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 Effectiveness 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:

\[
\text{Precision (Präzision)} = \frac{\# \text{relevant docs among top } r}{r}
\]

typically for \( r = 10, 100, 1000 \)

Capability to return all relevant documents:

\[
\text{Recall (Ausbeute)} = \frac{\# \text{relevant docs among top } r}{\# \text{relevant docs}}
\]

typically for \( r = \text{corpus size} \)

Typical quality:

![Precision vs Recall graph]

Ideal quality:

![Precision vs Recall graph]
IR Effectiveness: Aggregated Measures

Combining precision and recall into \( F \) measure (e.g. with \( \alpha=0.5 \): harmonic mean \( F_1 \)): 
\[
F = \frac{1}{\alpha \frac{1}{\text{precision}} + (1 - \alpha) \frac{1}{\text{recall}}}
\]

**Macro evaluation** (user-oriented) = 
\[
\frac{1}{n} \sum_{i=1}^{n} \text{precision (}\text{qi}\text{)}
\]

**Micro evaluation** (system-oriented) = 
\[
\frac{\sum_{i=1}^{n} \# \text{ relevant & found docs for qi}}{\sum_{i=1}^{n} \# \text{ found docs for qi}}
\]

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

for a set of \( n \) queries \( q_1, \ldots, q_n \) (e.g. TREC benchmark)

analogous for recall and \( F_1 \)
IR Effectivness: Integrated Measures

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

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

- **Uninterpolated average precision** of query q
  with top-m search result rank list $d_1, \ldots, d_m$, relevant results $d_{i_1}, \ldots, d_{i_k}$ ($k \leq m$, $i_j \leq i_{j+1} \leq m$):

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

- **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)

$$\frac{1}{|Q|} \sum_{q \in Q} \frac{1}{1/\Delta} \sum_{i=1}^{1/\Delta} \text{precision}(\text{recall} = i\Delta)$$
IR Effectiveness: Integrated Measures

plot **ROC curve** (receiver operating characteristics):
true-positives rate vs. false-positives rate

corresponds to:

Recall vs. Fallout
where $\text{Fallout} = \frac{\text{# irrelevant docs among top } r}{\text{# irrelevant docs in corpus}}$

**good ROC curve:**

area under curve (AUC) is quality indicator
IR Effectiveness: Weighted Measures

Mean reciprocal rank (MRR) over query set Q:

\[
\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\text{First Relevant Rank}(q)}
\]

Discounted Cumulative Gain (DCG) for query q:

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

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

Normalized Discounted Cumulative Gain (NDCG) for query q:

\[
\text{NDCG} = \frac{\text{DCG}}{\text{DCG}(\text{Perfect Result})}
\]
IR Effectiveness: Ordered List Measures

Consider top-k of two rankings \( \tau_1 \) and \( \tau_2 \) or full permutations of 1..n

- **overlap similarity** \( OSim(\tau_1, \tau_2) = \frac{|\text{top}(k, \tau_1) \cap \text{top}(k, \tau_2)|}{k} \)

- **Kendall's \( \tau \) measure** \( KD\text{Dist}(\tau_1, \tau_2) = \frac{|\{(u,v) | u, v \in U, u \neq v, \text{ and } \tau_1, \tau_2 \text{ disagree on relative order of } u, v\}|}{|U| \cdot (|U|-1)} \)
  
  with \( U = \text{top}(k, \tau_1) \cup \text{top}(k, \tau_2) \) (with missing items set to rank k+1)

  with ties in one ranking and order in the other, count \( p \) with \( 0 \leq p \leq 1 \)
  
  \( \rightarrow p = 0: \) weak \( KD\text{Dist}, \rightarrow p = 1: \) strict \( KD\text{Dist} \)

- **footrule distance** \( FD\text{ist}(\tau_1, \tau_2) = \frac{1}{|U|} \sum_{u \in U} |\tau_1(u) - \tau_2(u)| \)

  (normalized) \( FD\text{ist} \) is upper bound for \( KD\text{Dist} \) and \( FD\text{ist}/2 \) 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

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

with vocabulary $W$ and parameters $\phi_w = P[\text{randomly drawn word is w}]$

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

- 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) =
  \[ P[R|d] = P \text{ [ doc d is relevant for query q } | \]
  \[ d \text{ has term vector X1, ..., Xm } ] \]

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:
  \[ \text{cost}(d) := C_R \ast P[R|d] + C_{notR} \ast P[not R|d] \]
with cost constants
  \[ C_R = \text{cost of retrieving a relevant doc} \]
  \[ C_{notR} = \text{cost of retrieving an irrelevant doc} \]
For \(C_R < C_{notR}\), the cost is minimized by choosing
  \[ \text{argmax}_d \ P[R|d] \]
Derivation of PRP

Consider doc \( d \) to be retrieved next, i.e., preferred over all other candidate docs \( d' \)

\[
\text{cost}(d) = C_R P[R|d] + C_{\text{not}R} P[\text{not}R|d] \leq C_R P[R|d'] + C_{\text{not}R} P[\text{not}R|d']
\]

\[
= \text{cost}(d')
\]

\[\Leftrightarrow C_R P[R|d] + C_{\text{not}R} (1 - P[R|d]) \leq C_R P[R|d'] + C_{\text{not}R} (1 - P[R|d']) \]

\[\Leftrightarrow C_R P[R|d] - C_{\text{not}R} P[R|d] \leq C_R P[R|d'] - C_{\text{not}R} P[R|d'] \]

\[\Leftrightarrow (C_R - C_{\text{not}R}) P[R|d] \leq (C_R - C_{\text{not}R}) P[R|d'] \]

\[\Leftrightarrow P[R|d] \geq P[R|d'] \]

as \( C_R < C_{\text{not}R} \),

for all \( d' \)
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

\[ \text{sim}(d, q) = O(R \mid d) = \frac{P[R \mid d]}{P[\neg R \mid d]} \]

\[ = \frac{P[d \mid R] \times P[R]}{P[d \mid \neg R] \times P[\neg R]} \]

\[ \sim \frac{P[d \mid R]}{P[d \mid \neg R]} = \prod_{i=1}^{m} \frac{P[d_i \mid R]}{P[d_i \mid \neg R]} \]

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

\( d_i = 1 \) if \( d \) includes term \( i \), 0 otherwise

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

(odds for relevance)

(Bayes’ theorem)

(independence or linked dependence)

\((P[d_i \mid R] = P[d_i \mid \neg R] \text{ for } i \not\in q)\)
Ranking Proportional to Relevance Odds

\[ = \prod_{i \in d} \frac{p_i}{q_i} \cdot \prod_{i \notin d} \frac{1-p_i}{1-q_i} \]

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

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

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

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

\[ \sim \sum_{i \in q} d_i \log \frac{p_i}{1-p_i} + \sum_{i \in q} d_i \log \frac{1-q_i}{q_i} \sim \text{sim}(d, q) \]
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:  
\( p_i = \frac{r_i}{R} \)
\( q_i = \frac{n_i - r_i}{N - R} \)

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

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

\[ \Rightarrow \] 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:

<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ni</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ri</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>pi</td>
<td>5/6</td>
<td>1/2</td>
<td>1/2</td>
<td>5/6</td>
<td>1/2</td>
<td>1/6</td>
<td></td>
</tr>
<tr>
<td>qi</td>
<td>1/6</td>
<td>1/6</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/6</td>
<td></td>
</tr>
<tr>
<td>R=2, N=4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Score of new document d5 (with Lidstone smoothing):

\[ \text{d5} \cap q: <1 1 0 0 0 1> \quad \rightarrow \text{sim}(d5, q) = \log 5 + \log 1 + \log 0.2 \]
\[ + \log 5 + \log 5 + \log 5 \]

\[ \text{sim}(d, q) = \sum_{i \in q} d_i \log \frac{p_i}{1-p_i} + \sum_{i \in q} d_i \log \frac{1-q_i}{q_i} \]
**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:

\[ \sum_{i \in q} d_i \log \frac{p_i}{1 - p_i} = c \sum_{i \in q} d_i \quad \text{with constant } c \]

- $q_i = P[X_i = 1|\neg R] \approx \frac{df_i}{N}$
- $\frac{1 - q_i}{q_i} = \frac{N - df_i}{df_i} \approx \frac{N}{df_i}$

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

\[ \approx c \sum_{i \in q} d_i + \sum_{i \in q} d_i \cdot \log \text{idf}_i \]

*scalar product over the product of tf and dampened 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):

$$p_i = \frac{k+1}{n+2}$$

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

$$p_i = \frac{k+\lambda}{n+2\lambda} \text{ with } \lambda > 0 \text{ (e.g. } \lambda=0.5)$$

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

$$p_i = \frac{k_i + 1}{n + w}$$
Laplace Smoothing as Bayesian Parameter Estimation

\[ P[\text{param } \theta | \text{data } d] = \frac{P[d|\theta]P[\theta]}{P[d]} \]

**posterior** \hspace{1cm} **likelihood** \hspace{1cm} **prior**

consider:

binom(n,x) with observation \( k \)

assume:

\( \text{uniform}(x) \) as prior for \( \text{param } x \in [0,1] \)

\( f_{\text{uniform}}(x) = 1 \)

\[ P[x|k,n] = \frac{P[k,n|x]P[x]}{P[k,n]} \]

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

= \[ \frac{x^k(1-x)^{n-k}}{\int_0^1 x^k(1-x)^{n-k} \, dx} \]

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

\[ \text{posterior expectation} \]

= \[ \frac{B(k+2,n-k+1)}{B(k+1,n-k+1)} \]

= \[ \frac{\Gamma(k+2)\Gamma(n+2)}{\Gamma(n+3)\Gamma(k+1)} \]

= \[ \frac{(k+1)!(n+1)!}{(n+2)!k!} \]

= \[ \frac{k+1}{n+2} \]

with Beta function \( B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1} \, dt \)

and Gamma function \( \Gamma(z) = \int_0^\infty t^{z-1}e^{-t} \, dt \)

\( \Gamma(z+1) = z! \text{ for } z \in \mathbb{N} \)
13.2.2 Poisson Model

For generating doc $x$

- consider counting RVs: $x_w = \text{number of occurrences of } w \text{ in } x$
- still postulate independence among these RVs

Poisson model with word-specific parameters $\mu_w$:

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

MLE for $\mu_w$ 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:

\[ p_{tf} = e^{-\lambda} \frac{\lambda^{tf}}{tf!} \quad q_{tf} = e^{-\mu} \frac{\mu^{tf}}{tf!} \]

combined into 2-Poisson mixture

relevant docs     irrelevant docs     all docs
Okapi BM25 Scoring Function

Approximation of Poisson model by similarly-shaped function:

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

finally leads to Okapi BM25 weights:

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

or in the most comprehensive, tunable form: \[ w_j(d) = \]

\[ \log \frac{N - df_j + 0.5}{df_j + 0.5} \cdot \frac{(k_1 + 1)tf_j}{k_1((1-b) + b \frac{\text{len}(d)}{\Delta}) + tf_j} \cdot \frac{(k_3 + 1)qtf_j}{k_3 + tf_j} + k_2 \left| q \right| \frac{\Delta - \text{len}(d)}{\Delta + \text{len}(d)} \]

with \( \Delta = \text{avgdoclength} \) and tuning parameters \( k_1, k_2, k_3, b \),

**sub-linear influence of tf** (via \( k_1 \)), **consideration of doc length** (via \( b \))

BM25 performs very well has won many benchmark competitions (TREC etc.)
Poisson Mixtures for Capturing tf Distribution

Katz’s K-mixture:

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 $X_i$)
   → Problem of estimating m-dimensional prob. distribution
     \[ P[X_1 = \ldots \land X_2 = \ldots \land \ldots \land X_m = \ldots] =: f_X(X_1, \ldots, X_m) \]
     (curse of dimensionality)

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

a) Consider only 2-dimensional probabilities (for term pairs)
   \[ f_{ij}(X_i, X_j) = P[X_i = \ldots \land X_j = \ldots] = \sum \ldots \sum \sum \ldots \sum P[X_1 = \ldots \land \ldots \land X_m = \ldots] \]

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:

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

**Variant 2:**
Correlation coefficient for term pairs:

\[
\rho(X_i, X_j) := \frac{\text{Cov}(X_i, X_j)}{\sqrt{\text{Var}(X_i)} \sqrt{\text{Var}(X_j)}}
\]

**Variant 3:**
level-\( \alpha \) values or p-values of Chi-square independence test
Example for Approximation Error $\varepsilon$ 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 $f$:
    $f(1,1) = P[X_1=1 \land X_2=1] = 2/4$
    $f(0,0) = 1/4$, $f(0,1) = 1/4$, $f(1,0) = 0$
estimation of 1-dimensional marginal distributions $g_1$ and $g_2$:
    $g_1(1) = P[X_1=1] = 2/4$, $g_1(0) = 2/4$
    $g_2(1) = P[X_2=1] = 3/4$, $g_2(0) = 1/4$
estimation of 2-dim. distribution $g$ with independent $X_i$:
    $g(1,1) = g_1(1) \ast g_2(1) = 3/8,$
    $g(0,0) = 1/8$, $g(0,1) = 3/8$, $g(1,0) =1/8$
approximation error $\varepsilon$ (KL divergence):
    $\varepsilon = 2/4 \log 4/3 \ + \ 1/4 \log 2 \ + \ 1/4 \log 2/3 \ + \ 0$
Constructing the Term Dependence Tree

Given:
complete graph \((V, E)\) with \(m\) nodes \(X_i \in V\) and
\(m^2\) undirected edges \(e \in E\) with weights \(\varepsilon\) (or \(\rho\))

Wanted:
spanning tree \((V, E')\) with maximal sum of weights

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

Example: Web \(\xrightarrow{0.7}\) Surf
\(0.9\)
\(0.1\)
Internet \(\xrightarrow{0.5}\) Swim
\(0.3\)
}\n\(0.1\)

\(\rightarrow\)
Web \(\xrightarrow{0.9}\) Internet \(\xrightarrow{0.7} \) Surf
\(0.3\)
\(\rightarrow\)
\(\xrightarrow{0.1}\)
Swim
Estimation of Multidimensional Probabilities with Term Dependence Tree

Given is a term dependence tree \((V = \{X_1, ..., X_m\}, E')\).
Let \(X_1\) be the root, nodes are preorder-numbered, and assume that \(X_i\) and \(X_j\) are independent for \((i,j) \not\in E'\). Then:

\[
P[X_1 = .. \land .. \land X_m = ..] = P[X_1 = ..] P[X_2 = .. \land X_m = .. | X_1 = ..]
\]

\[
= \prod_{i=1..m} P[X_i = .. | X_1 = .. \land X (i-1) = ..]
\]

\[
= P[X_1] \cdot \prod_{(i,j) \in E'} P[X_j | X_i]
\]

\[
= P[X_1] \cdot \prod_{(i,j) \in E'} \frac{P[X_i, X_j]}{P[X_i]}
\]

Example:

\[
P[\text{Web}, \text{Internet}, \text{Surf}, \text{Swim}] = \frac{P[\text{Web}, \text{Internet}]}{P[\text{Web}]} \cdot \frac{P[\text{Web}, \text{Surf}]}{P[\text{Web}]} \cdot \frac{P[\text{Surf}, \text{Swim}]}{P[\text{Surf}]}\]
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 \(\text{parents}(X) = \{P_1, ..., P_k\}\)
  the BN captures the conditional probability \(P[X=... | P_1, ..., P_k]\).
- Node \(X\) is conditionally independent of a non-parent node \(Y\)
given its parents \(\text{parents}(X) = \{P_1, ..., P_k\}\):
  \[P[X | P_1, ..., P_k, Y] = P[X | P_1, ..., P_k].\]

This implies:
\[
P[X_1...X_n] = P[X_1|X_2...X_n] P[X_2...X_n]
\]

- by the chain rule:
  \[= \prod_{i=1}^{n} P[X_i|X(i+1)...X_n]\]
- by cond. independence:
  \[= \prod_{i=1}^{n} P[X_i|\text{parents}(X_i), \text{other nodes}]\]
  \[= \prod_{i=1}^{n} P[X_i|\text{parents}(X_i)]\]
Example of Bayesian Network

\[ P[C] : \]
\[
\begin{array}{l|cc}
\text{C} & P[C] & P[\neg C] \\
\hline
\text{F} & 0.5 & 0.5 \\
\text{T} & 0.1 & 0.9 \\
\end{array}
\]

\[ P[S | C] : \]
\[
\begin{array}{l|cc}
\text{C} & P[S] & P[\neg S] \\
\hline
\text{F} & 0.5 & 0.5 \\
\text{T} & 0.1 & 0.9 \\
\end{array}
\]

\[ P[R | C] : \]
\[
\begin{array}{l|cc}
\text{C} & P[R] & P[\neg R] \\
\hline
\text{F} & 0.2 & 0.8 \\
\text{T} & 0.8 & 0.2 \\
\end{array}
\]

\[ P[W | S,R] : \]
\[
\begin{array}{l|cc|cc}
\text{S} & \text{R} & P[W] & P[\neg W] \\
\hline
\text{F} & \text{F} & 0.0 & 1.0 \\
\text{F} & \text{T} & 0.9 & 0.1 \\
\text{T} & \text{F} & 0.9 & 0.1 \\
\text{T} & \text{T} & 0.99 & 0.01 \\
\end{array}
\]
Bayesian Inference Networks for IR

with binary random variables

\[ P[d_j] = \frac{1}{N} \]

\[ P[t_i \mid d_j \in \text{parents}(t_i)] = \begin{cases} 1 & \text{if } t_i \text{ occurs in } d_j, \\ 0 & \text{otherwise} \end{cases} \]

\[ P[q \mid \text{parents}(q)] = \begin{cases} 1 & \text{if } \exists t \in \text{parents}(q): t \text{ is relevant for } q, \\ 0 & \text{otherwise} \end{cases} \]

\[
P[q \land d_j] = \sum_{t_1 \ldots t_M} P[q \land d_j \mid t_1 \ldots t_M] P[t_1 \ldots t_M]
\]

\[
= \sum_{t_1 \ldots t_M} P[q \land d_j \land t_1 \land \ldots \land t_M]
\]

\[
= \sum_{t_1 \ldots t_M} P[q \mid d_j \land t_1 \land \ldots \land t_M] P[d_j \land t_1 \land \ldots \land t_M]
\]

\[
= \sum_{t_1 \ldots t_M} P[q \mid t_1 \land \ldots \land t_M] P[t_1 \land \ldots \land t_M \mid d_j] P[d_j]
\]
Advanced Bayesian Network for IR

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

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

\[ P[ck|ti, tl] = \frac{P[ti \land tl]}{P[ti \lor tl]} \approx \frac{df_{il}}{df_i + df_l - df_{il}} \]
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


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


K.W. Church, W.A. Gale: Poisson Mixtures, Natural Language Engineering 1(2), 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: $P[\text{"Today is Tuesday"}] = 0.001$
$P[\text{"Today Wednesday is"}] = 0.00000000001$
$P[\text{"The Eigenvalue is positive"}] = 0.000001$

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]

\[ \hat{X} = \arg \max_x P[X | Y] = \arg \max_x P[Y | X] \cdot P[X] \]

\( X \) 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 $\theta$: $P[\text{word} \mid \theta] \xrightarrow{\text{sample}} \text{document } d$

LM for topic 1: IR&DM

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

different $\theta_d$ for different $d$

LM for topic 2: Health

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

may also define LMs over n-grams

food

nutrition

paper

paper

IRDM WS 2015
LMs for Ranked Retrieval

parameter estimation

\[
\cdots
\text{text} \ ?
\text{mining} \ ?
n\text{-gram} \ ?
\text{cluster} \ ?
\cdots
\text{food} \ ?
\]

\text{LM(doc1)}

Which \text{LM} is more likely to generate \text{q}? (better explains \text{q})

\text{query q: data mining algorithms}

\text{LM(doc2)}
**LM Parameter Estimation**

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

**Query q:**

data mining algorithms

- LM(doc1)
- LM(doc2)
LM Illustration: Document as Model and Query as Sample

model M
A A A A
B B
C C C C
D
E E E E E E

document d: sample of M used for parameter estimation

estimate likelihood of observing query

\[ P \left[ A A B C E E \mid M \right] \]
LM Illustration: Need for Smoothing

model M

A A A A
B B
C C C C
D
E E E E E

estimate likelihood of observing query

P [ A B C E F | M]

document d + background corpus and/or smoothing

used for parameter estimation
## Probabilistic IR vs. Language Models

User considers doc relevant given that it has features $d$ and user has posed query $q$.

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

Probabilistic IR ranks according to relevance odds.

Statistical LMs rank according to query likelihood.

\[
\begin{align*}
    \sim & \quad \frac{P[q,d|R]}{P[q,d|\bar{R}]} \\
    = & \quad \frac{P[q|R,d]}{P[q|\bar{R},d]} \cdot \frac{P[R|d]}{P[\bar{R}|d]} = \ldots \sim \ldots \\
    \sim & \quad P[q|R,d]
\end{align*}
\]
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

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

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

\[ = \prod_{t \in q} P[t \mid d] \sim \sum_{t \in q} \log P[t \mid d] \]

Parameters \( \theta \) of LM(d) are \( P[t \mid d] \)
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 → can capture relevance feedback and user context (relative importance of terms)

\[
P[q \mid d] = \left( \frac{|q|}{f(t_1) f(t_2) \ldots f(t_{|q|})} \right) \prod_{t \in q} P_t(d)^{f_t(q)}
\]

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

Parameters \(\theta\) of LM(d) are \(P[t \mid d]\) and \(P[t \mid q]\)

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

$$\log_2 P[q \mid d] = \log_2 \left( \left| q \right| \prod_{j \in q} p_j(d)^{f_j(q)} \right)$$

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

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

neg. cross-entropy

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

$$= -D(f(q) \parallel p(d))$$

makes query LM explicit

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

neg. KL divergence of $\theta_q$ and $\theta_d$
Smoothing Methods

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

possible methods:
• Laplace smoothing
• Absolute Discounting
• 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)
estimation of $\theta_d$: $p_j(d)$ 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 $\sigma = \frac{\delta \cdot \#\text{distinct terms in } d}{|d|}$
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 \( \lambda \) 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 \( \lambda \) 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 \( \sum_w \theta_w = 1 \) and \( \theta_w \geq 0 \) and \( \alpha_w \geq 0 \) for all \( w \)

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

(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 $\theta$
- choose statistical model (**generative model**) $f(x | \theta)$
  that reflects our beliefs about RV $X$
- given RVs $X_1, ..., X_n$ for observed data,
  the **posterior distribution** is $h(\theta | x_1, ..., x_n)$

for $X_1 = x_1, ..., X_n = x_n$ the likelihood is

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

which implies

$$h(\theta | x_1...x_n) \sim L(x_1...x_n, \theta) \cdot g(\theta) \quad \text{(posterior is proportional to likelihood times prior)}$$

**MAP estimator** (**maximum a posteriori**):
compute $\theta$ that maximizes $h(\theta | x_1, ..., x_n)$ given a prior for $\theta$
Dirichlet-Prior Smoothing

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

\[ = \text{Dirichlet}(x + \alpha) \]

Posterior for \( \theta \) with Dirichlet distribution as prior with term frequencies \( x \) in document \( d \)

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

with \( \alpha_i \) set to \( \mu \cdot P[i \mid C] + 1 \) for the Dirichlet hypergenerator and \( \mu > 1 \) set to multiple of average document length

Dirichlet (\( \alpha \)):

\[ f(\theta_1, \ldots, \theta_m) = \frac{\Pi_{j=1..m} \Gamma(\alpha_j)}{\Gamma(\Sigma_{j=1..m} \alpha_j)} \Pi_{j=1..m} \theta_j^{\alpha_j-1} \]

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)

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

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

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 \( \mu \) 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)

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

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

setting \( \lambda = \frac{|d|}{|d| + \mu} \)

Can also integrate \( P[j \mid R] \) with relevance feedback LM
or \( P[j \mid U] \) 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
- Explain unseen words
- Dirichlet prior (Bayesian)

Stage-2
- Explain noise in query
- 2-component mixture

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

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 → $P[t_i \mid d] \sim P[t_i \mid t_{i-1}] P[t_{i-1} \mid d]$ …

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 \( P[j \mid w] \)

Opportunities and difficulties:
• synonymy, hypernymy/hyponymy, etc.
• efficiency
• training

estimate \( P[j \mid w] \) 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 \( P[j \mid w] \) 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:

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

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

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

\[
P[w \mid \theta_k] = \alpha P[w \mid q_k] + (1 - \alpha) P[w \mid H_q, H_c]
\]
LM for Temporal Search  [K. Berberich et al.: ECIR 2010]

keyword queries that express temporal interest
example: \( q = \) ”FIFA world cup 1990s“
→ 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

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

\[
P[time(q)|time(d)] = \prod_{\text{temp} \in q} \sum_{\text{temp} \in d} P[x|y]
\]

\[
P[x|y] \sim \frac{|x \cap y|}{|x| \cdot |y|}
\]
with \( |x| = \text{end}(x) - \text{begin}(x) \)
Entity Search with LM  

[Nie et al.: WWW’07]

Assume entities marked in docs by information extraction methods

**query: keywords → answer: entities**

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

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

**docs weighted by extraction confidence**

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
Language Models for Question Answering (QA)

**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

**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.

More on QA in Chapter 16 of this course
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.
Statistical Language Models in General:
Additional Literature for Section 13.3

LMs for Specific Retrieval Tasks:

- 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
- C. Clarke, E.L. Terra: Passage retrieval vs. document retrieval for factoid question answering. SIGIR 2003
- H. Zaragoza et al.: Ranking very many typed entities on wikipedia. CIKM 2007
- 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