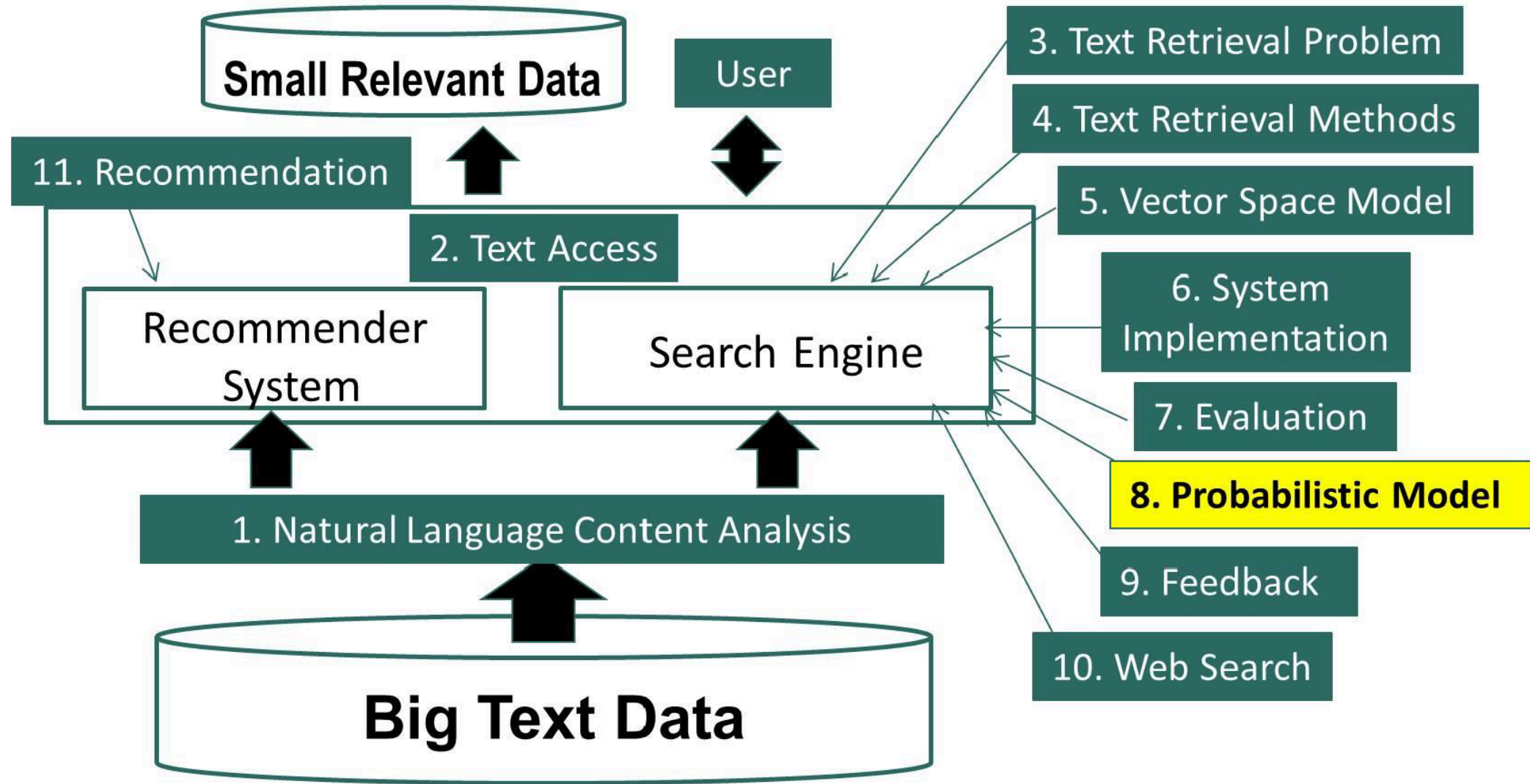


Information Retrieval & Text Mining

Probabilistic Retrieval Model: Smoothing

Dr. Iqra Safder

Probabilistic Retrieval Model: Smoothing



Ranking Function based on Query Likelihood

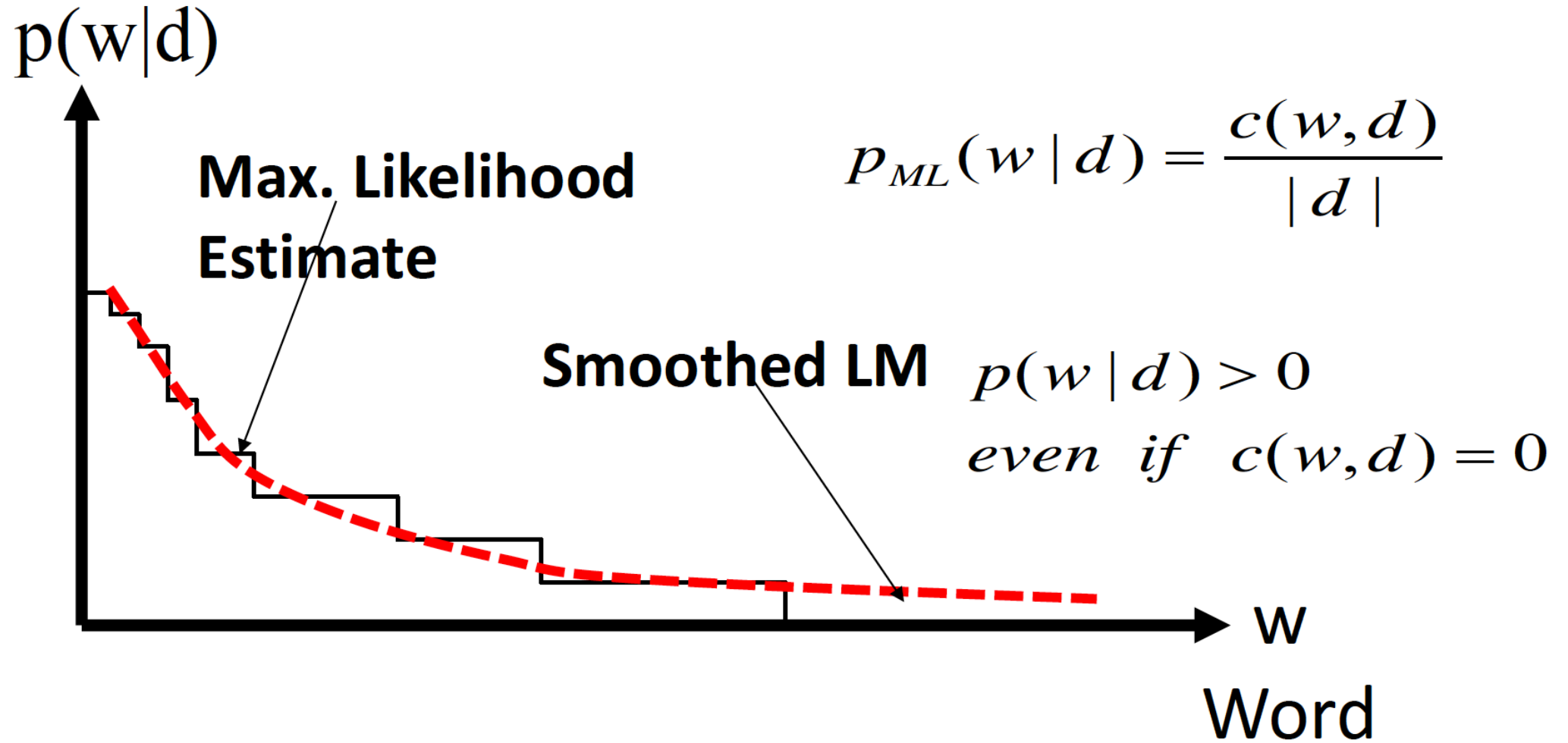
$$q = w_1 w_2 \dots w_n \quad p(q | d) = p(w_1 | d) \times \dots \times p(w_n | d)$$

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

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

In order to assign a non-zero probability to words that have not been observed in the document, we would have to take away some probability mass from seen words because we need some extra probability mass for the unseen words—otherwise, they won't sum to one.

How to Estimate $p(w | d)$



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

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

Discounted ML estimate

if w is seen in d

otherwise

Collection language model

Regardless of whether the word w is seen in the document or not, all these probabilities must sum to one, so α is constrained.

Rewriting the Ranking Function with Smoothing

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

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

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

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

This is actually quite useful, since part of the sum over all $w \in V$ can now be written as $|q| \log \alpha_d$. Additionally, the sum of query words matched in d is in terms of words that we observe in the query. Just like in the vector space model, we are now able to take a sum of terms in the intersection of the query vector and the document vector.

Benefit of Rewriting

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

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

- Enable efficient computation

Benefit of Rewriting

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

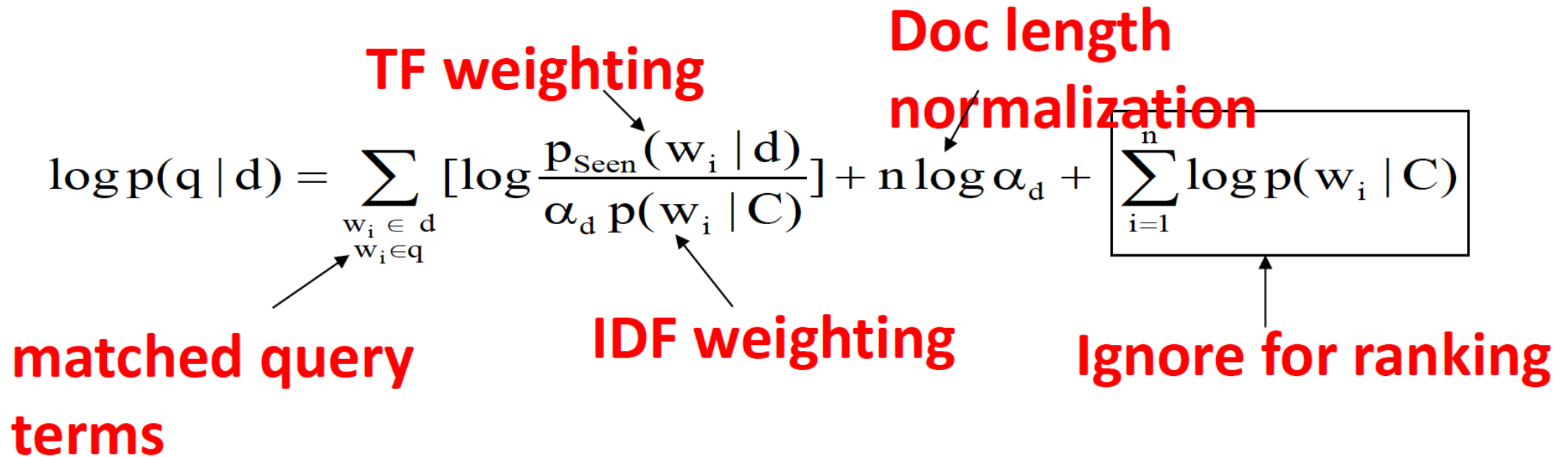
TF weighting

Doc length normalization

IDF weighting

Ignore for ranking

matched query terms

$$\log p(q | d) = \sum_{\substack{w_i \in d \\ w_i \in q}} \left[\log \frac{p_{\text{Seen}}(w_i | d)}{\alpha_d p(w_i | C)} \right] + n \log \alpha_d + \sum_{i=1}^n \log p(w_i | C)$$


- Enable efficient computation

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

Matched query terms TF weighting Doc length normalization
IDF weighting Ignore for ranking

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?