Relevance Feedback and Query Expansion

7 Jan, 2025

Outline

Relevance Feedback and Query Expansion [5 Hours]

7.1. Relevance feedback mechanisms

7.2. Pseudo-relevance feedback

7.3. Query expansion techniques

Recap

Probabilistic retrieval

Q: Why probability theory in IR, and why probabilistic ranking?

Q: What are the uncertainties of the IR process that we model probabilistically?

Language modeling

Q: What is a language model?

Q: How do we estimate probabilities of sequences of words?

Q: What language models are used in IR? Why? Explain the sparseness issue of LMs.

Language modeling for IR

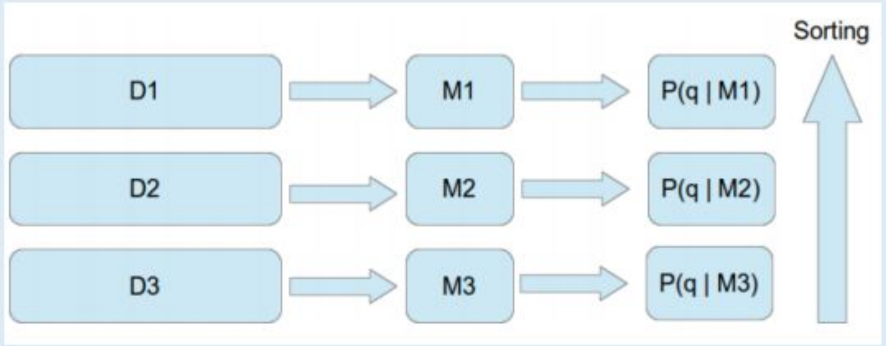
Q: What probability does the query likelihood model estimate?

Q: How do we estimate the probability of the query given the document?

Q: What happens if some query term is not present in the document?

Q: Explain the differences between smoothing techniques.

- Given a document collection D and a query q we need to estimate the probability P(q | d) for every document d in D
- In the **query likelihood model**, we estimate the probability P(q | d) as the probability that the language model built from d generates the query q
- Algorithm
  1. Compute the language model M_i for every document d_i in D
  2. Compute the probability P(q | Mi) for every language model M_i



- **Intuition**: Language models of relevant documents should assign higher probability for the query

- We are given a toy collection consisting of three documents
  - d_1: „Sam chased the orc with the sword”
  - d_2: „Frodo and Sam stabbed orcs”
  - d_3: „Sam took the sword”
- We are given the query „Sam and orc and sword”
- Let’s rank the documents according to unigram LM for IR (ignore stopwords)

- **Step 1:** Compute language models of individual documents
  - M_1: P(„sam”) = 0.25, P(„chase”) = 0.25, P(„orc”) = 0.25, P(„sword”) = 0.25
  - M_2: P(„frodo”) = 0.25, P(„sam”) = 0.25, P(„stab”) = 0.25, P(„orc”) = 0.25
  - M_3: P(„sam”) = 0.33, P(„took”) = 0.33, P(„sword”) = 0.33

- Laplace smoothing assumes that all unseen words are equally likely
- **Jelinek-Mercer smoothing** (also known as **interpolated smoothing**)
  1. Additionally builds a language model M_D from the whole document collection D
  2. Interpolates between probabilities of the query according to the
     - **Local LM** – language model M_d built from the particular document d
     - **Global LM** – language model M_D built from the whole collection

$$P(t_i|M_d) = \lambda \cdot P(t_i|M_d) + (1-\lambda) \cdot P(t_i|M_D)$$

- The probability of a word unseen in the document d still gets some probability from the global language model
  - Probability of an unseen word depends on its frequency in whole collection
- **Q:** What is some query term doesn’t appear in the whole collection D?

Improving recall of IR systems

- Most ranked retrieval systems optimize precision
  - Implicit assumptions:
    1. Plenty of relevant documents
    2. Users only look at top-ranked results
  - Top-ranked results should be relevant
  - Not so bad if we miss some of the relevant documents

- However, sometimes **recall** matters more than precision
  - Security & intelligence
  - Patent & publication search

…

- Sometimes recall matters more than precision
  - Retrieving **all** relevant documents is essential
  - Even at cost of generating many high-ranked irrelevant documents

- **Q:** How to increase recall?
- **A:** By modifying the initial query
  1. By adding new terms
     - E.g., terms that are semantically related to the terms of the original query
  2. By making the query vector more similar to vectors of relevant documents
     - We need some indication of relevance
     - Either provided by the user or assumed by the model

- Methods for improving recall of IR systems

1. **Global method:** query expansion
   - Adding new terms to the initial query
   - New terms somehow related to original terms
   - New terms identified using thesauri, distributional semantics, or lexical association

2. **Local methods**
   - Relevance feedback – requires additional input from the user
   - Pseudo relevance feedback (automated assumption of relevance)
   - Relevance model (automated assumption of relevance)

Relevance Feedback

- **Relevance feedback** means improving the query based on user feedback on relevance of the documents in the initial set of results
- Workflow:
  1. User issues a (usually short and simple) initial query
  2. Search engine retrieves and ranks the results
  3. User explicitly marks some of the results as relevant and/or non-relevant
  4. The search engine computes the new query (i.e., a better representation of the information need) based on the feedback
- There may be more than one iteration of the steps listed above

- **Assumption** motivating relevance feedback: it is often difficult to formulate a good (discriminative) query when you don’t know the collection well

…

Relevance feedback can go through one or more iterations of this sort.

The process exploits the idea that it may be difficult to formulate a good query when you don't know the collection well,

- but it is easy to judge particular documents, and so it makes sense to engage in iterative query refinement of this sort.

In such a scenario, relevance feedback can also be effective in tracking a user's evolving information need:

- seeing some documents may lead users to refine their understanding of the information they are seeking.

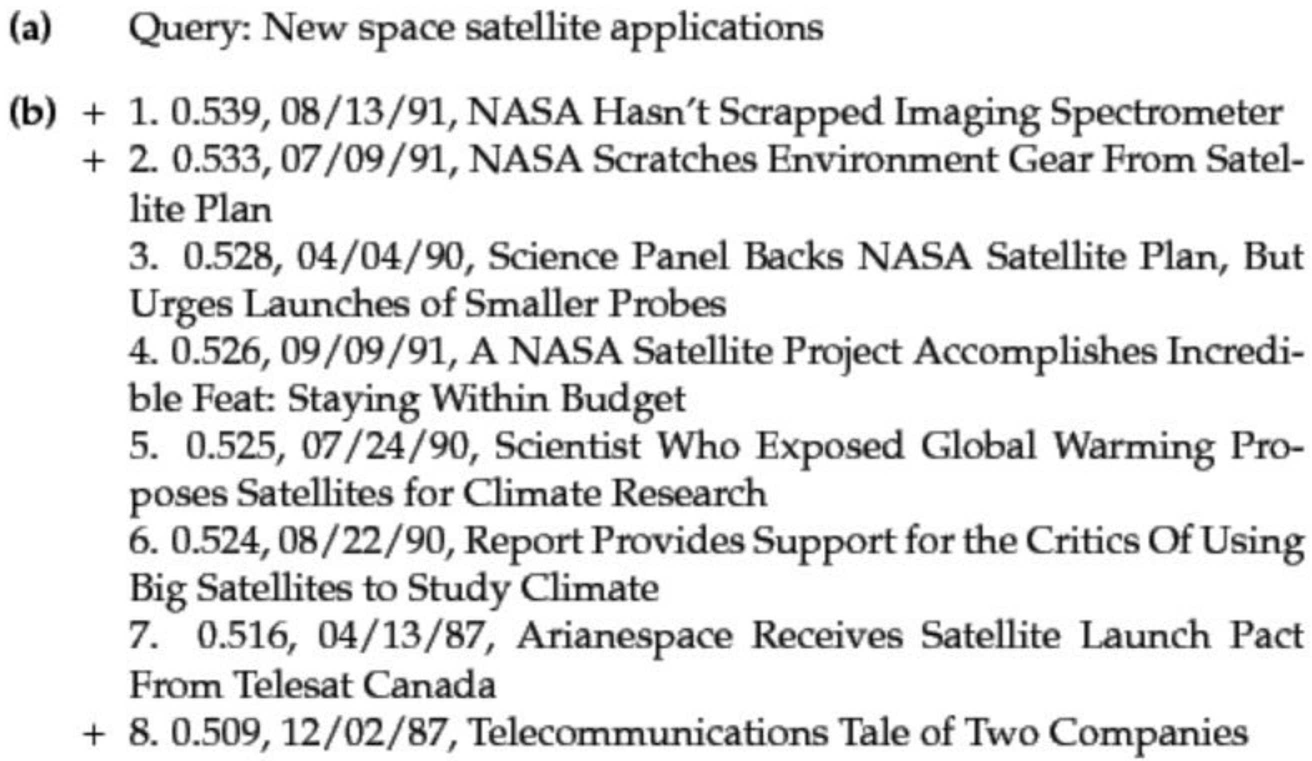
…

- Some definitions
  - **Ad hoc retrieval** – regular retrieval without relevance feedback
  - We'll call the first (uninformed) query being executed an **initial query** (q_0)
  - **Feedback set** – the first set of documents that are retrieved

...

- **Algorithm** (direct addition of query terms)
  1. Retrieve an initial ranked list of hits for the user's initial query q_0
  2. Ask user: which documents in the feedback set are relevant?
  3. Identify the best terms with which to extend the query
     - Good terms occur frequently in relevant documents
     - Good terms occur infrequently in non-relevant documents
     - Terms can have different importance weights
     - Form the vector q_{RF} containing the weighted new terms
  4. Expand the query with new terms
     - New terms could be additionally weighted with respect to original terms
     - $q' = f(q_0, q_{RF})$
  5. Retrieve the ranking for the expanded query q'

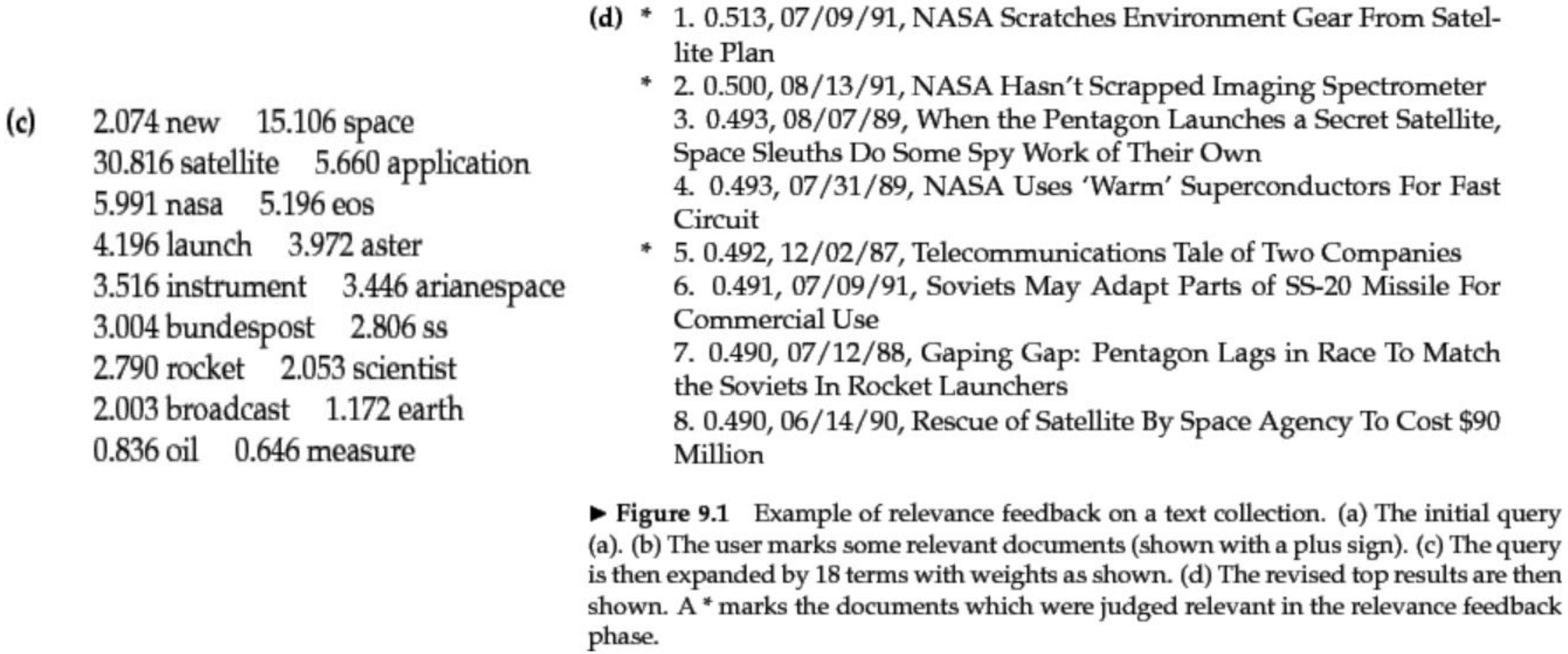
Initial query and results



(a) Query: New space satellite applications

(b)
- + 1. 0.539, 08/13/91, NASA Hasn't Scrapped Imaging Spectrometer
- + 2. 0.533, 07/09/91, NASA Scratches Environment Gear From Satellite Plan
- 3. 0.528, 04/04/90, Science Panel Backs NASA Satellite Plan, But Urges Launches of Smaller Probes
- 4. 0.526, 09/09/91, A NASA Satellite Project Accomplishes Incredible Feat: Staying Within Budget
- 5. 0.525, 07/24/90, Scientist Who Exposed Global Warming Proposes Satellites for Climate Research
- 6. 0.524, 08/22/90, Report Provides Support for the Critics Of Using Big Satellites to Study Climate
- 7. 0.516, 04/13/87, Arianespace Receives Satellite Launch Pact From Telesat Canada
- + 8. 0.509, 12/02/87, Telecommunications Tale of Two Companies

…



(c)

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

(d)
* 1. 0.513, 07/09/91, NASA Scratches Environment Gear From Satellite Plan
* 2. 0.500, 08/13/91, NASA Hasn't Scrapped Imaging Spectrometer
3. 0.493, 08/07/89, When the Pentagon Launches a Secret Satellite, Space Sleuths Do Some Spy Work of Their Own
4. 0.493, 07/31/89, NASA Uses 'Warm' Superconductors For Fast Circuit
* 5. 0.492, 12/02/87, Telecommunications Tale of Two Companies
6. 0.491, 07/09/91, Soviets May Adapt Parts of SS-20 Missile For Commercial Use
7. 0.490, 07/12/88, Gaping Gap: Pentagon Lags in Race To Match the Soviets In Rocket Launchers
8. 0.490, 06/14/90, Rescue of Satellite By Space Agency To Cost $90 Million

► **Figure 9.1** Example of relevance feedback on a text collection. (a) The initial query (a). (b) The user marks some relevant documents (shown with a plus sign). (c) The query is then expanded by 18 terms with weights as shown. (d) The revised top results are then shown. A * marks the documents which were judged relevant in the relevance feedback phase.

Relevance feedback in probabilistic retrieval

- **Recap**: In probabilistic retrieval, relevance feedback can be incorporated into estimations of probabilities $P(D_t \mid Q, r)$ and $P(D_t \mid Q, \neg r)$

- Estimates without relevance feedback
  - $P(D_t \mid Q, r) = 0.5$
  - $P(D_t \mid Q, \neg r) = N_t / N$

- Estimates **with** relevance feedback
  - $P(D_t \mid Q, r) = (r_t + 0.5) / (R + 1)$
  - $P(D_t \mid Q, \neg r) = (N_t - r_t + 0.5) / (N - R + 1)$

Relevance feedback in VSM

- **Rocchio algorithm** –used to incorporate relevance feedback into the vector space model
- This algorithm is looking to rewrite the query so that it’s vector is:
  - As similar as possible to the vectors of relevant documents
  - As dissimilar as possible to the vectors of non-relevant documents
- Open questions
  - How do we aggregate vectors of all (non-)relevant docs in one vector?
  - Do we take the original query into account?

Rocchio algorithm

- How do we aggregate vectors of all (non-)relevant docs in one vector?
  - Key concept: **centroid**
- Centroid is the center of mass of a set of points
  - In VSM, we represent documents as points in a high-dimensional space
  - So, given a set of documents (their VSM vectors), we can compute their centroid
- Centroid μ of the set of document vectors C is computed simply as the mean of vectors of individual documents

$$\frac{1}{|C|}\sum_{d \in C} d$$

…

- **Q:** Do we take the original query q_0 into account?
- **A:** Theoretically, we don't have to
  - Because Rocchio just needs relevant and non-relevant documents to compute the „optimal" query:

$$q_{opt} = argmax_q \left(\cos\left(q, \frac{1}{|C_r|} \sum_{d_r \in C_r} d_r \right) - \cos\left(q, \frac{1}{|C_{nr}|} \sum_{d_{nr} \in C_{nr}} d_{nr} \right) \right)$$

- **Q:** What might be the problem with computing the optimal query only from relevance feedback?
- **A:** Users provide relevance judgements only for a small number of documents, (not all documents in the collection!)

...

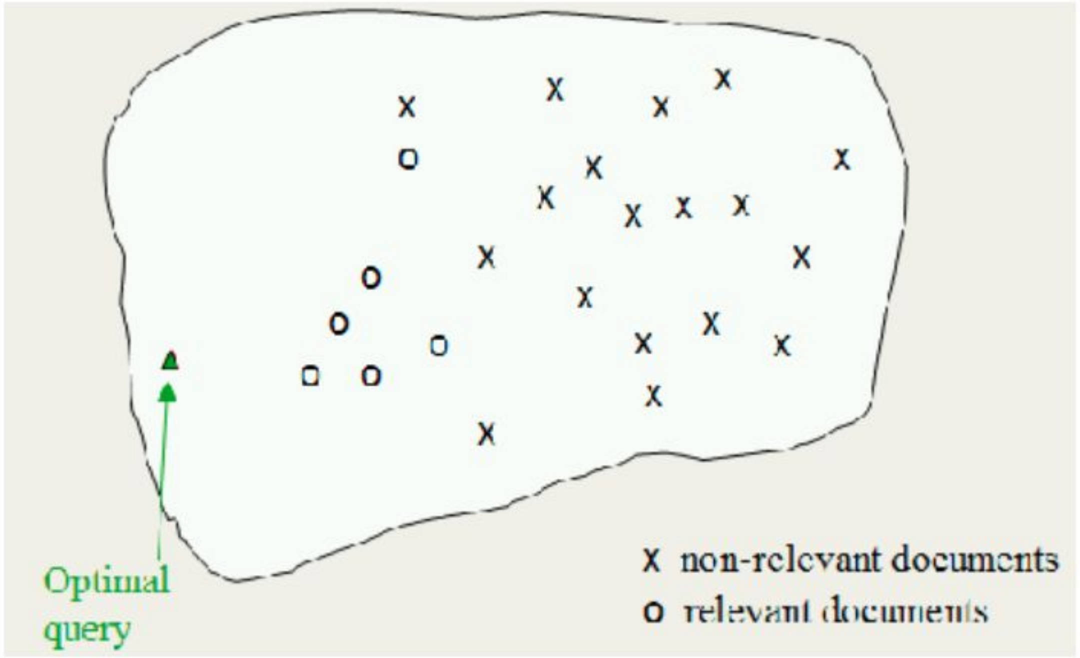


Figure 9.3: The Rocchio optimal query for separating relevant and nonrelevant documents.

…

- We are given only a handful of relevance feedback annotations
- Thus, we re-estimate the query by combining
  1. Centroid of relevant documents
  2. Centroid of non-relevant documents
  3. Initial query vector q_0

$$q_m = \alpha \cdot q_0 + \left(\beta \cdot \frac{1}{|D_r|} \sum_{d_j \in D_r} d_j \right) - \left(\gamma \cdot \frac{1}{|D_{nr}|} \sum_{d_j \in D_{nr}} d_j \right)$$

- D_r is the set of vectors of known relevant documents (different from C_r)
- D_{nr} is the set of vectors of known non-relevant documents (different from C_{nr})
- α, β, and γ are weights, determining the contribution of each component (set beforehand or empirically)
- New query moves towards the relevant and away from non-relevant documents

Rocchio relevance feedback - Example

- Given:
    - Initial query = “cheap CDs cheap DVDs extremely cheap CDs”.
    - d_{1} = “CDs cheap software cheap CDs” is judged as relevant.
    - d_{2} = “cheap thrills DVDs” is judged as nonrelevant
- What would the revised query vector be after relevance feedback?

Let us solve this together

Assume that we are using direct term frequency (with no scaling and no document frequency). There is no need to length-normalize vectors. Assume $\alpha = 1$, $\beta = 0.75$, $\gamma = 0.25$.

Rocchio relevance feedback - Example

Quiz: Can you complete the following table?

q_0 = "cheap CDs cheap DVDs extremely cheap CDs".

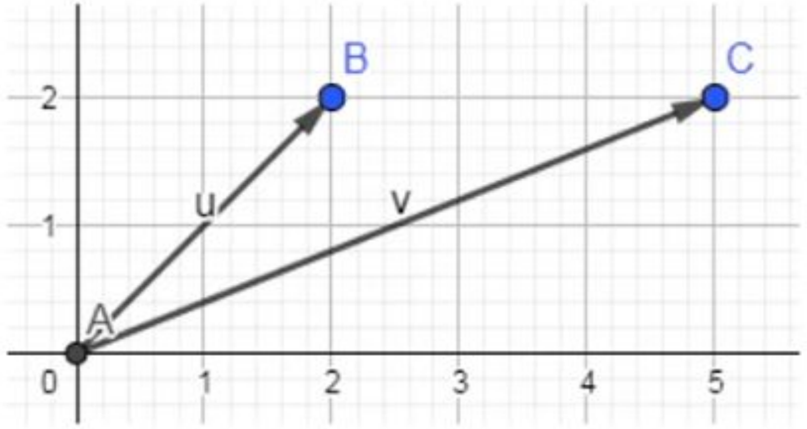
d_1 = "CDs cheap software cheap CDs".

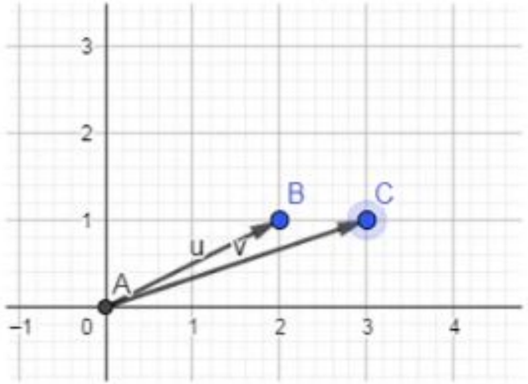
d_2 = "cheap thrills DVDs".

	cheap	CDs	DVDs	extremely	software	thrills
q_0	3	2	1	1	0	0
d_1	2	2	0	0	1	0
d_2	1	0	1	0	0	1

Moving Vectors

- Move (2,2) to (5,2) by adding 3 to x.





Rocchio relevance feedback - Example

Quiz: How to calculate the modified query vector, q_m?

d_1 is judged as relevant. d_2 is judged as nonrelevant.

Assume $\alpha = 1$, $\beta = 0.75$, $\gamma = 0.25$.

	cheap	CDs	DVDs	extremely	software	thrills
q_0	3	2	1	1	0	0
d_1	2	2	0	0	1	0
d_2	1	0	1	0	0	1

$q_m = q_0 + 0.75*d_1 - 0.25*d_2$

q_m	4.25	3.5	0.75	1	0.75	0

Negative weight does not make sense. So, leave them as zero.

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

…

- Tradeoff between α and β/γ depends on the number of judged documents
  - A few documents judged – $\alpha > \beta$ and γ
    - More emphasis on the original query
  - A lot of documents judged – $\alpha < \beta$ and γ
    - More emphasis on relevance feedback

- In the modified query, weights of some terms may become negative
  - Terms that appear more frequently in non-relevant queries
  - We just ignore negative weights (i.e., we reset them to 0)

…

- Positive vs. Negative feedback
  - **Q:** Should we emphasize more relevant or non-relevant documents?
    - I.e., should β be larger than γ or vice-versa?
  - **A:** positive feedback is more valuable than negative feedback
    - So, we should set $\gamma < \beta$, e.g., $\beta = 0.75$ and $\gamma = 0.25$

- Many systems allow only positive feedback (i.e., $\gamma = 0$)
  - **Q:** Why?

Relevance feedback – issues

- **Q:** When does relevance feedback work?
- **A:** Relevance feedback works when
  1. User has sufficient knowledge for initial query
  2. Relevance for initial query „behaves nicely”
     - **Case #1:** All relevant documents are tightly clustered around a single relevance vector
     - **Case #2:** There are several different clusters of relevant documents, but they have significant vocabulary overlap
     - Similarities (term overlap) between relevant and non-relevant documents are small

Relevance feedback – issues

- Cases when relevance feedback won’t work:

1. User’s knowledge insufficient for initial query
   - Misspellings (e.g., „Brittany Speers”), vocabulary mismatches („cosmonaut” vs. „astronaut”)

2. Multiple relevance vectors (groups) with little lexical overlap
   - When query contains general concepts with multiple senses
   - E.g., „contradictory goverment policies”
     „pop stars that worked at Burger King”

Relevance feedback – issues

- **Efficiency**
  - Extending the queries with additional terms (directly on indirectly, e.g., via Rocchio) may lead to long queries
  - Long queries are inefficient to process
  - Solution: limit the number of terms to be added
    - Criteria – e.g., collection-wide frequency

- **User issues**
  - Users often reluctant to provide explicit feedback
    - Time consuming, sometimes also quite cognitively demanding
  - Harder for users to understand why a particular document was retrieved after applying relevance feedback

Pseudo-relevance feedback also known as blind relevance feedback

Pseudo relevance feedback , also known as *blind relevance feedback* , provides a method for automatic local analysis.

It automates the manual part of relevance feedback, so that the user gets improved retrieval performance without an extended interaction.

The method is to do normal retrieval to find an initial set of most relevant documents,

- to then *assume* that the top k ranked documents are relevant,
- and finally to do relevance feedback as before under this assumption.

Pseudo-relevance feedback also known as blind relevance feedback

- **Pseudo-relevance feedback** is relevance feedback without the human in the loop
  - I.e., we automate the „manual" part of relevance feedback
  - Eliminate the need for human relevance judgements
- Algorithm
  1. Retrieve an initial ranked list of hits for the user's initial query q_0
  2. Assume that the top K documents are relevant
  3. Compute expansions terms, q
  4. Expand the query (the vector q_{RF} containing the weighted new terms)
  5. Retrieve the ranking for the expanded query q'

$$q' = \lambda \cdot q_0 + (1 - \lambda) \cdot q_{RF}$$

- **Q:** How do we compute the terms with which to expand the query?

Pseudo-relevance feedback

- Often works very well, but may go wrong
  - Crucially depends on how good the initially top-ranked K documents are
  - On average works better than query expansion
    - As evaluated on standard benchmark datasets

- Pitfall of pseudo-relevance feedback: **query drift**
  - E.g., imagine that for initial query „copper mines”, top K documents retrieved mostly describe copper mines in Chile
  - The final result may be more about Chile than about copper mines

- Query drift is the reason why typically only one iteration of pseudo-relevance feedback is performed

Relevance model (Lavrenko, 2001)

- Input
  - Initial query q_0
  - Top K documents in the ranking for initial query – $d_1, d_2, ..., d_K$
  - Relevance probabilities of top ranked documents for the initial query – $P(d_i|q_0)$
- Output
  - A distribution of terms denoting how well they describe the initial query q_0
  - An importance/probability of term w for q_0 query is computed as follows:

$$P(w|q_0) = \sum_{i=1}^{K} P(w|d_i) \cdot P(d_i|q_0)$$

  - Rank the terms in decreasing order of $P(w|q_0)$, take top N terms and combine them into a weighted expansion query q_{PRF}

Relevance Model vs. Rocchio algorithm

- Let's compare Lavrenko's relevance model with Rocchio algorithm
  - Assume Rocchio considers top K initially ranked documents as relevant (D_r) and does not consider non-relevant documents ($\gamma = 0$)
- Lavrenko's relevance model

$$q' = \lambda \cdot q_0 + (1 - \lambda) \cdot q_{RF} \qquad P(w|q_0) = \sum_{i=1}^{K} P(w|d_i) \cdot P(d_i|q_0)$$

- Rocchio algorithm

$$q_m = \alpha \cdot q_0 + \beta \cdot \frac{1}{|D_r|} \sum_{d_j \in D_r} d_j$$

Rocchio uses all terms, **RM uses only top N terms**

Rocchio computes simple average, **RM weighted average with document relevances for query $P(d_i|q_0)$ as weight**

Rocchio uses TF-IDF weights, **RM uses $P(w|d_i)$**

Query Expansion

- **Query expansion** techniques try to improve retrieval results by adding terms to the user query
- Relevance model is a form of automated query expansion

- Next, we will consider other ways of expanding queries
  - Independent of the initial ranking of documents
  - I.e., when the content of top-ranked documents is not used for query expansion
  - We use **external resources** to expand the query
    1. Dictionaries / thesauri
    2. Large corpora

…

- There are two types of query expansions:

1. **Explicit** query expansion
   - Initial query is executed and results retrieved and ranked
   - Expanded version(s) of the initial query proposed to the user for subsequent searches

2. **Implicit** query expansion
   - The original query is implicitly expanded and results are obtained by executing the expanded query

- **Q:** How do we identify good expansion words?

…

- The following are common methods of query expansion:
  1. **Controlled vocabulary**
     - There is a canonical term for each concept, all other terms for the same concept are replaced by the canonical term
     - E.g., {„hotel", „apartment", „room"} → „accommodation"; „Burma" → „Myanmar"

  2. **Manual thesaurus**
     - Human annotators build sets of synonymous terms for concepts, without designating a canonical terms
     - E.g., {„hotel", „apartment", „room", „accommodation"}; {„Burma", „Myanmar"}
     - **WordNet (https://wordnet.princeton.edu/)**
       - General large lexical database for English language, contains 117K synonym sets (*synsets*)
     - Domain specific Thesauri – e.g., NIH for medicine (https://ncit.nci.nih.gov/ncitbrowser)
     - Knowledge bases, e.g., DBPedia or Yago can also be used for query expansion

...

- The following are common methods of query expansion:

3. **Automatically generated thesaurus**
   - Co-occurrence statistics (lexical association metrics) on a large external corpus (e.g., Wikipedia) are used to automatically induce a large thesaurus
   - E.g., „hotel” often co-appears in text with „accommodation”

4. **Query log mining**
   - Query reformulations done by users are logged
   - These reformulations can be used for query expansion for similar queries of new users
   - Requires huge amounts of logged queries – feasible only in web search

Thesaurus-based query expansion

- **Algorithm** is simple
  - For each term *t* from the initial query q_0, look up synonyms and related terms in the thesaurus
    - E.g., „frodo” -> „frodo baggins”, „hobbit frodo”
  - Optionally, assign new terms lower weights than the original terms

- Thesaurus-based query expansion
  - **Generally increases recall**
    - Especially in specific domains, with rich domain-specific thesauri
  - May **significantly decrease precision**
    - Particularly when expanding an ambiguous query term
      - E.g., „interest rate” -> „interest, fascinate, rate, evaluate”

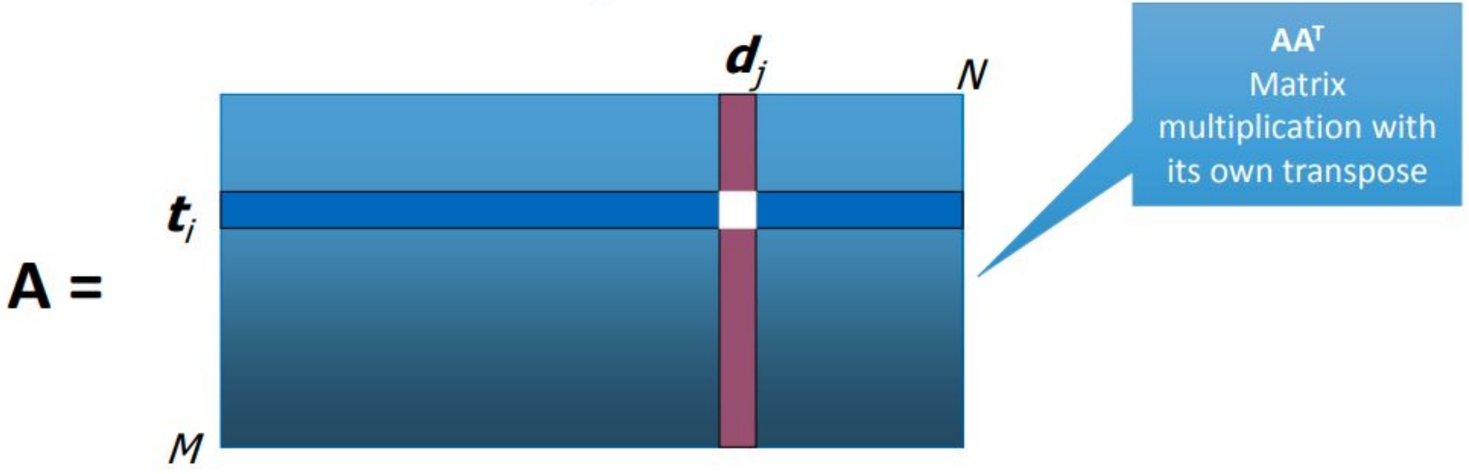
...

- Manually producing a thesaurus is time-consuming and expensive
  - Additionally, it needs to be constantly updated to reflect changes in the domain

- **Automated thesaurus generation**
  - Generating thesaurus by detecting similarity/relatedness of terms in a large corpora
  - **Distributional hypothesis** – words are similar if they occur in similar contexts
    - E.g., „apple" is similar to „pear" as you can both *harvest*, *peel*, *prepare* and *eat* both
  - **Related words** – words that often co-appear are semantically related
    - E.g., „pilot" and „airplane"

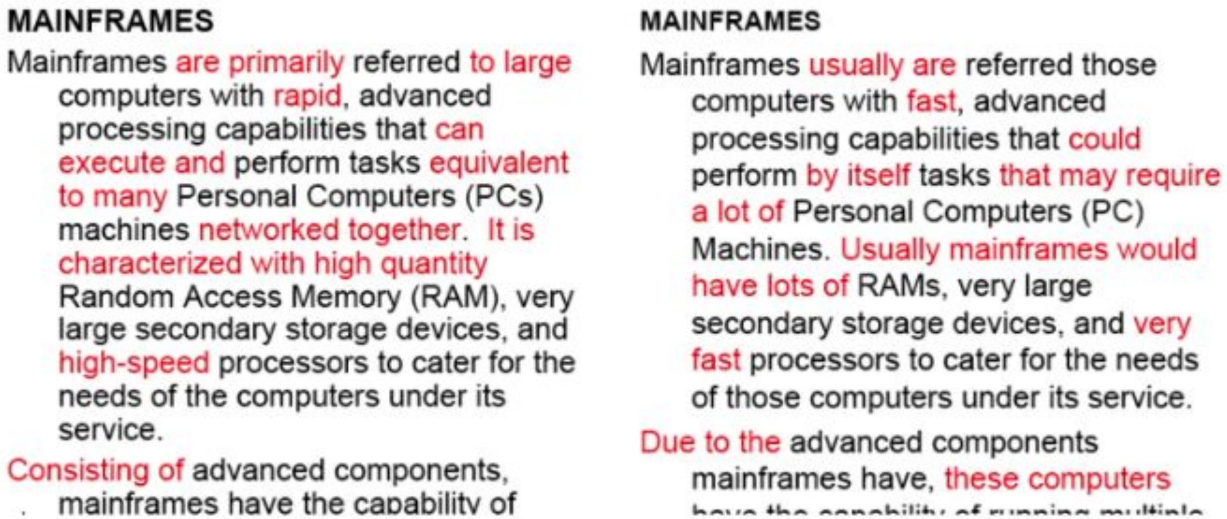
Co-occurrence thesaurus

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



- For each t_i, pick terms with high values in C (term-term matrix)

Co-occurrence Analysis



MAINFRAMES

Mainframes are primarily referred to large computers with rapid, advanced processing capabilities that can execute and perform tasks equivalent to many Personal Computers (PCs) machines networked together. It is characterized with high quantity Random Access Memory (RAM), very large secondary storage devices, and high-speed processors to cater for the needs of the computers under its service.

Consisting of advanced components, mainframes have the capability of

MAINFRAMES

Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater for the needs of those computers under its service.

Due to the advanced components mainframes have, these computers have the capability of running multiple

Source: https://web.stanford.edu/~jurafsky

Co-occurrence Analysis

- Term-Document Matrix
  - How often does individual terms appear in a document?
- Term-Term Matrix
  - How often terms co-occur?

Quiz: Which books are similar?

	Book1	Book2	Book3	Book4
cricket	400	10	355	3
football	5	5	4	4
hockey	9	330	10	200
tennis	2	6	12	4

Co-occurrence Analysis

- Two documents are similar if the document vectors are similar.

	Book1	Book2	Book3	Book4
cricket	400	10	355	3
football	5	5	4	4
hockey	9	330	10	200
tennis	2	6	12	4

Book1 and Book3 seem to be on cricket.
Book2 and Book4 are about hockey.

A =

	D1	D2	D3	D4	D5	d6
T1	1	0	1	0	1	0
T2	1	1	0	1	0	0
T3	0	1	1	0	1	0
T4	0	1	0	0	1	0
T5	1	0	0	1	1	1
T6	1	0	1	0	1	0

Terms 1 & 6 appear in the same documents

A^T =

	T1	T2	T3	T4	T5	T6
D1	1	1	0	0	1	1
D2	0	1	1	1	0	0
D3	1	0	1	0	0	1
D4	0	1	0	0	1	0
D5	1	0	1	1	1	1
D6	0	0	0	0	1	0

AA^T =

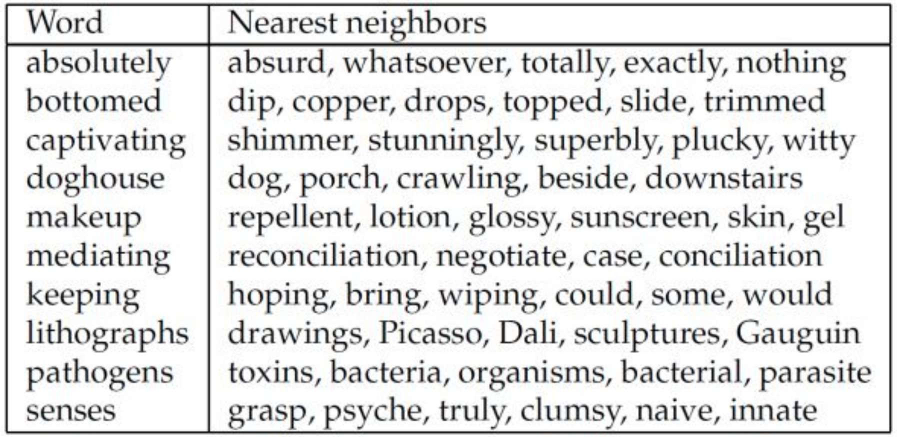
	T1	T2	T3	T4	T5	T6
T1	3	1	2	1	2	3
T2	1	3	1	1	2	1
T3	2	1	3	2	1	2
T4	1	1	2	2	1	1
T5	2	2	1	1	4	2
T6	3	1	2	1	2	3

Term 6 seems to be best related to itself and term 1.

Automated thesaurus generation

- Expansion based on distributional vectors has shortcomings
  - Distributional vectors cannot distinguish similarity from relatedness
    - E.g., two synonyms might be as similar as two antonyms
    - Making distributional vectors encode only similarity, not relatedness – active research area
    - Semantic similarity and relatedness are vague, not crisp relations
  - Word with multiple senses especially problematic

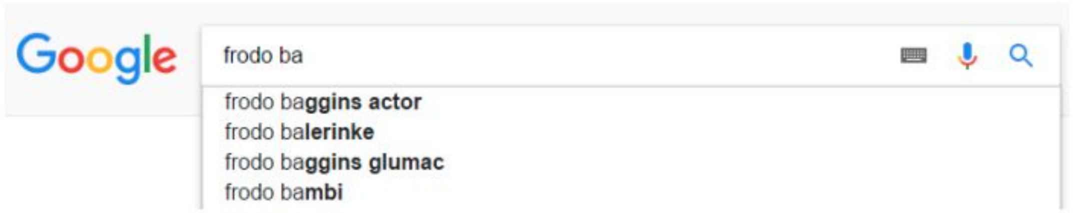
- Quality of produced term associations often not good enough for IR tasks



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

User interaction data („click” data)

- Using data collected from users’ interaction with the search engine to improve the search process
  - Can be used to assist the user in formulating the query
- Common example
  - Query auto-completion

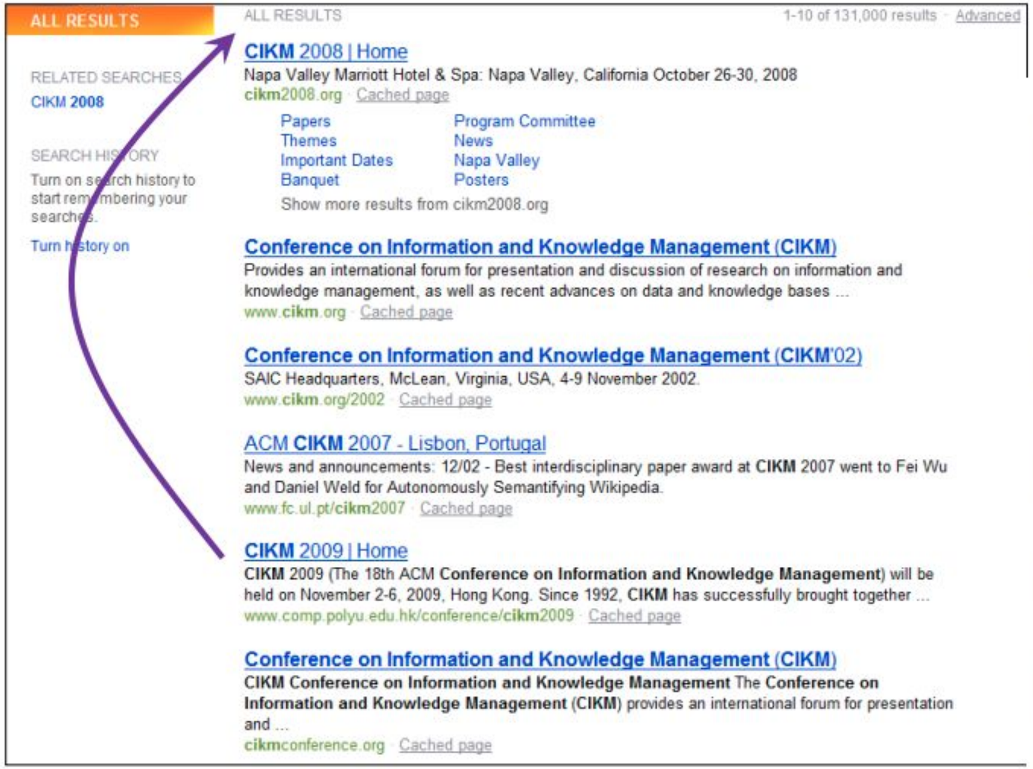


frodo ba

frodo ba**ggins actor**
frodo ba**lerinke**
frodo ba**ggins glumac**
frodo ba**mbi**

- Indirect relevance feedback
  - Assumption: ranked documents clicked by the user are relevant for the query
  - **Clickstream mining**:
    - Documents are considered more relevant, the more clicked they are
    - Global metric of document relevance on the web (e.g., like PageRank)
    - One feature in machine learning-based ranking functions

Click log data for re-ranking



ALL RESULTS

RELATED SEARCHES
CIKM 2008

SEARCH HISTORY
Turn on search history to start remembering your searches.
Turn history on

ALL RESULTS 1-10 of 131,000 results · Advanced

CIKM 2008 | Home
Napa Valley Marriott Hotel & Spa: Napa Valley, California October 26-30, 2008
cikm2008.org · Cached page
Papers | Program Committee
Themes | News
Important Dates | Napa Valley
Banquet | Posters
Show more results from cikm2008.org

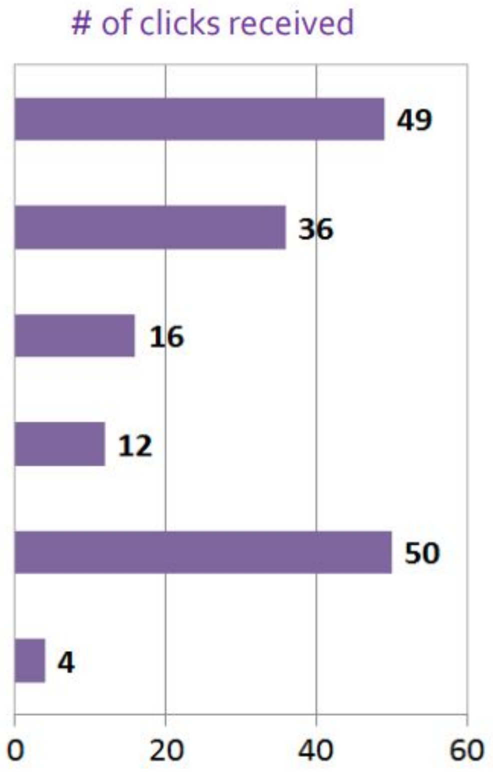
Conference on Information and Knowledge Management (CIKM)
Provides an international forum for presentation and discussion of research on information and knowledge management, as well as recent advances on data and knowledge bases ...
www.cikm.org · Cached page

Conference on Information and Knowledge Management (CIKM'02)
SAIC Headquarters, McLean, Virginia, USA, 4-9 November 2002.
www.cikm.org/2002 · Cached page

ACM CIKM 2007 - Lisbon, Portugal
News and announcements: 12/02 - Best interdisciplinary paper award at CIKM 2007 went to Fei Wu and Daniel Weld for Autonomously Semantifying Wikipedia.
www.fc.ul.pt/cikm2007 · Cached page

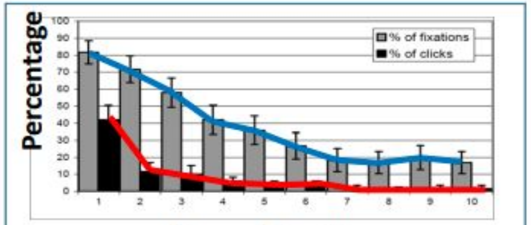
CIKM 2009 | Home
CIKM 2009 (The 18th ACM Conference on Information and Knowledge Management) will be held on November 2-6, 2009, Hong Kong. Since 1992, CIKM has successfully brought together ...
www.comp.polyu.edu.hk/conference/cikm2009 · Cached page

Conference on Information and Knowledge Management (CIKM)
CIKM Conference on Information and Knowledge Management The Conference on Information and Knowledge Management (CIKM) provides an international forum for presentation and ...
cikmconference.org · Cached page

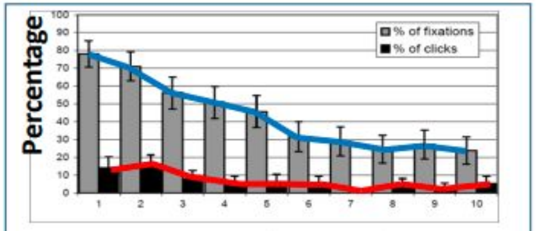


…

- How large is the click-log?
  - bing search logs grow 10+ TB every day
  - Efficient algorithms for mining click-logs are needed
- Click-position bias
  - Higher positions receive more user attention (eye fixation) and clicks than lower positions
  - True, even when the order or results is reversed!
  - Clicks are informative, but biased – top-ranked documents clicked even when non-relevant



Normal Position



Reversed Impression

Interaction data – conclusion

- User behavior is an intriguing source of relevance data
  - Users make (somewhat) informed choices when they interact with search engines
  - Potentially a lot of data available in search logs

- But there are significant caveats
  - User behavior data can be very noisy
  - Interpreting user behavior can be tricky
  - Spam can be a significant problem
  - Not all queries will have user behavior

Thank you for your time and attention.