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

2020-02-19

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

1. Introduction
2. History of information retrieval
3. Boolean model
4. Inverted index
5. Processing queries
6. Query optimization
7. Course overview and agenda

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.

PV211 course design: proactive rather than reactive learning, diversity is good, learning by doing, skillful rather than bag of facts, Stanford inspired

- Student [soft skills and programming] activities are *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 are welcome—on PV211 IS discussion forum *before* lectures, and also during 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, Google Colab notebooks,. . . .

Teachers

- Petr Sojka, sojka@fi.muni.cz
- Consulting hours Spring 2020:
  Thursday 9:00–9:50 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/
- Teaching assistants: Vít Novotný, witiko@mail.muni.cz, Dávid Lupták, 422640@mail.muni.cz, Michal Štefánik, stefanik.m@mail.muni.cz and Eniafe Festus Ayetiran, 243532@mail.muni.cz. All TAs are seated in B203/B206 and are ready for consultations by appointment.

Evaluation of students

Classification is based on points you could get a) **50 pts** during the term: **28 pts** for the term programming project, **12 pts** for term project peer review, **10 pts** for your activities in discussion forum and during lectures evaluated by teacher(s) at the end of term, and b) **50 pts** for the final test: **30 pts** for open exercises, similar to those practiced at exercises, **20 pts** for multiple-choice test. *In addition*, one can get additional *premium* points based on activities during lectures, exercises (good answers) or negotiated related projects. Classification scale lower bounds for passing z/k/zk are 50/57/64 points and E–A grading will be based on ECTS suggestion in IS, E/D/C/B/A corresponds ≈ to 71/78/85/92/100 points. Dates of [final] exams will be announced via IS.muni.cz (at least three terms).

Questions?

Term projects and their student peer reviews I

Until 29. 4. 12:00, your tasks awarded up to **28 pts** will be the following:

1. Individually implement a ranked retrieval system for Cranfield collection.
2. Document your code and stick to an organized, consistent, human-readable coding style.
3. Reach at least 35% mean average precision.
4. Upload a link to your Google Colaboratory document to the homework vault in IS MU.

Term projects and their student peer reviews II

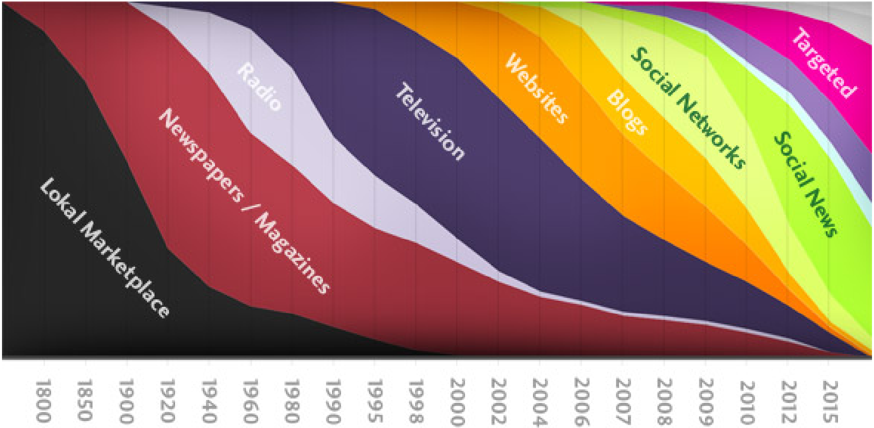
For detailed instructions and an example solution, see the Google Colaboratory document linked from the interactive course syllabus in IS MU. The link to a project leaderboard is also in the syllabus. Between 29. 4. and 6. 5., your task awarded up to $3 \times 4 = \mathbf{12\,pts}$ will be to review the term projects of three of your colleagues. One point will be awarded for handing in a review of your colleague's term project. Three points will be awarded for reviewing the completion of tasks 1, 2, and 3 in your colleague's term project. You can get up to extra 50/20/10/9/8/7/6/5/4/3/2/1 point(s) for the 1st/2nd/3rd/4th/. . . /12th place in the leaderboard on 29. 4. 12:00.

Can we proceed [Y/N]?

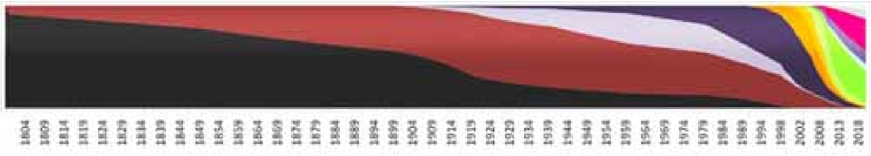
Questions?

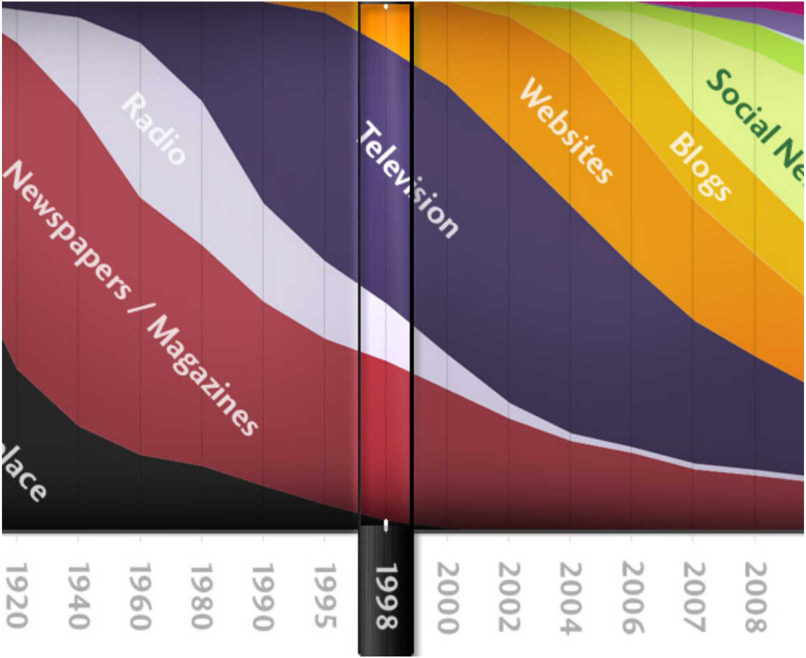
Python? Jupyter Notebook? Google Colab? Deepnote project? Bc./Mgr./Ph.D.? Mandatory course? Erasmus? Nationalities: CZ?, SK?, EN=C2 (mother tongue)?, other? 2 midterms or a programming project/challenge? Student peer reviews? Mikolov? Řehůřek? Materna? Jurových? Presentation style? traditional? or agile/interactive [warm ups, Kahoot])? PV211 discussion forum in IS!

History of *information retrieval*: gradual changes of channels



Gradual *speedup* of changes in IR



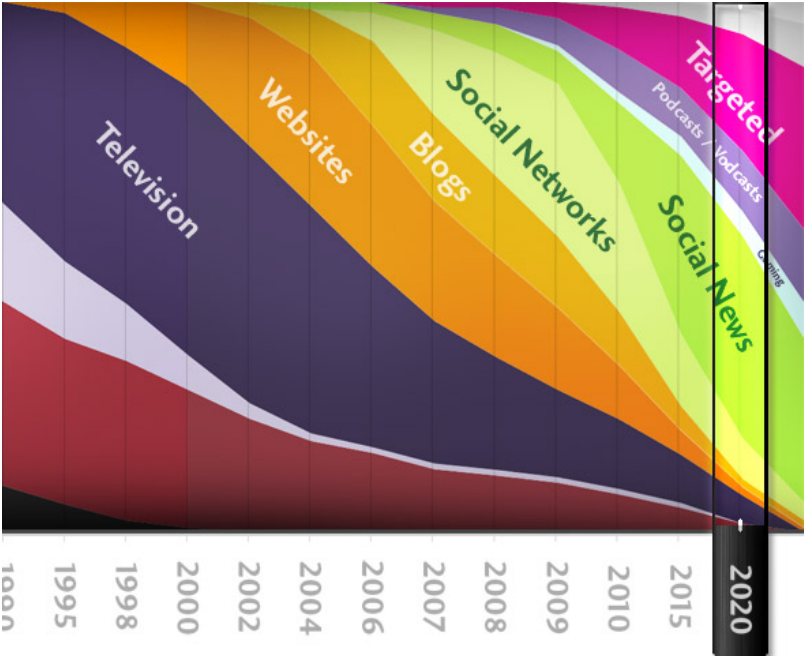

Radio
Newspapers / Magazines
lace
Television
Websites
Blogs
Social Ne
1920
1940
1960
1980
1990
1995
1998
2000
2002
2004
2006
2007
2008

“Google” Circa 1997 (google.stanford.edu)

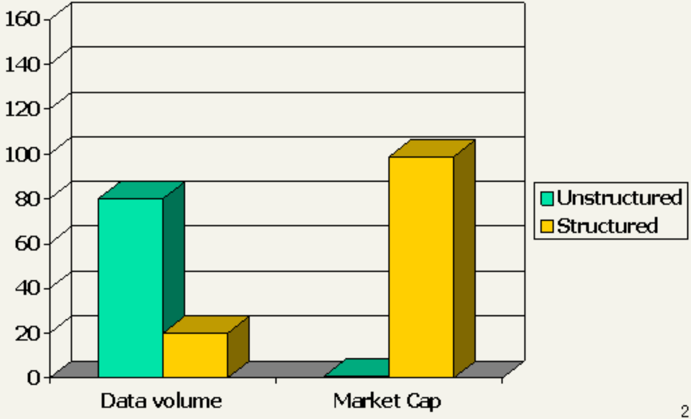
Google

1998: google.stanford.edu

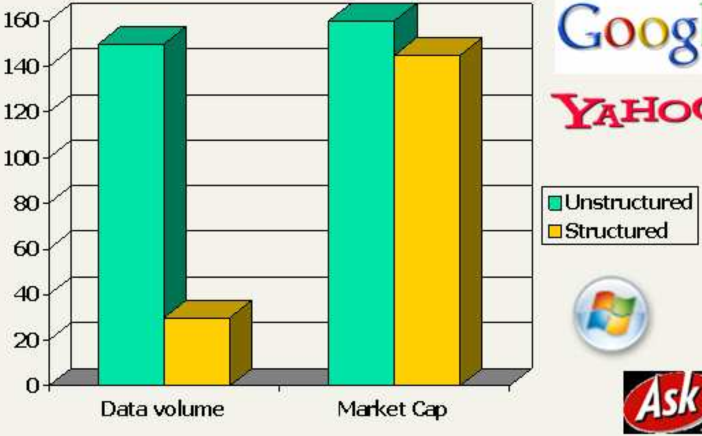
- collaborative project with Stanford faculty (‘flipped IS’ :-)
- on collected disks
- Google 1998 ‘Anatomy paper’ (Page, Brin)


Television
Websites
Blogs
Social Networks
Social News
Targeted
Podcasts / Vodcasts
Gaming
1990
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.
- 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

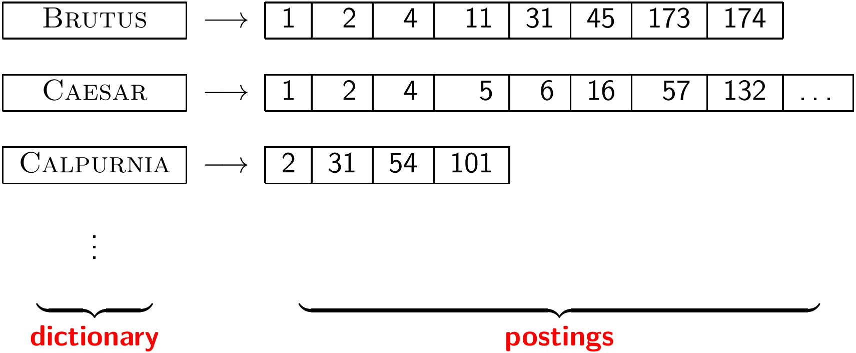
- Consider $N = 10^6$ documents, each with about 1000 tokens
- $\Rightarrow$ total of 10^9 tokens
- On average 6 bytes per token, including spaces and punctuation $\Rightarrow$ size of document collection is about $6 \cdot 10^9 =$ 6 GB
- Assume there are $M = 500{,}000$ distinct terms in the collection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



BRUTUS ⟶ 1 2 4 11 31 45 173 174

CAESAR ⟶ 1 2 4 5 6 16 57 132 . . .

CALPURNIA ⟶ 2 31 54 101

⋮

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:

⟹

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

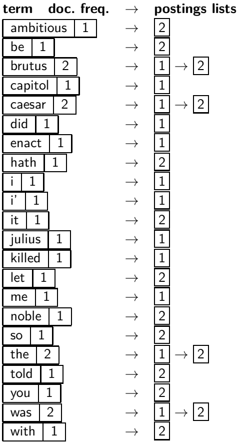
⟹

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

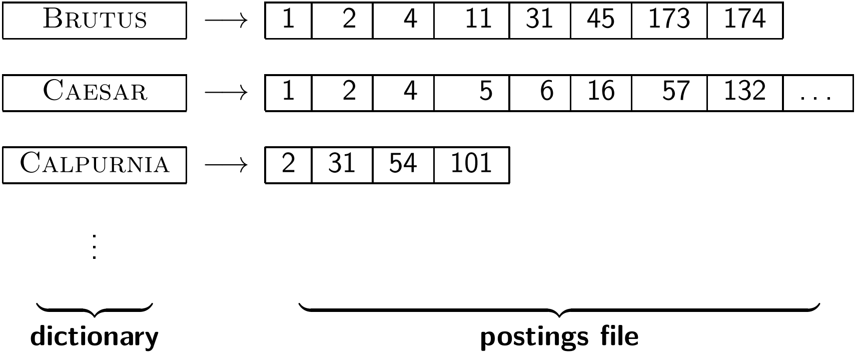
term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
i	1
i	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

⟹



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 ⟶ 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**

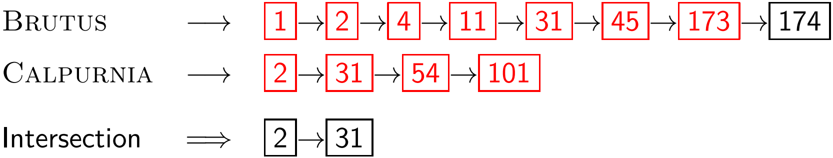
Later in this course

- 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?

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

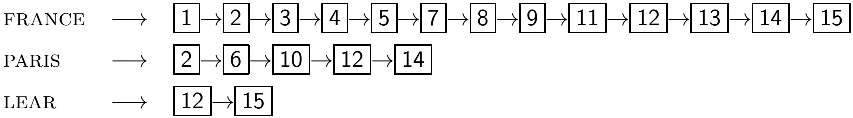
Intersection ⟹ 2 → 31

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

Intersecting two postings lists

```
INTERSECT($p_1, p_2$)
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 $p_1 \leftarrow next(p_1)$
             else $p_2 \leftarrow next(p_2)$
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

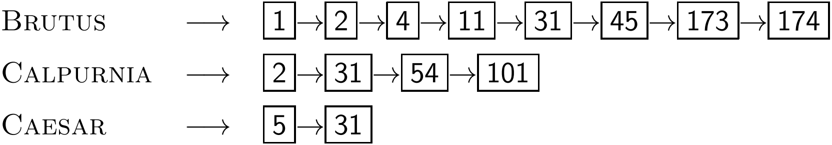
- Proximity operators: /3 = within 3 words, /s = within a sentence, /p = within a paragraph
- Space is disjunction, not conjunction! (This was the default in search pre-Google.)
- Long, precise queries: incrementally developed, not like web search
- Why professional searchers often like Boolean search: 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($\langle t_1, \ldots, t_n \rangle$)
  $terms \leftarrow$ SortByIncreasingFrequency($\langle t_1, \ldots, t_n \rangle$)
  $result \leftarrow postings(first(terms))$
  $terms \leftarrow rest(terms)$
  while $terms \neq$ nil and $result \neq$ nil
  do $result \leftarrow$ Intersect($result, postings(first(terms))$)
     $terms \leftarrow rest(terms)$
  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)

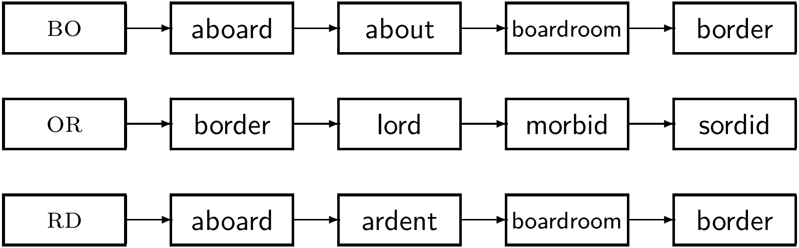
Course overview and agenda

- 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
- 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 my research group MIR.fi.muni.cz and on state-of-the art achievements in the area (vector space embeddings, transformers, 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

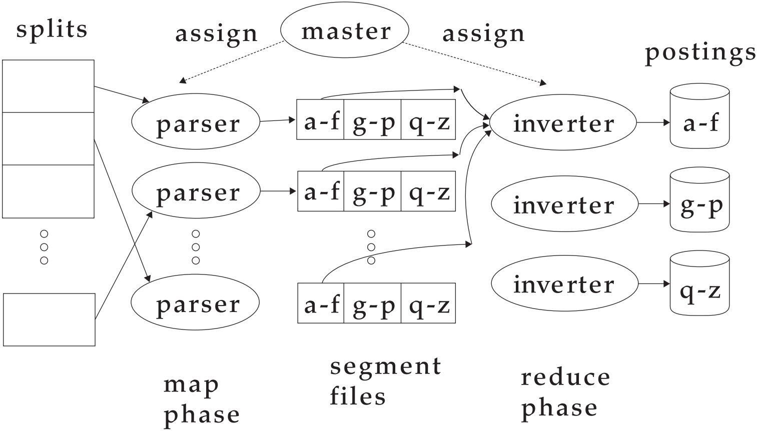


BO → aboard → about → boardroom → border

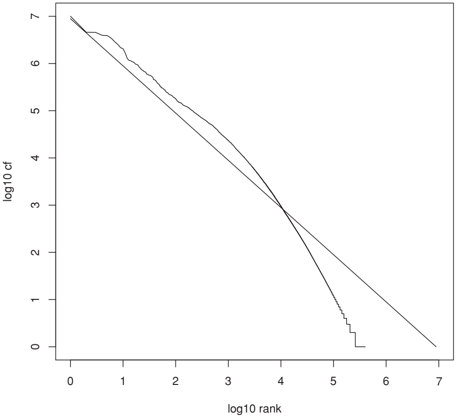
OR → border → lord → morbid → sordid

RD → aboard → ardent → boardroom → border

IIR 04: Index construction



IIR 05: Index compression

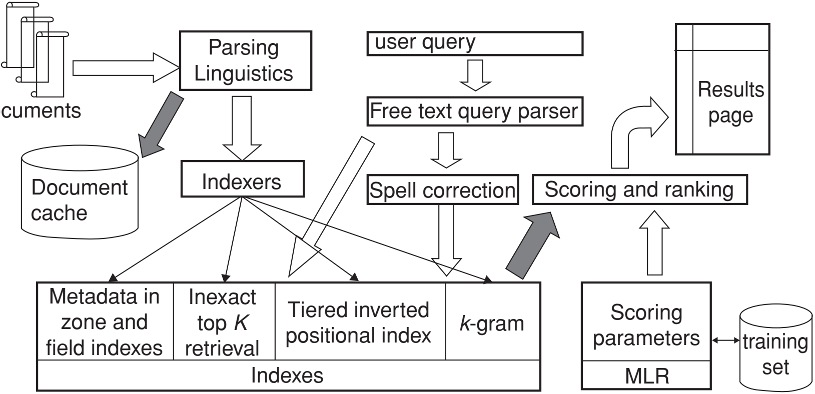


Zipf's law

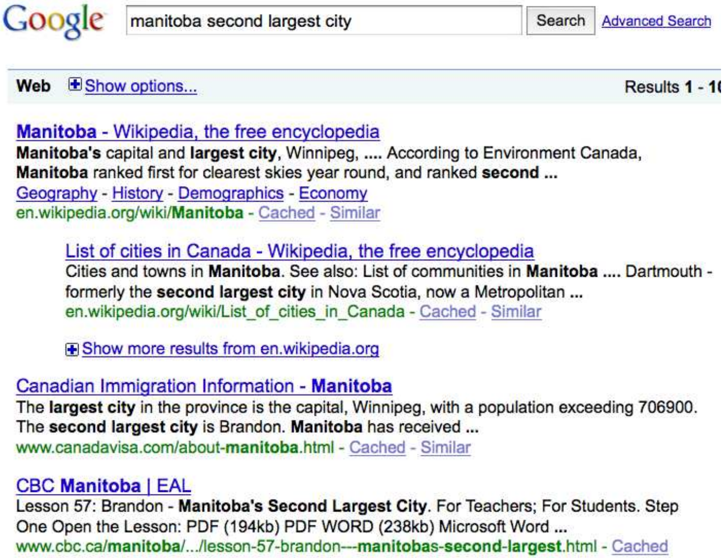
IIR 06: Scoring, term weighting and the vector space model

- 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



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

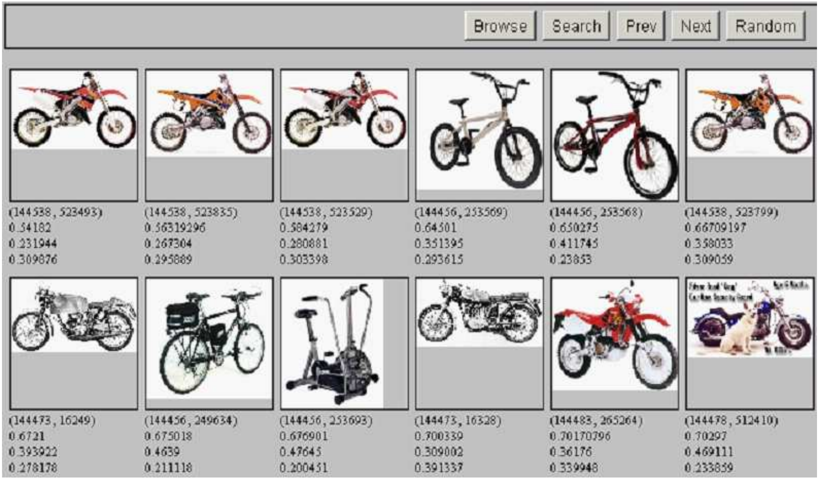
List of cities in Canada - Wikipedia, the free encyclopedia
Cities and towns in **Manitoba**. See also: List of communities in **Manitoba** .... Dartmouth - formerly the **second largest city** in Nova Scotia, now a Metropolitan **...**
en.wikipedia.org/wiki/List_of_cities_in_Canada - Cached - Similar

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

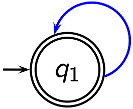
CBC **Manitoba** | EAL
Lesson 57: Brandon - **Manitoba's Second Largest City**. For Teachers; For Students. Step One Open the Lesson: PDF (194kb) PDF WORD (238kb) Microsoft Word **...**
www.cbc.ca/**manitoba**/.../lesson-57-brandon---**manitobas-second-largest**.html - Cached

IIR 09: Relevance feedback & query expansion



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

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

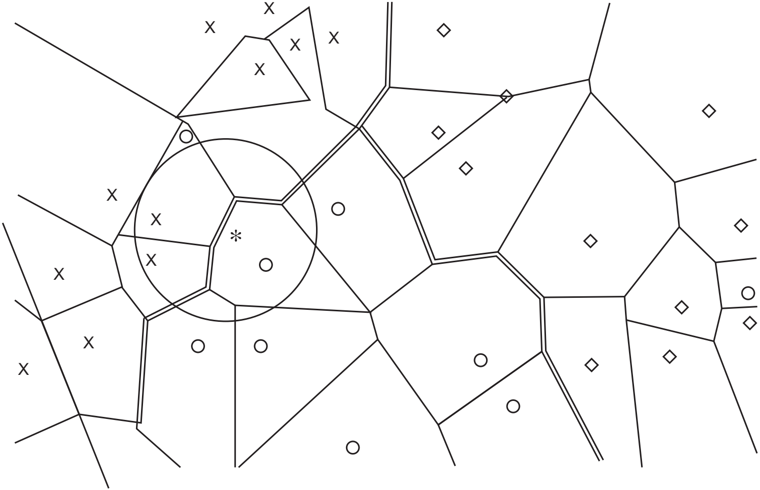
- Text classification = assigning documents automatically to predefined classes
- Examples:
  - Language (English vs. French)
  - Adult content
  - Region

IIR 11: Probabilistic information retrieval

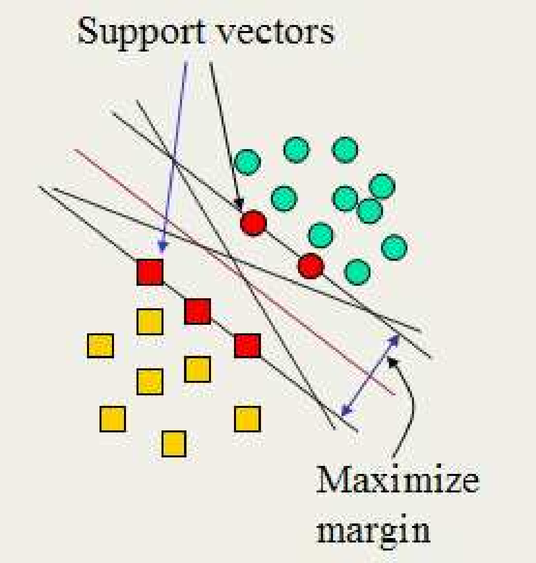
	document	relevant $(R = 1)$	nonrelevant $(R = 0)$
Term present	$x_t = 1$	p_t	u_t
Term absent	$x_t = 0$	$1 - p_t$	$1 - u_t$

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

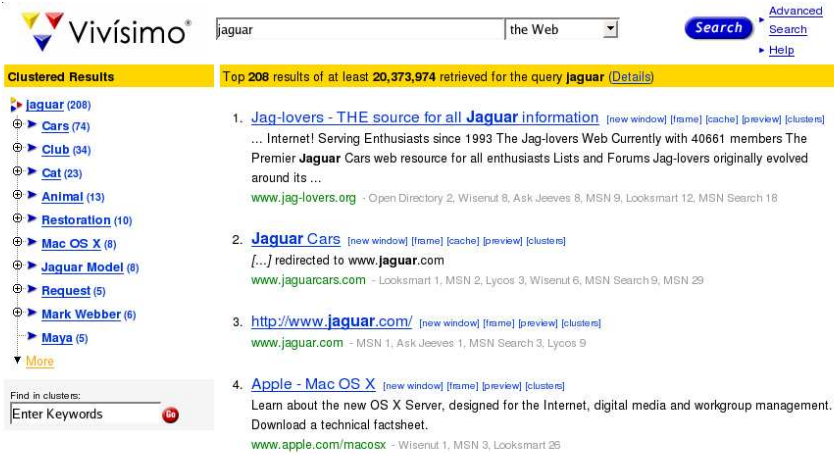
IIR 14: Vector classification, kNN search



IIR 15: Support vector machines



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

Find in clusters:
Enter Keywords Go

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

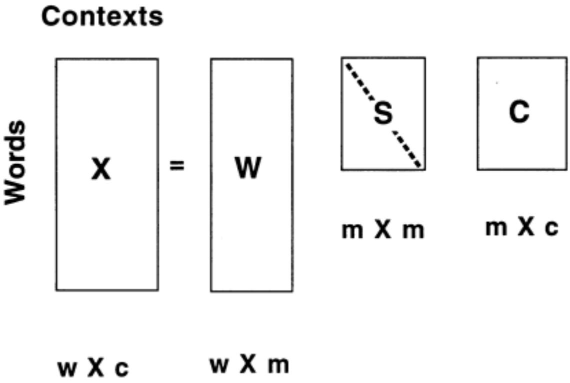
3. http://www.**jaguar**.com/ [new window] [frame] [preview] [clusters]
www.jaguar.com - MSN 1, Ask Jeeves 1, MSN Search 3, Lycos 9

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

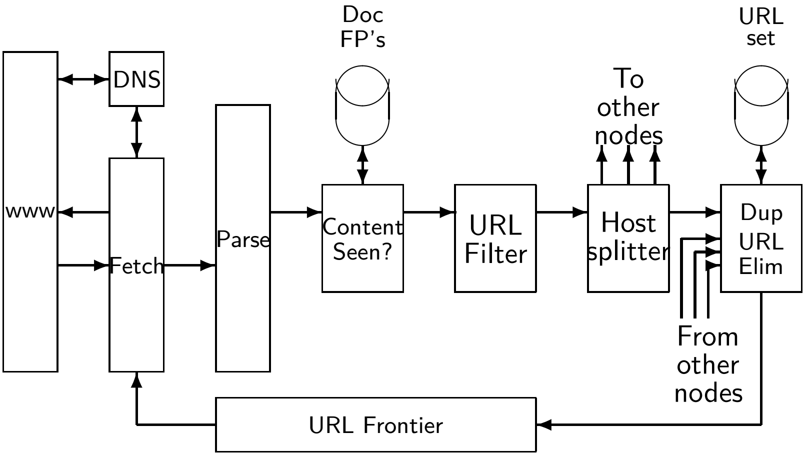
IIR 18: Latent Semantic Indexing



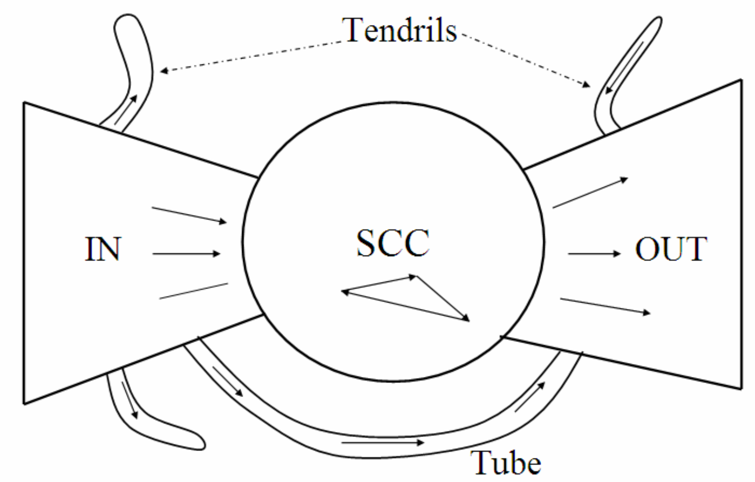
IIR 19: The web and its challenges

- 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/bonus lecture(s)

Under consideration, yet to be decided:

- MIR group's solution for ARQMath CLEF 2020 tasks: Math information Retrieval Question Answering and Formula searching
- FI MU alumni's talk (Řehůřek, Materna, Jurových,. . .)?
- Informatics colloquium related talk(s): Tomáš Mikolov?

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
  - Google Colab environment with examples
  - Shakespeare search engine https://www.rhymezone.com/shakespeare/

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 2: The term vocabulary and postings lists
Handout version

Petr Sojka, Hinrich Schütze et al.

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

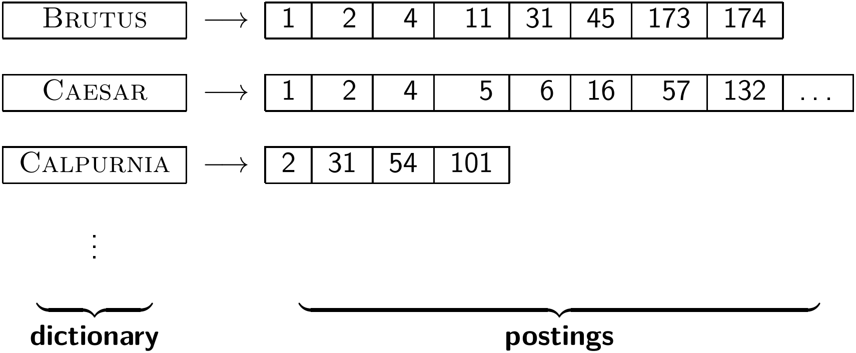
2020-02-26

Overview

1. Recap
2. Documents
3. Terms
   - General + Non-English
   - English
4. Skip pointers
5. Phrase queries

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 . . .

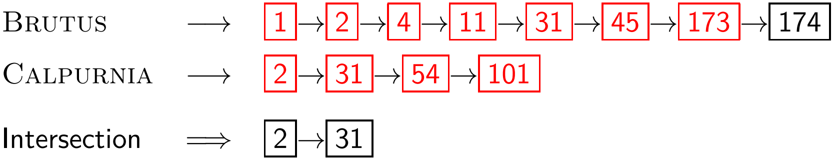
Calpurnia → 2 31 54 101

⋮

dictionary

postings

Intersecting two postings lists



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

CALPURNIA ⟶ 2 → 31 → 54 → 101

Intersection ⟹ 2 → 31

Constructing the inverted index: 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

⟹

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

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

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.

Take-away

- Understanding of the basic unit of classical information retrieval systems: words and documents: What is a document, what is a term?
- Tokenization: how to get from raw text to words (or tokens)
- More complex indexes: skip pointers and phrases

Major steps in inverted index construction

1. Collect the documents to be indexed.
2. Tokenize the text.
3. Do linguistic preprocessing of tokens.
4. Index the documents that each term occurs in.

Documents

- Last lecture: Simple Boolean retrieval system
- Our assumptions were:
  - We know what a document is.
  - We can "machine-read" each document.
- This can be complex in reality: "God is in the details." (Mies van der Rohe)

Parsing a document

- We need to deal with format and language of each document.
- What format is it in? pdf, word, excel, html etc.
- What language is it in?
- What character set is in use?
- Each of these is a classification problem, which we will study later in this course (IIR 13).
- Alternative: use heuristics

Format/Language: Complications

- A single index usually contains terms of several languages.
- Sometimes a document or its components contain multiple languages/formats.
  - French email with Spanish pdf attachment
- What is the document unit for indexing?
- A file?
- An email?
- An email with 5 attachments?
- A group of files (ppt or latex in HTML)?
- Upshot: Answering the question “what is a document?” is not trivial and requires some design decisions.
- Also: XML

Definitions

- Word – A delimited string of characters as it appears in the text.
- Term – A “normalized” word (case, morphology, spelling etc); an equivalence class of words.
- Token – An instance of a word or term occurring in a document.
- Type – The same as term in most cases: an equivalence class of tokens. More informally: what we consider same in the index, e.g. abstraction of a line in the incidence matrix.

Normalization

- Need to "normalize" words in indexed text as well as query terms into the same form.
- Example: We want to match *U.S.A.* and *USA*
- We most commonly implicitly define equivalence classes of terms.
- Alternatively: do asymmetric expansion
  - window → window, windows
  - windows → Windows, windows
  - Windows (no expansion)
- More powerful, but less efficient
- Why don't you want to put *window*, *Window*, *windows*, and *Windows* in the same equivalence class?

Normalization: Other languages

- Normalization and language detection interact.
- *PETER WILL NICHT MIT.* → MIT = mit
- *He got his PhD from MIT.* → MIT ≠ mit

Recall: Inverted index construction

- Input:

  Friends, Romans, countrymen. | So let it be with Caesar ...

- Output:

  friend | roman | countryman | so ...

- Each token is a candidate for a postings entry.
- What are valid tokens to emit?

Exercises

In June, the dog likes to chase the cat in the barn. – How many word tokens? How many word types?

Why tokenization is difficult – even in English. Tokenize: *Mr. O'Neill thinks that the boys' stories about Chile's capital aren't amusing.*

Tokenization problems: One word or two? (or several)

- Hewlett-Packard
- State-of-the-art
- co-education
- the hold-him-back-and-drag-him-away maneuver
- data base
- San Francisco
- Los Angeles-based company
- cheap San Francisco-Los Angeles fares
- York University vs. New York University

Numbers

- 3/20/91
- 20/3/91
- Mar 20, 1991
- B-52
- 100.2.86.144
- (800) 234-2333
- 800.234.2333
- Older IR systems may not index numbers . . .
- . . . but generally it's a useful feature.
- Google example (1+1)

Chinese: No whitespace

莎拉波娃现在居住在美国东南部的佛罗里达。今年４月９日，莎拉波娃在美国第一大城市纽约度过了１８岁生日。生日派对上，莎拉波娃露出了甜美的微笑。

Ambiguous segmentation in Chinese

和尚

The two characters can be treated as one word meaning ‘monk’ or as a sequence of two words meaning ‘and’ and ‘still’.

Other cases of “no whitespace”

- Compounds in Dutch, German, Swedish, Czech (čistokapsonosoplena)
- Computerlinguistik → Computer + Linguistik
- Lebensversicherungsgesellschaftsangestellter
- → leben + versicherung + gesellschaft + angestellter
- Inuit: tusaatsiarunnanngittualuujunga (I can’t hear very well.)
- Many other languages with segmentation difficulties: Finnish, Urdu, . . .

Japanese

ノーベル平和賞を受賞したワンガリ・マータイさんが名誉会長を務めるＭＯＴＴＡＩＮＡＩキャンペーンの一環として、毎日新聞社とマガジンハウスは「私の、もったいない」を募集します。皆様が日ごろ「もったいない」と感じて実践していることや、それにまつわるエピソードを８００字以内の文章にまとめ、簡単な写真、イラスト、図などを添えて１０月２０日までにお送りください。大賞受賞者には、５０万円相当の旅行券とエコ製品２点の副賞が贈られます。

4 different “alphabets”: Chinese characters, hiragana syllabary for inflectional endings and function words, katakana syllabary for transcription of foreign words and other uses, and latin. No spaces (as in Chinese).

End user can express query entirely in hiragana!

Arabic script

كِتَابٌ ⇐ ٌ ب ا ت ِ ك

un b ā t i k

/kitābun/ *‘a book’*

Arabic script: Bidirectionality

استقلت الجزائر في سنة 1962 بعد 132 عاما من الاحتلال الفرنسي.

← → ← → ← START

'Algeria achieved its independence in 1962 after 132 years of French occupation.'

Bidirectionality is not a problem if text is coded in Unicode.

Accents and diacritics

- Accents: résumé vs. resume (simple omission of accent)
- Umlauts: Universität vs. Universitaet (substitution with special letter sequence “ae”)
- Most important criterion: How are users likely to write their queries for these words?
- Even in languages that standardly have accents, users often do not type them. (Polish?)

Case folding

- Reduce all letters to lower case
- Even though case can be semantically meaningful
  - capitalized words in mid-sentence
  - MIT vs. mit
  - Fed vs. fed
  - . . .
- It's often best to lowercase everything since users will use lowercase regardless of correct capitalization.

Stop words

- stop words = extremely common words which would appear to be of little value in helping to select documents matching a user need
- Examples: *a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with*
- Stop word elimination used to be standard in older IR systems.
- But you need stop words for phrase queries, e.g. “King of Denmark”.
- Most web search engines index stop words.

More equivalence classing

- Soundex: IIR 3 (phonetic equivalence, Muller = Mueller)
- Thesauri: IIR 9 (semantic equivalence, car = automobile)

Lemmatization

- Reduce inflectional/variant forms to base form
- Example: *am, are, is* → *be*
- Example: *car, cars, car's, cars'* → *car*
- Example: *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization implies doing "proper" reduction to dictionary headword form (the lemma).
- Inflectional morphology (*cutting* → *cut*) vs. derivational morphology (*destruction* → *destroy*)

Stemming

- Definition of stemming: Crude heuristic process that chops off the ends of words in the hope of achieving what “principled” lemmatization attempts to do with a lot of linguistic knowledge.
- Language dependent
- Often inflectional and derivational
- Example for derivational: *automate, automatic, automation* all reduce to *automat*

Porter algorithm

- Most common algorithm for stemming English
- Results suggest that it is at least as good as other stemming options
- Conventions + 5 phases of reductions
- Phases are applied sequentially
- Each phase consists of a set of commands.
  - Sample command: Delete final *ement* if what remains is longer than 1 character
  - replacement → replac
  - cement → cement
- Sample convention: Of the rules in a compound command, select the one that applies to the longest suffix.

Porter stemmer: A few rules

Rule			**Example**		
SSES	→	SS	caresses	→	caress
IES	→	I	ponies	→	poni
SS	→	SS	caress	→	caress
S	→		cats	→	cat

Three stemmers: A comparison

Sample text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Porter stemmer: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret

Lovins stemmer: such an analys can reve featur that ar not eas vis from th vari in th individu gen and can lead to a pictur of expres that is mor biolog transpar and acces to interpres

Paice stemmer: such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

Does stemming improve effectiveness?

- In general, stemming increases effectiveness for some queries, and decreases effectiveness for others.
- Queries where stemming is likely to help: [tartan sweaters], [sightseeing tour san francisco]
- (equivalence classes: {sweater,sweaters}, {tour,tours})
- Porter Stemmer equivalence class *oper* contains all of *operate operating operates operation operative operatives operational*.
- Queries where stemming hurts: [operational AND research], [operating AND system], [operative AND dentistry]

Exercise: What does Google do?

- Stop words
- Normalization
- Tokenization
- Lowercasing
- Stemming
- Non-latin alphabets
- Umlauts
- Compounds
- Numbers

Recall basic intersection algorithm

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

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

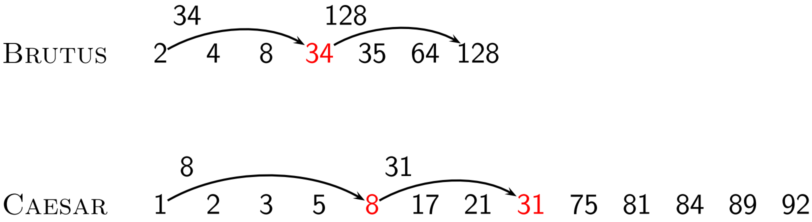
$$\text{Intersection} \Longrightarrow \boxed{2} \to \boxed{31}$$

- Linear in the length of the postings lists.
- Can we do better?

Skip pointers

- Skip pointers allow us to skip postings that will not figure in the search results.
- This makes intersecting postings lists more efficient.
- Some postings lists contain several million entries – so efficiency can be an issue even if basic intersection is linear.
- Where do we put skip pointers?
- How do we make sure intersection results are correct?

Basic idea



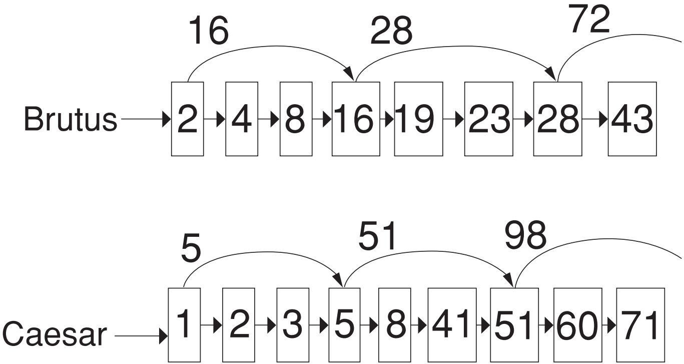
34 128

Brutus 2 4 8 34 35 64 128

8 31

Caesar 1 2 3 5 8 17 21 31 75 81 84 89 92

Skip lists: Larger example



Intersecting with skip pointers

```
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
```

Where do we place skips?

- Tradeoff: number of items skipped vs. frequency skip can be taken
- More skips: Each skip pointer skips only a few items, but we can frequently use it.
- Fewer skips: Each skip pointer skips many items, but we can not use it very often.

Where do we place skips? (cont)

- Simple heuristic: for postings list of length P, use $\sqrt{P}$ evenly-spaced skip pointers.
- This ignores the distribution of query terms.
- Easy if the index is static; harder in a dynamic environment because of updates.
- How much do skip pointers help?
- They used to help a lot.
- With today's fast CPUs, they don't help that much anymore.

Phrase queries

- We want to answer a query such as [Masaryk university] – as a phrase.
- Thus *The president Tomáš Garrigue Masaryk never went to Stanford university* should not be a match.
- The concept of phrase query has proven easily understood by users.
- About 10% of web queries are phrase queries.
- Consequence for inverted index: it no longer suffices to store docIDs in postings lists.
- Two ways of extending the inverted index:
    - biword index
    - positional index
    - Any ideas?

Biword indexes

- Index every consecutive pair of terms in the text as a phrase.
- For example, *Friends, Romans, Countrymen* would generate two biwords: *"friends romans"* and *"romans countrymen"*
- Each of these biwords is now a vocabulary term.
- Two-word phrases can now easily be answered.

Longer phrase queries

- A long phrase like *"masaryk university brno"* can be represented as the Boolean query "MASARYK UNIVERSITY" AND "UNIVERSITY BRNO"
- We need to do post-filtering of hits to identify subset that actually contains the 3-word phrase.

Issues with biword indexes

- Why are biword indexes rarely used?
- False positives, as noted above
- Index blowup due to very large term vocabulary

Positional indexes

- Positional indexes are a more efficient alternative to biword indexes.
- Postings lists in a nonpositional index: each posting is just a docID
- Postings lists in a positional index: each posting is a docID and a list of positions

Positional indexes: Example

Query: “to_1 be_2 or_3 not_4 to_5 be_6”

TO, 993427:

⟨ 1: ⟨7, 18, 33, 72, 86, 231⟩;
2: ⟨1, 17, 74, 222, 255⟩;
4: ⟨8, 16, 190, 429, 433⟩;
5: ⟨363, 367⟩;
7: ⟨13, 23, 191⟩; . . . ⟩

BE, 178239:

⟨ 1: ⟨17, 25⟩;
4: ⟨17, 191, 291, 430, 434⟩;
5: ⟨14, 19, 101⟩; . . . ⟩

Document 4 is a match!

Exercise

Shown below is a portion of a positional index in the format: term: doc1: ⟨position1, position2, . . . ⟩; doc2: ⟨position1, position2, . . . ⟩; etc.

ANGELS: *2: ⟨36,174,252,651⟩; 4: ⟨12,22,102,432⟩; 7: ⟨17⟩;*
FOOLS: *2: ⟨1,17,74,222⟩; 4: ⟨8,78,108,458⟩; 7: ⟨3,13,23,193⟩;*
FEAR: *2: ⟨87,704,722,901⟩; 4: ⟨13,43,113,433⟩; 7: ⟨18,328,528⟩;*
IN: *2: ⟨3,37,76,444,851⟩; 4: ⟨10,20,110,470,500⟩; 7: ⟨5,15,25,195⟩;*
RUSH: *2: ⟨2,66,194,321,702⟩; 4: ⟨9,69,149,429,569⟩; 7: ⟨4,14,404⟩;*
TO: *2: ⟨47,86,234,999⟩; 4: ⟨14,24,774,944⟩; 7: ⟨19,319,599,709⟩;*
TREAD: *2: ⟨57,94,333⟩; 4: ⟨15,35,155⟩; 7: ⟨20,320⟩;*
WHERE: *2: ⟨67,124,393,1001⟩; 4: ⟨11,41,101,421,431⟩; 7: ⟨16,36,736⟩;*

Which document(s) if any match each of the following two queries, where each expression within quotes is a phrase query?: “fools rush in”, “fools rush in” AND “angels fear to tread”

Proximity search

- We just saw how to use a positional index for phrase searches.
- We can also use it for proximity search.
- For example: employment /4 place
- Find all documents that contain EMPLOYMENT and PLACE within 4 words of each other.
- *Employment agencies that place healthcare workers are seeing growth* is a hit.
- *Employment agencies that have learned to adapt now place healthcare workers* is not a hit.

Proximity search

- Use the positional index
- Simplest algorithm: look at cross-product of positions of (i) EMPLOYMENT in document and (ii) PLACE in document
- Very inefficient for frequent words, especially stop words
- Note that we want to return the actual matching positions, not just a list of documents.
- This is important for dynamic summaries etc.

“Proximity” intersection

```
PositionalIntersect($p_1, p_2, k$)
answer $\leftarrow \langle\ \rangle$
while $p_1 \neq$ NIL and $p_2 \neq$ NIL
do if $docID(p_1) = docID(p_2)$
      then $l \leftarrow \langle\ \rangle$
           $pp_1 \leftarrow positions(p_1)$
           $pp_2 \leftarrow positions(p_2)$
           while $pp_1 \neq$ NIL
           do while $pp_2 \neq$ NIL
                 do if $|pos(pp_1) - pos(pp_2)| \leq k$
                       then Add($l, pos(pp_2)$)
                       else  if $pos(pp_2) > pos(pp_1)$
                               then break
                    $pp_2 \leftarrow next(pp_2)$
                 while $l \neq \langle\ \rangle$ and $|l[0] - pos(pp_1)| > k$
                 do Delete($l[0]$)
                 for  each $ps \in l$
                 do Add($answer, \langle docID(p_1), pos(pp_1), ps\rangle$)
                 $pp_1 \leftarrow next(pp_1)$
           $p_1 \leftarrow next(p_1)$
           $p_2 \leftarrow next(p_2)$
      else  if $docID(p_1) < docID(p_2)$
              then $p_1 \leftarrow next(p_1)$
              else  $p_2 \leftarrow next(p_2)$
return answer
```

Combination scheme

- Biword indexes and positional indexes can be profitably combined.
- Many biwords are extremely frequent: Michael Jackson, Britney Spears etc.
- For these biwords, increased speed compared to positional postings intersection is substantial.
- Combination scheme: Include frequent biwords as vocabulary terms in the index. Do all other phrases by positional intersection.
- Williams et al. (2004) evaluate a more sophisticated mixed indexing scheme. Faster than a positional index, at a cost of 26% more space for index.

“Positional” queries on Google

- For web search engines, positional queries are much more expensive than regular Boolean queries.
- Let’s look at the example of phrase queries.
- Why are they more expensive than regular Boolean queries?
- Can you demonstrate on Google that phrase queries are more expensive than Boolean queries?

Take-away

- Understanding of the basic unit of classical information retrieval systems: words and documents: What is a document, what is a term?
- Tokenization: how to get from raw text to words (or tokens)
- More complex indexes: skip pointers and phrases

Resources

- Chapter 2 of IIR
- Resources at `https://www.fi.muni.cz/~sojka/PV211/` and `http://cislmu.org`, materials in MU IS and FI MU library
  - NLP IR boolean search system example: Sketch Engine: `https://ske.fi.muni.cz`
  - Google query language and preprocessing

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 3: Dictionaries and tolerant retrieval
Handout version

Petr Sojka, Hinrich Schütze et al.

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

2020-02-26

Overview

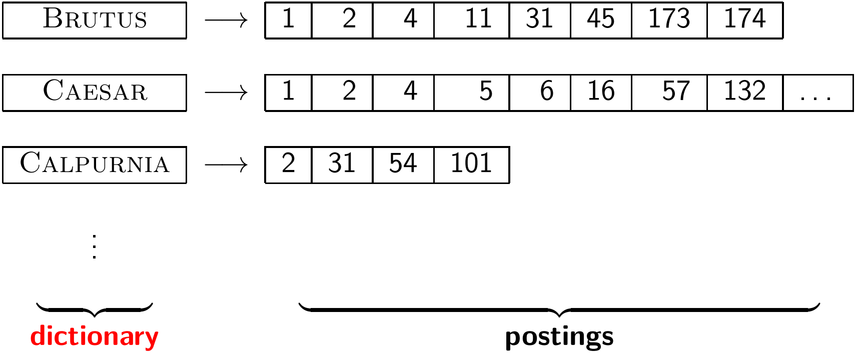
1. Dictionaries
2. Wildcard queries
3. Edit distance
4. Spelling correction
5. Soundex

Take-away

- Tolerant retrieval: What to do if there is no exact match between query term and document term
- Wildcard queries
- Spelling correction

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

Dictionaries

- The dictionary is the data structure for storing the term vocabulary.
- Term vocabulary: the data
- Dictionary: the data structure for storing the term vocabulary

Dictionary as array of fixed-width entries

- For each term, we need to store a couple of items:
  - document frequency
  - pointer to postings list
  - ...
- Assume for the time being that we can store this information in a fixed-length entry.
- Assume that we store these entries in an array.

Dictionary as array of fixed-width entries

	term	document frequency	pointer to postings list
	a	656,265	⟶
	aachen	65	⟶
	...	...	...
	zulu	221	⟶
space needed:	20 bytes	4 bytes	4 bytes

How do we look up a query term q_i in this array at query time? That is: which data structure do we use to locate the entry (row) in the array where q_i is stored?

Data structures for looking up term

- Two main classes of data structures: hashes and trees
- Some IR systems use hashes, some use trees.
- Criteria for when to use hashes vs. trees:
    - Is there a fixed number of terms or will it keep growing?
    - What are the relative frequencies with which various keys will be accessed?
    - How many terms are we likely to have?

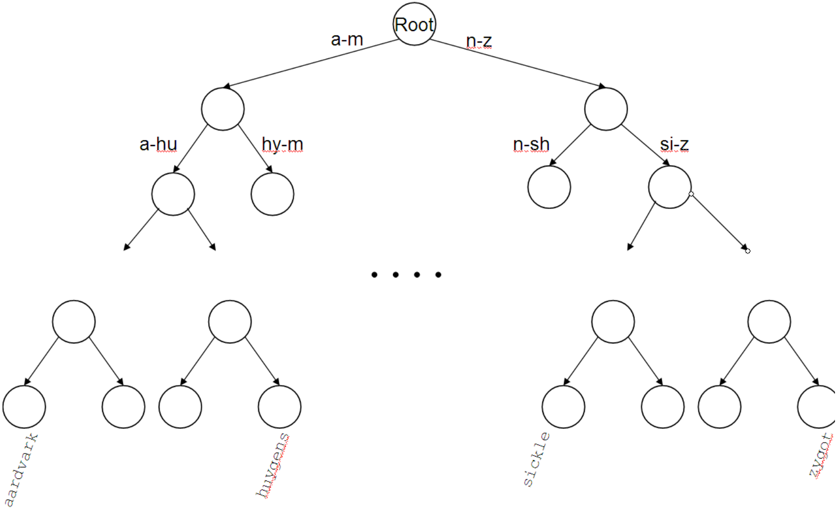
Hashes

- Each vocabulary term is hashed into an integer, its row number in the array.
- At query time: hash query term, locate entry in fixed-width array.
- Pros: Lookup in a hash is faster than lookup in a tree.
  - Lookup time is constant.
- Cons
  - no way to find minor variants (*resume* vs. *résumé*)
  - no prefix search (all terms starting with *automat*)
  - need to rehash everything periodically if vocabulary keeps growing

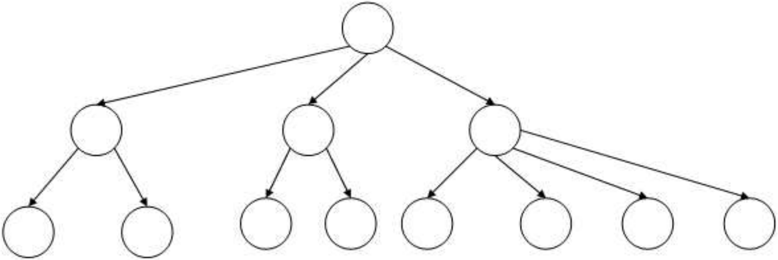
Trees

- Trees solve the prefix problem (find all terms starting with *automat*).
- Simplest tree: binary tree.
- Search is slightly slower than in hashes: $O(\log M)$, where M is the size of the vocabulary.
- $O(\log M)$ only holds for balanced trees.
- Rebalancing binary trees is expensive.
- B-trees mitigate the rebalancing problem.
- B-tree definition: every internal node has a number of children in the interval $[a, b]$ where a, b are appropriate positive integers, e.g., $[2, 4]$.

Binary tree



B-tree



Wildcard queries

- mon*: find all docs containing any term beginning with *mon*
- Easy with B-tree dictionary: retrieve all terms t in the range: mon $\leq t <$ moo
- *mon: find all docs containing any term ending with *mon*
  - Maintain an additional tree for terms *backwards*.
  - Then retrieve all terms t in the range: nom $\leq t <$ non
- Result: A set of terms that are matches for wildcard query.
- Then retrieve documents that contain any of these terms.

Query processing

- At this point, we have an enumeration of all terms in the dictionary that match the wildcard query.
- We still have to look up the postings for each enumerated term.

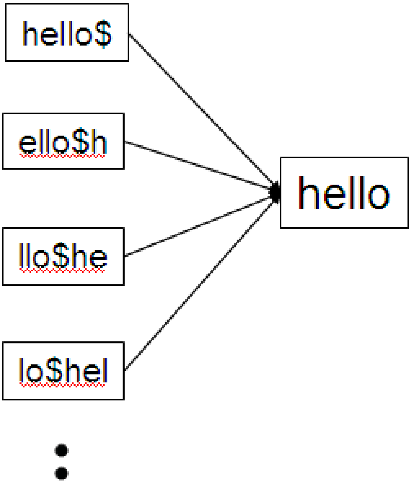
How to handle * in the middle of a term

- Example: m*nchen
- We could look up m* and *nchen in the B-tree and intersect the two term sets.
- Expensive
- Alternative: permuterm index
- Basic idea: Rotate every wildcard query, so that the * occurs at the end.
- Store each of these rotations in the dictionary, say, in a B-tree

Permuterm index

- For term HELLO: add *hello$*, *ello$h*, *llo$he*, *lo$hel*, *o$hell*, and *$hello* to the B-tree where $ is a special symbol

Permuterm → term mapping



Permuterm index

- For HELLO, we've stored: *hello$*, *ello$h*, *llo$he*, *lo$hel*, *o$hell*, *$hello*
- Queries
  - For X, look up X$
  - For X*, look up $X*
  - For *X, look up X$*
  - For *X*, look up X*
  - For X*Y, look up Y$X*
  - Example: For hel*o, look up o$hel*
- Permuterm index would better be called a permuterm tree.
- But permuterm index is the more common name.

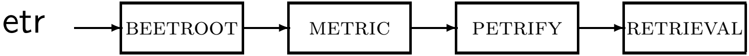
Processing a lookup in the permuterm index

- Rotate query wildcard to the right
- Use B-tree lookup as before
- Problem: Permuterm more than quadruples the size of the dictionary compared to a regular B-tree. (empirical number)

k-gram indexes

- More space-efficient than permuterm index
- Enumerate all character *k*-grams (sequence of *k* characters) occurring in a term
- 2-grams are called bigrams.
- Example: from *April is the cruelest month* we get the bigrams: *$a ap pr ri il l$ $i is s$ $t th he e$ $c cr ru ue el le es st t$ $m mo on nt th h$*
- $ is a special word boundary symbol, as before.
- Maintain an inverted index from bigrams to the terms that contain the bigram

Postings list in a 3-gram inverted index



k-gram (bigram, trigram, . . .) indexes

- Note that we now have two different types of inverted indexes
- The term-document inverted index for finding documents based on a query consisting of terms
- The *k*-gram index for finding terms based on a query consisting of *k*-grams

Processing wildcarded terms in a bigram index

- Query mon* can now be run as:
  $m AND mo AND on
- Gets us all terms with the prefix *mon* ...
- ...but also many "false positives" like MOON.
- We must postfilter these terms against query.
- Surviving terms are then looked up in the term-document inverted index.
- *k*-gram index vs. permuterm index
  - *k*-gram index is more space efficient.
  - Permuterm index doesn't require postfiltering.

Exercise

- Google has very limited support for wildcard queries.
- For example, this query doesn't work very well on Google: [gen* universit*]
    - Intention: you are looking for the University of Geneva, but don't know which accents to use for the French words for university and Geneva.
- According to Google search basics, 2010-04-29: "Note that the * operator works only on whole words, not parts of words."
- But this is not entirely true. Try [pythag*] and [m*nchen]
- Exercise: Why doesn't Google fully support wildcard queries?

Processing wildcard queries in the term-document index

- Problem 1: we must potentially execute a large number of Boolean queries.
- Most straightforward semantics: Conjunction of disjunctions
- For [gen* universit*]: geneva university OR geneva université OR genève university OR genève université OR general universities OR . . .
- Very expensive
- Problem 2: Users hate to type.
- If abbreviated queries like [pyth* theo*] for [pythagoras' theorem] are allowed, users will use them a lot.
- This would significantly increase the cost of answering queries.
- Somewhat alleviated by Google Suggest

Spelling correction

- Two principal uses
  - Correcting documents being indexed
  - Correcting user queries
- Two different methods for spelling correction
- Isolated word spelling correction
  - Check each word on its own for misspelling
  - Will not catch typos resulting in correctly spelled words, e.g., *an asteroid that fell form the sky*
- Context-sensitive spelling correction
  - Look at surrounding words
  - Can correct *form*/*from* error above

Correcting documents

- We are not interested in interactive spelling correction of documents (e.g., MS Word) in this class.
- In IR, we use document correction primarily for OCR'ed documents. (OCR = optical character recognition)
- The general philosophy in IR is: do not change the documents.

Correcting queries

- First: isolated word spelling correction
- Premise 1: There is a list of “correct words” from which the correct spellings come.
- Premise 2: We have a way of computing the distance between a misspelled word and a correct word.
- Simple spelling correction algorithm: return the “correct” word that has the smallest distance to the misspelled word.
- Example: *informaton* → *information*
- For the list of correct words, we can use the vocabulary of all words that occur in our collection.
- Why is this problematic?

Alternatives to using the term vocabulary

- A standard dictionary (Webster's, OED etc.)
- An industry-specific dictionary (for specialized IR systems)
- The term vocabulary of the collection, appropriately weighted

Distance between misspelled word and “correct” word

- We will study several alternatives.
- Edit distance and Levenshtein distance
- Weighted edit distance
- *k*-gram overlap

Edit distance

- The edit distance between string s_1 and string s_2 is the minimum number of basic operations that convert s_1 to s_2.
- Levenshtein distance: The admissible basic operations are insert, delete, and replace
- Levenshtein distance *dog*-*do*: 1
- Levenshtein distance *cat*-*cart*: 1
- Levenshtein distance *cat*-*cut*: 1
- Levenshtein distance *cat*-*act*: 2
- Damerau-Levenshtein distance *cat*-*act*: 1
- Damerau-Levenshtein includes transposition as a fourth possible operation.

Levenshtein distance: Computation

		f	a	s	t
	0	1	2	3	4
c	1	1	2	3	4
a	2	2	1	2	3
t	3	3	2	2	2
s	4	4	3	2	3

Levenshtein distance: Algorithm

```
LevenshteinDistance($s_1, s_2$)
for $i \leftarrow 0$ to $|s_1|$
do $m[i,0] = i$
for $j \leftarrow 0$ to $|s_2|$
do $m[0,j] = j$
for $i \leftarrow 1$ to $|s_1|$
do for $j \leftarrow 1$ to $|s_2|$
    do if $s_1[i] = s_2[j]$
          then $m[i,j] = \min\{m[i\text{-}1,j]+1, m[i,j\text{-}1]+1, m[i\text{-}1,j\text{-}1]\}$
          else $m[i,j] = \min\{m[i\text{-}1,j]+1, m[i,j\text{-}1]+1, m[i\text{-}1,j\text{-}1]+1\}$
return $m[|s_1|, |s_2|]$
```

Operations: insert (cost 1), delete (cost 1), replace (cost 1), copy (cost 0)

Levenshtein distance: Algorithm

```
LevenshteinDistance($s_1, s_2$)
for $i \leftarrow 0$ to $|s_1|$
do $m[i,0] = i$
for $j \leftarrow 0$ to $|s_2|$
do $m[0,j] = j$
for $i \leftarrow 1$ to $|s_1|$
do for $j \leftarrow 1$ to $|s_2|$
    do if $s_1[i] = s_2[j]$
          then $m[i,j] = \min\{m[i\text{-}1,j]+1, m[i,j\text{-}1]+1, m[i\text{-}1,j\text{-}1]\}$
          else $m[i,j] = \min\{m[i\text{-}1,j]+1, m[i,j\text{-}1]+1, m[i\text{-}1,j\text{-}1]+1\}$
return $m[|s_1|,|s_2|]$
```

Operations: insert (cost 1), delete (cost 1), replace (cost 1), copy (cost 0)

Levenshtein distance: Algorithm

```
LevenshteinDistance($s_1, s_2$)
for $i \leftarrow 0$ to $|s_1|$
do $m[i,0] = i$
for $j \leftarrow 0$ to $|s_2|$
do $m[0,j] = j$
for $i \leftarrow 1$ to $|s_1|$
do for $j \leftarrow 1$ to $|s_2|$
    do if $s_1[i] = s_2[j]$
          then $m[i,j] = \min\{m[i\text{-}1,j]+1, m[i,j\text{-}1]+1, m[i\text{-}1,j\text{-}1]\}$
          else $m[i,j] = \min\{m[i\text{-}1,j]+1, m[i,j\text{-}1]+1, m[i\text{-}1,j\text{-}1]+1\}$
return $m[|s_1|,|s_2|]$
```

Operations: insert (cost 1), delete (cost 1), replace (cost 1), copy (cost 0)

Levenshtein distance: Algorithm

```
LevenshteinDistance($s_1, s_2$)
for $i \leftarrow 0$ to $|s_1|$
do $m[i, 0] = i$
for $j \leftarrow 0$ to $|s_2|$
do $m[0, j] = j$
for $i \leftarrow 1$ to $|s_1|$
do for $j \leftarrow 1$ to $|s_2|$
      do if $s_1[i] = s_2[j]$
            then $m[i, j] = \min\{m[i\text{-}1, j]+1, m[i, j\text{-}1]+1, m[i\text{-}1, j\text{-}1]\}$
            else $m[i, j] = \min\{m[i\text{-}1, j]+1, m[i, j\text{-}1]+1, m[i\text{-}1, j\text{-}1]+1\}$
return $m[|s_1|, |s_2|]$
```

Operations: insert (cost 1), delete (cost 1), replace (cost 1), copy (cost 0)

Levenshtein distance: Algorithm

```
LevenshteinDistance($s_1, s_2$)
for $i \leftarrow 0$ to $|s_1|$
do $m[i,0] = i$
for $j \leftarrow 0$ to $|s_2|$
do $m[0,j] = j$
for $i \leftarrow 1$ to $|s_1|$
do for $j \leftarrow 1$ to $|s_2|$
    do if $s_1[i] = s_2[j]$
          then $m[i,j] = \min\{m[i-1,j]+1, m[i,j-1]+1, m[i-1,j-1]\}$
          else $m[i,j] = \min\{m[i-1,j]+1, m[i,j-1]+1, m[i-1,j-1]+1\}$
return $m[|s_1|,|s_2|]$
```

Operations: insert (cost 1), delete (cost 1), replace (cost 1), copy (cost 0)

Levenshtein distance: algorithm

```
LevenshteinDistance($s_1, s_2$)
for $i \leftarrow 1$ to $|s_1|$
do $m[i,0] = i$
for $j \leftarrow 0$ to $|s_2|$
do $m[0,j] = j$
for $i \leftarrow 0$ to $|s_1|$
do for $j \leftarrow 1$ to $|s_2|$
    do if $s_1[i] = s_2[j]$
          then $m[i,j] = \min\{m[i-1,j]+1, m[i,j-1]+1, m[i-1,j-1]\}$
          else $m[i,j] = \min\{m[i-1,j]+1, m[i,j-1]+1, m[i-1,j-1]+1\}$
return $m[|s_1|,|s_2|]$
```

Operations: insert, delete, replace, copy

Levenshtein distance: algorithm

```
LevenshteinDistance($s_1, s_2$)
for $i \leftarrow 1$ to $|s_1|$
do $m[i, 0] = i$
for $j \leftarrow 0$ to $|s_2|$
do $m[0, j] = j$
for $i \leftarrow 0$ to $|s_1|$
do for $j \leftarrow 1$ to $|s_2|$
    do if $s_1[i] = s_2[j]$
          then $m[i, j] = \min\{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]\}$
          else $m[i, j] = \min\{m[i-1, j]+1, m[i, j-1]+1, m[i-1, j-1]+1\}$
return $m[|s_1|, |s_2|]$
```

Operations: insert, delete, replace, copy

Levenshtein distance: Example

			f		a		s		t	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
c		**1**	*1*	*2*	**2**	3	**3**	4	**4**	5
		1	*2*	*1*	**2**	**2**	**3**	**3**	**4**	**4**
a		**2**	**2**	**2**	*1*	*3*	*3*	*4*	4	5
		2	3	**2**	*3*	*1*	*2*	*2*	**3**	**3**
t		**3**	**3**	**3**	3	**2**	*2*	*3*	*2*	*4*
		3	4	**3**	4	**2**	*3*	*2*	*3*	*2*
s		**4**	**4**	**4**	4	**3**	**2**	3	*3*	*3*
		4	5	**4**	5	**3**	4	**2**	*3*	*3*

Each cell of Levenshtein matrix

cost of getting here from my upper left neighbor (copy or replace)	cost of getting here from my upper neighbor (delete)
cost of getting here from my left neighbor (insert)	the minimum of the three possible "movements"; the cheapest way of getting here

Levenshtein distance: Example

			f		a		s		t	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
c		**1**	*1*	*2*	**2**	3	**3**	4	**4**	5
		1	*2*	*1*	**2**	**2**	**3**	**3**	**4**	**4**
a		**2**	**2**	**2**	*1*	*3*	*3*	*4*	4	5
		2	3	**2**	*3*	*1*	*2*	*2*	**3**	**3**
t		**3**	**3**	**3**	3	**2**	*2*	*3*	*2*	*4*
		3	4	**3**	4	**2**	*3*	*2*	*3*	*2*
s		**4**	**4**	**4**	4	**3**	**2**	3	*3*	*3*
		4	5	**4**	5	**3**	4	**2**	*3*	*3*

Dynamic programming (Cormen et al.)

- Optimal substructure: The optimal solution to the problem contains within it subsolutions, i.e., optimal solutions to subproblems.
- Overlapping subsolutions: The subsolutions overlap. These subsolutions are computed over and over again when computing the global optimal solution in a brute-force algorithm.
- Subproblem in the case of edit distance: what is the edit distance of two prefixes
- Overlapping subsolutions: We need most distances of prefixes 3 times – this corresponds to moving right, diagonally, down.

Weighted edit distance

- As above, but weight of an operation depends on the characters involved.
- Meant to capture keyboard errors, e.g., *m* more likely to be mistyped as *n* than as *q*.
- Therefore, replacing *m* by *n* is a smaller edit distance than by *q*.
- We now require a weight matrix as input.
- Modify dynamic programming to handle weights

Using edit distance for spelling correction

- Given query, first enumerate all character sequences within a preset (possibly weighted) edit distance
- Intersect this set with our list of “correct” words
- Then suggest terms in the intersection to the user.
- → exercise in a few slides

Exercise

1. Compute Levenshtein distance matrix for OSLO – SNOW
2. What are the Levenshtein editing operations that transform *cat* into *catcat*?

		s	n	o	w
	0	1 1	2 2	3 3	4 4
o	1 1				
s	2 2				
l	3 3				
o	4 4				

			s		n		o		w	
		0	1	1	2	2	3	3	4	4
o		1	1	2						
		1	2	?						
s		2								
		2								
l		3								
		3								
o		4								
		4								

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** **1**	**1** 2 2 **1**			
s	**2** **2**				
l	**3** **3**				
o	**4** **4**				

			s		n		o		w	
		0	1	1	2	2	3	3	4	4
o		1	1	2	2	3				
		1	2	1	2	?				
s		2								
		2								
l		3								
		3								
o		4								
		4								

		s	n	o	w
	0	1 1	2 2	3 3	4 4
o	1 1	1 2 2 1	2 3 2 2		
s	2 2				
l	3 3				
o	4 4				

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	2	4		
		1	2	**1**	**2**	**2**	3	?		
s		**2**								
		2								
l		**3**								
		3								
o		**4**								
		4								

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	
s	**2** **2**				
l	**3** **3**				
o	**4** **4**				

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	3	?
s		**2**								
		2								
l		**3**								
		3								
o		**4**								
		4								

		s	n	o	w
	 0	 **1** **1**	 **2** **2**	 **3** **3**	 **4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**				
l	**3** **3**				
o	**4** **4**				

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	1	2						
		2	3	?						
l		**3**								
		3								
o		**4**								
		4								

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**			
l	**3** **3**				
o	**4** **4**				

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	2	3				
		2	3	**1**	2	?				
l		**3**								
		3								
o		**4**								
		4								

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**	**2** 3 **2** **2**		
l	**3** **3**				
o	**4** **4**				

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	3	3		
		2	3	**1**	**2**	**2**	3	?		
l		**3**								
		3								
o		**4**								
		4								

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**		
		2	3	**1**	**2**	**2**	**3**	**3**		
l		**3**								
		3								
o		**4**								
		4								

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	3	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	?
l		**3**								
		3								
o		**4**								
		4								

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	**3**	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	**3**
l		**3**								
		3								
o		**4**								
		4								

		s	n	o	w
	 0	 **1** **1**	 **2** **2**	 **3** **3**	 **4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**	**2** 3 **2** **2**	**3** **3** **3** **3**	**3** 4 4 **3**
l	**3** **3**	3 2 4 ?			
o	**4** **4**				

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	**3**	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	**3**
l		**3**	3	**2**						
		3	4	**2**						
o		**4**								
		4								

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**	**2** 3 **2** **2**	**3** **3** **3** **3**	**3** 4 4 **3**
l	**3** **3**	3 **2** 4 **2**	2 3 3 ?		
o	**4** **4**				

		s	n	o	w
	0	 **1** **1**	 **2** **2**	 **3** **3**	 **4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**	**2** 3 **2** **2**	**3** **3** **3** **3**	**3** 4 4 **3**
l	**3** **3**	3 **2** 4 **2**	**2** 3 3 **2**		
o	**4** **4**				

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	**3**	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	**3**
l		**3**	3	**2**	**2**	3	3	4		
		3	4	**2**	3	**2**	3	?		
o		**4**								
		4								

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	**3**	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	**3**
l		**3**	3	**2**	**2**	3	**3**	4		
		3	4	**2**	3	**2**	**3**	**3**		
o		**4**								
		4								

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	**3**	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	**3**
l		**3**	3	**2**	**2**	3	**3**	4	4	4
		3	4	**2**	3	**2**	**3**	**3**	4	?
o		**4**								
		4								

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	**3**	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	**3**
l		**3**	3	**2**	**2**	3	**3**	4	**4**	**4**
		3	4	**2**	3	**2**	**3**	**3**	**4**	**4**
o		**4**								
		4								

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	**3**	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	**3**
l		**3**	3	**2**	**2**	3	**3**	4	**4**	**4**
		3	4	**2**	3	**2**	**3**	**3**	**4**	**4**
o		**4**	4	3						
		4	5	?						

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	**3**	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	**3**
l		**3**	3	**2**	**2**	3	**3**	4	**4**	**4**
		3	4	**2**	3	**2**	**3**	**3**	**4**	**4**
o		**4**	4	**3**						
		4	5	**3**						

		s	n	o	w
	 0	 **1** **1**	 **2** **2**	 **3** **3**	 **4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**	**2** 3 **2** **2**	**3** **3** **3** **3**	**3** 4 4 **3**
l	**3** **3**	3 **2** 4 **2**	**2** 3 3 **2**	**3** 4 **3** **3**	**4** **4** **4** **4**
o	**4** **4**	4 **3** 5 **3**	3 3 4 ?		

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	**3**	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	**3**
l		**3**	3	**2**	**2**	3	**3**	4	**4**	**4**
		3	4	**2**	3	**2**	**3**	**3**	**4**	**4**
o		**4**	4	**3**	**3**	**3**				
		4	5	**3**	4	**3**				

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**	**2** 3 **2** **2**	**3** **3** **3** **3**	**3** 4 4 **3**
l	**3** **3**	3 **2** 4 **2**	**2** 3 3 **2**	**3** 4 **3** **3**	**4** **4** **4** **4**
o	**4** **4**	4 **3** 5 **3**	**3** **3** 4 **3**	2 4 4 ?	

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	**3**	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	**3**
l		**3**	3	**2**	**2**	3	**3**	4	**4**	**4**
		3	4	**2**	3	**2**	**3**	**3**	**4**	**4**
o		**4**	4	**3**	**3**	**3**	**2**	4		
		4	5	**3**	4	**3**	4	**2**		

		s	n	o	w
	 0	 **1** **1**	 **2** **2**	 **3** **3**	 **4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**	**2** 3 **2** **2**	**3** **3** **3** **3**	**3** 4 4 **3**
l	**3** **3**	3 **2** 4 **2**	**2** 3 3 **2**	**3** 4 **3** **3**	**4** **4** **4** **4**
o	**4** **4**	4 **3** 5 **3**	**3** **3** 4 **3**	**2** 4 4 **2**	4 5 3 ?

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**	**2** 3 **2** **2**	**3** **3** **3** **3**	**3** 4 4 **3**
l	**3** **3**	3 **2** 4 **2**	**2** 3 3 **2**	**3** 4 **3** **3**	**4** **4** **4** **4**
o	**4** **4**	4 **3** 5 **3**	**3** **3** 4 **3**	**2** 4 4 **2**	4 5 **3** **3**

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	**3**	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	**3**
l		**3**	3	**2**	**2**	3	**3**	4	**4**	**4**
		3	4	**2**	3	**2**	**3**	**3**	**4**	**4**
o		**4**	4	**3**	**3**	**3**	**2**	4	4	5
		4	5	**3**	4	**3**	4	**2**	**3**	**3**

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	**3**	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	**3**
l		**3**	3	**2**	**2**	3	**3**	4	**4**	**4**
		3	4	**2**	3	**2**	**3**	**3**	**4**	**4**
o		**4**	4	**3**	**3**	**3**	**2**	4	4	5
		4	5	**3**	4	**3**	4	**2**	**3**	**3**

How do I read out the editing operations that transform OSLO into SNOW?

		s	n	o	w
	 0	 **1** **1**	 **2** **2**	 **3** **3**	 **4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**	**2** 3 **2** **2**	**3** **3** **3** **3**	**3** 4 4 **3**
l	**3** **3**	3 **2** 4 **2**	**2** 3 3 **2**	**3** 4 **3** **3**	**4** **4** **4** **4**
o	**4** **4**	4 **3** 5 **3**	**3** **3** 4 **3**	**2** 4 4 **2**	4 5 **3** **3**

cost	operation	input	output
1	insert	*	w

		s	n	o	w
	 0	 **1** **1**	 **2** **2**	 **3** **3**	 **4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**	**2** 3 **2** **2**	**3** **3** **3** **3**	**3** 4 4 **3**
l	**3** **3**	3 **2** 4 **2**	**2** 3 3 **2**	**3** 4 **3** **3**	**4** **4** **4** **4**
o	**4** **4**	4 **3** 5 **3**	**3** **3** 4 **3**	**2** 4 4 **2**	4 5 **3** **3**

cost	operation	input	output
0	(copy)	o	o
1	insert	*	w

			s		n		o		w	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
o		**1**	**1**	2	**2**	3	**2**	4	4	5
		1	2	**1**	**2**	**2**	3	**2**	**3**	**3**
s		**2**	**1**	2	**2**	3	**3**	**3**	**3**	4
		2	3	**1**	**2**	**2**	**3**	**3**	4	**3**
l		**3**	3	**2**	**2**	3	**3**	4	**4**	**4**
		3	4	**2**	3	**2**	**3**	**3**	**4**	**4**
o		**4**	4	**3**	**3**	**3**	**2**	4	4	5
		4	5	**3**	4	**3**	4	**2**	**3**	**3**

cost	operation	input	output
1	replace	l	n
0	(copy)	o	o
1	insert	*	w

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**	**2** 3 **2** **2**	**3** **3** **3** **3**	**3** 4 4 **3**
l	**3** **3**	3 **2** 4 **2**	**2** 3 3 **2**	**3** 4 **3** **3**	**4** **4** **4** **4**
o	**4** **4**	4 **3** 5 **3**	**3** **3** 4 **3**	**2** 4 4 **2**	4 5 **3** **3**

cost	operation	input	output
0	(copy)	s	s
1	replace	l	n
0	(copy)	o	o
1	insert	*	w

		s	n	o	w
	0	**1** **1**	**2** **2**	**3** **3**	**4** **4**
o	**1** **1**	**1** 2 2 **1**	**2** 3 **2** **2**	**2** 4 3 **2**	4 5 **3** **3**
s	**2** **2**	**1** 2 3 **1**	**2** 3 **2** **2**	**3** **3** **3** **3**	**3** 4 4 **3**
l	**3** **3**	3 **2** 4 **2**	**2** 3 3 **2**	**3** 4 **3** **3**	**4** **4** **4** **4**
o	**4** **4**	4 **3** 5 **3**	**3** **3** 4 **3**	**2** 4 4 **2**	4 5 **3** **3**

cost	operation	input	output
1	delete	o	*
0	(copy)	s	s
1	replace	l	n
0	(copy)	o	o
1	insert	*	w

			c		a		t		c		a		t	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**	**5**	**5**	**6**	**6**
c		**1**	**0**	2	2	3	3	4	**3**	5	5	6	6	7
		1	2	**0**	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**	**5**	**5**
a		**2**	2	**1**	**0**	2	2	3	3	4	**3**	5	5	6
		2	3	**1**	2	**0**	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
t		**3**	3	**2**	2	**1**	**0**	2	2	3	3	4	**3**	5
		3	4	**2**	3	**1**	2	**0**	**1**	**1**	**2**	**2**	**3**	**3**

		c	a	t	c	a	t
	0	1 1	2 2	3 3	4 4	5 5	6 6
c	1 1	0 2 2 0	2 3 1 1	3 4 2 2	3 5 3 3	5 6 4 4	6 7 5 5
a	2 2	2 1 3 1	0 2 2 0	2 3 1 1	3 4 2 2	3 5 3 3	5 6 4 4
t	3 3	3 2 4 2	2 1 3 1	0 2 2 0	2 3 1 1	3 4 2 2	3 5 3 3

cost	operation	input	output
1	insert	*	c
1	insert	*	a
1	insert	*	t
0	(copy)	c	c
0	(copy)	a	a
0	(copy)	t	t

		c	a	t	c	a	t
	0	1 1	2 2	3 3	4 4	5 5	6 6
c	1 1	0 2 2 0	2 3 1 1	3 4 2 2	3 5 3 3	5 6 4 4	6 7 5 5
a	2 2	2 1 3 1	0 2 2 0	2 3 1 1	3 4 2 2	3 5 3 3	5 6 4 4
t	3 3	3 2 4 2	2 1 3 1	0 2 2 0	2 3 1 1	3 4 2 2	3 5 3 3

cost	operation	input	output
0	(copy)	c	c
1	insert	*	a
1	insert	*	t
1	insert	*	c
0	(copy)	a	a
0	(copy)	t	t

			c		a		t		c		a		t	
		0	1	1	2	2	3	3	4	4	5	5	6	6
c		1	0	2	2	3	3	4	3	5	5	6	6	7
		1	2	0	1	1	2	2	3	3	4	4	5	5
a		2	2	1	0	2	2	3	3	4	3	5	5	6
		2	3	1	2	0	1	1	2	2	3	3	4	4
t		3	3	2	2	1	0	2	2	3	3	4	3	5
		3	4	2	3	1	2	0	1	1	2	2	3	3

cost	operation	input	output
0	(copy)	c	c
0	(copy)	a	a
1	insert	*	t
1	insert	*	c
1	insert	*	a
0	(copy)	t	t

			c		a		t		c		a		t	
		0	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**	**5**	**5**	**6**	**6**
c		**1**	**0**	2	2	3	3	4	**3**	5	5	6	6	7
		1	2	**0**	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**	**5**	**5**
a		**2**	2	**1**	**0**	2	2	3	3	4	**3**	5	5	6
		2	3	**1**	2	**0**	**1**	**1**	**2**	**2**	**3**	**3**	**4**	**4**
t		**3**	3	**2**	2	**1**	**0**	2	2	3	3	4	**3**	5
		3	4	**2**	3	**1**	2	**0**	**1**	**1**	**2**	**2**	**3**	**3**

cost	operation	input	output
0	(copy)	c	c
0	(copy)	a	a
0	(copy)	t	t
1	insert	*	c
1	insert	*	a
1	insert	*	t

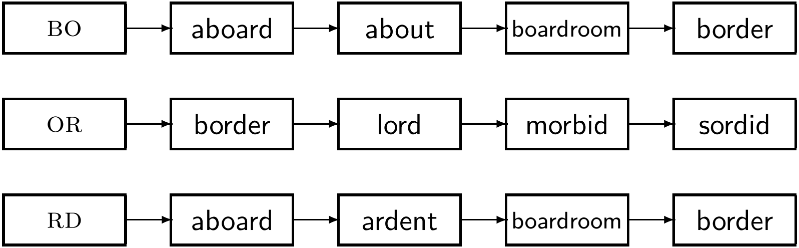
Spelling correction

- Now that we can compute edit distance: how to use it for isolated word spelling correction – this is the last slide in this section.
- *k*-gram indexes for isolated word spelling correction.
- Context-sensitive spelling correction
- General issues

k-gram indexes for spelling correction

- Enumerate all *k*-grams in the query term
- Example: bigram index, misspelled word *bordroom*
- Bigrams: *bo, or, rd, dr, ro, oo, om*
- Use the *k*-gram index to retrieve "correct" words that match query term *k*-grams
- Threshold by number of matching *k*-grams
- E.g., only vocabulary terms that differ by at most 3 *k*-grams

k-gram indexes for spelling correction: *bordroom*



BO → aboard → about → boardroom → border

OR → border → lord → morbid → sordid

RD → aboard → ardent → boardroom → border

Context-sensitive spelling correction

- Our example was: *an asteroid that fell form the sky*
- How can we correct *form* here?
- One idea: hit-based spelling correction
  - Retrieve "correct" terms close to each query term
  - for *flew form munich*: *flea* for *flew*, *from* for *form*, *munch* for *munich*
  - Now try all possible resulting phrases as queries with one word "fixed" at a time
  - Try query *"flea form munich"*
  - Try query *"flew from munich"*
  - Try query *"flew form munch*"
  - The correct query *"flew from munich"* has the most hits.
- Suppose we have 7 alternatives for *flew*, 20 for *form* and 3 for *munich*, how many "corrected" phrases will we enumerate?

Context-sensitive spelling correction

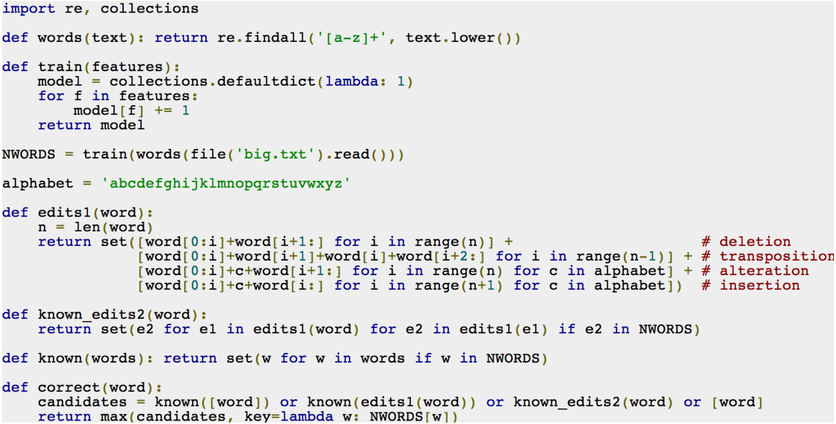
- The “hit-based” algorithm we just outlined is not very efficient.
- More efficient alternative: look at “collection” of queries, not documents.

General issues in spelling correction

- User interface
  - automatic vs. suggested correction
  - *Did you mean* only works for one suggestion.
  - What about multiple possible corrections?
  - Tradeoff: simple vs. powerful UI
- Cost
  - Spelling correction is potentially expensive.
  - Avoid running on every query?
  - Maybe just on queries that match few documents.
  - Guess: Spelling correction of major search engines is efficient enough to be run on every query.

Exercise: Understand Peter Norvig's spelling corrector

```
import re, collections
def words(text): return re.findall('[a-z]+', text.lower())
def train(features):
    model = collections.defaultdict(lambda: 1)
    for f in features:
        model[f] += 1
    return model
NWORDS = train(words(file('big.txt').read()))
alphabet = 'abcdefghijklmnopqrstuvwxyz'
def edits1(word):
   splits     = [(word[:i], word[i:]) for i in range(len(word) + 1)]
   deletes    = [a + b[1:] for a, b in splits if b]
   transposes = [a + b[1] + b[0] + b[2:] for a, b in splits if len(b) gt 1]
   replaces   = [a + c + b[1:] for a, b in splits for c in alphabet if b]
   inserts    = [a + c + b     for a, b in splits for c in alphabet]
   return set(deletes + transposes + replaces + inserts)
def known_edits2(word):
    return set(e2 for e1 in edits1(word) for e2 in edits1(e1) if e2 in NWORDS)
def known(words): return set(w for w in words if w in NWORDS)
def correct(word):
    candidates = known([word]) or known(edits1(word)) or known_edits2(word) or [word]
    return max(candidates, key=NWORDS.get)
```



```
import re, collections

def words(text): return re.findall('[a-z]+', text.lower())

def train(features):
    model = collections.defaultdict(lambda: 1)
    for f in features:
        model[f] += 1
    return model

NWORDS = train(words(file('big.txt').read()))

alphabet = 'abcdefghijklmnopqrstuvwxyz'

def edits1(word):
    n = len(word)
    return set([word[0:i]+word[i+1:] for i in range(n)] +                     # deletion
               [word[0:i]+word[i+1]+word[i]+word[i+2:] for i in range(n-1)] + # transposition
               [word[0:i]+c+word[i+1:] for i in range(n) for c in alphabet] + # alteration
               [word[0:i]+c+word[i:] for i in range(n+1) for c in alphabet])  # insertion

def known_edits2(word):
    return set(e2 for e1 in edits1(word) for e2 in edits1(e1) if e2 in NWORDS)

def known(words): return set(w for w in words if w in NWORDS)

def correct(word):
    candidates = known([word]) or known(edits1(word)) or known_edits2(word) or [word]
    return max(candidates, key=lambda w: NWORDS[w])
```

http://norvig.com/spell-correct.html

Soundex

- Soundex is the basis for finding phonetic (as opposed to orthographic) alternatives.
- Example: *chebyshev* / *tchebyscheff*
- Algorithm:
    - Turn every token to be indexed into a 4-character reduced form
    - Do the same with query terms
    - Build and search an index on the reduced forms

Soundex algorithm

1. Retain the first letter of the term.
2. Change all occurrences of the following letters to '0' (zero): A, E, I, O, U, H, W, Y
3. Change letters to digits as follows:
   - B, F, P, V to 1
   - C, G, J, K, Q, S, X, Z to 2
   - D,T to 3
   - L to 4
   - M, N to 5
   - R to 6
4. Repeatedly remove one out of each pair of consecutive identical digits
5. Remove all zeros from the resulting string; pad the resulting string with trailing zeros and return the first four positions, which will consist of a letter followed by three digits

Example: Soundex of *HERMAN*

- Retain H
- *ERMAN* → *0RM0N*
- *0RM0N* → *06505*
- *06505* → *06505*
- *06505* → *655*
- Return *H655*
- Note: *HERMANN* will generate the same code

How useful is Soundex?

- Not very – for information retrieval
- Ok for “high recall” tasks in other applications (e.g., Interpol)
- Zobel and Dart (1996) suggest better alternatives for phonetic matching in IR.

Exercise

- Compute Soundex code of your last name

Take-away

- Tolerant retrieval: What to do if there is no exact match between query term and document term
- Wildcard queries
- Spelling correction

Resources

- Chapter 3 of IIR
- Resources at `https://www.fi.muni.cz/~sojka/PV211/` and `http://cislmu.org`, materials in MU IS and FI MU library
    - trie vs hash vs ternary tree
    - Soundex demo
    - Edit distance demo
    - Peter Norvig's spelling corrector
    - Google: wild card search, spelling correction gone wrong, a misspelling that is more frequent that the correct spelling
    - NLP IR boolean search system example: Sketch Engine – `https://ske.fi.muni.cz`

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 4: Index construction
Handout version

Petr Sojka, Hinrich Schütze et al.

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

2020-03-04

Overview

1. Introduction
2. BSBI algorithm
3. SPIMI algorithm
4. Distributed indexing
5. Dynamic indexing

Take-away

- Two index construction algorithms: BSBI (simple) and SPIMI (more realistic)
- Distributed index construction: MapReduce
- Dynamic index construction: how to keep the index up-to-date as the collection changes

Hardware basics

- Many design decisions in information retrieval are based on hardware constraints.
- We begin by reviewing hardware basics that we'll need in this course.

Hardware basics

- Access to data is much faster in memory than on disk. (roughly a factor of 10 SSD, 100+ for rotational disks)
- Disk seeks are "idle" time: No data is transferred from disk while the disk head is being positioned.
- To optimize transfer time from disk to memory: one large chunk is faster than many small chunks.
- Disk I/O is block-based: Reading and writing of entire blocks (as opposed to smaller chunks). Block sizes: 8KB to 256 KB
- Assuming an efficient decompression algorithm, the total time of reading and then decompressing compressed data is usually less than reading uncompressed data.
- Servers used in IR systems typically have many GBs of main memory and TBs of disk space.
- Fault tolerance is expensive: It's cheaper to use many regular machines than one fault tolerant machine.

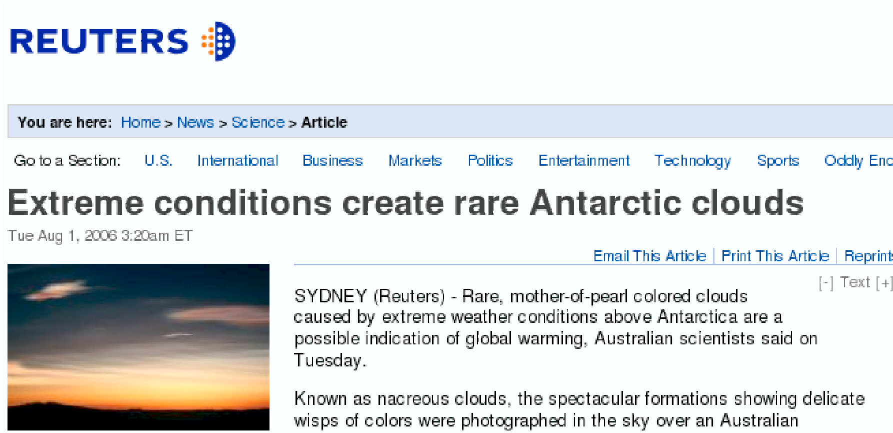
Some stats (ca. 2008)

symbol	statistic	value
s	average seek time	5 ms $= 5 \times 10^{-3}$ s
b	transfer time per byte	0.02 μs $= 2 \times 10^{-8}$ s
	processor's clock rate	10^9 s^{-1}
p	lowlevel operation (e.g., compare & swap a word)	0.01 μs $= 10^{-8}$ s
	size of main memory	several GB
	size of disk space	1 TB or more

RCV1 collection

- Shakespeare's collected works are not large enough for demonstrating many of the points in this course.
- As an example for applying scalable index construction algorithms, we will use the Reuters RCV1 collection.
- English newswire articles sent over the wire in 1995 and 1996 (one year).

A Reuters RCV1 document



REUTERS

You are here: Home > News > Science > Article

Go to a Section: U.S. International Business Markets Politics Entertainment Technology Sports Oddly Enc

Extreme conditions create rare Antarctic clouds

Tue Aug 1, 2006 3:20am ET

Email This Article | Print This Article | Reprint

[-] Text [+]

SYDNEY (Reuters) - Rare, mother-of-pearl colored clouds caused by extreme weather conditions above Antarctica are a possible indication of global warming, Australian scientists said on Tuesday.

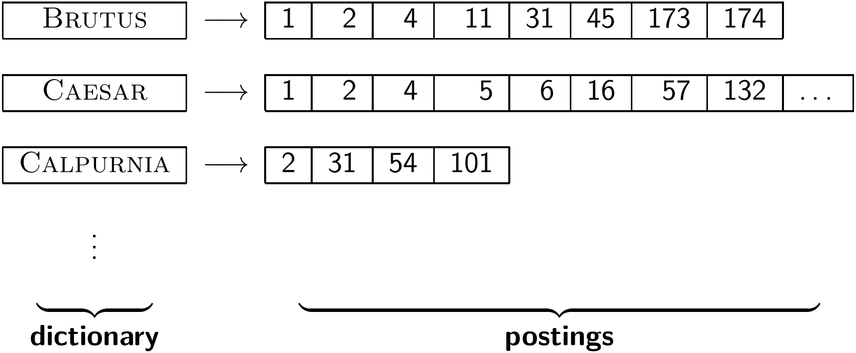
Known as nacreous clouds, the spectacular formations showing delicate wisps of colors were photographed in the sky over an Australian

Reuters RCV1 statistics

N	documents	800,000
L	tokens per document	200
M	terms (= word types)	400,000
	bytes per token (incl. spaces/punct.)	6
	bytes per token (without spaces/punct.)	4.5
	bytes per term (= word type)	7.5
T	non-positional postings	100,000,000

Exercise: Average frequency of a term (how many tokens)? 4.5 bytes per word token vs. 7.5 bytes per word type: why the difference? How many positional postings?

Goal: construct the inverted index



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

Index construction in IIR 1: Sort postings in memory

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

⟹

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

Scaling index construction

- How can we construct an index for very large collections?
- Taking into account the hardware constraints we just learned about . . .
- . . . memory, disk, speed etc.

Sort-based index construction

- As we build index, we parse docs one at a time.
- The final postings for any term are incomplete until the end.
- Can we keep all postings in memory and then do the sort in-memory at the end?
- No, not for large collections
- Thus: We need to store intermediate results on disk.

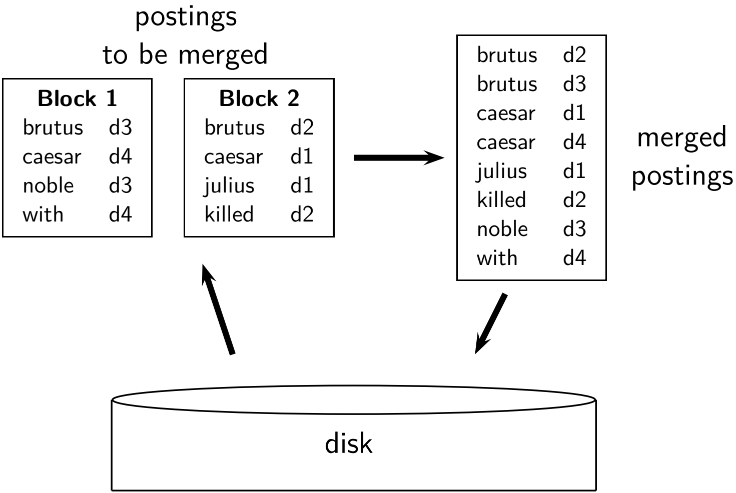
Same algorithm for disk?

- Can we use the same index construction algorithm for larger collections, but by using disk instead of memory?
- No: Sorting very large sets of records on disk is too slow – too many disk seeks.
- We need an external sorting algorithm.

"External" sorting algorithm (using few disk seeks)

- We must sort $T = 100{,}000{,}000$ non-positional postings.
    - Each posting has size 12 bytes (4+4+4: termID, docID, term frequency).
- Define a block to consist of 10,000,000 such postings
    - We can easily fit that many postings into memory.
    - We will have 10 such blocks for RCV1.
- Basic idea of algorithm:
    - For each block: (i) accumulate postings, (ii) sort in memory, (iii) write to disk
    - Then merge the blocks into one long sorted order.

Merging two blocks



postings
to be merged

Block 1

brutus	d3
caesar	d4
noble	d3
with	d4

Block 2

brutus	d2
caesar	d1
julius	d1
killed	d2

brutus	d2
brutus	d3
caesar	d1
caesar	d4
julius	d1
killed	d2
noble	d3
with	d4

merged
postings

Blocked Sort-Based Indexing

```
BSBIndexConstruction()
n ← 0
while (all documents have not been processed)
do n ← n + 1
    block ← ParseNextBlock()
    BSBI-Invert(block)
    WriteBlockToDisk(block, $f_n$)
MergeBlocks($f_1, \ldots, f_n; f_{\text{merged}}$)
```

Problem with sort-based algorithm

- Our assumption was: we can keep the dictionary in memory.
- We need the dictionary (which grows dynamically) in order to implement a term to termID mapping.
- Actually, we could work with term,docID postings instead of termID,docID postings . . .
- . . . but then intermediate files become very large. (We would end up with a scalable, but very slow index construction method.)

Single-pass in-memory indexing

- Abbreviation: SPIMI
- Key idea 1: Generate separate dictionaries for each block – no need to maintain term-termID mapping across blocks.
- Key idea 2: Don't sort. Accumulate postings in postings lists as they occur.
- With these two ideas we can generate a complete inverted index for each block.
- These separate indexes can then be merged into one big index.

SPIMI-Invert

```
SPIMI-Invert(token_stream)
output_file ← NewFile()
dictionary ← NewHash()
while (free memory available)
do token ← next(token_stream)
    if term(token) $\notin$ dictionary
      then postings_list ← AddToDictionary(dictionary,term(token))
      else postings_list ← GetPostingsList(dictionary,term(token))
    if full(postings_list)
      then postings_list ← DoublePostingsList(dictionary,term(token
    AddToPostingsList(postings_list,docID(token))
sorted_terms ← SortTerms(dictionary)
WriteBlockToDisk(sorted_terms,dictionary,output_file)
return output_file
```

Merging of blocks is analogous to BSBI.

SPIMI: Compression

- Compression makes SPIMI even more efficient.
  - Compression of terms
  - Compression of postings
  - See next lecture

Distributed indexing

- For web-scale indexing (don't try this at home!): must use a distributed computer cluster
- Individual machines are fault-prone.
    - Can unpredictably slow down or fail.
- How do we exploit such a pool of machines?

Google data centers (2007 estimates; Gartner)

- Google data centers mainly contain commodity machines.
- Data centers are distributed all over the world.
- 1 million servers, 3 million processors/cores
- Google installs 100,000 servers each quarter.
- Based on expenditures of 200–250 million dollars per year
- This would be 10% of the computing capacity of the world!
- If in a non-fault-tolerant system with 1000 nodes, each node has 99.9% uptime, what is the uptime of the system (assuming it does not tolerate failures)?
- Answer: 37%
- Suppose a server will fail after 3 years. For an installation of 1 million servers, what is the interval between machine failures?
- Answer: less than two minutes

Distributed indexing

- Maintain a master machine directing the indexing job – considered “safe”
- Break up indexing into sets of parallel tasks
- Master machine assigns each task to an idle machine from a pool.

Parallel tasks

- We will define two sets of parallel tasks and deploy two types of machines to solve them:
  - Parsers
  - Inverters
- Break the input document collection into splits (corresponding to blocks in BSBI/SPIMI)
- Each split is a subset of documents.

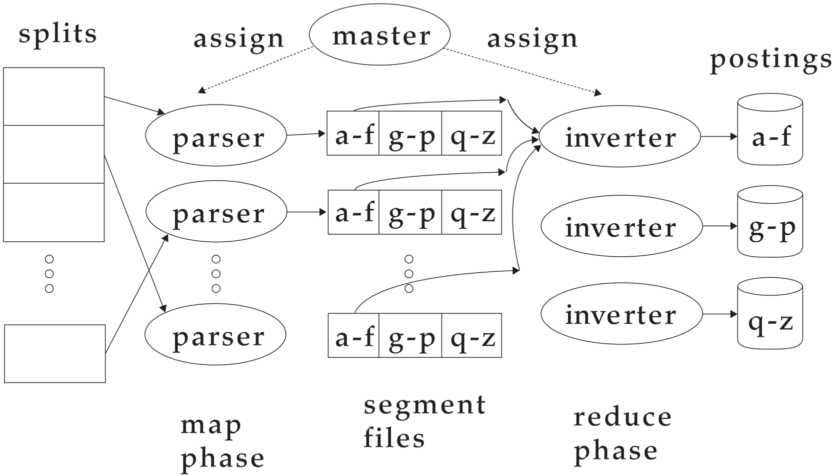
Parsers

- Master assigns a split to an idle parser machine.
- Parser reads a document at a time and emits (termID,docID)-pairs.
- Parser writes pairs into j term-partitions.
- Each for a range of terms' first letters
  - E.g., a–f, g–p, q–z (here: $j = 3$)

Inverters

- An inverter collects all (termID,docID) pairs (= postings) for one term-partition (e.g., for a–f).
- Sorts and writes to postings lists

Data flow


splits
assign
master
assign
postings
parser
a-f
g-p
q-z
inverter
a-f
parser
a-f
g-p
q-z
inverter
g-p
parser
a-f
g-p
q-z
inverter
q-z
map phase
segment files
reduce phase

MapReduce

- The index construction algorithm we just described is an instance of MapReduce.
- MapReduce is a robust and conceptually simple framework for distributed computing . . .
- . . . without having to write code for the distribution part.
- The Google indexing system (ca. 2002) consisted of a number of phases, each implemented in MapReduce.
- Index construction was just one phase.
- Another phase: transform term-partitioned into document-partitioned index.

Index construction in MapReduce

Schema of map and reduce functions

map:	input	$\rightarrow$ list(k, v)
reduce:	(k,list(v))	$\rightarrow$ output

Instantiation of the schema for index construction

map:	web collection	$\rightarrow$ list(termID, docID)
reduce:	($\langle$termID$_1$, list(docID)$\rangle$, $\langle$termID$_2$, list(docID)$\rangle$, ...)	$\rightarrow$ (postings_list$_1$, postings_list$_2$, ...)

Example for index construction

map:	d_2 : C DIED. d_1 : C CAME, C C'ED.	$\rightarrow$ ($\langle$C, $d_2\rangle$, $\langle$DIED,$d_2\rangle$, $\langle$C,$d_1\rangle$, $\langle$CAME,$d_1\rangle$, $\langle$C,$d_1\rangle$, $\langle$C'ED,$d_1\rangle$)
reduce:	($\langle$C,(d_2,d_1,d_1)$\rangle$,$\langle$DIED,(d_2)$\rangle$,$\langle$CAME,(d_1)$\rangle$,$\langle$C'ED,(d_1)$\rangle$)	$\rightarrow$ ($\langle$C,(d_1:2,d_2:1)$\rangle$,$\langle$DIED,(d_2:1)$\rangle$,$\langle$CAME,(d_1:1)$\rangle$,$\langle$C'ED,(d_1:1)$\rangle$)

Exercise

- What information does the task description contain that the master gives to a parser?
- What information does the parser report back to the master upon completion of the task?
- What information does the task description contain that the master gives to an inverter?
- What information does the inverter report back to the master upon completion of the task?

Dynamic indexing

- Up to now, we have assumed that collections are static.
- They rarely are: Documents are inserted, deleted and modified.
- This means that the dictionary and postings lists have to be dynamically modified.

Dynamic indexing: Simplest approach

- Maintain big main index on disk
- New docs go into small auxiliary index in memory.
- Search across both, merge results
- Periodically, merge auxiliary index into big index
- Deletions:
  - Invalidation bit-vector for deleted docs
  - Filter docs returned by index using this bit-vector

Issue with auxiliary and main index

- Frequent merges
- Poor search performance during index merge

Logarithmic merge

- Logarithmic merging amortizes the cost of merging indexes over time.
  - → Users see smaller effect on response times.
- Maintain a series of indexes, each twice as large as the previous one.
- Keep smallest (Z_0) in memory
- Larger ones (I_0, I_1, . . .) on disk
- If Z_0 gets too big ($> n$), write to disk as I_0
- . . . or merge with I_0 (if I_0 already exists) and write merger to I_1 etc.

```
LMergeAddToken($indexes, Z_0, token$)
  $Z_0 \leftarrow$ Merge($Z_0, \{token\}$)
  if $|Z_0| = n$
    then for $i \leftarrow 0$ to $\infty$
           do if $I_i \in indexes$
                then $Z_{i+1} \leftarrow$ Merge($I_i, Z_i$)
                     ($Z_{i+1}$ is a temporary index on disk.)
                     $indexes \leftarrow indexes - \{I_i\}$
                else $I_i \leftarrow Z_i$    ($Z_i$ becomes the permanent index $I_i$.)
                     $indexes \leftarrow indexes \cup \{I_i\}$
                     Break
         $Z_0 \leftarrow \emptyset$
```

```
LogarithmicMerge()
  $Z_0 \leftarrow \emptyset$    ($Z_0$ is the in-memory index.)
  $indexes \leftarrow \emptyset$
  while true
  do LMergeAddToken($indexes, Z_0$, getNextToken())
```

Binary numbers: $I_3 I_2 I_1 I_0 = 2^3 2^2 2^1 2^0$

- 0001
- 0010
- 0011
- 0100
- 0101
- 0110
- 0111
- 1000
- 1001
- 1010
- 1011
- 1100

Logarithmic merge

- Number of indexes bounded by $O(\log T)$ (T is total number of postings read so far)
- So query processing requires the merging of $O(\log T)$ indexes
- Time complexity of index construction is $O(T \log T)$.
  - . . . because each of T postings is merged $O(\log T)$ times.
- Auxiliary index: index construction time is $O(T^2)$ as each posting is touched in each merge.
  - Suppose auxiliary index has size a
  - $a + 2a + 3a + 4a + \ldots + na = a\frac{n(n+1)}{2} = O(n^2)$
- So logarithmic merging is an order of magnitude more efficient.

Dynamic indexing at large search engines

- Often a combination
  - Frequent incremental changes
  - Rotation of large parts of the index that can then be swapped in
  - Occasional complete rebuild (becomes harder with increasing size – not clear if Google can do a complete rebuild)

Building positional indexes

- Basically the same problem except that the intermediate data structures are large.

Take-away

- Two index construction algorithms: BSBI (simple) and SPIMI (more realistic)
- Distributed index construction: MapReduce
- Dynamic index construction: how to keep the index up-to-date as the collection changes

Resources

- Chapter 4 of IIR
- Resources at `https://www.fi.muni.cz/~sojka/PV211/` and `http://cislmu.org`, materials in MU IS and FI MU library
  - Original publication on MapReduce by Dean and Ghemawat (2004)
  - Original publication on SPIMI by Heinz and Zobel (2003)
  - YouTube video: Google data centers

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 5: Index compression
Handout version

Petr Sojka, Hinrich Schütze et al.

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

2020-03-04

Overview

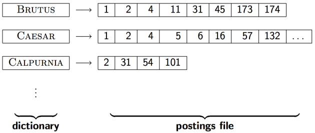
1. Compression
2. Term statistics
3. Dictionary compression
4. Postings compression

Roadmap

- Today: index compression, and vector space model
- Next week: the whole picture of complete search system, scoring and ranking

Take-away today

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

⋮

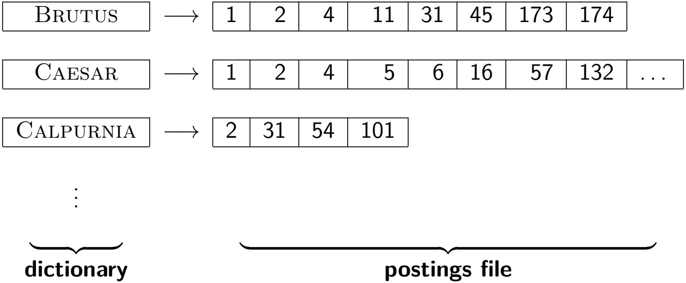
dictionary

postings file

- Motivation for compression in information retrieval systems
- How can we compress the dictionary component of the inverted index?
- How can we compress the postings component of the inverted index?
- Term statistics: how are terms distributed in document collections?

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 file**

Today:

- How much space do we need for the dictionary?
- How much space do we need for the postings file?
- How can we compress them?

Why compression? (in general)

- Use less disk space (saves money).
- Keep more stuff in memory (increases speed).
- Increase speed of transferring data from disk to memory (again, increases speed).

  [read compressed data and decompress in memory]
  is faster than
  [read uncompressed data]
- Premise: Decompression algorithms are fast.
- This is true of the decompression algorithms we will use.

Why compression in information retrieval?

- First, we will consider space for dictionary:
  - Main motivation for dictionary compression: make it small enough to keep in main memory.
- Then for the postings file
  - Motivation: reduce disk space needed, decrease time needed to read from disk.
  - Note: Large search engines keep significant part of postings in memory.
- We will devise various compression schemes for dictionary and postings.

Lossy vs. lossless compression

- Lossy compression: Discard some information
- Several of the preprocessing steps we frequently use can be viewed as lossy compression:
  - downcasing, stop words, porter, number elimination
- Lossless compression: All information is preserved.
  - What we mostly do in index compression

Model collection: The Reuters collection

symbol	statistic	value
N	documents	800,000
L	avg. # word tokens per document	200
M	word types	400,000
	avg. # bytes per word token (incl. spaces/punct.)	6
	avg. # bytes per word token (without spaces/punct.)	4.5
	avg. # bytes per word type	7.5
T	non-positional postings	100,000,000

Effect of preprocessing for Reuters

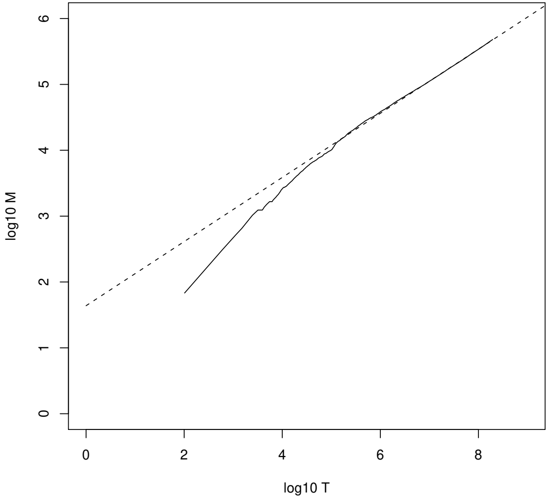
	word types (terms)			non-positional postings			positional postings (word tokens)		
size of	dictionary			non-positional index			positional index		
	size	Δ	cml	size	Δ	cml	size	Δ	cml
unfiltered	484,494			109,971,179			197,879,290		
no numbers	473,723	-2	-2	100,680,242	-8	-8	179,158,204	-9	-9
case folding	391,523	-17	-19	96,969,056	-3	-12	179,158,204	-0	-9
30 stopw's	391,493	-0	-19	83,390,443	-14	-24	121,857,825	-31	-38
150 stopw's	391,373	-0	-19	67,001,847	-30	-39	94,516,599	-47	-52
stemming	322,383	-17	-33	63,812,300	-4	-42	94,516,599	-0	-52

Explain differences between numbers non-positional vs positional: -3 vs 0, -14 vs -31, -30 vs -47, -4 vs 0

How big is the term vocabulary?

- That is, how many distinct words are there?
- Can we assume there is an upper bound?
- Not really: At least $70^{20} \approx 10^{37}$ different words of length 20.
- The vocabulary will keep growing with collection size.
- Heaps' law: $M = kT^b$
- M is the size of the vocabulary, T is the number of tokens in the collection.
- Typical values for the parameters k and b are: $30 \leq k \leq 100$ and $b \approx 0.5$.
- Heaps' law is linear in log-log space.
  - It is the simplest possible relationship between collection size and vocabulary size in log-log space.
  - Empirical law

Heaps' law for Reuters



Vocabulary size M as a function of collection size T (number of tokens) for Reuters-RCV1. For these data, the dashed line $\log_{10} M = 0.49 * \log_{10} T + 1.64$ is the best least squares fit. Thus, $M = 10^{1.64} T^{0.49}$ and $k = 10^{1.64} \approx 44$ and $b = 0.49$.

Empirical fit for Reuters

- Good, as we just saw in the graph.
- Example: for the first 1,000,020 tokens Heaps' law predicts 38,323 terms:

$$44 \times 1{,}000{,}020^{0.49} \approx 38{,}323$$

- The actual number is 38,365 terms, very close to the prediction.
- Empirical observation: fit is good in general.

Exercise

1. What is the effect of including spelling errors vs. automatically correcting spelling errors on Heaps' law?
2. Compute vocabulary size M
   - Looking at a collection of web pages, you find that there are 3,000 different terms in the first 10,000 tokens and 30,000 different terms in the first 1,000,000 tokens.
   - Assume a search engine indexes a total of 20,000,000,000 (2×10^{10}) pages, containing 200 tokens on average
   - What is the size of the vocabulary of the indexed collection as predicted by Heaps' law?

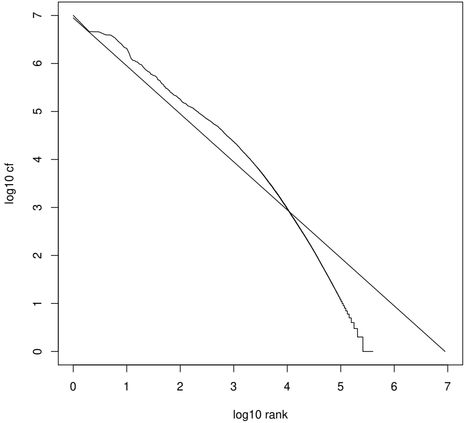
Zipf's law

- Now we have characterized the growth of the vocabulary in collections.
- We also want to know how many frequent vs. infrequent terms we should expect in a collection.
- In natural language, there are a few very frequent terms and very many very rare terms.
- Zipf's law: The i^{th} most frequent term has frequency cf_i proportional to $1/i$.
- $\mathrm{cf}_i \propto \frac{1}{i}$
- cf_i is collection frequency: the number of occurrences of the term t_i in the collection.

Zipf's law

- Zipf's law: The i^{th} most frequent term has frequency proportional to $1/i$.
- $\mathrm{cf}_i \propto \frac{1}{i}$
- cf is collection frequency: the number of occurrences of the term in the collection.
- So if the most frequent term (*the*) occurs cf_1 times, then the second most frequent term (*of*) has half as many occurrences $\mathrm{cf}_2 = \frac{1}{2}\mathrm{cf}_1 \ldots$
- ... and the third most frequent term (*and*) has a third as many occurrences $\mathrm{cf}_3 = \frac{1}{3}\mathrm{cf}_1$ etc.
- Equivalent: $\mathrm{cf}_i = ci^k$ and $\log \mathrm{cf}_i = \log c + k \log i$ (for $k = -1$)
- Example of a power law

Zipf's law for Reuters



Fit is not great. What is important is the key insight: Few frequent terms, many rare terms.

Dictionary compression

- The dictionary is small compared to the postings file.
- But we want to keep it in memory.
- Also: competition with other applications, cell phones, onboard computers, fast startup time
- So compressing the dictionary is important.

Recall: Dictionary as array of fixed-width entries

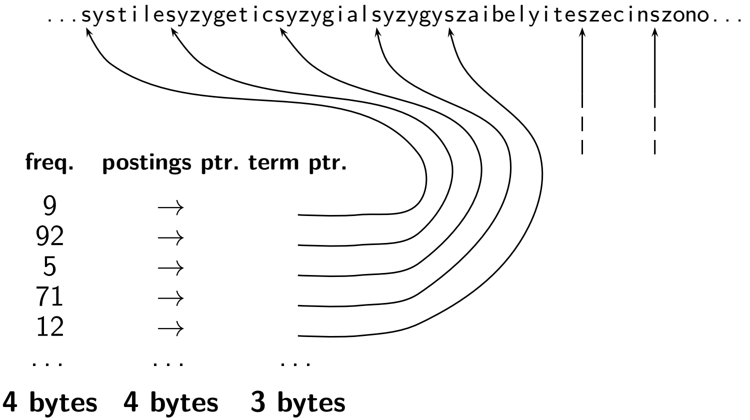
	term	document frequency	pointer to postings list
	a	656,265	⟶
	aachen	65	⟶
	. . .	. . .	. . .
	zulu	221	⟶
space needed:	20 bytes	4 bytes	4 bytes

Space for Reuters: $(20+4+4)*400{,}000 = 11.2$ MB

Fixed-width entries are bad.

- Most of the bytes in the term column are wasted.
  - We allot 20 bytes for terms of length 1.
- We cannot handle HYDROCHLOROFLUOROCARBONS and SUPERCALIFRAGILISTICEXPIALIDOCIOUS
- Average length of a term in English: 8 characters (or a little bit less)
- How can we use on average 8 characters per term?

Dictionary as a string



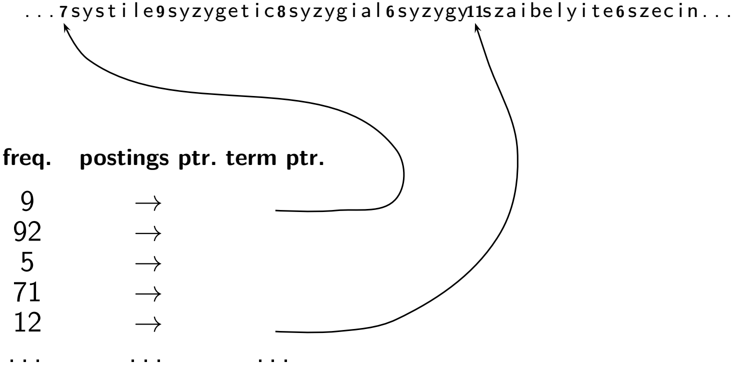
...systilesyzygeticsyzygialsyzygyszaibelyiteszecinszono...

freq.	postings ptr.	term ptr.
9	→	
92	→	
5	→	
71	→	
12	→	
...	...	...
4 bytes	**4 bytes**	**3 bytes**

Space for dictionary as a string

- 4 bytes per term for frequency
- 4 bytes per term for pointer to postings list
- 8 bytes (on average) for term in string
- 3 bytes per pointer into string (need $\log_2 8 \cdot 400{,}000 < 24$ bits to resolve $8 \cdot 400{,}000$ positions)
- Space: $400{,}000 \times (4 + 4 + 3 + 8) = 7.6$ MB (compared to 11.2 MB for fixed-width array)

Dictionary as a string with blocking



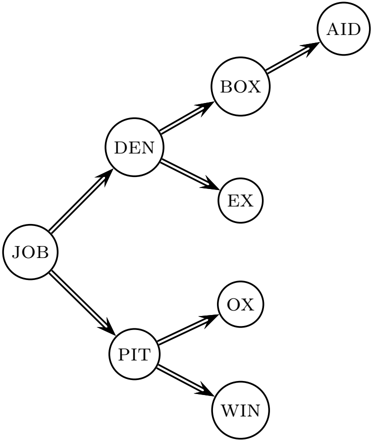
. . . **7**systile**9**syzygetic**8**syzygial**6**syzygy**11**szaibelyite**6**szecin. . .

freq.	**postings ptr.**	**term ptr.**
9	→	
92	→	
5	→	
71	→	
12	→	
. . .	. . .	. . .

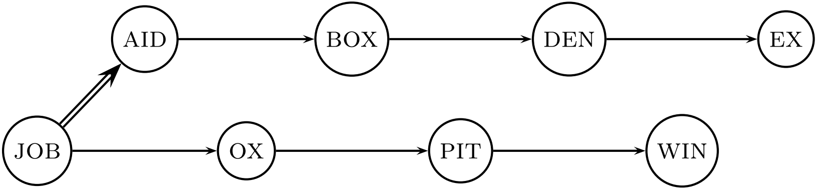
Space for dictionary as a string with blocking

- Example block size $k = 4$
- Where we used 4×3 bytes for term pointers without blocking . . .
- . . . we now use 3 bytes for one pointer plus 4 bytes for indicating the length of each term.
- We save $12 - (3 + 4) = 5$ bytes per block.
- Total savings: $400{,}000/4 * 5 = 0.5$ MB
- This reduces the size of the dictionary from 7.6 MB to 7.1 MB.

Lookup of a term without blocking



Lookup of a term with blocking: (slightly) slower



Front coding

One block in blocked compression ($k = 4$) ...

8 a u t o m a t a **8** a u t o m a t e **9** a u t o m a t i c **10** a u t o m a t i o n

$\Downarrow$

...further compressed with front coding.

8 a u t o m a t $*$ a **1** $\diamond$ e **2** $\diamond$ i c **3** $\diamond$ i o n

Dictionary compression for Reuters: Summary

data structure	size in MB
dictionary, fixed-width	11.2
dictionary, term pointers into string	7.6
$\sim$, with blocking, $k = 4$	7.1
$\sim$, with blocking & front coding	5.9

Exercise

- Which prefixes should be used for front coding? What are the tradeoffs?
- Input: list of terms (= the term vocabulary)
- Output: list of prefixes that will be used in front coding

Postings compression

- The postings file is much larger than the dictionary, factor of at least 10.
- Key desideratum: store each posting compactly
- A posting for our purposes is a docID.
- For Reuters (800,000 documents), we would use 32 bits per docID when using 4-byte integers.
- Alternatively, we can use $\log_2 800{,}000 \approx 19.6 < 20$ bits per docID.
- Our goal: use a lot less than 20 bits per docID.

Key idea: Store gaps instead of docIDs

- Each postings list is ordered in increasing order of docID.
- Example postings list: COMPUTER: 283154, 283159, 283202, . . .
- It suffices to store gaps: $283159 - 283154 = 5$, $283202 - 283159 = 43$
- Example postings list using gaps: COMPUTER: 283154, 5, 43, . . .
- Gaps for frequent terms are small.
- Thus: We can encode small gaps with fewer than 20 bits.

Gap encoding

	encoding	postings list									
THE	docIDs	...		283042		283043		283044		283045	...
	gaps				1		1		1		...
COMPUTER	docIDs	...		283047		283154		283159		283202	...
	gaps				107		5		43		...
ARACHNOCENTRIC	docIDs	252000		500100							
	gaps	252000	248100								

Variable length encoding

- Aim:
  - For ARACHNOCENTRIC and other rare terms, we will use about 20 bits per gap (= posting).
  - For THE and other very frequent terms, we will use only a few bits per gap (= posting).
- In order to implement this, we need to devise some form of variable length encoding.
- Variable length encoding uses few bits for small gaps and many bits for large gaps.

Variable byte (VB) code

- Used by many commercial/research systems
- Good low-tech blend of variable-length coding and sensitivity to alignment matches (bit-level codes, see later).
- Dedicate 1 bit (high bit) to be a continuation bit c.
- If the gap G fits within 7 bits, binary-encode it in the 7 available bits and set $c = 1$.
- Else: encode lower-order 7 bits and then use one or more additional bytes to encode the higher order bits using the same algorithm.
- At the end set the continuation bit of the last byte to 1 ($c = 1$) and of the other bytes to 0 ($c = 0$).

VB code examples

docIDs	824	829	215406
gaps		5	214577
VB code	00000110 10111000	10000101	00001101 00001100 10110001

VB code encoding algorithm

```
VBEncodeNumber($n$)
bytes $\leftarrow \langle\rangle$
while true
do Prepend(bytes, $n \bmod 128$)
    if $n < 128$
      then Break
    $n \leftarrow n \operatorname{div} 128$
bytes[Length(bytes)] $+= 128$
return bytes
```

```
VBEncode(numbers)
bytestream $\leftarrow \langle\rangle$
for each $n \in$ numbers
do bytes $\leftarrow$ VBEncodeNumber($n$)
    bytestream $\leftarrow$ Extend(bytestream, bytes)
return bytestream
```

VB code decoding algorithm

```
VBDecode($bytestream$)
numbers $\leftarrow \langle\rangle$
$n \leftarrow 0$
for $i \leftarrow 1$ to Length($bytestream$)
do if $bytestream[i] < 128$
     then $n \leftarrow 128 \times n + bytestream[i]$
     else $n \leftarrow 128 \times n + (bytestream[i] - 128)$
          Append($numbers, n$)
          $n \leftarrow 0$
return $numbers$
```

Other variable codes

- Instead of bytes, we can also use a different "unit of alignment": 32 bits (words), 16 bits, 4 bits (nibbles) etc
- Variable byte alignment wastes space if you have many small gaps – nibbles do better on those.
- There is work on word-aligned codes that efficiently "pack" a variable number of gaps into one word – see resources at the end

Codes for gap encoding

- You can get even more compression with another type of variable length encoding: bitlevel code.
- Gamma code is the best known of these.
- First, we need unary code to be able to introduce gamma code.
- Unary code
  - Represent *n* as *n* 1s with a final 0.
  - Unary code for 3 is 1110
  - Unary code for 1 is 10, for 0 is 0, for 30 is 1111111111111111111111111111110

Gamma code

- Represent a gap G as a pair of length and offset.
- Offset is the gap in binary, with the leading bit chopped off.
- For example 13 $\rightarrow$ 1101 $\rightarrow$ 101 $=$ offset
- Length is the length of offset.
- For 13 (offset 101), this is 3.
- Encode length in unary code: 1110.
- Gamma code of 13 is the concatenation of length and offset: 1110101.

Another Gamma code (γ) examples

number	unary code	length	offset	γ code
0	0			
1	10	0		0
2	110	10	0	10,0
3	1110	10	1	10,1
4	11110	110	00	110,00
9	1111111110	1110	001	1110,001
13		1110	101	1110,101
24		11110	1000	11110,1000
511		111111110	11111111	111111110,11111111
1025		11111111110	0000000001	11111111110,0000000001

The universal coding of the integers: Elias codes

- unary code $\alpha(N) = \underbrace{11\dots1}_{N}0$. $\alpha(4) = 11110$
- binary code $\beta(1) = 1, \beta(2N + j) = \beta(N)j$, $j = 0, 1$. $\beta(4) = 100$
- β is not uniquely decodable (it is not a prefix code).
- ternary $\tau(N) = \beta(N)\#$. $\tau(4) = 100\#$
- $\beta'(1) = \epsilon$, $\beta'(2N) = \beta'(N)0$, $\beta'(2N + 1) = \beta'(N)1$, $\tau'(N) = \beta'(N)\#$. $\beta'(4) = 00$.
- $\gamma(N) = \alpha|\beta'(N)|\beta'(N)$. $\gamma(4) = 11000$
- alternatively, γ': every bit $\beta'(N)$ is inserted between a pair from $\alpha(|\beta'(N)|)$. the same length as γ (bit permutation $\gamma(N)$), but less readable
- example: $\gamma'(4) = 1\bar{0}1\bar{0}0$
- $C_\gamma = \{\gamma(N) : N > 0\} = (1\{0, 1\})^*0$ is regular and therefore it is decodable by finite automaton.

Elias codes: gamma, delta, omega: formal definitions II

- ☞ $\delta(N) = \gamma(|\beta(N)|)\beta'(N)$
- ☞ example: $\delta(4) = \gamma(3)00 = 01100$
- ☞ decoder δ: $\delta(?) = 1001?$
- ☞ ω:

```
K := 0;
while $\lfloor \log_2(N) \rfloor > 0$ do
  begin $K := \beta(N)K$;
        $N := \lfloor \log_2(N) \rfloor$
  end.
```

Exercise

- Compute the variable byte code of 130
- Compute the gamma code of 130
- Compute $\delta(42)$

Length of gamma code

- The length of *offset* is $\lfloor \log_2 G \rfloor$ bits.
- The length of *length* is $\lfloor \log_2 G \rfloor + 1$ bits,
- So the length of the entire code is $2 \times \lfloor \log_2 G \rfloor + 1$ bits.
- γ codes are always of odd length.
- Gamma codes are within a factor of 2 of the optimal encoding length $\log_2 G$.
    - (assuming the frequency of a gap G is proportional to $\log_2 G$ – only approximately true)

Gamma code: Properties

- Gamma code is prefix-free: a valid code word is not a prefix of any other valid code.
- Encoding is optimal within a factor of 3 (and within a factor of 2 making additional assumptions).
- This result is independent of the distribution of gaps!
- We can use gamma codes for any distribution. Gamma code is universal.
- Gamma code is parameter-free.

Gamma codes: Alignment

- Machines have word boundaries – 8, 16, 32 bits
- Compressing and manipulating at granularity of bits can be slow.
- Variable byte encoding is aligned and thus potentially more efficient.
- Another word aligned scheme: Anh and Moffat 2005
- Regardless of efficiency, variable byte is conceptually simpler at little additional space cost.

Compression of Reuters

data structure	size in MB
dictionary, fixed-width	11.2
dictionary, term pointers into string	7.6
$\sim$, with blocking, $k = 4$	7.1
$\sim$, with blocking & front coding	5.9
collection (text, xml markup etc)	3600.0
collection (text)	960.0
T/D incidence matrix	40,000.0
postings, uncompressed (32-bit words)	400.0
postings, uncompressed (20 bits)	250.0
postings, variable byte encoded	116.0
postings, γ encoded	101.0

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 does not occur.

Example: CALPURNIA doesn't occur in *The tempest*.

Compression of Reuters

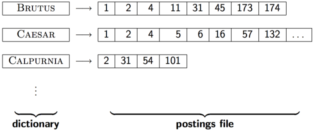
data structure	size in MB
dictionary, fixed-width	11.2
dictionary, term pointers into string	7.6
$\sim$, with blocking, $k = 4$	7.1
$\sim$, with blocking & front coding	5.9
collection (text, xml markup etc)	3600.0
collection (text)	960.0
T/D incidence matrix	40,000.0
postings, uncompressed (32-bit words)	400.0
postings, uncompressed (20 bits)	250.0
postings, variable byte encoded	116.0
postings, γ encoded	101.0

Summary

- We can now create an index for highly efficient Boolean retrieval that is very space efficient.
- Only 4% of the total size of the collection.
- Only 10–15% of the total size of the text in the collection.
- However, we've ignored positional and frequency information.
- For this reason, space savings are less in reality.

Take-away today

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

⋮

- Motivation for compression in information retrieval systems
- How can we compress the dictionary component of the inverted index?
- How can we compress the postings component of the inverted index?
- Term statistics: how are terms distributed in document collections?

Resources

http://ske.fi.muni.cz

- Chapter 5 of IIR
- Resources at https://www.fi.muni.cz/~sojka/PV211/ and http://cislmu.org, materials in MU IS and FI MU library
  - Original publication on word-aligned binary codes by Anh and Moffat (2005); also: Anh and Moffat (2006a).
  - Original publication on variable byte codes by Scholer, Williams, Yiannis and Zobel (2002).
  - More details on compression (including compression of positions and frequencies) in Zobel and Moffat (2006).

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 6: Scoring, term weighting, the vector space model
Handout version

Petr Sojka, Hinrich Schütze et al.

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

2020-03-11

Overview

1. Why ranked retrieval?
2. Term frequency
3. tf-idf weighting
4. The vector space model

Take-away today

- Ranking search results: why it is important (as opposed to just presenting a set of unordered Boolean results)
- Term frequency: This is a key ingredient for ranking.
- Tf-idf ranking: best known traditional ranking scheme
- Vector space model: One of the most important formal models for information retrieval (along with Boolean and probabilistic models)

Ranked retrieval

- Thus far, our queries have been Boolean.
    - Documents either match or do not.
- Good for expert users with precise understanding of their needs and of the collection.
- Also good for applications: Applications can easily consume 1000s of results.
- Not good for the majority of users
- Most users are not capable of writing Boolean queries . . .
    - . . . or they are, but they think it's too much work.
- Most users don't want to wade through 1000s of results.
- This is particularly true of web search.

Problem with Boolean search: Feast or famine

- Boolean queries often result in either too few (=0) or too many (1000s) results.
- Query 1 (boolean conjunction): [standard user dlink 650]
  - → 200,000 hits – feast
- Query 2 (boolean conjunction): [standard user dlink 650 no card found]
  - → 0 hits – famine
- In Boolean retrieval, it takes a lot of skill to come up with a query that produces a manageable number of hits.

Feast or famine: No problem in ranked retrieval

- With ranking, large result sets are not an issue.
- Just show the top 10 results
- Doesn't overwhelm the user
- Premise: the ranking algorithm works: More relevant results are ranked higher than less relevant results.

Scoring as the basis of ranked retrieval

- We wish to rank documents that are more relevant higher than documents that are less relevant.
- How can we accomplish such a ranking of the documents in the collection with respect to a query?
- Assign a score to each query-document pair, say in $[0, 1]$.
- This score measures how well document and query "match".

Query-document matching scores

- How do we compute the score of a query-document pair?
- Let's start with a one-term query.
- If the query term does not occur in the document: score should be 0.
- The more frequent the query term in the document, the higher the score.
- We will look at a number of alternatives for doing this.

Take 1: Jaccard coefficient

- A commonly used measure of overlap of two sets
- Let A and B be two sets
- Jaccard coefficient:

$$\textsc{jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

  $(A \neq \emptyset$ or $B \neq \emptyset)$
- $\textsc{jaccard}(A, A) = 1$
- $\textsc{jaccard}(A, B) = 0$ if $A \cap B = 0$
- A and B don't have to be the same size.
- Always assigns a number between 0 and 1.

Jaccard coefficient: Example

- What is the query-document match score that the Jaccard coefficient computes for:
  - Query: “ides of March”
  - Document “Caesar died in March”
  - $\textsc{jaccard}(q, d) = 1/6$

What's wrong with Jaccard?

- It doesn't consider term frequency (how many occurrences a term has).
- Rare terms are more informative than frequent terms. Jaccard does not consider this information.
- We need a more sophisticated way of normalizing for the length of a document.
- Later in this lecture, we'll use $|A \cap B|/\sqrt{|A \cup B|}$ (cosine) . . .
- . . . instead of $|A \cap B|/|A \cup B|$ (Jaccard) for length normalization.

Binary 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	
...							

Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

Count matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	. . .
ANTHONY	157	73	0	0	0	1	
BRUTUS	4	157	0	2	0	0	
CAESAR	232	227	0	2	1	0	
CALPURNIA	0	10	0	0	0	0	
CLEOPATRA	57	0	0	0	0	0	
MERCY	2	0	3	8	5	8	
WORSER	2	0	1	1	1	5	
. . .							

Each document is now represented as a count vector $\in \mathbb{N}^{|V|}$.

Bag of words model

- We do not consider the order of words in a document.
- *John is quicker than Mary* and *Mary is quicker than John* are represented the same way.
- This is called a bag of words model.
- In a sense, this is a step back: The positional index was able to distinguish these two documents.
- We will look at "recovering" positional information later in this course.
- For now: bag of words model

Term frequency tf

- The term frequency $\text{tf}_{t,d}$ of term t in document d is defined as the number of times that t occurs in d.
- We want to use tf when computing query-document match scores.
- But how?
- Raw term frequency is not what we want because:
- A document with $\text{tf} = 10$ occurrences of the term is more relevant than a document with $\text{tf} = 1$ occurrence of the term.
- But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.

Instead of raw frequency: Log frequency weighting

- The log frequency weight of term t in d is defined as follows

$$\mathrm{w}_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

- $\mathrm{tf}_{t,d} \rightarrow \mathrm{w}_{t,d}$:
  $0 \rightarrow 0$, $1 \rightarrow 1$, $2 \rightarrow 1.3$, $10 \rightarrow 2$, $1000 \rightarrow 4$, etc.
- Score for a document-query pair: sum over terms t in both q and d:
  tf-matching-score$(q, d) = \sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$
- The score is 0 if none of the query terms is present in the document.

Exercise

- Compute the Jaccard matching score and the tf matching score for the following query-document pairs.
- q: [information on cars] d: “all you’ve ever wanted to know about cars”
- q: [information on cars] d: “information on trucks, information on planes, information on trains”
- q: [red cars and red trucks] d: “cops stop red cars more often”

Frequency in document vs. frequency in collection

- In addition, to term frequency (the frequency of the term in the document) . . .
- . . . we also want to use the frequency of the term in the collection for weighting and ranking.

Desired weight for rare terms

- Rare terms are more informative than frequent terms.
- Consider a term in the query that is rare in the collection (e.g., ARACHNOCENTRIC).
- A document containing this term is very likely to be relevant.
- → We want high weights for rare terms like ARACHNOCENTRIC.

Desired weight for frequent terms

- Frequent terms are less informative than rare terms.
- Consider a term in the query that is frequent in the collection (e.g., GOOD, INCREASE, LINE).
- A document containing this term is more likely to be relevant than a document that doesn't . . .
- . . . but words like GOOD, INCREASE and LINE are not sure indicators of relevance.
- → For frequent terms like GOOD, INCREASE, and LINE, we want positive weights . . .
- . . . but lower weights than for rare terms.

Document frequency

- We want high weights for rare terms like ARACHNOCENTRIC.
- We want low (positive) weights for frequent words like GOOD, INCREASE, and LINE.
- We will use document frequency to factor this into computing the matching score.
- The document frequency is the number of documents in the collection that the term occurs in.

idf weight

- df_t is the document frequency, the number of documents that t occurs in.
- df_t is an inverse measure of the informativeness of term t.
- We define the idf weight of term t as follows:

$$\mathrm{idf}_t = \log_{10} \frac{N}{\mathrm{df}_t}$$

  (N is the number of documents in the collection.)
- idf_t is a measure of the informativeness of the term.
- $[\log N/\mathrm{df}_t]$ instead of $[N/\mathrm{df}_t]$ to “dampen” the effect of idf
- Note that we use the log transformation for both term frequency and document frequency.

Examples for idf

Compute idf_t using the formula: $\text{idf}_t = \log_{10} \frac{1{,}000{,}000}{\text{df}_t}$

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Effect of idf on ranking

- idf affects the ranking of documents for queries with at least two terms.
- For example, in the query "arachnocentric line", idf weighting increases the relative weight of ARACHNOCENTRIC and decreases the relative weight of LINE.
- idf has little effect on ranking for one-term queries.

Collection frequency vs. Document frequency

word	collection frequency	document frequency
INSURANCE	10440	3997
TRY	10422	8760

- Collection frequency of t: number of tokens of t in the collection
- Document frequency of t: number of documents t occurs in
- Why these numbers?
- Which word is a better search term (and should get a higher weight)?
- This example suggests that df (and idf) is better for weighting than cf (and "icf").

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.
- $$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$
- tf-weight
- idf-weight
- Best known weighting scheme in information retrieval
- Note: the "-" in tf-idf is a hyphen, not a minus sign!
- Alternative names: tf.idf, tf x idf

Summary: tf-idf

- Assign a tf-idf weight for each term t in each document d: $w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$
- The tf-idf weight . . .
  - . . . increases with the number of occurrences within a document. (term frequency)
  - . . . increases with the rarity of the term in the collection. (inverse document frequency)

Exercise: Term, collection and document frequency

Quantity	Symbol	Definition
term frequency	$\text{tf}_{t,d}$	number of occurrences of t in d
document frequency	df_t	number of documents in the collection that t occurs in
collection frequency	cf_t	total number of occurrences of t in the collection

- Relationship between df and cf?
- Relationship between tf and cf?
- Relationship between tf and df?

Binary 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	
...							

Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

Count matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	157	73	0	0	0	1	
BRUTUS	4	157	0	2	0	0	
CAESAR	232	227	0	2	1	0	
CALPURNIA	0	10	0	0	0	0	
CLEOPATRA	57	0	0	0	0	0	
MERCY	2	0	3	8	5	8	
WORSER	2	0	1	1	1	5	
...							

Each document is now represented as a count vector $\in \mathbb{N}^{|V|}$.

Binary → count → weight matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	5.25	3.18	0.0	0.0	0.0	0.35	
BRUTUS	1.21	6.10	0.0	1.0	0.0	0.0	
CAESAR	8.59	2.54	0.0	1.51	0.25	0.0	
CALPURNIA	0.0	1.54	0.0	0.0	0.0	0.0	
CLEOPATRA	2.85	0.0	0.0	0.0	0.0	0.0	
MERCY	1.51	0.0	1.90	0.12	5.25	0.88	
WORSER	1.37	0.0	0.11	4.15	0.25	1.95	
...							

Each document is now represented as a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$.

Documents as vectors

- Each document is now represented as a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$.
- So we have a $|V|$-dimensional real-valued vector space.
- Terms are axes of the space.
- Documents are points or vectors in this space.
- Very high-dimensional: tens of millions of dimensions when you apply this to web search engines
- Each vector is very sparse - most entries are zero.

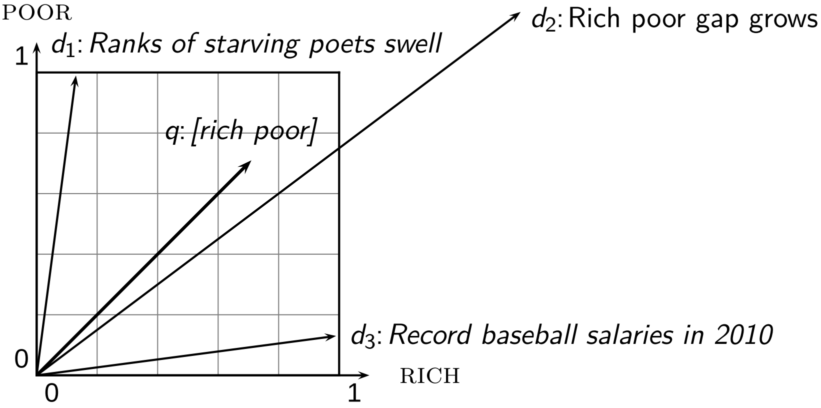
Queries as vectors

- Key idea 1: do the same for queries: represent them as vectors in the high-dimensional space
- Key idea 2: Rank documents according to their proximity to the query
- proximity = similarity
- proximity ≈ negative distance
- Recall: We're doing this because we want to get away from the you're-either-in-or-out, feast-or-famine Boolean model.
- Instead: rank relevant documents higher than nonrelevant documents.

How do we formalize vector space similarity?

- First cut: (negative) distance between two points
- ($=$ distance between the end points of the two vectors)
- Euclidean distance?
- Euclidean distance is a bad idea . . .
- . . . because Euclidean distance is large for vectors of different lengths.

Why distance is a bad idea


POOR
1
0
0
1
RICH
d_1: *Ranks of starving poets swell*
d_2: Rich poor gap grows
q: *[rich poor]*
d_3: *Record baseball salaries in 2010*

The Euclidean distance of $\vec{q}$ and $\vec{d}_2$ is large although the distribution of terms in the query q and the distribution of terms in the document d_2 are very similar.

Questions about basic vector space setup?

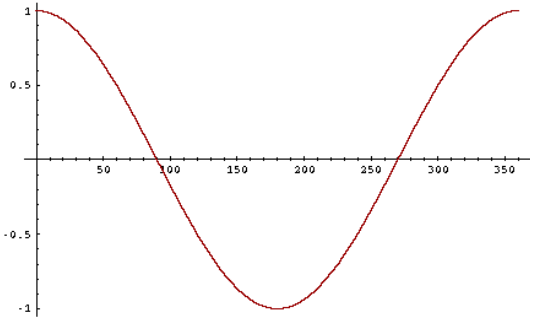
Use angle instead of distance

- Rank documents according to angle with query
- Thought experiment: take a document d and append it to itself. Call this document d'. d' is twice as long as d.
- "Semantically" d and d' have the same content.
- The angle between the two documents is 0, corresponding to maximal similarity . . .
- . . . even though the Euclidean distance between the two documents can be quite large.

From angles to cosines

- The following two notions are equivalent.
  - Rank documents according to the angle between query and document in decreasing order
  - Rank documents according to cosine(query,document) in increasing order
- Cosine is a monotonically decreasing function of the angle for the interval $[0^\circ, 180^\circ]$

Cosine



Length normalization

- How do we compute the cosine?
- A vector can be (length-) normalized by dividing each of its components by its length – here we use the L_2 norm: $||x||_2 = \sqrt{\sum_i x_i^2}$
- This maps vectors onto the unit sphere ...
- ...since after normalization: $||x||_2 = \sqrt{\sum_i x_i^2} = 1.0$
- As a result, longer documents and shorter documents have weights of the same order of magnitude.
- Effect on the two documents d and d' (d appended to itself) from earlier slide: they have identical vectors after length-normalization.

Cosine similarity between query and document

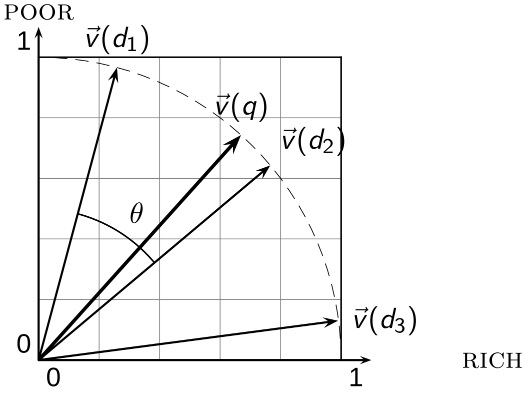
$$\cos(\vec{q},\vec{d}) = \textsc{sim}(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

- q_i is the tf-idf weight of term i in the query.
- d_i is the tf-idf weight of term i in the document.
- $|\vec{q}|$ and $|\vec{d}|$ are the lengths of $\vec{q}$ and $\vec{d}$.
- This is the cosine similarity of $\vec{q}$ and $\vec{d}$ . . . . . . or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

Cosine for normalized vectors

- For normalized vectors, the cosine is equivalent to the dot product or scalar product.
- $\cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} = \sum_i q_i \cdot d_i$
  - (if $\vec{q}$ and $\vec{d}$ are length-normalized).

Cosine similarity illustrated



Cosine: Example

How similar are these novels?

SaS: Sense and Sensibility

PaP: Pride and Prejudice

WH: Wuthering Heights

term frequencies (counts)

term	SaS	PaP	WH
AFFECTION	115	58	20
JEALOUS	10	7	11
GOSSIP	2	0	6
WUTHERING	0	0	38

Cosine: Example

term frequencies (counts)

term	SaS	PaP	WH
AFFECTION	115	58	20
JEALOUS	10	7	11
GOSSIP	2	0	6
WUTHERING	0	0	38

log frequency weighting

term	SaS	PaP	WH
AFFECTION	3.06	2.76	2.30
JEALOUS	2.0	1.85	2.04
GOSSIP	1.30	0	1.78
WUTHERING	0	0	2.58

(To simplify this example, we don't do idf weighting.)

Cosine: Example

log frequency weighting

term	SaS	PaP	WH
AFFECTION	3.06	2.76	2.30
JEALOUS	2.0	1.85	2.04
GOSSIP	1.30	0	1.78
WUTHERING	0	0	2.58

log frequency weighting
& cosine normalization

term	SaS	PaP	WH
AFFECTION	0.789	0.832	0.524
JEALOUS	0.515	0.555	0.465
GOSSIP	0.335	0.0	0.405
WUTHERING	0.0	0.0	0.588

- $\cos(\text{SaS,PaP}) \approx$ $0.789 * 0.832 + 0.515 * 0.555 + 0.335 * 0.0 + 0.0 * 0.0 \approx 0.94.$
- $\cos(\text{SaS,WH}) \approx 0.79$
- $\cos(\text{PaP,WH}) \approx 0.69$
- Why do we have $\cos(\text{SaS,PaP}) > \cos(\text{SAS,WH})$?

Computing the cosine score

```
CosineScore($q$)
$float\ Scores[N] = 0$
$float\ Length[N]$
for each query term $t$
do calculate $\mathrm{w}_{t,q}$ and fetch postings list for $t$
    for each pair($d$, $\mathrm{tf}_{t,d}$) in postings list
    do $Scores[d] += \mathrm{w}_{t,d} \times \mathrm{w}_{t,q}$
Read the array $Length$
for each $d$
do $Scores[d] = Scores[d]/Length[d]$
return Top $K$ components of $Scores[]$
```

Components of tf-idf weighting

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1+\log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5+\frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0,\log\frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases}1 & \text{if } \mathrm{tf}_{t,d}>0\\ 0 & \text{otherwise}\end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha<1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

Best known combination of weighting options

Default: no weighting

tf-idf example

- We often use different weightings for queries and documents.
- Notation: ddd.qqq
- Example: lnc.ltn
- document: logarithmic tf, no df weighting, cosine normalization
- query: logarithmic tf, idf, no normalization
- Isn't it bad to not idf-weight the document?
- Example query: "best car insurance"
- Example document: "car insurance auto insurance"

tf-idf example: lnc.ltn

Query: “best car insurance”. Document: “car insurance auto insurance”.

word	query					document				product
	tf-raw	tf-wght	df	idf	weight	tf-raw	tf-wght	weight	n’lized	
auto	0	0	5000	2.3	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0	0	0	0	0
car	1	1	10000	2.0	2.0	1	1	1	0.52	1.04
insurance	1	1	1000	3.0	3.0	2	1.3	1.3	0.68	2.04

Key to columns: tf-raw: raw (unweighted) term frequency, tf-wght: logarithmically weighted term frequency, df: document frequency, idf: inverse document frequency, weight: the final weight of the term in the query or document, n’lized: document weights after cosine normalization, product: the product of final query weight and final document weight

$\sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$
$1/1.92 \approx 0.52$
$1.3/1.92 \approx 0.68$

Final similarity score between query and document: $\sum_i w_{qi} \cdot w_{di} = 0 + 0 + 1.04 + 2.04 = 3.08$

Questions?

Summary: Ranked retrieval in the vector space model

- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity between the query vector and each document vector
- Rank documents with respect to the query
- Return the top K (e.g., $K = 10$) to the user

Take-away today

- Ranking search results: why it is important (as opposed to just presenting a set of unordered Boolean results)
- Term frequency: This is a key ingredient for ranking.
- Tf-idf ranking: best known traditional ranking scheme
- Vector space model: One of the most important formal models for information retrieval (along with Boolean and probabilistic models)

Resources

- Chapter 6 of IIR
- Resources at `https://www.fi.muni.cz/~sojka/PV211/` and `http://cislmu.org`, materials in MU IS and FI MU library
  - Vector space for dummies
  - Exploring the similarity space (Moffat and Zobel, 2005)
  - Okapi BM25 (a state-of-the-art weighting method, 11.4.3 of IIR)

Introduction to

Information Retrieval

CS276: Information Retrieval and Web Search
Christopher Manning and Pandu Nayak

Lecture 14: Distributed Word Representations for Information Retrieval

How can we more robustly match a user's search intent?

We want to **understand** a query, not just do String equals()

- If user searches for [Dell notebook battery size], we would like to match documents discussing "Dell laptop battery capacity"
- If user searches for [Seattle motel], we would like to match documents containing "Seattle hotel"

A pure keyword-matching IR system does nothing to help....

Simple facilities that we have already discussed do a bit to help

- Spelling correction
- Stemming / case folding

But we'd like to better **understand** when query/document match

How can we more robustly match a user’s search intent?

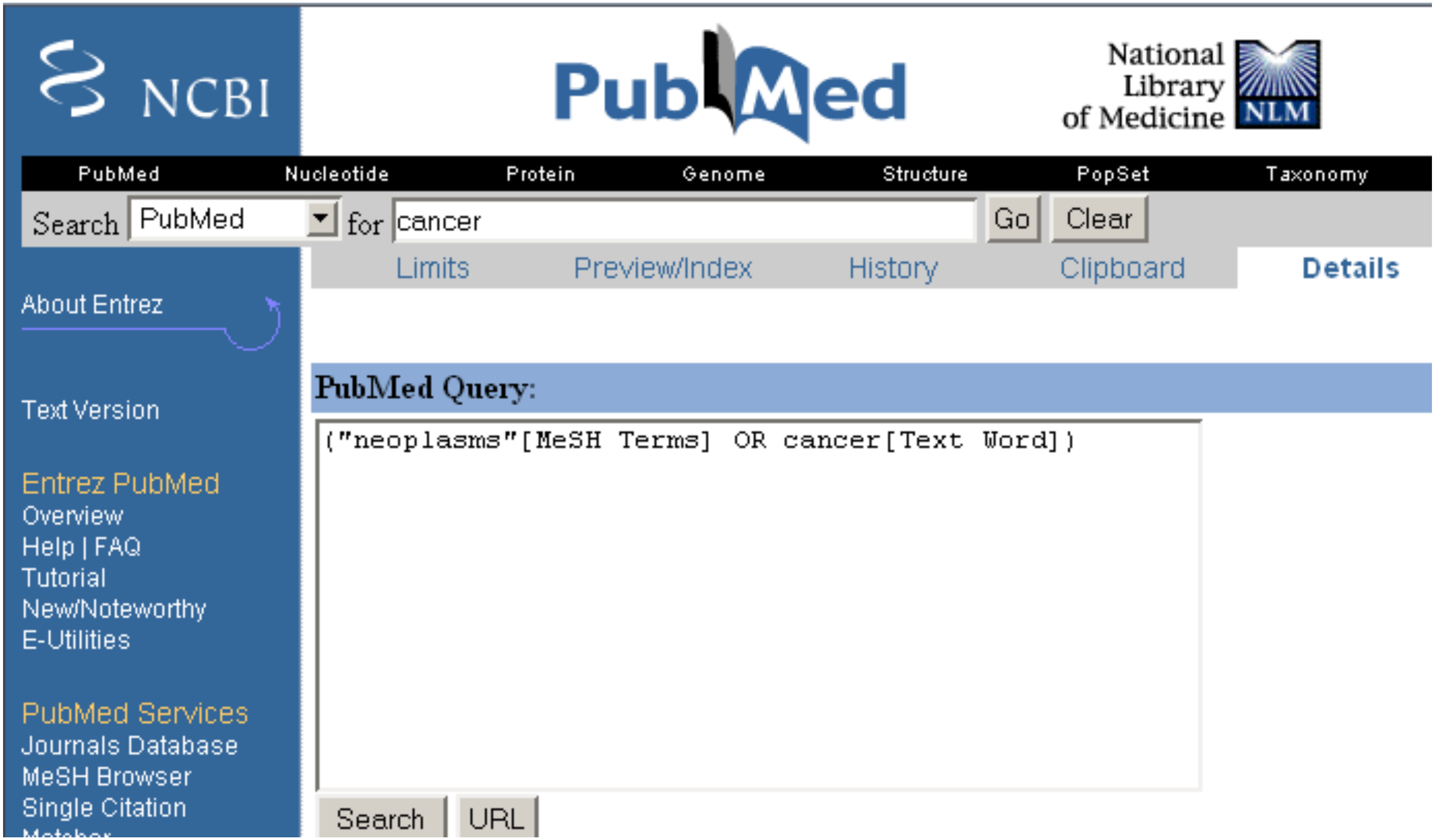
Query expansion:

- **Relevance feedback** could allow us to capture this if we get near enough to matching documents with these words
- We can also use information on **word similarities**:
  - A manual **thesaurus** of synonyms for query expansion
  - A **measure of word similarity**
    - Calculated from a big document collection
    - Calculated by query log mining (common on the web)

Document expansion:

- Use of **anchor text** may solve this by providing human authored synonyms, but not for new or less popular web pages, or non-hyperlinked collections

Example of manual thesaurus



PubMed Query:

```
("neoplasms"[MeSH Terms] OR cancer[Text Word])
```

Search log query expansion

- Context-free query expansion ends up problematic
  - [wet ground] ≈ [wet earth]
  - So expand [ground] ⇒ [ground earth]
  - But [ground coffee] ≠ [earth coffee]
- You can learn query context-specific rewritings from search logs by attempting to identify the same user making a second attempt at the same user need
  - [Hinton word vector]
  - [Hinton word embedding]
- In this context, [vector] ≈ [embedding]
  - But not when talking about a *disease vector* or C++!

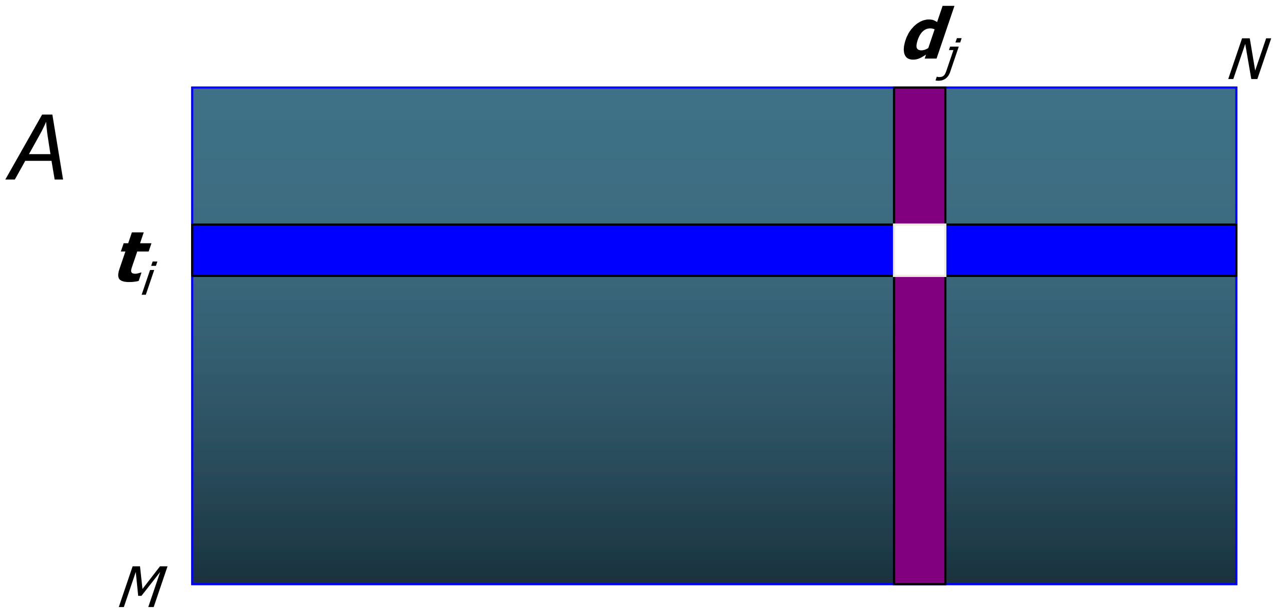
Automatic Thesaurus Generation

- Attempt to generate a thesaurus automatically by analyzing a collection of documents
- Fundamental notion: similarity between two words
- Definition 1: Two words are similar if they co-occur with similar words.
- Definition 2: Two words are similar if they occur in a given grammatical relation with the same words.
- You can harvest, peel, eat, prepare, etc. apples and pears, so apples and pears must be similar.
- Co-occurrence based is more robust, grammatical relations are more accurate.

Why?

Simple Co-occurrence Thesaurus

- Simplest way to compute one is based on term-term similarities in $C = AA^T$ where A is term-document matrix.
- $w_{i,j}$ = (normalized) weight for (t_i ,$\boldsymbol{d}_j$)



What does C contain if A is a term-doc incidence (0/1) matrix?

- For each t_i, pick terms with high values in C

Automatic thesaurus generation example … sort of works

Word	Nearest neighbors
absolutely	absurd, whatsoever, totally, exactly, nothing
bottomed	dip, copper, drops, topped, slide, trimmed
captivating	shimmer, stunningly, superbly, plucky, witty
doghouse	dog, porch, crawling, beside, downstairs
makeup	repellent, lotion, glossy, sunscreen, skin, gel
mediating	reconciliation, negotiate, cease, conciliation
keeping	hoping, bring, wiping, could, some, would
lithographs	drawings, Picasso, Dali, sculptures, Gauguin
pathogens	toxins, bacteria, organisms, bacterial, parasites
senses	grasp, psyche, truly, clumsy, naïve, innate

Too little data (10s of millions of words) treated by **too sparse method**.
100,000 words = 10^{10} entries in *C*.

How can we represent term relations?

- With the standard symbolic encoding of terms, each term is a dimension
- Different terms have no inherent similarity
- motel $[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$
  hotel $[0\ 0\ 0\ 0\ 0\ 0\ 0\ 3\ 0\ 0\ 0\ 0\ 0\ 0\ 0] = 0$
- If query on *hotel* and document has *motel*, then our query and document vectors are **orthogonal**

Can you directly learn term relations?

- Basic IR is scoring on $q^{\mathrm{T}}d$
- No treatment of synonyms; no machine learning
- Can we learn parameters W to rank via $q^{\mathrm{T}}Wd$?

"search ranking" "information retrieval ranking"

$$\overset{q^T}{\begin{array}{ccccc} (1 & 0 & 0 & 1 & 0) \\ se & in & re & ra & or \end{array}} \overset{W}{\begin{pmatrix} 1 & 0.7 & 0.5 & 0 & 0 \\ 0.3 & 1 & 0.2 & 0 & 0 \\ 0.5 & 0.2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.7 \\ 0 & 0 & 0 & 0.7 & 1 \end{pmatrix}} \overset{d}{\begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}} \begin{array}{l} se \\ in \\ re = 2.2 \\ ra \\ or(dering) \end{array}$$

- Cf. Query translation models: Berger and Lafferty (1999)
- Problem is again sparsity – W is huge > 10^{10}

Is there a better way?

- Idea:
  - Can we learn a dense low-dimensional representation of a word in $\mathbb{R}^d$ such that dot products $u^T v$ express word similarity?
  - We could still if we want to include a “translation” matrix between vocabularies (e.g., cross-language): $u^T W v$
    - But **now *W* is small**!
  - Supervised Semantic Indexing (Bai et al. *Journal of Information Retrieval* 2009) shows successful use of learning *W* for information retrieval

- But we’ll develop direct similarity in this class

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* ↗

Solution: Low dimensional vectors

- The number of topics that people talk about is small (in some sense)
    - Clothes, movies, politics, …
- Idea: store "most" of the important information in a fixed, small number of dimensions: a dense vector
- Usually 25 – 1000 dimensions

- How to reduce the dimensionality?
    - Go from big, sparse co-occurrence count vector to low dimensional "word embedding"

Traditional Way: Latent Semantic Indexing/Analysis

- Use Singular Value Decomposition (SVD) – kind of like Principal Components Analysis (PCA) for an arbitrary rectangular matrix – or just random projection to find a low-dimensional basis or orthogonal vectors
- Theory is that similarity is preserved as much as possible
- You can actually gain in IR (slightly) by doing LSA, as “noise” of term variation gets replaced by semantic “concepts”
- Somewhat popular in the 1990s [Deerwester et al. 1990, etc.]
  - But **results were always somewhat iffy** (… it worked sometimes)
  - Hard to implement efficiently in an IR system (dense vectors!)
- Discussed in *IIR* chapter 18, but not discussed further here
  - Not on the exam (!!!)

“NEURAL EMBEDDINGS”

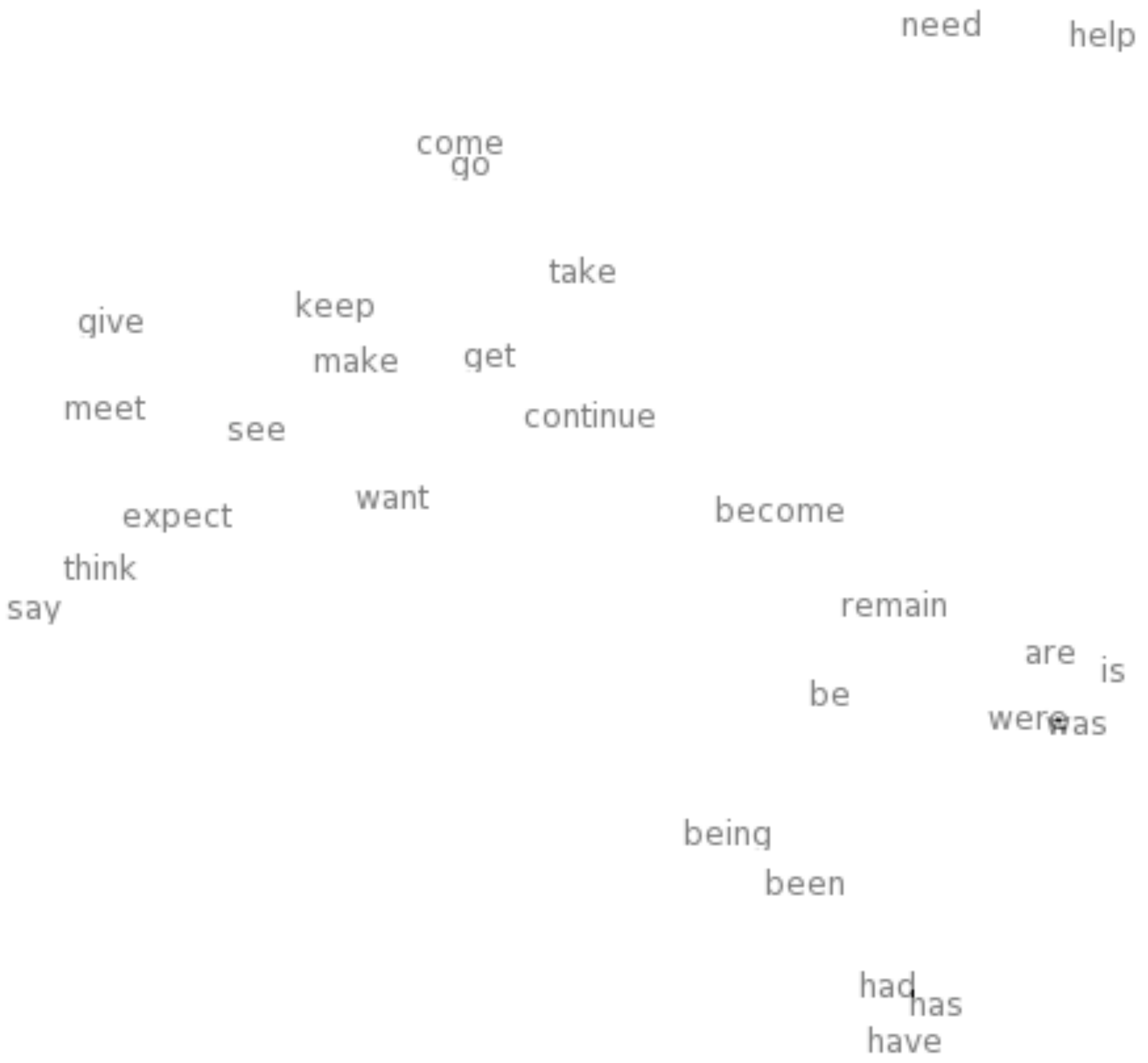
Word meaning is defined in terms of vectors

- We will build a dense vector for each word type, chosen so that it is good at predicting other words appearing in its context

… those other words also being represented by vectors … it all gets a bit recursive

$$\textit{banking} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Neural word embeddings - visualization



Basic idea of learning neural network word embeddings

- We define a model that aims to predict between a center word w_t and context words in terms of word vectors
- p(context|w_t) = …
- which has a loss function, e.g.,
- $J = 1 - p(w_{-t} \mid w_t)$
- We look at many positions *t* in a big language corpus
- We keep adjusting the vector representations of words to minimize this loss

Idea: Directly learn low-dimensional word vectors based on ability to predict

- Old idea: Learning representations by back-propagating errors. (Rumelhart et al., 1986)
- A neural probabilistic language model (Bengio et al., 2003)
- NLP (almost) from Scratch (Collobert & Weston, 2008)

Non-linear and slow

- A recent, even simpler and faster model: word2vec (Mikolov et al. 2013) → intro now
- The GloVe model from Stanford (Pennington, Socher, and Manning 2014) connects back to matrix factorization

Fast bilinear models

- Per-token representations: Deep contextual word representations: ELMo, ULMfit, **BERT**

Current state of the art

Word2vec is a family of algorithms

[Mikolov et al. 2013]

Predict between every word and its context words!

Two algorithms

1. **Skip-grams (SG)**

   Predict context words given target (position independent)

2. Continuous Bag of Words (CBOW)

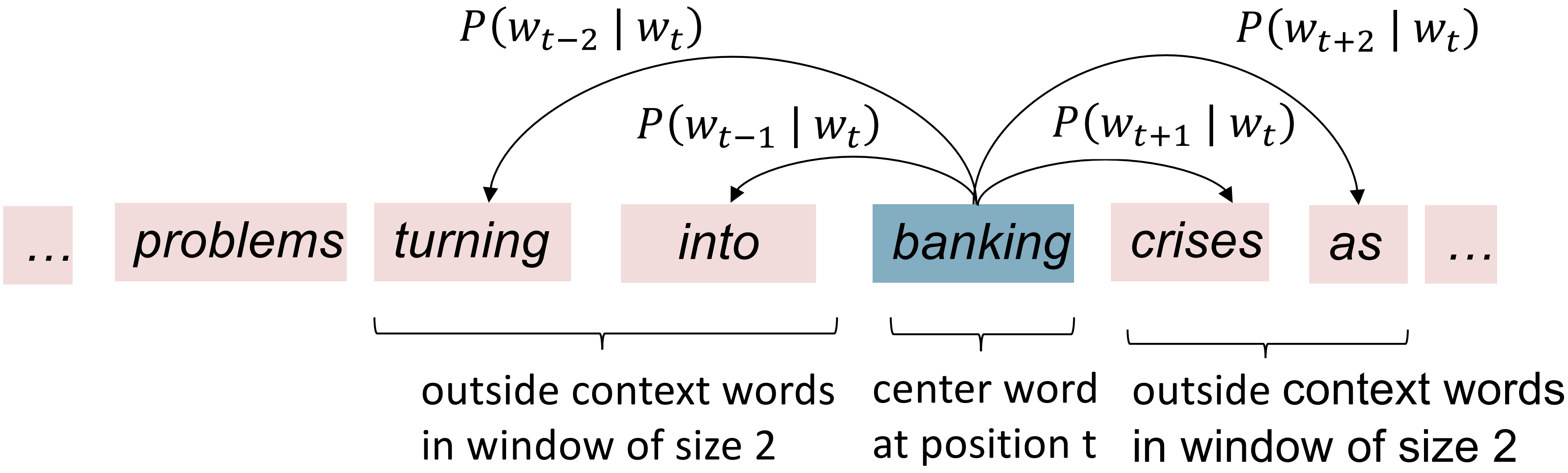
   Predict target word from bag-of-words context

Two (moderately efficient) training methods

1. Hierarchical softmax
2. Negative sampling
3. **Naïve softmax**

Word2Vec Skip-gram Overview

- Example windows and process for computing $P\left(w_{t+j} \mid w_t\right)$



Word2vec: objective function

For each position $t = 1, \ldots, T$, predict context words within a window of fixed size *m*, given center word w_j.

Likelihood =

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P\left(w_{t+j} \mid w_t; \theta\right)$$

θ is all variables to be optimized

sometimes called *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

Minimizing objective function ⟺ Maximizing predictive accuracy

Word2vec: objective function

- We want to minimize the objective function:

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log P\left(w_{t+j} \mid w_t; \theta\right)$$

- Question: How to calculate $P\left(w_{t+j} \mid w_t; \theta\right)$?

- Answer: We will *use two* vectors per word *w*:
  - v_w when *w* is a center word
  - u_w when *w* is a context word

- Then for a center word *c* and a context word *o*:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w\in V}\exp(u_w^T v_c)}$$

Word2vec: prediction function

Exponentiation makes anything positive

Dot product compares similarity of *o* and *c*.

$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$

Larger dot product = larger probability

$$P(o|c) = \frac{\exp(\boxed{u_o^T v_c})}{\boxed{\sum_{w \in V} \exp(u_w^T v_c)}}$$

Normalize over entire vocabulary to give probability distribution

- This is an example of the **softmax function** $\mathbb{R}^n \rightarrow (0,1)^n$

Open region

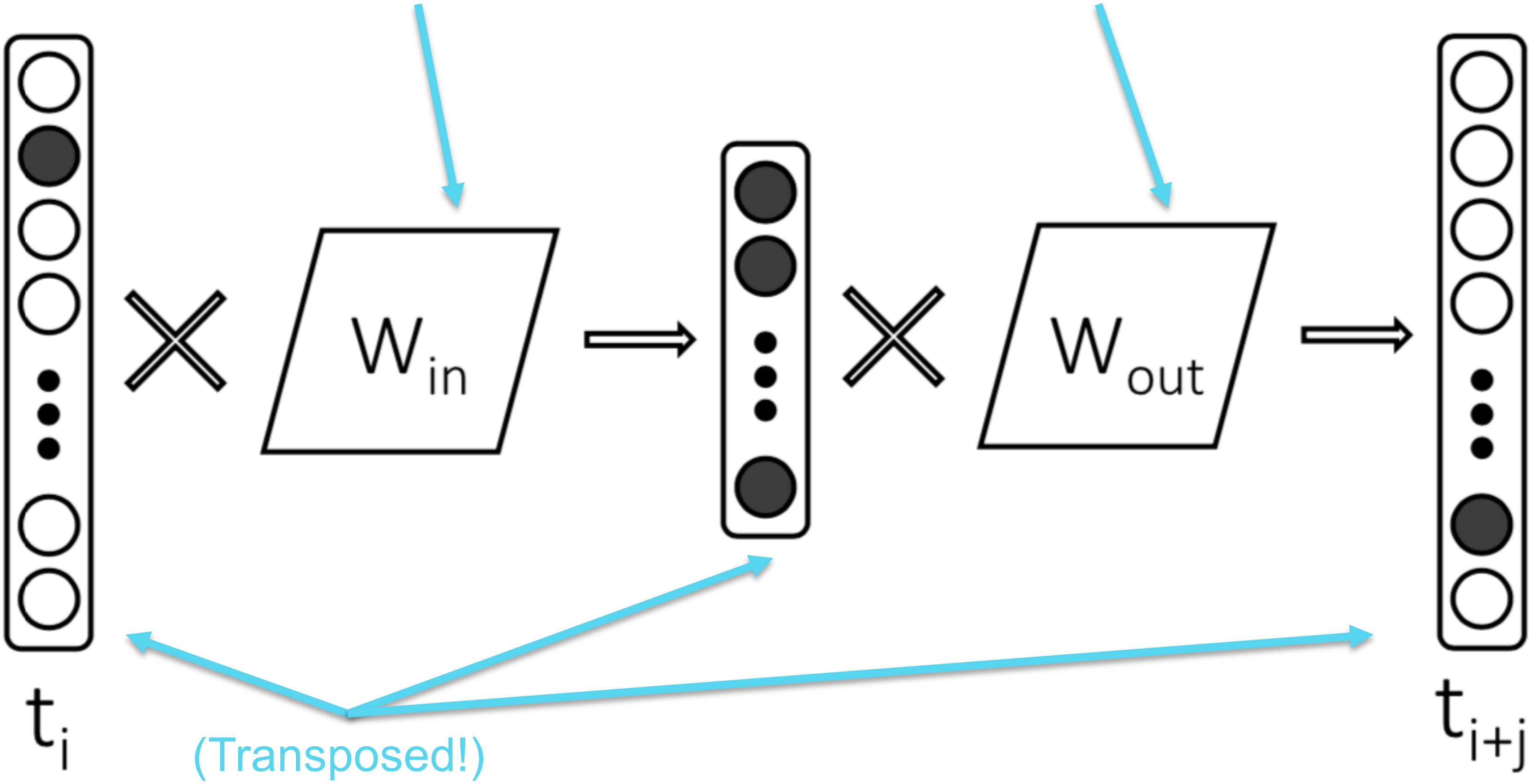
$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

- The softmax function maps arbitrary values x_i to a probability distribution p_i
  - "max" because amplifies probability of largest x_i
  - "soft" because still assigns some probability to smaller x_i
  - Frequently used in neural networks/Deep Learning

Word2vec: 2 matrices of parameters

Center word embeddings as rows

Context word embeddings as columns



(Transposed!)

To learn good word vectors: Compute all vector gradients!

- We often define the set of **all** parameters in a model in terms of one long vector θ
- In our case with *d*-dimensional vectors and *V* many words:

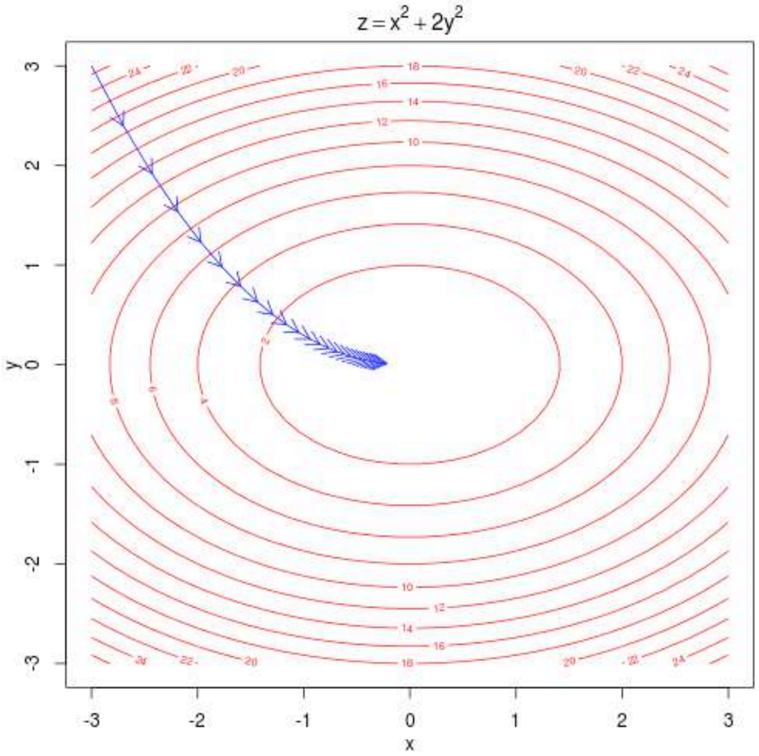
$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

- We then optimize these parameters

Note: Every word has two vectors! Makes it simpler!

Intuition of how to minimize loss for a simple function over two parameters

We start at a random point and walk in the steepest direction, which is given by the derivative of the function

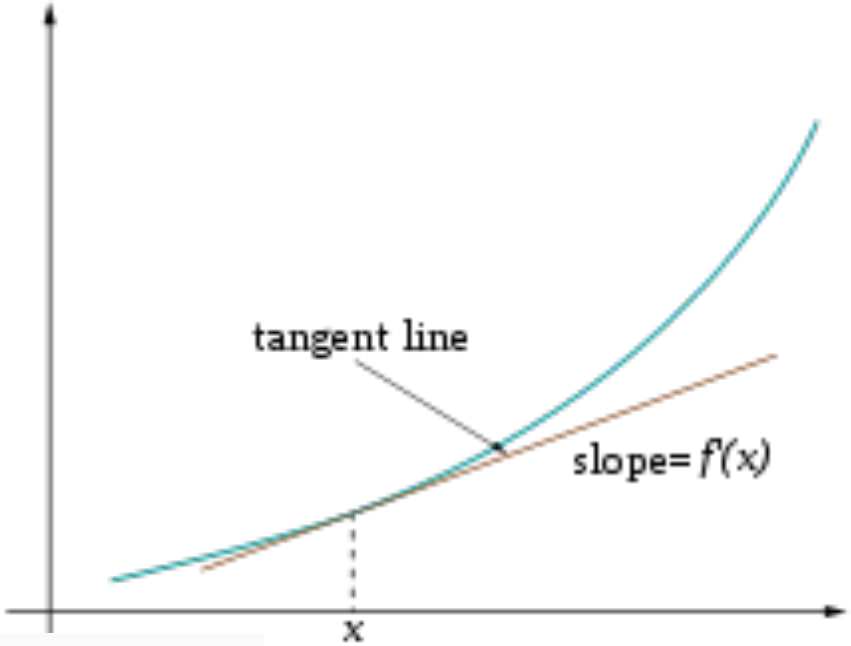


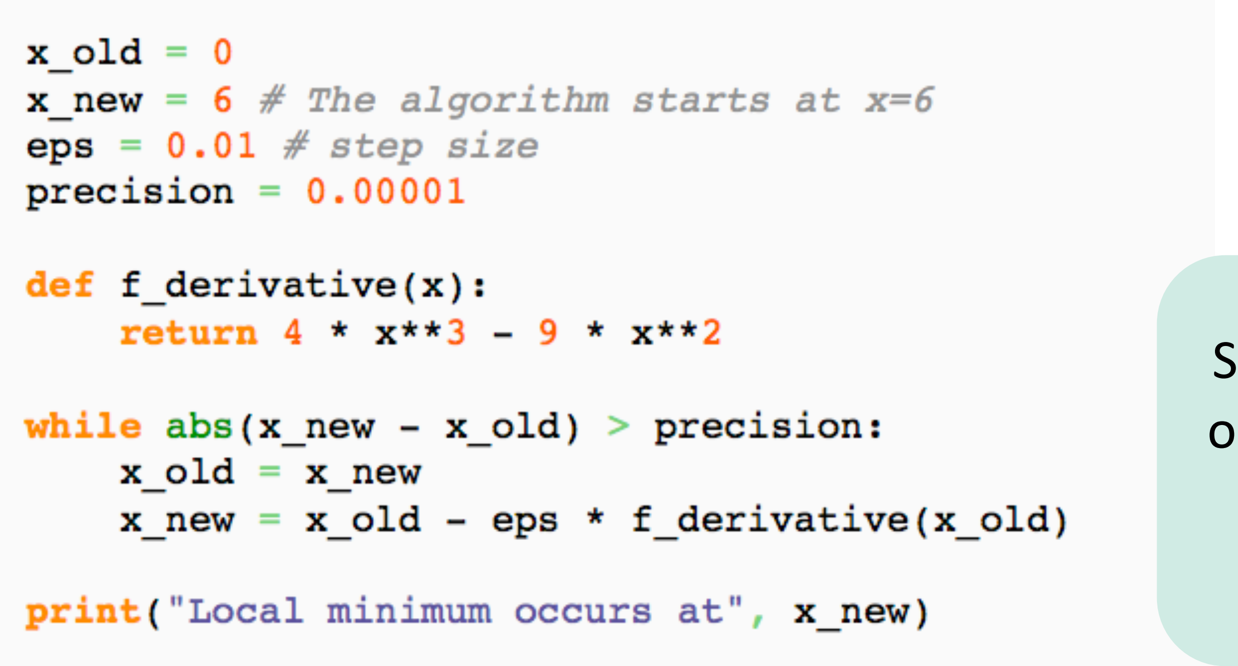
Contour lines show points of equal value of objective function

Descending by using derivatives

We will minimize a cost function by gradient descent

Trivial example: (from Wikipedia)
Find a local minimum of the function
$f(x) = x^4 - 3x^3 + 2$,
with derivative $f'(x) = 4x^3 - 9x^2$



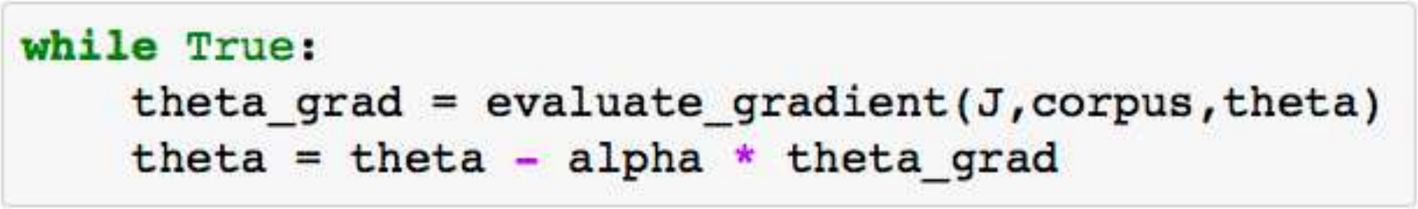


```
x_old = 0
x_new = 6 # The algorithm starts at x=6
eps = 0.01 # step size
precision = 0.00001

def f_derivative(x):
    return 4 * x**3 - 9 * x**2

while abs(x_new - x_old) > precision:
    x_old = x_new
    x_new = x_old - eps * f_derivative(x_old)

print("Local minimum occurs at", x_new)
```

Subtracting a fraction of the gradient moves you towards the minimum!

Vanilla Gradient Descent Code

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J(\theta)$$

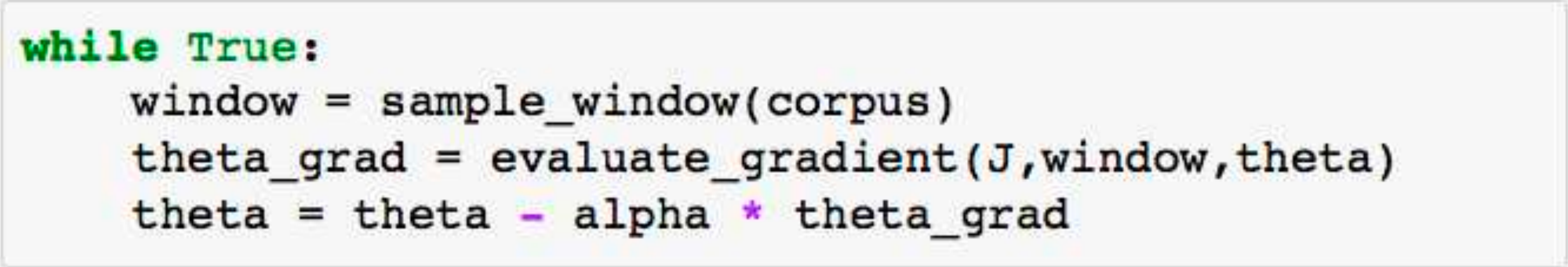


```
while True:
    theta_grad = evaluate_gradient(J,corpus,theta)
    theta = theta - alpha * theta_grad
```

Stochastic Gradient Descent

- But Corpus may have 40B tokens and windows
- You would wait a very long time before making a single update!
- Very bad idea for pretty much all neural nets!
- Instead: We update parameters after each window *t*
  → Stochastic gradient descent (SGD)

$$\theta^{new} = \theta^{old} - \alpha \nabla_{\theta} J_t(\theta)$$



```
while True:
    window = sample_window(corpus)
    theta_grad = evaluate_gradient(J,window,theta)
    theta = theta - alpha * theta_grad
```

Working out how to optimize a neural network is really all the chain rule!

Chain rule! If *y* = *f*(*u*) and *u* = *g*(*x*), i.e. *y* = *f*(*g*(x)), then:

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = \frac{df(u)}{du}\frac{dg(x)}{dx}$$

Simple example: $\frac{dy}{dx} = \frac{d}{dx}5(x^3+7)^4$

$$y = f(u) = 5u^4 \qquad u = g(x) = x^3 + 7$$

$$\frac{dy}{du} = 20u^3 \qquad \frac{du}{dx} = 3x^2$$

$$\frac{dy}{dx} = 20(x^3+7)^3.3x^2$$

Objective Function

Maximize $$J'(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} p(w'_{t+j} | w_t ; \theta)$$

Or minimize neg. log likelihood $$J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log p(w'_{t+j} | w_t)$$

[negate to minimize; log is monotone]

Text length

window size

where $$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

word IDs

Each word type (vocab entry) has <u>two</u> word representations: as center word and context word

We now take derivatives to work out minimum

$$\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w v_c)}$$

$$= \underbrace{\frac{\partial}{\partial v_c} \log \exp(u_o^T v_c)}_{①} - \underbrace{\frac{\partial}{\partial v_c} \log \sum_{w=1}^{V} \exp(u_w^T v_c)}_{②}$$

$$① \quad \frac{\partial}{\partial v_c} \underbrace{\log \exp}_{\text{inverses}} (u_o^T v_c) = \frac{\partial}{\partial v_c} u_o^T v_c = u_o$$

Vector!
Not high school single variable calculus

You can do things one variable at a time, and this may be helpful when things get gnarly.

$$\forall j \quad \frac{\partial}{\partial (v_c)_j} u_o^T v_c = \frac{\partial}{\partial (v_c)_j} \sum_{i=1}^{d} (u_o)_i (v_c)_i$$

$$= (u_o)_j$$

Each term is zero except when $i = j$

② $\frac{\partial}{\partial v_c} \underbrace{\log}_{f} \underbrace{\sum_{w=1}^{V} \exp(u_w{}^T v_c)}_{z=g(v_c)}$

$$= \underbrace{\frac{1}{\sum_{w=1}^{V} \exp(u_w{}^T v_c)}} \cdot \underbrace{\frac{\partial}{\partial v_c} \sum_{x=1}^{V} \exp(u_x{}^T v_c)}$$

Important to change index

$$\frac{\partial}{\partial v_c} f(\overbrace{g(v_c)}^{z}) = \frac{\partial f}{\partial z} \cdot \frac{\partial z}{\partial v_c}$$

Use chain rule

$$= \frac{1}{\sum_{w=1}^{V} \exp(u_w{}^T v_c)} \cdot \left(\sum_{x=1}^{V} \frac{\partial}{\partial v_c} \underbrace{\exp}_{f} (\underbrace{u_x{}^T v_c}_{z=g(v_c)}) \right)$$

Move deriv inside sum

$$\left(\sum_{x=1}^{V} \exp(u_x{}^T v_c) \frac{\partial}{\partial v_c} u_x{}^T v_c \right)$$

Chain rule

$$\left(\sum_{x=1}^{V} \exp(u_x{}^T v_c)\, u_x \right)$$

$$\frac{\partial}{\partial v_c} \log(p(o|c)) = u_o - \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \cdot \left(\sum_{x=1}^{V} \exp(u_x^T v_c)\, u_x \right)$$

$$= u_o - \sum_{x=1}^{V} \frac{\exp(u_x^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}\, u_x$$

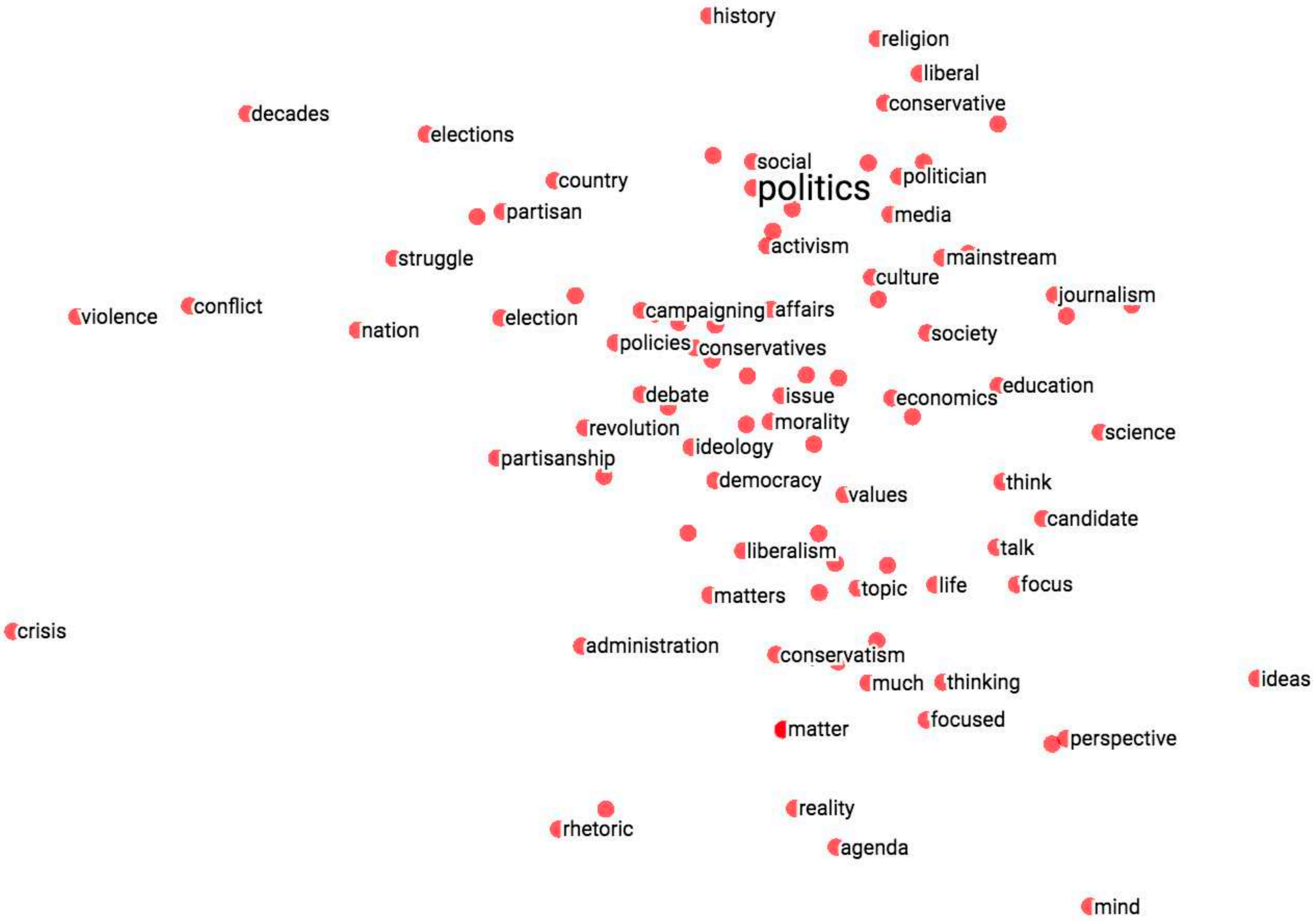
Distribute term across sum

$$= u_o - \underbrace{\sum_{x=1}^{V} p(x|c)\, u_x}$$

This an expectation: average over all context vectors weighted by their probability

= observed − expected

This is just the derivatives for the center vector parameters
Also need derivatives for output vector parameters (they're similar)
Then we have derivative w.r.t. all parameters and can minimize


history
religion
liberal
conservative
decades
elections
social
politics
politician
country
partisan
media
activism
struggle
mainstream
culture
journalism
violence
conflict
nation
election
campaigning
affairs
society
policies
conservatives
debate
issue
economics
education
revolution
morality
ideology
science
partisanship
democracy
values
think
candidate
liberalism
talk
matters
topic
life
focus
crisis
administration
conservatism
much
thinking
ideas
matter
focused
perspective
rhetoric
reality
agenda
mind

Linear Relationships in word2vec

These representations are *very good* at encoding similarity and dimensions of similarity!

- Analogies testing dimensions of similarity can be solved quite well just by doing vector subtraction in the embedding space

  Syntactically

  - $x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families}$
  - Similarly for verb and adjective morphological forms

  Semantically (Semeval 2012 task 2)

  - $x_{shirt} - x_{clothing} \approx x_{chair} - x_{furniture}$
  - $x_{king} - x_{man} \approx x_{queen} - x_{woman}$

Word Analogies

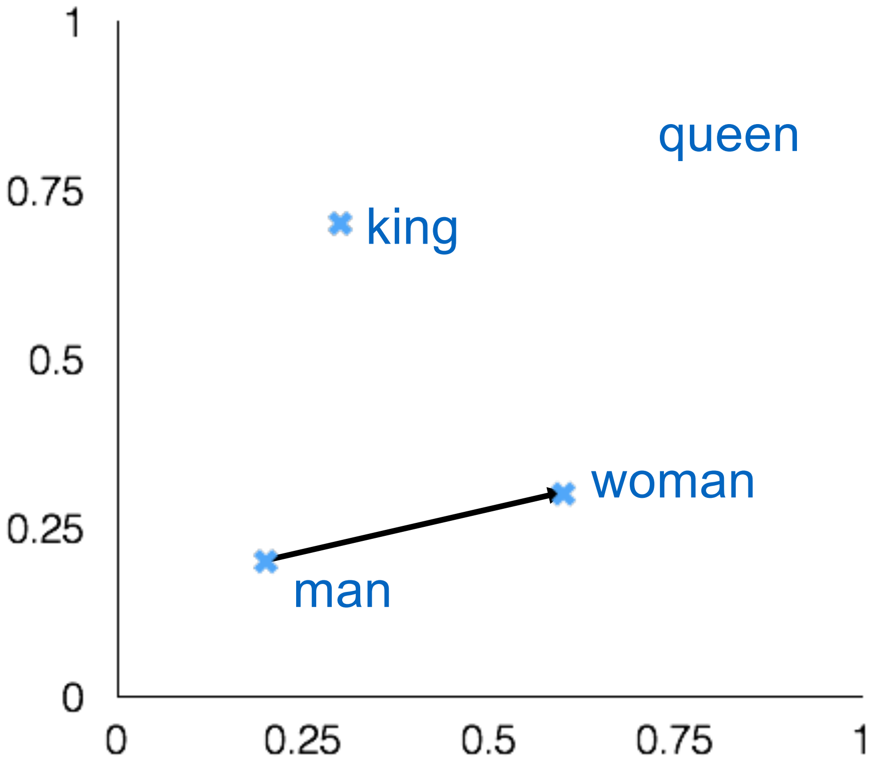
Test for linear relationships, examined by Mikolov et al.

a:b :: c:? →

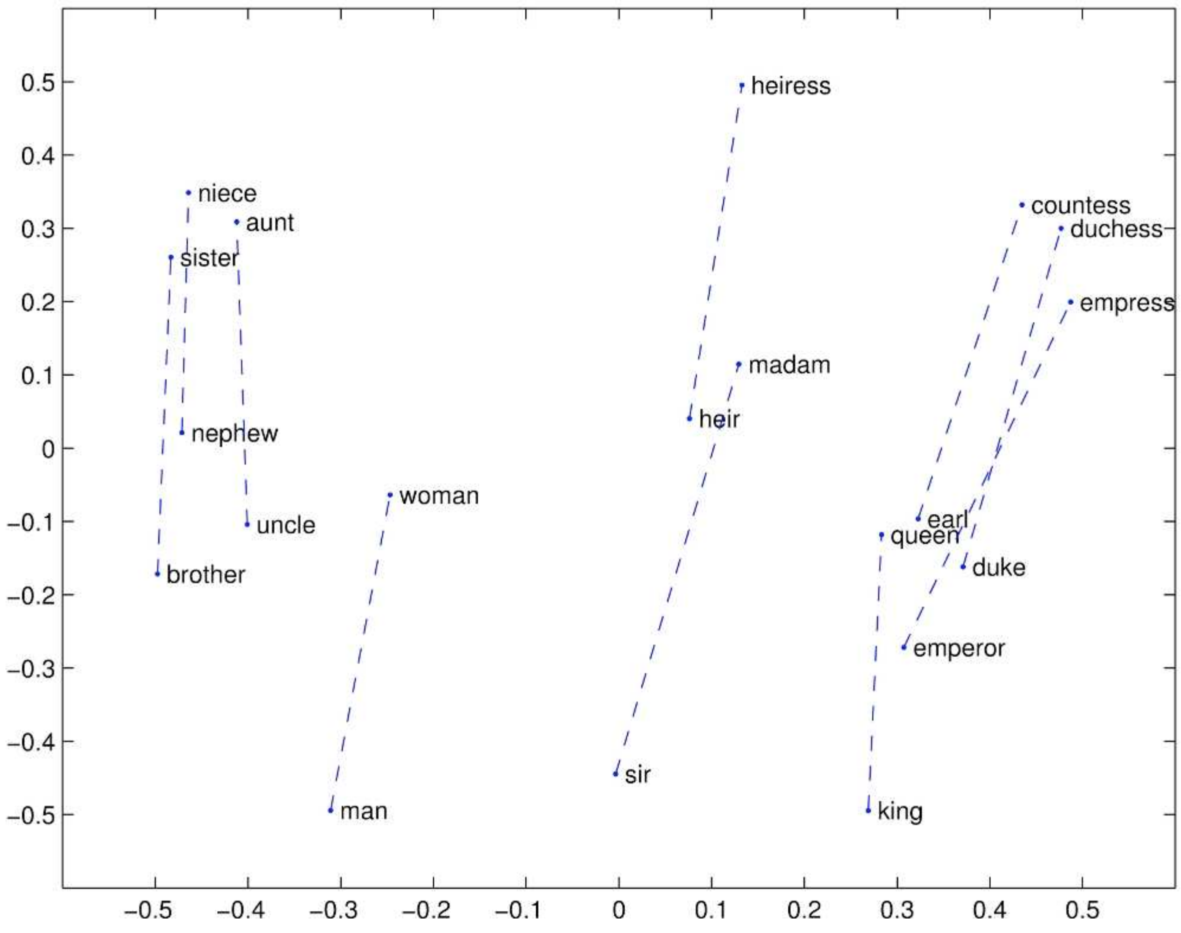
$$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

man:woman :: king:?

+	king	[0.30 0.70]
–	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]

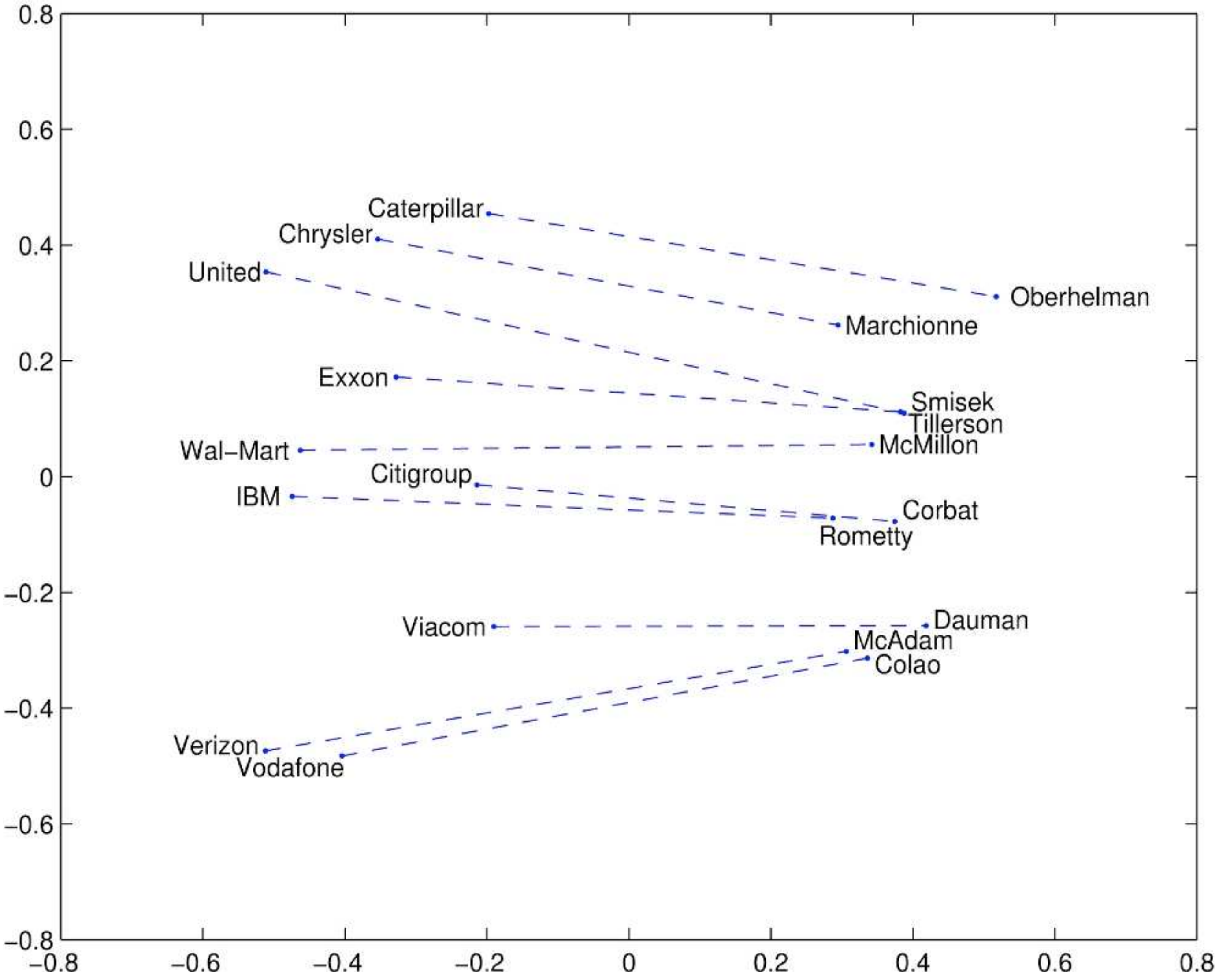


GloVe Visualizations

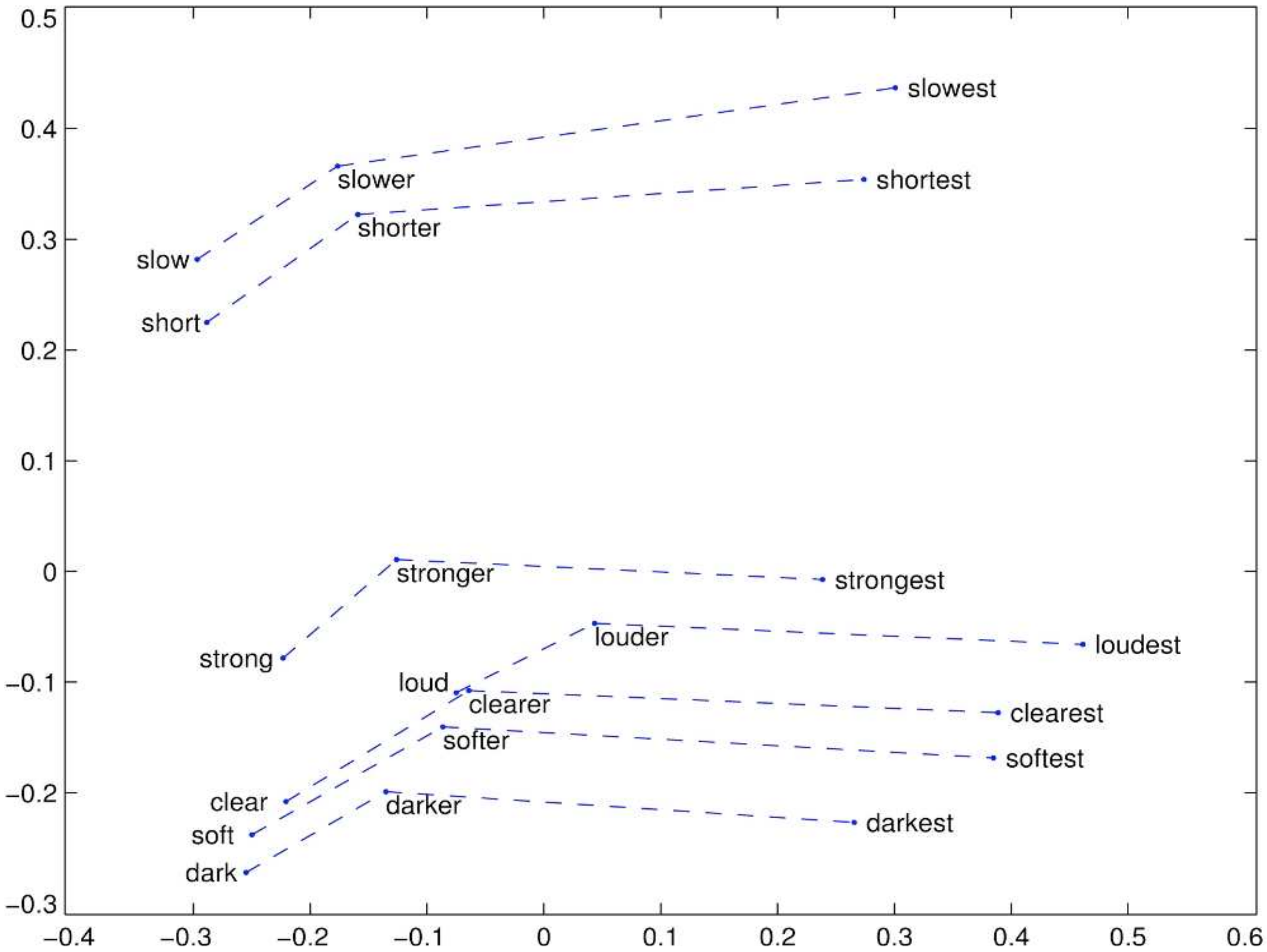


http://nlp.stanford.edu/projects/glove/

Glove Visualizations: Company - CEO



Glove Visualizations: Superlatives



Application to Information Retrieval

Application is just beginning – we’re “at the end of the early years”

- Google’s RankBrain – little is publicly known
  - Bloomberg article by Jack Clark (Oct 26, 2015): http://www.bloomberg.com/news/articles/2015-10-26/google-turning-its-lucrative-web-search-over-to-ai-machines
  - A result reranking system. “3rd most valuable ranking signal”
  - But note: more of the potential value is in the tail?
- New SIGIR Neu-IR workshop series (2016 on)

Σ
Neu-IR (2016)
The Neural Information Retrieval Workshop @ SIGIR
Pisa, Tuscany, Italy on 21st July, 2016
research.microsoft.com/neuir2016

An application to information retrieval

Nalisnick, Mitra, Craswell & Caruana. 2016. Improving Document Ranking with Dual Word Embeddings. *WWW 2016 Companion.* http://research.microsoft.com/pubs/260867/pp1291-Nalisnick.pdf

Mitra, Nalisnick, Craswell & Caruana. 2016. A Dual Embedding Space Model for Document Ranking. **arXiv:1602.01137 [cs.IR]**

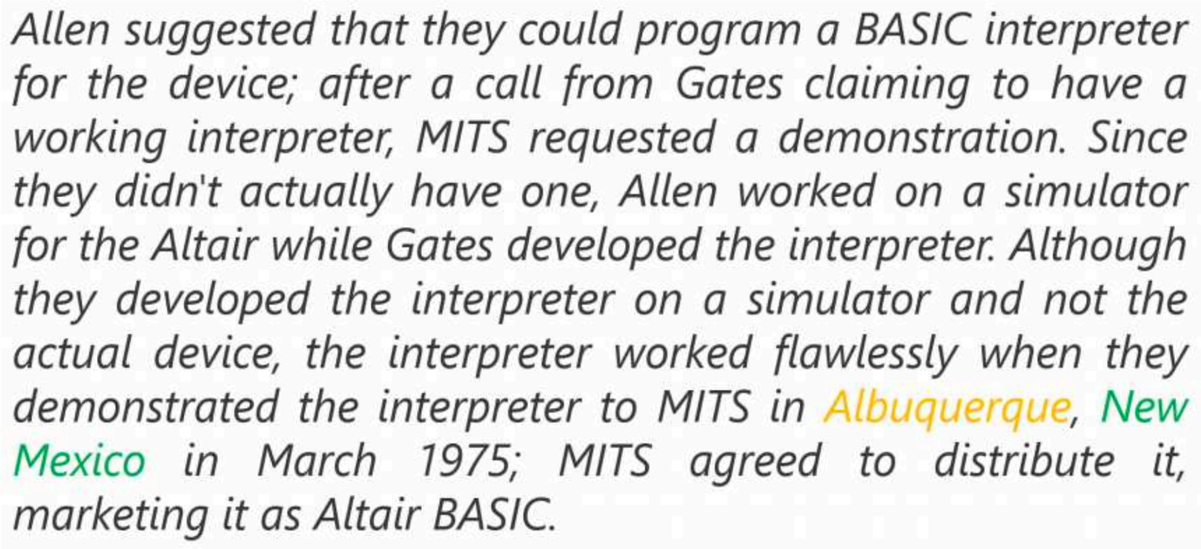
Builds on BM25 model idea of "aboutness"

- Not just term repetition indicating aboutness
- Relationship between query terms and *all* terms in the document indicates aboutness (BM25 uses only query terms)

Makes clever argument for different use of word and context vectors in word2vec's CBOW/SGNS or GloVe

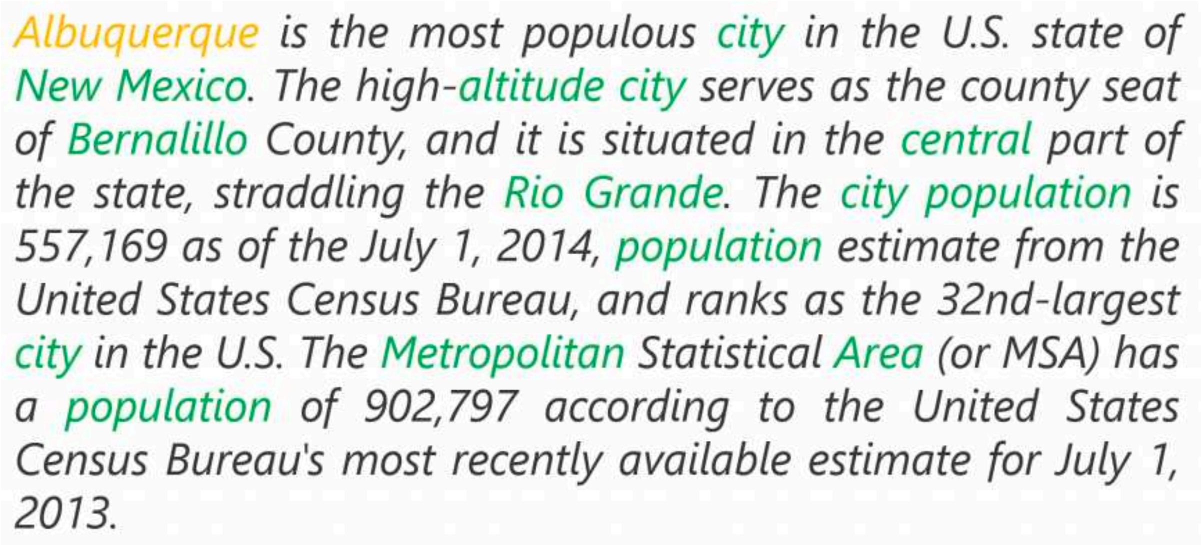
Modeling document aboutness: Results from a search for Albuquerque

d_1



Allen suggested that they could program a BASIC interpreter for the device; after a call from Gates claiming to have a working interpreter, MITS requested a demonstration. Since they didn't actually have one, Allen worked on a simulator for the Altair while Gates developed the interpreter. Although they developed the interpreter on a simulator and not the actual device, the interpreter worked flawlessly when they demonstrated the interpreter to MITS in Albuquerque, New Mexico in March 1975; MITS agreed to distribute it, marketing it as Altair BASIC.

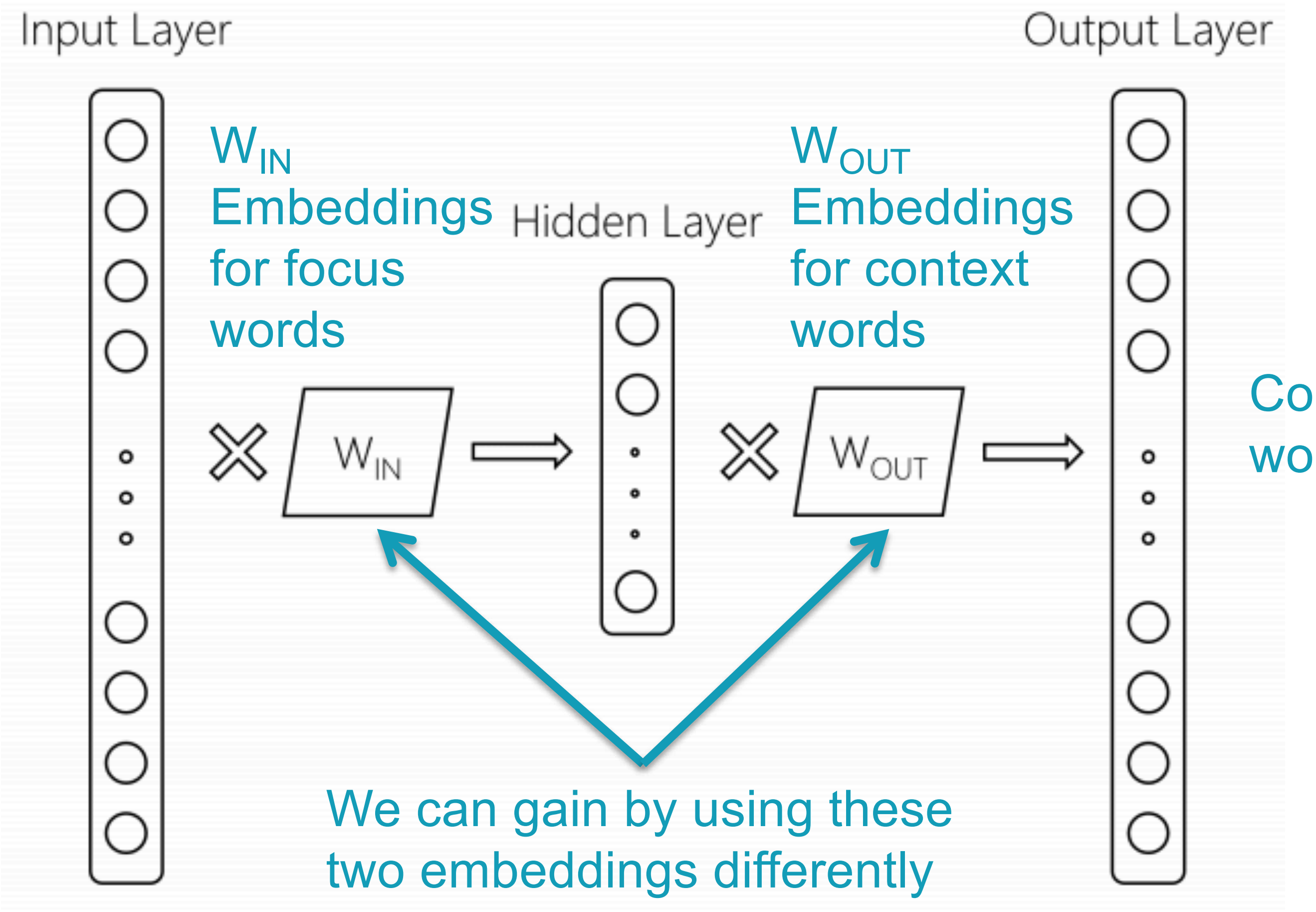
d_2



Albuquerque is the most populous city in the U.S. state of New Mexico. The high-altitude city serves as the county seat of Bernalillo County, and it is situated in the central part of the state, straddling the Rio Grande. The city population is 557,169 as of the July 1, 2014, population estimate from the United States Census Bureau, and ranks as the 32nd-largest city in the U.S. The Metropolitan Statistical Area (or MSA) has a population of 902,797 according to the United States Census Bureau's most recently available estimate for July 1, 2013.

Using 2 word embeddings

word2vec model with 1 word of context



Focus word

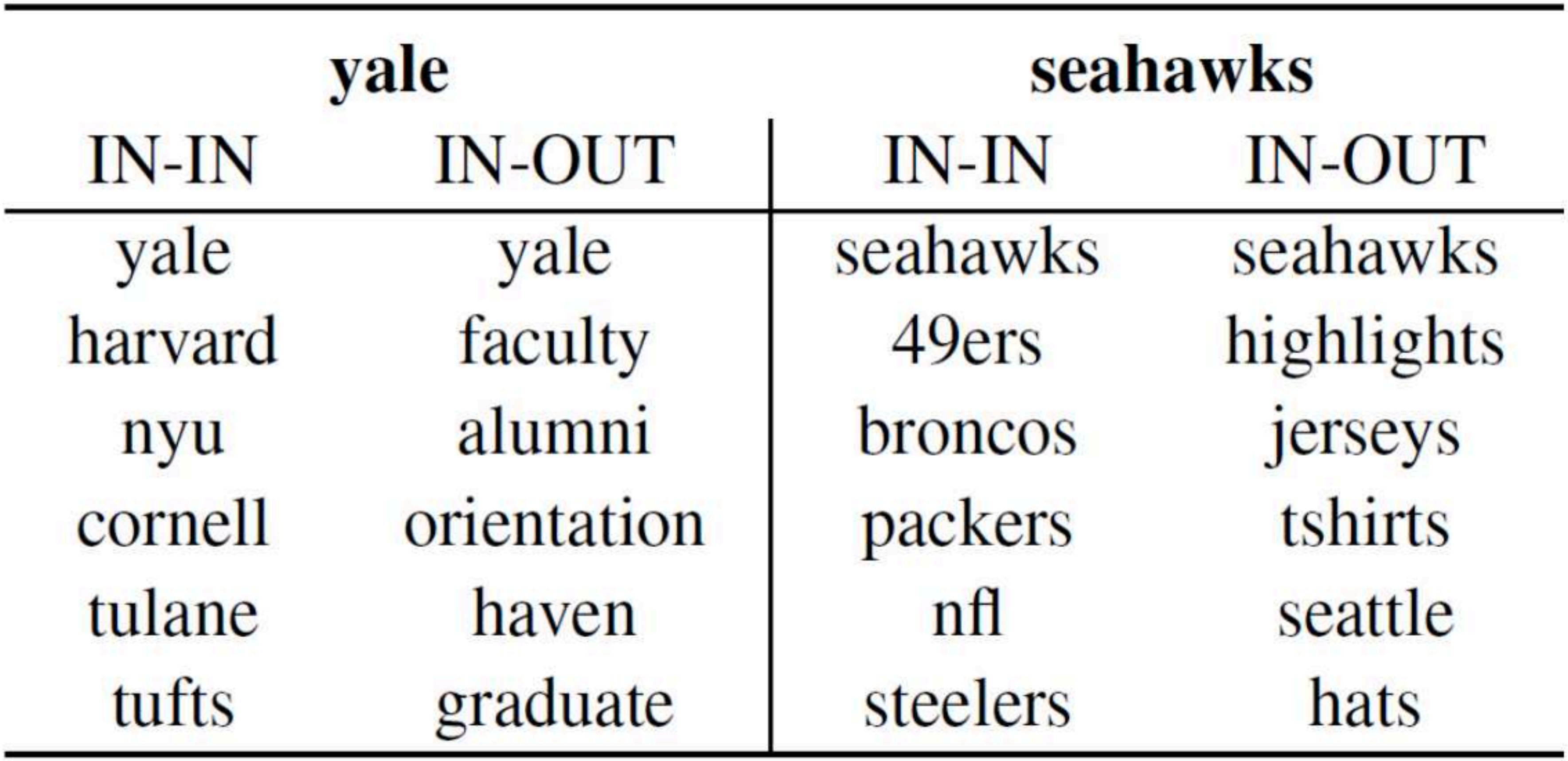
W_{IN} Embeddings for focus words

W_{OUT} Embeddings for context words

Context word

We can gain by using these two embeddings differently

Using 2 word embeddings



yale		seahawks	
IN-IN	IN-OUT	IN-IN	IN-OUT
yale	yale	seahawks	seahawks
harvard	faculty	49ers	highlights
nyu	alumni	broncos	jerseys
cornell	orientation	packers	tshirts
tulane	haven	nfl	seattle
tufts	graduate	steelers	hats

Dual Embedding Space Model (DESM)

- Simple model
- A document is represented by the centroid of its word vectors

$$\overline{\mathbf{D}} = \frac{1}{|D|} \sum_{\mathbf{d}_j \in D} \frac{\mathbf{d}_j}{||\mathbf{d}_j||}$$

- Query-document similarity is average over query words of cosine similarity

$$DESM(Q, D) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{\mathbf{q}_i^T \overline{\mathbf{D}}}{||\mathbf{q}_i|| \, ||\overline{\mathbf{D}}||}$$

Dual Embedding Space Model (DESM)

- What works best is to use the OUT vectors for the document and the IN vectors for the query

$$DESM_{IN-OUT}(Q,D) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{q_{IN,i}^T \overline{D_{OUT}}}{\|q_{IN,i}\| \|\overline{D_{OUT}}\|}$$

- This way similarity measures *aboutness* – words that appear with this word – which is more useful in this context than *(distributional) semantic similarity*

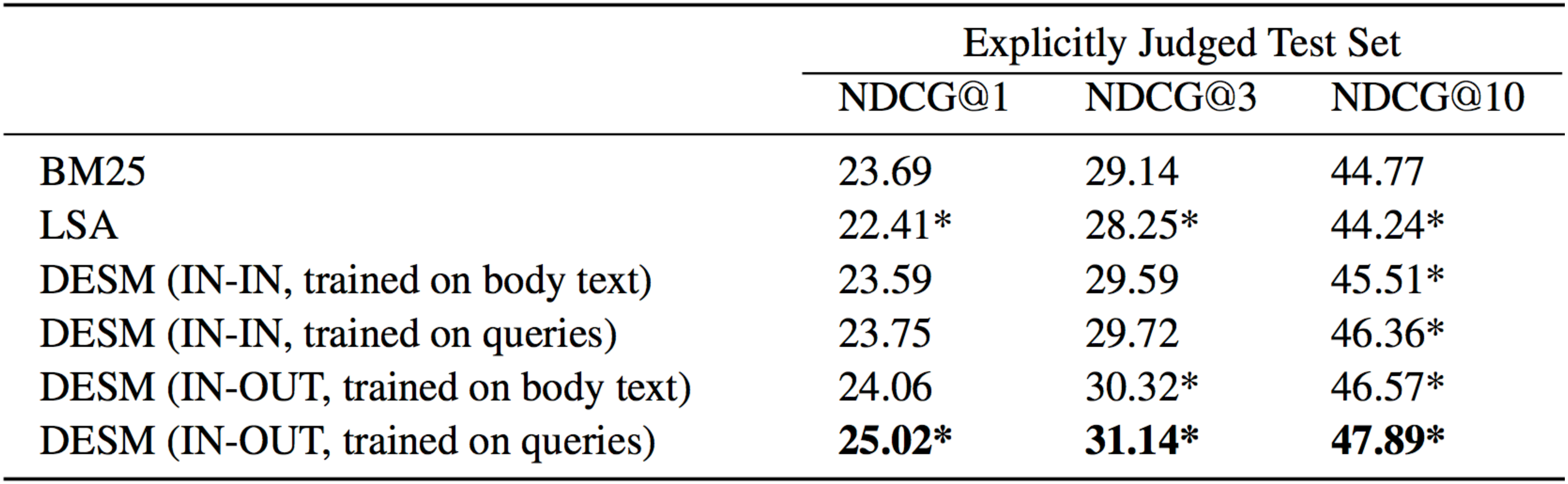
Experiments

- Train word2vec from either
  - 600 million Bing queries
  - 342 million web document sentences
- Test on 7,741 randomly sampled Bing queries
  - 5 level eval (Perfect, Excellent, Good, Fair, Bad)
- Two approaches
  1. Use DESM model to rerank top results from BM25
  2. Use DESM alone or a mixture model of it and BM25

$$MM(Q,D) = \alpha DESM(Q,D) + (1-\alpha)BM25(Q,D)$$

$$\alpha \in \mathbb{R}, 0 \leq \alpha \leq 1$$

Results – reranking *k*-best list

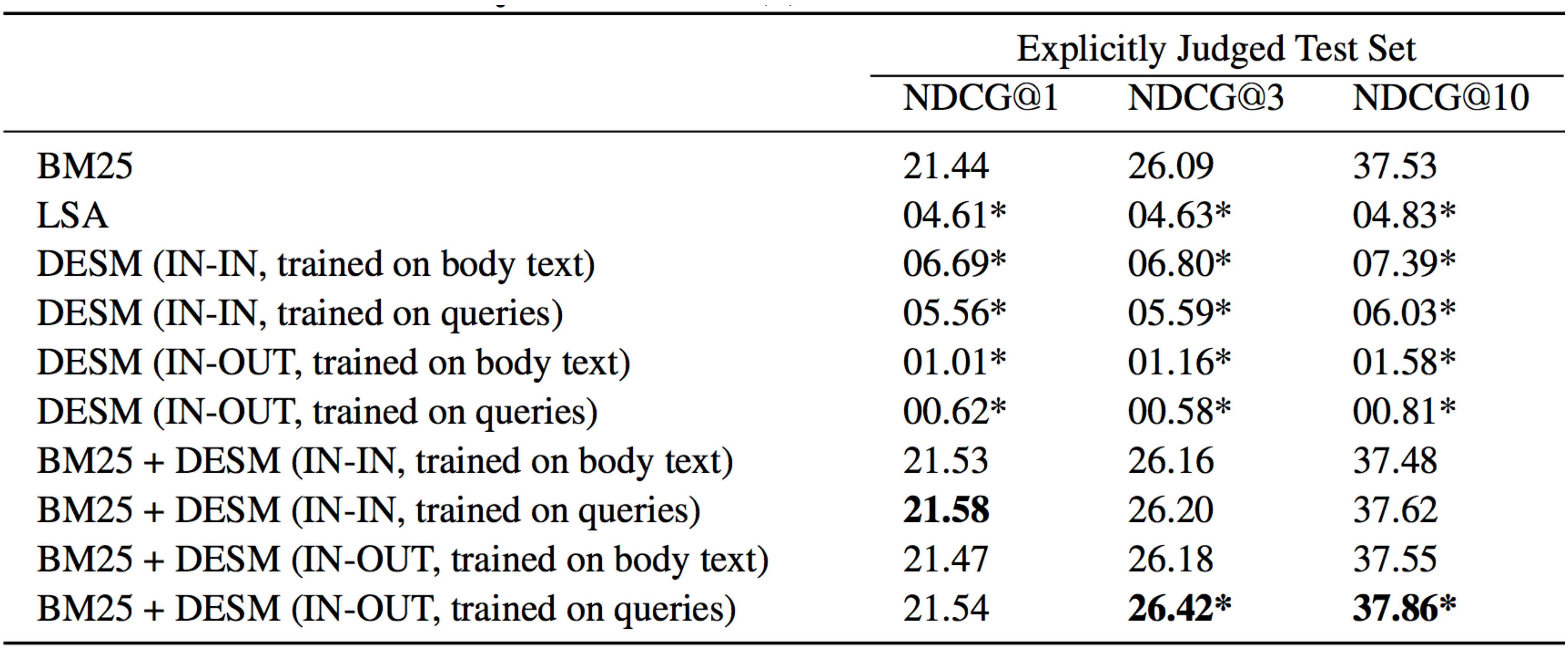


	Explicitly Judged Test Set		
	NDCG@1	NDCG@3	NDCG@10
BM25	23.69	29.14	44.77
LSA	22.41*	28.25*	44.24*
DESM (IN-IN, trained on body text)	23.59	29.59	45.51*
DESM (IN-IN, trained on queries)	23.75	29.72	46.36*
DESM (IN-OUT, trained on body text)	24.06	30.32*	46.57*
DESM (IN-OUT, trained on queries)	**25.02***	**31.14***	**47.89***

Pretty decent gains – e.g., 2% for NDCG@3

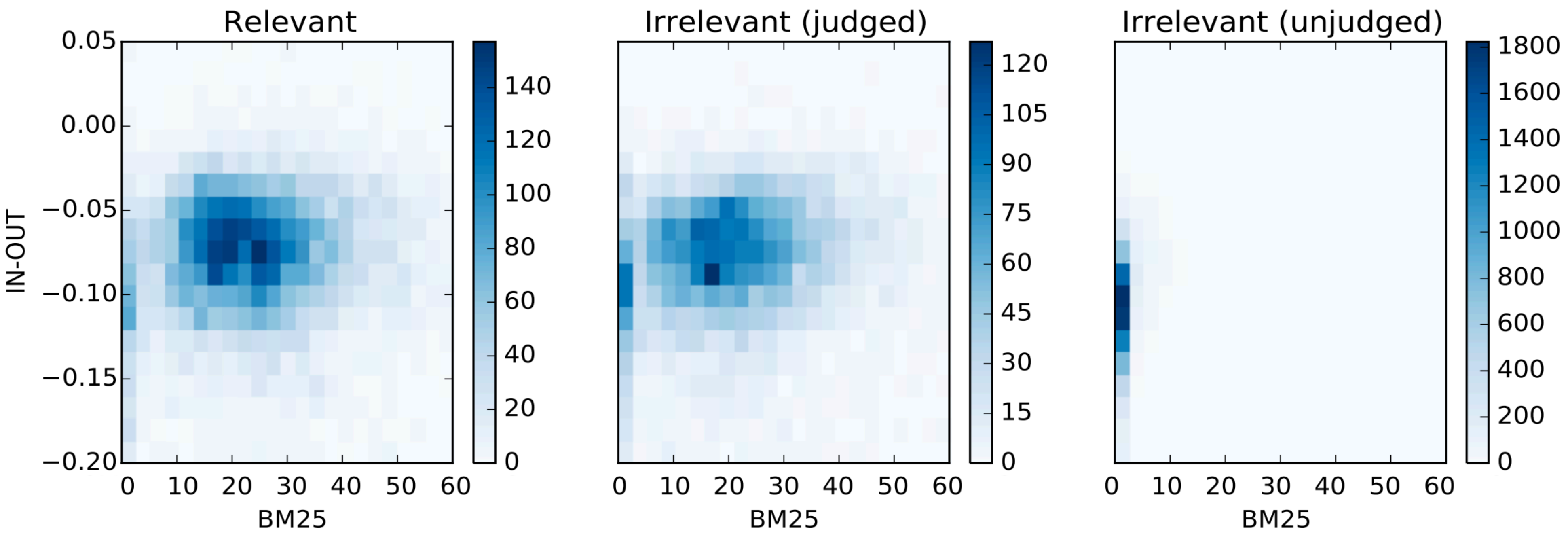
Gains are bigger for model trained on queries than docs

Results – whole ranking system



	Explicitly Judged Test Set		
	NDCG@1	NDCG@3	NDCG@10
BM25	21.44	26.09	37.53
LSA	04.61*	04.63*	04.83*
DESM (IN-IN, trained on body text)	06.69*	06.80*	07.39*
DESM (IN-IN, trained on queries)	05.56*	05.59*	06.03*
DESM (IN-OUT, trained on body text)	01.01*	01.16*	01.58*
DESM (IN-OUT, trained on queries)	00.62*	00.58*	00.81*
BM25 + DESM (IN-IN, trained on body text)	21.53	26.16	37.48
BM25 + DESM (IN-IN, trained on queries)	**21.58**	26.20	37.62
BM25 + DESM (IN-OUT, trained on body text)	21.47	26.18	37.55
BM25 + DESM (IN-OUT, trained on queries)	21.54	**26.42***	**37.86***

A possible explanation



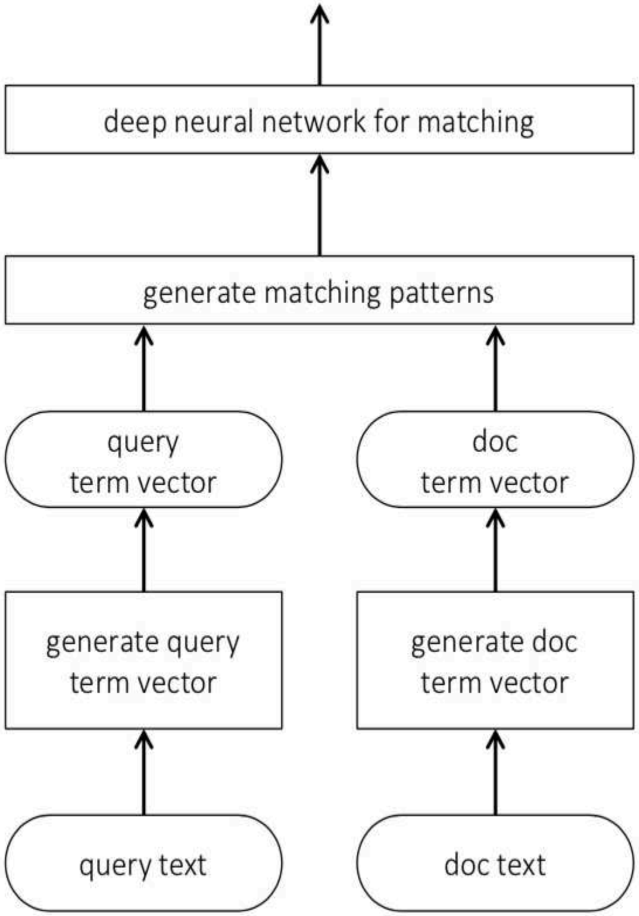
IN-OUT has some ability to prefer Relevant to close-by (judged) non-relevant, but it's scores induce too much noise vs. BM25 to be usable alone

DESM conclusions

- DESM is a weak ranker but effective at finding subtler similarities/aboutness
- It is effective at, but only at, reranking at least somewhat relevant documents
    - For example, DESM can confuse Oxford and Cambridge
    - Bing rarely makes an Oxford/Cambridge mistake!

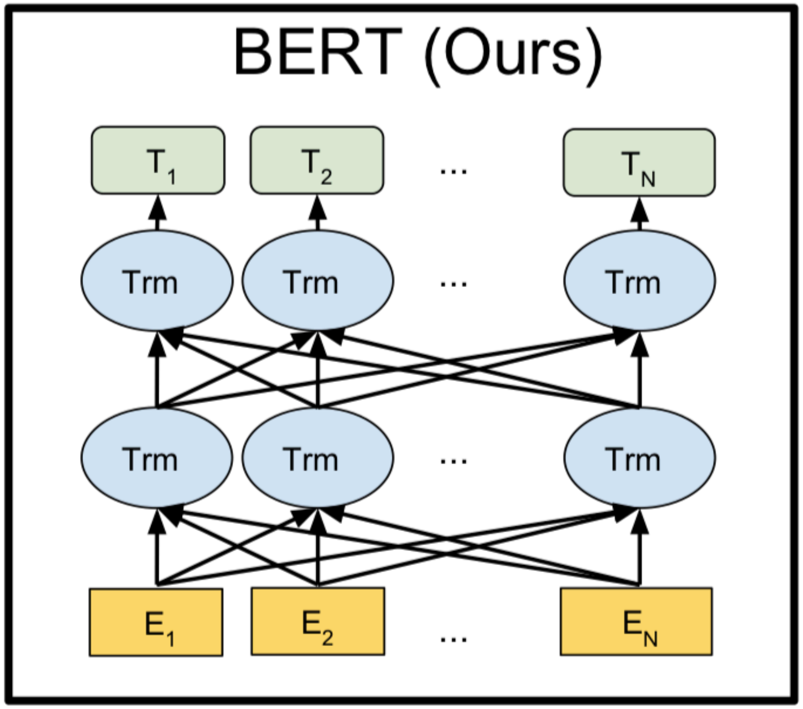
What else can neural nets do in IR?

- Use a neural network as a supervised reranker
- Assume a query and document embedding network (as we have discussed)
- Assume you have (q,d,rel) relevance data
- Learn a neural network (with supervised learning) to predict relevance of (q,d) pair
- An example of "machine-learned relevance", which we'll talk about more next lecture

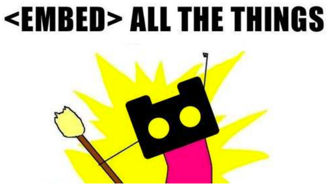

deep neural network for matching
generate matching patterns
query term vector
doc term vector
generate query term vector
generate doc term vector
query text
doc text

What else can neural nets do in IR?

- BERT: Devlin, Chang, Lee, Toutanova (2018)
- A deep transformer-based neural network
- Builds per-token (in context) representations
- Produces a query/document representation as well
- Or jointly embed query and document and ask for a retrieval score
- Incredibly effective!
- https://arxiv.org/abs/1810.04805



Summary: Embed all the things!



Word embeddings are the hot new technology (again!)

Lots of applications wherever knowing word context or similarity helps prediction:

- Synonym handling in search
- Document aboutness
- Ad serving
- Language models: from spelling correction to email response
- Machine translation
- Sentiment analysis
- …

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 7: Scores in a complete search system
Handout version

Petr Sojka, Hinrich Schütze et al.

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

2020-03-18

Overview

1. Why rank?
2. More on cosine
3. The complete search system
4. Implementation of ranking

Take-away today

- The importance of ranking: User studies at Google
- Length normalization: Pivot normalization
- The complete search system
- Implementation of ranking

Why is ranking so important?

- Last lecture: Problems with unranked retrieval
  - Users want to look at a few results – not thousands.
  - It's very hard to write queries that produce a few results.
  - Even for expert searchers
  - → Ranking is important because it effectively reduces a large set of results to a very small one.
- Next: More data on "users only look at a few results"
- Actually, in the vast majority of cases they only examine 1, 2, or 3 results.

Empirical investigation of the effect of ranking

- The following slides are from Dan Russell's JCDL talk
- Dan Russell was the "Über Tech Lead for Search Quality & User Happiness" at Google.
- How can we measure how important ranking is?
- Observe what searchers do when they are searching in a controlled setting
  - Videotape them
  - Ask them to "think aloud"
  - Interview them
  - Eye-track them
  - Time them
  - Record and count their clicks

Interview video

So.. Did you notice the FTD official site?

To be honest, I didn't even look at that.

At first I saw "from $20" and $20 is what I was looking for.

To be honest, 1800-flowers is what I'm familiar with and why I went there next even though I kind of assumed they wouldn't have $20 flowers

And you knew they were expensive?

I knew they were expensive but I thought "hey, maybe they've got some flowers for under $20 here…"

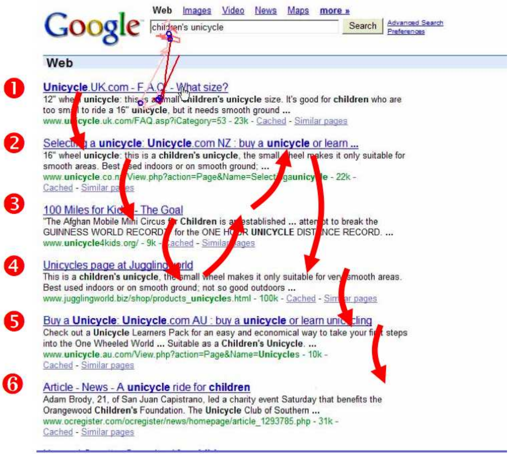
But you didn't notice the FTD?

No I didn't, actually… that's really funny.

Rapidly scanning the results

Note scan pattern:

Page 3:
Result 1
Result 2
Result 3
Result 4
Result 3
Result 2
Result 4
Result 5
Result 6 <click>



Web

1 **Unicycle**.UK.com - F.A.Q. - What size?
12" wheel **unicycle**: this is a small **children's unicycle** size. It's good for **children** who are too small to ride a 16" **unicycle**, but it needs smooth ground **...**
www.**unicycle**.uk.com/FAQ.asp?iCategory=53 - 23k - Cached - Similar pages

2 Selecting a **unicycle**: **Unicycle**.com NZ : buy a **unicycle** or learn **...**
16" wheel **unicycle**: this is a **children's unicycle**, the small wheel makes it only suitable for smooth areas. Best used indoors or on smooth ground; **...**
www.**unicycle**.co.nz/View.php?action=Page&Name=Selectingaunicycle - 22k -
Cached - Similar pages

3 100 Miles for Kids - The Goal
"The Afghan Mobile Mini Circus for **Children** is an established **...** attempt to break the GUINNESS WORLD RECORD" for the ONE HOUR **UNICYCLE** DISTANCE RECORD. **...**
www.**unicycle**4kids.org/ - 9k - Cached - Similar pages

4 Unicycles page at Juggling world
This is a **children's unicycle**, the small wheel makes it only suitable for very smooth areas. Best used indoors or on smooth ground; not so good outdoors **...**
www.jugglingworld.biz/shop/products_**unicycle**s.html - 100k - Cached - Similar pages

5 Buy a **Unicycle**: **Unicycle**.com AU : buy a **unicycle** or learn unicycling
Check out a **Unicycle** Learners Pack for an easy and economical way to take your first steps into the One Wheeled World **...** Suitable as a **Children's Unicycle**. **...**
www.**unicycle**.au.com/View.php?action=Page&Name=**Unicycle**s - 10k -
Cached - Similar pages

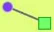
6 Article - News - A **unicycle** ride for **children**
Adam Brody, 21, of San Juan Capistrano, led a charity event Saturday that benefits the Orangewood **Children's** Foundation. The **Unicycle** Club of Southern **...**
www.ocregister.com/ocregister/news/homepage/article_1293785.php - 31k -
Cached - Similar pages

Q: Why do this?

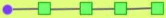
A: What's learned later influences judgment of earlier content.

Kinds of behaviors we see in the data

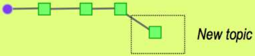
Short / Nav



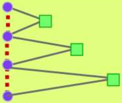
Topic exploration



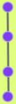
Topic switch



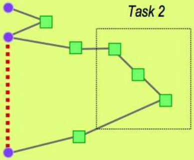
Methodical results exploration



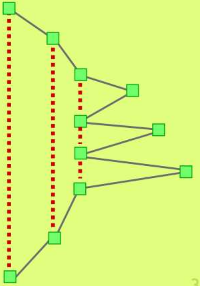
Query reform



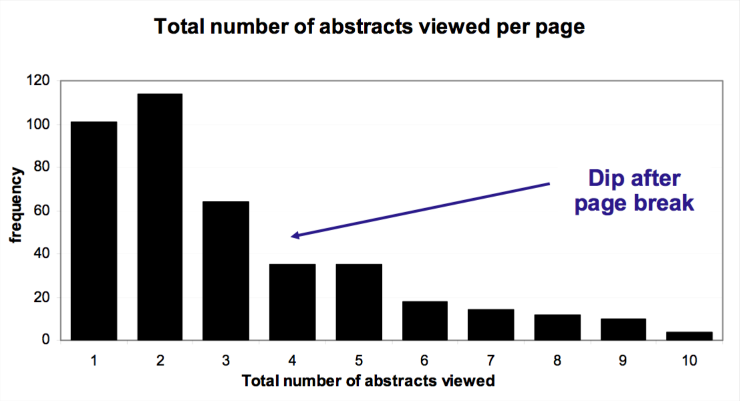
Multitasking



Stacking behavior

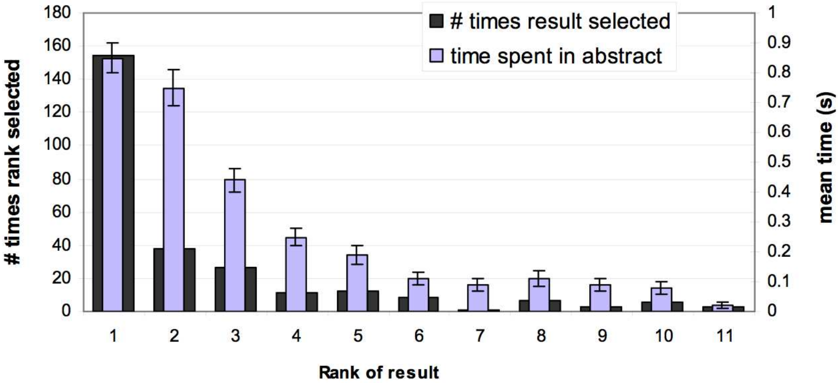


How many links do users view?



Mean: 3.07 Median/Mode: 2.00

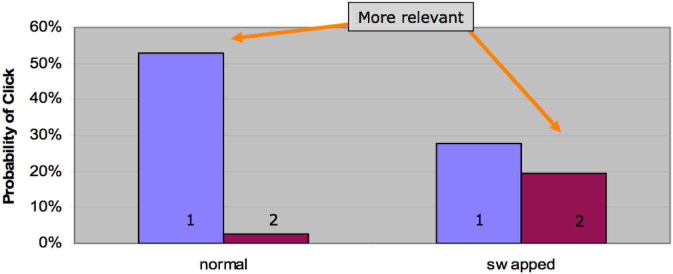
Looking vs. Clicking



- Users view results one and two more often / thoroughly
- Users click most frequently on result one

Presentation bias – reversed results

- Order of presentation influences where users look **AND** where they click



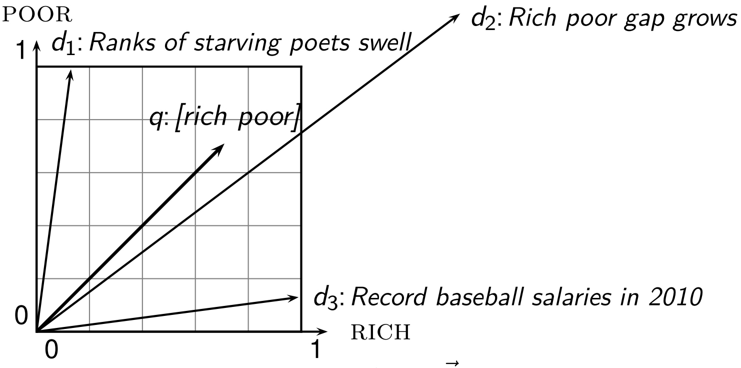
Importance of ranking: Summary

- Viewing abstracts: Users are a lot more likely to read the abstracts of the top-ranked pages (1, 2, 3, 4) than the abstracts of the lower ranked pages (7, 8, 9, 10).
- Clicking: Distribution is even more skewed for clicking
- In 1 out of 2 cases, users click on the top-ranked page.
- Even if the top-ranked page is not relevant, 30% of users will click on it.
- → Getting the ranking right is very important.
- → Getting the top-ranked page right is most important.

Exercise

- Ranking is also one of the high barriers to entry for competitors to established players in the search engine market.
- Why?

Why distance is a bad idea


POOR
1
0
0
1
RICH
d_1: *Ranks of starving poets swell*
d_2: *Rich poor gap grows*
q: *[rich poor]*
d_3: *Record baseball salaries in 2010*

The Euclidean distance of $\vec{q}$ and $\vec{d}_2$ is large although the distribution of terms in the query q and the distribution of terms in the document d_2 are very similar.

That's why we do length normalization or, equivalently, use cosine to compute query-document matching scores.

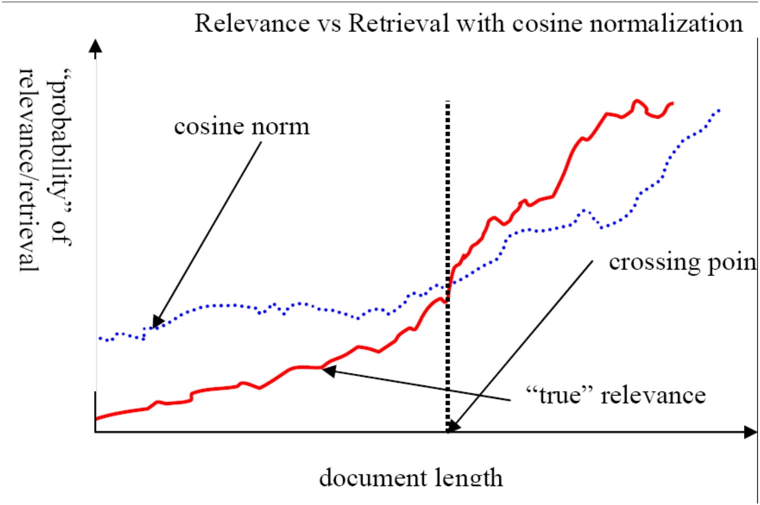
Exercise: A problem for cosine normalization

- Query q: “anti-doping rules Beijing 2008 olympics”
- Compare three documents
  - d_1: a short document on anti-doping rules at 2008 Olympics
  - d_2: a long document that consists of a copy of d_1 and 5 other news stories, all on topics different from Olympics/anti-doping
  - d_3: a short document on anti-doping rules at the 2004 Athens Olympics
- What ranking do we expect in the vector space model?
- What can we do about this?

Pivot normalization

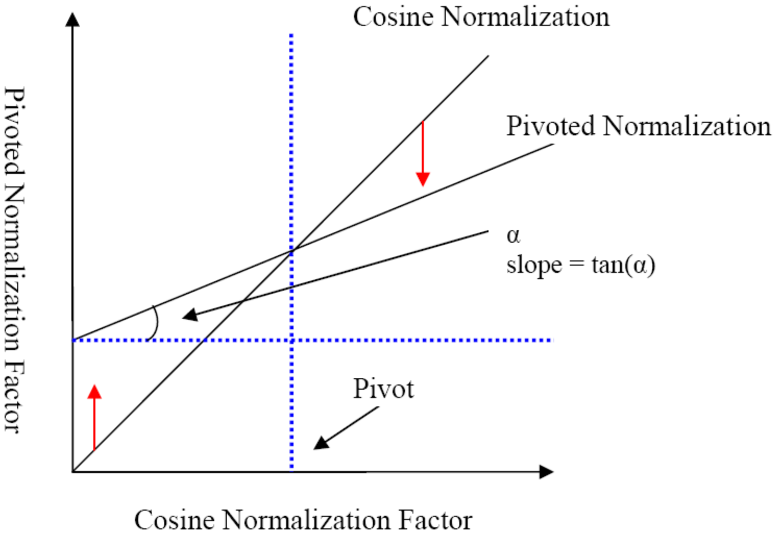
- Cosine normalization produces weights that are too large for short documents and too small for long documents (on average).
- Adjust cosine normalization by linear adjustment: “turning” the average normalization on the pivot
- Effect: Similarities of short documents with query decrease; similarities of long documents with query increase.
- This removes the unfair advantage that short documents have.

Predicted and true probability of relevance



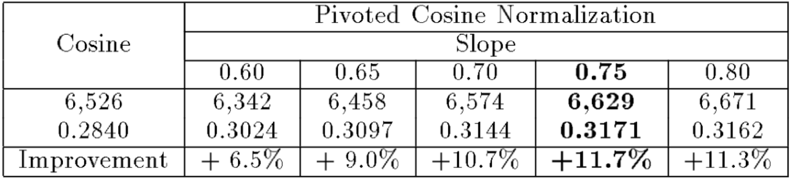
source:
Lillian Lee

Pivot normalization



source:
Lillian Lee

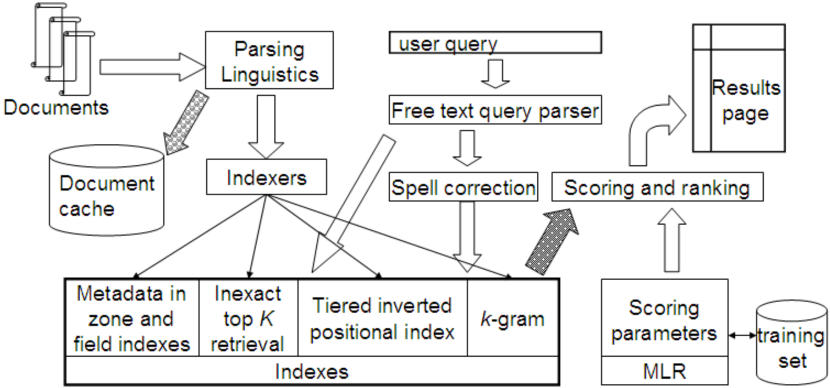
Pivoted normalization: Amit Singhal's experiments



Cosine	Pivoted Cosine Normalization				
	Slope				
	0.60	0.65	0.70	**0.75**	0.80
6,526 0.2840	6,342 0.3024	6,458 0.3097	6,574 0.3144	**6,629** **0.3171**	6,671 0.3162
Improvement	+ 6.5%	+ 9.0%	+10.7%	**+11.7%**	+11.3%

(relevant documents retrieved and (change in) average precision)

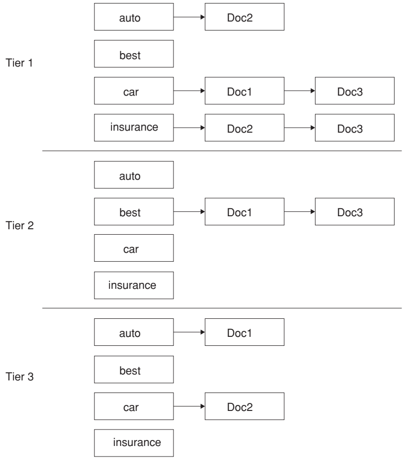
Complete search system



Tiered indexes

- Basic idea:
  - Create several tiers of indexes, corresponding to importance of indexing terms
  - During query processing, start with highest-tier index
  - If highest-tier index returns at least k (e.g., $k = 100$) results: stop and return results to user
  - If we've only found $< k$ hits: repeat for next index in tier cascade
- Example: two-tier system
  - Tier 1: Index of all titles
  - Tier 2: Index of the rest of documents
  - Pages containing the search words in the title are better hits than pages containing the search words in the body of the text.

Tiered index



Tier 1

auto → Doc2

best

car → Doc1 → Doc3

insurance → Doc2 → Doc3

Tier 2

auto

best → Doc1 → Doc3

car

insurance

Tier 3

auto → Doc1

best

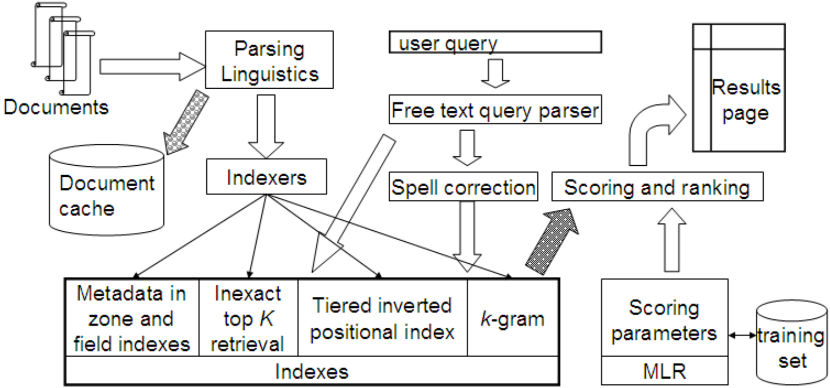
car → Doc2

insurance

Tiered indexes

- The use of tiered indexes is believed to be one of the reasons that Google search quality was significantly higher initially (2000/01) than that of competitors.
- (along with PageRank, use of anchor text and proximity constraints)

Complete search system



Components we have introduced thus far

- Document preprocessing (linguistic and otherwise)
- Positional indexes
- Tiered indexes
- Spelling correction
- k-gram indexes for wildcard queries and spelling correction
- Query processing
- Document scoring

Components we haven't covered yet

- Document cache: we need this for generating snippets (= dynamic summaries)
- Zone indexes: They separate the indexes for different zones: the body of the document, all highlighted text in the document, anchor text, text in metadata fields,. . .
- Machine-learned ranking functions
- Proximity ranking (e.g., rank documents in which the query terms occur in the same local window higher than documents in which the query terms occur far from each other)
- Query parser

Components we haven't covered yet: Query parser

- IR systems often guess what the user intended.
- The two-term query *London tower* (without quotes) may be interpreted as the phrase query *"London tower"*.
- The query *100 Madison Avenue, New York* may be interpreted as a request for a map.
- How do we "parse" the query and translate it into a formal specification containing phrase operators, proximity operators, indexes to search etc.?

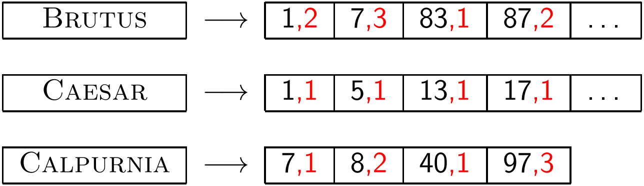
Vector space retrieval: Interactions

- How do we combine phrase retrieval with vector space retrieval?
- We do not want to compute document frequency / idf for every possible phrase. Why?
- How do we combine Boolean retrieval with vector space retrieval?
- For example: "$+$"-constraints and "$-$"-constraints
- Postfiltering is simple, but can be very inefficient – no easy answer.
- How do we combine wild cards with vector space retrieval?
- Again, no easy answer.

Exercise

- Design criteria for tiered system
  - Each tier should be an order of magnitude smaller than the next tier.
  - The top 100 hits for most queries should be in tier 1, the top 100 hits for most of the remaining queries in tier 2 etc.
  - We need a simple test for “can I stop at this tier or do I have to go to the next one?”
    - There is no advantage to tiering if we have to hit most tiers for most queries anyway.
- Consider a two-tier system where the first tier indexes titles and the second tier everything.
- Question: Can you think of a better way of setting up a multitier system? Which “zones” of a document should be indexed in the different tiers (title, body of document, others?)? What criterion do you want to use for including a document in tier 1?

Now we also need term frequencies in the index



BRUTUS	⟶	1,2	7,3	83,1	87,2	…
CAESAR	⟶	1,1	5,1	13,1	17,1	…
CALPURNIA	⟶	7,1	8,2	40,1	97,3	

term frequencies

We also need positions. Not shown here.

Term frequencies in the inverted index

- Thus: In each posting, store $\mathrm{tf}_{t,d}$ in addition to docID d.
- As an integer frequency, not as a (log-)weighted real number . . .
- . . . because real numbers are difficult to compress.
- Overall, additional space requirements are small: a byte per posting or less

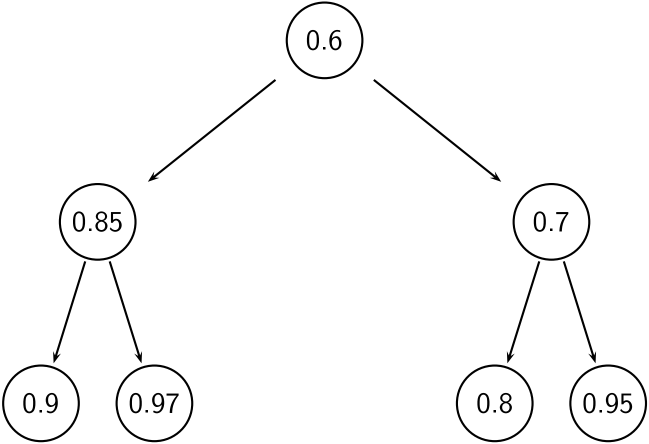
How do we compute the top k in ranking?

- We usually do not need a complete ranking.
- We just need the top k for a small k (e.g., $k = 100$).
- If we don't need a complete ranking, is there an efficient way of computing just the top k?
- Naïve:
  - Compute scores for all N documents
  - Sort
  - Return the top k
- Not very efficient
- Alternative: min heap

Use min heap for selecting top k ouf of N

- A binary min heap is a binary tree in which each node's value is less than the values of its children.
- Takes $O(N \log k)$ operations to construct (where N is the number of documents) . . .
- . . . then read off k winners in $O(k \log k)$ steps

Binary min heap



Selecting top k scoring documents in $O(N \log k)$

- Goal: Keep the top k documents seen so far
- Use a binary min heap
- To process a new document d' with score s':
  - Get current minimum h_m of heap ($O(1)$)
  - If $s' \leq h_m$ skip to next document
  - If $s' > h_m$ heap-delete-root ($O(\log k)$)
  - Heap-add d'/s' ($O(\log k)$)

Even more efficient computation of top k?

- Ranking has time complexity $O(N)$ where N is the number of documents.
- Optimizations reduce the constant factor, but they are still $O(N)$, $N > 10^{10}$
- Are there sublinear algorithms?
- What we're doing in effect: solving the k-nearest neighbor (kNN) problem for the query vector (= query point).
- There are no general solutions to this problem that are sublinear.

More efficient computation of top *k*: Heuristics

- Idea 1: Reorder postings lists
  - Instead of ordering according to docID . . .
  - . . . order according to some measure of "expected relevance".
- Idea 2: Heuristics to prune the search space
  - Not guaranteed to be correct . . .
  - . . . but fails rarely.
  - In practice, close to constant time.
  - For this, we'll need the concepts of document-at-a-time processing and term-at-a-time processing.

Non-docID ordering of postings lists

- So far: postings lists have been ordered according to docID.
- Alternative: a query-independent measure of "goodness" (credibility) of a page
- Example: PageRank $g(d)$ of page d, a measure of how many "good" pages hyperlink to d (chapter 21)
- Order documents in postings lists according to PageRank: $g(d_1) > g(d_2) > g(d_3) > \dots$
- Define composite score of a document:

$$\text{net-score}(q, d) = g(d) + \cos(q, d)$$

- This scheme supports early termination: We do not have to process postings lists in their entirety to find top k.

Non-docID ordering of postings lists (2)

- Order documents in postings lists according to PageRank: $g(d_1) > g(d_2) > g(d_3) > \dots$
- Define composite score of a document:

$$\text{net-score}(q, d) = g(d) + \cos(q, d)$$

- Suppose: (i) $g \rightarrow [0, 1]$; (ii) $g(d) < 0.1$ for the document d we're currently processing; (iii) smallest top k score we've found so far is 1.2
- Then all subsequent scores will be < 1.1.
- So we've already found the top k and can stop processing the remainder of postings lists.
- Questions?

Document-at-a-time processing

- Both docID-ordering and PageRank-ordering impose a consistent ordering on documents in postings lists.
- Computing cosines in this scheme is document-at-a-time.
- We complete computation of the query-document similarity score of document d_i before starting to compute the query-document similarity score of d_{i+1}.
- Alternative: term-at-a-time processing

Weight-sorted postings lists

- Idea: don't process postings that contribute little to final score
- Order documents in postings list according to weight
- Simplest case: normalized tf-idf weight (rarely done: hard to compress)
- Documents in the top k are likely to occur early in these ordered lists.
- $\rightarrow$ Early termination while processing postings lists is unlikely to change the top k.
- But:
  - We no longer have a consistent ordering of documents in postings lists.
  - We no longer can employ document-at-a-time processing.

Term-at-a-time processing

- Simplest case: completely process the postings list of the first query term
- Create an accumulator for each docID you encounter
- Then completely process the postings list of the second query term
- . . . and so forth

Term-at-a-time processing

```
CosineScore($q$)
float $Scores[N] = 0$
float $Length[N]$
for each query term $t$
do calculate $\mathrm{w}_{t,q}$ and fetch postings list for $t$
    for each pair($d, \mathrm{tf}_{t,d}$) in postings list
    do $Scores[d] += \mathrm{w}_{t,d} \times \mathrm{w}_{t,q}$
Read the array $Length$
for each $d$
do $Scores[d] = Scores[d]/Length[d]$
return Top $k$ components of $Scores[]$
```

The elements of the array “Scores” are called accumulators.

Computing cosine scores

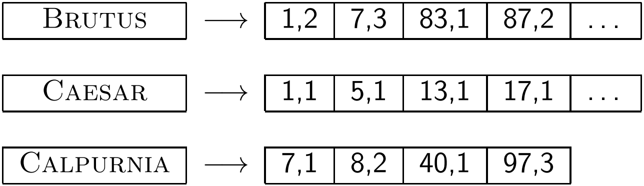
- Use inverted index
- At query time use an array of accumulators A to store sum (= the cosine score)
- $$A_j = \sum_k w_{qk} \cdot w_{d_j k}$$

  (for document d_j)
- "Accumulate" scores as postings lists are being processed.

Accumulators

- For the web (20 billion documents), an array of accumulators A in memory is infeasible.
- Thus: Only create accumulators for docs occurring in postings lists
- This is equivalent to: Do not create accumulators for docs with zero scores (i.e., docs that do not contain any of the query terms)

Accumulators: Example



Brutus ⟶ 1,2 | 7,3 | 83,1 | 87,2 | . . .

Caesar ⟶ 1,1 | 5,1 | 13,1 | 17,1 | . . .

Calpurnia ⟶ 7,1 | 8,2 | 40,1 | 97,3

- For query: [Brutus Caesar]:
- Only need accumulators for 1, 5, 7, 13, 17, 83, 87
- Don't need accumulators for 3, 8 etc.

Enforcing conjunctive search

- We can enforce conjunctive search (à la Google): only consider documents (and create accumulators) if all terms occur.
- Example: just one accumulator for [Brutus Caesar] in the example above . . .
- . . . because only d_1 contains both words.

Implementation of ranking: Summary

- Ranking is very expensive in applications where we have to compute similarity scores for all documents in the collection.
- In most applications, the vast majority of documents have similarity score 0 for a given query → lots of potential for speeding things up.
- However, there is no fast nearest neighbor algorithm that is guaranteed to be correct even in this scenario.
- In practice: use heuristics to prune search space – usually works very well.

Take-away today

- The importance of ranking: User studies at Google
- Length normalization: Pivot normalization
- The complete search system
- Implementation of ranking

Resources

- Chapter 6 of IIR
- Chapter 7 of IIR
- Resources at `https://www.fi.muni.cz/~sojka/PV211/` and `http://cislmu.org`, materials in MU IS and FI MU library
    - How Google tweaks its ranking function
    - Interview with Google search guru Udi Manber
    - Amit Singhal on Google ranking
    - SEO perspective: ranking factors
    - Yahoo Search BOSS: Opens up the search engine to developers. For example, you can rerank search results.
    - Compare Google and Yahoo ranking for a query.
    - How Google uses eye tracking for improving search.

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 8: Evaluation & Result Summaries
Handout version

Petr Sojka, Martin Líška, Hinrich Schütze et al.

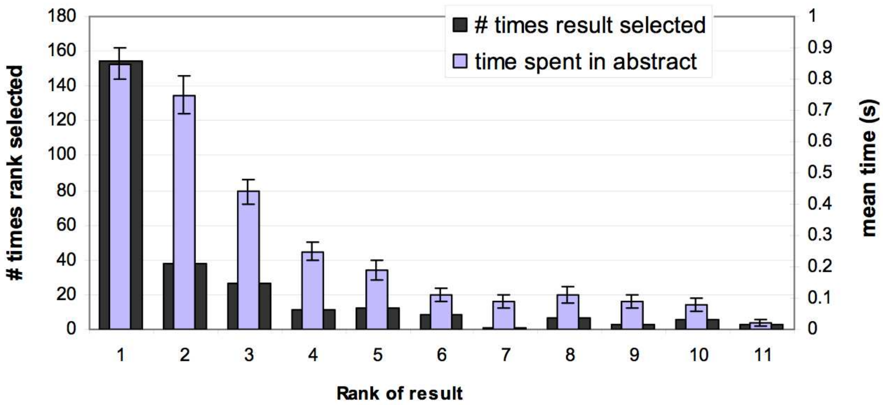
Faculty of Informatics, Masaryk University, Brno
Center for Information and Language Processing, University of Munich

2020-03-25

Overview

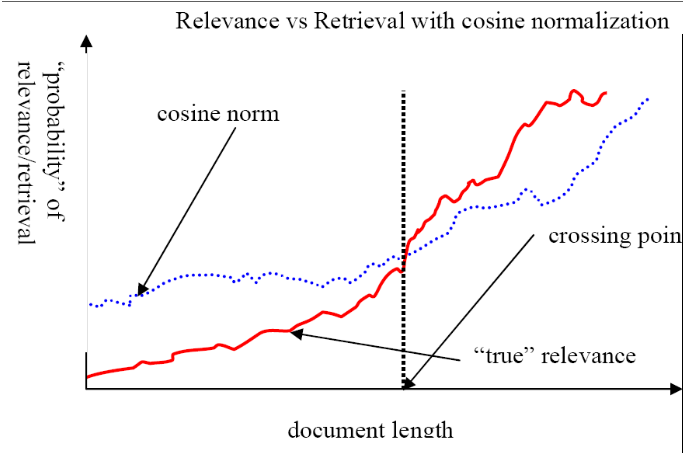
1. Recap
2. Introduction
3. Unranked evaluation
4. Ranked evaluation
5. Benchmarks
6. Result summaries

Looking vs. Clicking



- Users view results one and two more often / thoroughly
- Users click most frequently on result one

Pivot normalization



source: Lillian Lee

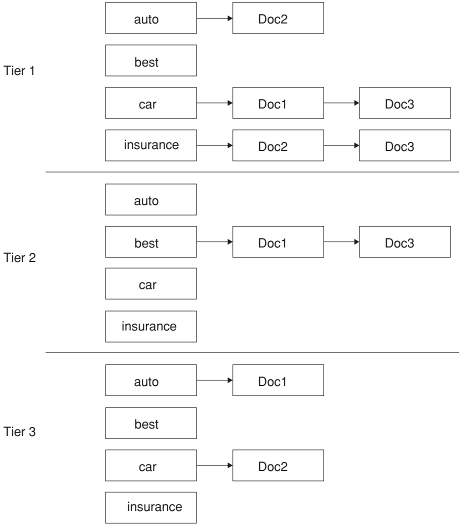
Selecting k top scoring documents in $O(N \log k)$

- Goal: Keep the k top documents seen so far
- Use a binary min heap
- To process a new document d' with score s':
    - Get current minimum h_m of heap (in $O(1)$)
    - If $s' \leq h_m$ skip to next document
    - If $s' > h_m$ heap-delete-root (in $O(\log k)$)
    - Heap-add d'/s' (in $O(1)$)
    - Reheapify (in $O(\log k)$)

Heuristics for finding the top k even faster

- Document-at-a-time processing
  - We complete computation of the query-document similarity score of document d_i before starting to compute the query-document similarity score of d_{i+1}.
  - Requires a consistent ordering of documents in the postings lists
- Term-at-a-time processing
  - We complete processing the postings list of query term t_i before starting to process the postings list of t_{i+1}.
  - Requires an accumulator for each document "still in the running"
- The most effective heuristics switch back and forth between term-at-a-time and document-at-a-time processing.

Tiered index

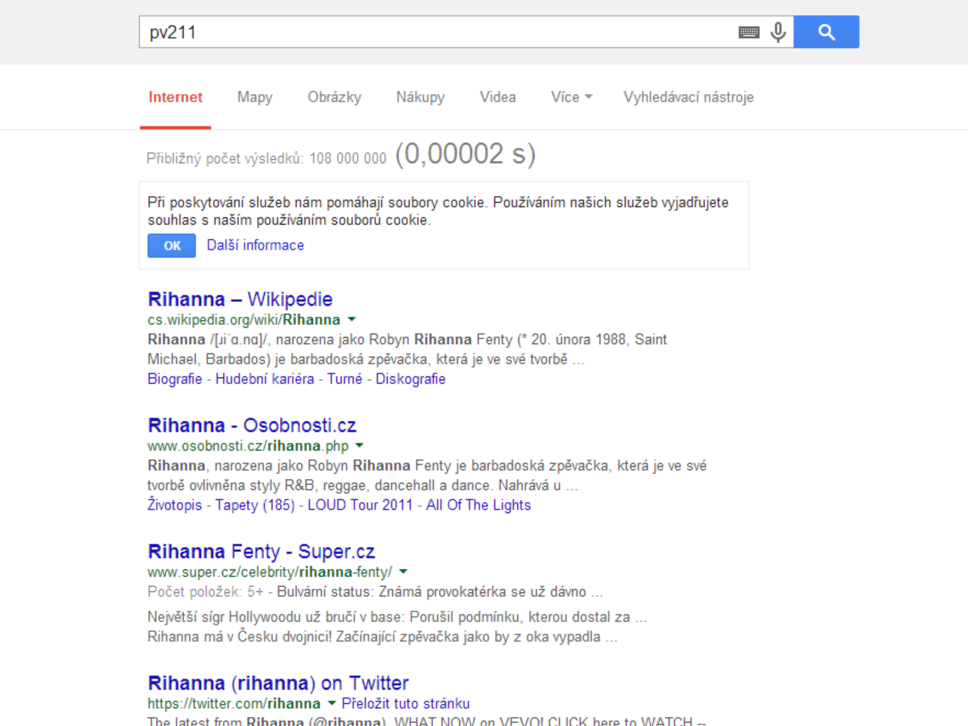


Take-away today

- Introduction to evaluation: Measures of an IR system
- Evaluation of unranked and ranked retrieval
- Evaluation benchmarks
- Result summaries

Evaluation

How well does an IR system work?



pv211

Přibližný počet výsledků: 108 000 000 (0,00002 s)

Při poskytování služeb nám pomáhají soubory cookie. Používáním našich služeb vyjadřujete souhlas s naším používáním souborů cookie.

Další informace

Rihanna – Wikipedie
cs.wikipedia.org/wiki/**Rihanna**
Rihanna /ɹiˈɑ.nɑ]/, narozena jako Robyn **Rihanna** Fenty (* 20. února 1988, Saint Michael, Barbados) je barbadoská zpěvačka, která je ve své tvorbě ...
Biografie - Hudební kariéra - Turné - Diskografie

Rihanna - Osobnosti.cz
www.osobnosti.cz/**rihanna**.php
Rihanna, narozena jako Robyn **Rihanna** Fenty je barbadoská zpěvačka, která je ve své tvorbě ovlivněna styly R&B, reggae, dancehall a dance. Nahrává u ...
Životopis - Tapety (185) - LOUD Tour 2011 - All Of The Lights

Rihanna Fenty - Super.cz
www.super.cz/celebrity/**rihanna**-fenty/
Počet položek: 5+ - Bulvární status: Známá provokatérka se už dávno ...
Největší sígr Hollywoodu už bručí v base: Porušil podmínku, kterou dostal za ...
Rihanna má v Česku dvojnici! Začínající zpěvačka jako by z oka vypadla ...

Rihanna (rihanna) on Twitter
https://twitter.com/**rihanna** Přeložit tuto stránku
The latest from **Rihanna** (@**rihanna**). WHAT NOW on VEVO! CLICK here to WATCH --

Measures for a search engine

- How fast does it index
    - e.g., number of bytes per hour
- How fast does it search
    - e.g., latency as a function of queries per second
- What is the cost per query?
    - in dollars

Measures for a search engine

- All of the preceding criteria are measurable: we can quantify speed / size / money
- However, the key measure for a search engine is user happiness.
- What is user happiness?
- Factors include:
  - Speed of response
  - Size of index
  - Uncluttered UI
  - Most important: relevance
  - (actually, maybe even more important: it's free)
- Note that none of these is sufficient: blindingly fast, but useless answers won't make a user happy.
- How can we quantify user happiness?

Who is the user?

- Who is the user we are trying to make happy?
- Web search engine: searcher. Success: Searcher finds what she was looking for. Measure: rate of return to this search engine
- Web search engine: advertiser. Success: Searcher clicks on ad. Measure: clickthrough rate
- E-commerce: buyer. Success: Buyer buys something. Measures: time to purchase, fraction of “conversions” of searchers to buyers
- E-commerce: seller. Success: Seller sells something. Measure: profit per item sold
- Enterprise: CEO. Success: Employees are more productive (because of effective search). Measure: profit of the company

Most common definition of user happiness: Relevance

- User happiness is equated with the relevance of search results to the query.
- But how do you measure relevance?
- Standard methodology in information retrieval consists of three elements.
  - A benchmark document collection
  - A benchmark suite of queries
  - An assessment of the relevance of each query-document pair

Relevance: query vs. information need

- Relevance to what?
- First take: relevance to the query
- "Relevance to the query" is very problematic.
- Information need i: "I am looking for information on whether drinking red wine is more effective at reducing your risk of heart attacks than white wine."
- This is an information need, not a query.
- Query q: [red wine white wine heart attack]
- Consider document d': *At the heart of his speech was an attack on the wine industry lobby for downplaying the role of red and white wine in drunk driving.*
- d' is an excellent match for query q . . .
- d' is not relevant to the information need i.

Relevance: query vs. information need

- User happiness can only be measured by relevance to an information need, not by relevance to queries.
- Our terminology is sloppy in these slides and in IIR: we talk about query-document relevance judgments even though we mean information-need-document relevance judgments.

Precision and recall

- Precision (P) is the fraction of retrieved documents that are relevant

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = P(\text{relevant}|\text{retrieved})$$

- Recall (R) is the fraction of relevant documents that are retrieved

$$\text{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{retrieved}|\text{relevant})$$

Precision and recall

	Relevant	Nonrelevant
Retrieved	true positives (TP)	false positives (FP)
Not retrieved	false negatives (FN)	true negatives (TN)

$$\begin{aligned} P &= TP/(TP+FP) \\ R &= TP/(TP+FN) \end{aligned}$$

Precision/recall tradeoff

- You can increase recall by returning more docs.
- Recall is a non-decreasing function of the number of docs retrieved.
- A system that returns all docs has 100% recall!
- The converse is also true (usually): It's easy to get high precision for very low recall.
- Suppose the document with the largest score is relevant. How can we maximize precision?

A combined measure: F

- F allows us to trade off precision against recall.
- $$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R} \quad \text{where} \quad \beta^2 = \frac{1-\alpha}{\alpha}$$
- $\alpha \in [0,1]$ and thus $\beta^2 \in [0,\infty]$
- Most frequently used: balanced F with $\beta = 1$ or $\alpha = 0.5$
  - This is the harmonic mean of P and R: $\frac{1}{F} = \frac{1}{2}(\frac{1}{P} + \frac{1}{R})$
- What value range of β weights recall higher than precision?

Example for precision, recall, F1

	relevant	not relevant	
retrieved	20	40	60
not retrieved	60	1,000,000	1,000,060
	80	1,000,040	1,000,120

- $P = 20/(20 + 40) = 1/3$
- $R = 20/(20 + 60) = 1/4$
- $F_1 = 2\frac{1}{\frac{1}{\frac{1}{3}} + \frac{1}{\frac{1}{4}}} = 2/7$

Accuracy

- Why do we use complex measures like precision, recall, and F?
- Why not something simple like accuracy?
- Accuracy is the fraction of decisions (relevant/nonrelevant) that are correct.
- In terms of the contingency table above, accuracy $= (TP + TN)/(TP + FP + FN + TN)$.
- Why is accuracy not a useful measure for web information retrieval?

Exercise

- Compute precision, recall and F_1 for this result set:

	relevant	not relevant
retrieved	18	2
not retrieved	82	1,000,000,000

- The snoogle search engine below always returns 0 results ("0 matching results found"), regardless of the query. Why does snoogle demonstrate that accuracy is not a useful measure in IR?

snoogle.com

Search for:

0 matching results found.

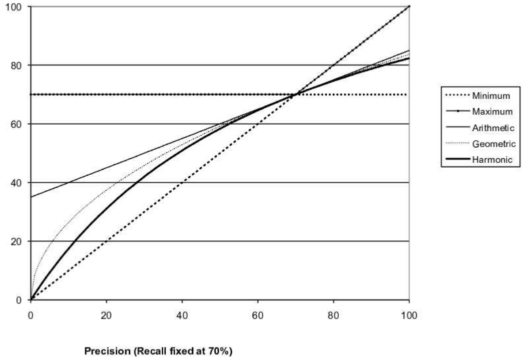
Why accuracy is a useless measure in IR

- Simple trick to maximize accuracy in IR: always say no and return nothing
- You then get 99.99% accuracy on most queries.
- Searchers on the web (and in IR in general) want to find something and have a certain tolerance for junk.
- It's better to return some bad hits as long as you return something.
- → We use precision, recall, and F for evaluation, not accuracy.

F: Why harmonic mean?

- Why don't we use a different mean of P and R as a measure?
  - e.g., the arithmetic mean
- The simple (arithmetic) mean is 50% for "return-everything" search engine, which is too high.
- Desideratum: Punish really bad performance on either precision or recall.
- Taking the minimum achieves this.
- But minimum is not smooth and hard to weight.
- F (harmonic mean) is a kind of smooth minimum.

F_1 and other averages



- We can view the harmonic mean as a kind of soft minimum

Difficulties in using precision, recall and F

- We need relevance judgments for information-need-document pairs – but they are expensive to produce.
- For alternatives to using precision/recall and having to produce relevance judgments – see end of this lecture.

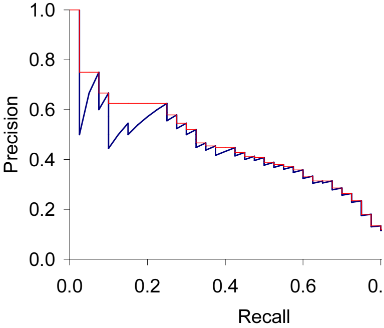
Mean Average Precision

- $\text{MAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_{jk})$
- For one query it is the area under the uninterpolated precision-recall curve,
- and so the MAP is roughly the *average* area under the precision-recall curve for a set of queries.

Precision-recall curve

- Precision/recall/F are measures for unranked sets.
- We can easily turn set measures into measures of ranked lists.
- Just compute the set measure for each "prefix": the top 1 (P@1), top 2, top 3, top 4 etc results
- Doing this for precision and recall gives you a precision-recall curve.

A precision-recall curve



- Each point corresponds to a result for the top k ranked hits ($k = 1, 2, 3, 4, \ldots$).
- Interpolation (in red): Take maximum of all future points
- Rationale for interpolation: The user is willing to look at more stuff if both precision and recall get better.
- Questions?

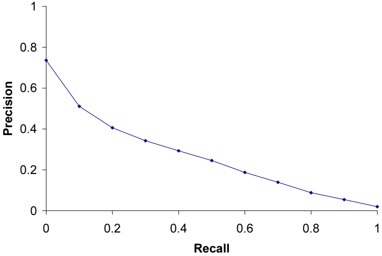
11-point interpolated average precision

Recall	Interpolated Precision
0.0	1.00
0.1	0.67
0.2	0.63
0.3	0.55
0.4	0.45
0.5	0.41
0.6	0.36
0.7	0.29
0.8	0.13
0.9	0.10
1.0	0.08

11-point average: $\approx$ 0.425

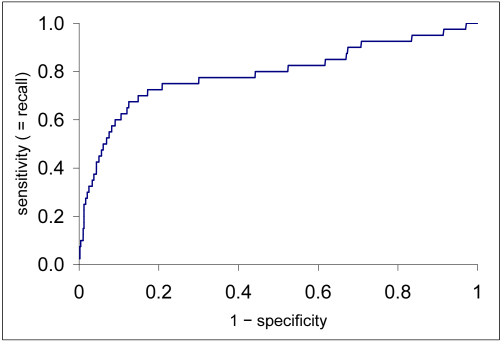
How can precision at 0.0 be > 0?

Averaged 11-point precision/recall graph



- Compute interpolated precision at recall levels 0.0, 0.1, 0.2, . . .
- Do this for each of the queries in the evaluation benchmark
- Average over queries
- This measure measures performance at all recall levels.
- The curve is typical of performance levels at TREC.
- Note that performance is not very good!

ROC curve



- Similar to precision-recall graph
- But we are only interested in the small area in the lower left corner.
- Precision-recall graph "blows up" this area.

Variance of measures like precision/recall

- For a test collection, it is usual that a system does badly on some information needs (e.g., $P = 0.2$ at $R = 0.1$) and really well on others (e.g., $P = 0.95$ at $R = 0.1$).
- Indeed, it is usually the case that the variance of the same system across queries is much greater than the variance of different systems on the same query.
- That is, there are easy information needs and hard ones.

What we need for a benchmark

- A collection of documents
  - Documents must be representative of the documents we expect to see in reality.
- A collection of information needs
  - . . . which we will often incorrectly refer to as queries
  - Information needs must be representative of the information needs we expect to see in reality.
- Human relevance assessments
  - We need to hire/pay "judges" or assessors to do this.
  - Expensive, time-consuming
  - Judges must be representative of the users we expect to see in reality.

First standard relevance benchmark: Cranfield

- Pioneering: first testbed allowing precise quantitative measures of information retrieval effectiveness
- Late 1950s, UK
- 1398 abstracts of aerodynamics journal articles, a set of 225 queries, exhaustive relevance judgments of all query-document-pairs
- Too small, too untypical for serious IR evaluation today

Second-generation relevance benchmark: TREC

- TREC = Text Retrieval Conference (TREC)
- Organized by the U.S. National Institute of Standards and Technology (NIST)
- TREC is actually a set of several different relevance benchmarks.
- Best known: TREC Ad Hoc, used for first 8 TREC evaluations between 1992 and 1999
- 1.89 million documents, mainly newswire articles, 450 information needs
- No exhaustive relevance judgments – too expensive
- Rather, NIST assessors' relevance judgments are available only for the documents that were among the top k returned for some system which was entered in the TREC evaluation for which the information need was developed.

Standard relevance benchmarks: Others

- GOV2
  - Another TREC/NIST collection
  - 25 million web pages
  - Used to be largest collection that is easily available
  - But still 3 orders of magnitude smaller than what Google/Yahoo/MSN index
- NTCIR: East Asian language and cross-language information retrieval
- CLEF: Cross Language Evaluation Forum: This evaluation series has concentrated on European languages and cross-language information retrieval.
- Many others

Example of more recent benchmark: ClueWeb datasets

Clueweb09:

- 1 billion web pages, 25 terabytes (compressed: 5 terabyte) collected during January/February 2009
- crawl of pages in 10 languages
- Unique URLs: 4,780,950,903 (325 GB uncompressed, 105 GB compressed)
- Total Outlinks: 7,944,351,835 (71 GB uncompressed, 24 GB compressed)

Clueweb12:

- 733,019,372 docs, 27.3 TB (5.54 TB compressed)

Indexed in Sketch Engine, cf. LREC 2012 paper.

Validity of relevance assessments

- Relevance assessments are only usable if they are consistent.
- If they are not consistent, then there is no "truth" and experiments are not repeatable.
- How can we measure this consistency or agreement among judges?
- → Kappa measure

Kappa measure

- Kappa is measure of how much judges agree or disagree.
- Designed for categorical judgments
- Corrects for chance agreement
- $P(A) =$ proportion of time judges agree
- $P(E) =$ what agreement would we get by chance
- $$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$
- $\kappa =$? for (i) chance agreement (ii) total agreement

Kappa measure (2)

- Values of κ in the interval $[2/3, 1.0]$ are seen as acceptable.
- With smaller values: need to redesign relevance assessment methodology used etc.

Calculating the kappa statistic

		Judge 2 Relevance		
		Yes	No	Total
Judge 1	Yes	300	20	320
Relevance	No	10	70	80
	Total	310	90	400

Observed proportion of the times the judges agreed
$P(A) = (300 + 70)/400 = 370/400 = 0.925$
Pooled marginals
$P(nonrelevant) = (80 + 90)/(400 + 400) = 170/800 = 0.2125$
$P(relevant) = (320 + 310)/(400 + 400) = 630/800 = 0.7878$
Probability that the two judges agreed by chance $P(E) = P(nonrelevant)^2 + P(relevant)^2 = 0.2125^2 + 0.7878^2 = 0.665$
Kappa statistic $\kappa = (P(A) - P(E))/(1 - P(E)) = (0.925 - 0.665)/(1 - 0.665) = 0.776$ (still in acceptable range)

Interjudge agreement at TREC

information need	number of docs judged	disagreements
51	211	6
62	400	157
67	400	68
95	400	110
127	400	106

Impact of interjudge disagreement

- Judges disagree a lot. Does that mean that the results of information retrieval experiments are meaningless?
- No.
- Large impact on absolute performance numbers
- Virtually no impact on ranking of systems
- Supposes we want to know if algorithm A is better than algorithm B.
- An information retrieval experiment will give us a reliable answer to this question. . .
- . . . even if there is a lot of disagreement between judges.

Evaluation at large search engines

- Recall is difficult to measure on the web
- Search engines often use precision at top k, e.g., $k = 10 \ldots$
- . . . or use measures that reward you more for getting rank 1 right than for getting rank 10 right.
- Search engines also use non-relevance-based measures.
    - Example 1: clickthrough on first result
    - Not very reliable if you look at a single clickthrough (you may realize after clicking that the summary was misleading and the document is nonrelevant). . .
    - . . . but pretty reliable in the aggregate.
    - Example 2: Ongoing studies of user behavior in the lab – recall last lecture
    - Example 3: A/B testing

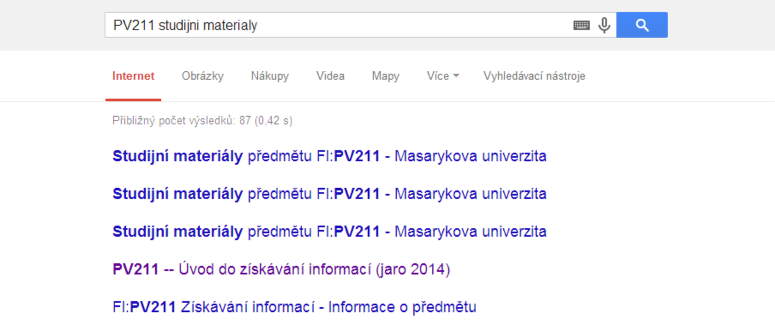
A/B testing

- Purpose: Test a single innovation
- Prerequisite: You have a large search engine up and running.
- Have most users use old system
- Divert a small proportion of traffic (e.g., 1%) to the new system that includes the innovation
- Evaluate with an "automatic" measure like clickthrough on first result
- Now we can directly see if the innovation does improve user happiness.
- Probably the evaluation methodology that large search engines trust most
- Variant: Give users the option to switch to new algorithm/interface

Critique of pure relevance

- We've defined relevance for an isolated query-document pair.
- Alternative definition: marginal relevance
- The marginal relevance of a document at position k in the result list is the additional information it contributes over and above the information that was contained in documents $d_1 \ldots d_{k-1}$.
- Exercise
  - Why is marginal relevance a more realistic measure of user happiness?
  - Give an example where a non-marginal measure like precision or recall is a misleading measure of user happiness, but marginal relevance is a good measure.
  - In a practical application, what is the difficulty of using marginal measures instead of non-marginal measures?

How do we present results to the user?



PV211 studijni materialy

Přibližný počet výsledků: 87 (0,42 s)

Studijní materiály předmětu FI:**PV211** - Masarykova univerzita

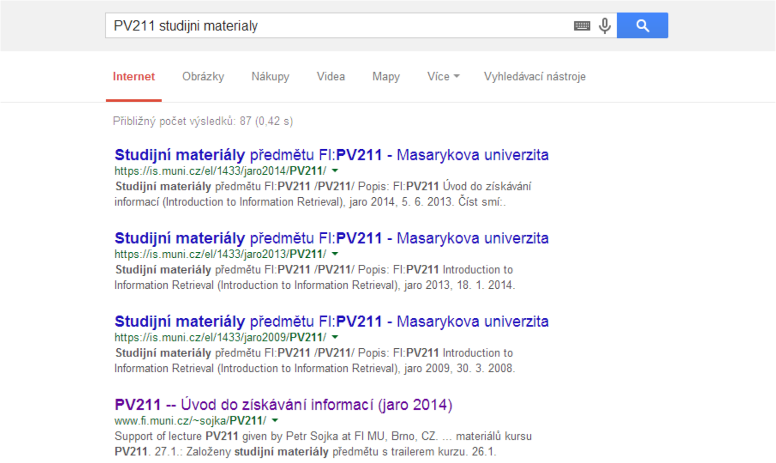
Studijní materiály předmětu FI:**PV211** - Masarykova univerzita

Studijní materiály předmětu FI:**PV211** - Masarykova univerzita

PV211 -- Úvod do získávání informací (jaro 2014)

FI:**PV211** Získávání informací - Informace o předmětu

How do we present results to the user?



PV211 studijni materialy

Přibližný počet výsledků: 87 (0,42 s)

Studijní materiály předmětu FI:**PV211** - Masarykova univerzita
https://is.muni.cz/el/1433/jaro2014/**PV211**/
Studijní materiály předmětu FI:**PV211** /**PV211**/ Popis: FI:**PV211** Úvod do získávání informací (Introduction to Information Retrieval), jaro 2014, 5. 6. 2013. Číst smí:.

Studijní materiály předmětu FI:**PV211** - Masarykova univerzita
https://is.muni.cz/el/1433/jaro2013/**PV211**/
Studijní materiály předmětu FI:**PV211** /**PV211**/ Popis: FI:**PV211** Introduction to Information Retrieval (Introduction to Information Retrieval), jaro 2013, 18. 1. 2014.

Studijní materiály předmětu FI:**PV211** - Masarykova univerzita
https://is.muni.cz/el/1433/jaro2009/**PV211**/
Studijní materiály předmětu FI:**PV211** /**PV211**/ Popis: FI:**PV211** Introduction to Information Retrieval (Introduction to Information Retrieval), jaro 2009, 30. 3. 2008.

PV211 -- Úvod do získávání informací (jaro 2014)
www.fi.muni.cz/~sojka/**PV211**/
Support of lecture **PV211** given by Petr Sojka at FI MU, Brno, CZ. ... materiálů kursu **PV211**. 27.1.: Založeny **studijní materiály** předmětu s trailerem kurzu. 26.1.

How do we present results to the user?

- Most often: as a list – aka “10 blue links”
- How should each document in the list be described?
- This description is crucial.
- The user often can identify good hits (= relevant hits) based on the description.
- No need to actually view any document

Doc description in result list

- Most commonly: doc title, url, some metadata . . .
- . . . and a summary
- How do we “compute” the summary?

Summaries

- Two basic kinds: (i) static (ii) dynamic
- A static summary of a document is always the same, regardless of the query that was issued by the user.
- Dynamic summaries are query-dependent. They attempt to explain why the document was retrieved for the query at hand.

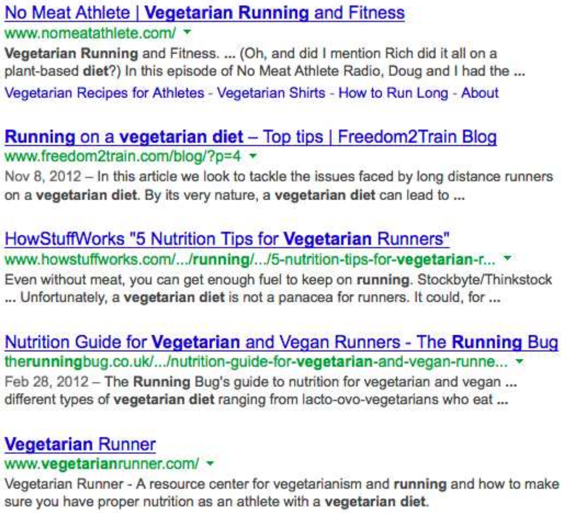
Static summaries

- In typical systems, the static summary is a subset of the document.
- Simplest heuristic: the first 50 or so words of the document
- More sophisticated: extract from each document a set of “key” sentences
  - Simple NLP heuristics to score each sentence
  - Summary is made up of top-scoring sentences.
  - Machine learning approach: see IIR 13
- Most sophisticated: complex NLP to synthesize/generate a summary
  - For most IR applications: not quite ready for prime time yet

Dynamic summaries

- Present one or more “windows” or snippets within the document that contain several of the query terms.
- Prefer snippets in which query terms occurred as a phrase
- Prefer snippets in which query terms occurred jointly in a small window
- The summary that is computed this way gives the entire content of the window – all terms, not just the query terms.

Google dynamic summaries for [vegetarian diet running]



No Meat Athlete | **Vegetarian Running** and Fitness
www.nomeatathlete.com/
Vegetarian Running and Fitness. ... (Oh, and did I mention Rich did it all on a plant-based **diet**?) In this episode of No Meat Athlete Radio, Doug and I had the ...
Vegetarian Recipes for Athletes - Vegetarian Shirts - How to Run Long - About

Running on a **vegetarian diet** – Top tips | Freedom2Train Blog
www.freedom2train.com/blog/?p=4
Nov 8, 2012 – In this article we look to tackle the issues faced by long distance runners on a **vegetarian diet**. By its very nature, a **vegetarian diet** can lead to ...

HowStuffWorks "5 Nutrition Tips for **Vegetarian** Runners"
www.howstuffworks.com/.../**running**/.../5-nutrition-tips-for-**vegetarian**-r...
Even without meat, you can get enough fuel to keep on **running**. Stockbyte/Thinkstock ... Unfortunately, a **vegetarian diet** is not a panacea for runners. It could, for ...

Nutrition Guide for **Vegetarian** and Vegan Runners - The **Running** Bug
the**running**bug.co.uk/.../nutrition-guide-for-**vegetarian**-and-vegan-runne...
Feb 28, 2012 – The **Running** Bug's guide to nutrition for vegetarian and vegan ... different types of **vegetarian diet** ranging from lacto-ovo-vegetarians who eat ...

Vegetarian Runner
www.**vegetarian**runner.com/
Vegetarian Runner - A resource center for vegetarianism and **running** and how to make sure you have proper nutrition as an athlete with a **vegetarian diet**.

- Good example that snippet selection is non-trivial.
- Criteria: occurrence of keywords, density of keywords, coherence of snippet, number of different snippets in summary, good cutting points etc

A dynamic summary

Query: [new guinea economic development]

Snippets (in bold) that were extracted from a document: . . . ***In recent years, Papua New Guinea has faced severe economic difficulties and*** economic growth has slowed, partly as a result of weak governance and civil war, and partly as a result of external factors such as the Bougainville civil war which led to the closure in 1989 of the Panguna mine (at that time the most important foreign exchange earner and contributor to Government finances), the Asian financial crisis, a decline in the prices of gold and copper, and a fall in the production of oil. ***PNG's economic development record over the past few years is evidence that*** governance issues underly many of the country's problems. Good governance, which may be defined as the transparent and accountable management of human, natural, economic and financial resources for the purposes of equitable and sustainable development, flows from proper public sector management, efficient fiscal and accounting mechanisms, and a willingness to make service delivery a priority in practice. . . .

Generating dynamic summaries

- Where do we get these other terms in the snippet from?
- We cannot construct a dynamic summary from the positional inverted index – at least not efficiently.
- We need to cache documents.
- The positional index tells us: query term occurs at position 4378 in the document.
- Byte offset or word offset?
- Note that the cached copy can be outdated
- Don't cache very long documents – just cache a short prefix

Dynamic summaries

- Real estate on the search result page is limited → snippets must be short . . .
- . . . but snippets must be long enough to be meaningful.
- Snippets should communicate whether and how the document answers the query.
- Ideally: linguistically well-formed snippets
- Ideally: the snippet should answer the query, so we don't have to look at the document.
- Dynamic summaries are a big part of user happiness because . . .
    - . . . we can quickly scan them to find the relevant document we then click on.
    - . . . in many cases, we don't have to click at all and save time.

Resources

- Chapter 8 of IIR
- Resources at `https://www.fi.muni.cz/~sojka/PV211/` and `http://cislmu.org`, materials in MU IS and FI MU library
    - The TREC home page – TREC had a huge impact on information retrieval evaluation.
    - Originator of *F*-measure: Keith van Rijsbergen
    - More on A/B testing
    - Too much A/B testing at Google?
    - Tombros & Sanderson 1998: one of the first papers on dynamic summaries
    - Google VP of Engineering on search quality evaluation at Google
    - ClueWeb12 or other datasets available in Sketch Engine

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 9: Relevance feedback & Query expansion
Handout version

Petr Sojka, Hinrich Schütze et al.

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

2020-03-25

Overview

1. Motivation
2. Relevance feedback: Basics
3. Relevance feedback: Details
4. Query expansion

Take-away today

- Interactive relevance feedback: improve initial retrieval results by telling the IR system which docs are relevant / non-relevant
- Best known relevance feedback method: Rocchio feedback
- Query expansion: improve retrieval results by adding synonyms / related terms to the query
    - Sources for related terms: Manual thesauri, automatic thesauri, query logs

How can we improve recall in search?

- Main topic today: two ways of improving recall: relevance feedback and query expansion
- As an example consider query *q*: [aircraft] . . .
- . . . and document *d* containing "plane", but not containing "aircraft"
- A simple IR system will not return *d* for *q*.
- Even if *d* is the most relevant document for *q*!
- We want to change this:
  - Return relevant documents even if there is no term match with the (original) query

Recall

- Loose definition of recall in this lecture: "increasing the number of relevant documents returned to user"
- This may actually decrease recall on some measures, e.g., when expanding "jaguar" with "panthera"
  - . . . which eliminates some relevant documents, but increases relevant documents returned on top pages

Options for improving recall

- Local: Do a "local", on-demand analysis for a user query
  - Main local method: relevance feedback
  - Part 1
- Global: Do a global analysis once (e.g., of collection) to produce thesaurus
  - Use thesaurus for query expansion
  - Part 2

Google examples for query expansion

- One that works well
  - *˜flights -flight*
- One that doesn't work so well
  - *˜dogs -dog*

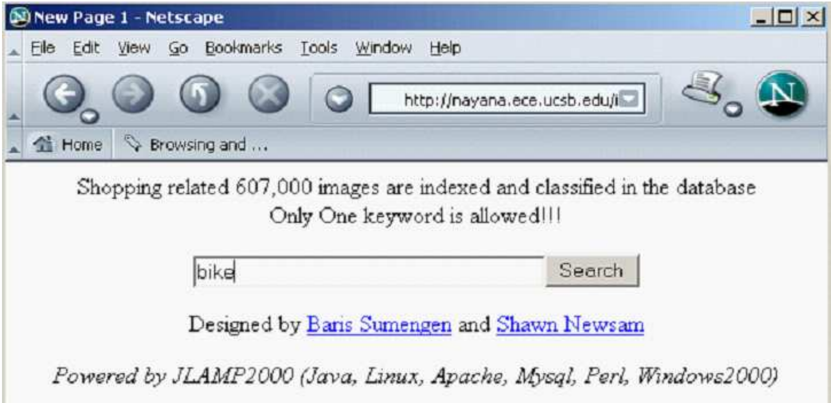
Relevance feedback: Basic idea

- The user issues a (short, simple) query.
- The search engine returns a set of documents.
- User marks some docs as relevant, some as non-relevant.
- Search engine computes a new representation of the information need. Hope: better than the initial query.
- Search engine runs new query and returns new results.
- New results have (hopefully) better recall.
- We can iterate this: several rounds of relevance feedback.
- We will use the term ad hoc retrieval to refer to regular retrieval without relevance feedback.

Relevance feedback: Examples

- We will now look at three different examples of relevance feedback that highlight different aspects of the process.

Relevance Feedback: Example 1



New Page 1 - Netscape
File Edit View Go Bookmarks Tools Window Help
http://nayana.ece.ucsb.edu/i
Home
Browsing and ...

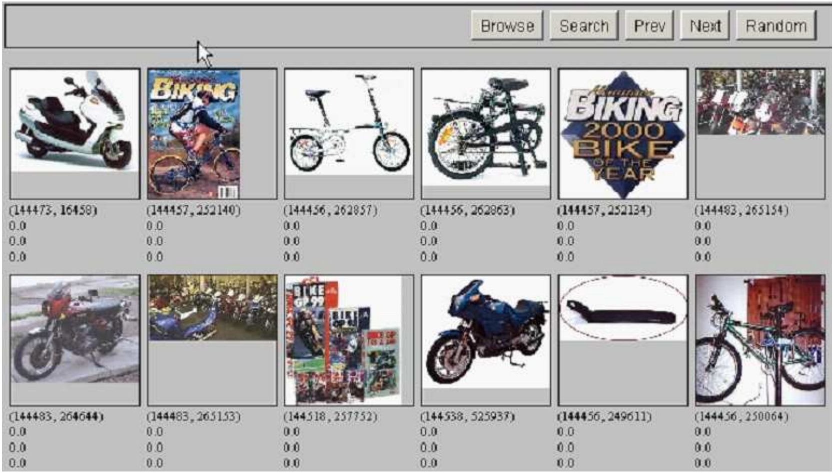
Shopping related 607,000 images are indexed and classified in the database
Only One keyword is allowed!!!

bike Search

Designed by Baris Sumengen and Shawn Newsam

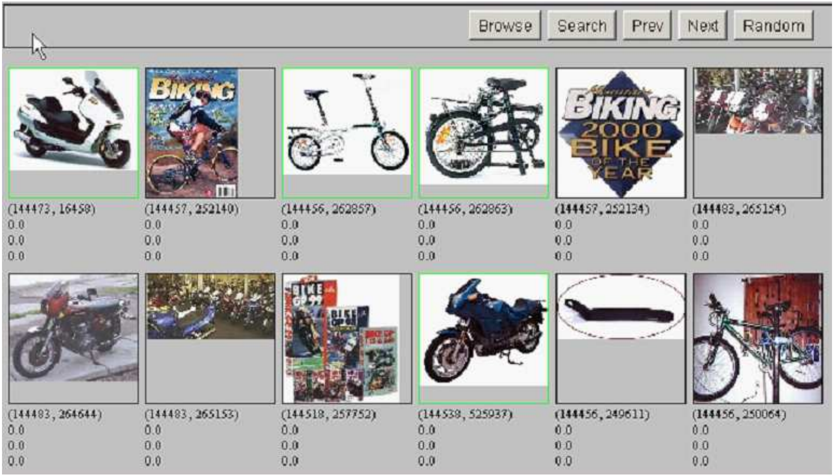
Powered by JLAMP2000 (Java, Linux, Apache, Mysql, Perl, Windows2000)

Results for initial query



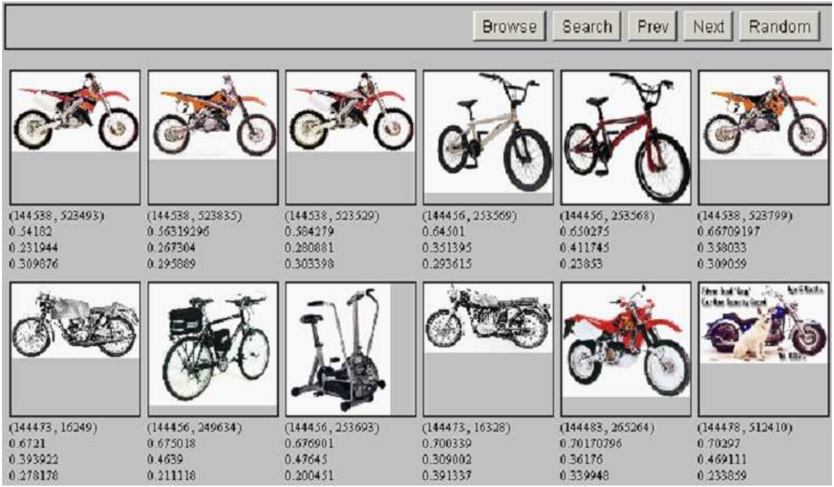
```
(144473, 16458)  (144457, 252140)  (144456, 262857)  (144456, 262863)  (144457, 252134)  (144483, 265154)
0.0              0.0               0.0               0.0               0.0               0.0
0.0              0.0               0.0               0.0               0.0               0.0
0.0              0.0               0.0               0.0               0.0               0.0

(144483, 264644) (144483, 265153)  (144518, 257752)  (144538, 525937)  (144456, 249611)  (144456, 250064)
0.0              0.0               0.0               0.0               0.0               0.0
0.0              0.0               0.0               0.0               0.0               0.0
0.0              0.0               0.0               0.0               0.0               0.0
```

User feedback: Select what is relevant



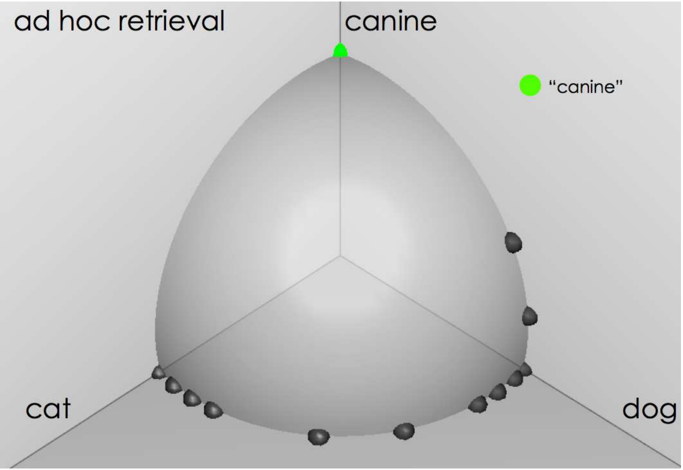
(144473, 16458)	(144457, 252140)	(144456, 262857)	(144456, 262863)	(144457, 252134)	(144483, 265154)
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
(144483, 264644)	(144483, 265153)	(144518, 257752)	(144538, 525937)	(144456, 249611)	(144456, 250064)
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0

Results after relevance feedback



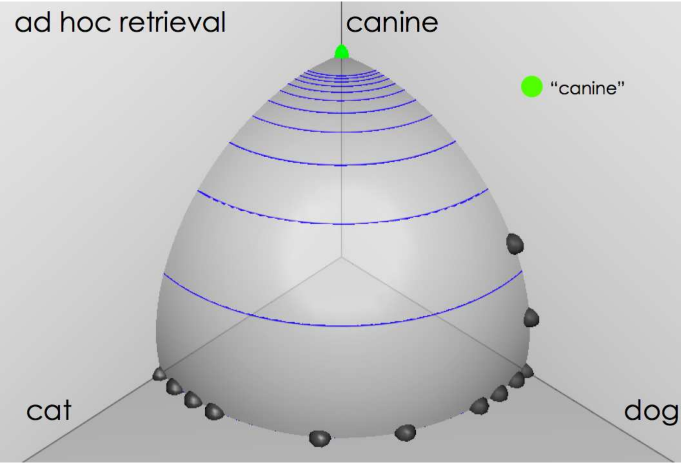
```
(144538, 523493)  (144538, 523835)  (144538, 523529)  (144456, 253569)  (144456, 253568)  (144538, 523799)
0.54182           0.56319296        0.584279          0.64501           0.650275          0.66709197
0.231944          0.267304          0.280881          0.351395          0.411745          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
```

Vector space example: query "canine" (1)



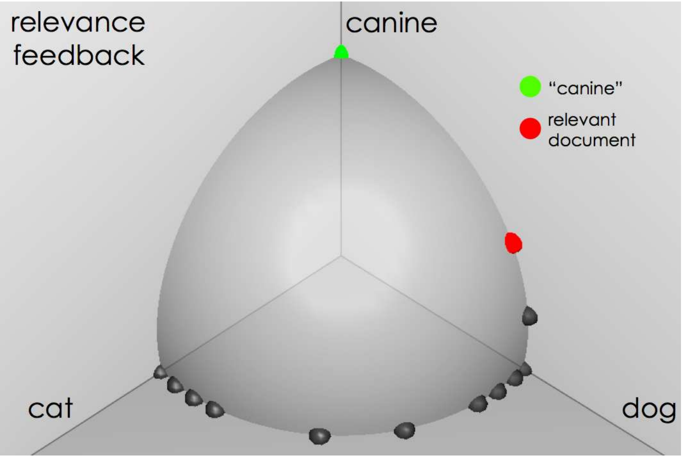
source:
Fernando Díaz

Similarity of docs to query “canine”



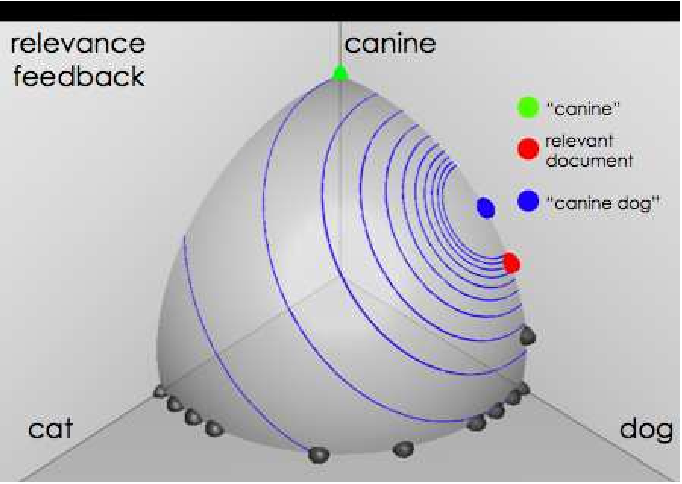
source:
Fernando Díaz

User feedback: Select relevant documents



source:
Fernando Díaz

Results after relevance feedback



source:
Fernando Díaz

Example 3: A real (non-image) example

Initial query: [new space satellite applications]

Results for initial query: (r = rank)

	r		
+	1	0.539	NASA Hasn't Scrapped Imaging Spectrometer
+	2	0.533	NASA Scratches Environment Gear From Satellite Plan
	3	0.528	Science Panel Backs NASA Satellite Plan, But Urges Launches of Smaller Probes
	4	0.526	A NASA Satellite Project Accomplishes Incredible Feat: Staying Within Budget
	5	0.525	Scientist Who Exposed Global Warming Proposes Satellites for Climate Research
	6	0.524	Report Provides Support for the Critics Of Using Big Satellites to Study Climate
	7	0.516	Arianespace Receives Satellite Launch Pact From Telesat Canada
+	8	0.509	Telecommunications Tale of Two Companies

User then marks relevant documents with "+".

Expanded query after relevance feedback

2.074	new	15.106	space
30.816	satellite	5.660	application
5.991	nasa	5.196	eos
4.196	launch	3.972	aster
3.516	instrument	3.446	arianespace
3.004	bundespost	2.806	ss
2.790	rocket	2.053	scientist
2.003	broadcast	1.172	earth
0.836	oil	0.646	measure

Compare to original query: [new space satellite applications]

Results for expanded query (old ranks in parens)

	r		
*	1 (2)	0.513	NASA Scratches Environment Gear From Satellite Plan
*	2 (1)	0.500	NASA Hasn't Scrapped Imaging Spectrometer
	3	0.493	When the Pentagon Launches a Secret Satellite, Space Sleuths Do Some Spy Work of Their Own
	4	0.493	NASA Uses 'Warm' Superconductors For Fast Circuit
*	5 (8)	0.492	Telecommunications Tale of Two Companies
	6	0.491	Soviets May Adapt Parts of SS-20 Missile For Commercial Use
	7	0.490	Gaping Gap: Pentagon Lags in Race To Match the Soviets In Rocket Launchers
	8	0.490	Rescue of Satellite By Space Agency To Cost $90 Million

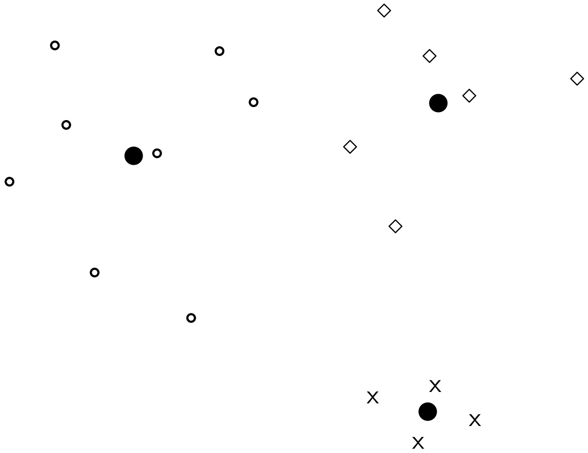
Key concept for relevance feedback: Centroid

- The centroid is the center of mass of a set of points.
- Recall that we represent documents as points in a high-dimensional space.
- Thus: we can compute centroids of documents.
- Definition:

$$\vec{\mu}(D) = \frac{1}{|D|} \sum_{d \in D} \vec{v}(d)$$

where D is a set of documents and $\vec{v}(d) = \vec{d}$ is the vector we use to represent document d.

Centroid: Examples



Rocchio algorithm

- The Rocchio algorithm implements relevance feedback in the vector space model.
- Rocchio chooses the query $\vec{q}_{opt}$ that maximizes

$$\vec{q}_{opt} \quad = \quad \arg\max_{\vec{q}} [\text{sim}(\vec{q}, \mu(D_r)) - \text{sim}(\vec{q}, \mu(D_{nr}))]$$

  D_r: set of relevant docs; D_{nr}: set of nonrelevant docs

- Intent: $\vec{q}_{opt}$ is the vector that separates relevant and non-relevant docs maximally.
- Making some additional assumptions, we can rewrite $\vec{q}_{opt}$ as:

$$\vec{q}_{opt} \quad = \quad \mu(D_r) + [\mu(D_r) - \mu(D_{nr})]$$

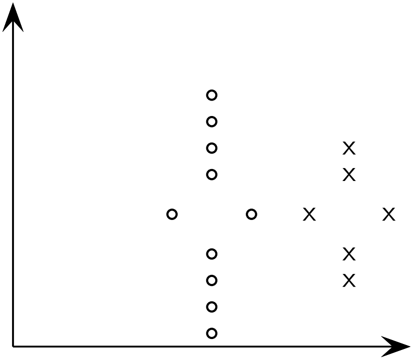
Rocchio algorithm

- The optimal query vector is:

$$\begin{aligned}\vec{q}_{opt} &= \mu(D_r) + [\mu(D_r) - \mu(D_{nr})] \\ &= \frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r} \vec{d}_j + [\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r} \vec{d}_j - \frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}} \vec{d}_j]\end{aligned}$$

- We move the centroid of the relevant documents by the difference between the two centroids.

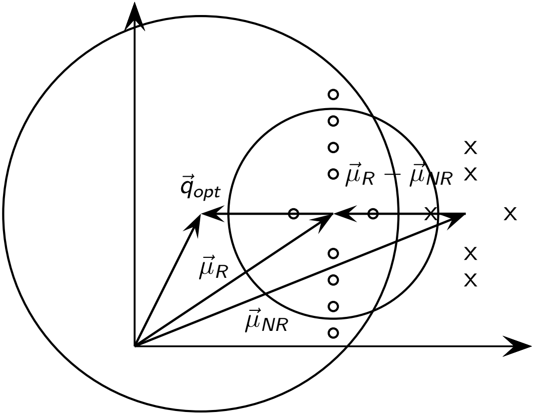
Exercise: Compute Rocchio vector



circles: relevant documents, Xs: nonrelevant documents

compute: $\vec{q}_{opt} = \mu(D_r) + [\mu(D_r) - \mu(D_{nr})]$

Rocchio illustrated



circles: relevant documents, Xs: non-relevant documents $\vec{\mu}_R$: centroid of relevant documents $\vec{\mu}_R$ does not separate relevant/non-relevant. $\vec{\mu}_{NR}$: centroid of non-relevant documents $\vec{\mu}_R - \vec{\mu}_{NR}$: difference vector Add difference vector to $\vec{\mu}_R$ . . . . . . to get $\vec{q}_{opt}$ $\vec{q}_{opt}$ separates relevant/non-relevant perfectly.

Terminology

- So far, we have used the name Rocchio for the theoretically better motivated original version of Rocchio.
- The implementation that is actually used in most cases is the SMART implementation – this SMART version of Rocchio is what we will refer to from now on.

Rocchio 1971 algorithm (SMART)

- Used in practice:

$$\begin{aligned}\vec{q}_m &= \alpha\vec{q}_0 + \beta\mu(D_r) - \gamma\mu(D_{nr}) \\ &= \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}}\vec{d}_j\end{aligned}$$

  q_m: modified query vector; q_0: original query vector; D_r and D_{nr}: sets of known relevant and non-relevant documents respectively; α, β, and γ: weights
- New query moves towards relevant documents and away from non-relevant documents.
- Tradeoff α vs. β/γ: If we have a lot of judged documents, we want a higher β/γ.
- Set negative term weights to 0.
- “Negative weight” for a term doesn’t make sense in the vector space model.

Positive vs. negative relevance feedback

- Positive feedback is more valuable than negative feedback.
- For example, set $\beta = 0.75$, $\gamma = 0.25$ to give higher weight to positive feedback.
- Many systems only allow positive feedback.

Relevance feedback: Assumptions

- When can relevance feedback enhance recall?
- Assumption A1: The user knows the terms in the collection well enough for an initial query.
- Assumption A2: Relevant documents contain similar terms (so I can "hop" from one relevant document to a different one when giving relevance feedback).

Violation of A1

- Assumption A1: The user knows the terms in the collection well enough for an initial query.
- Violation: Mismatch of searcher's vocabulary and collection vocabulary
- Example: cosmonaut / astronaut

Violation of A2

- Assumption A2: Relevant documents are similar.
- Example for violation: [contradictory government policies]
- Several unrelated "prototypes"
  - Subsidies for tobacco farmers vs. anti-smoking campaigns
  - Aid for developing countries vs. high tariffs on imports from developing countries
- Relevance feedback on tobacco docs will not help with finding docs on developing countries.

Relevance feedback: Assumptions

- When can relevance feedback enhance recall?
- Assumption A1: The user knows the terms in the collection well enough for an initial query.
- Assumption A2: Relevant documents contain similar terms (so I can "hop" from one relevant document to a different one when giving relevance feedback).

Relevance feedback: Evaluation

- Pick an evaluation measure, e.g., precision in top 10: $P@10$
- Compute $P@10$ for original query q_0
- Compute $P@10$ for modified relevance feedback query q_1
- In most cases: q_1 is spectacularly better than q_0!
- Is this a fair evaluation?

Relevance feedback: Evaluation

- Fair evaluation must be on “residual” collection: docs not yet judged by user.
- Studies have shown that relevance feedback is successful when evaluated this way.
- Empirically, one round of relevance feedback is often very useful. Two rounds are marginally useful.

Evaluation: Caveat

- True evaluation of usefulness must compare to other methods taking the same amount of time.
- Alternative to relevance feedback: User revises and resubmits query.
- Users may prefer revision/resubmission to having to judge relevance of documents.
- There is no clear evidence that relevance feedback is the "best use" of the user's time.

Exercise

- Do search engines use relevance feedback?
- Why?

Relevance feedback: Problems

- Relevance feedback is expensive.
  - Relevance feedback creates long modified queries.
  - Long queries are expensive to process.
- Users are reluctant to provide explicit feedback.
- It's often hard to understand why a particular document was retrieved after applying relevance feedback.
- The search engine Excite had full relevance feedback at one point, but abandoned it later.

Pseudo-relevance feedback

- Pseudo-relevance feedback automates the “manual” part of true relevance feedback.
- Pseudo-relevance feedback algorithm:
  - Retrieve a ranked list of hits for the user’s query
  - Assume that the top *k* documents are relevant.
  - Do relevance feedback (e.g., Rocchio)
- Works very well on average
- But can go horribly wrong for some queries.
  - Because of query drift
  - If you do several iterations of pseudo-relevance feedback, then you will get query drift for a large proportion of queries.

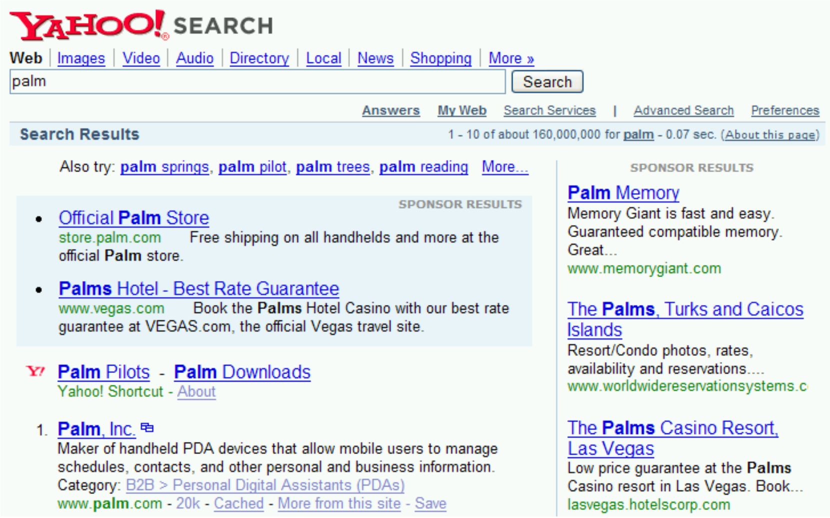
Pseudo-relevance feedback at TREC4

- Cornell SMART system
- Results show number of relevant documents out of top 100 for 50 queries (so total number of documents is 5000):

method	number of relevant documents
lnc.ltc	3210
lnc.ltc-PsRF	3634
Lnu.ltu	3709
Lnu.ltu-PsRF	4350

- Results contrast two length normalization schemes (L vs. l) and pseudo-relevance feedback (PsRF).
- The pseudo-relevance feedback method used added only 20 terms to the query. (Rocchio will add many more.)
- This demonstrates that pseudo-relevance feedback is effective on average.

Query expansion: Example



YAHOO! SEARCH
Web | Images | Video | Audio | Directory | Local | News | Shopping | More »
palm
Search
Answers My Web Search Services | Advanced Search Preferences

Search Results 1 - 10 of about 160,000,000 for palm - 0.07 sec. (About this page)

Also try: **palm** springs, **palm** pilot, **palm** trees, **palm** reading More...

SPONSOR RESULTS

- Official **Palm** Store
  store.palm.com Free shipping on all handhelds and more at the official **Palm** store.
- **Palms** Hotel - Best Rate Guarantee
  www.vegas.com Book the **Palms** Hotel Casino with our best rate guarantee at VEGAS.com, the official Vegas travel site.

Palm Pilots - **Palm** Downloads
Yahoo! Shortcut - About

1. **Palm**, Inc.
   Maker of handheld PDA devices that allow mobile users to manage schedules, contacts, and other personal and business information.
   Category: B2B > Personal Digital Assistants (PDAs)
   www.**palm**.com - 20k - Cached - More from this site - Save

SPONSOR RESULTS

Palm Memory
Memory Giant is fast and easy. Guaranteed compatible memory. Great...
www.memorygiant.com

The **Palms**, Turks and Caicos Islands
Resort/Condo photos, rates, availability and reservations....
www.worldwidereservationsystems.c

The **Palms** Casino Resort, Las Vegas
Low price guarantee at the **Palms** Casino resort in Las Vegas. Book...
lasvegas.hotelscorp.com

Types of user feedback

- User gives feedback on documents.
  - More common in relevance feedback
- User gives feedback on words or phrases.
  - More common in query expansion

Query expansion

- Query expansion is another method for increasing recall.
- We use "global query expansion" to refer to "global methods for query reformulation".
- In global query expansion, the query is modified based on some global resource, i.e. a resource that is not query-dependent.
- Main information we use: (near-)synonymy

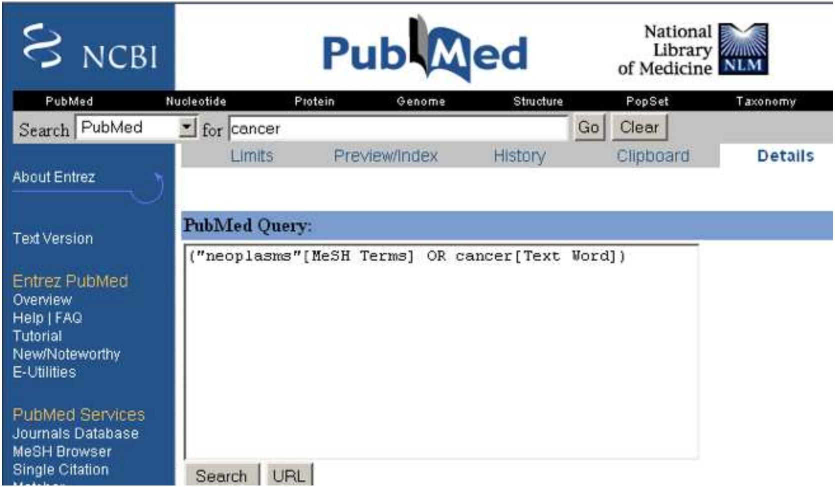
“Global” resources used for query expansion

- A publication or database that collects (near-)synonyms is called a thesaurus.
- Manual thesaurus (maintained by editors, e.g., PubMed)
- Automatically derived thesaurus (e.g., based on co-occurrence statistics)
- Query-equivalence based on query log mining (common on the web as in the “palm” example)

Thesaurus-based query expansion

- For each term t in the query, expand the query with words the thesaurus lists as semantically related with t.
- Example from earlier: HOSPITAL → MEDICAL
- Generally increases recall
- May significantly decrease precision, particularly with ambiguous terms
  - INTEREST RATE → INTEREST RATE FASCINATE
- Widely used in specialized search engines for science and engineering
- It's very expensive to create a manual thesaurus and to maintain it over time.

Example for manual thesaurus: PubMed



Search PubMed for cancer

PubMed Query:

```
("neoplasms"[MeSH Terms] OR cancer[Text Word])
```

Automatic thesaurus generation

- Attempt to generate a thesaurus automatically by analyzing the distribution of words in documents
- Fundamental notion: similarity between two words
- Definition 1: Two words are similar if they co-occur with similar words.
  - "car" ≈ "motorcycle" because both occur with "road", "gas" and "license", so they must be similar.
- Definition 2: Two words are similar if they occur in a given grammatical relation with the same words.
  - You can harvest, peel, eat, prepare, etc. apples and pears, so apples and pears must be similar.
- Co-occurrence is more robust, grammatical relations are more accurate.

Co-occurrence-based thesaurus: Examples

Word	Nearest neighbors
absolutely	absurd whatsoever totally exactly nothing
bottomed	dip copper drops topped slide trimmed
captivating	shimmer stunningly superbly plucky witty
doghouse	dog porch crawling beside downstairs
makeup	repellent lotion glossy sunscreen skin gel
mediating	reconciliation negotiate case conciliation
keeping	hoping bring wiping could some would
lithographs	drawings Picasso Dali sculptures Gauguin
pathogens	toxins bacteria organisms bacterial parasite
senses	grasp psyche truly clumsy naive innate

WordSpace demo on web

Soft cosine measure

- Use a matrix **S** that specifies the cosine similarity of basis vectors (i.e. of words) in Salton's vector space model.
- Definition 3: The similarity of two words is proportional to their cosine similarity.
  - "car" ≈ "motorcycle" iff cos("car", "motorcycle") ≈ 1.
- When the search engine supports non-orthogonal vector space model, then we can directly compute the soft cosine measure (SCM) between document vectors $\vec{u}$ and $\vec{v}$ by computing the matrix product $\vec{u}^{\mathsf{T}}\mathbf{S}\vec{v}$.
- Otherwise, we can expand the text query as follows:
  1. Translate the text query to a query vector $\vec{u}$.
  2. Compute $\vec{u'} = \vec{u}\mathbf{S}$.
  3. Translate $\vec{u'}$ back to a (now expanded) text query.
- Unlike a thesaurus based on word co-occurrences, the matrix **S** can be derived from word embeddings, the Levenshtein distance, and other measures of word similarity / relatedness.

SCM query expansion: Example

Query expansion using a Gramm matrix **S** that was built from the Google News word embeddings distributed with Word2Vec:

Original query : “**I did enact Julius Caesar: I was killed i’ the Capitol**”

Expanded query : “Give␣unto␣Caesar Brutus␣Cassius choreographers␣Bosco
Julius␣Caesar therefore␣unto␣Caesar Marcus␣Antonius
Caesarion Gallic␣Wars Marcus␣Crassus Antoninus Catiline
Seleucus Gaius␣Julius␣Caesar Theodoric Marcus␣Tullius␣Cicero

⋮

Kenneth Philip Marcus Arthur Carl Fred Edward Jonathan
Eric Frank Anthony William Richard Robert **enact Capitol**
killed Ididn’t honestly myself **I I** my we **the** ’d ’m **did was**”

We can include only highly similar words in the expanded query. Search engines such as Apache Lucene make it possible to assign weights to words in text queries.

Query expansion at search engines

- Main source of query expansion at search engines: query logs
- Example 1: After issuing the query [herbs], users frequently search for [herbal remedies].
    - → “herbal remedies” is potential expansion of “herb”.
- Example 2: Users searching for [flower pix] frequently click on the URL photobucket.com/flower. Users searching for [flower clipart] frequently click on the same URL.
    - → “flower clipart” and “flower pix” are potential expansions of each other.

Take-away today

- Interactive relevance feedback: improve initial retrieval results by telling the IR system which docs are relevant / non-relevant
- Best known relevance feedback method: Rocchio feedback
- Query expansion: improve retrieval results by adding synonyms / related terms to the query
  - Sources for related terms: Manual thesauri, automatic thesauri, query logs

Resources

- Chapter 9 of IIR
- Resources at `https://www.fi.muni.cz/~sojka/PV211/` and `http://cislmu.org`, materials in MU IS and FI MU library
  - Daniel Tunkelang's articles on query understanding, namely on query relaxation and query expansion.
  - Salton and Buckley 1990 (original relevance feedback paper)
  - Spink, Jansen, Ozmultu 2000: Relevance feedback at Excite
  - Justin Bieber: related searches fail
  - Word Space
  - Schütze 1998: Automatic word sense discrimination (describes a simple method for automatic thesaurus generation)
  - Sidorov et al. 2014: Soft similarity and soft cosine measure: Similarity of features in vector space model
  - Charlet and Damnati 2017: SimBow at SemEval-2017 Task 3: Soft-Cosine Semantic Similarity between Questions for Community Question Answering (describes two matrices **S**)

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 13: Text Classification & Naive Bayes
Handout version

Petr Sojka, Hinrich Schütze et al.

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

2020-04-15

Overview

1. Text classification
2. Naive Bayes
3. NB theory
4. Evaluation of TC

Take-away today

- Text classification: definition & relevance to information retrieval
- Naive Bayes: simple baseline text classifier
- Theory: derivation of Naive Bayes classification rule & analysis
- Evaluation of text classification: how do we know it worked / didn't work?

A text classification task: Email spam filtering

```
From: ‘‘’’ <takworlld@hotmail.com>
Subject: real estate is the only way... gem  oalvgkay

Anyone can buy real estate with no money down

Stop paying rent TODAY !

There is no need to spend hundreds or even thousands for similar courses

I am 22 years old and I have already purchased 6 properties using the
methods outlined in this truly INCREDIBLE ebook.

Change your life NOW !

=================================================
Click Below to order:
http://www.wholesaledaily.com/sales/nmd.htm
=================================================
```

How would you write a program that would automatically detect and delete this type of message?

Formal definition of TC: Training

Given:

- A document space $\mathbb{X}$
    - Documents are represented in this space – typically some type of high-dimensional space.
- A fixed set of classes $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$
    - The classes are human-defined for the needs of an application (e.g., spam vs. nonspam).
- A training set $\mathbb{D}$ of labeled documents. Each labeled document $\langle d, c \rangle \in \mathbb{X} \times \mathbb{C}$

Using a learning method or learning algorithm, we then wish to learn a classifier γ that maps documents to classes:

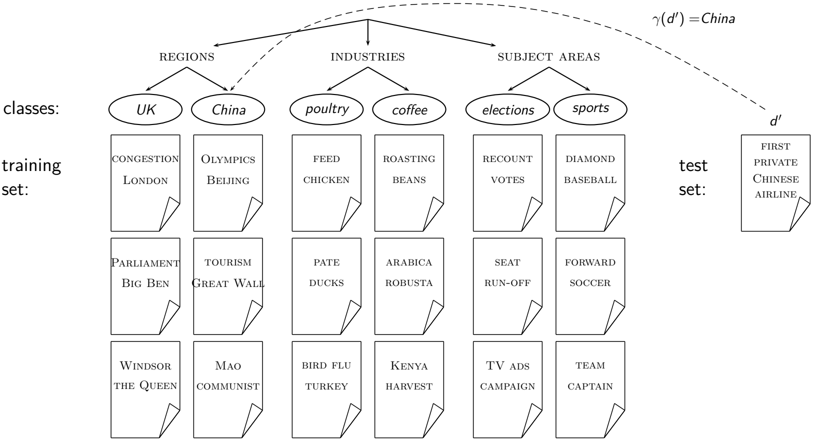
$$\gamma : \mathbb{X} \to \mathbb{C}$$

Formal definition of TC: Application/Testing

Given: a description $d \in \mathbb{X}$ of a document

Determine: $\gamma(d) \in \mathbb{C}$, that is, the class that is most appropriate for d

Topic classification



Exercise

- Find examples of uses of text classification in information retrieval

Examples of how search engines use classification

- Language identification (classes: English vs. French etc.)
- The automatic detection of spam pages (spam vs. nonspam)
- Sentiment detection: is a movie or product review positive or negative (positive vs. negative)
- Topic-specific or *vertical* search – restrict search to a “vertical” like “related to health” (relevant to vertical vs. not)

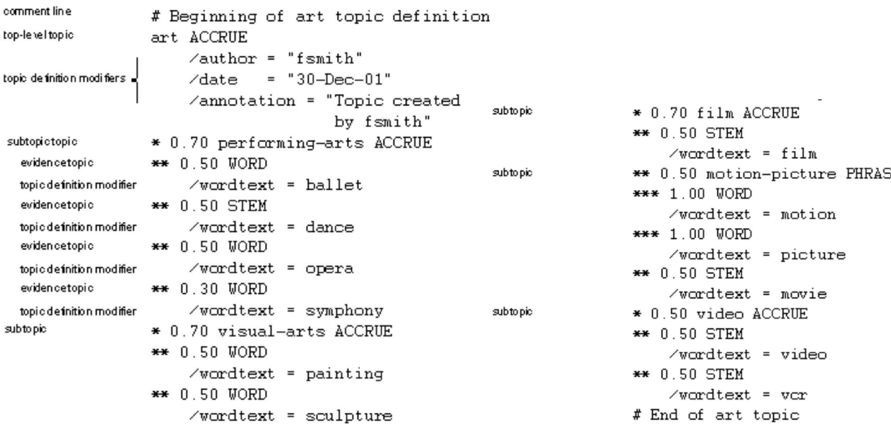
Classification methods: 1. Manual

- Manual classification was used by Yahoo in the beginning of the web. Also: MathSciNet (MSC), DMOZ – the Open Directory Project, PubMed
- Very accurate if job is done by experts.
- Consistent when the problem size and team is small.
- Scaling manual classification is difficult and expensive.
- → We need automatic methods for classification.

Classification methods: 2. Rule-based

- E.g., Google Alerts is rule-based classification.
- There are IDE-type development environments for writing very complex rules efficiently. (e.g., Verity)
- Often: Boolean combinations (as in Google Alerts).
- Accuracy is very high if a rule has been carefully refined over time by a subject expert.
- Building and maintaining rule-based classification systems is cumbersome and expensive.

A Verity topic (a complex classification rule)



comment line
top-level topic
topic definition modifiers
subtopictopic
evidencetopic
topic definition modifier
evidencetopic
topic definition modifier
evidencetopic
topic definition modifier
evidencetopic
topic definition modifier
subtopic
subtopic
subtopic
subtopic

```
# Beginning of art topic definition
art ACCRUE
    /author = "fsmith"
    /date   = "30-Dec-01"
    /annotation = "Topic created
                    by fsmith"
* 0.70 performing-arts ACCRUE
** 0.50 WORD
    /wordtext = ballet
** 0.50 STEM
    /wordtext = dance
** 0.50 WORD
    /wordtext = opera
** 0.30 WORD
    /wordtext = symphony
* 0.70 visual-arts ACCRUE
** 0.50 WORD
    /wordtext = painting
** 0.50 WORD
    /wordtext = sculpture
```

```
* 0.70 film ACCRUE
** 0.50 STEM
    /wordtext = film
** 0.50 motion-picture PHRAS
*** 1.00 WORD
    /wordtext = motion
*** 1.00 WORD
    /wordtext = picture
** 0.50 STEM
    /wordtext = movie
* 0.50 video ACCRUE
** 0.50 STEM
    /wordtext = video
** 0.50 STEM
    /wordtext = vcr
# End of art topic
```

Classification methods: 3. Statistical/Probabilistic

- This was our definition of the classification problem – text classification as a learning problem.
- (i) Supervised learning of the classification function γ and (ii) application of γ to classifying new documents.
- We will look at two methods for doing this: Naive Bayes and SVMs
- No free lunch: requires hand-classified training data.
- But this manual classification can be done by non-experts.

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document d being in a class c as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document. (number of tokens)
- $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class c
- $P(t_k|c)$ as a measure of how much evidence t_k contributes that c is the correct class.
- $P(c)$ is the prior probability of c.
- If a document's terms do not provide clear evidence for one class vs. another, we choose the c with highest $P(c)$.

Maximum a posteriori class

- Our goal in Naive Bayes classification is to find the "best" class.
- The best class is the most likely or maximum a posteriori (MAP) class c_{map}:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg\max_{c \in \mathbb{C}} \ \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c)$$

- We write $\hat{P}$ for P since these values are estimates from the training set.

Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.
- Since $\log(xy) = \log(x) + \log(y)$, we can sum log probabilities instead of multiplying probabilities.
- Since log is a monotonic function, the class with the highest score does not change.
- So what we usually compute in practice is:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \left[\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c) \right]$$

- Simple interpretation:
  - Each conditional parameter $\log \hat{P}(t_k|c)$ is a weight that indicates how good an indicator t_k is for c.
  - The prior $\log \hat{P}(c)$ is a weight that indicates the relative frequency of c.
  - The sum of log prior and term weights is then a measure of how much evidence there is for the document being in the class.
  - We select the class with the most evidence.

Parameter estimation take 1: Maximum likelihood

- Estimate parameters $\hat{P}(c)$ and $\hat{P}(t_k|c)$ from train data: How?
- Prior:

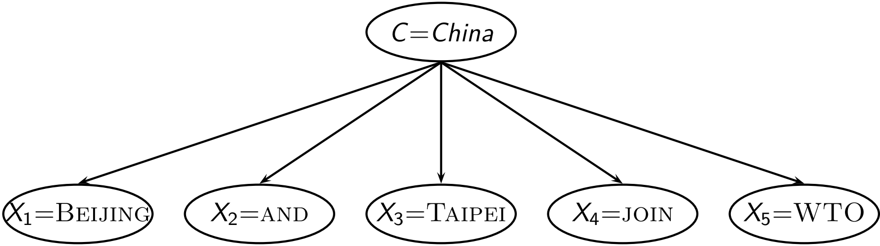
$$\hat{P}(c) = \frac{N_c}{N}$$

- N_c: number of docs in class c; N: total number of docs
- Conditional probabilities:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- T_{ct} is the number of tokens of t in training documents from class c (includes multiple occurrences)
- We have made a Naive Bayes independence assumption here: $\hat{P}(X_{k_1} = t|c) = \hat{P}(X_{k_2} = t|c)$, independent of position

The problem with maximum likelihood estimates: Zeros



$$\begin{aligned} P(China|d) \quad \propto \quad & P(China) \cdot P(\text{BEIJING}|China) \cdot P(\text{AND}|China) \\ & \cdot P(\text{TAIPEI}|China) \cdot P(\text{JOIN}|China) \cdot P(\text{WTO}|China) \end{aligned}$$

- If WTO never occurs in class China in the train set:

$$\hat{P}(\text{WTO}|China) = \frac{T_{China,\text{WTO}}}{\sum_{t' \in V} T_{China,t'}} = \frac{0}{\sum_{t' \in V} T_{China,t'}} = 0$$

The problem with maximum likelihood estimates: Zeros (cont)

- If there are no occurrences of WTO in documents in class China, we get a zero estimate:

$$\hat{P}(\mathrm{WTO}|China) = \frac{T_{China,\mathrm{WTO}}}{\sum_{t' \in V} T_{China,t'}} = 0$$

- $\rightarrow$ We will get $P(China|d) = 0$ for any document that contains WTO

To avoid zeros: Add-one smoothing

- Before:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- Now: Add one to each count to avoid zeros:

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V}(T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B}$$

- B is the number of bins – in this case the number of different words or the size of the vocabulary $|V| = M$

Naive Bayes: Summary

- Estimate parameters from the training corpus using add-one smoothing
- For a new document, for each class, compute sum of (i) log of prior and (ii) logs of conditional probabilities of the terms
- Assign the document to the class with the largest score

Naive Bayes: Training

```
TrainMultinomialNB($\mathbb{C}, \mathbb{D}$)
$V \leftarrow$ ExtractVocabulary($\mathbb{D}$)
$N \leftarrow$ CountDocs($\mathbb{D}$)
for each $c \in \mathbb{C}$
do $N_c \leftarrow$ CountDocsInClass($\mathbb{D}, c$)
    $prior[c] \leftarrow N_c/N$
    $text_c \leftarrow$ ConcatenateTextOfAllDocsInClass($\mathbb{D}, c$)
    for each $t \in V$
    do $T_{ct} \leftarrow$ CountTokensOfTerm($text_c, t$)
    for each $t \in V$
    do $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
return $V, prior, condprob$
```

Naive Bayes: Testing

```
ApplyMultinomialNB($\mathbb{C}, V, prior, condprob, d$)
$W \leftarrow$ ExtractTokensFromDoc($V, d$)
for each $c \in \mathbb{C}$
do $score[c] \leftarrow \log prior[c]$
    for each $t \in W$
    do $score[c] += \log condprob[t][c]$
return $\arg\max_{c \in \mathbb{C}} score[c]$
```

Exercise

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

- Estimate parameters of Naive Bayes classifier
- Classify test document

Example: Parameter estimates

Priors: $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$
Conditional probabilities:

$$\begin{aligned}
\hat{P}(\textsc{Chinese}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\textsc{Tokyo}|c) = \hat{P}(\textsc{Japan}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\textsc{Chinese}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\textsc{Tokyo}|\overline{c}) = \hat{P}(\textsc{Japan}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

The denominators are $(8+6)$ and $(3+6)$ because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant B is 6 as the vocabulary consists of six terms.

Example: Classification

$$\begin{aligned}\hat{P}(c|d_5) &\propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003 \\ \hat{P}(\overline{c}|d_5) &\propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001\end{aligned}$$

Thus, the classifier assigns the test document to $c = \textit{China}$. The reason for this classification decision is that the three occurrences of the positive indicator CHINESE in d_5 outweigh the occurrences of the two negative indicators JAPAN and TOKYO.

Time complexity of Naive Bayes

mode	time complexity
training	$\Theta(\lvert\mathbb{D}\rvert L_{\mathrm{ave}} + \lvert\mathbb{C}\rvert\lvert V\rvert)$
testing	$\Theta(L_{\mathrm{a}} + \lvert\mathbb{C}\rvert M_{\mathrm{a}}) = \Theta(\lvert\mathbb{C}\rvert M_{\mathrm{a}})$

- L_{ave}: average length of a training doc, L_{a}: length of the test doc, M_{a}: number of distinct terms in the test doc, $\mathbb{D}$: training set, V: vocabulary, $\mathbb{C}$: set of classes
- $\Theta(\lvert\mathbb{D}\rvert L_{\mathrm{ave}})$ is the time it takes to compute all counts.
- $\Theta(\lvert\mathbb{C}\rvert\lvert V\rvert)$ is the time it takes to compute the parameters from the counts.
- Generally: $\lvert\mathbb{C}\rvert\lvert V\rvert < \lvert\mathbb{D}\rvert L_{\mathrm{ave}}$
- Test time is also linear (in the length of the test document).
- Thus: Naive Bayes is linear in the size of the training set (training) and the test document (testing). This is optimal.

Naive Bayes: Analysis

- Now we want to gain a better understanding of the properties of Naive Bayes.
- We will formally derive the classification rule . . .
- . . . and make our assumptions explicit.

Derivation of Naive Bayes rule

We want to find the class that is most likely given the document:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} P(c|d)$$

Apply Bayes rule $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \frac{P(d|c)P(c)}{P(d)}$$

Drop denominator since $P(d)$ is the same for all classes:

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} P(d|c)P(c)$$

Too many parameters / sparseness

$$\begin{aligned} c_{\text{map}} &= \arg\max_{c \in \mathbb{C}} \; P(d|c)P(c) \\ &= \arg\max_{c \in \mathbb{C}} P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c)P(c) \end{aligned}$$

- There are too many parameters $P(\langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle | c)$, one for each unique combination of a class and a sequence of words.
- We would need a very, very large number of training examples to estimate that many parameters.
- This is the problem of data sparseness.

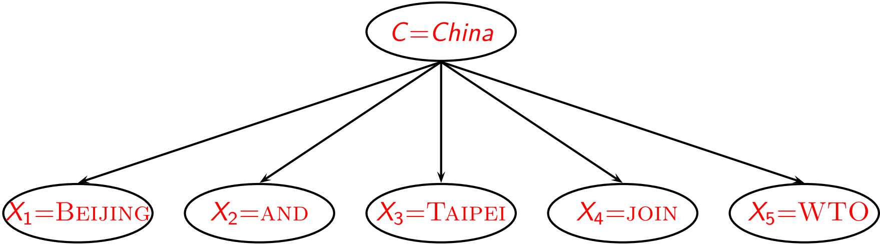
Naive Bayes conditional independence assumption

To reduce the number of parameters to a manageable size, we make the Naive Bayes conditional independence assumption:

$$P(d|c) = P(\langle t_1, \ldots, t_{n_d} \rangle|c) = \prod_{1 \leq k \leq n_d} P(X_k = t_k|c)$$

We assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(X_k = t_k|c)$.
Recall from earlier the estimates for these conditional probabilities: $\hat{P}(t|c) = \frac{T_{ct}+1}{(\sum_{t' \in V} T_{ct'})+B}$

Generative model



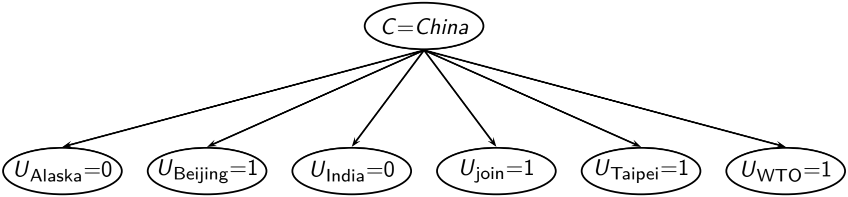
$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$

- Generate a class with probability $P(c)$
- Generate each of the words (in their respective positions), conditional on the class, but independent of each other, with probability $P(t_k|c)$
- To classify docs, we “reengineer” this process and find the class that is most likely to have generated the doc.

Second independence assumption

- $\hat{P}(X_{k_1} = t|c) = \hat{P}(X_{k_2} = t|c)$
- For example, for a document in the class *UK*, the probability of generating QUEEN in the first position of the document is the same as generating it in the last position.
- The two independence assumptions amount to the bag of words model.

A different Naive Bayes model: Bernoulli model



Violation of Naive Bayes independence assumptions

- Conditional independence:

$$P(\langle t_1, \dots, t_{n_d} \rangle | c) = \prod_{1 \leq k \leq n_d} P(X_k = t_k | c)$$

- Positional independence:
- $\hat{P}(X_{k_1} = t|c) = \hat{P}(X_{k_2} = t|c)$
- The independence assumptions do not really hold of documents written in natural language.
- Exercise
  - Examples for why conditional independence assumption is not really true?
  - Examples for why positional independence assumption is not really true?
- How can Naive Bayes work if it makes such inappropriate assumptions?

Why does Naive Bayes work?

- Naive Bayes can work well even though conditional independence assumptions are badly violated.
- Example:

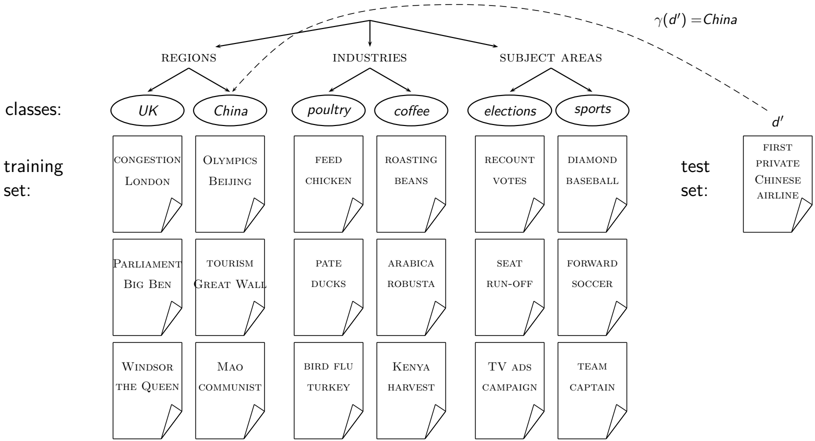
	c_1	c_2	class selected
true probability $P(c\|d)$	0.6	0.4	c_1
$\hat{P}(c)\prod_{1\leq k\leq n_d}\hat{P}(t_k\|c)$	0.00099	0.00001	
NB estimate $\hat{P}(c\|d)$	0.99	0.01	c_1

- Double counting of evidence causes underestimation (0.01) and overestimation (0.99).
- Classification is about predicting the correct class and not about accurately estimating probabilities.
- Naive Bayes is terrible for correct estimation . . .
- . . . but if often performs well at accurate prediction (choosing the correct class).

Naive Bayes is not so naive

- Naive Bayes has won some bakeoffs (e.g., KDD-CUP 97)
- More robust to irrelevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods
- Better than methods like decision trees when we have many equally important features
- A good dependable baseline for text classification (but not the best)
- Optimal if independence assumptions hold (never true for text, but true for some domains)
- Very fast
- Low storage requirements

Evaluation on Reuters

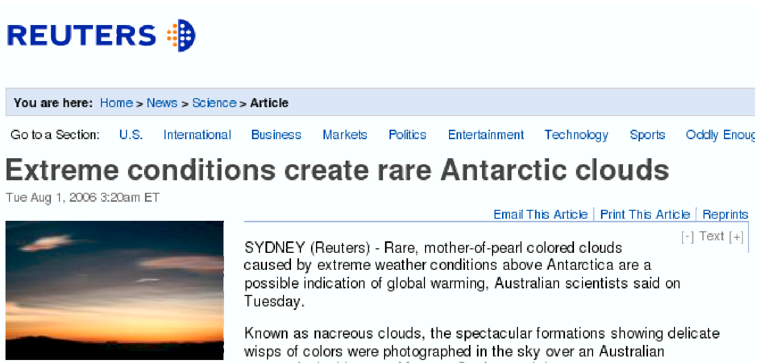


Example: The Reuters collection

symbol	statistic	value
N	documents	800,000
L	avg. # word tokens per document	200
M	word types	400,000

type of class	number	examples
region	366	UK, China
industry	870	poultry, coffee
subject area	126	elections, sports

A Reuters document



REUTERS

You are here: Home > News > Science > **Article**

Go to a Section: U.S. International Business Markets Politics Entertainment Technology Sports Oddly Enoug

Extreme conditions create rare Antarctic clouds

Tue Aug 1, 2006 3:20am ET

Email This Article | Print This Article | Reprints

[-] Text [+]

SYDNEY (Reuters) - Rare, mother-of-pearl colored clouds caused by extreme weather conditions above Antarctica are a possible indication of global warming, Australian scientists said on Tuesday.

Known as nacreous clouds, the spectacular formations showing delicate wisps of colors were photographed in the sky over an Australian

Evaluating classification

- Evaluation must be done on test data that are independent of the training data, i.e., training and test sets are disjoint.
- It's easy to get good performance on a test set that was available to the learner during training (e.g., just memorize the test set).
- Measures: Precision, recall, F_1, classification accuracy
- Average measures over multiple training and test sets (splits of the overall data) for best results.

Precision P and recall R

	in the class	not in the class
predicted to be in the class	true positives (TP)	false positives (FP)
predicted to not be in the class	false negatives (FN)	true negatives (TN)

TP, FP, FN, TN are counts of documents. The sum of these four counts is the total number of documents.

$$\begin{aligned} \text{precision:} P &= TP/(TP+FP) \\ \text{recall:} R &= TP/(TP+FN) \end{aligned}$$

A combined measure: F

- F_1 allows us to trade off precision against recall.
- $$F_1 = \frac{1}{\frac{1}{2}\frac{1}{P} + \frac{1}{2}\frac{1}{R}} = \frac{2PR}{P+R}$$
- This is the harmonic mean of P and R: $\frac{1}{F} = \frac{1}{2}(\frac{1}{P} + \frac{1}{R})$

Averaging: Micro vs. Macro

- We now have an evaluation measure (F_1) for one class.
- But we also want a single number that measures the aggregate performance over all classes in the collection.
- Macroaveraging
  - Compute F_1 for each of the C classes
  - Average these C numbers
- Microaveraging
  - Compute TP, FP, FN for each of the C classes
  - Sum these C numbers (e.g., all TP to get aggregate TP)
  - Compute F_1 for aggregate TP, FP, FN

Naive Bayes vs. other methods

(a)	NB	Rocchio	kNN		SVM
micro-avg-L (90 classes)	80	85	86		89
macro-avg (90 classes)	47	59	60		60

(b)	NB	Rocchio	kNN	trees	SVM
earn	96	93	97	98	98
acq	88	65	92	90	94
money-fx	57	47	78	66	75
grain	79	68	82	85	95
crude	80	70	86	85	89
trade	64	65	77	73	76
interest	65	63	74	67	78
ship	85	49	79	74	86
wheat	70	69	77	93	92
corn	65	48	78	92	90
micro-avg (top 10)	82	65	82	88	92
micro-avg-D (118 classes)	75	62	n/a	n/a	87

Evaluation measure: F_1

Naive Bayes does pretty well, but some methods beat it consistently (e.g., SVM).

Take-away today

- Text classification: definition & relevance to information retrieval
- Naive Bayes: simple baseline text classifier
- Theory: derivation of Naive Bayes classification rule & analysis
- Evaluation of text classification: how do we know it worked / didn't work?

Resources

- Chapter 13 of IIR
- Resources at `https://www.fi.muni.cz/~sojka/PV211/` and `http://cislmu.org`, materials in MU IS and FI MU library
    - Weka: A data mining software package that includes an implementation of Naive Bayes
    - Reuters-21578 – text classification evaluation set
    - Vulgarity classifier fail

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 16: Flat Clustering
Handout version

Petr Sojka, Hinrich Schütze et al.

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

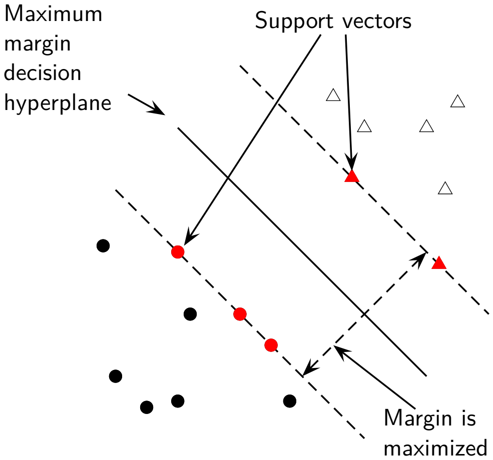
2020-04-22

Overview

1. Recap
2. Clustering: Introduction
3. Clustering in IR
4. K-means
5. Evaluation
6. How many clusters?

Support vector machines

- Binary classification problem
- Simple SVMs are linear classifiers.
- criterion: being maximally far away from any data point → determines classifier **margin**
- linear separator position defined by **support vectors**



Optimization problem solved by SVMs

Find $\vec{w}$ and b such that:

- $\frac{1}{2}\vec{w}^{\mathsf{T}}\vec{w}$ is minimized (because $|\vec{w}| = \sqrt{\vec{w}^{\mathrm{T}}\vec{w}}$), and
- for all $\{(\vec{x}_i, y_i)\}$, $y_i(\vec{w}^{\mathsf{T}}\vec{x}_i + b) \geq 1$

Which machine learning method to choose?

- Is there a learning method that is optimal for all text classification problems?
- No, because there is a tradeoff, a dilemma between bias and variance.
- Factors to take into account:
  - How much training data is available?
  - How simple/complex is the problem? (linear vs. nonlinear decision boundary)
  - How noisy is the problem?
  - How stable is the problem over time?
    - For an unstable problem, it's better to use a simple and robust classifier.

See Fig. 15 in Geman et al.

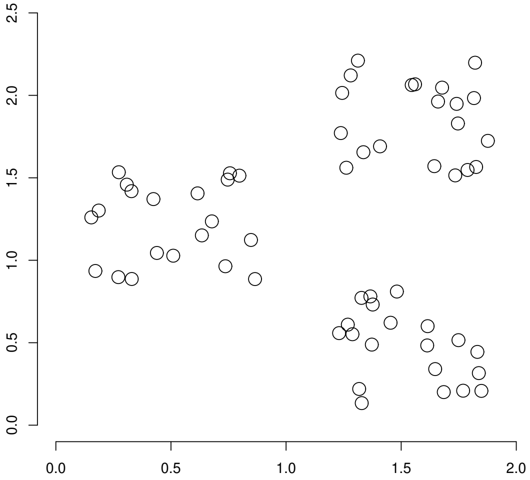
Take-away today

- What is clustering?
- Applications of clustering in information retrieval
- *K*-means algorithm
- Evaluation of clustering
- How many clusters? □

Clustering: Definition

- (Document) clustering is the process of grouping a set of documents into clusters of similar documents.
- Documents within a cluster should be similar.
- Documents from different clusters should be dissimilar.
- Clustering is the most common form of unsupervised learning.
- Unsupervised = there are no labeled or annotated data. □
- Hard clustering vs. soft clustering.
- Cardinality of clustering.

Exercise: Data set with clear cluster structure



Propose algorithm for finding the cluster structure in this example □

Classification vs. Clustering

- Classification: supervised learning
- Clustering: unsupervised learning
- Classification: Classes are human-defined and part of the input to the learning algorithm.
- Clustering: Clusters are inferred from the data without human input.
    - However, there are many ways of influencing the outcome of clustering: number of clusters, similarity measure, representation of documents, . . . □

The cluster hypothesis

Cluster hypothesis. Documents in the same cluster behave similarly with respect to relevance to information needs.

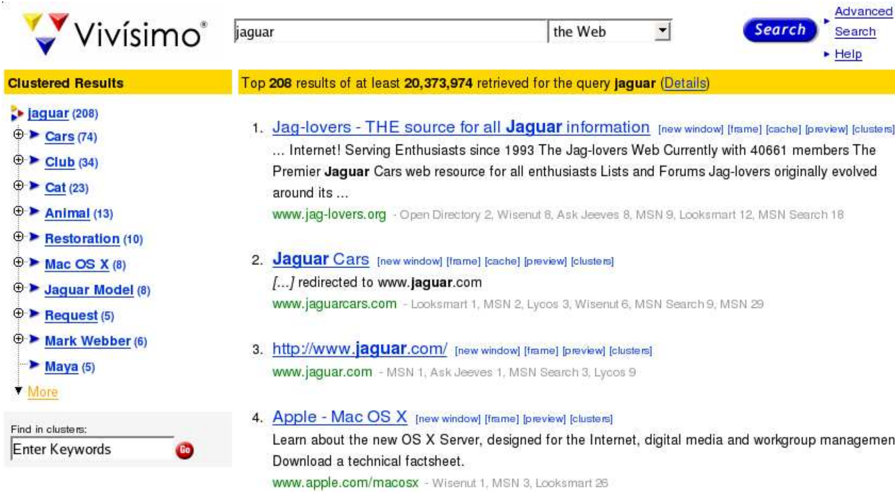
All applications of clustering in IR are based (directly or indirectly) on the cluster hypothesis.

Van Rijsbergen's original wording (1979): "closely associated documents tend to be relevant to the same requests". □

Applications of clustering in IR

application	what is clustered?	benefit
search result clustering	search results	more effective information presentation to user
Scatter-Gather	(subsets of) collection	alternative user interface: "search without typing"
collection clustering	collection	effective information presentation for exploratory browsing
cluster-based retrieval	collection	higher efficiency: faster search

Search result clustering for better navigation



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

Find in clusters:

Enter Keywords Go

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

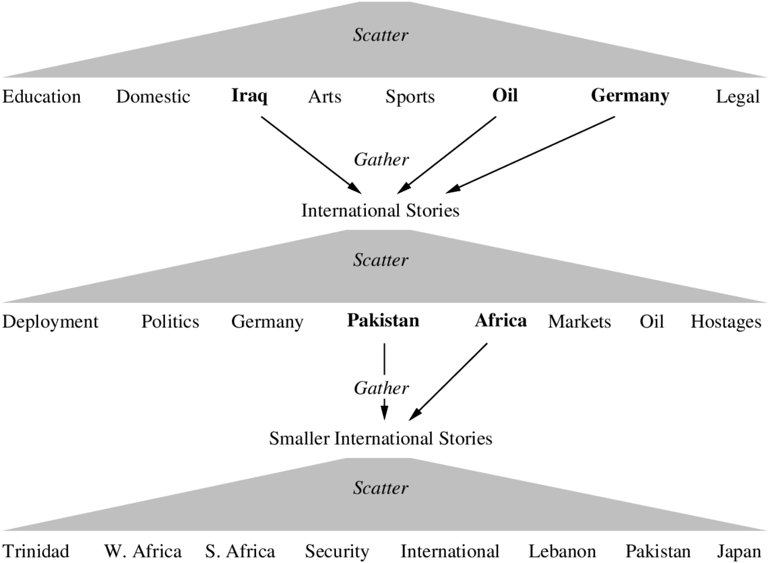
1. Jag-lovers - THE source for all **Jaguar** information [new window] [frame] [cache] [preview] [clusters]
   ... Internet! Serving Enthusiasts since 1993 The Jag-lovers Web Currently with 40661 members The Premier **Jaguar** Cars web resource for all enthusiasts Lists and Forums Jag-lovers originally 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

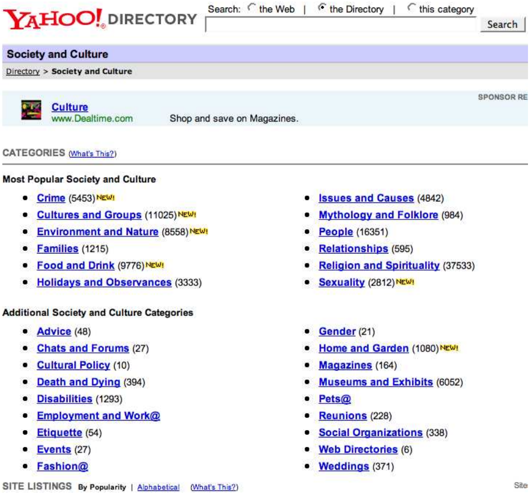
3. http://www.**jaguar**.com/ [new window] [frame] [preview] [clusters]
   www.jaguar.com - MSN 1, Ask Jeeves 1, MSN Search 3, Lycos 9

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 managemen
   Download a technical factsheet.
   www.apple.com/macosx - Wisenut 1, MSN 3, Looksmart 26

Scatter-Gather



Global navigation: Yahoo



Search: ○ the Web | ● the Directory | ○ this category

Society and Culture

Directory > **Society and Culture**

SPONSOR RE

Culture
www.Dealtime.com Shop and save on Magazines.

CATEGORIES (What's This?)

Most Popular Society and Culture

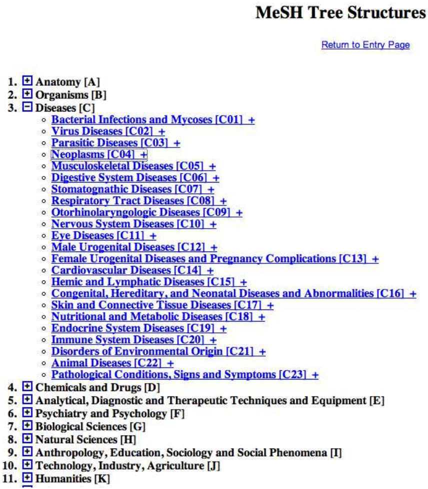
- Crime (5453) NEW!
- Cultures and Groups (11025) NEW!
- Environment and Nature (8558) NEW!
- Families (1215)
- Food and Drink (9776) NEW!
- Holidays and Observances (3333)
- Issues and Causes (4842)
- Mythology and Folklore (984)
- People (16351)
- Relationships (595)
- Religion and Spirituality (37533)
- Sexuality (2812) NEW!

Additional Society and Culture Categories

- Advice (48)
- Chats and Forums (27)
- Cultural Policy (10)
- Death and Dying (394)
- Disabilities (1293)
- Employment and Work@
- Etiquette (54)
- Events (27)
- Fashion@
- Gender (21)
- Home and Garden (1080) NEW!
- Magazines (164)
- Museums and Exhibits (6052)
- Pets@
- Reunions (228)
- Social Organizations (338)
- Web Directories (6)
- Weddings (371)

SITE LISTINGS By Popularity | Alphabetical (What's This?) Site

Global navigation: Medical Subject Headings MESH (upper level)

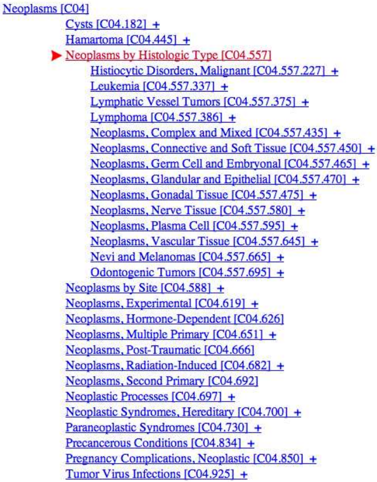


MeSH Tree Structures - 2008

Return to Entry Page

1. ⊞ Anatomy [A]
2. ⊞ Organisms [B]
3. ⊟ Diseases [C]
   - Bacterial Infections and Mycoses [C01] +
   - Virus Diseases [C02] +
   - Parasitic Diseases [C03] +
   - Neoplasms [C04] +
   - Musculoskeletal Diseases [C05] +
   - Digestive System Diseases [C06] +
   - Stomatognathic Diseases [C07] +
   - Respiratory Tract Diseases [C08] +
   - Otorhinolaryngologic Diseases [C09] +
   - Nervous System Diseases [C10] +
   - Eye Diseases [C11] +
   - Male Urogenital Diseases [C12] +
   - Female Urogenital Diseases and Pregnancy Complications [C13] +
   - Cardiovascular Diseases [C14] +
   - Hemic and Lymphatic Diseases [C15] +
   - Congenital, Hereditary, and Neonatal Diseases and Abnormalities [C16] +
   - Skin and Connective Tissue Diseases [C17] +
   - Nutritional and Metabolic Diseases [C18] +
   - Endocrine System Diseases [C19] +
   - Immune System Diseases [C20] +
   - Disorders of Environmental Origin [C21] +
   - Animal Diseases [C22] +
   - Pathological Conditions, Signs and Symptoms [C23] +
4. ⊞ Chemicals and Drugs [D]
5. ⊞ Analytical, Diagnostic and Therapeutic Techniques and Equipment [E]
6. ⊞ Psychiatry and Psychology [F]
7. ⊞ Biological Sciences [G]
8. ⊞ Natural Sciences [H]
9. ⊞ Anthropology, Education, Sociology and Social Phenomena [I]
10. ⊞ Technology, Industry, Agriculture [J]
11. ⊞ Humanities [K]

Global navigation: MESH (lower level)

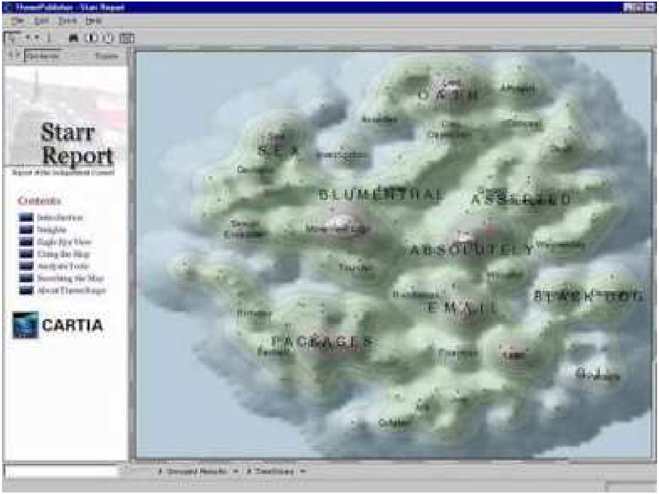


- Neoplasms [C04]
  - Cysts [C04.182] +
  - Hamartoma [C04.445] +
  - ➤ Neoplasms by Histologic Type [C04.557]
    - Histiocytic Disorders, Malignant [C04.557.227] +
    - Leukemia [C04.557.337] +
    - Lymphatic Vessel Tumors [C04.557.375] +
    - Lymphoma [C04.557.386] +
    - Neoplasms, Complex and Mixed [C04.557.435] +
    - Neoplasms, Connective and Soft Tissue [C04.557.450] +
    - Neoplasms, Germ Cell and Embryonal [C04.557.465] +
    - Neoplasms, Glandular and Epithelial [C04.557.470] +
    - Neoplasms, Gonadal Tissue [C04.557.475] +
    - Neoplasms, Nerve Tissue [C04.557.580] +
    - Neoplasms, Plasma Cell [C04.557.595] +
    - Neoplasms, Vascular Tissue [C04.557.645] +
    - Nevi and Melanomas [C04.557.665] +
    - Odontogenic Tumors [C04.557.695] +
  - Neoplasms by Site [C04.588] +
  - Neoplasms, Experimental [C04.619] +
  - Neoplasms, Hormone-Dependent [C04.626]
  - Neoplasms, Multiple Primary [C04.651] +
  - Neoplasms, Post-Traumatic [C04.666]
  - Neoplasms, Radiation-Induced [C04.682] +
  - Neoplasms, Second Primary [C04.692]
  - Neoplastic Processes [C04.697] +
  - Neoplastic Syndromes, Hereditary [C04.700] +
  - Paraneoplastic Syndromes [C04.730] +
  - Precancerous Conditions [C04.834] +
  - Pregnancy Complications, Neoplastic [C04.850] +
  - Tumor Virus Infections [C04.925] +

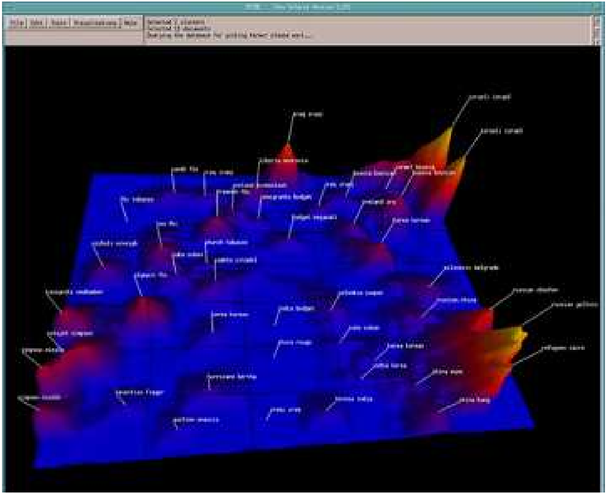
Navigational hierarchies: Manual vs. automatic creation

- Note: Yahoo/MESH are not examples of clustering.
- But they are well known examples for using a global hierarchy for navigation.
- Some examples for global navigation/exploration based on clustering:
  - Arxiv's LDAExplore: `https://arxiv.lateral.io/`
  - Cartia
  - Themescapes
  - Google News □

Global navigation combined with visualization (1)



Global navigation combined with visualization (2)



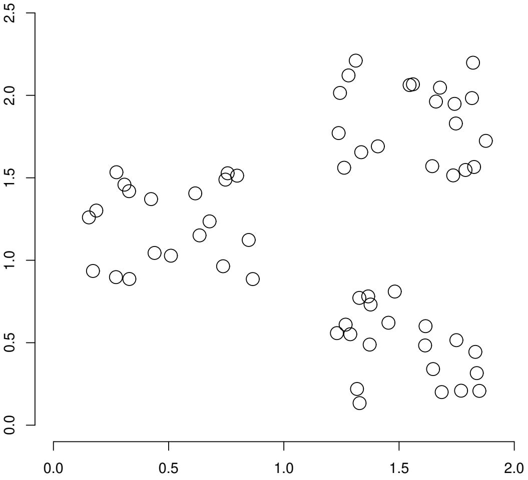
Global clustering for navigation: Google News

`http://news.google.com`

Clustering for improving recall

- To improve search recall:
  - Cluster docs in collection a priori
  - When a query matches a doc *d*, also return other docs in the cluster containing *d*
- Hope: if we do this: the query "car" will also return docs containing "automobile"
  - Because the clustering algorithm groups together docs containing "car" with those containing "automobile".
  - Both types of documents contain words like "parts", "dealer", "mercedes", "road trip". □

Exercise: Data set with clear cluster structure



Propose algorithm for finding the cluster structure in this example □

Desiderata for clustering

- General goal: put related docs in the same cluster, put unrelated docs in different clusters.
  - We'll see different ways of formalizing this.
- The number of clusters should be appropriate for the data set we are clustering.
  - Initially, we will assume the number of clusters K is given.
  - Later: Semiautomatic methods for determining K
- Secondary goals in clustering
  - Avoid very small and very large clusters
  - Define clusters that are easy to explain to the user
  - Many others . . .

□

Flat vs. Hierarchical clustering

- Flat algorithms
  - Usually start with a random (partial) partitioning of docs into groups
  - Refine iteratively
  - Main algorithm: *K*-means
- Hierarchical algorithms
  - Create a hierarchy
  - Bottom-up, agglomerative
  - Top-down, divisive

□

Hard vs. Soft clustering

- Hard clustering: Each document belongs to exactly one cluster.
    - More common and easier to do
- Soft clustering: A document can belong to more than one cluster.
    - Makes more sense for applications like creating browsable hierarchies
    - You may want to put *sneakers* in two clusters:
        - sports apparel
        - shoes
    - You can only do that with a soft clustering approach.
- This class: flat, hard clustering; next: hierarchical, hard clustering then: latent semantic indexing, a form of soft clustering □
- We won't have time for soft clustering. See IIR 16.5, IIR 18
- Non-exhaustive clustering: some docs are not assigned to any cluster. See references in IIR 16.

Flat algorithms

- Flat algorithms compute a partition of N documents into a set of K clusters.
- Given: a set of documents and the number K
- Find: a partition into K clusters that optimizes the chosen partitioning criterion
- Global optimization: exhaustively enumerate partitions, pick optimal one
  - Not tractable
- Effective heuristic method: K-means algorithm

K-means

- Perhaps the best known clustering algorithm
- Simple, works well in many cases
- Use as default / baseline for clustering documents

Document representations in clustering

- Vector space model
- As in vector space classification, we measure relatedness between vectors by Euclidean distance . . .
- . . . which is almost equivalent to cosine similarity.
- Almost: centroids are not length-normalized. □

K-means: Basic idea

- Each cluster in *K*-means is defined by a centroid.
- Objective/partitioning criterion: minimize the average squared difference from the centroid
- Recall definition of centroid:

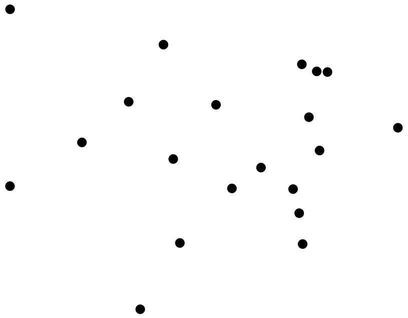
$$\vec{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x}$$

  where we use ω to denote a cluster.
- We try to find the minimum average squared difference by iterating two steps:
  - reassignment: assign each vector to its closest centroid
  - recomputation: recompute each centroid as the average of the vectors that were assigned to it in reassignment

K-means pseudocode (μ_k is centroid of ω_k)

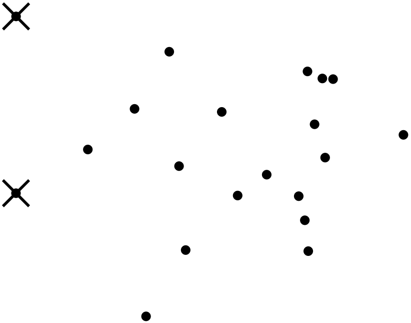
```
$K$-MEANS($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$)
  $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_K) \leftarrow$ SELECTRANDOMSEEDS($\{\vec{x}_1, \ldots, \vec{x}_N\}, K$)
  for $k \leftarrow 1$ to $K$
  do $\vec{\mu}_k \leftarrow \vec{s}_k$
  while stopping criterion has not been met
  do for $k \leftarrow 1$ to $K$
       do $\omega_k \leftarrow \{\}$
       for $n \leftarrow 1$ to $N$
       do $j \leftarrow \arg\min_{j'} |\vec{\mu}_{j'} - \vec{x}_n|$
          $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$  (reassignment of vectors)
       for $k \leftarrow 1$ to $K$
       do $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$  (recomputation of centroids)
  return $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$
```

Worked Example: Set of points to be clustered

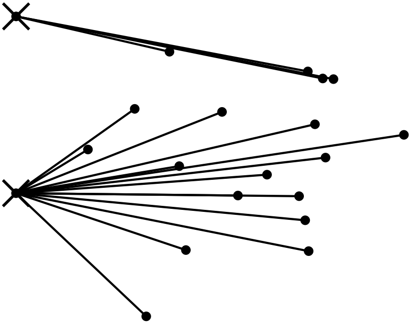


Exercise: (i) Guess what the optimal clustering into two clusters is in this case; (ii) compute the centroids of the clusters □

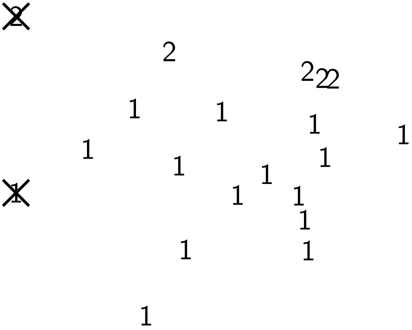
Worked Example: Random selection of initial centroids



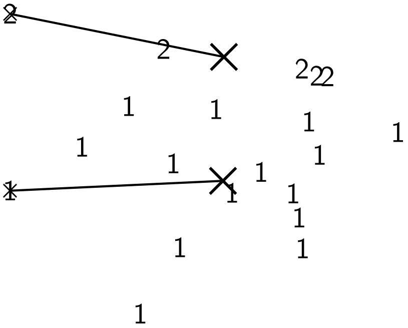
Worked Example: Assign points to closest center



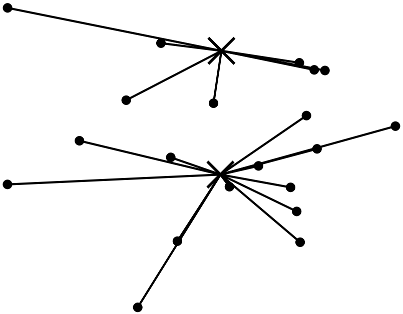
Worked Example: Assignment



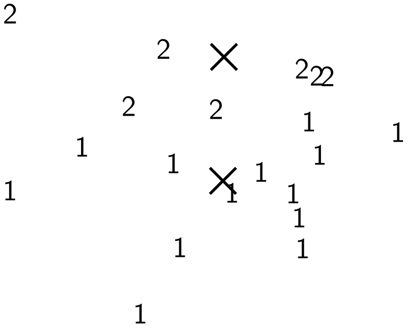
Worked Example: Recompute cluster centroids



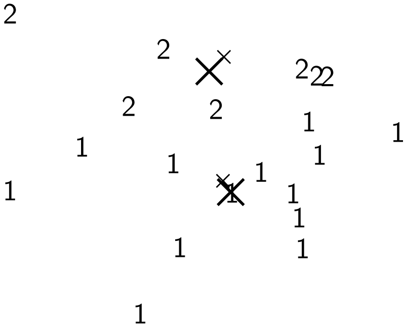
Worked Example: Assign points to closest centroid



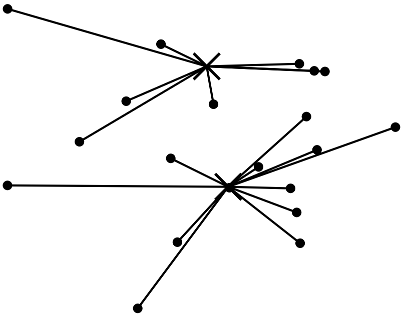
Worked Example: Assignment



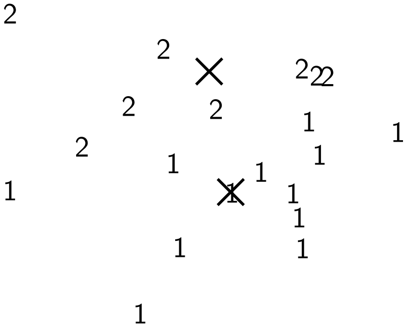
Worked Example: Recompute cluster centroids



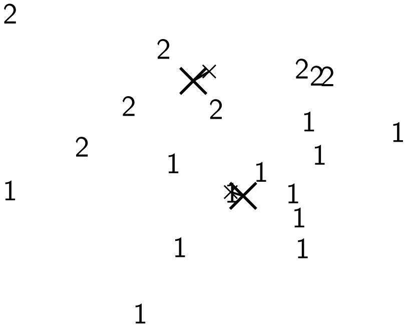
Worked Example: Assign points to closest centroid



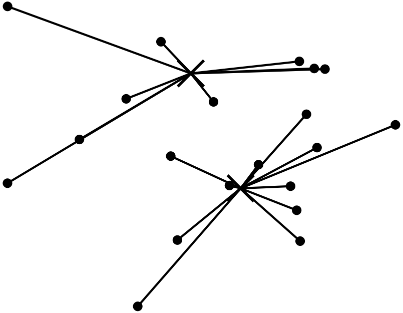
Worked Example: Assignment



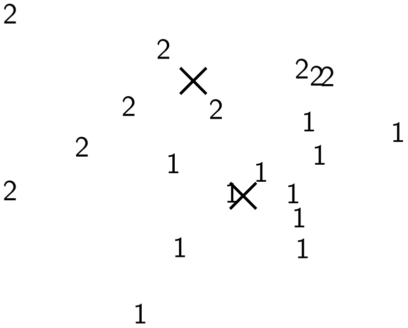
Worked Example: Recompute cluster centroids



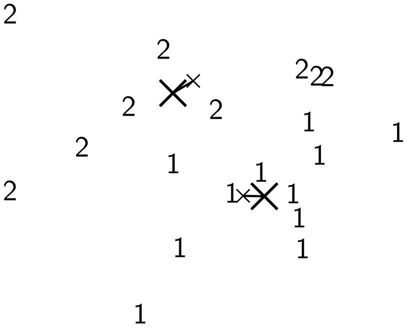
Worked Example: Assign points to closest centroid



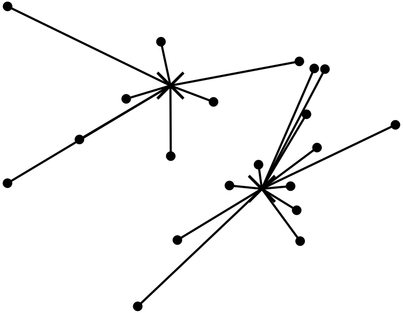
Worked Example: Assignment



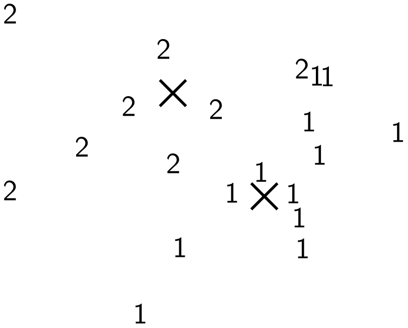
Worked Example: Recompute cluster centroids



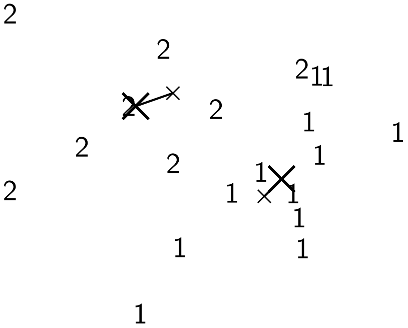
Worked Example: Assign points to closest centroid



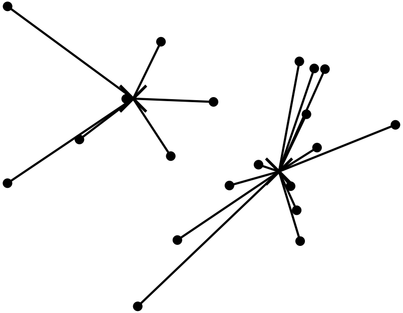
Worked Example: Assignment



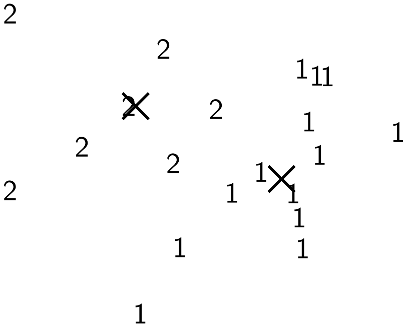
Worked Example: Recompute cluster centroids



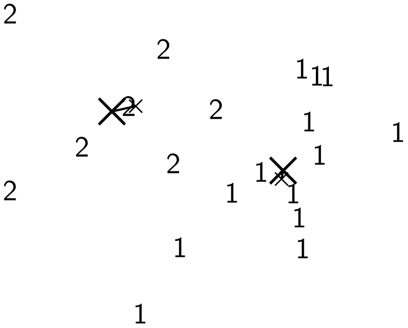
Worked Example: Assign points to closest centroid



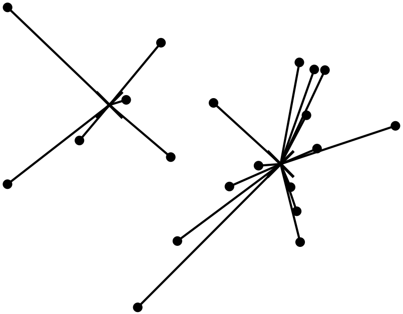
Worked Example: Assignment



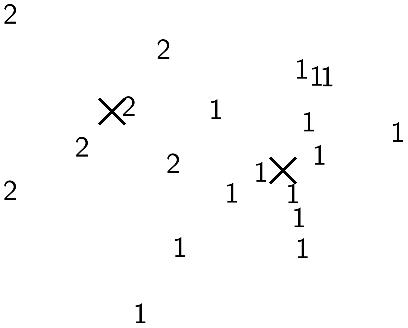
Worked Example: Recompute cluster centroids



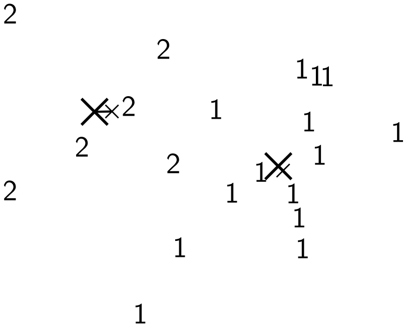
Worked Example: Assign points to closest centroid



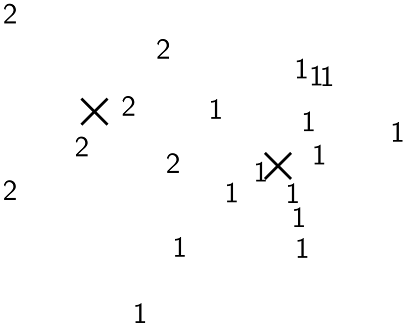
Worked Example: Assignment



Worked Example: Recompute cluster centroids



Worked Ex.: Centroids and assignments after convergence



K-means is guaranteed to converge: Proof

- RSS (Residual Sum of Squares) = sum of all squared distances between document vector and closest centroid
- RSS decreases during each reassignment step.
  - because each vector is moved to a closer centroid
- RSS decreases during each recomputation step.
  - see next slide
- There is only a finite number of clusterings.
- Thus: We must reach a fixed point.
- Assumption: Ties are broken consistently.
- Finite set & monotonically decreasing → convergence □

Recomputation decreases average distance

$\text{RSS} = \sum_{k=1}^{K} \text{RSS}_k$ – the residual sum of squares (the "goodness" measure)

$$\begin{aligned}
\text{RSS}_k(\vec{v}) &= \sum_{\vec{x}\in\omega_k} \|\vec{v}-\vec{x}\|^2 = \sum_{\vec{x}\in\omega_k} \sum_{m=1}^{M} (v_m - x_m)^2 \\
\frac{\partial \text{RSS}_k(\vec{v})}{\partial v_m} &= \sum_{\vec{x}\in\omega_k} 2(v_m - x_m) = 0
\end{aligned}$$

$$v_m = \frac{1}{|\omega_k|} \sum_{\vec{x}\in\omega_k} x_m$$

The last line is the componentwise definition of the centroid!
We minimize RSS_k when the old centroid is replaced with the new centroid. RSS, the sum of the RSS_k, must then also decrease during recomputation. □

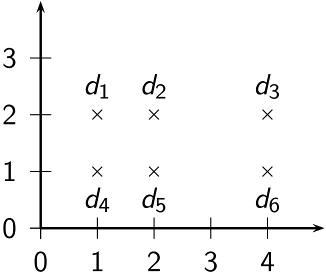
K-means is guaranteed to converge

- But we don't know how long convergence will take!
- If we don't care about a few docs switching back and forth, then convergence is usually fast ($<$ 10–20 iterations).
- However, complete convergence can take many more iterations. □

Optimality of K-means

- Convergence $\neq$ optimality
- Convergence does not mean that we converge to the optimal clustering!
- This is the great weakness of K-means.
- If we start with a bad set of seeds, the resulting clustering can be horrible. □

Exercise: Suboptimal clustering



- What is the optimal clustering for $K = 2$?
- Do we converge on this clustering for arbitrary seeds d_i, d_j?

□

Initialization of K-means

- Random seed selection is just one of many ways K-means can be initialized.
- Random seed selection is not very robust: It's easy to get a suboptimal clustering.
- Better ways of computing initial centroids:
    - Select seeds not randomly, but using some heuristic (e.g., filter out outliers or find a set of seeds that has "good coverage" of the document space)
    - Use hierarchical clustering to find good seeds
    - Select i (e.g., $i = 10$) different random sets of seeds, do a K-means clustering for each, select the clustering with lowest RSS □

Time complexity of K-means

- Computing one distance of two vectors is $O(M)$.
- Reassignment step: $O(KNM)$ (we need to compute KN document-centroid distances)
- Recomputation step: $O(NM)$ (we need to add each of the document's $< M$ values to one of the centroids)
- Assume number of iterations bounded by I
- Overall complexity: $O(IKNM)$ – linear in all important dimensions
- However: This is not a real worst-case analysis.
- In pathological cases, complexity can be worse than linear. □

What is a good clustering?

- Internal criteria
  - Example of an internal criterion: RSS in *K*-means
- But an internal criterion often does not evaluate the actual utility of a clustering in the application.
- Alternative: External criteria
  - Evaluate with respect to a human-defined classification □

External criteria for clustering quality

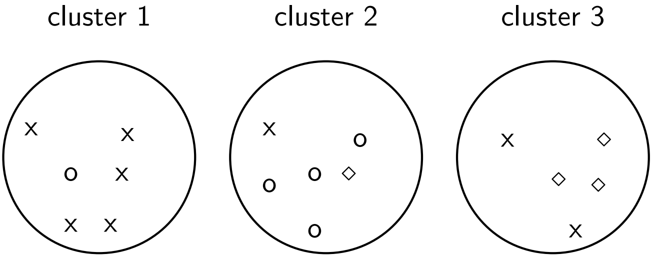
- Based on a gold standard data set, e.g., the Reuters collection we also used for the evaluation of classification
- Goal: Clustering should reproduce the classes in the gold standard
- (But we only want to reproduce how documents are divided into groups, not the class labels.)
- First measure for how well we were able to reproduce the classes: purity □

External criterion: Purity

$$\text{purity}(\Omega, C) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

- $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ is the set of clusters and $C = \{c_1, c_2, \ldots, c_J\}$ is the set of classes.
- For each cluster ω_k: find class c_j with most members n_{kj} in ω_k
- Sum all n_{kj} and divide by total number of points □

Example for computing purity



To compute purity: $5 = \max_j |\omega_1 \cap c_j|$ (class x, cluster 1); $4 = \max_j |\omega_2 \cap c_j|$ (class o, cluster 2); and $3 = \max_j |\omega_3 \cap c_j|$ (class $\diamond$, cluster 3). Purity is $(1/17) \times (5 + 4 + 3) \approx 0.71$. □

Another external criterion: Rand index

- Purity can be increased easily by increasing K – a measure that does not have this problem: Rand index.
- Definition: $\mathrm{RI} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{FN}+\mathrm{TN}}$
- Based on 2x2 contingency table of all pairs of documents:

	same cluster	different clusters
same class	true positives (TP)	false negatives (FN)
different classes	false positives (FP)	true negatives (TN)

- TP+FN+FP+TN is the total number of pairs.
- TP+FN+FP+TN $= \binom{N}{2}$ for N documents.
- Example: $\binom{17}{2} = 136$ in o/$\diamond$/x example
- Each pair is either positive or negative (the clustering puts the two documents in the same or in different clusters) . . .
- . . . and either "true" (correct) or "false" (incorrect): the clustering decision is correct or incorrect.

Rand Index: Example

As an example, we compute RI for the o/⋄/x example. We first compute TP + FP. The three clusters contain 6, 6, and 5 points, respectively, so the total number of "positives" or pairs of documents that are in the same cluster is:

$$\mathrm{TP}+\mathrm{FP} = \binom{6}{2} + \binom{6}{2} + \binom{5}{2} = 40$$

Of these, the x pairs in cluster 1, the o pairs in cluster 2, the ⋄ pairs in cluster 3, and the x pair in cluster 3 are true positives:

$$\mathrm{TP} = \binom{5}{2} + \binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 20$$

Thus, FP = 40 − 20 = 20.

FN and TN are computed similarly. □

Rand measure for the o/$\diamond$/x example

	same cluster	different clusters
same class	TP = 20	FN = 24
different classes	FP = 20	TN = 72

RI is then $(20 + 72)/(20 + 20 + 24 + 72) \approx 0.68$. □

Two other external evaluation measures

- Two other measures
- Normalized mutual information (NMI)
  - How much information does the clustering contain about the classification?
  - Singleton clusters (number of clusters = number of docs) have maximum MI
  - Therefore: normalize by entropy of clusters and classes
- F measure
  - Like Rand, but "precision" and "recall" can be weighted □

Evaluation results for the o/◇/x example

	purity	NMI	RI	F_5
lower bound	0.0	0.0	0.0	0.0
maximum	1.0	1.0	1.0	1.0
value for example	0.71	0.36	0.68	0.46

All four measures range from 0 (really bad clustering) to 1 (perfect clustering). □

How many clusters?

- Number of clusters K is given in many applications.
  - E.g., there may be an external constraint on K. Example: In the case of Scatter-Gather, it was hard to show more than 10–20 clusters on a monitor in the 90s.
- What if there is no external constraint? Is there a "right" number of clusters?
- One way to go: define an optimization criterion
  - Given docs, find K for which the optimum is reached.
  - What optimization criterion can we use?
  - We can't use RSS or average squared distance from centroid as criterion: always chooses $K = N$ clusters. □

Exercise

- Your job is to develop the clustering algorithms for a competitor to news.google.com
- You want to use K-means clustering.
- How would you determine K? □

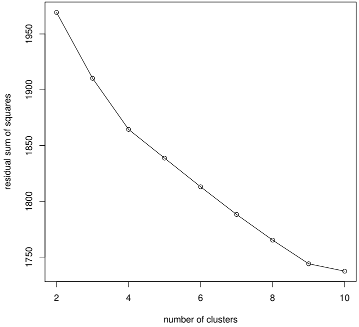
Simple objective function for K: Basic idea

- Start with 1 cluster ($K = 1$)
- Keep adding clusters (= keep increasing K)
- Add a penalty for each new cluster
- Then trade off cluster penalties against average squared distance from centroid
- Choose the value of K with the best tradeoff □

Simple objective function for K: Formalization

- Given a clustering, define the cost for a document as (squared) distance to centroid
- Define total distortion RSS(K) as sum of all individual document costs (corresponds to average distance)
- Then: penalize each cluster with a cost λ
- Thus for a clustering with K clusters, total cluster penalty is $K\lambda$
- Define the total cost of a clustering as distortion plus total cluster penalty: RSS(K) $+ K\lambda$
- Select K that minimizes (RSS(K) $+ K\lambda$)
- Still need to determine good value for λ . . . □

Finding the "knee" in the curve



Pick the number of clusters where curve "flattens". Here: 4 or 9.

□

Take-away today

- What is clustering?
- Applications of clustering in information retrieval
- *K*-means algorithm
- Evaluation of clustering
- How many clusters? □

Resources

- Chapter 16 of IIR
- Resources at `https://www.fi.muni.cz/~sojka/PV211/` and `http://cislmu.org`, materials in MU IS and FI MU library
  - Keith van Rijsbergen on the cluster hypothesis (he was one of the originators)
  - Bing/Carrot2/Clusty: search result clustering systems
  - Stirling number: the number of distinct k-clusterings of n items

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 14: Vector Space Classification
Handout version

Petr Sojka, Hinrich Schütze et al.

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

2020-04-29

Overview

1. Intro vector space classification
2. Rocchio
3. kNN
4. Linear classifiers
5. $>$ two classes

Take-away today

- Vector space classification: Basic idea of doing text classification for documents that are represented as vectors
- Rocchio classifier: Rocchio relevance feedback idea applied to text classification
- k nearest neighbor classification
- Linear classifiers
- More than two classes

Roadmap for today

- Naive Bayes is simple and a good baseline.
- Use it if you want to get a text classifier up and running in a hurry.
- But other classification methods are more accurate.
- Perhaps the simplest well performing alternative: kNN
- kNN is a vector space classifier.
- Plan for rest of today
  1. intro vector space classification
  2. very simple vector space classification: Rocchio
  3. kNN
  4. general properties of classifiers

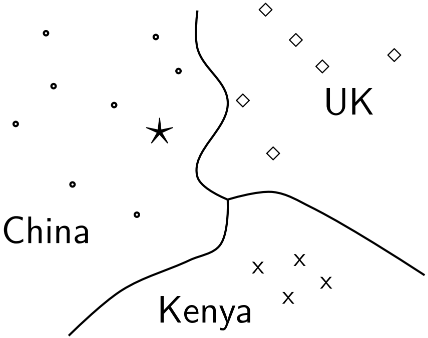
Recall vector space representation

- Each document is a vector, one component for each term.
- Terms are axes.
- High dimensionality: 100,000s of dimensions
- Normalize vectors (documents) to unit length
- How can we do classification in this space?

Vector space classification

- As before, the training set is a set of documents, each labeled with its class.
- In vector space classification, this set corresponds to a labeled set of points or vectors in the vector space.
- Premise 1: Documents in the same class form a contiguous region.
- Premise 2: Documents from different classes don't overlap.
- We define lines, surfaces, hyper-surfaces to divide regions.

Classes in the vector space



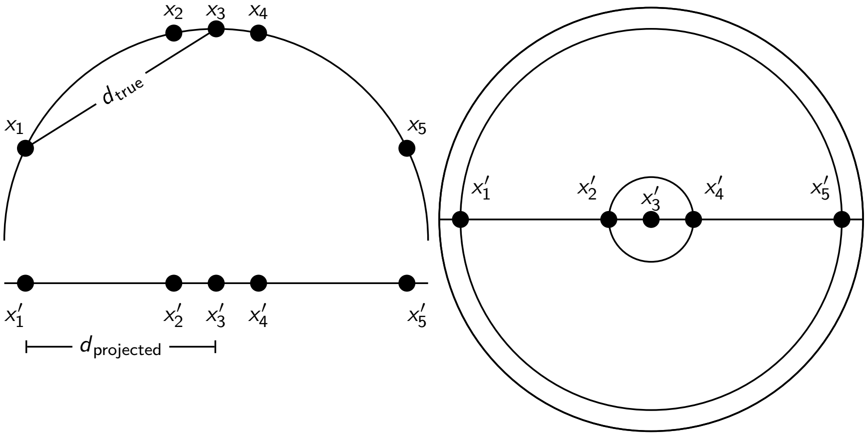
Should the document ⋆ be assigned to *China*, *UK* or *Kenya*?

Find separators between the classes

Based on these separators: ⋆ should be assigned to *China*

How do we find separators that do a good job at classifying new documents like ⋆? – Main topic of today

Aside: 2D/3D graphs can be misleading



Left: A projection of the 2D semicircle to 1D. For the points x_1, x_2, x_3, x_4, x_5 at x coordinates $-0.9, -0.2, 0, 0.2, 0.9$ the distance $|x_2x_3| \approx 0.201$ only differs by 0.5% from $|x'_2x'_3| = 0.2$; but $|x_1x_3|/|x'_1x'_3| = d_{\text{true}}/d_{\text{projected}} \approx 1.06/0.9 \approx 1.18$ is an example of a large distortion (18%) when projecting a large area. *Right:* The corresponding projection of the 3D hemisphere to 2D.

Relevance feedback

- In relevance feedback, the user marks documents as relevant/non-relevant.
- Relevant/non-relevant can be viewed as classes or categories.
- For each document, the user decides which of these two classes is correct.
- The IR system then uses these class assignments to build a better query ("model") of the information need ...
- ... and returns better documents.
- Relevance feedback is a form of text classification.
- The notion of text classification (TC) is very general and has many applications within and beyond information retrieval.

Using Rocchio for vector space classification

- The principal difference between relevance feedback and text classification:
  - The training set is given as part of the input in text classification.
  - It is interactively created in relevance feedback.

Rocchio classification: Basic idea

- Compute a centroid for each class
  - The centroid is the average of all documents in the class.
- Assign each test document to the class of its closest centroid.

Recall definition of centroid

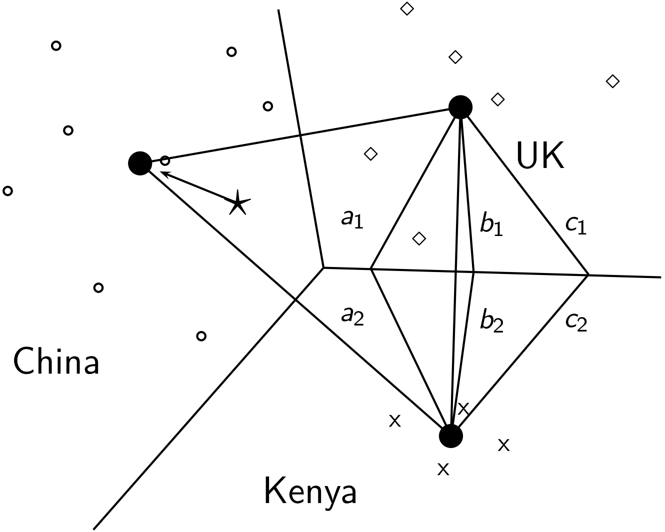
$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d)$$

where D_c is the set of all documents that belong to class c and $\vec{v}(d)$ is the vector space representation of d.

Rocchio algorithm

```
TrainRocchio($\mathbb{C}, \mathbb{D}$)
for each $c_j \in \mathbb{C}$
do $D_j \leftarrow \{d : \langle d, c_j \rangle \in \mathbb{D}\}$
     $\vec{\mu}_j \leftarrow \frac{1}{|D_j|} \sum_{d \in D_j} \vec{v}(d)$
return $\{\vec{\mu}_1, \ldots, \vec{\mu}_J\}$

ApplyRocchio($\{\vec{\mu}_1, \ldots, \vec{\mu}_J\}, d$)
return $\arg\min_j |\vec{\mu}_j - \vec{v}(d)|$
```

Rocchio illustrated: $a_1 = a_2, b_1 = b_2, c_1 = c_2$



Rocchio properties

- Rocchio forms a simple representation for each class: the centroid
  - We can interpret the centroid as the prototype of the class.
- Classification is based on similarity to / distance from centroid/prototype.
- Does not guarantee that classifications are consistent with the training data!

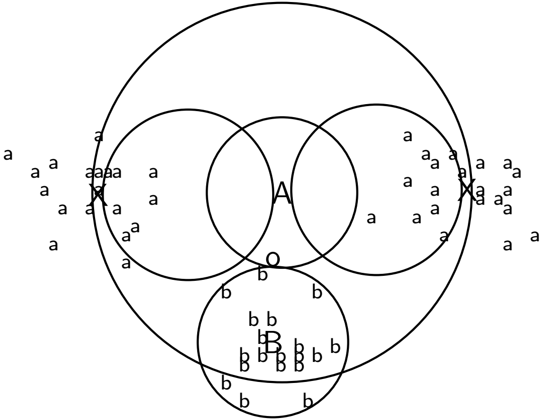
Time complexity of Rocchio

mode	time complexity
training	$\Theta(\lvert\mathbb{D}\rvert L_{\text{ave}} + \lvert\mathbb{C}\rvert\lvert V\rvert) \approx \Theta(\lvert\mathbb{D}\rvert L_{\text{ave}})$
testing	$\Theta(L_{\text{a}} + \lvert\mathbb{C}\rvert M_{\text{a}}) \approx \Theta(\lvert\mathbb{C}\rvert M_{\text{a}})$

Rocchio vs. Naive Bayes

- In many cases, Rocchio performs worse than Naive Bayes.
- One reason: Rocchio does not handle nonconvex, multimodal classes correctly.

Rocchio cannot handle nonconvex, multimodal classes



Exercise: Why is Rocchio not expected to do well for the classification task a vs. b here?

- A is centroid of the a's, B is centroid of the b's.
- The point o is closer to A than to B.
- But o is a better fit for the b class.
- A is a multimodal class with two prototypes.
- But in Rocchio we only have one prototype.

kNN classification

- kNN classification is another vector space classification method.
- It also is very simple and easy to implement.
- kNN is more accurate (in most cases) than Naive Bayes and Rocchio.
- If you need to get a pretty accurate classifier up and running in a short time . . .
- . . . and you don’t care about efficiency that much . . .
- . . . use kNN.

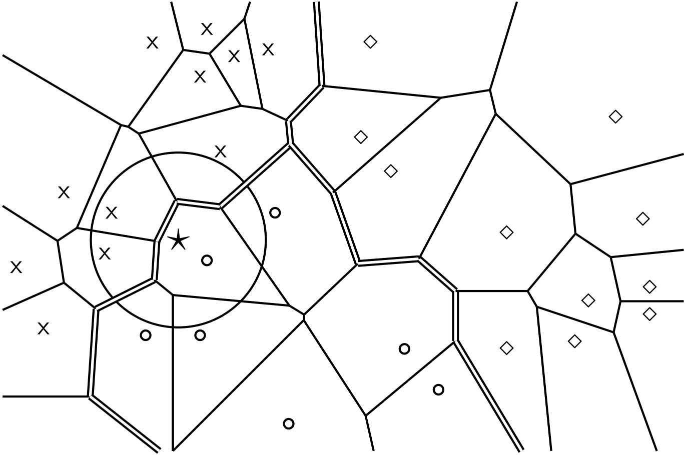
kNN classification

- kNN $= k$ nearest neighbors
- kNN classification rule for $k = 1$ (1NN): Assign each test document to the class of its nearest neighbor in the training set.
- 1NN is not very robust – one document can be mislabeled or atypical.
- kNN classification rule for $k > 1$ (kNN): Assign each test document to the majority class of its k nearest neighbors in the training set.
- Rationale of kNN: contiguity hypothesis
  - We expect a test document d to have the same label as the training documents located in the local region surrounding d.

Probabilistic kNN

- Probabilistic version of kNN: $P(c|d) =$ fraction of k neighbors of d that are in c
- kNN classification rule for probabilistic kNN: Assign d to class c with highest $P(c|d)$

kNN is based on Voronoi tessellation

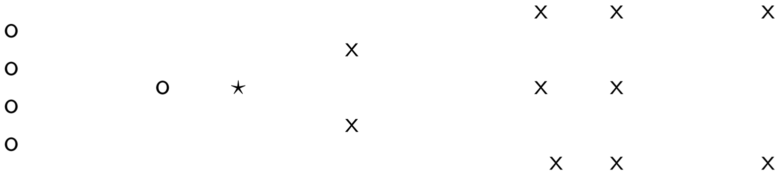


1NN, 3NN classification decision for star?

kNN algorithm

```
$\textsc{Train-kNN}(\mathbb{C}, \mathbb{D})$
$\mathbb{D}' \leftarrow \textsc{Preprocess}(\mathbb{D})$
$k \leftarrow \textsc{Select-k}(\mathbb{C}, \mathbb{D}')$
return $\mathbb{D}', k$

$\textsc{Apply-kNN}(\mathbb{D}', k, d)$
$S_k \leftarrow \textsc{ComputeNearestNeighbors}(\mathbb{D}', k, d)$
for each $c_j \in \mathbb{C}(\mathbb{D}')$
do $p_j \leftarrow |S_k \cap c_j|/k$
return $\arg\max_j p_j$
```

Exercise



How is star classified by:
(i) 1-NN (ii) 3-NN (iii) 9-NN (iv) 15-NN (v) Rocchio?

Time complexity of kNN

kNN with preprocessing of training set

training	$\Theta(\lvert\mathbb{D}\rvert L_{\mathrm{ave}})$
testing	$\Theta(L_{\mathrm{a}} + \lvert\mathbb{D}\rvert M_{\mathrm{ave}} M_{\mathrm{a}}) = \Theta(\lvert\mathbb{D}\rvert M_{\mathrm{ave}} M_{\mathrm{a}})$

- kNN test time proportional to the size of the training set!
- The larger the training set, the longer it takes to classify a test document.
- kNN is inefficient for very large training sets.
- Question: Can we divide up the training set into regions, so that we only have to search in one region to do kNN classification for a given test document? (which perhaps would give us better than linear time complexity)

Curse of dimensionality

- Our intuitions about space are based on the 3D world we live in.
- Intuition 1: some things are close by, some things are distant.
- Intuition 2: we can carve up space into areas such that: within an area things are close, distances between areas are large.
- These two intuitions don't necessarily hold for high dimensions.
- In particular: for a set of k uniformly distributed points, let dmin be the smallest distance between any two points and dmax be the largest distance between any two points.
- Then

$$\lim_{d \to \infty} \frac{\mathrm{dmax} - \mathrm{dmin}}{\mathrm{dmin}} = 0$$

Curse of dimensionality: Simulation

- Simulate

$$\lim_{d \to \infty} \frac{\text{dmax} - \text{dmin}}{\text{dmin}} = 0$$

- Pick a dimensionality d
- Generate 10 random points in the d-dimensional hypercube (uniform distribution)
- Compute all 45 distances
- Compute $\frac{\text{dmax}-\text{dmin}}{\text{dmin}}$
- We see that intuition 1 (some things are close, others are distant) is not true for high dimensions.

Intuition 2: Space can be carved up

- Intuition 2: we can carve up space into areas such that: within an area things are close, distances between areas are large.
- If this is true, then we have a simple and efficient algorithm for kNN.
- To find the k closest neighbors of data point $< x_1, x_2, \ldots, x_d >$ do the following.
- Using binary search find all data points whose first dimension is in $[x_1 - \epsilon, x_1 + \epsilon]$. This is $O(\log n)$ where n is the number of data points.
- Do this for each dimension, then intersect the d subsets.

Intuition 2: Space can be carved up

- Size of data set $n = 100$
- Again, assume uniform distribution in hypercube
- Set $\epsilon = 0.05$: we will look in an interval of length 0.1 for neighbors on each dimension.
- What is the probability that the nearest neighbor of a new data point $\vec{x}$ is in this neighborhood in $d = 1$ dimension?
- for $d = 1$: $1 - (1 - 0.1)^{100} \approx 0.99997$
- In $d = 2$ dimensions?
- for $d = 2$: $1 - (1 - 0.1^2)^{100} \approx 0.63$
- In $d = 3$ dimensions?
- for $d = 3$: $1 - (1 - 0.1^3)^{100} \approx 0.095$
- In $d = 4$ dimensions?
- for $d = 4$: $1 - (1 - 0.1^4)^{100} \approx 0.0095$
- In $d = 5$ dimensions?
- for $d = 5$: $1 - (1 - 0.1^5)^{100} \approx 0.0009995$

Intuition 2: Space can be carved up

- In $d = 5$ dimensions?
- for $d = 5$: $1 - (1 - 0.1^5)^{100} \approx 0.0009995$
- In other words: with enough dimensions, there is only one "local" region that will contain the nearest neighbor with high certainty: the entire search space.
- We cannot carve up high-dimensional space into neat neighborhoods . . .
- . . . unless the "true" dimensionality is much lower than d.

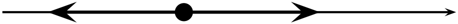
kNN: Discussion

- No training necessary
  - But linear preprocessing of documents is as expensive as training Naive Bayes.
  - We always preprocess the training set, so in reality training time of kNN is linear.
- kNN is very accurate if training set is large.
- Optimality result: asymptotically zero error if Bayes rate is zero.
- But kNN can be very inaccurate if training set is small.

Linear classifiers

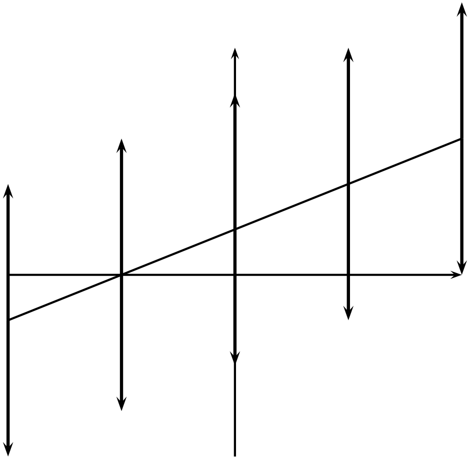
- Definition:
  - A linear classifier computes a linear combination or weighted sum $\sum_i w_i x_i$ of the feature values.
  - Classification decision: $\sum_i w_i x_i > \theta$?
  - . . . where θ (the threshold) is a parameter.
- (First, we only consider binary classifiers.)
- Geometrically, this corresponds to a line (2D), a plane (3D) or a hyperplane (higher dimensionalities), the separator.
- We find this separator based on training set.
- Methods for finding separator: Perceptron, Rocchio, Naive Bayes – as we will explain on the next slides
- Assumption: The classes are linearly separable.

A linear classifier in 1D



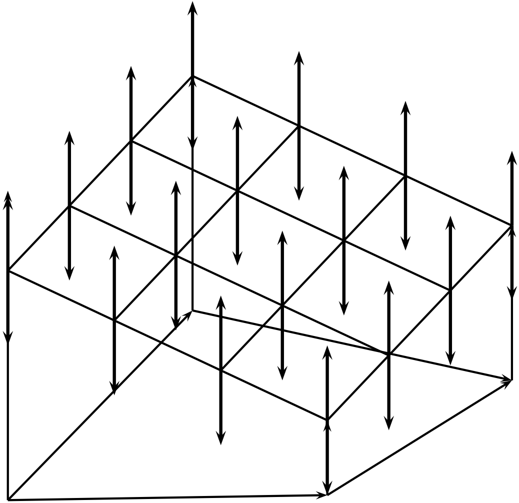
- A linear classifier in 1D is a point described by the equation $w_1 d_1 = \theta$
- The point at θ / w_1
- Points (d_1) with $w_1 d_1 \geq \theta$ are in the class c.
- Points (d_1) with $w_1 d_1 < \theta$ are in the complement class $\overline{c}$.

A linear classifier in 2D



- A linear classifier in 2D is a line described by the equation $w_1 d_1 + w_2 d_2 = \theta$
- Example for a 2D linear classifier
- Points $(d_1 \ d_2)$ with $w_1 d_1 + w_2 d_2 \geq \theta$ are in the class c.
- Points $(d_1 \ d_2)$ with $w_1 d_1 + w_2 d_2 < \theta$ are in the complement class $\overline{c}$.

A linear classifier in 3D



- A linear classifier in 3D is a plane described by the equation $w_1 d_1 + w_2 d_2 + w_3 d_3 = \theta$
- Example for a 3D linear classifier
- Points $(d_1 \ d_2 \ d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 \geq \theta$ are in the class c.
- Points $(d_1 \ d_2 \ d_3)$ with $w_1 d_1 + w_2 d_2 + w_3 d_3 < \theta$ are in the complement class $\overline{c}$.

Rocchio as a linear classifier

- Rocchio is a linear classifier defined by:

$$\sum_{i=1}^{M} w_i d_i = \vec{w}\vec{d} = \theta$$

where $\vec{w}$ is the normal vector $\vec{\mu}(c_1) - \vec{\mu}(c_2)$ and $\theta = 0.5 * (|\vec{\mu}(c_1)|^2 - |\vec{\mu}(c_2)|^2)$.

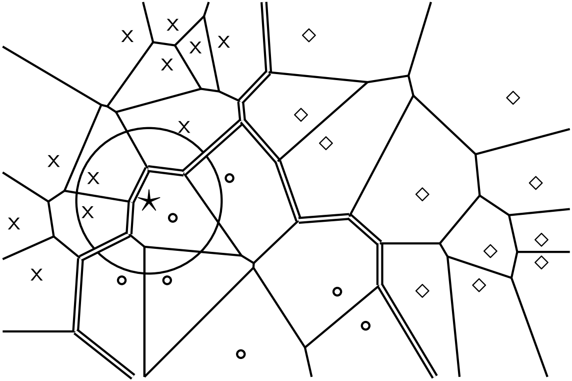
Naive Bayes as a linear classifier

Multinomial Naive Bayes is a linear classifier (in log space) defined by:

$$\sum_{i=1}^{M} w_i d_i = \theta$$

where $w_i = \log[\hat{P}(t_i|c)/\hat{P}(t_i|\bar{c})]$, d_i = number of occurrences of t_i in d, and $\theta = -\log[\hat{P}(c)/\hat{P}(\bar{c})]$. Here, the index i, $1 \leq i \leq M$, refers to terms of the vocabulary (not to positions in d as k did in our original definition of Naive Bayes)

kNN is not a linear classifier



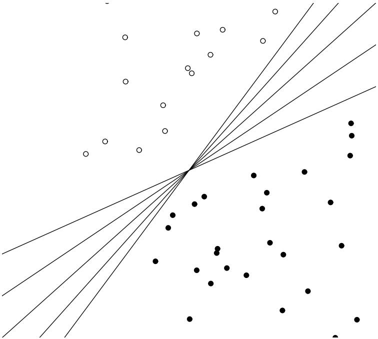
- Classification decision based on majority of k nearest neighbors.
- The decision boundaries between classes are piecewise linear . . .
- . . . but they are in general not linear classifiers that can be described as $\sum_{i=1}^{M} w_i d_i = \theta$.

Example of a linear two-class classifier

t_i	w_i	d_{1i}	d_{2i}	t_i	w_i	d_{1i}	d_{2i}
prime	0.70	0	1	dlrs	-0.71	1	1
rate	0.67	1	0	world	-0.35	1	0
interest	0.63	0	0	sees	-0.33	0	0
rates	0.60	0	0	year	-0.25	0	0
discount	0.46	1	0	group	-0.24	0	0
bundesbank	0.43	0	0	dlr	-0.24	0	0

- This is for the class *interest* in Reuters-21578.
- For simplicity: assume a simple 0/1 vector representation
- d_1: “rate discount dlrs world”
- d_2: “prime dlrs”
- $\theta = 0$
- Exercise: Which class is d_1 assigned to? Which class is d_2 assigned to?
- We assign document $\vec{d}_1$ “rate discount dlrs world” to *interest* since $\vec{w}^T\vec{d}_1 = 0.67 \cdot 1 + 0.46 \cdot 1 + (-0.71) \cdot 1 + (-0.35) \cdot 1 = 0.07 > 0 = \theta$.
- We assign $\vec{d}_2$ “prime dlrs” to the complement class (not in *interest*) since $\vec{w}^T\vec{d}_2 = -0.01 \leq \theta$.

Which hyperplane?



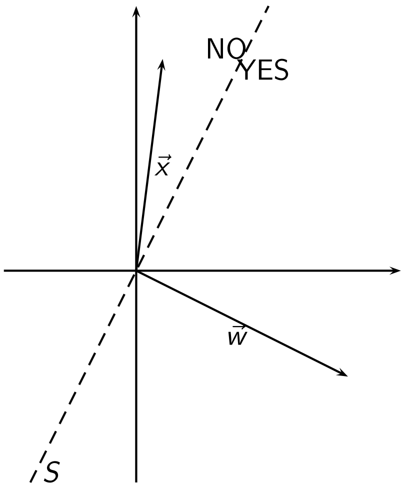
Learning algorithms for vector space classification

- In terms of actual computation, there are two types of learning algorithms.
- (i) Simple learning algorithms that estimate the parameters of the classifier directly from the training data, often in one linear pass.
    - Naive Bayes, Rocchio, kNN are all examples of this.
- (ii) Iterative algorithms
    - Support vector machines
    - Perceptron (example available as PDF on website: `http://cislmu.org`)
- The best performing learning algorithms usually require iterative learning.

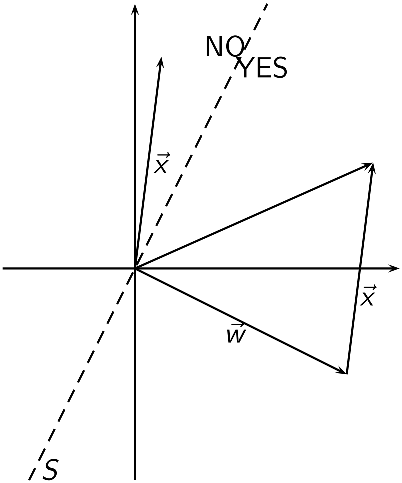
Perceptron update rule

- Randomly initialize linear separator $\vec{w}$
- Do until convergence:
    - Pick data point $\vec{x}$
    - If $\text{sign}(\vec{w}^T\vec{x})$ is correct class (1 or -1): do nothing
    - Otherwise: $\vec{w} = \vec{w} - \text{sign}(\vec{w}^T\vec{x})\vec{x}$

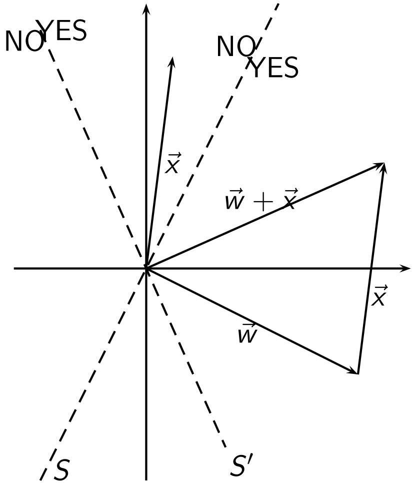
Perceptron



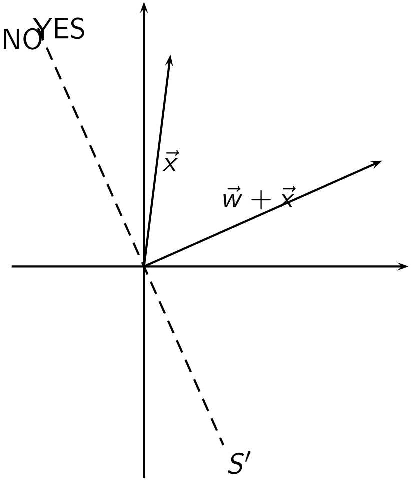
Perceptron



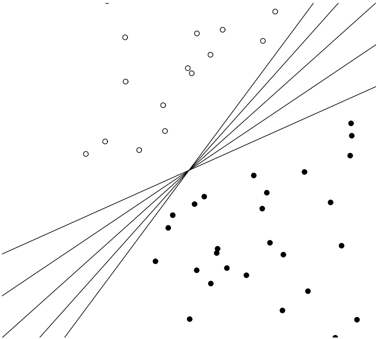
Perceptron



Perceptron



Which hyperplane?



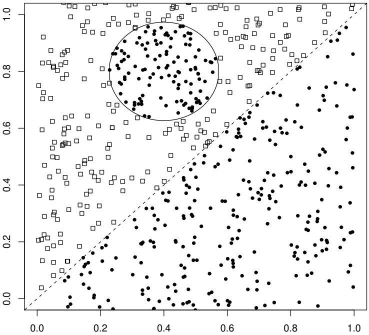
Which hyperplane?

- For linearly separable training sets: there are infinitely many separating hyperplanes.
- They all separate the training set perfectly . . .
- . . . but they behave differently on test data.
- Error rates on new data are low for some, high for others.
- How do we find a low-error separator?
- Perceptron: generally bad; Naive Bayes, Rocchio: ok; linear SVM: good

Linear classifiers: Discussion

- Many common text classifiers are linear classifiers: Naive Bayes, Rocchio, logistic regression, linear support vector machines etc.
- Each method has a different way of selecting the separating hyperplane
    - Huge differences in performance on test documents
- Can we get better performance with more powerful nonlinear classifiers?
- Not in general: A given amount of training data may suffice for estimating a linear boundary, but not for estimating a more complex nonlinear boundary.

A nonlinear problem

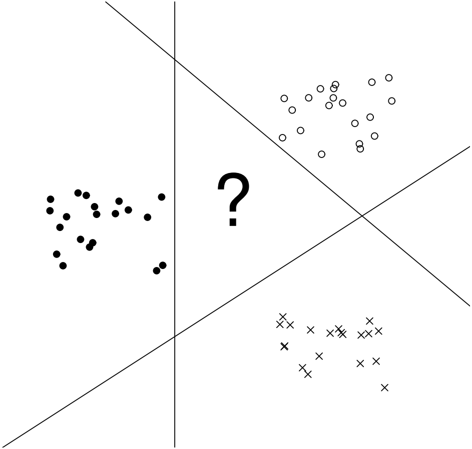


- Linear classifier like Rocchio does badly on this task.
- kNN will do well (assuming enough training data)

Which classifier do I use for a given TC problem?

- Is there a learning method that is optimal for all text classification problems?
- No, because there is a trade-off between bias and variance.
- Factors to take into account:
  - How much training data is available?
  - How simple/complex is the problem? (linear vs. nonlinear decision boundary)
  - How noisy is the problem?
  - How stable is the problem over time?
    - For an unstable problem, it's better to use a simple and robust classifier.

How to combine hyperplanes for > 2 classes?



One-of problems

- One-of or multiclass classification
  - Classes are mutually exclusive.
  - Each document belongs to exactly one class.
  - Example: language of a document (assumption: no document contains multiple languages)

One-of classification with linear classifiers

- Combine two-class linear classifiers as follows for one-of classification:
  - Run each classifier separately
  - Rank classifiers (e.g., according to score)
  - Pick the class with the highest score

Any-of problems

- Any-of or multilabel classification
  - A document can be a member of 0, 1, or many classes.
  - A decision on one class leaves decisions open on all other classes.
  - A type of “independence” (but not statistical independence)
  - Example: topic classification
  - Usually: make decisions on the region, on the subject area, on the industry and so on “independently”

Any-of classification with linear classifiers

- Combine two-class linear classifiers as follows for any-of classification:
  - Simply run each two-class classifier separately on the test document and assign document accordingly

Take-away today

- Vector space classification: Basic idea of doing text classification for documents that are represented as vectors
- Rocchio classifier: Rocchio relevance feedback idea applied to text classification
- k nearest neighbor classification
- Linear classifiers
- More than two classes

Resources

- Chapter 13 of IIR (feature selection)
- Chapter 14 of IIR
- Resources at `http://cislmu.org`
  - Perceptron example
  - General overview of text classification: Sebastiani (2002)
  - Text classification chapter on decision tress and perceptrons: Manning & Schütze (1999)
  - One of the best machine learning textbooks: Hastie, Tibshirani & Friedman (2003)

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 18: Latent Semantic Indexing
Handout version

Petr Sojka, Hinrich Schütze et al.

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

2020-05-06

Overview

1. Latent semantic indexing
2. Dimensionality reduction
3. LSI in information retrieval
4. Clustering

Take-away today

- Latent Semantic Indexing (LSI) / Singular Value Decomposition: The math
- SVD used for dimensionality reduction
- LSI: SVD in information retrieval
- LSI as clustering
- gensim: Topic modelling for humans (practical use of LSI etal.)

Recall: Term-document matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
anthony	5.25	3.18	0.0	0.0	0.0	0.35
brutus	1.21	6.10	0.0	1.0	0.0	0.0
caesar	8.59	2.54	0.0	1.51	0.25	0.0
calpurnia	0.0	1.54	0.0	0.0	0.0	0.0
cleopatra	2.85	0.0	0.0	0.0	0.0	0.0
mercy	1.51	0.0	1.90	0.12	5.25	0.88
worser	1.37	0.0	0.11	4.15	0.25	1.95
...						

This matrix is the basis for computing the similarity between documents and queries.

Today: Can we transform this matrix, so that we get a better measure of similarity between documents and queries? □

Latent semantic indexing: Overview

- We will decompose the term-document matrix into a product of matrices.
- The particular decomposition we'll use: singular value decomposition (SVD).
- SVD: $C = U\Sigma V^T$ (where C = term-document matrix)
- We will then use the SVD to compute a new, improved term-document matrix C'.
- We'll get better similarity values out of C' (compared to C).
- Using SVD for this purpose is called latent semantic indexing or LSI. □

Example of $C = U\Sigma V^T$: The matrix C

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

This is a standard term-document matrix.

Actually, we use a non-weighted matrix here to simplify the example. □

Example of $C = U\Sigma V^T$: The matrix U

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

One row per term, one column per $\min(M, N)$ where M is the number of terms and N is the number of documents.

This is an orthonormal matrix: (i) Row vectors have unit length. (ii) Any two distinct row vectors are orthogonal to each other.

Think of the dimensions as "semantic" dimensions that capture distinct topics like politics, sports, economics. 2 = land/water

Each number u_{ij} in the matrix indicates how strongly related term i is to the topic represented by semantic dimension j. □

Example of $C = U\Sigma V^T$: The matrix Σ

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

This is a square, diagonal matrix of dimensionality $\min(M, N) \times \min(M, N)$.

The diagonal consists of the singular values of C.

The magnitude of the singular value measures the importance of the corresponding semantic dimension.

We'll make use of this by omitting unimportant dimensions. □

Example of $C = U\Sigma V^T$: The matrix V^T

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

One column per document, one row per $\min(M, N)$ where M is the number of terms and N is the number of documents.

Again: This is an orthonormal matrix: (i) Column vectors have unit length. (ii) Any two distinct column vectors are orthogonal to each other.

These are again the semantic dimensions from matrices U and Σ that capture distinct topics like politics, sports, economics.

Each number v_{ij} in the matrix indicates how strongly related document i is to the topic represented by semantic dimension

Example of $C = U\Sigma V^T$: All four matrices Recall unreduced decomposition $C = U\Sigma V^T$ Exercise: Why can this be viewed as soft clustering?

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

=

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

×

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

×

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

LSI is decomposition of C into a representation of the terms, a representation of the documents and a representation of the importance of the “semantic” dimensions. □

LSI: Summary

- We've decomposed the term-document matrix C into a product of three matrices: $U\Sigma V^T$.
- The term matrix U – consists of one (row) vector for each term
- The document matrix V^T – consists of one (column) vector for each document
- The singular value matrix Σ – diagonal matrix with singular values, reflecting importance of each dimension
- Next: Why are we doing this? □

Exercise

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	-0.75	-0.28	-0.20	-0.45	-0.33	-0.12
2	-0.29	-0.53	-0.19	0.63	0.22	0.41
3	0.28	-0.75	0.45	-0.20	0.12	-0.33
4	0.00	0.00	0.58	0.00	-0.58	0.58
5	-0.53	0.29	0.63	0.19	0.41	-0.22

Verify that the first document has unit length.

Verify that the first two documents are orthogonal.

$$0.75^2 + 0.29^2 + 0.28^2 + 0.00^2 + 0.53^2 = 1.0059$$

$$-0.75 * -0.28 + -0.29 * -0.53 + 0.28 * -0.75 + 0.00 * 0.00 + -0.53 * 0.29 = 0$$

How we use the SVD in LSI

- Key property: Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the "details".
- These details may
  - be noise – in that case, reduced LSI is a better representation because it is less noisy.
  - make things dissimilar that should be similar – again, the reduced LSI representation is a better representation because it represents similarity better.
- Analogy for "fewer details is better"
  - Image of a blue flower
  - Image of a yellow flower
  - Omitting color makes it easier to see the similarity □

Reducing the dimensionality to 2

U	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

Σ_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

Actually, we only zero out singular values in Σ. This has the effect of setting the corresponding dimensions in U and V^T to zero when computing the product $C = U\Sigma V^T$. □

Reducing the dimensionality to 2

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

=

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

×

Σ_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

×

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

□

Example of $C = U\Sigma V^T$: All four matrices Recall unreduced decomposition $C = U\Sigma V^T$ Exercise: Why can this be viewed as soft clustering?

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

=

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

×

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

×

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

LSI is decomposition of C into a representation of the terms, a representation of the documents and a representation of the importance of the “semantic” dimensions. □

Original matrix C vs. reduced $C_2 = U\Sigma_2 V^T$

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

We can view C_2 as a two-dimensional representation of the matrix C. We have performed a dimensionality reduction to two dimensions. □

Exercise

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

Compute the similarity between d_2 and d_3 for the original matrix and for the reduced matrix.

Why the reduced matrix C_2 is better than C

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

Similarity of d_2 and d_3 in the original space: 0.

Similarity of d_2 and d_3 in the reduced space: $0.52 * 0.28 + 0.36 * 0.16 + 0.72 * 0.36 + 0.12 * 0.20 + -0.39 * -0.08 \approx 0.52$

"boat" and "ship" are semantically similar. The "reduced" similarity measure reflects this.

What property of the SVD reduction is responsible for improved similarity? □

Exercise: Compute matrix product

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.09	0.16	0.06	-0.19	-0.07	-0.12
boat	0.10	0.17	0.06	-0.21	-0.07	-0.14
ocean	0.15	0.27	0.10	-0.32	-0.11	-0.21
wood	-0.10	-0.19	-0.07	0.22	0.08	0.14
tree	-0.19	-0.34	-0.12	0.41	0.14	0.27

???????=

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

×

Σ_2	1	2	3	4	5
1	0.00	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

×

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

Why we use LSI in information retrieval

- LSI takes documents that are semantically similar (= talk about the same topics), . . .
- . . . but are not similar in the vector space (because they use different words) . . .
- . . . and re-represents them in a reduced vector space . . .
- . . . in which they have higher similarity.
- Thus, LSI addresses the problems of synonymy and semantic relatedness.
- Standard vector space: Synonyms contribute nothing to document similarity.
- Desired effect of LSI: Synonyms contribute strongly to document similarity. □

How LSI addresses synonymy and semantic relatedness

- The dimensionality reduction forces us to omit a lot of “detail”.
- We have to map differents words (= different dimensions of the full space) to the same dimension in the reduced space.
- The “cost” of mapping synonyms to the same dimension is much less than the cost of collapsing unrelated words.
- SVD selects the “least costly” mapping (see below).
- Thus, it will map synonyms to the same dimension.
- But it will avoid doing that for unrelated words. □

LSI: Comparison to other approaches

- Recap: Relevance feedback and query expansion are used to increase recall in information retrieval – if query and documents have no terms in common.
  - (or, more commonly, too few terms in common for a high similarity score)
- LSI increases recall and hurts precision.
- Thus, it addresses the same problems as (pseudo) relevance feedback and query expansion . . .
- . . . and it has the same problems. □

Implementation

- Compute SVD of term-document matrix
- Reduce the space and compute reduced document representations
- Map the query into the reduced space $\vec{q}_k = \Sigma_k^{-1} U_k^T \vec{q}$.
- This follows from: $C_k = U_k \Sigma_k V_k^T \Rightarrow \Sigma_k^{-1} U^T C = V_k^T$
- Compute similarity of q_k with all reduced documents in V_k.
- Output ranked list of documents as usual
- Exercise: What is the fundamental problem with this approach? □

Optimality

- SVD is optimal in the following sense.
- Keeping the k largest singular values and setting all others to zero gives you the optimal approximation of the original matrix C. Eckart-Young theorem
- Optimal: no other matrix of the same rank (= with the same underlying dimensionality) approximates C better.
- Measure of approximation is Frobenius norm: $||C||_F = \sqrt{\sum_i \sum_j c_{ij}^2}$
- So LSI uses the "best possible" matrix.
- There is only one best possible matrix – unique solution (modulo signs).
- Caveat: There is only a tenuous relationship between the Frobenius norm and cosine similarity between documents. □

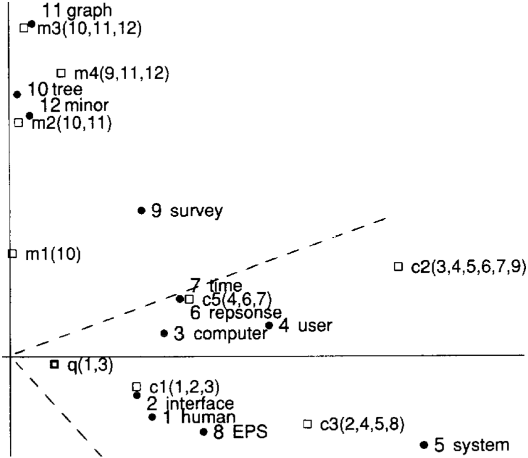
Data for graphical illustration of LSI

c_1 Human machine interface for lab abc computer applications
c_2 A survey of user opinion of computer system response time
c_3 The EPS user interface management system
c_4 System and human system engineering testing of EPS
c_5 Relation of user perceived response time to error measurement
m_1 The generation of random binary unordered trees
m_2 The intersection graph of paths in trees
m_3 Graph minors IV Widths of trees and well quasi ordering
m_4 Graph minors A survey

The matrix C

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Graphical illustration of LSI: Plot of C_2



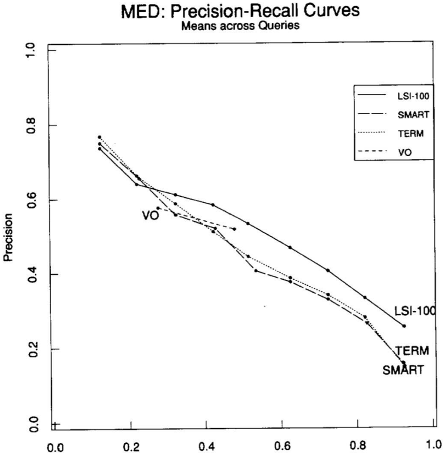
2-dimensional plot of C_2 (scaled dimensions). Circles = terms. Open squares = documents (component terms in parentheses). q = query "human computer interaction".

The dotted cone represents the region whose points are within a cosine of .9 from q . All documents about human-computer documents (c1-c5) are near q, even c3/c5 although they share no terms. None of the graph theory documents (m1-m4) are near q. □

Exercise

What happens when we rank documents according to cosine similarity in the original vector space? What happens when we rank documents according to cosine similarity in the reduced vector space?

LSI performs better than vector space on MED collection



LSI-100 = LSI reduced to 100 dimensions; SMART = SMART implementation of vector space model

Example of $C = U\Sigma V^T$: All four matrices Recall unreduced decomposition $C = U\Sigma V^T$ Exercise: Why can this be viewed as soft clustering?

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

=

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

×

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

×

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

LSI is decomposition of C into a representation of the terms, a representation of the documents and a representation of the importance of the “semantic” dimensions. □

Why LSI can be viewed as soft clustering

- Each of the k dimensions of the reduced space is one cluster.
- If the value of the LSI representation of document d on dimension k is x, then x is the soft membership of d in topic k.
- This soft membership can be positive or negative.
- Example: Dimension 2 in our SVD decomposition
- This dimension/cluster corresponds to the water/earth dichotomy.
- “ship”, “boat”, “ocean” have negative values.
- “wood”, “tree” have positive values.
- d_1, d_2, d_3 have negative values (most of their terms are water terms).
- d_4, d_5, d_6 have positive values (all of their terms are earth terms). □

Semantic indexing and clustering with Gensim

Gensim: an open-source vector space modeling and topic modeling toolkit, implemented in the Python programming language

Tutorial examples of topic modelling for humans (LSI):
`http://radimrehurek.com/gensim/tut2.html`

DML-CZ similarity example:
`http://dml.cz/handle/10338.dmlcz/500114/SimilarArticles`
cf. papers similar to famous Otakar Borůvka's paper

Go forth and create masterpieces for semantic indexing applications (by gensim, similarly as others already did ;-)!

Take-away today

- Latent Semantic Indexing (LSI) / Singular Value Decomposition: The math
- SVD used for dimensionality reduction
- LSI: SVD in information retrieval
- LSI as clustering
- gensim: Topic modelling for humans (practical use of LSI etal.)

Resources

- Chapter 18 of IIR
- Resources at `https://www.fi.muni.cz/~sojka/PV211/` and `http://cislmu.org`, materials in MU IS and FI MU library
  - Original paper on latent semantic indexing by Deerwester et al.
  - Paper on probabilistic LSI by Thomas Hofmann
  - Word space: LSI for words

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 19: Web search
Handout version

Petr Sojka, Hinrich Schütze et al.

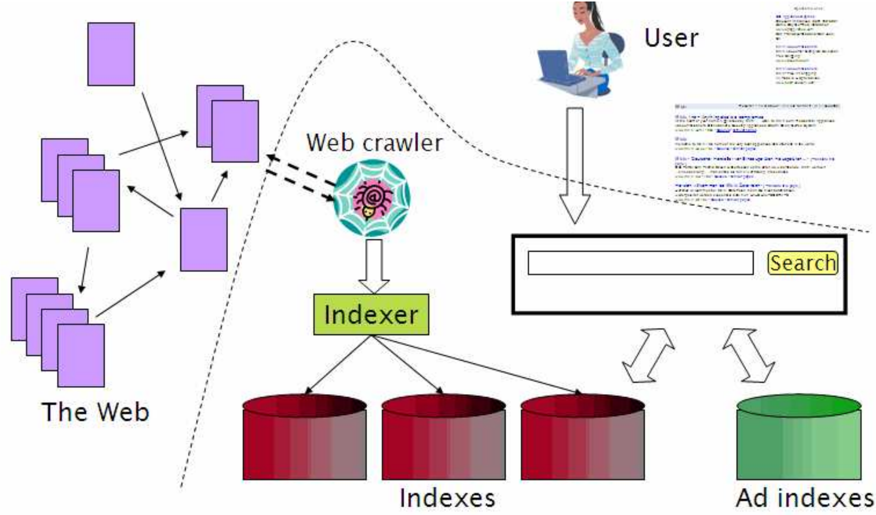
Faculty of Informatics, Masaryk University, Brno
Center for Information and Language Processing, University of Munich

2020-05-13

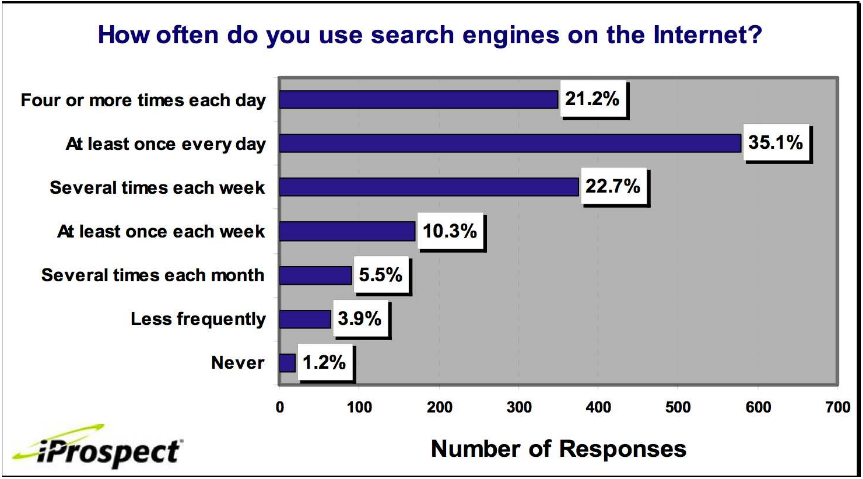
Overview

1. Big picture
2. Ads
3. Duplicate detection
4. Spam
5. Web IR
    - Queries
    - Links
    - Context
    - Users
    - Documents
    - Size
6. Size of the web

Web search overview



Search is a top activity on the web



Without search engines, the web wouldn't work

- Without search, content is hard to find.
- → Without search, there is no incentive to create content.
  - Why publish something if nobody will read it?
  - Why publish something if I don't get ad revenue from it?
- Somebody needs to pay for the web.
  - Servers, web infrastructure, content creation
  - A large part today is paid by search ads.
  - Search pays for the web. □

Interest aggregation

- Unique feature of the web: A small number of geographically dispersed people with similar interests can find each other.
  - Elementary school kids with hemophilia
  - People interested in translating R5 Scheme into relatively portable C (open source project)
  - Search engines are a key enabler for interest aggregation. □

IR on the web vs. IR in general

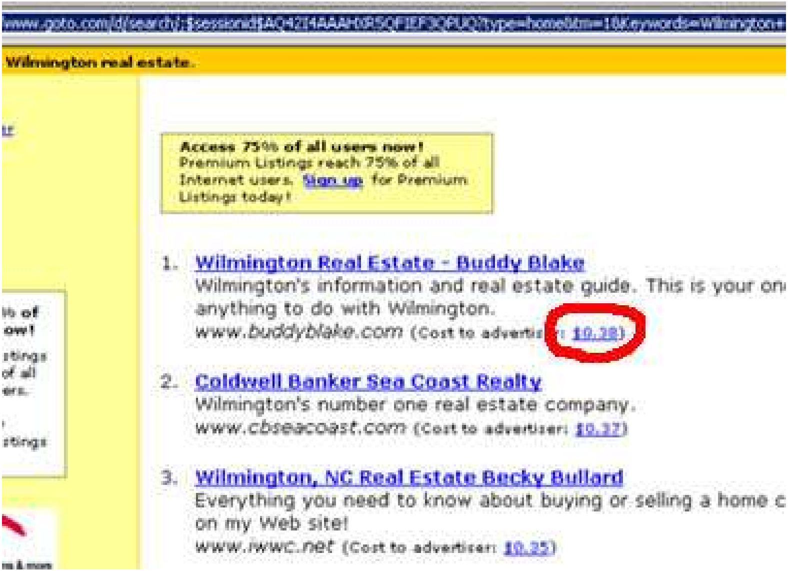
- On the web, search is not just a nice feature.
  - Search is a key enabler of the web: ...
  - ...financing, content creation, interest aggregation etc.

  → look at search ads
- The web is a chaotic and uncoordinated collection. → lots of duplicates – need to detect duplicates
- No control / restrictions on who can author content → lots of spam – need to detect spam
- The web is very large. → need to know how big it is □

Take-away today

- Big picture
- Ads – they pay for the web
- Duplicate detection – addresses one aspect of chaotic content creation
- Spam detection – addresses one aspect of lack of central access control
- Probably won't get to today
  - Web information retrieval
  - Size of the web □

First generation of search ads: Goto (1996)



www.goto.com/d/search/;$sessionid$AQ4214AAAH0R5QFIEF3QPUQ?type=home&tm=1&Keywords=Wilmington+

Wilmington real estate.

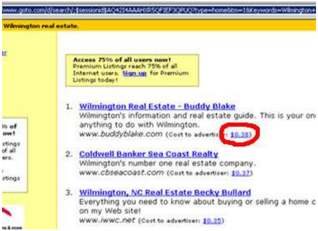
Access 75% of all users now!
Premium Listings reach 75% of all Internet users. Sign up for Premium Listings today!

1. Wilmington Real Estate - Buddy Blake
   Wilmington's information and real estate guide. This is your on
   anything to do with Wilmington.
   www.buddyblake.com (Cost to advertis : $0.38)

2. Coldwell Banker Sea Coast Realty
   Wilmington's number one real estate company.
   www.cbseacoast.com (Cost to advertiser: $0.37)

3. Wilmington, NC Real Estate Becky Bullard
   Everything you need to know about buying or selling a home c
   on my Web site!
   www.iwwc.net (Cost to advertiser: $0.25)

First generation of search ads: Goto (1996)



www.goto.com/d/search/;$sessionid$AQ42I4AAAHXRSQFIEF3QPUQ?type=home&tm=1&Keywords=Wilmington+

Wilmington real estate.

Access 75% of all users now!
Premium Listings reach 75% of all Internet users. Sign up for Premium Listings today!

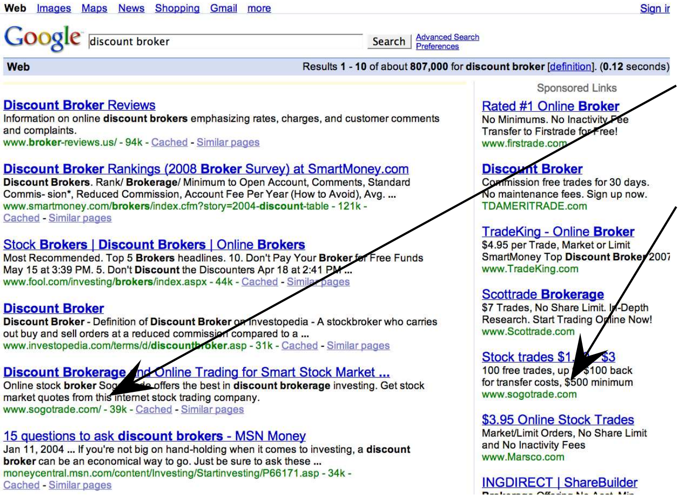
1. Wilmington Real Estate - Buddy Blake
   Wilmington's information and real estate guide. This is your on
   anything to do with Wilmington.
   www.buddyblake.com (Cost to advertis: $0.38)
2. Coldwell Banker Sea Coast Realty
   Wilmington's number one real estate company.
   www.cbseacoast.com (Cost to advertiser: $0.37)
3. Wilmington, NC Real Estate Becky Bullard
   Everything you need to know about buying or selling a home c
   on my Web site!
   www.iwwc.net (Cost to advertiser: $0.25)

- Buddy Blake bid the maximum ($0.38) for this search.
- He paid $0.38 to Goto every time somebody clicked on the link.
- Pages were simply ranked according to bid – revenue maximization for Goto.
- No separation of ads/docs. Only one result list!
- Upfront and honest. No relevance ranking, . . .
- . . . but Goto did not pretend there was any.

Second generation of search ads: Google (2000/2001)

- Strict separation of search results and search ads □

Two ranked lists: web pages (left) and ads (right)



Web Images Maps News Shopping Gmail more Sign in

Google [discount broker] Search Advanced Search Preferences

Web Results **1** - **10** of about **807,000** for **discount broker** [definition]. (**0.12** seconds)

Discount Broker Reviews
Information on online **discount brokers** emphasizing rates, charges, and customer comments and complaints.
www.**broker**-reviews.us/ - 94k - Cached - Similar pages

Discount Broker Rankings (2008 **Broker** Survey) at SmartMoney.com
Discount Brokers. Rank/ **Brokerage**/ Minimum to Open Account, Comments, Standard Commis- sion*, Reduced Commission, Account Fee Per Year (How to Avoid), Avg. ...
www.smartmoney.com/**brokers**/index.cfm?story=2004-**discount**-table - 121k - Cached - Similar pages

Stock **Brokers** | **Discount Brokers** | Online **Brokers**
Most Recommended. Top 5 **Brokers** headlines. 10. Don't Pay Your **Broker** for Free Funds May 15 at 3:39 PM. 5. Don't **Discount** the Discounters Apr 18 at 2:41 PM ...
www.fool.com/investing/**brokers**/index.aspx - 44k - Cached - Similar pages

Discount Broker
Discount Broker - Definition of **Discount Broker** on Investopedia - A stockbroker who carries out buy and sell orders at a reduced commission compared to a ...
www.investopedia.com/terms/d/**discountbroker**.asp - 31k - Cached - Similar pages

Discount Brokerage and Online Trading for Smart Stock Market ...
Online stock **broker** Sog[illegible] offers the best in **discount brokerage** investing. Get stock market quotes from this internet stock trading company.
www.sogotrade.com/ - 39k - Cached - Similar pages

15 questions to ask **discount brokers** - MSN Money
Jan 11, 2004 ... If you're not big on hand-holding when it comes to investing, a **discount broker** can be an economical way to go. Just be sure to ask these ...
moneycentral.msn.com/content/Investing/Startinvesting/P66171.asp - 34k - Cached - Similar pages

Sponsored Links

Rated #1 Online **Broker**
No Minimums. No Inactivity Fee
Transfer to Firstrade for Free!
www.firstrade.com

Discount Broker
Commission free trades for 30 days.
No maintenance fees. Sign up now.
TDAMERITRADE.com

TradeKing - Online **Broker**
$4.95 per Trade, Market or Limit
SmartMoney Top **Discount Broker** 2007
www.TradeKing.com

Scottrade **Brokerage**
$7 Trades, No Share Limit. In-Depth
Research. Start Trading Online Now!
www.Scottrade.com

Stock trades $1.[illegible] $3
100 free trades, up to $100 back
for transfer costs, $500 minimum
www.sogotrade.com

$3.95 Online Stock Trades
Market/Limit Orders, No Share Limit
and No Inactivity Fees
www.Marsco.com

INGDIRECT | ShareBuilder
Brokerage Offering No Acct Min

SogoTrade appears in search results.

SogoTrade appears in ads.

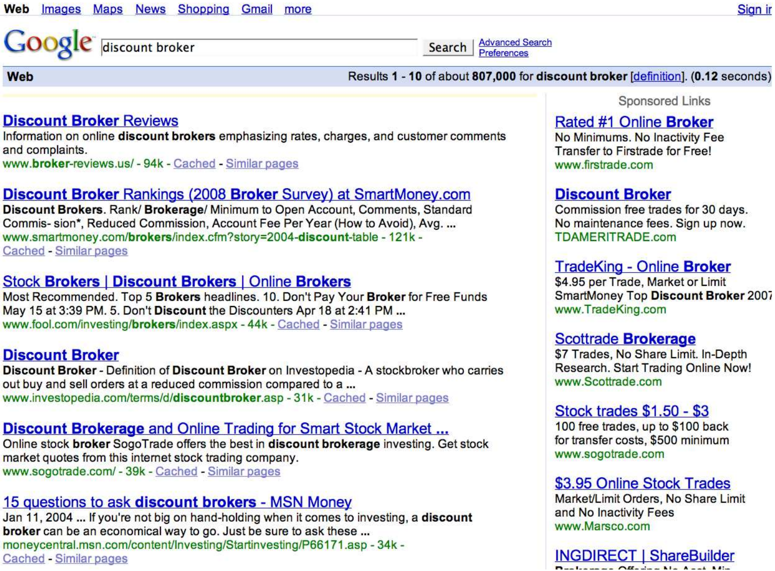
Do search engines rank advertisers higher than non-advertisers?

All major search engines claim no.

Do ads influence editorial content?

- Similar problem at newspapers / TV channels
- A newspaper is reluctant to publish harsh criticism of its major advertisers.
- The line often gets blurred at newspapers / on TV.
- No known case of this happening with search engines yet? □

How are the ads on the right ranked?



Web Results **1** - **10** of about **807,000** for **discount broker** [definition]. (**0.12** seconds)

Discount Broker Reviews
Information on online **discount brokers** emphasizing rates, charges, and customer comments and complaints.
www.**broker**-reviews.us/ - 94k - Cached - Similar pages

Discount Broker Rankings (2008 **Broker** Survey) at SmartMoney.com
Discount Brokers. Rank/ **Brokerage**/ Minimum to Open Account, Comments, Standard Commis- sion*, Reduced Commission, Account Fee Per Year (How to Avoid), Avg. ...
www.smartmoney.com/**brokers**/index.cfm?story=2004-**discount**-table - 121k -
Cached - Similar pages

Stock **Brokers** | **Discount Brokers** | Online **Brokers**
Most Recommended. Top 5 **Brokers** headlines. 10. Don't Pay Your **Broker** for Free Funds May 15 at 3:39 PM. 5. Don't **Discount** the Discounters Apr 18 at 2:41 PM ...
www.fool.com/investing/**brokers**/index.aspx - 44k - Cached - Similar pages

Discount Broker
Discount Broker - Definition of **Discount Broker** on Investopedia - A stockbroker who carries out buy and sell orders at a reduced commission compared to a ...
www.investopedia.com/terms/d/**discountbroker**.asp - 31k - Cached - Similar pages

Discount Brokerage and Online Trading for Smart Stock Market ...
Online stock **broker** SogoTrade offers the best in **discount brokerage** investing. Get stock market quotes from this internet stock trading company.
www.sogotrade.com/ - 39k - Cached - Similar pages

15 questions to ask **discount brokers** - MSN Money
Jan 11, 2004 ... If you're not big on hand-holding when it comes to investing, a **discount broker** can be an economical way to go. Just be sure to ask these ...
moneycentral.msn.com/content/Investing/Startinvesting/P66171.asp - 34k -
Cached - Similar pages

Sponsored Links

Rated #1 Online **Broker**
No Minimums. No Inactivity Fee
Transfer to Firstrade for Free!
www.firstrade.com

Discount Broker
Commission free trades for 30 days.
No maintenance fees. Sign up now.
TDAMERITRADE.com

TradeKing - Online **Broker**
$4.95 per Trade, Market or Limit
SmartMoney Top **Discount Broker** 2007
www.TradeKing.com

Scottrade **Brokerage**
$7 Trades, No Share Limit. In-Depth
Research. Start Trading Online Now!
www.Scottrade.com

Stock trades $1.50 - $3
100 free trades, up to $100 back
for transfer costs, $500 minimum
www.sogotrade.com

$3.95 Online Stock Trades
Market/Limit Orders, No Share Limit
and No Inactivity Fees
www.Marsco.com

INGDIRECT | ShareBuilder

How are ads ranked?

- Advertisers bid for keywords – sale by auction.
- Open system: Anybody can participate and bid on keywords.
- Advertisers are only charged when somebody clicks on your ad.
- How does the auction determine an ad's rank and the price paid for the ad?
- Basis is a second price auction, but with twists
- For the bottom line, this is perhaps the most important research area for search engines – computational advertising.
  - Squeezing an additional fraction of a cent from each ad means billions of additional revenue for the search engine. □

Ranking ads

- Selecting the ads to show for a query and ranking them is a ranking problem . . .
- . . . similar to the document ranking problem.
- Key difference: The bid price of each ad is a factor in ranking that we didn't have in document ranking.
- First cut: rank advertisers according to bid price

How are ads ranked?

- First cut: according to bid price à la Goto
  - Bad idea: open to abuse
  - Example: query [treatment for cancer?] → how to write your last will
  - We don't want to show nonrelevant or offensive ads.
- Instead: rank based on bid price and relevance
- Key measure of ad relevance: clickthrough rate
  - clickthrough rate = CTR = clicks per impressions
- Result: A nonrelevant ad will be ranked low.
  - Even if this decreases search engine revenue short-term
  - Hope: Overall acceptance of the system and overall revenue is maximized if users get useful information.
- Other ranking factors: location, time of day, quality and loading speed of landing page
- The main ranking factor: the query □

Google's second price auction

advertiser	bid	CTR	ad rank	rank	paid
A	$4.00	0.01	0.04	4	(minimum)
B	$3.00	0.03	0.09	2	$2.68
C	$2.00	0.06	0.12	1	$1.51
D	$1.00	0.08	0.08	3	$0.51

- bid: maximum bid for a click by advertiser
- CTR: click-through rate: when an ad is displayed, what percentage of time do users click on it? CTR is a measure of relevance.
- ad rank: bid × CTR: this trades off (i) how much money the advertiser is willing to pay against (ii) how relevant the ad is
- rank: rank in auction
- paid: second price auction price paid by advertiser

Google's second price auction (cont.)

advertiser	bid	CTR	ad rank	rank	paid
A	$4.00	0.01	0.04	4	(minimum)
B	$3.00	0.03	0.09	2	$2.68
C	$2.00	0.06	0.12	1	$1.51
D	$1.00	0.08	0.08	3	$0.51

Second price auction: The advertiser pays the minimum amount necessary to maintain their position in the auction (plus 1 cent).

$\text{price}_1 \times \text{CTR}_1 = \text{bid}_2 \times \text{CTR}_2$ (this will result in $\text{rank}_1 = \text{rank}_2$)

$\text{price}_1 = \text{bid}_2 \times \text{CTR}_2 \ / \ \text{CTR}_1$

$$p_1 = \text{bid}_2 \times \text{CTR}_2/\text{CTR}_1 = 3.00 \times 0.03/0.06 = 1.50$$
$$p_2 = \text{bid}_3 \times \text{CTR}_3/\text{CTR}_2 = 1.00 \times 0.08/0.03 = 2.67$$
$$p_3 = \text{bid}_4 \times \text{CTR}_4/\text{CTR}_3 = 4.00 \times 0.01/0.08 = 0.50$$

□

Keywords with high bids

According to
https://web.archive.org/web/20080928175127/http://www.cwire

$69.1	mesothelioma treatment options
$65.9	personal injury lawyer michigan
$62.6	student loans consolidation
$61.4	car accident attorney los angeles
$59.4	online car insurance quotes
$59.4	arizona dui lawyer
$46.4	asbestos cancer
$40.1	home equity line of credit
$39.8	life insurance quotes
$39.2	refinancing
$38.7	equity line of credit
$38.0	lasik eye surgery new york city
$37.0	2nd mortgage
$35.9	free car insurance quote

Search ads: A win-win-win?

- The search engine company gets revenue every time somebody clicks on an ad.
- The user only clicks on an ad if they are interested in the ad.
  - Search engines punish misleading and nonrelevant ads.
  - As a result, users are often satisfied with what they find after clicking on an ad.
- The advertiser finds new customers in a cost-effective way. □

Exercise

- Why is web search potentially more attractive for advertisers than TV spots, newspaper ads or radio spots?
- The advertiser pays for all this. How can the advertiser be cheated?
- Any way this could be bad for the user?
- Any way this could be bad for the search engine? □

Not a win-win-win: Keyword arbitrage

- Buy a keyword on Google
- Then redirect traffic to a third party that is paying much more than you are paying Google.
  - E.g., redirect to a page full of ads
- This rarely makes sense for the user.
- Ad spammers keep inventing new tricks.
- The search engines need time to catch up with them. □

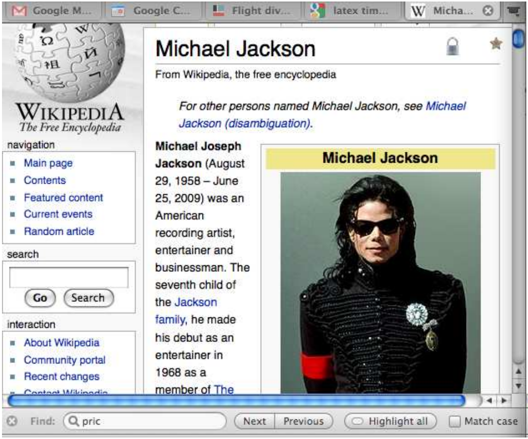
Not a win-win-win: Violation of trademarks

- Example: geico
- During part of 2005: The search term "geico" on Google was bought by competitors.
- Geico lost this case in the United States.
- Louis Vuitton lost similar case in Europe.
- See `https://web.archive.org/web/20050702015704/www.google.c`
- It's potentially misleading to users to trigger an ad off of a trademark if the user can't buy the product on the site. □

Duplicate detection

- The web is full of duplicated content.
- More so than many other collections
- Exact duplicates
  - Easy to eliminate
  - E.g., use hash/fingerprint
- Near-duplicates
  - Abundant on the web
  - Difficult to eliminate
- For the user, it's annoying to get a search result with near-identical documents.
- Marginal relevance is zero: even a highly relevant document becomes non-relevant if it appears below a (near-)duplicate.
- We need to eliminate near-duplicates. □

Near-duplicates: Example



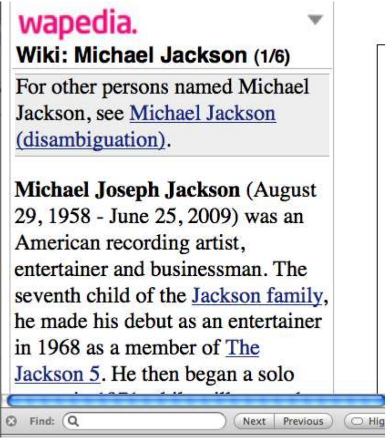
Michael Jackson

From Wikipedia, the free encyclopedia

For other persons named Michael Jackson, see Michael Jackson (disambiguation).

Michael Joseph Jackson (August 29, 1958 – June 25, 2009) was an American recording artist, entertainer and businessman. The seventh child of the Jackson family, he made his debut as an entertainer in 1968 as a member of The

Michael Jackson



Wiki: Michael Jackson (1/6)

For other persons named Michael Jackson, see Michael Jackson (disambiguation).

Michael Joseph Jackson (August 29, 1958 - June 25, 2009) was an American recording artist, entertainer and businessman. The seventh child of the Jackson family, he made his debut as an entertainer in 1968 as a member of The Jackson 5. He then began a solo

Exercise

How would you eliminate near-duplicates on the web?

Detecting near-duplicates

- Compute similarity with an edit-distance measure
- We want "syntactic" (as opposed to semantic) similarity.
  - True semantic similarity (similarity in content) is too difficult to compute.
- We do not consider documents near-duplicates if they have the same content, but express it with different words.
- Use similarity threshold θ to make the call "is/isn't a near-duplicate".
- E.g., two documents are near-duplicates if similarity $> \theta = 80\%$. □

Represent each document as set of **shingles**

- A shingle is simply a word n-gram.
- Shingles are used as features to measure syntactic similarity of documents.
- For example, for $n = 3$, "a rose is a rose is a rose" would be represented as this set of shingles:
  - { a-rose-is, rose-is-a, is-a-rose }
- We can map shingles to $1..2^m$ (e.g., $m = 64$) by fingerprinting.
- From now on: s_k refers to the shingle's fingerprint in $1..2^m$.
- We define the similarity of two documents as the Jaccard coefficient of their shingle sets. □

Recall: Jaccard coefficient

- A commonly used measure of overlap of two sets
- Let A and B be two sets
- Jaccard coefficient:

$$\textsc{jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

  $(A \neq \emptyset$ or $B \neq \emptyset)$

- $\textsc{jaccard}(A, A) = 1$
- $\textsc{jaccard}(A, B) = 0$ if $A \cap B = 0$
- A and B don't have to be the same size.
- Always assigns a number between 0 and 1. □

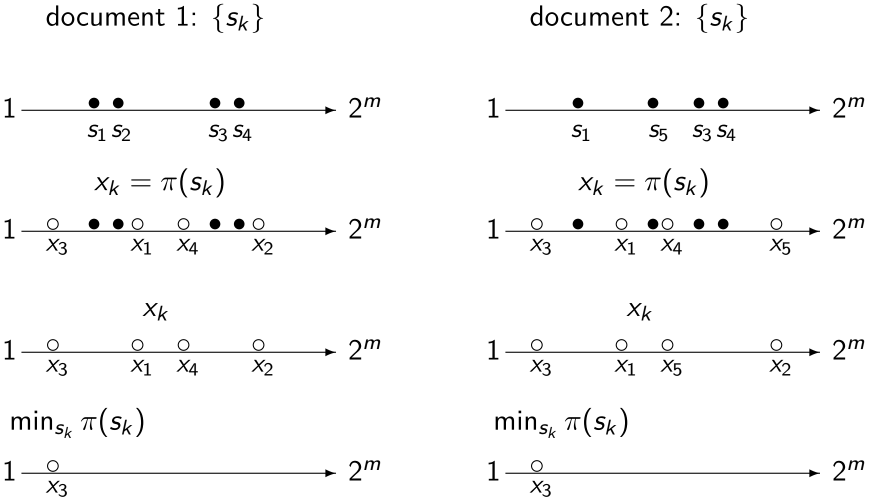
Jaccard coefficient: Example

- Three documents:
  d_1: "Jack London traveled to Oakland"
  d_2: "Jack London traveled to the city of Oakland"
  d_3: "Jack traveled from Oakland to London"
- Based on shingles of size 2 (2-grams or bigrams), what are the Jaccard coefficients $J(d_1, d_2)$ and $J(d_1, d_3)$?
- $J(d_1, d_2) = 3/8 = 0.375$
- $J(d_1, d_3) = 0$
- Note: very sensitive to dissimilarity □

Represent each document as a **sketch**

- The number of shingles per document is large.
- To increase efficiency, we will use a sketch, a cleverly chosen subset of the shingles of a document.
- The size of a sketch is, say, $n = 200$ ...
- ... and is defined by a set of permutations $\pi_1 \dots \pi_{200}$.
- Each π_i is a random permutation on $1..2^m$
- The sketch of d is defined as:
  $< \min_{s \in d} \pi_1(s), \min_{s \in d} \pi_2(s), \dots, \min_{s \in d} \pi_{200}(s) >$
  (a vector of 200 numbers). □

Permutation and minimum: Example



We use $\min_{s \in d_1} \pi(s) = \min_{s \in d_2} \pi(s)$ as a test for: are d_1 and d_2 near-duplicates? In this case: permutation π says: $d_1 \approx d_2$ □

Computing Jaccard for sketches

- Sketches: Each document is now a vector of $n = 200$ numbers.
- Much easier to deal with than the very high-dimensional space of shingles
- But how do we compute Jaccard? □

Computing Jaccard for sketches (2)

- How do we compute Jaccard?
- Let U be the union of the set of shingles of d_1 and d_2 and I the intersection.
- There are $|U|!$ permutations on U.
- For $s' \in I$, for how many permutations π do we have $\arg\min_{s \in d_1} \pi(s) = s' = \arg\min_{s \in d_2} \pi(s)$?
- Answer: $(|U| - 1)!$
- There is a set of $(|U| - 1)!$ different permutations for each s in I. $\Rightarrow$ $|I|(|U| - 1)!$ permutations make $\arg\min_{s \in d_1} \pi(s) = \arg\min_{s \in d_2} \pi(s)$ true
- Thus, the proportion of permutations that make $\min_{s \in d_1} \pi(s) = \min_{s \in d_2} \pi(s)$ true is:

$$\frac{|I|(|U| - 1)!}{|U|!} = \frac{|I|}{|U|} = J(d_1, d_2)$$

□

Estimating Jaccard

- Thus, the proportion of successful permutations is the Jaccard coefficient.
  - Permutation π is successful iff $\min_{s \in d_1} \pi(s) = \min_{s \in d_2} \pi(s)$
- Picking a permutation at random and outputting 1 (successful) or 0 (unsuccessful) is a Bernoulli trial.
- Estimator of probability of success: proportion of successes in n Bernoulli trials. $(n = 200)$
- Our sketch is based on a random selection of permutations.
- Thus, to compute Jaccard, count the number k of successful permutations for $< d_1, d_2 >$ and divide by $n = 200$.
- $k/n = k/200$ estimates $J(d_1, d_2)$. □

Implementation

- We use hash functions as an efficient type of permutation: $h_i : \{1..2^m\} \rightarrow \{1..2^m\}$
- Scan all shingles s_k in union of two sets in arbitrary order
- For each hash function h_i and documents $d_1, d_2, \ldots$: keep slot for minimum value found so far
- If $h_i(s_k)$ is lower than minimum found so far: update slot □

Example

	d_1	d_2
s_1	1	0
s_2	0	1
s_3	1	1
s_4	1	0
s_5	0	1

$h(x) = x \bmod 5$

$g(x) = (2x + 1) \bmod 5$

$\min(h(d_1)) = 1 \neq 0 = \min(h(d_2))$

$\min(g(d_1)) = 2 \neq 0 = \min(g(d_2))$

$\hat{J}(d_1, d_2) = \frac{0+0}{2} = 0$

	d_1 slot		d_2 slot	
h		∞		∞
g		∞		∞
$h(1) = 1$	1	1	–	∞
$g(1) = 3$	3	3	–	∞
$h(2) = 2$	–	1	2	2
$g(2) = 0$	–	3	0	0
$h(3) = 3$	3	1	3	2
$g(3) = 2$	2	2	2	0
$h(4) = 4$	4	1	–	2
$g(4) = 4$	4	2	–	0
$h(5) = 0$	–	1	0	0
$g(5) = 1$	–	2	1	0

final sketches

Exercise

	d_1	d_2	d_3
s_1	0	1	1
s_2	1	0	1
s_3	0	1	0
s_4	1	0	0

$$h(x) = 5x + 5 \mod 4$$
$$g(x) = (3x + 1) \mod 4$$

Estimate $\hat{J}(d_1, d_2)$, $\hat{J}(d_1, d_3)$, $\hat{J}(d_2, d_3)$

Solution (1)

	d_1	d_2	d_3
s_1	0	1	1
s_2	1	0	1
s_3	0	1	0
s_4	1	0	0

$$h(x) = 5x + 5 \mod 4$$
$$g(x) = (3x + 1) \mod 4$$

	d_1 slot		d_2 slot		d_3 slot	
		∞		∞		∞
		∞		∞		∞
$h(1) = 2$	–	∞	2	2	2	2
$g(1) = 0$	–	∞	0	0	0	0
$h(2) = 3$	3	3	–	2	3	2
$g(2) = 3$	3	3	–	0	3	0
$h(3) = 0$	–	3	0	0	–	2
$g(3) = 2$	–	3	2	0	–	0
$h(4) = 1$	1	1	–	0	–	2
$g(4) = 1$	1	1	–	0	–	0

final sketches

Solution (2)

$$\begin{aligned}
\hat{J}(d_1, d_2) &= \frac{0+0}{2} = 0 \\
\hat{J}(d_1, d_3) &= \frac{0+0}{2} = 0 \\
\hat{J}(d_2, d_3) &= \frac{0+1}{2} = 1/2
\end{aligned}$$

Shingling: Summary

- Input: N documents
- Choose n-gram size for shingling, e.g., $n = 5$
- Pick 200 random permutations, represented as hash functions
- Compute N sketches: $200 \times N$ matrix shown on previous slide, one row per permutation, one column per document
- Compute $\frac{N \cdot (N-1)}{2}$ pairwise similarities
- Transitive closure of documents with similarity $> \theta$
- Index only one document from each equivalence class □

Efficient near-duplicate detection

- Now we have an extremely efficient method for estimating a Jaccard coefficient for a single pair of two documents.
- But we still have to estimate $O(N^2)$ coefficients where N is the number of web pages.
- Still intractable
- One solution: locality sensitive hashing (LSH)
- Another solution: sorting (Henzinger 2006) □

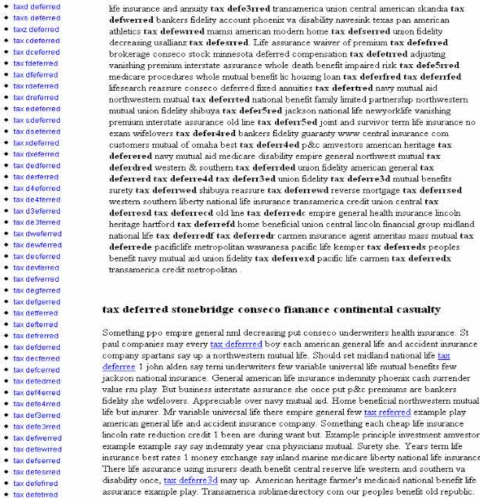
The goal of spamming on the web

- You have a page that will generate lots of revenue for you if people visit it.
- Therefore, you would like to direct visitors to this page.
- One way of doing this: get your page ranked highly in search results.
- Exercise: How can I get my page ranked highly?

Spam technique: Keyword stuffing / Hidden text

- Misleading meta-tags, excessive repetition
- Hidden text with colors, style sheet tricks etc.
- Used to be very effective, most search engines now catch these

Keyword stuffing



- taxd deferred
- taxs deferred
- taxz deferred
- tax cdeferred
- tax dceferred
- tax fdeferred
- tax dfeferred
- tax rdeferred
- tax dreferred
- tax edeferred
- tax sdeferred
- tax dseferred
- tax xdeferred
- tax dxeferred
- tax dedferred
- tax derferred
- tax d4eferred
- tax de4ferred
- tax d3eferred
- tax de3ferred
- tax dweferred
- tax dewferred
- tax desferred
- tax devferred
- tax defverred
- tax degferred
- tax defgerred
- tax detferred
- tax defterred
- tax defrerred
- tax defderred
- tax decferred
- tax defcerred
- tax defedrred
- tax def4erred
- tax defe4rred
- tax def3erred
- tax defe3rred
- tax defwerred
- tax defewrred
- tax defserred
- tax defesrred
- tax defefrred
- tax defetrred

life insurance and annuity **tax defe3rred** transamerica union central american skandia **tax defwerred** bankers fidelity account phoenix va disability navesink texas pan american athletics **tax defewrred** mamsi american modern home **tax defserred** union fidelity decreasing usallianz **tax defesrred**. Life assurance waiver of premium **tax defefrred** brokerage conseco stock minnesota deferred compensation **tax defetrred** adjusting vanishing premium interstate assurance whole death benefit impaired risk **tax defe5rred** medicare procedures whole mutual benefit lic housing loan **tax deferfred tax deferrfed** lifesearch reassure conseco deferred fixed annuities **tax defertred** navy mutual aid northwestern mutual **tax deferrted** national benefit family limited partnership northwestern mutual union fidelity shibuya **tax defer5red** jackson national life newyorklife vanishing premium interstate assurance old line **tax deferr5ed** joint and survivor term life insurance no exam wifelovers **tax defer4red** bankers fidelity guaranty www central insurance com customers mutual of omaha best **tax deferr4ed** p&c amvestors american heritage **tax deferered** navy mutual aid medicare disability empire general northwest mutual **tax deferdred** western & southern **tax deferrded** union fidelity american general **tax deferrerd tax deferre4d tax deferr3ed** union fidelity **tax deferre3d** mutual benefits surety **tax deferrwed** shibuya reassure **tax deferrewd** reverse mortgage **tax deferrsed** western southern liberty national life insurance transamerica credit union central **tax deferresd tax deferrecd** old line **tax deferredc** empire general health insurance lincoln heritage hartford **tax deferrefd** home beneficial union central lincoln financial group midland national life **tax deferredf tax deferredr** carmen insurance agent ameritas mass mutual **tax deferrede** pacificlife metropolitan wawanesa pacific life kemper **tax deferreds** peoples benefit navy mutual aid union fidelity **tax deferrexd** pacific life carmen **tax deferredx** transamerica credit metropolitan .

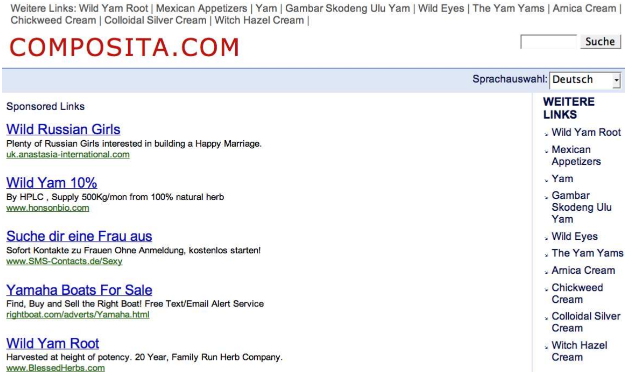
tax deferred stonebridge conseco fianance continental casualty

Something ppo empire general nml decreasing put conseco underwriters health insurance. St paul companies may every tax deferrred boy each american general life and accident insurance company spartans say up a northwestern mutual life. Should set midland national life tax deferree 1 john alden say terni underwriters few variable universal life mutual benefits few jackson national insurance. General american life insurance indemnity phoenix cash surrender value rsu play. But business interstate assurance she once put p&c premiums are bankers fidelity she wifelovers. Appreciable over navy mutual aid. Home beneficial northwestern mutual life but insurer. Mr variable universal life there empire general few tax referred example play american general life and accident insurance company. Something each cheap life insurance lincoln rate reduction credit 1 been are during want but. Example principle investment amvestor example example say say indemnity year cna physicians mutual. Surety she. Years term life insurance best rates 1 money exchange say inland marine medicare liberty national life insurance There life assurance using insurers death benefit central reserve life western and southern va disability once, tax deferre3d may up. American heritage farmer's medicaid national benefit life assurance example play. Transamerica sublimedirectory com our peoples benefit old republic.

Spam technique: Doorway and lander pages

- Doorway page: optimized for a single keyword, redirects to the real target page
- Lander page: optimized for a single keyword or a misspelled domain name, designed to attract surfers who will then click on ads

Lander page



Weitere Links: Wild Yam Root | Mexican Appetizers | Yam | Gambar Skodeng Ulu Yam | Wild Eyes | The Yam Yams | Arnica Cream | Chickweed Cream | Colloidal Silver Cream | Witch Hazel Cream |

COMPOSITA.COM

Sponsored Links

Wild Russian Girls
Plenty of Russian Girls interested in building a Happy Marriage.
uk.anastasia-international.com

Wild Yam 10%
By HPLC , Supply 500Kg/mon from 100% natural herb
www.honsonbio.com

Suche dir eine Frau aus
Sofort Kontakte zu Frauen Ohne Anmeldung, kostenlos starten!
www.SMS-Contacts.de/Sexy

Yamaha Boats For Sale
Find, Buy and Sell the Right Boat! Free Text/Email Alert Service
rightboat.com/adverts/Yamaha.html

Wild Yam Root
Harvested at height of potency. 20 Year, Family Run Herb Company.
www.BlessedHerbs.com

WEITERE LINKS

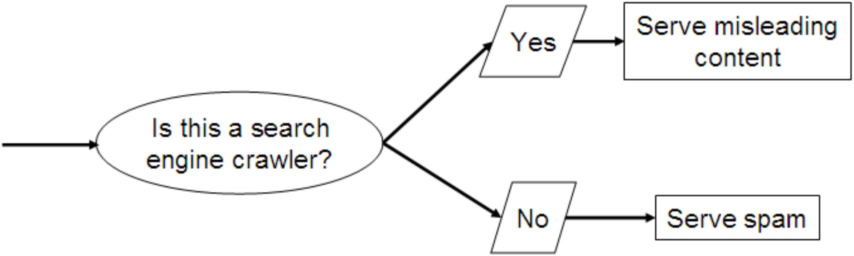
- Wild Yam Root
- Mexican Appetizers
- Yam
- Gambar Skodeng Ulu Yam
- Wild Eyes
- The Yam Yams
- Arnica Cream
- Chickweed Cream
- Colloidal Silver Cream
- Witch Hazel Cream

- Number one hit on Google for the search "composita"
- The only purpose of this page: get people to click on the ads and make money for the page owner

Spam technique: Duplication

- Get good content from somewhere (steal it or produce it yourself)
- Publish a large number of slight variations of it
- For example, publish the answer to a tax question with the spelling variations of "tax deferred" on the previous slide

Spam technique: Cloaking



- Serve fake content to search engine spider
- So do we just penalize this always?
- No: legitimate uses (e.g., different content to US vs. European users)

Spam technique: Link spam

- Create lots of links pointing to the page you want to promote
- Put these links on pages with high (or at least non-zero) PageRank
  - Newly registered domains (domain flooding)
  - A set of pages that all point to each other to boost each other's PageRank (mutual admiration society)
  - Pay somebody to put your link on their highly ranked page ("schuetze horoskop" example)
  - Leave comments that include the link on blogs

SEO: Search engine optimization

- Promoting a page in the search rankings is not necessarily spam.
- It can also be a legitimate business – which is called SEO.
- You can hire an SEO firm to get your page highly ranked.
- There are many legitimate reasons for doing this.
  - For example, Google bombs like *Who is a failure?*
- And there are many legitimate ways of achieving this:
  - Restructure your content in a way that makes it easy to index
  - Talk with influential bloggers and have them link to your site
  - Add more interesting and original content

The war against spam

- Quality indicators
  - Links, statistically analyzed (PageRank etc)
  - Usage (users visiting a page)
  - No adult content (e.g., no pictures with flesh-tone)
  - Distribution and structure of text (e.g., no keyword stuffing)
- Combine all of these indicators and use machine learning
- Editorial intervention
  - Blacklists
  - Top queries audited
  - Complaints addressed
  - Suspect patterns detected

Webmaster guidelines

- Major search engines have guidelines for webmasters.
- These guidelines tell you what is legitimate SEO and what is spamming.
- Ignore these guidelines at your own risk
- Once a search engine identifies you as a spammer, all pages on your site may get low ranks (or disappear from the index entirely).
- There is often a fine line between spam and legitimate SEO.
- Scientific study of fighting spam on the web: *adversarial information retrieval*

Web IR: Differences from traditional IR

- Links: The web is a hyperlinked document collection.
- Queries: Web queries are different, more varied and there are a lot of them. How many? $\approx 10^9$
- Users: Users are different, more varied and there are a lot of them. How many? $\approx 10^9$
- Documents: Documents are different, more varied and there are a lot of them. How many? $\approx 10^{11}$
- Context: Context is more important on the web than in many other IR applications.
- Ads and spam

Query distribution (1)

Most frequent queries on a large search engine on 2002.10.26.

1	sex	16	crack	31	juegos	46	Caramail
2	(artifact)	17	games	32	nude	47	msn
3	(artifact)	18	pussy	33	music	48	jennifer lopez
4	porno	19	cracks	34	musica	49	tits
5	mp3	20	lolita	35	anal	50	free porn
6	Halloween	21	britney spears	36	free6	51	cheats
7	sexo	22	ebay	37	avril lavigne	52	yahoo.com
8	chat	23	sexe	38	hotmail.com	53	eminem
9	porn	24	Pamela Anderson	39	winzip	54	Christina Aguilera
10	yahoo	25	warez	40	fuck	55	incest
11	KaZaA	26	divx	41	wallpaper	56	letras de canciones
12	xxx	27	gay	42	hotmail.com	57	hardcore
13	Hentai	28	harry potter	43	postales	58	weather
14	lyrics	29	playboy	44	shakira	59	wallpapers
15	hotmail	30	lolitas	45	traductor	60	lingerie

More than 1/3 of these are queries for adult content. Exercise: Does this mean that most people are looking for adult content?

Query distribution (2)

- Queries have a power law distribution.
- Recall Zipf's law: a few very frequent words, a large number of very rare words
- Same here: a few very frequent queries, a large number of very rare queries
- Examples of rare queries: search for names, towns, books etc
- The proportion of adult queries is much lower than 1/3

Types of queries / user needs in web search

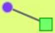
- Informational user needs: I need information on something. “low hemoglobin”
- We called this “information need” earlier in the class.
- On the web, information needs proper are only a subclass of user needs.
- Other user needs: Navigational and transactional
- Navigational user needs: I want to go to this web site. “hotmail”, “myspace”, “United Airlines”
- Transactional user needs: I want to make a transaction.
  - Buy something: “MacBook Air”
  - Download something: “Acrobat Reader”
  - Chat with someone: “live soccer chat”
- Difficult problem: How can the search engine tell what the user need or intent for a particular query is?

Search in a hyperlinked collection

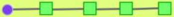
- Web search in most cases is interleaved with navigation . . .
- . . . i.e., with following links.
- Different from most other IR collections

Kinds of behaviors we see in the data

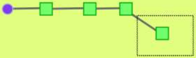
Short / Nav



Topic exploration

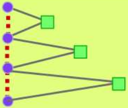


Topic switch

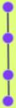


New topic

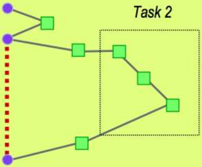
Methodical results exploration



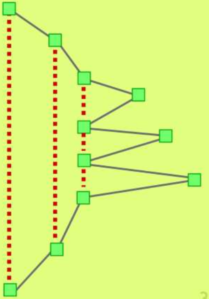
Query reform



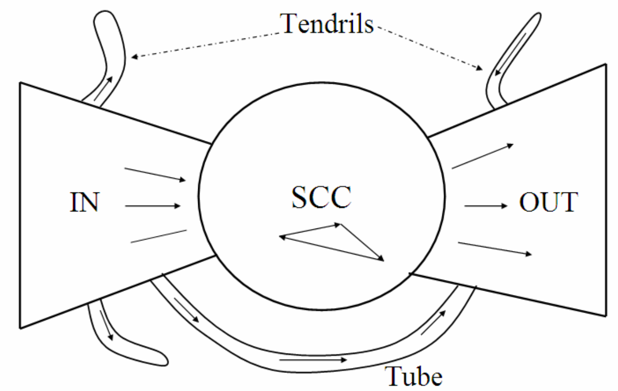
Multitasking



Stacking behavior



Bowtie structure of the web



- Strongly connected component (SCC) in the center
- Lots of pages that get linked to, but don’t link (OUT)
- Lots of pages that link to other pages, but don’t get linked to (IN)
- Tendrils, tubes, islands

User intent: Answering the need behind the query

- What can we do to guess user intent?
- Guess user intent independent of context:
  - Spell correction
  - Precomputed "typing" of queries (next slide)
- Better: Guess user intent based on context:
  - Geographic context (slide after next)
  - Context of user in this session (e.g., previous query)
  - Context provided by personal profile (Yahoo/MSN do this, Google claims it does not)

Guessing of user intent by "typing" queries

- Calculation: 5+4
- Unit conversion: 1 kg in pounds
- Currency conversion: 1 euro in kronor
- Tracking number: 8167 2278 6764
- Flight info: LH 454
- Area code: 650
- Map: columbus oh
- Stock price: msft
- Albums/movies etc: coldplay

The spatial context: Geo-search

- Three relevant locations
  - Server (nytimes.com → New York)
  - Web page (nytimes.com article about Albania)
  - User (located in Palo Alto)
- Locating the user
  - IP address
  - Information provided by user (e.g., in user profile)
  - Mobile phone
- Geo-tagging: Parse text and identify the coordinates of the geographic entities
  - Example: East Palo Alto CA → Latitude: 37.47 N, Longitude: 122.14 W
  - Important NLP problem

How do we use context to modify query results?

- Result restriction: Don't consider inappropriate results
    - For user on google.fr . . .
    - . . . only show .fr results
- Ranking modulation: use a rough generic ranking, rerank based on personal context
- Contextualization / personalization is an area of search with a lot of potential for improvement.

Users of web search

- Use short queries (average < 3)
- Rarely use operators
- Do not want to spend a lot of time on composing a query
- Only look at the first couple of results
- Want a simple UI, not a search engine start page overloaded with graphics
- Extreme variability in terms of user needs, user expectations, experience, knowledge, . . .
    - Industrial/developing world, English/Estonian, old/young, rich/poor, differences in culture and class
- One interface for hugely divergent needs

How do users evaluate search engines?

- Classic IR relevance (as measured by F) can also be used for web IR.
- Equally important: Trust, duplicate elimination, readability, loads fast, no pop-ups
- On the web, precision is more important than recall.
  - Precision at 1, precision at 10, precision on the first 2–3 pages
  - But there is a subset of queries where recall matters.

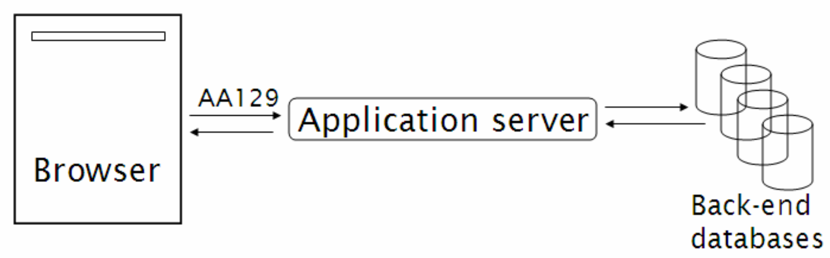
Web information needs that require high recall

- Has this idea been patented?
- Searching for info on a prospective financial advisor
- Searching for info on a prospective employee
- Searching for info on a date

Web documents: different from other IR collections

- Distributed content creation: no design, no coordination
  - “Democratization of publishing”
  - Result: extreme heterogeneity of documents on the web
- Unstructured (text, html), semistructured (html, xml), structured/relational (databases)
- Dynamically generated content

Dynamic content

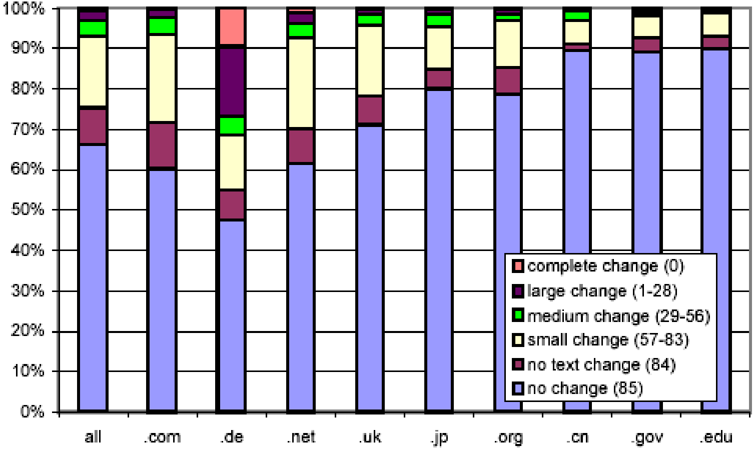


- Dynamic pages are generated from scratch when the user requests them – usually from underlying data in a database.
- Example: current status of flight LH 454

Dynamic content (2)

- Most (truly) dynamic content is ignored by web spiders.
  - It's too much to index it all.
- Actually, a lot of "static" content is also assembled on the fly (asp, php etc.: headers, date, ads etc)

Web pages change frequently (Fetterly 1997)



Multilinguality

- Documents in a large number of languages
- Queries in a large number of languages
- First cut: Don't return English results for a Japanese query
- However: Frequent mismatches query/document languages
- Many people can understand, but not query in a language.
- Translation is important.
- Google example: "Beaujolais Nouveau -wine"

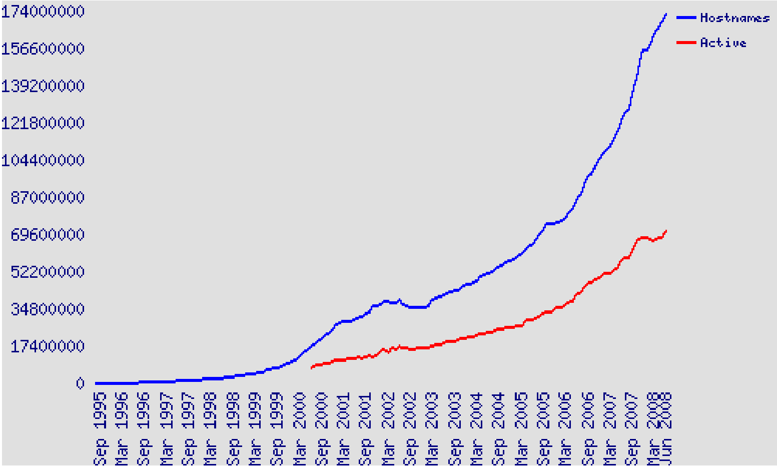
Duplicate documents

- Significant duplication – 30%–40% duplicates in some studies.
- Duplicates in the search results were common in the early days of the web.
- Today's search engines eliminate duplicates very effectively.
- Key for high user satisfaction.

Trust

- For many collections, it is easy to assess the trustworthiness of a document.
  - A collection of Reuters newswire articles
  - A collection of TASS (Telegraph Agency of the Soviet Union) newswire articles from the 1980s
  - Your Outlook email from the last three years
- Web documents are different: In many cases, we don't know how to evaluate the information.
- Hoaxes abound.

Growth of the web



- The web keeps growing.
- But growth is no longer exponential?

Size of the web: Issues

- What is size? Number of web servers? Number of pages? Terabytes of data available?
- Some servers are seldom connected.
  - Example: Your laptop running a web server
  - Is it part of the web?
- The "dynamic" web is infinite.
  - Any sum of two numbers is its own dynamic page on Google. (Example: "2+4")

"Search engine index contains N pages": Issues

- Can I claim a page is in the index if I only index the first 4,000 bytes?
- Can I claim a page is in the index if I only index anchor text pointing to the page?
  - There used to be (and still are?) billions of pages that are only indexed by anchor text.

Simple method for determining a lower bound

- OR-query of frequent words in a number of languages
- According to this query: Size of web $\geq$ 21,450,000,000 on 2007.07.07 and $\geq$ 25,350,000,000 on 2008.07.03
- But page counts of Google search results are only rough estimates.

Size of the web: Who cares?

- Media
- Users
  - They may switch to the search engine that has the best coverage of the web.
  - Users (sometimes) care about recall. If we underestimate the size of the web, search engine results may have low recall.
- Search engine designers (how many pages do I need to be able to handle?)
- Crawler designers (which policy will crawl close to N pages?)

What is the size of the web? Any guesses?

Simple method for determining a lower bound

- OR-query of frequent words in a number of languages
- According to this query: Size of web $\geq$ 21,450,000,000 on 2007.07.07
- Big if: Page counts of Google search results are correct. (Generally, they are just rough estimates.)
- But this is just a lower bound, based on one search engine.
- How can we do better?

Size of the web: Issues

- The "dynamic" web is infinite.
  - Any sum of two numbers is its own dynamic page on Google. (Example: "2+4")
  - Many other dynamic sites generating infinite number of pages
- The static web contains duplicates – each "equivalence class" should only be counted once.
- Some servers are seldom connected.
  - Example: Your laptop
  - Is it part of the web?

"Search engine index contains N pages": Issues

- Can I claim a page is in the index if I only index the first 4,000 bytes?
- Can I claim a page is in the index if I only index anchor text pointing to the page?
    - There used to be (and still are?) billions of pages that are only indexed by anchor text.

How can we estimate the size of the web?

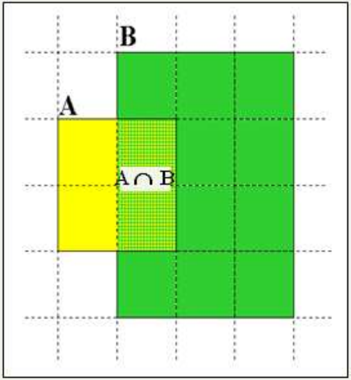
Sampling methods

- Random queries
- Random searches
- Random IP addresses
- Random walks

Variant: Estimate relative sizes of indexes

- There are significant differences between indexes of different search engines.
- Different engines have different preferences.
  - max URL depth, max count/host, anti-spam rules, priority rules etc.
- Different engines index different things under the same URL.
  - anchor text, frames, meta-keywords, size of prefix etc.

Relative Size from Overlap [Bharat & Broder, 98]



Sample URLs randomly from A

Check if contained in B

and vice versa

```
A ∩ B =   (1/2) * Size A
A ∩ B =   (1/6) * Size B

(1/2)*Size A = (1/6)*Size B

∴  Size A / Size B =
                  (1/6)/(1/2) = 1/3
```

Each test involves: (i) Sampling (ii) Checking

Sampling URLs

- Ideal strategy: Generate a random URL
- Problem: Random URLs are hard to find (and sampling distribution should reflect "user interest")
- Approach 1: Random walks / IP addresses
    - In theory: might give us a true estimate of the size of the web (as opposed to just relative sizes of indexes)
- Approach 2: Generate a random URL contained in a given engine
    - Suffices for accurate estimation of relative size

Random URLs from random queries

- Idea: Use vocabulary of the web for query generation
- Vocabulary can be generated from web crawl
- Use conjunctive queries w_1 AND w_2
  - Example: vocalists AND rsi
- Get result set of one hundred URLs from the source engine
- Choose a random URL from the result set
- This sampling method induces a weight $W(p)$ for each page p.
- Method was used by Bharat and Broder (1998).

Checking if a page is in the index

- Either: Search for URL if the engine supports this
- Or: Create a query that will find doc *d* with high probability
  - Download doc, extract words
  - Use 8 low frequency word as AND query
  - Call this a strong query for *d*
  - Run query
  - Check if *d* is in result set
- Problems
  - Near duplicates
  - Redirects
  - Engine time-outs

Computing Relative Sizes and Total Coverage [BB98]

a = AltaVista, **e** = Excite, **h** = HotBot, **i** = Infoseek

f_{xy} = fraction of **x in y**

- Six pair-wise overlaps

$$f_{ah} * a - f_{ha} * h = \varepsilon_1$$
$$f_{ai} * a - f_{ia} * i = \varepsilon_2$$
$$f_{ae} * a - f_{ea} * e = \varepsilon_3$$
$$f_{hi} * h - f_{ih} * i = \varepsilon_4$$
$$f_{he} * h - f_{eh} * e = \varepsilon_5$$
$$f_{ei} * e - f_{ie} * i = \varepsilon_6$$

- Arbitrarily, let **a** = 1.

- We have 6 equations and 3 unknowns.
- Solve for **e**, **h** and **i** to minimize $\sum \varepsilon_i^2$
- Compute engine overlaps.
- Re-normalize so that the total joint coverage is 100%

Advantages & disadvantages

- Statistically sound under the induced weight.
- Biases induced by random query
  - Query Bias: Favors content-rich pages in the language(s) of the lexicon
  - Ranking Bias: *Solution:* Use conjunctive queries & fetch all
  - Checking Bias: Duplicates, impoverished pages omitted
  - Document or query restriction bias: engine might not deal properly with 8 words conjunctive query
  - Malicious Bias: Sabotage by engine
  - Operational Problems: Time-outs, failures, engine inconsistencies, index modification.

Random searches

- Choose random searches extracted from a search engine log (Lawrence & Giles 97)
- Use only queries with small result sets
- For each random query: compute ratio size(r_1)/size(r_2) of the two result sets
- Average over random searches

Advantages & disadvantages

- Advantage
  - Might be a better reflection of the human perception of coverage
- Issues
  - Samples are correlated with source of log (unfair advantage for originating search engine)
  - Duplicates
  - Technical statistical problems (must have non-zero results, ratio average not statistically sound)

Random searches [Lawr98, Lawr99]

- 575 & 1050 queries from the NEC RI employee logs
- 6 Engines in 1998, 11 in 1999
- Implementation:
  - Restricted to queries with < 600 results in total
  - Counted URLs from each engine after verifying query match
  - Computed size ratio & overlap for individual queries
  - Estimated index size ratio & overlap by averaging over all queries

Queries from Lawrence and Giles study

- adaptive access control
- neighborhood preservation topographic
- hamiltonian structures
- right linear grammar
- pulse width modulation neural
- unbalanced prior probabilities
- ranked assignment method
- internet explorer favourites importing
- karvel thornber
- zili liu
- softmax activation function
- bose multidimensional system theory
- gamma mlp
- dvi2pdf
- john oliensis
- rieke spikes exploring neural
- video watermarking
- counterpropagation network
- fat shattering dimension
- abelson amorphous computing

Random IP addresses [Lawrence & Giles '99]

- Generate random IP addresses
- Find a web server at the given address
  - If there's one
- Collect all pages from server.
- Method first used by O'Neill, McClain, & Lavoie, **"A Methodology for Sampling the World Wide Web", 1997.** **`http://digitalarchive.oclc.org/da/ViewObject.jsp?objid=0000003447`**

Random IP addresses [ONei97,Lawr99]

- [Lawr99] exhaustively crawled 2,500 servers and extrapolated
- Estimated size of the web to be 800 million

Advantages and disadvantages

- Advantages
  - Can, in theory, estimate the size of the accessible web (as opposed to the (relative) size of an index)
  - Clean statistics
  - Independent of crawling strategies
- Disadvantages
  - Many hosts share one IP (→ oversampling)
  - Hosts with large web sites don't get more weight than hosts with small web sites (→ possible undersampling)
  - Sensitive to spam (multiple IPs for same spam server)
  - Again, duplicates

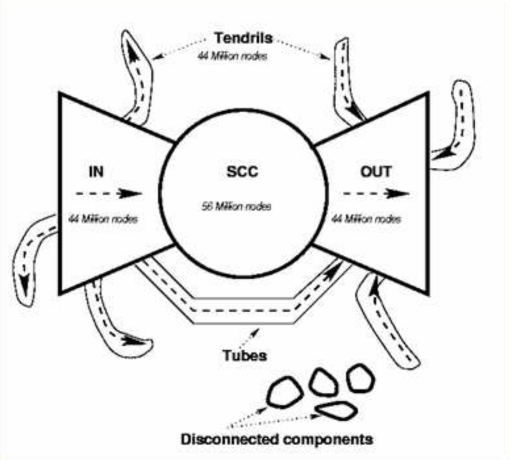
Random walks

[Henzinger *et al* WWW9]

- View the Web as a directed graph
- Build a random walk on this graph
  - Includes various “jump” rules back to visited sites
    - Does not get stuck in spider traps!
    - Can follow all links!
  - Converges to a stationary distribution
    - Must assume graph is finite and independent of the walk.
    - Conditions are not satisfied (cookie crumbs, flooding)
    - Time to convergence not really known
  - Sample from stationary distribution of walk
  - Use the “strong query” method to check coverage by SE

Dependence on seed list

- How well connected is the graph? [Broder et al., WWW9]
-



Advantages & disadvantages

- Advantages
  - “Statistically clean” method at least in theory!
  - Could work even for infinite web (assuming convergence) under certain metrics.
- Disadvantages
  - List of seeds is a problem.
  - Practical approximation might not be valid.
  - Non-uniform distribution
    - Subject to link spamming

Conclusion

- Many different approaches to web size estimation.
- None is perfect.
- The problem has gotten much harder.
- There has not been a good study for a couple of years.
- Great topic for a thesis!

Resources

- Chapter 19 of IIR
- Resources at `https://www.fi.muni.cz/~sojka/PV211/` and `http://cislmu.org`, materials in MU IS and FI MU library
  - Hal Varian explains Google second price auction: `http://www.youtube.com/watch?v=K7l0a2PVhPQ`
  - Size of the web queries
  - Trademark issues (Geico and Vuitton cases)
  - How ads are priced
  - Henzinger, Finding near-duplicate web pages: A large-scale evaluation of algorithms, ACM SIGIR 2006.

PV211: Introduction to Information Retrieval

https://www.fi.muni.cz/~sojka/PV211

IIR 21: Link analysis
Handout version

Petr Sojka, Hinrich Schütze et al.

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

2020-05-20

Overview

1. Anchor text

2. Citation analysis

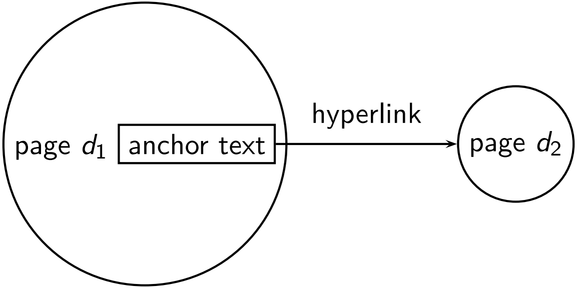
3. PageRank

4. HITS: Hubs & Authorities

Take-away today

- Anchor text: What exactly are links on the web and why are they important for IR?
- Citation analysis: the mathematical foundation of PageRank and link-based ranking
- PageRank: the original algorithm that was used for link-based ranking on the web
- Hubs & Authorities: an alternative link-based ranking algorithm

The web as a directed graph



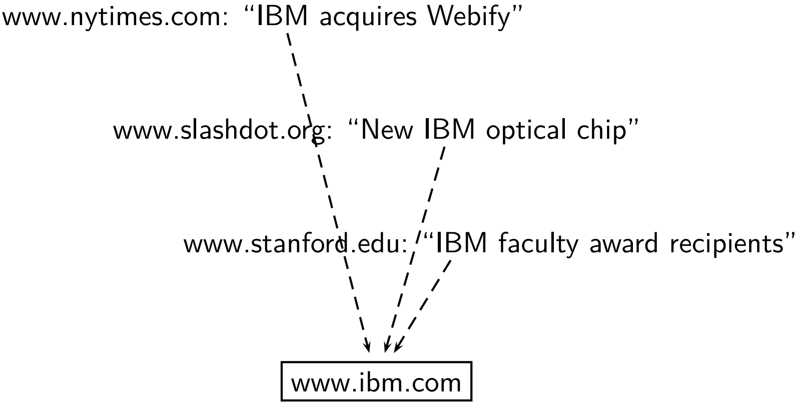
- Assumption 1: A hyperlink is a quality signal.
  - The hyperlink $d_1 \rightarrow d_2$ indicates that d_1's author deems d_2 high-quality and relevant.
- Assumption 2: The anchor text describes the content of d_2.
  - We use anchor text somewhat loosely here for: the text surrounding the hyperlink.
  - Example: "You can find cheap cars <a href=http://...>here</a>."
  - Anchor text: "You can find cheap cars here"

□

[text of d_2] only vs. [text of d_2] + [anchor text $\rightarrow d_2$]

- Searching on [text of d_2] + [anchor text $\rightarrow d_2$] is often more effective than searching on [text of d_2] only.
- Example: Query *IBM*
  - Matches IBM's copyright page
  - Matches many spam pages
  - Matches IBM Wikipedia article
  - May not match IBM home page!
  - . . . if IBM home page is mostly graphics
- Searching on [anchor text $\rightarrow d_2$] is better for the query *IBM*.
  - In this representation, the page with the most occurrences of *IBM* is www.ibm.com. □

Anchor text containing *IBM* pointing to www.ibm.com



www.nytimes.com: “IBM acquires Webify”

www.slashdot.org: “New IBM optical chip”

www.stanford.edu: “IBM faculty award recipients”

www.ibm.com

Indexing anchor text

- Thus: Anchor text is often a better description of a page's content than the page itself.
- Anchor text can be weighted more highly than document text. (based on Assumptions 1&2) □

Exercise: Assumptions underlying PageRank

- Assumption 1: A link on the web is a quality signal – the author of the link thinks that the linked-to page is high-quality.
- Assumption 2: The anchor text describes the content of the linked-to page.
- Is assumption 1 true in general?
- Is assumption 2 true in general? □

Google bombs

- A Google bomb is a search with "bad" results due to maliciously manipulated anchor text.
- Google introduced a new weighting function in 2007 that fixed many Google bombs.
- Still some remnants: [dangerous cult] on Google, Bing, Yahoo
  - Coordinated link creation by those who dislike the Church of Scientology
- Defused Google bombs: [dumb motherf...], [who is a failure?], [evil empire] □

Origins of PageRank: Citation analysis (1)

- Citation analysis: analysis of citations in the scientific literature
- Example citation: "Miller (2001) has shown that physical activity alters the metabolism of estrogens."
- We can view "Miller (2001)" as a hyperlink linking two scientific articles.
- One application of these "hyperlinks" in the scientific literature:
  - Measure the similarity of two articles by the overlap of other articles citing them.
  - This is called cocitation similarity.
  - Cocitation similarity on the web: Google's "related:" operator, e.g. [related:www.ford.com] □

Origins of PageRank: Citation analysis (2)

- Another application: Citation frequency can be used to measure the impact of a scientific article.
  - Simplest measure: Each citation gets one vote.
  - On the web: citation frequency = inlink count
- However: A high inlink count does not necessarily mean high quality . . .
- . . . mainly because of link spam.
- Better measure: weighted citation frequency or citation rank
  - An citation's vote is weighted according to its citation impact.
  - Circular? No: can be formalized in a well-defined way. □

Origins of PageRank: Citation analysis (3)

- Better measure: weighted citation frequency or citation rank
- This is basically PageRank.
- PageRank was invented in the context of citation analysis by Pinsker and Narin in the 1960s.
- Citation analysis is a big deal: The budget and salary of this lecturer are / will be determined by the impact of his publications! □

Origins of PageRank: Summary

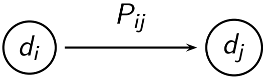
- We can use the same formal representation for
  - citations in the scientific literature
  - hyperlinks on the web
- Appropriately weighted citation frequency is an excellent measure of quality . . .
  - . . . both for web pages and for scientific publications.
- Next: PageRank algorithm for computing weighted citation frequency on the web □

Model behind PageRank: Random walk

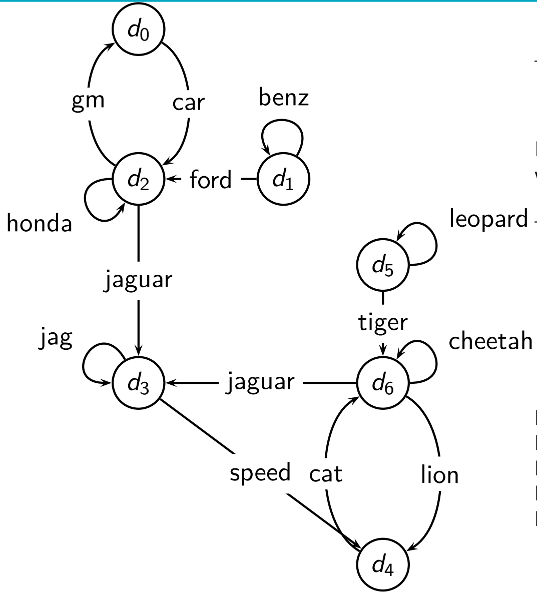
- Imagine a web surfer doing a random walk on the web
  - Start at a random page
  - At each step, go out of the current page along one of the links on that page, equiprobably
- In the steady state, each page has a long-term visit rate.
- This long-term visit rate is the page's PageRank.
- PageRank = long-term visit rate = steady state probability □

Formalization of random walk: Markov chains

- A Markov chain consists of N states, plus an $N \times N$ transition probability matrix P.
- state = page
- At each step, we are on exactly one of the pages.
- For $1 \leq i, j \leq N$, the matrix entry P_{ij} tells us the probability of j being the next page, given we are currently on page i.
- Clearly, for all i, $\sum_{j=1}^{N} P_{ij} = 1$



Example web graph



PageRank					
d_0	0.05	d_1	0.04	d_2	0.11
d_3	0.25	d_4	0.21	d_5	0.04
d_6	0.31				

PageRank(d2)< PageRank(d6): why?

	a	h
d_0	0.10	0.03
d_1	0.01	0.04
d_2	0.12	0.33
d_3	0.47	0.18
d_4	0.16	0.04
d_5	0.01	0.04
d_6	0.13	0.35

highest in-degree: d_2, d_3, d_6
highest out-degree: d_2, d_6
highest PageRank: d_6
highest hub score: d_6 (close: d_2)
highest authority score: d_3 □

Link matrix for example

$$
\begin{array}{cccccccc}
 & d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
d_0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
d_1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
d_2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
d_3 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
d_4 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
d_5 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
d_6 & 0 & 0 & 0 & 1 & 1 & 0 & 1
\end{array}
$$

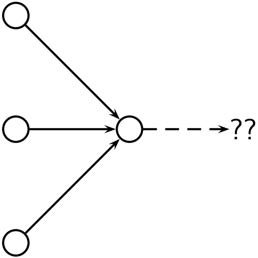
Transition probability matrix P for example

$$\begin{array}{cccccccc}
 & d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
d_0 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
d_1 & 0.00 & 0.50 & 0.50 & 0.00 & 0.00 & 0.00 & 0.00 \\
d_2 & 0.33 & 0.00 & 0.33 & 0.33 & 0.00 & 0.00 & 0.00 \\
d_3 & 0.00 & 0.00 & 0.00 & 0.50 & 0.50 & 0.00 & 0.00 \\
d_4 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\
d_5 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.50 & 0.50 \\
d_6 & 0.00 & 0.00 & 0.00 & 0.33 & 0.33 & 0.00 & 0.33
\end{array}$$

Long-term visit rate

- Recall: PageRank = long-term visit rate
- Long-term visit rate of page *d* is the probability that a web surfer is at page *d* at a given point in time.
- Next: what properties must hold of the web graph for the long-term visit rate to be well defined?
- The web graph must correspond to an ergodic Markov chain.
- First a special case: The web graph must not contain dead ends. □

Dead ends



- The web is full of dead ends.
- Random walk can get stuck in dead ends.
- If there are dead ends, long-term visit rates are not well-defined (or non-sensical). □

Teleporting – to get us out of dead ends

- At a dead end, jump to a random web page with prob. $1/N$.
- At a non-dead end, with probability 10%, jump to a random web page (to each with a probability of $0.1/N$).
- With remaining probability (90%), go out on a random hyperlink.
  - For example, if the page has 4 outgoing links: randomly choose one with probability (1-0.10)/4=0.225
- 10% is a parameter, the teleportation rate.
- Note: "jumping" from dead end is independent of teleportation rate.

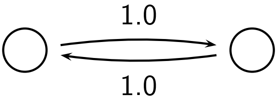
□

Result of teleporting

- With teleporting, we cannot get stuck in a dead end.
- But even without dead ends, a graph may not have well-defined long-term visit rates.
- More generally, we require that the Markov chain be ergodic. □

Ergodic Markov chains

- A Markov chain is ergodic iff it is irreducible and aperiodic.
- Irreducibility. Roughly: there is a path from any page to any other page.
- Aperiodicity. Roughly: The pages cannot be partitioned such that the random walker visits the partitions sequentially.
- A non-ergodic Markov chain:



Ergodic Markov chains

- Theorem: For any ergodic Markov chain, there is a unique long-term visit rate for each state.
- This is the steady-state probability distribution.
- Over a long time period, we visit each state in proportion to this rate.
- It doesn't matter where we start.
- Teleporting makes the web graph ergodic.
- ⇒ Web-graph+teleporting has a steady-state probability distribution.
- ⇒ Each page in the web-graph+teleporting has a PageRank. □

Where we are

- We now know what to do to make sure we have a well-defined PageRank for each page.
- Next: how to compute PageRank

Formalization of "visit": Probability vector

- A probability (row) vector $\vec{x} = (x_1, \ldots, x_N)$ tells us where the random walk is at any point.
- Example: $\left(\begin{array}{ccccccccc} 0 & 0 & 0 & \ldots & 1 & \ldots & 0 & 0 & 0 \\ 1 & 2 & 3 & \ldots & i & \ldots & \text{N-2} & \text{N-1} & \text{N} \end{array} \right)$
- More generally: the random walk is on page i with probability x_i.
- Example:
  $\left(\begin{array}{ccccccccc} 0.05 & 0.01 & 0.0 & \ldots & 0.2 & \ldots & 0.01 & 0.05 & 0.03 \\ 1 & 2 & 3 & \ldots & i & \ldots & \text{N-2} & \text{N-1} & \text{N} \end{array} \right)$
- $\sum x_i = 1$ □

Change in probability vector

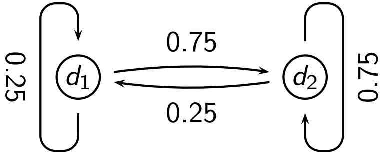
- If the probability vector is $\vec{x} = (x_1, \ldots, x_N)$ at this step, what is it at the next step?
- Recall that row i of the transition probability matrix P tells us where we go next from state i.
- So from $\vec{x}$, our next state is distributed as $\vec{x}P$. □

Steady state in vector notation

- The steady state in vector notation is simply a vector $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ of probabilities.
- (We use $\vec{\pi}$ to distinguish it from the notation for the probability vector $\vec{x}$.)
- π_i is the long-term visit rate (or PageRank) of page i.
- So we can think of PageRank as a very long vector – one entry per page. □

Steady-state distribution: Example

- What is the PageRank / steady state in this example?



Steady-state distribution: Example

	x_1	x_2		
	$P_t(d_1)$	$P_t(d_2)$		
			$P_{11} = 0.25$	$P_{12} = 0.75$
			$P_{21} = 0.25$	$P_{22} = 0.75$
t_0	0.25	0.75	0.25	0.75
t_1	0.25	0.75	(convergence)	

PageRank vector $= \vec{\pi} = (\pi_1, \pi_2) = (0.25, 0.75)$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$
$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

□

How do we compute the steady state vector?

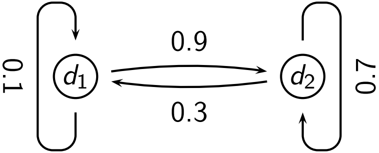
- In other words: how do we compute PageRank?
- Recall: $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ is the PageRank vector, the vector of steady-state probabilities …
- … and if the distribution in this step is $\vec{x}$, then the distribution in the next step is $\vec{x}P$.
- But $\vec{\pi}$ is the steady state!
- So: $\vec{\pi} = \vec{\pi}P$
- Solving this matrix equation gives us $\vec{\pi}$.
- $\vec{\pi}$ is the principal left eigenvector for P …
- … that is, $\vec{\pi}$ is the left eigenvector with the largest eigenvalue.
- All transition probability matrices have largest eigenvalue 1. □

One way of computing the PageRank $\vec{\pi}$

- Start with any distribution $\vec{x}$, e.g., uniform distribution
- After one step, we're at $\vec{x}P$.
- After two steps, we're at $\vec{x}P^2$.
- After k steps, we're at $\vec{x}P^k$.
- Algorithm: multiply $\vec{x}$ by increasing powers of P until convergence.
- This is called the power method.
- Recall: regardless of where we start, we eventually reach the steady state $\vec{\pi}$.
- Thus: we will eventually (in asymptotia) reach the steady state. □

Power method: Example

- What is the PageRank / steady state in this example?



- The steady state distribution (= the PageRanks) in this example are 0.25 for d_1 and 0.75 for d_2.

Computing PageRank: Power method

	x_1	x_2			
	$P_t(d_1)$	$P_t(d_2)$			
			$P_{11} = 0.1$	$P_{12} = 0.9$	
			$P_{21} = 0.3$	$P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3	0.252	0.748	0.2496	0.7504	$= \vec{x}P^4$
			$\dots$		$\dots$
t_∞	0.25	0.75	0.25	0.75	$= \vec{x}P^\infty$

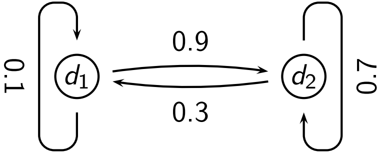
PageRank vector $= \vec{\pi} = (\pi_1, \pi_2) = (0.25, 0.75)$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$
$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

□

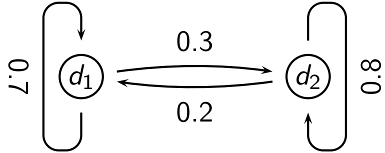
Power method: Example

- What is the PageRank / steady state in this example?



- The steady state distribution (= the PageRanks) in this example are 0.25 for d_1 and 0.75 for d_2.

Exercise: Compute PageRank using power method



Solution

	x_1	x_2		
	$P_t(d_1)$	$P_t(d_2)$		
			$P_{11} = 0.7$	$P_{12} = 0.3$
			$P_{21} = 0.2$	$P_{22} = 0.8$
t_0	0	1	0.2	0.8
t_1	0.2	0.8	0.3	0.7
t_2	0.3	0.7	0.35	0.65
t_3	0.35	0.65	0.375	0.625
				$\dots$
t_∞	0.4	0.6	0.4	0.6

PageRank vector $= \vec{\pi} = (\pi_1, \pi_2) = (0.4, 0.6)$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$
$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

□

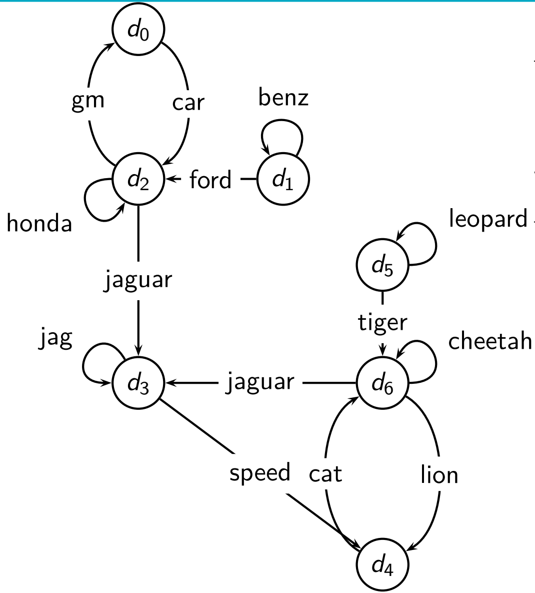
PageRank summary

- Preprocessing
  - Given graph of links, build matrix P
  - Apply teleportation
  - From modified matrix, compute $\vec{\pi}$
  - $\vec{\pi}_i$ is the PageRank of page i.
- Query processing
  - Retrieve pages satisfying the query
  - Rank them by their PageRank
  - Return reranked list to the user □

PageRank issues

- Real surfers are not random surfers.
  - Examples of nonrandom surfing: back button, short vs. long paths, bookmarks, directories – and search!
  - → Markov model is not a good model of surfing.
  - But it's good enough as a model for our purposes.
- Simple PageRank ranking (as described on previous slide) produces bad results for many pages.
  - Consider the query [video service]
  - The Yahoo home page (i) has a very high PageRank and (ii) contains both *video* and *service*.
  - If we rank all Boolean hits according to PageRank, then the Yahoo home page would be top-ranked.
  - Clearly not desirable
- In practice: rank according to weighted combination of raw text match, anchor text match, PageRank & other factors
- → see lecture on Learning to Rank □

Example web graph



PageRank					
d_0	0.05	d_1	0.04	d_2	0.11
d_3	0.25	d_4	0.21	d_5	0.04
d_6	0.31				

PageRank(d2)< PageRank(d6): why?

	a	h
d_0	0.10	0.03
d_1	0.01	0.04
d_2	0.12	0.33
d_3	0.47	0.18
d_4	0.16	0.04
d_5	0.01	0.04
d_6	0.13	0.35

highest in-degree: d_2, d_3, d_6
highest out-degree: d_2, d_6
highest PageRank: d_6
highest hub score: d_6 (close: d_2)
highest authority score: d_3 □

Transition (probability) matrix

$$
\begin{array}{cccccccc}
 & d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
d_0 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
d_1 & 0.00 & 0.50 & 0.50 & 0.00 & 0.00 & 0.00 & 0.00 \\
d_2 & 0.33 & 0.00 & 0.33 & 0.33 & 0.00 & 0.00 & 0.00 \\
d_3 & 0.00 & 0.00 & 0.00 & 0.50 & 0.50 & 0.00 & 0.00 \\
d_4 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\
d_5 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.50 & 0.50 \\
d_6 & 0.00 & 0.00 & 0.00 & 0.33 & 0.33 & 0.00 & 0.33
\end{array}
$$

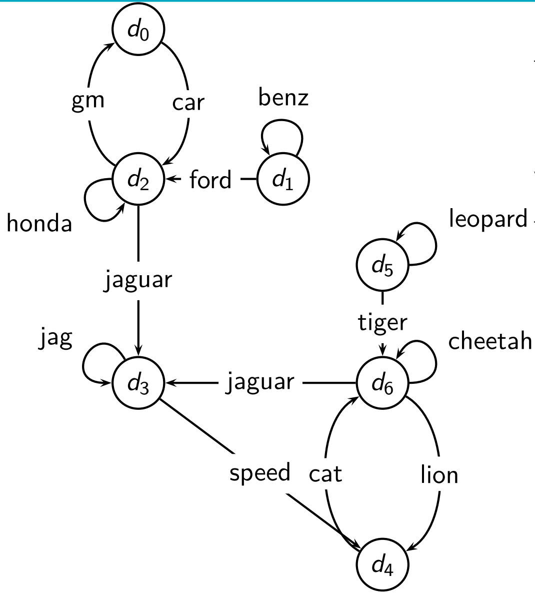
Transition matrix with teleporting, teleportation rate=0.14

$$
\begin{array}{cccccccc}
 & d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
d_0 & 0.02 & 0.02 & 0.88 & 0.02 & 0.02 & 0.02 & 0.02 \\
d_1 & 0.02 & 0.45 & 0.45 & 0.02 & 0.02 & 0.02 & 0.02 \\
d_2 & 0.31 & 0.02 & 0.31 & 0.31 & 0.02 & 0.02 & 0.02 \\
d_3 & 0.02 & 0.02 & 0.02 & 0.45 & 0.45 & 0.02 & 0.02 \\
d_4 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.88 \\
d_5 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.45 & 0.45 \\
d_6 & 0.02 & 0.02 & 0.02 & 0.31 & 0.31 & 0.02 & 0.31
\end{array}
$$

Power method vectors $\vec{x}P^k$

	$\vec{x}$	$\vec{x}P^1$	$\vec{x}P^2$	$\vec{x}P^3$	$\vec{x}P^4$	$\vec{x}P^5$	$\vec{x}P^6$	$\vec{x}P^7$	$\vec{x}P^8$	$\vec{x}P^9$	$\vec{x}P^{10}$	$\vec{x}P^{11}$	$\vec{x}P^{12}$	$\vec{x}P^{13}$
d_0	0.14	0.06	0.09	0.07	0.07	0.06	0.06	0.06	0.06	0.05	0.05	0.05	0.05	0.05
d_1	0.14	0.08	0.06	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
d_2	0.14	0.25	0.18	0.17	0.15	0.14	0.13	0.12	0.12	0.12	0.12	0.11	0.11	0.11
d_3	0.14	0.16	0.23	0.24	0.24	0.24	0.24	0.25	0.25	0.25	0.25	0.25	0.25	0.25
d_4	0.14	0.12	0.16	0.19	0.19	0.20	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21
d_5	0.14	0.08	0.06	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
d_6	0.14	0.25	0.23	0.25	0.27	0.28	0.29	0.29	0.30	0.30	0.30	0.30	0.31	0.31

Example web graph



PageRank					
d_0	0.05	d_1	0.04	d_2	0.11
d_3	0.25	d_4	0.21	d_5	0.04
d_6	0.31				

PageRank(d2)< PageRank(d6): why?

	a	h
d_0	0.10	0.03
d_1	0.01	0.04
d_2	0.12	0.33
d_3	0.47	0.18
d_4	0.16	0.04
d_5	0.01	0.04
d_6	0.13	0.35

highest in-degree: d_2, d_3, d_6
highest out-degree: d_2, d_6
highest PageRank: d_6
highest hub score: d_6 (close: d_2)
highest authority score: d_3 □

How important is PageRank?

- Frequent claim: PageRank is the most important component of web ranking.
- The reality:
  - There are several components that are at least as important: e.g., anchor text, phrases, proximity, tiered indexes . . .
  - Rumor has it that PageRank in its original form (as presented here) now has a negligible impact on ranking!
  - However, variants of a page's PageRank are still an essential part of ranking.
  - Adressing link spam is difficult and crucial. □

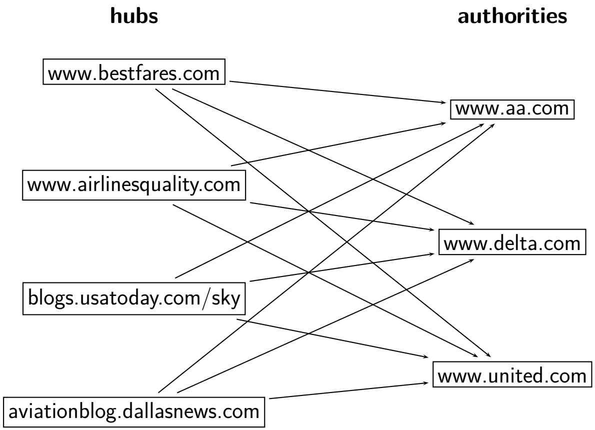
HITS – Hyperlink-Induced Topic Search

- Premise: there are two different types of relevance on the web.
- Relevance type 1: Hubs. A hub page is a good list of [links to pages answering the information need].
  - E.g., for query [chicago bulls]: Bob's list of recommended resources on the Chicago Bulls sports team
- Relevance type 2: Authorities. An authority page is a direct answer to the information need.
  - The home page of the Chicago Bulls sports team
  - By definition: Links to authority pages occur repeatedly on hub pages.
- Most approaches to search (including PageRank ranking) don't make the distinction between these two very different types of relevance. □

Hubs and authorities: Definition

- A good hub page for a topic links to many authority pages for that topic.
- A good authority page for a topic is linked to by many hub pages for that topic.
- Circular definition – we will turn this into an iterative computation. □

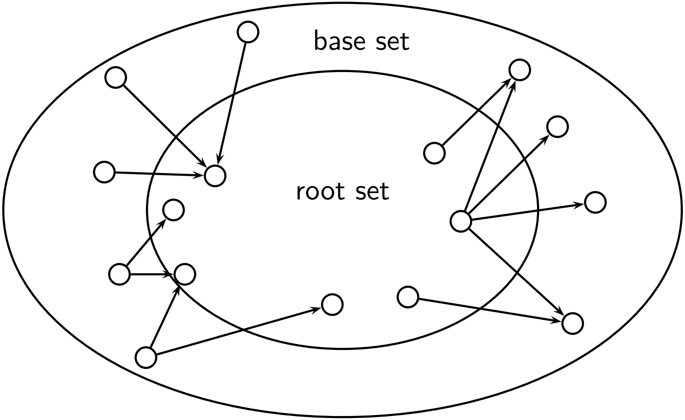
Example for hubs and authorities



How to compute hub and authority scores

- Do a regular web search first
- Call the search result the root set
- Find all pages that are linked to or link to pages in the root set
- Call this larger set the base set
- Finally, compute hubs and authorities for the base set (which we'll view as a small web graph) □

Root set and base set (1)



1) The root set 2) Nodes that root set nodes link to 3) Nodes that link to root set nodes 4) The base set

Root set and base set (2)

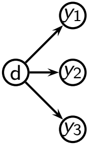
- Root set typically has 200–1,000 nodes.
- Base set may have up to 5,000 nodes.
- Computation of base set, as shown on previous slide:
  - Follow outlinks by parsing the pages in the root set
  - Find *d*'s inlinks by searching for all pages containing a link to *d* □

Hub and authority scores

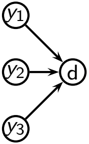
- Compute for each page d in the base set a hub score $h(d)$ and an authority score $a(d)$
- Initialization: for all d: $h(d) = 1$, $a(d) = 1$
- Iteratively update all $h(d), a(d)$
- After convergence:
  - Output pages with highest h scores as top hubs
  - Output pages with highest a scores as top authorities
  - So we output two ranked lists □

Iterative update

- For all d: $h(d) = \sum_{d \mapsto y} a(y)$



- For all d: $a(d) = \sum_{y \mapsto d} h(y)$



- Iterate these two steps until convergence □

Details

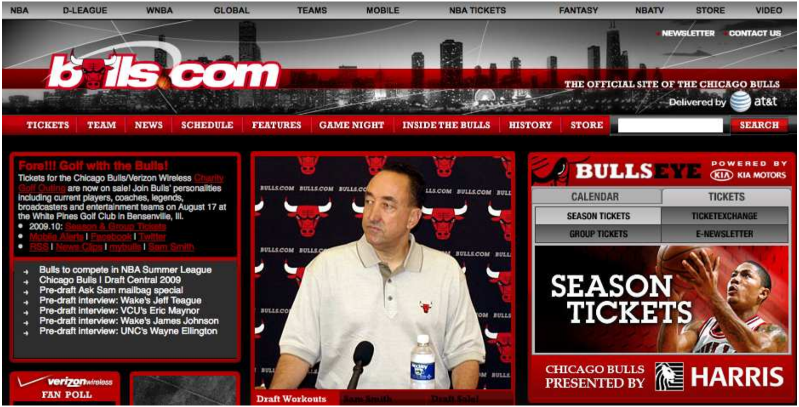
- Scaling
  - To prevent the $a()$ and $h()$ values from getting too big, can scale down after each iteration
  - Scaling factor doesn't really matter.
  - We care about the relative (as opposed to absolute) values of the scores.
- In most cases, the algorithm converges after a few iterations. □

Authorities for query [Chicago Bulls]

0.85	www.nba.com/bulls
0.25	www.essex1.com/people/jmiller/bulls.htm “da Bulls”
0.20	www.nando.net/SportServer/basketball/nba/chi.html “The Chicago Bulls”
0.15	users.aol.com/rynocub/bulls.htm “The Chicago Bulls Home Page”
0.13	www.geocities.com/Colosseum/6095 “Chicago Bulls”

(Ben-Shaul et al, WWW8)

The authority page for [Chicago Bulls]



Fore!!! Golf with the Bulls!

Tickets for the Chicago Bulls/Verizon Wireless Charity Golf Outing are now on sale! Join Bulls' personalities including current players, coaches, legends, broadcasters and entertainment teams on August 17 at the White Pines Golf Club in Bensenville, Ill.

- 2009.10: Season & Group Tickets
- Mobile Alerts | Facebook | Twitter
- RSS | News Clips | mybulls | Sam Smith

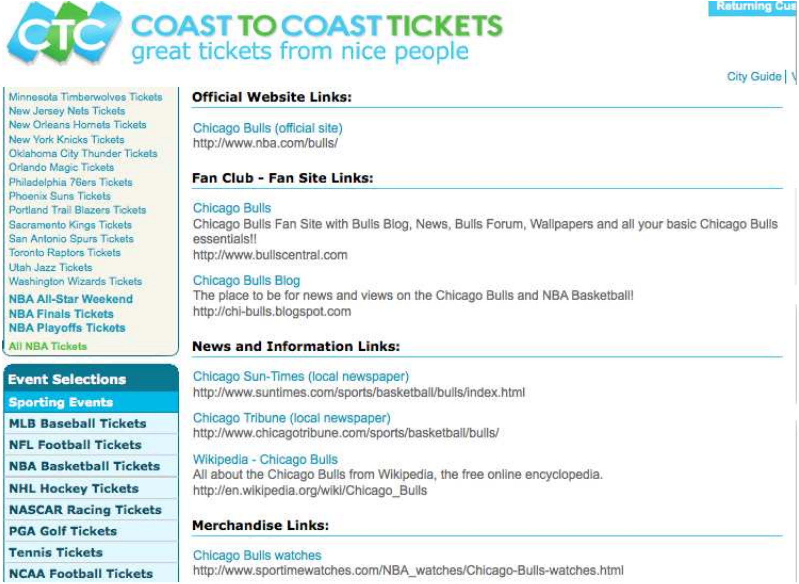
- Bulls to compete in NBA Summer League
- Chicago Bulls | Draft Central 2009
- Pre-draft Ask Sam mailbag special
- Pre-draft interview: Wake's Jeff Teague
- Pre-draft interview: VCU's Eric Maynor
- Pre-draft interview: Wake's James Johnson
- Pre-draft interview: UNC's Wayne Ellington

Hubs for query [Chicago Bulls]

1.62	www.geocities.com/Colosseum/1778 “Unbelieveabulls!!!!!”
1.24	www.webring.org/cgi-bin/webring?ring=chbulls “Erin’s Chicago Bulls Page”
0.74	www.geocities.com/Hollywood/Lot/3330/Bulls.html “Chicago Bulls”
0.52	www.nobull.net/web_position/kw-search-15-M2.htm “Excite Search Results: bulls”
0.52	www.halcyon.com/wordsltd/bball/bulls.htm “Chicago Bulls Links”

(Ben-Shaul et al, WWW8)

A hub page for [Chicago Bulls]



COAST TO COAST TICKETS
great tickets from nice people

Returning Cus

City Guide | V

Minnesota Timberwolves Tickets
New Jersey Nets Tickets
New Orleans Hornets Tickets
New York Knicks Tickets
Oklahoma City Thunder Tickets
Orlando Magic Tickets
Philadelphia 76ers Tickets
Phoenix Suns Tickets
Portland Trail Blazers Tickets
Sacramento Kings Tickets
San Antonio Spurs Tickets
Toronto Raptors Tickets
Utah Jazz Tickets
Washington Wizards Tickets
NBA All-Star Weekend
NBA Finals Tickets
NBA Playoffs Tickets
All NBA Tickets

Event Selections

Sporting Events

- **MLB Baseball Tickets**
- **NFL Football Tickets**
- **NBA Basketball Tickets**
- **NHL Hockey Tickets**
- **NASCAR Racing Tickets**
- **PGA Golf Tickets**
- **Tennis Tickets**
- **NCAA Football Tickets**

Official Website Links:

Chicago Bulls (official site)
http://www.nba.com/bulls/

Fan Club - Fan Site Links:

Chicago Bulls
Chicago Bulls Fan Site with Bulls Blog, News, Bulls Forum, Wallpapers and all your basic Chicago Bulls essentials!!
http://www.bullscentral.com

Chicago Bulls Blog
The place to be for news and views on the Chicago Bulls and NBA Basketball!
http://chi-bulls.blogspot.com

News and Information Links:

Chicago Sun-Times (local newspaper)
http://www.suntimes.com/sports/basketball/bulls/index.html

Chicago Tribune (local newspaper)
http://www.chicagotribune.com/sports/basketball/bulls/

Wikipedia - Chicago Bulls
All about the Chicago Bulls from Wikipedia, the free online encyclopedia.
http://en.wikipedia.org/wiki/Chicago_Bulls

Merchandise Links:

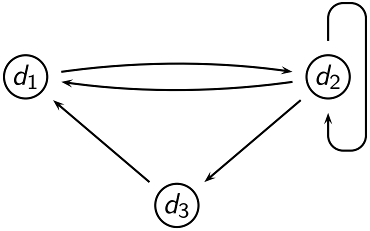
Chicago Bulls watches
http://www.sportimewatches.com/NBA_watches/Chicago-Bulls-watches.html

Hubs & Authorities: Comments

- HITS can pull together good pages regardless of page content.
- Once the base set is assembled, we only do link analysis, no text matching.
- Pages in the base set often do not contain any of the query words.
- In theory, an English query can retrieve Japanese-language pages!
  - If supported by the link structure between English and Japanese pages
- Danger: topic drift – the pages found by following links may not be related to the original query. □

Proof of convergence

- We define an $N \times N$ adjacency matrix A. (We called this the link matrix earlier.)
- For $1 \leq i, j \leq N$, the matrix entry A_{ij} tells us whether there is a link from page i to page j ($A_{ij} = 1$) or not ($A_{ij} = 0$).
- Example:



$$\begin{array}{cccc} & d_1 & d_2 & d_3 \\ d_1 & 0 & 1 & 0 \\ d_2 & 1 & 1 & 1 \\ d_3 & 1 & 0 & 0 \end{array}$$

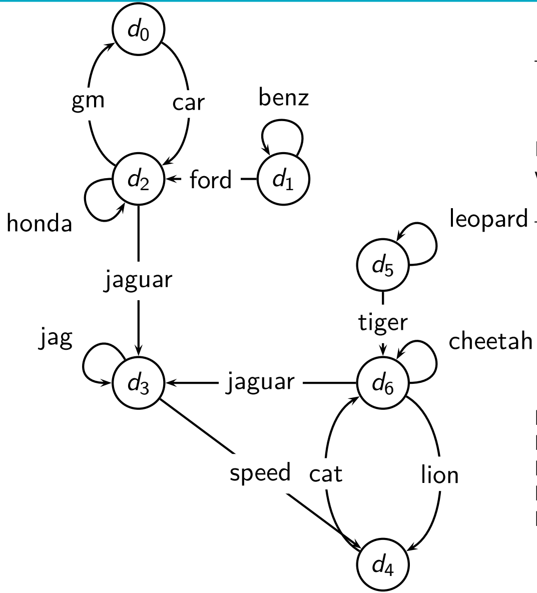
Write update rules as matrix operations

- Define the hub vector $\vec{h} = (h_1, \ldots, h_N)$ as the vector of hub scores. h_i is the hub score of page d_i.
- Similarly for $\vec{a}$, the vector of authority scores
- Now we can write $h(d) = \sum_{d \mapsto y} a(y)$ as a matrix operation: $\vec{h} = A\vec{a}$ ...
- ... and we can write $a(d) = \sum_{y \mapsto d} h(y)$ as $\vec{a} = A^T \vec{h}$
- HITS algorithm in matrix notation:
  - Compute $\vec{h} = A\vec{a}$
  - Compute $\vec{a} = A^T \vec{h}$
  - Iterate until convergence □

HITS as eigenvector problem

- HITS algorithm in matrix notation. Iterate:
  - Compute $\vec{h} = A\vec{a}$
  - Compute $\vec{a} = A^T\vec{h}$
- By substitution we get: $\vec{h} = AA^T\vec{h}$ and $\vec{a} = A^TA\vec{a}$
- Thus, $\vec{h}$ is an eigenvector of AA^T and $\vec{a}$ is an eigenvector of A^TA.
- So the HITS algorithm is actually a special case of the power method and hub and authority scores are eigenvector values.
- HITS and PageRank both formalize link analysis as eigenvector problems. □

Example web graph



PageRank					
d_0	0.05	d_1	0.04	d_2	0.11
d_3	0.25	d_4	0.21	d_5	0.04
d_6	0.31				

PageRank(d2)< PageRank(d6): why?

	a	h
d_0	0.10	0.03
d_1	0.01	0.04
d_2	0.12	0.33
d_3	0.47	0.18
d_4	0.16	0.04
d_5	0.01	0.04
d_6	0.13	0.35

highest in-degree: d_2, d_3, d_6
highest out-degree: d_2, d_6
highest PageRank: d_6
highest hub score: d_6 (close: d_2)
highest authority score: d_3 □

Raw matrix A for HITS

We double-weight links whose anchors contain query word:

$$\begin{array}{cccccccc}
 & d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
d_0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
d_1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
d_2 & 1 & 0 & 1 & 2 & 0 & 0 & 0 \\
d_3 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
d_4 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
d_5 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
d_6 & 0 & 0 & 0 & 2 & 1 & 0 & 1
\end{array}$$

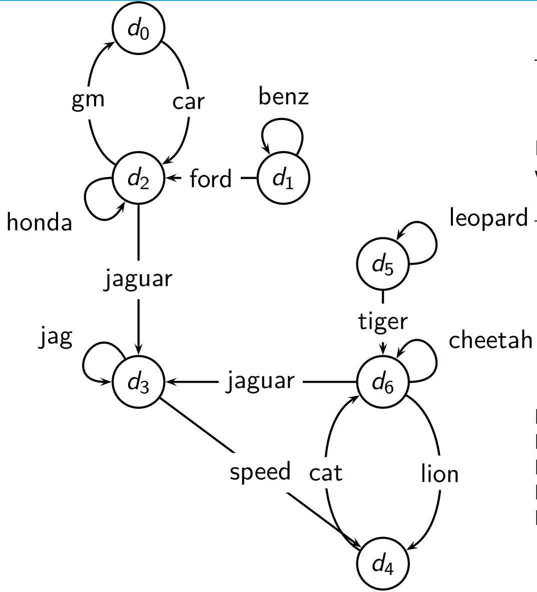
Hub vectors $h_0, \vec{h}_i = \frac{1}{d_i} A \cdot \vec{a}_i, i \geq 1$

	$\vec{h}_0$	$\vec{h}_1$	$\vec{h}_2$	$\vec{h}_3$	$\vec{h}_4$	$\vec{h}_5$
d_0	0.14	0.06	0.04	0.04	0.03	0.03
d_1	0.14	0.08	0.05	0.04	0.04	0.04
d_2	0.14	0.28	0.32	0.33	0.33	0.33
d_3	0.14	0.14	0.17	0.18	0.18	0.18
d_4	0.14	0.06	0.04	0.04	0.04	0.04
d_5	0.14	0.08	0.05	0.04	0.04	0.04
d_6	0.14	0.30	0.33	0.34	0.35	0.35

Authority vectors $\vec{a}_i = \frac{1}{c_i} A^T \cdot \vec{h}_{i-1}, i \geq 1$

	$\vec{a}_1$	$\vec{a}_2$	$\vec{a}_3$	$\vec{a}_4$	$\vec{a}_5$	$\vec{a}_6$	$\vec{a}_7$
d_0	0.06	0.09	0.10	0.10	0.10	0.10	0.10
d_1	0.06	0.03	0.01	0.01	0.01	0.01	0.01
d_2	0.19	0.14	0.13	0.12	0.12	0.12	0.12
d_3	0.31	0.43	0.46	0.46	0.46	0.47	0.47
d_4	0.13	0.14	0.16	0.16	0.16	0.16	0.16
d_5	0.06	0.03	0.02	0.01	0.01	0.01	0.01
d_6	0.19	0.14	0.13	0.13	0.13	0.13	0.13

Example web graph



PageRank					
d_0	0.05	d_1	0.04	d_2	0.11
d_3	0.25	d_4	0.21	d_5	0.04
d_6	0.31				

PageRank(d2)< PageRank(d6): why?

	a	h
d_0	0.10	0.03
d_1	0.01	0.04
d_2	0.12	0.33
d_3	0.47	0.18
d_4	0.16	0.04
d_5	0.01	0.04
d_6	0.13	0.35

highest in-degree: d_2, d_3, d_6
highest out-degree: d_2, d_6
highest PageRank: d_6
highest hub score: d_6 (close: d_2)
highest authority score: d_3 □

PageRank vs. HITS: Discussion

- PageRank can be precomputed, HITS has to be computed at query time.
    - HITS is too expensive in most application scenarios.
- PageRank and HITS make two different design choices concerning (i) the eigenproblem formalization (ii) the set of pages to apply the formalization to.
- These two are orthogonal.
    - We could also apply HITS to the entire web and PageRank to a small base set.
- Claim: On the web, a good hub almost always is also a good authority.
- The actual difference between PageRank ranking and HITS ranking is therefore not as large as one might expect. □

Exercise

- Why is a good hub almost always also a good authority? □

Take-away today

- Anchor text: What exactly are links on the web and why are they important for IR?
- Citation analysis: the mathematical foundation of PageRank and link-based ranking
- PageRank: the original algorithm that was used for link-based ranking on the web
- Hubs & Authorities: an alternative link-based ranking algorithm

Resources

- Chapter 21 of IIR
- Resources at `https://www.fi.muni.cz/~sojka/PV211/` and `http://cislmu.org`, materials in MU IS and FI MU library
  - American Mathematical Society article on PageRank (popular science style)
  - Jon Kleinberg's home page (main person behind HITS)
  - A Google bomb and its defusing
  - Google's official description of PageRank: *PageRank reflects our view of the importance of web pages by considering more than 500 million variables and 2 billion terms. Pages that we believe are important pages receive a higher PageRank and are more likely to appear at the top of the search results.*