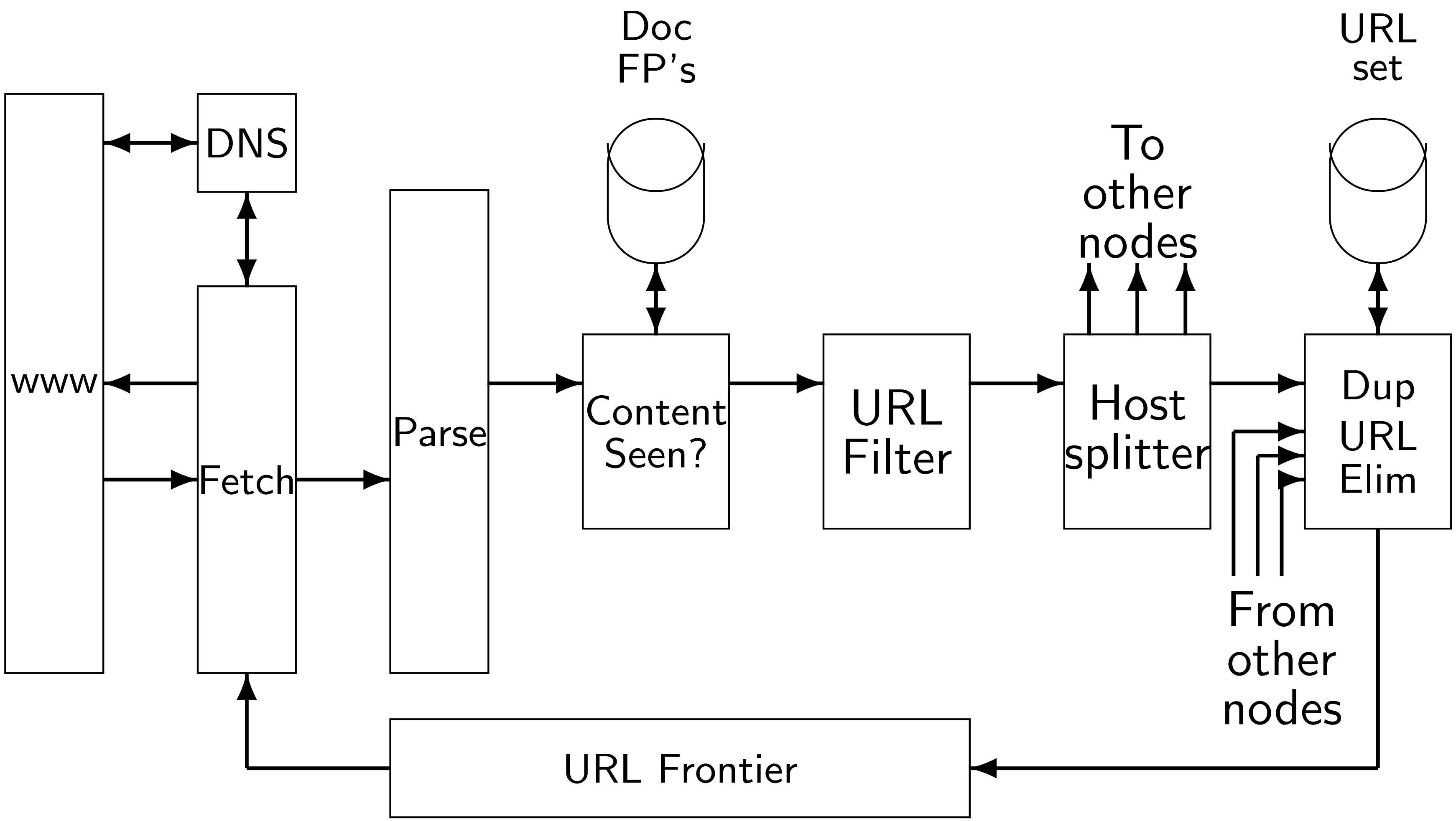
clustering


Vivísimo
jaguar
the Web
Search
Advanced Search
Help
Clustered Results
Top 208 results of at least 20,373,974 retrieved for the query jaguar (Details)
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
Find in clusters:
Enter Keywords
Go
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

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?

Crawling


Doc
FP's
URL
set
DNS
To
other
nodes
www
Parse
Content
Seen?
URL
Filter
Host
splitter
Dup
URL
Elim
Fetch
From
other
nodes
URL Frontier

Take-away

- Boolean Retrieval: Design and data structures of a simple information retrieval system
- Formalities
- What topics will be covered in this class?