

# Language Models

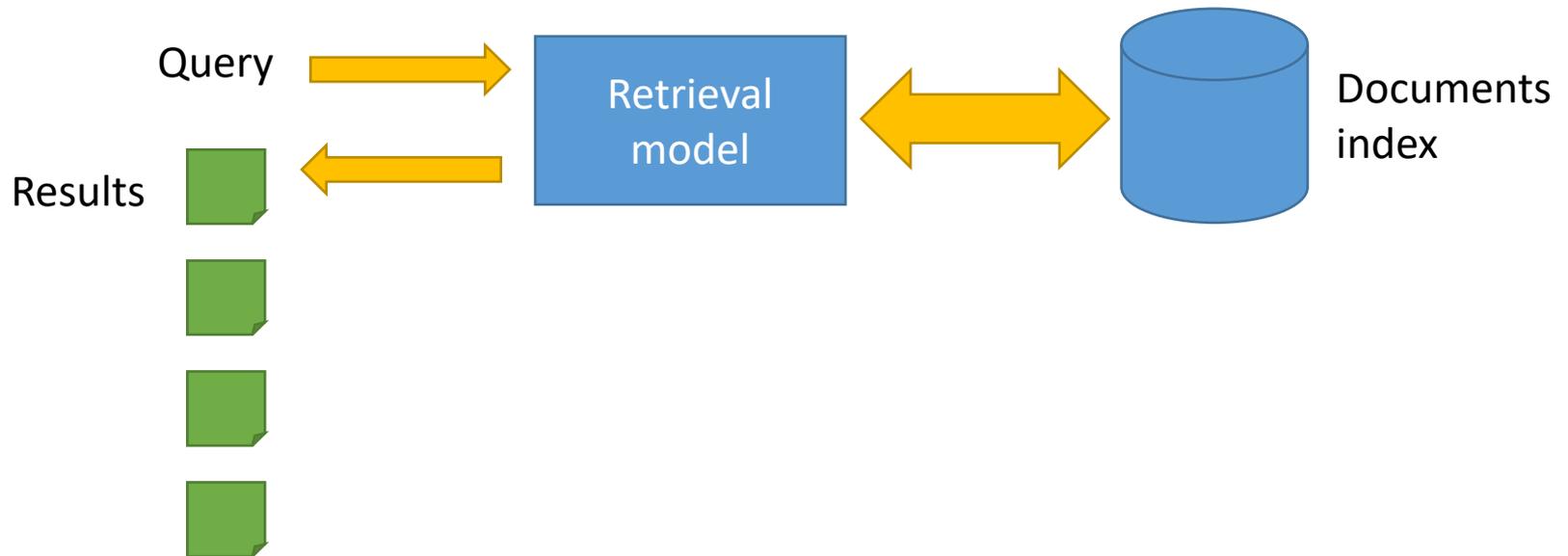
LM Jelinek-Mercer Smoothing and LM Dirichlet Smoothing

## Information Retrieval

Slides based on the books:



# Problem setting



# Retrieval models

- Geometric/linear spaces
  - Vector space model
- Probability ranking principle
- Language models approach to IR
  - An important emphasis in recent work
- Probabilistic retrieval model
  - Binary independence model
  - Okapi's BM25

# Recap of basic probability

Information Retrieval

# Recall a few probability basics

- For events A and B, the Bayes' Rule is:

$$p(A, B) = p(A|B)p(B) = p(B|A)p(A) \quad (\text{chain rule})$$

$$p(A|B) = \frac{p(A, B)}{p(B)} = \frac{p(A)p(B|A)}{p(B)} \quad (\text{Bayes' rule})$$

- Interpretation:

$$posterior = \frac{prior \cdot likelihood}{evidence} \Leftrightarrow p(A|B) = \frac{p(A)p(B|A)}{p(B)}$$

# Recall a few probability basics

- Independence assumption:

$$p(A|B) = \frac{p(A)p(B|A)}{p(B)} = \frac{p(A) \prod_i p(b_i|A)}{\prod_i p(b_i)}$$

- Odds:  $O(A) = \frac{p(A)}{p(\bar{A})} = \frac{p(A)}{1 - p(A)}$

$$O(A|B) = \frac{p(A|B)}{p(\bar{A}|B)} = \frac{\frac{p(A)p(B|A)}{p(B)}}{\frac{p(\bar{A})p(B|\bar{A})}{p(B)}} = \frac{p(A)p(B|A)}{p(\bar{A})p(B|\bar{A})}$$

# Recall a few probability basics

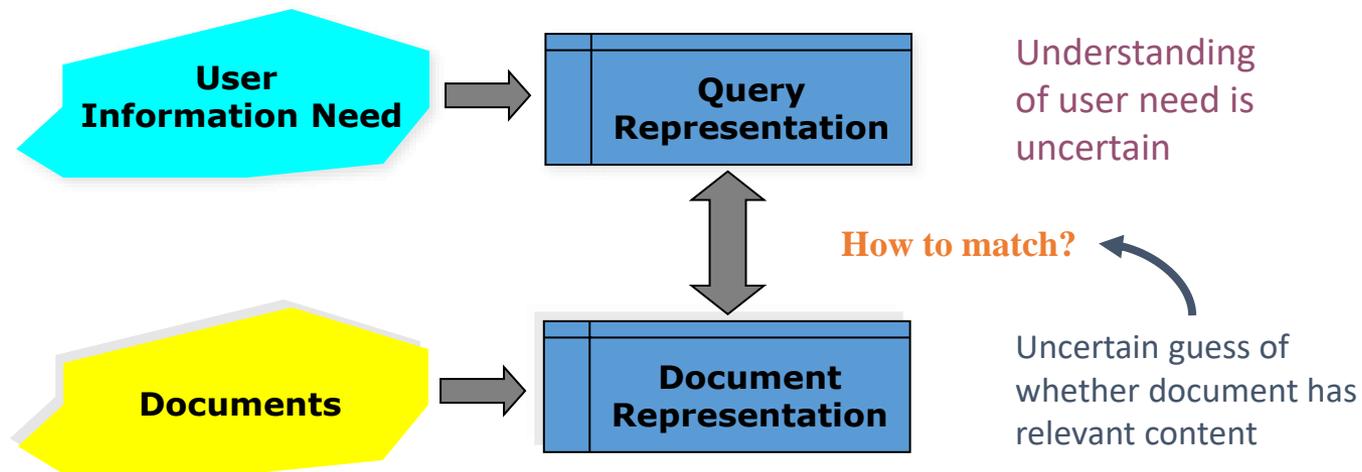
$$p(A|data) = \frac{p(A)p(data|A)}{p(data)}$$

$$p(SLB = campeão|data) = \frac{p(SLB = campeão)p(data|SLB = campeão)}{p(data)}$$

$$aposteriori = \frac{apriori \cdot verosimilhança}{evidencia}$$

# Why probabilities in IR?

- In traditional IR systems, matching between each document and query is attempted in a **semantically imprecise space** of index terms.



Probabilities provide a principled foundation for uncertain reasoning.

# The document ranking problem

- We have a collection of documents
- User issues a query
- A list of documents needs to be returned
- Ranking method is the core of an IR system:
  - In what order do we present documents to the user?
  - We want the “best” document to be first, second best second, etc....

**Idea: Rank by probability of  
relevance of the document w.r.t. information need**

# Modeling relevance

**P(R=1 | document, query)**

- Let **d** represent a document in the collection.
- Let **R** represent relevance of a document w.r.t. to a query **q**
- Let **R=1** represent relevant and **R=0** not relevant.

• Our goal is to estimate: 
$$p(r = 1|q, d) = \frac{p(d, q|r = 1)p(r = 1)}{p(d, q)}$$

$$p(r = 0|q, d) = \frac{p(d, q|r = 0)p(r = 0)}{p(d, q)}$$

# Probability Ranking Principle (PRP)

- PRP in action: Rank all documents by  $p(r = 1|q, d)$ 
  - Theorem: Using the PRP is optimal, in that it minimizes the loss (Bayes risk) under 1/0 loss
  - Provable if all probabilities correct, etc. [e.g., Ripley 1996]

$$p(r|q, d) = \frac{p(d, q|r)p(r)}{p(d, q)}$$

- Using odds, we reach a more convenient formulation of ranking :

$$O(R|q, d) = \frac{p(r = 1|q, d)}{p(r = 0|q, d)}$$

# Probabilistic retrieval models interpretation

- PRP in action: Rank all documents by  $p(r = 1|q, d)$ 
  - Theorem: Using the PRP is optimal, in that it minimizes the loss (Bayes risk) under 1/0 loss
  - Provable if all probabilities correct, etc. [e.g., Ripley 1996]

$$p(r|q, d) = \frac{p(d, q|r)p(r)}{p(d, q)}$$

- Using odds, we reach a more convenient ranking formulation:

$$O(R|q, d) = \frac{p(r = 1|q, d)}{p(r = 0|q, d)} \propto \frac{p(d|q, r = 1)}{p(d|q, r = 0)}$$

# Language models interpretation

- In language models, we do a different formulation towards the query posterior given the document as a model.

$$O(R|q, d) = \frac{p(r = 1|q, d)}{p(r = 0|q, d)} \propto \log \frac{p(q|d, r)p(r|d)}{p(q|d, \bar{r})p(\bar{r}|d)}$$

# The two families of Retrieval Models

## Probability Ranking Principle

$$O(R|q, d) = \frac{p(r = 1|q, d)}{p(r = 0|q, d)}$$

## Probabilistic Retrieval Models

$$O(R|q, d) \propto \frac{p(d|q, r = 1)}{p(d|q, r = 0)}$$

- Vector Space Model
- Binary Independent Model
- BM25

## Language Models

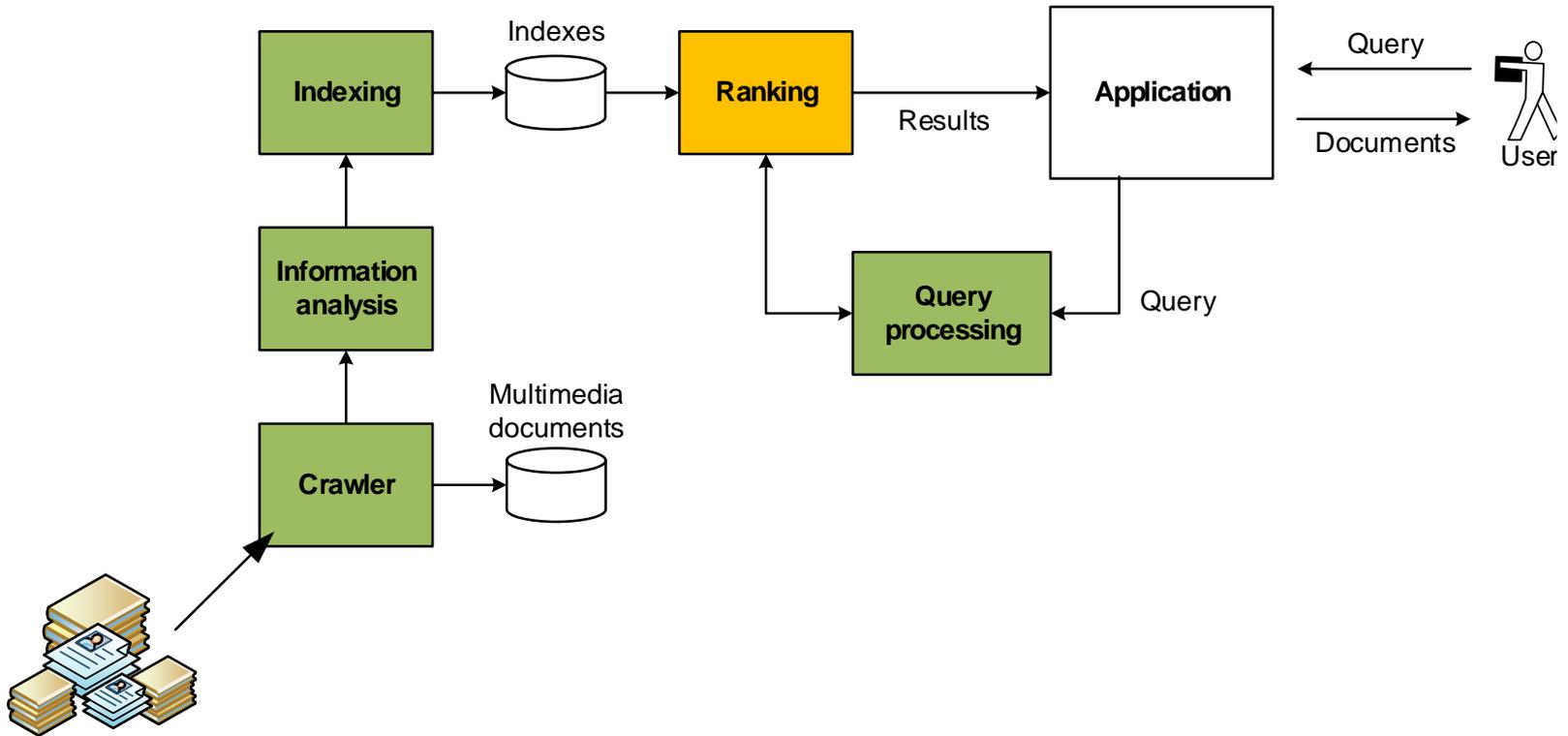
$$O(R|q, d) \propto \log \frac{p(q|d, r)p(r|d)}{p(q|d, \bar{r})p(\bar{r}|d)}$$

- LM Dirichlet
- LM Jelineck-Mercer

# Language Models

Information Retrieval

# Overview



# What is a language model?

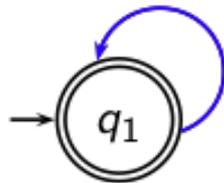
- We can view a finite state automaton as a deterministic language model.



- I wish I wish I wish I wish . . . Cannot generate: “wish I wish” or “I wish I”.
- Our basic model: each document was generated by a different automaton like this except that these automata are probabilistic.

# A probabilistic language model

- This is a one-state probabilistic finite-state automaton (a unigram LM) and the state emission distribution for its one state  $q_1$ .
  - STOP is not a word, but a special symbol indicating that the automaton stops.



$w$	$P(w q_1)$	$w$	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
		...	...

String = “frog said that toad likes frog STOP”

$$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2 = 0.00000000000048$$

# A language model per document

language model of $d_1$				language model of $d_2$			
$w$	$P(w .)$	$w$	$P(w .)$	$w$	$P(w .)$	$w$	$P(w .)$
STOP	.2	toad	.01	STOP	.2	toad	.02
the	.2	said	.03	the	.15	said	.03
a	.1	likes	.02	a	.08	likes	.02
frog	.01	that	.04	frog	.01	that	.05
		...	...			...	...

String = “frog said that toad likes frog STOP”

$$P(\text{string} | M_{d1}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2 = 0.0000000000048 = 4.8 \cdot 10^{-12}$$

$$P(\text{string} | M_{d2}) = 0.01 \cdot 0.03 \cdot 0.05 \cdot 0.02 \cdot 0.02 \cdot 0.01 \cdot 0.2 = 0.0000000000120 = 12 \cdot 10^{-12}$$

$$P(\text{string} | M_{d1}) < P(\text{string} | M_{d2})$$

- Thus, document  $d_2$  is “more relevant” to the string “frog said that toad likes frog STOP” than  $d_1$  is.

# Types of language models

- Unigrams:  $p_{uni}(t_1 t_2 t_3 t_4) = p(t_1)p(t_2)p(t_3)p(t_4)$
- Bigrams:  $p_{bi}(t_1 t_2 t_3 t_4) = p(t_1)p(t_2|t_1)p(t_3|t_2)p(t_4|t_3)$
- Multinomial distributions over words:

$$p(d) = \frac{l_d!}{f_{t_1,d}! f_{t_2,d}! \dots f_{t_n,d}!} p(t_1)^{f_{t_1,d}} p(t_2)^{f_{t_2,d}} \dots p(t_n)^{f_{t_n,d}}$$

# Probability Ranking Principle (PRP)

- PRP in action: Rank all documents by  $p(r = 1|q, d)$ 
  - Theorem: Using the PRP is optimal, in that it minimizes the loss (Bayes risk) under 1/0 loss
  - Provable if all probabilities correct, etc. [e.g., Ripley 1996]

$$p(r|q, d) = \frac{p(d, q|r)p(r)}{p(d, q)}$$

- Using odds, we reach a more convenient formulation of ranking :

$$O(R|q, d) = \frac{p(r = 1|q, d)}{p(r = 0|q, d)}$$

# Language models

- In language models, we do a formulation towards the query posterior given the document as a model.

$$\begin{aligned} O(R|q, d) &= \frac{p(r = 1|q, d)}{p(r = 0|q, d)} = \frac{p(d, q|r = 1)p(r = 1)}{p(d, q|r = 0)p(r = 0)} \\ &= \frac{p(q|d, r)p(d|r)p(r)}{p(q|d, \bar{r})p(d|\bar{r})p(\bar{r})} \propto \log \frac{p(q|d, r)p(r|d)}{p(q|d, \bar{r})p(\bar{r}|d)} \\ &= \log p(q|d, r) - \log p(q|d, \bar{r}) + \log \frac{p(r|d)}{p(\bar{r}|d)} \end{aligned}$$

# Language models

$$\log p(q|d, r) - \log p(q|d, \bar{r}) + \log \frac{p(r|d)}{p(\bar{r}|d)}$$

$$\approx \log p(q|d, r) + \text{logit}(p(r|d))$$

- The first term computes the probability that the query has been generated by the document model
- The second term can measure the quality of the document with respect to other indicators not contained in the query (e.g. PageRank or number of links)

# How to compute $p(q|d)$ ?

- We will make the same conditional independence assumption as for Naive Bayes (we dropped the  $r$  variable)

$$p(q|M_d) = \prod_{i=0}^{|q|} p(t_i|M_d)$$

- $|q|$  length of query;
- $t_i$  the token occurring at position  $i$  in the query

- This is equivalent to:  $p(q|M_d) = \prod_{t \in \{q \cap d\}} p(t|M_d)^{tf_{t,q}}$

- $tf_{t,q}$  is the term frequency (# occurrences) of  $t$  in  $q$

- Multinomial model (omitting constant factor)

# Parameter estimation

- The parameters  $p(t|M_d)$  are obtained from the document data as the maximum likelihood estimate:

$$p(t|M_d^{ml}) = \frac{f_{t,d}}{|d|}$$

- A single  $t$  with  $p(t|M_d) = 0$  will make  $p(q|M_d) = \prod p(t|M_d)$  zero.
- This can be smoothed with the prior knowledge we have about the collection.

# Smoothing

- Key intuition: A non-occurring term is possible (even though it didn't occur), . . .  
    . . . but no more likely than would be expected by chance in the collection.

- The maximum likelihood language model  $M_C^{ml}$  based on the term frequencies in the collection as a whole:

$$p(t|M_C^{ml}) = \frac{l_t}{l_C}$$

- $l_t$  is the number of times the term shows up in the collection
  - $l_C$  is the number of terms in the whole collection.
- We will use  $p(t|M_C^{ml})$  to “smooth”  $p(t|d)$  away from zero.

# LM with Jelineck-Mercer smoothing

- The first approach we can do is to create a mixture model with both distributions:

$$p(q|d, C) = \lambda \cdot p(q|M_d) + (1 - \lambda) \cdot p(q|M_c)$$

- Mixes the probability from the document with the general collection frequency of the word.
- High value of  $\lambda$ : “conjunctive-like” search – tends to retrieve documents containing all query words.
- Low value of  $\lambda$ : more disjunctive, suitable for long queries
- Correctly setting  $\lambda$  is very important for good performance.

# Mixture model: Summary

- What we model: *The user has some background knowledge about the collection and has a “document in mind” and generates the query from this document.*

$$p(q|d, C) \approx \prod_{t \in \{q \cap d\}} (\lambda \cdot p(t|M_d) + (1 - \lambda) \cdot p(t|M_c))$$

- The equation represents the probability that the document that the user had in mind was in fact this one.

# LM with Dirichlet smoothing

- We can use the prior knowledge about the mean of each term.
- The mean of the term in the collection should be our starting point when computing the term average on a document:
  - Imagine that we can add a fractional number occurrences to each term frequency.
  - Add  $\mu = 1000$  occurrences of terms to a document according to the collection distribution.
  - The frequency of each term  $t_i$  would increase  $\mu \cdot M_c(t_i)$
  - The length of each document increases by 1000.
- This will change the way we compute the mean of a term on a document.

# Dirichlet smoothing

- We end up with the maximum a posteriori estimate of the term average:

$$p(t|M_d^{MAP}) = \frac{f_{t,d} + \mu \cdot M_c(t)}{|d| + \mu}$$

- This is equivalent to using a Dirichlet prior with appropriate parameters.
- The ranking function becomes:

$$p(q|d) = \prod_{t \in q} \left( \frac{f_{t,d} + \mu \cdot M_c(t)}{|d| + \mu} \right)^{q_t}$$

# Experimental comparison

Method	TREC45				Gov2			
	1998		1999		2005		2006	
	P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP
Binary	0.256	0.141	0.224	0.148	0.069	0.050	0.106	0.083
2-Poisson	0.402	0.177	0.406	0.207	0.418	0.171	0.538	0.207
BM25	0.424	0.178	<b>0.440</b>	0.205	0.471	0.243	0.534	0.277
LMJM	0.390	0.179	0.432	0.209	0.416	0.211	0.494	0.257
LMD	<b>0.450</b>	<b>0.193</b>	0.428	<b>0.226</b>	<b>0.484</b>	<b>0.244</b>	<b>0.580</b>	<b>0.293</b>
BM25F					0.482	0.242	0.544	0.277
BM25+PRF	0.452	0.239	0.454	0.249	0.567	0.277	0.588	0.314
RRF	0.462	0.215	0.464	0.252	0.543	0.297	0.570	0.352
LR			0.446	0.266			0.588	0.309
RankSVM			0.420	0.234			0.556	0.268

# Experimental comparison

- For long queries, the Jelinek-Mercer smoothing performs better than the Dirichlet smoothing.
- For short queries, the Dirichlet smoothing performs better than the Jelinek-Mercer smoothing.

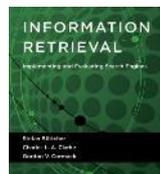
Method	Query	AP	Prec@10	Prec@20
LMJM	Title	0.227	0.323	0.265
LMD	Title	<b>0.256</b>	<b>0.352</b>	<b>0.289</b>
LMJM	Long	<b>0.280</b>	<b>0.388</b>	<b>0.315</b>
LMD	Long	0.279	0.373	0.303

# Summary

- Language Models
  - Jelinek-Mercer smoothing
  - Dirichlet smoothing
- Both models need to estimate one single parameter from the whole collection
  - (although there are known values that work well).
- References:



Chapter 12



Sections 9.1, 9.2 and 9.3