COEN 169

Statistical Language Modeling II

Yi Fang

Department of Computer Engineering

Santa Clara University

Outline

Introduction to language modeling

Language modeling for information retrieval

Query-likelihood Retrieval Model

Smoothing

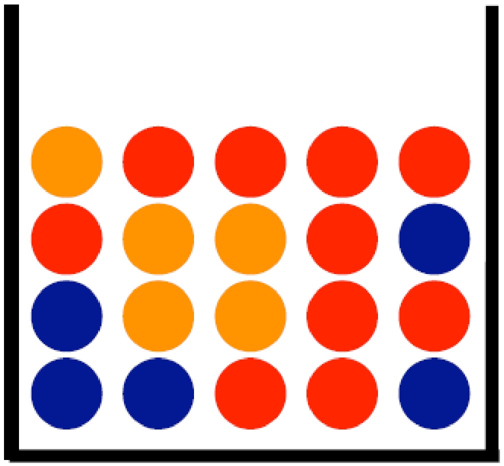
Priors

Smoothing Probability Estimates

- When estimating probabilities, we tend to ...
  - Over-estimate the probability of observed outcomes
  - Under-estimate the probability of unobserved outcomes
- The goal of smoothing is to ...
  - Decrease the probability of observed outcomes
  - Increase the probability of unobserved outcomes
- It’s usually a good idea

Smoothing Probability Estimates

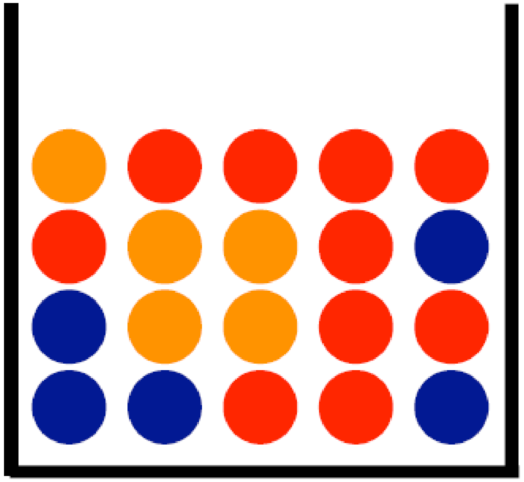
- Suppose that in reality this bag is a sample from a different, bigger bag ...
- And, our goal is to estimate the probabilities of that bigger bag ...
- And, we know that the bigger bag has red, blue, orange, yellow, and green balls.



P(RED) = 0.5
P(BLUE) = 0.25
P(ORANGE) = 0.25
P(YELLOW) = 0.00
P(GREEN) = 0.00

Smoothing Probability Estimates

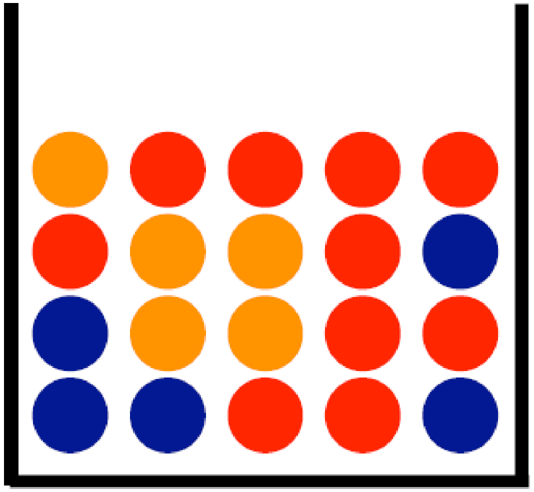
- Do we really want to assign YELLOW and GREEN balls a zero probability?
- What else can we do?



P(RED) = (10/20)
P(BLUE) = (5/20)
P(ORANGE) = (5/20)
P(YELLOW) = (0/20)
P(GREEN) = (0/20)

Add-One Smoothing

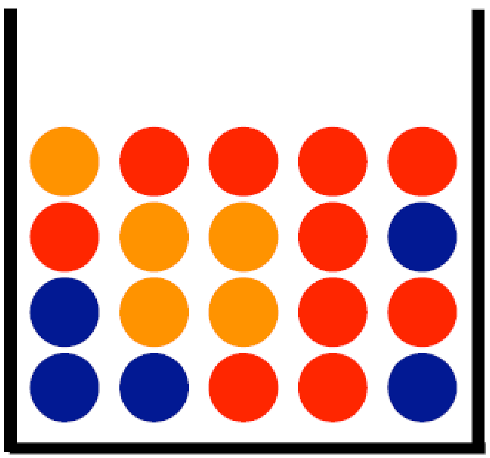
- We could add one ball of each color to the bag
- This gives a small probability to unobserved outcomes (YELLOW and GREEN)
- As a result, it also reduces the probability of observed outcomes (RED, BLUE, ORANGE) by a small amount
- Very common solution (also called 'discounting')



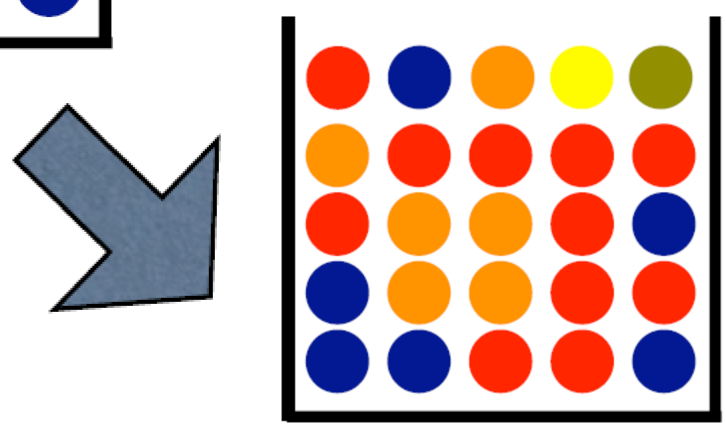
P(RED) = (11/25)
P(BLUE) = (6/25)
P(ORANGE) = (6/25)
P(YELLOW) = (1/25)
P(GREEN) = (1/25)

Add-One Smoothing

- Gives a small probability to unobserved outcomes (YELLOW and GREEN) and reduces the probability of observed outcomes (RED, BLUE, ORANGE) by a small amount



P(RED) = (10/20)
P(BLUE) = (5/20)
P(ORANGE) = (5/20)
P(YELLOW) = (0/20)
P(GREEN) = (0/20)



P(RED) = (11/25)
P(BLUE) = (6/25)
P(ORANGE) = (6/25)
P(YELLOW) = (1/25)
P(GREEN) = (1/25)

Document Language Models

- Estimating a document's language model:
  1. tokenize/split the document text into terms
  2. count the number of times each term occurs ($tf_{t,D}$)
  3. count the total number of term occurrences (N_D)
  4. assign term *t* a probability equal to:

$$\frac{tf_{t,D}}{N_D}$$

Smoothing Probability Estimates
for document language models

- In theory, we could use add-one smoothing
- To do this, we would add each indexed-term once into each document
  - Conceptually!
- Then, we would compute its language model probabilities

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 constant δ to the counts of each word

Counts of t in D

"Add one", Laplace smoothing

$$P(t|\theta_D) = \frac{tf_{t,D} + 1}{N_D + |V|}$$

Vocabulary size

Length of D (total counts)

Method 2: Dirichlet smoothing

Can we consider collection language model for additive smoothing?

$$P(t|\theta_D) = \frac{tf_{t,D} + uP(t|\theta_C)}{N_D + u}$$

$u = 2500$ in Lemur by default

This is Dirichlet smoothing

Linear Interpolation Smoothing

- Let θ_D denote the language model associated with document *D*
- Let θ_C denote the language model associated with the entire collection
- Using linear interpolation, the probability given by the document language model to term *t* is:

$$P(t|D) = \alpha P(t|\theta_D) + (1 - \alpha) P(t|\theta_C)$$

Linear Interpolation Smoothing

$$P(t|D) = \alpha P(t|\theta_D) + (1 - \alpha)P(t|\theta_C)$$

the probability given to the term by the document language model

the probability given to the term by the collection language model

Linear Interpolation Smoothing

$$P(t|D) = \alpha P(t|\theta_D) + (1 - \alpha)P(t|\theta_C)$$

every one of these numbers
is between 0 and 1, so P(t|D)
is between 0 and 1

Query Likelihood Retrieval Model

with linear interpolation smoothing

- As before, a document's score is given by the probability that it "generated" the query
- As before, this is given by multiplying the individual query-term probabilities
- However, the probabilities are obtained using the linearly interpolated language model

$$score(Q, D) = \prod_{i=1}^{n} (\alpha P(q_i|\theta_D) + (1 - \alpha) P(q_i|\theta_C))$$

Method 3: Linear Interpolation Smoothing (Jelinek-Mercer smoothing)

$$P(t|D) = \alpha \underbrace{P(t|\theta_D)} + (1-\alpha)\underbrace{P(t|\theta_C)}$$

0<α<1

the probability given to the term by the **document language model**

$$P(t|\theta_D) = \frac{tf_{t,D}}{N_D}$$

the probability given to the term by the **collection language model**

counts of t in the whole corpus

$$P(t|\theta_C) = \frac{tf_{t,C}}{N_C}$$

corpus size

Query Likelihood Retrieval Model

with linear interpolation smoothing

- Linear interpolation helps us avoid zero-probabilities
- Remember, because we're multiplying probabilities, if a document is missing a single query-term it will be given a score of zero!
- Linear interpolation smoothing has another added benefit, though it's not obvious
- Let's start with an example

Query Likelihood Retrieval Model

no smoothing

- Query: apple ipad
- Two documents (D_1 and D_2), each with 50 term occurrences

	D_1 $(N_{D1}=50)$	D_2 $(N_{D2}=50)$
apple	$2/50 = 0.04$	$3/50 = 0.06$
ipad	$3/50 = 0.06$	$2/50 = 0.04$
score	$(0.04 \times 0.06) = 0.0024$	$(0.06 \times 0.04) = 0.0024$

Query Likelihood Retrieval Model

no smoothing

- Query: apple ipad
- Two documents (D_1 and D_2), each with 50 term occurrences

	D_1 $(N_{D1}=50)$	D_2 $(N_{D2}=50)$
apple	2/50 = 0.04	3/50 = 0.06
ipad	3/50 = 0.06	2/50 = 0.04
score	(0.04 x 0.06) = 0.0024	(0.06 x 0.04) = 0.0024

- Which query-term is more important: apple or ipad?

Query Likelihood Retrieval Model

no smoothing

- A term is descriptive of the document if it occurs many times in the document
- But, not if it occurs many times in the document and also occurs frequently in the collection

Query Likelihood Retrieval Model

no smoothing

- Query: apple ipad
- Two documents (D_1 and D_2), each with 50 term occurrences

	D_1 $(N_{D1}=50)$	D_2 $(N_{D2}=50)$
apple	2/50 = 0.04	3/50 = 0.06
ipad	3/50 = 0.06	2/50 = 0.04
score	(0.04 x 0.06) = 0.0024	(0.06 x 0.04) = 0.0024

- Without smoothing, the query-likelihood model ignores how frequently the term occurs in general!

Query Likelihood Retrieval Model

with linear interpolation smoothing

- Suppose the corpus has 1,000,000 word-occurrences
- apple occurs 200 / 1,000,000 times
- ipad occurs 100 / 1,000,000 times
- Therefore:

$$P(\text{apple}|\theta_C) = \frac{200}{1000000} = 0.0002$$

$$P(\text{ipad}|\theta_C) = \frac{100}{1000000} = 0.0001$$

Query Likelihood Retrieval Model

with linear interpolation smoothing

$$score(Q, D) = \prod_{i=1}^{n} (\alpha P(q_i|\theta_D) + (1 - \alpha)P(q_i|\theta_C))$$

	D_1 $(N_{D1}=50)$	D_2 $(N_{D2}=50)$
P(apple|D)	0.04	0.06
P(apple|C)	0.0002	0.0002
score(apple)	0.0201	0.0301
P(ipad|D)	0.06	0.04
P(ipad|C)	0.0001	0.0001
score(ipad)	0.03005	0.02005
total score	0.000604005	0.000603505

$\alpha=0.50$

Query Likelihood Retrieval Model

with linear interpolation smoothing

- Linear interpolation smoothing does not only avoid zero probabilities ...
- It also introduces an IDF-like scoring of documents
  - terms are that are less frequent in the entire collection have a higher contribution to a document's score
- Yes, but we've only seen an example. Where is the mathematical proof!?

Mathematical Proof

out of the scope of this class

$$
\begin{aligned}
p(q \mid d) &= \prod_{q_i \in q} p(q_i \mid d) \\
&= \prod_{q_i \in q} \left(\lambda p_{MLE}(q_i \mid d) + (1-\lambda) p_{MLE}(q_i \mid C)\right) && \textbf{Mixture model} \\
&= \prod_{q_i \in q} \left(\lambda p_{MLE}(q_i \mid d) + (1-\lambda) p_{MLE}(q_i \mid C)\right) \frac{(1-\lambda) p_{MLE}(q_i \mid C)}{(1-\lambda) p_{MLE}(q_i \mid C)} && \textbf{Multiply by 1} \\
&= \prod_{q_i \in q} \left(\left(\frac{\lambda p_{MLE}(q_i \mid d)}{(1-\lambda) p_{MLE}(q_i \mid C)} + 1 \right) (1-\lambda) p_{MLE}(q_i \mid C) \right) && \textbf{Recombine} \\
&= \prod_{q_i \in q} \left(\frac{\lambda p_{MLE}(q_i \mid d)}{(1-\lambda) p_{MLE}(q_i \mid C)} + 1 \right) \prod_{q_i \in q} (1-\lambda) p_{MLE}(q_i \mid C) && \textbf{Recombine} \\
&\propto \prod_{q_i \in q} \left(\frac{\lambda p_{MLE}(q_i \mid d)}{(1-\lambda) p_{MLE}(q_i \mid C)} + 1 \right) && \textbf{Drop constant}
\end{aligned}
$$

“tf”

“idf”