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

2024-02-21

(compiled on 2024-06-19 19:15:23)

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

Start with why (Simon Sinek)

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).
Why important? Why you? Why now?
Information handling on Faculty of informatics in information age. . .

Prerequisites

Curiosity about how Information Retrieval works.
But seriously, based on Manning *et al.* IIR textbook (available in MU libraries):

- 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.

PV211 course design I

- proactive rather than reactive learning
- diversity is stability, e.g. welcomed
- learning by doing/programming
- skillful rather than bag of facts
- Stanford (TEX, Google, ...) inspired

PV211 course design II

- 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."
- Questions are welcome—on PV211 IS discussion forum *before* lectures, and also *during* lectures.
- Respect to the individual learning speed and knowledge.
- Student [soft skills and programming] activities (answering in discussion forums) are *explicitly welcomed*.
- 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, Google Colab notebooks,. . . .

Teachers

- Petr Sojka, sojka@fi.muni.cz
- Consulting hours Spring 2024:
  Thursday 8:45–9:30 A502 or by appointment by email.
- Room C523 (or C522), fifth floor, Botanická 68a.
- Course web page:
  https://www.fi.muni.cz/~sojka/PV211/
- Teaching assistants (TA):
  Mgr. Marek Toma 485275@mail.muni.cz,
  Ing. Martin Fajčík ifajcik@fit.vutbr.cz,
  Santosh Kesiraju, Ph.D. kesiraju@fit.vutbr.cz
  Mgr. Michal Štefánik, stefanik.m@mail.muni.cz
  RNDr. Vít Starý Novotný, Ph.D. witiko@mail.muni.cz,
  Mgr. Šárka Ščavnická, 527352@mail.muni.cz
  All TAs are ready for consultations after their teaching hours or by appointment.

Course Evaluations

Classification is based on points you could get

a) **48 pts** for projects done during the term:

- **10 pts** for the first project
- **6 pts** for giving a peer review
- **26 pts** for the second project
- **6 pts** for the peer review

and

b) **52 pts** for the final exam (multiple-choice test):

- **20 pts** for exercises, similar to those practiced at seminars
- **32 pts** for classical multiple-choice test

Projects evaluation and peer review evaluation I

Until 19. 3. 23:59 (resp. 7. 5. 23:59), your tasks awarded up to **10 pts** resp. **26 pts** will be the following:

Functional implementation of Information Retrieval system: Individually implement a ranked unsupervised (resp. supervised) retrieval system for Cranfield (resp. ARQMath3) collection.

Readability: Document your code and stick to an organized, consistent, human-readable coding style.

Reaching a minimal score: reach at least 22% (resp. 10%) mean average precision (MAP) score and record it in your Jupyter notebook or in the public leaderboard.

Innovativeness & creativity: Upload an `.ipynb` file with your Jupyter notebook to the homework vault in IS MU.

Projects evaluation and peer review evaluation II

For detailed instructions and an example solution, see the Google Colaboratory document linked from the interactive course syllabus in IS MU.

You will be instructed on the first practical on which institutional computational resources (Jupyter Hub, Google Colab, Deepnote,...) you will have at your disposal for solving projects.

Peer review evaluation

For each project, test and fill in a questionnaire about 3 solutions of your peers, including feedback with ideas for further improvements. Between 20. 3. and 26. 3., (resp. 8. 5. and 14. 5.) your task awarded up to $3 \times 2 =$ **6 pts** will be to review the term projects of three of your colleagues. **0.5 pts** will be awarded for handing in a review of your colleague's term project. **1.5 pts** will be awarded for reviewing the completion of tasks in your colleague's term project.

Bonus points motivation I

- You can get extra 10/9/8/.../1/0 point(s) for the $1^{st}/2^{nd}/3^{rd}/.../\ldots 10/11^{th}$ place in the first project assignment.
- You can get extra 20/18/16/.../2/0 point(s) for the $1^{st}/2^{nd}/3^{rd}/.../\ldots 10/11^{th}$ place in the second project assignment.
- *In addition*, one can get additional *premium* points based on activities during lectures, practicals (good answers), projects (PV211-utils pull requests), in IS discussion forum, or negotiated related (Kaggle) projects.

Summary of course grading I

Classification scale lower bounds for passing z/k are 48/53 points and E–A grading will be adjusted based on ECTS suggestion in IS (E/D/C/B/A ≈ **58/66/74/82/90 pts** . Dates of [final] exams will be announced via IS.muni.cz (at least three terms, probably four).

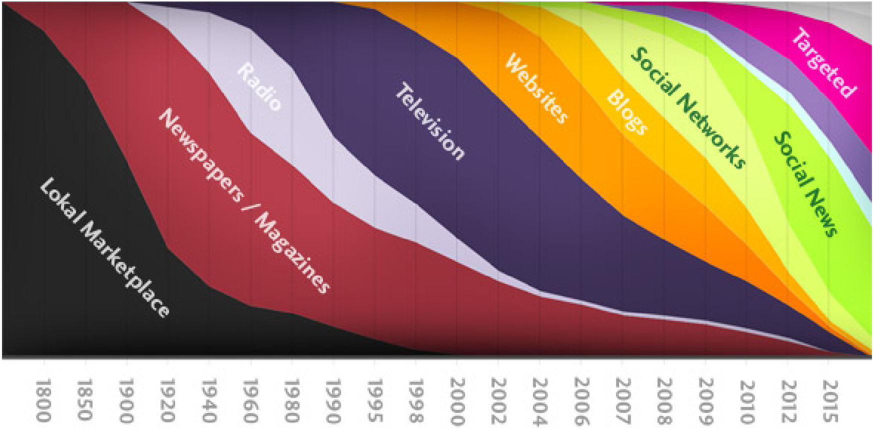
week/ deadline	pts	description
1–4/ 19. 3. 23:59	10	first assignment project (Cranfield)
5/ 26. 3. 23:59	6	peer review of Cranfield TFIDF
6–11/ 7. 5. 23:59	20	second assignment project (ARQMath3)
6–11/ 7. 5. 23:59	6	for justification and explanation of your solution and code of second assignment
12/ 14. 5. 23:59	6	peer review of explained second project
14+/ exam part 1	20	open exercises, similar to those practiced at seminars
14+/ exam part 2	32	classical multiple-choice test testing understanding of topics taught
1–14+/ extra points	X	extra activities during term or negotiated related projects
14+/ total points	100+X	points for ECTS gradings

Can we proceed [Y/N]?

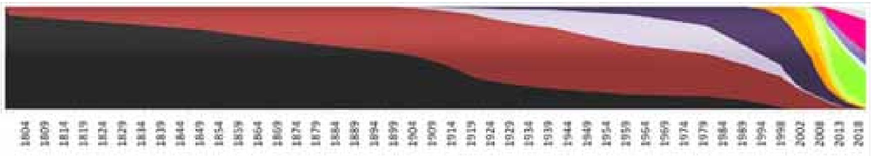
Questions?

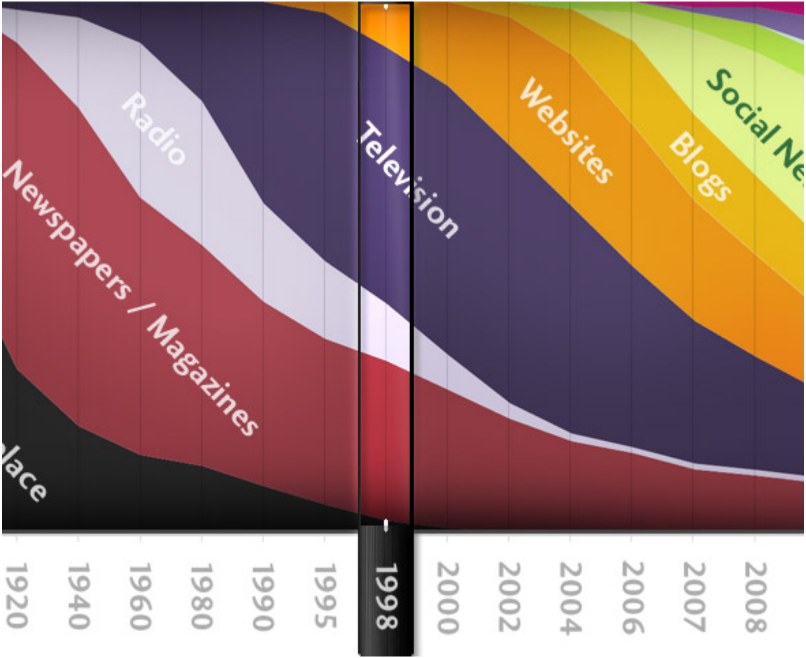
Python? Jupyter Notebook, Jupyter Hub? Google Colab? Deepnote project? Bc./Mgr./Ph.D.? Mandatory course z/k/zk? Erasmus? Nationalities: CZ?, SK?, EN=C2 (mother tongue)?, other? 2 programming projects?/challenges? Student peer reviews? Mikolov? Řehůřek? Materna? Jurových? Discord discussion forum with anonymous posts?!

History of *information retrieval*: gradual changes of channels and technologies



Gradual *speedup* of changes in IR





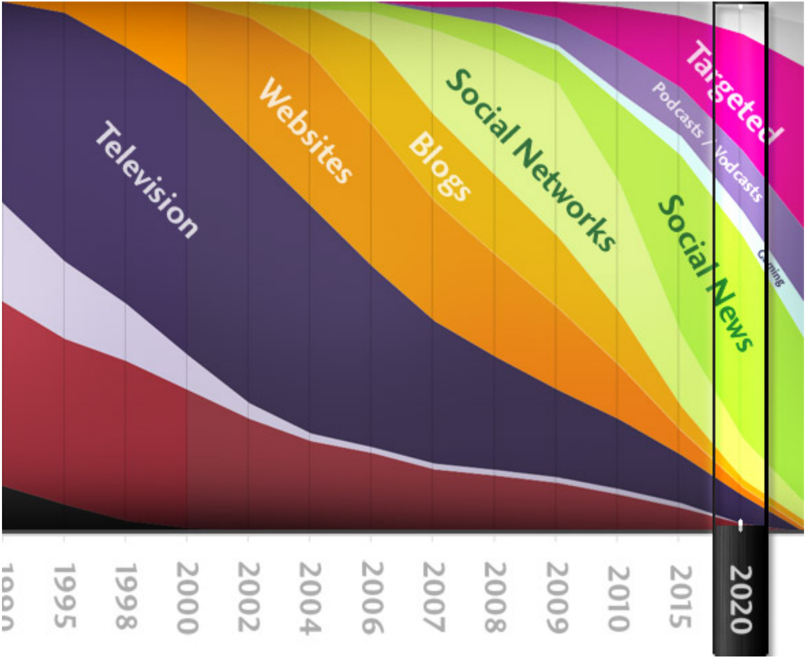
Radio
Television
Websites
Blogs
Social Ne
Newspapers / Magazines
1920
1940
1960
1980
1990
1995
1998
2000
2002
2004
2006
2007
2008

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

Google

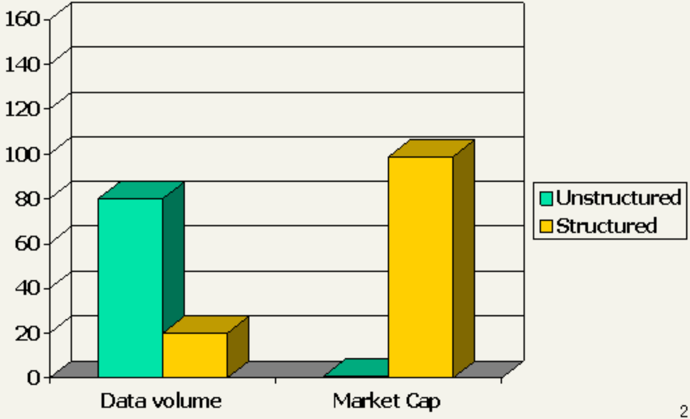
1998: google.stanford.edu

- collaborative project with Stanford faculty (‘flipped IS’ :-)
- on collected disks in the dormitories
- Google 1998 ‘Anatomy paper’ (Larry Page, Sergey Brin)

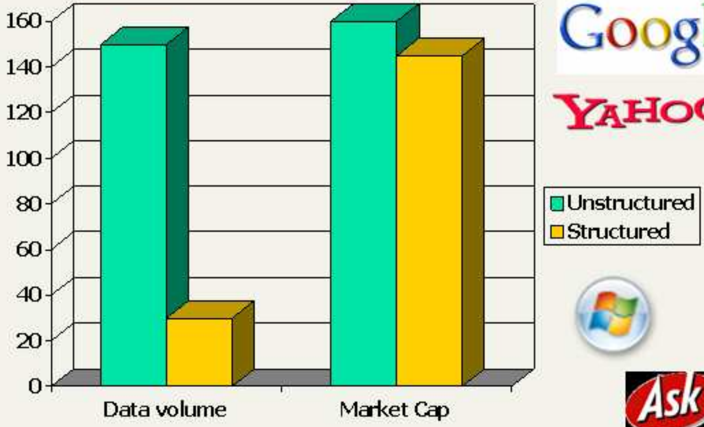


Television
Websites
Blogs
Social Networks
Social News
Podcasts / Vodcasts
Targeted
1995
1998
2000
2002
2004
2006
2007
2008
2009
2010
2015
2020

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



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



Unstructured (text) versus structured (database) data

in 2016 ?

in 2026 ?

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.

Unstructured data in 1650: collective works of 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

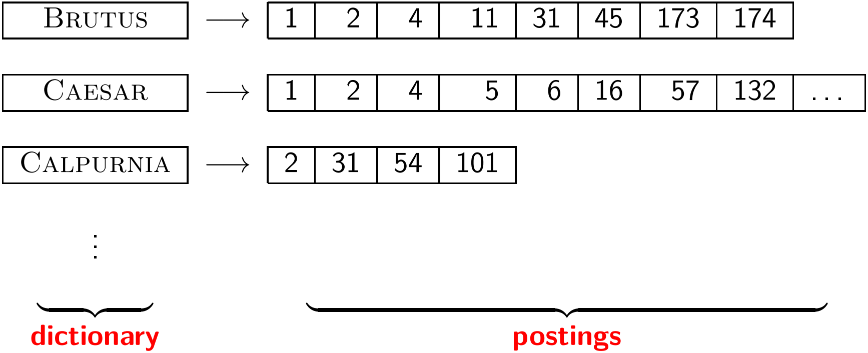
- Consider $N = 10^6$ documents, each with about 1000 tokens
- $\Rightarrow$ total of 10^9 tokens
- On average 6 bytes per token, including spaces and punctuation $\Rightarrow$ size of document collection is about $6 \cdot 10^9 =$ 6 GB
- Assume there are $M = 500{,}000$ distinct terms in the collection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



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

⟹

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

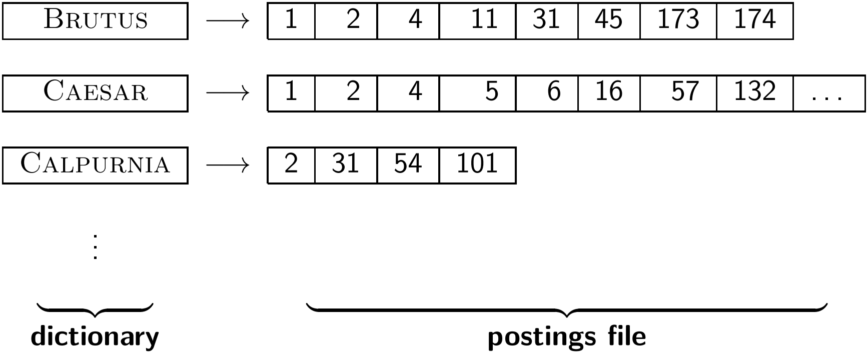
Create postings lists, determine document frequency

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



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

CALPURNIA ⟶ 2 → 31 → 54 → 101

Intersection ⟹ 2 → 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 ⟶ 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 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 ⟶ 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

CALPURNIA ⟶ 2 → 31 → 54 → 101

CAESAR ⟶ 5 → 31

Optimized intersection algorithm for conjunctive queries

```
INTERSECT(⟨t_1, ..., t_n⟩)
    terms ← SORTBYINCREASINGFREQUENCY(⟨t_1, ..., t_n⟩)
    result ← postings(first(terms))
    terms ← rest(terms)
    while terms ≠ NIL and result ≠ NIL
    do result ← INTERSECT(result, postings(first(terms)))
        terms ← rest(terms)
    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)

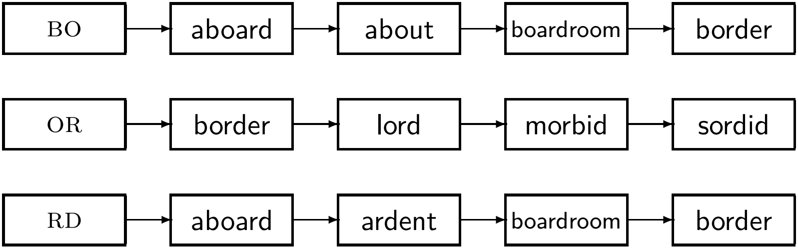
Course overview and agenda

- We are done with Chapter 1 of IIR (IIR 01).
- Plan for the rest of the semester: some 14 of the 21 chapters of IIR (cf. slides from previous years and those planned for this year – comments welcome).
- During weeks 6–9 will probably not be contact lectures but selfstudy from available study materials and recordigns from previous years.
- In addition to experts from FI lectures by leading experts are in preparation (probably weeks 5, 10 and 13).
- In what follows: teasers for most chapters – to give you a sense of what will be covered.
- One or two bonus invited lecture(s), and lecture(s) on IR topics researched in research group MIR.fi.muni.cz and on state-of-the art achievements in the area (vector space embeddings, transformers, Neural AI 4 IR, etc.).

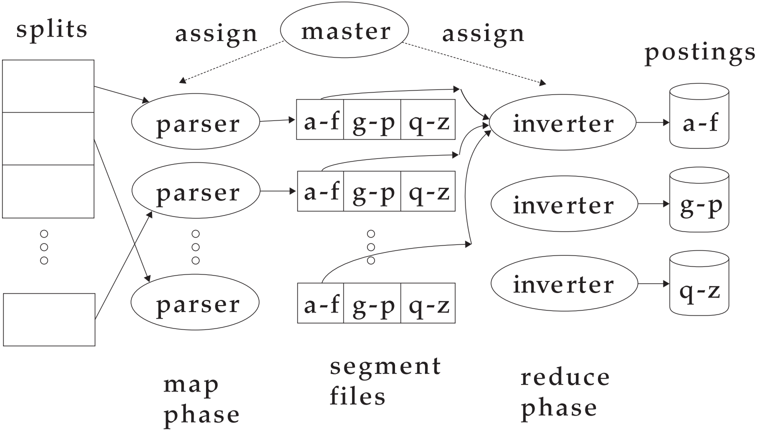
Week 2 – IIR 02: The term vocabulary and postings lists

- Phrase queries: "Stanford University"
- Proximity queries: Gates near Microsoft
- We need an index that captures position information for phrase queries and proximity queries.

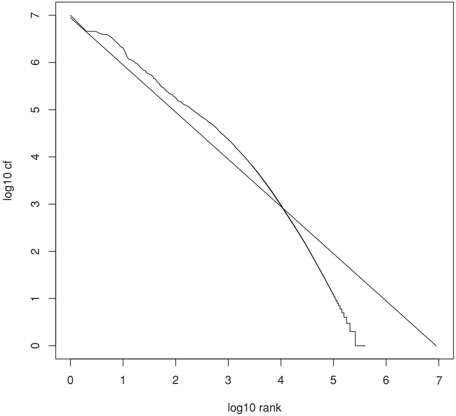
Week 2 – IIR 03: Dictionaries and tolerant retrieval



Week 3 – IIR 04: Index construction



Week 4 – IIR 05: Index compression

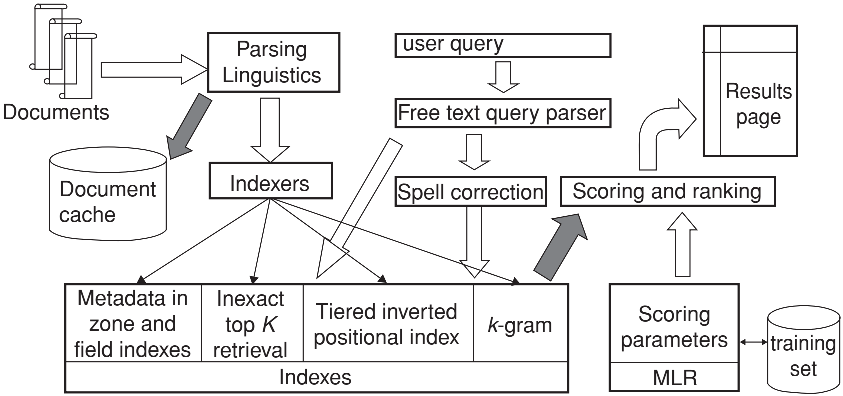


Zipf's law

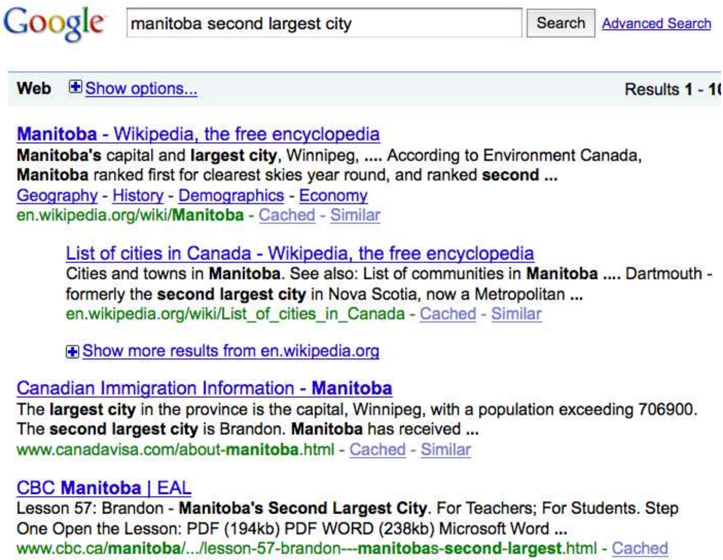
Week 4 – IIR 06: Scoring, term weighting and the vector space model

- 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

Week 5 – IIR 07: Scoring in a complete search system



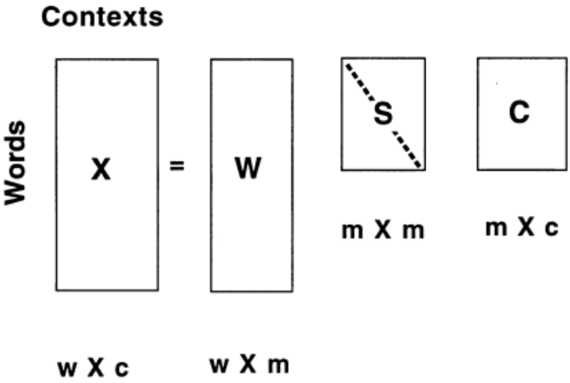
Week 5 – IIR 08: Evaluation and dynamic summaries



Week 6 – Anatomy of the web-scale IR system

Challenges in Building Large-Scale Information Retrieval Systems
by Jeff Dean, Google Senior Fellow, `jeff@google.com`

Week 7 – IIR 18: Latent Semantic Indexing



Week 7 – CS276 14: Distributed Word Representations for Information retrieval

Introduction to Information Retrieval

Distributional similarity based representations

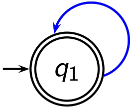
- You can get a lot of value by representing a word by means of its neighbors
- "You shall know a word by the company it keeps"
  - (J. R. Firth 1957: 11)
- One of the most successful ideas of modern statistical NLP

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

↖ These words will represent *banking* ↗

Week 8 – 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

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

Week 8 – IIR 13: Text classification & Naive Bayes

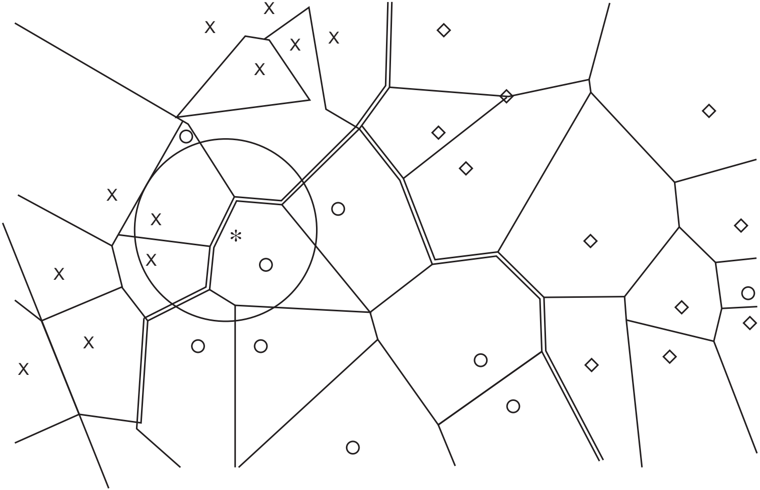
- Text classification = assigning documents automatically to predefined classes
- Examples:
  - Language (English vs. French)
  - Adult content
  - Region

NO week this term IIR 11: Probabilistic information retrieval

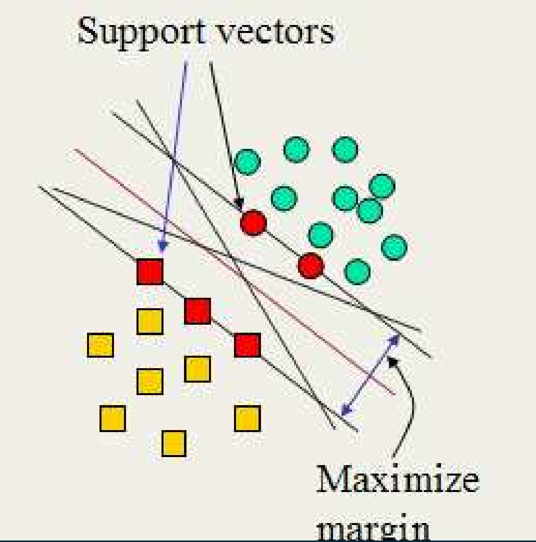
	document	relevant ($R = 1$)	nonrelevant ($R = 0$)
Term present	$x_t = 1$	p_t	u_t
Term absent	$x_t = 0$	$1 - p_t$	$1 - u_t$

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

Week 9 – IIR 14: Vector classification, kNN search



Week 10 – IIR 15: Support vector machines, Learning to rank



Week 11 – IIR 16: Flat clustering

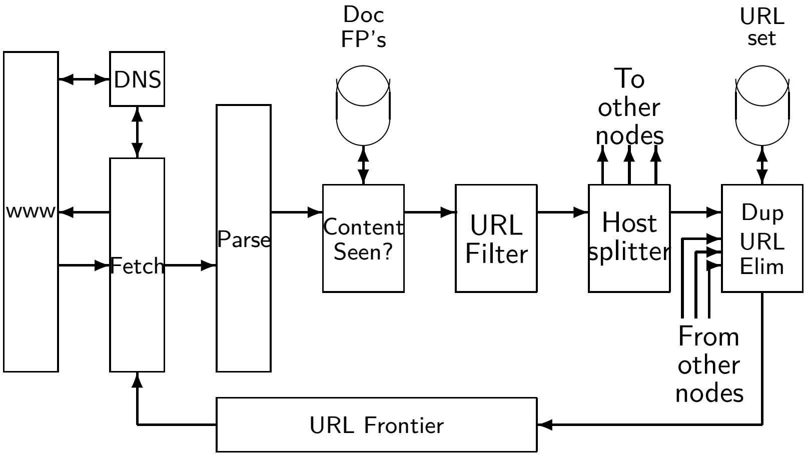
NO week this term – IIR 17: Hierarchical clustering

`http://news.google.com`

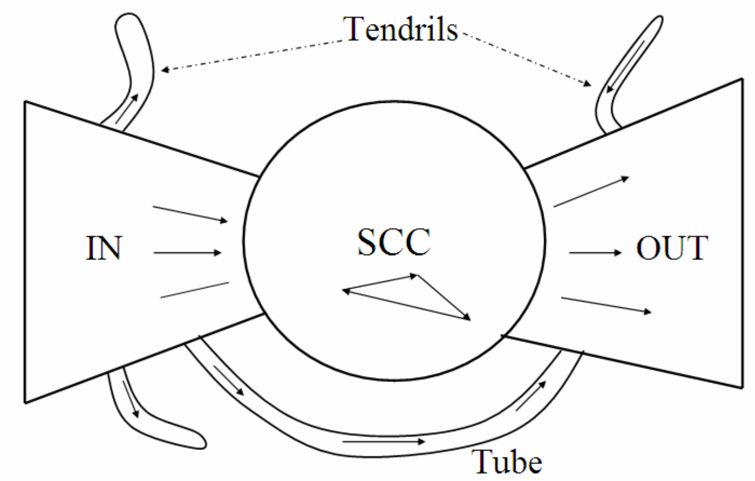
Week 12 – IIR 19: The web and its challenges

- 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?

Week 12 – IIR 20: Crawling



Week 13 – IIR 21: Link analysis / PageRank



Week 1 – Week 13: Related research seminars and courses

Semianr FI:PV212 of LEMMA/MIR labs.

- MIR group's solution for ARQMath 2022 @ CLEF2022 tasks: Math information Retrieval Question Answering and Formula searching
- Talks and brainstormings of TA's and FI MU alumni's talks (Řehůřek, Materna, Jurových,. . .)?
- Informatics colloquium related talk(s): Tomáš Mikolov 2019, or in 2017.

MU on Coursera

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
  - IIR textbook and other books (Baeta-Yates et al: Modern Information Retrieval, and other passed on during the lecture)
  - Jupyter Hub/ Google Colab/ Deepnote environments with examples