

Language Model

What is a language model ?

- Probability distribution over strings of text
 - how likely is a given string (observation) in a given “language”
 - for example, consider probability for the following four strings
 - $p1 = P(\text{“a quick brown dog”})$
 - $p2 = P(\text{“dog quick a brown”})$
 - $p3 = P(\text{“быстрая brown dog”})$
 - $p4 = P(\text{“быстрая собака”})$

English: $p1 > p2 > p3 > p4$

- ... depends on what “language” we are modeling
 - In most of IR, assume that $p1 == p2$

Language models

- estimate probabilities of certain "events" in the text
- based on these probabilities, use likelihood as similarity
- language model based on
 - letters?
 - words?
 - phrases?

Statistical text generation

1. *Zero-order approximation.* (The symbols are independent and equiprobable.)

XFOML RXKHRJFFJUJ ZLPWCFWKCYJ

FFJEYVKCQSGXYD QPAAMKBZAACIBZLHJQD

2. *First-order approximation.* (The symbols are independent. Frequency of letters matches English text.)

OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI

ALHENHTTPA OOBTTVA NAH BRL

3. *Second-order approximation.* (The frequency of pairs of letters matches English text.)

ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY

ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO

TIZIN ANDY TOBE SEACE CTISBE

4. *Third-order approximation.* (The frequency of triplets of letters matches English text.)

IN NO IST LAT WHEY CRATICT FROURE BERS GROCID

PONDENOME OF DEMONSTURES OF THE REPTAGIN IS

REGOACTIONA OF CRE

Statistical text generation

5. *Fourth-order approximation.* (The frequency of quadruplets of letters matches English text. Each letter depends on the previous three letters. This sentence is from Lucky's book, *Silicon Dreams* [183].)

THE GENERATED JOB PROVIDUAL BETTER TRAND THE
DISPLAYED CODE, ABOVERY UPONDULTS WELL THE
CODERST IN THESTICAL IT DO HOCK BOTHE MERG.
(INSTATES CONS ERATION. NEVER ANY OF PUBLE AND TO
THEORY. EVENTIAL CALLEGAND TO ELAST BENERATED IN
WITH PIES AS IS WITH THE)

Instead of continuing with the letter models, we jump to word models.

6. *First-order word model.* (The words are chosen independently but with frequencies as in English.)

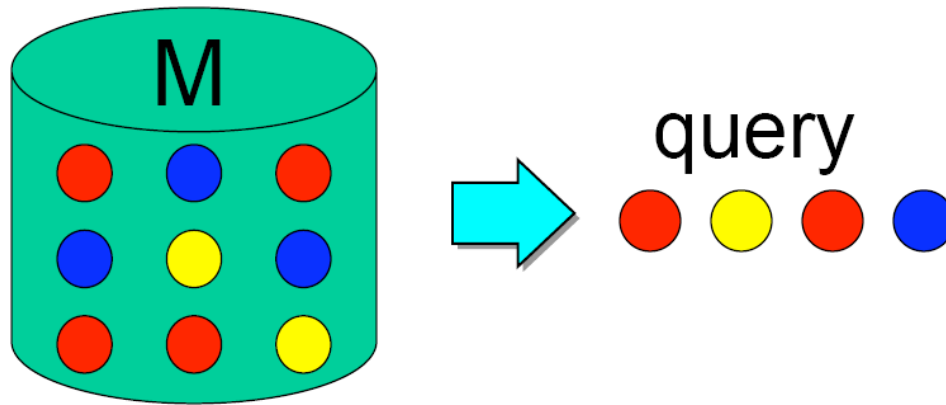
REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME
CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO
OF TO EXPERT GRAY COME TO FURNISHES THE LINE
MESSAGE HAD BE THESE.

7. *Second-order word model.* (The word transition probabilities match English text.)

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH
WRITER THAT THE CHARACTER OF THIS POINT IS
THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE
TIME OF WHO EVER TOLD THE PROBLEM FOR AN
UNEXPECTED

Unigram LM

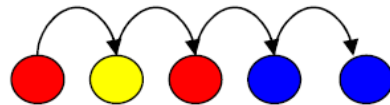
- words are sampled independently, with replacement
- order of the words $P(q_1 \dots q_k | M) = \prod_{i=1}^k P(q_i | M)$



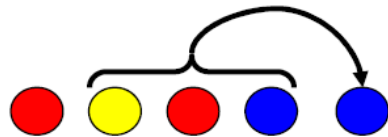
$$\begin{aligned} P(\text{red} \text{ yellow} \text{ red} \text{ blue}) &= P(\text{red}) P(\text{yellow}) P(\text{red}) P(\text{blue}) \\ &= 4/9 * 2/9 * 4/9 * 3/9 \end{aligned}$$

Higher-order LM

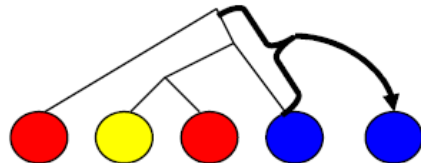
- Unigram model assumes word independence
 - cannot capture surface form: $P(\text{"brown dog"}) \neq P(\text{"dog brown"})$
- Higher-order models
 - n-gram: condition on preceding words



- cache: condition on a window (cache)



- grammar: condition on parse tree



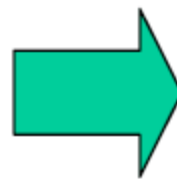
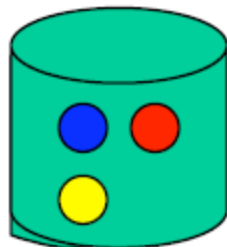
Higher-order LM

- Are they useful?
 - no improvements from n-gram, grammar-based models
 - some research on cache-like models (proximity, passages, etc.)

Maximum likelihood

- count relative frequencies of words in S
- maximum-likelihood property:
 - assigns highest possible likelihood to the observation

$$P_{ml}(w|M_S) = \#(w, S) / |S|$$



$$P(\text{blue}) = 1/3$$

$$P(\text{red}) = 1/3$$

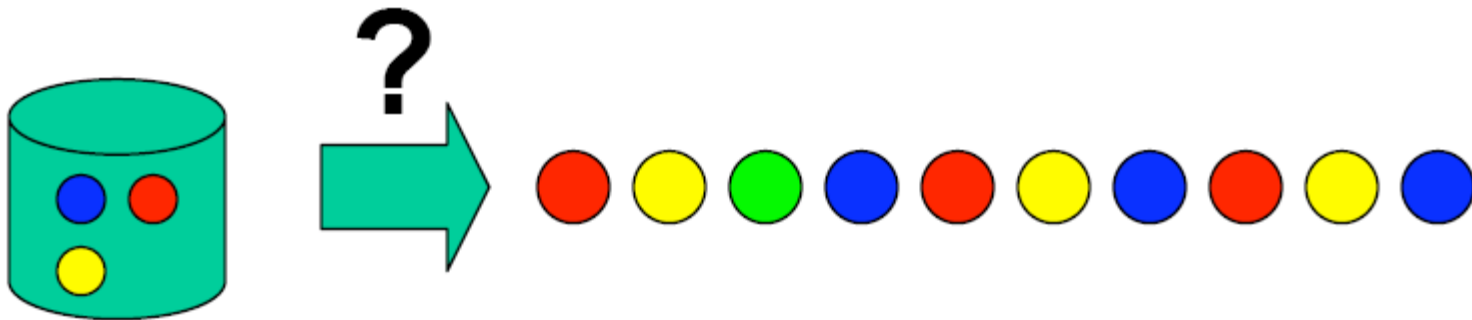
$$P(\text{yellow}) = 1/3$$

$$P(\text{green}) = 0$$

$$P(\text{grey}) = 0$$

Zero-frequency problem

- Suppose some event not in our observation S
 - Model will assign zero probability to that event
 - And to any set of events involving the unseen event
- Happens very frequently with language
- It is incorrect to infer zero probabilities
 - especially when creating a model from short samples



Laplace smoothing

- Counts events in observed data
- Add 1 to every count
- Renormalize to obtain probabilities
- It corresponds to uniform priors
- If even counts are (m_1, m_2, \dots, m_k) with $\sum_i m_i = N$
- Then maximum likelihood estimates are $\frac{m_1}{N}, \frac{m_2}{N}, \dots, \frac{m_k}{N}$
- Laplace estimates are $\frac{m_1 + 1}{N + k}, \frac{m_2 + 1}{N + k}, \dots, \frac{m_k + 1}{N + k}$

Laplace smoothing

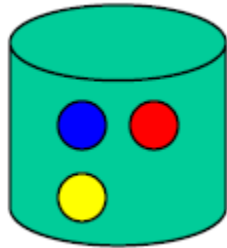
- Probability that sun will rise tomorrow
- $= (4.5 \text{ billion years} + 1) / (4.5 \text{ billion years} + 2)$
- $= (16425 * 10^{12} + 1) / (16425 * 10^{12} + 2)$
- $= 99.999999999994 \%$

Interpolation methods

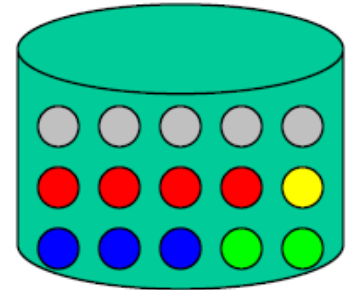
- Problem with all discounting methods:
 - discounting treats unseen words equally (add or subtract ϵ)
 - some words are more frequent than others
- Idea: use background probabilities
 - “interpolate” ML estimates with General English expectations
- (computed as relative frequency of a word in a large collection)
 - reflects expected frequency of events

Interpolation methods

ML estimate =

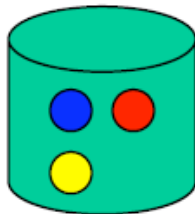


Background Probability=



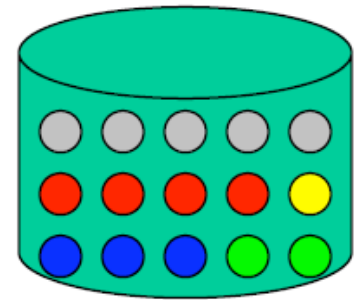
final estimate =

λ



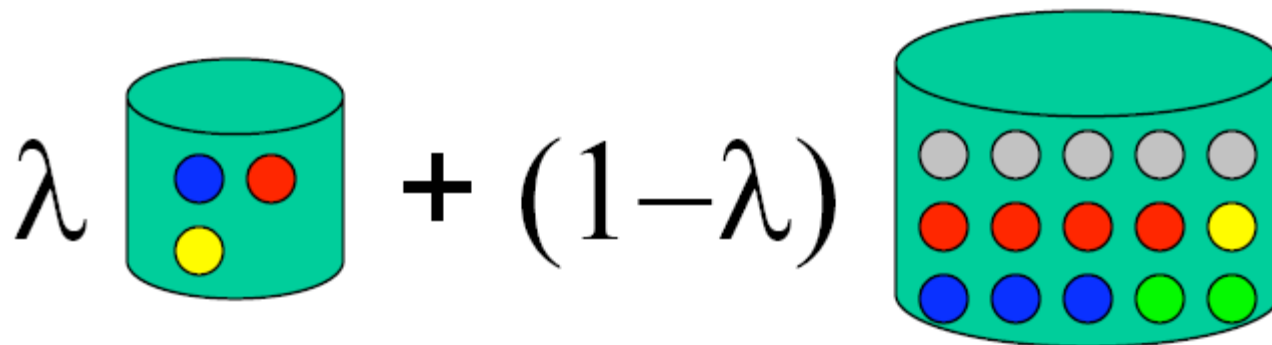
+

$(1-\lambda)$



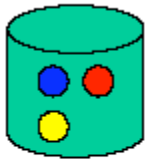
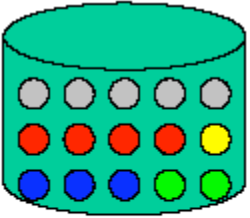
Jelinek Mercer smoothing

- Correctly setting λ is very important
- Start simple
 - set λ to be a constant, independent of document, query
- Tune to optimize retrieval performance
 - optimal value of λ varies with different collections, query sets, etc.



Dirichlet smoothing

- Problem with Jelinek-Mercer:
 - longer documents provide better estimates
 - could get by with less smoothing
- Make smoothing depend on sample size
- N is length of sample = document length
- μ is a constant

$$\underbrace{N / (N + \mu)}_{\lambda} \text{  } + \underbrace{\mu / (N + \mu)}_{(1-\lambda)} \text{  }$$

Witten-Bell smoothing

- A step further:
 - condition smoothing on “redundancy” of the example
 - long, redundant example requires little smoothing
 - short, sparse example requires a lot of smoothing
- Derived by considering the proportion of new events as we walk through example
 - N is total number of events = document length
 - V is number of unique events = number of unique terms in doc

$$\underbrace{N / (N + V)}_{\lambda} \text{ } \text{cylinder with 3 colored dots} + \underbrace{V / (N + V)}_{(1-\lambda)} \text{ } \text{cylinder with 12 colored dots}$$

The diagram illustrates the Witten-Bell smoothing formula. It shows two cylinders representing event sets. The first cylinder, associated with the term λ , contains 3 colored dots (blue, red, yellow). The second cylinder, associated with the term $(1-\lambda)$, contains 12 colored dots (4 red, 4 blue, 2 green, 2 yellow, and 2 grey). The formula is $\lambda \cdot \text{cylinder with 3 colored dots} + (1-\lambda) \cdot \text{cylinder with 12 colored dots}$.

Document 1	Shares in apple company stock prices rise
Document 2	An apple a day keeps the doctor away
Document 3	Apple health benefits apple fruits mango
Query	apple company
Stopwords	the, in, a,

LM: summary

- Goal: estimate a model M from a sample text S
- Use maximum-likelihood estimator
 - count the number of times each word occurs in S , divide by length
- Smoothing to avoid zero frequencies
 - 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)

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

Slide Credits

- Javed Aslam, Northeastern University
- Christopher Manning and Prabhakar Raghavan, Stanford University