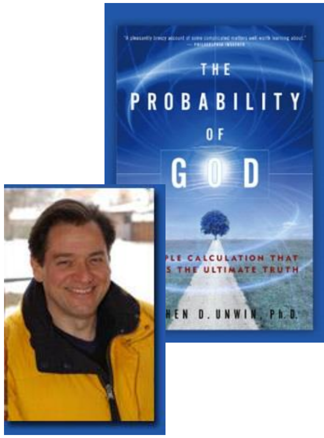
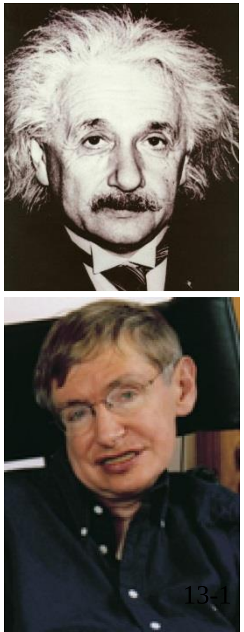
Chapter 13: Ranking Models

I apply some basic rules of probability theory
to calculate the probability of God's existence –
the odds of God, really. **-- Stephen Unwin**



God does not roll dice. **-- Albert Einstein**

Not only does God play dice, but He sometimes
confuses us by throwing them where they can't be seen.
-- Stephen Hawking



Outline

13.1 IR Effectiveness Measures

13.2 Probabilistic IR

13.3 Statistical Language Model

13.4 Latent-Topic Models

13.5 Learning to Rank

following Büttcher/Clarke/Cormack Chapters 12, 8, 9
and/or Manning/Raghavan/Schuetze Chapters 8, 11, 12, 18
plus additional literature for 13.4 and 13.5

13.1 IR Effectivness Measures

ideal measure is user satisfaction
heuristically approximated by benchmarking measures
(on test corpora with query suite and relevance assessment by experts)

Capability to return **only** relevant documents:

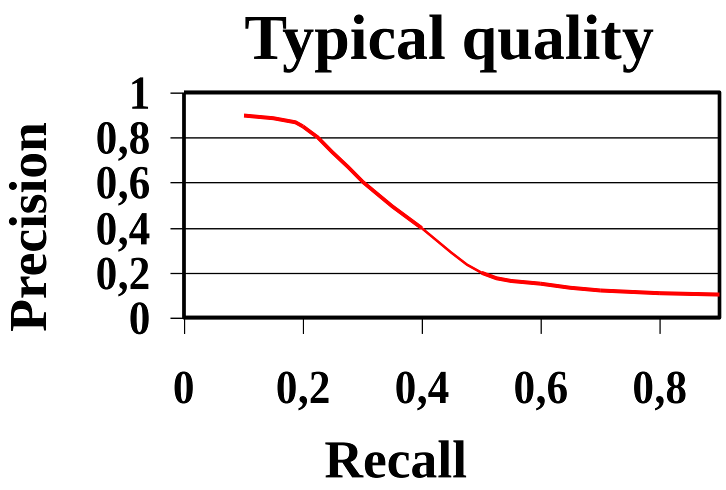
$$\textit{Precision (Präzision)} = \frac{\#\ relevant\ docs\ among\ top\ r}{r}$$

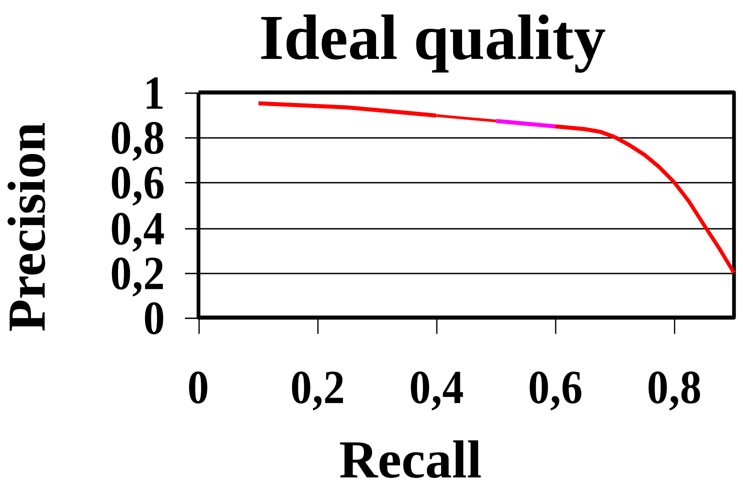
typically for
r = 10, 100, 1000

Capability to return **all** relevant documents:

$$\textit{Recall (Ausbeute)} = \frac{\#\ relevant\ docs\ among\ top\ r}{\#\ relevant\ docs}$$

typically for
r = corpus size





IR Effectiveness: Aggregated Measures

Combining precision and recall into **F measure**
(e.g. with α=0.5:
harmonic mean **F1**): $F = \dfrac{1}{\alpha \dfrac{1}{precision} + (1-\alpha)\dfrac{1}{recall}}$

Precision-recall breakeven point of query q:
point on precision-recall curve p = f(r) with p = r

for a set of n queries q1, ..., qn (e.g. TREC benchmark)

Macro evaluation
(user-oriented) $= \dfrac{1}{n}\sum_{i=1}^{n} precision(qi)$
of precision

Micro evaluation
(system-oriented) $= \dfrac{\sum_{i=1}^{n} \#\ relevant\ \&\ found\ docs\ for\ qi}{\sum_{i=1}^{n} \#\ found\ docs\ for\ qi}$
of precision

analogous for recall and F1

IR Effectivness: Integrated Measures

- **Interpolated average precision** of query q
  with precision p(x) at recall x
  and step width Δ (e.g. 0.1):

$$\frac{1}{1/\Delta}\sum_{i=1}^{1/\Delta} p(i\Delta)$$

area under precision-recall curve

- **Uninterpolated average precision** of query q
  with top-m search result rank list $d_1, \ldots, d_m$,
  relevant results $di_1, \ldots, di_k$ ($k \le m$, $i_j \le i_{j+1} \le m$):

$$\frac{1}{k}\sum_{j=1}^{k} \frac{j}{i_j}$$

- **Mean average precision (MAP)** of query benchmark suite
  macro-average of per-query interpolated average precision
  for top-m results (usually with recall width 0.01)

$$\frac{1}{|Q|}\sum_{q\in Q}\ \frac{1}{1/\Delta}\sum_{i=1}^{1/\Delta}\mathrm{precision}(\mathrm{recall}=\mathrm{i}\Delta)$$

IR Effectiveness: Integrated Measures

plot **ROC curve** (receiver operating characteristics):

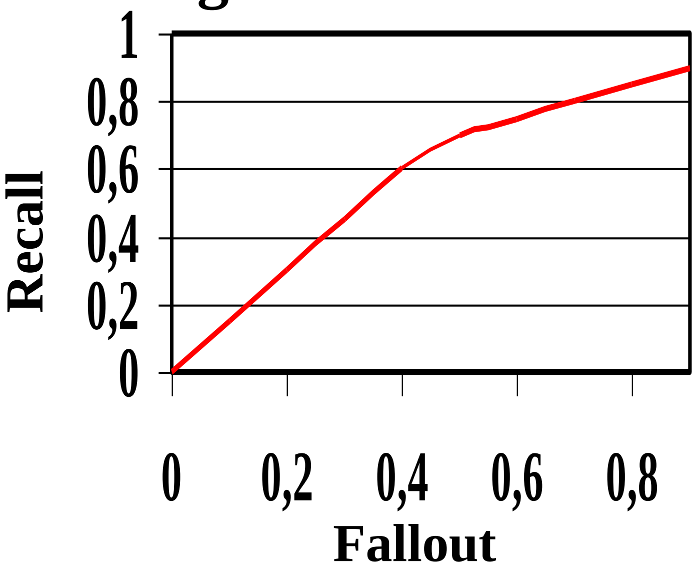
true-positives rate vs. false-positives rate

corresponds to:

Recall vs. Fallout

where Fallout = $\dfrac{\#\ irrelevant\ docs\ among\ top\ r}{\#\ irrelevant\ docs\ in\ corpus}$

good ROC curve:



area under curve (AUC)
is quality indicator

IR Effectiveness: Weighted Measures

Mean reciprocal rank (MRR) over query set Q:

$$MRR = \frac{1}{|Q|} \sum\nolimits_{q \in Q} \frac{1}{FirstRelevantRank(q)}$$

Variation:
summand 0 if
FirstRelevantRank > k

Discounted Cumulative Gain (DCG) for query q:

$$DCG = \sum_{i=1}^{k} \frac{2^{rating(i)} - 1}{\log_2(1+i)}$$

with finite set of result ratings: 0 (irrelevant), 1 (ok), 2(good), …

Normalized Discounted Cumulative Gain (NDCG) for query q:

$$NDCG = DCG \;/\; DCG(PerfectResult)$$

IR Effectiveness: Ordered List Measures

Consider top-k of two rankings τ1 and τ2 or full permutations of 1..n

- ***overlap similarity*** ***OSim*** (τ1,τ2) = | top(k,τ1) ∩ top(k,τ2) | / k

- ***Kendall's τ measure*** ***KDist*** (τ1,τ2) =

$$\frac{|\{(u,v) \mid u,v \in U, u \neq v, \text{ and } \tau 1, \tau 2 \text{ disagree on relative order of } u,v\}|}{|U| \cdot (|U|-1)}$$

with U = top(k,τ1) ∪ top(k,τ2) (with missing items set to rank k+1)

with ties in one ranking and order in the other, count p with 0≤p≤1
→ p=0: weak KDist, → p=1: strict KDist

- ***footrule distance*** ***Fdist*** (τ1,τ2) = $\frac{1}{|U|} \sum_{u \in U} |\tau 1(u) - \tau 2(u)|$

(normalized) Fdist is upper bound for KDist
and Fdist/2 is lower bound

Outline

13.1 IR Effectiveness Measures

13.2 Probabilistic IR

- 13.2.1 Prob. IR with the Binary Model
- 13.2.2 Prob. IR with Poisson Model (Okapi BM25)
- 13.2.3 Extensions with Term Dependencies

13.3 Statistical Language Model

13.4 Latent-Topic Models

13.5 Learning to Rank

13.2 Probabilistic IR

based on **generative model:**
probabilistic mechanism for producing document (or query)

usually with specific family of parameterized distribution

often with assumption of independence among words
justified by „**curse of dimensionality**“:

corpus with n docs and m terms has 2^m possible docs
would have to estimate model parameters from $n << 2^m$
(problems of sparseness & computational tractability)

13.2.1 Multivariate Bernoulli Model (aka. Multi-Bernoulli Model)

For generating doc x

- consider binary RVs: $x_w = 1$ if w occurs in x, 0 otherwise
- postulate independence among these RVs

$$P[x \mid \phi] = \prod_{w \in W} \phi_w^{\,x_w} \, (1-\phi_w)^{1-x_w}$$

with vocabulary W and parameters ϕ_w = P[randomly drawn word is w]

$$= \prod_{w \in x} \phi_w \prod_{w \in W, w \notin x} (1-\phi_w)$$

- product for absent words underestimates prob. of likely docs
- too much prob. mass given to very unlikely word combinations

Probability Ranking Principle (PRP)
[Robertson and Sparck Jones 1976]

Goal:

Ranking based on sim(doc d, query q) =

P[R|d] = P [doc d is relevant for query q |
d has term vector X1, ..., Xm]

Probability Ranking Principle (PRP) [Robertson 1977]:
For a given retrieval task, the cost of retrieving d as the next result in a ranked list is:

$$cost(d) := C_R * P[R|d] + C_{notR} * P[not\ R|d]$$

with cost constants

C_R = cost of retrieving a relevant doc

C_{notR} = cost of retrieving an irrrelevant doc

For $C_R < C_{notR}$, the cost is minimized by choosing

$$argmax_d \; P[R|d]$$

Derivation of PRP

Consider doc d to be retrieved next,
i.e., preferred over all other candidate docs d‘

$$\text{cost}(d) =$$

$$C_R\, P[R|d] + C_{notR}\, P[notR|d] \le C_R\, P[R|d'] + C_{notR}\, P[notR|d']$$

$$= \text{cost}(d')$$

$$\Leftrightarrow\; C_R\, P[R|d] + C_{notR}\,(1 - P[R|d]) \le C_R\, P[R|d'] + C_{notR}\,(1 - P[R|d'])$$

$$\Leftrightarrow\; C_R\, P[R|d] - C_{notR}\, P[R|d] \le C_R\, P[R|d'] - C_{notR}\, P[R|d']$$

$$\left.\begin{array}{l} \Leftrightarrow\; (C_R - C_{notR})\, P[R|d] \le (C_R - C_{notR})\, P[R|d'] \\ \Leftrightarrow\; P[R|d] \ge P[R|d'] \end{array}\right\}\ \text{as } C_R < C_{notR},$$

for all d‘

Probabilistic IR with Binary Independence Model
[Robertson and Sparck Jones 1976]

based on Multi-Bernoulli generative model
and Probability Ranking Principle

Assumptions:

- Relevant and irrelevant documents differ in their terms.
- **Binary Independence Retrieval (BIR) Model**:
  - Probabilities of term occurrence of **different terms** are pairwise **independent**
  - **Term frequencies are binary** $\in \{0,1\}$.
- for terms that do not occur in query q the probabilities for such a term occurring are the same for relevant and irrelevant documents.

BIR principle analogous to **Naive Bayes** classifier

Ranking Proportional to Relevance Odds

$$sim(d,q) = O(R \mid d) = \frac{P[R \mid d]}{P[\neg R \mid d]}$$ (odds for relevance)

$$= \frac{P[d \mid R] \times P[R]}{P[d \mid \neg R] \times P[\neg R]}$$ (Bayes' theorem)

$$\sim \frac{P[d \mid R]}{P[d \mid \neg R]} = \prod_{i=1}^{m} \frac{P[d_i \mid R]}{P[d_i \mid \neg R]}$$ (independence or linked dependence)

$$= \prod_{i \in q} \frac{P[d_i \mid R]}{P[d_i \mid \neg R]}$$ ($P[d_i|R]=P[d_i|\neg R]$ for $i \notin q$)

$$= \prod_{\substack{i \in d \\ i \in q}} \frac{P[X_i = 1 \mid R]}{P[X_i = 1 \mid \neg R]} \cdot \prod_{\substack{i \notin d \\ i \in q}} \frac{P[X_i = 0 \mid R]}{P[X_i = 0 \mid \neg R]}$$

d_i = 1 if d includes term i, 0 otherwise

X_i = 1 if random doc includes term i, 0 otherwise

Ranking Proportional to Relevance Odds

$$= \prod_{\substack{i \in d \\ i \in q}} \frac{p_i}{q_i} \cdot \prod_{\substack{i \notin d \\ i \in q}} \frac{1-p_i}{1-q_i}$$

with estimators $p_i = P[X_i = 1 | R]$ and $q_i = P[X_i = 1 | \neg R]$

$$= \prod_{i \in q} \frac{p_i^{\,d_i}}{q_i^{\,d_i}} \cdot \prod_{i \in q} \frac{(1-p_i)^{1-d_i}}{(1-q_i)^{1-d_i}}$$

$$\sim \sum_{i \in q} \log\left(\frac{p_i^{\,d_i}(1-p_i)}{(1-p_i)^{d_i}}\right) - \log\left(\frac{q_i^{\,d_i}(1-q_i)}{(1-q_i)^{d_i}}\right)$$

$$= \sum_{i \in q} d_i \log \frac{p_i}{1-p_i} + \sum_{i \in q} d_i \log \frac{1-q_i}{q_i} + \sum_{i \in q} \log \frac{1-p_i}{1-q_i}$$

$$\sim \sum_{i \in q} d_i \log \frac{p_i}{1-p_i} + \sum_{i \in q} d_i \log \frac{1-q_i}{q_i} \qquad \sim sim(d, q)$$

Estimating p_i and q_i values: Robertson / Sparck Jones Formula

Estimate p_i und q_i based on training sample
(query q on small sample of corpus) or based on
intellectual assessment of first round's results (*relevance feedback*):

Let N be #docs in sample,
R be # relevant docs in sample
n_i #docs in sample that contain term i,
r_i # relevant docs in sample that contain term i

$\Rightarrow$ Estimate: $p_i = \frac{r_i}{R}$ $\quad q_i = \frac{n_i - r_i}{N - R}$

or: $p_i = \frac{r_i + 0.5}{R + 1}$ $\quad q_i = \frac{n_i - r_i + 0.5}{N - R + 1}$ (Lidstone smoothing with λ=0.5)

$$\Rightarrow \quad sim(d,q) = \sum_{i \in q} d_i \log \frac{r_i + 0.5}{R - r_i + 0.5} + \sum_{i \in q} d_i \log \frac{N - n_i - R + r_i + 0.5}{n_i - r_i + 0.5}$$

$\Rightarrow$ Weight of term i in doc d: $\log \frac{(r_i + 0.5)\,(N - n_i - R + r_i + 0.5)}{(R - r_i + 0.5)(n_i - r_i + 0.5)}$

Example for Probabilistic Retrieval

Documents with relevance feedback:

q: t1 t2 t3 t4 t5 t6

	t1	t2	t3	t4	t5	t6	R
d1	1	0	1	1	0	0	1
d2	1	1	0	1	1	0	1
d3	0	0	0	1	1	0	0
d4	0	0	1	0	0	0	0
ni	2	1	2	3	2	0	
ri	2	1	1	2	1	0	
pi	5/6	1/2	1/2	5/6	1/2	1/6	
qi	1/6	1/6	1/2	1/2	1/2	1/6	

R=2, N=4

with Lidstone smoothing(λ=0.5)

Score of new document d5 (with Lidstone smoothing):

d5$\cap$q: <1 1 0 0 0 1> $\rightarrow$ sim(d5, q) = log 5 + log 1 + log 0.2
+ log 5 + log 5 + log 5

$$\mathrm{sim}(d,q) = \sum_{i \in q} d_i \log \frac{p_i}{1-p_i} + \sum_{i \in q} d_i \log \frac{1-q_i}{q_i}$$

Relationship to tf*idf Formula

Assumptions (**without training sample or relevance feedback**):

- p_i is the same for all i
- Most documents are irrelevant.
- Each individual term i is infrequent.

This implies:

- $\sum_{i \in q} d_i \log \frac{p_i}{1-p_i} = c \sum_{i \in q} d_i$ with constant c
- $q_i = P[X_i = 1 \mid \neg R] \approx \frac{df_i}{N}$
- $\frac{1-q_i}{q_i} = \frac{N - df_i}{df_i} \approx \frac{N}{df_i}$

$$\Rightarrow \quad sim(d,q) = \sum_{i \in q} d_i \log \frac{p_i}{1-p_i} + \sum_{i \in q} d_i \log \frac{1-q_i}{q_i}$$

$$\approx c \sum_{i \in q} d_i + \sum_{i \in q} d_i \cdot log\ idf_i$$

scalar product over the product of tf and dampend idf values for query terms

Laplace Smoothing (with Uniform Prior)

Probabilities p_i and q_i for term i are estimated
by **MLE for binomial distribution**
(repeated coin tosses for relevant docs, showing term i with p_i,
repeated coin tosses for irrelevant docs, showing term i with q_i)

To avoid overfitting to feedback/training,
the estimates should be **smoothed** (e.g. **with uniform prior**):

Instead of estimating $p_i = k/n$ estimate (**Laplace's law of succession**):

$$p_i = (k+1) / (n+2)$$

or with heuristic generalization (**Lidstone's law of succession**):

$$p_i = (k+\lambda) / (n+2\lambda)$$ with $\lambda > 0$ (e.g. $\lambda=0.5$)

And for multinomial distribution (n times w-faceted dice) estimate:

$$p_i = (k_i + 1) / (n + w)$$

Laplace Smoothing as Bayesian Parameter Estimation

posterior likelihood prior

$$P[param\ \theta | data\ d] = P[d|\theta]\mathrm{P}[\theta]\ /\ \mathrm{P[d]}$$

consider:
binom(n,x) with observation *k*

assume:
uniform(x) as prior for param $x \in [0,1]$
$f_{uniform}(x) = 1$

$$P[x|k,n] = P[k,n|x]\mathrm{P[x]}\ /\ \mathrm{P[k,n]}$$

$$= \frac{P[k,n|x]\ f_{uniform}(x)}{\int_0^1 P[k,n|x]\ f_{uniform}(x)\, dx} = \frac{x^k(1-x)^{n-k}}{\int_0^1 x^k(1-x)^{n-k}\ dx}$$

$$\mathrm{E}[x|k,n] = \int_0^1 P[x|k,n]dx = \int_0^1 \frac{x^k(1-x)^{n-k}}{\int_0^1 y^k(1-y)^{n-k}\ dy}\, dx$$

posterior expectation

$$= \frac{B(k+2,n-k+1)}{B(k+1,n-k+1)} = \frac{\Gamma(k+2)\Gamma(n+2)}{\Gamma(n+3)\Gamma(k+1)} = \frac{(k+1)!(n+1)!}{(n+2)!k!} = \frac{k+1}{n+2}$$

with Beta function $B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}\, dt$ $\quad B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$

and Gamma function $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\, dt$ $\quad \Gamma(z+1) = z!$ for $z \in \mathrm{N}$

13.2.2 Poisson Model

For generating doc x

- consider couting RVs: x_w = number of occurrences of w in x
- still postulate independence among these RVs

Poisson model with word-specific parameters μ_w:

$$P[x \mid \mu] = \prod_{w \in W} \frac{e^{-\mu_w} \cdot \mu_w^{\,x_w}}{x_w!} \quad = e^{-\sum_{w \in W} \mu_w} \prod_{w \in x} \frac{\mu_w^{\,x_w}}{x_w!}$$

MLE for μ_w is straightforward
no likelihood penalty by absent words
no control of doc length

$$\hat{\mu}_w = \frac{1}{n} \sum_{i=1..n} tf(w, d_i)$$

Probabilistic IR with Poisson Model (Okapi BM25)

Generalize term weight $w = \log \frac{p(1-q)}{q(1-p)}$

into $w = \log \frac{p_{tf}\, q_0}{q_{tf}\, p_0}$

with p_j, q_j denoting prob. that term occurs j times

in relevant / irrelevant doc

Postulate Poisson distributions:

$p_{tf} = e^{-\lambda} \frac{\lambda^{tf}}{tf\,!}$ relevant docs

$q_{tf} = e^{-\mu} \frac{\mu^{tf}}{tf\,!}$ irrelevant docs

combined into 2-Poisson mixture all docs

Okapi BM25 Scoring Function

Approximation of Poisson model by similarly-shaped function:

$$w := \log \frac{p(1-q)}{q(1-p)} \cdot \frac{tf}{k_1 + tf}$$

BM25 performs very well
has won many benchmark
competitions (TREC etc.)

finally leads to Okapi BM25 weights:

$$w_j(d) := \frac{(k_1+1)tf_j}{k_1((1-b)+b\frac{length(d)}{avgdoclength})+tf_j} \cdot \log \frac{N-df_j+0.5}{df_j+0.5}$$

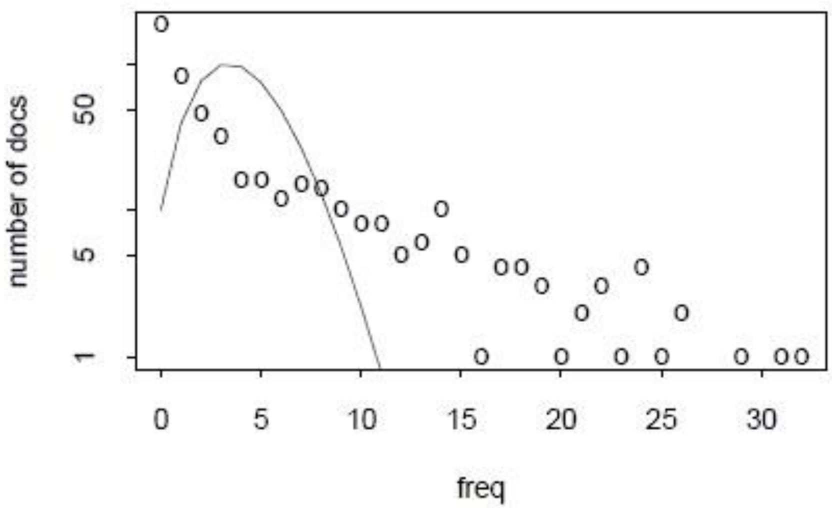
or in the most comprehensive, tunable form: $w_j(d)$ =

$$\log \frac{N-df_j+0.5}{df_j+0.5} \cdot \frac{(k_1+1)tf_j}{k_1((1-b)+b\frac{len(d)}{\Delta})+tf_j} \cdot \frac{(k_3+1)qtf_j}{k_3+tf_j} + k_2\,|q|\,\frac{\Delta - len(d)}{\Delta + len(d)}$$

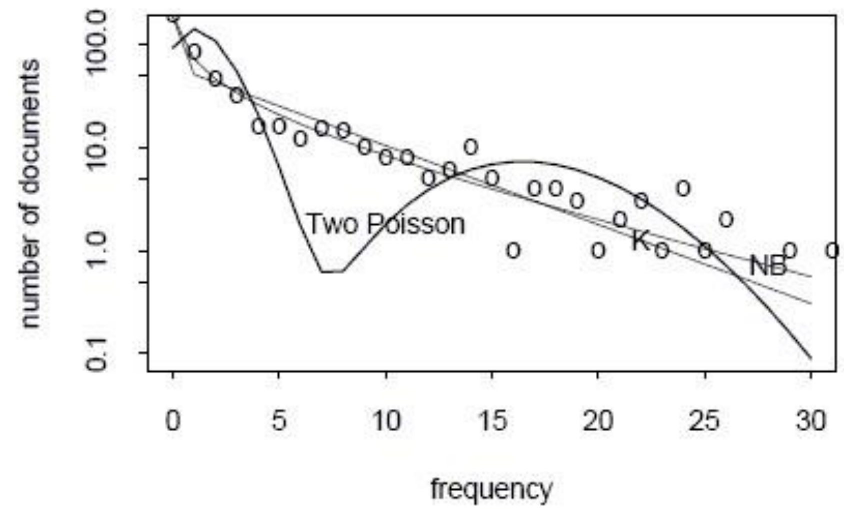
with Δ=avgdoclength and tuning parameters k_1, k_2, k_3, b,
sub-linear influence of tf (via k_1), **consideration of doc length** (via b)

Poisson Mixtures for Capturing tf Distribution

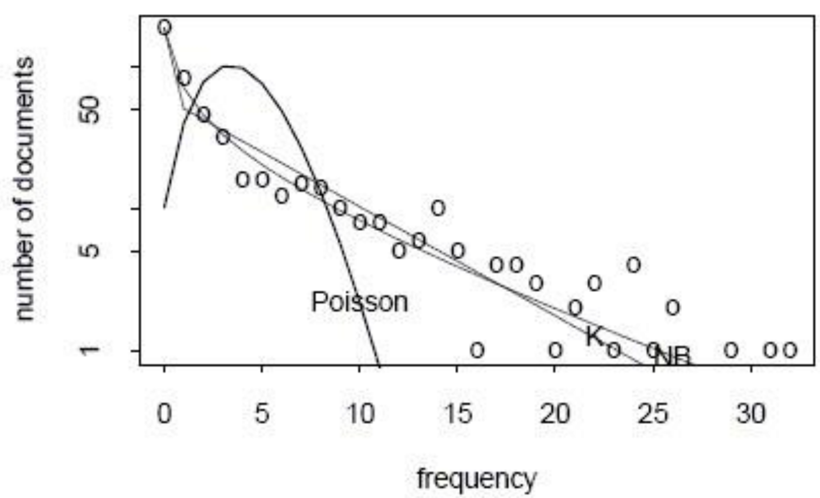
Poisson Doesn't Fit



Two Poissons Are Not Enough



Katz's K-mixture: Poisson Mixtures Fit Better



distribution of
tf values
for term „said"

Source:
Church/Gale 1995

13.2.3 Extensions with Term Dependencies

Consider term correlations in documents (with binary X_i)
$\rightarrow$ Problem of estimating m-dimensional prob. distribution
$P[X_1 = \ldots \wedge X_2 = \ldots \wedge \ldots \wedge X_m = \ldots] =: f_X(X_1, \ldots, X_m)$
(curse of dimensionality)

One possible approach: **Tree Dependence Model:**

a) Consider only 2-dimensional probabilities (for term pairs)

$f_{ij}(X_i, X_j) = P[X_i = .. \wedge X_j = ..] =$

$$\sum_{X_1} .. \sum_{X_{i-1}} \sum_{X_{i+1}} .. \sum_{X_{j-1}} \sum_{X_{j+1}} .. \sum_{X_m} P[X_1 = \ldots \wedge .. \wedge X_m = \ldots]$$

b) For each term pair

estimate the error between independence and the actual correlation

c) Construct a tree with terms as nodes and the

m-1 highest error (or correlation) values as weighted edges

Considering Two-dimensional Term Correlation

Variant 1:

Error of approximating f by g (Kullback-Leibler divergence) with g assuming pairwise term independence:

$$\varepsilon(f,g) := \sum_{\vec{X} \in \{0,1\}^m} f(\vec{X}) \log \frac{f(\vec{X})}{g(\vec{X})} = \sum_{\vec{X} \in \{0,1\}^m} f(\vec{X}) \log \frac{f(\vec{X})}{\prod_{i=1}^{m} g_i(X_i)}$$

Variant 2:

Correlation coefficient for term pairs:

$$\rho(Xi, Xj) := \frac{Cov(Xi, Xj)}{\sqrt{Var(Xi)}\sqrt{Var(Xj)}}$$

Variant 3:

level-α values or p-values

of Chi-square independence test

Example for Approximation Error ε by KL Divergence

m=2:

given are documents:

d1=(1,1), d2(0,0), d3=(1,1), d4=(0,1)

estimation of 2-dimensional prob. distribution f:

$f(1,1) = P[X1=1 \wedge X2=1] = 2/4$

$f(0,0) = 1/4$, $f(0,1) = 1/4$, $f(1,0) = 0$

estimation of 1-dimensional marginal distributions g1 and g2:

$g1(1) = P[X1=1] = 2/4$, $g1(0) = 2/4$

$g2(1) = P[X2=1] = 3/4$, $g2(0) = 1/4$

estimation of 2-dim. distribution g with independent Xi:

$g(1,1) = g1(1)*g2(1) = 3/8$,

$g(0,0) = 1/8$, $g(0,1) = 3/8$, $g(1,0) =1/8$

approximation error ε (KL divergence):

$\varepsilon = 2/4 \log 4/3 + 1/4 \log 2 + 1/4 \log 2/3 + 0$

Constructing the Term Dependence Tree

<u>Given:</u>

complete graph (V, E) with m nodes Xi $\in$V and
m^2 undirected edges $\in$ E with weights ε (or ρ)

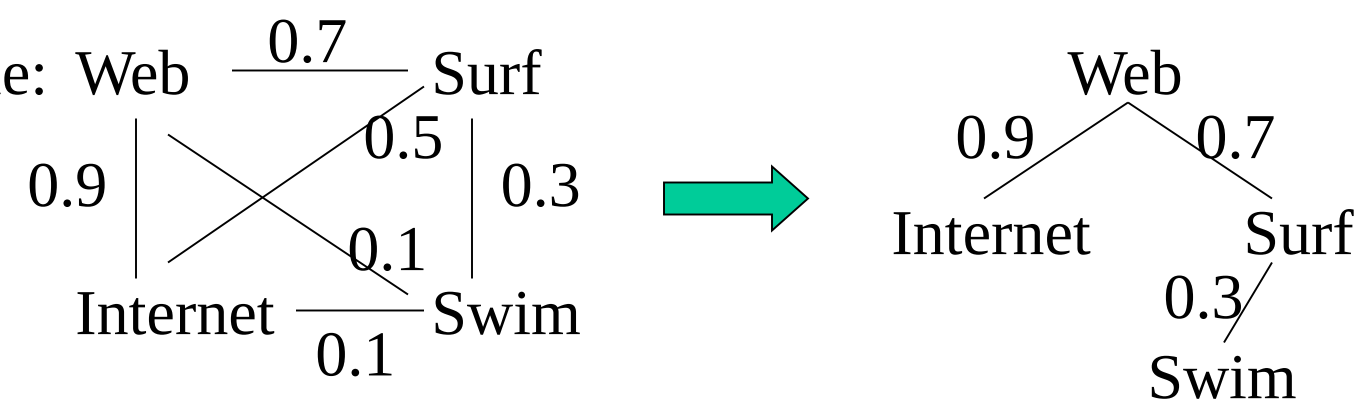
<u>Wanted:</u>

spanning tree (V, E‘) with maximal sum of weights

<u>Algorithm:</u>

Sort the m^2 edges of E in descending order of weight
E‘ := $\emptyset$
Repeat until |E‘| = m-1
  E‘ := E‘ $\cup$ {(i,j) $\in$E | (i,j) has max. weight in E}
  provided that E‘ remains acyclic;
  E := E – {(i,j) $\in$E | (i,j) has max. weight in E}

Example:



Estimation of Multidimensional Probabilities with Term Dependence Tree

Given is a term dependence tree (V = {X1, ..., Xm}, E‘).
Let X1 be the root, nodes are preorder-numbered, and assume that
Xi and Xj are independent for $(i,j) \notin E'$. Then:

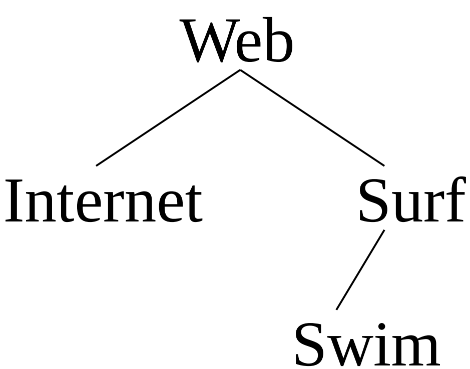
$$P[X1=.. \wedge .. \wedge Xm=..] = P[X1=..]\, P[X2=.. \wedge Xm=.. \mid X1=..]$$

$$= \prod_{i=1..m} P[Xi=.. \mid X1=.. \wedge X(i-1)=..]$$

$$= P[X1] \cdot \prod_{(i,j)\in E'} P[Xj \mid Xi]$$

$$= P[X1] \cdot \prod_{(i,j)\in E'} \frac{P[Xi, Xj]}{P[Xi]}$$

<u>Example:</u>



P[Web, Internet, Surf, Swim] =

$$P[Web] \frac{P[Web, Internet]}{P[Web]} \frac{P[Web, Surf]}{P[Web]} \frac{P[Surf, Swim]}{P[Surf]}$$

Digression: Bayesian Networks

Bayesian network (BN) is a directed, acyclic graph (V, E) with the following properties:

- Nodes $\in$ V representing random variables and
- Edges $\in$ E representing dependencies.
- For a root R $\in$ V the BN captures the prior probability P[R = ...].
- For a node X $\in$ V with parents parents(X) = {P1, ..., Pk} the BN captures the conditional probability P[X=... | P1, ..., Pk].
- Node X is conditionally independent of a non-parent node Y given its parents parents(X) = {P1, ..., Pk}: P[X | P1, ..., Pk, Y] = P[X | P1, ..., Pk].

This implies:

$$P[X1 \ldots Xn] = P[X1 | X2 \ldots Xn]\, P[X2 \ldots Xn]$$

- by the chain rule:

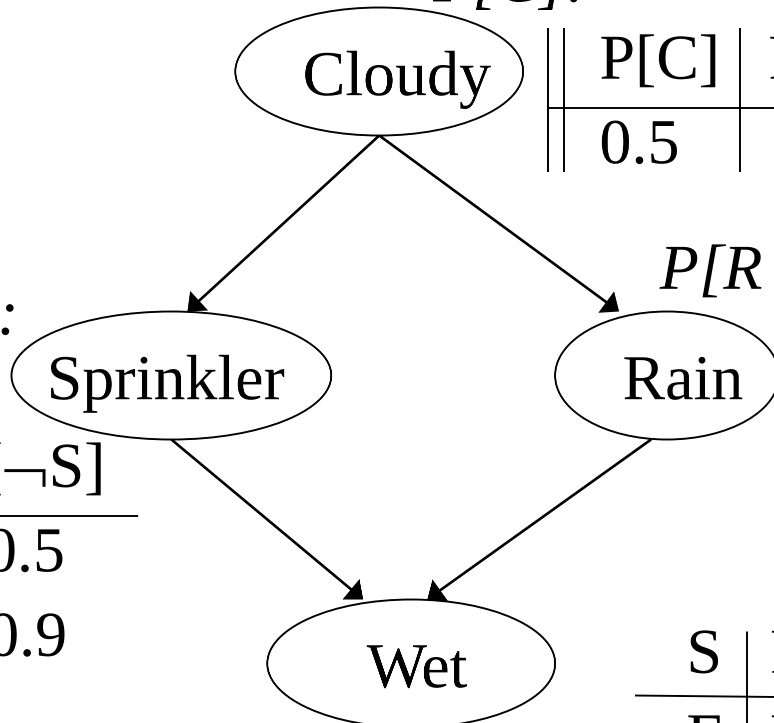
$$= \prod_{i=1}^{n} P[Xi | X(i+1) \ldots Xn]$$

- by cond. independence:

$$= \prod_{i=1}^{n} P[Xi | parents(Xi), other\ nodes]$$

$$= \prod_{i=1}^{n} P[Xi | parents(Xi)]$$

Example of Bayesian Network



P[C]:

P[C]	P[¬C]
0.5	0.5

P[S | C]:

C	P[S]	P[¬S]
F	0.5	0.5
T	0.1	0.9

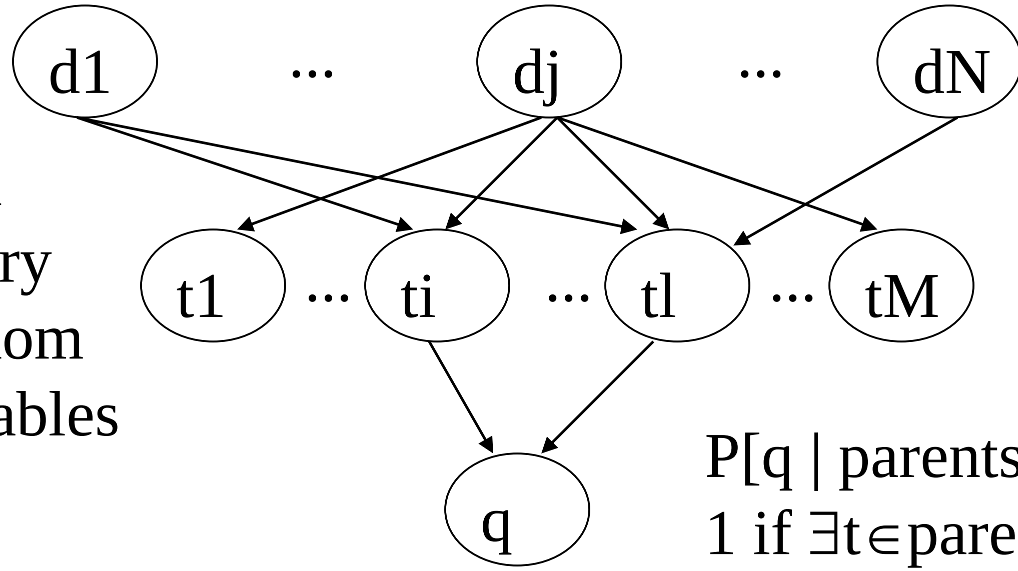
P[R | C]:

C	P[R]	P[¬R]
F	0.2	0.8
T	0.8	0.2

P[W | S,R]:

S	R	P[W]	P[¬W]
F	F	0.0	1.0
F	T	0.9	0.1
T	F	0.9	0.1
T	T	0.99	0.01

Bayesian Inference Networks for IR



with binary random variables

P[dj]=1/N

P[ti | dj∈parents(ti)] =
1 if ti occurs in dj,
0 otherwise

P[q | parents(q)] =
1 if ∃t∈parents(q): t is relevant for q,
0 otherwise

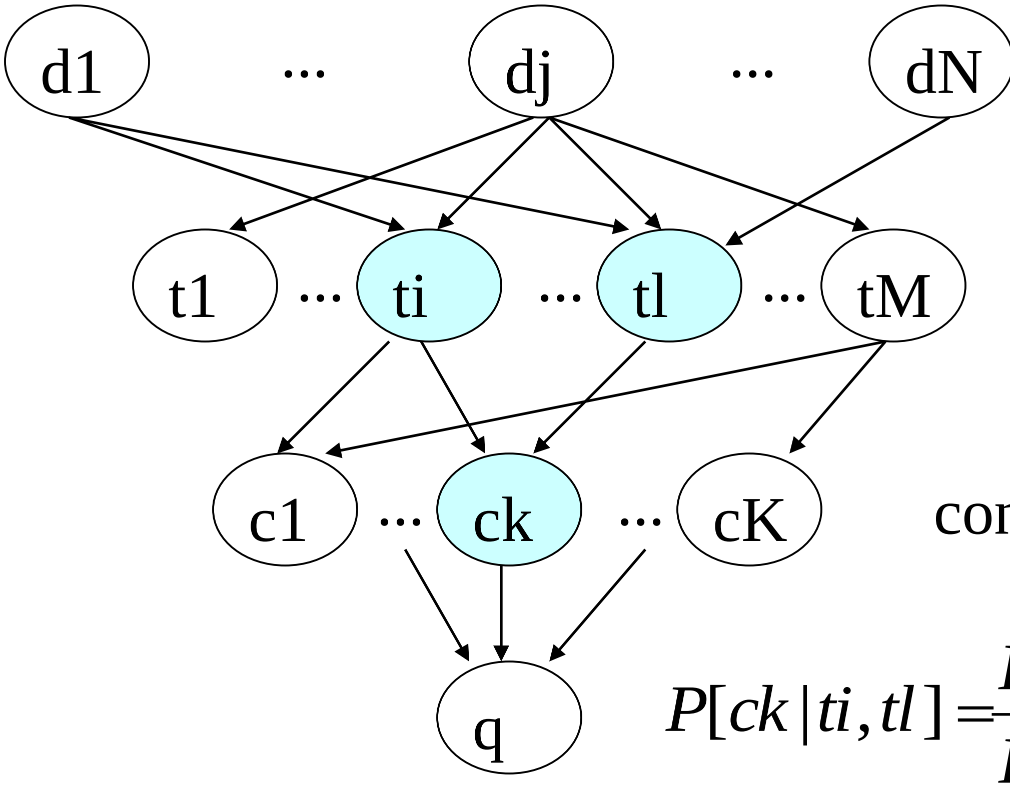
$$P[\,q \wedge dj\,] = \sum_{(t1 \ldots tM)} P[\,q \wedge dj \,|\, t1 \ldots tM\,]\; P[\,t1 \ldots tM\,]$$

$$= \sum_{(t1 \ldots tM)} P[\,q \wedge dj \wedge t1 \wedge \ldots \wedge tM\,]$$

$$= \sum_{(t1 \ldots tM)} P[\,q \,|\, dj \wedge t1 \wedge \ldots \wedge tM\,]\; P[\,dj \wedge t1 \wedge \ldots \wedge tM\,]$$

$$= \sum_{(t1 \ldots tM)} P[\,q \,|\, t1 \wedge \ldots \wedge tM\,]\; P[\,t1 \wedge \ldots \wedge tM \,|\, dj\,]\; P[\,dj\,]$$

Advanced Bayesian Network for IR



Alternative to BN is **MRF (Markov Random Field)** to model query term dependencies [Metzler/Croft 2005]

concepts / topics

$$P[ck \mid ti, tl] = \frac{P[ti \wedge tl]}{P[ti \vee tl]} \approx \frac{df_{il}}{df_i + df_l - df_{il}}$$

Problems:

- parameter estimation (sampling / training)
- (non-) scalable representation
- (in-) efficient prediction
- lack of fully convincing experiments

Summary of Section 13.2

- **Probabilistic IR** reconciles principled foundations with practically effective ranking
- **Binary Independence Retrieval** (Multi-Bernoulli model) can be thought of as a Naive Bayes classifier: simple but effective
- Parameter estimation requires **smoothing**
- **Poisson-model**-based **Okapi BM25** often performs best
- Extensions with **term dependencies** (e.g. **Bayesian Networks**) are (too) expensive for Web IR but may be interesting for specific apps

Additional Literature for Section 13.2

- K. van Rijsbergen: Information Retrieval, Chapter 6: Probabilistic Retrieval, 1979, http://www.dcs.gla.ac.uk/Keith/Preface.html
- R. Madsen, D. Kauchak, C. Elkan: Modeling Word Burstiness Using the Dirichlet Distribution, ICML 2005
- S.E. Robertson, K. Sparck Jones: Relevance Weighting of Search Terms, JASIS 27(3), 1976
- S.E. Robertson, S. Walker: Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval, SIGIR 1994
- A. Singhal: Modern Information Retrieval – a Brief Overview, IEEE CS Data Engineering Bulletin 24(4), 2001
- K.W. Church, W.A. Gale: Poisson Mixtures, Natural Language Engineering 1(2), 1995
- C.T. Yu, W. Meng: Principles of Database Query Processing for Advanced Applications, Morgan Kaufmann, 1997, Chapter 9
- D. Heckerman: A Tutorial on Learning with Bayesian Networks, Technical Report MSR-TR-95-06, Microsoft Research, 1995
- D. Metzler, W.B. Croft: A Markov Random Field Model for Term Dependencies. SIGIR 2005

Outline

13.1 IR Effectiveness Measures

13.2 Probabilistic IR

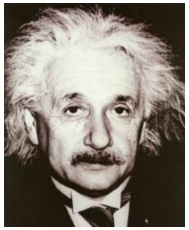
13.3 Statistical Language Model

- 13.3.1 Principles of LMs
- 13.3.2 LMs with Smoothing
- 13.3.3 Extended LMs

13.4 Latent-Topic Models

13.5 Learning to Rank

God does not roll dice. -- Albert Einstein



13.3.1 Key Idea of Statistical Language Models

generative model for word sequence

(generates probability distribution of word sequences, or bag-of-words, or set-of-words, or structured doc, or ...)

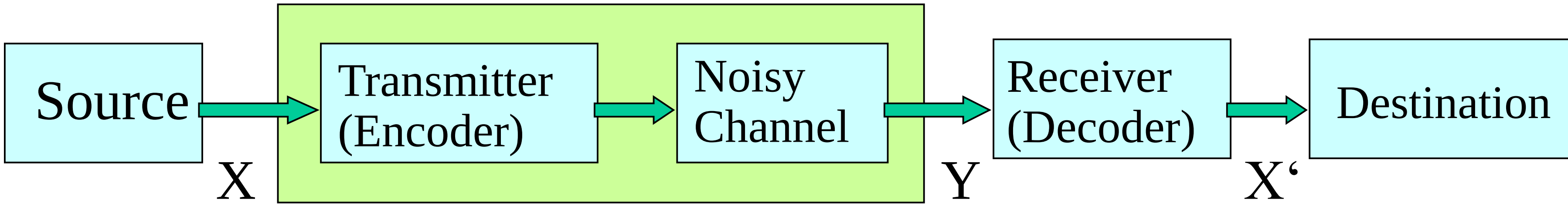
Example: P[„Today is Tuesday“] = 0.001
P[„Today Wednesday is“] = 0.00000000001
P[„The Eigenvalue is positive“] = 0.000001

LM itself highly context- / application-dependent

Examples:

- **speech recognition**: given that we heard „Julia“ and „feels“, how likely will we next hear „happy“ or „habit“?
- **text classification**: given that we saw „soccer“ 3 times and „game“ 2 times, how likely is the news about sports?
- **information retrieval**: given that the user is interested in math, how likely would the user use „distribution“ in a query?

Historical Background: Source-Channel Framework [Shannon 1948]



P[X] P[Y|X] P[X|Y]=?

$$\hat{X} = \arg\max_X P[X \mid Y] = \arg\max_X P[Y \mid X]\, P[X]$$

X is text → P[X] is language model

Applications:

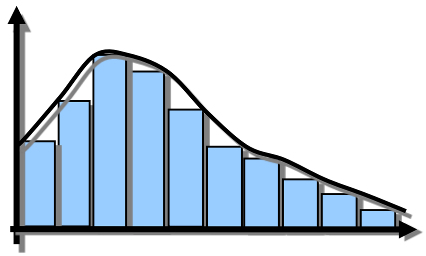
speech recognition	X: word sequence	Y: speech signal
machine translation	X: English sentence	Y: German sentence
OCR error correction	X: correct word	Y: erroneous word
summarization	X: document	Y: summary
information retrieval	X: document	Y: query

Text Generation with (Unigram) LMs

LM θ: P[word | θ] ← sample — document d

LM for topic 1: IR&DM

...	
text	0.2
mining	0.1
n-gram	0.01
cluster	0.02
...	
food	0.000001

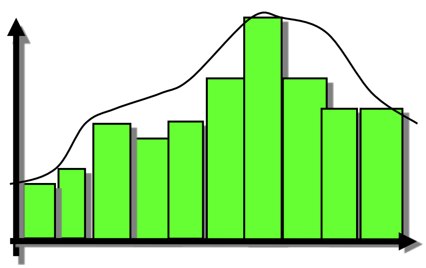


text mining paper

different θ_d for different d

LM for topic 2: Health

...	
food	0.25
nutrition	0.1
healthy	0.05
diet	0.02
...	

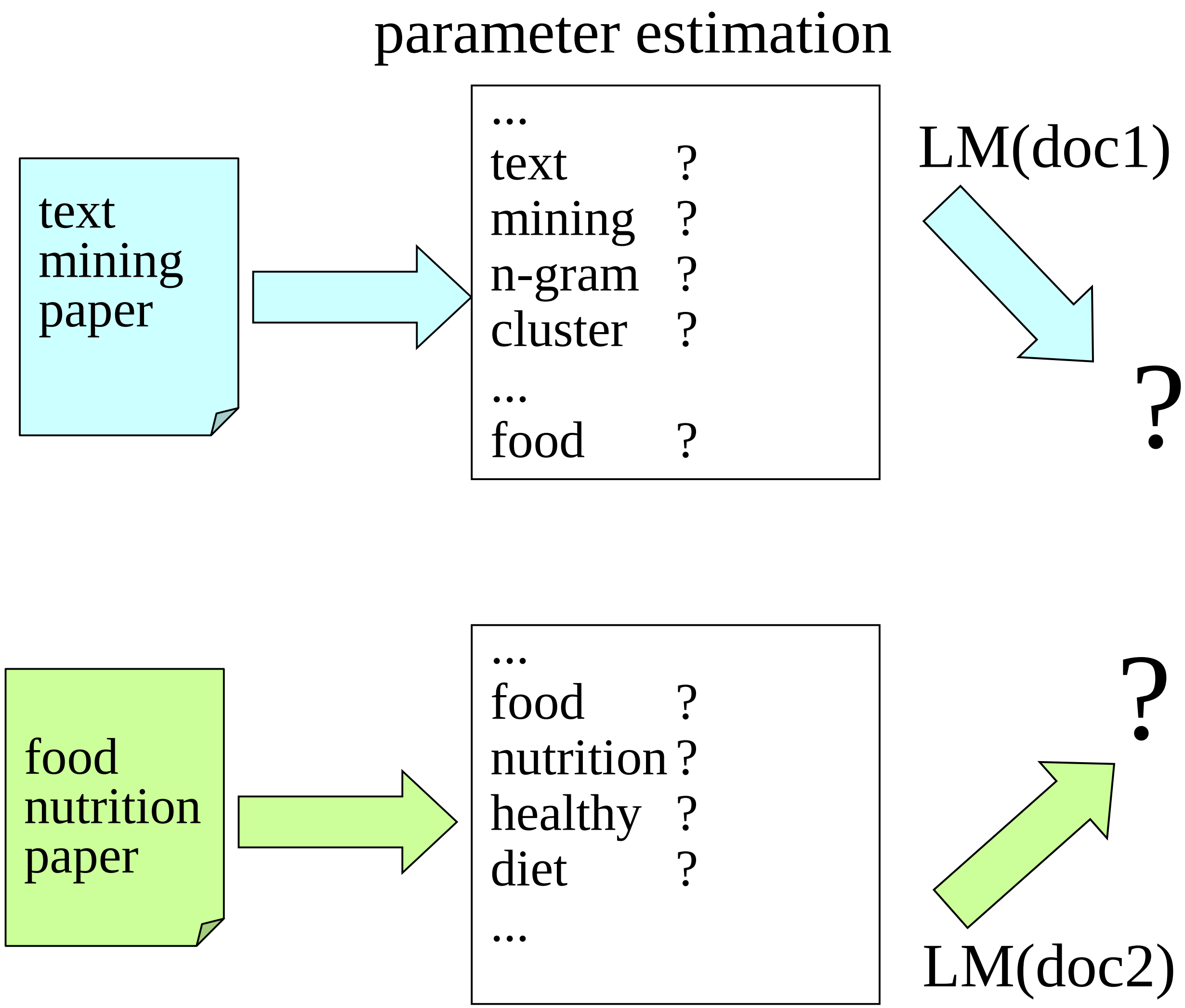


food nutrition paper

may also define LMs over n-grams

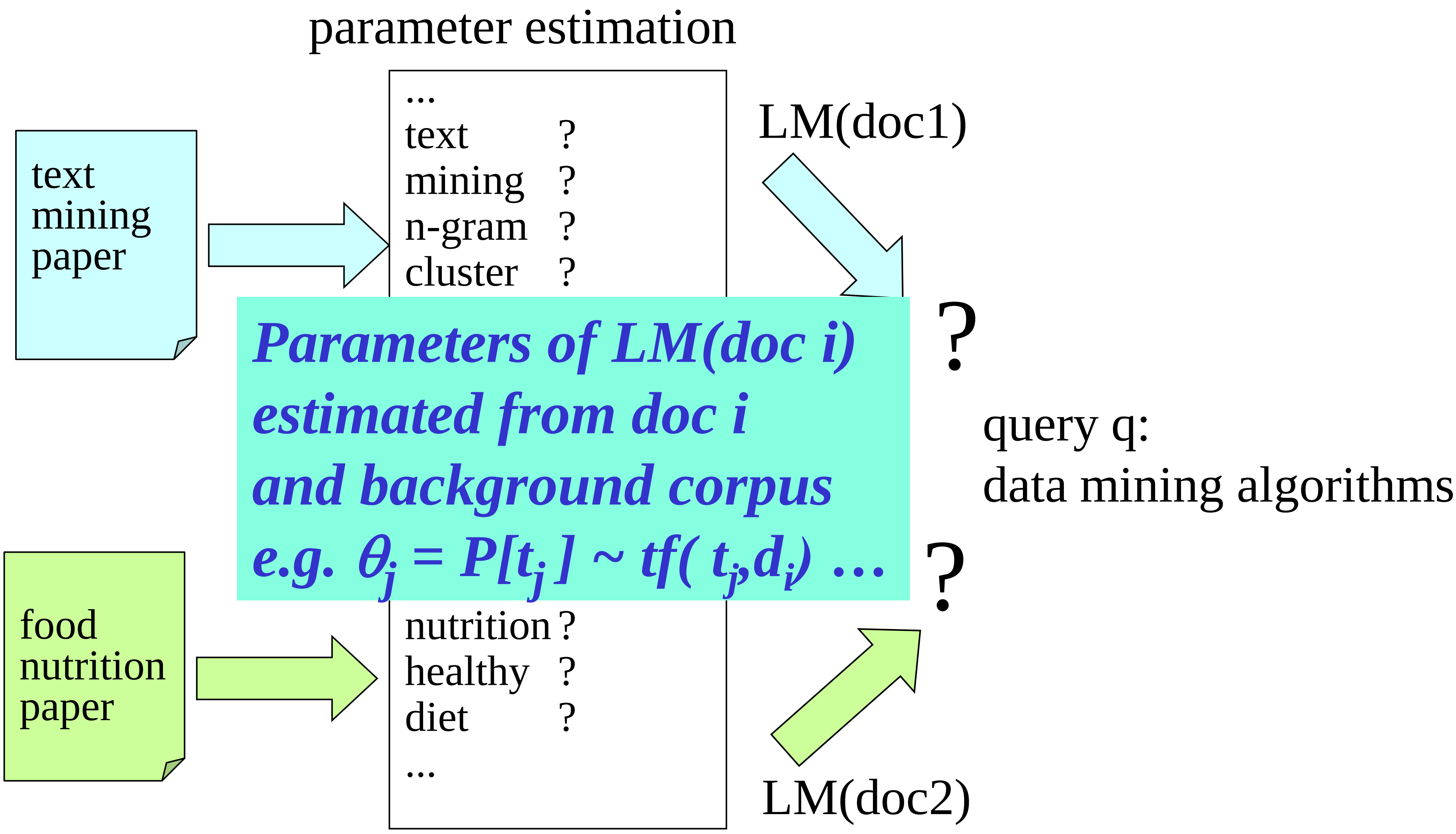
LMs for Ranked Retrieval

Which LM is more likely to generate q? (better explains q)



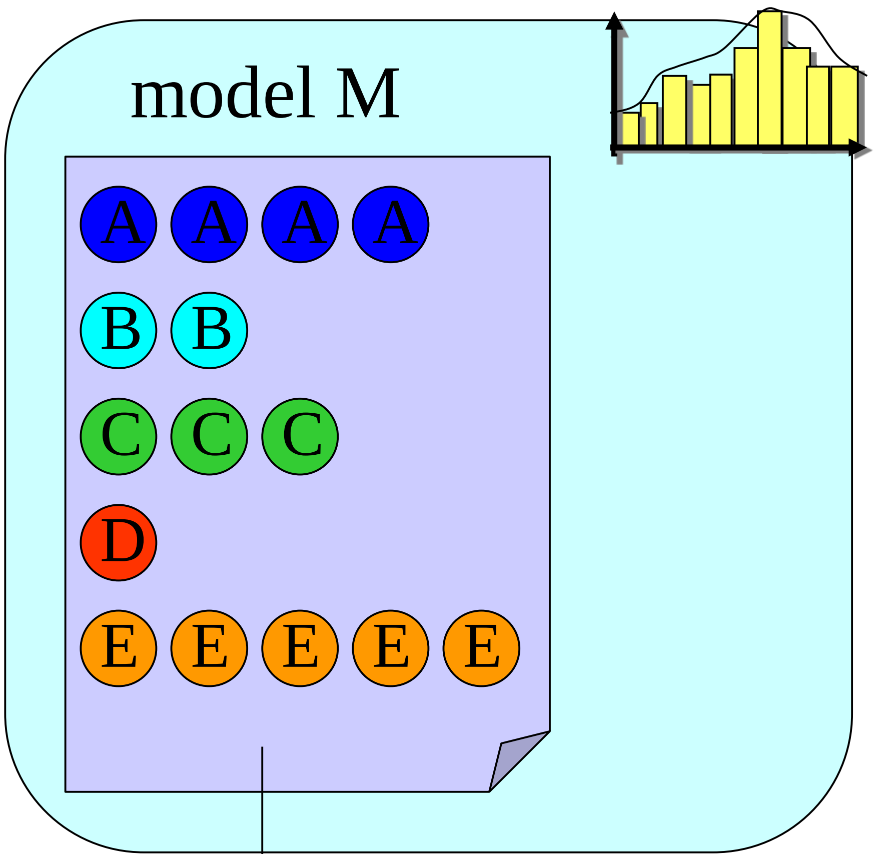
query q:
data mining algorithms

LM Parameter Estimation

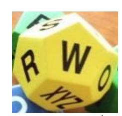


Parameters of LM(doc i) estimated from doc i and background corpus
e.g. $\theta_j = P[t_j] \sim tf(t_j, d_i)$ ***…***

LM Illustration:
Document as Model and Query as Sample



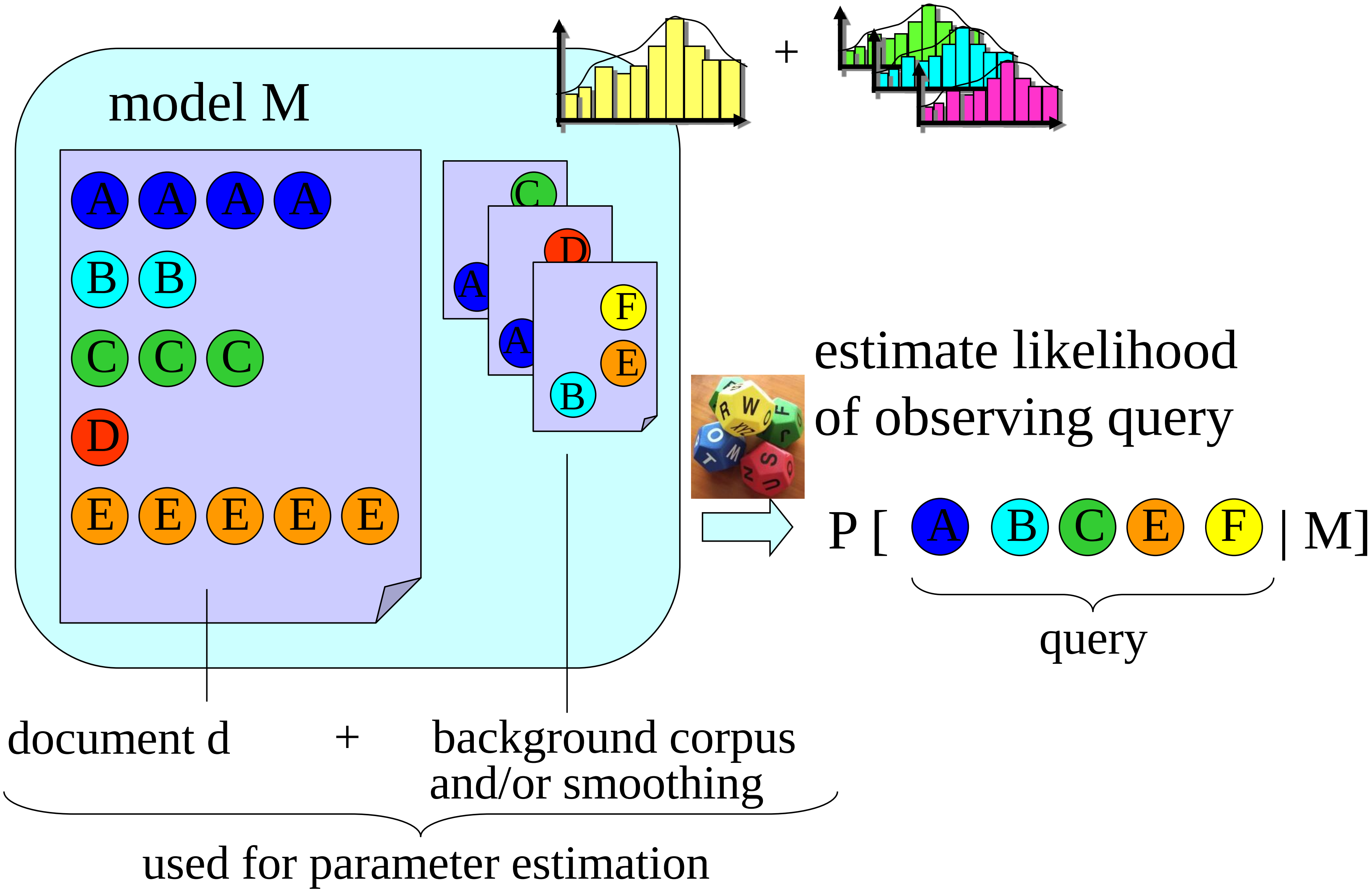
document d: sample of M
used for parameter estimation



estimate likelihood
of observing query

$$P[\,\underbrace{A\;A\;B\;C\;E\;E}_{\text{query}}\mid M]$$

LM Illustration: Need for Smoothing



Probabilistic IR vs. Language Models

P[R | d, q] user considers doc relevant given that it has features d and user has posed query q

$$\sim \frac{P[d \mid R, q]}{P[d \mid \bar{R}, q]}$$

$$\sim \frac{P[q,d|R]}{P[q,d|\bar{R}]}$$

$$= \frac{P[q|R,d]}{P[q|\bar{R},d]} \frac{P[R|d]}{P[\bar{R}|d]} = \ldots \sim \ldots$$

$$\sim \mathrm{P[q|R,d]}$$

Prob. IR
ranks according to
relevance odds

Statistical LMs
rank according to
query likelihood

13.3.2 Query Likelihood Model with Multi-Bernoulli LM

Query is set of terms
generated by d by tossing coin for every term in vocabulary V

$$P[q \mid d] = \prod_{t \in V} \quad p_t(d)^{X_t(q)} \cdot (1 - p_t(d))^{1 - X_t(q)}$$

with $X_t(q)=1$ if $t \in q$, 0 otherwise

$$= \prod_{t \in q} P[t|d] \qquad \sim \sum_{t \in q} \log P[t|d]$$

Parameters θ of LM(d) are P[t|d]
MLE is tf(t,d) / len(d), but model works better with smoothing
→ MAP: Maximum Posterior Likelihood given a prior for parameters

Query Likelihood Model with Multinomial LM

Query is bag of terms
generated by d by rolling a dice for every term in vocabulary V
$\rightarrow$ can capture relevance feedback and user context
(relative importance of terms)

$$P[q \mid d] = \binom{|q|}{f(t_1)\, f(t_2) \ldots f(t_{|q|})} \Pi_{t \in q}\, p_t(d)^{f_t(q)}$$

with $f_t(q)$ = frequency of t in q

Parameters θ of LM(d) are P[t|d] and P[t|q]

Multinomial LM more expressive as a generative model
and thus usually preferred over Multi-Bernoulli LM

Alternative Form of Multinomial LM: Ranking by Kullback-Leibler Divergence

$$\log_2 P[q \mid d] = \log_2 \left(\begin{matrix} |q| \\ f(j_1)\, f(j_2) \ldots f(j_{|q|}) \end{matrix} \right) \Pi_{j \in q}\, p_j(d)^{f_j(q)}$$

$$\sim \Sigma_{j \in q} f_j(q) \log_2 p_j(d)$$

$$= -H(f(q), p(d))$$

neg. cross-entropy

$$\sim -H(f(q), p(d)) + H(f(q))$$

makes **query LM** explicit

$$= -D(f(q) \,\|\, p(d))$$

$$= -\sum_j f_j(q) \log_2 \frac{f_j(q)}{p_j(d)}$$

neg. KL divergence of θ_q and θ_d

Smoothing Methods

absolutely crucial to avoid overfitting and make LMs useful
(one LM per doc, one LM per query !)

<u>possible methods:</u>

- Laplace smoothing
- Absolute Discouting
- Jelinek-Mercer smoothing
- Dirichlet-prior smoothing
- Katz smoothing
- Good-Turing smoothing
- ...

most with their own parameters

choice and parameter setting still pretty much black art (or empirical)

Laplace Smoothing and Absolute Discounting

estimation of θ_d: $p_j(d)$ by MLE would yield $\frac{freq(j,d)}{|d|}$

where $|d| = \sum_j freq(j,d)$

Additive Laplace smoothing:

$$\hat{p}_j(d) = \frac{freq(j,d)+1}{|d|+m}$$

for multinomial over vocabulary W with |W|=m

Absolute discounting:

$$\hat{p}_j(d) = \frac{\max(freq(j,d)-\delta,0)}{|d|} + \sigma \frac{freq(j,C)}{|C|}$$

with corpus C, $\delta \in [0,1]$

where $\sigma = \frac{\delta \cdot \#distinct\ terms\ in\ d}{|d|}$

Jelinek-Mercer Smoothing

Idea:
use linear combination of doc LM with
background LM (corpus LM, common language);

$$\hat{p}_j(d) = \lambda \frac{freq(j,d)}{|d|} + (1-\lambda) \frac{freq(j,C)}{|C|}$$

could also consider query log as background LM for query

parameter tuning of λ by **cross-validation** with held-out data:

- divide set of relevant (d,q) pairs into n partitions
- build LM on the pairs from n-1 partitions
- choose λ to maximize precision (or recall or F1) on n^{th} partition
- iterate with different choice of n^{th} partition and average

Jelinek-Mercer Smoothing: Relationship to tf*idf

$$P[q \mid \theta] = \lambda P[q \mid d] + (1-\lambda) P[q]$$

$$\sim \frac{\lambda}{1-\lambda} \frac{P[q \mid d]}{P[q]} + 1$$

$$\sim \sum_{i \in q} \log P[q_i \mid d] + \log \frac{1}{P[q_i]}$$

$$\sim \sum_{i \in q} \log \frac{tf(i,d)}{\sum_k tf(k,d)} + \log \frac{\sum_k df(k)}{df(i)}$$

Burstiness and the Dirichlet Model

Problem:

- Poisson/multinomial underestimate likelihood of doc with high tf
- **bursty word occurrences** are not unlikely:
  - rare term may be frequent in doc
  - P[tf>0] is low, but P[tf=10 | tf>0] is high

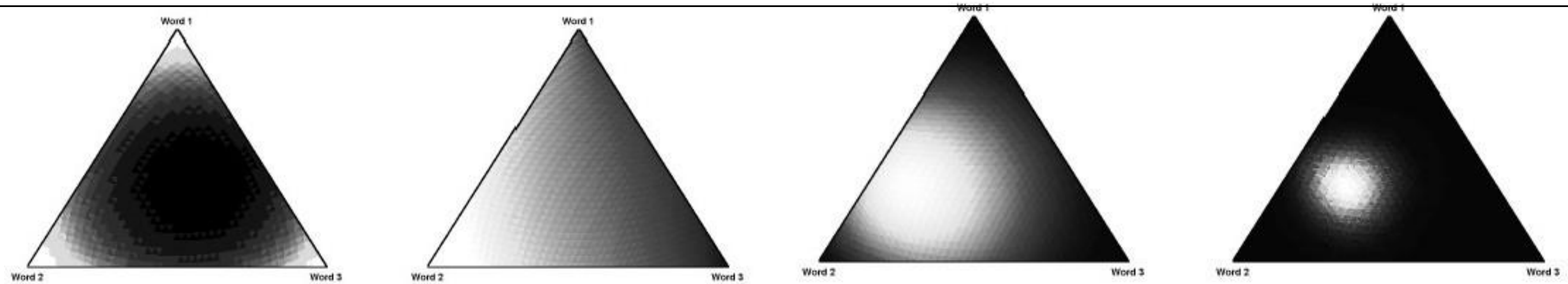
Solution: **two-level model**

- **hypergenerator:**
  to generate doc, first generate **word distribution in corpus**
  (parameters of doc-specific generative model)
- **generator:**
  then generate **word frequencies in doc**, using doc-specific model

Dirichlet Distribution as Hypergenerator for Two-Level Multinomial Model

$$P[\theta \,|\, \alpha] = \frac{\Gamma(\sum_w \alpha_w)}{\prod_w \Gamma(\alpha_w)} \prod_w \theta_w^{\alpha_w - 1} \quad \text{with} \quad \Gamma(x) = \int_0^\infty z^{x-1} e^{-z} dz$$

where $\Sigma_w \theta_w = 1$ and $\theta_w \geq 0$ and $\alpha_w \geq 0$ for all w



$\alpha = (0.44, 0.25, 0.31)$ $\alpha = (1.32, 0.75, 0.93)$ $\alpha = (3.94, 2.25, 2.81)$ $\theta = (0.44, 0.25, 0.31)$

(a) low-scaled DCM (b) mid-scaled DCM (c) high-scaled DCM (d) multinomial

3-dimensional examples of Dirichlet and Multinomial
(Source: R.E. Madsen et al.: Modeling Word Burstiness Using the Dirichlet Distribution)

MAP (Maximum Posterior) of Multinomial with Dirichlet prior is again Dirichlet (with different parameter values)
(„Dirichlet is the conjugate prior of Multinomial“)

Bayesian Viewpoint of Parameter Estimation

- assume **prior distribution** $g(\theta)$ of parameter θ
- choose statistical model (**generative model**) $f(x \mid \theta)$ that reflects our beliefs about RV X
- given RVs $X_1, \ldots, X_n$ for observed data, the **posterior distribution** is $h(\theta \mid x_1, \ldots, x_n)$

for $X_1=x_1, \ldots, X_n=x_n$ the likelihood is

$$L(x_1 \ldots x_n, \theta) = \prod_{i=1}^{n} f(x_i \mid \theta) = \prod_{i=1}^{n} \frac{h(\theta \mid x_i) \cdot \sum_{\theta'} f(x_i \mid \theta') g(\theta')}{g(\theta)}$$

which implies

$$h(\theta \mid x_1 \ldots x_n) \sim L(x_1 \ldots x_n, \theta) \cdot g(\theta)$$

(posterior is proportional to likelihood times prior)

MAP estimator (maximum a posteriori):

compute θ that maximizes $h(\theta \mid x_1, \ldots, x_n)$ *given a prior for* θ

Dirichlet-Prior Smoothing

$$M(\theta) := P[\theta \mid x] = \frac{P[x \mid \theta] \cdot P[\theta]}{\int_\theta P[x \mid \theta][\theta] d\theta}$$

$$= Dirichlet(x + \alpha)$$

Posterior for θ with Dirichlet distribution as prior

with term frequencies x in document d

MAP estimator

$$\hat{p}_j(d) = \hat{\theta}_j = \arg\max M(\theta) = \frac{x_i + \alpha_i - 1}{n + \sum \alpha_i - m} = \frac{|d| \cdot P[j \mid d]}{|d| + \mu} + \frac{\mu \cdot P[j \mid C]}{|d| + \mu}$$

with α_i set to μ P[i|C]+1 for the Dirichlet hypergenerator and $\mu > 1$ set to multiple of average document length

Dirichlet (α): $$f(\theta_1, \ldots, \theta_m) = \frac{\Pi_{j=1..m} \Gamma(\alpha_j)}{\Gamma(\Sigma_{j=1..m} \alpha_j)} \Pi_{j=1..m} \theta_j^{\alpha_j - 1}$$ with $\sum_{j=1..m} \theta_j = 1$

(Dirichlet is conjugate prior for parameters of multinomial distribution: Dirichlet prior implies Dirichlet posterior, only with different parameters)

Dirichlet-Prior Smoothing (cont'd)

$$\hat{p}_j(d) = \left(\lambda P[j\,|\,d] + (1-\lambda) P[j\,|\,C]\right)$$

tf

$$= \frac{|d| \cdot P[j\,|\,d]}{|d| + \mu} + \frac{\mu \cdot P[j\,|\,C]}{|d| + \mu}$$

α_j from corpus

with MLEs P[j|d], P[j|C]

with $\lambda = \frac{|d|}{|d| + \mu}$

where $\alpha_1 = \mu\, P[1|C]$, ..., $\alpha_m = \mu\, P[m|C]$ are the parameters of the underlying Dirichlet distribution, with constant $\mu > 1$ typically set to multiple of average document length

Note 1: conceptually extend d by μ terms randomly drawn from corpus

Note 2: Dirichlet smoothing thus takes the syntactic form of Jelinek-Mercer smoothing

Multinomial LM with Dirichlet Smoothing (Final Wrap-Up)

$$score(d,q) = P[q\,|\,d] = \sum_{j \in q} \left(\lambda P[\,j\,|\,d\,] + (1-\lambda) P[\,j\,|\,C\,] \right)$$

$$= \sum_{j \in q} \left(\frac{|\,d\,| \cdot P[\,j\,|\,d\,]}{|\,d\,| + \mu} + \frac{\mu \cdot P[\,j\,|\,C\,]}{|\,d\,| + \mu} \right)$$

setting $\lambda = \frac{|\,d\,|}{|\,d\,| + \mu}$

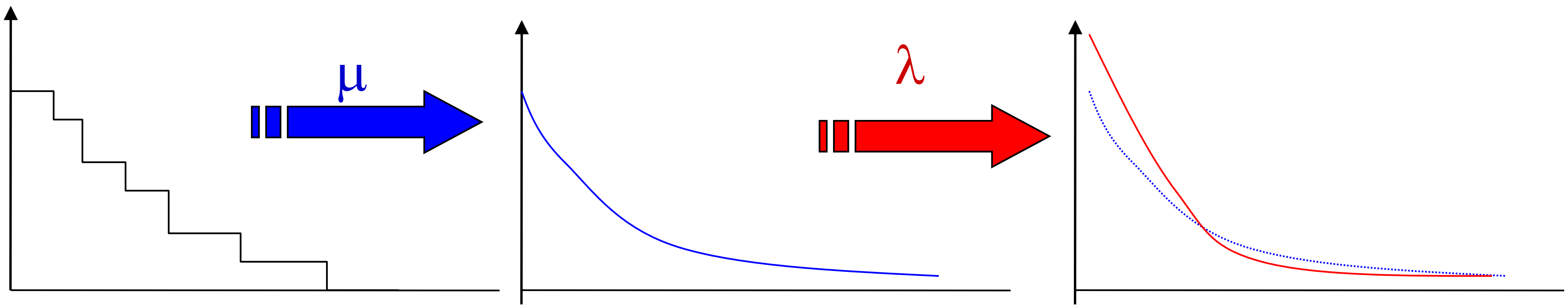
Can also integrate P[j|R] with **relevance feedback LM** or P[j|U] with **user (context) LM**

Multinomial LMs with Dirichlet smoothing the - often best performing – **method of choice** for ranking

LMs of this kind are **composable building blocks** (via probabilistic mixture models)

Two-Stage Smoothing [Zhai/Lafferty, TOIS 2004]

Stage-1	Stage-2
-Explain unseen words	-Explain noise in query
-Dirichlet prior(Bayesian)	-2-component mixture



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

Source: Manning/Raghavan/Schütze, lecture12-lmodels.ppt

13.3.3 Extended LMs

large variety of extensions and combinations:

- N-gram (Sequence) Models and Mixture Models
- Semantic) Translation Models
- Cross-Lingual Models
- Query-Log- and Click-Stream-based LM
- Temporal Search
- LMs for Entity Search
- LMs for Passage Retrieval for Question Answering

N-Gram and Mixture Models

Mixture of LM for **bigrams** and LM for **unigrams**
for both docs and queries,
aiming to capture query phrases / term dependencies,
e.g.: Bob Dylan cover songs by African singers
→ **query segmentation** / query understanding

HMM-style models to capture **informative N-grams**
→ $P[t_i \mid d] \sim P[t_i \mid t_{i-1}]\ P[t_{i-1} \mid d] \ldots$

Mixture models with LMs for unigrams, bigrams,
ordered term pairs in **window**, **unordered** term pairs in **window**, …

Parameter estimation needs Big Data
→ tap **n-gram web/book collections**, query logs, dictionaries, etc.
→ data mining to obtain most informative correlations

(Semantic) Translation Model

$$P[q \mid d] = \prod_{j \in q} \sum_{w} P[j \mid w] \cdot P[w \mid d]$$

with word-word translation model P[j|w]

<u>Opportunities and difficulties:</u>
- synonymy, hypernymy/hyponymy, etc.
- efficiency
- training

estimate P[j|w] by overlap statistics on background corpus (Dice coefficients, Jaccard coefficients, etc.)

Translation Models for Cross-Lingual IR

$$P[q \mid d] = \prod_{j \in q} \sum_{w} P[j \mid w] \cdot P[w \mid d]$$

with q in language F (e.g. French)
and d in language E (e.g. English)

can rank docs in E (or F) for queries in F
Example: q: „moteur de recherche“
returns
d: „Quaero is a French initiative for developing a search engine that can serve as a European alternative to Google ... “

needs estimations of P[j|w] from parallel corpora
(docs available in both F and E)

see also benchmark CLEF: http://www.clef-campaign.org/

Query-Log-Based LM (User LM)

Idea:

for current query q_k leverage

prior **query history $H_q = q_1 \ldots q_{k-1}$** and

prior **click stream $H_c = d_1 \ldots d_{k-1}$** as background LMs

Example:

q_k = „java library" benefits from q_{k-1} = „python programming"

Mixture Model with Fixed Coefficient Interpolation:

$$P[w \mid q_i] = \frac{freq(w, q_i)}{\mid q_i \mid} \qquad P[w \mid H_q] = \frac{1}{k-1} \sum_{i=1..k-1} P[w \mid q_i]$$

$$P[w \mid d_i] = \frac{freq(w, d_i)}{\mid d_i \mid} \qquad P[w \mid H_c] = \frac{1}{k-1} \sum_{i=1..k-1} P[w \mid d_i]$$

$$P[w \mid H_q, H_c] = \beta\, P[w \mid H_q] + (1-\beta)\, P[w \mid H_c]$$

$$P[w \mid \theta_k] = \alpha\, P[w \mid q_k] + (1-\alpha)\, P[w \mid H_q, H_c]$$

LM for Temporal Search [K. Berberich et al.: ECIR 2010]

keyword queries that express temporal interest
example: q = „FIFA world cup 1990s“
$\rightarrow$ would not retrieve doc
d = „France won the FIFA world cup in 1998“

Approach:

- extract temporal phrases from docs
- normalize temporal expressions
- split query and docs into text $\times$ time

$$P[q \mid d] = P[text(q) \mid text(d)] \cdot P[time(q) \mid time(d)]$$

$$P[time(q) \mid time(d)] = \prod_{temp\ expr\ x \in q} \sum_{temp\ expr\ y \in d} P[x \mid y]$$

plus smoothing

$$P[x \mid y] \sim \frac{|x \cap y|}{|x| \cdot |y|}$$

with $|x| = end(x) - begin(x)$

Entity Search with LM [Nie et al.: WWW'07]

Assume entities marked in docs by information extraction methods

query: keywords → answer: entities

$$score(e,q) = \lambda P[q \mid e] + (1-\lambda) P[q] \sim \prod \frac{P[q_i \mid e_i]}{P[q_i]} \sim KL(LM(q) \mid LM(e))$$

LM (entity e) = prob. distr. of words seen in context of e

query q:
„French soccer player Bayern“

candidate entities:

e1: Franck Ribery

e2: Manuel Neuer

e3: Kingsley Coman

e4: Zinedine Zidane

e5: Real Madrid

French soccer champions
champions league with Bayern
French national team Equipe Tricolore
played soccer FC Bayern Munich

Zizou champions league 2002
Real Madrid Johan Cruyff Dutch
soccer world cup best player
2002 won against Bayern

docs weighted by extraction confidence

Language Models for Question Answering (QA)

question

↓ question-type-specific NL parsing

query

↓ finding most promising short text passages

passages

↓ NL parsing and entity extraction

answers

More on QA in Chapter 16 of this course

e.g. factoid questions:
who? where? when? ...

example:
Where is the Louvre museum located?

Louvre museum location

...
The Louvre is the most visited and one of the oldest, largest, and most famous art galleries and museums in the world. It is located in Paris, France. Its address is Musée du Louvre, 75058 Paris cedex 01.
...

The Louvre museum is in Paris.

Use of LMs:

- **Passage retrieval**: likelihood of passage generating question
- **Translation model**: likelihood of answer generating question with param. estim. from manually compiled **question-answer corpus**

Summary of Section 13.3

- LMs are a clean form of **generative models** for docs, corpora, queries:
  - one LM per doc (with doc itself for parameter estimation)
  - **likelihood of LM generating query** yields ranking of docs
  - for **multinomial model**: equivalent to ranking by KL (q || d)
- **parameter smoothing** is essential:
  - use **background corpus**, query&click log, etc.
  - Jelinek-Mercer and **Dirichlet smoothing** perform very well
- LMs very useful for advanced IR: cross-lingual, passages for QA, entity search, etc.

Additional Literature for Section 13.3

Statistical Language Models in General:

- Djoerd Hiemstra: Language Models, Smoothing, and N-grams, in: Encyclopedia of Database Systems, Springer, 2009
- ChengXiang Zhai, Statistical Language Models for Information Retrieval, Morgan & Claypool Publishers, 2008
- ChengXiang Zhai, Statistical Language Models for Information Retrieval: A Critical Review, Foundations and Trends in Information Retrieval 2(3), 2008
- X. Liu, W.B. Croft: Statistical Language Modeling for Information Retrieval, Annual Review of Information Science and Technology 39, 2004
- J. Ponte, W.B. Croft: A Language Modeling Approach to Information Retrieval, SIGIR 1998
- C. Zhai, J. Lafferty: A Study of Smoothing Methods for Language Models Applied to Information Retrieval, TOIS 22(2), 2004
- C. Zhai, J. Lafferty: A Risk Minimization Framework for Information Retrieval, Information Processing and Management 42, 2006
- M.E. Maron, J.L. Kuhns: On Relevance, Probabilistic Indexing, and Information Retrieval, Journal of the ACM 7, 1960

Additional Literature for Section 13.3

LMs for Specific Retrieval Tasks:

- X. Shen, B. Tan, C. Zhai: Context-Sensitive Information Retrieval Using Implicit Feedback, SIGIR 2005
- Y. Lv, C. Zhai, Positonal Language Models for Information Retrieval, SIGIR 2009
- V. Lavrenko, M. Choquette, W.B. Croft: Cross-lingual relevance models. SIGIR'02
- D. Nguyen, A. Overwijk, C. Hauff, D. Trieschnigg, D. Hiemstra, F. de Jong: WikiTranslate: Query Translation for Cross-Lingual Information Retrieval Using Only Wikipedia. CLEF 2008
- C. Clarke, E.L. Terra: Passage retrieval vs. document retrieval for factoid question answering. SIGIR 2003
- Z. Nie, Y. Ma, S. Shi, J.-R. Wen, W.-Y. Ma: Web object retrieval. WWW 2007
- H. Zaragoza et al.: Ranking very many typed entities on wikipedia. CIKM 2007
- P. Serdyukov, D. Hiemstra: Modeling Documents as Mixtures of Persons for Expert Finding. ECIR 2008
- S. Elbassuoni, M. Ramanath, R. Schenkel, M. Sydow, G. Weikum: Language-model-based Ranking for Queries on RDF-Graphs. CIKM 2009
- K. Berberich, O. Alonso, S. Bedathur, G. Weikum: A Language Modeling Approach for Temporal Information Needs. ECIR 2010
- D. Metzler, W.B. Croft: A Markov Random Field Model for Term Dependencies. SIGIR 2005
- S. Huston, W.B. Croft: A Comparison of Retrieval Models using Term Dependencies. CIKM 2014