TEXT RETRIEVAL AND SEARCH ENGINES

The basic concepts, principles, and the major techniques in text retrieval, which is the underlying science of search engines.

Course author:

ChengXiang Zhai

University of Illinois at Urbana-Champaign
&
Coursera

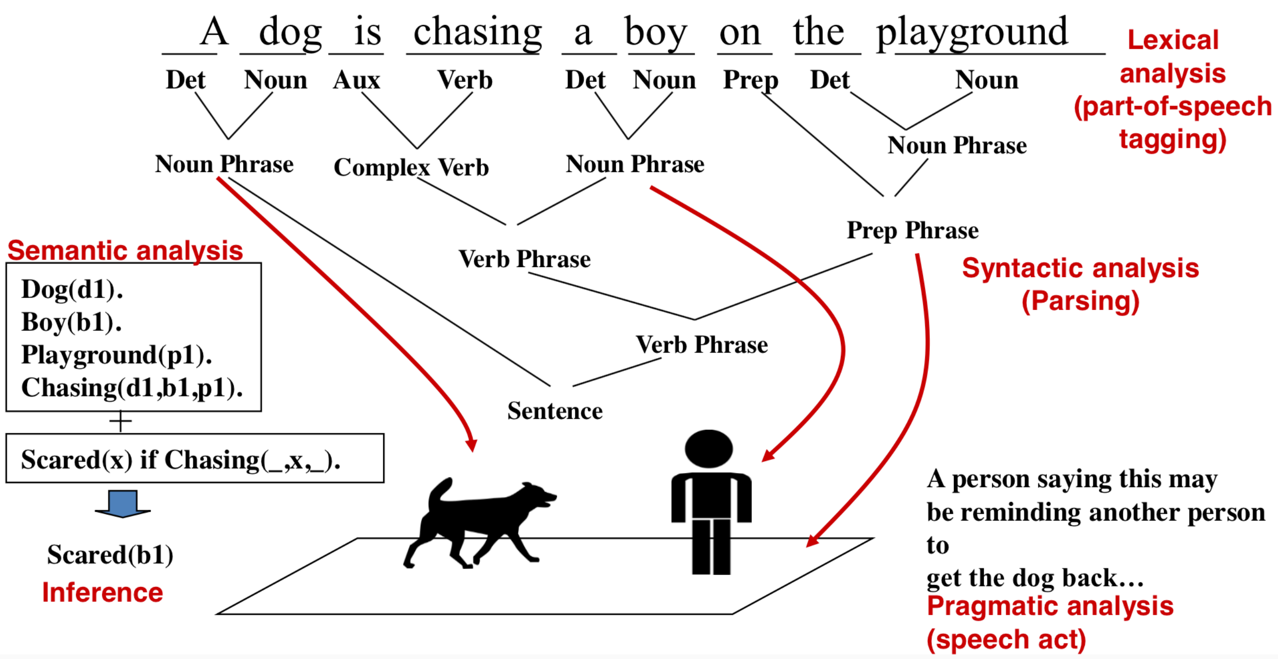
2015

Contents

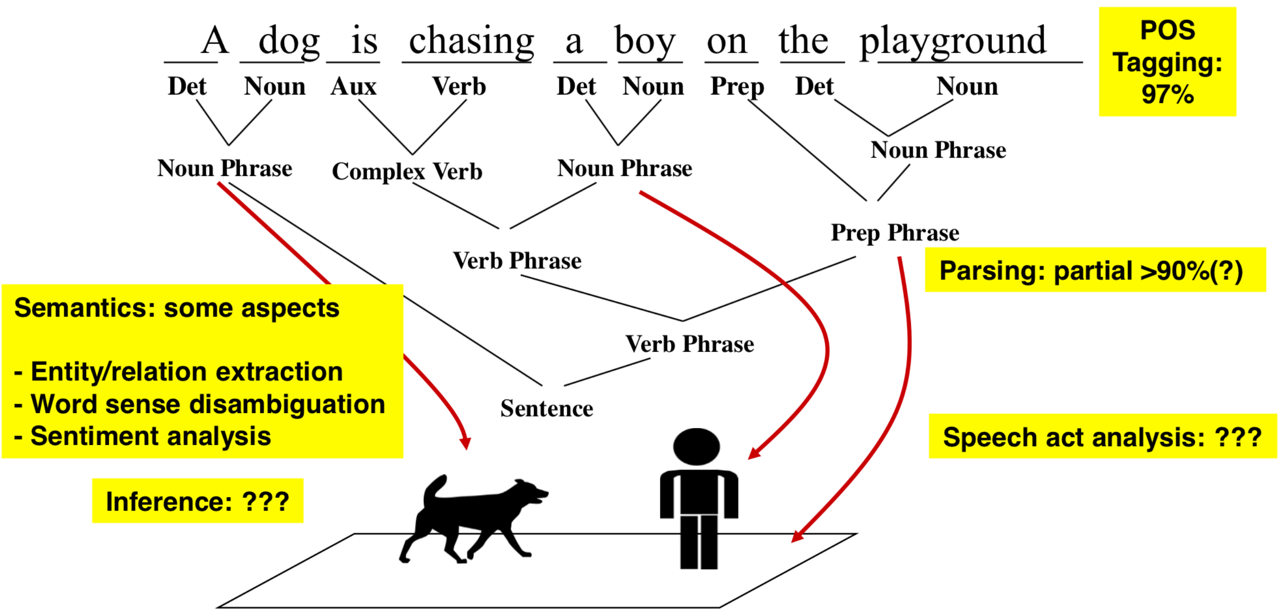
1 Natural Language Content Analysis

NLP = Natural Language Processing

1.1 An Example of NLP



1.2 The State of the Art



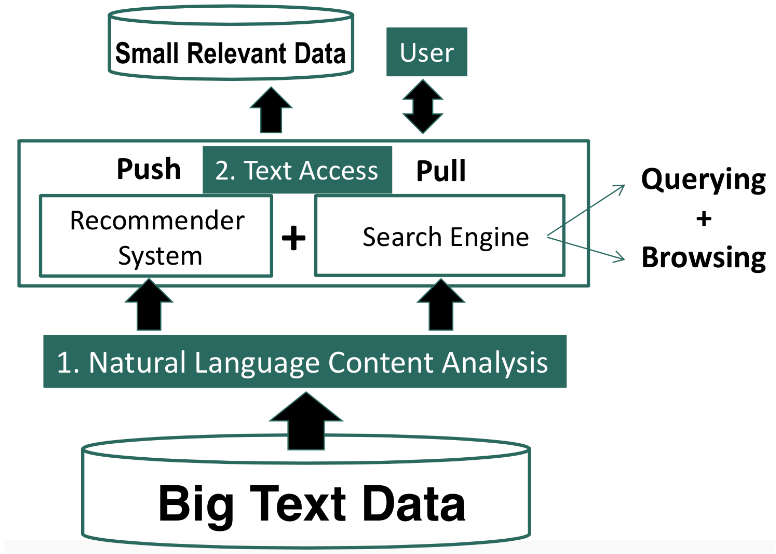
1.3 Recommended reading

- Chris Manning and Hinrich Schütze, «Foundations of Statistical Natural Language Processing», MIT Press. Cambridge, MA: May 1999.

2 Text Access

2.1 Two Modes of Text Access: Pull vs. Push

- Pull Mode (search engines) – Users take initiative
  - Ad hoc information need
- Push Mode (recommender systems)
  - Systems take initiative
  - Stable information need or system has good knowledge about a user's need



2.2 Pull Mode: Querying vs. Browsing

- Querying
  - User enters a (keyword) query
  - System returns relevant documents
  - Works well when the user knows what keywords to use
- Browsing
  - User navigates into relevant information by following a path enabled by the structures on the documents
  - Works well when the user wants to explore information, doesn't know what keywords to use, or can't conveniently enter a query

2.3 Recommended reading

- N. J. Belkin and W. B. Croft. 1992. «Information filtering and information retrieval: two sides of the same coin?» Commun. ACM 35, 12 (Dec. 1992), 29-38.

3 Text Retrieval Problem

3.1 What Is Text Retrieval?

TR = Text Retrieval[1]

- Collection of text documents exists
- User gives a query to express the information need
- Search engine system returns relevant documents to users
- Often called "information retrieval" (IR), but IR is actually much broader
- Known as «search technology» in industry

TR is an empirically defined problem:

- Can't mathematically prove one method is better than another
- Must rely on empirical evaluation involving users!

3.2 Formal Formulation of TR

- **Vocabulary**: $V = \{w_1, w_2, \dots, w_N\}$ of language
- **Query**: $q = q_1, \dots, q_m$, where $q_i \in V$
- **Document**: $d_i = d_{i1}, \dots, d_{im_i}$, where $d_{ij} \in V$
- **Collection**: $C = \{d_1, \dots, d_M\}$
- **Set of relevant documents**: $R(q) \subseteq C$
  - Generally unknown and user-dependent
  - Query is a «hint» on which doc is in $R(q)$
- **Task**: compute $R'(q)$, an approximation of $R(q)$

3.3 How to Compute $R'(q)$

- Strategy 1: Document selection
  - $R'(q) = \{d \in C \,|\, f(d,q) = 1\}$, where $f(d,q) \in \{0,1\}$ is an indicator function or binary classifier
  - System must decide if a doc is relevant or not (absolute relevance)
- Strategy 2 (generally preferred): Document ranking
  - $R'(q) = \{d \in C \,|\, f(d,q) > \theta\}$, where $f(d,q) \in \mathfrak{R}$ is a relevance measure function; θ is a cutoff determined by the user
  - System only needs to decide if one doc is more likely relevant than another (relative relevance)

[1] Retrieval - поиск

3.4 Theoretical Justification for Ranking

Probability Ranking Principle [Robertson 77]: Returning a ranked list of documents in descending order of probability that a document is relevant to the query is the optimal strategy under the following two assumptions:

- The utility of a document (to a user) is independent of the utility of any other document
- A user would browse the results sequentially

3.5 Recommended reading

- S.E. Robertson, «The probability ranking principle in IR». Journal of Documentation 33, 294-304, 1977
- **C. J. van Rijsbergen, «Information Retrieval», 2nd Edition**, Butterworth-Heinemann, Newton, MA, USA, 1979

4 Overview of Text Retrieval Methods

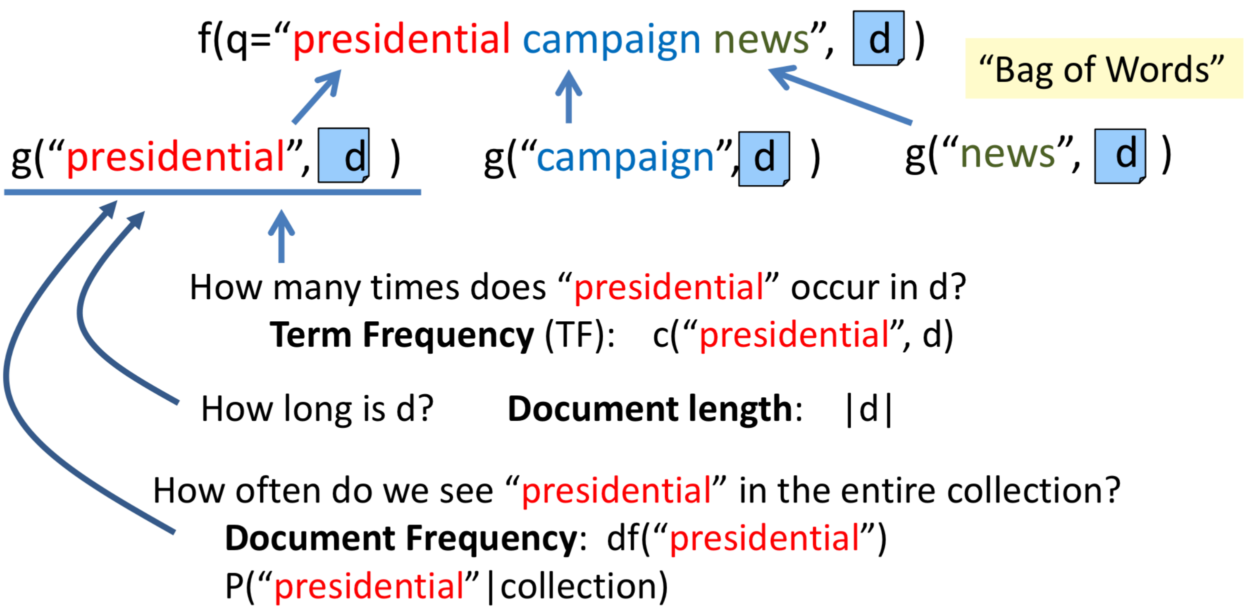
4.1 How to Design a Ranking Function

- **Query**: $q = q_1, \dots, q_m$, where $q_i \in V$
- **Document**: $d = d_1, \dots, d_n$, where $d_i \in V$
- **Ranking function**: $f(q, d) \in \mathfrak{R}$
- **Key challenge**: how to measure the likelihood that document d is relevant to query q
- **Retrieval model**: formalization of relevance (give a computational definition of relevance)

4.2 Retrieval Models

- **Similarity-based models**: $f(q, d) = similarity(q, d)$
  - Vector space model
- **Probabilistic models**: $f(d, q) = p(R = 1 \mid d, q)$, where $R \in 0, 1$
  - Classic probabilistic model
  - Language model
  - Divergence-from-randomness model
- **Probabilistic inference model**: $f(q, d) = p(d \to q)$
- **Axiomatic model**: $f(q, d)$ must satisfy a set of constraints

4.3 Common Ideas in State of the Art Retrieval Models



State of the art ranking functions tend to rely on:

- Bag of words representation
- Term Frequency (TF) and Document Frequency (DF) of words
- Document length

4.4 Which Model Works the Best?

When optimized, the following models tend to perform equally well [Fang et al. 11]:

- **Pivoted length normalization – BM25**
- Query likelihood
- PL2

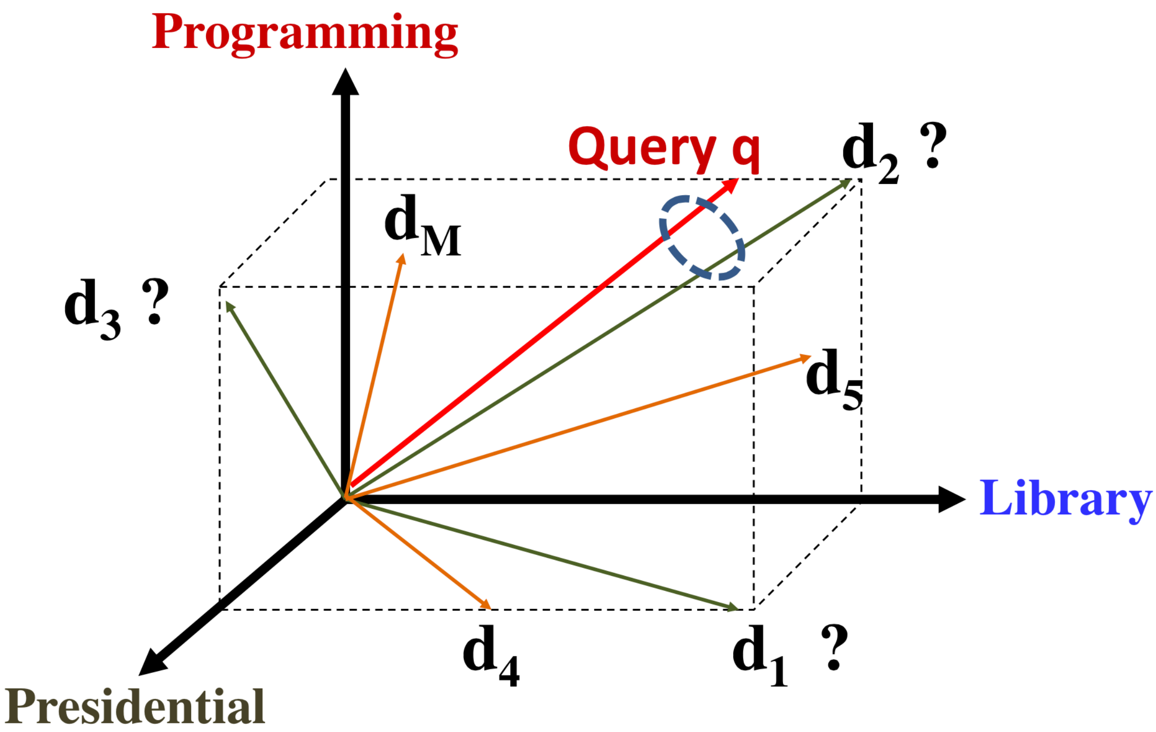
4.5 Recommended reading

- Hui Fang, Tao Tao, and Chengxiang Zhai. 2011. «Diagnostic Evaluation of Information Retrieval Models». ACM Trans. Inf. Syst. 29, 2, Article 7 (April 2011)
- ChengXiang Zhai, «Statistical Language Models for Information Retrieval», Morgan & Claypool Publishers, 2008. (Chapter 2)

5 Vector Space Retrieval Model

VSM - Vector Space Model

5.1 Vector Space Model (VSM): Illustration

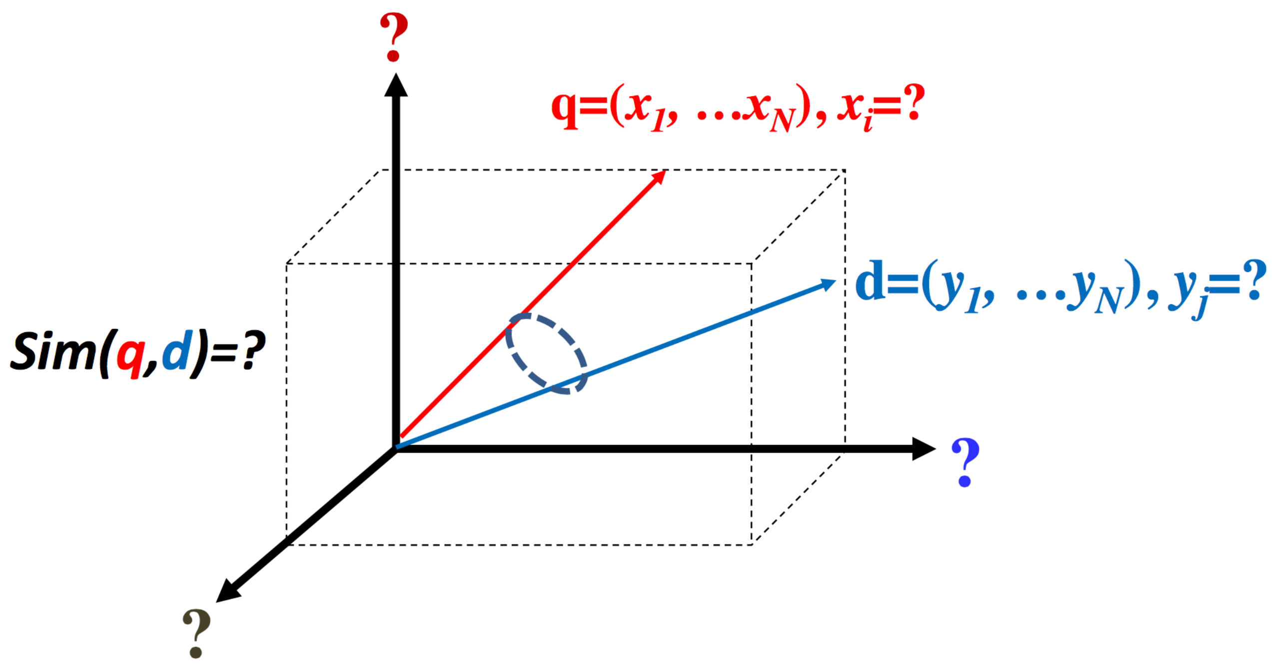


5.2 VSM Is a Framework

- Represent a doc/query by a term vector
  - **Term**: basic concept, e.g., word or phrase
  - Each term defines one dimension
  - N terms define an **N-dimensional space**
  - **Query vector**: $q = (x_1, ... x_N), x_i \in \Re$ is query term weight
  - **Doc** vector: $d = (y_1, ... y_N), y_j \in \Re$ is doc term weight
- $relevance(q, d) \propto similarity(q, d) = f(q, d)$

5.3 What VSM Doesn't Say

- How to define/select the "basic concept" – Concepts are assumed to be orthogonal
- How to place docs and query in the space (= how to assign term weights)
  - Term weight in query indicates importance of term
  - Term weight in doc indicates how well the term characterizes the doc
- How to define the similarity measure



5.4 Simplest VSM = Bit-Vector + Dot-Product + BOW

q=(x_1, …x_N)

d=(y_1, …y_N)

$x_i, y_i \in \{0,1\}$

1: word W_i is **present**

0: word W_i is **absent**

$$Sim(q,d) = q.d = x_1 y_1 + \ldots + x_N y_N = \sum_{i=1}^{N} x_i y_i$$

Simplest VSM:

- Dimension = word
- Vector = 0-1 bit vector (word presence/absence)
- Similarity = dot product
- f(q,d) = number of distinct query words matched in d

5.5 Improved Instantiation

Improved VSM:

- Dimension = word
- Vector = TF-IDF weight vector
- Similarity = dot product

5.6 Improved VSM with Term Frequency (TF) Weighting

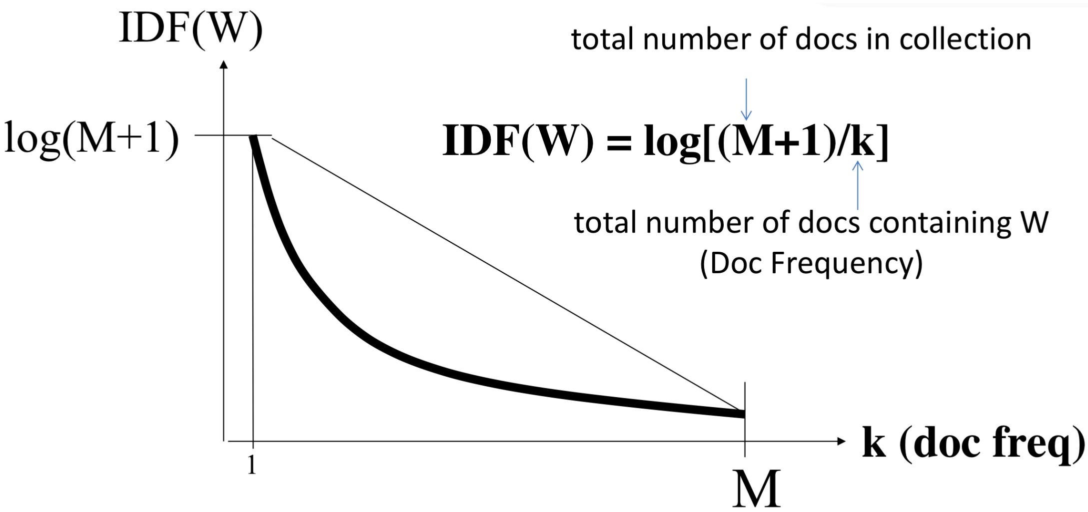
$q=(x_1, \ldots x_N)$ $\boxed{x_i = \text{count of word } W_i \text{ in query}}$

$d=(y_1, \ldots y_N)$ $\boxed{y_i = \text{count of word } W_i \text{ in doc}}$

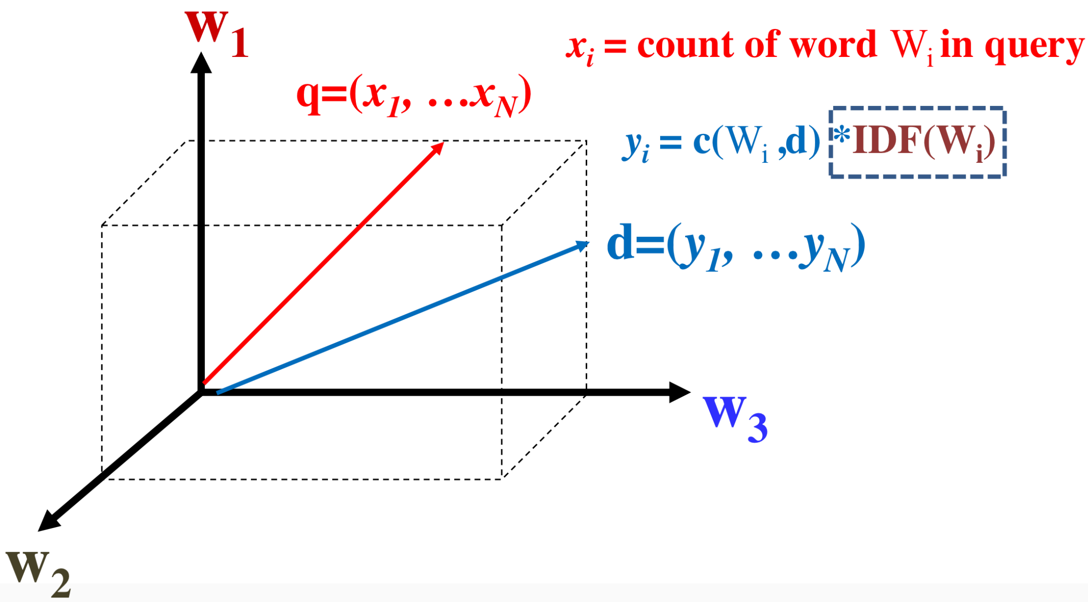
$$Sim(q,d)=q.d= x_1 y_1 + \ldots + x_N y_N = \sum_{i=1}^{N} x_i\, y_i$$

5.7 IDF Weighting: Penalizing Popular Terms

IDF — inverse document frequency



5.8 Adding Inverse Document Frequency (IDF)



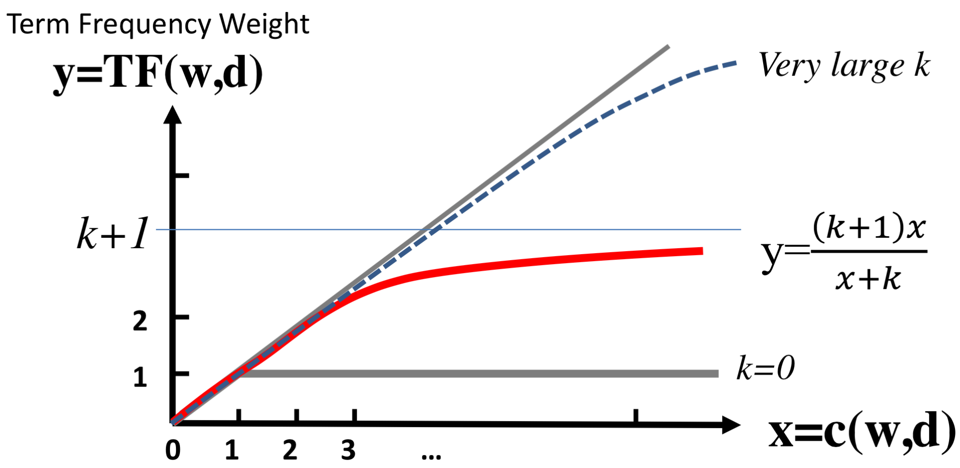
5.9 Ranking Function with TF-IDF Weighting

$$f(q,d) = \sum_{i=1}^{N} x_i y_i = \sum_{w \in q \cap d} c(w,q)\, c(w,d) \log \frac{M+1}{df(w)}$$

- $w \in q \cap d$ - all matched query (q) words in document (d)
- $c(w,q)$ - count of word w in document d
- M - total number of documents in collection
- $df(w)$ - Doc Frequency (total number of documents containing word w)

5.10 TF Transformation: BM25 Transformation

BM = Best Matching



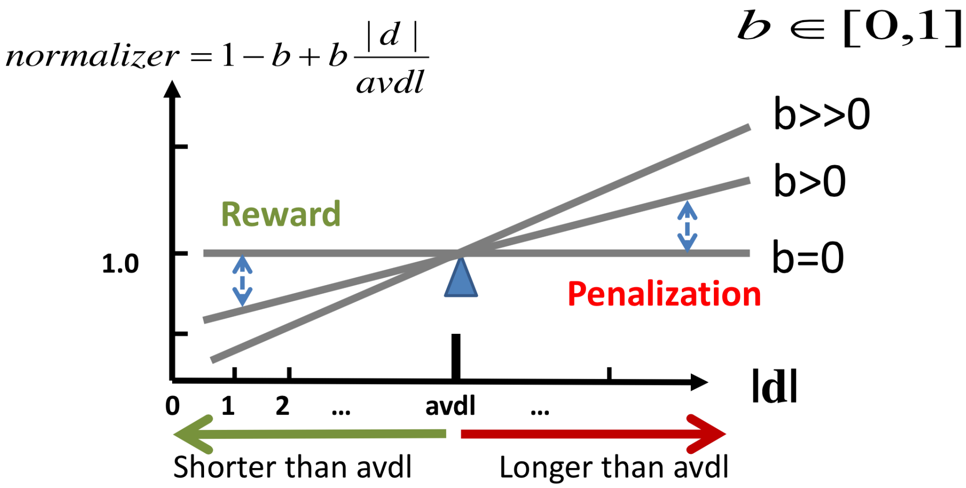
5.11 TF Transformation: summary

- Sublinear TF Transformation is needed to
  - capture the intuition of «diminishing return» from higher TF
  - avoid dominance by one single term over all others
- BM25 Transformation
  - has an upper bound
  - is robust and effective
- Ranking function with BM25 TF ($k \geqslant 0$):

$$f(q,d) = \sum_{i=1}^{N} x_i y_i = \sum_{w \in q \cap d} c(w,q) \frac{(k+1)c(w,d)}{c(w,d)+k} \log \frac{M+1}{df(w)}$$

5.12 Pivoted Length Normalization

Pivoted length normalizer: use average doc length as «pivot»[2]. Normalizer = 1 if $|d|$ = average doc length (avdl).



5.13 State of the Art VSM Ranking Functions

Pivoted Length Normalization VSM [Singhal et al 96]:

$$f(q,d) = \sum_{w \in q \cap d} c(w,q) \frac{\ln[1+\ln(1+c(w,d))]}{1-b+b\frac{|d|}{avdl}} \log \frac{M+1}{df(w)}$$

BM25/Okapi [Robertson & Walker 94]:

$$f(q,d) = \sum_{w \in q \cap d} c(w,q) \frac{(k+1)\, c(w,d)}{c(w,d)+k\left(1-b+b\frac{|d|}{avdl}\right)} \log \frac{M+1}{df(w)}$$

5.14 Further Improvement of VSM?

- Improved instantiation of dimension?
  - stemmed words, stop word removal, phrases, latent semantic indexing (word clusters), character n-grams, ...
  - bag-of-words with phrases is often sufficient in practice
  - Language-specific and domain-specific tokenization is important to ensure "normalization of terms"
- Improved instantiation of similarity function?
  - cosine of angle between two vectors?
  - Euclidean?
  - dot product seems still the best (sufficiently general especially with appropriate term weighting)

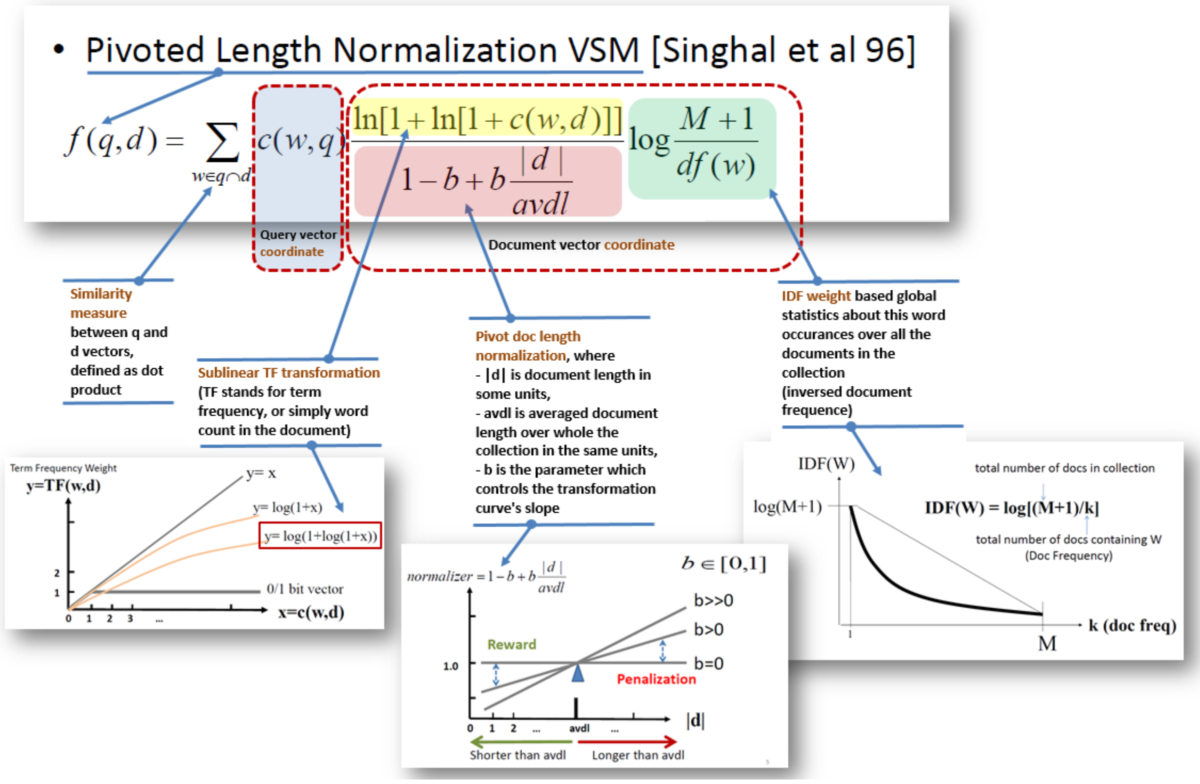
[2] Pivot - стержень; точка опоры, вращения

5.15 Further Improvement of BM25

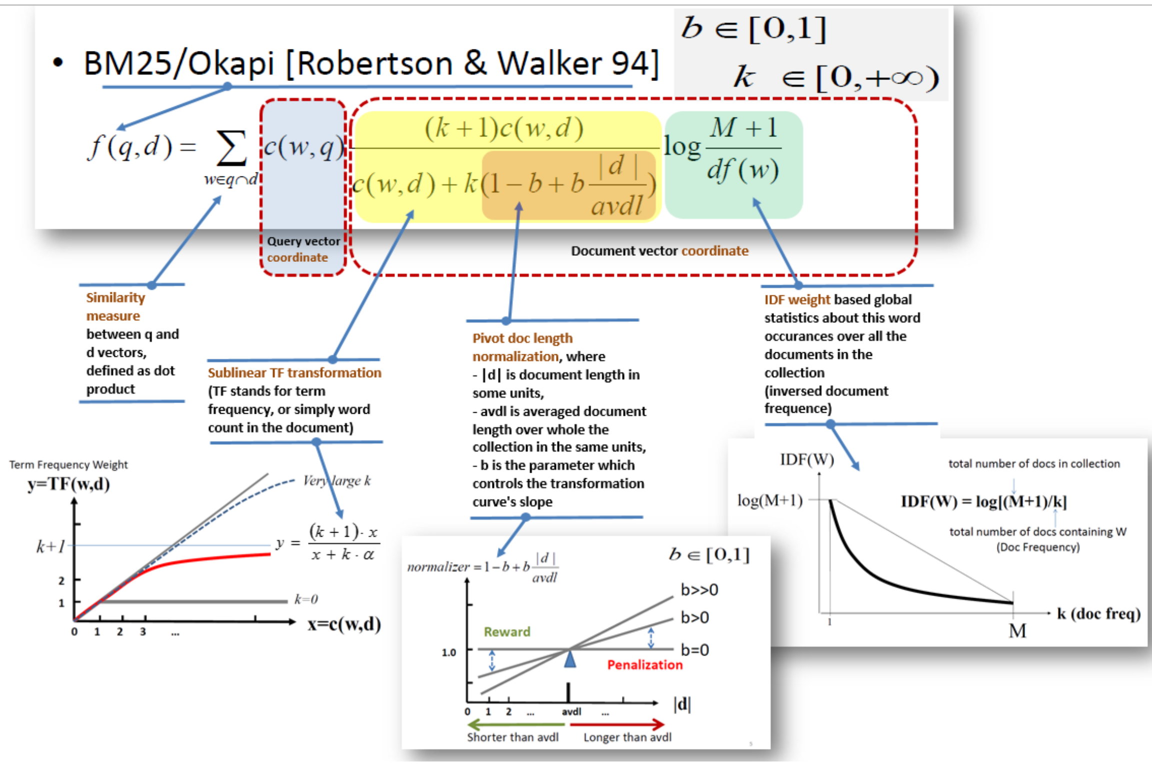
- BM25F [Robertson & Zaragoza 09]
  - Use BM25 for documents with structures («F»=fields)
  - Key idea: combine the frequency counts of terms in all fields and then apply BM25 (instead of the other way)
- BM25+ [Lv & Zhai 11]
  - Address the problem of over penalization of long documents by BM25 by adding a small constant to TF
  - Empirically and analytically shown to be better than BM25

5.16 Summary of Vector Space Model

- Relevance(q,d) = similarity(q,d)
- Query and documents are represented as vectors
- Heuristic[3] design of ranking function
- Major term weighting heuristics
  - TF weighting and transformation
  - IDF weighting
  - Document length normalization
- BM25 and Pivoted normalization seem to be most effective



[3] Heuristic - эвристический

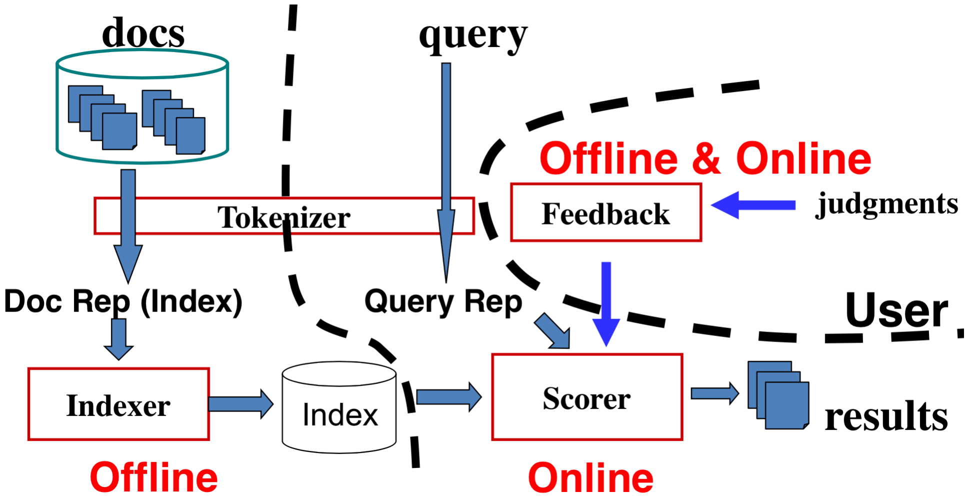


5.17 Recommended reading

- A.Singhal, C.Buckley, and M.Mitra. «Pivoted document length normalization». In Proceedings of ACM SIGIR 1996.
- S. E. Robertson and S. Walker. «Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval», Proceedings of ACM SIGIR 1994.
- S. Robertson and H. Zaragoza. «The Probabilistic Relevance Framework: BM25 and Beyond», Found. Trends Inf. Retr. 3, 4 (April 2009).
- Y. Lv, C. Zhai, «Lower-bounding term frequency normalization». In Proceedings of ACM CIKM 2011.

6 Implementation of TR Systems

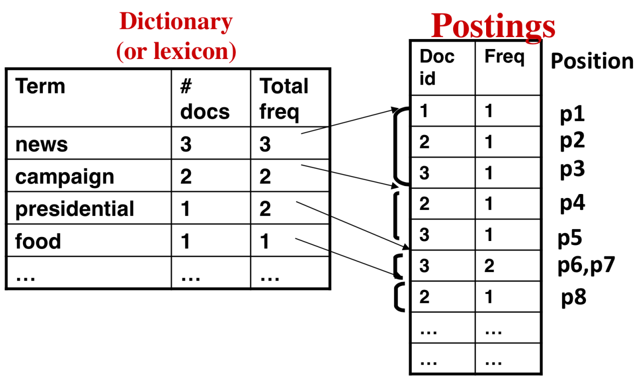
6.1 Typical TR System Architecture



6.2 Tokenization

- Normalize lexical units: words with similar meanings should be mapped to the same indexing term
- Stemming: mapping all inflectional forms of words to the same root form
- Some languages (e.g., Chinese) pose challenges in word segmentation

6.3 Inverted Index

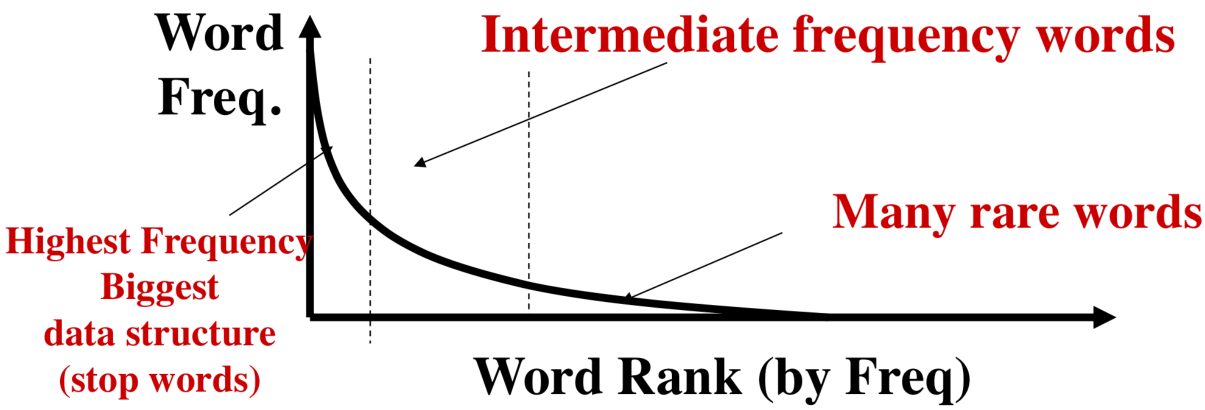


6.4 Empirical Distribution of Words

There are stable language-independent patterns in how people use natural languages:

- A few words occur very frequently; most occur rarely. E.g., in news articles:
  - Top 4 words: 10 15% word occurrences
  - Top 50 words: 35 40% word occurrences
- The most frequent word in one corpus may be rare in another

6.5 Zipf's Law



$$F(w) = \frac{C}{r(w)^{\alpha}}, \alpha \approx 1, C \approx 0.1$$

rank $\times$ frequency $\approx$ constant:

- $F(w)$ - word frequency
- $r(w)$ - word rank

6.6 Data Structures for Inverted Index

- Dictionary: modest size
  - Needs fast random access
  - Preferred to be in memory
  - Hash table, B-tree, trie, ...
- Postings: huge
  - Sequential access is expected
  - Can stay on disk
  - May contain docID, term freq., term pos, etc
  - Compression is desirable

6.7 Constructing Inverted Index

Sort-based method:

- Step 1: Collect local (termID, docID, freq) tuples from documents
- Step 2: Sort local tuples by termID (to make «runs») and save to files
- Step 3: Pair-wise merge runs
- Step 4: Output inverted file

6.8 Inverted Index Compression

In general, leverage skewed distribution of values and use variable-length encoding:

- TF compression:
  - Small numbers tend to occur far more frequently than large numbers (Zipf's law)
  - Fewer bits for small (high frequency) integers at the cost of more bits for large integers
- Doc ID compression:
  - «d-gap» (store difference): $d_1, d_2 - d_1, d_3 - d_2, \dots$
  - Feasible due to sequential access

6.9 Integer Compression Methods

- **Binary**: equal-length coding
- **Unary**: $x \geqslant 1$ is coded as $x - 1$ one bits followed by 0, e.g., 3=> 110; 5=>11110
- γ-**code**: x => unary code for $1 + \lfloor \log x \rfloor$ followed by uniform code for $x - 2^{\lfloor \log x \rfloor}$ in $\lfloor \log x \rfloor$ bits, e.g., 3=>101, 5=>11001
- δ-**code**: same as γ-code, but replace the unary prefix with γ-code. E.g., 3=>1001, 5=>10101

6.10 General Form of Scoring Function

$$f(q, d) = f_a \left(h \left(g(t_1, d, q), \dots, g(t_k, d, q) \right), f_d(d), f_q(q) \right)$$

- $f_d(d), f_q(q)$ - adjustment factors of document and query
- $g(t_i, d, q)$ - weight of a **matched** query term t_i in d
- $h()$ - weights aggregation function
- $f_a()$ - final score adjustment function

6.11 A General Algorithm for Ranking Documents

- $f_d(d)$ - can be precomputed at index time, $f_q(q)$ - at query time
- Maintain a score accumulator for each d to compute h
- For each query term t_i
  - Fetch the inverted list $\{(d_1, f_1), \dots, (d_n, f_n)\}$
  - For each entry (d_j, f_j), compute $g(t_i, d_j, q)$, and update score accumulator for doc d_i to incrementally compute h
- Adjust the score to compute f_a, and sort

6.12 Further Improving Efficiency

- Caching (e.g., query results, list of inverted index)
- Keep only the most promising accumulators
- Scaling up to the Web-scale? (need parallel processing)

6.13 Some Text Retrieval Toolkits

- Lucene
- Lemur/Indri
- Terrier
- MeTA
- More can be found here

6.14 Summary of System Implementation

- Inverted index and its construction
  - Preprocess data as much as we can
  - Compression when appropriate
- Fast search using inverted index
  - Exploit inverted index to accumulate scores for documents matching a query term
  - Exploit Zipf's law to avoid touching many documents not matching any query term
  - Can support a wide range of ranking algorithms
- Further scaling up using distributed file system, parallel processing, and caching

6.15 Recommended reading

- Ian H. Witten, Alistair Moffat, Timothy C. Bell: «Managing Gigabytes: Compressing and Indexing Documents and Images», Second Edition. Morgan Kaufmann, 1999.
- Stefan Büttcher, Charles L. A. Clarke, Gordon V. Cormack: «Information Retrieval - Implementing and Evaluating Search Engines». MIT Press, 2010.

7 Evaluation of Text Retrieval Systems

7.1 The Cranfield Evaluation Methodology

A methodology for laboratory testing of system components developed in 1960s. General idea is to build reusable test collections and define measures. A test collection can then be reused many times to compare different systems.

- A sample collection of documents (simulate real document collection)
- A sample set of queries/topics (simulate user queries)
- Relevance judgments (ideally made by users who formulated the queries) => ideal ranked list
- Measures to quantify how well a system's result matches the ideal ranked list

7.2 Evaluating a Set of Retrieved Docs

	Retrieved	**Not Retrieved**
Relevant	a	b
Not Relevant	c	d

- Precision: are the retrieved results all relevant?

$$Precision = \frac{a}{a+c}$$

- Recall: have all the relevant documents been retrieved?

$$Recall = \frac{a}{a+b}$$

- In reality, high recall tends to be associated with low precision

7.3 Combine Precision and Recall: F-Measure

$$F_\beta = \frac{1}{\frac{\beta^2}{\beta^2+1}\frac{1}{R} + \frac{1}{\beta^2+1}\frac{1}{P}} = \frac{(\beta^2+1)\cdot P\cdot R}{\beta^2\cdot P+R}$$

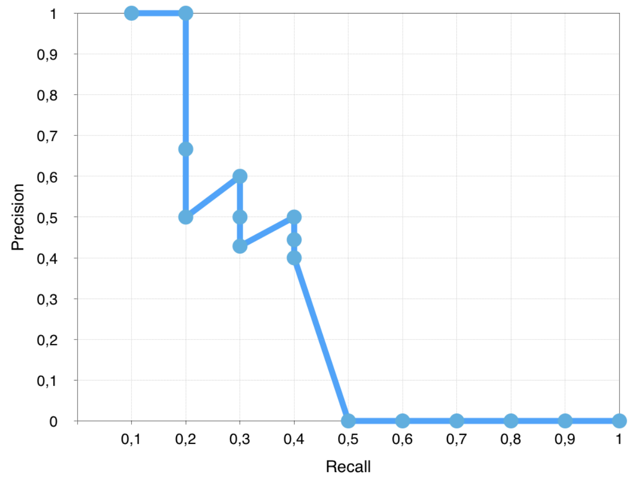
- P - precision
- R - recall
- β - parameter, often set to 1:

$$F_1 = \frac{2\cdot P\cdot R}{P+R}$$

7.4 Evaluating Ranking: Precision-Recall (PR) Curve

- Total number of relevant documents in collection: $a+b=10$
- Number of retrieved documents: $a+c=10$

Relevance	Precision	Recall
D1 +	1/1	1/10
D2 +	2/2	2/10
D3 -	2/3	2/10
D4 -	2/4	2/10
D5 +	3/5	3/10
D6 -	3/6	3/10
D7 -	3/7	3/10
D8 +	4/8	4/10
D9 -	4/9	4/10
D10 -	4/10	4/10



7.5 How to Summarize a Ranking

Average Precision is sensitive to the rank of each relevant document:

$$AveP = \frac{\sum_{k=1}^{a+c} P(k) \cdot rel(k)}{a+b} = \sum_{k=1}^{a+c} P(k) \cdot \Delta r(k)$$

- $a + c$ - number of retrieved documents
- $a + b$ - total number of relevant documents in collection
- $P(k)$ - the precision at cut-off k in the list
- $rel(k)$ - indicator function equaling 1 if the item at rank k is a relevant document, zero otherwise
- $\Delta r(k)$ - the change in recall that happened between cut-off $k-1$ and cut-off k

In special case, when there's only one relevant document in the collection (e.g., known item search):

- Average Precision = **Reciprocal Rank** = 1/r, where r is the rank position of the single relevant doc

7.6 Mean Average Precision (MAP)

In case of multiple queries:

- MAP = arithmetic mean of average precision over a set of queries

$$MAP = \frac{\sum_{q=1}^{N} AveP(q)}{N}, \textit{ where N is the number of queries}$$

- gMAP = geometric mean of average precision over a set of queries

$$gMAP = \sqrt[N]{\prod_{q=1}^{N} AveP(q)} = \exp \frac{\sum_{q=1}^{N} \log\left(AveP(q)\right)}{N}$$

7.7 Summary on Average Precision

- Precision-Recall curve characterizes the overall accuracy of a ranked list
- The **actual** utility of a ranked list depends on how many top-ranked results a user would examine
- Average Precision is the standard measure for comparing two ranking methods
  - Combines precision and recall
  - Sensitive to the rank of **every** relevant document

7.8 Multi-level Relevance Judgments

Discounted cumulative gain (DCG) is a measure of ranking quality. Two assumptions are made in using DCG and its related measures:

- Highly relevant documents are more useful when appearing earlier in a search engine result list (have higher ranks)
- Highly relevant documents are more useful than marginally relevant documents, which are in turn more useful than irrelevant documents.

For a rank position p:

- **Cumulative Gain**: $\mathrm{CG_p} = \sum_{i=1}^{p} rel_i$, where rel_i is the graded relevance of the result at position i
- **Discounted Cumulative Gain**: $\mathrm{DCG_p} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$
- Alternative version of **Discounted Cumulative Gain**: $\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}$
- **Normalized DCG**: $\mathrm{nDCG_p} = \frac{\mathrm{DCG_p}}{\mathrm{IDCG_p}}$, where $\mathrm{IDCG_p}$ is an Ideal DCG (the maximum possible DCG till position p)

For example, each document is to be judged on a scale of 0-3 with 0 meaning irrelevant, 3 meaning completely relevant, and 1 and 2 meaning «somewhere in between»

Document	rel_i	$\frac{rel_i}{\log_2 i}$
D1	3	-
D2	2	2
D3	3	1.892
D4	0	0
D5	1	0.431
D6	2	0.774

- $\mathrm{DCG_6} = rel_1 + \sum_{i=2}^{6} \frac{rel_i}{\log_2 i} = 3 + (2 + 1.892 + 0 + 0.431 + 0.774) = 8.10$
- $\mathrm{IDCG_6} = 8.69 \quad (rel_i = 3, 3, 2, 2, 1, 0)$
- $\mathrm{nDCG_6} = \frac{DCG_6}{IDCG_6} = \frac{8.10}{8.69} = 0.932$

7.9 Statistical Significance Tests

Query	System A	System B	Sign Test	Wilcoxon
1	0.02	0.76	+	+0.74
2	0.39	0.07	-	- 0.32
3	0.16	0.37	+	+0.21
4	0.58	0.21	-	- 0.37
5	0.04	0.02	-	- 0.02
6	0.09	0.91	+	+0.82
7	0.12	0.46	+	+0.34
Average	0.20	0.40	p=1.0	p=0.9375

Нулевая гипотеза - гипотеза об отсутствии взаимосвязи или корреляции между исследуемыми переменными, об отсутствии различий (однородности) в распределениях (параметрах распределений) двух и/или более выборках.

- H0: median difference between the pairs is zero
- H1: median difference is not zero.

7.9.1 Sign test

Критерий знаков используется при проверке нулевой гипотезы о равенстве медиан двух непрерывно распределенных случайных величин

Рассмотрим две непрерывно распределенные случайные величины X и Y, и пусть нулевая гипотеза выполняется, то есть их медианы равны. Тогда $p = \mathbb{P}(X > Y) = 0.5$. Иными словами, каждая из случайных величин равновероятно больше другой.

Рассмотрим пару связных выборок $\{(x_1, y_1), \dots, (x_n, y_n)\}$. Будем считать, что в выборке нет элементов, для которых $x_i = y_i$ (иначе уберем эти элементы из выборки). Построим статистику w, равную числу элементов в выборке, при которых $x_i > y_i$. При выполнении нулевой гипотезы, эта величина имеет биномиальное распределение: $w \sim B(n, 0.5)$ с функцией вероятности

$$p_Y(k) \equiv \mathbb{P}(Y = k) = \binom{n}{k} p^k q^{n-k}, \quad k = 0, \dots, n,$$

где $\binom{n}{k} = C_n^i = \dfrac{n!}{(n-k)!\ k!}$ — биномиальный коэффициент.

Для применения критерия необходимо вычислить «левый хвост» биномиального распределения до w:

$$b = 2^{-n} \sum_{i=0}^{w} \binom{n}{i}$$

Согласно критерию, при уровне значимости α: если $b \notin [\alpha/2,\ 1 - \alpha/2]$, то нулевая гипотеза $p \neq 0.5$ отвергается.

7.9.2 Wilcoxon signed-rank test

The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test used when comparing two related samples or repeated measurements on a single sample to assess whether their population mean ranks differ.

Let N be the sample size, the number of pairs. Thus, there are a total of $2N$ data points. For $i = 1, \ldots, N$, let $x_{1,i}$ and $x_{2,i}$ denote the measurements.

- For $i = 1, \ldots, N$, calculate $|x_{2,i} - x_{1,i}|$ and $\text{sign}(x_{2,i} - x_{1,i})$.
- Exclude pairs with $|x_{2,i} - x_{1,i}| = 0$. Let N_r be the reduced sample size.
- Order the remaining N_r pairs from smallest absolute difference to largest absolute difference, $|x_{2,i} - x_{1,i}|$.
- Rank the pairs, starting with the smallest as 1. Ties receive a rank equal to the average of the ranks they span. Let R_i denote the rank.
- Calculate the test statistic W, the absolute value of the sum of the signed ranks:

$$W = \left| \sum_{i=1}^{N_r} [\text{sign}(x_{2,i} - x_{1,i}) \cdot R_i] \right|$$

- As N_r increases, the sampling distribution of W converges to a normal distribution. Thus,
  - For $N_r \geqslant 10$, a z-score can be calculated as $z = \frac{W - 0.5}{\sigma_W}, \sigma_W = \sqrt{\frac{N_r(N_r+1)(2N_r+1)}{6}}$. If $z > z_{critical}$ then reject H_0
  - For $N_r < 10$, W is compared to a critical value from a reference table. If $W \geqslant W_{critical,N_r}$ then reject H_0

Example:

			$x_{2,i} - x_{1,i}$	
i	$x_{2,i}$	$x_{1,i}$	sgn	abs
1	125	110	1	15
2	115	122	–1	7
3	130	125	1	5
4	140	120	1	20
5	140	140		0
6	115	124	–1	9
7	140	123	1	17
8	125	137	–1	12
9	140	135	1	5
10	135	145	–1	10

order by absolute difference

			$x_{2,i} - x_{1,i}$			
i	$x_{2,i}$	$x_{1,i}$	sgn	abs	R_i	sgn $\cdot R_i$
5	140	140		0		
3	130	125	1	5	1.5	1.5
9	140	135	1	5	1.5	1.5
2	115	122	–1	7	3	–3
6	115	124	–1	9	4	–4
10	135	145	–1	10	5	–5
8	125	137	–1	12	6	–6
1	125	110	1	15	7	7
7	140	123	1	17	8	8
4	140	120	1	20	9	9

Notice that pairs 3 and 9 are tied in absolute value. They would be ranked 1 and 2, so each gets the average of those ranks, 1.5.

$N_r = 10 - 1 = 9, W = |1.5 + 1.5 - 3 - 4 - 5 - 6 + 7 + 8 + 9| = 9$
$W < W_{\alpha=0.05,9} = 39 \therefore$ fail to reject H_0

7.10 Pooling: Avoid Judging all Documents

Pooling strategy:

- Choose a diverse set of ranking methods (TR systems)
- Have each to return top-K documents
- Combine all the top-K sets to form a pool for human assessors to judge
- Other (unjudged) documents are usually assumed to be non-relevant (though they don't have to)

Pooling strategy is okay for comparing systems that contributed to the pool, but problematic for evaluating new systems.

7.11 Summary of TR Evaluation

Evaluation is extremely important:

- TR is an empirically defined problem
- Inappropriate experiment design misguides research and applications
- Make sure to get it right for your research or application

Cranfield evaluation methodology is the main paradigm:

- MAP and nDCG: appropriate for comparing ranking algorithms
- Precision@10docs is easier to interpret from a user's perspective

Not covered:

- A-B Test [Sanderson 10]
- User studies [Kelly 09]

7.12 Recommended reading

- Donna Harman, «Information Retrieval Evaluation. Synthesis Lectures on Information Concepts, Retrieval, and Services», Morgan & Claypool Publishers 2011
- Mark Sanderson, «Test Collection Based Evaluation of Information Retrieval Systems». Foundations and Trends in Information Retrieval 4(4): 247-375 (2010)
- Diane Kelly, «Methods for Evaluating Interactive Information Retrieval Systems with Users». Foundations and Trends in Information Retrieval 3(1-2): 1-224 (2009)

8 Probabilistic Model

8.1 Basic Idea of Probabilistic Model

- Probabilistic models ranking function:

$$f(d, q) = p(R = 1 \,|\, d, q),\ \ R \in \{0, 1\}$$

- **Query Likelihood**: if a user likes document d, how likely would the user enter query q (in order to retrieve d)?
- Assumption: a user formulates a query based on an «imaginary relevant document»:

$$p(R = 1 \,|\, d, q) \approx p(q \,|\, d, R = 1)$$

- Basic idea based on user clicks (R=1):

$$f(q, d) = p(R = 1 \,|\, d, q) = \frac{count(q, d, R = 1)}{count(q, d)}$$

- How to compute $p(q \,|\, d)$? How to compute probability of text in general? $\rightarrow$ **Language Model**

8.2 Language Model

The term language model (LM) refers to a probabilistic model of text (i.e., it defines a probability distribution over sequences of words).

Uses of a Language Model:

- Representing topics
- Discovering word associations

8.3 The Simplest Language Model: Unigram LM

Unigram Language Model = word distribution

- Generate text by generating each word **independently**
- Thus, $p(w_1, w_2 \dots w_n) = p(w_1)p(w_2) \dots p(w_n)$
- Parameters: $\{p(w_i)\} : p(w_1) + \dots + p(w_N) = 1$ (N is vocabulary size)
- Text = sample drawn according to this **word distribution**

Maximum Likelihood (ML) Estimator:

$$p(w \,|\, \theta) = p(w \,|\, d) = \frac{c(w, d)}{|d|}$$

- θ - document language model
- $c(w, d)$ - count on word w in document d
- $|d|$ - length of document d

8.4 Recommended reading

- Chris Manning and Hinrich Schütze, «Foundations of Statistical Natural Language Processing», MIT Press. Cambridge, MA: May 1999.
- Rosenfeld, R., «Two decades of statistical language modeling: where do we go from here?», Proceedings of the IEEE , vol.88, no.8, pp.1270,1278, Aug. 2000

8.5 Ranking based on Query Likelihood

- Query: $q = w_1 w_2 \dots w_n$
- Vocabulary of the language of the documents: $V = \{w_1, \dots, w_{|V|}\}$
- $c(w, q)$ - count of word w in query q

How likely would we observe this query from this document model?

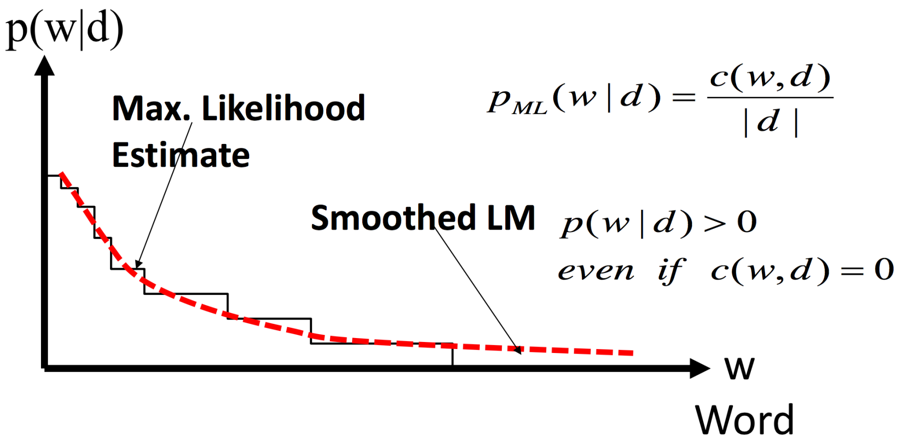
$$p(q \mid d) = p(w_1 \mid d) \times \dots \times p(w_n \mid d)$$

Retrieval problem $\rightarrow$ estimation of $p(w_i \mid d)$

$$f(q, d) = \log p(q \mid d) = \sum_{i=1}^{n} \log p(w_i \mid d) = \sum_{w \in V} c(w, q) \cdot \log p(w \mid d)$$

8.6 How to Estimate $p(w|d)$

Smoothing of $p(w|d)$ is necessary for query likelihood:



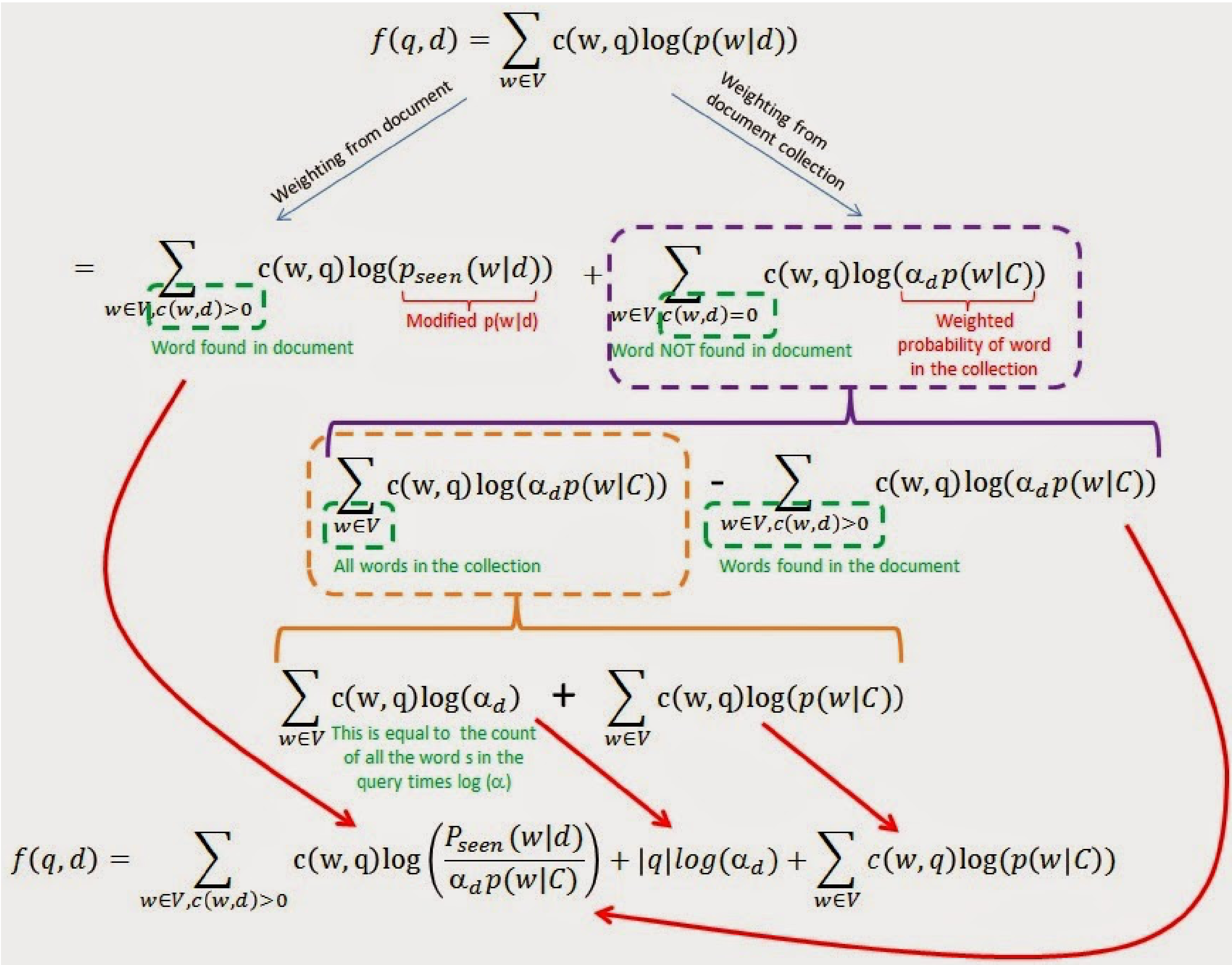
Key Question: what probability should be assigned to an unseen word? Let the probability of an unseen word be proportional to its probability given by a reference LM. One possibility: Reference LM = Collection LM:

Discounted ML estimate

$$p(w \mid d) = \begin{cases} p_{Seen}(w \mid d) & \text{if } w \text{ is seen in } d \\ \alpha_d \, p(w \mid C) & \text{otherwise} \end{cases}$$

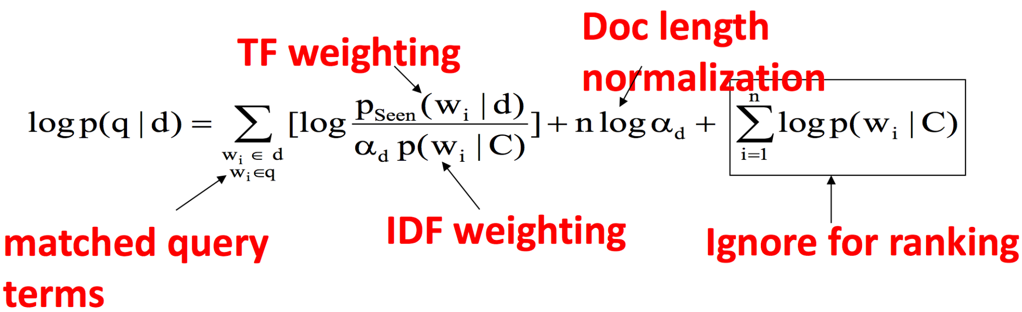
Collection language model

8.7 Rewriting the Ranking Function with Smoothing



8.8 Benefit of Rewriting

Smoothing with $p(w|d)$ leads to a general ranking formula for query likelihood with TF-IDF weighting and document length normalization:



8.9 Smoothing Methods

$$f(q,d) = \sum_{w_i \in d, w_i \in q} c(w_i, q) \log \frac{p_{seen}(w_i|d)}{\alpha_d p(w_i|C)} + n \log \alpha_d$$

8.9.1 Jelinek-Mercer Smoothing

Jelinek-Mercer: Fixed coefficient linear interpolation

$$p_{seen}(w \,|\, d) = (1-\lambda)\,\frac{c(w,d)}{|d|} + \lambda\, p(w \,|\, C), \quad \alpha_d = \lambda, \quad \lambda \in [0,1]$$

Ranking Function:

$$f_{JM}(q,d) = \sum_{w\in d, w\in q} c(w,q) \log\left(1 + \frac{1-\lambda}{\lambda}\,\frac{c(w,d)}{|d|\,p(w \,|\, C)}\right)$$

8.9.2 Dirichlet Prior (Bayesian) Smoothing

Dirichlet Prior: Adding pseudo counts; adaptive interpolation

$$p_{seen}(w \,|\, d) = \frac{c(w,d) + \mu\, p(w \,|\, C)}{|d| + \mu} = \frac{|d|}{|d|+\mu}\cdot\frac{c(w,d)}{|d|} + \frac{\mu}{|d|+\mu}\cdot p(w \,|\, C)$$

$$\alpha_d = \frac{\mu}{|d|+\mu}, \quad \mu \in [0, +\infty)$$

Ranking Function:

$$f_{DIR}(q,d) = \sum_{w\in d, w\in q} c(w,q) \log\left(1 + \frac{c(w,d)}{\mu\, p(w \,|\, C)}\right) + n\, \log\frac{\mu}{\mu + |d|}$$

8.10 Summary of Query Likelihood Probabilistic Model

- Effective ranking functions obtained using pure probabilistic modeling
  - Assumption 1: $Relevance(q,d) = p(R=1 \,|\, q,d) \approx p(q \,|\, d, R=1) \approx p(q \,|\, d)$
  - Assumption 2: Query words are generated independently
  - Assumption 3: Smoothing with $p(w \,|\, C)$
  - Assumption 4: JM or Dirichlet prior smoothing
- Less heuristic compared with VSM
- Many extensions have been made [Zhai 08]

8.11 Recommended reading

- ChengXiang Zhai, «Statistical Language Models for Information Retrieval» (Synthesis Lectures Series on Human Language Technologies), Morgan & Claypool Publishers, 2008.