Probabilistic Information Retrieval Models

- Lecture 4
  SI 650 / EECS 549
  Information Retrieval

- Sept. 17, 2025

Some slides based on slide deck by Dr. Jurgens

Administrative Notes

- Homework 1 due in two week!
  - Please try to go end-to-end first *before* implementing all of one part—get the IR system running!
- Project proposals due in three weeks! Attend discussion sections for more insights on this.

Today's Goals

- Think about retrieval beyond TF-IDF and VS Models
- Learn a bit more about probabilities

The Notion of Relevance

- Relevance
  - Δ(Rep(q), Rep(d)) — **Similarity**
    - Different rep & similarity
      - Vector space model (Salton et al., 75)
      - . . .
      - Prob. distr. model (Wong & Yao, 89)
  - $P(r=1|q,d)$ $r \in \{0,1\}$ — **Probability of Relevance**
    - Regression Model (Fox 83)
    - Generative Model
      - Doc generation
        - Classical prob. Model (Robertson & Sparck Jones, 76)
      - Query generation
        - LM approach (Ponte & Croft, 98) (Lafferty & Zhai, 01a)
  - $P(d \rightarrow q)$ or $P(q \rightarrow d)$ — **Probabilistic inference**
    - Different inference system
      - Prob. concept space model (Wong & Yao, 95)
      - Inference network model (Turtle & Croft, 91)

Prob/Statistics & Text Management

- Probability & statistics provide a principled way to quantify the uncertainties associated with natural language

- Allow us to answer questions like:
  - Given that we observe “baseball” three times and “game” once in a news article, how likely is it about “sports”?

    (text categorization, information retrieval)
  - Given that a user is interested in sports news, how likely would the user use “baseball” in a query?

    (information retrieval)

Things we will use in the course

- Events
- Probability, joint probability, conditional probability
- Distributions
- Bayes Rule
- Basic Statistics
- Parameter estimation
- Maximum likelihood estimation

Revisit: Essential Probability & Statistics

Random variable

- A variable that can take values within a fixed set (discrete) or within some range (continuous).

$$X \in \{1, 2, 3, 4, 5, 6\}$$

$$X \in \{the, a, dog, cat, runs, to, store\}$$

Example: Flip a Coin

- Sample space: {Head, Tail}
- Fair coin:
  - p(H) = 0.5, p(T) = 0.5
- Unfair coin, e.g.:
  - p(H) = 0.3, p(T) = 0.7
- Flipping two fair coins:
  - Sample space: {HH, HT, TH, TT}
- Modeling text:
  - Flip a coin to decide whether or not to include a word in a document
  - Sample space = {appear, absence}

Example: Toss a Dice

- Sample space: S = {1,2,3,4,5,6}
- Fair dice:
  - p(1) = p(2) = p(3) = p(4) = p(5) = p(6) = 1/6
- Unfair dice: p(1) = 0.3, p(2) = 0.2, ...
- N-dimensional dice:
  - S = {1, 2, 3, 4, …, N}
- Modeling text:
  - Toss a die to decide which word to write in the next position
  - Sample space = {cat, dog, tiger, …}

What if the Dice has More Faces?

- Suitable to represent documents
- Every face corresponds to a word in vocabulary
- The author toss a dice
  to write a word
- Apparently, an unfair dice

Maximum Likelihood Estimate

- Data: a document d with counts c(w1), …, c(wN), and length |d|
- Model: multinomial distribution M with parameters {p(wi)}

$$p(w_i) = \frac{c(w_i)}{d}$$

- Likelihood: p(d|M)
- Maximum likelihood estimator: M=argmax M p(d|M)

Problem: a document is short
if $c(w_i) = 0$, does this mean $p(w_i) = 0$?

Revisit: Basic Concept of Probability

- Joint probability: P(AB) , also written as P(A, B)
- Conditional Probability: P(B|A) = P(A, B)/P(A)
  - P(A, B) = P(A)P(B|A) = P(B)P(A|B)
  - For independent events, P(A, B) = P(A)P(B), and P(A|B) = P(A), P(B|A) = P(B).
- Bayes' Rule:
  - P(A|B) = P(B|A)P(A)/P(B)
  - P(H|E) = P(E|H)P(H)/P(E)
  - P(C|D) = P(D|C)P(C)/P(D)

H: hypothesis; E: Evidence

In text classification: C: class label; D: data (document

Getting to Statistics ...

- We are flipping an unfair coin, but P(Head)=? (parameter estimation)
  - If we see the results of a huge number of random experiments, then $$\hat{P}(Head) = \frac{count(Heads)}{count(Flips)} = \frac{count(Heads)}{count(Heads) + count(Tails)}$$
  - But, what if we only see a small sample (e.g., 2)? Is this estimate still reliable? We flip twice and got two tails, does it mean P(Head) = 0?
- In general, statistics has to do with drawing conclusions on the whole population based on observations of a sample (data)

Parameter Estimation of a Random Variable

- General setting:
  - Given a (hypothesized & probabilistic) model that governs the random experiment
  - The model gives a probability of any data $p(D|\theta)$ that depends on the parameter θ
  - Now, given actual sample data $X=\{x_1,\ldots,x_n\}$, what can we say about the value of θ?
- Intuitively, take your best guess of θ – "best" means "best explaining/fitting the data"
- Generally an optimization problem

$$P(X = x)$$

Probability that the random variable *X* takes the value *x* (e.g., 1)

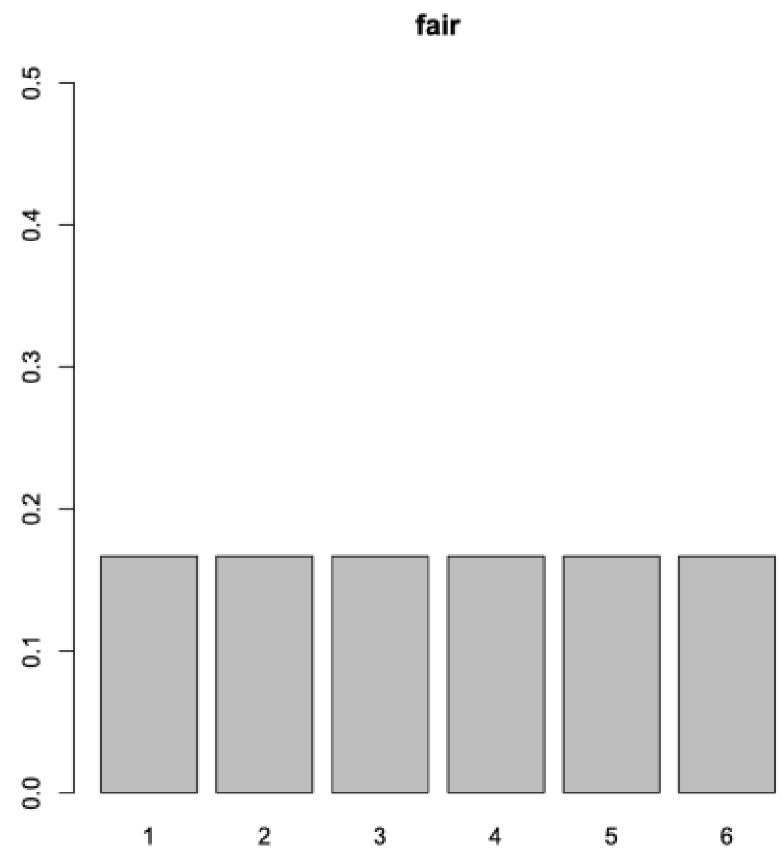
$$X \in \{1, 2, 3, 4, 5, 6\}$$

Two conditions:

1. Between 0 and 1: $$0 \leq P(X = x) \leq 1$$
2. Sum of all probabilities = 1 $$\sum_x P(X = x) = 1$$

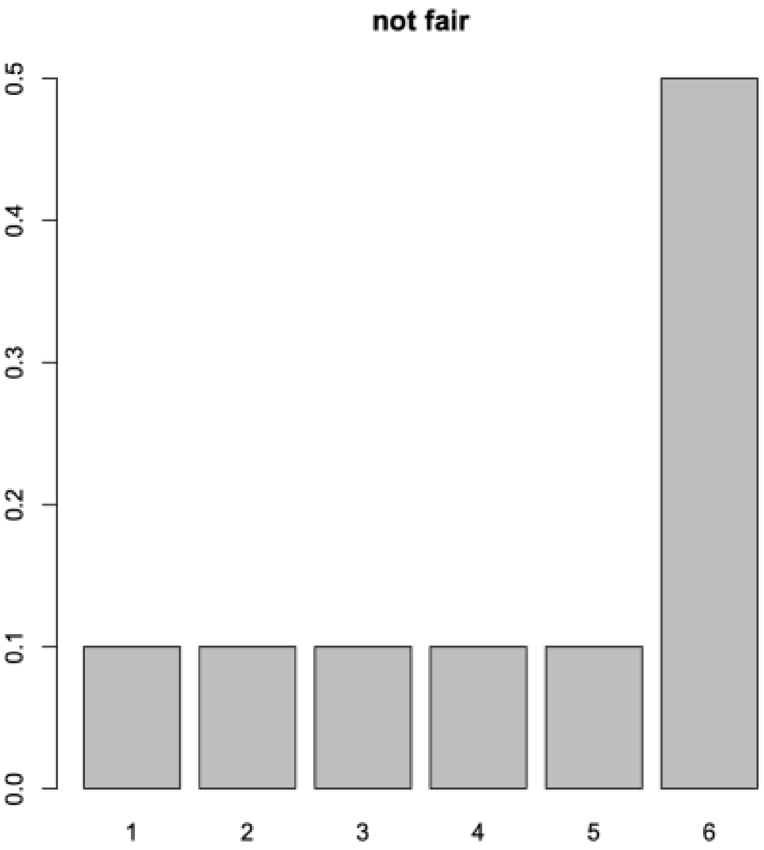
Fair dice

$$X \in \{1, 2, 3, 4, 5, 6\}$$



Weighted dice

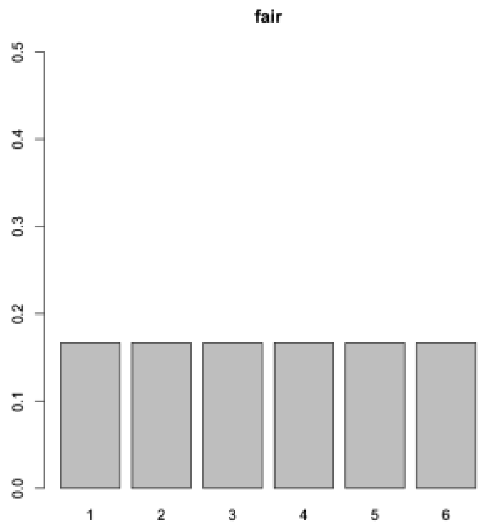
$$X \in \{1, 2, 3, 4, 5, 6\}$$



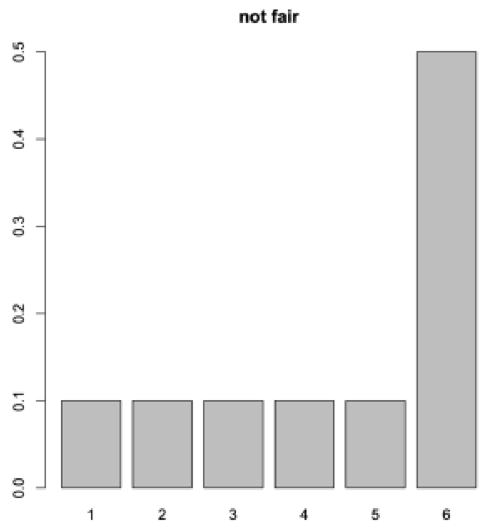
Inference

$$X \in \{1, 2, 3, 4, 5, 6\}$$

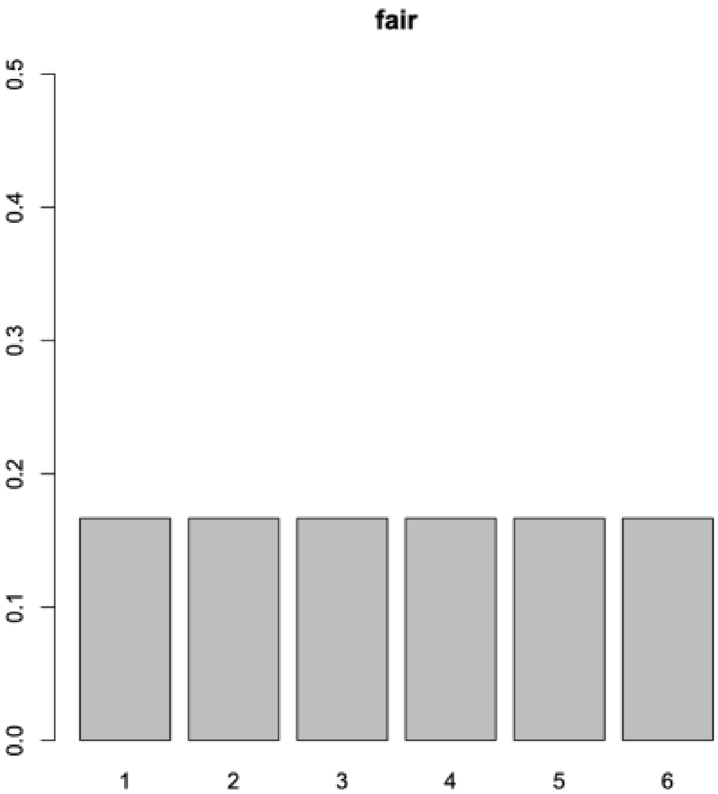
We want to *infer* the probability distribution that generated the data we see.

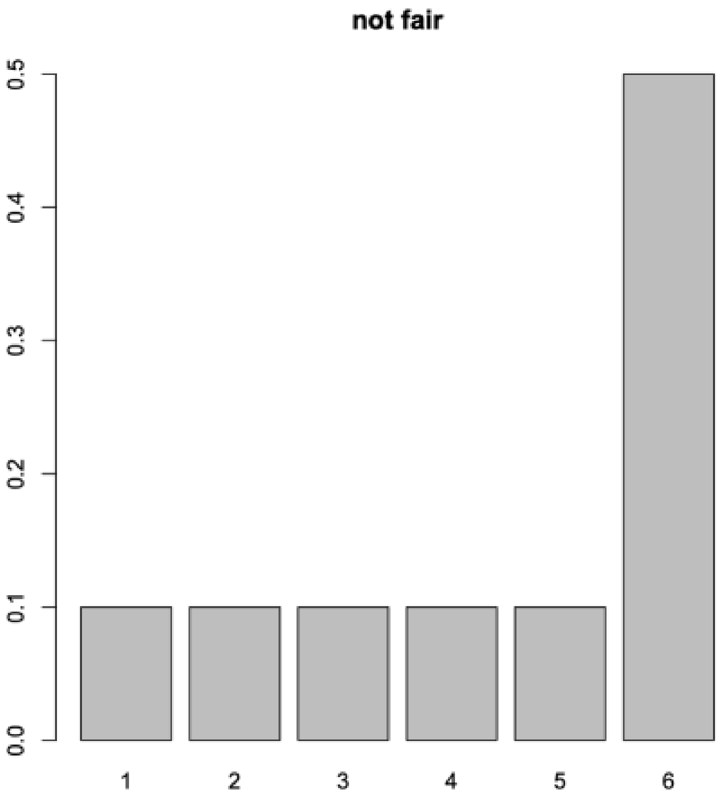


?

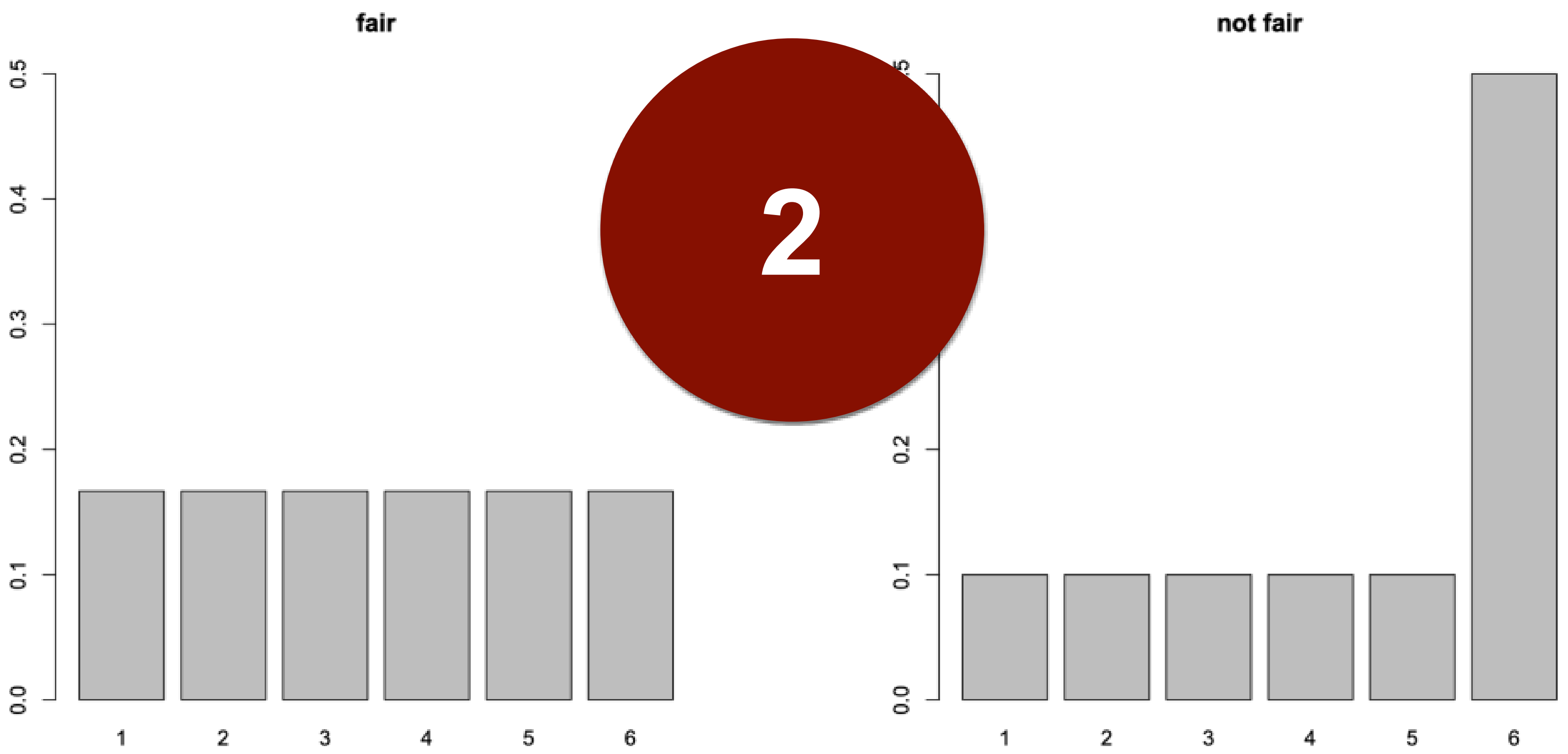


Probability



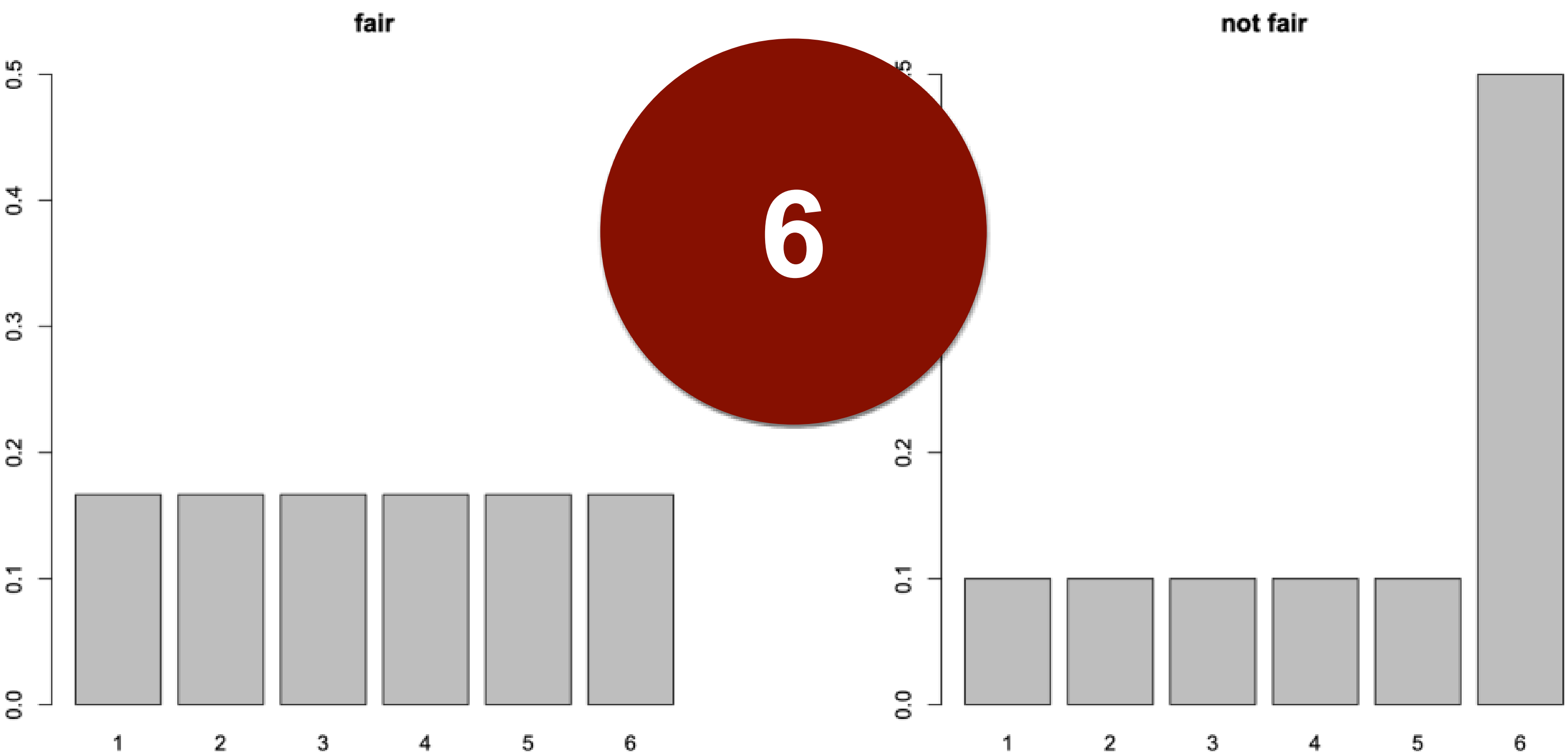


Probability

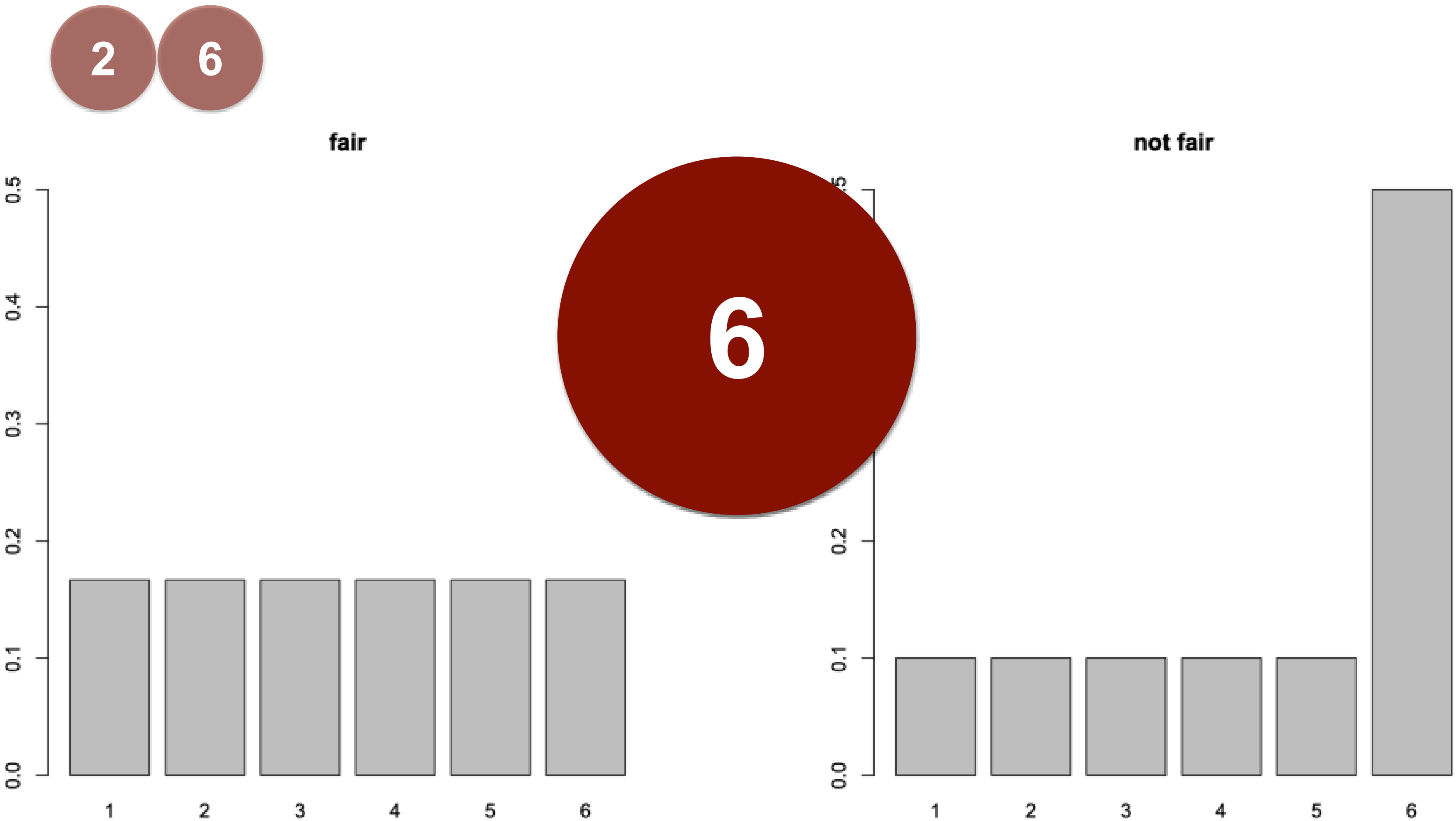


Probability

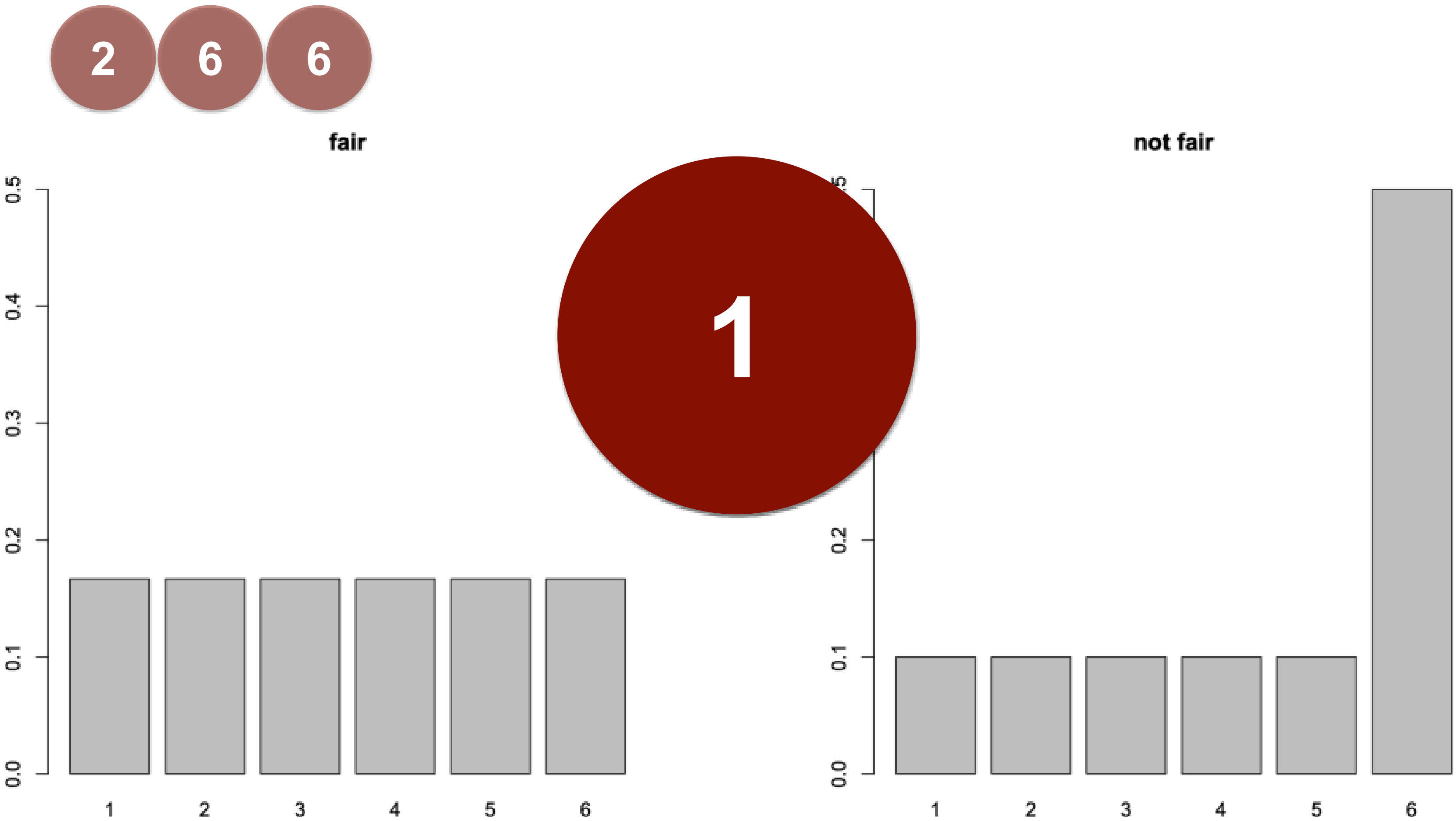
2



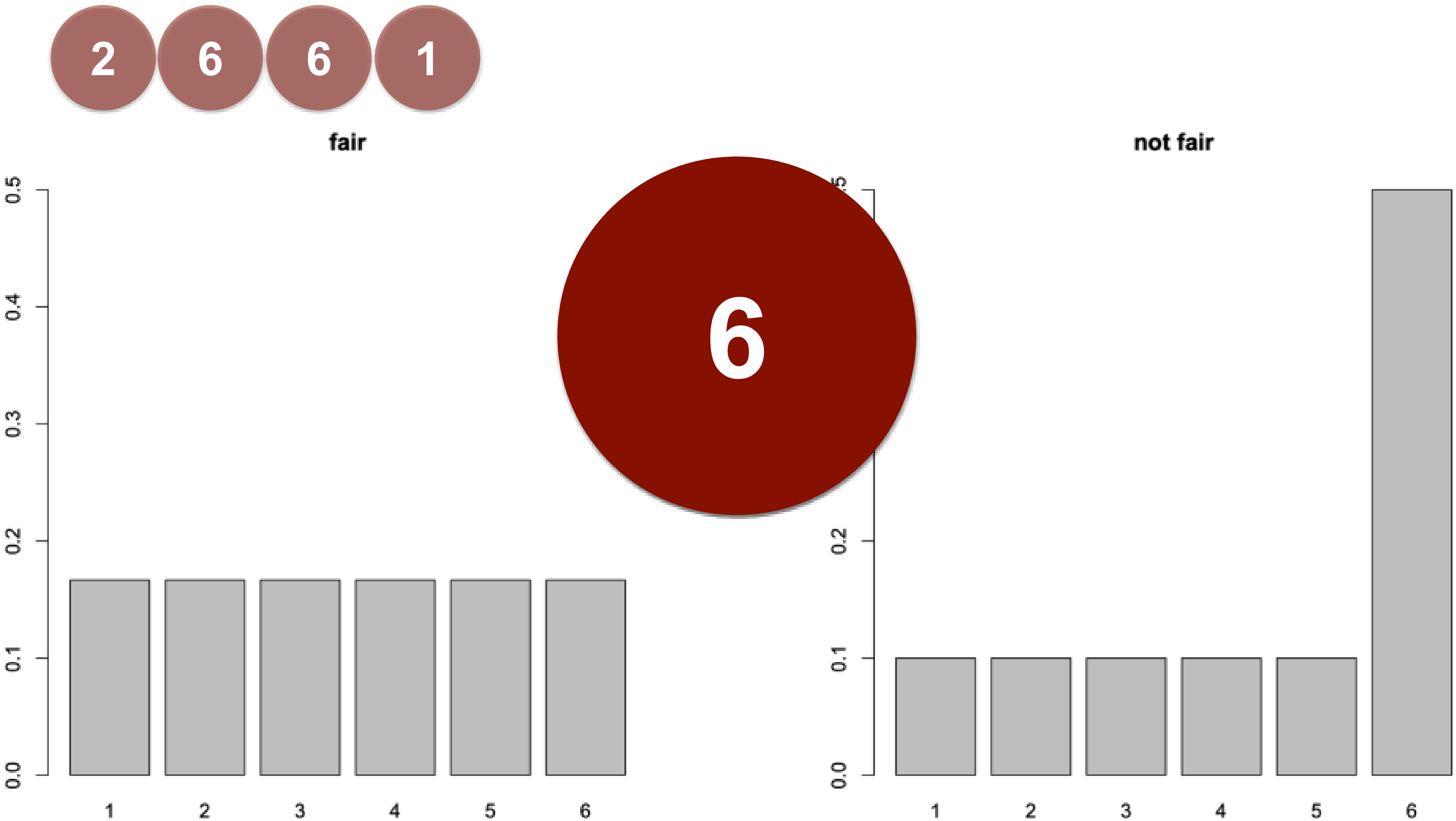
Probability



Probability

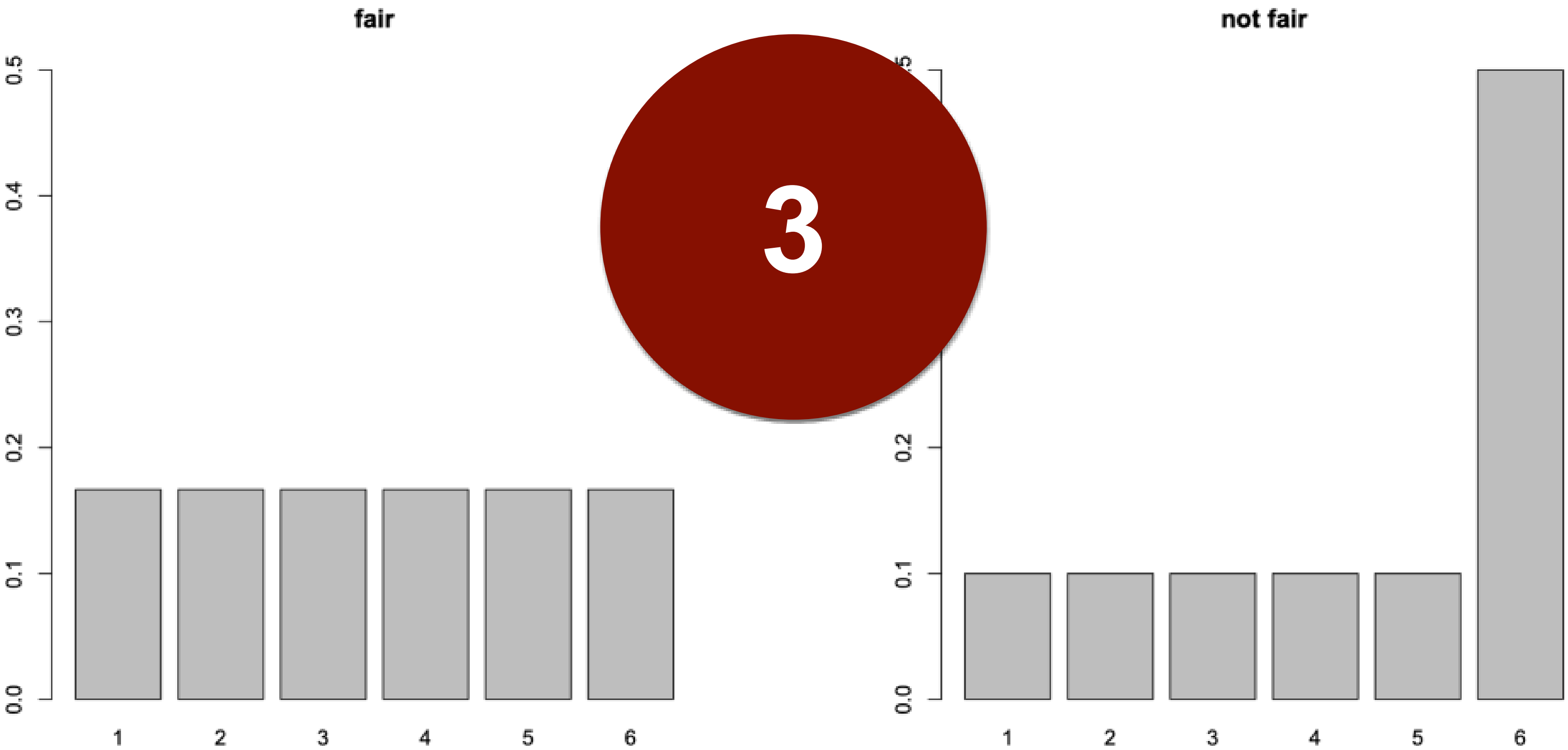


Probability

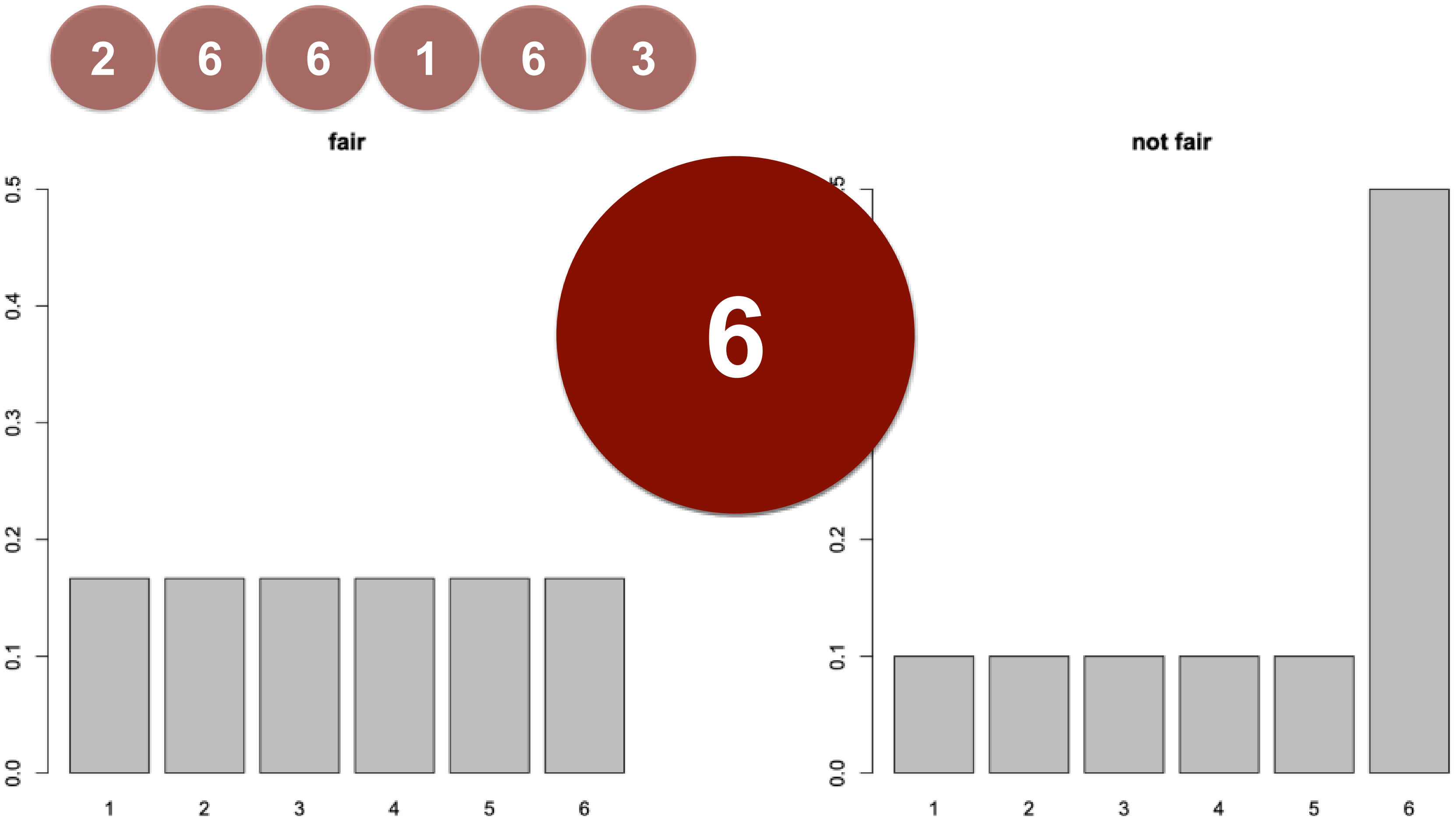


Probability

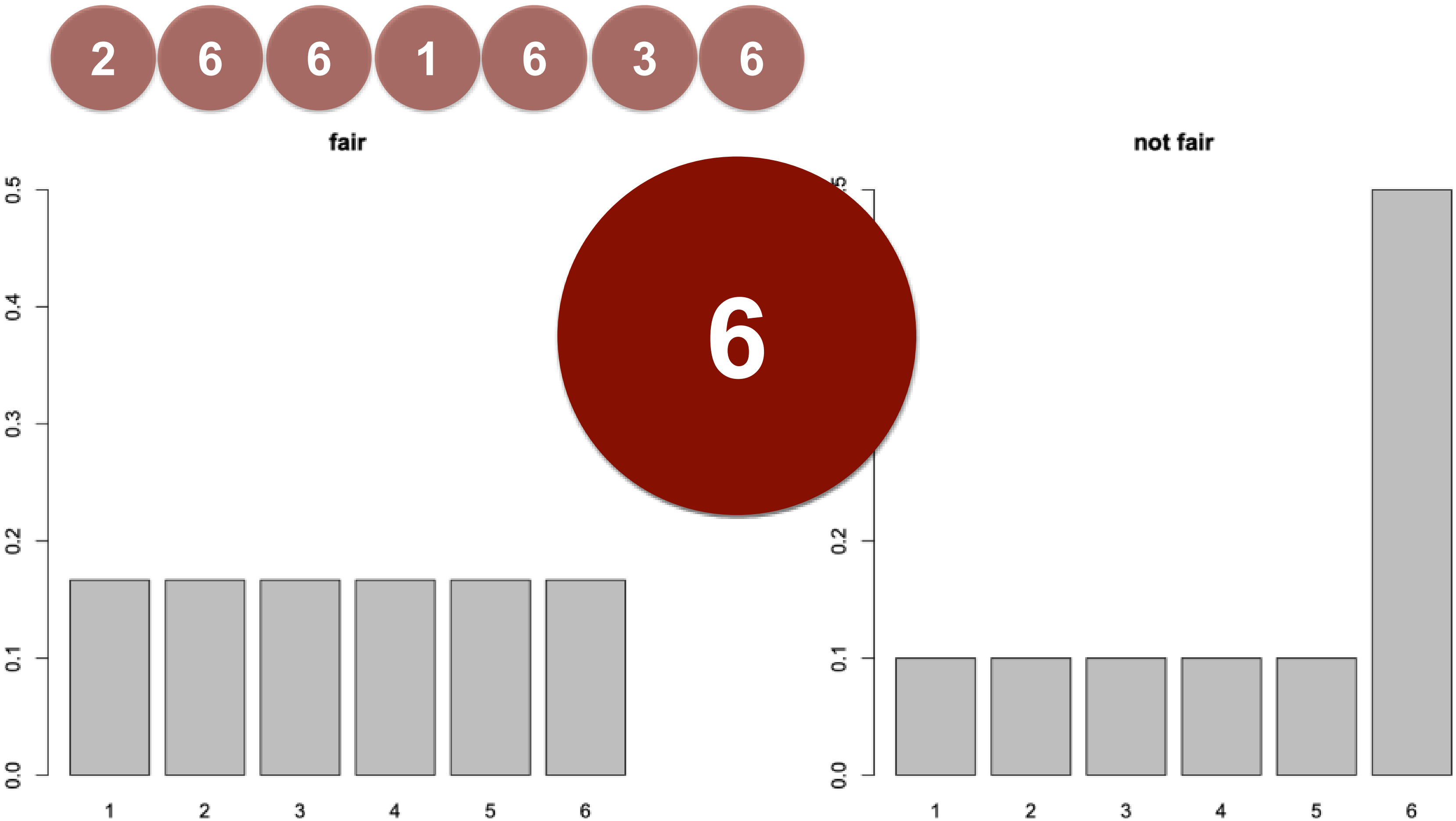
2 6 6 1 6



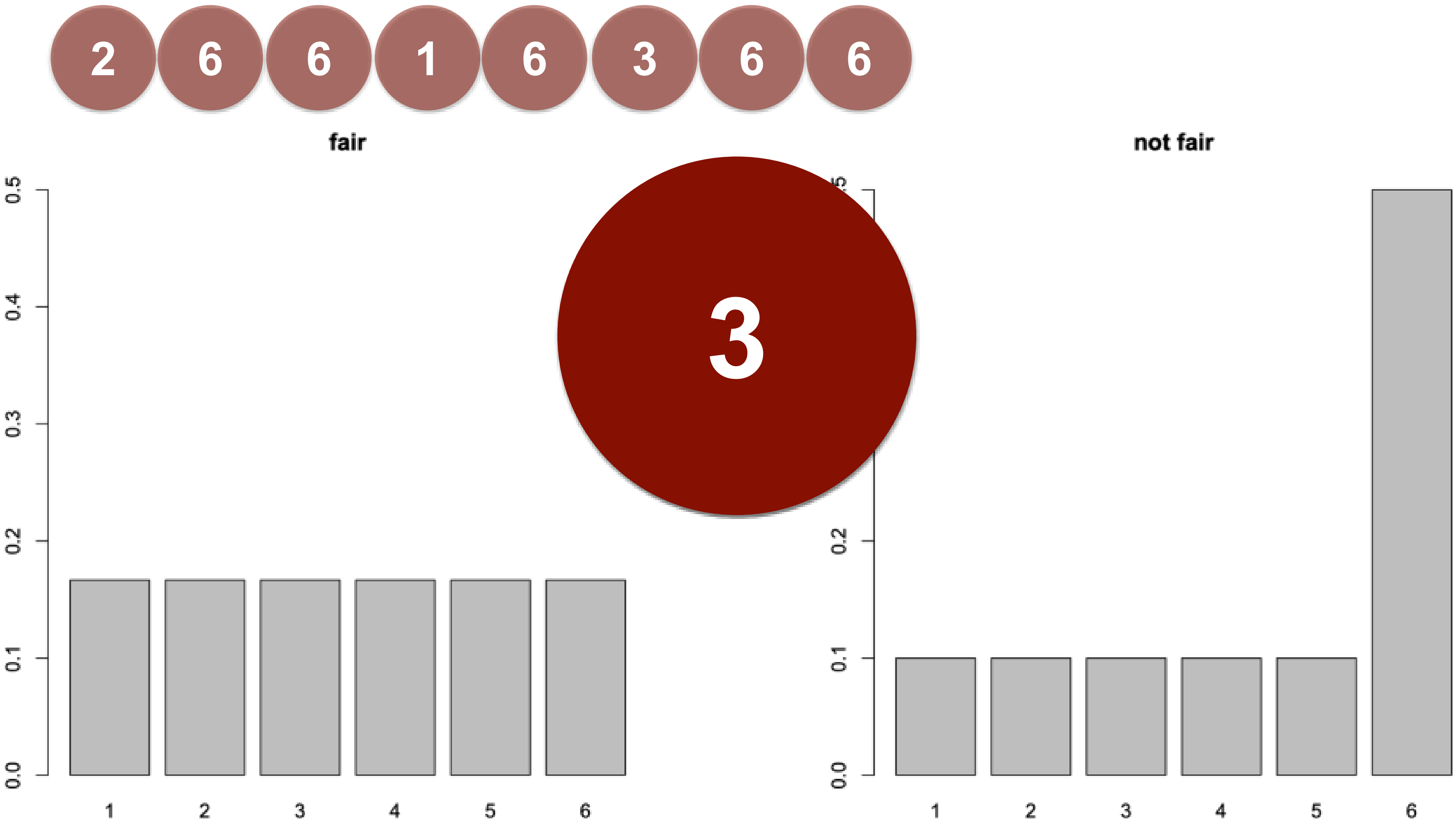
Probability



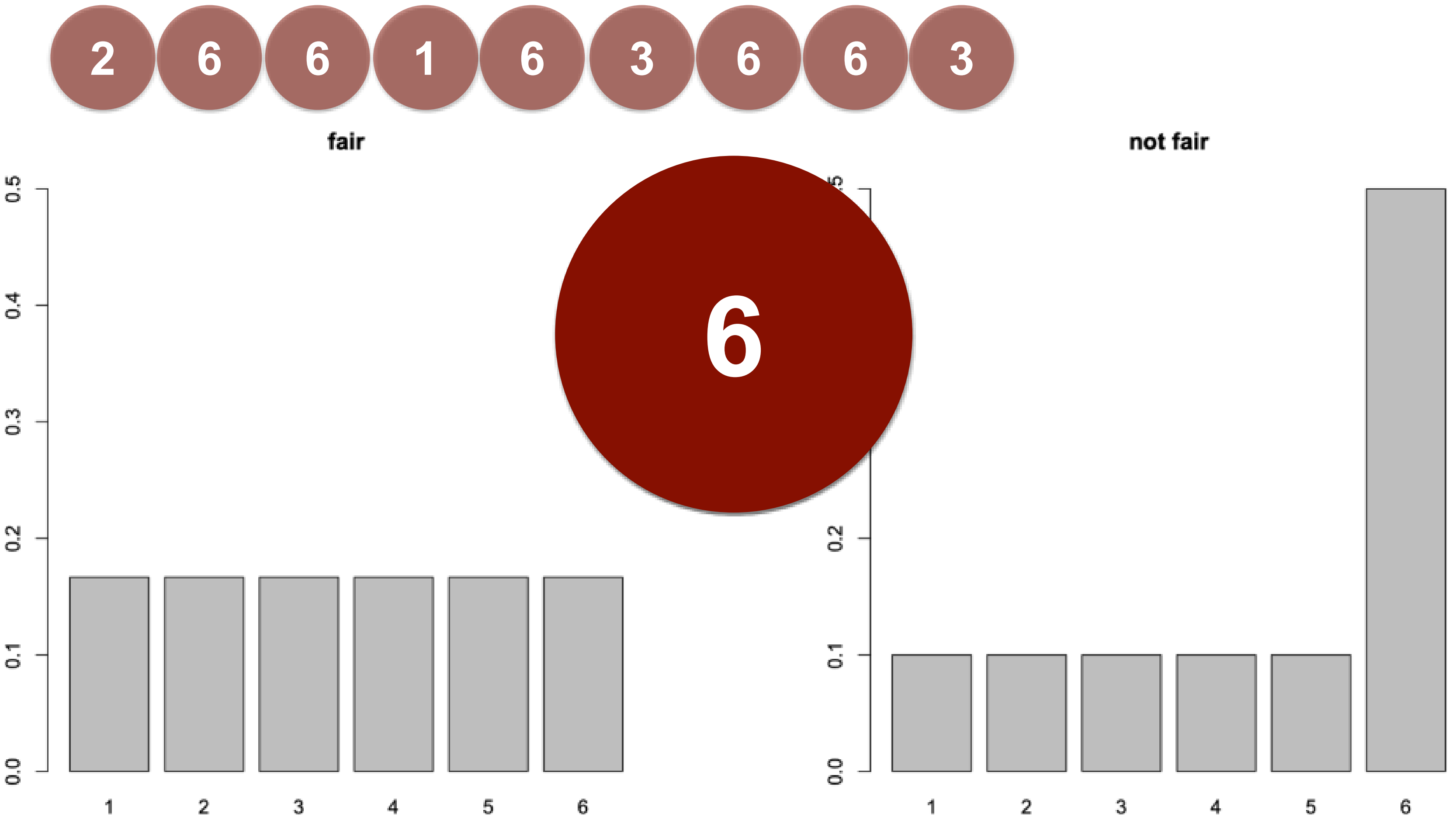
Probability



Probability

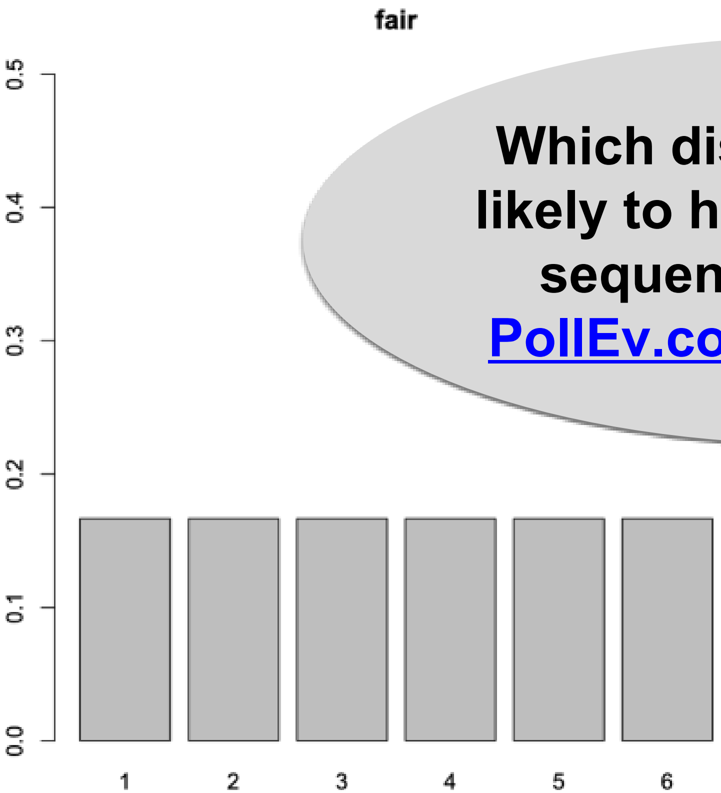


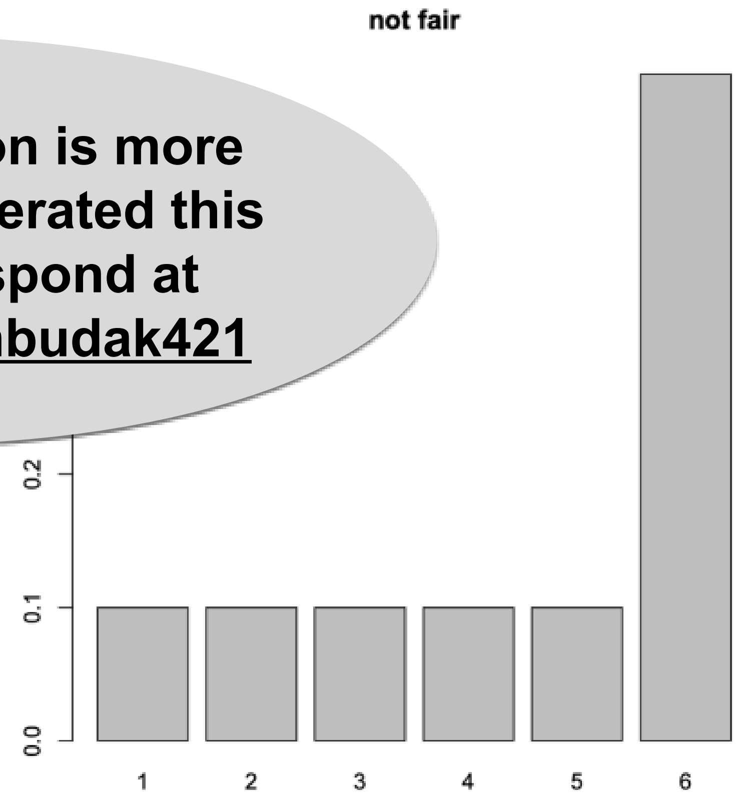
Probability



Probability

2 6 6 1 6 3 6 6 3 6





Which distribution is more likely to have generated this sequence? Respond at PollEv.com/cerenbudak421

Log in to Poll Everywhere

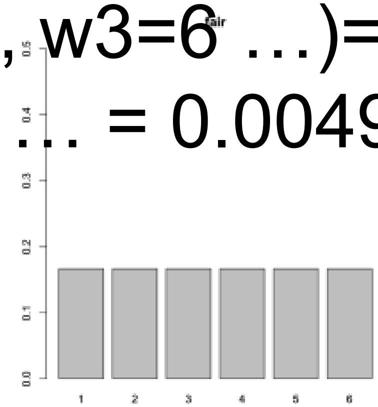
To present live activities, please log in to your Poll Everywhere account in a separate window.

Launch log-in window

Data Likelihood

- P(w1=2, w2=6, w3=6 …)=
  .17 x .17 x .17 … = 0.004913

P(2 6 6 |

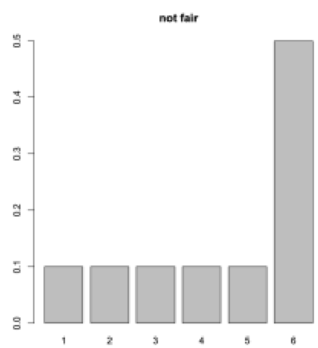


)

=.17 x .17 x .17
= 0.004913

- P(w1=2, w2=6,w3=6

P(2 6 6 |



)

= .1 x .5 x .5
= 0.025

Data Likelihood

- The likelihood gives us a way of discriminating between possible alternative parameters, but also a strategy for picking a single best* parameter (θ) among all possibilities

Word choice as weighted dice

Unigram probability

A document
with positive
reviews (or can
be a collection
for
classification)

A document
with negative
reviews
(or can be a
collection for
classification)

Maximum Likelihood Estimate

- This is a maximum likelihood estimate for P(X); the parameter values for which the data we observe (X) is most likely.

$$P(X = the) = \frac{\#the}{\#\text{total words}}$$

Conditional Probability

- Probability that one random variable takes a particular value *given* the fact that a different variable takes another

$$P(X = x|Y = y)$$

$$P(X_i = hate \mid Y = \oplus)$$

Maximum Likelihood vs. Bayesian

- Maximum likelihood estimation
  - "Best" means "data likelihood reaches maximum"

$$\hat{\theta} = \arg\max_{\theta} P(X|\theta)$$

  - Problem: small sample
- Bayesian estimation
  - "Best" means being consistent with our "prior" knowledge and explaining data well

  - Problem: how to define prior?

2. Maximum Likelihood (MLE)

- **目标**：选择能让观测数据概率最大的参数值 $\hat{\theta}$。
- **做法**：

$$\hat{\theta}_{MLE} = \arg\max_{\theta} P(\text{data}|\theta)$$

- **特点**：
  - 只依赖 **观测到的数据。**
  - 不考虑任何先验知识（prior）。
  - 通常给出一个点估计（single best estimate）。
  - 简单、直接，但容易过拟合。

3. Bayesian Estimation

- **目标**：求出参数的后验分布 $P(\theta|\text{data})$，然后用它做决策。
- **公式**：

$$P(\theta|\text{data}) = \frac{P(\text{data}|\theta)P(\theta)}{P(\text{data})}$$

  其中：
  - $P(\theta)$：先验概率（prior）
  - $P(\text{data}|\theta)$：似然（likelihood）
  - $P(\theta|\text{data})$：后验概率（posterior）
- **特点**：
  - 同时利用 **观测数据** 和 **先验知识。**
  - 得到的不是单一数值，而是整个概率分布。
  - 更鲁棒，尤其是在数据量少的情况下。

Probabilistic Approaches to Text Retrieval

Probabilistic Approach to IR

- Document ranking based on the conditional probability of relevance

$$P(R=1 \mid D, Q) \quad \mathbf{R} \in \{0,1\}$$

Relevance

Document

Query

Ranking based on

$$O(R \mid D, Q) = \frac{P(R=1 \mid D, Q)}{P(R=0 \mid D, Q)}$$

Odds-ratio

Problem: Hard to compute P(R|D, Q) directly

Odds（几率）

几率是“发生的概率”和“不发生的概率”的比值：

$$\text{odds} = \frac{P(\text{事件发生})}{P(\text{事件不发生})}$$

比如：

- 事件发生概率 $P = 0.8$ → 几率 = $0.8/0.2 = 4$
- 事件发生概率 $P = 0.5$ → 几率 = $0.5/0.5 = 1$
- 事件发生概率 $P = 0.2$ → 几率 = $0.2/0.8 = 0.25$

几率越大，表示事件更有可能发生。

Odds Ratio（几率比）

几率比是两个几率的比值。

在 IR 中，我们用的是相关和不相关的几率比：

$$\text{odds-ratio} = \frac{\text{相关的几率}}{\text{不相关的几率}} = \frac{P(R=1|D,Q)}{P(R=0|D,Q)}$$

当这个值 > 1 → 文档更可能是相关的；

当这个值 < 1 → 文档更可能是不相关的。

为什么用odd:

1. 数学上更方便
2. 区分度更大

Probabilistic Retrieval Models: Computing p(R|Q,D)

- Basic idea
  - Compute P(R|Q,D) using Bayes' rule

$$P(R \mid Q, D) = \frac{P(Q, D \mid R) P(R)}{P(Q, D)}$$

- P(D,Q|R=r) is the probability that if we know that a relevant document is retrieved, it's D (for query Q)
- P(R=r) is the probability of retrieving a relevant document
- P(D,Q) probability of retrieving D and issuing Q
- Assumption: computing P(Q, D|R) is easier.
- But what about P(R) and P(Q,D)?

- 对所有文档，$p(Q|D)$ 是常数 → 排序时可以忽略
- 如果假设相关性 R 和文档内容 D 无关（先验相同），$p(R|D)$ 也是常数 → 排序时也可以忽略

Probabilistic Retrieval Models: Computing p(R|Q,D)

- Assumption: computing P(Q, D|R) is easier.

$$O(R|Q,D)=\frac{P(R=1|Q,D)}{P(R=0|Q,D)}=\frac{P(Q,D|R=1)P(R=1)/P(Q,D)}{P(Q,D|R=0)P(R=0)/P(Q,D)}=\frac{P(Q,D|R=1)}{P(Q,D|R=0)}\boxed{\frac{P(R=1)}{P(R=0)}}$$

Odds ratio

Ignored for ranking D. Why?

- How to compute P(Q,D|R)?
  - Document "generation": P(Q,D|R) = P(D|Q,R) P(Q|R)

  考虑给定查询的情况下，文档 D 被"生成"的概率
  - Query "generation": P(Q,D|R) = P(Q|D,R) P(D|R)

  考虑给定文档的情况下，查询 Q被"生成"的概率

Query Generation and Document Generation

Ranking based on

假设文档 D 确实是相关的（R=1）
然后计算“用户发出了 Q 这个查询，且我们看到了 D 这篇文档”同时发生的概率

$$O(D,Q) = \frac{P(R=1 \mid D,Q)}{P(R=0 \mid D,Q)} \sim \frac{P(Q,D \mid R=1)}{P(Q,D \mid R=0)}$$

$P(R \mid D,Q)$ ➡ $P(Q,D \mid R)$

$P(D \mid Q,R)$ Document generation → Okapi/BM25

$P(Q \mid D,R)$ Query generation → Language models

The Notion of Relevance

Relevance

- Δ(Rep(q), Rep(d)) **Similarity**
  - Different rep & similarity
    - Vector space model (Salton et al., 75)
    - …
    - Prob. distr. model (Wong & Yao, 89)
- P(r=1|q,d) r $\in$ {0,1} **Probability of Relevance**
  - Regression Model (Fox 83)
  - Generative Model
    - Doc generation
      - Classical prob. Model (Robertson & Sparck Jones, 76)
    - Query generation
      - LM approach (Ponte & Croft, 98) (Lafferty & Zhai, 01a)
- P(d $\rightarrow$ q) or P(q $\rightarrow$ d) **Probabilistic inference**
  - Different inference system
    - Prob. concept space model (Wong & Yao, 95)
    - Inference network model (Turtle & Croft, 91)

Binary Independence Retrieval Model

- Assumes that query and doc terms are independent: it's a Naive Bayes Classifier for IR
- Documents are represented by binary vectors
  - if a term is present in a document, it's 1, otherwise it's 0

$$score(Q,D) = log\frac{P(R=r|D,Q)}{P(R=\neg r|D,Q)}$$

We're not going to produce a probability here. why?

the log-odds ratio of being relevant or not

Binary Independence Retrieval Model

Bayes Rule

$$score(Q,D) = log\frac{P(R=r|D,Q)}{P(R=\neg r|D,Q)} = log\frac{P(D,Q|R=r)P(R=r)}{P(D,Q|R=\neg r)P(R=\neg r)}$$

P(R=r) and P(R=¬r) are just constants and will not change relative positions of documents in the rating, so let's remove them:

$$score(Q,D) = log\frac{P(D,Q|R=r)}{P(D,Q|R=\neg r)}$$

Binary Independence Retrieval Model

$$score(Q, D) = log \frac{P(D, Q|R = r)}{P(D, Q|R = \neg r)}$$

- We assume each term in D is independent, so

$$P(D, Q|R) = \prod_i P(T_i|Q, R)$$

Distribute the product

Log rule for products

$$s(Q, D) = \log \frac{\prod P(T_i \mid Q, R = r)}{\prod P(T_i \mid Q, R = \neg r)} = \log \prod \frac{P(T_i \mid Q, R = r)}{P(T_i \mid Q, R = \neg r)} = \sum \log \frac{P(T_i \mid Q, R = r)}{P(T_i \mid Q, R = \neg r)}$$

T_i is presence of a term T_i in D

Binary Independence Retrieval Model

$$s(Q,D) = \sum_{i \in Q} \log \frac{P(T_i \mid Q, R = r)}{P(T_i \mid Q, R = \neg r)}$$

- Let's define some terms:

$p_i = P(T_1 = 1 \mid Q, R = r)$ $q_i = P(T_1 = 1 \mid Q, R = \neg r)$

	$T_i = 1$	p_i	q_i
	$T_i = 0$	$(1-p_i)$	$(1 - q_i)$

- If we assume for all terms not occurring in the query that $p_i = q_i$, we can redefine the score as (a few steps skipped here. Included in additional slides)

$$s(Q,D) = \sum_{i \in Q \cap D} \log \frac{p_i(1-q_i)}{(1-p_i)q_i}$$

- $p_i = P(T_i = 1|Q, R = 1)$：词 w_i 出现在相关文档里的概率
- $q_i = P(T_i = 1|Q, R = 0)$：词 w_i 出现在不相关文档里的概率

How do we get p_i and q_i?

$$s(Q,D) = \sum_{i \in Q \cap D} \log \frac{p_i(1-q_i)}{(1-p_i)q_i}$$

$$p_i = P(T_1 = 1 \mid Q, R = r)$$

$$q_i = P(T_1 = 1 \mid Q, R = \neg r)$$

- given a query Q and a corpus C
  - let N be the number of documents in the co
  - let R be the number of documents relevant
  - n_i be the # of docs that have term w_i
  - r_i be the # of relevant docs that have term

- N：语料库中文档总数
- R：与查询相关的文档总数
- n_i：包含词 w_i 的文档数
- r_i：包含词 w_i 且相关的文档数

$$p_i = \frac{r_i + \lambda}{R + 2\lambda} \qquad q_i = \frac{n_i - r_i + \lambda}{N - R + 2\lambda}$$

Typically set λ=0.5 to avoid 0 in log.
Why one alpha for the nominator and two for denominator: intuition is that you add one document that includes and one that doesn’t for term t_i

The Okapi/BM25 Retrieval Formula

- Classical probabilistic model (among the best performers!)
- Basic Idea: P(D,Q|R) = P(D|Q, R)P(Q|R) Ignored for ranking
- Many ways to define the distribution P(D|Q, R), Okapi uses a 2-Poisson model

Doc. Term frequency

Query term frequency

$$Score(d,q) = \sum_{w\in q, w\in d} \left(\log \frac{N - df(w) + 0.5}{df(w) + 0.5} \cdot \frac{(k_1 + 1)\cdot c(w,d)}{k_1((1-b) + b\frac{|d|}{avdl}) + c(w,d)} \cdot \frac{(k_3 + 1)\cdot c(w,q)}{k_3 + c(w,q)}\right)$$

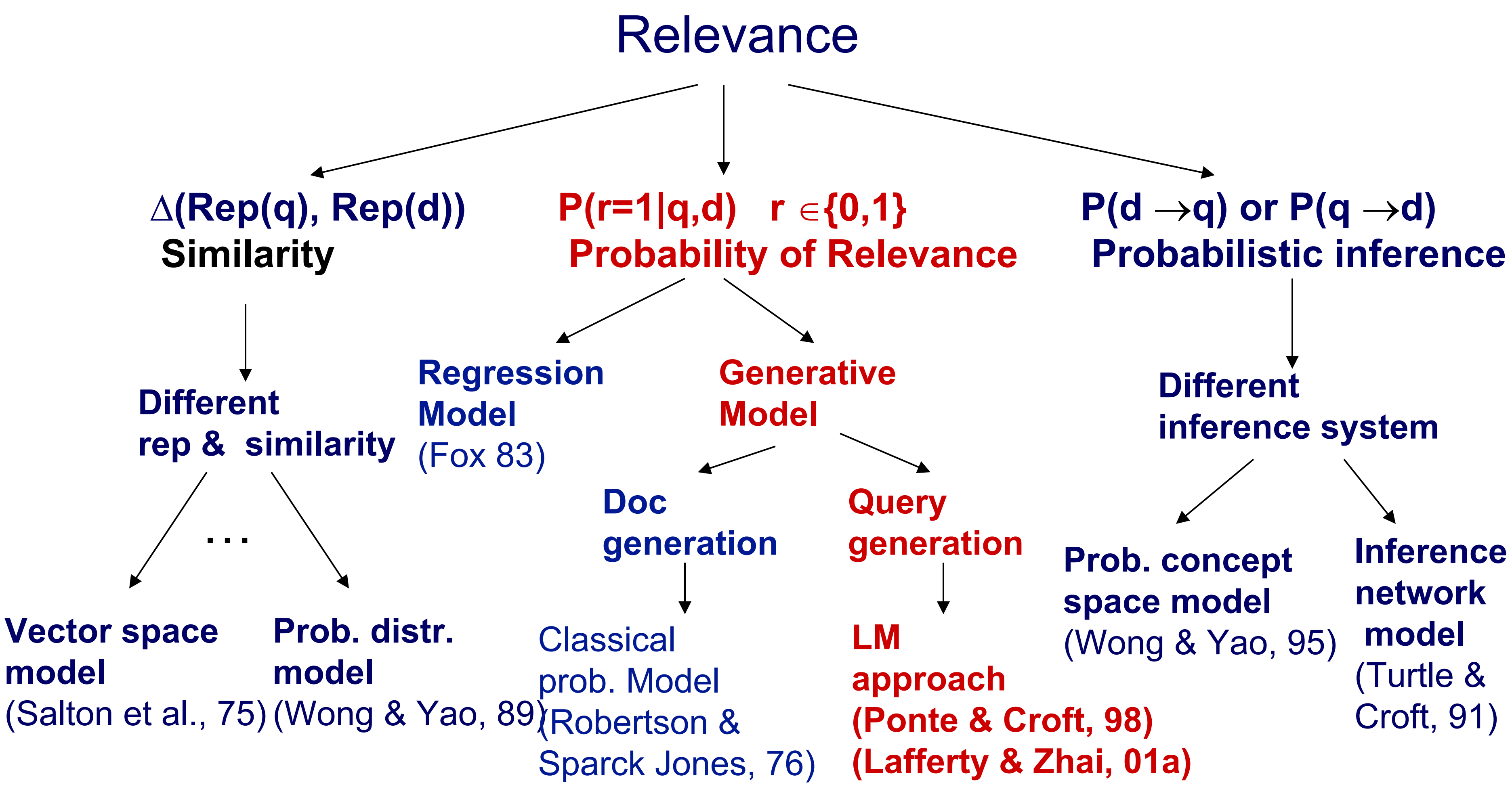
Only consider words in the query

IDF

Doc. length

More info for those curious: https://www.staff.city.ac.uk/~sbrp622/papers/foundations_bm25_review.pdf

The Notion of Relevance



Statistical Language Models

We already know a very simple language model

Word choice as weighted dice

This is a simple language model

Here, we have *two* language models

positive reviews

negative reviews

Language Model

- Vocabulary $\mathcal{V}$ is a finite set of discrete symbols (e.g., words, characters); V = | $\mathcal{V}$ |
- $\mathcal{V}^+$ is the infinite set of sequences of symbols from $\mathcal{V}$; each sequence ends with STOP
- $x \in \mathcal{V}^+$

Language Model

$$P(w) = P(w_1, \ldots, w_n)$$

P("all is well") =
P(w_1 = "all", w_2 = "is", w_3 = "well") x P(STOP)

$$\sum_{w \in V^+} P(w) = 1$$

$$0 \leq P(w) \leq 1$$

over all sequence lengths!

Language Model

- Language models provide us with a way to quantify the likelihood of sequence of words — i.e., plausible sentences.

Language Model

- Language modeling is the task of estimating P(w)
- How do we compute this?

P("Do or do not. There is no try.")

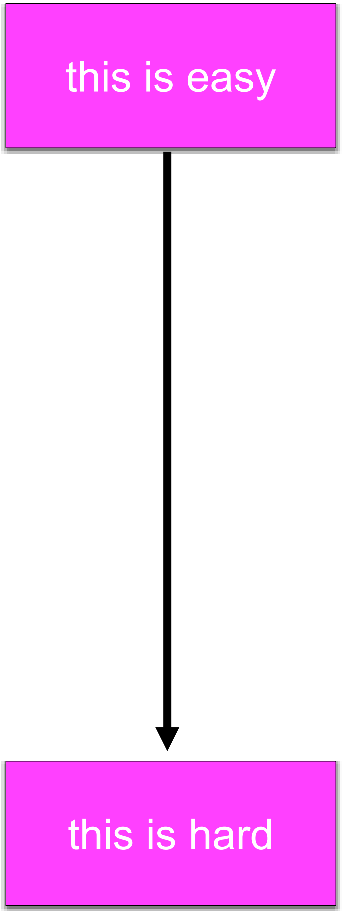
Chain rule (of probability)

$$
\begin{aligned}
P(x_1, x_2, x_3, x_4, x_5) = {} & P(x_1) \\
& \times P(x_2 \mid x_1) \\
& \times P(x_3 \mid x_1, x_2) \\
& \times P(x_4 \mid x_1, x_2, x_3) \\
& \times P(x_5 \mid x_1, x_2, x_3, x_4)
\end{aligned}
$$

Chain rule (of probability)

P("It")

P("was" | "It")



$$P(w_1)$$

$$P(w_2 \mid w_1)$$

$$P(w_3 \mid w_1, w_2)$$

$$P(w_4 \mid w_1, w_2, w_3)$$

$$P(w_n \mid w_1, \ldots, w_{n-1})$$

P("try" | "Do or do not. There is no")

Markov assumption

first-order

$$P(x_i \mid x_1, \ldots x_{i-1}) \approx P(x_i \mid x_{i-1})$$

second-order

$$P(x_i \mid x_1, \ldots x_{i-1}) \approx P(x_i \mid x_{i-2}, x_{i-1})$$

Which one of these two give us a bi-gram model?

Respond at PollEv.com/cerenbudak421

First-order

Log in to Poll Everywhere

To present live activities, please log in to your Poll Everywhere account in a separate window.

Launch log-in window

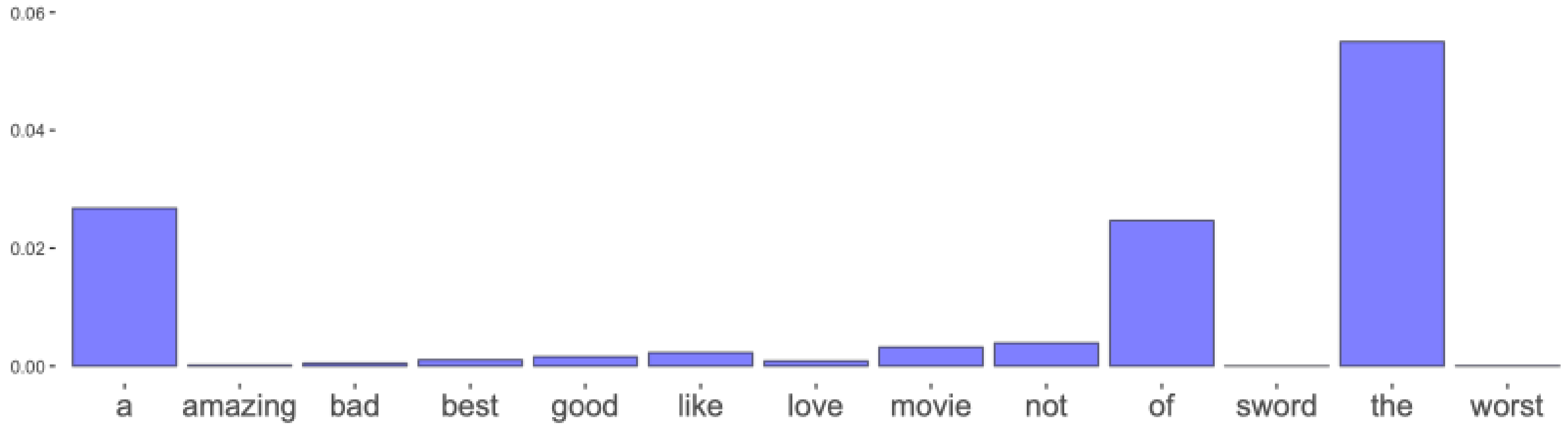
Estimation

unigram	bigram	trigram
$\prod_{i}^{n} P(w_i)$	$\prod_{i}^{n} P(w_i \mid w_{i-1})$	$\prod_{i}^{n} P(w_i \mid w_{i-2}, w_{i-1})$
$\times P(STOP)$	$\times P(STOP \mid w_n)$	$\times P(STOP \mid w_{n-1}, w_n)$

Maximum likelihood estimate

$\frac{c(w_i)}{N}$	$\frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$	$\frac{c(w_{i-2}, w_{i-1}, w_i)}{c(w_{i-2}, w_{i-1})}$

Generating

- 

- (including STOP) for each context

Text Generation with Unigram LM

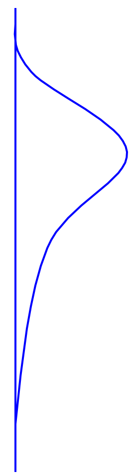
(Unigram) Language Model θ
$p(w|\theta)$

Sampling

Document

Topic 1:
Text mining

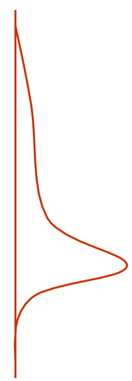
…
text 0.2
mining 0.1
association 0.01
clustering 0.02
…
food 0.00001
…



Text mining paper

Topic 2:
Health

…
food 0.25
nutrition 0.1
healthy 0.05
diet 0.02
…



Food nutrition paper

Estimation of Unigram LM

(Unigram) Language Model θ **Estimation** **Document**

$p(w|\theta)$=?

	…	
10/100	text ?	text 10
5/100	mining ?	mining 5
3/100	association ?	association 3
3/100	database ?	database 3
		algorithm 2
	…	…
1/100	query ?	query 1
		efficient 1
	…	

A "text mining paper" (total #words=100)

Revisit: Maximum Likelihood Estimate

Data: a document d with counts $c(w_1)$, …, $c(w_N)$, and length |d|
Model: multinomial distribution M with parameters $\{p(w_i)\}$
Likelihood: p(d|M)
Maximum likelihood estimator: $M = \text{argmax}_M \; p(d|M)$

$$p(w_i) = \frac{c(w_i)}{|d|}$$

Problem: in a short document, if $c(w_i) = 0$, does this mean $p(w_i) = 0$?

Let's go back to our simple unigram model

unigram

$$\prod_{i}^{n} P(w_i)$$

$$\times P(STOP)$$

- What is the probability if one of the words does not occur in the document?
- Solution?

We can combine language models through interpolation（插值）

- A linear interpolation of any two language models p and q (with $\lambda \in [0,1]$) is also a valid language model.
- Often useful if we have some “background” language model with a much larger vocabulary

$$\lambda p + (1 - \lambda) q$$

p = the web

q = political speeches

Interpolation

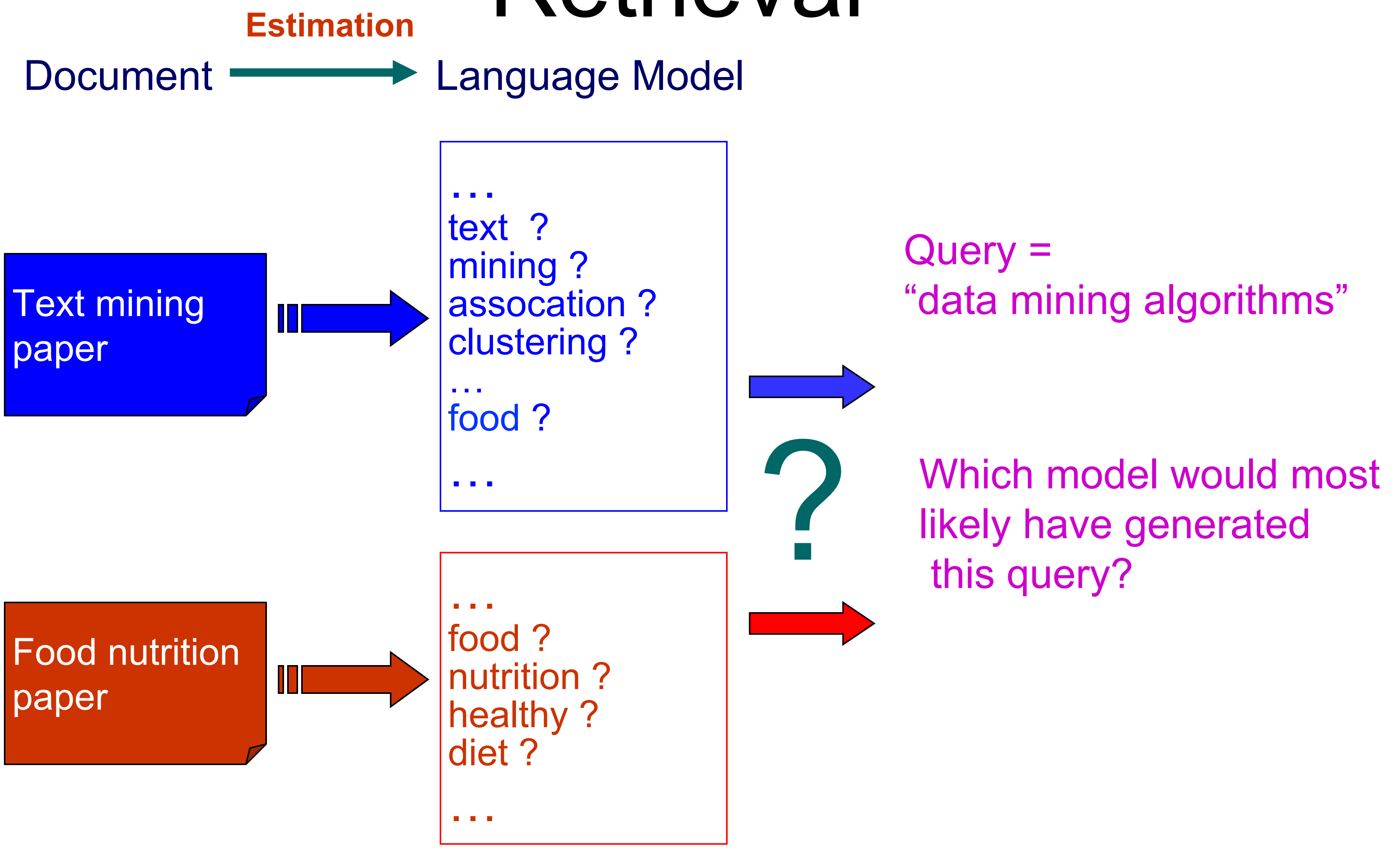
$$\lambda p + (1 - \lambda) q$$

p = Star Wars text

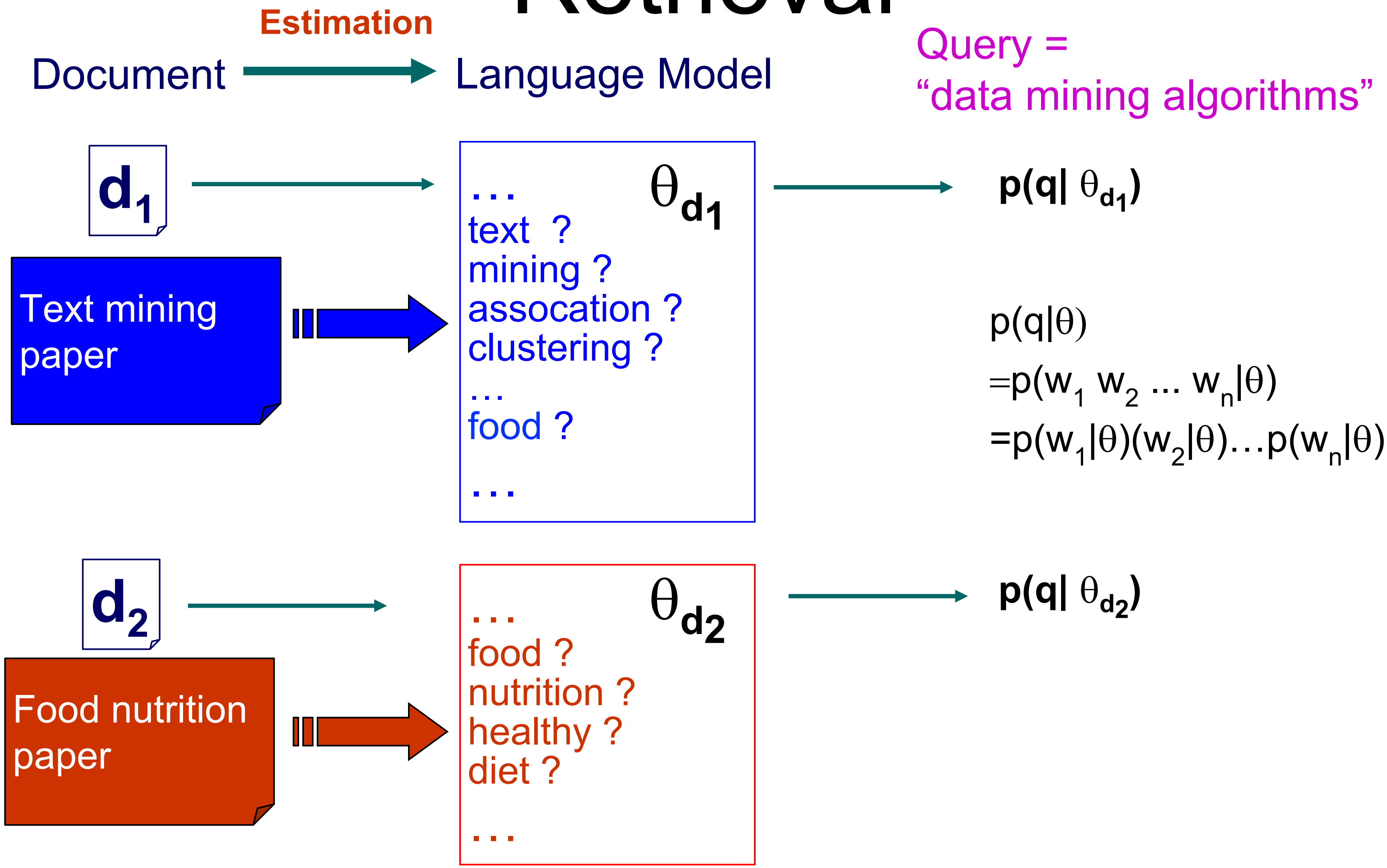
q = Programming text

Language Models for IR

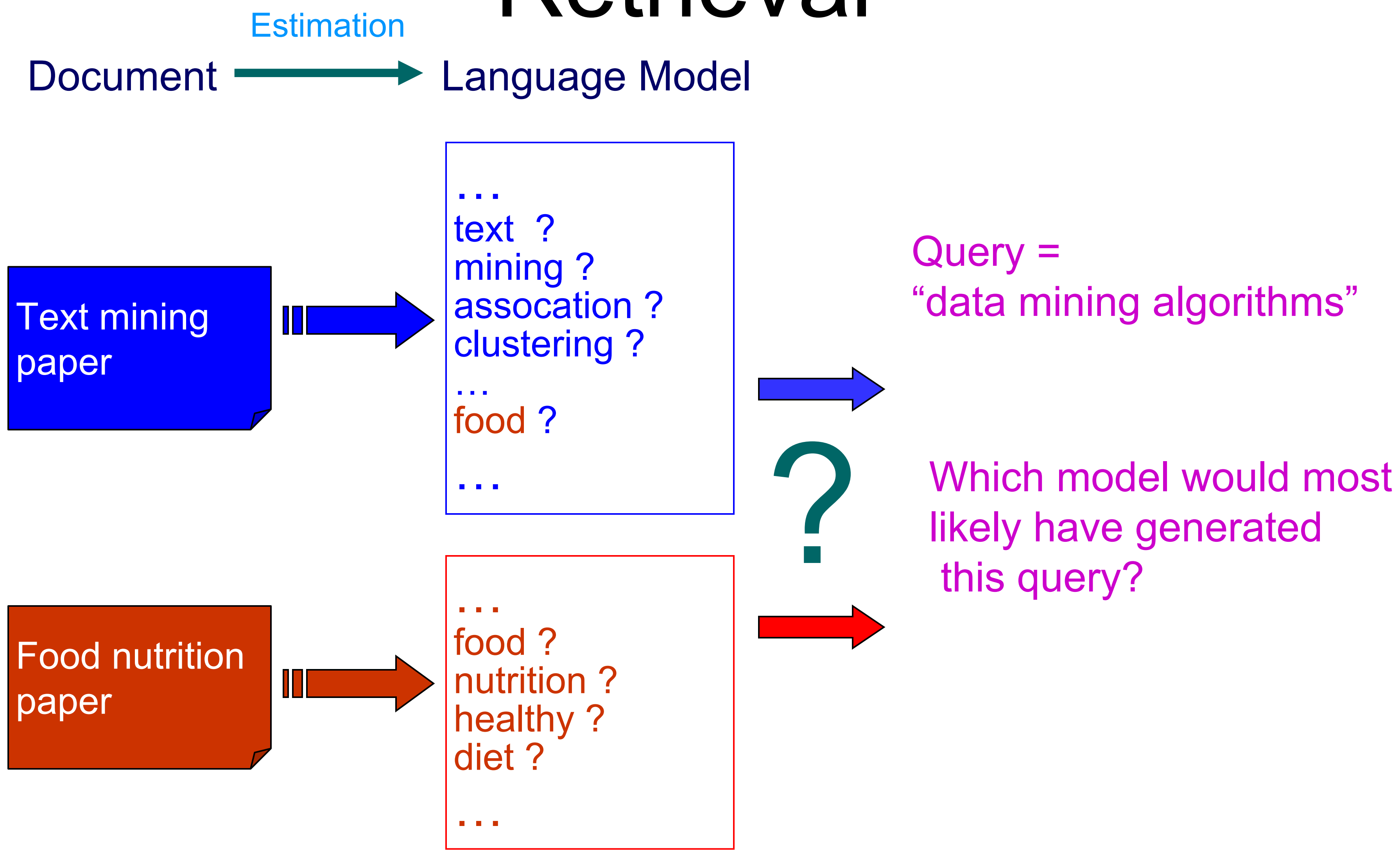
Language Models for Retrieval



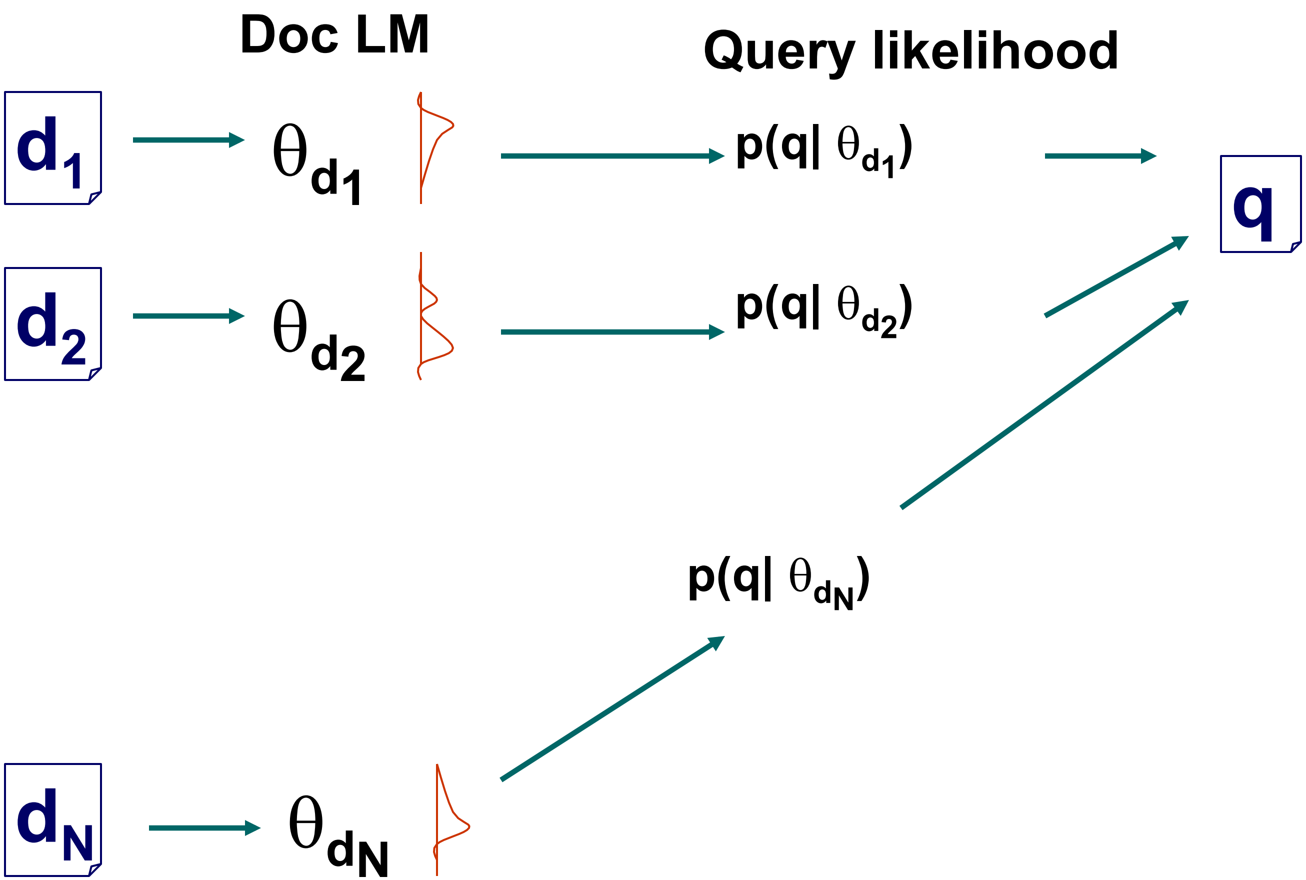
Language Models for Retrieval



Language Models for Retrieval



Ranking Docs by Query Likelihood



Retrieval as Language Model Estimation

- Document ranking based on query likelihood

$$\log p(q \mid d) = \sum_i \log \boxed{p(w_i \mid d)}$$

$$where,\ q = w_1 w_2 \ldots w_n$$

Document language model

- Retrieval problem ≈ Estimation of $p(w_i|d)$

Problem with the Maximum Likelihood Estimator

- What if a query word doesn't appear in a document?

  $p(\text{“information retrieval umiich”}|\boldsymbol{\theta})$

- In general, what probability should we give a word that has not been observed?
- If we want to assign non-zero probabilities to such words, we'll have to discount the probabilities of observed words
- This is what “smoothing” is about …

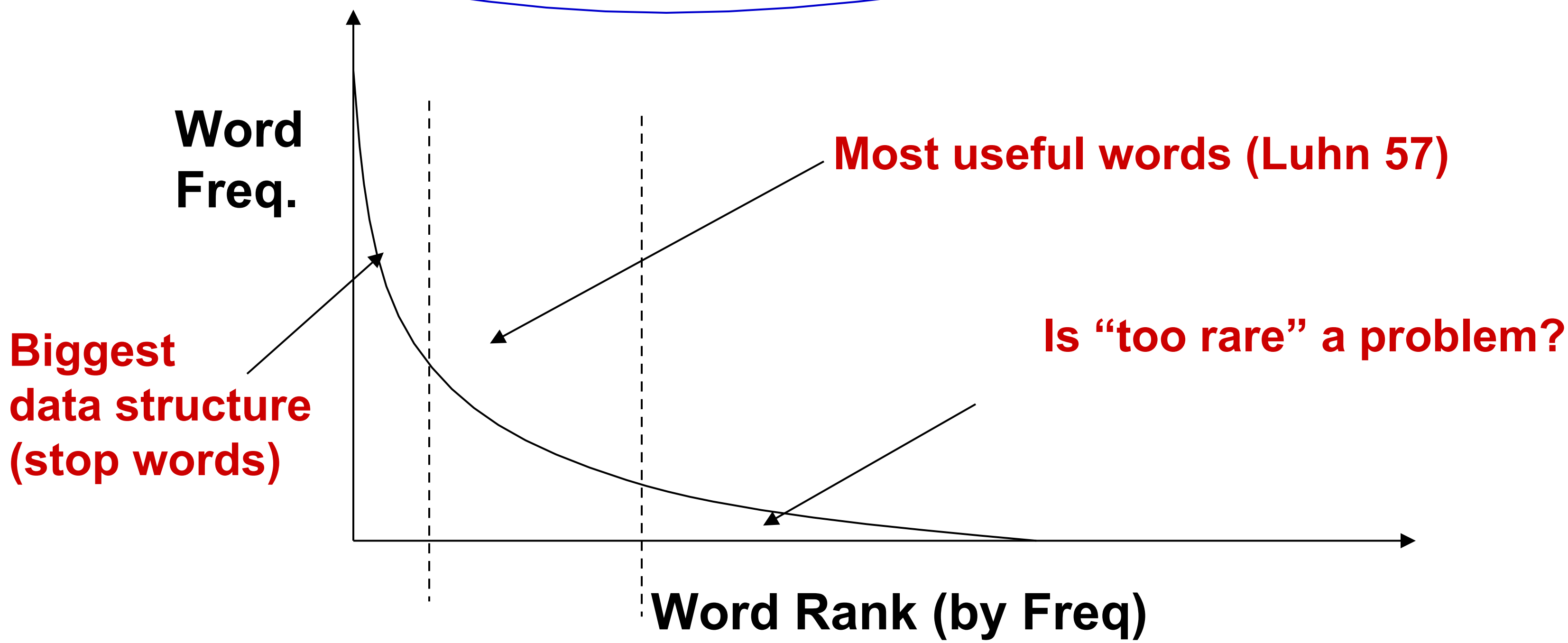
Empirical distribution of words

- There are stable language-independent patterns in how people use natural languages
- A few words occur very frequently; most occur rarely. E.g., in news articles,
  – Top 4 words: 10~15% word occurrences
  – Top 50 words: 35~40% word occurrences
- The most frequent word in one corpus may be rare in another

Revisit: Zipf's Law

- rank * frequency ≈ constant

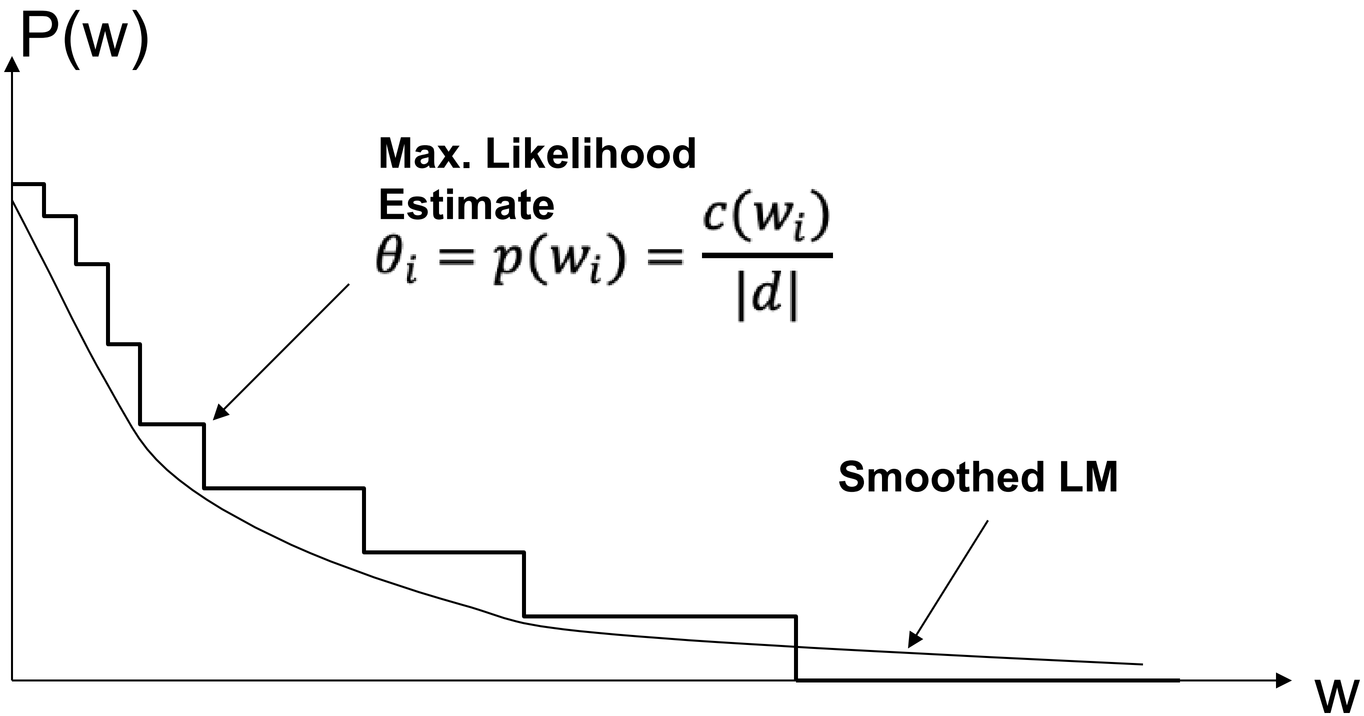
$$F(w) = \frac{C}{r(w)^{\alpha}} \quad \alpha \approx 1, C \approx 0.1$$



Generalized Zipf's law: $$F(w) = \frac{C}{[r(w) + B]^{\alpha}}$$ **Applicable in many domains**

Language Model Smoothing (Illustration)

平滑（Smoothing），它是语言模型里非常重要的一步，主要目的是解决“某个词没在文档里出现导致概率为 0”的问题。



How to Smooth?

- All smoothing methods try to
  - discount the probability of words seen in a document
  - re-allocate the extra counts so that unseen words will have a non-zero count
- Method 1 (Additive smoothing): Add a

平滑方法通过：
- 降低（discount）已出现过的词的概率
- 把"省下的"概率分配给没出现的词，让它们概率 > 0

unts of each word

Counts of w in d

"Add one", Laplace smoothing

$$p(w|d) = \frac{c(w,d)+1}{|d|+|V|}$$

Vocabulary size

Length of d (total counts)

Additive smoothing

保证即便没出现也有非零概率

Laplace smoothing:
α = 1

$$P(w_i) = \frac{c(w_i) + \alpha}{N + V\alpha}$$

保证概率和为 1

Same idea for bigram language models

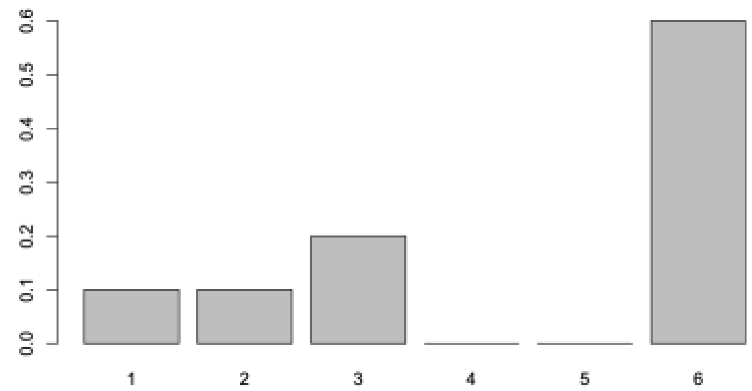
$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + \alpha}{c(w_{i-1}) + V\alpha}$$

Smoothing

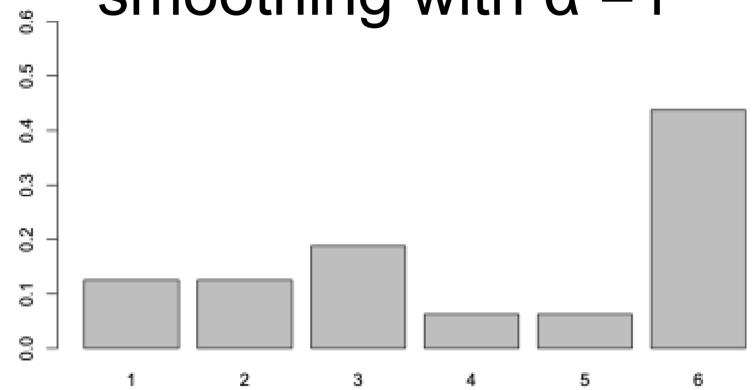
- Maximum likelihood estimates can fail miserably when features are never observed with a particular class.

Smoothing is the re-allocation of probability mass

MLE



smoothing with α =1



How to Smooth? (cont.)

- Should all unseen words get equal probabilities?
- We can use a reference model to discriminate unseen words

Discounted ML estimate

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

Reference language model

$$\alpha_d = \frac{1 - \sum_{w \text{ is seen}} p_{seen}(w|d)}{\sum_{w \text{ is unseen}} p(w|REF)}$$

更智能的平滑方法，不是简单给所有没出现的词分配相同概率，而是借助一个“参考语言模型”（reference model），让分配给未出现词的概率更合理。

Other Smoothing Methods

- Method 2 (Absolute discounting): Subtract a constant δ from the counts of each word

uniq words

$$p(w|d) = \frac{\max(c(w;d)-\delta,0)+\delta|d|_u p(w|REF)}{|d|}$$

$\alpha_d = \delta|d|_u/|d|$

- Method 3 (Linear interpolation, Jelinek-Mercer): "Shrink" uniformly toward p(w|REF)

$$p(w|d) = (1-\lambda)\boxed{\frac{c(w,d)}{|d|}} + \lambda\, p(w|REF)$$

ML estimate

parameter

$\alpha_d = \lambda$

Jelinek-Mercer Smoothing

Idea: use linear combination of document-specific LM with background LM (corpus, common language):

$$\hat{p}_j(d) = \lambda \frac{freq(j,d)}{|d|} + (1-\lambda)\frac{freq(j,C)}{|C|}$$

could also consider query log as background LM for query

Parameter tuning of λ by cross-validation with held-out data:

- divide set of relevant (d,q) pairs into n partitions
- build LM on the pairs from n-1 partitions
- choose λ to maximize precision (or recall or F1) on n^{th} partition
- iterate with different choice of n^{th} partition and average

Other Smoothing Methods

(cont.)

- Method 4 (Dirichlet Prior/Bayesian):

  - Assume adding pseudo counts $\mu p(w|REF)$
  - A dynamic coefficient interpolation w.r.t. |d|

$$p(w \mid d) = \frac{c(w;d) + \mu\, p(w|REF)}{|d| + \mu} = \frac{|d|}{|d| + \mu} \frac{c(w,d)}{|d|} + \underbrace{\frac{\mu}{|d| + \mu}}_{\text{parameter}} p(w \mid REF)$$

How to rank using Dirichlet Smoothing

μ is a hyper parameter that lets us control how much smoothing to do

$$p(w|d) = \frac{c(w;d)+\mu\, p(w|C)}{|d|+\mu} = \frac{|d|}{|d|+\mu}\frac{c(w,d)}{|d|} + \frac{\mu}{|d|+\mu}p(w|C)$$

$$\log p(q|d)$$

Query term frequency

Doc. Term frequency

$$Score(d,q) = \sum_{\boxed{w\in q, w\in d}} \boxed{c(w,q)} \cdot \log\left(1 + \frac{\boxed{c(w,d)}}{\mu \cdot \boxed{p(w|C)}}\right) + |q| \cdot \log\frac{\mu}{\boxed{|d|}+\mu}$$

Only consider words in the query **Similar to IDF** **Doc. length**

So, which smoothing method is the best?

- It depends on the data and the task!
- Many other sophisticated smoothing methods have been proposed…
- Cross validation is generally used to choose the best method and/or set the smoothing parameters…
- For retrieval, Dirichlet prior performs well…

A General Smoothing Scheme

- All smoothing methods try to
  - discount the probability of words seen in a doc
  - re-allocate the extra probability so that unseen words will have a non-zero probability
- Most use a reference model (collection language model) to discriminate unseen words

Discounted ML estimate

$$p(w \mid d) = \begin{cases} p_{seen}(w \mid d) & \textit{if } w \textit{ is seen in } d \\ \alpha_d \, p(w \mid C) & \textit{otherwise} \end{cases}$$

Collection language model

Comparison of Three Smoothing Methods

(Zhai & Lafferty 01)

- Simplified Jelinek-Mercer: Shrink uniformly toward p(w|C)

$$p(w|d) = (1-\lambda)p_{ml}(w|d) + \lambda(p|C)$$

- Dirichlet prior (Bayesian): Assume pseudo counts μp(w|C)

$$p(w|d) = \frac{c(w;d)+\mu\, p(w|C)}{|d|+\mu} = \frac{|d|}{|d|+\mu} p_{ml}(w|d) + \frac{\mu}{|d|+\mu} p(w|C)$$

- Absolute discounting: Subtract a constant δ

$$p(w|d) = \frac{\max(c(w;d)-\delta,0)+\delta|d|_u\, p(w|C)}{|d|}$$

① Jelinek-Mercer 平滑 (线性插值)

$$p(w|d) = (1-\lambda)p_{ml}(w|d) + \lambda p(w|C)$$

- **p_ml(w|d)**：最大似然估计 (文档内频率)。
- **p(w|C)**：背景语言模型。
- **λ ∈ [0,1]**：平衡两部分的权重。
- **特点**：把文档概率直接往全局概率"拉近" (Shrink uniformly)，适合文档很短的情况。

② Dirichlet Prior 平滑 (贝叶斯方法)

$$p(w|d) = \frac{c(w;d) + \mu p(w|C)}{|d| + \mu} = \frac{|d|}{|d|+\mu}p_{ml}(w|d) + \frac{\mu}{|d|+\mu}p(w|C)$$

- **c(w;d)**：文档中词 w 的出现次数。
- **|d|**：文档长度。
- **μ**：平滑参数，相当于加入了 μ 个"伪计数 (pseudo counts)"。
- **特点**：文档越长，越信任文档自己的统计；文档越短，越依赖全局统计。

③ Absolute Discounting 平滑 (绝对折扣)

$$p(w|d) = \frac{\max(c(w;d) - \delta, 0) + \delta|d|_u p(w|C)}{|d|}$$

- **δ**：从每个已出现词的计数中减去一个固定值。
- **|d|_u**：文档中不同词的数量（unique terms）。
- 被减去的概率总和再分配给未见词，按背景模型分配。

> **特点**：相比 Jelinek-Mercer 和 Dirichlet，这种方法对高频词影响较小，只是简单减去一个常数。

Evaluation

How do we know if our language model captures queries or documents?

Evaluation: How good is our model?

- Does our language model prefer good sentences to bad ones?
  - Assign higher probability to “real” or “frequently observed” sentences
    - Than “ungrammatical” or “rarely observed” sentences?
- We train parameters of our model on a training set.
- We test the model’s performance on data we haven’t seen.
  - A test set is an unseen dataset that is different from our training set, totally unused.
  - An evaluation metric tells us how well our model does on the test set.

Training on the test set

- We can’t allow test sentences into the training set
- We will assign it an artificially high probability when we set it in the test set
- “Training on the test set”
- Bad science!
- And violates the honor code

Evaluation

- The best evaluation metrics are external — how does a better language model influence the application you care about?

Extrinsic evaluation（外部评估） of language models

- Best evaluation for comparing models A and B
  - Put each model in a task
    - IR system, query prediction, etc.
  - Run the task, get an accuracy for A and for B
    - How many times people clicked links
    - How many auto-suggests were used
  - Compare accuracy for A and B

给两个模型（A 和 B）同样的任务

运行任务，统计每个模型的效果

Intrinsic Evaluation

- A good language model should judge unseen real language to have high probability
- Perplexity（困惑度） = inverse probability of test data, averaged by word.测试数据的逆概率（取平均后开方），越低说明模型越好。
- Not a good idea for IR.
- To be reliable, the test data must be truly unseen, including knowledge of its vocabulary (could have unseen words!).

- **核心思想**：直接测语言模型本身，而不是放到应用里。

Experiment design

	training	development	testing
size	80%	10%	10%
purpose	training models	model selection; hyperparameter tuning	evaluation; never look at it until the very end

Summary of Language Model

- Goal: estimate a language model from a sample text
- When using maximum-likelihood estimator
  - count the number of times each word occurs in S, divide by length
- Smoothing to avoid zero frequencies (and probabilities!)
  - discounting methods: add or subtract a constant, redistribute mass
  - better: interpolate with background probability of a word
  - smoothing has a role similar to IDF in classical models
- Smoothing parameters very important
  - Dirichlet works well for short queries (need to tune the parameter)
  - Jelinek-Mercer works well for longer queries (also needs tuning)
  - Lots of other ideas being worked on

You Should Know

- How probabilistic approaches work in retrieval in general.
- Okapi/BM25 is a probabilistic model, although the function looks very similar to a VSM
- The basic idea of ranking docs by query likelihood (“the language modeling approach”)
- How smoothing is connected with TF-IDF weighting and document length normalization

Additional slides

Binary Independence Model

- Traditionally used in conjunction with PRP
- **"Binary" = Boolean**: documents are represented as binary incidence vectors of terms (cf. IIR Chapter 1):
  - $\vec{x} = (x_1, \ldots, x_n)$
  - $x_i = 1$ iff term *i* is present in document *x*.
- **"Independence":** terms occur in documents independently
- Different documents can be modeled as the same vector

Binary Independence Model

- Queries: binary term incidence vectors
- Given query *q*,
  - for each document *d* need to compute *p*(*R*|*q*,*d*).
  - replace with computing *p*(*R*|*q*,*x*) where *x* is binary term incidence vector representing *d*.

    （d 是二进制向量 x）
  - Interested only in ranking
- Will use odds and Bayes' Rule:

$$O(R|q,\vec{x}) = \frac{p(R=1|q,\vec{x})}{p(R=0|q,\vec{x})} = \frac{\dfrac{p(R=1|q)\,p(\vec{x}|R=1,q)}{p(\vec{x}|q)}}{\dfrac{p(R=0|q)\,p(\vec{x}|R=0,q)}{p(\vec{x}|q)}}$$

Binary Independence Model

$$O(R \mid q, \vec{x}) = \frac{p(R=1 \mid q, \vec{x})}{p(R=0 \mid q, \vec{x})} = \frac{p(R=1 \mid q)}{p(R=0 \mid q)} \cdot \frac{p(\vec{x} \mid R=1, q)}{p(\vec{x} \mid R=0, q)}$$

Constant for a given query

Needs estimation

- Using **Independence** Assumption:

$$\frac{p(\vec{x} \mid R=1, q)}{p(\vec{x} \mid R=0, q)} = \prod_{i=1}^{n} \frac{p(x_i \mid R=1, q)}{p(x_i \mid R=0, q)}$$

$$O(R \mid q, \vec{x}) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R=1, q)}{p(x_i \mid R=0, q)}$$

Binary Independence Model

$$O(R|q,\vec{x}) = O(R|q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R=1,q)}{p(x_i \mid R=0,q)}$$

- Since x_i is either 0 or 1:

$$O(R|q,\vec{x}) = O(R|q) \cdot \prod_{x_i=1} \frac{p(x_i=1 \mid R=1,q)}{p(x_i=1 \mid R=0,q)} \cdot \prod_{x_i=0} \frac{p(x_i=0 \mid R=1,q)}{p(x_i=0 \mid R=0,q)}$$

- Let $p_i = p(x_i=1 \mid R=1,q);\ \ r_i = p(x_i=1 \mid R=0,q);$

- Assume, for all terms not occurring in the query $(q_i=0)$ $p_i = r_i$

$$O(R|q,\vec{x}) = O(R|q) \cdot \prod_{\substack{x_i=1\\ q_i=1}} \frac{p_i}{r_i} \cdot \prod_{\substack{x_i=0\\ q_i=1}} \frac{(1-p_i)}{(1-r_i)}$$

	document	relevant (R=1)	not relevant (R=0)
term present	$x_i = 1$	p_i	r_i
term absent	$x_i = 0$	$(1 - p_i)$	$(1 - r_i)$

Binary Independence Model

$$O(R|q,\vec{x}) = O(R|q)\cdot \underbrace{\prod_{x_i=q_i=1}\frac{p_i}{r_i}}_{\text{All matching terms}} \cdot \underbrace{\prod_{\substack{x_i=0\\ q_i=1}}\frac{1-p_i}{1-r_i}}_{\text{Non-matching query terms}}$$

All matching terms

Non-matching query terms

$$O(R|q,\vec{x}) = O(R|q)\cdot \prod_{\substack{x_i=1\\ q_i=1}}\frac{p_i}{r_i}\cdot \prod_{\substack{x_i=1\\ q_i=1}}\left(\frac{1-r_i}{1-p_i}\cdot\frac{1-p_i}{1-r_i}\right)\prod_{\substack{x_i=0\\ q_i=1}}\frac{1-p_i}{1-r_i}$$

$$O(R|q,\vec{x}) = O(R|q)\cdot \underbrace{\prod_{x_i=q_i=1}\frac{p_i(1-r_i)}{r_i(1-p_i)}}_{\text{All matching terms}}\cdot \underbrace{\prod_{q_i=1}\frac{1-p_i}{1-r_i}}_{\text{All query terms}}$$

All matching terms

All query terms

Binary Independence Model

$$O(R\,|\,q,\vec{x}) = \boxed{O(R\,|\,q)} \cdot \boxed{\prod_{x_i=q_i=1} \frac{p_i(1-r_i)}{r_i(1-p_i)}} \cdot \boxed{\prod_{q_i=1} \frac{1-p_i}{1-r_i}}$$

如果某个词在相关文档里出现的概率 p_i 远大于在不相关文档里出现的概率 r_i，那么它就是一个“好词”，能强烈提升文档得分。

Constant for each query

Only quantity to be estimated for rankings

Retrieval Status Value:

$$RSV = \log \prod_{x_i=q_i=1} \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_{x_i=q_i=1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$