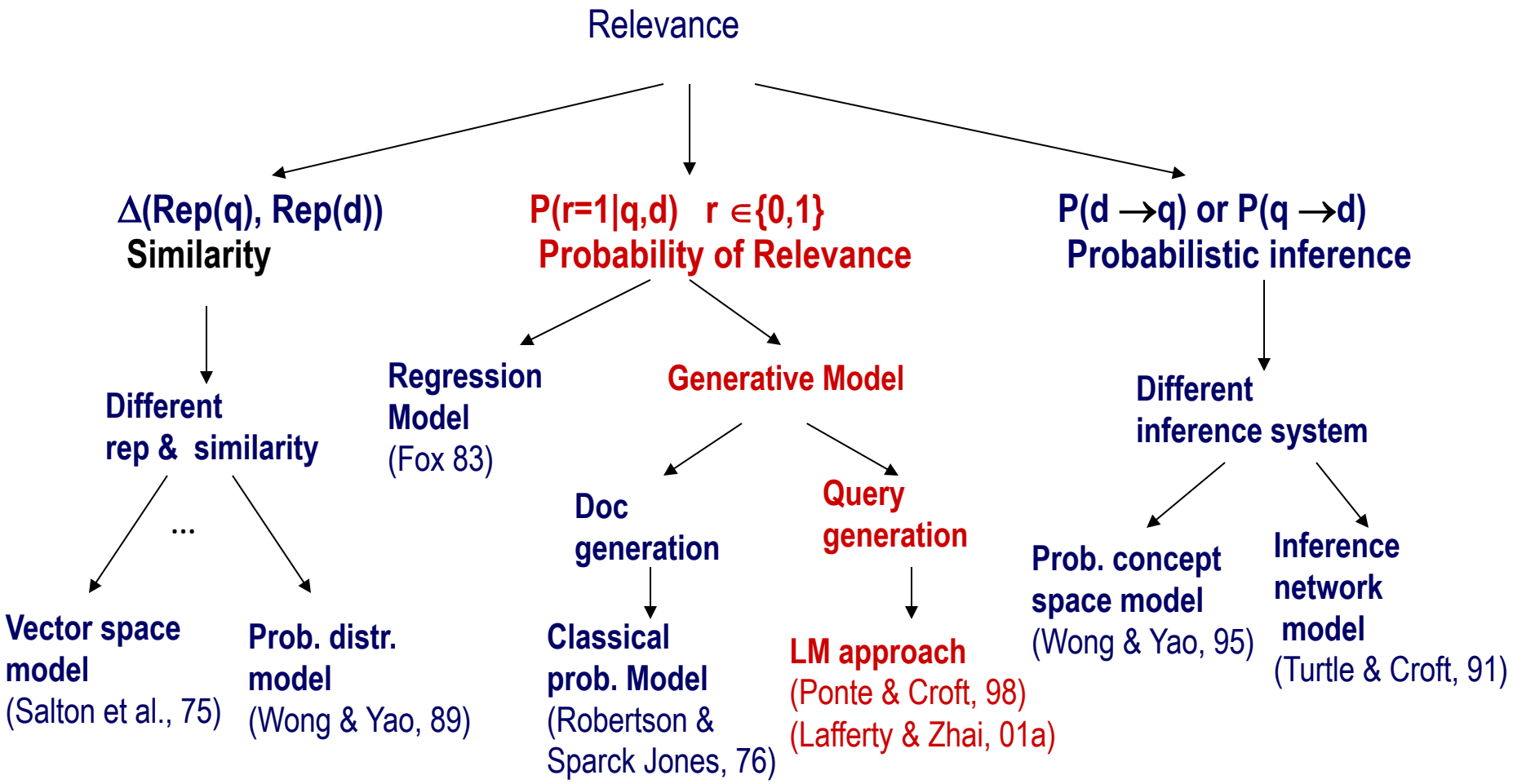
Language Models

Hongning Wang
CS@UVa

Notion of Relevance



What is a statistical LM?

- A model specifying a probability distribution over word sequences
  - p("*Today is Wednesday*") ≈ 0.001
  - p("*Today Wednesday is*") ≈ 0.000000000001
  - 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)

Language model for text

We need independence assumptions!

- Probability distribution over word sequences
  - $p(w_1\, w_2\, \ldots\, w_n) =$
    $p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \ldots p(w_n|w_1, w_2, \ldots, w_{n-1})$
  - Complexity - $O(V^{n^*})$
    - n^* - maximum ~~document~~ length sentence

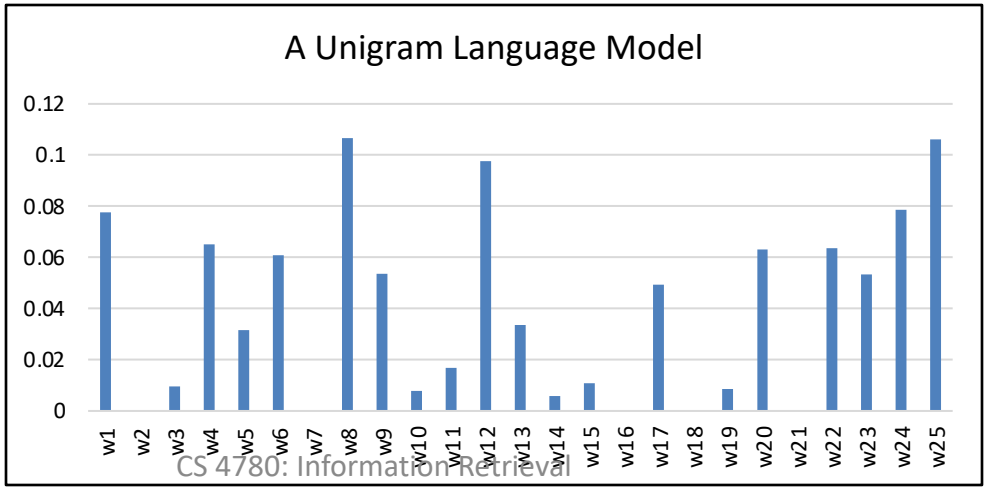
Chain rule: from conditional probability to joint probability

- 475,000 main headwords in Webster's Third New International Dictionary
- Average English sentence length is 14.3 words
- A rough estimate: $O(475000^{14})$

How large is this? $\frac{475000^{14}}{8bytes \times (1024)^4} \approx 3.38e^{66}TB$

Unigram language model

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



The simplest and most popular choice!

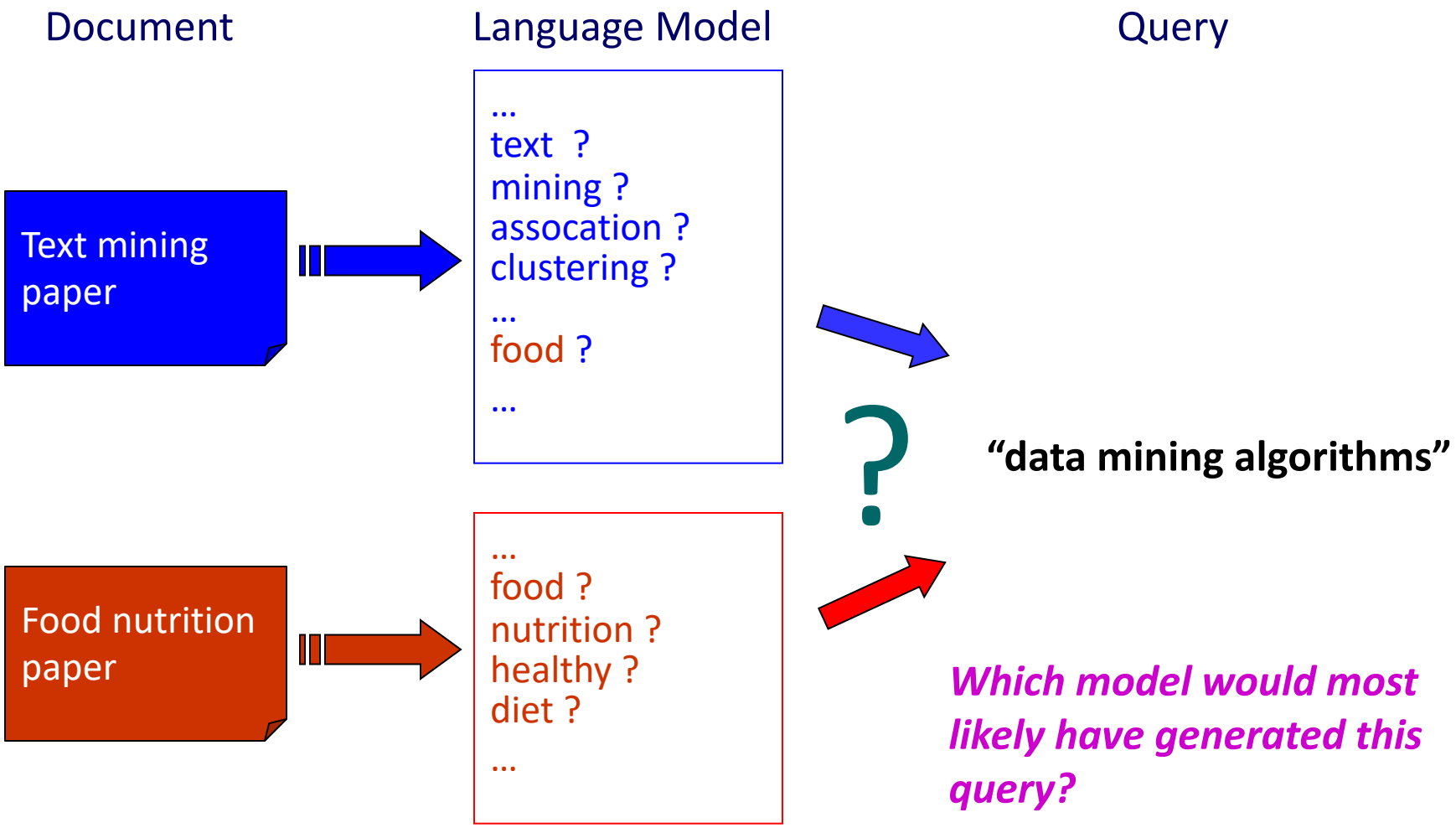
More sophisticated LMs

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

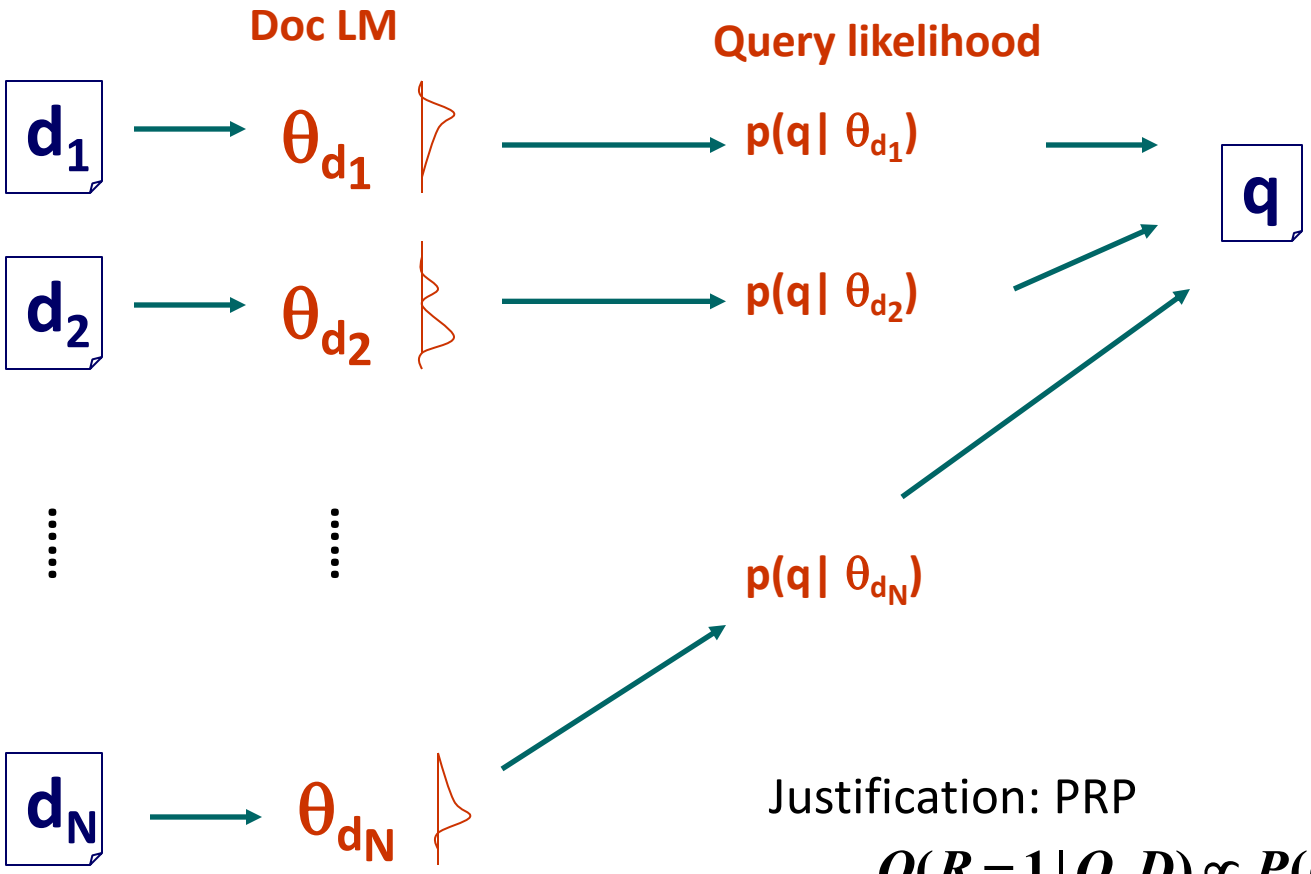
Why just unigram models?

- 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 ...

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



Ranking docs by query likelihood



Justification: PRP

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

Retrieval as language model estimation

- Document ranking based on *query likelihood*

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

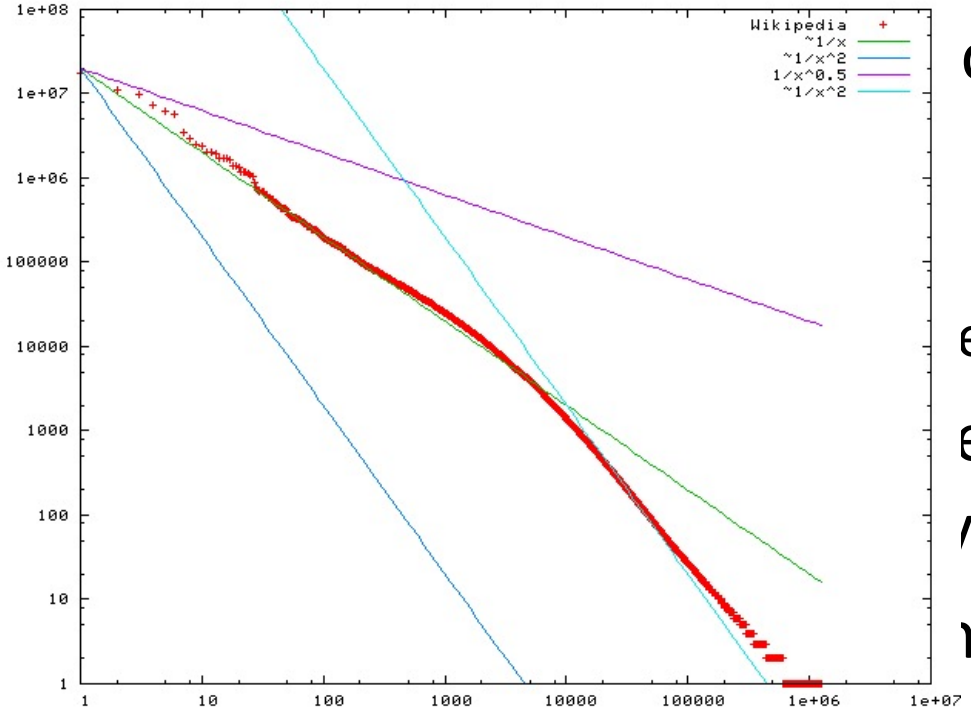
$$where,\ \ q = w_1 w_2 \ldots w_n$$

Document language model

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

Problem with MLE

- Unseen events $f(k; s, N) \propto 1/k^s$
  - There ction of Yelp rev
    - (
    - (
  - ➢ Thi es not occur in the with MLE!
  - ➢ No fu ntain those unseen words/

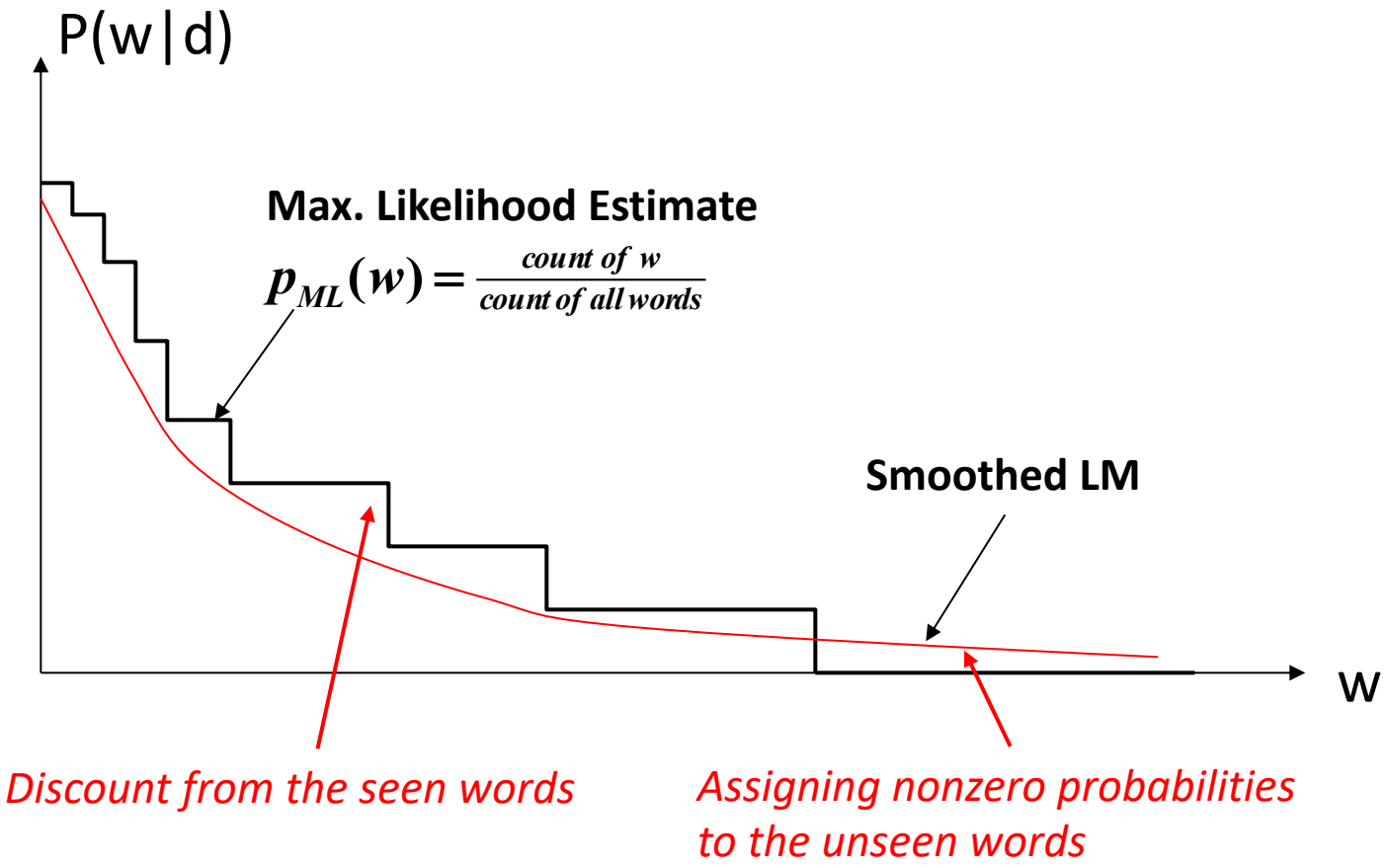


A plot of word frequency in Wikipedia (Nov 27, 2006)

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"

Illustration of language model smoothing

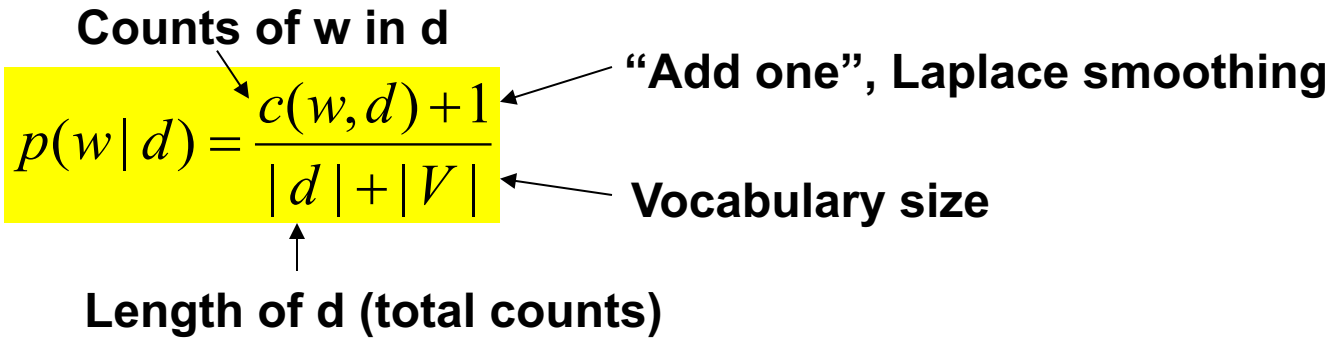


General idea of smoothing

- 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

Smoothing methods

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



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

Add one smoothing for bigrams

Original:

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Smoothed:

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

After smoothing

- Giving too much to the unseen events without discrimination

Original:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Smoothed:

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

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

- 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

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

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

MLE

parameter

  - Problems?
    - Hint: what is missing?

Smoothing methods

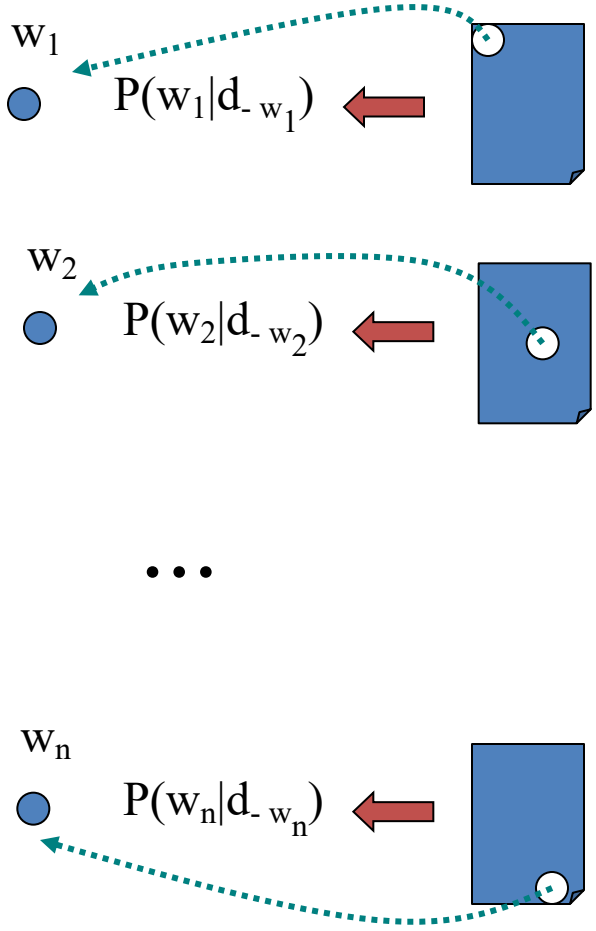
- Method 4: Dirichlet Prior/Bayesian
  - Assume pseudo counts μp(w|REF)

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

parameter

  - Problems?

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



Leave-one-out

log-likelihood

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

Maximum Likelihood Estimator

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

Understanding smoothing

Topical 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("algorithms"|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 $p(w|d) = 0.1p_{DML}(w|d) + 0.9p(w|REF)$, $p(q|d1) < p(q|d2)$!

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

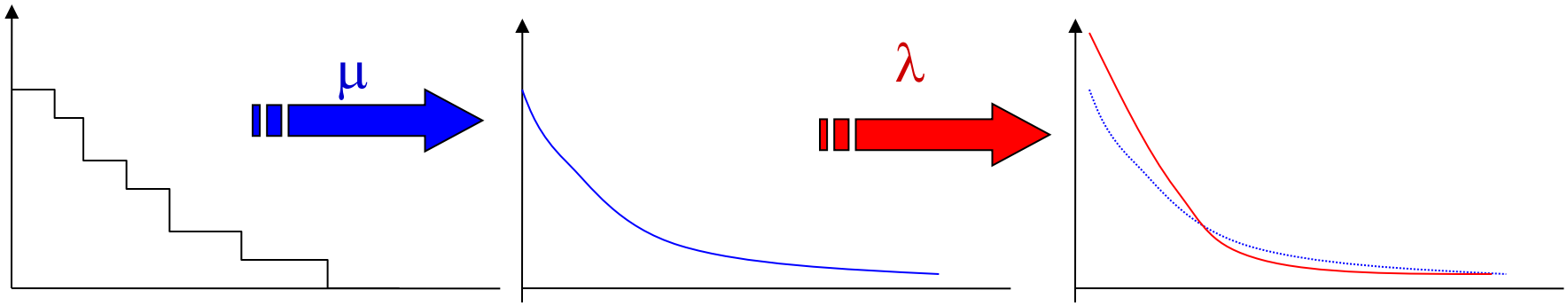
Two-stage smoothing [Zhai & Lafferty 02]

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

Smoothing & TF-IDF weighting

Smoothed ML estimate

Retrieval formula using the general smoothing scheme

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

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…

Understanding smoo

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

- 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)

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

IDF weighting

Ignore for ranking

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

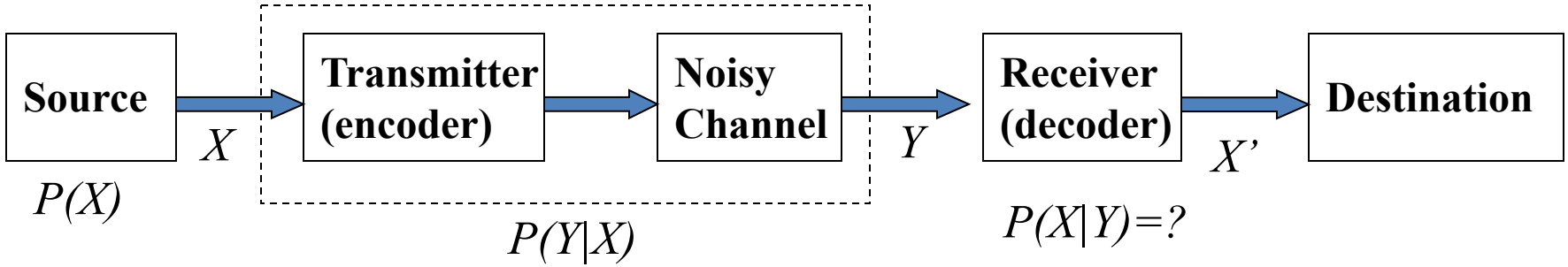
What you should know

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

Today's reading

- Introduction to information retrieval
  - Chapter 12: Language models for 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

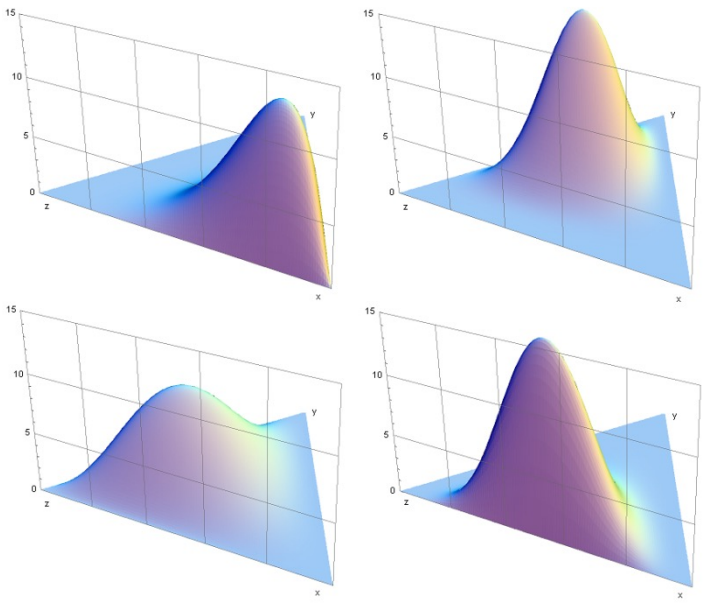
Many Examples:

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**

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

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

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

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

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:

$$p(w_i \mid \hat{\theta}) = \int p(w_i \mid \theta) Dir(\theta \mid \alpha) d\theta$$

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

Dirichlet prior smoothing