Midterm Review

Hongning Wang

CS@UVa

Core concepts

- Search Engine Architecture
  - Key components in a modern search engine
- Crawling & Text processing
  - Different strategies for crawling
  - Challenges in crawling
  - Text processing pipeline
  - Zipf's law
- Inverted index
  - Index compression
  - Phase queries

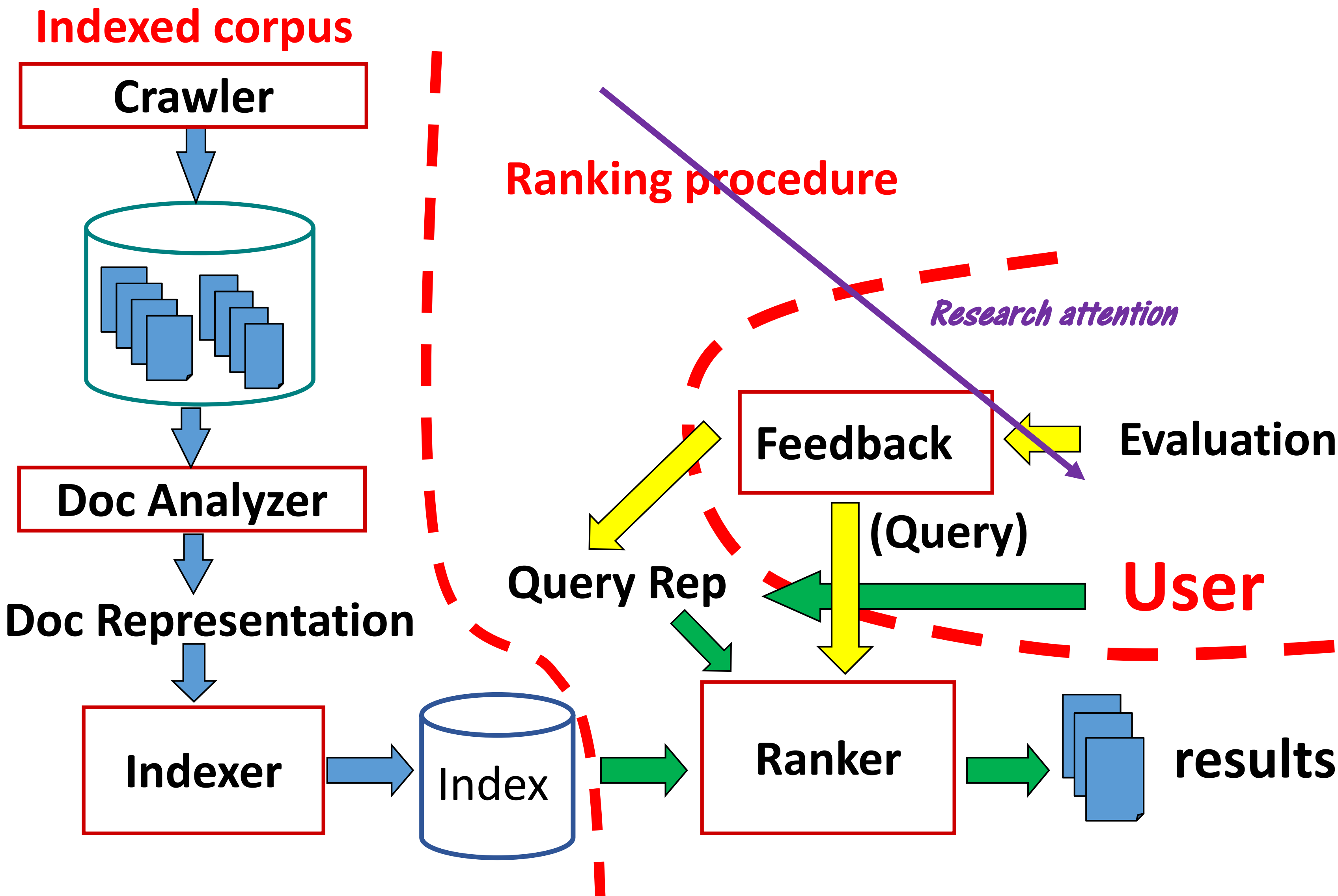
Core concepts

- Vector space model
  - Basic term/document weighting schemata
- Latent semantic analysis
  - Word space to concept space
- Probabilistic ranking principle
  - Risk minimization
  - Document generation model
- Language model
  - N-gram language models
  - Smoothing techniques

Core concepts

- Classical IR evaluations
  - Basic components in IR evaluations
  - Evaluation metrics
  - Annotation strategy and annotation consistency

Abstraction of search engine architecture



IR v.s. DBs

- Information Retrieval:
  - Unstructured data
  - Semantics of objects are subjective
  - Simple keyword queries
  - Relevance-drive retrieval
  - Effectiveness is primary issue, though efficiency is also important
- Database Systems:
  - Structured data
  - Semantics of each object are well defined
  - Structured query languages (e.g., SQL)
  - Exact retrieval
  - Emphasis on efficiency

Crawler: visiting strategy

- Breadth first
  - Uniformly explore from the entry page
  - Memorize all nodes on the previous level
  - As shown in pseudo code
- Depth first
  - Explore the web by branch
  - Biased crawling given the web is not a tree structure
- Focused crawling
  - Prioritize the new links by predefined strategies
- Challenges
  - Avoid duplicate visits
  - Re-visit policy

Automatic text indexing

- Tokenization
  - Regular expression based
  - Learning-based
- Normalization
- Stemming
- Stopword removal

Statistical property of language

discrete version of power law

- Zipf's law
  - Frequency of any word is inversely proportional to its rank in the frequency table
  - Formally
    - $f(k; s, N) = \frac{1/k^s}{\sum_{n=1}^{N} 1/n^s}$

      where k is rank of the word; N is the vocabulary size; s is language-specific parameter

Automatic text indexing

Remove non-informative words

Remove 1s

Terms	Documents													
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14
abnormalities	0	0	0	0	0	0	0	1	0	1	0	0	0	0
age	1	0	0	0	0	0	0	0	0	0	0	1	0	0
behavior	0	0	0	0	1	1	0	0	0	0	0	0	0	0
blood	0	0	0	0	0	0	0	1	0	0	1	0	0	0
close	0	0	0	0	0	0	1	0	0	0	1	0	0	0
culture	1	1	0	0	0	0	0	1	1	0	0	0	0	0
depressed	1	0	1	1	1	0	0	0	0	0	0	0	0	0
discharge	1	1	0	0	0	1	0	0	0	0	0	0	0	0
disease	0	0	0	0	0	0	0	0	1	0	1	0	0	0
fast	0	0	0	0	0	0	0	0	0	1	0	1	1	1
generation	0	0	0	0	0	0	0	0	1	0	0	0	1	0
oestrogen	0	0	1	1	0	0	0	0	0	0	0	0	0	0
patients	1	1	0	1	0	0	0	1	0	0	0	0	0	0
pressure	0	0	0	0	0	0	0	0	0	0	1	0	0	1
rats	0	0	0	0	0	0	0	0	0	0	0	0	1	1
respect	0	0	0	0	0	0	0	1	0	0	0	1	0	0
rise	0	0	0	1	0	0	0	0	0	0	0	0	0	1
study	1	0	1	0	0	0	0	0	1	0	0	0	0	0

Remove 0s

Remove rare words

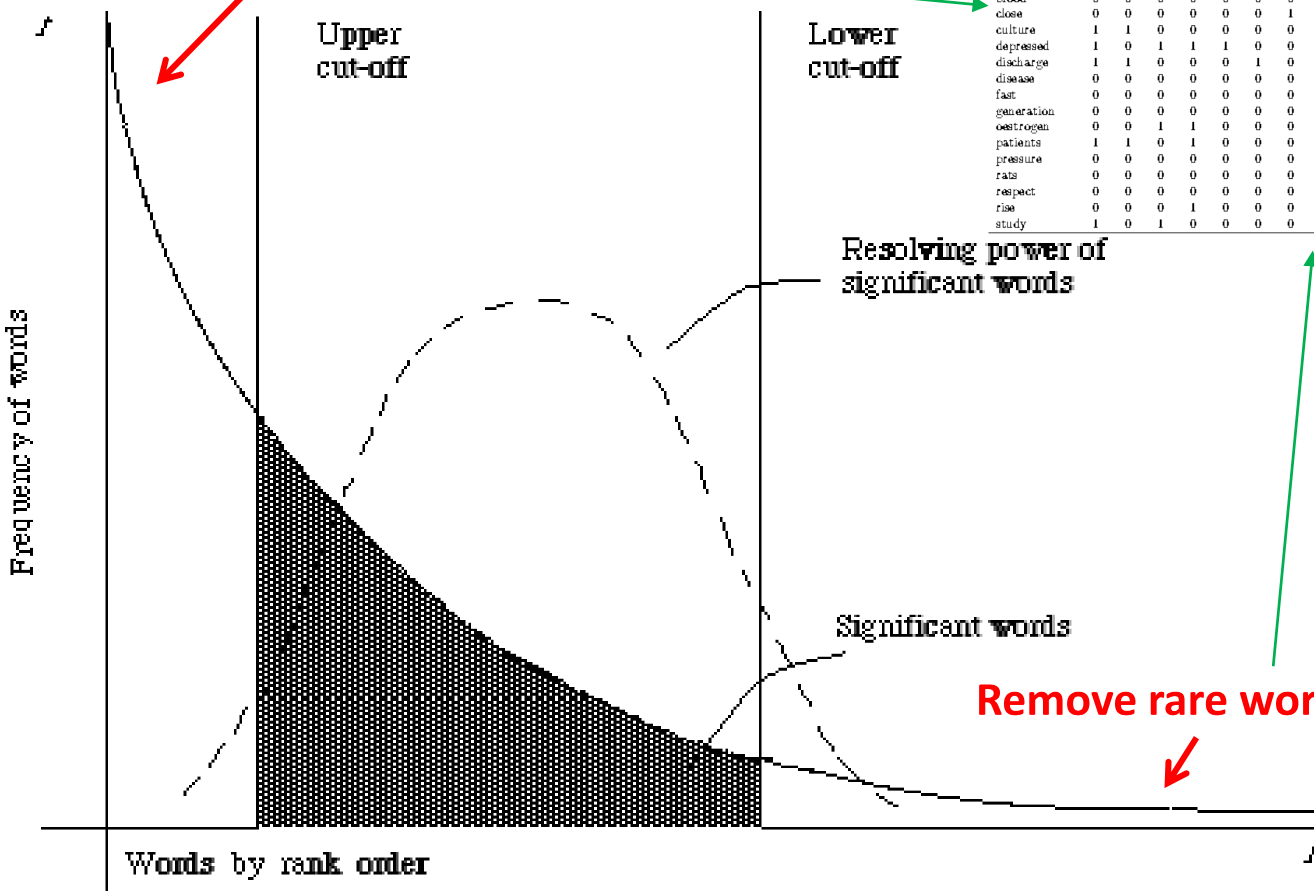
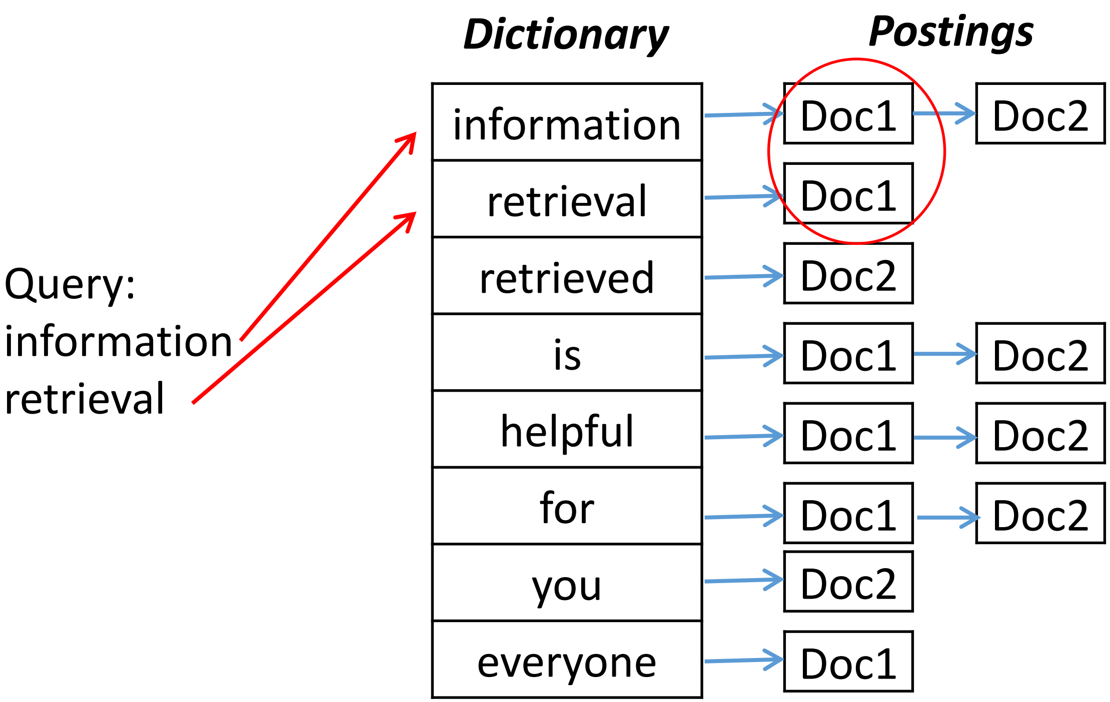


Figure 2.1. A plot of the hyperbolic curve relating f, the frequency of occurrence and r, the rank order (Adaped from Schultz[44] page 120)

Inverted index

- Build a look-up table for each word in vocabulary
  - From word to find documents!



Time complexity:

- $O(|q| * |L|)$, $|L|$ is the average length of posting list
- By Zipf's law, $|L| \ll D$

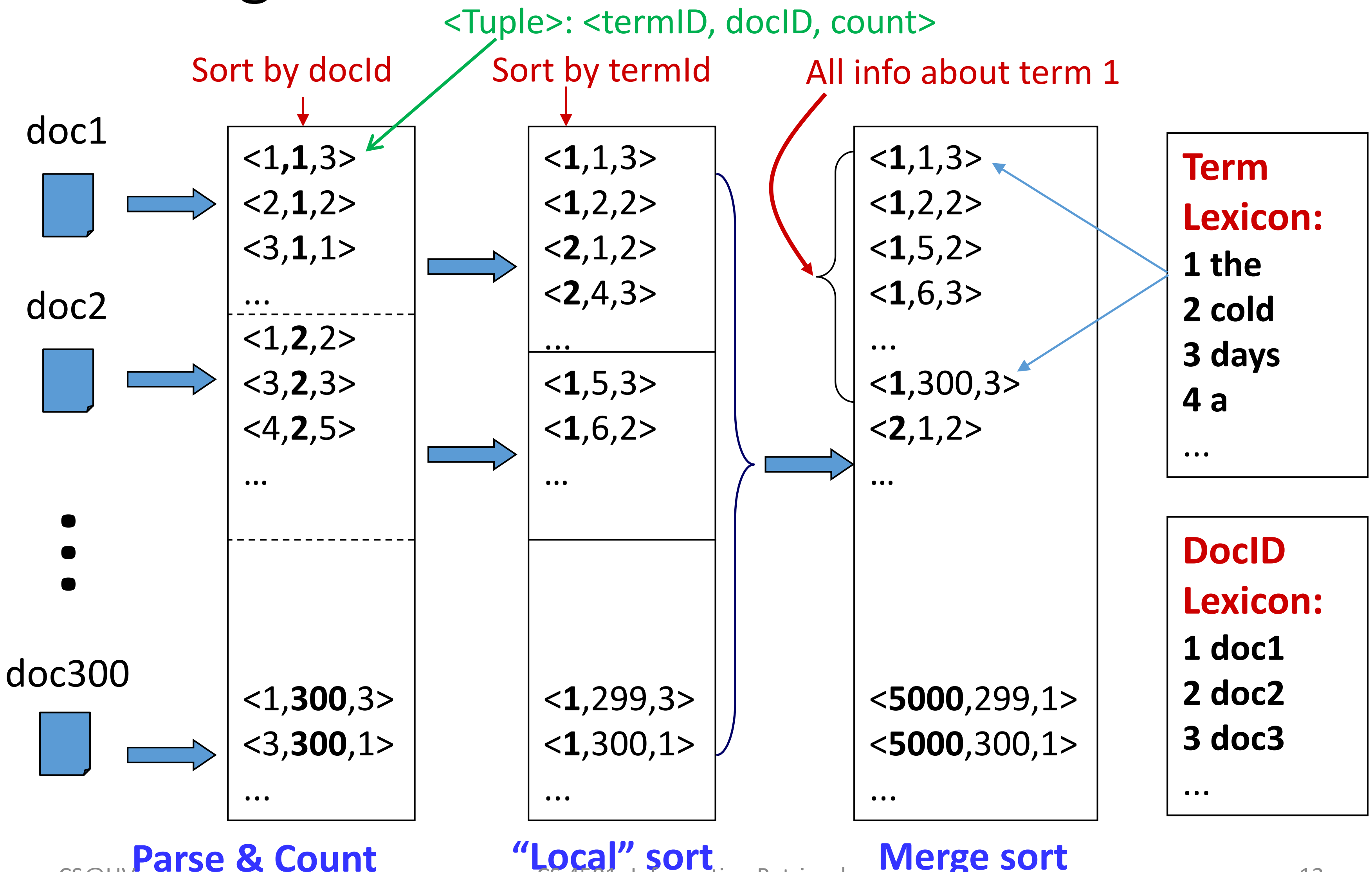
Structures for inverted index

- Dictionary: modest size
  - Needs fast random access
  - Stay in memory
    - Hash table, B-tree, trie, …
- Postings: huge
  - Sequential access is expected
  - Stay on disk
  - Contain docID, term freq, term position, …
  - Compression is needed

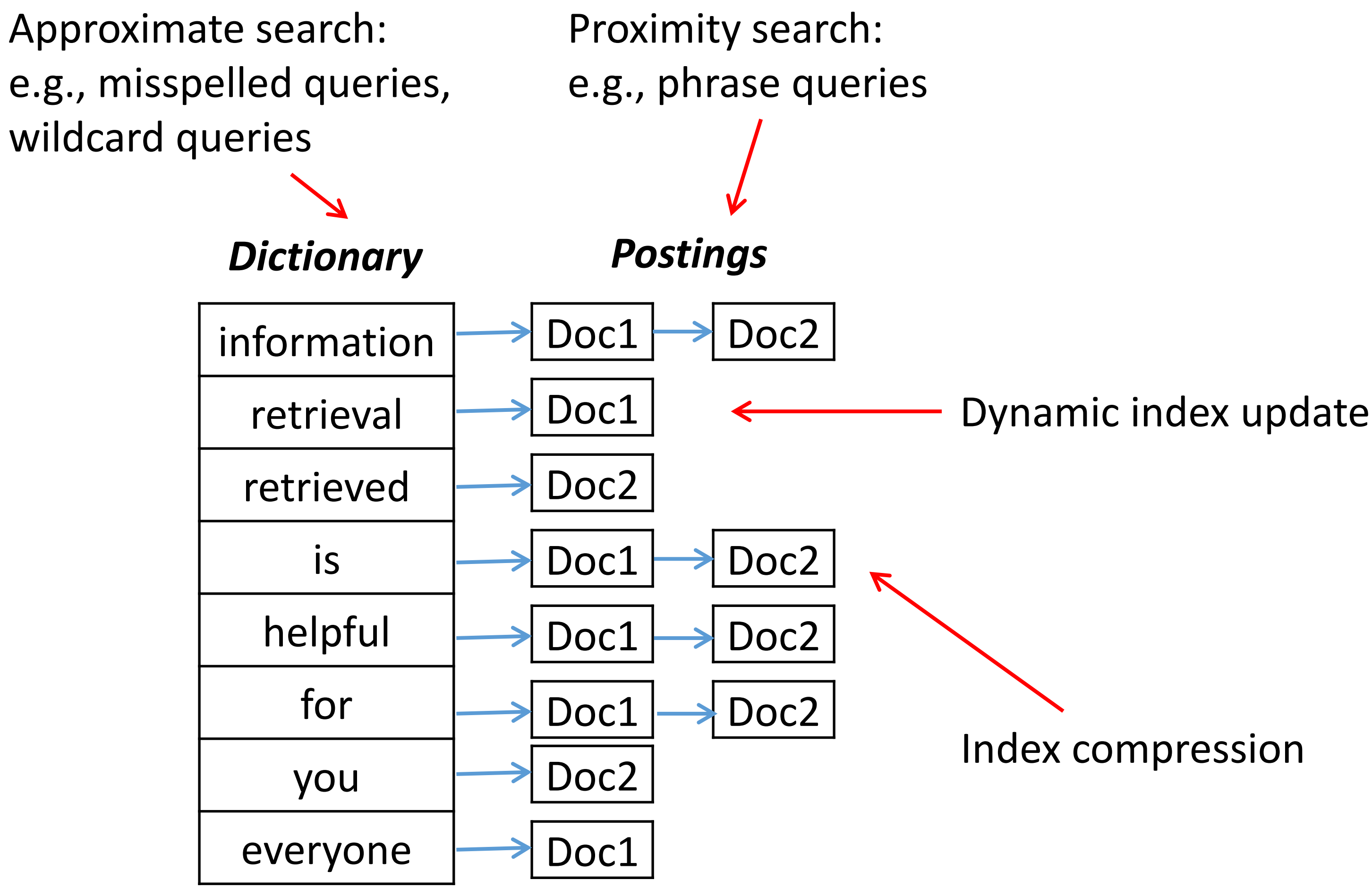
"Key data structure underlying modern IR"

- Christopher D. Manning

Sorting-based inverted index construction



A close look at inverted index



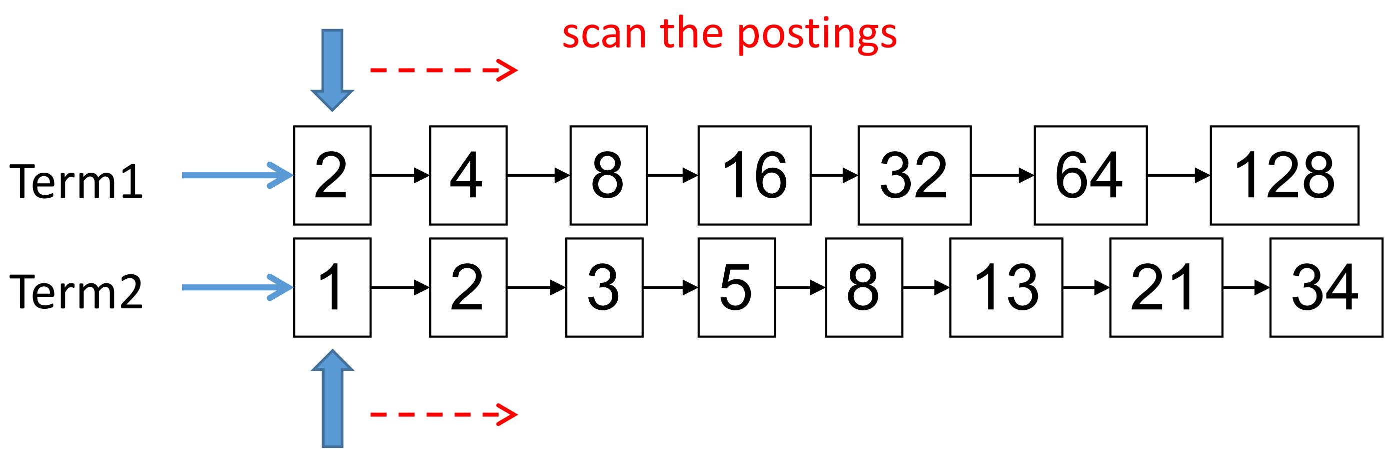
Index compression

- Observation of posting files
  - Instead of storing docID in posting, we store gap between docIDs, since they are ordered
  - Zipf's law again:
    - The more frequent a word is, the smaller the gaps are
    - The less frequent a word is, the shorter the posting list is
  - Heavily biased distribution gives us great opportunity of compression!

Information theory: entropy measures compression difficulty.

Phrase query

- Generalized postings matching
  - Equality condition check with requirement of position pattern between two query terms
    - e.g., T2.pos-T1.pos = 1 (T1 must be immediately before T2 in any matched document)
  - Proximity query: |T2.pos-T1.pos| ≤ k



Considerations in result display

- Relevance
  - Order the results by relevance
- Diversity
  - Maximize the topical coverage of the displayed results
- Navigation
  - Help users easily explore the related search space
    - Query suggestion
    - Search by example

Deficiency of Boolean model

- The query is unlikely precise
  - “Over-constrained” query (terms are too specific): no relevant documents can be found
  - “Under-constrained” query (terms are too general): over delivery
  - It is hard to find the right position between these two extremes (hard for users to specify constraints)
- Even if it is accurate
  - Not all users would like to use such queries
  - All relevant documents are **not equally** important
    - No one would go through all the matched results
- Relevance is a matter of degree!

Vector space model

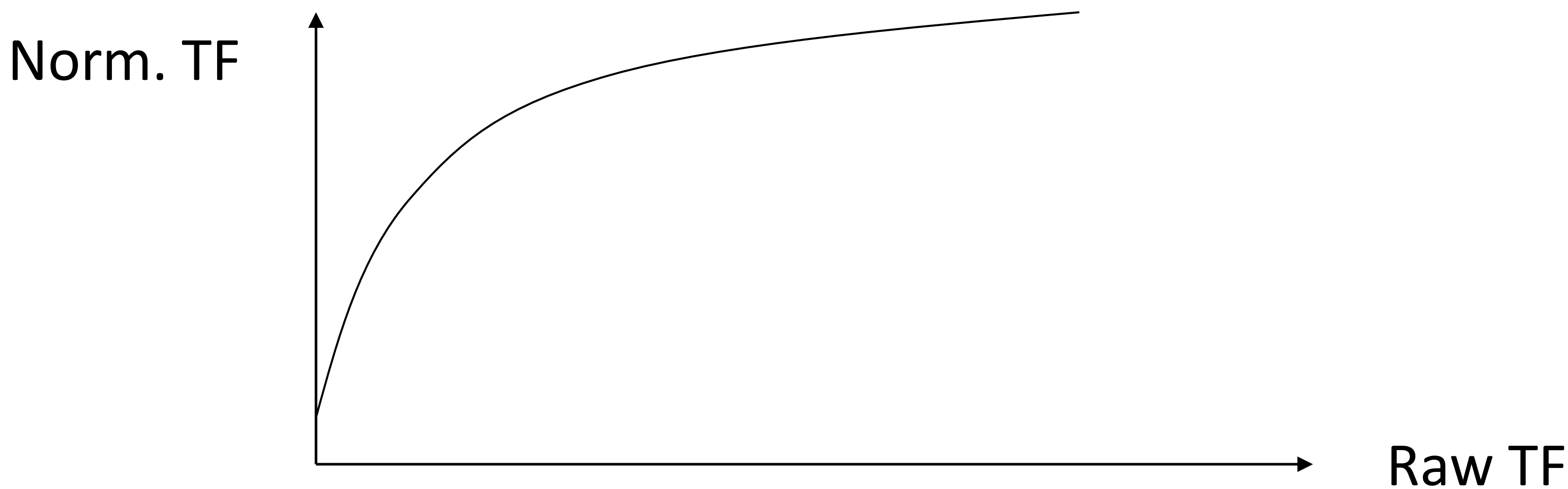
- Represent both doc and query by <u>concept</u> vectors
  - Each concept defines one dimension
  - *K* concepts define a high-dimensional space
  - Element of vector corresponds to concept weight
    - E.g., $d=(x_1,\dots,x_k)$, x_i is "importance" of concept i
- Measure relevance
  - Distance between the query vector and document vector in this concept space

What is a good “basic concept”?

- Orthogonal
  - Linearly independent basis vectors
    - “Non-overlapping” in meaning
    - No ambiguity
- Weights can be assigned automatically and accurately
- Existing solutions
  - Terms or N-grams, i.e., bag-of-words
  - Topics, i.e., topic model

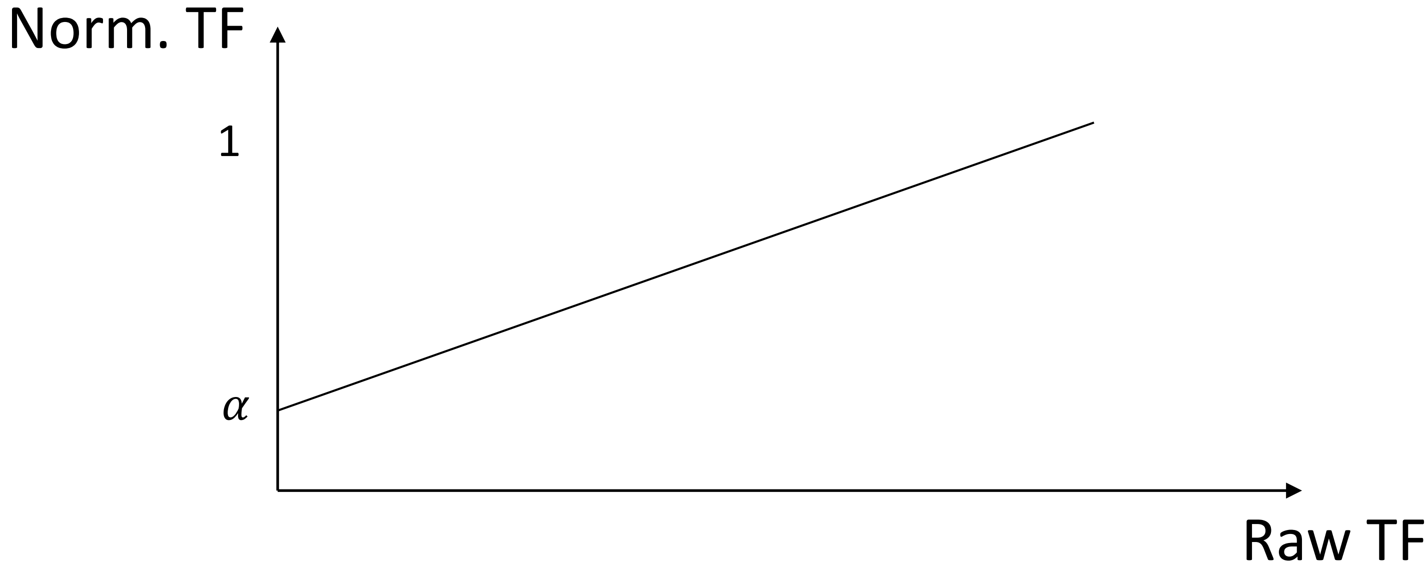
TF normalization

- Sublinear TF scaling
  - $tf(t,d) = \begin{cases} 1 + \log f(t,d), if\ f(t,d) > 0 \\ 0, otherwise \end{cases}$



TF normalization

- Maximum TF scaling
  - $tf(t,d) = \alpha + (1-\alpha)\frac{f(t,d)}{\max_t f(t,d)}$
  - Normalize by the most frequent word in this doc



IDF weighting

- Solution
  - Assign higher weights to the rare terms
  - Formula
    - $IDF(t) = 1 + \log(\frac{N}{df(t)})$

Non-linear scaling

Total number of docs in collection

Number of docs containing term t

  - A corpus-specific property
    - Independent of a single document

TF-IDF weighting

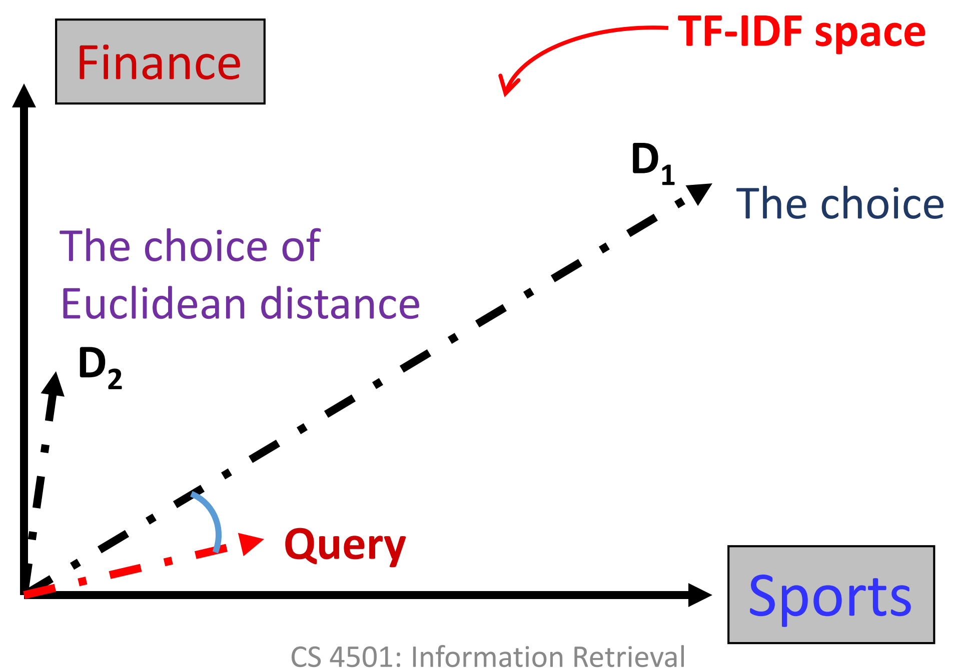
- Combining TF and IDF
  - Common in doc → high tf → high weight
  - Rare in collection→ high idf→ high weight
  - $w(t,d) = TF(t,d) \times IDF(t)$
- Most well-known document representation schema in IR! (G Salton et al. 1983)

"Salton was perhaps the leading computer scientist working in the field of information retrieval during his time." - wikipedia

[Gerard Salton Award](#) – highest achievement award in IR

From distance to angle

- Angle: how vectors are overlapped
  - Cosine similarity – projection of one vector onto another



Advantages of VS Model

- Empirically effective! (Top TREC performance)
- Intuitive
- Easy to implement
- Well-studied/Mostly evaluated
- The Smart system
    - Developed at Cornell: 1960-1999
    - Still widely used
- Warning: Many variants of TF-IDF!

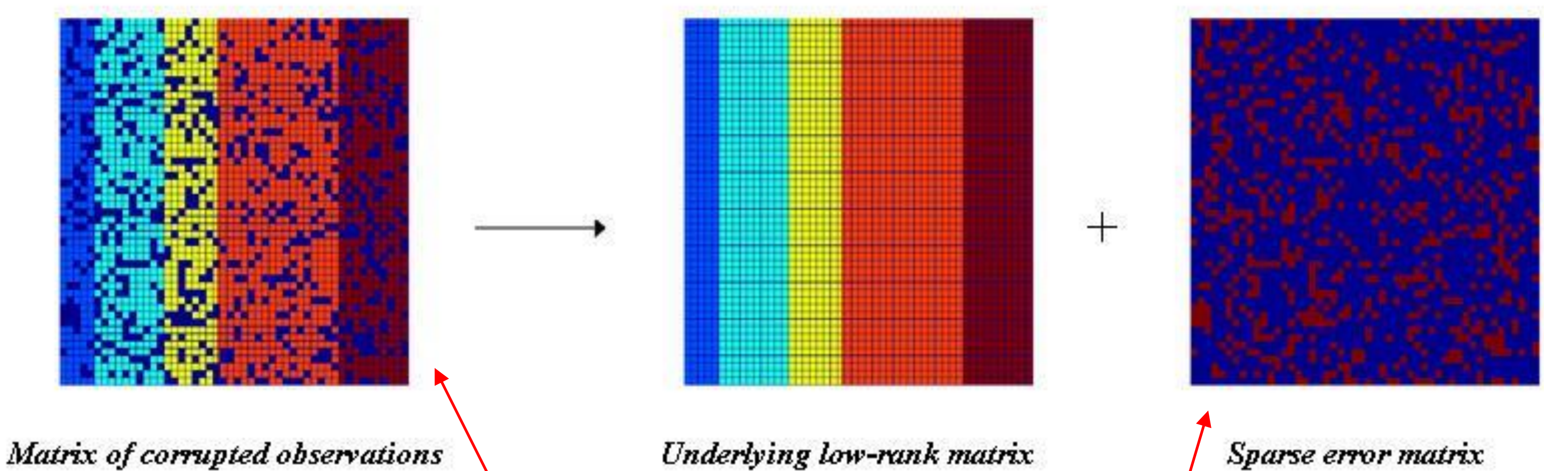
Disadvantages of VS Model

- Assume term independence
- Assume query and document to be the same
- Lack of “predictive adequacy”
    - Arbitrary term weighting
    - Arbitrary similarity measure
- Lots of parameter tuning!

Latent semantic analysis

- Solution
  - Low rank matrix approximation

Imagine this is *true* concept-document matrix



Imagine this is our observed term-document matrix

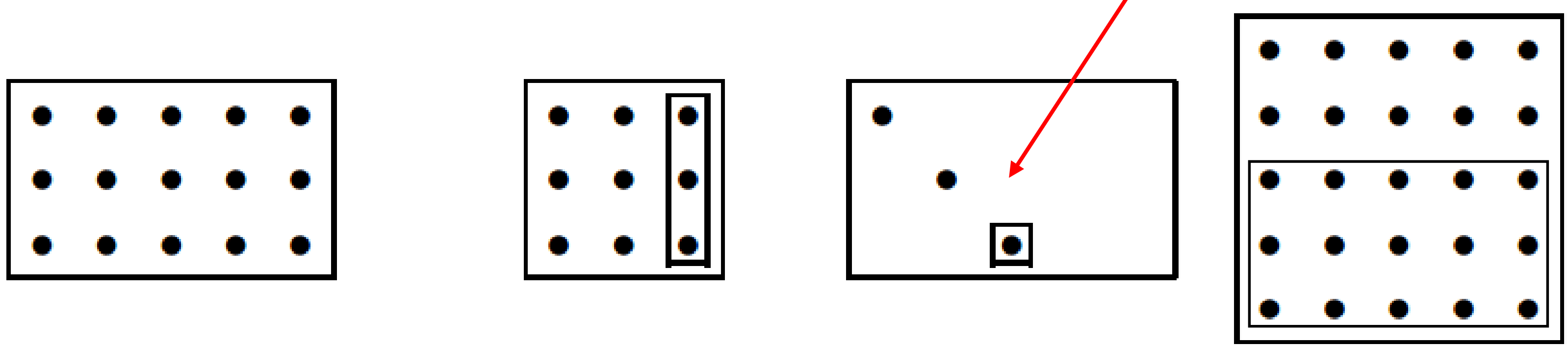
Random noise over the word selection in each document

Latent Semantic Analysis (LSA)

- Solve LSA by SVD

Map to a lower dimensional space

$$C_k \quad = \quad U \quad \Sigma_k \quad V^T$$



1. Perform SVD on document-term adjacency matrix
2. Construct $C_{M\times N}^{k}$ by only keeping the largest k singular values in Σ non-zero

Probabilistic ranking principle

- From decision theory
  - Two types of loss
    - Loss(retrieved|non-relevant)=a_1
    - Loss(not retrieved|relevant)=a_2
  - $\phi(d_i, q)$: probability of d_i being relevant to q
  - Expected loss regarding to the decision of including d_i in the final results
    - Retrieve: $\left(1 - \phi(d_i, q)\right)a_1$
    - Not retrieve: $\phi(d_i, q)a_2$

Your decision criterion?

Probabilistic ranking principle

- From decision theory
  - We make decision by
    - If $\big(1-\phi(d_i,q)\big)a_1 < \phi(d_i,q)a_2$, retrieve d_i
    - Otherwise, not retrieve d_i
  - Check if $\phi(d_i,q) > \frac{a_1}{a_1+a_2}$
  - Rank documents by descending order of $\phi(d_i,q)$ would minimize the loss

Conditional models for $P(R=1|Q,D)$

- Basic idea: relevance depends on how well a query matches a document
  - $P(R=1|Q,D)=g(Rep(Q,D)|\theta)$ ⟵ **a functional form**
    - Rep(Q,D): feature representation of query-doc pair
      - E.g., #matched terms, highest IDF of a matched term, docLen
  - Using training data (with known relevance judgments) to estimate parameter θ
  - Apply the model to rank new documents
- Special case: logistic regression

Generative models for P(R=1|Q,D)

- Basic idea
  - Compute Odd(R=1|Q,D) using Bayes' rule

$$Odd(R=1|Q,D) = \frac{P(R=1|Q,D)}{P(R=0|Q,D)} = \frac{P(Q,D|R=1)}{P(Q,D|R=0)} \boxed{\frac{P(R=1)}{P(R=0)}}$$

Ignored for ranking

- Assumption
  - Relevance is a binary variable
- Variants
  - Document "generation"
    - P(Q,D|R)=P(D|Q,R)P(Q|R)
  - Query "generation"
    - P(Q,D|R)=P(Q|D,R)P(D|R)

Document generation model

$$Odd(R=1|Q,D) \propto \prod_{i=1}^{k} \frac{P(A_i=d_i|Q,R=1)}{P(A_i=d_i|Q,R=0)}$$

Terms occur in doc

Terms do not occur in doc

$$= \boxed{\prod_{i=1,d_i=1}^{k} \frac{P(A_i=1|Q,R=1)}{P(A_i=1|Q,R=0)}} \boxed{\prod_{i=1,d_i=0}^{k} \frac{P(A_i=0|Q,R=1)}{P(A_i=0|Q,R=0)}}$$

Important tricks

$$\approx \prod_{i=1,d_i=q_i=1}^{k} \frac{p_i}{u_i} \prod_{i=1,d_i=0,q_i=1}^{k} \frac{1-p_i}{1-u_i}$$

Assumption: terms not occurring in the query are equally likely to occur in relevant and nonrelevant documents, i.e., $p_t=u_t$

$$= \prod_{i=1,d_i=q_i=1}^{k} \frac{p_i(1-u_i)}{u_i(1-p_i)} \prod_{i=1,q_i=1}^{k} \frac{1-p_i}{1-u_i}$$

document	relevant(R=1)	nonrelevant(R=0)
term present A_i=1	p_i	u_i
term absent A_i=0	1-p_i	1-u_i

Maximum likelihood estimation

- Data: a document d with counts $c(w_1), \ldots, c(w_N)$
- Model: multinomial distribution $p(W|\theta)$ with parameters $\theta_i = p(w_i)$
- Maximum likelihood estimator: $\hat{\theta} = argmax_\theta p(W|\theta)$

$$p(W|\theta) = \binom{N}{c(w_1), \ldots, c(w_N)} \prod_{i=1}^{N} \theta_i^{c(w_i)} \propto \prod_{i=1}^{N} \theta_i^{c(w_i)} \Rightarrow \log p(W|\theta) = \sum_{i=1}^{N} c(w_i) \log \theta_i$$

$$\Rightarrow L(W,\theta) = \sum_{i=1}^{N} c(w_i) \log \theta_i + \lambda \left(\sum_{i=1}^{N} \theta_i - 1 \right)$$

Using Lagrange multiplier approach, we'll tune $\boldsymbol{\theta_i}$ to maximize $\boldsymbol{L(W,\theta)}$

$$\Rightarrow \frac{\partial L}{\partial \theta_i} = \frac{c(w_i)}{\theta_i} + \lambda \quad \rightarrow \quad \theta_i = -\frac{c(w_i)}{\lambda}$$

Set partial derivatives to zero

$\Rightarrow$ Since $\sum_{i=1}^{N} \theta_i = 1$ we have $\lambda = -\sum_{i=1}^{N} c(w_i)$

Requirement from probability

$$\Rightarrow \theta_i = \frac{c(w_i)}{\sum_{i=1}^{N} c(w_i)}$$

ML estimate

The BM25 formula

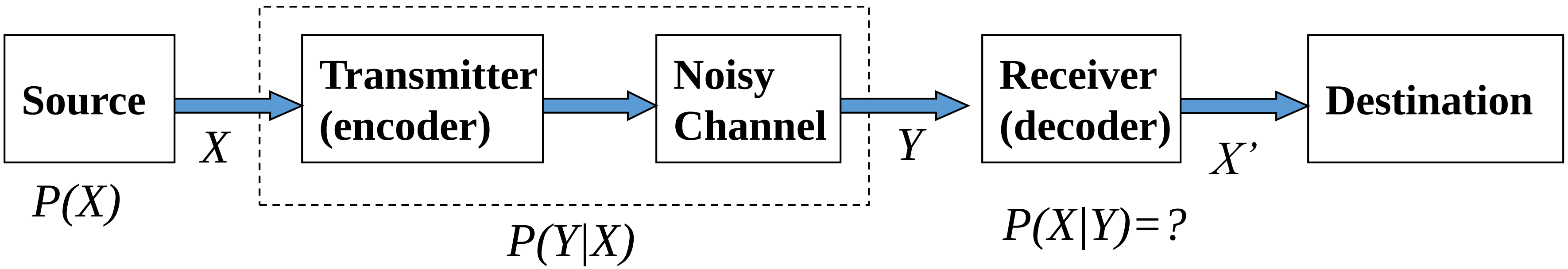
- A closer look

$$rel(q,D)=\sum_{i=1}^{n}\overbrace{IDF(q_i)\frac{tf_i(k_1+1)}{tf_i+k_1(1-b+b\frac{|D|}{avg|D|})}}^{\textit{TF-IDF component for document}}\overbrace{\frac{qtf_i(k_2+1)}{k_2+qtf_i}}^{\textit{TF component for query}}$$

  - b is usually set to [0.75, 1.2]
  - k_1 is usually set to [1.2, 2.0]
  - k_2 is usually set to (0, 1000]

Vector space model with TF-IDF schema!

Source-Channel framework [Shannon 48]



$$\hat{X} = \arg\max_X p(X|Y) = \arg\max_X p(Y|X)p(X) \quad \textbf{(Bayes Rule)}$$

When X is text, p(X) is a language model

Many Examples:

Speech recognition:	**X=Word sequence**	**Y=Speech signal**
Machine translation:	**X=English sentence**	**Y=Chinese sentence**
OCR Error Correction:	**X=Correct word**	**Y= Erroneous word**
Information Retrieval:	**X=Document**	**Y=Query**
Summarization:	**X=Summary**	**Y=Document**

More sophisticated LMs

- N-gram language models
  - In general, $p(w_1\, w_2\ \dots w_n) = p(w_1)p(w_2|w_1) \dots p(w_n|w_1 \dots w_{n-1})$
  - N-gram: conditioned only on the past N-1 words
  - E.g., bigram: $p(w_1\ \dots\ w_n) = p(w_1)p(w_2|w_1)\, p(w_3|w_2)\ \dots p(w_n|w_{n-1})$
- Remote-dependence language models (e.g., Maximum Entropy model)
- Structured language models (e.g., probabilistic context-free grammar)

Justification from PRP

$$O(R=1|Q,D) \propto \frac{P(Q,D|R=1)}{P(Q,D|R=0)}$$

Query generation

$$= \frac{P(Q|D,R=1)P(D|R=1)}{P(Q|D,R=0)P(D|R=0)}$$

$$\propto \underbrace{P(Q|D,R=1)}\underbrace{\frac{P(D|R=1)}{P(D|R=0)}} \quad (Assume\ P(Q|D,R=0) \approx P(Q|R=0))$$

Query likelihood p(q| θ_d) **Document prior**

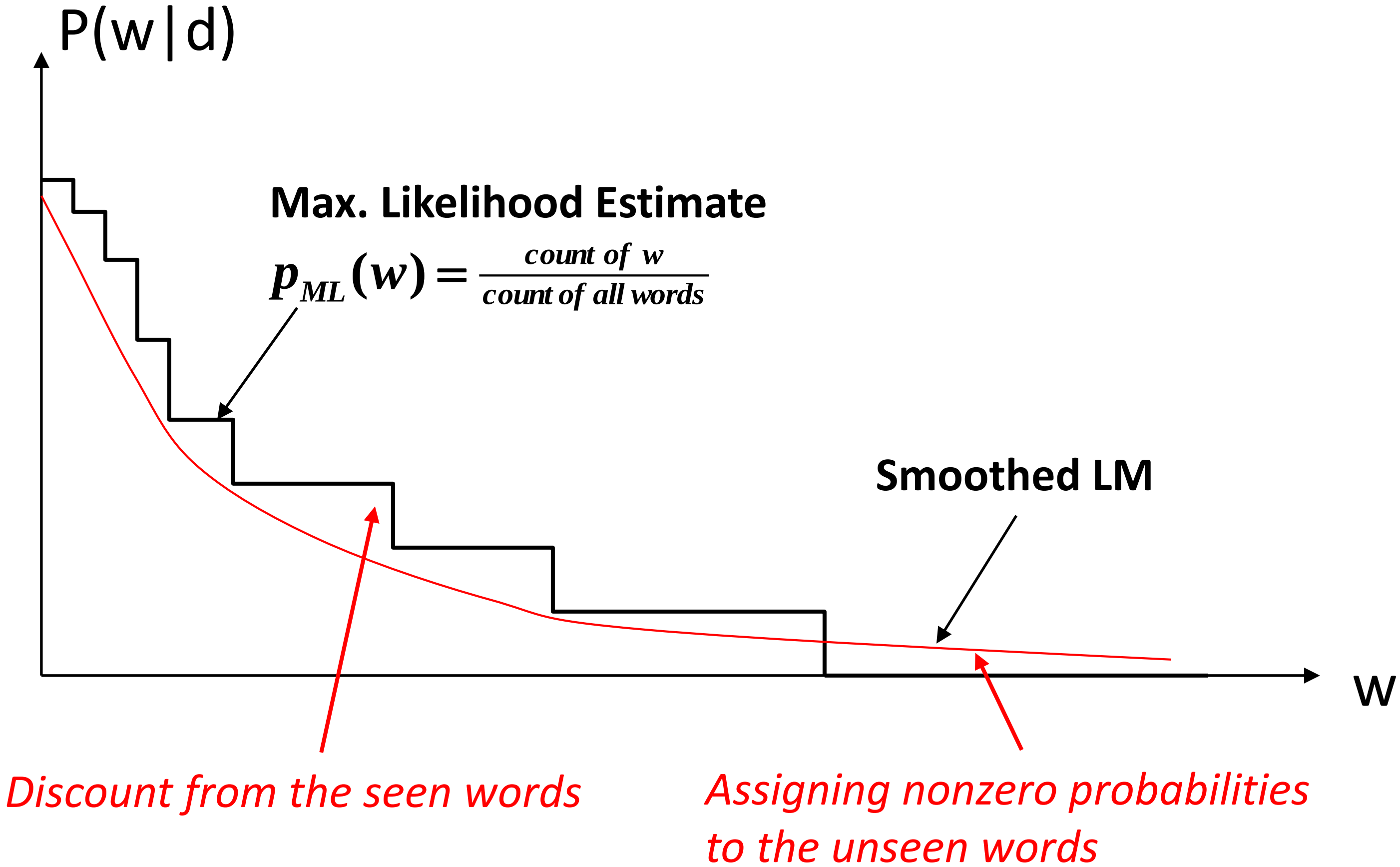
Assuming uniform document prior, we have

$$O(R=1|Q,D) \propto P(Q|D,R=1)$$

Problem with MLE

- What probability should we give a word that has not been observed in the document?
  - log0?
- If we want to assign non-zero probabilities to such words, we'll have to discount the probabilities of observed words
- This is so-called "smoothing"

Illustration of language model smoothing



Refine the idea of smoothing

- Should all unseen words get equal probabilities?
- We can use a reference model to discriminate unseen words

Discounted ML estimate

$$p(w|d) = \begin{cases} p_{seen}(w|d) & if\ w\ is\ seen\ in\ d \\ \alpha_d\, p(w|REF) & otherwise \end{cases}$$

Reference language model

$$\alpha_d = \frac{1 - \sum_{w\ is\ seen} p_{seen}(w|d)}{\sum_{w\ is\ unseen} p(w|REF)}$$

Smoothing methods

- Method 1: Additive smoothing
  - Add a constant δ to the counts of each word

Counts of w in d

$$p(w|d) = \frac{c(w,d)+1}{|d|+|V|}$$

"Add one", Laplace smoothing

Vocabulary size

Length of d (total counts)

  - Problems?
    - Hint: all words are equally important?

Smoothing methods

- Method 2: Absolute discounting
  - Subtract a constant δ from the counts of each word

 # uniq words

$$p(w|d) = \frac{\max(c(w;d)-\delta,0)+\delta|d|_u \, p(w|REF)}{|d|}$$

  - Problems?
    - Hint: varied document length?

Smoothing methods

- Method 3: Linear interpolation, Jelinek-Mercer
  - "Shrink" uniformly toward p(w|REF)

$$p(w|d) = (1-\lambda)\underbrace{\boxed{\frac{c(w,d)}{|d|}}}_{\textbf{MLE}} + \underbrace{\lambda}_{\textbf{parameter}}\, p(w|REF)$$

  - Problems?
    - Hint: what is missing?

Smoothing methods

- Method 4: Dirichlet Prior/Bayesian
  - Assume pseudo counts $\mu p(w|REF)$

$$p(w|d) = \frac{c(w;d) + \mu p(w|REF)}{|d| + \mu} = \frac{|d|}{|d| + \mu} \frac{c(w,d)}{|d|} + \frac{\mu}{|d| + \underset{\text{parameter}}{\mu}} p(w|REF)$$

  - Problems?

parameter

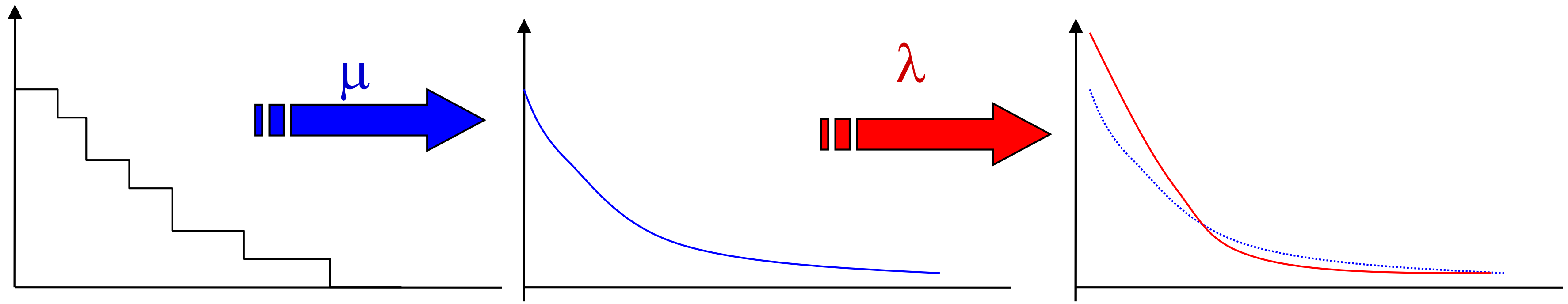
Two-stage smoothing [Zhai & Lafferty 02]

Stage-1

-Explain unseen words
-Dirichlet prior (Bayesian)

Stage-2

-Explain noise in query
-2-component mixture



$$P(w|d) = (1-\lambda)\frac{c(w,d) + \mu p(w|C)}{|d| + \mu} + \lambda p(w|U)$$

Collection LM

User background model

Understanding smoothing

Topical words

	Query = "the	algorithms	for	data	mining"	
p_{ML}(w|d1):	*0.04*	*0.001*	*0.02*	*0.002*	*0.003*	4.8×10^{-12}
p_{ML}(w|d2):	*0.02*	*0.001*	*0.01*	*0.003*	*0.004*	2.4×10^{-12}

p("algorithms"|d1) = p("algorithms"|d2)
p("data"|d1) < p("data"|d2)
p("mining"|d1) < p("mining"|d2)

Intuitively, d2 should have a higher score, but p(q|d1)>p(q|d2)…

So we should make p("the") and p("for") less different for all docs, and smoothing helps to achieve this goal…

2.35×10^{-13}

4.53×10^{-13}

After smoothing with $p(w|d) = 0.1 p_{DML}(w|d) + 0.9 p(w|REF)$, $p(q|d1) < p(q|d2)$!

Query	= "the	algorithms	for	data	mining"
P(w|REF)	0.2	0.00001	0.2	0.00001	0.00001
Smoothed p(w|d1):	0.184	0.000109	0.182	0.000209	0.000309
Smoothed p(w|d2):	0.182	0.000109	0.181	0.000309	0.000409

Smoothing & TF-IDF weighting

Smoothed ML estimate

Retrieval formula using the general smoothing scheme

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

$$\alpha_d = \frac{1 - \sum_{w \text{ is seen}} p_{Seen}(w|d)}{\sum_{w \text{ is unseen}} p(w|C)}$$

Reference language model

$$\log p(q|d) = \sum_{w \in V, c(w,q)>0} c(w,q) \log p(w|d)$$

$$= \sum_{\substack{w \in V, c(w,d)>0, \\ c(w,q)>0}} c(w,q) \log p_{Seen}(w|d) + \boxed{\sum_{w \in V, c(w,q)>0, c(w,d)=0} c(w,q) \log \alpha_d p(w|C)}$$

$$= \sum_{\substack{w \in V, c(w,d)>0 \\ c(w,q)>0}} c(w,q) \log p_{Seen}(w|d) + \boxed{\sum_{w \in V, c(w,q)>0} c(w,q) \log \alpha_d p(w|C) - \sum_{w \in V, c(w,q)>0, c(w,d)>0} c(w,q) \log \alpha_d p(w|C)}$$

$$= \sum_{\substack{w \in V, c(w,d)>0 \\ c(w,q)>0}} c(w,q) \log \frac{p_{Seen}(w|d)}{\alpha_d p(w|C)} + |q| \log \alpha_d + \cancel{\sum_{w \in V, c(w,q)>0} c(w,q) p(w|C)}$$

Key rewriting step (where did we see it before?)

Similar rewritings are very common when using probabilistic models for IR…

Retrieval evaluation

- Aforementioned evaluation criteria are all good, but not essential
  - Goal of any IR system
    - Satisfying users' <u>information need</u>
  - Core <u>quality</u> measure criterion
    - *"how well a system meets the information needs of its users." – wiki*
    - Unfortunately vague and hard to execute

Classical IR evaluation

- Three key elements for IR evaluation
    1. A document collection
    2. A test suite of information needs, expressible as queries
    3. A set of relevance judgments, e.g., binary assessment of either *relevant* or *nonrelevant* for each query-document pair

Evaluation of unranked retrieval sets

- Summarizing precision and recall to a single value
  - In order to compare different systems
  - F-measure: weighted harmonic mean of precision and recall, α balances the trade-off

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} \qquad \left(F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}\right)$$

Equal weight between precision and recall

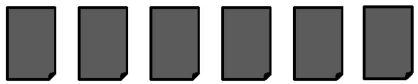
  - Why harmonic mean?
    - System1: P:0.53, R:0.36
    - System2: P:0.01, R:0.99

H	A
0.429	0.445
0.019	0.500

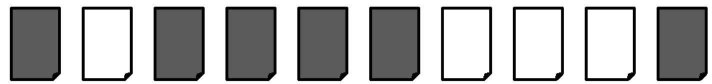
Evaluation of ranked retrieval results

- Summarize the ranking performance with a single number
  - Binary relevance
    - Eleven-point interpolated average precision
    - Precision@K (P@K)
    - Mean Average Precision (MAP)
    - Mean Reciprocal Rank (MRR)
  - Multiple grades of relevance
    - Normalized Discounted Cumulative Gain (NDCG)

AvgPrec is about one query

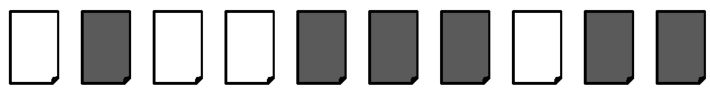
 = the relevant documents

Ranking #1



Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6

Ranking #2



Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

Figure from Manning Stanford CS276, Lecture 8

AvgPrec of the two rankings

$$\text{Ranking \#1:}\ (1.0 + 0.67 + 0.75 + 0.8 + 0.83 + 0.6)/6 = 0.78$$

$$\text{Ranking \#2:}\ (0.5 + 0.4 + 0.5 + 0.57 + 0.56 + 0.6)/6 = 0.52$$

What does query averaging hide?

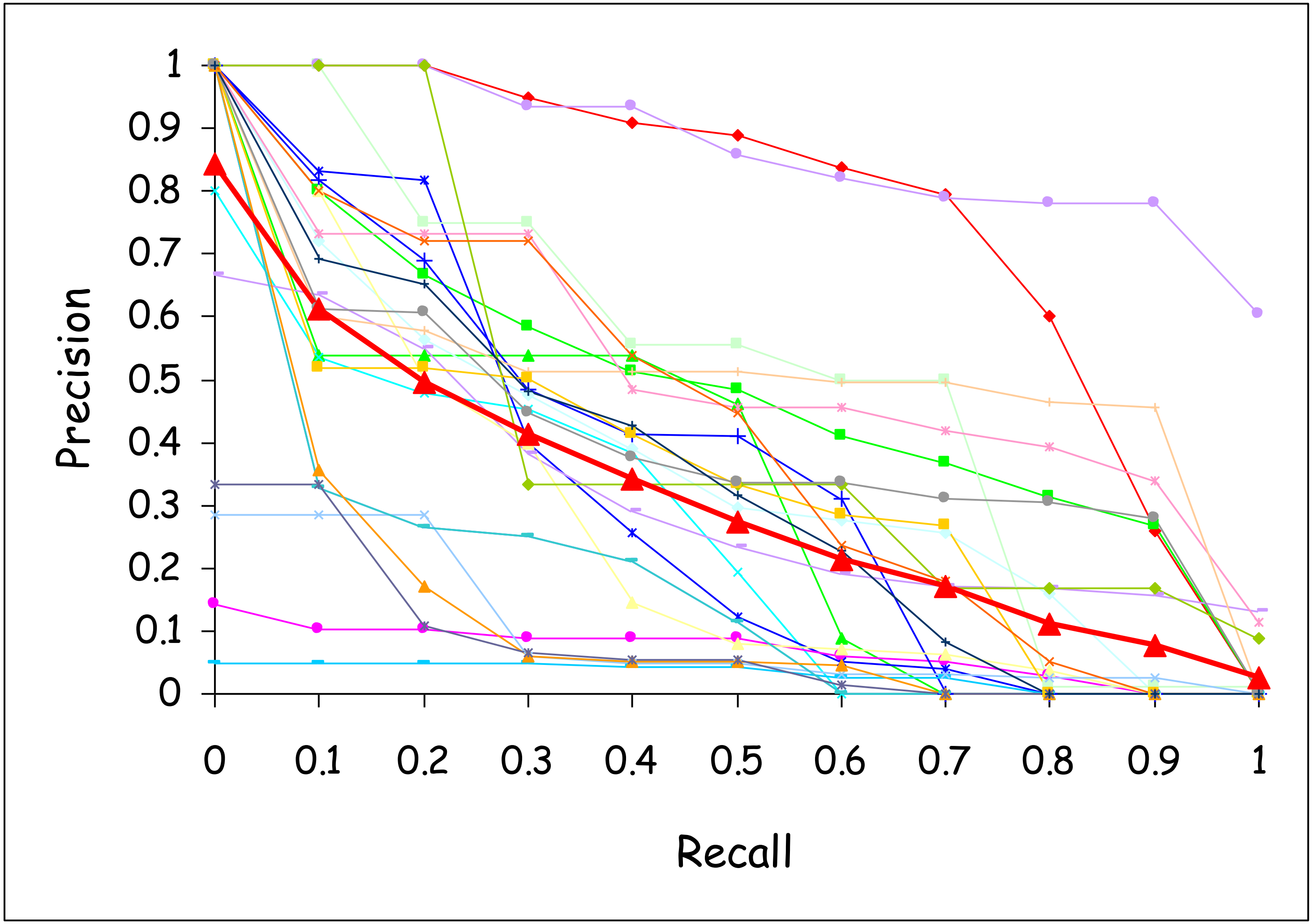


Figure from Doug Oard's presentation, originally from Ellen Voorhees' presentation

Measuring assessor consistency

- *kappa* statistic
  - A measure of agreement between judges

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

    - $P(A)$ is the proportion of the times judges agreed
    - $P(E)$ is the proportion of times they would be expected to agree by chance
  - $\kappa = 1$ if two judges always agree
  - $\kappa = 0$ if two judges agree by chance
  - $\kappa < 0$ if two judges always disagree

What we have not considered

- The physical form of the output
  - User interface
- The effort, intellectual or physical, demanded of the user
  - User effort when using the system
- Bias IR research towards optimizing relevance-centric metrics