

Information Storage and Retrieval

CSCE 670

Texas A&M University

Department of Computer Science & Engineering

Instructor: Prof. James Caverlee

Statistical Language Models

31 January 2017

Three “classic” approaches to IR

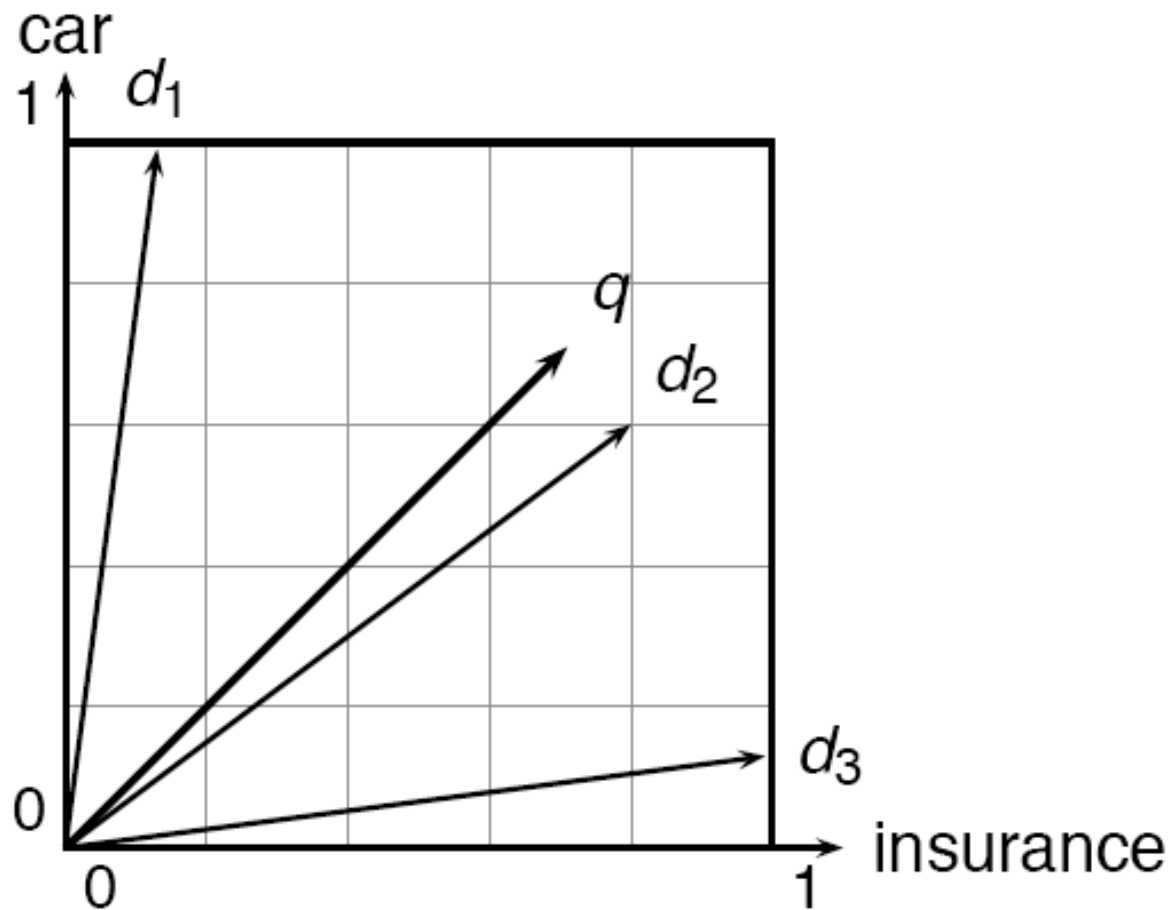
Recall: Boolean Retrieval

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Brutus AND Caesar but NOT Calpurnia

I if **play** contains **word**, 0 otherwise

Recall: Vector Space Retrieval

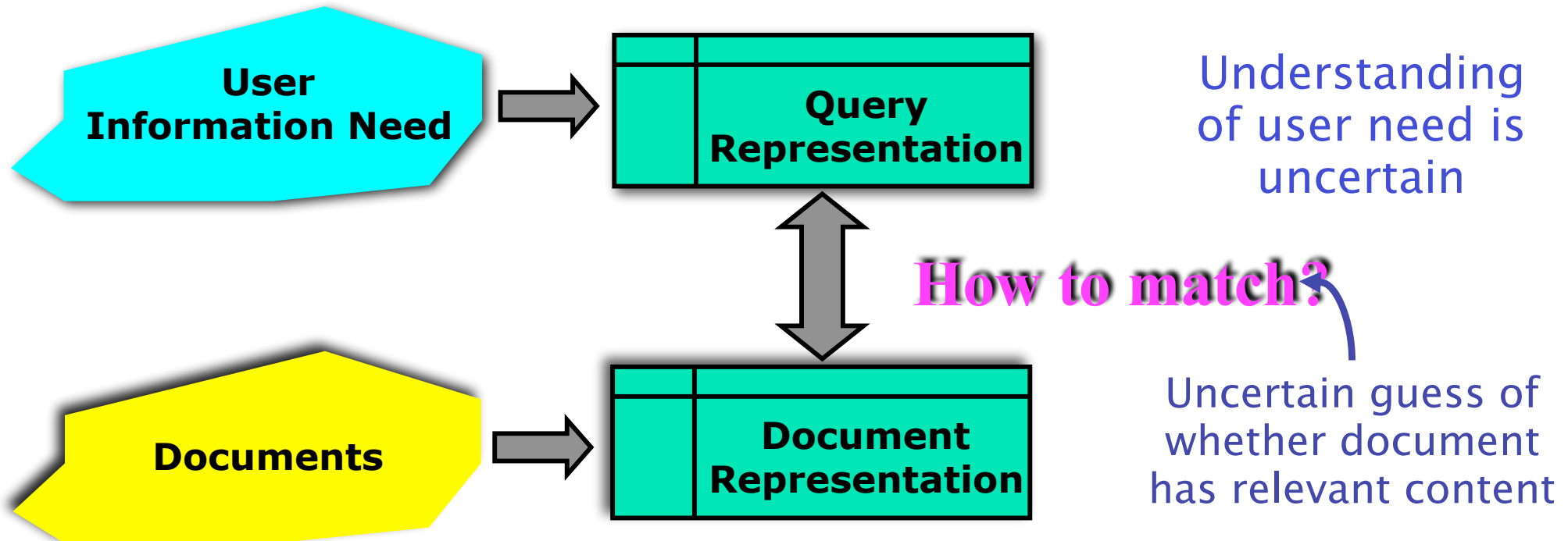


$$\text{sim}(d_j, d_k) = \frac{\vec{d}_j \cdot \vec{d}_k}{\|\vec{d}_j\| \|\vec{d}_k\|} = \frac{\sum_{i=1}^n w_{i,j} w_{i,k}}{\sqrt{\sum_{i=1}^n w_{i,j}^2} \sqrt{\sum_{i=1}^n w_{i,k}^2}}$$

Probabilistic IR

- Chapter 11
 - Traditional Probabilistic IR model
 - *Traditionally: neat ideas, but they've never won on performance.*
- Chapter 12
 - Statistical Language Models
 - Very hot right now

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.
Can we use probabilities to quantify our uncertainties?

But first ...

Probability review

Independent events

Let a , b be two events, with probability $P(a)$ and $P(b)$.

The events a and b are independent if and only if:

$$P(a \cap b) = P(a)P(b)$$

In general, a_1, a_2, \dots, a_n are independent if and only if:

$$P(a_1 \cap a_2 \cap \dots \cap a_n) = P(a_1)P(a_2)\dots P(a_n)$$

Probability review

Let a , b be two events, with probability $P(a)$ and $P(b)$.

Conditional probability

$P(a \mid b)$ is the probability of a given b , also called the conditional probability of a given b .

Conditional independence

The events a_1, \dots, a_n are conditionally independent if and only if:

$$P(a_i \mid a_j) = P(a_i) \text{ for all } i \text{ and } j.$$

Example

Independent

a and b are the results of throwing two dice

$$P(a=5 \mid b=3) = P(a=5) = 1/6$$

Not independent

a and b are the results of throwing two dice

t is the sum of the two dice

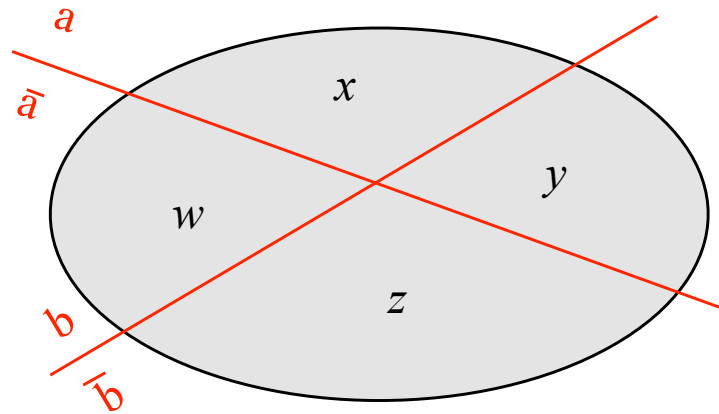
$$t = a + b$$

$$P(t=8 \mid a=2) = 1/6$$

$$P(t=8 \mid a=1) = 0$$

Example

where \bar{a} is
the event *not*
 a



$$P(a) = x + y$$

$$P(b) = w + x$$

$$P(a \mid b) = x / (w + x)$$

$$P(a \mid b) P(b) = P(a \cap b) = P(b \mid a) P(a)$$

Bayes theorem

Notation

Let a , b be two events.

$P(a \mid b)$ is the probability of a given b

Bayes Theorem

$$P(a \mid b) = \frac{P(b \mid a) P(a)}{P(b)}$$

Derivation

$$P(a \mid b) P(b) = P(a \cap b) = P(b \mid a) P(a)$$

Bayes theorem

Terminology used with Bayes Theorem

$$P(a | b) = \frac{P(b | a) P(a)}{P(b)}$$

$P(a)$ is called the **prior** probability of a

$P(a | b)$ is called the **posterior** probability of a given b

Example of Bayes theorem

Example

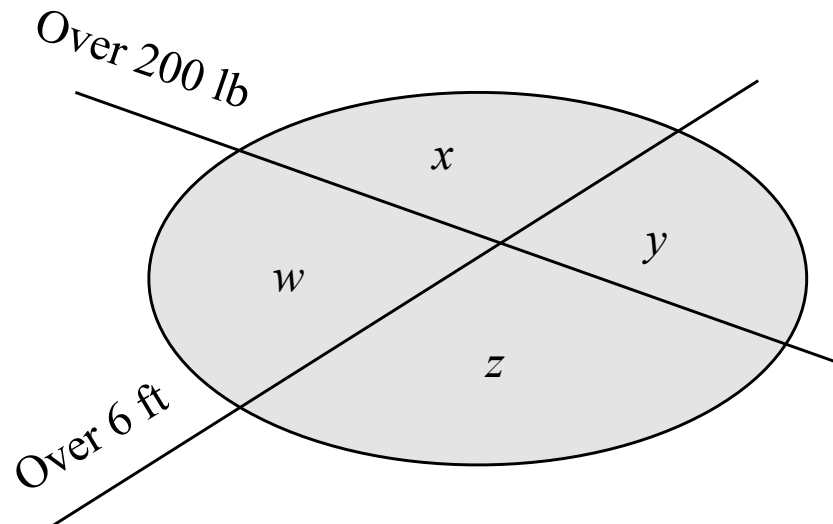
a Weight over 200 lb.

b Height over 6 ft.

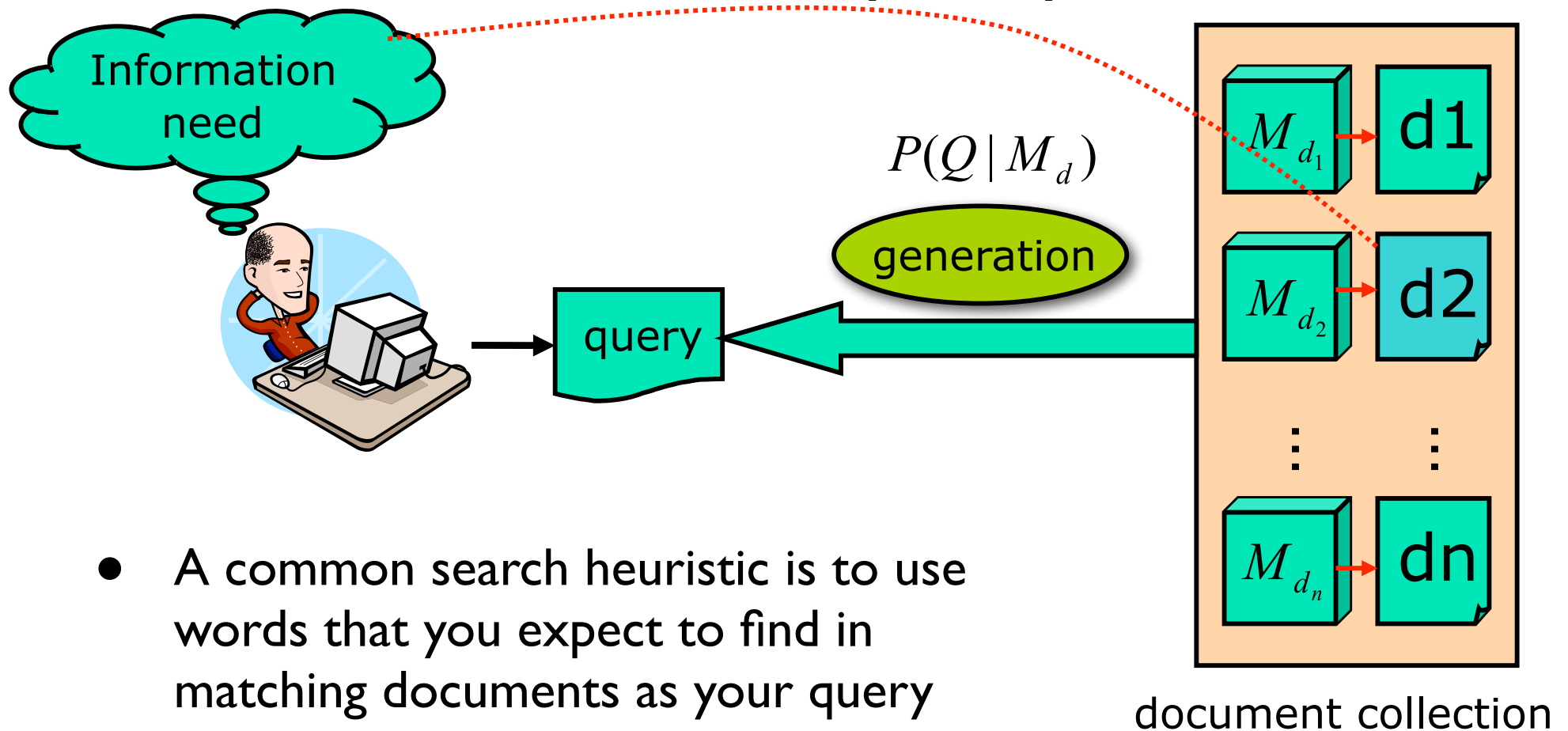
$$P(a \mid b) = x / (w+x) = x / P(b)$$

$$P(b \mid a) = x / (x+y) = x / P(a)$$

x is $P(a \cap b)$



IR based on Language Model (LM)

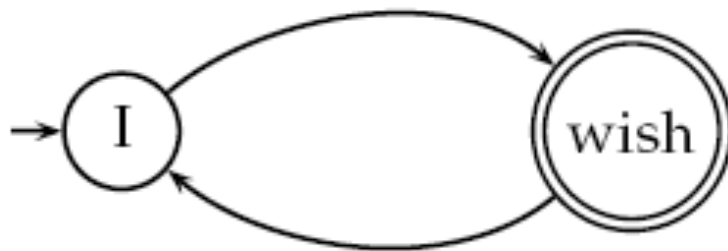


- A common search heuristic is to use words that you expect to find in matching documents as your query
- The LM approach directly exploits that idea!

Formal Language (Model)

- Traditional generative model: generates strings
 - Finite state machines or regular grammars, etc.

- Example:



I wish
I wish I wish
I wish I wish I wish
I wish I wish I wish I wish
...

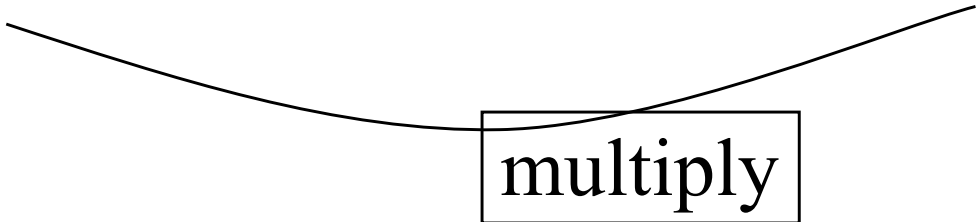
*wish I wish

Stochastic Language Models

- Models *probability* of generating strings in the language (commonly all strings over alphabet Σ)

Model M

0.2	the	<u>the</u>	<u>man</u>	<u>likes</u>	<u>the</u>	<u>woman</u>
0.1	a					
0.01	man	0.2	0.01	0.02	0.2	0.01
0.01	woman					
0.03	said					
0.02	likes					
...						

multiply

$P(s \mid M) =$
0.00000008

Stochastic Language Models

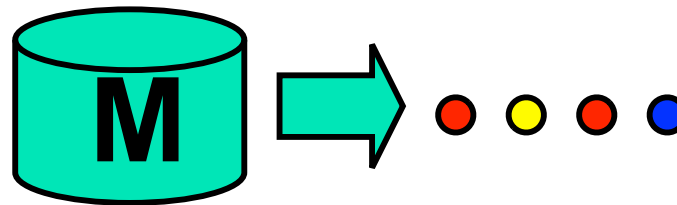
- Model *probability* of generating any string

Model M1	Model M2					
0.2 the	0.2 the					
0.01 class	0.0001 class	the	class	pleaseth	yon	maiden
0.0001 sayst	0.03 sayst					
0.0001 pleaseth	0.02 pleaseth	0.2	0.01	0.0001	0.0001	0.0005
0.0001 yon	0.1 yon	0.2	0.0001	0.02	0.1	0.01
0.0005 maiden	0.01 maiden					
0.01 woman	0.0001 woman					

$$P(s|M2) > P(s|M1)$$

Stochastic Language Models

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



$$\begin{aligned} P(\text{red yellow red blue} \mid M) &= P(\text{red} \mid M) \\ &\quad P(\text{yellow} \mid M, \text{red}) \\ &\quad P(\text{red} \mid M, \text{red yellow}) \\ &\quad P(\text{blue} \mid M, \text{red yellow red}) \end{aligned}$$

Language Models for IR

- Language Modeling Approaches
 - Attempt to model query generation process
 - Documents are ranked by the probability that a query would be observed as a random sample from the respective document model

Retrieval based on probabilistic LM

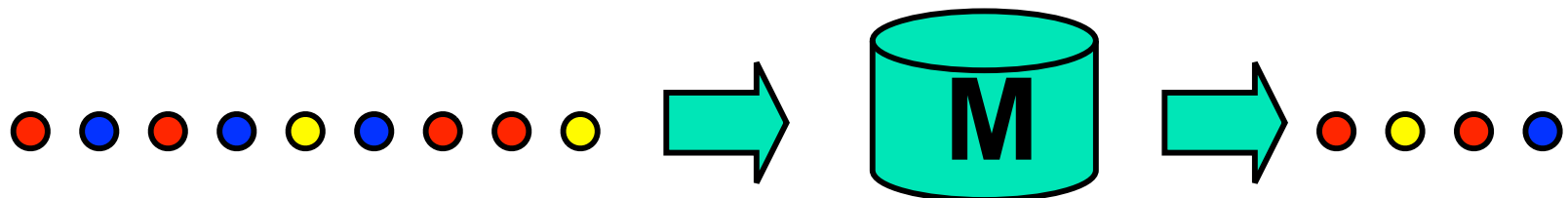
- Intuition
 - Users ...
 - Have a reasonable idea of terms that are likely to occur in documents of interest.
 - They will choose query terms that distinguish these documents from others in the collection.
- Collection statistics ...
 - Are integral parts of the language model.
 - Are not used heuristically as in many other approaches.
 - In theory. In practice, there's usually some wiggle room for empirically set parameters

The fundamental problem of LMs

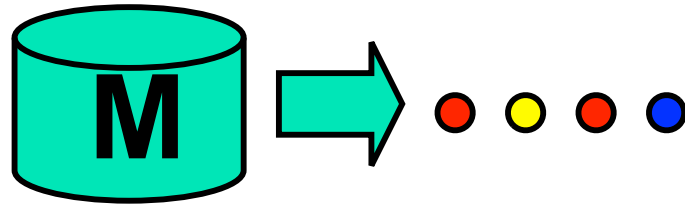
- Usually we don't know the model **M**
 - But have a sample of text representative of that model

$$P(\text{red yellow red blue} \mid M(\text{red blue red blue yellow blue red red yellow}))$$

- Estimate a language model from a sample
- Then compute the observation probability

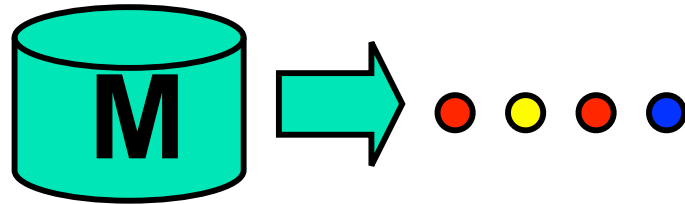


Example



- Doc 1 = “Today is a beautiful day.”
- $p(\text{today} \mid M1) =$
- $p(\text{is} \mid M1) =$
- $p(a \mid M1) =$
- $p(\text{beautiful} \mid M1) =$

Example



- Doc 2 = “Beautiful beautiful beautiful day!”
- $p(\text{today} \mid M2) =$
- $p(\text{is} \mid M2) =$
- $p(a \mid M2) =$
- $p(\text{beautiful} \mid M2) =$

Query generation probability

- Rank docs according to:

$$\hat{P}(q|M_d)$$

- The probability of producing the query given the language model of document d using MLE is:

$$\hat{P}(q|M_d) = \prod_{t \in q} \hat{P}_{\text{mle}}(t|M_d) = \prod_{t \in q} \frac{\text{tf}_{t,d}}{L_d}$$

M_d : language model of document d

$\text{tf}_{t,d}$: raw tf of term t in document d

L_d : total number of tokens in document d

MLE

- Maximum Likelihood Parameter Estimation
- We have some sample data
- We have an idea about how to model the process that generated the data
- What parameters θ should we pick to guide the generative process?
- The ones that maximize the probability (likelihood) of the sample data

MLE (example)

- We have a coin; could be fair, maybe not
- Toss it 100 times
 - That's 100 sample data values (call it Z)
- We get 90 heads, 10 tails
- We model the data generation process as a series of Bernoulli trials
 - Each outcome is independent Heads/Tails
- What is the probability of tossing heads?
 - p , or $\theta \leftarrow$ What is the most likely p ?

$$L(\theta;Z) = \Pr(\text{HTHHHH} \dots | p) =$$

$$p^{z_1} (1-p)^{1-z_1} * \dots * p^{z_n} (1-p)^{1-z_n}$$

where $z = 1$ (heads) or $z = 0$ (tails)

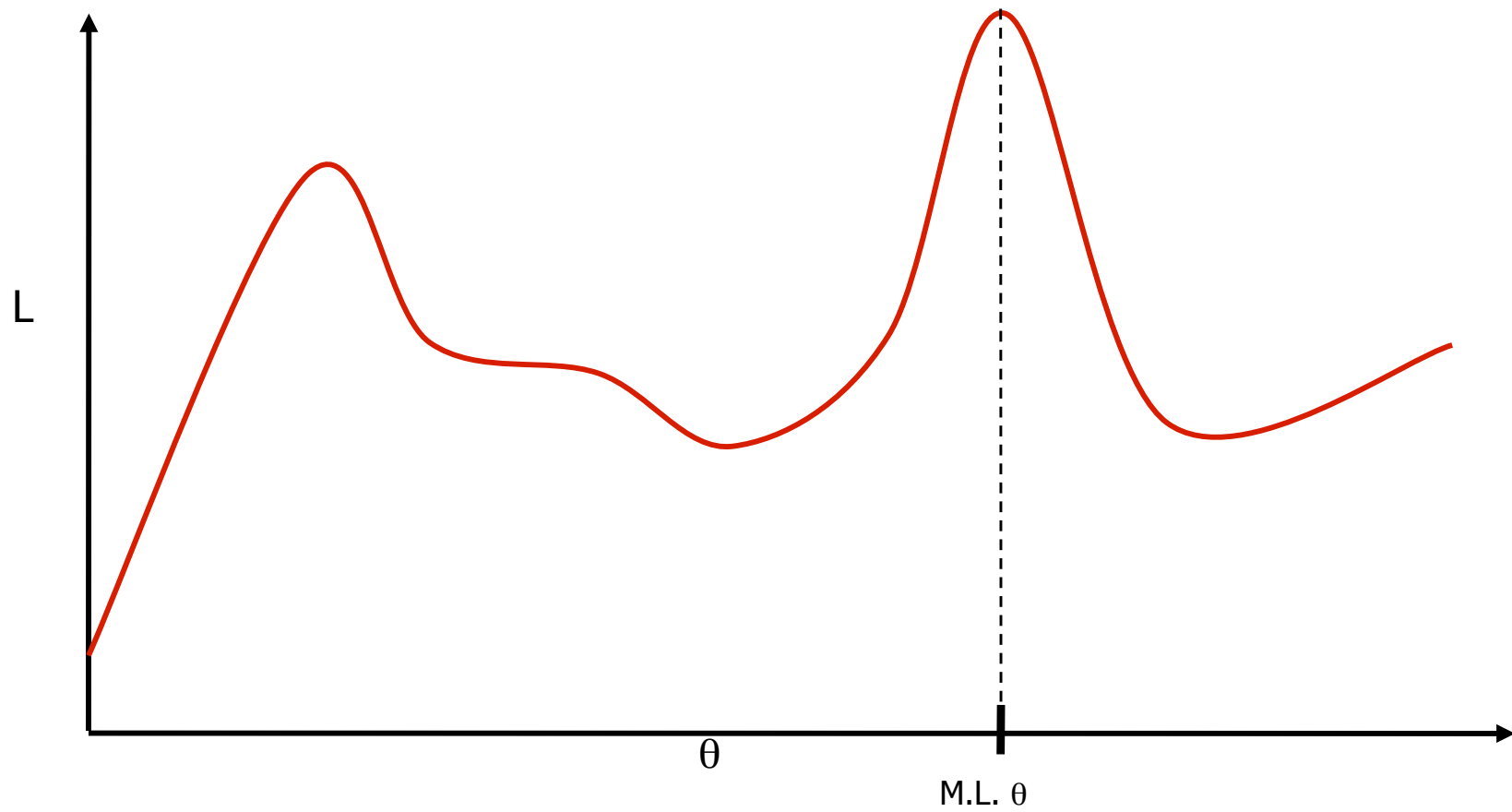
Take Logs

$$LL(\theta;Z) = \sum z_i \ln(p) + (n - \sum z_i) \ln(1 - p)$$

Take Derivative wrt p and Set to 0

$$\frac{\sum z_i}{p} - \frac{n - \sum z_i}{1 - p} = 0 \quad \Rightarrow \quad p^* = \frac{\sum z_i}{n} = \frac{90}{100}$$

- So if we toss a coin 100 times and see 90 heads, the MLE for the coin is that it is a biased coin with $\text{Pr}(\text{Heads}) = 0.9$



Example

$$\hat{P}(q|M_d) = \prod_{t \in q} \hat{P}_{\text{mle}}(t|M_d) = \prod_{t \in q} \frac{\text{tf}_{t,d}}{L_d}$$

- Doc 1 = “Today is a beautiful day.”
- Doc 2 = “Beautiful beautiful beautiful day!”
- Query = “beautiful”

Insufficient data

- Zero probability $\hat{P}(t|M_d) = 0$
- May not wish to assign a probability of zero to a document that is missing one or more of the query terms [gives conjunction semantics]
- General approach
 - A non-occurring term is possible, but no more likely than would be expected by chance in the collection.
 - If $tf_{(t,d)} = 0$, $\hat{P}(t|M_d) \leq cf_t/T$

cf_t : raw count of term t in the collection

T : raw collection size (total number of tokens in the collection)

Insufficient data

- Zero probabilities spell disaster
 - We need to smooth probabilities
 - Discount nonzero probabilities
 - Give some probability mass to unseen things
- There's a wide space of approaches to smoothing probability distributions to deal with this problem, such as adding 1, $\frac{1}{2}$ or ϵ to counts, Dirichlet priors, discounting, and interpolation
- A simple idea that works well in practice is to use a mixture between the document multinomial and the collection multinomial distribution

Mixture model

$$\hat{P}(t|d) = \lambda \hat{P}_{\text{mle}}(t|M_d) + (1 - \lambda) \hat{P}_{\text{mle}}(t|M_c)$$

- Mixes the probability from the document with the general collection frequency of the word.
- Correctly setting λ is very important
- A high value of lambda makes the search “conjunctive-like” – suitable for short queries
- A low value is more suitable for long queries
- Can tune λ to optimize performance
 - Perhaps make it dependent on document size

Basic mixture model summary

- General formulation of the LM for IR

$$P(d|q) \propto P(d) \prod_{t \in q} ((1 - \lambda)P(t|M_c) + \lambda P(t|M_d))$$

general language model



individual-document model

- The user has a document in mind, and generates the query from this document.
- The equation represents the probability that the document that the user had in mind was in fact this one.

Example

- Document collection (2 documents)
 - d_1 : Xerox reports a profit but revenue is down
 - d_2 : Lucent narrows quarter loss but revenue decreases further
- Model: MLE unigram from documents; $\lambda = 1/2$
- Query: *revenue down*
 - $P(Q|d_1) = [(1/8 + 2/16)/2] \times [(1/8 + 1/16)/2]$
 $= 1/8 \times 3/32 = 3/256$
 - $P(Q|d_2) = [(1/8 + 2/16)/2] \times [(0 + 1/16)/2]$
 $= 1/8 \times 1/32 = 1/256$
- Ranking: $d_1 > d_2$

Exercise

- Suppose, we've got 4 documents:

DocID	Document text
1	click go the shears boys click click click
2	click click
3	metal here
4	metal shears click here

- Using the mixture model with $\lambda = 0.5$, work out the per-doc probabilities for the query “click”

- collection model for “click” is $7/16$
- collection model for “shears” is $2/16$
- click in doc1: $0.5 * 1/2 + 0.5 * 7/16 = 0.4688$
- doc2: 0.7188
- doc3: 0.2188
- doc4: 0.3438

Exercise

DocID	Document text
1	click go the shears boys click click click
2	click click
3	metal here
4	metal shears click here

- For the query “click shears”, what’s the ranking of the four documents?

click shears

- **Doc 4: 0.0645**
- Doc 1: 0.0586
- Doc 2: 0.0449
- Doc 3: 0.0137

Summary: LM

- LM approach assumes that documents and expressions of information problems are of the same type
- Computationally tractable, intuitively appealing

Drawbacks

- Assumption of equivalence between document and information problem representation is unrealistic
- Very simple models of language
- Relevance feedback is difficult to integrate, as are user preferences, and other general issues of relevance
- Can't easily accommodate phrases, passages, Boolean operators
- Current extensions focus on putting relevance back into the model, etc.

Language models: pro & con

- Novel way of looking at the problem of text retrieval based on probabilistic language modeling
 - Conceptually simple and explanatory
 - Formal mathematical model
 - Natural use of collection statistics, not heuristics (almost...)
- LMs provide effective retrieval and can be improved to the extent that the following conditions can be met
 - Our language models are accurate representations of the data.
 - Users have some sense of term distribution.

Comparison With Vector Space

- There's some relation to traditional tf.idf models:
 - (unscaled) term frequency is directly in model
 - the probabilities do length normalization of term frequencies
 - the effect of doing a mixture with overall collection frequencies is a little like idf: terms rare in the general collection but common in some documents will have a greater influence on the ranking

Comparison With Vector Space

- Similar in some ways
 - Term weights based on frequency
 - Terms often used as if they were independent
 - Inverse document/collection frequency used
 - Some form of length normalization useful
- Different in others
 - Based on probability rather than similarity
 - Intuitions are probabilistic rather than geometric
 - Details of use of document length and term, document, and collection frequency differ