Statistical Language Models

Intelligent Information Retrieval

What is a Statistical LM?

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

Today is Wednesday

Today Wednesday is

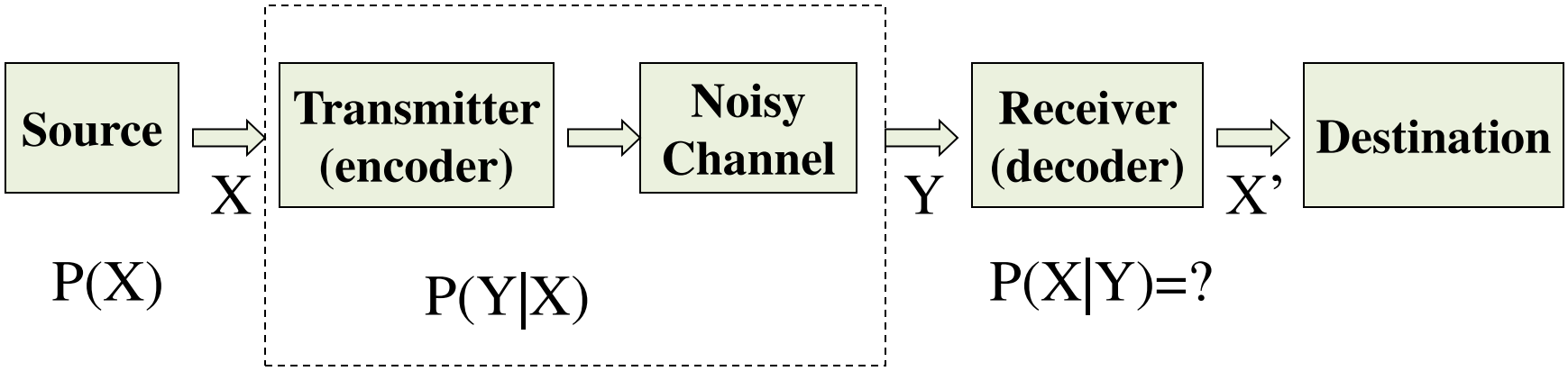
...

The eigenvalue is positive

Why is a LM Useful?

- Provides a principled way to quantify the uncertainties associated with natural language
- Allows us to answer questions like:
  - Given that we see "*John*" and "*feels*", how likely will we see "*happy*" as opposed to "*habit*" as the next word?
    (speech recognition)
  - Given that we observe "baseball" three times and "game" once in a news article, how likely is it about "sports"?
    (text categorization, information retrieval)
  - Given that a user is interested in sports news, how likely would the user use "baseball" in a query?
    (information retrieval)

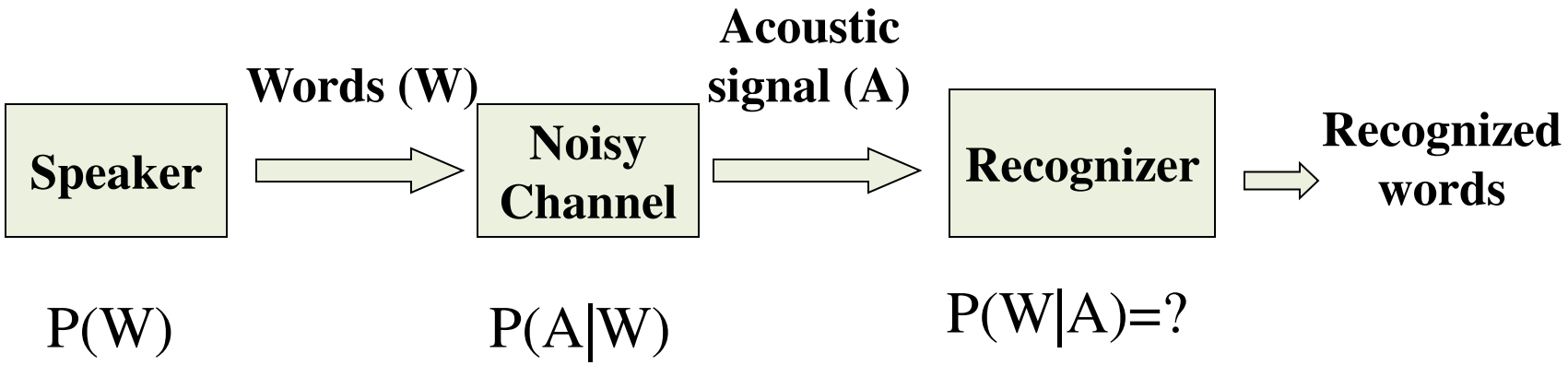
Source-Channel Framework
(Communication System)



$$\hat{X} = \arg\max_{X} p(X \mid Y) = \arg\max_{X} p(Y \mid X)p(X)$$

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

Speech Recognition



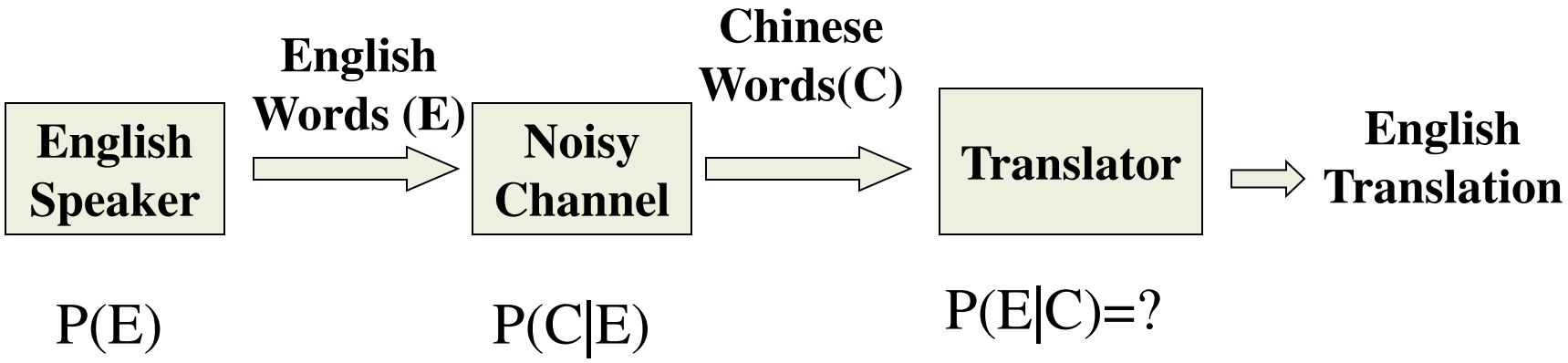
$$\hat{W} = \arg\max_{W} p(W \mid A) = \arg\max_{W} p(A \mid W)\, p(W)$$

Acoustic model

Language model

Given acoustic signal A, find the word sequence W

Machine Translation



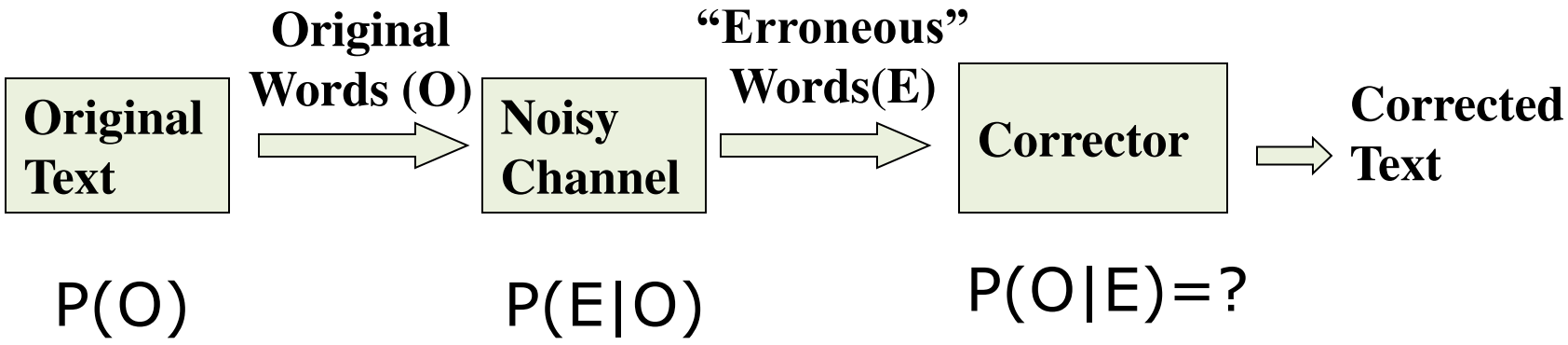
$$\hat{E} = \arg\max_{E} p(E \mid C) = \arg\max_{E} p(C \mid E)\, p(E)$$

English->Chinese Translation model

English Language model

Given Chinese sentence C, find its English translation E

Spelling/OCR Error Correction



$$\hat{O} = \arg\max_{o} p(O \mid E) = \arg\max_{o} p(E \mid O)p(O)$$

Spelling/OCR Error model

"Normal" Language model

Given corrupted text E, find the original text O

Basic Issues

- Define the probabilistic model
  - Event, Random Variables, Joint/Conditional Prob's
  - $P(w_1 w_2 \ \ldots \ w_n) = f(\ \ _1, \theta_2, \ldots, \theta_m)$
- Estimate model parameters
  - Tune the model to best fit the data and our prior knowledge
  - $\theta_i = ?$
- Apply the model to a particular task
  - Many applications

The Simplest Language Model
(Unigram Model)

$$p(t_1\ t_2\ t_3\ t_4) = p(t_1)p(t_2|t_1)p(t_3|t_1\ t_2)p(t_4|t_1\ t_2\ t_3)$$

$$p_{uni}(t_1\ t_2\ t_3\ t_4) = p(t_1)p(t_2)p(t_3)p(t_4)$$

$$p_{bi}(t_1\ t_2\ t_3\ t_4) = p(t_1)p(t_2|t_1)p(t_3|t_2)p(t_4|t_3)$$

- A piece of text can be regarded as a sample drawn according to this word distribution

Wednesday

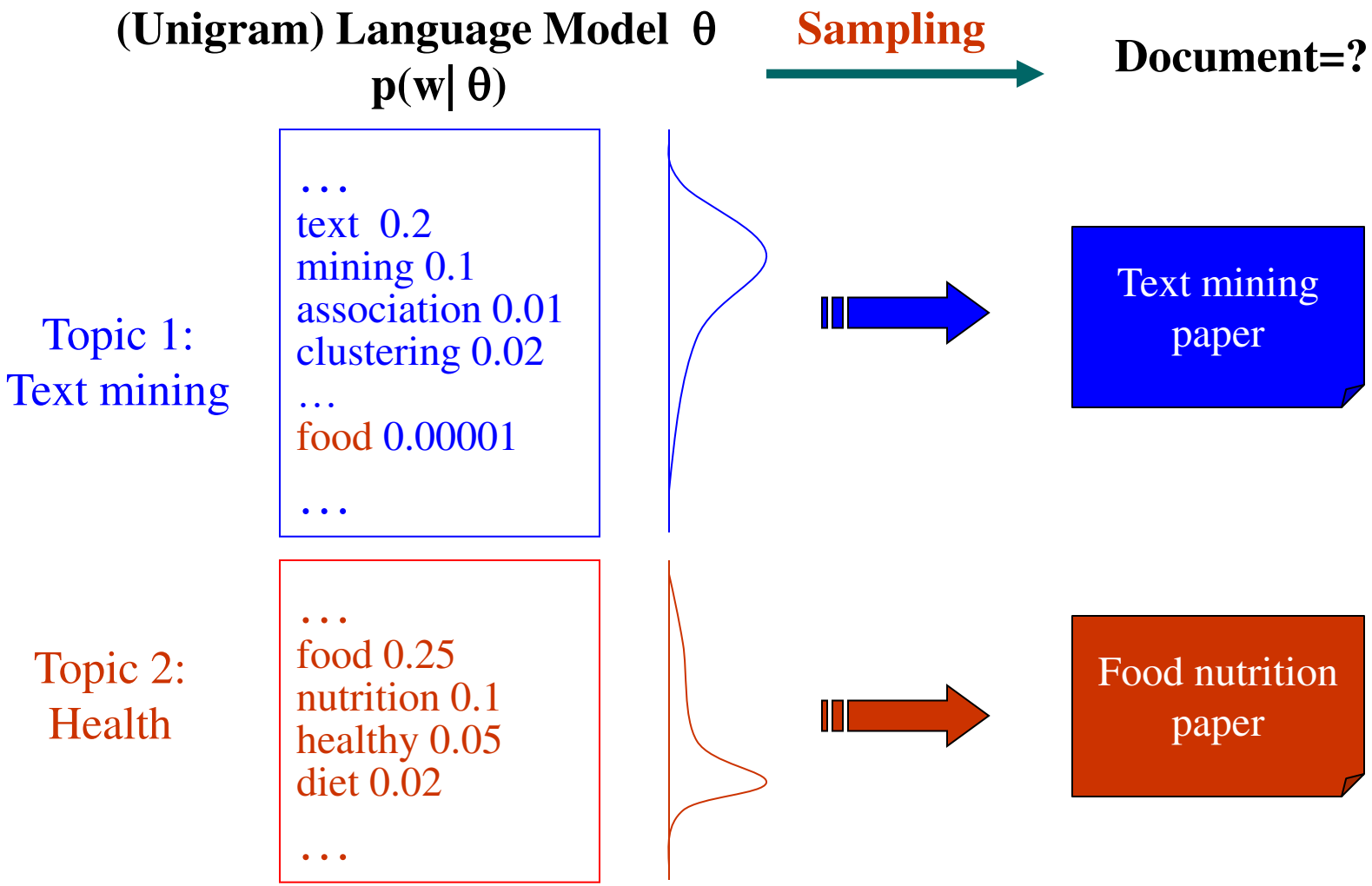
today

…

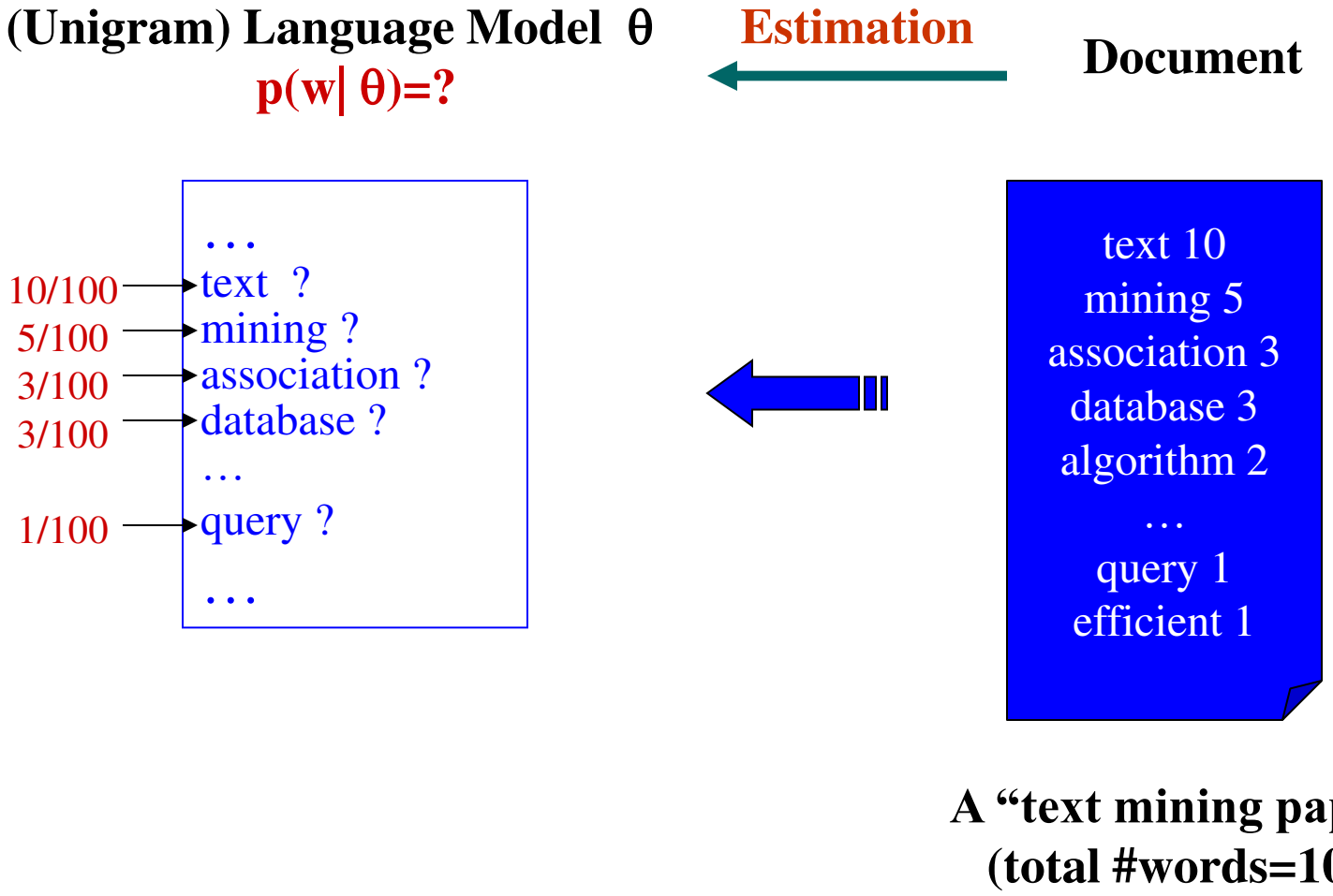
eigenvalue

P("today is Wednesday")
= P("today)P("is")P("Wednesday")
= 0.0002 × 0.001 × 0.000015

Text Generation with Unigram LM



Estimation of Unigram LM



Maximum Likelihood Estimate

Data: a document d with counts $c(w_1), \ldots, c(w_N)$, and length $|d|$

Model: multinomial (unigram) M with parameters $\{p(w_i)\}$

Likelihood: $p(d|M)$

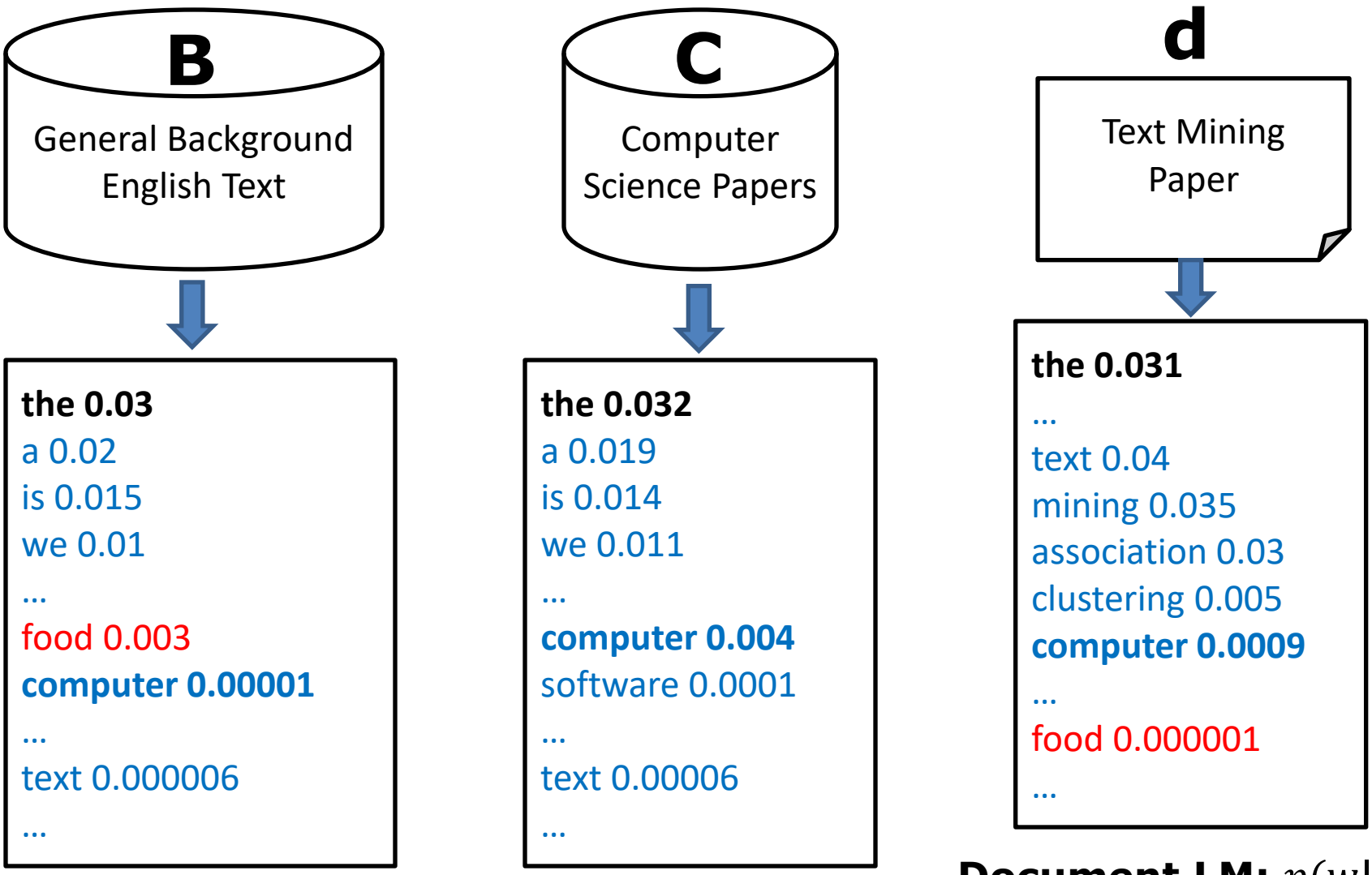
Maximum likelihood estimator: $M = argmax_M\, p(d|M)$

It can be analytically shown that

$$\theta_i = p(w_i) = \frac{c(w_i)}{|d|}$$

ML estimate

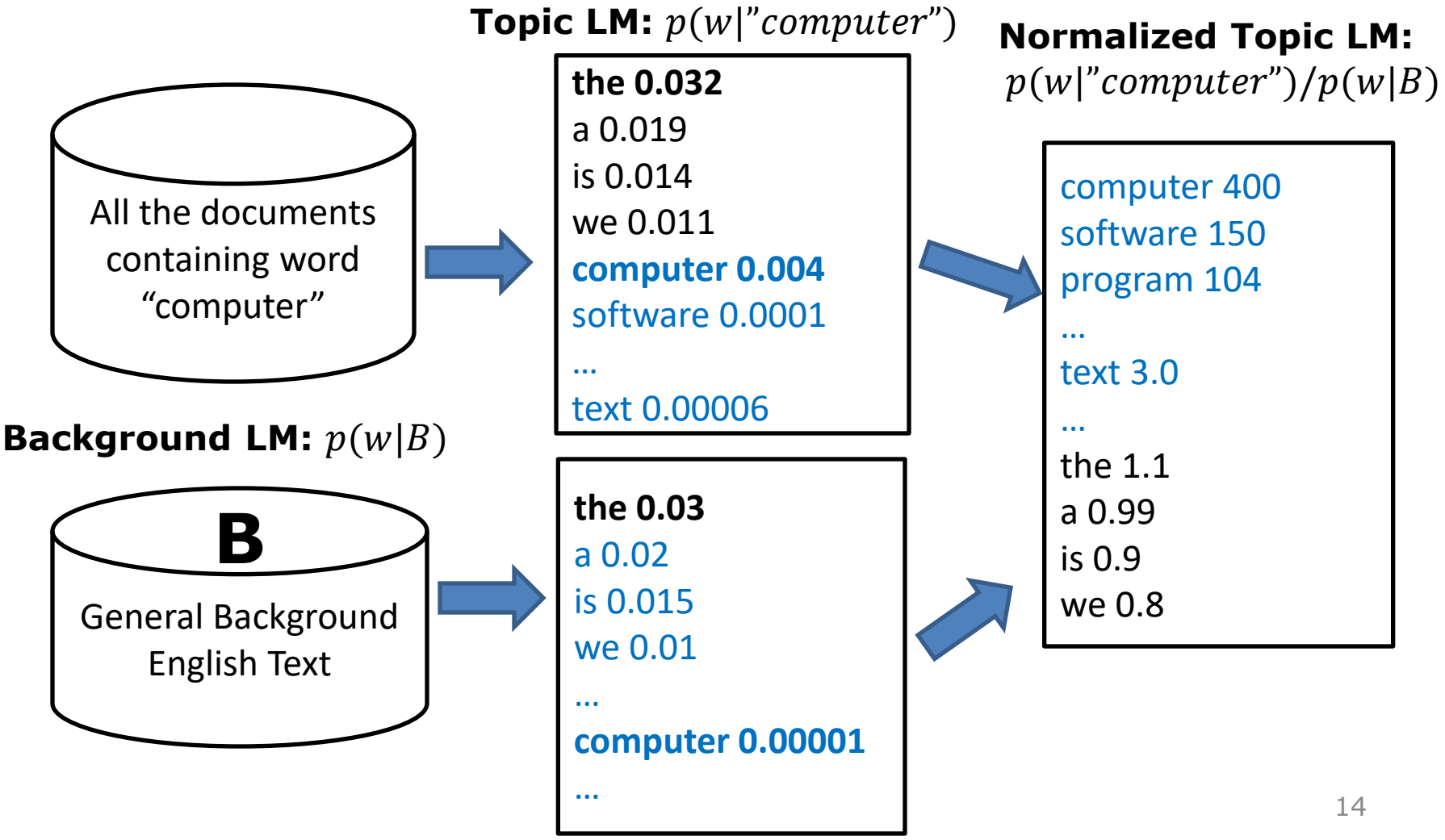
LMs for Topic Representation



Background LM: $p(w|B)$ **Collection LM:** $p(w|C)$ **Document LM:** $p(w|d)$

LMs for Association Analysis

What words are semantically related to "computer"?

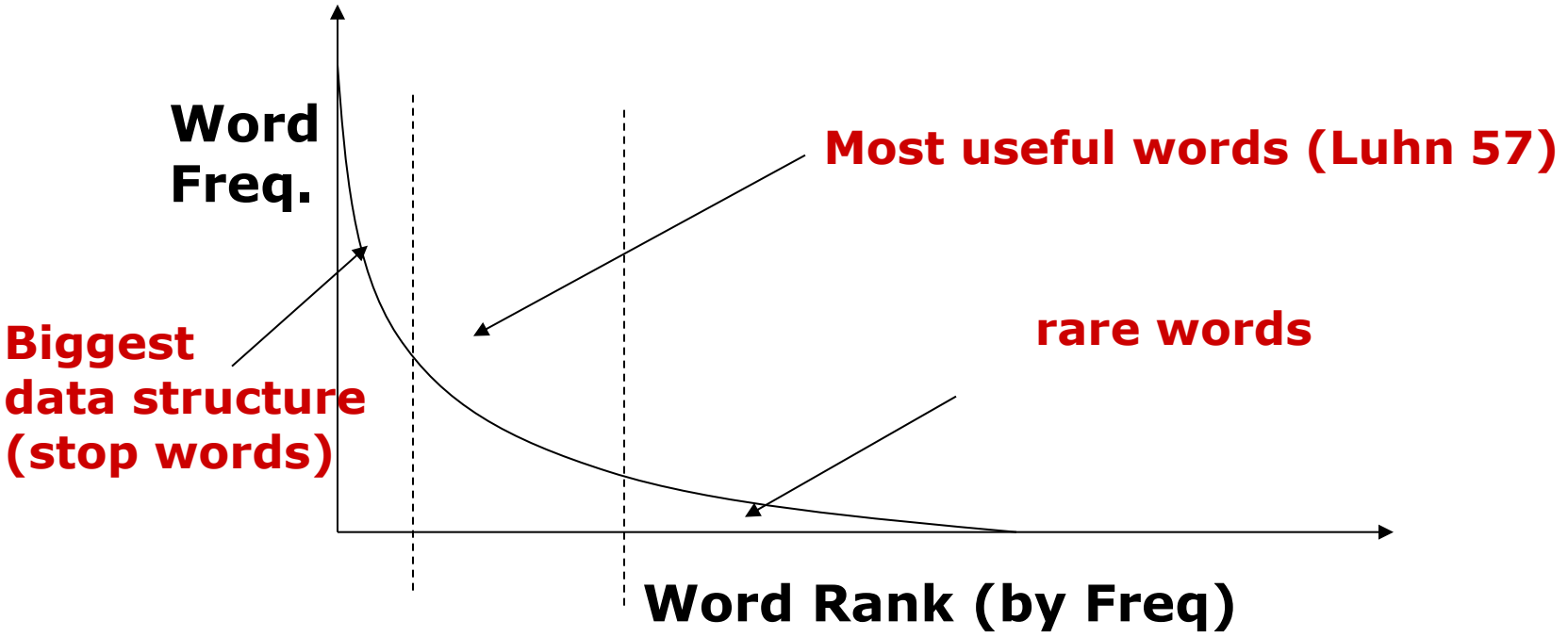


Empirical distribution of words

- There are stable language-independent patterns in how people use natural languages
- A few words occur very frequently; most occur rarely. E.g., in news articles,
  - Top 4 words: 10~15% word occurrences
  - Top 50 words: 35~40% word occurrences
- The most frequent word in one corpus may be rare in another

Zipf's Law

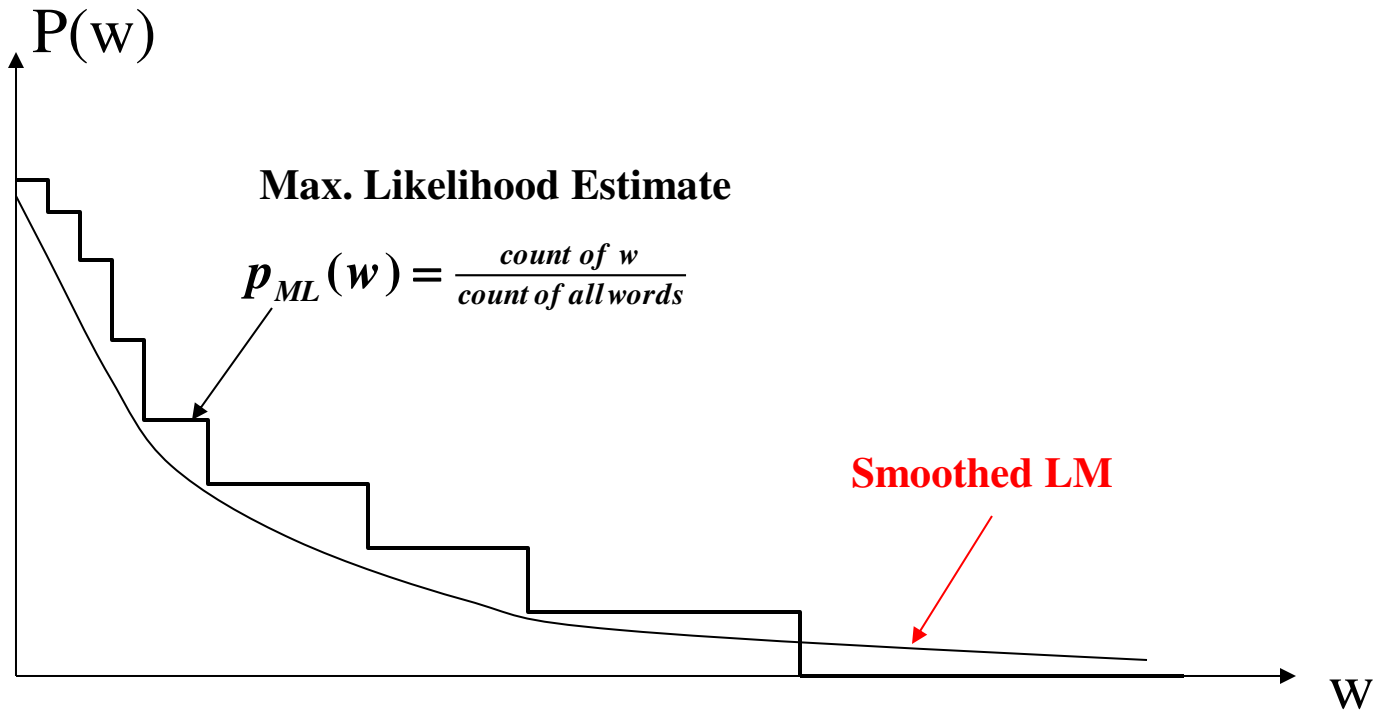
- rank * frequency ≈ constant



Problem with the ML Estimator

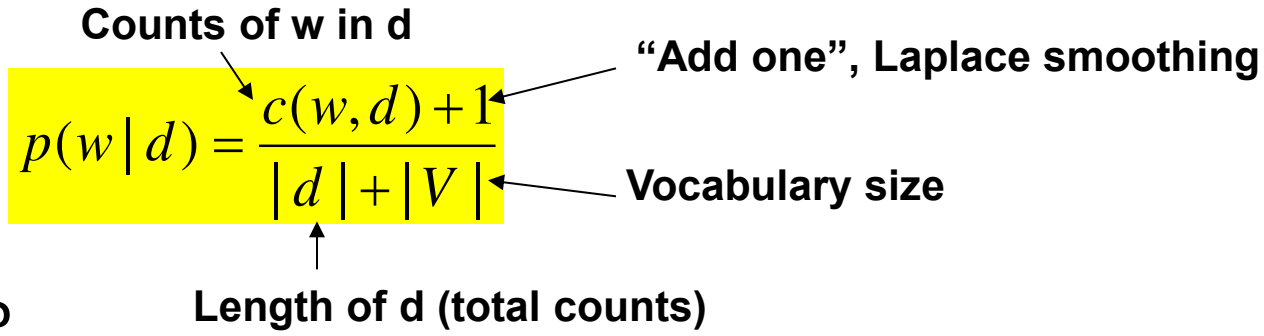
- What if a word doesn't appear in the text?
- In general, what probability should we give a word that has not been observed?
- If we want to assign non-zero probabilities to such words, we'll have to discount the probabilities of observed words
- This is what "smoothing" is about …

Language Model Smoothing (Illustration)



How to Smooth?

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



- Problems?

How to Smooth? (cont.)

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

Discounted ML estimate

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

Reference language model

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

How to Estimate P(w|C)?

- $P(w|C) = \frac{\sum_{D \in C} c(w,D)}{\sum_{D \in C} |D|}$
- $P(w|C) = \frac{1}{|C|} \sum_{D \in C} \frac{c(w,D)}{|D|}$
- $P(w|C) = \frac{\sum_{D \in C} \delta(w,D)}{\sum_{w \in V} \sum_{D \in C} \delta(w,D)}$

Other Smoothing Methods

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

unique words in document d

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

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

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

ML estimate

parameter

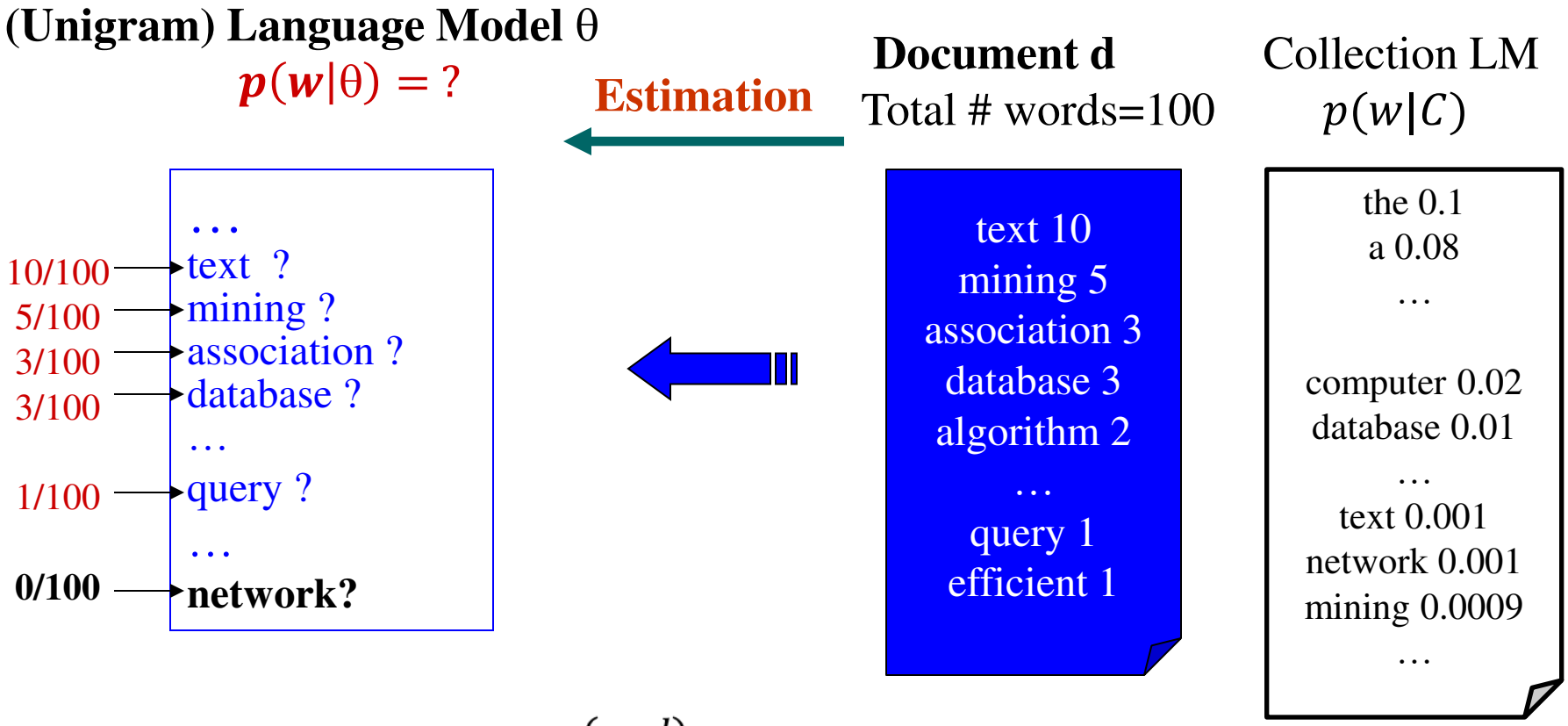
Other Smoothing Methods (cont.)

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

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

parameter

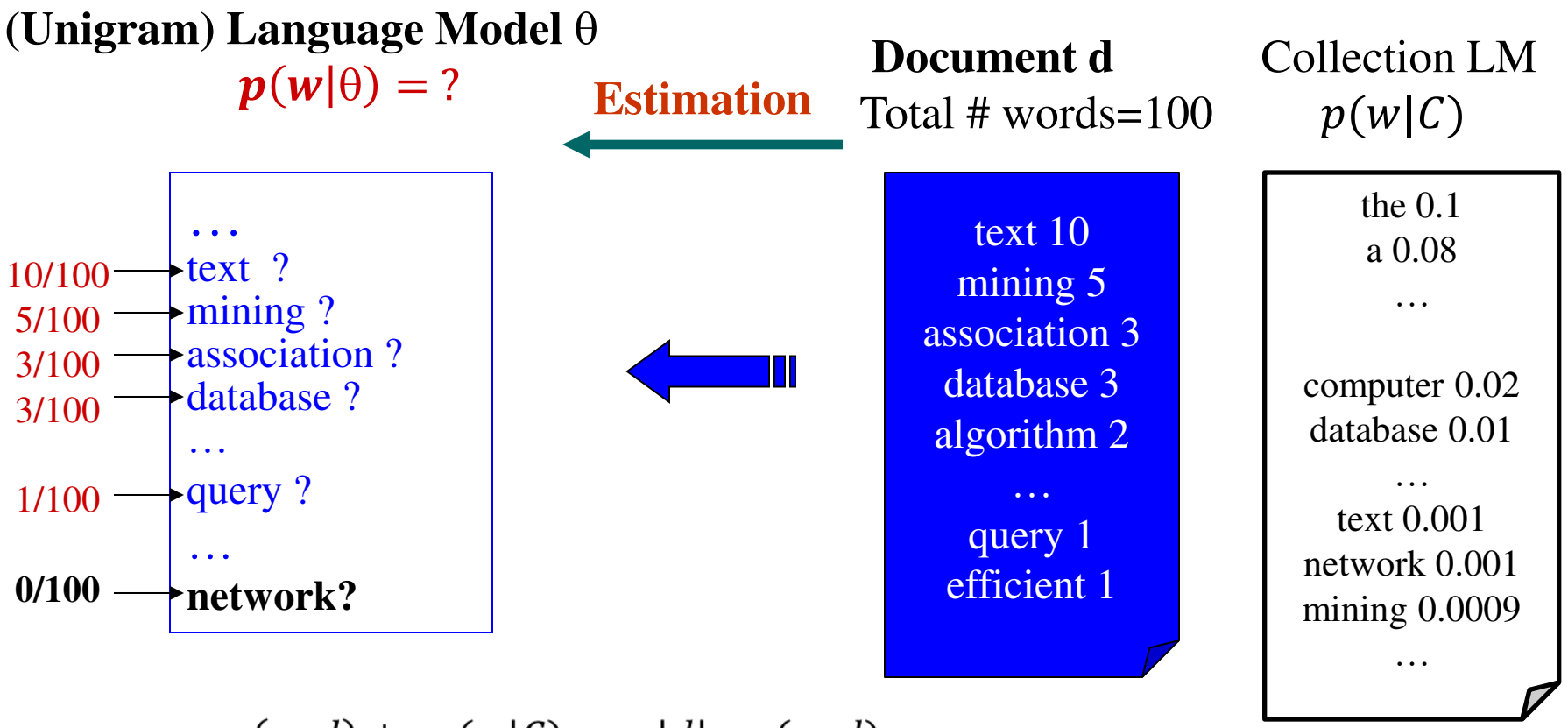
Linear Interpolation (Jelinek-Mercer) Smoothing



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

$$p("text"|d) = (1-\lambda)\frac{10}{100} + \lambda * 0.001 \qquad p("network"|d) = \lambda * 0.001$$

Dirichlet Prior (Bayesian) Smoothing



$$p(w|d) = \frac{c(w,d) + \mu p(w|C)}{|d| + \mu} = \frac{|d|}{|d| + \mu}\frac{c(w,d)}{|d|} + \frac{\mu}{|d| + \mu}p(w|C) \qquad \mu \in [0, +\infty)$$

$$p("text"|d) = \frac{10 + \mu * 0.001}{100 + \mu} \qquad p("network"|d) = \frac{\mu}{100 + \mu} * 0.001$$

So, which method is the best?

It depends on the data and the task!

Many other sophisticated smoothing methods have been proposed…

Cross validation is generally used to choose the best method and/or set the smoothing parameters…

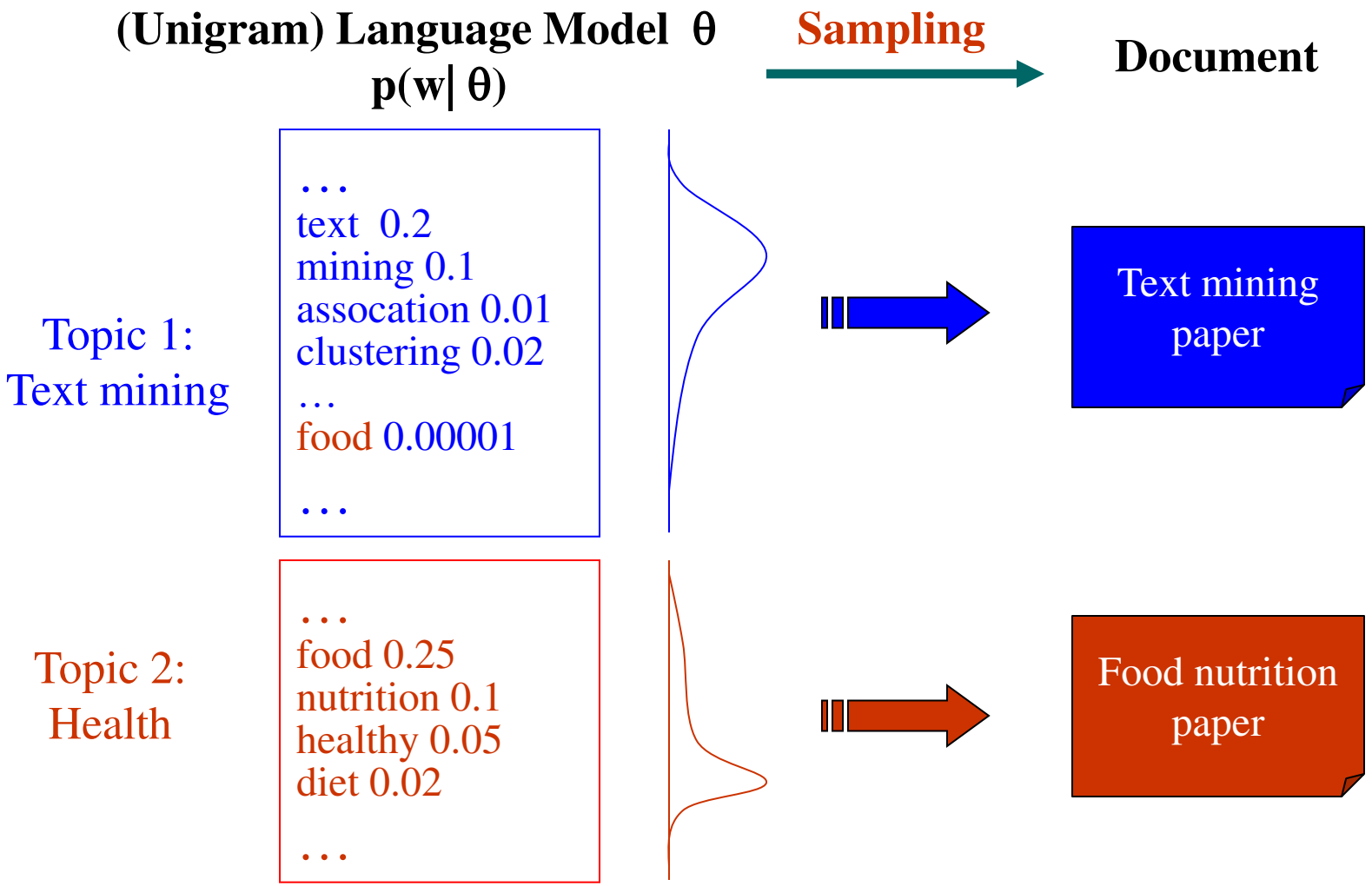
For retrieval, Dirichlet prior performs well…

Smoothing will be discussed further in the course…

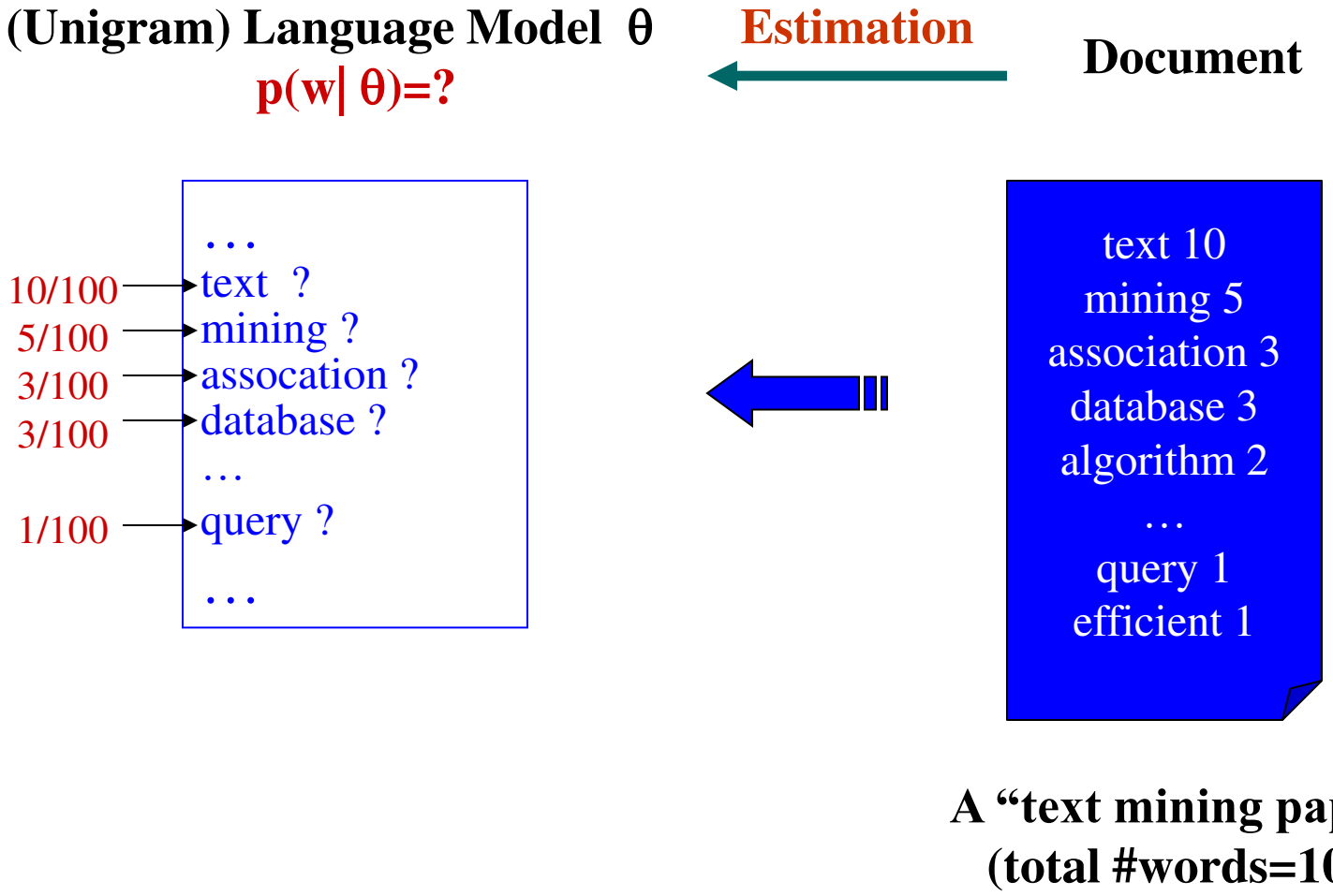
Questions?

Language Models for TR

Text Generation with Unigram LM



Estimation of Unigram LM

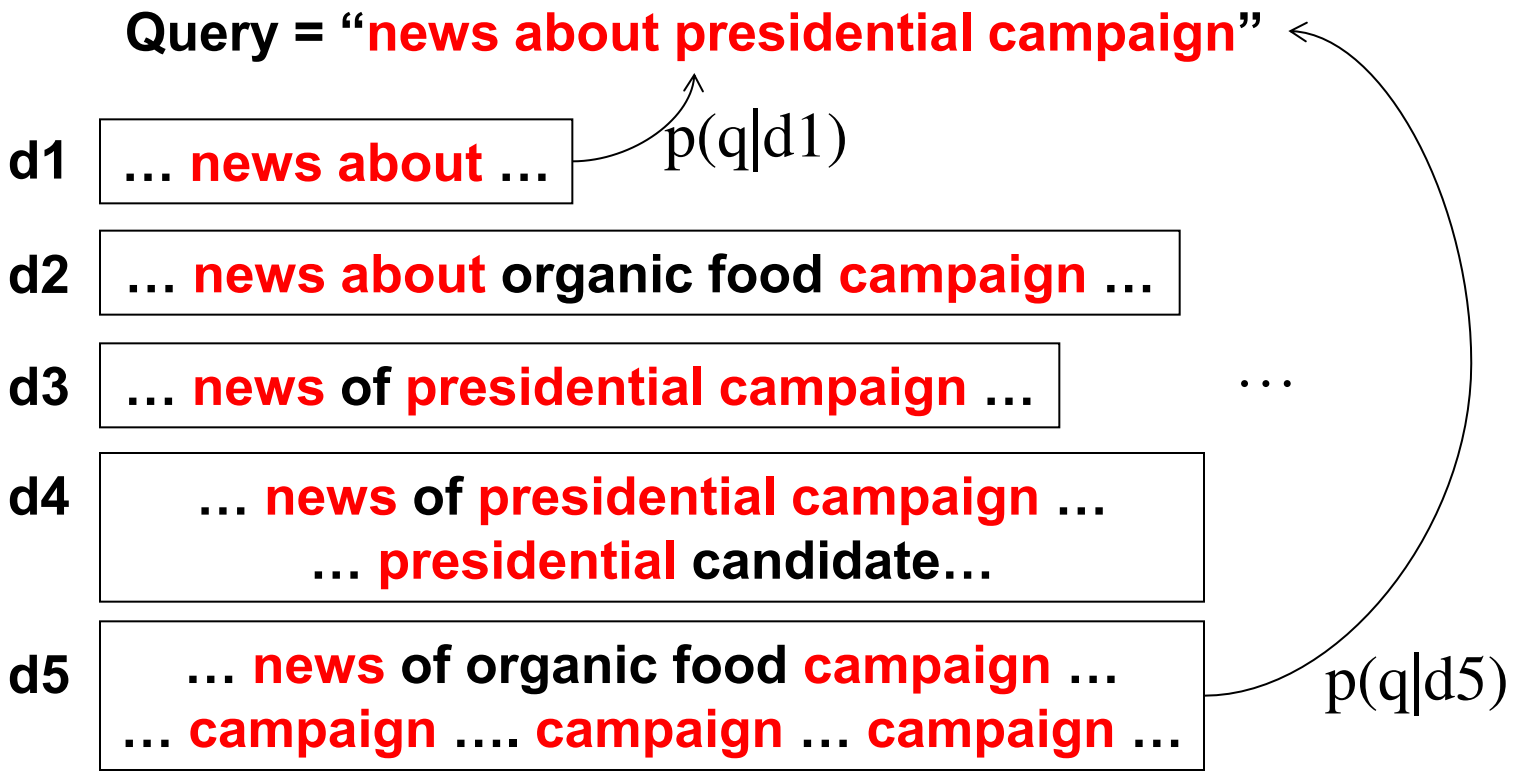


Query Likelihood Retrieval Model

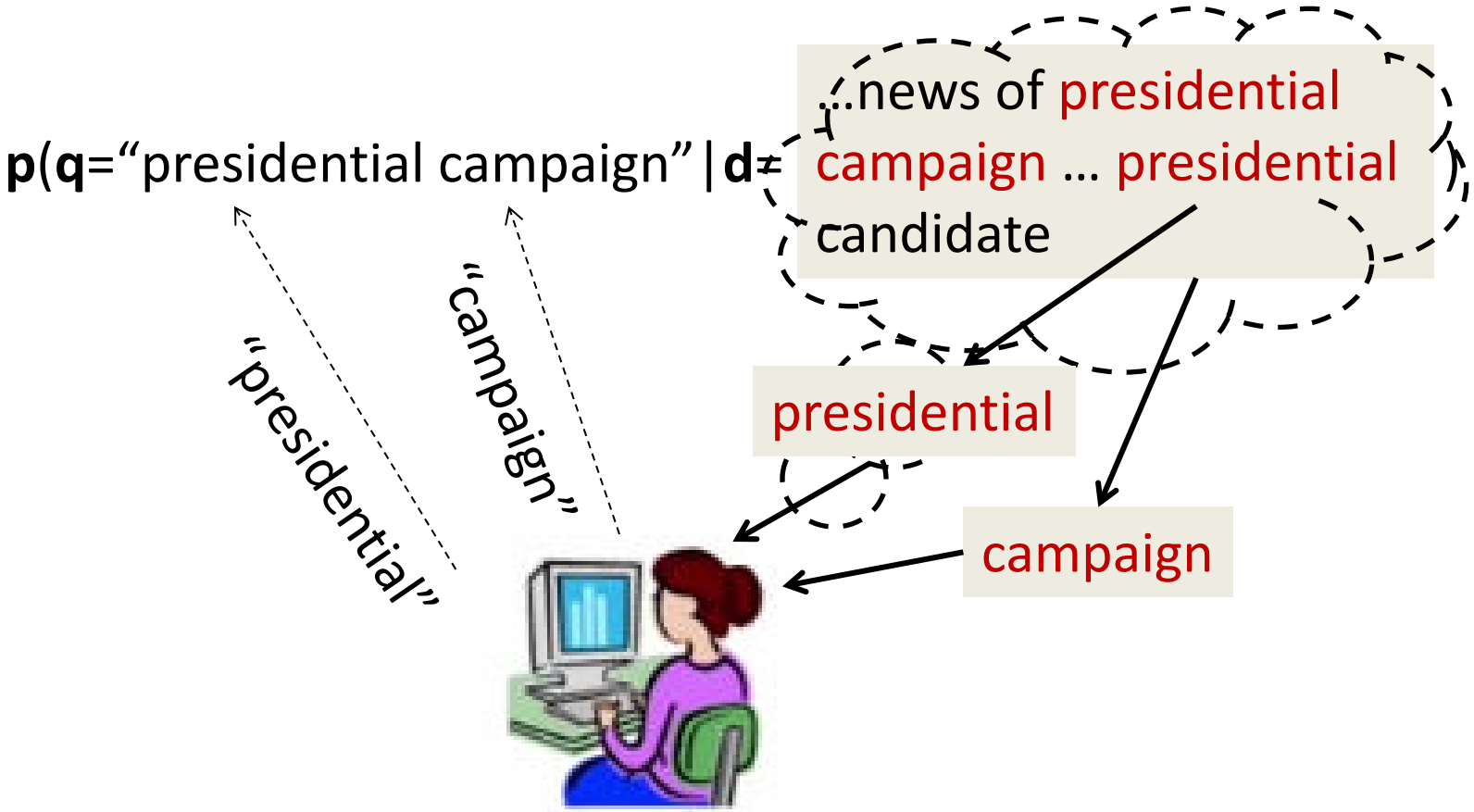
$$f(q,d) = p(q|d, R = 1)$$

Assumption:
A user formulates a query based on an
"**imaginary relevant document**"

Which Doc is Most Likely the "Imaginary Relevant Doc"?



Query Generation by Sampling Words from Doc



If the user is **thinking of this doc**,
how likely would she **pose this query**?

Unigram Query Likelihood

$$p(q = "presidential\ campaign"|d = \text{...news of presidential campaign ... presidential candidate})$$

$$= p("\text{presidential}"|d) * p("campaign"|d)$$

$$= \frac{c(\text{presidential}, d)}{|d|} * \frac{c(\text{campaign}, d)}{|d|}$$

Assumption:
Each query word is generated independently

Does Query Likelihood Make Sense?

$$p(q = \text{"presidential campaign"}|d) = \frac{c(\text{"presidential"}, d)}{|d|} \times \frac{c(\text{"campaign"}, d)}{|d|}$$

$p(q|d4 =$ … news of **presidential campaign** **… presidential** candidate … $) = \frac{2}{|d4|} * \frac{1}{|d4|}$

$p(q|d3 =$ … news of **presidential campaign** … $) = \frac{1}{|d3|} * \frac{1}{|d3|}$

$p(q|d2 =$ … news about organic food **campaign …** $) = \frac{0}{|d2|} * \frac{1}{|d2|} = 0$

d4 > d3 > d2 as we expected

Try a Different Query?

q = "presidential campaign update"

$p(q|d4 =$ … news of **presidential campaign** **… presidentia**l candidate … $) = \frac{2}{|d4|} * \frac{1}{|d4|} * \frac{0}{|d4|} = 0$

$p(q|d3 =$ … news of **presidential campaign** … $) = \frac{1}{|d3|} * \frac{1}{|d3|} * \frac{0}{|d3|} = 0$

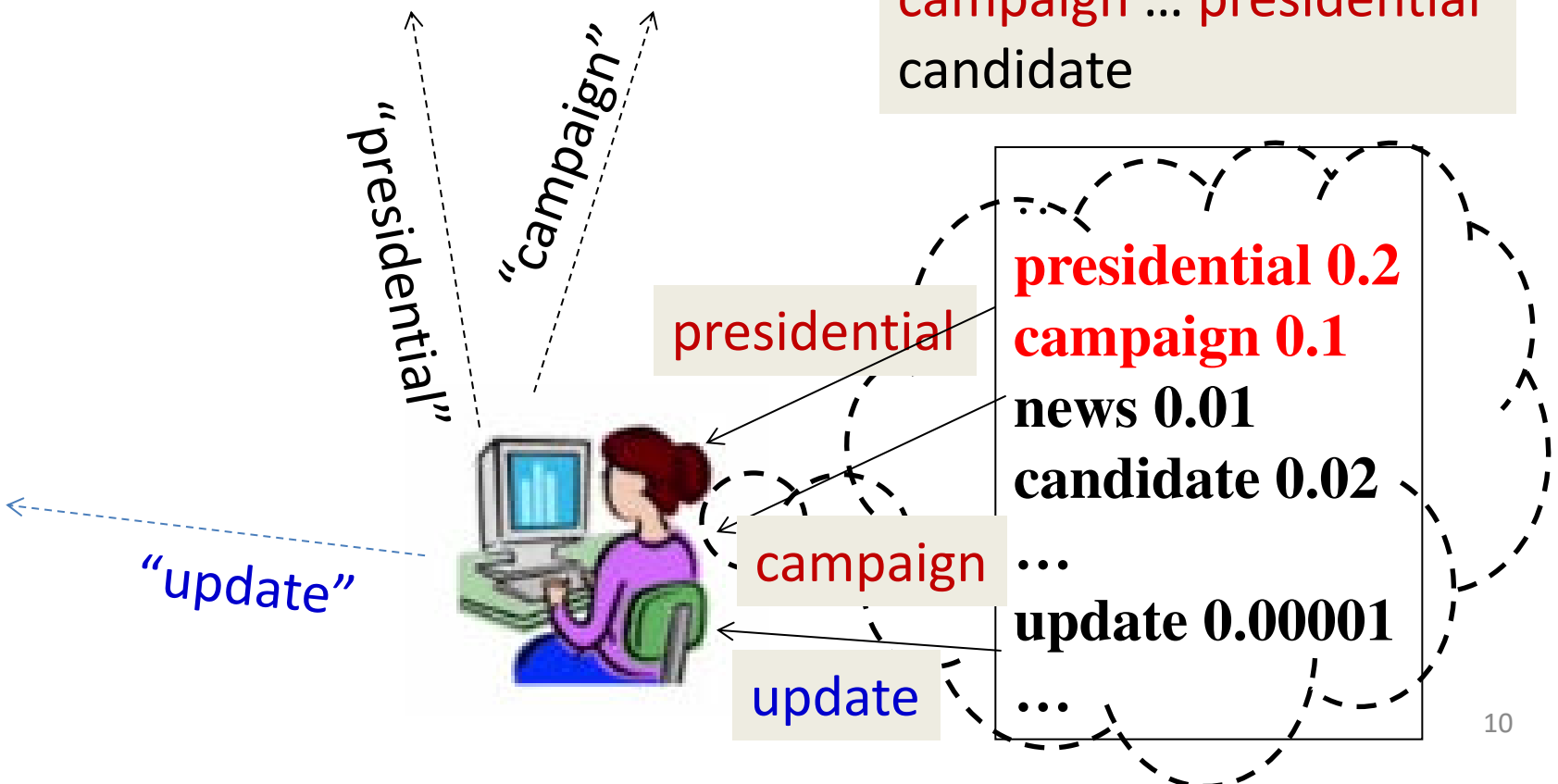
$p(q|d2 =$ … news about organic food **campaign …** $) = \frac{0}{|d2|} * \frac{1}{|d2|} * \frac{0}{|d2|} = 0$

What assumption has caused this problem? How do we fix it?

Improved Model: Sampling Words from a Doc Model

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

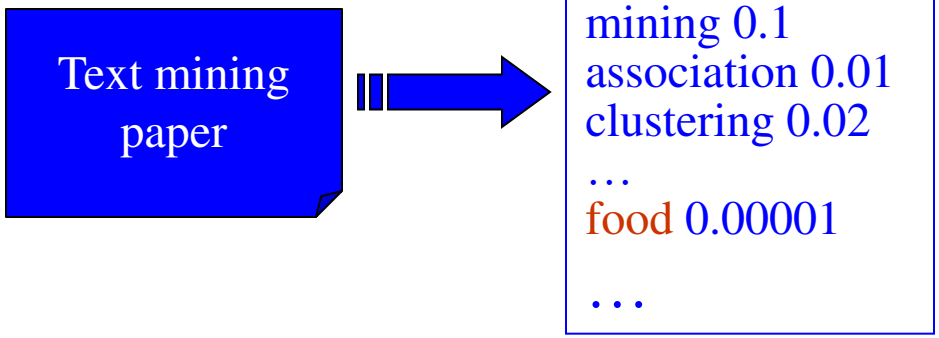
p(**q**="presidential campaign"|**d**= …news of presidential campaign … presidential candidate)



Computation of Query Likelihood

Document

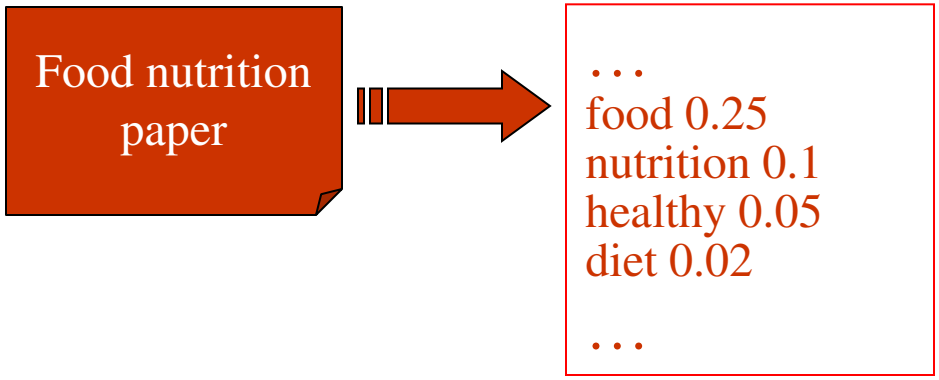
Language Model
p(w|d1)



Query =
"data mining algorithms"

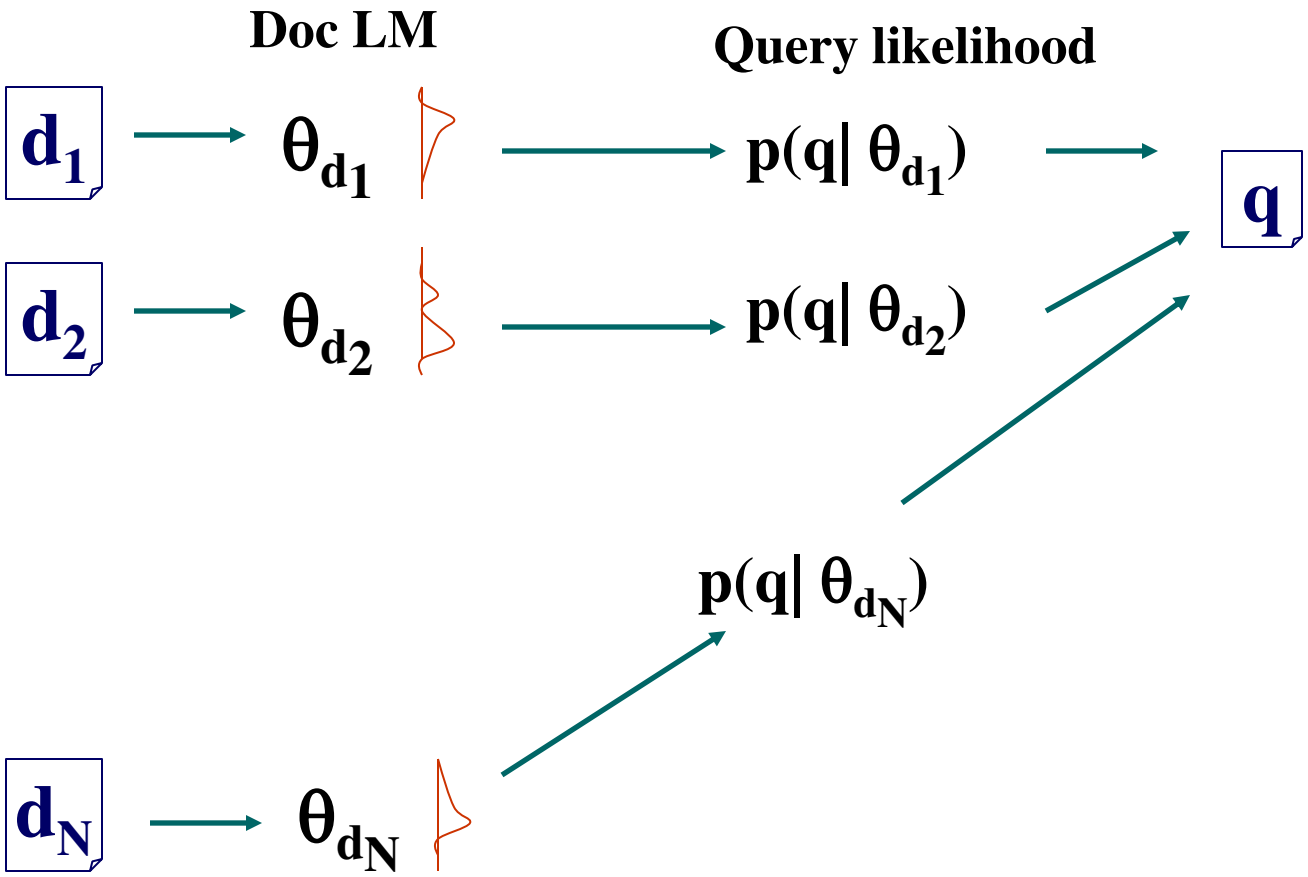
$$p('data\ mining\ alg'|d1) = p('data'|d1) \times p('mining'|d1) \times p('alg'|d1)$$

p(w|d2)



$$p('data\ mining\ alg'|d2) = p('data'|d2) \times p('mining'|d2) \times p('alg'|d2)$$

Ranking Docs by Query Likelihood



Retrieval as Language Model Estimation

- Document ranking based on *query likelihood*

$$\log p(q \mid d) = \sum_i \log p(w_i \mid d)$$

$$where,\ q = w_1 w_2 \ldots w_n$$

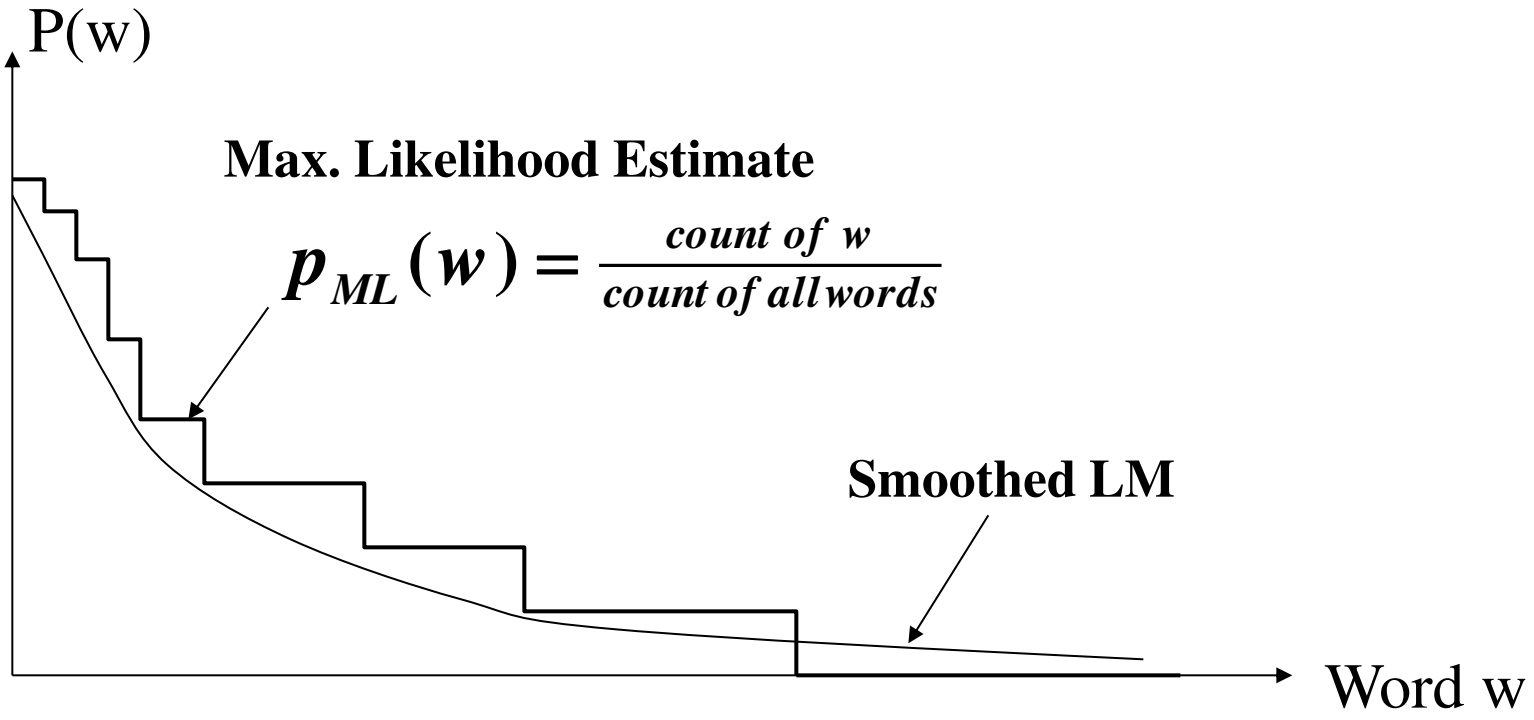
Document language model

- Retrieval problem $\approx$ Estimation of $p(w_i|d)$
- Smoothing is an important issue, and distinguishes different approaches

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?
- If we want to assign non-zero probabilities to such words, we'll have to discount the probabilities of observed words
- This is what "smoothing" is about …

Language Model Smoothing (Illustration)



A General Smoothing Scheme

- All smoothing methods try to
  - discount the probability of words seen in a doc
  - re-allocate the extra probability so that unseen words will have a non-zero probability
- Most use a reference model (collection language model) to discriminate unseen words

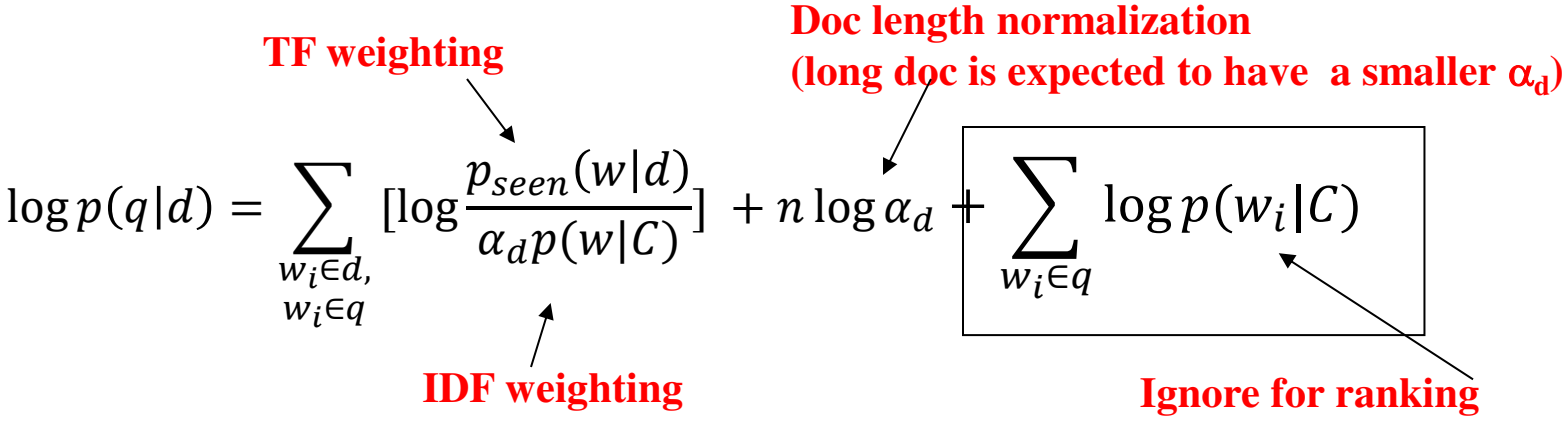
$$p(w \mid d) = \begin{cases} p_{seen}(w \mid d) & \text{if } w \text{ is seen in } d \\ \alpha_d p(w \mid C) & \text{otherwise} \end{cases}$$

Discounted ML estimate

Collection language model

Smoothing & TF-IDF Weighting

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



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

Derivation of the QL Retrieval Formula

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

- Retrieval formula using the general smoothing scheme:

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

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

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

Derivation of the QL Retrieval Formula (cond't)

$$\log p(q|d) = \cdots$$

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

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

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

Key rewriting step

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

Similar rewritings are very common when using LMs for IR…

Three Smoothing Methods

(Zhai & Lafferty 01)

- Simplified Jelinek-Mercer: Shrink uniformly toward p(w|C)

$$p(w|d) = (1-\lambda)\, p_{ML}(w|d) + \lambda p(w|C)$$

- Dirichlet prior (Bayesian): Assume pseudo counts μp(w|C)

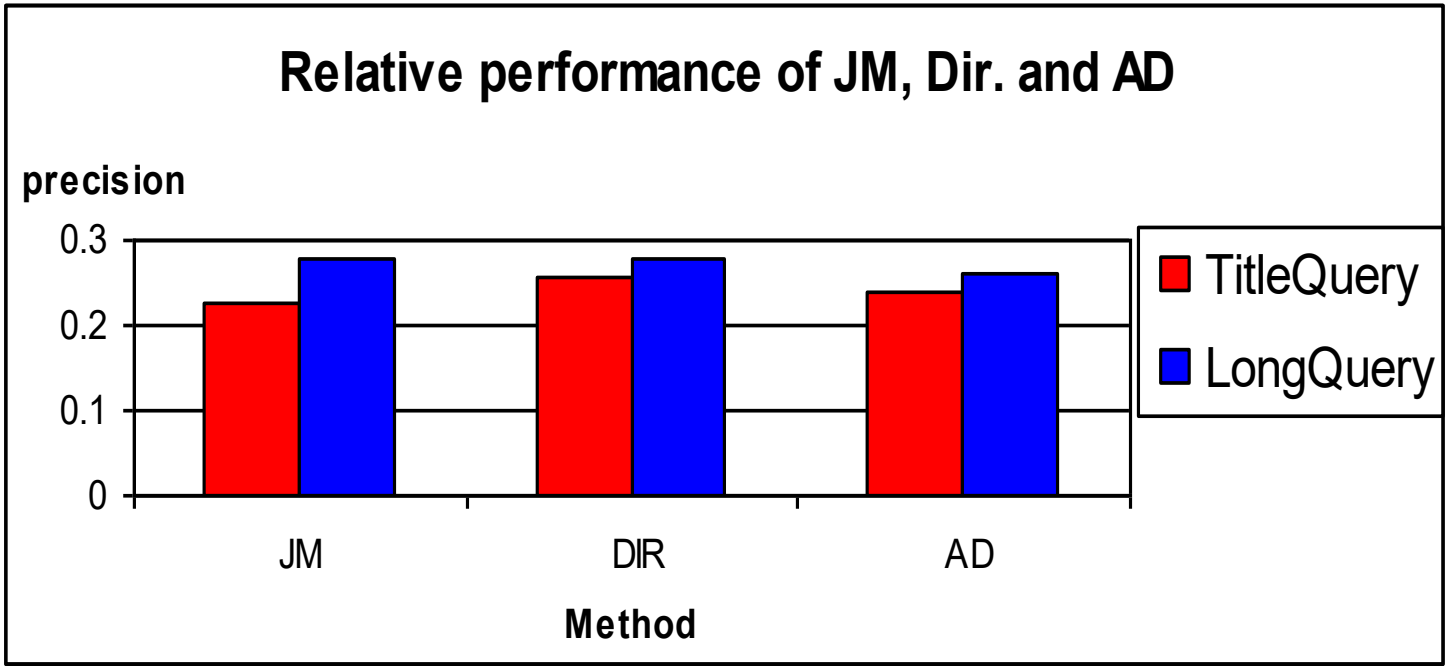
$$p(w|d) = \frac{c(w,d) + \mu p(w|C)}{|d| + \mu} = \frac{|d|}{|d| + \mu}\, p_{ML}(w|d) + \frac{\mu}{|d| + \mu}\, p(w|C)$$

- Absolute discounting: Subtract a constant δ

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

Comparison of Three Methods

Query Type	JM	Dir	AD
Title	0.228	**0.256**	0.237
Long	**0.278**	0.276	0.260



Long verbose

Short keyword

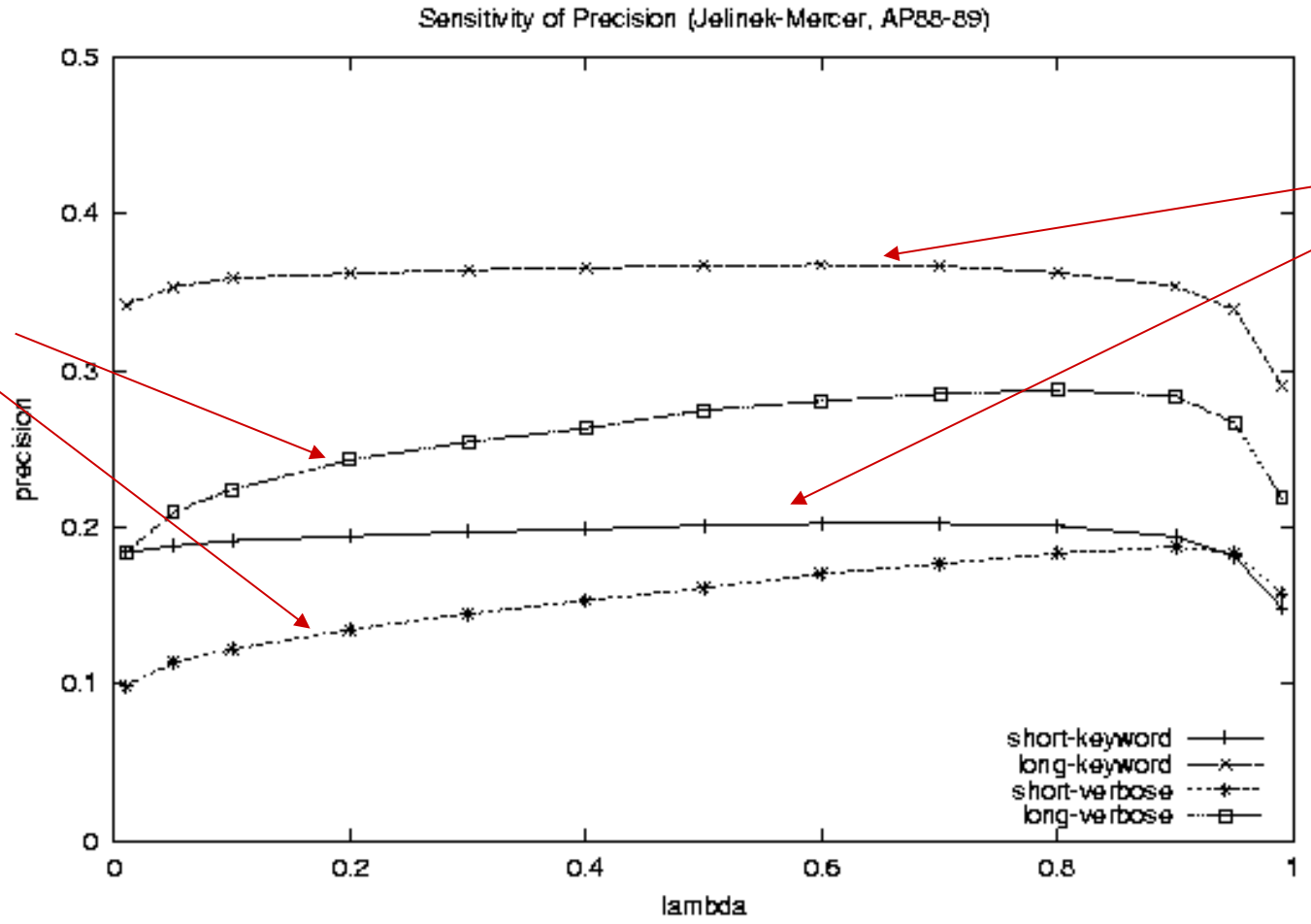
Short verbose

Title: South African Sanctions
Description: Document discusses sanctions against South Africa.
Narrative:
A relevant document will discuss any aspect of South African sanctions, such as: sanctions declared/proposed by a country against the South African government in response to its apartheid policy, or in response to pressure by an individual, organization or another country; international sanctions against Pretoria imposed by the United Nations; the effects of sanctions against S. Africa; opposition to sanctions; or, compliance with sanctions by a company. The document will identify the sanctions instituted or being considered, e.g., corporate disinvestment, trade ban, academic boycott, arms embargo.
Concepts:

(1) sanctions, international sanctions, economic sanctions
(2) corporate exodus, corporate disinvestment, stock divestiture, ban on new investment, trade ban, import ban on South African diamonds, U.N. arms embargo, curtailment of defense contracts, cutoff of nonmilitary goods, academic boycott, reduction of cultural ties
(3) apartheid, white domination, racism
(4) antiapartheid, black majority rule
(5) Pretoria

long keyword

The Need of Query-Modeling
(Dual-Role of Smoothing)



Keyword queries

Verbose queries

Why does query type affect smoothing sensitivity?

Another Reason for Smoothing

Content words

Query = "the	algorithms	for	data	mining"
$p_{DML}(w\|d1)$: 0.04	0.001	0.02	0.002	0.003
$p_{DML}(w\|d2)$: 0.02	0.001	0.01	0.003	0.004

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

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

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

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

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

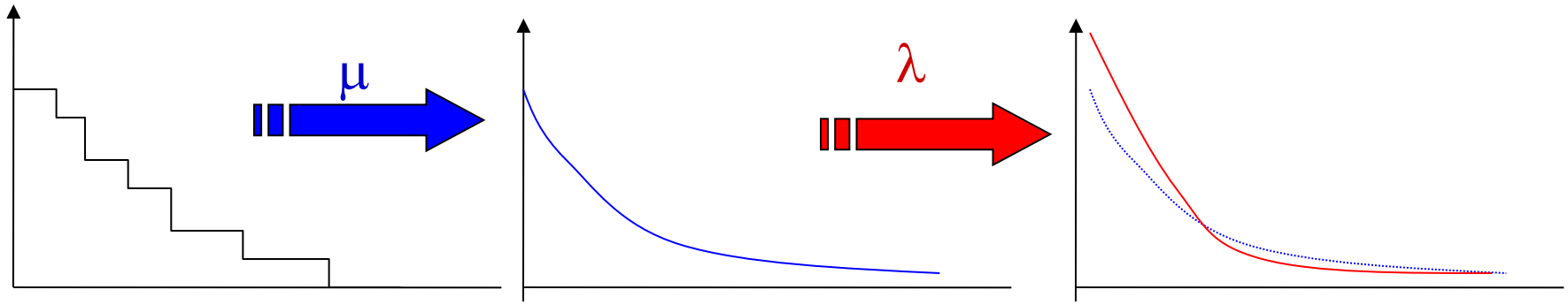
Two-stage Smoothing

Stage-1

-Explain unseen words
-Dirichlet prior(Bayesian)

Stage-2

-Explain noise in query



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

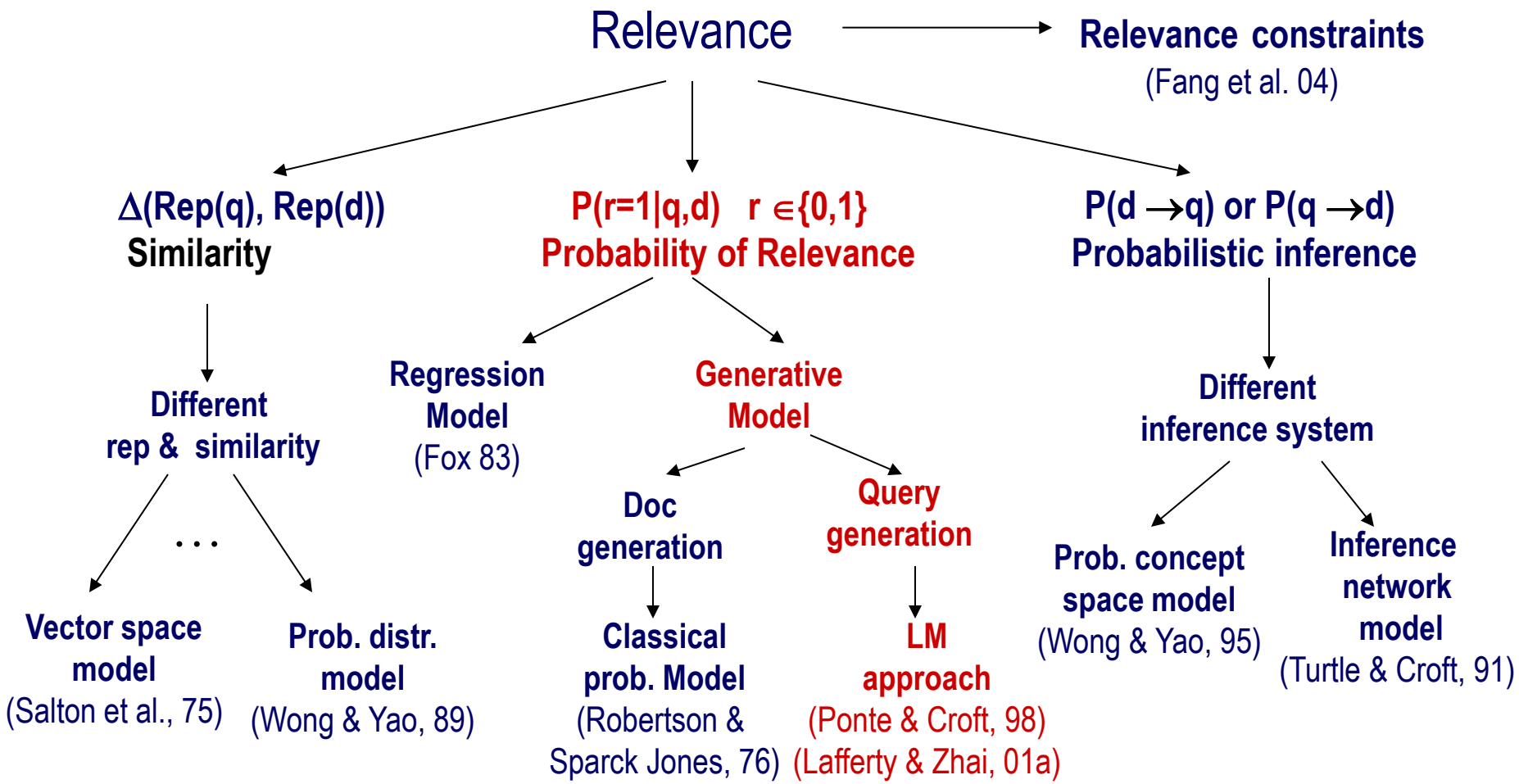
Query noise model

λ and μ can be automatically set through statistical estimation

But, where is the relevance?

And, what's good about this approach?

The Notion of Relevance



Probabilistic Retrieval Models: Computing p(R|Q,D)

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

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

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

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

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

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

Query likelihood $p(q|\theta_d)$ **Document prior**

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

Computing 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

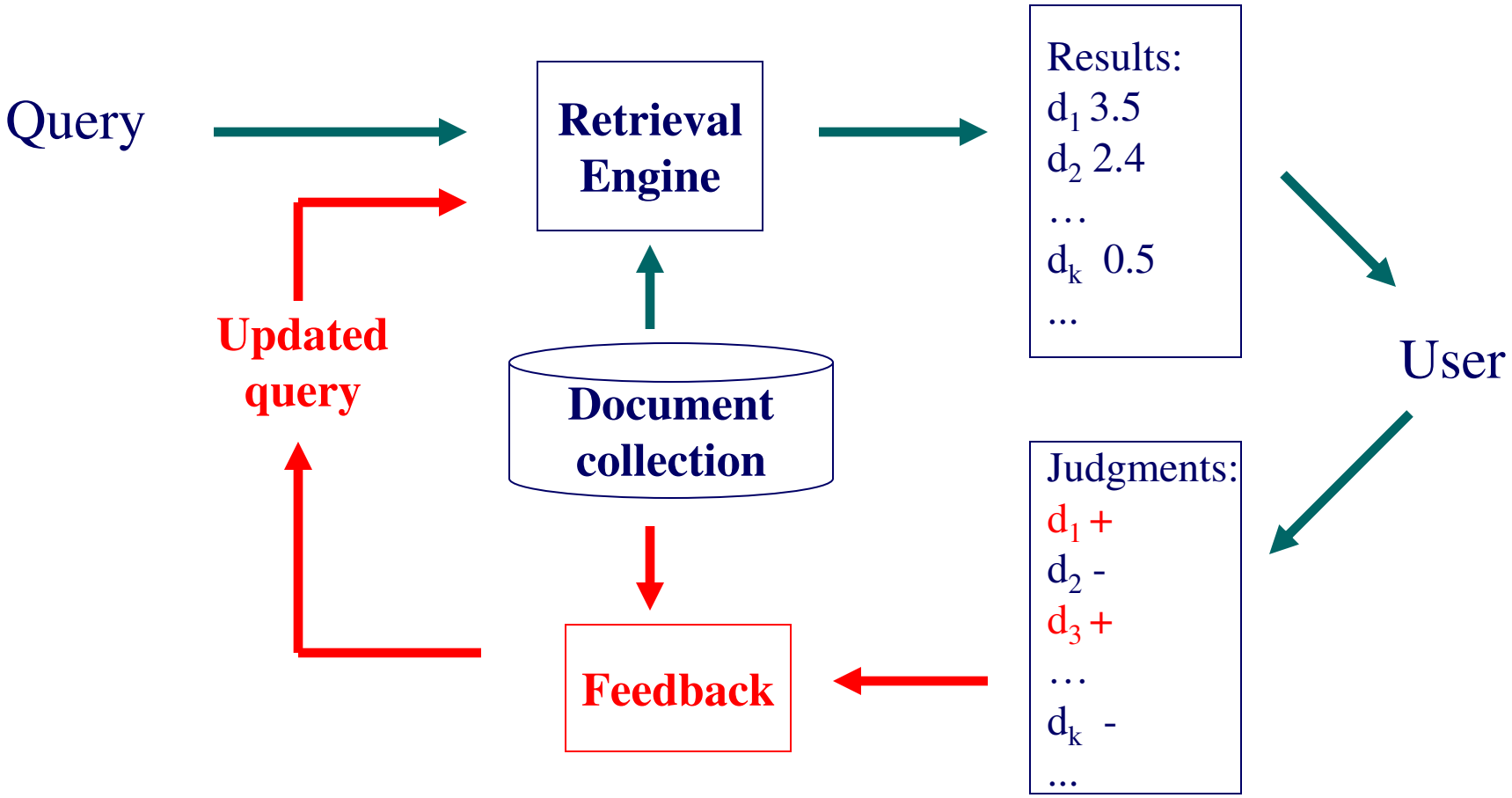
$P(Q|D)=P(Q|D, R=1)!$ Prob. that a user who likes D would pose query Q

Relevance-based interpretation of the so-called "document language model"

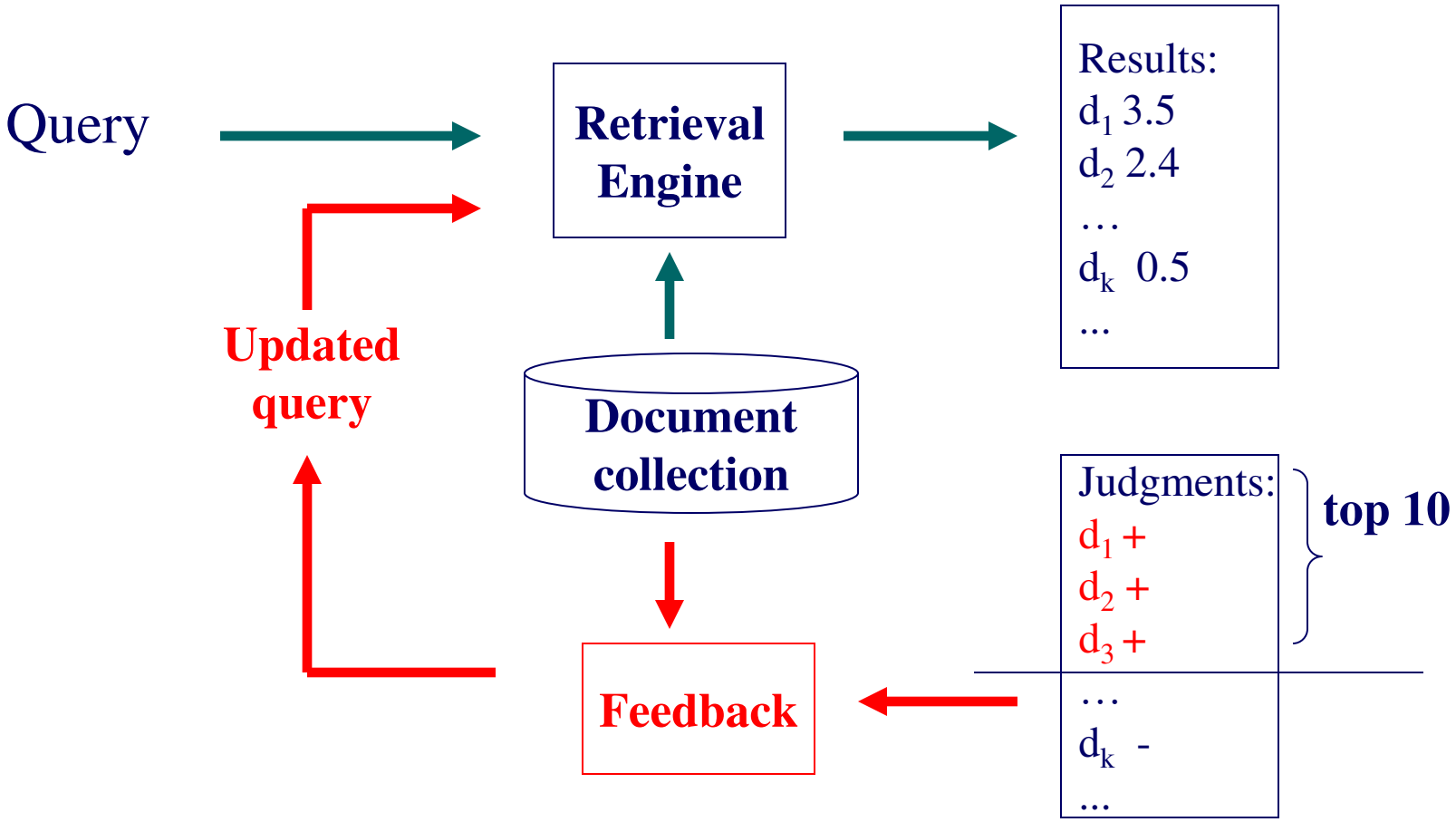
Question?

Feedback in Language Models

Relevance Feedback



Pseudo/Blind/Automatic Feedback



Intuition in Feedback

- Query expansion: feedback can help discover related query terms
  - Query = “information retrieval”
  - Relevant or pseudo-relevant docs would likely share words related to “information retrieval”, e.g., “search engine”, “search”, “user”, “query”, etc.
  - These words generally have higher frequency in these relevant or pseudo-relevant documents than in the whole collection
  - They can be used to expand the original query to increase recall and sometimes also precision
- Machine learning/pattern recognition
  - Relevant documents = labeled examples

Overview of Feedback Techniques

- Feedback as machine learning: many possibilities
  - Standard ML: Given examples of relevant (and non-relevant) documents, learn how to classify a new document as either "relevant" or "non-relevant".
  - "Modified" ML: Given a query and examples of relevant (and non-relevant) documents, learn how to rank new documents based on relevance
- Feedback as query expansion: traditional IR
  - Step 1: Term selection
  - Step 2: Query expansion
  - Step 3: Query term re-weighting
- Traditional IR is still robust (Rocchio), but ML approaches can potentially be more accurate

Question: How to exploit language modeling to perform natural and effective feedback?

Answer: **Introduce a query model & treat feedback as query model updating**

Retrieval function:

Query-likelihood => KL-Divergence

Feedback:

Expansion-based => Model-based

Kullback-Leibler (KL) Divergence Measure

- Given two probability mass functions p(x) and q(x), the KL divergence between p and q is defined as:

$$D(p \| q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

- Properties:
  - $D(p \| q) \geq 0$
  - $D(p \| q) = 0 \quad \text{iff} \quad p = q$

KL-divergence is often used to measure the distance between two distributions

Kullback-Leibler (KL) Divergence Retrieval Model

- Unigram similarity model

$$Sim(q,d) \propto -D(\hat{\theta}_Q \| \hat{\theta}_D)$$

$$\propto \sum_w p(w|\hat{\theta}_Q) \log p(w|\hat{\theta}_D) + (-\sum_w p(w|\hat{\theta}_Q) \log p(w|\hat{\theta}_Q))$$

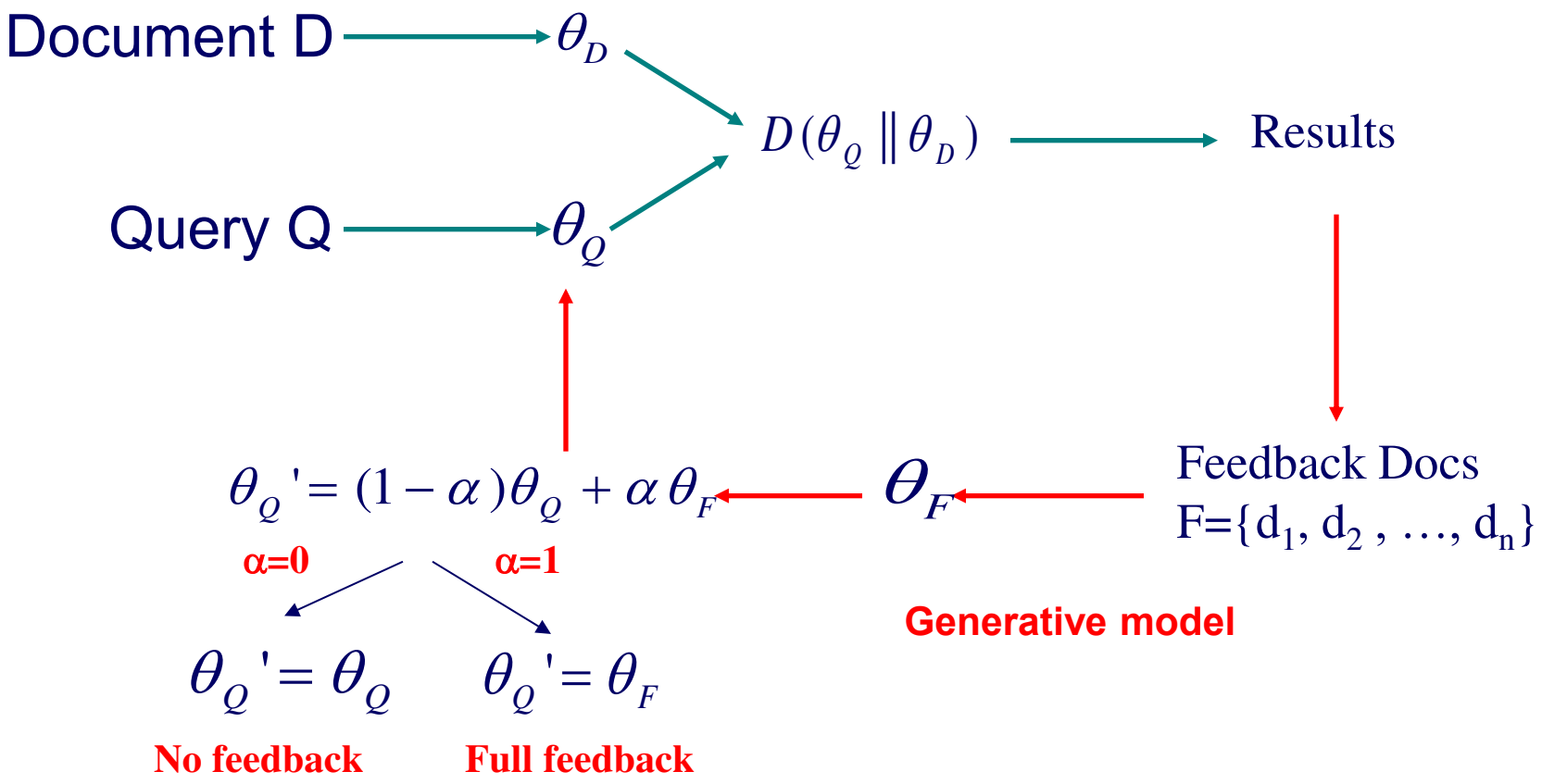
ignored for ranking

- Retrieval ≈ Estimation of θ_Q and θ_D

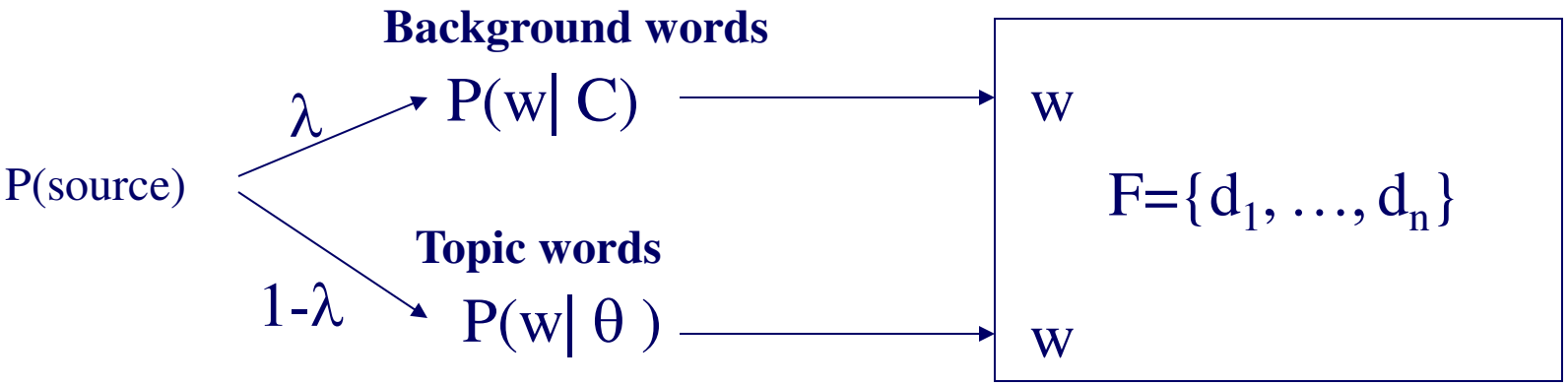
$$sim(q,d) \approx \sum_{w \in d, p(w|\theta_Q)>0} [p(w|\hat{\theta}_Q) \log \frac{p_{seen}(w|d)}{\alpha_d p(w|C)}] + \log \alpha_d$$

- Special case: $\hat{\theta}_Q$ = empirical distribution of *q* recovers "query-likelihood"

Feedback as Model Interpolation



Generative Mixture Model



$$\log p(F \mid \theta) = \sum_i \sum_w c(w; d_i) \log[(1-\lambda)\, p(w \mid \theta) + \lambda\, p(w \mid C)]$$

Maximum Likelihood $\theta_F = \arg\max_{\theta} \log p(F \mid \theta)$

λ = Noise in feedback documents

Understanding a Mixture Model

Known
Background
$p(w|C)$

the 0.2
a 0.1
we 0.01
to 0.02
…
text 0.0001
mining 0.00005
…

Unknown
query topic
$p(w|\theta_F)$=?

"Text mining"

…
text =?
mining =?
association =?
word =?
…

Suppose each model would be selected with equal probability $\lambda = 0.5$

The probability of observing word "text":
$\lambda p(\text{"text"}|C) + (1-\lambda)p(\text{"text"}|\theta_F) =$
$0.5 * 0.0001 + 0.5 * p(\text{"text"}|\theta_F)$

The probability of observing word "the":
$\lambda p(\text{"the"}|C) + (1-\lambda)p(\text{"the"}|\theta_F) =$
$0.5 * 0.2 + 0.5 * p(\text{"the"}|\theta_F)$

The probability of observing "the"&"text" (likelihood)
$[0.5 * 0.0001 + 0.5 * p(\text{"text"}|\theta_F)]$
$* [0.5 * 0.2 + 0.5 * p(\text{"the"}|\theta_F)]$

How to set $p(\text{"the"}|\theta_F)$ and $p(\text{"text"}|\theta_F)$ so as to maximize this likelihood?
Assume $p(\text{"the"}|\theta_F) + p(\text{"text"}|\theta_F) =$ constant
=> Give $p(\text{"text"}|\theta_F)$ a higher probability than $p(\text{"the"}|\theta_F)$

How to Estimate θ_F?

Known
Background
$p(w|C)$

the 0.2
a 0.1
we 0.01
to 0.02
…
text 0.0001
mining 0.00005
…

Unknown
query topic
$p(w|\theta_F)$=?

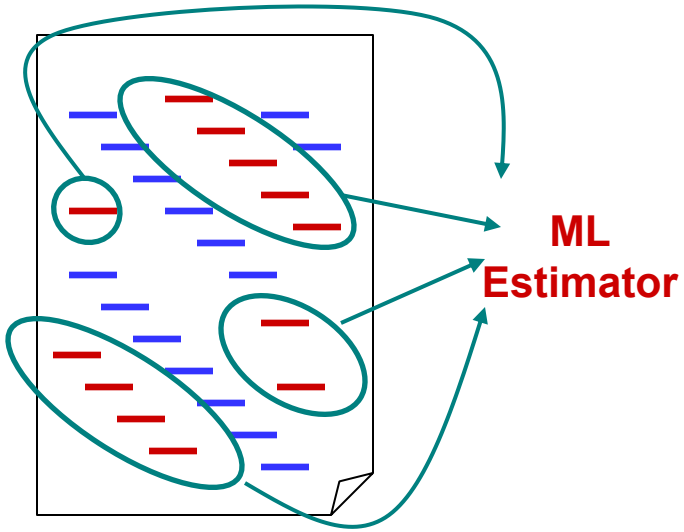
"Text mining"

…
text =?
mining =?
association =?
word =?
…

λ=0.7

λ=0.3

Observed
Doc(s)



Suppose,
we know
the identity of each word ...

Can We Guess the Identity?

Identity ("hidden") variable: $z_i \in \{1(\text{background}), 0(\text{topic})\}$

	z_i
the	1
paper	1
presents	1
a	1
text	0
mining	0
algorithm	0
the	1
paper	0
...	...

Suppose the parameters are all known, what's a reasonable guess of z_i?

- depends on λ
- depends on p(w|C) and $p(w|\theta_F)$

$$p(z_i = 1|w_i) = \frac{p(z_i = 1)p(w_i|z_i = 1)}{p(z_i = 1)p(w_i|z_i = 1) + p(z_i = 0)p(w_i|z_i = 0)}$$

$$= \frac{\lambda p(w_i|C)}{\lambda p(w_i|C) + (1-\lambda)p(w_i|\Theta_F)} \quad \text{E-step}$$

$$p^{new}(w_i \mid \theta_F) = \frac{c(w_i, F)(1 - p^{(n)}(z_i = 1 \mid w_i))}{\sum_{w_j \in vocabulary} c(w_j, F)(1 - p^{(n)}(z_j = 1 \mid w_j))} \quad \text{M-step}$$

Initially, set p(w| θ_F) to some random value, then iterate …

An Example of EM Computation

$$p^{(n)}(z_i = 1 \mid w_i) = \frac{\lambda p(w_i \mid C)}{\lambda p(w_i \mid C) + (1-\lambda) p^{(n)}(w_i \mid \theta_F)}$$

Expectation-Step:
Augmenting data by guessing hidden variables

$$p^{(n+1)}(w_i \mid \theta_F) = \frac{c(w_i, F)(1 - p^{(n)}(z_i = 1 \mid w_i))}{\sum_{w_j \in vocabulary} c(w_j, F)(1 - p^{(n)}(z_j = 1 \mid w_j))}$$

Maximization-Step
With the "augmented data", estimate parameters using maximum likelihood

Assume λ=0.5

Word	#	P(w|C)	Iteration 1		Iteration 2		Iteration 3	
			P(w|θ_F)	P(z=1)	P(w|θ_F)	P(z=1)	P(w|θ_F)	P(z=1)
The	4	0.5	**0.25**	0.67	**0.20**	0.71	**0.18**	0.74
Paper	2	0.3	**0.25**	0.55	**0.14**	0.68	**0.10**	0.75
Text	4	0.1	**0.25**	0.29	**0.44**	0.19	**0.50**	0.17
Mining	2	0.1	**0.25**	0.29	**0.22**	0.31	**0.22**	0.31
Log-Likelihood			-16.96		-16.13		-16.02	

Example of Feedback Query Model

Trec topic 412: "airport security"

Mixture model approach

λ=0.9

w	$p(w\|\Theta_F)$
security	0.0558
airport	0.0546
beverage	0.0488
alcohol	0.0474
bomb	0.0236
terrorist	0.0217
author	0.0206
license	0.0188
bond	0.0186
counter-terror	0.0173
terror	0.0142
newsnet	0.0129
attack	0.0124
operation	0.0121
headline	0.0121

Web database

Top 10 docs

λ=0.7

w	$p(w\|\Theta_F)$
the	0.0405
security	0.0377
airport	0.0342
beverage	0.0305
alcohol	0.0304
to	0.0268
of	0.0241
and	0.0214
author	0.0156
bomb	0.0150
terrorist	0.0137
in	0.0135
license	0.0127
state	0.0127
by	0.0125

Negative Relevance Feedback

- How we can learn from examples of nonrelevant documents?
- It is nontrivial to use negative information in the KL-divergence retrieval model
  - We use a generative query language model to represent the information need
  - The model cannot assign a negative probability to any term => hard to penalize a term.

Negative Relevance Feedback (Cont'd)

- Heuristic modification:
  - Introduce θ_N: Negative topic language model
  - Use θ_N to compute a "distraction score" for each document.
  - Combine the distraction score with the original score

$$Score(Q,D) \overset{rank}{=} -D(\theta_Q \| \theta_D) + \beta D(\theta_N \| \theta_D)$$

  - θ_N can be estimated based on negative feedback documents in the same way as query model estimation

Questions?