Language Models for Text Retrieval

ChengXiang Zhai

Department of Computer Science

University of Illinois, Urbana-Champaign

Outline

- General questions to ask about a language model
- Probabilistic model for text retrieval
- Document-generation models
- Query-generation models

Central Questions to Ask about a LM: "ADMI"

- **Application**: Why do you need a LM? For what purpose?

  → Evaluation metric for a LM

  Information Retrieval

- **Data**: What kind of data do you want to model?

  → Data set for estimation & evaluation

  Documents & Queries

- **Model**: How do you define the model?

  → Assumptions to be made

  Doc. vs. Query generation, independence

- **Inference**: How do you infer/estimate the parameters?

  → Inference/Estimation algorithm

  Smoothing methods, Pseudo feedback

The Basic Question

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

Formally…

3 random variables: query Q, document D, relevance R $\in\{0,1\}$

Given a particular query q, a particular document d, $p(R=1|Q=q,D=d)=?$

The Data and Information Systems Laboratories at The University of Illinois at Urbana-Champaign Large Scale Information Management and Mining

Probability of Relevance

- Three random variables
  - Query Q
  - Document D
  - Relevance $R \in \{0,1\}$
- Goal: rank D based on $P(R=1|Q,D)$
  - Evaluate $P(R=1|Q,D)$
  - Actually, only need to compare $P(R=1|Q,D1)$ with $P(R=1|Q,D2)$, I.e., rank documents
- Several different ways to refine $P(R=1|Q,D)$

Probabilistic Retrieval Models: Intuitions

Suppose we have a large number of relevance judgments (e.g., clickthroughs: "1"=clicked; "0"= skipped)

Query(Q)	Doc (D)	Rel (R) ?
Q1	D1	1
Q1	D2	1
Q1	D3	0
Q1	D4	0
Q1	D5	1
…		
Q1	D1	0
Q1	D2	1
Q1	D3	0
Q2	D3	1
Q3	D1	1
Q4	D2	1
Q4	D3	0
…		

We can score documents based on

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

P(R=1|Q1, D1)=1/2
P(R=1|Q1,D2)=2/2
P(R=1|Q1,D3)=0/2
…

What if we don't have (sufficient) search log?

We can approximate p(R=1|Q,D)!

Different assumptions lead to different models

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

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

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

Ignored for ranking D

- Special cases
  - Document "generation": P(Q,D|R)=P(D|Q,R)P(Q|R)
  - Query "generation": P(Q,D|R)=P(Q|D,R)P(D|R)

Document Generation

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

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

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

← **Model of relevant docs for Q**

← **Model of non-relevant docs for Q**

Assume independent attributes $A_1 \ldots A_k$ ….(why?)

Let $D=d_1 \ldots d_k$, where $d_k \in \{0,1\}$ is the value of attribute A_k (Similarly $Q=q_1 \ldots q_k$)

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

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

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

$$\approx \prod_{i=1,d_i=q_i=1}^{k} \frac{P(A_i=1|Q,R=1)P(A_i=0|Q,R=0)}{P(A_i=1|Q,R=0)P(A_i=0|Q,R=1)} \quad (Assume\ \ P(A_i=1|Q,R=1)=P(A_i=1|Q,R=0), if\ \ q_i=0)$$

The Data and Information Systems Laboratories at The University of Illinois at Urbana-Champaign Large Scale Information Management and Mining

TIMAN

Robertson-Sparck Jones Model
(Robertson & Sparck Jones 76)

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

Two parameters for each term A_i:

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

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

How to estimate parameters?

Suppose we have relevance judgments,

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

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

RSJ Model: No Relevance Info

(Croft & Harper 79)

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

How to estimate parameters?

Suppose we do not have relevance judgments,

- We will assume p_i to be a constant
- Estimate q_i by assuming all documents to be non-relevant

$$\log O(R=1|Q,D) \overset{Rank}{\approx} \sum_{i=1, d_i=q_i=1}^{k} \log \frac{N-n_i+0.5}{n_i+0.5}$$

$$IDF' = \log \frac{N-n_i}{n_i}$$

N: # documents in collection

n_i: # documents in which term A_i occurs

RSJ Model: Summary

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

Improving RSJ: Adding TF

Basic doc. generation model: $$\frac{P(R=1|Q,D)}{P(R=0|Q,D)} \propto \frac{P(D|Q,R=1)}{P(D|Q,R=0)}$$

Let $D=d_1 \ldots d_k$, where d_k is the frequency count of term A_k

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

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

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

2-Poisson mixture model $$p(A_i=f|Q,R) = p(E|Q,R)p(A_i=f|E) + P(\overline{E}|Q,R)p(A_i=f|\overline{E})$$

$$= p(E|Q,R)\frac{\mu_E^{\ f}}{f!}e^{-\mu_E} + P(\overline{E}|Q,R)\frac{\mu_{\overline{E}}^{\ f}}{f!}e^{-\mu_{\overline{E}}}$$

Many more parameters to estimate! (how many exactly?)

BM25/Okapi Approximation
(Robertson et al. 94)

- Idea: Approximate p(R=1|Q,D) with a simpler function that share similar properties
- Observations:
  - log O(R=1|Q,D) is a sum of term weights W_i
  - W_i= 0, if TF_i=0
  - W_i increases monotonically with Tfi
  - W_i has an asymptotic limit
- The simple function is

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

Adding Doc. Length & Query TF

- Incorporating doc length
  - Motivation: The 2-Poisson model assumes equal document length
  - Implementation: "Carefully" penalize long doc
- Incorporating query TF
  - Motivation: Appears to be not well-justified
  - Implementation: A similar TF transformation
- The final formula is called BM25, achieving top TREC performance

The BM25 Formula

$$\sum_{T \in Q} w^{(1)} \frac{(k_1 + 1)tf}{K + tf} \frac{(k_3 + 1)\, qtf}{k_3 + qtf} \quad (1)$$

"Okapi TF/BM25 TF"

where

Q is a query, containing terms T
$w^{(1)}$ is the Robertson/Sparck Jones weight [5] of T in Q

$$\log \frac{(r + 0.5)/(R - r + 0.5)}{(n - r + 0.5)/(N - n - R + r + 0.5)} \quad (2)$$

N is the number of items (documents) in the collection
n is the number of documents containing the term
R is the number of documents known to be relevant to a specific topic
r is the number of relevant documents containing the term
K is $k_1((1 - b) + b.dl/avdl)$
k_1, b and k_3 are parameters which depend on the on the nature of the queries and possibly on the database; k_1 and b default to 1.2 and 0.75 respectively, but smaller values of b are sometimes advantageous; in long queries k_3 is often set to 7 or 1000 (effectively infinite)
tf is the frequency of occurrence of the term within a specific document
qtf is the frequency of the term within the topic from which Q was derived
dl and $avdl$ are respectively the document length and average document length measured in some suitable unit.

Extensions of "Doc Generation" Models

- Capture term dependence (Rijsbergen & Harper 78)
- Alternative ways to incorporate TF (Croft 83, Kalt96)
- Feature/term selection for feedback (Okapi's TREC reports)
- Estimate of the relevance model based on pseudo feedback, to be covered later [Lavrenko & Croft 01]

Query Generation (➔ Language Models for IR)

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

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

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

Query likelihood p(Q| D,R=1)

Document prior

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

Now, the question is how to compute $P(Q|D,R=1)$?

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, R=1) **Prob. that a user who likes D would pose query Q. How to estimate it?**

The Basic LM Approach

[Ponte & Croft 98]

Document

Language Model

Text mining paper

…
text ?
mining ?
assocation ?
clustering ?
…
food ?
…

Food nutrition paper

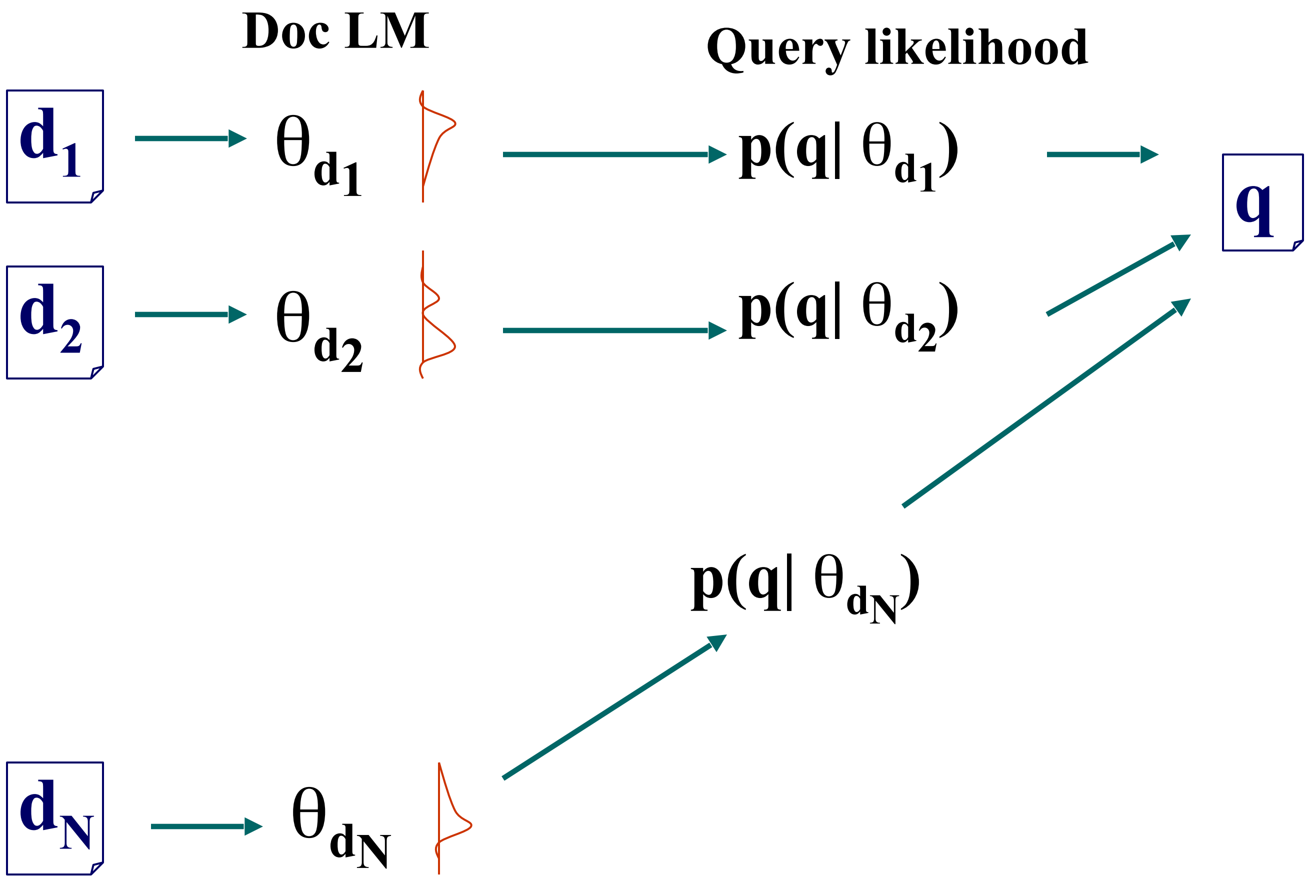
…
food ?
nutrition ?
healthy ?
diet ?
…

?

Query = "data mining algorithms"

Which model would most likely have generated this query?

Ranking Docs by Query Likelihood



Modeling Queries: Different Assumptions

- Multi-Bernoulli: Modeling word presence/absence
  - $q = (x_1, \ldots, x_{|V|})$, $x_i = 1$ for presence of word w_i; $x_i = 0$ for absence

$$p(q=(x_1,\ldots,x_{|V|})\,|\,d) = \prod_{i=1}^{|V|} p(w_i = x_i\,|\,d) = \prod_{i=1,x_i=1}^{|V|} p(w_i = 1\,|\,d) \prod_{i=1,x_i=0}^{|V|} p(w_i = 0\,|\,d)$$

  - Parameters: $\{p(w_i=1|d), p(w_i=0|d)\}$ $p(w_i=1|d) + p(w_i=0|d) = 1$
- Multinomial (Unigram LM): Modeling word frequency
  - $q = q_1, \ldots q_m$, where q_j is a query word

$$p(q = q_1 \ldots q_m\,|\,d) = \prod_{j=1}^{m} p(q_j\,|\,d) = \prod_{i=1}^{|V|} p(w_i\,|\,d)^{c(w_i,q)}$$

  - $c(w_i,q)$ is the count of word w_i in query q
  - Parameters: $\{p(w_i|d)\}$ $p(w_1|d) + \ldots p(w_{|V|}|d) = 1$

[Ponte & Croft 98] **uses Multi-Bernoulli; most other work uses multinomial**
Multinomial seems to work better [Song & Croft 99, McCallum & Nigam 98,Lavrenko 04]

Retrieval as LM Estimation

- Document ranking based on *query likelihood*

$$\log p(q \mid d) = \sum_{i=1}^{m} \log p(q_i \mid d) = \sum_{i=1}^{|V|} c(w_i, q) \log \boxed{p(w_i \mid d)}$$

$$where,\ \ q = q_1 q_2 \ldots q_m$$

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? Smoothing!

How to smooth a LM

- Key Question: what probability should be assigned to an unseen word?
- Let the probability of an unseen word be proportional to its probability given by a reference LM
- One possibility: Reference LM = Collection LM

Discounted ML estimate

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

Collection language model

Rewriting the Ranking Function with Smoothing

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

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

Query words **matched** in d

Query words **not matched** in d

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

All query words

Query words **matched** in d

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

The Data and Information Systems Laboratories at The University of Illinois at Urbana-Champaign

Large Scale Information Management and Mining

Benefit of Rewriting

- Better understanding of the ranking function
  - Smoothing with p(w|C) ➔ TF-IDF weighting + length norm.

TF weighting

Doc length normalization

$$\log p(q|d) = \sum_{\substack{w_i \in d \\ w_i \in q}} c(w,q)[\log \frac{p_{Seen}(w_i|d)}{\alpha_d\, p(w_i|C)}] + n \log \alpha_d + \boxed{\sum_{i=1}^{n} \log p(w_i|C)}$$

matched query terms

IDF weighting

Ignore for ranking

- Enable efficient computation

Query Likelihood Retrieval Functions

$$\log p(q\,|\,d) = \sum_{\substack{w_i \in d \\ w_i \in q}} [\log \frac{p_{seen}(w_i\,|\,d)}{\alpha_d\, p(w_i\,|\,C)}] + n \log \alpha_d + \sum_{i=1}^{n} \log p(w_i\,|\,C)$$

$$p(w\,|\,C) = \frac{c(w,C)}{\sum_{w' \in V} c(w',C)}$$

With Jelinek-Mercer (JM):

$$S_{JM}(q,d) = \log p(q\,|\,d) = \sum_{\substack{w \in d \\ w \in q}} \log[1 + \frac{1-\lambda}{\lambda} \frac{c(w,d)}{|d|\,p(w\,|\,C)}]$$

With Dirichlet Prior (DIR):

$$S_{DIR}(q,d) = \log p(q\,|\,d) = \sum_{\substack{w \in d \\ w \in q}} \log[1 + \frac{c(w,d)}{\mu p(w\,|\,C)}] + n \log \frac{\mu}{|d| + \mu}$$

What assumptions have we made in order to derive these functions?
Do they capture the same retrieval heuristics (TF-IDF, Length Norm) as a vector space retrieval function?

So, which method is the best?

It depends on the data and the task!

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

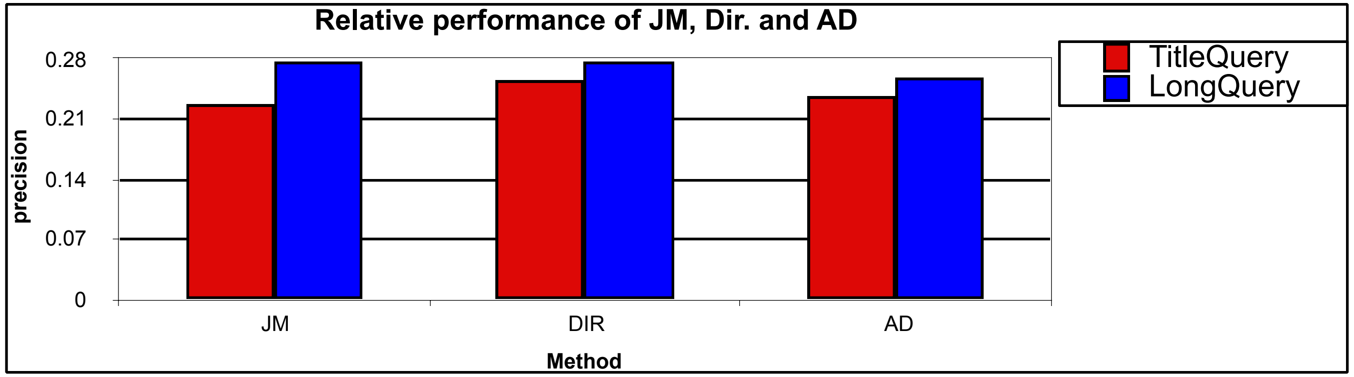
For retrieval, Dirichlet prior performs well…

Backoff smoothing [Katz 87] doesn't work well due to a lack of 2nd-stage smoothing…

Comparison of Three Methods

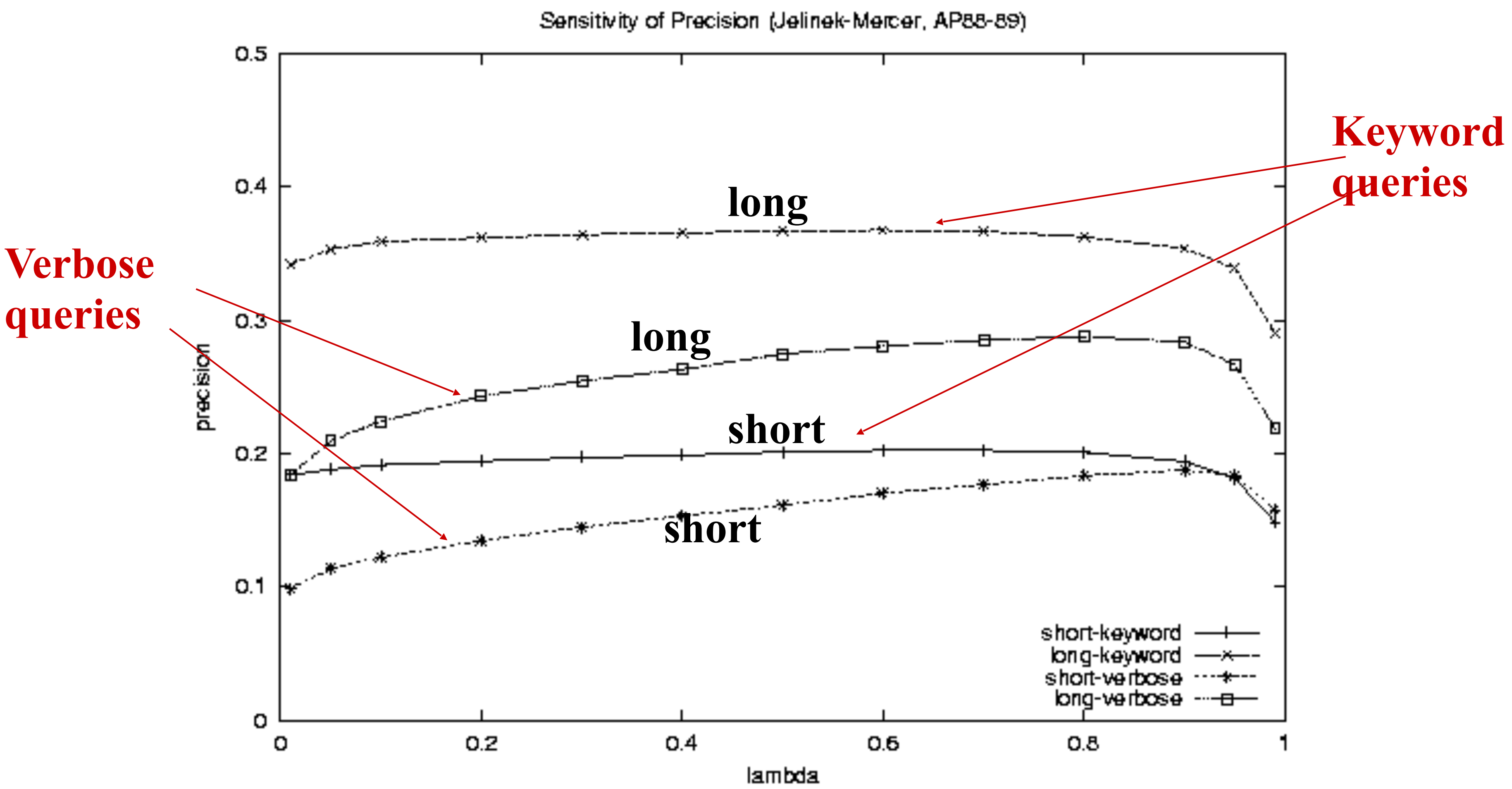
[Zhai & Lafferty 01a]

Query Type	Jelinek-Mercer	Dirichlet	Abs. Discounting
Title	0.228	**0.256**	0.237
Long	**0.278**	0.276	0.260



Comparison is performed on a variety of test collections

The Dual-Role of Smoothing [Zhai & Lafferty 02]



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

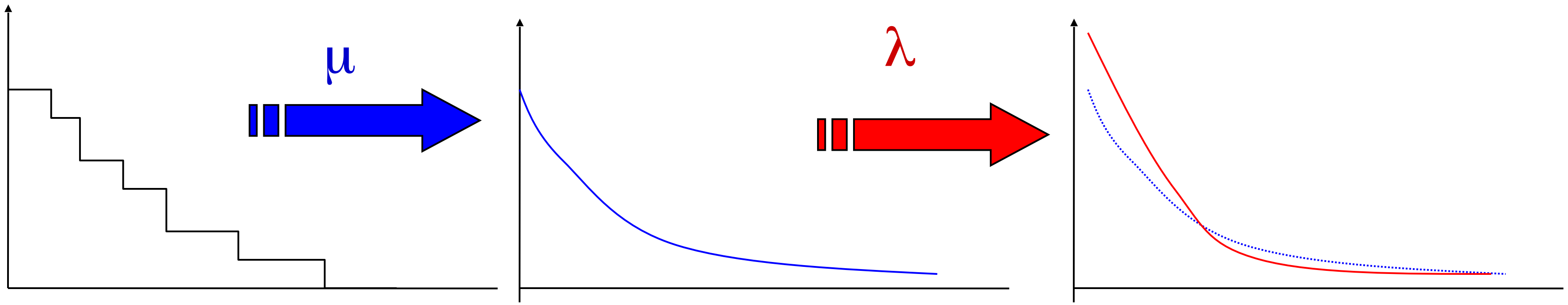
Two-stage Smoothing [Zhai & Lafferty 02]

Stage-1

-Explain unseen words
-Dirichlet prior(Bayesian)

Stage-2

-Explain noise in query
-2-component mixture

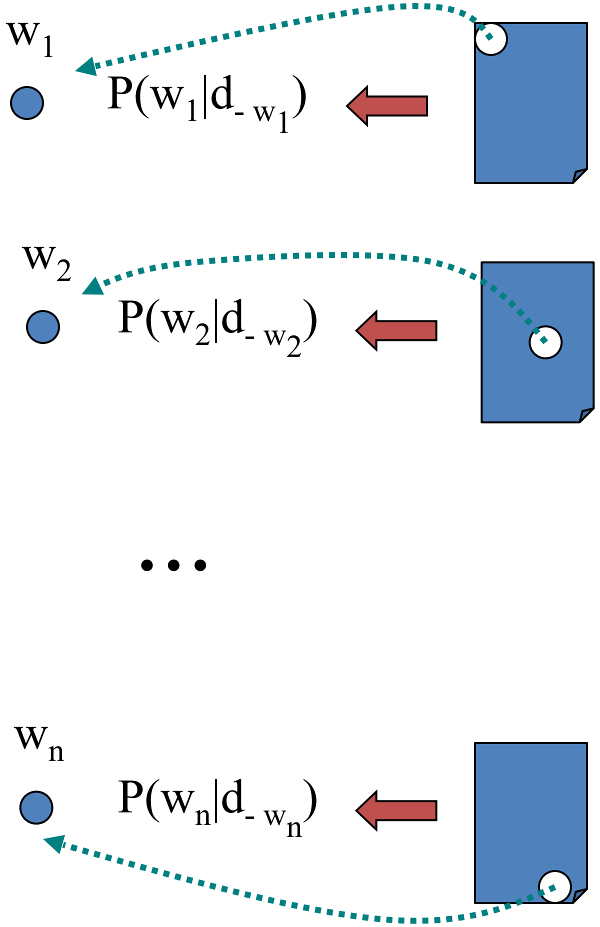


$$P(w|d) = (1-\lambda) \frac{c(w,d) + \mu p(w|C)}{|d| + \mu} + \lambda p(w|U)$$

Collection LM

User background model
Can be approximated by p(w|C)

Estimating μ using leave-one-out [Zhai & Lafferty 02]



log-likelihood

Leave-one-out

$$l_{-1}(\mu \mid C) = \sum_{i=1}^{N} \sum_{w \in V} c(w, d_i) \log\left(\frac{c(w, d_i) - 1 + \mu p(w \mid C)}{|d_i| - 1 + \mu}\right)$$

Maximum Likelihood Estimator

$$\hat{\mu} = \underset{\mu}{\operatorname{argmax}} \; l_{-1}(\mu \mid C)$$

Newton's Method

Why would "leave-one-out" work?

20 word by author1

abc abc ab c d d
abc cd d d
abd ab ab ab ab
cd d e cd e

Suppose we keep sampling and get 10 more words. Which author is likely to "write" <u>more new words</u>?

Now, suppose we leave "e" out…

μ doesn't have to be big

$$p_{ml}("e"|author1) = \frac{1}{19} \qquad p_{smooth}("e"|author1) = \frac{20}{20+\mu}\frac{1}{19} + \frac{\mu}{20+\mu}p("e"|REF)$$

20 word by author2

abc abc ab c d d
abe cb e f
acf fb ef aff abef
cdc db ge f s

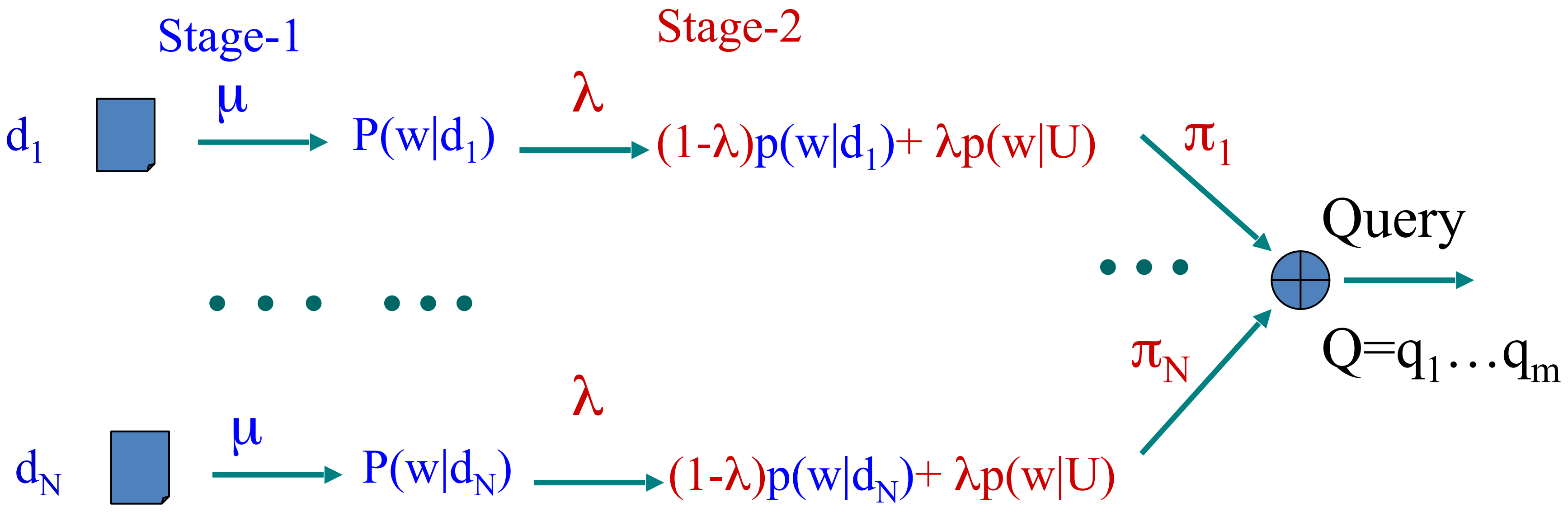
$$p_{ml}("e"|author2) = \frac{0}{19} \qquad p_{smooth}("e"|author2) = \frac{20}{20+\mu}\frac{0}{19} + \frac{\mu}{20+\mu}p("e"|REF)$$

μ must be big! <u>more smoothing</u>

The amount of smoothing is closely related to the underlying vocabulary size

Estimating λ using Mixture Model

[Zhai & Lafferty 02]



$$p(Q|\lambda,U)=\sum_{i=1}^{N}\pi_i\prod_{j=1}^{m}((1-\lambda)p(q_j|d_i)+\lambda p(q_j|U))$$

$$\hat{\lambda}=\underset{\lambda}{\operatorname{argmax}}\ p(Q|\lambda,U)$$

Estimated in stage-1

$$p(q_j|d_i)=\frac{c(q_j,d_i)+\hat{\mu}\,p(q_j|C)}{|d_i|+\hat{\mu}}$$

Maximum Likelihood Estimator
Expectation-Maximization (EM) algorithm

Automatic 2-stage results ≈ Optimal 1-stage results [Zhai & Lafferty 02]

Average precision (3 DB's + 4 query types, 150 topics)
*** Indicates significant difference**

Collection	query	Optimal-JM	Optimal-Dir	Auto-2stage
AP88-89	SK	20.3%	23.0%	22.2%*
	LK	36.8%	37.6%	37.4%
	SV	18.8%	20.9%	20.4%
	LV	28.8%	29.8%	29.2%
WSJ87-92	SK	19.4%	22.3%	21.8%*
	LK	34.8%	35.3%	35.8%
	SV	17.2%	19.6%	19.9%
	LV	27.7%	28.2%	28.8%*
ZIFF1-2	SK	17.9%	21.5%	20.0%
	LK	32.6%	32.6%	32.2%
	SV	15.6%	18.5%	18.1%
	LV	26.7%	27.9%	27.9%*

Completely automatic tuning of parameters IS POSSIBLE!

Feedback and Doc/Query Generation

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

Rel. doc model

NonRel. doc model

Query likelihood ("Language Model") $O(R=1|Q,D) \propto P(Q|D,R=1)$ ← **"Rel. query" model**

Parameter Estimation

$$\left.\begin{array}{l}(q_1,d_1,1)\\(q_1,d_2,1)\\(q_1,d_3,1)\end{array}\right\} P(D|Q,R=1)$$

$$\left.\begin{array}{l}(q_1,d_4,0)\\(q_1,d_5,0)\end{array}\right\} P(D|Q,R=0)$$

$$\left.\begin{array}{l}(q_3,d_1,1)\\(q_4,d_1,1)\\(q_5,d_1,1)\end{array}\right\} P(Q|D,R=1)$$

$(q_6,d_2,1)$

$(q_6,d_3,0)$

Initial retrieval:

- query as rel doc vs. doc as rel query
- P(Q|D,R=1) is more accurate

Feedback:

- P(D|Q,R=1) can be improved for the current query and future doc
- P(Q|D,R=1) can also be improved, but for current doc and future query

Query-based feedback **Doc-based feedback**

Difficulty in Feedback with Query Likelihood

- Traditional query expansion [Ponte 98, Miller et al. 99, Ng 99]
  – Improvement is reported, but there is a conceptual inconsistency
  – What's an expanded query, a piece of text or a set of terms?
- Avoid expansion
  – Query term reweighting [Hiemstra 01, Hiemstra 02]
  – Translation models [Berger & Lafferty 99, Jin et al. 02]
  – Only achieving limited feedback
- Doing relevant query expansion instead [Nallapati et al 03]
- The difficulty is due to the lack of a query/relevance model
- The difficulty can be overcome with alternative ways of using LMs for retrieval (e.g., relevance model [Lavrenko & Croft 01] , Query model estimation [Lafferty & Zhai 01b; Zhai & Lafferty 01b])

Two Alternative Ways of Using LMs

- Classic Probabilistic Model :Doc-Generation as opposed to Query-generation

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

  - Natural for relevance feedback
  - Challenge: Estimate p(D|Q,R=1) without relevance feedback; relevance model [Lavrenko & Croft 01] provides a good solution
- Probabilistic Distance Model :Similar to the vector-space model, but with LMs as opposed to TF-IDF weight vectors
  - A popular distance function: Kullback-Leibler (KL) divergence, covering query likelihood as a special case
  - Retrieval is now to estimate query & doc models and feedback is treated as query LM updating [Lafferty & Zhai 01b; Zhai & Lafferty 01b]

$$score(Q,D) = -D(\theta_Q \| \theta_D), \ essentially \sum_{w \in V} p(w|\theta_Q) \log p(w|\theta_D)$$

Both methods outperform the basic LM significantly

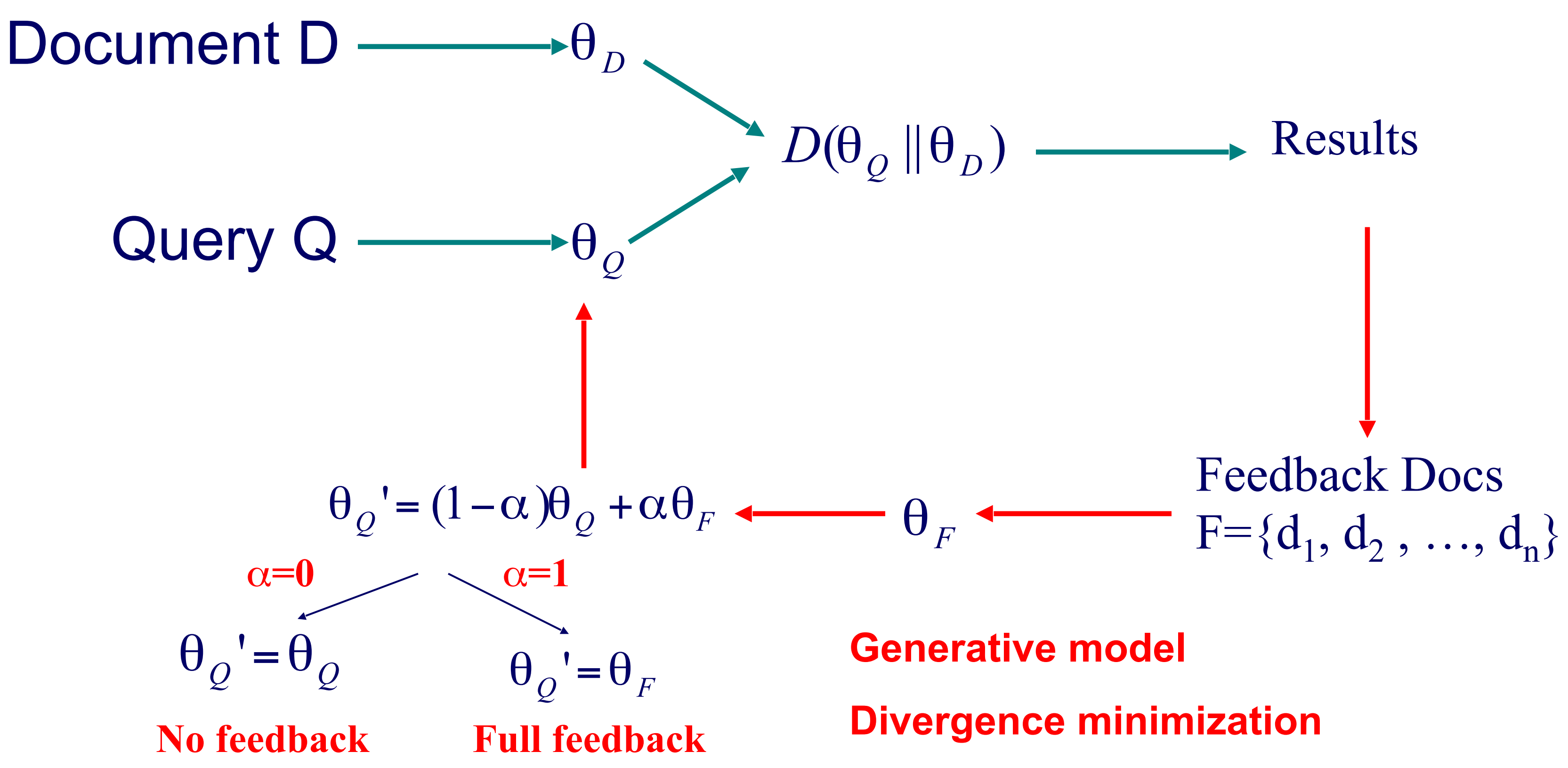
Query Model Estimation

[Lafferty & Zhai 01b, Zhai & Lafferty 01b]

- Question: How to estimate a better query model than the ML estimate based on the original query?
- "Massive feedback": Improve a query model through co-occurrence pattern learned from
  - A document-term Markov chain that outputs the query [Lafferty & Zhai 01b]
  - Thesauri, corpus [Bai et al. 05,Collins-Thompson & Callan 05]
- Model-based feedback: Improve the estimate of query model by exploiting pseudo-relevance feedback
  - Update the query model by interpolating the original query model with a learned feedback model [Zhai & Lafferty 01b]
  - Estimate a more integrated mixture model using pseudo-feedback documents [Tao & Zhai 06]

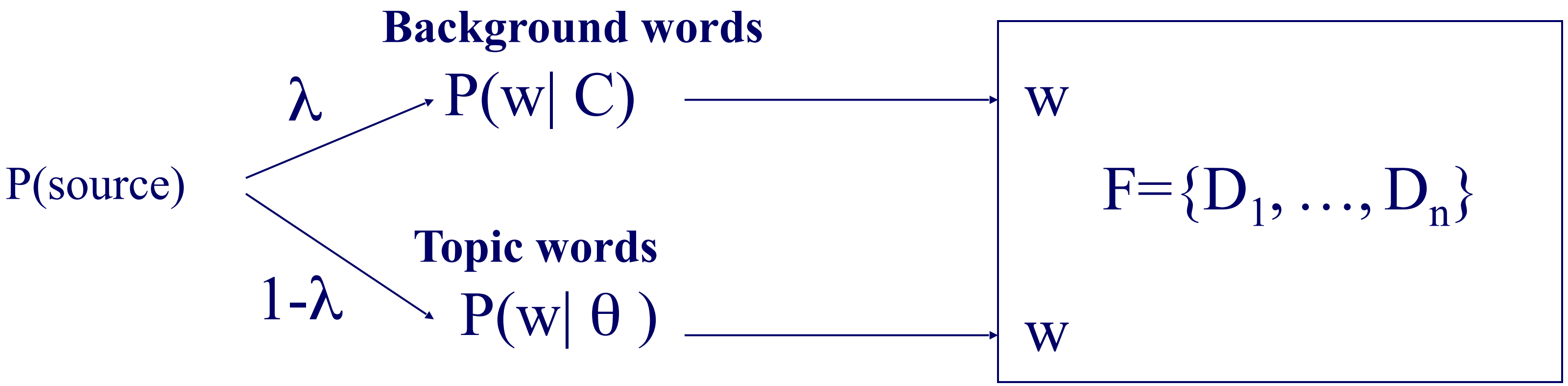
Feedback as Model Interpolation

[Zhai & Lafferty 01b]



θ_F Estimation Method I:

Generative Mixture Model



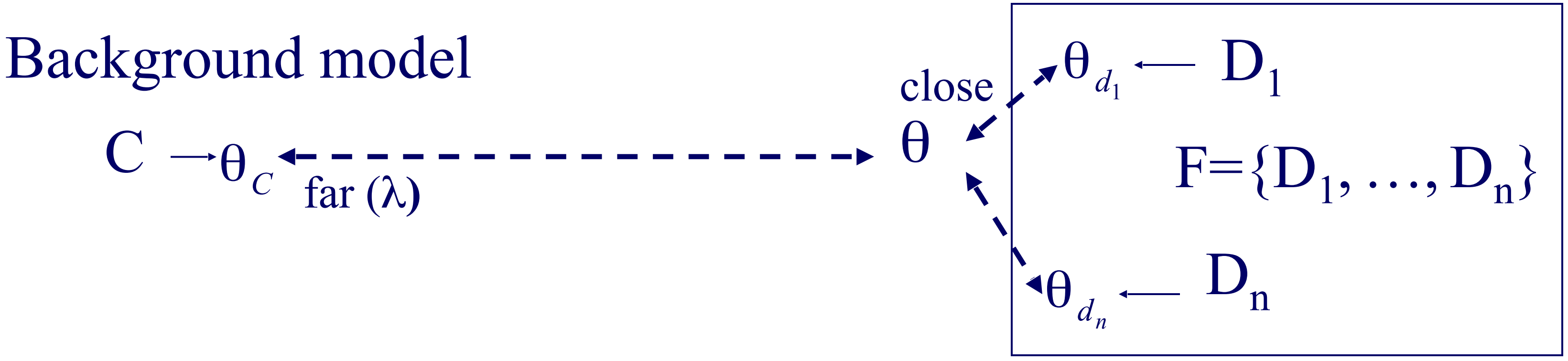
$$\log p(F|\theta) = \sum_{D \in F} \prod_{w \in D} c(w; D) \log((1-\lambda) p(w|\theta) + \lambda p(w|C))$$

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

The learned topic model is called a "parsimonious language model" in [Hiemstra et al. 04]

θ_F Estimation Method II:

Empirical Divergence Minimization



Empirical divergence

$$D_\lambda(\theta, F, C) = \frac{1}{|F|}\sum_{i=1}^{n} D(\theta \,||\, \theta_{D_j}) - \lambda D(\theta \,||\, \theta_C))$$

Divergence minimization

$$\theta_F = \underset{\theta}{\operatorname{argmin}}\, D_\lambda(\theta, F, C)$$

Example of Feedback Query Model

Trec topic 412: "airport security"

$\lambda=0.9$

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

Mixture model approach

Web database

Top 10 docs

$\lambda=0.7$

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

Model-based feedback Improves over Simple LM [Zhai & Lafferty 01b]

collection		Simple LM	Mixture	Improv.	Div.Min.	Improv.
AP88-89	AvgPr	0.21	0.296	+41%	0.295	+40%
	InitPr	0.617	0.591	-4%	0.617	+0%
	Recall	3067/4805	3888/4805	+27%	3665/4805	+19%
TREC8	AvgPr	0.256	0.282	+10%	0.269	+5%
	InitPr	0.729	0.707	-3%	0.705	-3%
	Recall	2853/4728	3160/4728	+11%	3129/4728	+10%
WEB	AvgPr	0.281	0.306	+9%	0.312	+11%
	InitPr	0.742	0.732	-1%	0.728	-2%
	Recall	1755/2279	1758/2279	+0%	1798/2279	+2%

What You Should Know

- Basic idea of probabilistic retrieval models
- How to use Bayes Rule to derive a general document-generation retrieval model
- How to derive the RSJ retrieval model (i.e., binary independence model)
- Assumptions that have to be made in order to derive the RSJ model

What You Should Know (cont.)

- Derivation of query likelihood retrieval model using query generation (what are the assumptions made?)
- Connection between query likelihood and TF-IDF weighting + doc length normalization
- The basic idea of two-stage smoothing
- KL-divergence retrieval model
- Basic idea of divergence minimization feedback method