III.4 Statistical Language Models

- III.4 Statistical LM *(MRS book, Chapter 12*)*
  - 4.1 What is a statistical language model?
  - 4.2 Smoothing Methods
  - 4.3 Extended LMs

*With extensions from: C. Zhai, J. Lafferty: A Study of Smoothing Methods for Language Models Applied to Information Retrieval, TOIS 22(2), 2004

III.4.1 What is a Statistical Language Model?

Generative model for word sequences

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

Example: P[“Today is Tuesday”] = 0.01
P[“The Eigenvalue is positive”] = 0.001
P[“Today Wednesday is”] = 0.000001

LM itself highly context- / application-dependent

Application 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?

Types of Language Models

<u>Key idea:</u> A document is a good match to a query if the *document model is likely to generate the query*, i.e., if P(q|d) "is high".

A language model is **well-formed** over alphabet ∑ if $\sum_{s \in \Sigma^*} P(s) = 1$.

Generic Language Model

"Today is Tuesday"	0.01
"The Eigenvalue is positive"	0.001
"Today Wednesday is"	0.00001
…	

Unigram Language Model

"Today"	0.1
"is"	0.3
"Tuesday"	0.2
"Wednesday"	0.2
…	

Bigram Language Model

"Today"	0.1
"is" | "Today"	0.4
"Tuesday" | "is"	0.8
…	

How to handle sequences?

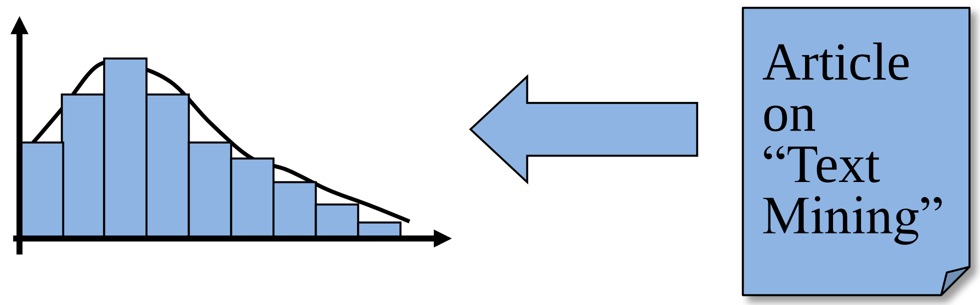
- Chain Rule (requires long chains of cond. prob.):
  $P(t_1 t_2 t_3 t_4) = P(t_1) P(t_2 | t_1) P(t_3 | t_1 t_2) P(t_4 | t_1 t_2 t_3)$
- Bigram LM (pairwise cond. prob.):
  $P_{bi}(t_1 t_2 t_3 t_4) = P(t_1) P(t_2 | t_1) P(t_3 | t_2) P(t_4 | t_3)$
- Unigram LM (no cond. prob.):
  $P_{uni}(t_1 t_2 t_3 t_4) = P(t_1) P(t_2) P(t_3) P(t_4)$

Text Generation with (Unigram) LM

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

LM for topic 1: *IR&DM*

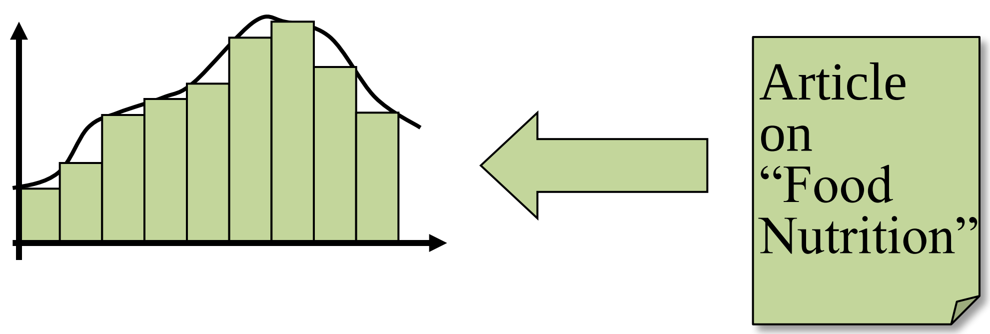
text	0.2
mining	0.1
n-gram	0.01
cluster	0.02
...	
healthy	0.000001
…	



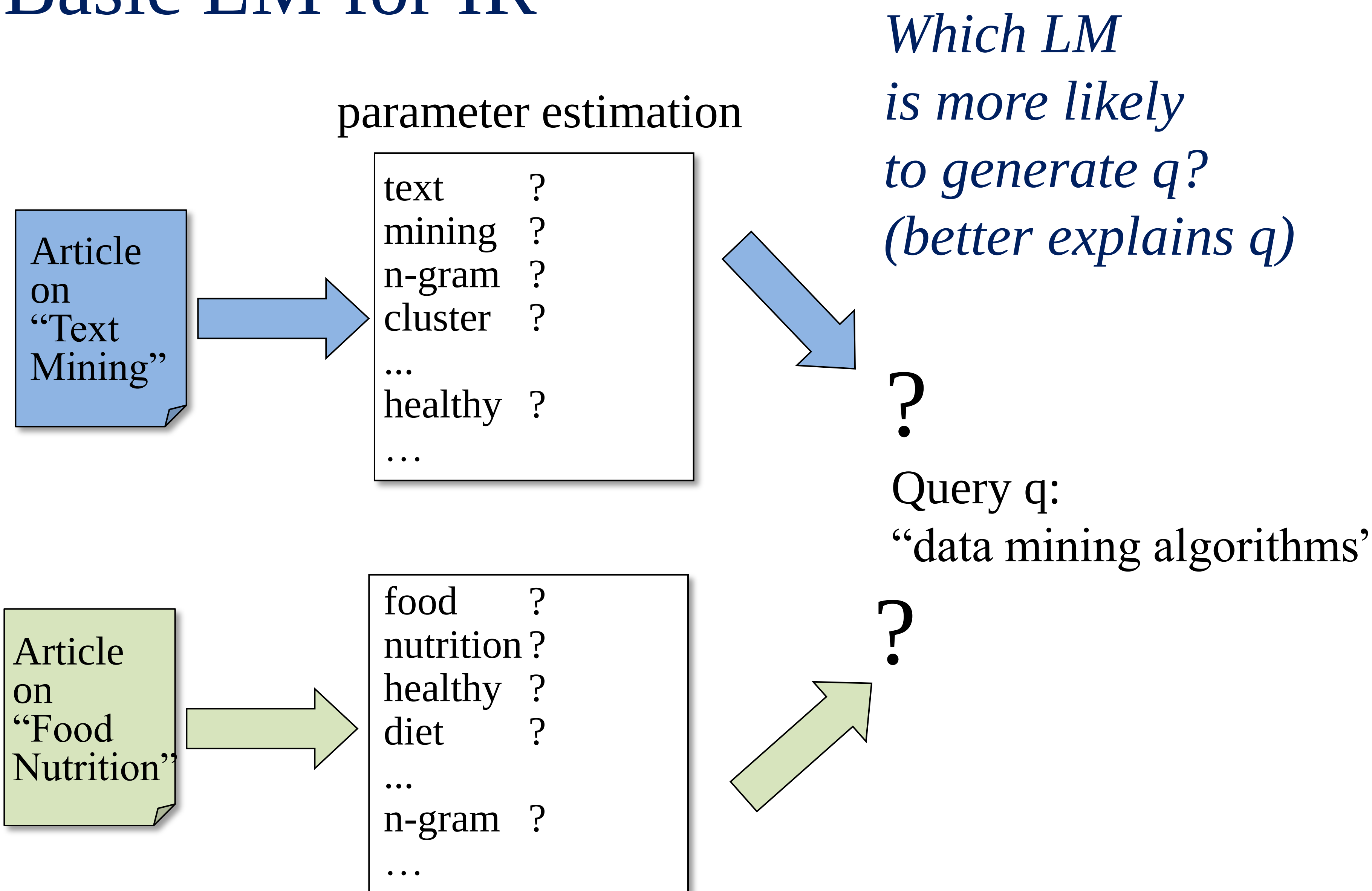
different θ_d for different d

LM for topic 2: *Health*

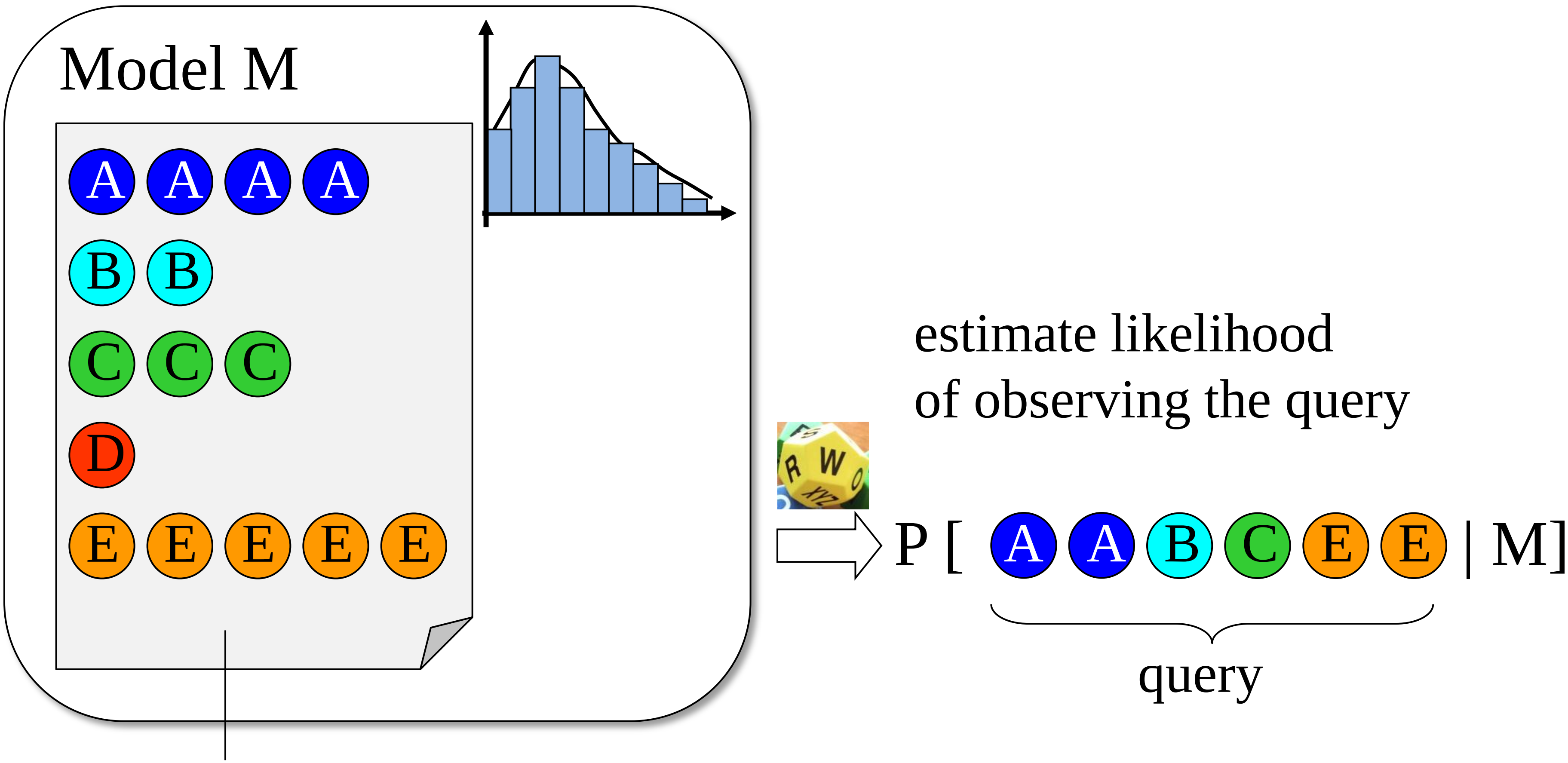
food	0.25
nutrition	0.1
healthy	0.05
diet	0.02
...	
n-gram	0.00002
…	



Basic LM for IR

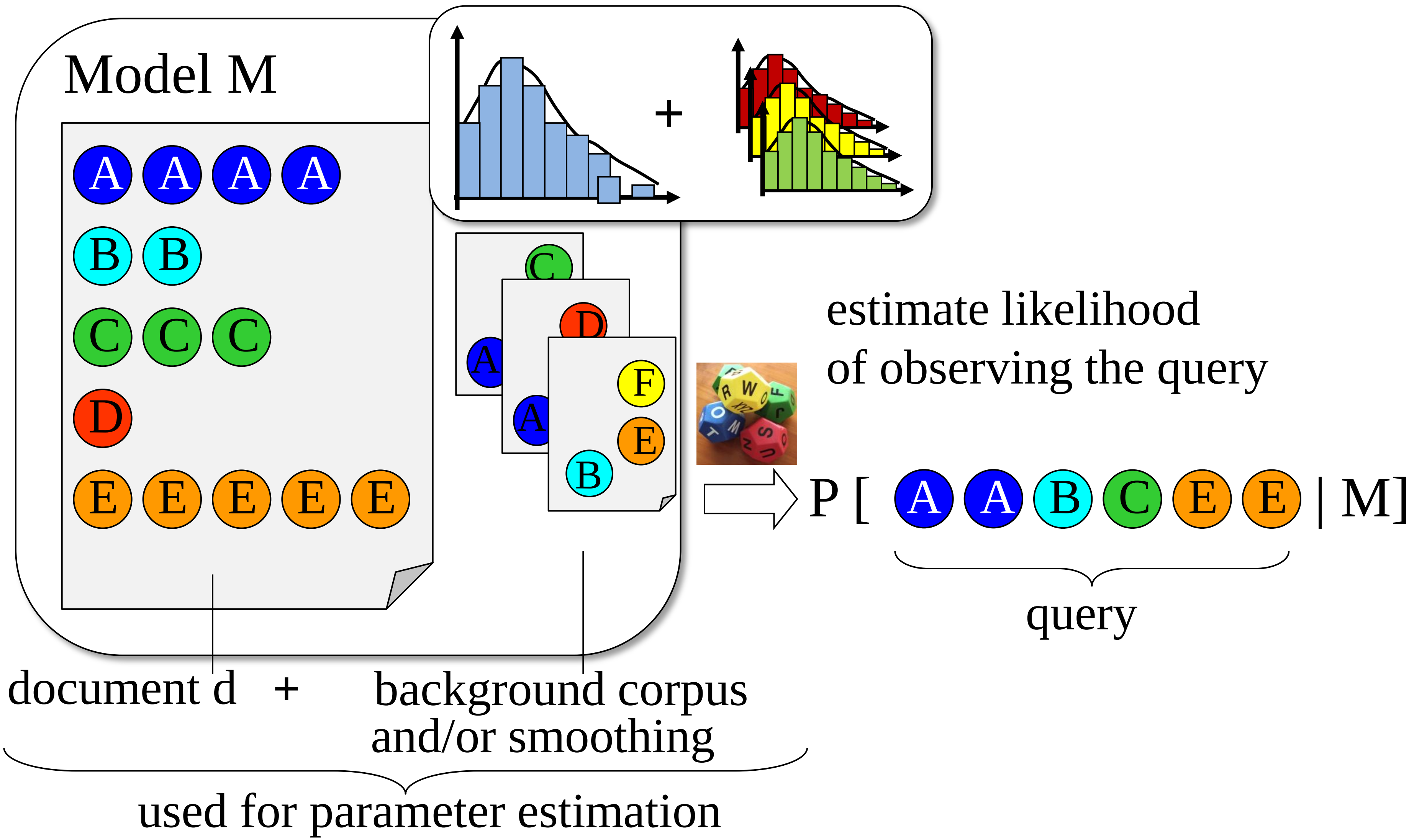


LM Illustration:
Document as Model and Query as Sample



document d: sample of M
used for parameter estimation

LM Illustration:
Document as Model and Query as Sample



Prob.-IR vs. Language Models

$P[R|d,q]$

User likes doc (R) given that it has features d and user poses query q

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

prob. IR

(ranking proportional to relevance odds)

$$\propto P[q, d \mid R] \cdot P[R]$$
$$= P[q \mid d, R] \cdot P[d \mid R] \cdot P[R]$$
$$\propto P[q \mid d]$$

statist. LM

(ranking proportional to query likelihood)

query likelihood:

$$s(q,d) = \log P[q \mid d] = \sum_{j \in q} \log P[j \mid \theta_d]$$

top-k query result:

$$k\text{-}\operatorname{argmax}_d \log P[q \mid d]$$

MLE would be $tf_j / |d|$

Multi-Bernoulli vs. Multinomial LM

Multi-Bernoulli:

$$P[q \mid d] = \Pi_j \; p_j(d)^{X_j(q)} \cdot (1 - p_j(d))^{1 - X_j(q)}$$

with $X_j(q)=1$ *if* $j \in q$, *0 otherwise*

Multinomial:

$$P[q \mid d] = \binom{|q|}{f(j_1)\, f(j_2) \ldots f(j_{|q|})} \Pi_{j \in q}\, p_j(d)^{f_j(q)}$$

with $f_j(q) = f(j) =$ *frequency of j in q* and $\sum_j f(j) = |q|$

multinomial LM more expressive and usually preferred

LM Scoring by Kullback-Leibler Divergence

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

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

$$= -H(f(q), p(d))$$ neg. cross-entropy

$$\propto -H(f(q), p(d)) + H(f(q))$$ neg. cross-entropy + entropy

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

$$= -\sum_j f_j(q) \log_2 \frac{f_j(q)}{p_j(d)}$$ neg. KL divergence of θ_q and θ_d

III.4.2 Smoothing Methods

Absolutely crucial to avoid overfitting and make LMs useful in practice (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 mostly "black art" (or empirical)

Laplace Smoothing and Absolute Discounting

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

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

Additive Laplace smoothing:

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

for multinomial over vocabulary W with |W|=m

Absolute discounting:

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

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

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

Jelinek-Mercer Smoothing

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

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

could also consider query log as background LM for query

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

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

Jelinek-Mercer Smoothing: Relationship to TF*IDF

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

$$\propto \sum_{i \in q} \log \left(\lambda \frac{tf(i,d)}{\sum_k tf(k,d)} + (1-\lambda) \frac{df(i)}{\sum_k df(k)} \right)$$

with absolute frequencies tf, df

$$\propto \sum_{i \in q} \log \left(1 + \underbrace{\frac{tf(i,d)}{\sum_k tf(k,d)}}_{\text{relative tf}} \cdot \frac{\lambda}{1-\lambda} \underbrace{\frac{\sum_k df(k)}{df(i)}}_{\sim \text{ relative idf}} \right)$$

relative tf ~ relative idf

Dirichlet-Prior Smoothing

$\sim Dirichlet(\alpha)$
prior

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

$\sim Dirichlet(f + \alpha)$
posterior

MAP for θ with Dirichlet distribution as prior

with term frequencies f in document d

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

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

Dirichlet: $f(\theta_1, \ldots, \theta_m; \alpha_1, \ldots, \alpha_m) = \frac{\Pi_{j=1..m} \Gamma(\alpha_j)}{\Gamma(\Sigma_{j=1..m} \alpha_j)} \Pi_{j=1..m} \theta_j^{\alpha_j - 1}$ 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: Relationship to Jelinek-Mercer Smoothing

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

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

tf

α_j from corpus

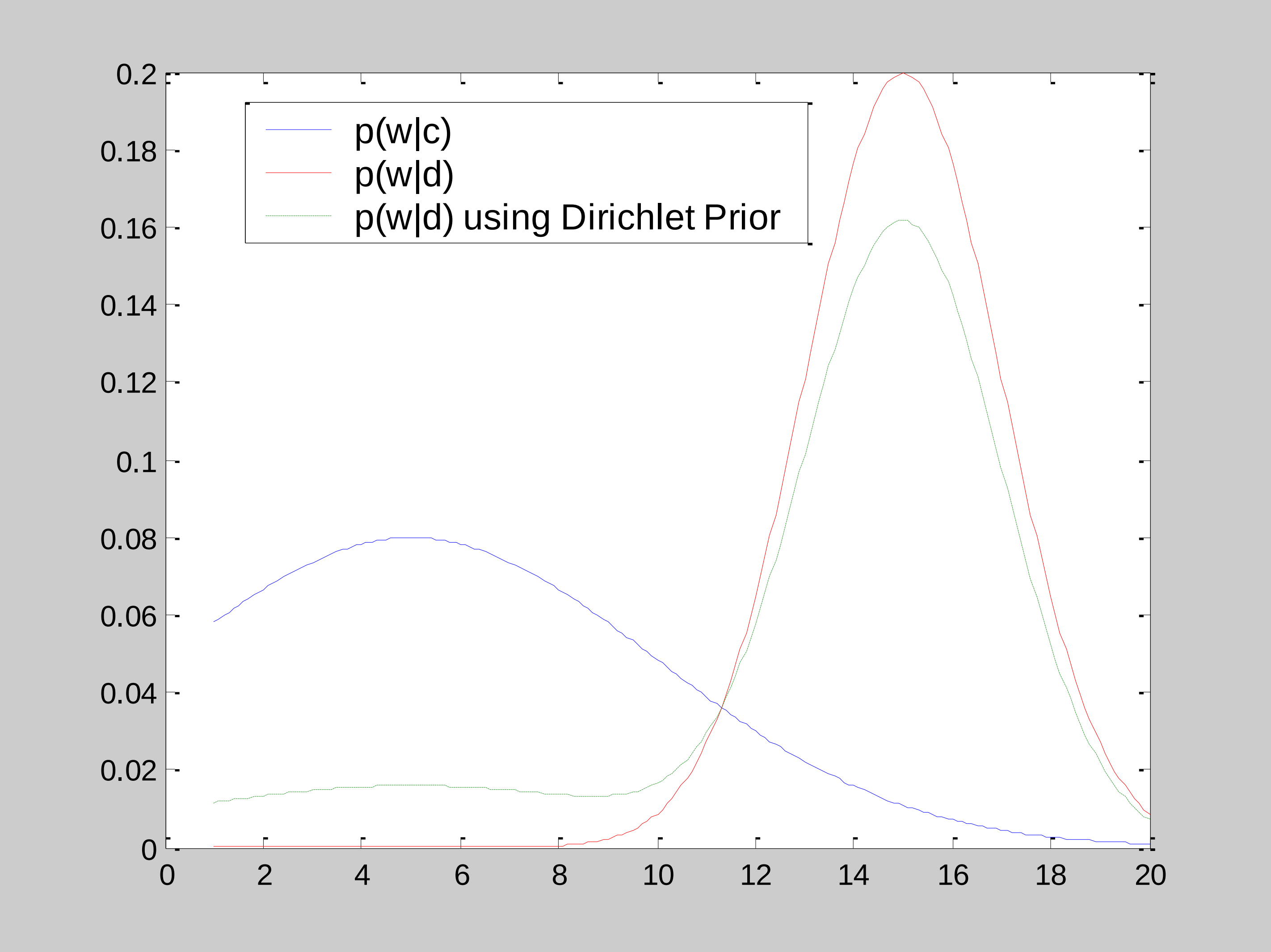
with MLEs P[j|d], P[j|C]

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

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

→ Jelinek-Mercer special case of Dirichlet!

Effect of Dirichlet Smoothing



Source: Rong Jin, Language Modeling Approaches for Information Retrieval, http://www.cse.msu.edu/~cse484/lectures/lang_model.ppt

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

Query =	"the	algorithms	for	data	mining"
d1:	0.04	0.001	0.02	0.002	0.003
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)

But: p(q|d1) > p(q|d2) !

We should make p("the") and p("for") less different for all docs.

→Combine Dirichlet (good at short keyword queries)
and Jelinek-Mercer smoothing (good at verbose queries)!

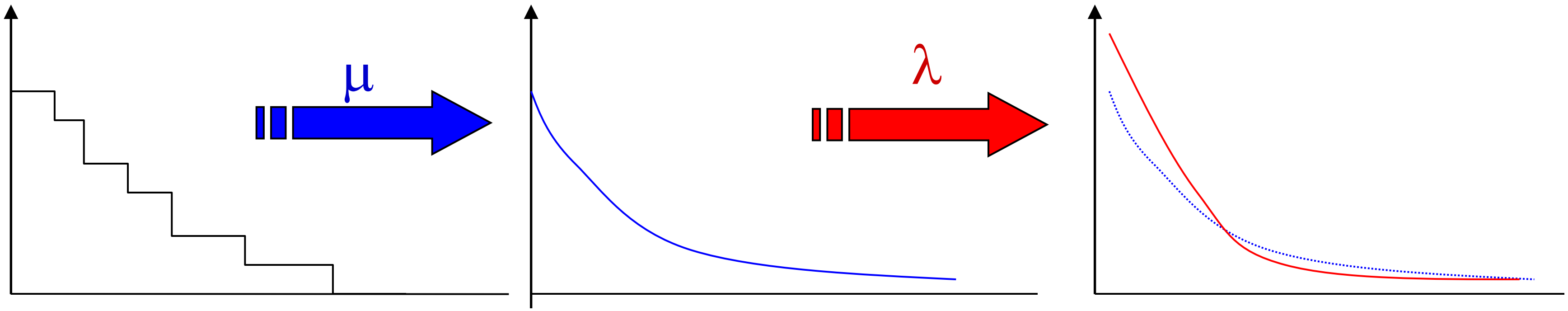
Two-Stage Smoothing [Zhai/Lafferty, TOIS 2004]

Stage-1

-Explain unseen words
-Dirichlet prior

Stage-2

-Explain noise in query
-2-component mixture



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

U: user's background LM, or approximated by corpus LM C

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

III.4.3 Extended LMs

Large variety of extensions:

- Term-specific smoothing
  (JM with term-specific λ_j, e.g., based on idf values)
- Parsimonious LM
  (JM-style smoothing with smaller feature space)
- N-gram (Sequence) Models (e.g. HMMs)
- (Semantic) Translation Models
- Cross-Lingual Models
- Query-Log- & Click-Stream-based LM
- LMs for Question Answering

(Semantic) Translation Model

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

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

<u>Opportunities and difficulties:</u>

- synonymy, hypernymy/hyponymy, polysemy
- efficiency
- training

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

Translation Models for Cross-Lingual IR

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

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

Can rank docs in E (or F) for queries in F

Example: q: “moteur de recherche”
returns
d: “Quaero is a French initiative for developing a search engine that can serve as a European alternative to Google ... ”

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

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

Query-Log-Based LM (User LM)

<u>Idea:</u>

For current query $q_{k,}$ leverage the following:

- 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

<u>Example:</u>

q_k = "Java library" benefits from q_{k-1} = "cgi programming"

Simple Mixture Model with Fixed Coefficient Interpolation:

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

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

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

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

More advanced models with Dirichlet priors in the literature…

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

query: keywords → answer: entities

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

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

Query q:
"Dutch soccer player Barca"

Candidate entities:

- e_1: *Johan Cruyff*
- e_2: *Ruud van Nistelroy*
- e_3: *Ronaldinho*
- e_4: *Zinedine Zidane*
- e_5: *FC Barcelona*

e_1 (Johan Cruyff):
Dutch goalgetter soccer champion
Dutch player Ajax Amsterdam
trainer Barca 8 years Camp Nou
played soccer FC Barcelona
Jordi Cruyff son

e_4 (Zinedine Zidane):
Zizou champions league 2002
Real Madrid van Nistelroy Dutch
soccer world cup best player
2005 lost against Barca

Additionally weighted by extraction accuracy

Language Models for Question Answering (QA)

question

↓ question-type-specific NL parsing

query

↓ finding most promising short text passages

passages

↓ NL parsing and entity extraction

answers

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

Example:
Where is the Louvre museum located?

Q: 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.
...

A: The Louvre museum is in Paris.

Use of LMs:

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

LM for Temporal Search

Keyword queries that express temporal interest
Example: q = “FIFA world cup <u>1990s</u>”
→ would not retrieve doc
d = “France won the FIFA world cup <u>in 1998</u>”

<u>Approach:</u>

- extract temporal phrases from docs
- normalize temporal expressions
- split query and docs into text × time

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

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

(plus smoothing)

$$P[x \mid y] := \frac{|x \cap y|}{|y|} \quad \text{with } |x| = end(x) - begin(x)$$

Summary of Section III.4

- 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 specialized IR: cross-lingual, passages, etc.

Additional Literature for Section III.4

Statistical Language Models in General:

- Manning/Raghavan/Schütze book, Chapter 12
- Djoerd Hiemstra: Language Models, Smoothing, and N-grams, in: Encyclopedia of Database Systems, Springer, 2009
- Cheng Xiang Zhai, Statistical Language Models for Information Retrieval, Morgan & Claypool Publishers, 2008
- Cheng Xiang Zhai, Statistical Language Models for Information Retrieval: A Critical Review, Foundations and Trends in Information Retrieval 2(3), 2008
- X. Liu, W.B. Croft: Statistical Language Modeling for Information Retrieval, Annual Review of Information Science and Technology 39, 2004
- J. Ponte, W.B. Croft: A Language Modeling Approach to Information Retrieval, SIGIR 1998
- C. Zhai, J. Lafferty: A Study of Smoothing Methods for Language Models Applied to Information Retrieval, TOIS 22(2), 2004
- C. Zhai, J. Lafferty: A Risk Minimization Framework for Information Retrieval, Information Processing and Management 42, 2006
- M.E. Maron, J.L. Kuhns: On Relevance, Probabilistic Indexing, and Information Retrieval, Journal of the ACM 7, 1960

Additional Literature for Section III.4

LMs for Specific Retrieval Tasks:

- X. Shen, B. Tan, C. Zhai: Context-Sensitive Information Retrieval Using Implicit Feedback, SIGIR 2005
- Y. Lv, C. Zhai, Positonal Language Models for Information Retrieval, SIGIR 2009
- V. Lavrenko, M. Choquette, W.B. Croft: Cross-lingual relevance models. SIGIR'02
- D. Nguyen, A. Overwijk, C. Hauff, D. Trieschnigg, D. Hiemstra, F. de Jong: WikiTranslate: Query Translation for Cross-Lingual Information Retrieval Using Only Wikipedia. CLEF 2008
- C. Clarke: Web Question Answering. Encyclopedia of Database Systems 2009
- C. Clarke, E.L. Terra: Passage retrieval vs. document retrieval for factoid question answering. SIGIR 2003
- D. Shen, J.L. Leidner, A. Merkel, D. Klakow: The Alyssa System at TREC 2006: A Statistically-Inspired Question Answering System. TREC 2006
- Z. Nie, Y. Ma, S. Shi, J.-R. Wen, W.-Y. Ma: Web object retrieval. WWW 2007
- H. Zaragoza et al.: Ranking very many typed entities on wikipedia. CIKM 2007
- P. Serdyukov, D. Hiemstra: Modeling Documents as Mixtures of Persons for Expert Finding. ECIR 2008
- S. Elbassuoni, M. Ramanath, R. Schenkel, M. Sydow, G. Weikum: Language-model-based Ranking for Queries on RDF-Graphs. CIKM 2009
- K. Berberich, O. Alonso, S. Bedathur, G. Weikum: A Language Modeling Approach for Temporal Information Needs. ECIR 2010