PV211: Introduction to Information Retrieval
https://www.fi.muni.cz/~sojka/PV211

IIR 1: Boolean Retrieval
Handout version

Petr Sojka, Hinrich Schütze et al.

Faculty of Informatics, Masaryk University, Brno
Center for Information and Language Processing, University of Munich

2019-02-21

Take-away

- Basic information about the course, teachers, evaluation, exercises
- Boolean Retrieval: Design and data structures of a simple information retrieval system
- What topics will be covered in this class (overview)?

Overview

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).

Prerequisites

Curiosity about how Information Retrieval works.
But seriously:

- Chapters 1–5 benefit from basic course on algorithms and data structures.
- Chapters 6–7 need in addition linear algebra, vectors and dot products.
- For Chapters 11–13 basic probability notions are needed.
- Chapters 18–21 demand course in linear algebra, notions of matrix rank, eigenvalues and eigenvectors.

Active learning features in PV211

- Student activities *explicitly welcomed* and built as part of classification system (10 pts).
- Mentoring rather than 'ex cathedra' lectures: "The *flipped classroom* is a pedagogical model in which the typical lecture and homework elements of a course are reversed."
- Respect to individual learning speed and knowledge.
- Questions on PV211 IS discussion forum is welcomed especially *before* lectures.
- Richness of materials available in advance: MOOC (Massive open online course) becoming widespread, parts of IIR Stanford courses being available, together with other freely available teaching materials, including the whole IIR book.

Teachers

- Petr Sojka, `sojka@fi.muni.cz`
- Consulting hours Spring 2019:
  Thursday 13:00–14:00 and Friday 10:00–11:00
  or by appointment by email.
- Room C523 or C522 or A502, fifth floor, Botanická 68a.
- Course web page:
  `https://www.fi.muni.cz/~sojka/PV211/`
- TA: Vít Novotný, `witiko@mail.muni.cz`,
  Consulting hours: Fri 10:00–12:00 (A502) or by appointment
- TA: Dávid Lupták, `422640@mail.muni.cz`,
  Consulting hours: Fri 10:00–12:00 (A502) or by appointment

Evaluation of students

Classification system is based on points achieved (100 pts max). You could get 50 points during the term: 20 pts for each of 2 midterm tests, 10 pts for your activity during term (lectures or discussion forums,. . .) evaluated *subjectively* by teachers of the course, and 50 pts for the final test. Final written exam will consist of open exercises (30 pts, similar to midterm ones) and multiple choice questions (20 pts). *In addition*, one can get additional *premium* points based on activities during lectures, exercises (good answers) or negotiated related projects. Classification scale (*adjustments based on ECTS suggestions*) z/k[/E/D/C/B/A] corresponds ≈ 50/57/[64/71/78/85/92] points.

Dates of [final] exams will be announced via IS.muni.cz (at least three terms). There wiil be a possibility to make midterm tests on the first exam term for those ill.

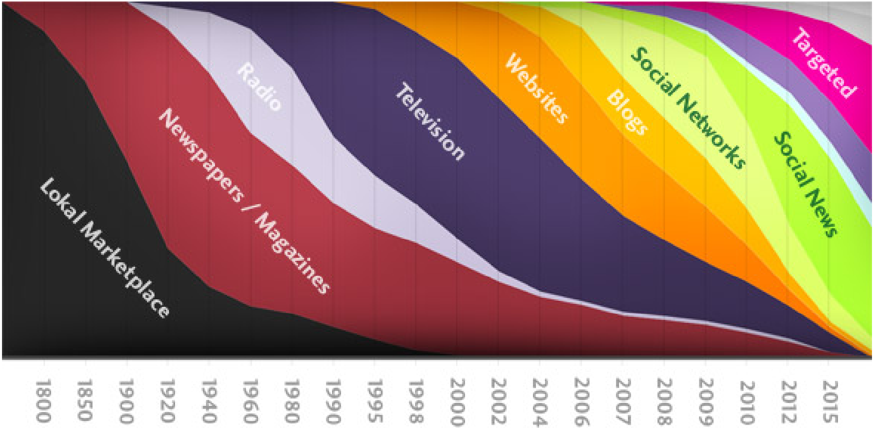
Questions?

Can we proceed [Y/N]?

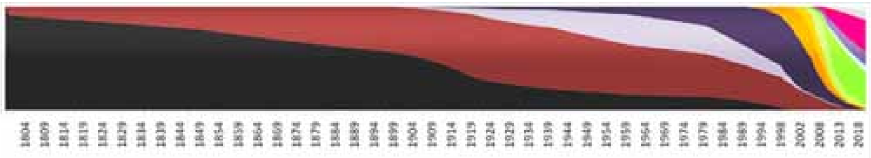
Questions?

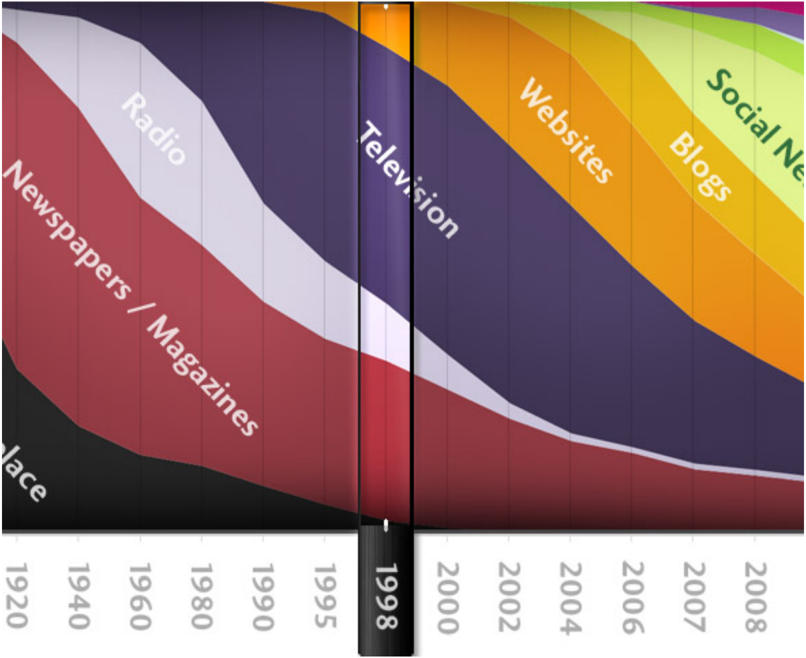
Presentation style? Warm ups? Personal cards.
Erasmus? Bc. or Mgr.? Discussion forum in IS!

History of *information retrieval*: gradual changes of channels



Gradual *speedup* of changes in IR





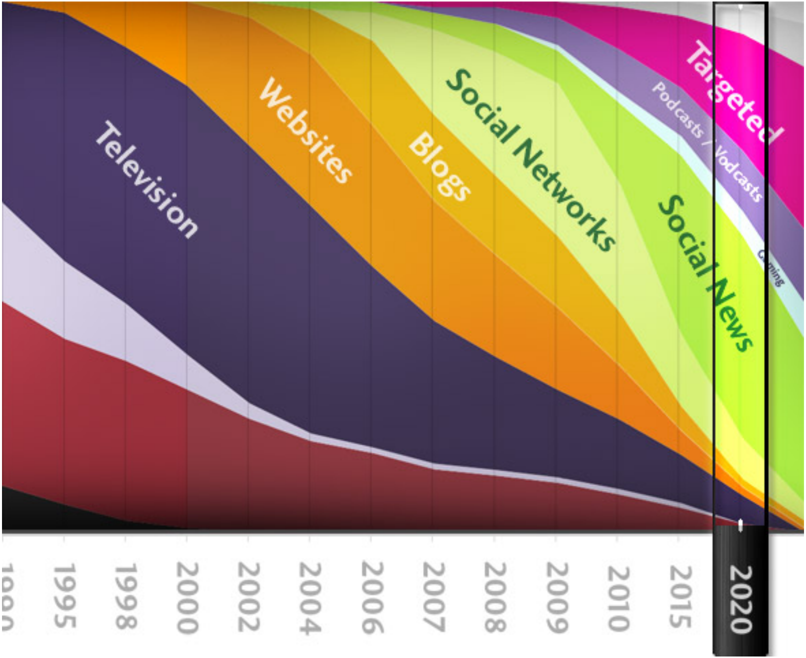
lace
Newspapers / Magazines
Radio
Television
Websites
Blogs
Social Ne
1920
1940
1960
1980
1990
1995
1998
2000
2002
2004
2006
2007
2008

"Google" Circa 1997 (google.stanford.edu)

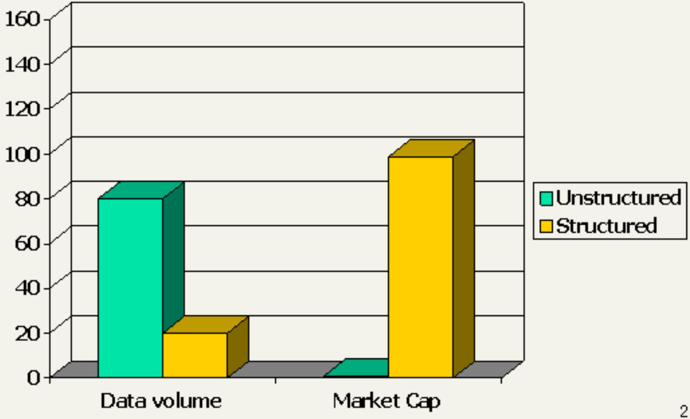
Google

1998: google.stanford.edu

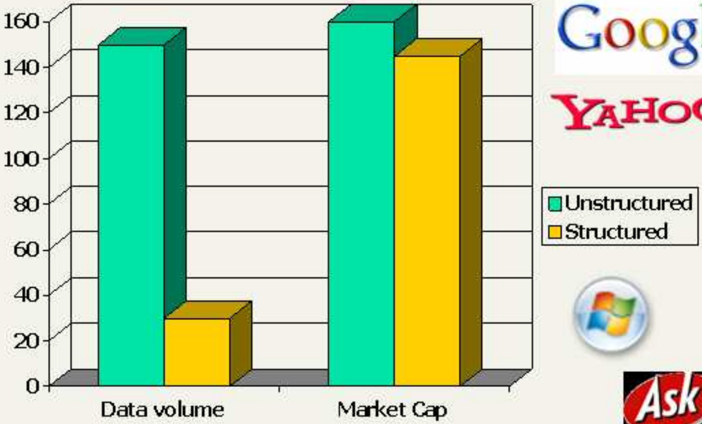
- ‘flipped IS’, collaborative project with Stanford faculty
- on collected disks
- Google 1998 ‘Anatomy paper’ (Page, Brin)


Television
Websites
Blogs
Social Networks
Social News
Targeted
Podcasts / Vodcasts
1995
1998
2000
2002
2004
2006
2007
2008
2009
2010
2015
2020

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



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



Boolean retrieval

- The Boolean model is arguably the simplest model to base an information retrieval system on.
- 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 $\dots w_n$] is w_1 AND w_2 AND $\dots$ 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.

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?
  - 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
  - Ranked retrieval (best documents to return) – focus of later lectures, but not this one

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 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!

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

$\underbrace{\hspace{3em}}_{\textbf{dictionary}}$ $\underbrace{\hspace{12em}}_{\textbf{postings file}}$

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?

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

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

Intersecting two postings lists

```
INTERSECT(p1, p2)
  answer ← ⟨ ⟩
  while p1 ≠ NIL and p2 ≠ NIL
  do if docID(p1) = docID(p2)
       then ADD(answer, docID(p1))
            p1 ← next(p1)
            p2 ← next(p2)
       else if docID(p1) < docID(p2)
               then p1 ← next(p1)
               else p2 ← next(p2)
  return answer
```

Query processing: Exercise

FRANCE $\longrightarrow$ $\boxed{1} \to \boxed{2} \to \boxed{3} \to \boxed{4} \to \boxed{5} \to \boxed{7} \to \boxed{8} \to \boxed{9} \to \boxed{11} \to \boxed{12} \to \boxed{13} \to \boxed{14} \to \boxed{15}$

PARIS $\longrightarrow$ $\boxed{2} \to \boxed{6} \to \boxed{10} \to \boxed{12} \to \boxed{14}$

LEAR $\longrightarrow$ $\boxed{12} \to \boxed{15}$

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 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.

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,. . .

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

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

CAESAR $\longrightarrow$ $\boxed{5} \rightarrow \boxed{31}$

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** $terms \neq \textsc{nil}$ and $result \neq \textsc{nil}$
$\quad$ **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)
- Process in increasing order of OR sizes

Exercise

Recommend a query processing order for: (TANGERINE OR TREES) AND (MARMALADE OR SKIES) AND (KALEIDOSCOPE OR EYES)

Course overview and agenda

- We are done with Chapter 1 of IIR (IIR 01).
- Plan for the rest of the semester: 16–18 of the 21 chapters of IIR
- In addition to experts from FI lectures by leading industry experts from Facebook (Tomáš Mikolov on March 12th as part of FI Informatics Colloquium), Seznam.cz (Vláďa Kadlec) or RaRe Technologies (Radim Řehůřek).
- In what follows: teasers for most chapters – to give you a sense of what will be covered.
- Last two or three lectures on IR topics researched in my research group MIR.fi.muni.cz and on state-of-the art achievements in the area (vector space embeddings etc.).

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.

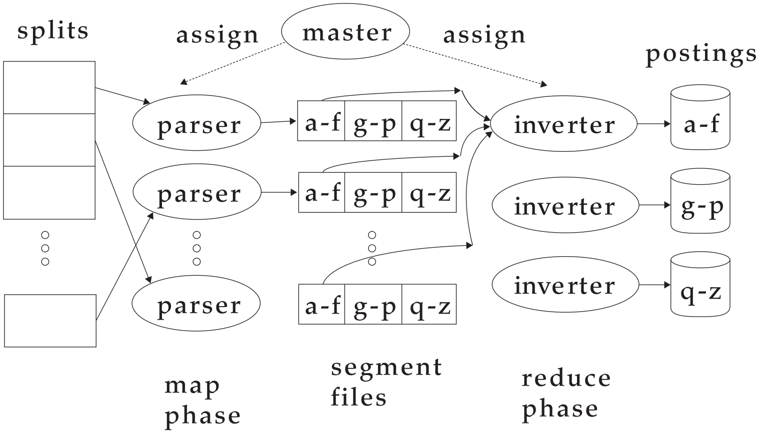
IIR 03: Dictionaries and tolerant retrieval

BO → aboard → about → boardroom → border

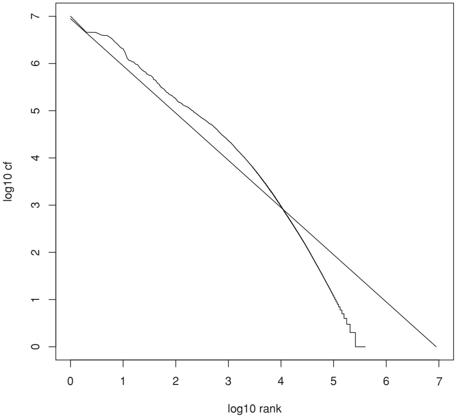
OR → border → lord → morbid → sordid

RD → aboard → ardent → boardroom → border

IIR 04: Index construction



IIR 05: Index compression

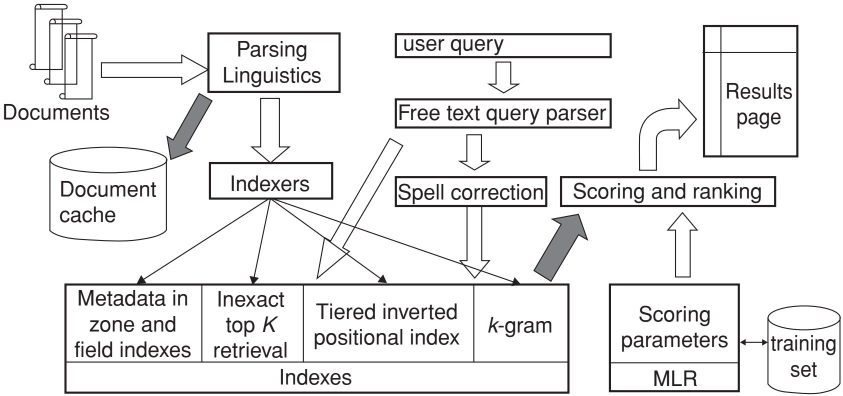


Zipf's law

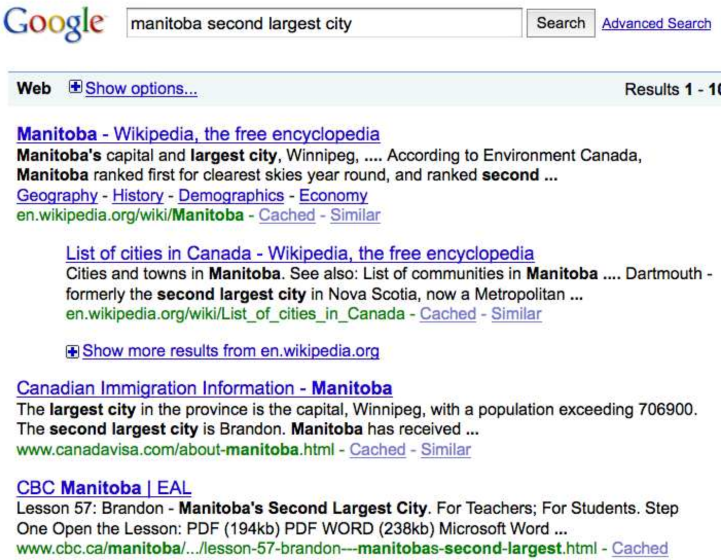
IIR 06: Scoring, term weighting and the vector space model

- 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 occurrence of a query term in the document
  - 3 vs. 2 occurrences 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

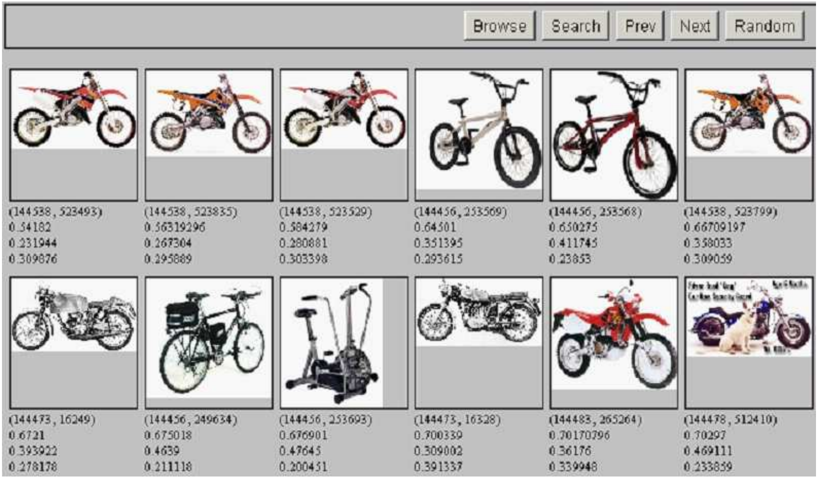
IIR 07: Scoring in a complete search system



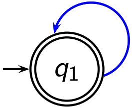
IIR 08: Evaluation and dynamic summaries



IIR 09: Relevance feedback & query expansion



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.

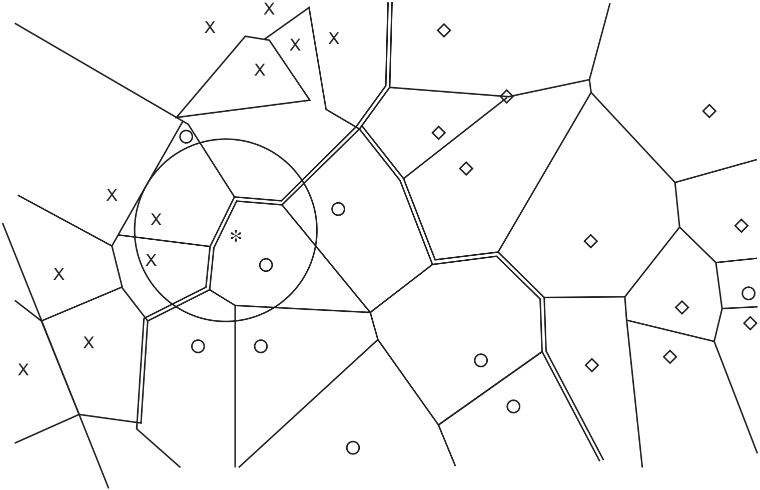
frog said that toad likes frog STOP

$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2$
$= 0.000000000048$

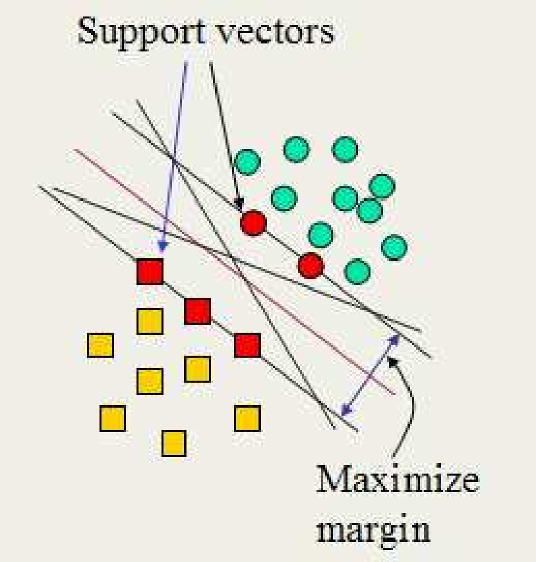
IIR 13: Text classification & Naive Bayes

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

IIR 14: Vector classification, kNN search



IIR 15: Support vector machines

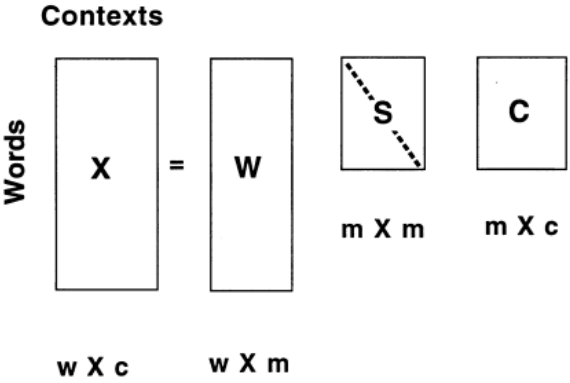


IIR 16: Flat clustering

IIR 17: Hierarchical clustering

`http://news.google.com`

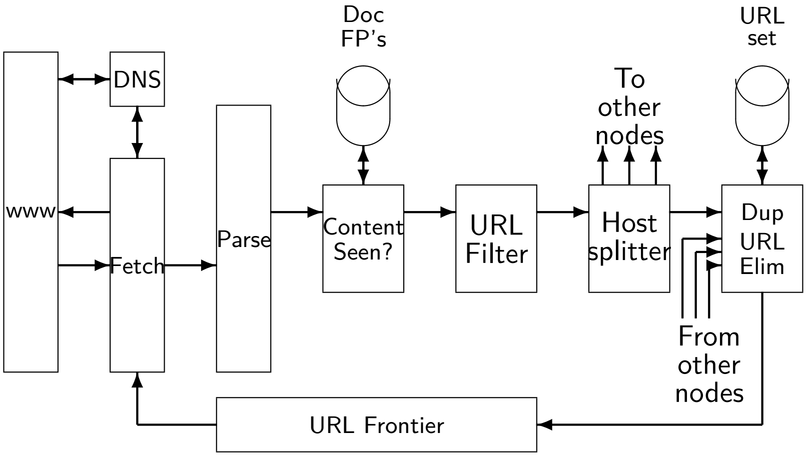
IIR 18: Latent Semantic Indexing



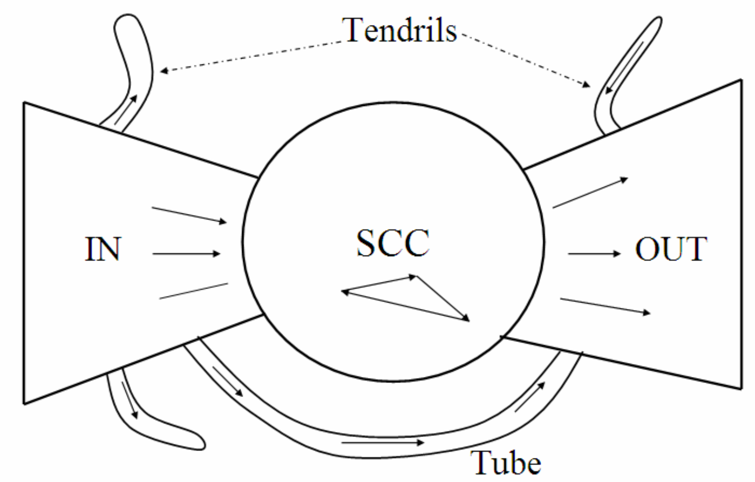
IIR 19: The web and its challenges

- 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



Invited lecture: Fulltext architecture in Seznam

Introduction to the Seznam.cz fulltext search architecture by Seznam research team lead (Vladimír Kadlec).

Abstract: The talk covers all basic web search engine blocks: crawling, indexing, query reformulation, relevance. Explanation of inner parts of the user interface such as: auto completer, query corrector, suggested searches. Real statistics from Seznam's traffic. As a bonus: Image/video search.

Take-away

- Basic information about the course, teachers, evaluation, exercises
- Boolean Retrieval: Design and data structures of a simple information retrieval system
- What topics will be covered in this class (overview)?

Resources

- Chapter 1 of IIR
- Resources at https://www.fi.muni.cz/~sojka/PV211/ and http://cislmu.org, materials in MU IS and FI MU library
  - course schedule and overview
  - information retrieval links
  - Shakespeare search engine https://www.rhymezone.com/shakespeare/