Probabilistic Retrieval Models

Quach Dinh Hoang

Slides are obtained from [Zhai and Massung, 2016]

Probabilistic Retrieval Models

- Framework

$$f(d,q) = p(R = 1 | d,q), R \in \{0,1\}$$

  – Classic probabilistic model: BM25
  – **Language model: Query Likelihood**
  – Divergence-from-randomness model: PL2
- Language model

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

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

Query Likelihood Retrieval Model

User likes d

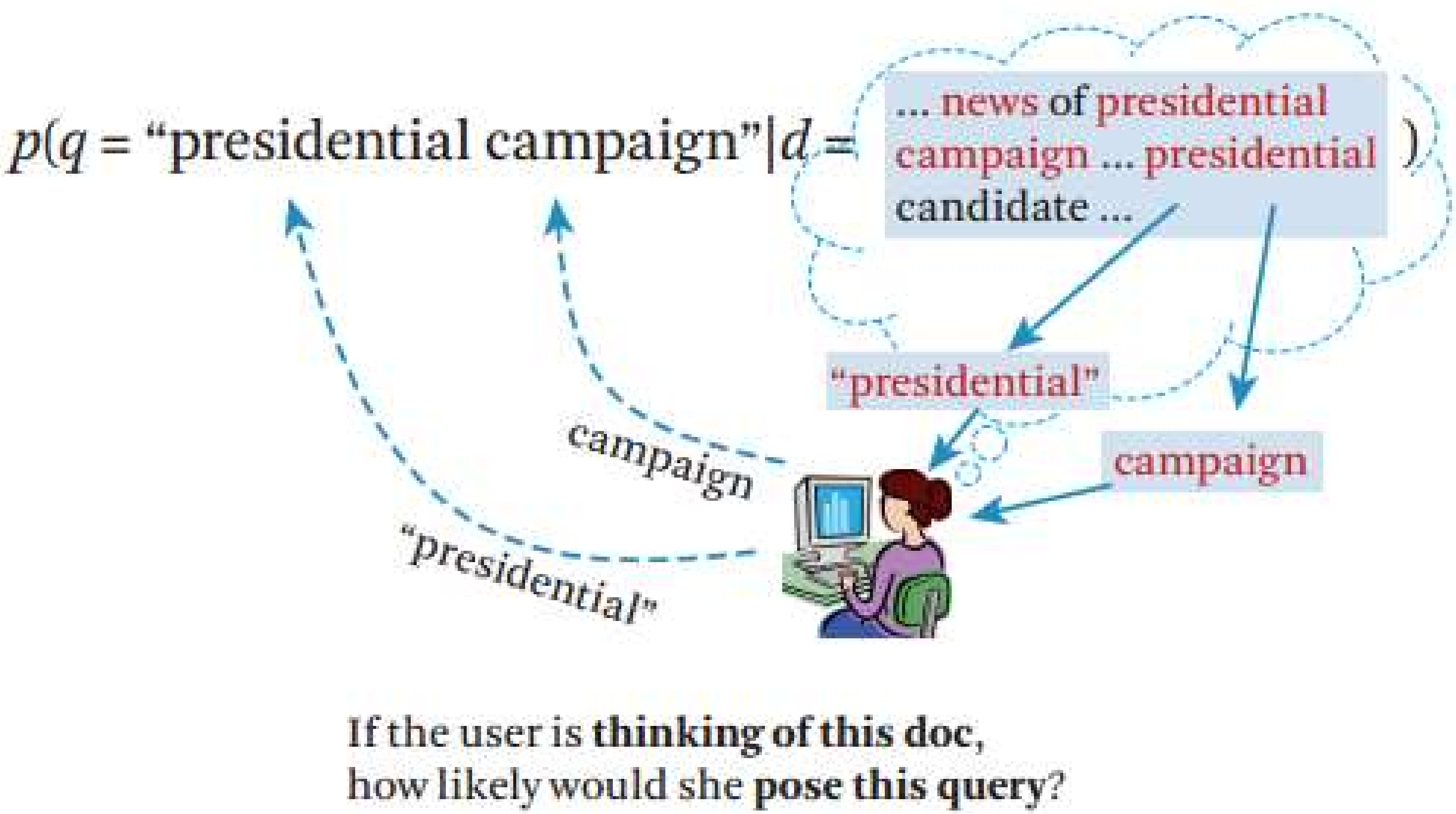
$$f(q,d) = p(R = 1| d,q) \sim \mathbf{p(q \mid d, R = 1)}$$

How likely the user enters q

Assumption:

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

Query Likelihood Retrieval Model



Language Model

- It is a probability distribution over word sequences
- A document is considered as a sample have drawn from an underline language model
- Each document is ranked based on the capability of its language model generate the words in query

The Simplest Language Model: Unigram LM

- Generate text by generating each word INDEPENDENTLY

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

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

- Parameters: $\{p(w_i)\}$

$$p(w_1) + \ldots + p(w_N) = 1$$

  – N is vocabulary size
- Document = sample drawn according to this **word distribution**

Unigram Query Likelihood Model Example

Assumption: each query word is independent

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

$$p(q \mid d_4 = \text{... news of presidential campaign ... presidential candidate ...}) = \frac{2}{|d_4|} * \frac{1}{|d_4|}$$

$$p(q \mid d_3 = \text{... news of presidential campaign ...}) = \frac{1}{|d_3|} * \frac{1}{|d_3|}$$

$$p(q \mid d_2 = \text{... news about organic food campaign ...}) = \frac{0}{|d_2|} * \frac{1}{|d_2|} = 0$$

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

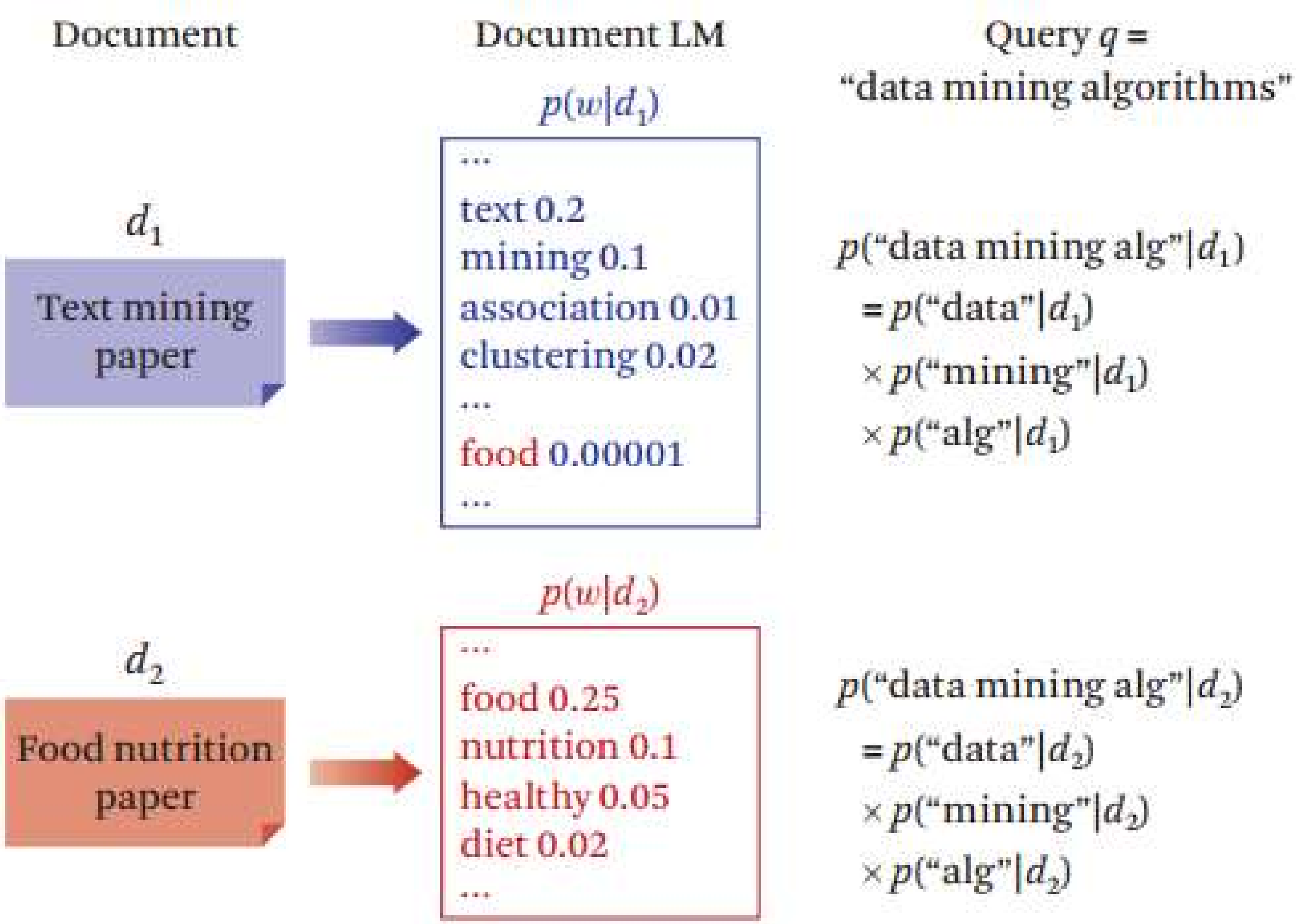
- All of documents have zero probability of generating this query.

Clearly, this's not desirable. How to fix it?

Document Language Model

- Assume that the user could have drawn a query from a document language model

Document Language Model Example



Summary of Language Model

- Generate text by generating each word INDEPENDENTLY

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

- Parameters: $\{p(w_i)\}$

$$p(w_1) + \ldots + p(w_N) = 1$$

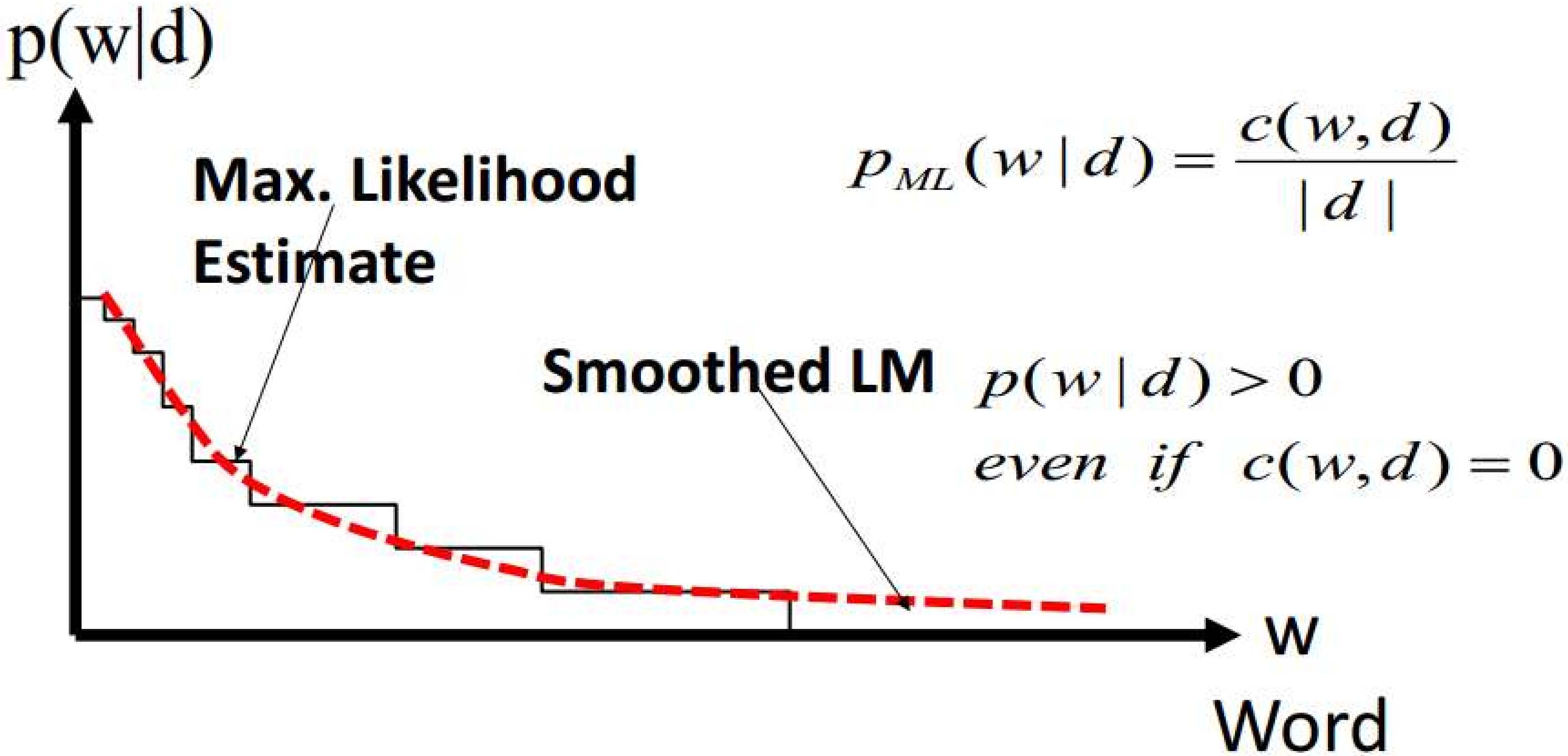
  – N is vocabulary size

- In practice

Document language model

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

How to Estimate p(w|d)



Smoothing a Language Model

- 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 language model (LM)
- One possibility: Reference LM = Collection LM

Document language model

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

Collection language model

Rewriting the Ranking Function with Smoothing

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

$$= \underbrace{\sum_{w\in V, c(w,d)>0} c(w,q)\log p_{\text{Seen}}(w|d)}_{\text{Query words } \textbf{matched} \text{ in } d} + \boxed{\underbrace{\sum_{w\in V, c(w,d)=0} c(w,q)\log \alpha_d\, p(w|C)}_{\text{Query words } \textbf{not matched} \text{ in } d}}$$

$$\underbrace{\sum_{w\in V} c(w,q)\log \alpha_d\, p(w|C)}_{\textbf{All} \text{ query words}} - \underbrace{\sum_{w\in V, c(w,d)>0} c(w,q)\log \alpha_d\, p(w|C)}_{\text{Query words } \textbf{matched} \text{ in } d}$$

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

Rewriting the Ranking Function with Smoothing

$$\log p(q\,|\,d) = \underbrace{\sum_{\substack{w\in d\\ w\in q}}}_{\text{Matched query terms}} c(w,q)\left[\log \frac{\overbrace{p_{\text{Seen}}(w\,|\,d)}^{\text{TF weighting}}}{\underbrace{\alpha_d\, p(w\,|\,C)}_{\text{IDF weighting}}}\right] + \overbrace{n \log \alpha_d}^{\text{Doc length normalization}} + \underbrace{\boxed{\sum_{i=1}^{N} \log p(w_i\,|\,C)}}_{\text{Ignore for ranking}}$$

Summary of Smoothing

- 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 TFIDF weighting and document length normalization
  - Scoring is primarily based on sum of weights on matched query terms

Smoothing Methods

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

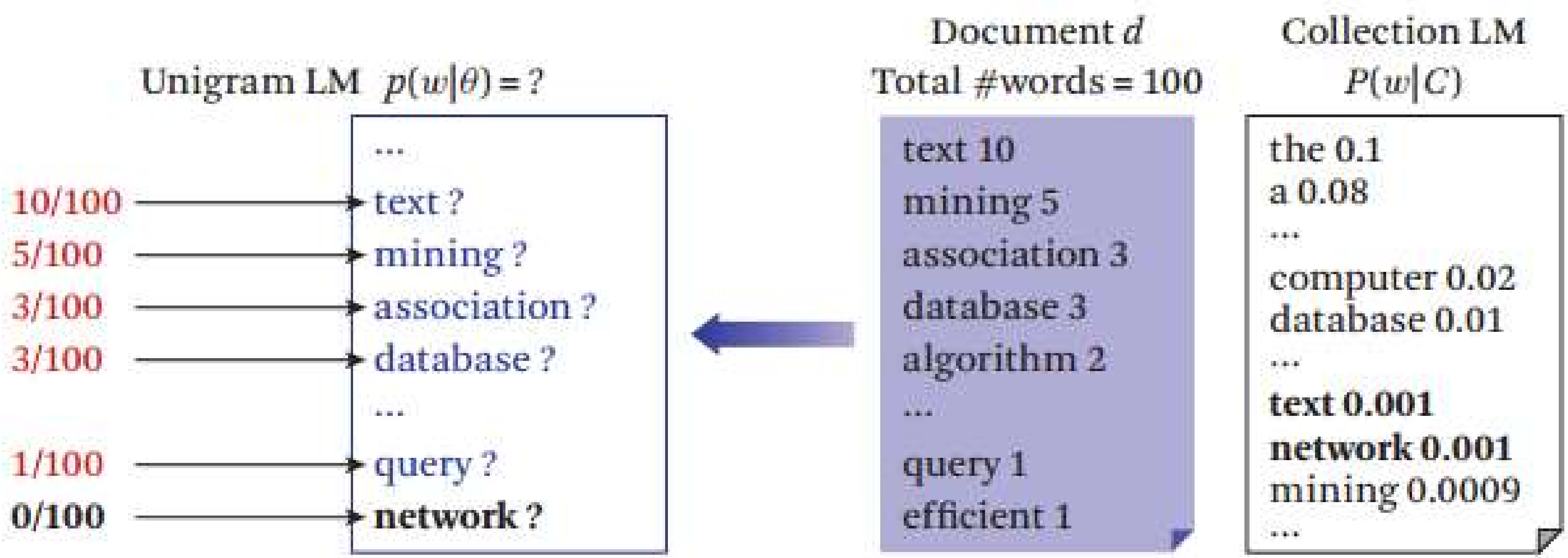
$$f(d,q) = \sum_{\substack{w \in d \\ w \in q}} c(w,q)[\log \frac{p_{\text{Seen}}(w|d)}{\alpha_d\, p(w|C)}] + n \log \alpha_d$$

$$\begin{cases} p_{seen}(w_i \mid d) = ? \\ \alpha_d = ? \end{cases}$$

How to smooth p(w|d)?

Jelinek-Mercer Smoothing

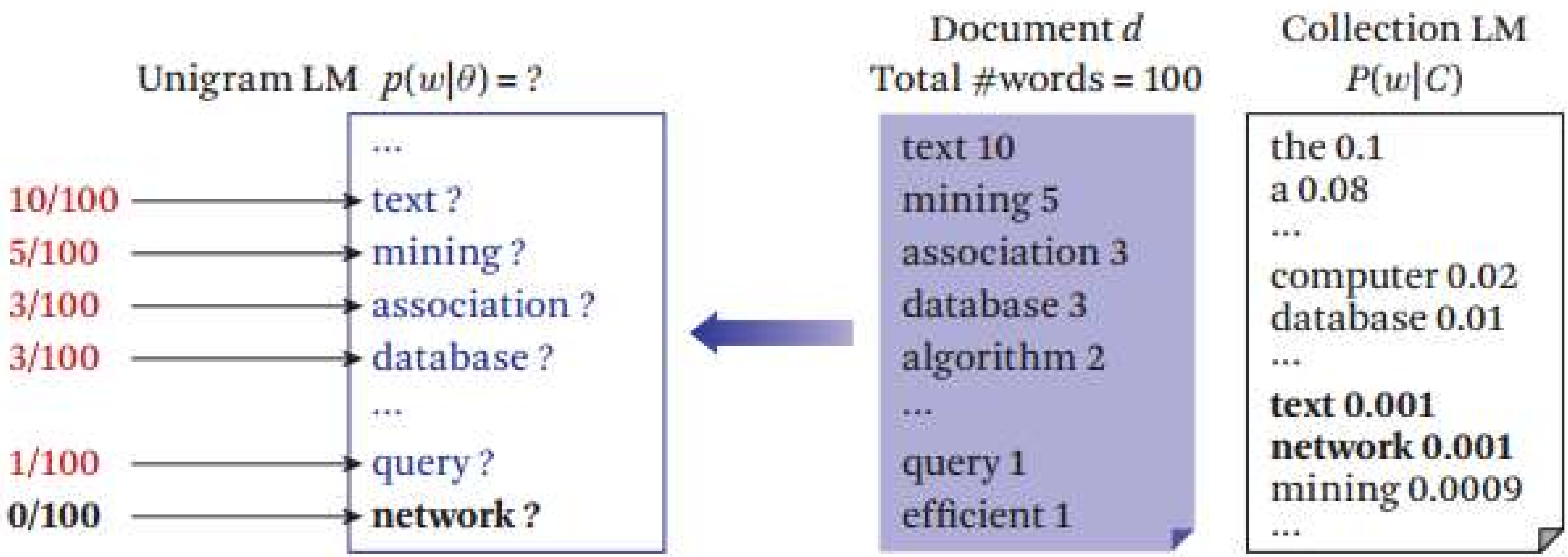
$$p(w|d) = (1-\lambda)\frac{c(w,d)}{|d|} + \lambda\, p(w|C) \qquad \lambda \in [0,1]$$



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

Dirichlet Prior Smoothing

$$p(w|d) = \frac{c(w,d) + \mu p(w|C)}{|d| + \mu} = \frac{|d|}{|d| + \mu}\frac{c(w,d)}{|d|} + \frac{\mu}{|d| + \mu}p(w|C) \qquad \mu \in [0, +\infty)$$



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

Ranking Function for Jelinek-Mercer Smoothing

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

$$p(w|d) = (1-\lambda)\frac{c(w, d)}{|d|} + \lambda p(w|C) \qquad \lambda \in [0, 1]$$

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

$$\text{score}_{JM}(q, d) = \sum_{w \in q, d} c(w, q) \log \left(1 + \frac{1-\lambda}{\lambda} \cdot \frac{c(w, d)}{|d| \cdot p(w \mid C)}\right)$$

Ranking Function for Dirichlet Prior Smoothing

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

$$p(w|d) = \frac{c(w, d) + \mu p(w|C)}{|d| + \mu} = \frac{|d|}{|d| + \mu} \frac{c(w, d)}{|d|} + \frac{\mu}{|d| + \mu} p(w|C) \qquad \mu \in [0, +\infty)$$

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

$$\text{score}_{DIR}(q, d) = \sum_{w \in q, d} c(w, q) \log \left(1 + \frac{c(w, d)}{\mu \cdot p(w \mid C)}\right) + |q| \log \frac{\mu}{\mu + |d|}$$

Summary of Smoothing Methods

- 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 Language 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