THE UNIVERSITY *of* EDINBURGH

Text Technologies for Data Science

INFR11145

Ranked Retrieval (2)

Instructor:
Youssef Al Hariri

11-Oct-2023

Lecture Objectives

- Learn about Probabilistic models
  - BM25

- Learn about LM for IR

Recall: VSM & TFIDF term weighting

- Combines TF and IDF to find the weight of terms

$$w_{t.d} = \big(1 + log_{10} tf(t,d)\big) \times log_{10}\left(\frac{N}{df(t)}\right)$$

- For a query *q* and document *d*, retrieval score *f(q,d)*:

$$Score(q,d) = \sum_{t \in q \cap d} w_{t.d}$$

- TFIDF observations

Can we do better?

  - Term appearing more in a doc gets higher weight (TF)
  - First occurrences are more important (log)
  - Rare terms are more important (IDF)
  - Bias towards longer documents

IR Model

- VSM is very heuristic in nature
  - No notion of relevance is there (still works well)
  - Any weighting scheme, similarity measure can be used
    - Components not interpretable → no guide for what to try next
    - More engineering rather than theory → tweak, run, observe, tweak …
  - Very popular, hard to beat, strong baseline
    - Easy to adapt good ideas from other models

- **Probabilistic Model** of retrieval
  - Mathematical formulisation for relevant / irrelevant sets
    - Explicitly defines random variables (R,Q,D)
    - Specific about what their values are
    - State the assumptions behind each step
    - Watch out for contradictions

Probabilistic Models

- Concept: Uncertainty is inherent part of IR process
- Probability theory is strong foundation for representing and manipulating uncertainty
- Probability Ranking Principle (1977)

Stephan Robertson

Probability Ranking Principle

- "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."

- Basis for most probabilistic approaches for IR

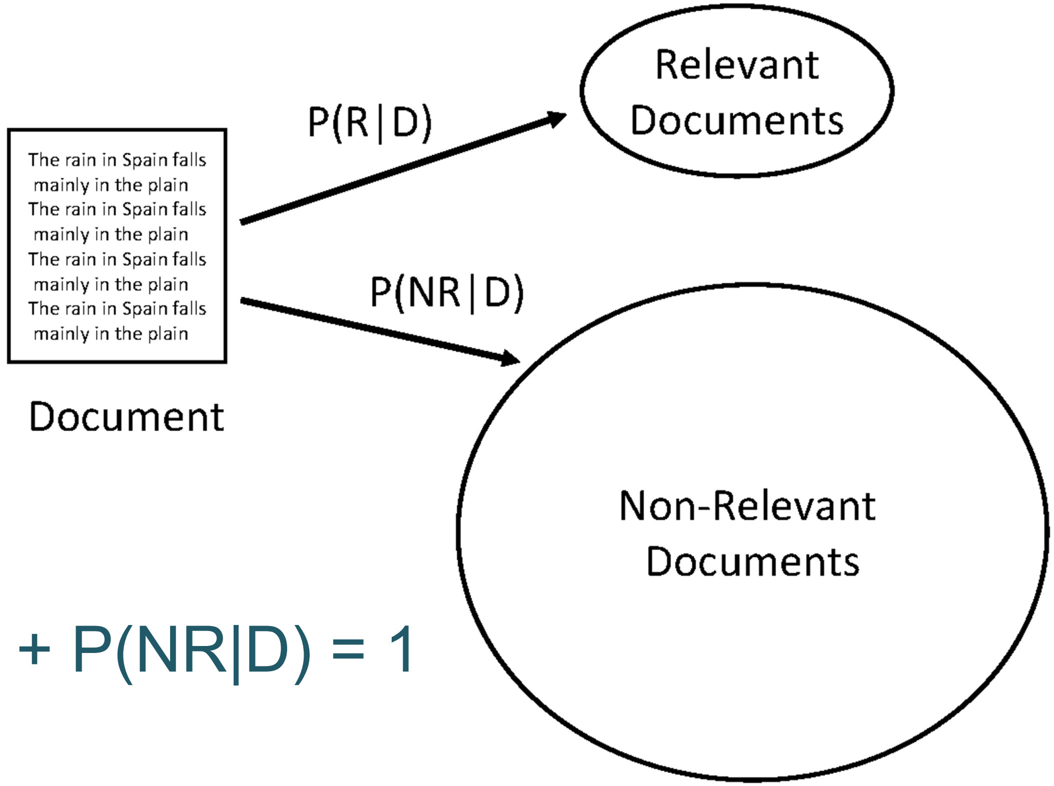
Formulation of PRP

- 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
  - $P_{est}(R \mid \text{doc, session, context, user profile, } \ldots)$
- Best possible accuracy can be achieved with that data
  - → the perfect IR system
  - Is it really doable?

- **How to estimate the probability of relevance?**

PRP Concept

- Imagine IR as a classification problem



$$P(R|D) + P(NR|D) = 1$$

- Document *D* is **relevant** if ***P*(*R*|*D*) > *P*(*NR*|*D*)**

THE UNIVERSITY of EDINBURGH

Probability of Relevance

- What is P_{true}(rel | doc, query, session, user, …)?
  - Isn't relevance just the user's opinion?
    - User decides relevant or not, what is the "probability" thing?
- Search algorithm cannot look into your head (yet!)
  - Relevance depends on factors that algorithm cannot observe
    - SIGIR 2016 best paper award: *Understanding Information Need: an fMRI Study*
- Different users may disagree on relevance of the same doc
  - Even similar users, doing the same task, in the same context
- P_{true} (rel | Q, D):
  - Proportion of all unseen users / context / tasks for which D would have judged relevant to Q
- Similar to: P(die=6 | even and not square)

Okapi BM25 Model

- Based on the probabilistic model
  - A document D is relevant if **P(R=1|D) > P(R=0|D)**
- Extension to the "binary independence model"
  - **Binary features**: Document represented by a vector of binary features indicating term occurrence
  - Assume **term independence** (Naïve Bayes assumption) → BOW trick
- In 1995, *Stephan Robertson* with his group came up with the **BM25** Formula as part of the **Okapi** project.
- It outperformed all other systems in TREC
- Popular and effective ranking algorithm

Okapi BM25 Ranking Function

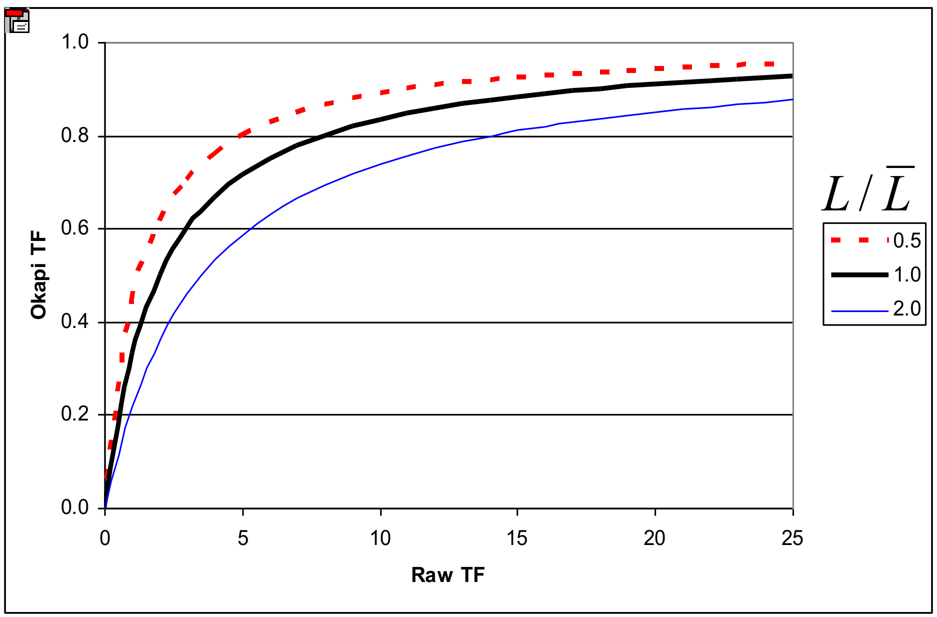
- Let L_d be the number of terms in document d
- Let $\bar{L}$ be the average number of terms in a document

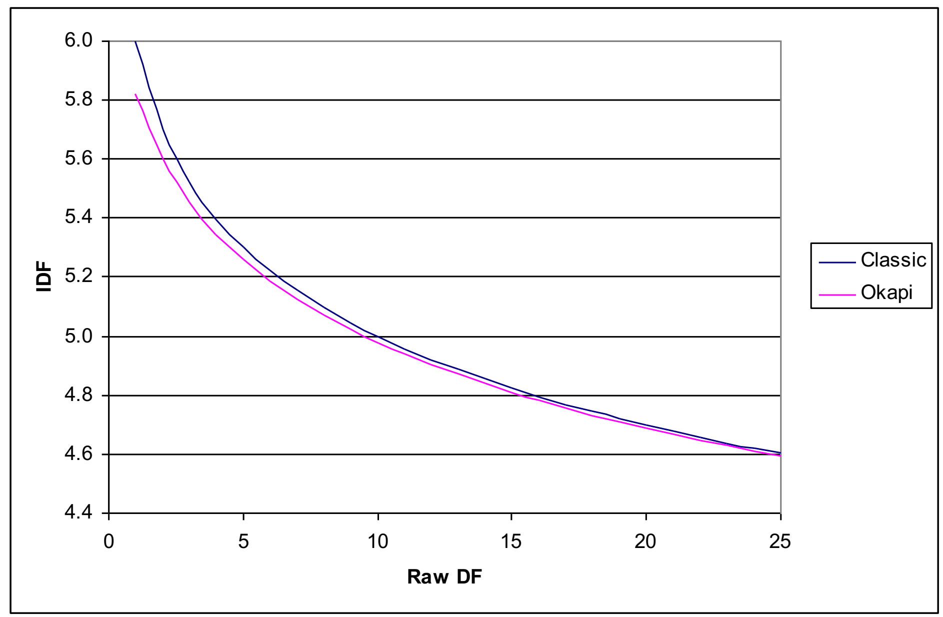
$$w_{t.d} = \frac{tf_{t,d}}{k.\frac{L_d}{\bar{L}} + tf_{t,d} + 0.5} \times log_{10}\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right)$$

- Best practices: k=1.5

Okapi BM25 Ranking Function

$$w_{t.d} = \underbrace{\frac{tf_{t,d}}{1.5\frac{L_d}{\bar{L}} + tf_{t,d} + 0.5}}_{tf\text{ component}} \times log_{10}\underbrace{\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right)}_{idf\text{ component}}$$



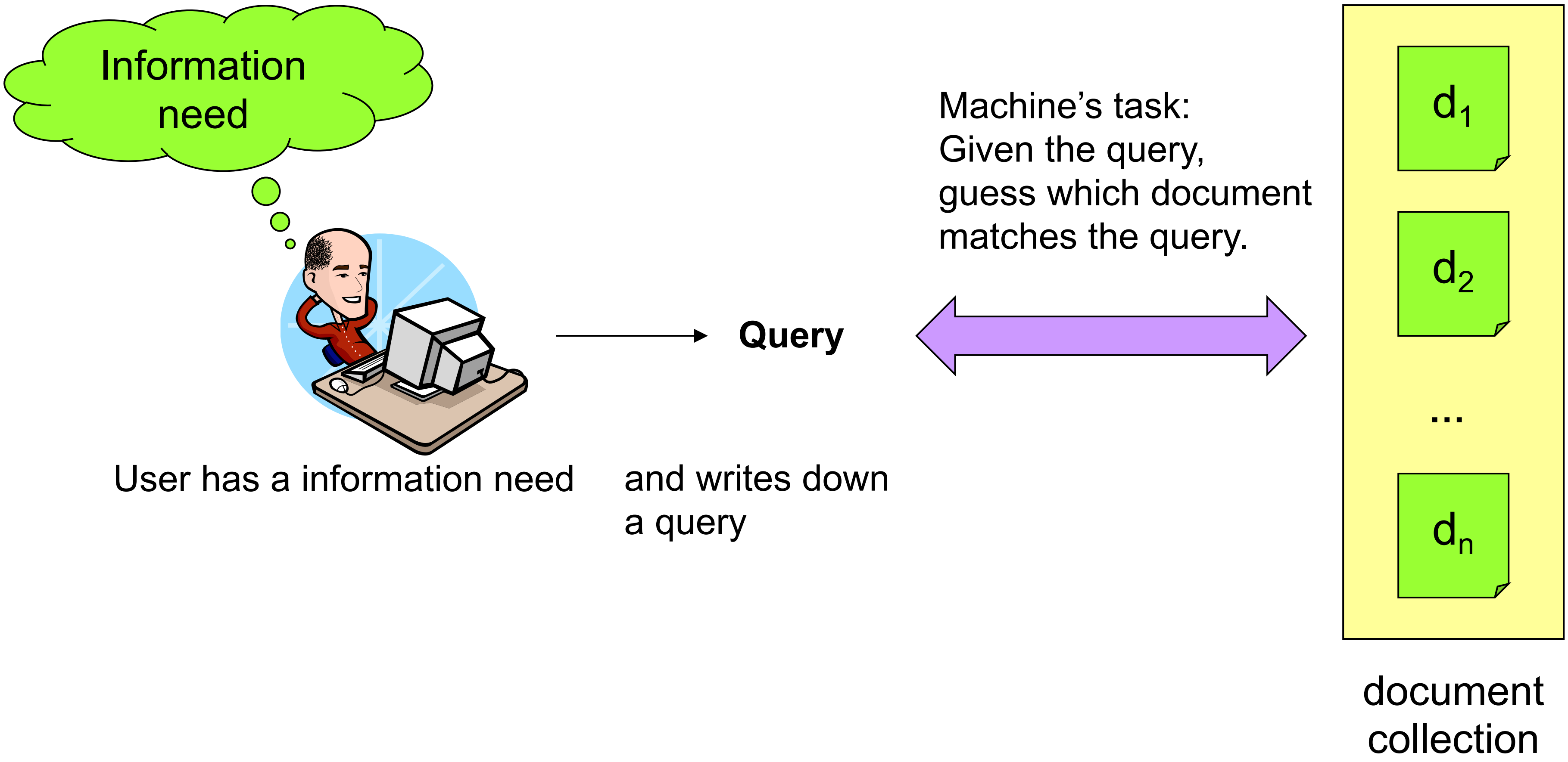


Probabilistic Model in IR

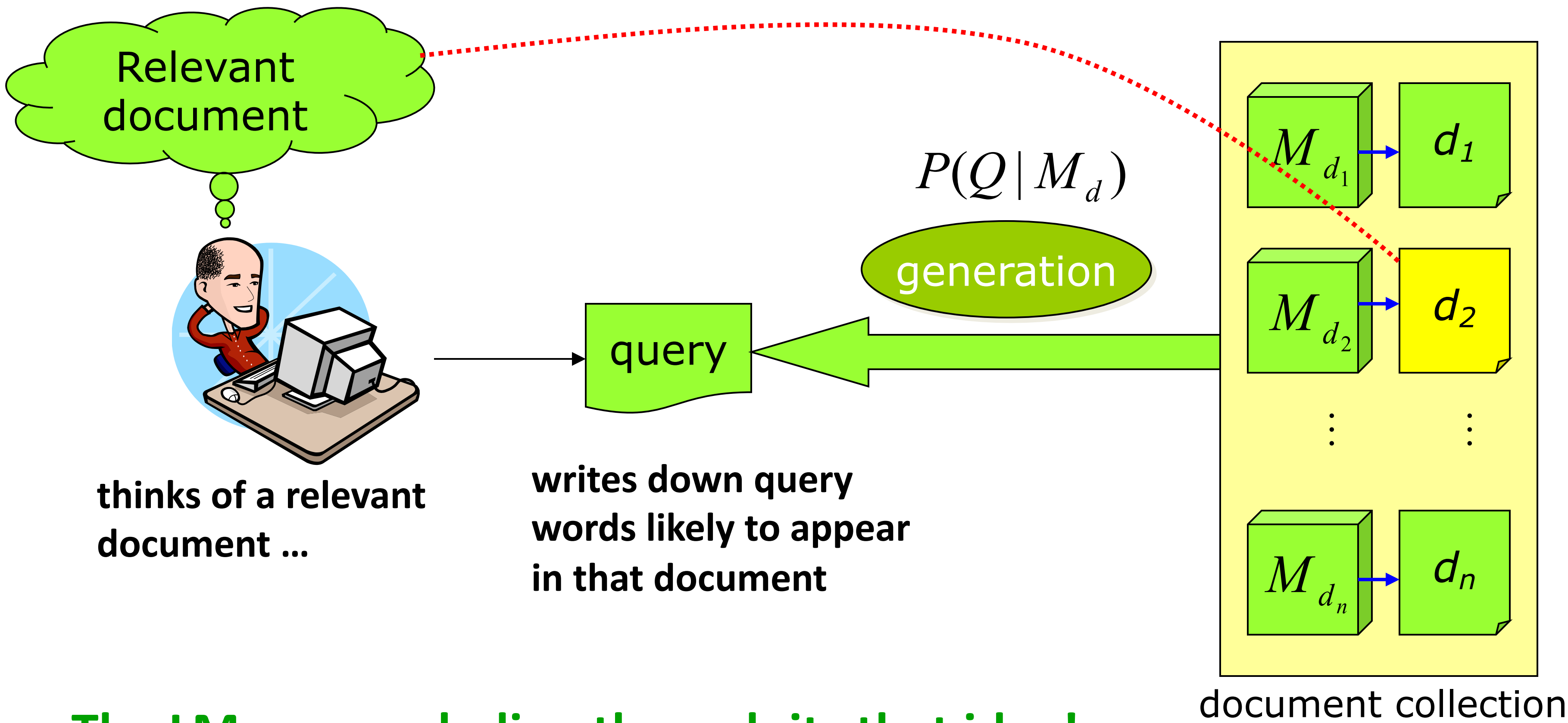
- Focuses on the probability of relevance of docs
- Could be mathematically proved
- Different ways to apply it
- BM25 is the most common formula for it

- What other models could be still used in IR?

“Noisy-Channel” Model of IR



IR based on Language Model (LM)



- **The LM approach directly exploits that idea!**
  - a document is a good match to a query if the document model is likely to generate the query

Concept

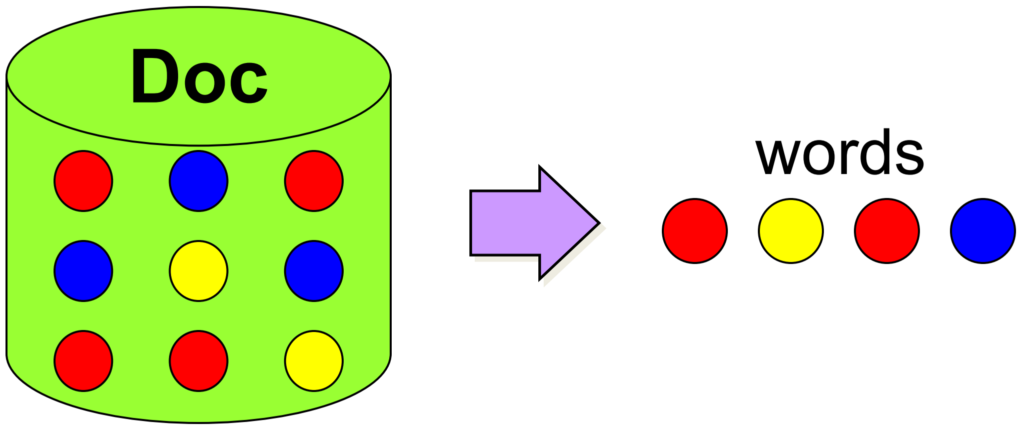
- Coming up with good queries?
  - Think of words that would likely appear in a relevant doc
  - Use those words as the query
- The language modeling approach to IR directly models that idea
  - a document is a good match to a query if the document model is likely to generate the query
    - happens if the document contains the query words often.
- Build a probabilistic language model M_d from each document d
- Rank documents based on the probability of the model generating the query: $P(q|M_d)$.

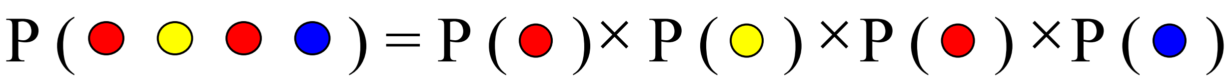
Language Model (LM)

- A language model is a probability distribution over strings drawn from some vocabulary
- A topic in a document or query can be represented as a language model
  - i.e., words that tend to occur often when discussing a topic will have high probabilities in the corresponding language model

Unigram LM

- Terms are randomly drawn from a document (with replacement)





$$= (4/9) \times (2/9) \times (4/9) \times (3/9)$$

Example

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

- This is a one-state probabilistic finite-state automaton – a unigram language model.
- S = "frog said that toad likes frog STOP"
  P(S) = 0.01 × 0.03 × 0.04 × 0.01 × 0.02 × 0.01 × 0.02
  = 0.0000000000048

Comparing LMs

- **M_{d1}**
  LM generated from Doc 1
- **M_{d2}**
  LM generated from Doc 2
- Try to generate sentence **S** from **M_{d1}** & **M_{d2}**

Model M_{d1}

P(w)	w
0.2	the
0.0001	yon
0.01	class
0.0005	maiden
0.0003	sayst
0.0001	pleaseth
…	

Model M_{d2}

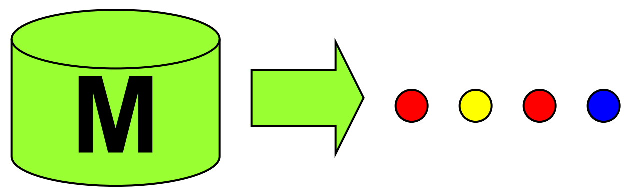
P(w)	w
0.2	the
0.1	yon
0.001	class
0.01	maiden
0.03	sayst
0.02	pleaseth
…	

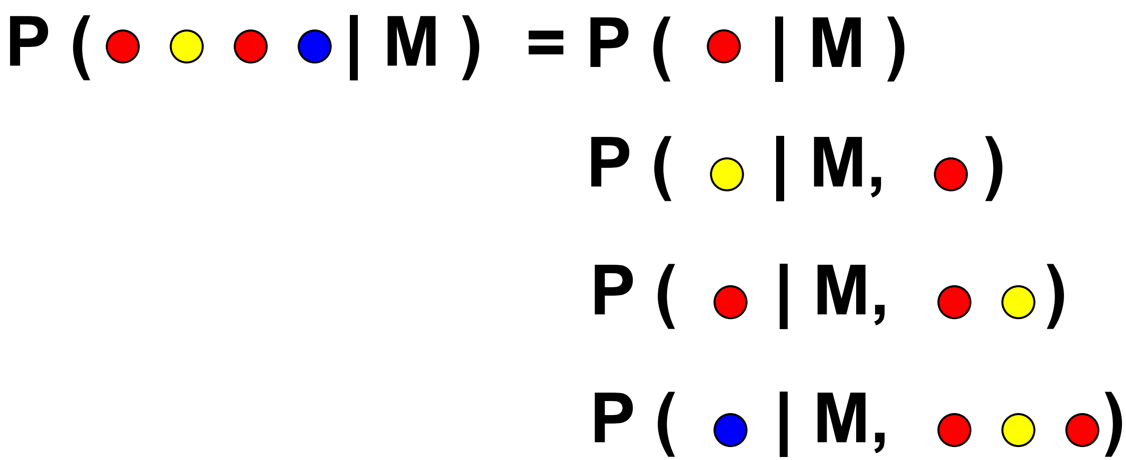
text:	the	class	pleaseth	yon	maiden	P(S)
M_{d1}:	0.2	0.01	0.0001	0.0001	0.0005	0.00000000000001
M_{d2}:	0.2	0.001	0.02	0.1	0.01	0.000000004

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

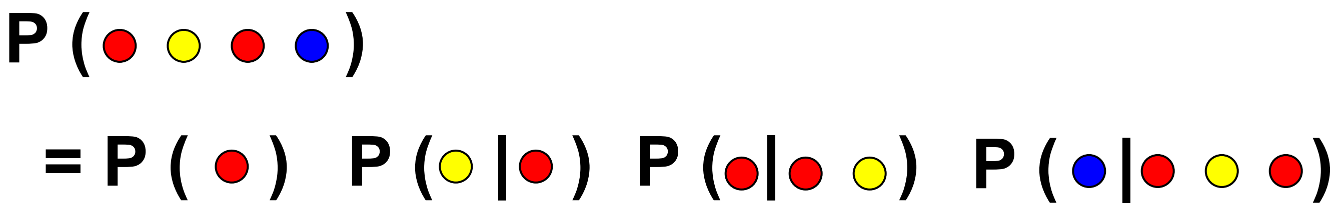
Stochastic Language Models

- A statistical model for generating text
  - Probability distribution over strings in a given language

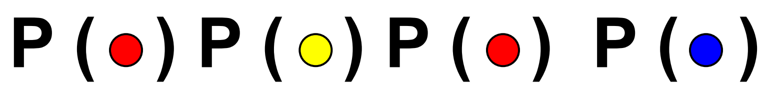




Unigram and Higher-order LM



- **Unigram Language Models**



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

LM in IR

- Each document is treated as basis for a LM.
- Given a query q, rank documents based on *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(q|d)* and *P(d|q)* 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.
- That is, we rank according to ***P(q|d)***.

P(q|d)

Query Likelihood Model

- We will make the conditional independence assumption.

$$P(q|M_d) = P\left(\langle t_1, \ldots, t_{|q|} \rangle \middle| M_d\right) = \prod_{1 \le k \le |q|} P(t_k|M_d)$$

$|q|$: length of q; t_k : token occurring at position k in q

- This is equivalent to:

$$P(q|M_d) = \prod_{each\ term\ t\ in\ q} P(t|M_d)^{tf_{t,q}}$$

$tf_{t,q}$: term frequency (# occurrences) of t in q

- Multinomial model (omitting constant factor)

Parameter estimation

- Probability of a term t in a LM M_d using Maximum Likelihood Estimation (MLE)

$$P(t|M_d) = \frac{tf_{t,d}}{|d|}$$

|*d*|: length of *d*;
$tf_{t,d}$: # occurrences of *t* in *d*

- Probability of a query *q* to be noticed in a LM M_d:

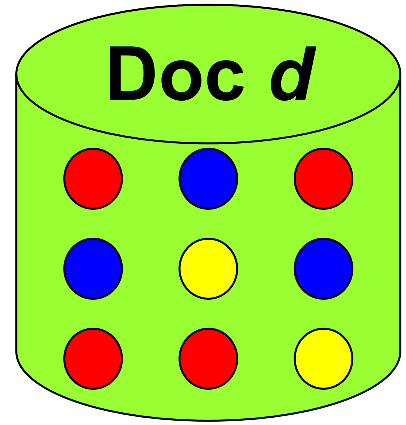
$$P(q|M_d) = \prod_{\forall\, t \in q} \left(\frac{tf_{t,d}}{|d|}\right)^{tf_{t,q}}$$

Example

P(🔴 🟡 🔴 🔵) = P(🔴)^2 × P(🟡) × P(🔵)

$$= (4/9)^2 \times (2/9) \times (3/9) = 0.0146$$

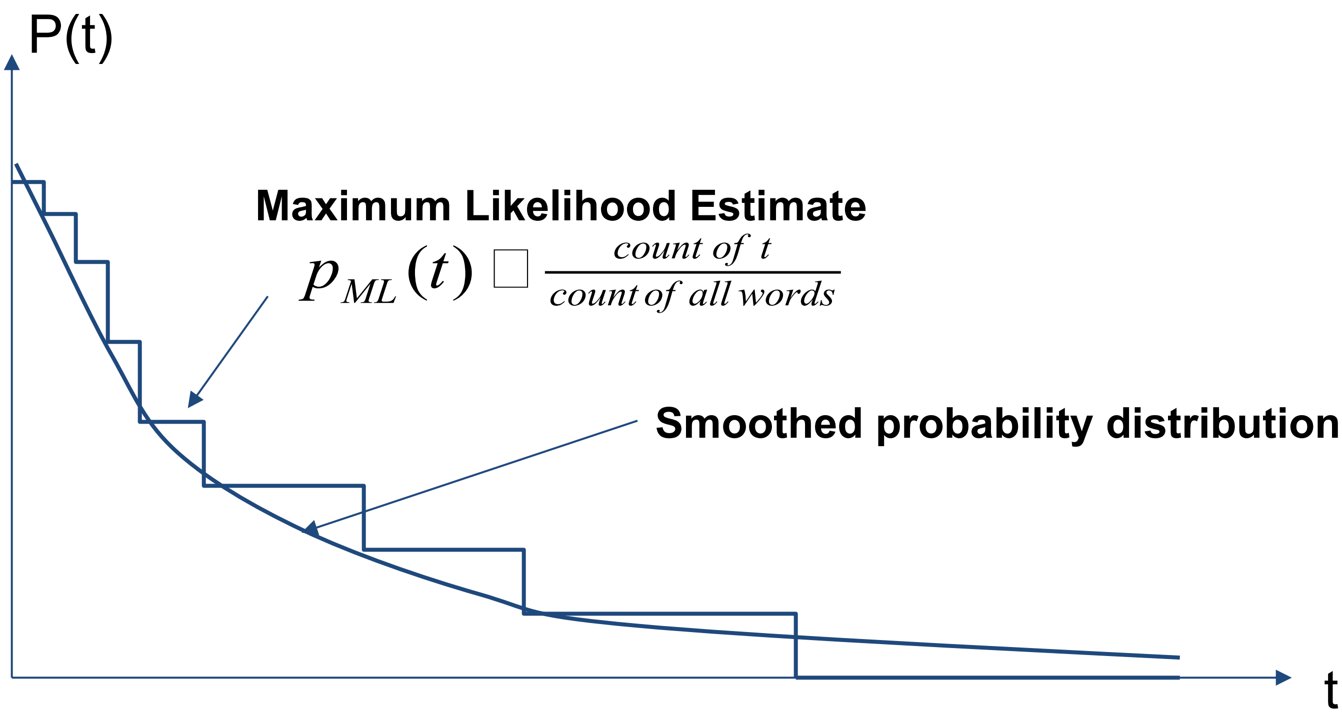
P(🔴 🟡 🟢 🔵)



- Is that fair?
  - In VSM, S(Q,D) was summation, works more like OR in Boolean search. Missing one term reduces score only
  - In language model, S(Q,D) is P(Q|D) → Multiplication of probabilities → missing one term makes score = 0
  - Is there a better way to handle unseen terms?

Smoothing

- Problem: Zero frequency
- Solution: “Smooth” terms probability



Smoothing

- Document texts are a sample 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.
- A technique for estimating probabilities for missing (or unseen) words
  - Overcomes data-sparsity problem
  - lower (or discount) the probability estimates for words that are seen in the document text
  - assign that "left-over" probability to the estimates for the words that are not seen in the text (and also on the seen ones)

Mixture Model

$$P(t|d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$$

- Mixes the probability from the document with the general collection frequency of the word.
- Estimate for unseen words is $(1-\lambda)\ P(t|M_c)$
  - Based on collection language model (background LM)
  - $P(t|M_c)$ is the probability for query word *i* in the collection language model for collection C (background probability)
  - λ is a parameter controlling probability for unseen words
- Estimate for observed words is

CF

$$\lambda\ P(t|M_d) + (1-\lambda)\ P(t|M_c)$$

Jelinek-Mercer Smoothing

$$P(t|d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$$

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

- Final Ranking function:

$$P(q|M_d) \propto \prod_{1 \leq k \leq |q|} \left(\lambda \cdot P(t_k|M_d) + (1 - \lambda) \cdot P(t_k|M_c)\right)$$

Example

- **Collection**: d_1 and d_2
- $\boldsymbol{d_1}$: "Jackson was one of the most talented entertainers of all time"
- $\boldsymbol{d_2}$: "Michael Jackson anointed himself King of Pop"
- **Query *q***: Michael Jackson
- Use mixture model with λ = 1/2
- $P(q|d_1) = \overbrace{[(0/11 + 1/18)/2]} \cdot \overbrace{[(1/11 + 2/18)/2]} \approx 0.003$
- $P(q|d_2) = [(1/7 + 1/18)/2] \cdot [(1/7 + 2/18)/2] \approx 0.013$
- Ranking: $d_2 > d_1$

Notes on Query Likelihood Model

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

n-grams LMs

- 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

Summary

- **Three ways to model IR**
- VSM
  How query vector aligns with document vector?
- Probabilistic Model
  What is the relevance probability of document D given query Q?
- LM
  How likely is it possible to observe/generate sequence of terms Q in a language model of document D?

Resources

- Text book 1: Intro to IR, Chapter 12
- Text book 2: IR in Practice, Chapter 7.2, 7.3
- Readings:
  - *Robertson, Stephen E., et al.*
    ***"Okapi at TREC-3*.“**
    NIST Special Publication 109 (1995): 109.
  - *J. Ponte and W. B. Croft.*
    A language modeling approach to information retrieval.
    In Proceedings on the 21st annual international ACM SIGIR conference, pages 275–281, 1998
    https://dl.acm.org/doi/10.1145/290941.291008