Welcome to the class of

Advanced Topics in Information Retrieval !

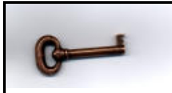
Min ZHANG （张敏）

z-m@tsinghua.edu.cn

Tea Time

Towards **Fairness** in Machine Learning

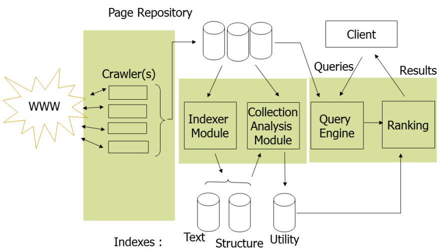
Chenyang Wang



Brief introduction to IR fundamentals – III

(content-based) Ranking

Outline



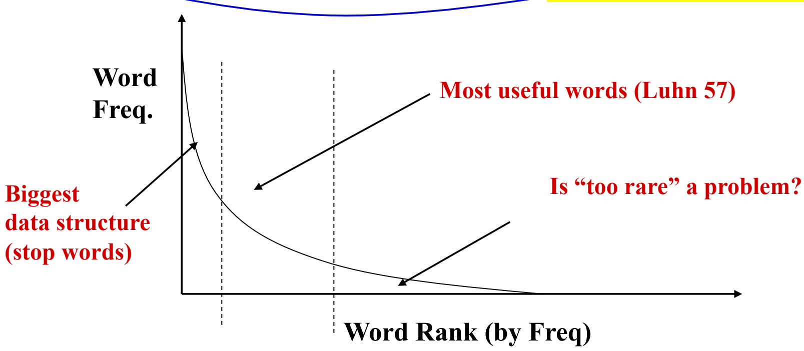
- What is IR?
- Basic IR procedure
  - Data acquisition
  - Indexing
  - Ranking
    - Term weighting
    - Ranking models
  - System evaluation

Term Weighting

- Not all terms are equally important
- How to select *important / good* keywords?
  - Simplest way: using middle-frequency words according to Zipf's law

Zipf's Law

- rank * frequency ≈ constant $F(w) = \frac{C}{r(w)^{\alpha}}$ $\alpha \approx 1, C \approx 0.1$



Generalized Zipf's law: $F(w) = \frac{C}{[r(w)+B]^{\alpha}}$ **Applicable in many domains**

Term weighting

- **TFIDF**

  tf = term frequency

  (the frequency of a term/keyword in a document)

  idf = inverse document frequency

  (the unevenness of term distribution in the corpus)

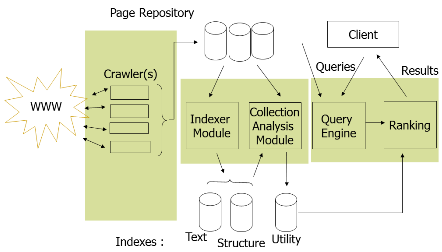
  ***weight* (*t*, *D*) = *tf* (*t*,*D*) * *idf* (*t*)**

- Some commonly used tf*idf schemes:

$tf(t, D) = freq(t, D)$	$idf(t) = \log(N/n + \alpha)$
$tf(t, D) = \log[freq(t, D)]$	n = # of docs containing t
$tf(t, D) = \log[freq(t, D) + 1]$	N = # of docs in the corpus
$tf(t, D) = freq(t, D) / \mathrm{Max}[f(t, D)]$	α = 1, 0.5, etc

- Sometimes, additional normalizations (e.g. *length*).

Outline



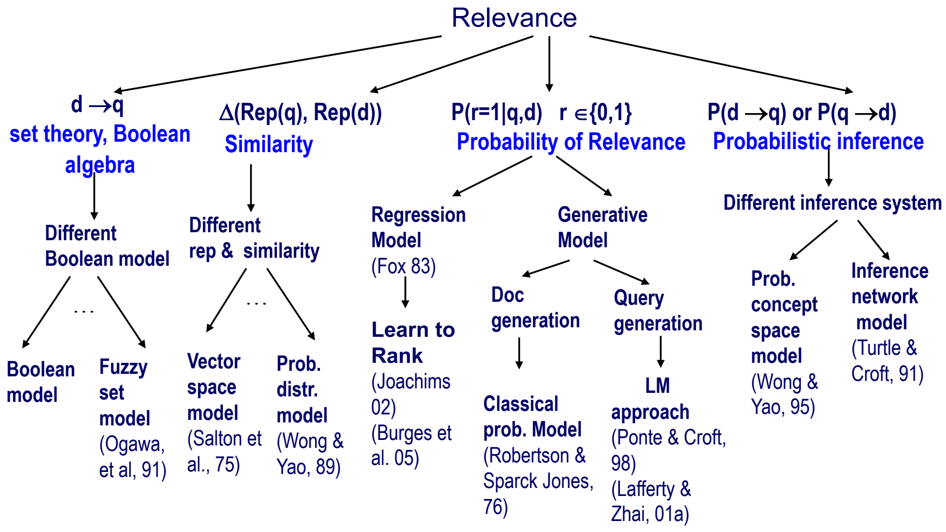
- What is IR?
- Basic IR procedure
  - Data acquisition
  - Indexing
  - Ranking
    - Term weighting -- TFIDF
    - Ranking models
  - System evaluation

Ranking

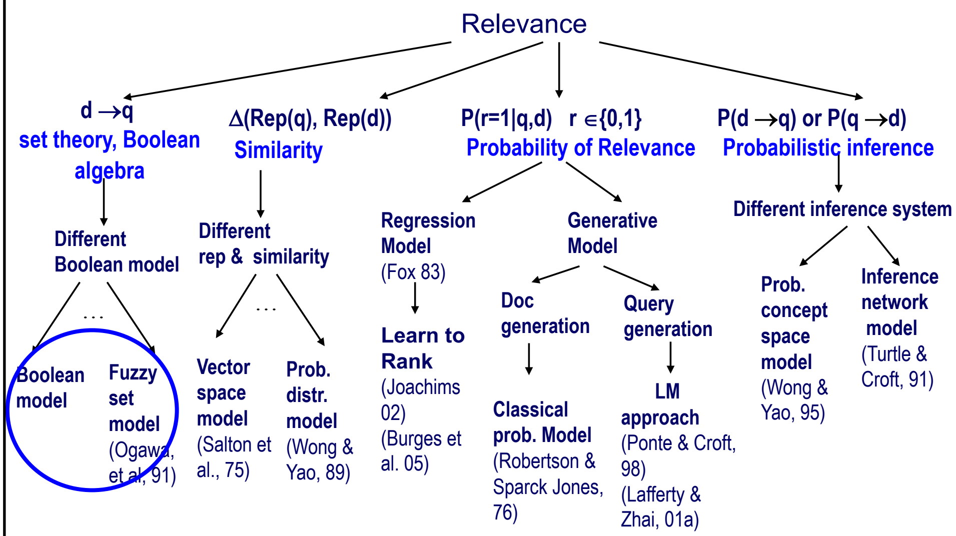
- The problems underlying ranking
  - How is a document/query represented with the selected keywords?
  - How are two representations compared?

  - (ordered) measure of relevance

Overview of Retrieval Models



Overview of Retrieval Models



Boolean model

- Boolean model
  - Document = Logical conjunction of keywords
  - Query = Boolean expression of keywords
    (AND, OR, NOT, with brackets)
  - R(D, Q) = D →Q
- Example:
  $D = t_1 \wedge t_2 \wedge \ldots \wedge t_n \qquad Q = (t_1 \wedge t_2) \vee (t_3 \wedge \neg t_4)$
  We have D →Q. Thus R(D, Q) = 1.
- Popular/earliest retrieval model because:
  - Easy to understand for simple queries.
  - Clean formalism.
  - Reasonably efficient implementations possible for normal queries.

Boolean Models – Problems

- Very rigid: AND means all; OR means any.
- Difficult to express complex user requests.
- Difficult to control the number of documents retrieved.
  - ***All*** matched documents will be returned.
- Difficult to rank output.
  - ***All*** matched documents logically satisfy the query.
- Difficult to perform relevance feedback.
  - If a document is identified by the user as relevant or irrelevant, how should the query be modified?

Extensions to Boolean model

- $D = \{\ldots, (t_i, a_i), \ldots\}$ i.e. keywords are weighted
- Interpretation:

  D is a member of class t_i to degree a_i.

  In terms of fuzzy sets:

  $\mu_{ti}(D) = a_i$
- Evaluation:

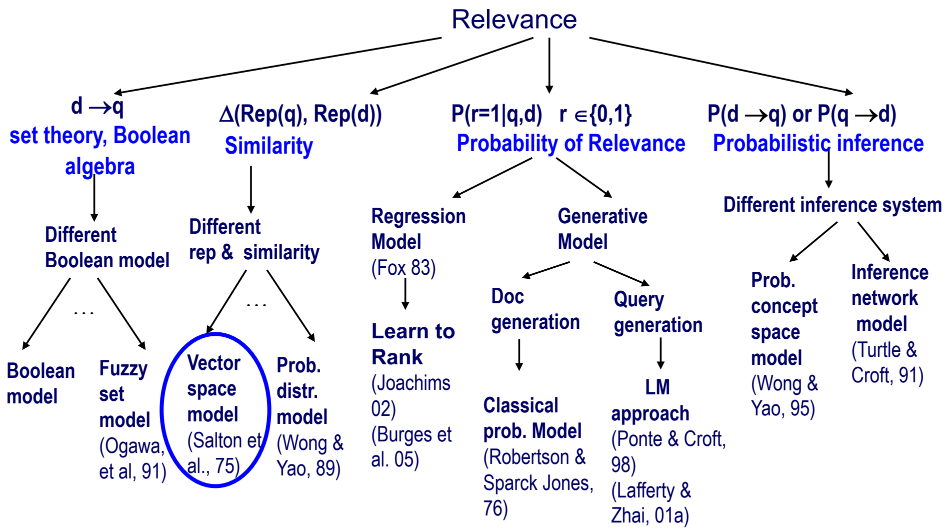
  $R(D, t_i) = \mu_{ti}(D)$

  $R(D, Q_1 \wedge Q_2) = \min(R(D, Q_1), R(D, Q_2))$.

  $R(D, Q_1 \vee Q_2) = \max(R(D, Q_1), R(D, Q_2))$.

  $R(D, \neg Q_1) = 1 - R(D, Q_1)$.

Overview of Retrieval Models



Vector space model

- Vector space = all the keywords encountered

  $<t_1, \ t_2, \ t_3, ..., t_n>$

  Dimension = n = |vocabulary|
- Document: a weighted vector

  $D =< a_1, a_2, a_3, ..., a_n>$

  a_i = weight of t_i in D
- Query: a weighted vector

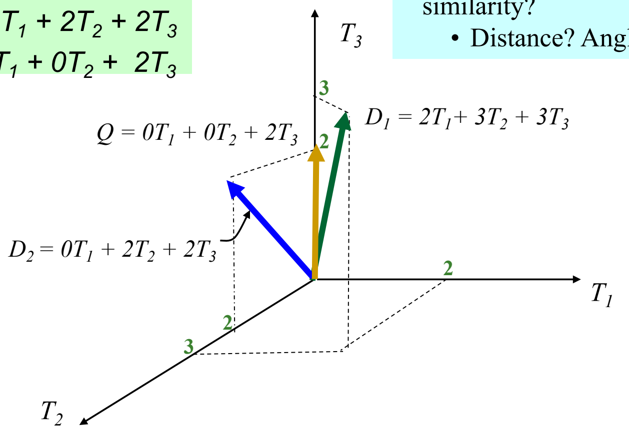
  $Q =< b_1, b_2, b_3, ..., b_n>$

  b_i = weight of t_i in Q
- R(D,Q) = Similarity(D,Q)

Graphic Representation

Example:
$D_1 = 2T_1 + 3T_2 + 3T_3$
$D_2 = 0T_1 + 2T_2 + 2T_3$
$Q = 0T_1 + 0T_2 + 2T_3$

- Is D_1 or D_2 more similar to Q?
- How to measure the degree of similarity?
  - Distance? Angle? Projection?



Similarity Measure (1) - Inner Product

- Using inner product:

$$\text{sim}(\boldsymbol{d}_j, \boldsymbol{q}) = \boldsymbol{d}_j \bullet \boldsymbol{q} = \sum_{i=1}^{t} w_{ij} \cdot w_{iq}$$

w_{ij} -- the weight of term *i* in document *j*

w_{iq} -- the weight of term *i* in the query

- For binary vectors, it's # of matched query terms in the document (size of intersection).
- For weighted term vectors, it's the sum of the products of the weights of the matched terms.

$Q = 0T_1 + 0T_2 + 2T_3$ $D_1 = 2T_1 + 3T_2 + 3T_3$ $D_2 = 0T_1 + 2T_2 + 2T_3$

$\text{Sim}(D_1, Q) = 6$
$\text{Sim}(D_2, Q) = 4$

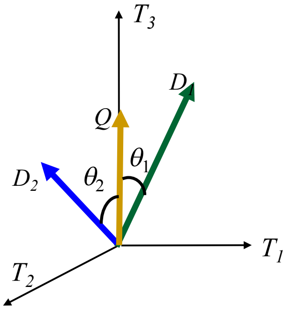
Properties of Inner Product

- The inner product is unbounded.
- Favors long documents with a large number of unique terms.
- Measures how many terms matched but not how many terms are *not* matched.

Similarity Measure (2) -- Cosine Similarity

- Cosine similarity measures the cosine of the angle between two vectors.
- It is inner product normalized by the vector lengths.

$$\text{CosSim}(\boldsymbol{d}_j, \boldsymbol{q}) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2 \cdot \sum_{i=1}^{t} w_{iq}^2}}$$



$D_1 = 2T_1 + 3T_2 + 3T_3$ CosSim(D_1 , Q) = 6 / $\sqrt{(4+9+9)(0+0+4)}$ = 0.64
$D_2 = 0T_1 + 2T_2 + 2T_3$ CosSim(D_2 , Q) = 4 / $\sqrt{(0+4+4)(0+0+4)}$ = 0.71
$Q = 0T_1 + 0T_2 + 2T_3$

D_2 is better than D_1 using cosine similarity but D_1 is better using inner product.

Typical VSM weighting formula

$$\sum_{t \in Q,D} \frac{1 + \ln(1 + \ln(tf))}{(1 - s) + s\frac{dl}{avdl}} \times qtf \times \ln \frac{N+1}{df}$$

where s is an empirical parameter (usually 0.20), and

tf	is the term's frequency in document
qtf	is the term's frequency in query
N	is the total number of documents in the collection
df	is the number of documents that contain the term
dl	is the document length, and
$avdl$	is the average document length.

Comments on Vector Space Models

- Simple, mathematics-based approach.
- Provides partial matching and ranked results.
- Tends to work quite well in practice despite obvious weaknesses.
- Allows efficient implementation for large document collections.

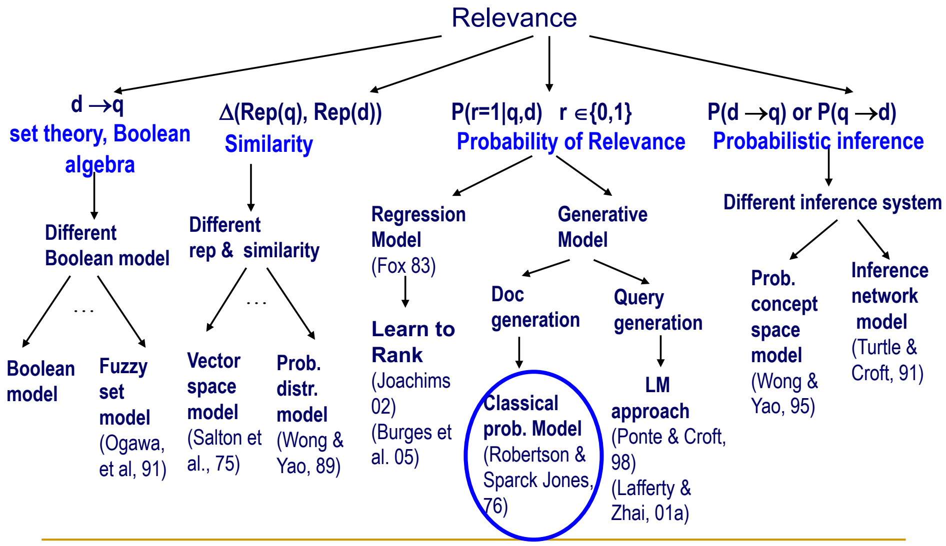
Problems with Vector Space Model

- Missing semantic information (e.g. word sense).
- Missing syntactic information (e.g. phrase structure, word order, proximity information).
- Assumption of term independence.

(*the flaws of not only VSM*, *but All Bag-of-Word models*)

- Lacks the control of a Boolean model (e.g., *requiring* a term to appear in a document).
  - Given a two-term query "A B", may prefer a document containing A frequently but not B, over a document that contains both A and B, but both less frequently.

Overview of Retrieval Models



Probabilistic model

The Basic Question

- What is the probability that ***THIS*** document is relevant to ***THIS*** query?

Formally…

- 3 random variables:
  - query Q, document D, relevance $R \in \{0,1\}$
- Given a particular query q, a particular document d, $p(R=1|Q=q, D=d)$=?
- In fact, we only need to ***compare*** $P(R=1|Q, D_1)$ with $P(R=1|Q, D_2)$, i.e., rank documents

Probabilistic model: Generative models

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

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

Ignored for ranking D

Probabilistic model: Generative models

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

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

Ignored for ranking D

- Special cases
  - 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

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

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

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

Model of relevant docs for Q

Model of non-relevant docs for Q

Assume independent attributes $A_1 \ldots A_k$ (generally we take terms as attributes)
Let $D = d_1 \ldots d_k$, where $d_k \in \{0,1\}$ is the value of attribute A_k (Similarly $Q = t_1 \ldots t_k$)

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

$$\underline{\underline{rank}}\prod_{i=1}^{k}\frac{P(A_i=1|Q,R=1)}{1-P(A_i=1|Q,R=1)}\cdot\frac{1-P(A_i=1|Q,R=0)}{P(A_i=1|Q,R=0)}$$

odds transformation (strictly monotonic) $\frac{P}{1-P}$

$$\propto\sum_{i=1,t_i=1}^{k}\log\frac{p_i(1-q_i)}{(1-p_i)q_i}$$

Robertson-Sparck Jones Model

Note: Non-query terms (t_i=0) are equally likely to appear in relevant and non-relevant docs

$p_i = P(A_i = 1|Q, R = 1)$: prob. that term A_i occurs in a relevant doc
$q_i = P(A_i = 1|Q, R = 0)$: prob. that term A_i occurs in a non-relevant doc

Robertson-Sparck Jones Model

(Robertson & Sparck Jones 76)

$$\log O(R=1\mid Q,D) \overset{Rank}{\approx} \sum_{i=1,t_i=1}^{k} \log \frac{p_i(1-q_i)}{q_i(1-p_i)} \qquad \text{(RSJ model)}$$

Two parameters for each term A_i:

$p_i = P(A_i = 1|Q, R = 1)$: prob. that term A_i occurs in a relevant doc

$q_i = P(A_i = 1|Q, R = 0)$: prob. that term A_i occurs in a non-relevant doc

How to estimate parameters (probabilities)?

I. Suppose we have relevance judgments,

$$\hat{p}_i = \frac{\#(rel.\ doc\ with\ A_i)+0.5}{\#(rel.doc)+1} \qquad \hat{q}_i = \frac{\#(nonrel.\ doc\ with\ A_i)+0.5}{\#(nonrel.doc)+1}$$

"+0.5" and "+1" can be justified by Bayesian estimation

RSJ Model: No Relevance Info

(Croft & Harper 79)

$$\log O(R=1\mid Q,D) \overset{Rank}{\approx} \sum_{i=1,t_i=1}^{k} \log \frac{p_i(1-q_i)}{q_i(1-p_i)} \qquad \text{(RSJ model)}$$

How to estimate parameters (probabilities)?

II. Suppose we do not have relevance judgments,

- We will assume p_i to be a constant

$$\log O(R=1\mid Q,D) \overset{Rank}{\approx} \sum_{i=1,t_i=1}^{k} \log \frac{(1-q_i)}{q_i}$$

- Estimate q_i by assuming all documents to be non-relevant

$$q_i = \frac{n+0.5}{N+1}$$

N: # documents in collection

n_i: # documents in which term A_i occurs

$$\log O(R=1\mid Q,D) \overset{Rank}{\approx} \sum_{i=1,t_i=1}^{k} \log \frac{N-n_i+0.5}{n_i+0.5} \qquad IDF' = \log \frac{N-n_i}{n_i}$$

RSJ Model: Summary

- The most important classic prob. IR model
- Use only term presence/absence, thus also referred to as Binary Independence Model
- Essentially Naïve Bayes for doc ranking
- Most natural for relevance/pseudo feedback
- When without relevance judgments, the model parameters must be estimated in an ad hoc way
- Performance isn't as good as tuned VS model

↓

- Many improvements …

Improvements -- BM25/Okapi Approximation (Robertson et al. 94)

- Idea: Approximate $p(R=1|Q,D)$ with a simpler function that share similar properties
- Observations:

$$\log O(R=1 \mid Q,D) \overset{Rank}{\approx} \sum_{i=1,t_i=1}^{k} \log \frac{p_i(1-q_i)}{q_i(1-p_i)}$$

  - $log\ O(R=1|Q,D)$ is a sum of term weights W_i
- Adding TF (d_i is no longer of binary value 0/1)
  - $W_i = 0,\ \ if\ TF_i = 0$
  - W_i increases monotonically with TF_i
  - W_i has an asymptotic limit

$$W_i = \frac{TF_i(k_1+1)}{k_1+TF_i} \log \frac{p_i(1-q_i)}{q_i(1-p_i)}$$

- Adding document length
  - "Carefully" penalize long doc
- Adding query TF

The most famous ranking function in the doc generation branch – BM25 series

- Final: BM25 – achieving top performances in TREC

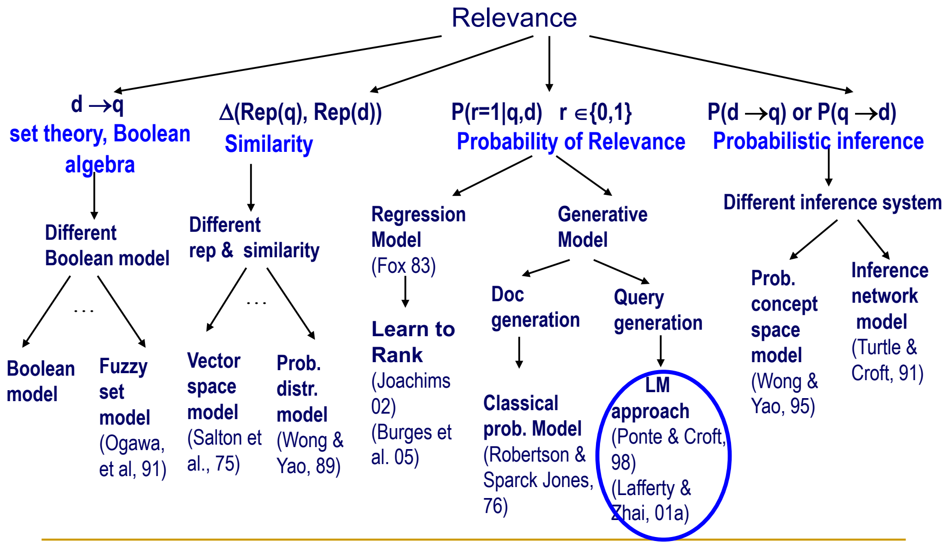
$$\sum_{T\in Q} w^{(1)} \frac{(k_1+1)tf}{K+tf}\,\frac{(k_3+1)qtf}{k_3+qtf} \qquad w^{(1)} = log\frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)}$$

$$K = k_1((1-b)+b\frac{|d|}{avdl})$$

		Are the documents relevant to the term?		
		1 = Yes (Relevant)	0 = No (Non-Relevant)	Collection-wide Incidence
Is the term present	1 = Yes (Present)	r	n - r	n
in the documents?	0 = No (Absent)	R - r	N – n – R + r	N - n
Total number of documents		R	N - R	N

- *tf* : the count of word *T* in the document *d*, *qtf* : the count of word *T* in the query *q*,
- |*d*|: the length of document *d*, *avdl* : the average document length of the collection,
- k_1(1.0 to 2.0),*b* (usually 0.75) and k_3 (0 to 1000) : constants.

Overview of Retrieval Models



Review: Probabilistic model: Generative models

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

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

Ignored for ranking D

- Special cases
  - 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)

Query Generation Model

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

Query Generation Model

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

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

$$\propto \underbrace{P(Q|D,R=1)}_{\text{Query likelihood } p(q|\theta_d)} \underbrace{\frac{P(D|R=1)}{P(D|R=0)}}_{\text{Document prior}}$$

Various ways to represent document prior, e.g. PageRank, … Will be introduced in later lectures.

Assuming uniform document prior, we have $\boldsymbol{O(R=1|Q,D) \propto P(Q|D,R=1)}$

Now, the question is how to compute $\boldsymbol{P(Q|D,R=1)}$ **?**

Generally involves two steps:

(1) Estimate a language model based on D

(2) Compute the query likelihood according to the estimated model

Leading to the so-called "Language Modeling Approach" …

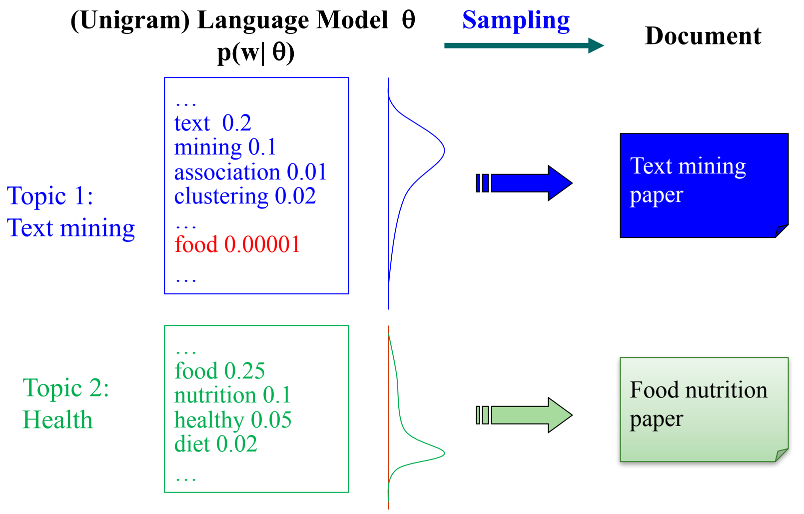
What is a Statistical LM?

- A probability distribution over word sequences
  - p("*Today is Thursday*") ≈ 0.001
  - p("*Today Thursday is*") ≈ 0.0000000000001
  - p("*The eigenvalue is positive*") ≈ 0.00001
- Context-dependent!
- Can also be regarded as a probabilistic mechanism for "generating" text, thus also called a "generative" model

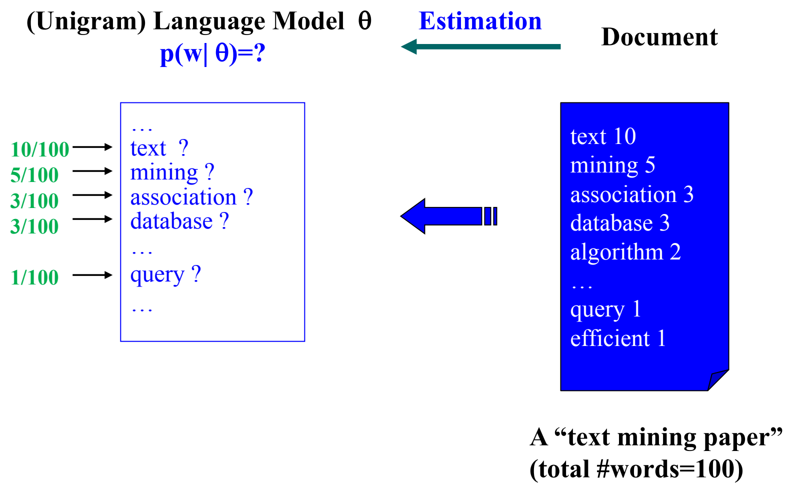
The Simplest Language Model
(Unigram Model)

- Generate a piece of text by generating each word independently
- Thus, $p(w_1\ w_2\ ...\ w_N) = p(w_1)p(w_2)\ldots p(w_N)$
- Parameters: $\{p(w_i)\}$ $p(w_1)+\ldots+p(w_N)=1$
  - (N is voc. size)
- Essentially a multinomial distribution over words
- A piece of text can be regarded as a sample drawn according to this word distribution

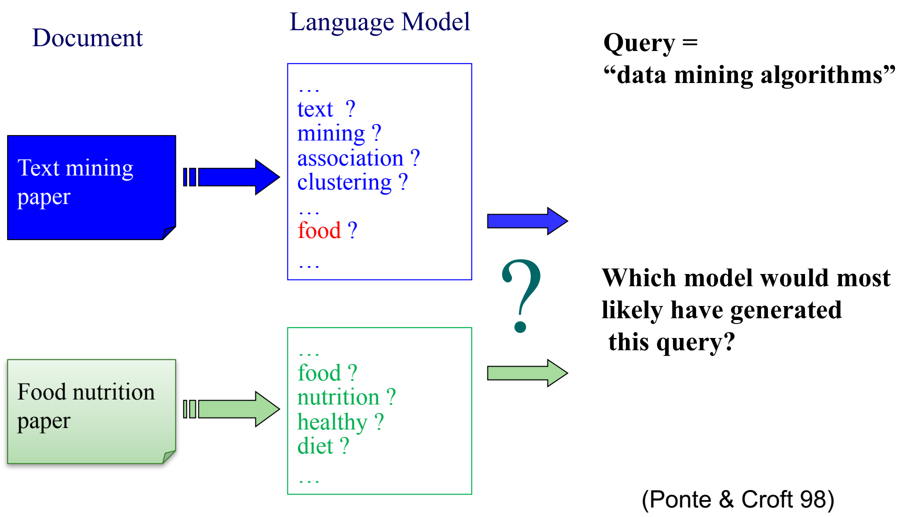
Text Generation with Unigram LM



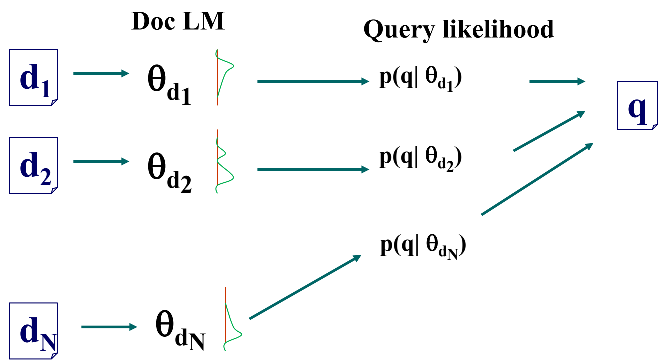
Estimation of Unigram LM



Language Models for Retrieval



Ranking Docs by Query Likelihood



- Document ranking based on *query likelihood*

$$\log p(q\,|\,d) = \sum_i \log \boxed{p(w_i\,|\,d)} \quad \textit{where, } q = w_1 w_2 \ldots w_n$$

Document language model

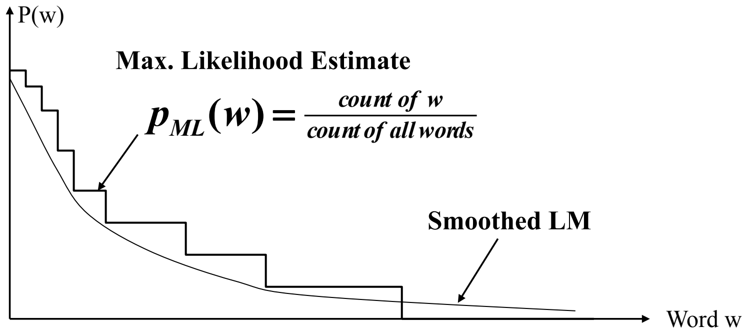
- Retrieval problem $\approx$ Estimation of $p(w_i|d)$

How to Estimate p(w|d)?

- Simplest solution: Maximum Likelihood Estimator
  - $P(w|d)$ = relative frequency of word w in d
  - What if a word doesn't appear in the text? $P(w|d)$ =0 ? ?
- In general, what probability should we give a word that has not been observed?

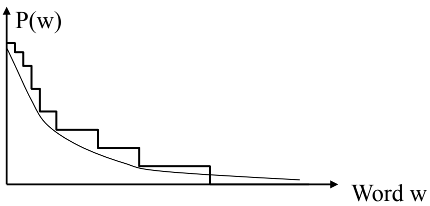
→ "smoothing"

Language Model Smoothing (Illustration)



Language Model Smoothing (Illustration)

- All smoothing methods try to
  - Discount the probability of words seen in a document
  - Re-allocate the extra counts so that unseen words will have a non-zero count



- Many ways for smoothing
  - Add One smoothing (Laplace smoothing): all unseen words take the same count of 1
  - Use a reference corpus
  - ……

For more information, ref. to: Lafferty, J. and Zhai, C., A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval, 2003

Smoothing & TF-IDF Weighting

- A general smoothing schema: using a reference model

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

Discounted Max. LL. estimate

Collection language model

Derivation of the Query Likelihood Retrieval Formula

Discounted Max. LL. estimate

Retrieval formula using the general smoothing scheme

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

Reference language model

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

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

c (w, q) count of word in the query

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

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

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

Key rewriting step

Similar rewritings are very common when using LMs for IR…

Smoothing & TF-IDF Weighting

- A general smoothing schema: using a reference model

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

Discounted ML estimate

Collection language model

- Plug in the general smoothing scheme to the query likelihood retrieval formula, we obtain

TF weighting

Doc length normalization (long doc is expected to have a smaller α_d)

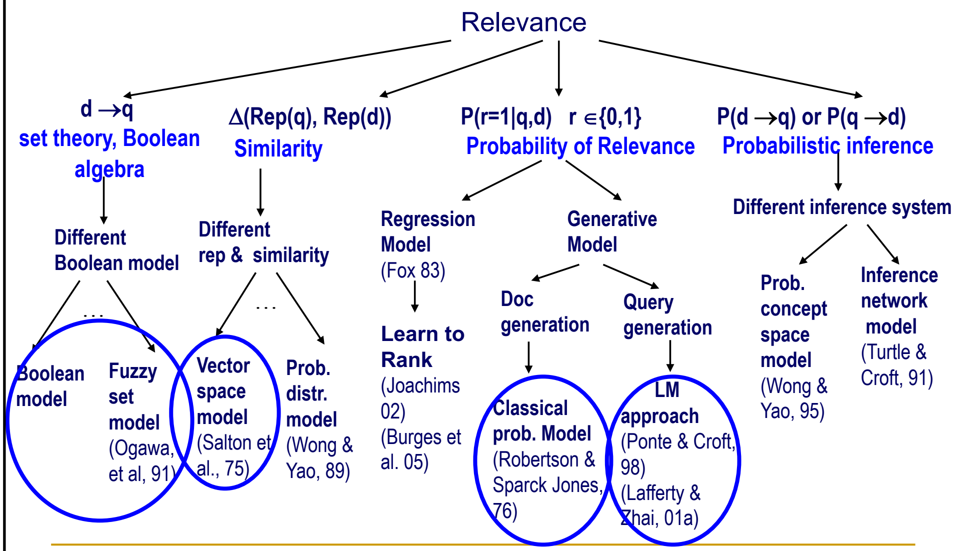
$$\log p(q|d)=\sum_{\substack{w_i \in d \\ w_i \in q}} [\log \frac{p_{seen}(w_i|d)}{\alpha_d\, p(w_i|C)}] + n\log\alpha_d + \boxed{\sum_i \log p(w_i|C)}$$

IDF weighting

Ignore for ranking

- Smoothing with *p(w|C)* ≈ TF-IDF + length norm.

Review: Retrieval Models



Toolkits

- SMART   Developed at Cornell University in 1960s

http://en.wikipedia.org/wiki/SMART_Information_Retrieval_System

Software and test collections: ftp://ftp.cs.cornell.edu/pub/smart/

- Lucene
  - http://lucene.apache.org/
- Indri (Lemur Project) by Umass, Amherst, CMU
  - http://www.lemurproject.org/