Computing Cosine Scores?

TAAT
Query Processing

CosineScore(q)

float $Scores[N] = 0$
Initialize $Length[N]$
for each query term t
do calculate $w_{t,q}$ and fetch postings list for t
    for each pair(d, $tf_{t,d}$) in postings list
    do $Scores[d] += wf_{t,d} \times w_{t,q}$
Read the array $Length[d]$
for each d
do $Scores[d] = Scores[d] / Length[d]$
return Top K components of $Scores[]$

بسم الله الرحمن الرحيم

دورة "استرجاع المعلومات" باللغة العربية - صيف ٢٠٢١

Information Retrieval – Summer 2021

5. Ranked Retrieval II: Language Modelling

Tamer Elsayed

Qatar University

Today’s Roadmap

- Probability Ranking Principle (PRP)
- Language Models
- Language Models in IR: Query Likelihood Model

PROBABILITY RANKING PRINCIPLE (PRP)

Probabilities .. Why?

- Uncertainty is inherent part of IR process.
- Probability theory is strong foundation for representing and manipulating uncertainty.

Probability Ranking Principle

Stephan Robertson

Probability Ranking Principle (PRP)

"If a reference retrieval system's response to each request is a **ranking of the documents** in the collection in order of **decreasing probability of relevance** to the user who submitted the request,

[*where the probabilities are **estimated as accurately as possible** on the basis of whatever data have been made available to the system for this purpose*], the overall **effectiveness** of the system to its user will be the **best that is obtainable** on the basis of those data."

Probability Ranking Principle (PRP)

"If a reference retrieval system's response to each request is a **ranking of the documents** in the collection in order of **decreasing probability of relevance** to the user who submitted the request,

[*where the probabilities are* ***estimated as accurately as possible*** *on the basis of whatever data have been made available to the system for this purpose*], the overall **effectiveness** of the system to its user will be the **best that is obtainable** on the basis of those data."

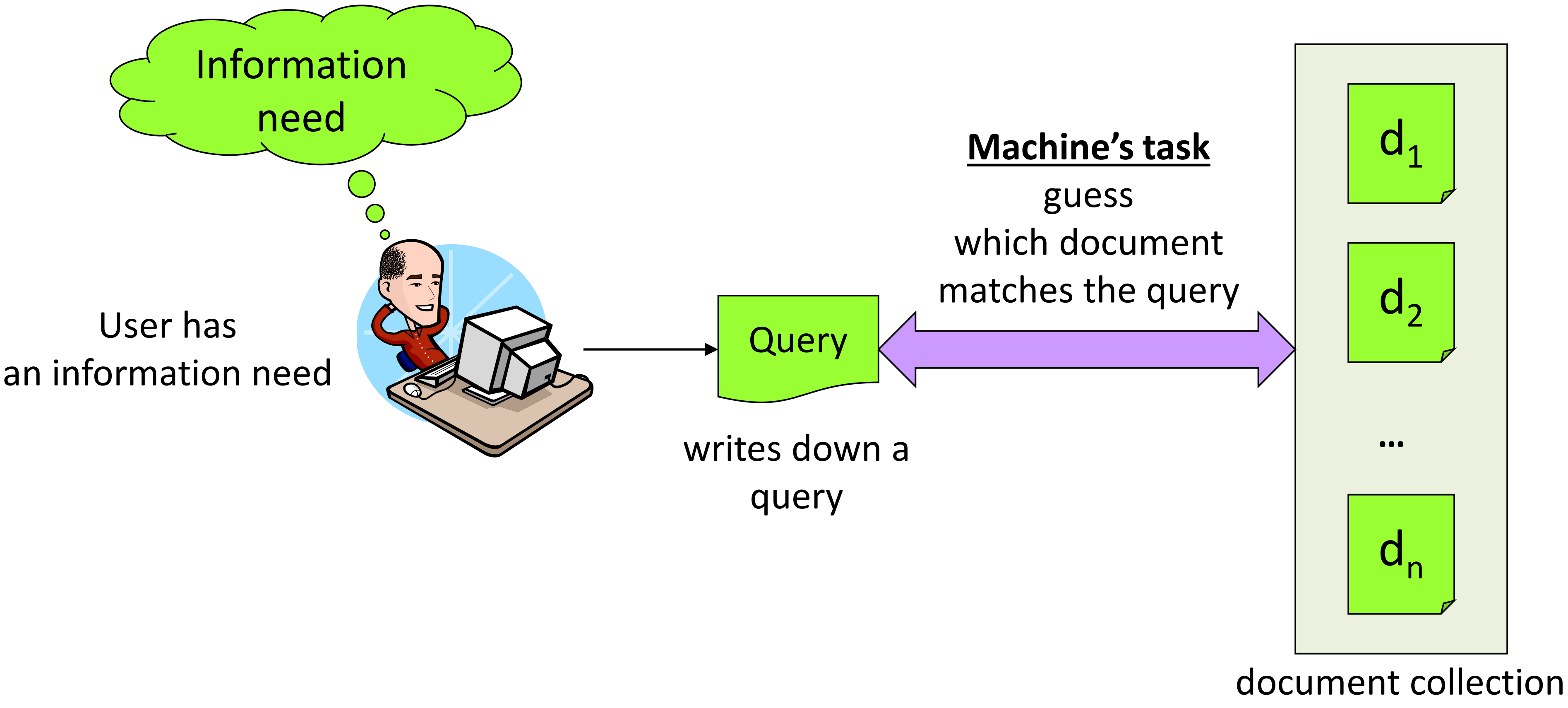
Formulation of PRP

- Rank docs by probability of relevance
  - $P(R|D_{r1}) > P(R|D_{r2}) > P(R|D_{r3}) > P(R|D_{r4}) > \ldots.$
- Estimate probability as accurate as possible
  - $P_{est}(R|D) \approx P_{true}(R|D)$
- Estimate with all possibly-available data
  - $P_{est}(R \mid \text{doc, session, user profile, …})$
- Best possible accuracy can be achieved with that data
  - the perfect IR system!
  - Is it really doable?

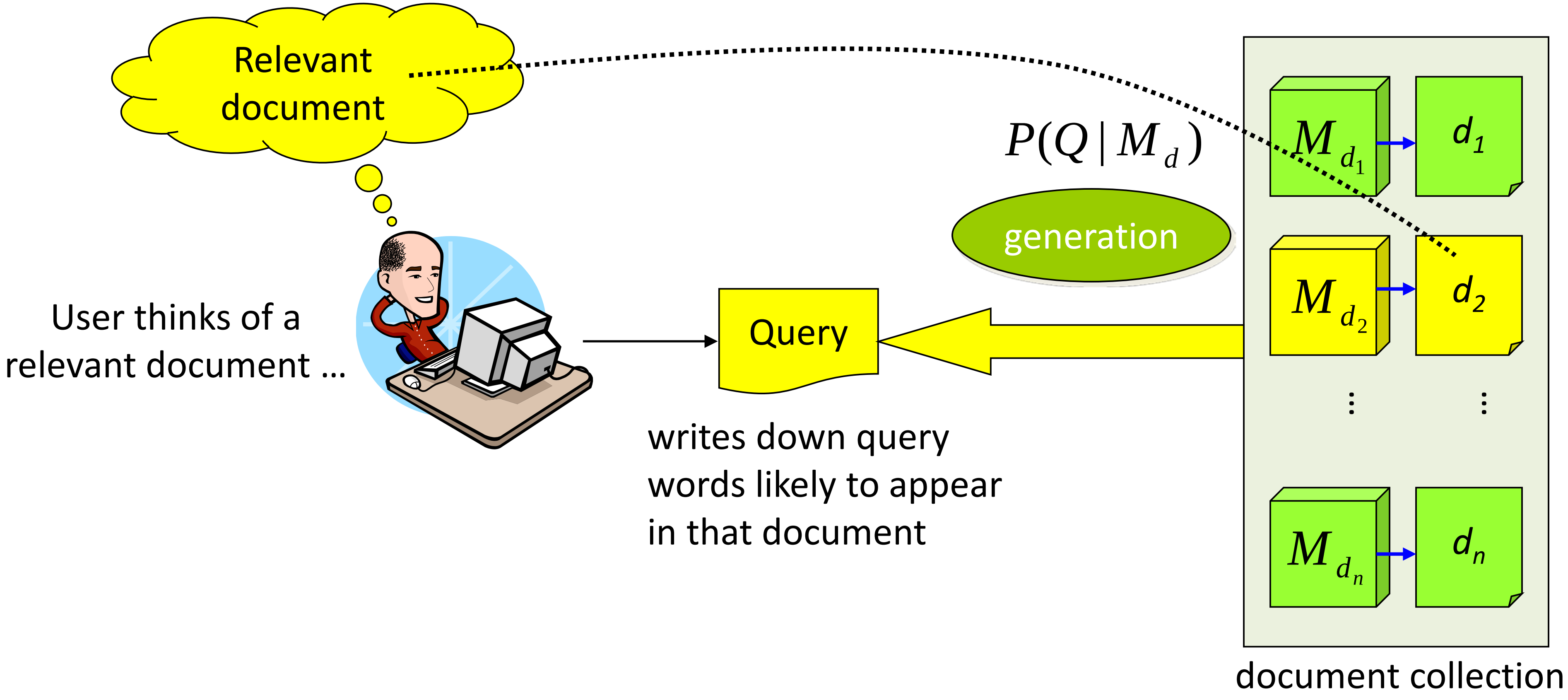
How to estimate the probability of relevance?

LANGUAGE MODELS

"Noisy-Channel" Model of IR



IR based on Language Model (LM)



The LM approach directly exploits that idea!

Concept …

- Coming up with good queries?
  - Think of words that would likely appear in a relevant document
  - Use those words as the query

- A document is a good match to a query if the document model is likely to generate the query
  - happens if the document contains the query words often.

Language Model Approach …

1. Build a probabilistic language model M_d from each document d

2. Rank documents based on the probability of the model generating the query: $P(q|M_d)$.

What & How?

What's a Language Model?

A** language model **is a probability distribution over strings drawn from some vocabulary

Unigram LM M_d

w	P(w)
qatar	0.015
campus	0.003
house	0.00000
education	0.006
buildings	0.0002
university	0.02
masjid	0.0001
health	0.0001
green	0.00004
…	…

- A *topic* in a document (or query) can be represented as a language model
- Words that tend to occur often when discussing a topic will have high probabilities in the corresponding language model.

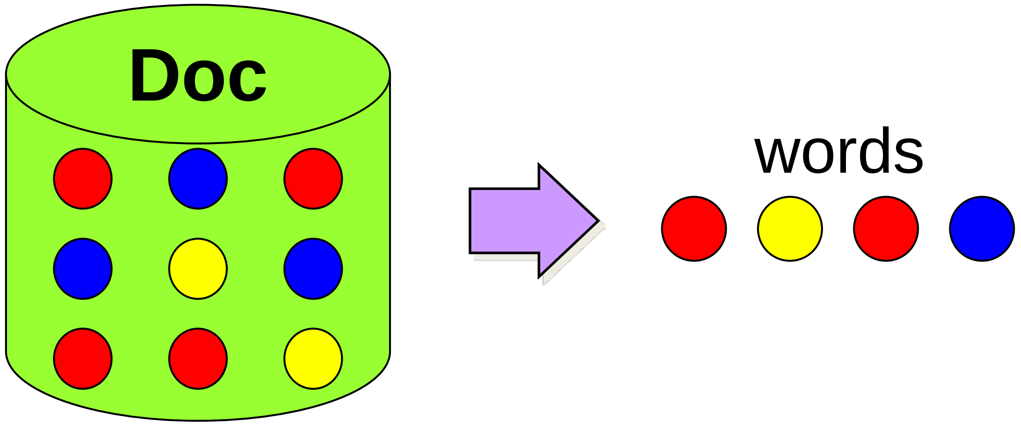
Can we estimate probability of generating text from language models?

Unigram Language Model

- A statistical model for generating text
  - Terms are randomly drawn from a document (with replacement)
  - Probability Distribution: $P(q_i)$

M_d ?

P (●) = 4/9
P (●) = 2/9
P (●) = 3/9



P (● ● ● ●) = P (●) × P (●) × P (●) × P (●)
= (4/9) × (2/9) × (4/9) × (3/9)

Comparing Language Models

Model M_{d1}	
P(w)	**w**
0.2	qatar
0.0001	grant
0.01	university
0.0005	profile
0.0003	campus
0.0001	research
…	

Model M_{d2}	
P(w)	**w**
0.2	qatar
0.1	grant
0.001	university
0.01	profile
0.03	campus
0.02	research
…	

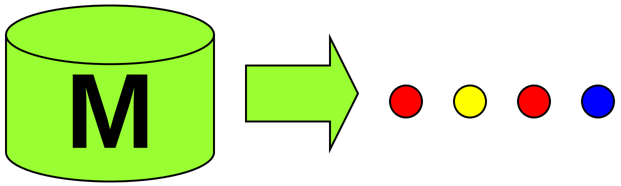
Useful?!

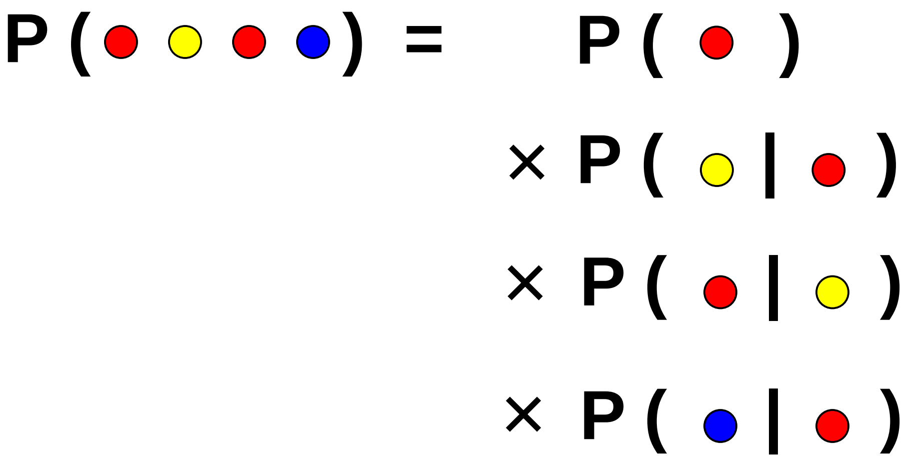
text:	qatar	university	research	grant	profile	P(text|M_d)
M_{d1}:	0.2	0.01	0.0001	0.0001	0.0005	0.0000000000001
M_{d2}:	0.2	0.001	0.02	0.1	0.01	0.000000004

P(text|M_{d2}) > P(text|M_{d1})

Bigram Language Models

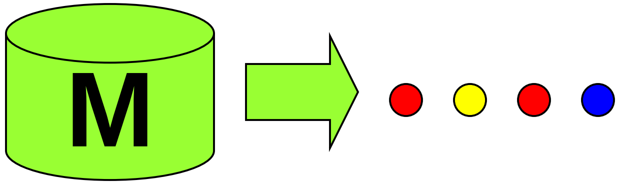
- A statistical model for generating text
  - Probability Distribution: $P(q_i|q_{i-1})$

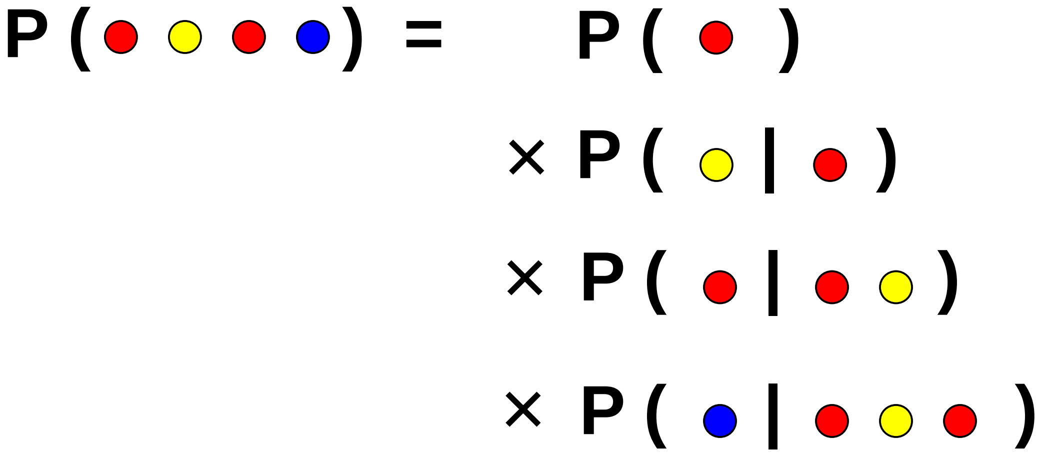




Stochastic Language Models

- A statistical model for generating text
  - Probability distribution: $P(q_i|q_{i-1}q_{i-2}\ldots q_1)$





Summary: Unigram and higher-order models

Easy & Effective!

- **Unigram Language Models**

  P (🔴 🟡 🔴 🔵)

  = P (🔴) P (🟡) P (🔴) P (🔵)

- **Bigram** (generally, *n*-gram) Language Models

  P (🔴 🟡 🔴 🔵)

  = P (🔴) P (🟡 | 🔴) P (🔴 | 🟡) P (🔵 | 🔴)

- ….

- **Stochastic**

  P (🔴 🟡 🔴 🔵)

  = P (🔴) P (🟡 | 🔴) P (🔴 | 🔴 🟡) P (🔵 | 🔴 🟡 🔴)


15
POLL

15

In Bigram Language Models, we assume that the probability of observing a word …

- is independent of any previous word
- depends only on its previous word
- depends only on its previous 2 words

Text generation probabilities computed using Bigram Language Models are more accurate than Unigram Language Models.

- Yes
- No

Today's Roadmap

- Probability Ranking Principle (PRP)
- Language Models
- Language Models in IR: Query Likelihood Model

Bruce Croft

Using Language Models in IR

- Each document is treated as basis for a language model.
- Given a query q, **rank documents based on $\boldsymbol{P(d|q)}$.**

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)}$$

  - $P(q)$ is the same for all documents → ignore
  - $P(d)$ is the prior – often treated as the same for all d
    - But we can give a prior to "high-quality" documents, e.g., those with high PageRank (later to be discussed).
  - $P(q|d)$ is the probability of q given d.
- So to rank documents according to relevance to q, ranking according to $P(d|q)$ or $\boldsymbol{P(q|d)}$ is equivalent.

LM in IR: Basic idea

- We attempt to model the query generation process.
- Then we rank documents by the probability that a query would be observed as a random sample from the respective document model.

Rank according to $P(q|d)$

Rank according to $P(q|M_d)$

Query Likelihood Model

- We will make the conditional independence assumption.

$$P(q|M_d) = P(\langle q_1, \ldots, q_n \rangle | M_d) = \prod_{1 \le k \le n} P(q_i|M_d)$$

n : length of q; q_i: term i in q

How to estimate?

Parameter estimation

- Start with maximum likelihood estimates (MLE)

$$P_{MLE}(q_i|M_d) = \frac{tf_{q_i,d}}{|d|}$$

|d|: length of d; $tf_{q_i,d}$: term freq of q_i in d

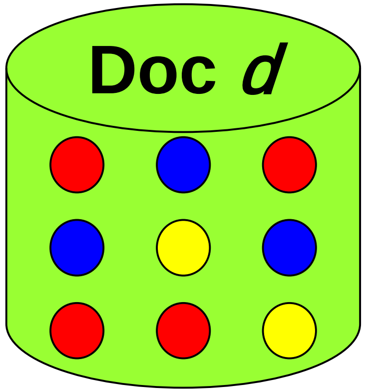
- Probability of a query q to be generated by a LM M_d:

$$P_{MLE}(q|M_d) = \prod_{q_i \in q} \frac{tf_{q_i,d}}{|d|}$$

Example

$$P(\color{red}{\bullet}\ \color{gold}{\bullet}\ \color{red}{\bullet}\ \color{blue}{\bullet}) = P(\color{red}{\bullet})^2 \times P(\color{gold}{\bullet}) \times P(\color{blue}{\bullet})$$
$$= (4/9)^2 \times (2/9) \times (3/9) = 0.0146$$

$$P(\color{red}{\bullet}\ \color{gold}{\bullet}\ \color{green}{\bullet}\ \color{blue}{\bullet}) = P(\color{red}{\bullet}) \times P(\color{gold}{\bullet}) \times P(\color{green}{\bullet}) \times P(\color{blue}{\bullet})$$
$$= (4/9) \times (2/9) \times (0/9) \times (3/9) = 0 \ \color{red}{!!}$$

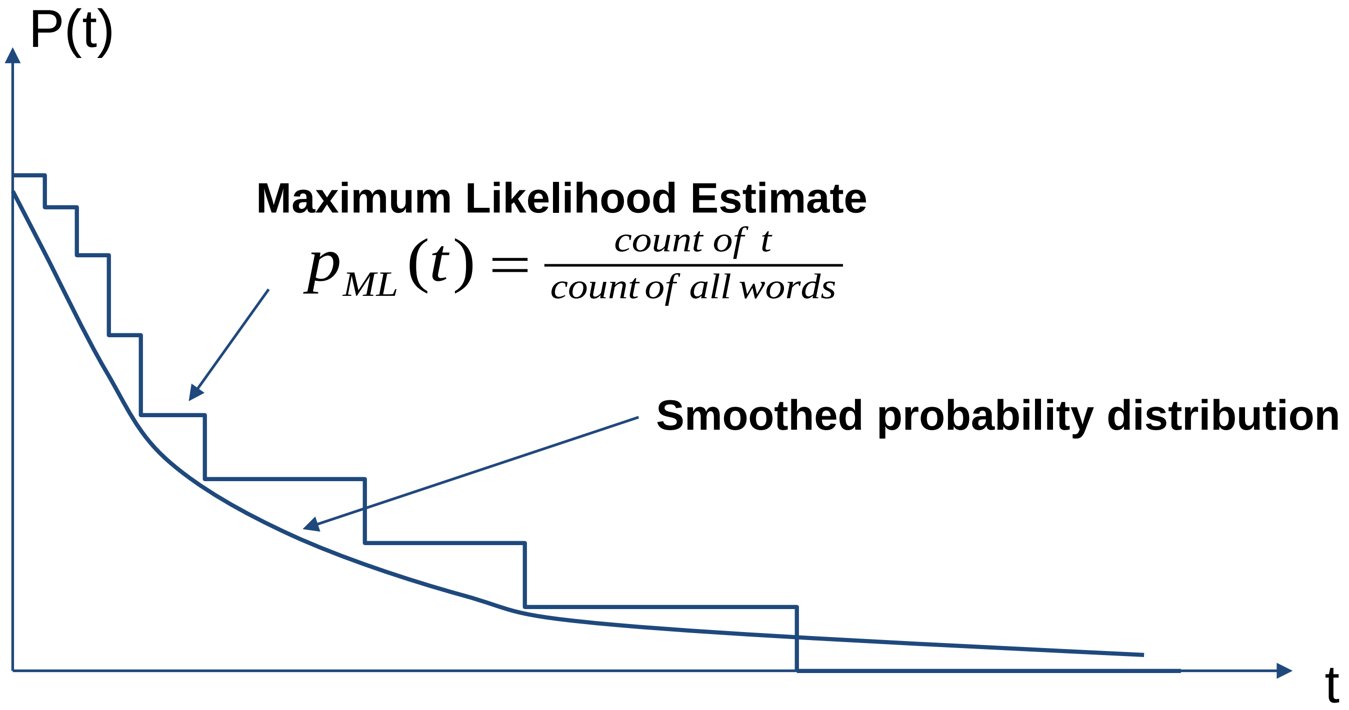


Model assigns zero probability to unseen term!
We would give a single term "veto power"!

Is there a better way to handle unseen terms?

Smoothing

The solution: *"smooth" the term probabilities*



- lower (or discount) the probability estimates for ***seen*** words in the document text
- assign that "left-over" probability to the estimates for ***unseen*** (and also the seen ones)

Is smoothing logical?

- Document texts are *samples* from the language model
- Missing words should not have zero probability of occurring
- A missing term is possible (even though it didn't occur)
  - . . . but no more likely than would be expected by chance in the collection.

Jelinek-Mercer Smoothing

$$P(q_i|M_d) = (1-\lambda)\, P_{MLE}(q_i|M_d) + \lambda\, P_{MLE}(q_i|M_C)$$

$$P_{MLE}(q_i|M_C) = \frac{ctf_{q_i,C}}{|C|}$$

- Mixes the probability from the document with the general collection frequency of the word.
- Estimate for **unseen words** is $\lambda\, P_{MLE}(q_i|M_C)$
  - λ is a parameter controlling probability for unseen words
  - Based on *collection language model* (***background LM***) M_C
  - $P_{MLE}(q_i|M_C)$ is the probability for query word q_i in M_C (***background probability***)
- Estimate for **observed words** is $(1-\lambda)\, P_{MLE}(q_i|M_d) + \lambda\, P_{MLE}(q_i|M_C)$

Jelinek-Mercer Smoothing

$$P(q_i|M_d) = (1-\lambda)\, P_{MLE}(q_i|M_d) + \lambda\, P_{MLE}(q_i|M_C)$$

- **Low value of λ**: "conjunctive-like" search – tends to retrieve documents containing all query words.
- **High value of λ:** more disjunctive, suitable for long queries
- Correctly setting λ is important for good performance.

- **Final Ranking function:**

$$P_{MLE}(q|M_d) = \prod_{\forall q_i \in q} \left((1-\lambda)\, P_{MLE}(q_i|M_d) + \lambda\, P_{MLE}(q_i|M_C)\,\right)$$

Example with JM Smoothing

- **Collection**: d_1 and d_2
- ***d*$_1$** : “Jackson was one of the most talented entertainers of all time”
- ***d*$_2$**: “Michael Jackson anointed himself King of Pop”
- **Query *q***: Michael Jackson
- Use mixture model with $\lambda = 1/3$

- $P(q|d_1) = [2/3 * 0/11 + 1/3 * 1/18] \cdot [2/3 * 1/11 + 1/3 * 2/18]$
- $P(q|d_2) = [2/3 * 1/7 + 1/3 * 1/18)] \cdot [2/3 * 1/7 + 2/3 * 2/18]$
- Ranking: $d_2 > d_1$

15

16

Smoothing allows query words that are not observed in a document to ...

- contribute to the score of that document with a non-zero probability
- make the score of that document zero

The background model assigns a probability for a word based on ...

- its collection frequency
- its document frequency
- its term frequency in the given document

Notes on Query Likelihood Model

- It generally has similar effectiveness to BM25
- With more sophisticated techniques, it outperforms BM25
  - Topic models
- There are several alternative smoothing techniques
  - JM was just an example ☺

Language Models

Unigram language model

- probability distribution over the words in a language
  - associates a probability of occurrence with every word
- generation of text consists of pulling words out of a “bucket” according to the probability distribution and replacing them

n-gram language model

- some applications use bigram and trigram language models where probabilities depend on previous words
- predicts a word based on the previous n-1 words

LMs for IR: 3 Possibilities

- Probability of generating the query text from a document language model
- Probability of generating the document text from a query language model
- Comparing the language models representing the query and document topics


Feedback

Can we automatically "modify" the query to get better results?