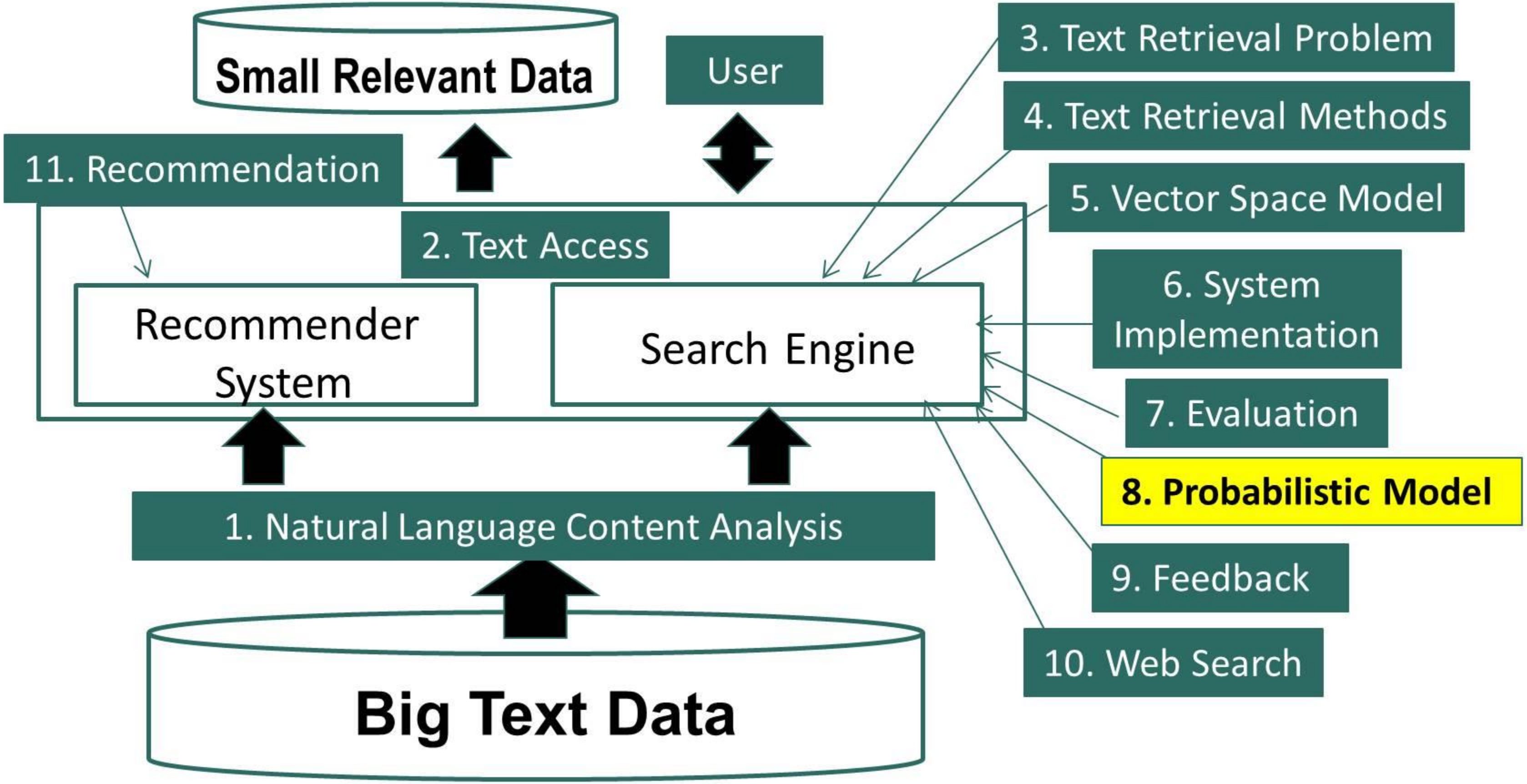
Text Retrieval and Search Engines

Probabilistic Retrieval Model: Basic Idea

ChengXiang "Cheng" Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Probabilistic Retrieval Model: Basic Idea



1. Probabilistic Retrieval Model - Basic Idea
Many Different Retrieval Models

- **Probabilistic models**: $f(d,q) = p(R=1|d,q)$, $R \in \{0,1\}$
  - Classic probabilistic model ➔ BM25
  - **Language model ➔Query Likelihood**
  - Divergence-from-randomness model ➔PL2

$$p(R=1|d,q) \approx p(q|d,R=1)$$

If a user likes document d, how likely would the user enter query q (in order to retrieve d)?

Probabilistic Retrieval Models: Basic Idea

Query	Doc	Rel
q	**d**	**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

f(q,d)=p(R=1|d,q)=? $\frac{count(q,d,R=1)}{count(q,d)}$

P(R=1|q1,d1) = ? 1/2

P(R=1|q1,d2) = ? 2/2

P(R=1|q1,d3) = ? 0/2

What about unseen documents?

Unseen queries?

Query Likelihood Retrieval Model

Query	Doc	Rel
q	**d**	**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

User likes d

$$f(q,d)=p(R=1|d,q) \approx \mathbf{p(q|d,R=1)}$$

How likely the user enters q

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

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

q = "**news about presidential campaign**"

p(q|d1)

p(q|d2)

p(q|d4)

p(q|d5)

d1 … **news about** …

d2 … **news about** organic food **campaign**…

d3 … **news** of **presidential campaign** …

d4 … **news** of **presidential campaign** …
… **presidential** candidate …

d5 … **news** of organic food **campaign**…
campaign…**campaign**…**campaign**…

Summary

- Relevance(q,d) = p(R=1|q,d) ➔p(q|d,R=1)
- **Query likelihood** ranking function: f(q,d)=p(q|d)
  - Probability that a user who likes d would pose query q
- How to compute p(q|d)? How to compute probability of text in general? ➔ Language Model

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

2. Statistical Language Model Overview

- What is a Language Model?
- Unigram Language Model
- Uses of a Language Model

What is a Statistical Language Model (LM)?

- A probability distribution over word sequences
  - p("*Today is Wednesday*") ≈ 0.001
  - p("*Today Wednesday is*") ≈ 0.0000000000001
  - p("*The eigenvalue is positive*") ≈ 0.00001
- Context-dependent!
- Can also be regarded as a probabilistic mechanism for "generatin[g]" text ... also called a "generative" model

Why is a LM Useful?

- Quantify the uncertainties in 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)

The Simplest Language Model: Unigram LM

- Generate text by generating each word INDEPENDENTLY
- Thus, $p(w_1\ w_2\ \ldots\ w_n)=p(w_1)p(w_2)\ldots p(w_n)$
- Parameters: $\{p(w_i)\}$ $p(w_1)+\ldots+p(w_N)=1$ (N is voc. size)
- Text = sample drawn according to this **word distribution**

Wednesday

today

…

eigenvalue

p("today is Wed")

= p("today")p("is")p("Wed")

= 0.0002 × 0.001 × 0.000015

Text Generation with Unigram LM

Unigram LM p(w|θ) **Sampling** **Document =?**

Topic 1:
Text mining

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

Text mining paper

Topic 2:
Health

…
food 0.25
nutrition 0.1
healthy 0.05
diet 0.02
…

Food nutrition paper

Estimation of Unigram LM

Unigram LM p(w|θ)=? ← **Estimation** — **Text Mining Paper d**

Total #words=**100**

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

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

Maximum Likelihood (ML) Estimator:

$$p(w \mid \theta) = p(w \mid d) = \frac{c(w,d)}{|d|}$$

Is this the best estimate?

LMs for Topic Representation

General Background English Text

B

the 0.03
a 0.02
is 0.015
we 0.01
...
food 0.003
computer 0.00001
text 0.000006
…

Background LM: p(w|B)

Computer Science Papers

C

the 0.032
a 0.019
is 0.014
we 0.011
...
computer 0.004
software 0.0001
text 0.00006
…

Collection LM: p(w|C)

Text mining paper

d

the 0.031
…
text 0.04
mining 0.035
association 0.03
clustering 0.005
computer 0.0009
…
food 0.000001

Document LM: p(w|

LMs for Association Analysis

What words are semantically related to "computer"?

Documents containing word "computer"

Topic LM: p(w|"computer")

the 0.032
a 0.019
is 0.014
we 0.008
computer 0.004
software 0.0001

B

General Background English Text

Background LM: p(w|B)

the 0.03
a 0.02
is 0.015
we 0.01
...
computer 0.00001

Normalized Topic LM:
p(w|"computer")/p(w|B)

computer 400
software 150
program 104
…
text 3.0
…
the 1.1
a 0.99
is 0.9
we 0.8

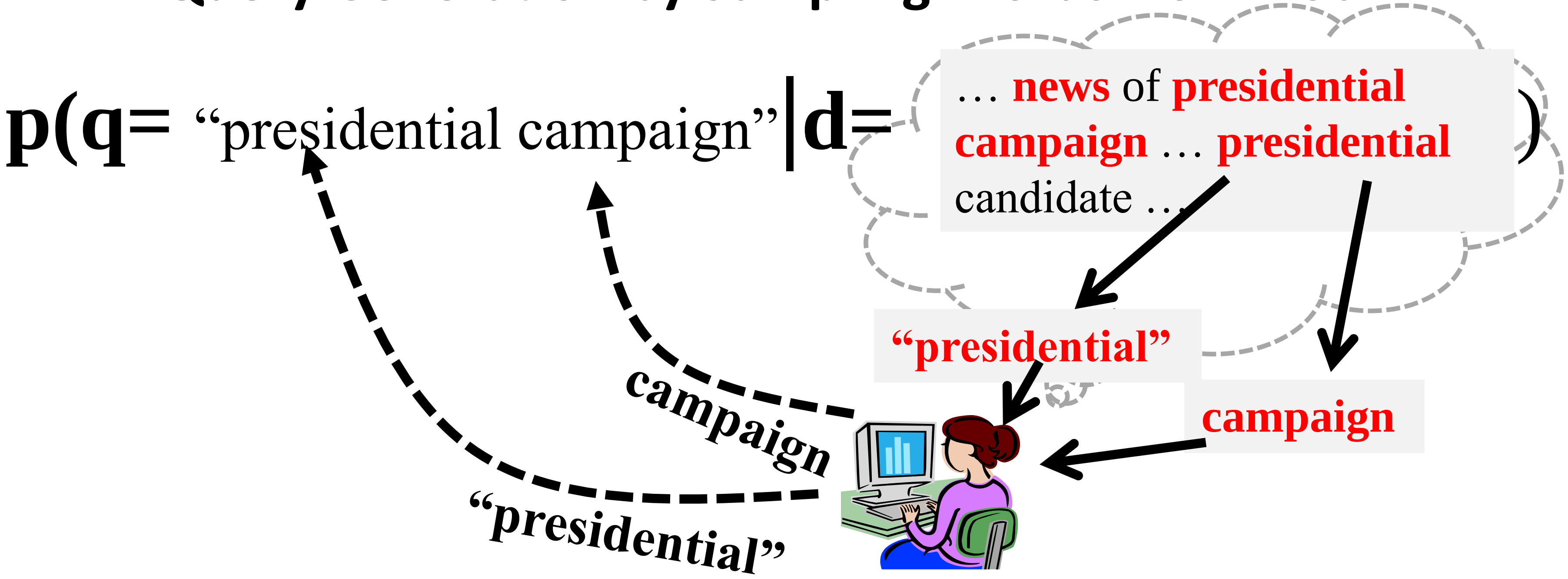
Summary

- Language Model = probability distribution over text
- Unigram Language Model = word distribution
- Uses of a Language Model
  - Representing topics
  - Discovering word associations

Additional Readings

- Chris Manning and Hinrich Schütze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA: May 1999.
- Rosenfeld, R., "Two decades of statistical language modeling: where do we go from here?," *Proceedings of the IEEE* , vol.88, no.8, pp.1270,1278, Aug. 2000

3. Query Likelihood Retrieval Function
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

$$p(q=\text{“presidential campaign”} \mid d= \text{… news of presidential campaign … presidential candidate …})$$

… **news** of **presidential campaign** … **presidential** candidate …

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

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

Assumption:
Each query word is generated independently

Does Query Likelihood Make Sense?

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

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** … **presidential** 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!**

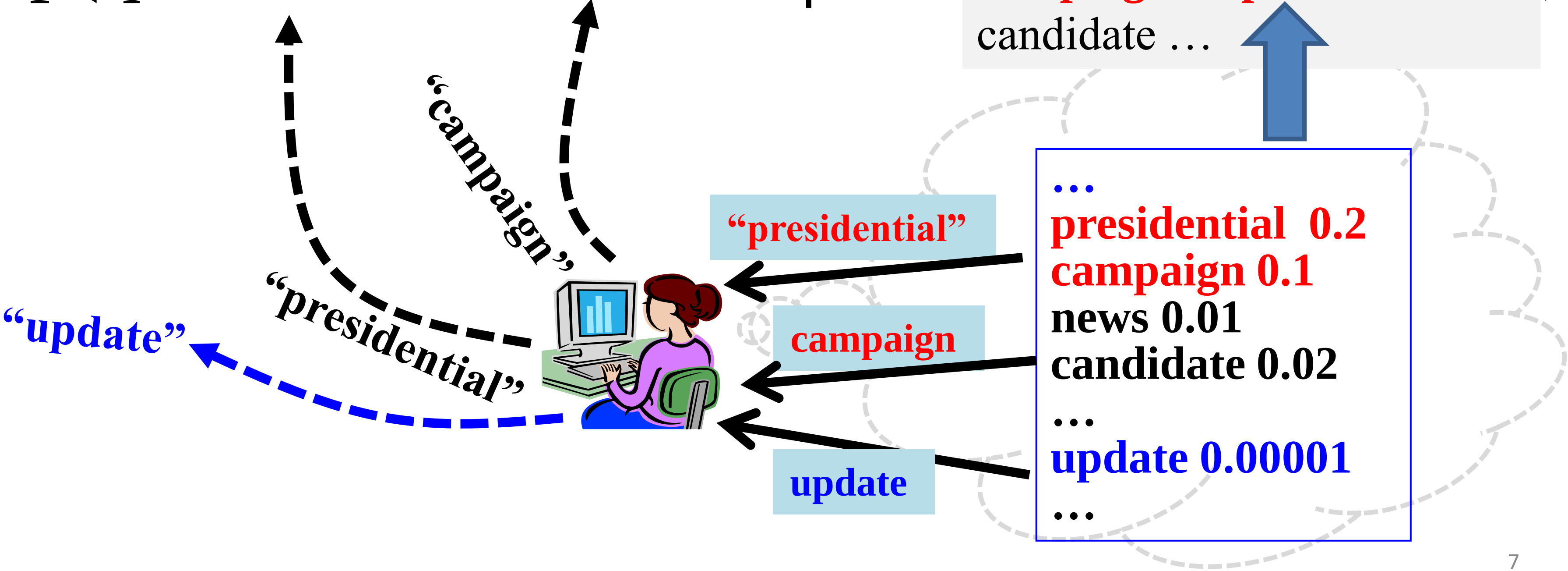
p(q|d2= … news about organic food **campaign**… **)** $=\frac{0}{|d2|} * \frac{1}{|d2|} * \frac{0}{|d2|}$ **=0**

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

d1

Text mining paper

d2

Food nutrition paper

Document LM

p(w|d1)

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

p(w|d2)

…
food 0.25
nutrition 0.1
healthy 0.05
diet 0.02
…

Query q =

"data mining algorithms"

$$p(\text{"data mining alg"}|d1) = p(\text{"data"}|d1) \times p(\text{"mining"}|d1) \times p(\text{"alg"}|d1)$$

$$p(\text{"data mining alg"}|d2) = p(\text{"data"}|d2) \times p(\text{"mining"}|d2) \times p(\text{"alg"}|d2)$$

Summary: Ranking based on Query Likelihood

$$q = w_1 w_2 \ldots w_n \qquad\qquad p(q \mid d) = p(w_1 \mid d) \times \ldots \times p(w_n \mid d)$$

$$f(q,d) = \log p(q \mid d) = \sum_{i=1}^{n} \log p(w_i \mid d) = \sum_{w \in V} c(w,q) \log \boxed{p(w \mid d)}$$

Document language model

Retrieval problem ➔ Estimation of $p(w_i|d)$

Different estimation methods ➔ different ranking functions

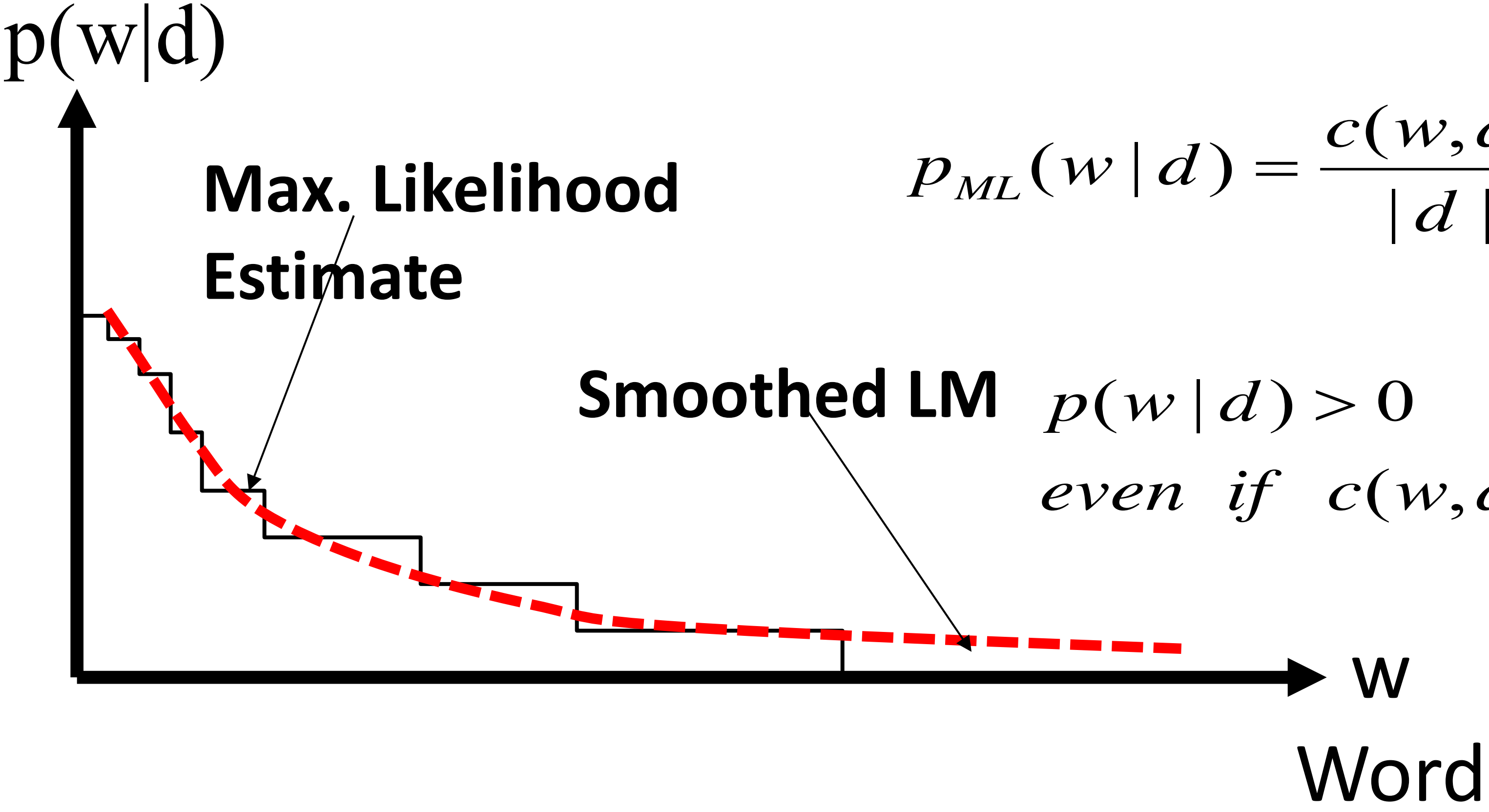
4. Statistical Language Model (1)
Ranking Function based on Query Likelihood

$$q = w_1 w_2 \ldots w_n \qquad p(q \mid d) = p(w_1 \mid d) \times \ldots \times p(w_n \mid d)$$

$$f(q,d) = \log p(q \mid d) = \sum_{i=1}^{n} \log p(w_i \mid d) = \sum_{w \in V} c(w,q) \log \boxed{p(w \mid d)}$$

How should we estimate *p(w|d)*?

How to Estimate p(w|d)



$$p_{ML}(w \mid d) = \frac{c(w,d)}{|d|}$$

$$p(w \mid d) > 0$$
$$even \;\; if \;\; c(w,d) = 0$$

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

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

Collection language model

Rewriting the Ranking Function with Smoothing

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

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

Query words **matched** in d

Query words **not matched** in d

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

All query words

Query words **matched** in d

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

Benefit of Rewriting

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

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

- Enable efficient computation

5. Statistical Language Model (2)
Benefit of Rewriting

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

Doc length normalization

TF weighting

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

matched query terms

IDF weighting

Ignore for ranking

- Enable efficient computation

Summary

- Smoothing of p(w|d) is necessary for query likelihood
- General idea: smoothing with p(w|C)
  - The probability of an unseen word in d is assumed to be proportional to p(w|C)
  - Leads to a general ranking formula for query likelihood with TF-IDF weighting and document length normalization
  - Scoring is primarily based on sum of weights on matched query terms
- However, how exactly should we smooth?

6. Smoothing Methods (1)
Query Likelihood + Smoothing with p(w|C)

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

$$f(q,d) = \sum_{\substack{w_i \in d \\ w_i \in q}} c(w,q)[\log \frac{p_{Seen}(w_i \mid d)}{\alpha_d\, p(w_i \mid C)}] + n \log \alpha_d$$

$$\boxed{\begin{aligned} &p_{Seen}(w_i \mid d) = ? \\ &\alpha_d = ? \end{aligned}}$$

How to smooth p(w|d)?

Linear Interpolation (Jelinek-Mercer) Smoothing

Unigram LM p(w|θ)=?

	…
10/100 →	**text** ?
5/100 →	mining ?
3/100 →	association ?
3/100 →	database ?
	…
1/100 →	query ?
0/100 →	**network?**

Document d

Total #words=**100**

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

Collection LM

P(w|C)

the 0.1
a 0.08
..
computer 0.02
database 0.01
……
text 0.001
network 0.001
mining 0.0009
…

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

$$\lambda \in [0,1]$$

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

$$p(\text{"network"}|d) = \lambda * 0.001$$

Dirichlet Prior (Bayesian) Smoothing

Unigram LM p(w|θ)=?

10/100	… **text** ?
5/100	mining ?
3/100	association ?
3/100	database ?
1/100	…
0/100	query ?
	network?

Document d
Total #words=**100**

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

Collection LM
P(w|C)

the 0.1
a 0.08
..
computer 0.02
database 0.01
……
text 0.001
network 0.001
mining 0.0009
…

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

$$\mu \in [0, +\infty)$$

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

7. Smoothing Methods (2)
Ranking Function for JM Smoothing

$$f(q,d) = \sum_{\substack{w_i \in d \\ w_i \in q}} c(w,q)[\log \frac{p_{Seen}(w_i \mid d)}{\alpha_d\, p(w_i \mid C)}] + n \log \alpha_d$$

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

$\lambda \in [0,1]$

$$\frac{p_{seen}(w \mid d)}{\alpha_d p(w \mid C)} = \frac{(1-\lambda)p_{ML}(w \mid d) + \lambda p(w \mid C)}{\lambda p(w \mid C)} = 1 + \frac{1-\lambda}{\lambda}\frac{c(w,d)}{|d|\, p(w \mid C)}$$

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

Ranking Function for Dirichlet Prior Smoothing

$$f(q,d) = \sum_{\substack{w_i \in d \\ w_i \in q}} c(w,q)[\log \frac{p_{Seen}(w_i \mid d)}{\alpha_d \, p(w_i \mid C)}] + n \log \alpha_d$$

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

$\mu \in [0, +\infty)$

$$\frac{p_{seen}(w \mid d)}{\alpha_d p(w \mid C)} = \frac{\frac{c(w,d) + \mu p(w \mid C)}{\mid d \mid + \mu}}{\frac{\mu p(w \mid C)}{\mid d \mid + \mu}} = 1 + \frac{c(w,d)}{\mu p(w \mid C)} \qquad \alpha_d = \frac{\mu}{\mid d \mid + \mu}$$

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

Summary

- Two smoothing methods
  - Jelinek-Mercer: Fixed coefficient linear interpolation
  - Dirichlet Prior: Adding pseudo counts; adaptive interpolation
- Both lead to state of the art retrieval functions with assumptions clearly articulated (less heuristic)
  - Also implementing TF-IDF weighting and doc length normalization
  - Has precisely one (smoothing) parameter

Summary of Query Likelihood Probabilistic Model

- Effective ranking functions obtained using pure probabilistic modeling
  - Assumption 1: Relevance(q,d) = p(R=1|q,d) ≈ p(q|d,R=1) ≈**p(q|d)**
  - Assumption 2: Query words are generated independently
  - Assumption 3: Smoothing with p(w|C)
  - Assumption 4: JM **or** Dirichlet prior smoothing
- Less heuristic compared with VSM
- Many extensions have been made [Zhai 08]

Additional Readings

- ChengXiang Zhai, *Statistical Language Models for Information Retrieval* (Synthesis Lectures Series on Human Language Technologies), Morgan & Claypool Publishers, 2008.

http://www.morganclaypool.com/doi/abs/10.2200/S00158ED1V01Y200811HLT001