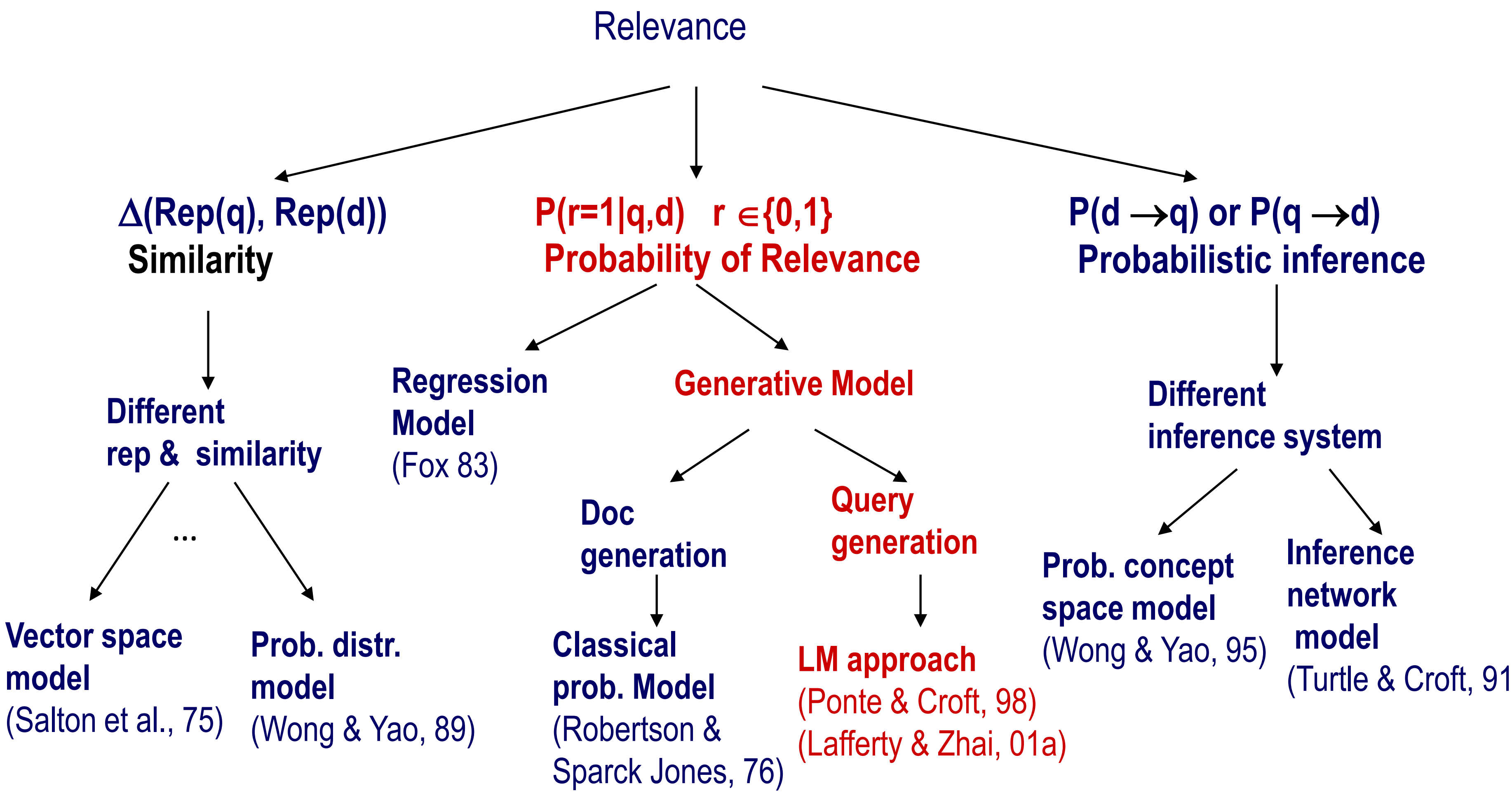
Language Models

Hongning Wang
CS@UVa

Notion of Relevance



What is a statistical LM?

- A model specifying probability distribution over word <u>sequences</u>
  - p("*Today is Wednesday*") ≈ 0.001
  - p("*Today Wednesday is*") ≈ 0.0000000000001
  - p("*The eigenvalue is positive*") ≈ 0.00001
- It can be regarded as a probabilistic mechanism for "generating" text, thus also called a "generative" model

Why is a LM useful?

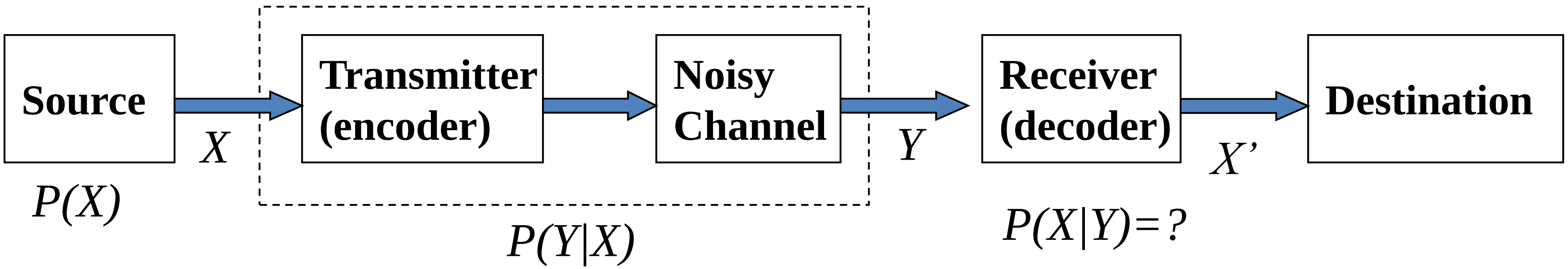
- Provides a principled way to quantify the uncertainties associated with natural language
- Allows us to answer questions like:
  - Given that we see "*John*" and "*feels*", how likely will we see "*happy*" as opposed to "*habit*" as the next word?

    (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 [Shannon 48]



$$\hat{X} = \arg\max_{X} p(X \mid Y) = \arg\max_{X} p(Y \mid X)\, p(X) \quad \textbf{(Bayes Rule)}$$

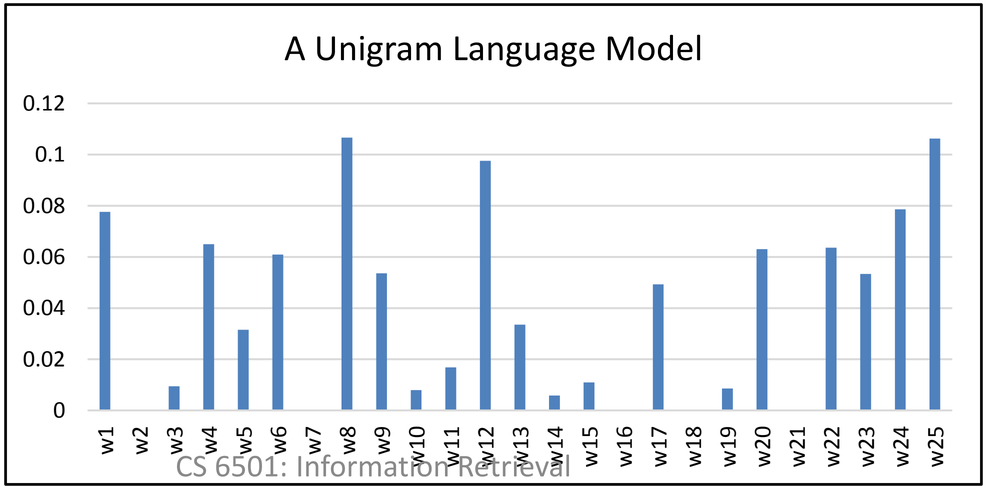
When X is text, p(X) is a language model

Many Examples:

Speech recognition:	**X=Word sequence**	**Y=Speech signal**
Machine translation:	**X=English sentence**	**Y=Chinese sentence**
OCR Error Correction:	**X=Correct word**	**Y= Erroneous word**
Information Retrieval:	**X=Document**	**Y=Query**
Summarization:	**X=Summary**	**Y=Document**

Unigram language model

- Generate a piece of text by generating each word <u>independently</u>
  - $p(w_1\ w_2\ \dots\ w_n) = p(w_1)p(w_2)\dots p(w_n)$
  - $s.t. \{p(w_i)\}_{i=1}^{N}, \sum_i p(w_i) = 1, p(w_i) \geq 0$
- Essentially a multinomial distribution over the vocabulary



The simplest and most popular choice!

More sophisticated LMs

- N-gram language models
  - In general, $p(w_1\, w_2\ \dots w_n) = p(w_1)p(w_2|w_1) \dots p(w_n|w_1 \dots w_{n-1})$
  - n-gram: conditioned only on the past n-1 words
  - E.g., bigram: $p(w_1\ \dots\ w_n) = p(w_1)p(w_2|w_1)\, p(w_3|w_2)\ \dots p(w_n|w_{n-1})$
- Remote-dependence language models (e.g., Maximum Entropy model)
- Structured language models (e.g., probabilistic context-free grammar)

Why just unigram models?

- Difficulty in moving toward more complex models
  - They involve more parameters, so need more data to estimate
  - They increase the computational complexity significantly, both in time and space
- Capturing word order or structure may not add so much value for “topical inference”
- But, using more sophisticated models can still be expected to improve performance ...

Generative view of text documents

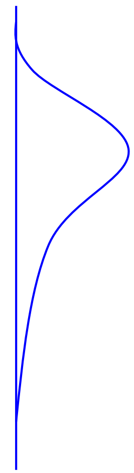
(Unigram) Language Model θ
p(w| θ)

Sampling

Document

Topic 1:
Text mining

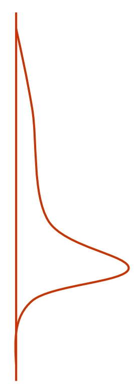
…
text 0.2
mining 0.1
assocation 0.01
clustering 0.02
…
food 0.00001
…



Text mining document

Topic 2:
Health

…
text 0.01
food 0.25
nutrition 0.1
healthy 0.05
diet 0.02
…



Food nutrition document

Estimation of language models

- Maximum likelihood estimation

~~Unigram~~ Language Model θ

$p(w|\theta)=?$

Estimation

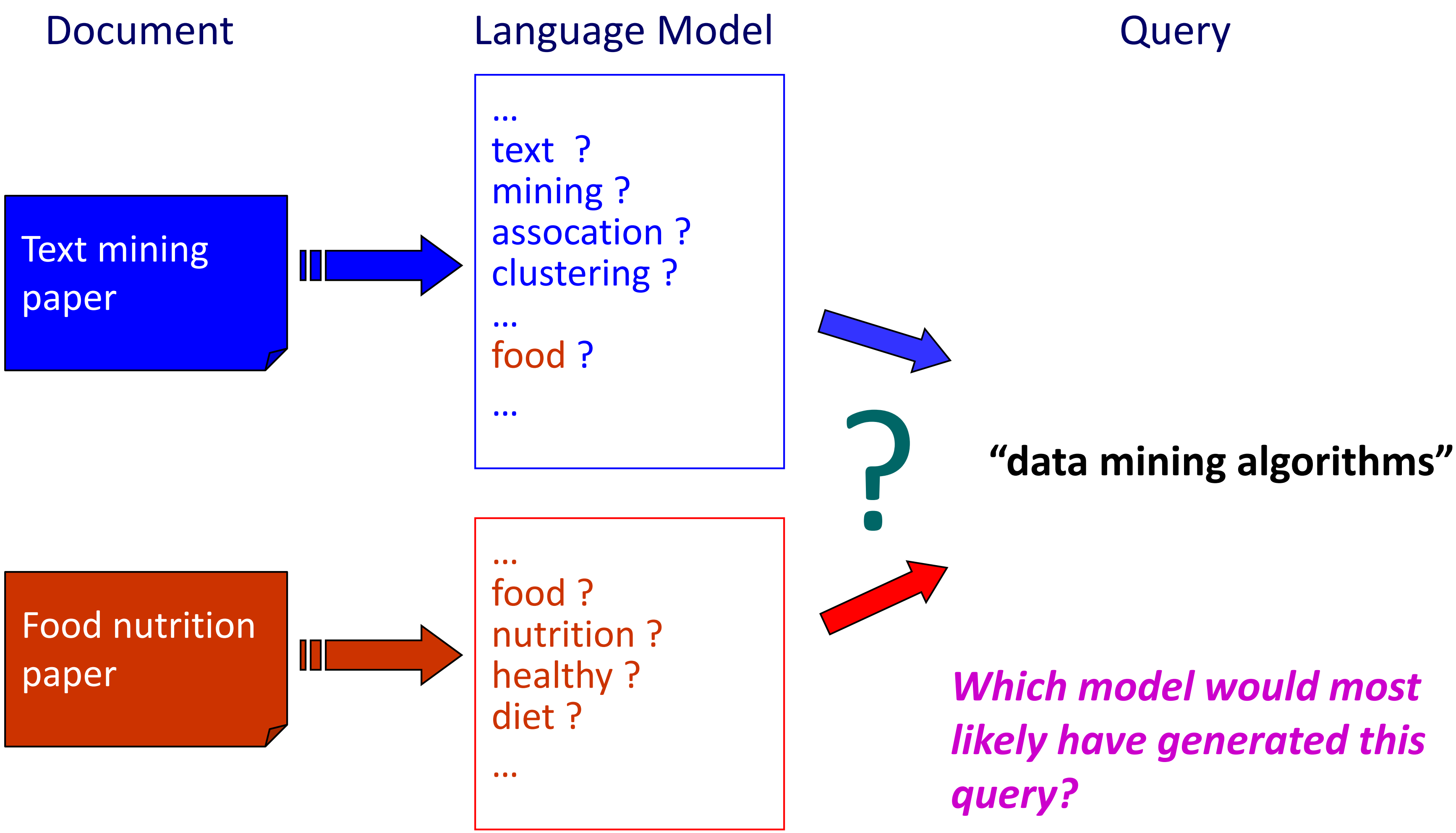
Document

	…
10/100	text ?
5/100	mining ?
3/100	assocation ?
3/100	database ?
	…
1/100	query ?
	…

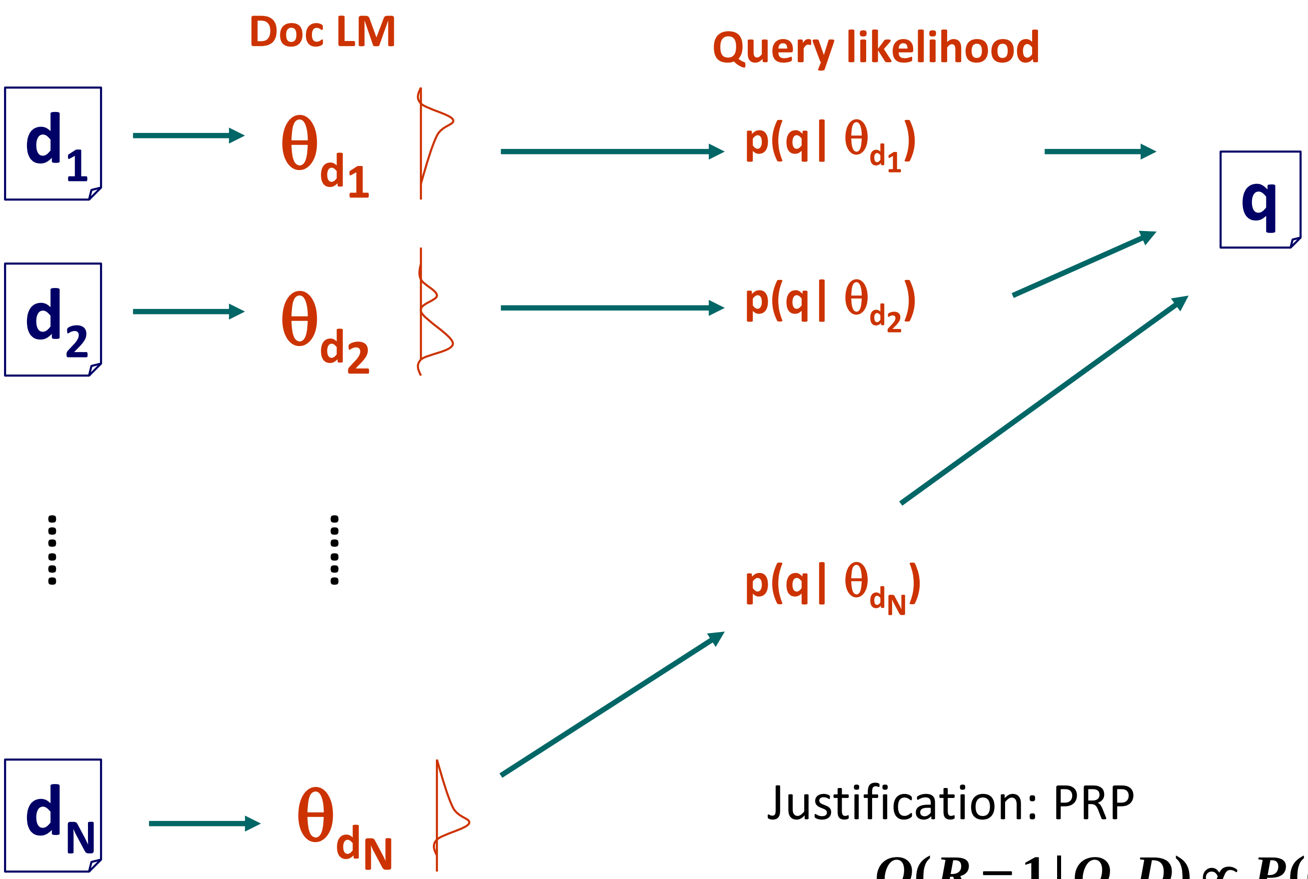
text 10
mining 5
association 3
database 3
algorithm 2
…
query 1
efficient 1

A “text mining” paper (total #words=100)

Language models for IR
[Ponte & Croft SIGIR’98]



Ranking docs by query likelihood



Justification: PRP

$$O(R=1|Q,D) \propto P(Q|D,R=1)$$

Justification from PRP

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

Query generation

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

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

Query likelihood p(q| θ_d) **Document prior**

Assuming uniform document prior, we have

$$O(R=1|Q,D) \propto P(Q|D,R=1)$$

Retrieval as language model estimation

- Document ranking based on *query likelihood*

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

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

Document language model

  - Retrieval problem $\approx$ Estimation of $p(w_i|d)$
  - Common approach
    - Maximum likelihood estimation (MLE)

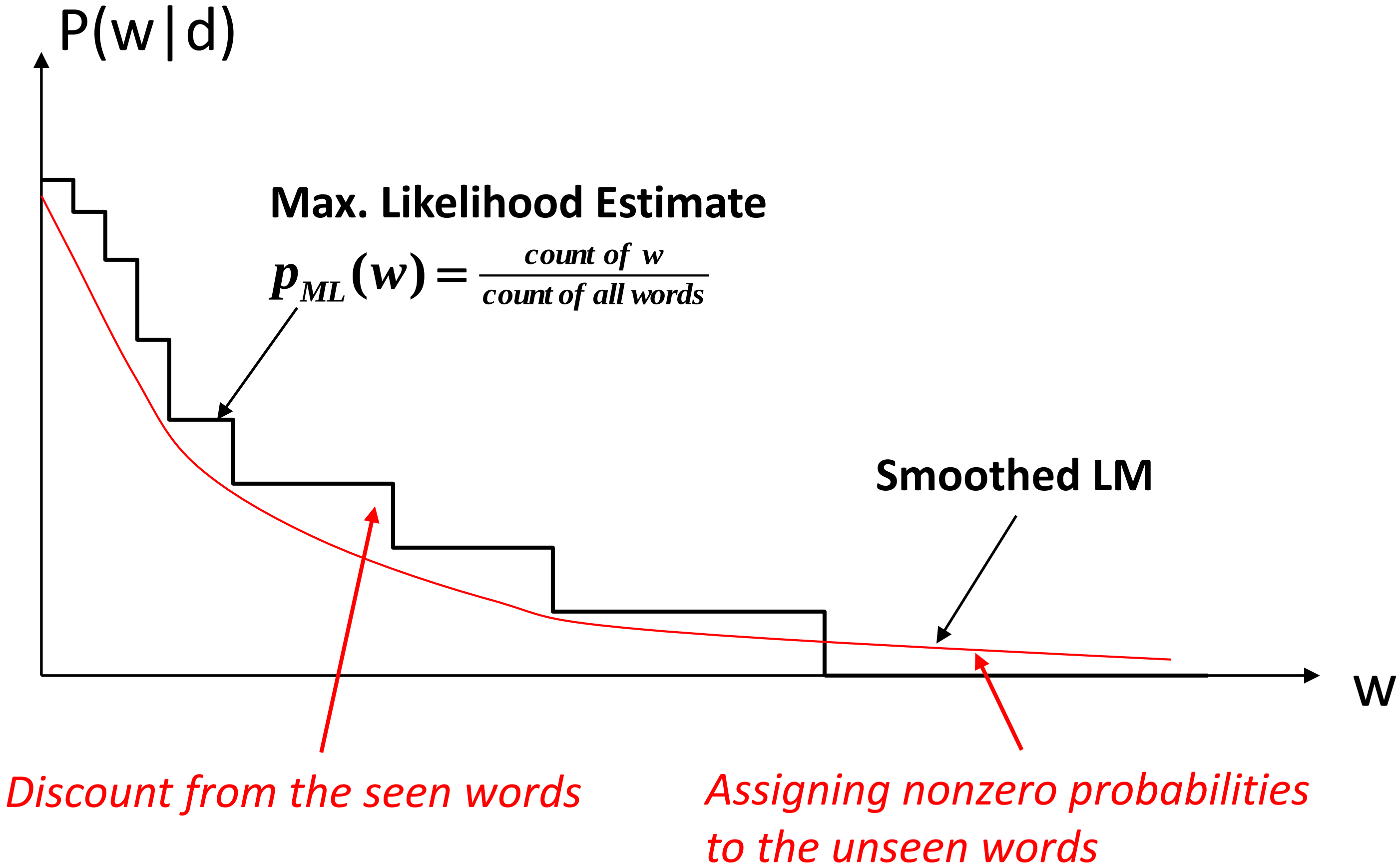
Problem with MLE

- What probability should we give a word that has not been observed in the document?
  - log0?
- If we want to assign non-zero probabilities to such words, we'll have to discount the probabilities of observed words
- This is so-called "smoothing"

General idea of smoothing

- All smoothing methods try to
  1. Discount the probability of words seen in a document
  2. Re-allocate the extra counts such that unseen words will have a non-zero count

Illustration of language model smoothing



Smoothing methods

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

Counts of w in d

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

"Add one", Laplace smoothing

Vocabulary size

Length of d (total counts)

  - Problems?
    - Hint: all words are equally important?

Refine the idea of smoothing

- Should all unseen words get equal probabilities?
- 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

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

Smoothing methods (cont)

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

 # uniq words

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

  - Problems?
    - Hint: varied document length?

Smoothing methods (cont)

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

$$p(w|d) = (1-\lambda)\underbrace{\boxed{\frac{c(w,d)}{|d|}}}_{\text{MLE}} + \underbrace{\lambda}_{\text{parameter}}\, p(w|REF)$$

  - Problems?
    - Hint: what is missing?

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

  - Problems?

Dirichlet prior smoothing

likelihood of doc given the model

prior over models

- Bayesian estimator
  - Posterior of LM: $p(\theta|d) \propto p(d|\theta)p(\theta)$
- Conjugate prior
  - Posterior will be in the same form as prior
  - Prior can be interpreted as “extra”/“pseudo” data
- Dirichlet distribution is a conjugate prior for multinomial distribution

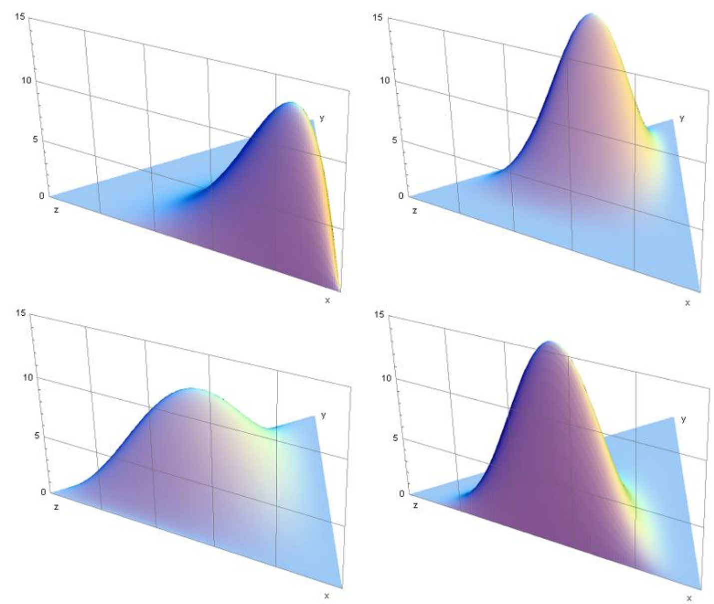
$$Dir(\theta \mid \underline{\alpha_1, \ldots, \alpha_N}) = \frac{\Gamma(\alpha_1 + \cdots + \alpha_N)}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_N)} \prod_{i=1}^{N} \theta_i^{\alpha_i - 1}$$

“extra”/“pseudo” word counts, we set $\alpha_i = \mu\, p(w_i|REF)$

Some background knowledge

- Conjugate prior
  - Posterior dist in the same family as prior
- Dirichlet distribution
  - Continuous
  - Samples from it will be the parameters in a multinomial distribution

Gaussian -> Gaussian
Beta -> Binomial
Dirichlet -> Multinomial



Dirichlet prior smoothing (cont)

Posterior distribution of parameters:

$$p(\theta \mid d) = Dir(\theta \mid c(w_1)+\alpha_1, \ldots, c(w_N)+\alpha_N)$$

$$\textbf{Property : If } \theta \sim Dir(\theta \mid \alpha), \textbf{ then } \mathrm{E}(\theta) = \{\frac{\alpha_i}{\sum \alpha_i}\}$$

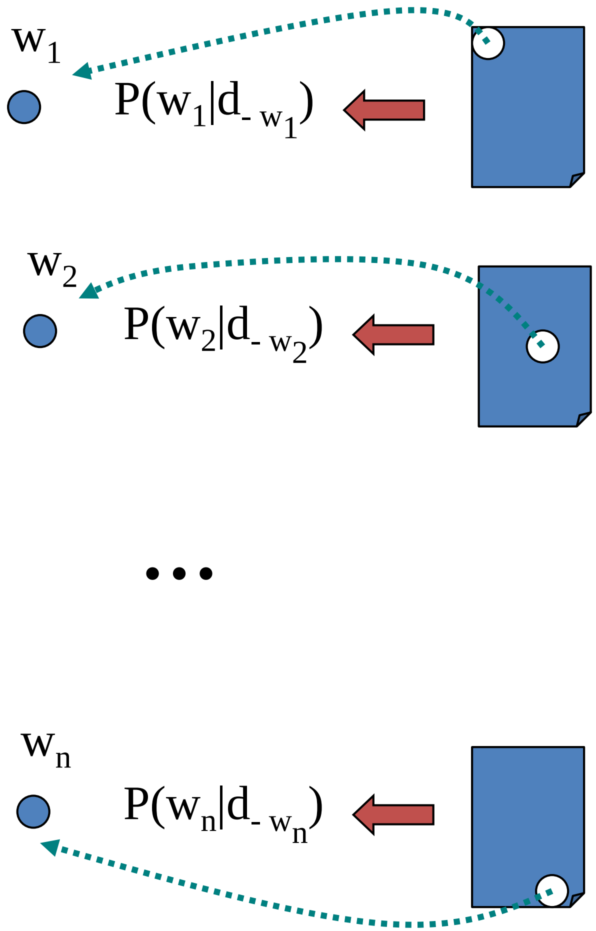
The predictive distribution is the same as the mean:

$$\mathbf{p}(\mathbf{w}_\mathrm{i} \mid \hat{\theta}) = \int \mathbf{p}(\mathbf{w}_\mathrm{i} \mid \theta) Dir(\theta \mid \alpha) d\theta$$

$$= \frac{c(\mathbf{w}_\mathrm{i}) + \alpha_i}{|\mathbf{d}| + \sum_{\mathrm{i}=1}^{N} \alpha_i} = \boxed{\frac{c(\mathbf{w}_\mathrm{i}) + \mu p(w_i \mid REF)}{|\mathbf{d}| + \mu}}$$

Dirichlet prior smoothing

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



Leave-one-out

log-likelihood

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

Maximum Likelihood Estimator

$$\hat{\mu} = \underset{\mu}{\operatorname{argmax}} \ L_{-1}(\mu | C)$$

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

Now, suppose we leave "e" out…

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

μ doesn't have to be big

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

Understanding smoothing

Content words

	Query = "the	algorithms	for	data	mining"	
$p_{ML}(w\|d1)$:	*0.04*	*0.001*	*0.02*	*0.002*	*0.003*	4.8×10^{-12}
$p_{ML}(w\|d2)$:	*0.02*	*0.001*	*0.01*	*0.003*	*0.004*	2.4×10^{-12}

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 to achieve this goal…

2.35×10^{-13}

4.53×10^{-13}

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

Understanding smoo

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

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

TF weighting

Doc length normalization
(longer doc is expected to have a smaller α_d)

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

IDF weighting

Ignore for ranking

- **Smoothing with *p(w|C)* ≈ TF-IDF + doc-length normalization**

Smoothing & TF-IDF weighting

Smoothed ML estimate

Retrieval formula using the general smoothing scheme

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

Reference language model

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

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

Key rewriting step (where did we see it before?)

Similar rewritings are very common when using probabilistic models for IR…

What you should know

- How to estimate a language model
- General idea and different ways of smoothing
- Effect of smoothing