Retrieval models II

IN4325 – Information Retrieval

Assignment 1

- Deadline Wednesday means as long as it is Wednesday you are still on time
  - In practice, any time before I make it to the office on Thursday (~8am) is okay
- Amazon #instances still limited, please keep to 2-3 instances per run with **m1.xlarge**
  - 15GB memory, 8 EC2 compute units ➔ the same as 8x m1.small
  - Between 2-6 jobs ran yesterday at any time
- Hadoop is difficult to learn, but well worth it ☺

  ➔LinkedIn: do a job search on „Hadoop" or „Big data"

Assignment I

```
public class CategoryPerPageMapper extends Mapper<Object, Text, Text, Text>
{
        public void map(Object key, Text value, Context context) throws Exception
        {
                SAXBuilder builder = new SAXBuilder();
                Pattern catPattern = Pattern.compile("\\[\\[[cC]ategory:.*?]]");
                Text emptyText = new Text("");

                try
                {
                        String xml = value.toString();
                        Reader in = new StringReader(xml);
                        String content ="";
                        Text title = new Text();
                         Document doc = builder.build(in);
                        …
                }
                …
        }
}
```

Assignment I

```
public class CategoryPerPageMapper extends Mapper<Object, Text, Text, Text>
{
        SAXBuilder builder = new SAXBuilder();
        Pattern catPattern = Pattern.compile("\\[\\[[cC]ategory:.*?]]");
        Text emptyText = new Text("");

        public void map(Object key, Text value, Context context) throws Exception
        {
                try
                {
                        String xml = value.toString();
                        Reader in = new StringReader(xml);
                        String content ="";
                        Text title = new Text();
                         Document doc = builder.build(in);
                        …
                }
                …
        }
}
```

Assignment I

- *Data-Intensive Text Processing with MapReduce* by Jimmy Lin et al
  - Highly recommended reading!
  - ~170 pages full of useful knowledge
  - A lot of information about design patterns in MapReduce
  - No cookbook (no source code)

Last time

- Term weighting strategies
- Vector space model
- Pivoted document length normalization

Today

- A look at relevance feedback
- The cluster hypothesis
- A first look at probabilistic models
  - Language modeling!

A vector space model for automatic indexing.
A classic paper by G. Salton et al. from 1975.

Vector space model

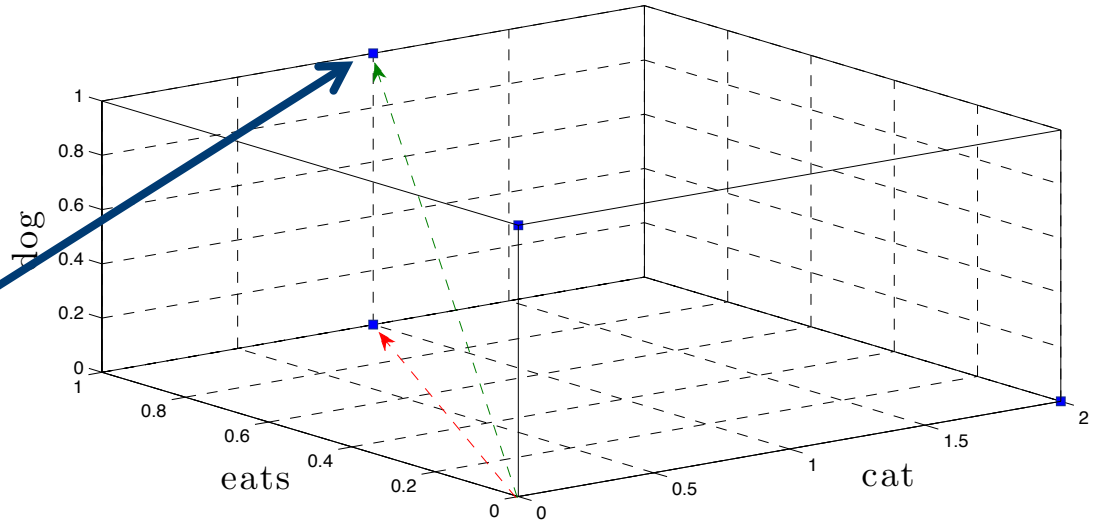
In one slide

$$tf - idf_{t,D} = tf_{t,D} \times idf_t$$

retrieval by $\cos\theta$

- A classic information retrieval model
- Corpus: a set of vectors in a vector space with one dimension for each term
- *S(Q,D)*: cosine similarity between query and document vector

$$D_1 = \{cat, eats\}$$
$$D_2 = \{dog\}$$
$$D_3 = \{dog, eats, cat\}$$
$$D_4 = \{cat, cat\}$$



The problem of synonymy

- Synonymy: different terms with the same meaning

synset

Noun

- S: (n) **king**, male monarch, Rex (a male sovereign; ruler of a kingdom)
- S: (n) **king**, queen, world-beater (a competitor who holds a preeminent position)
- S: (n) baron, big businessman, business leader, **ki** power, top executive, tycoon (a very wealthy or pc *oil baron"*
- S: (n) **king** (preeminence in a particular category c *is the king of beasts"*
- S: (n) **King**, Billie Jean King, Billie Jean Moffitt King tennis player (born in 1943))
- S: (n) **King**, B. B. King, Riley B King (United States guitar player and singer of the blues (born in 1925))
- S: (n) **King**, Martin Luther King, Martin Luther King Jr. (United States charismatic civil rights leader and Baptist minister who campaigned against the segregation of Blacks (1929-1968))
- S: (n) **king** (a checker that has been moved to the opponent's first row where it is promoted to a piece that is free to move either forward or backward)

Users try to combat these situations by adding terms they see in the retrieved documents to the query

WordNet 3.1

http://wordnet.princeton.edu/

Automatic query reformulation independent of the user/results

Word sense disambiguation

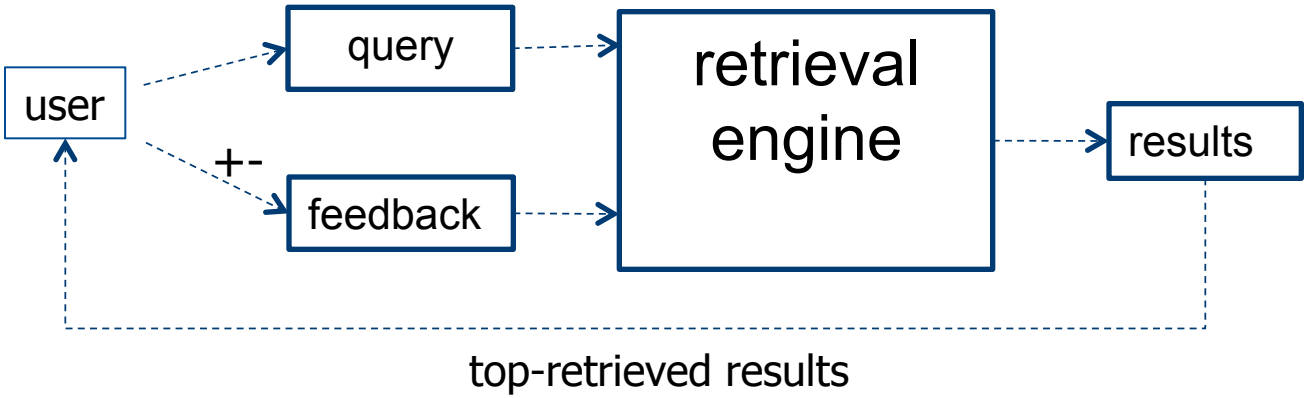
- Query expansion: additional query terms derived from a thesaurus or WordNet
  "King 1968" ➔ (King 1968), (Martin Luther King), (Martin Luther King jr.)
- Query expansion through automatic thesaurus generation
- Query reformulation by spelling correction
  "Martin Luter King" ➔ Martin Luther King

Automatic query reformulation wrt. the user/results

- Given an initial set of results, **relevance feedback** can be
  - Obtained explicitly/interactively (ask the user!)
    - +/- voting on the top results
  - Inferred (observe the user!)
    - How long does the user spend on viewing document X?
    - What is his eye movement on the result page?
  - Assumed (observe the top results!)
    - Also called blind feedback, pseudo-relevance feedback
    - The system takes the top returned results and assumes all of them to be relevant

Basic relevance feedback loop

evolving information needs served



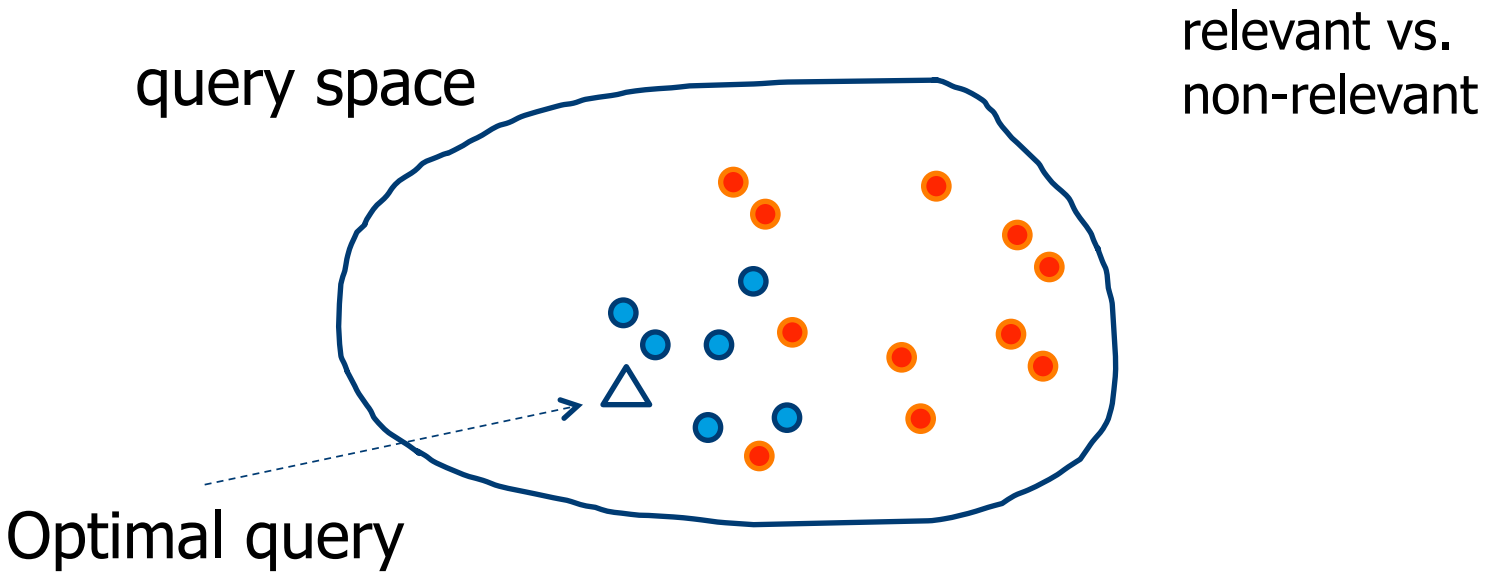
iterative refinement

Idea: users do not know the corpus (difficult to come up with all possible query reformulations), easy though for the user to decide if a given document is relevant

Rocchio algorithm

An algorithm to take advantage of relevance feedback

- Incorporates relevance feedback into the vector space model



Rocchio algorithm

Theory

- Goal: query vector with max. similarity to the relevant documents and min. similarity to the non-relevant documents

$$\vec{q} = \arg\max_{\vec{q}}[sim(\vec{q}, C_r) - sim(\vec{q}, C_{nr})]$$

cosθ

- Cosine similarity based:

$$\vec{q} = \frac{1}{|C_r|}\sum_{\vec{d}_j \in C_r} \vec{d}_j - \frac{1}{|C_{nr}|}\sum_{\vec{d}_j \in C_{nr}} \vec{d}_j$$

Optimal query is the vector difference between centroids of the relevant and non-relevant documents

Rocchio algorithm

Developed 1971

- Given: a query and some knowledge about relevant/non-relevant documents

$$\vec{q}_m = \alpha q_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

original query known (non-)relevant documents

- Model parameters: α, β, γ

Rocchio algorithm

Q: what are good weights for "Find similar pages"?

- Given: a query and some knowledge about relevant/non-relevant documents

$$\vec{q}_m = \alpha q_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

  - Higher alpha: large influence of original query
  - Higher beta: large influence of the rel docs
  - Higher gamma: go away from non-rel docs

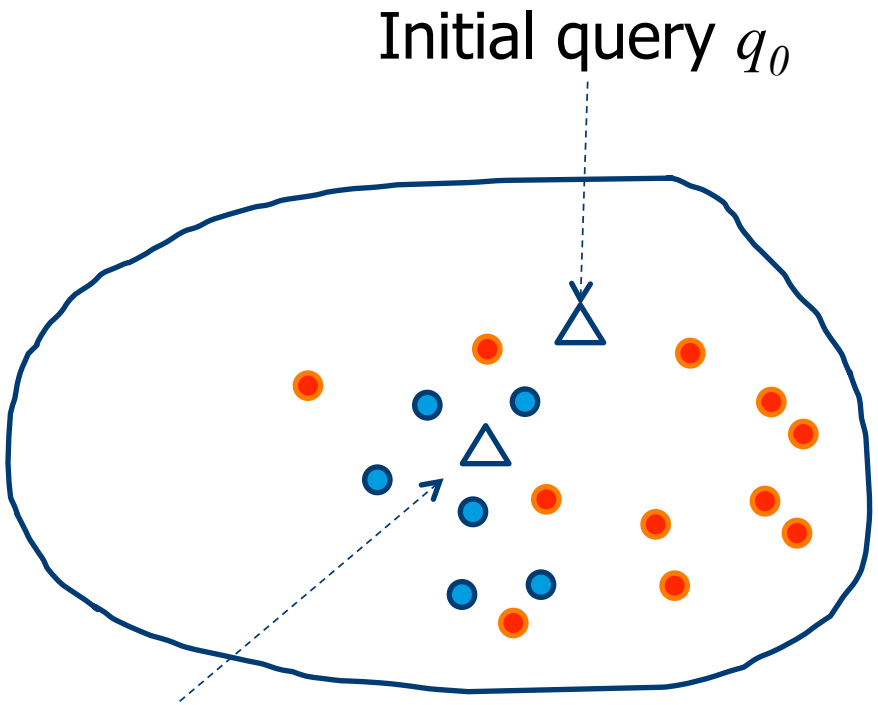
- Q: should the weights change over time?

X terms are added to the expanded query

Rocchio algorithm

Graphically

Due to the subtraction we may end up with negative term weights; ignore those.



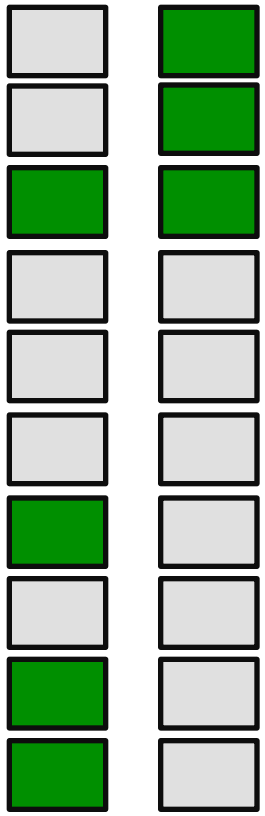
Use revised query for retrieval

Effects of relevance feedback

- Improves precision and recall

$$precision = \frac{\#\,rel.\,retrieved}{\#\,retrieved} \quad recall = \frac{\#\,rel.\,retrieved}{\#\,relevant \in corpus}$$

- Most important for recall
- Positive feedback more valuable than negative feedback (why?)
  - $\gamma < \beta$
  - Common values: $\alpha = 1.0, \beta = 0.75, \gamma = 0.15$
  - Often only positive feedback: $\gamma = 0$



	S_4	S_5
P@10	0.4	0.3
R=5, Recall	0.8	0.6

Rocchio algorithm

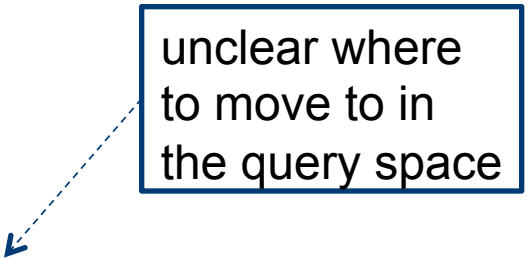
Empirical evaluation: Buckley et al. 1994 [7]

- What influence does Rocchio have on the effectiveness of a system?
- TREC routing task evaluation
  - One long standing information need and new documents appear continuously, hundreds of positive qrels
  - Query is updated depending on user feedback

- Effects studied of the #known relevant documents and #terms added to the query
  - ➔the more known relevant documents the better
  - ➔the more terms added to the query the better

Relevance feedback

Issues



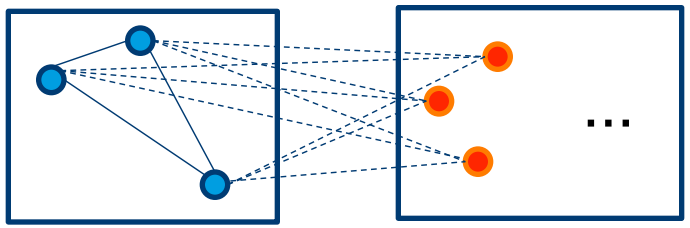
- If the initial query performs poorly, an expanded version is likely to perform poorly too, even with relevance feedback
  - Misspellings
  - Cross-language retrieval (document vectors in other languages are not nearby)
  - Total mismatch of searcher's vocabulary and collection vocabulary
- Relevance feedback can only work, if the cluster hypothesis holds

Cluster hypothesis

Keith van Rijsbergen [2]

- "Closely associated documents tend to be relevant to the same requests."
- Basic assumption of IR systems: relevant documents are more similar to each other than they are to non-relevant documents

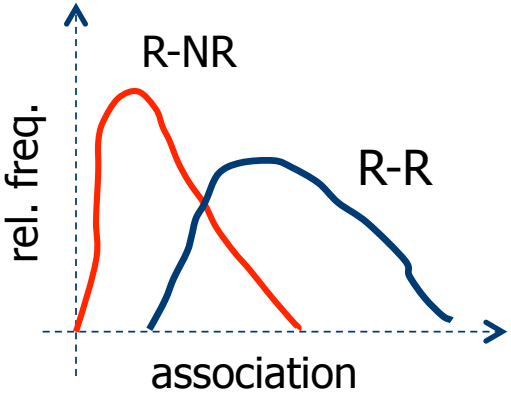
association between all document pairs (R-R), (R-NR)



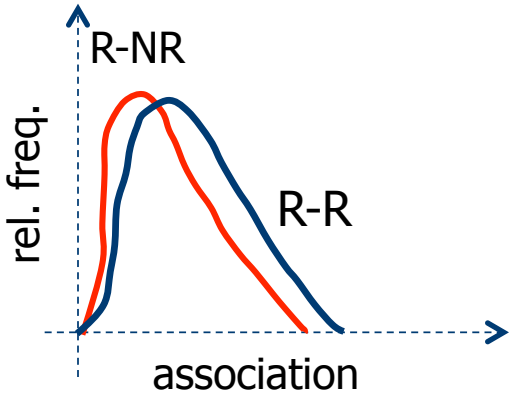
Cluster hypothesis

Keith van Rijsbergen [2]

- Plot the relative frequency against the strength of association



good separation
high R-R association



poor separation

Cluster hypothesis

Keith van Rijsbergen [2]

- Clustering methods should
  - Produce a stable clustering: no sudden changes when items are added
  - Be tolerant to errors: small errors should lead to small changes in clustering
  - Be independent of the initial item ordering
- Clustering fails
  - Subsets of documents have very different important terms (Burma vs. Myanmar, TREC vs. CLEF)
  - Queries that are inherently disjunctive (there is no document that contains all terms)
  - Polysemy

semantic approach: link both terms to the same concept in Wikipedia

Relevance feedback & users

- Not popular with users (4% on Excite search engine, 2000)
  - Explicit feedback is difficut to get (makes interaction with the search interface cumbersome)
  - Users get annoyed if after a feedback round the results do not (seemingly) get better
- Efficiency issues
  - The longer the query, the longer it takes to process it
  - Use top weighted query terms only, though according to Buckley [7], the more terms the better
    - Up to 4000 terms per query

Pseudo-relevance feedback

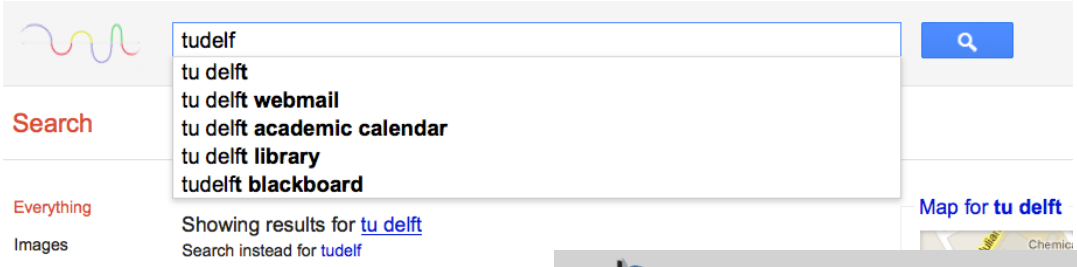
- Blind relevance feedback
- No user

- Proxy for relevance judgments: assume top k documents to be relevant, same computation otherwise

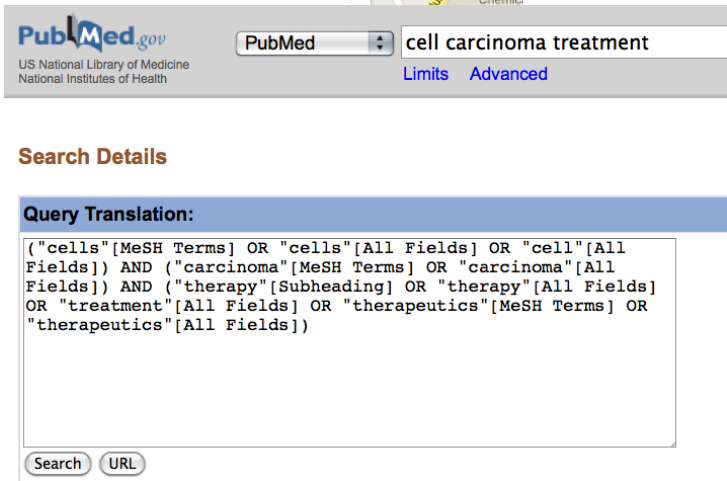
- Works well in situations where many relevant documents appear in the top-k
  - If the query performs well, pseudo-relevance feedback improves it further
  - If the initial results are poor, they remain poor

Query expansion

- Users explicitly provide extra terms (instead of feedback on documents)



- Thesaurus-based expansion for the user
  - No user input
  - E.g. PubMed automatically expands the query



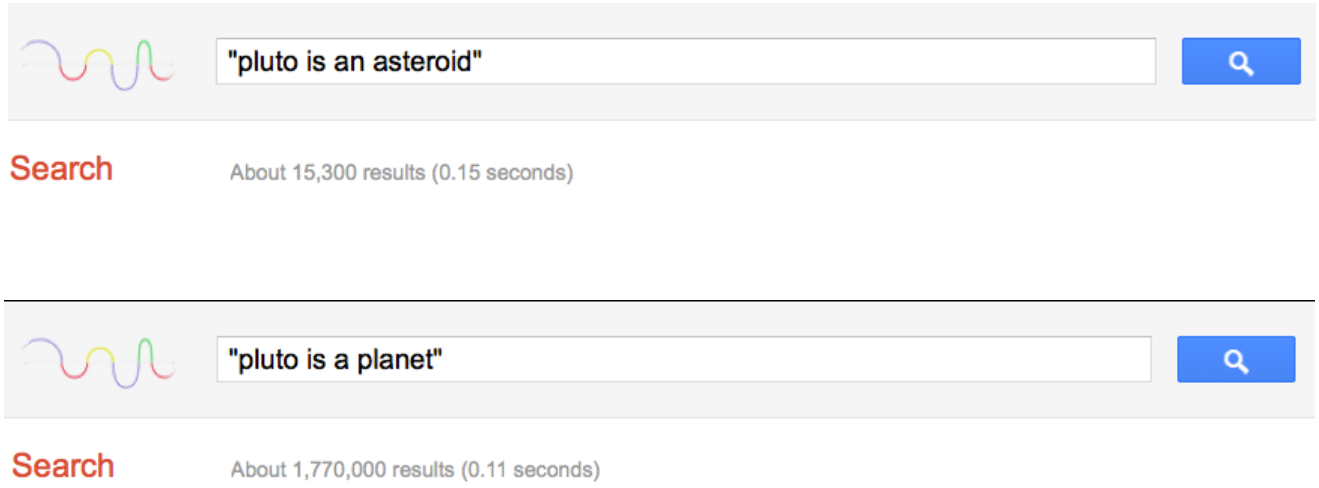
Query expansion

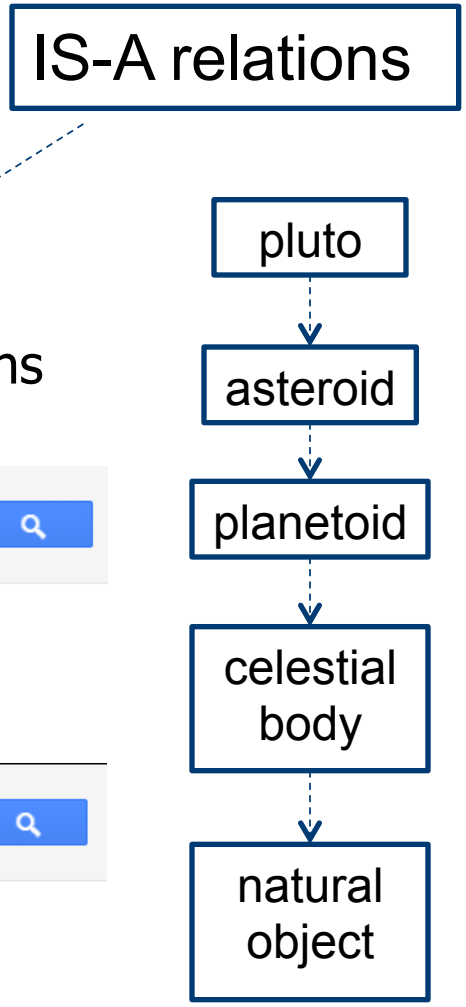
- Thesauri
  - Can be built manually (time, money, people)
  - Can be built automatically (co-occurrence statistics, large-scale corpora, processing power)
- Query reformulations based on log data
  - Exploits 'manual' data (*your* reformulation ideas)
  - What about long-tail queries?

Hearst patterns

Hearst [8]

- Use simple patterns to decide on taxonomic relations





Pluto must be a planet!

Hearst patterns

Hearst [8]

- NP_0 *such as* $\{NP_1, NP_2, \ldots, (and|or)\}$ NP_n
  - "American cars such as " ➔ Chevrolet, Pontiac
- *Such NP as {NP, }* {(or|and)} NP*
  - "such colors as red or orange"
- *NP {, NP}* {,} or other NP*
- *NP {, NP}* {,} and other NP*
- *NP {,} including {NP,}* {or|and} NP*
- *NP {,} especially {NP,}* {or|and} NP*

- Bootstrapping the discovery of new patterns
  1. Use standard patterns to find entity pairs
  2. Where else do these pairs co-occur and with what terms?

A new type of retrieval models

Probabilistic models

- Binary independence model
- 2-Poisson model
- Language modeling *
- …

Basic probabilities

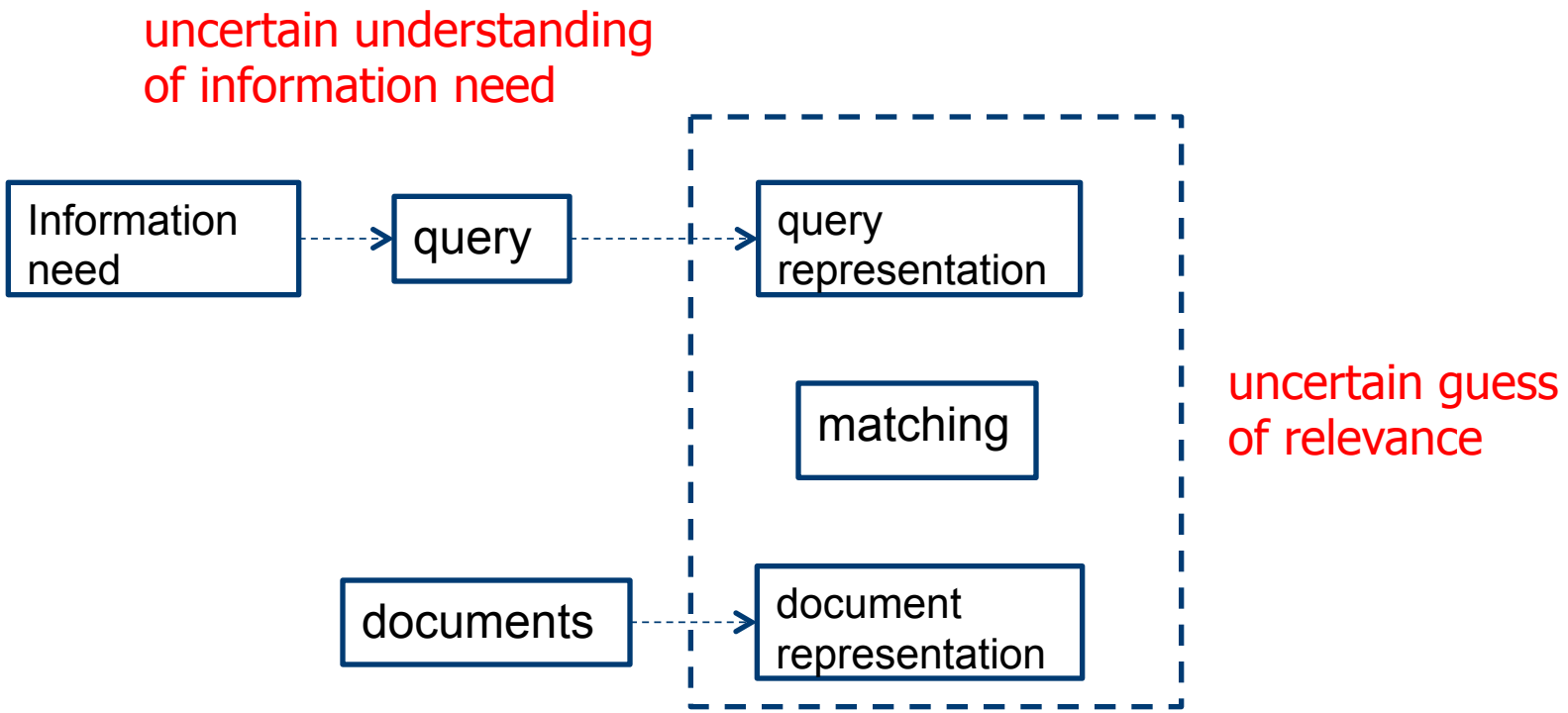
$$P(A,B) = P(A \mid B)P(B) = P(B \mid A)P(A)$$ chain rule

joint probability cond. probability

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$$ Bayes' Theorem

$$P(B) = P(\overline{A},B) + P(A,B)$$ partition rule

Probabilistic models in IR



$$P(D \mid Q) = \frac{P(Q \mid D) \times P(D)}{P(Q)}$$

central equation to language modeling

Probabilistic models in IR

- Probability models have a sounder theoretical foundation (reasoning under uncertainty) than boolean or vector space models
  - And they usually perform better
- **Documents are ranked according to their probability of being relevant, they are ranked accoring to** $P(D|Q)$
- Many models exist
- We start with language modeling

Language modeling

Ponte & Croft, 1998.

$$P(D \mid Q) = \frac{P(Q \mid D) \times P(D)}{P(Q)}$$

- Idea: rank the documents by the **likelihood of the query** according to the language model

"CSKA"

If we throw all terms into a bag and randomly draw terms, for which document is the probability greater of drawing CSKA?

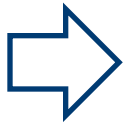
Pontus Wernbloom's last-minute volley earned a draw to keep CSKA Moscow in the hunt in their last-16 Champions League match against Real Madrid.
Real had looked set to return to Spain with a lead at the tie's halfway stage, but paid for their wastefulness.
Cristiano Ronaldo fired under Sergei Chepchugov to open the scoring but fluffed a fine 84th-minute chance.
CSKA had rarely threatened but Wernbloom crashed home with virtually the game's final kick to snatch a draw.

Greece has avoided a nightmare scenario by agreeing to a 130bn euros (£110bn; $170bn) bailout deal, Finance Minister Evangelos Venizelos has said.
He said the deal was probably the most important in Greece's post-war history. The cabinet was meeting to discuss how to pass the reforms stipulated by international lenders, which include huge spending cuts and beefed-up monitoring by eurozone officials. Trade unions have called strikes and protests for Wednesday.

Language modeling

- Query Q is "generated" by a **probabilistic model** based on document D

Pontus Wernbloom's last-minute volley earned a draw to keep CSKA Moscow in the hunt in their last-16 Champions League match against Real Madrid.
Real had looked set to return to Spain with a lead at the tie's halfway stage, but paid for their wastefulness.
Cristiano Ronaldo fired under Sergei Chepchugov to open the scoring but fluffed a fine 84th-minute chance.
CSKA had rarely threatened but Wernbloom crashed home with virtually the game's final kick to snatch a draw.



to	0.0617
the	0.0493
a	0.0493
s	0.0370
but	0.0370
in	0.0246
their	0.0246
with	0.0246
wernbloom	0.0246
real	0.0246
had	0.0246
draw	0.0246
cska	0.0246
…	
moscow	0.0123

Maximum likelihood estimate

posterior likelihood prior

$$P(D \mid Q) \propto P(Q \mid D) \times P(D)$$

Goal! Hard!

Relatively easy!

Ignore for now!

$P(Q)$ *has no use*

Language modeling

uniform

$$P(D \mid Q) \propto P(Q \mid D) \times P(D)$$

Unigram (multinomial) language model:

$$P(Q \mid D) = \prod_i P(q_i \mid D)$$

Assuming term independence makes the problem more tractable

➔Problem reduced to estimating $P(q_i|D)$

term frequencies

What about the IDF component? (none in LM)

Language modeling

Smoothing

$$P(Q \mid D) = \prod_i P(q_i \mid D)$$

$$Q = \{CSKA, Moscow, Greece\}$$

$$P(Q \mid D) = P(CSKA \mid D)P(Moscow \mid D)P(Greece \mid D)$$

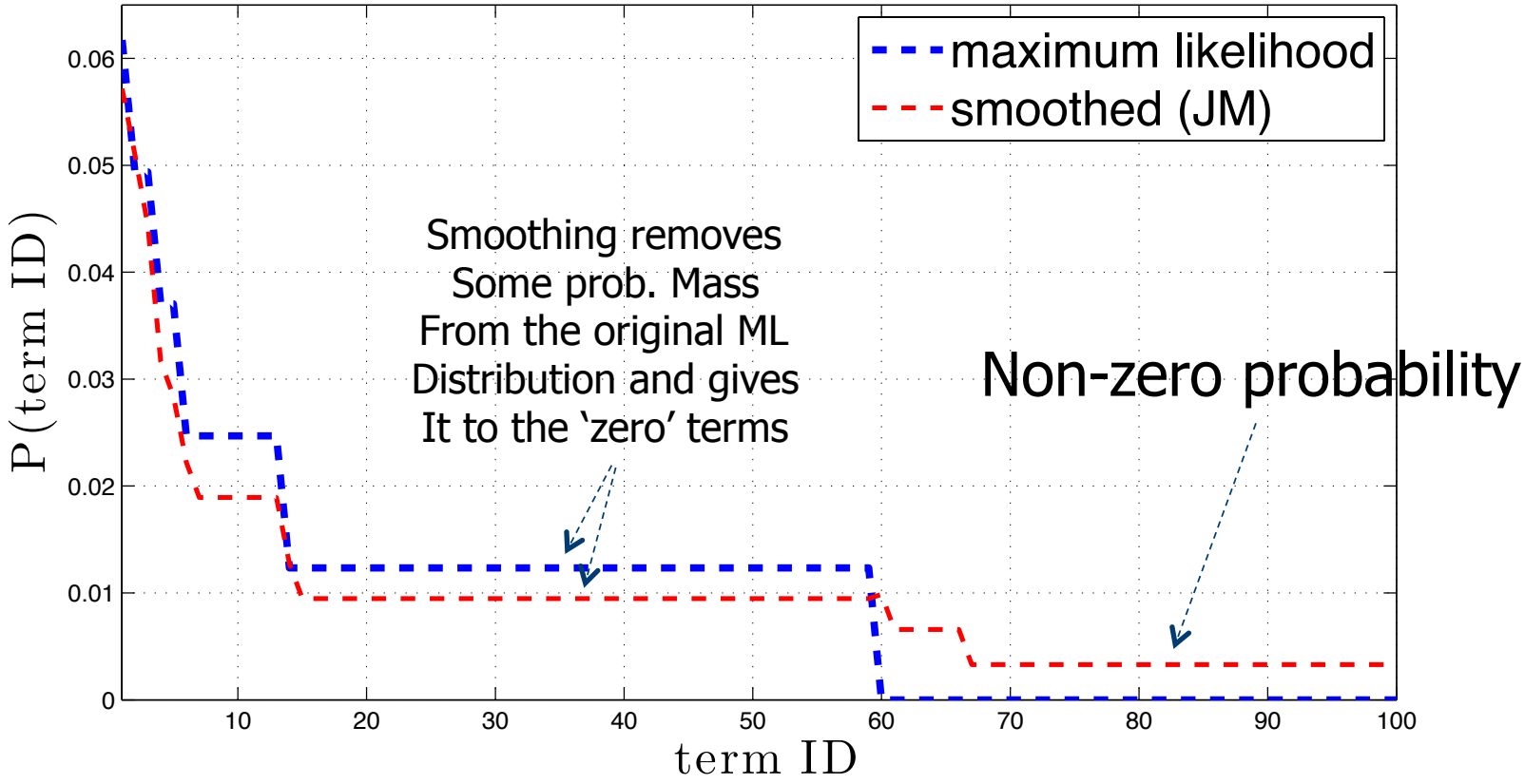
$$P(Q \mid D) = 0.0246 \times 0.0123 \times 0$$

$$P(Q \mid D) = 0$$

- Smoothing methods 'smooth' the document's language model (maximum likelihood prob. distribution) to avoid terms with zero probability

Language modeling

Smoothing



Language modeling

Smoothing

- Smoothing methods consider 2 distributions
  - Model $P_{seen}(t|D)$ for terms occurring in the document
  - Model $P_{unseen}(t|D)$ for "unseen" terms not in the document

$$\log P(Q \mid D) = \sum_i \log P(q_i \mid D)$$

$$= \sum_{i:c(q_i;D)>0} \log P_{seen}(q_i \mid D) + \sum_{i:c(q_i;D)=0} \log P_{unseen}(q_i \mid D)$$

$$+ \frac{\sum_{i:c(q_i;D)>0} \log P_{unseen}(q_i \mid D)}{\sum_{i:(q_i;D)>0} \log P_{unseen}(q_i \mid D)}$$

$c(w;D)$ *count of word w in D*

$$= \sum_{i:c(q_i|D)>0} \log \frac{P_{seen}(q_i \mid D)}{P_{unseen}(q_i \mid D)} + \sum_i \log P_{unseen}(q_i \mid D)$$

Language modeling

Smoothing

- $P_{unseen}(w)$ is typically proportional to the general frequency of w

$$P_{unseen}(q_i \mid D) = \alpha_d P(q_i \mid \mathbb{C})$$

document dependent constant

collection language model

(previous slide)

$$\log P(Q \mid D) = \sum_{i:c(q_i \mid D)>0} \log \frac{P_{seen}(q_i \mid D)}{P_{unseen}(q_i \mid D)} + \sum_i \log P_{unseen}(q_i \mid D)$$

query length

$$= \sum_{i:c(q_i \mid D)>0} \log \frac{P_{seen}(q_i \mid D)}{\alpha_d P(q_i \mid \mathbb{C})} + n \log \alpha_d + \sum_i \log P(q_i \mid \mathbb{C})$$

doc. independent

term weight

prob. mass to unseen words

Language modeling

Smoothing

$$\log P(Q \mid D) \propto \sum_{i:c(q_i|D)>0} \log \frac{P_{seen}(q_i \mid D)}{\alpha_d P(q_i \mid \mathbb{C})} + n \log \alpha_d$$

- What happens to $P(q_i \mid \mathbb{C})$ if the query term is a stopword?
  - Similar to IDF
- $n \log \alpha_d$: product of a document dependent constant and the query lengh
  - Longer documents should receive less smoothing (greater penalty if a query term is missing)
  - Similar to document length normalization

Language modeling

Smoothing

$$P_{ml}(w \mid D) = \frac{c(w;D)}{\sum_{w} c(w;D)}$$

- How to smooth $P(w|D)$ in practice?
  - We look at 3 ideas (there are more)

- General idea: discount probabilities of seen words, assign extra probability mass to unseen words with a fallback model (the collection language model)

$$P(w \mid D) = \begin{cases} P_{smoothed}(w \mid D) & if\ word\ w\ is\ seen \\ \alpha_d P(w \mid \mathbb{C}) & otherwise \end{cases}$$

all probabilities must sum to one.

- Laplace smoothing: simplest approach
  - Add one count to every word

$$P_{laplace}(w \mid D) = \frac{c(w;D)+1}{\sum_{w} (\ c(w;D)+1 \ )}$$

Language modeling

Smoothing

- Jelineck-Mercer (JM) smoothing
  - Linear interpolation between ML and collection LM

$$P_\lambda(w \mid D) = (1-\lambda)P_{ml}(w \mid D) + \lambda P(w \mid \mathbb{C}), \quad \lambda \in (0,1)$$

parameter controls amount of smoothing

- Term-dependent Jelineck-Mercer smoothing

$$P_{\lambda_w}(w \mid D) = (1-\lambda_w)P_{ml}(w \mid D) + \lambda_w P(w \mid \mathbb{C})$$

Different terms are smoothed to different degrees

Language modeling

Smoothing

What is the effect of the denominator when |D|=100 and |D|=1000?

- Dirichlet smoothing *

$$P_\mu(w \mid D) = \frac{c(w;D) + \mu P(w \mid \mathbb{C})}{\sum_w c(w;D) + \mu}, \; usually \; \mu > 100$$

➔The longer the document, the less smoothing is applied

- Absolute discounting

subtract constant from counts

$$P_\delta(w \mid D) = \frac{\max(c(w;D) - \delta, 0)}{\sum_w c(w;D)} + \sigma P(w \mid \mathbb{C}), \delta \in (0,1) \; and \; \sigma = \delta \frac{|D_{unique}|}{|D_{all}|}$$

Assignment II

- Retrieval experiments
- You will get a number of queries to "run" on the Wikipedia corpus (training data)
- You have to implement the vector space model + 1 modification (term normalization, language modeling, your own ideas)
  - I make sure the output format and everything else is clear
- We will run a benchmark between the groups

Sources

① Introduction to Information Retrieval. Manning et al. 2008
② Information retrieval. Keith van Rijsbergen, 1979.
③ Managing gigabytes, Witten et al. 1999.
④ The probability ranking principle in IR, S.E. Robertson, 1977
⑤ A study of smoothing methods for language models applied to ad hoc information retrieval. Zhai & Lafferty. 2001.
⑥ The importance of prior probabilities for entry page search. Kraaij et al. 2002.
⑦ The effect of adding relevance information in a relevance feedback environment. Buckley et al. 1994
⑧ Automatic acquisition of hyponyms from large text corpora. MA Hearst. 1992.