# Statistical Language Models

Hongning Wang

CS@UVa

# Today's lecture

1. How to represent a document?
    – Make it computable

2. How to infer the relationship among documents or identify the structure within a document?

    – Knowledge discovery

# What is a statistical LM?

- A model specifying probability distribution over <u>word sequences</u>
  - p(*"Today is Wednesday"*) ≈ 0.001
  - p(*"Today Wednesday is"*) ≈ 0.0000000000001
  - p(*"The eigenvalue is positive"*) ≈ 0.00001
- It can be regarded as a probabilistic mechanism for "generating" text, thus also called a "generative" model

# Why is a LM useful?

- Provide a principled way to quantify the uncertainties associated with natural language

- Allow us to answer questions like:
  - Given that we see "*John*" and "*feels*", how likely will we see "*happy*" as opposed to "*habit*" as the next word?
    (speech recognition)
  - Given that we observe "baseball" three times and "game" once in a news article, how likely is it about "sports" v.s. "politics"?
    (text categorization)
  - Given that a user is interested in sports news, how likely would the user use "baseball" in a query?
    (information retrieval)

# Measure the fluency of documents

- How likely this document is generated by a given language model
  - If $p_{text-mining}(d) > p_{health}(d)$, $d$ belongs to text mining related documents
  - If $p_{user_a}(d_1) > p_{user_a}(d_2)$, recommend $d_1$ to $user_a$
  - A <u>relative</u> concept

# Source-Channel framework [Shannon 48]
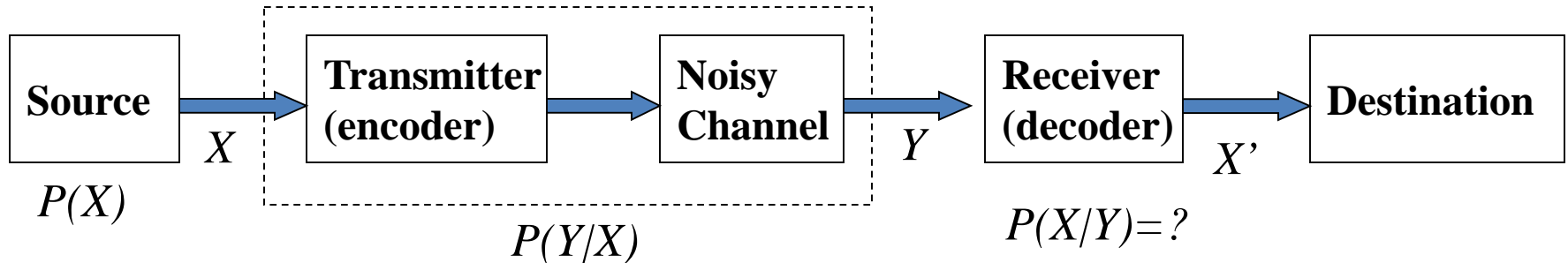


$$\hat{X} = \arg\max_{X} \; p(X \mid Y) = \arg\max_{X} \; p(Y \mid X)p(X) \quad \textbf{(Bayes Rule)}$$

**When X is text, p(X) is a language model**

## Many Examples:

| | | |
|---|---|---|
| **Speech recognition:** | **X=Word sequence** | **Y=Speech signal** |
| **Machine translation:** | **X=English sentence** | **Y=Chinese sentence** |
| **OCR Error Correction:** | **X=Correct word** | **Y= Erroneous word** |
| **Information Retrieval:** | **X=Document** | **Y=Query** |
| **Summarization:** | **X=Summary** | **Y=Document** |

# Basic concepts of probability

- **Random experiment**
  - An experiment with uncertain outcome (e.g., tossing a coin, picking a word from text)

- **Sample space (S)**
  - All possible outcomes of an experiment, e.g., tossing 2 fair coins, S={HH, HT, TH, TT}

- **Event (E)**
  - $E \subseteq S$, E happens iff outcome is in S, e.g., E={HH} (all heads), E={HH,TT} (same face)
  - Impossible event ({}), certain event (S)

- **Probability of event**
  - $0 \leq P(E) \leq 1$

# Sampling with replacement
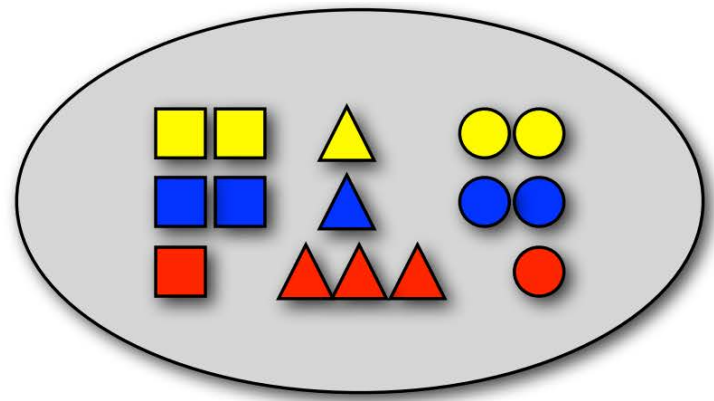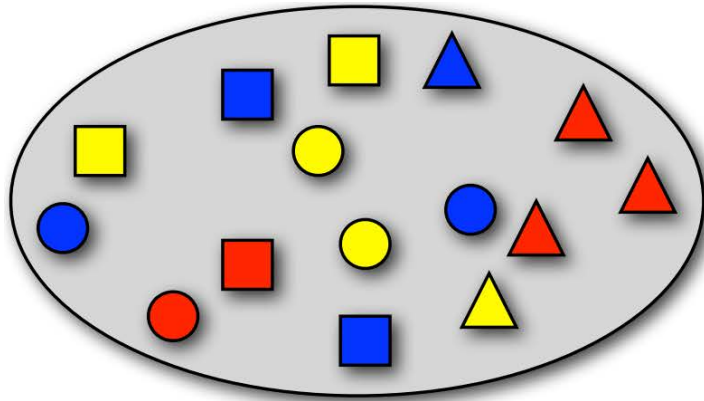
- Pick a random shape, then put it back in the bag



$P(\blacksquare)$ $= 2/15$

$P(\text{blue}) = 5/15$

$P(\text{blue} | \square) = 2/5$

$P(\blacksquare)$ $= 1/15$

$P(\text{red})$ $= 5/15$

$P(\square)$ $= 5/15$

$P(\blacksquare \text{ or} \blacktriangle) = 2/15$

$P(\triangle | \text{red}) = 3/5$

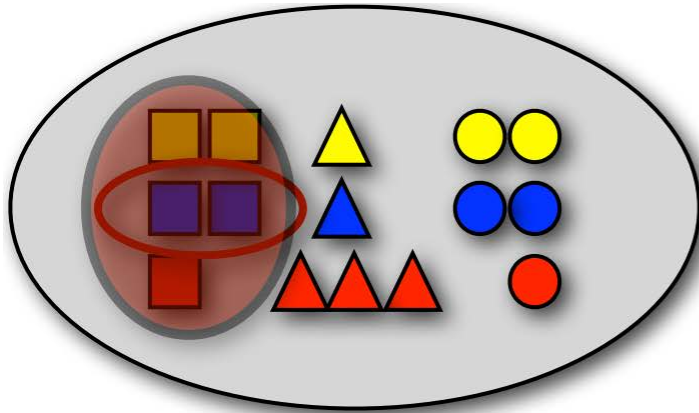# Essential probability concepts

- Probability of events
  - Mutually exclusive events
    - $P(A \cup B) = P(A) + P(B)$
  - General events
    - $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
  - Independent events
    - $P(A \cap B) = P(A)P(B)$

Joint probability, or simply as $P(A, B)$

# Essential probability concepts

- Conditional probability
  - $P(B|A) = P(A,B)/P(A)$
  - Bayes' Rule: $P(B|A) = P(A|B)P(B)/P(A)$
  - For independent events, $P(B|A) = P(B)$

$$P(\text{blue} \mid \blacksquare) = 2/5$$

# Language model for text

*We need independence assumptions!*

- A probability distribution over <u>word sequences</u>
  - $p(w_1\ w_2\ \ldots\ w_n) =$
    $p(w_1)p(w_2|w_1)p(w_3|w_1,w_2)\ldots p(w_n|w_1,w_2,\ldots,w_{n-1})$
  - Complexity - $O(V^{n^*})$
    - $n^*$ - maximum ~~document~~ length sentence

Chain rule: from conditional probability to joint probability

- 475,000 main headwords in Webster's Third New International Dictionary
- Average English sentence length is 14.3 words
- A rough estimate: $O(475000^{14})$

How large is this?  $\dfrac{475000^{14}}{8bytes \times (1024)^4} \approx 3.38e^{66}TB$

# Unigram language model

- Generate a piece of text by generating each word <u>independently</u>

  $$- p(w_1\, w_2\, \ldots\, w_n) = p(w_1)p(w_2)\ldots p(w_n)$$

  $$- s.t.\, \{p(w_i)\}_{i=1}^{N}\,, \sum_i p(w_i) = 1\,, p(w_i) \geq 0$$

- Essentially a multinomial distribution over the vocabulary

*The simplest and most popular choice!*

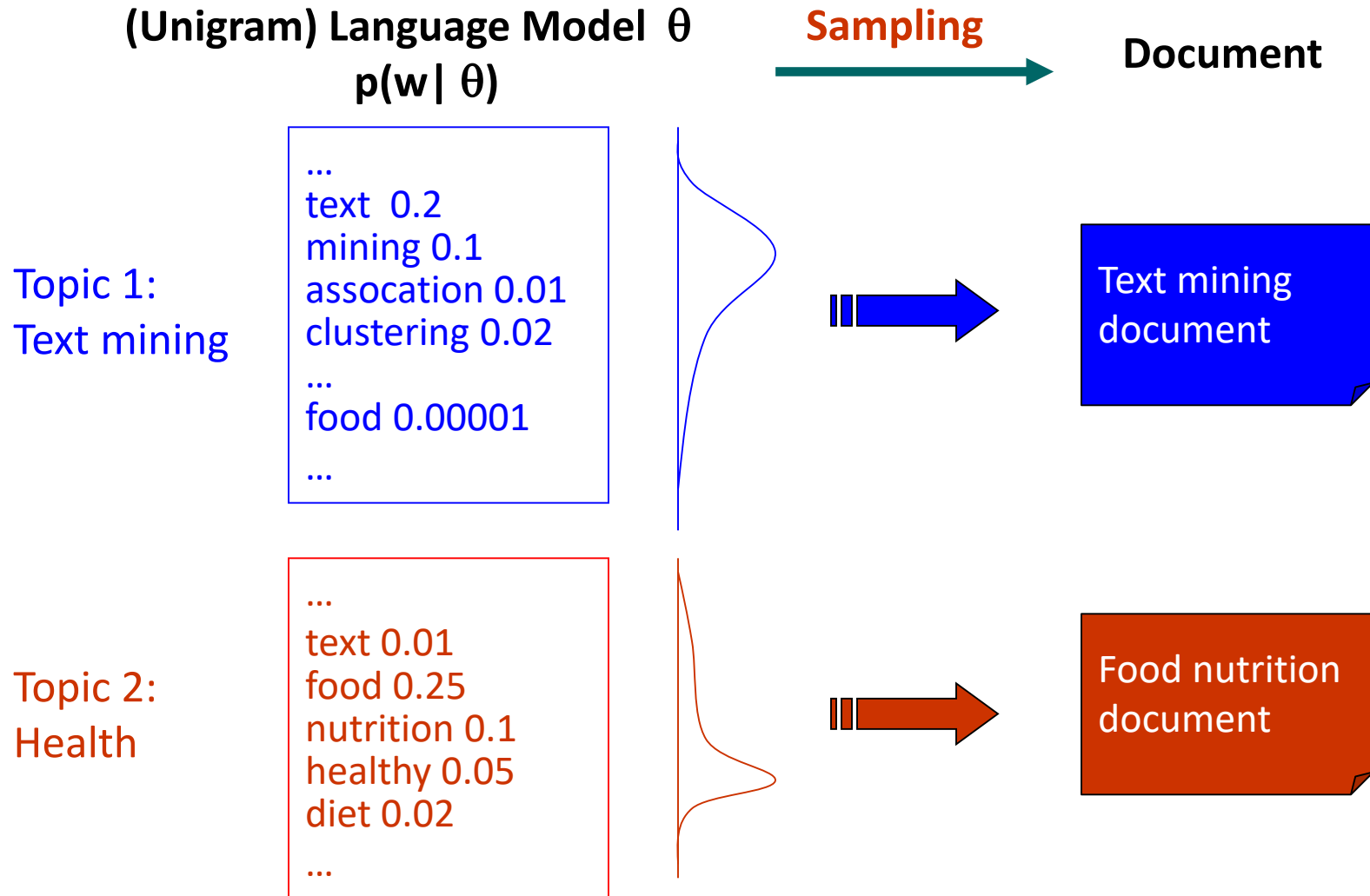A Unigram Language Model

CS 6501: Text Mining
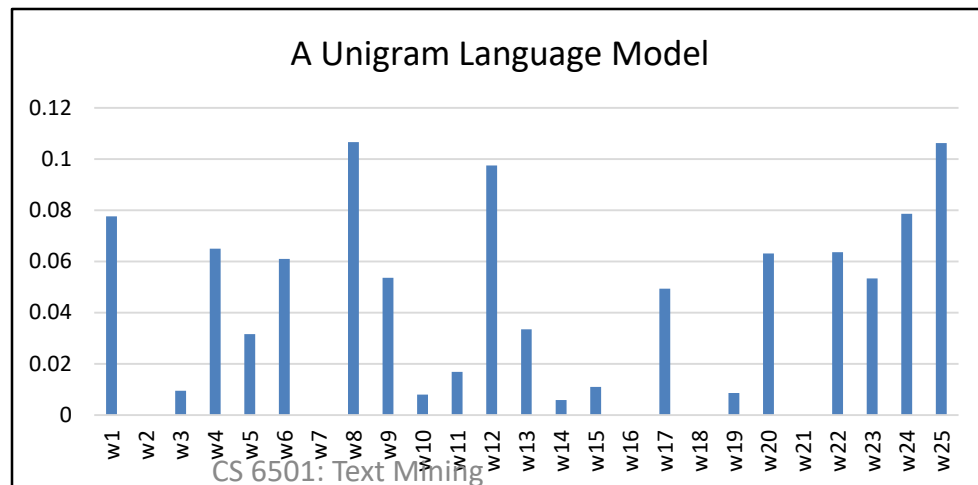
# More sophisticated LMs

- N-gram language models
  - Conditioned only on the past n-1 words
  - E.g., bigram: $p(w_1 \ \dots \ w_n) = p(w_1)p(w_2|w_1) \ p(w_3|w_2) \ \dots p(w_n|w_{n-1})$

- Remote-dependence language models (e.g., Maximum Entropy model)

- Structured language models (e.g., probabilistic context-free grammar)

# Why just unigram models?

- Difficulty in moving toward more complex models
  - They involve more parameters, so need more data to estimate
  - They increase the computational complexity significantly, both in time and space
- Capturing word order or structure may not add so much value for "topical inference"
- But, using more sophisticated models can still be expected to improve performance …

# Generative view of text documents

**(Unigram) Language Model θ**
**p(w| θ)**

**Sampling**

**Document**

Topic 1:
Text mining

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

Text mining document

Topic 2:
Health
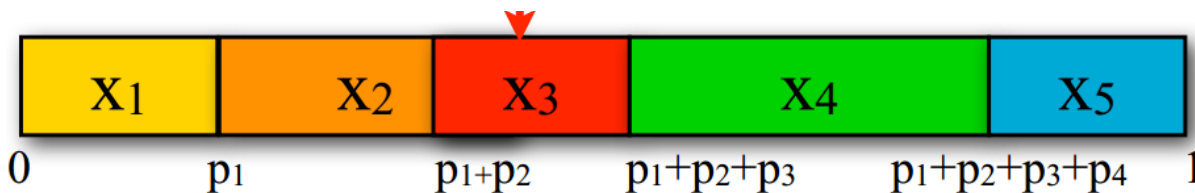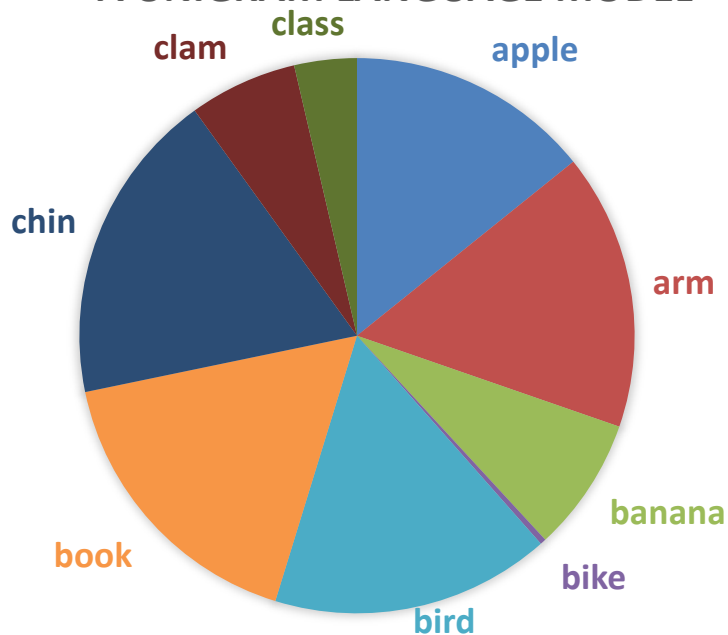
…
text 0.01
food 0.25
nutrition 0.1
healthy 0.05
diet 0.02
…

Food nutrition document

# How to generate text from an N-gram language model?

- Sample from a discrete distribution $p(X)$

**A UNIGRAM LANGUAGE MODEL**

# Generating text from language models

$P(\text{of}) = 3/66$

$P(\text{Alice}) = 2/66$

$P(\text{was}) = 2/66$

$P(\text{to}) = 2/66$

$P(\text{her}) = 2/66$

$P(\text{sister}) = 2/66$

$P(,) = 4/66$

$P(') = 4/66$

**Under a unigram language model:**

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

# Generating text from language models

$P(\text{of}) = 3/66$

$P(\text{Alice}) = 2/66$

$P(\text{was}) = 2/66$

$P(\text{to}) = 2/66$

$P(\text{her}) = 2/66$

$P(\text{sister}) = 2/66$

$P(,) = 4/66$

$P(') = 4/66$

**Under a unigram language model:**

**The same likelihood!**

beginning by, very Alice but was and?
reading no tired of to into sitting
sister the, bank, and thought of without
her nothing: having conversations Alice
once do or on she it get the book her had
peeped was conversation it pictures or
sister in, 'what is the use had twice of
a book''pictures or' to

# N-gram language models will help

- Unigram

  *Generated from language models of New York Times*

  - Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a q acquire to six executives.

- Bigram

  - Last December through the way to preserve the Hudson corporation N.B.E.C. Taylor would seem to complete the major central planners one point five percent of U.S.E. has already told M.X. corporation of living on information such as more frequently fishing to keep her.

- Trigram

  - They also point to ninety nine point six billon dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions.

# Turing test: generating Shakespeare

| | |
|---|---|
| A | • To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have<br>• Every enter now severally so, let<br>• Hill he late speaks; or! a more to leg less first you enter<br>• Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like |
| B | • What means, sir. I confess she? then all sorts, he is trim, captain.<br>•Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.<br>•What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first~ ~~ ~~~ ~<br>•E~ SCIgen - An Automatic CS Paper Generator :om-<br>mand of fear not a liberal largess given away, Falstaff! Exeunt |
| C | • Sweet prince, Falstaff shall die. Harry of Monmouth's grave.<br>• This shall forbid it should be branded, if renown made it empty.<br>• Indeed the duke; and had a very good friend.<br>• Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. |
| D | • King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;<br>• Will you not tell me who I am?<br>• It cannot be but so.<br>• Indeed the short and the long. Marry, 'tis a noble Lepidus. |

# Estimation of language models

**Unigram Language Model** $\theta$
$p(w|\theta)=?$

**Estimation**

**Document**
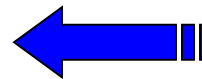
...
text ?
mining ?
assocation ?
database ?
...
query ?
...

text 10
mining 5
association 3
database 3
algorithm 2
...
query 1
efficient 1

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

# Sampling with replacement
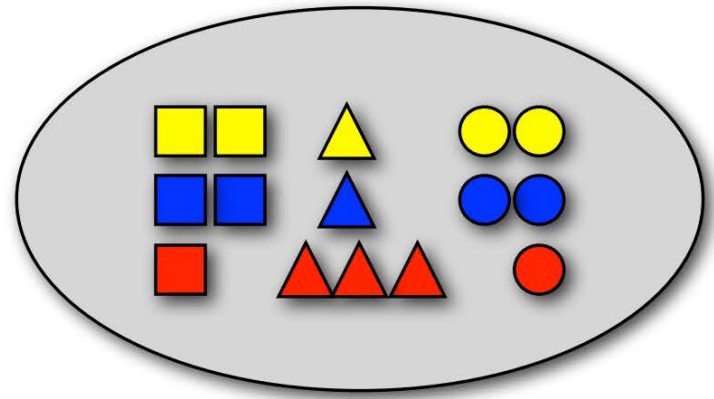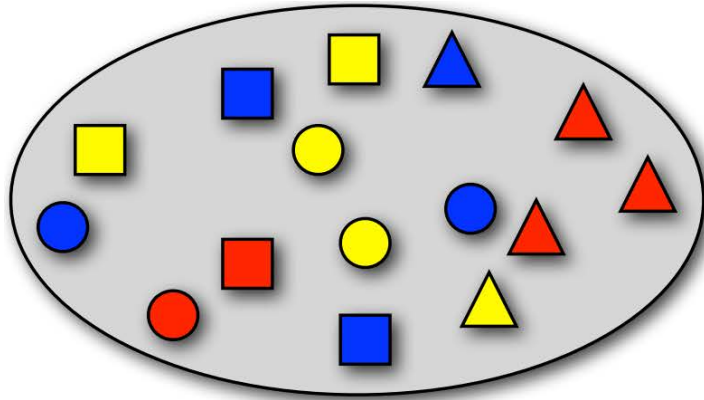
- Pick a random shape, then put it back in the bag



P(■) = 2/15
P(blue) = 5/15
P(blue |□) = 2/5

P(■) = 1/15
P(red) = 5/15
P(□) = 5/15

P(■ or ▲) = 2/15
P(△ |red) = 3/5

# Parameter estimation

- General setting:
  - Given a (hypothesized & probabilistic) model that governs the random experiment
  - The model gives a probability of any data $p(X|\theta)$ that depends on the parameter $\theta$
  - Now, given actual sample data X={$x_1$,...,$x_n$}, what can we say about the value of $\theta$?

- Intuitively, take our best guess of $\theta$ -- "best" means "best explaining/fitting the data"

- Generally an optimization problem

# Maximum likelihood vs. Bayesian

- Maximum likelihood estimation
  - "Best" means "data likelihood reaches maximum"
  $$\widehat{\boldsymbol{\theta}} = \mathbf{argmax}_{\theta} \mathbf{P}(\mathbf{X}|\boldsymbol{\theta})$$
  - Issue: small sample size

  *ML: Frequentist's point of view*

- Bayesian estimation
  - "Best" means being consistent with our "prior" knowledge and explaining data well
  $$\widehat{\boldsymbol{\theta}} = \mathbf{argmax}_{\theta} \boldsymbol{P}(\boldsymbol{\theta}|X) = \mathbf{argmax}_{\theta} \mathbf{P}(\mathbf{X}|\boldsymbol{\theta})\mathbf{P}(\boldsymbol{\theta})$$
  - A.k.a, Maximum a Posterior estimation
  - Issue: how to define prior?

  *MAP: Bayesian's point of view*

# Illustration of Bayesian estimation

Posterior:
$p(\theta|X) \propto p(X|\theta)p(\theta)$

Likelihood:
$p(X|\theta)$ $X=(x_1,...,x_N)$

Prior: $p(\theta)$

$\theta_\alpha$: prior mode

$\theta$: posterior mode

$\theta_{ml}$: ML estimate

$\theta$

# Maximum likelihood


$y = \log_2(x)$
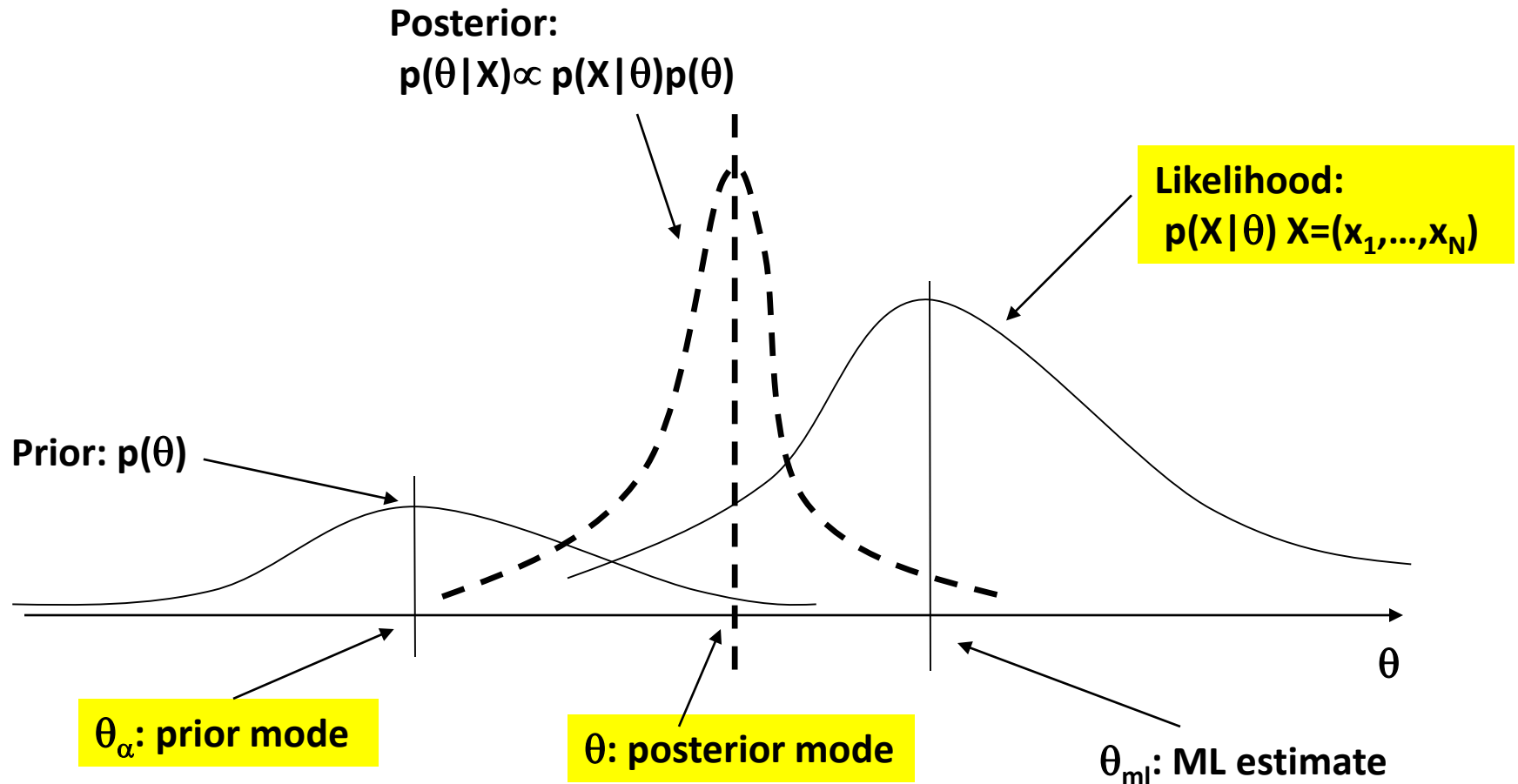
- Data: a collection of words, $w_1, w_2, \dots$
- Model: multinomial distribution $p(W$ 
  $p(w_i)$
- Maximum likelihood estimator: $\hat{\theta} = argmax_\theta\, p(W|\theta)$
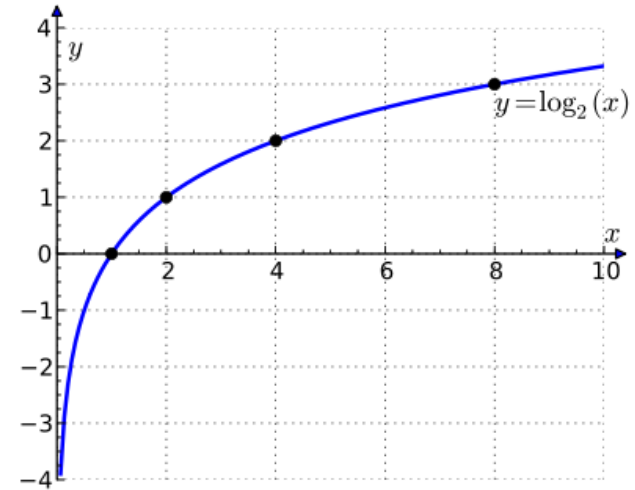
$$p(W|\theta) = \binom{N}{c(w_1), \dots, c(w_N)} \prod_{i=1}^{N} \theta_i^{c(w_i)} \propto \prod_{i=1}^{N} \theta_i^{c(w_i)} \implies \log p(W|\theta) = \sum_{i=1}^{N} c(w_i) \log \theta_i$$

$$\implies L(W, \theta) = \sum_{i=1}^{N} c(w_i) \log \theta_i + \lambda\left(\sum_{i=1}^{N} \theta_i - 1\right)$$

**Using Lagrange multiplier approach, we'll tune $\theta_i$ to maximize $L(W, \theta)$**

$$\implies \frac{\partial L}{\partial \theta_i} = \frac{c(w_i)}{\theta_i} + \lambda \quad \rightarrow \quad \theta_i = -\frac{c(w_i)}{\lambda}$$

**Set partial derivatives to zero**

$$\implies \text{Since } \sum_{i=1}^{N} \theta_i = 1 \text{ we have } \quad \lambda = -\sum_{i=1}^{N} c(w_i)$$

**Requirement from probability**

$$\implies \theta_i = \frac{c(w_i)}{\sum_{i=1}^{N} c(w_i)}$$

**ML estimate**

CS 6501: Text Mining

# Maximum likelihood estimation

- For N-gram language models
    - $p(w_i | w_{i-1}, \ldots, w_{i-n+1})$
        - $c(\emptyset) = N$

    *Length of document or total number of words in a corpus*

# Problem with MLE

- Unseen events
  - We est... tokens, but:
    - Onl...
    - O... ...ed
  - ➤ This ... es not occur in th... ty!
  - ➤ No tut... ...se unseen words

$$f(k; s, N) \propto 1/k^s$$
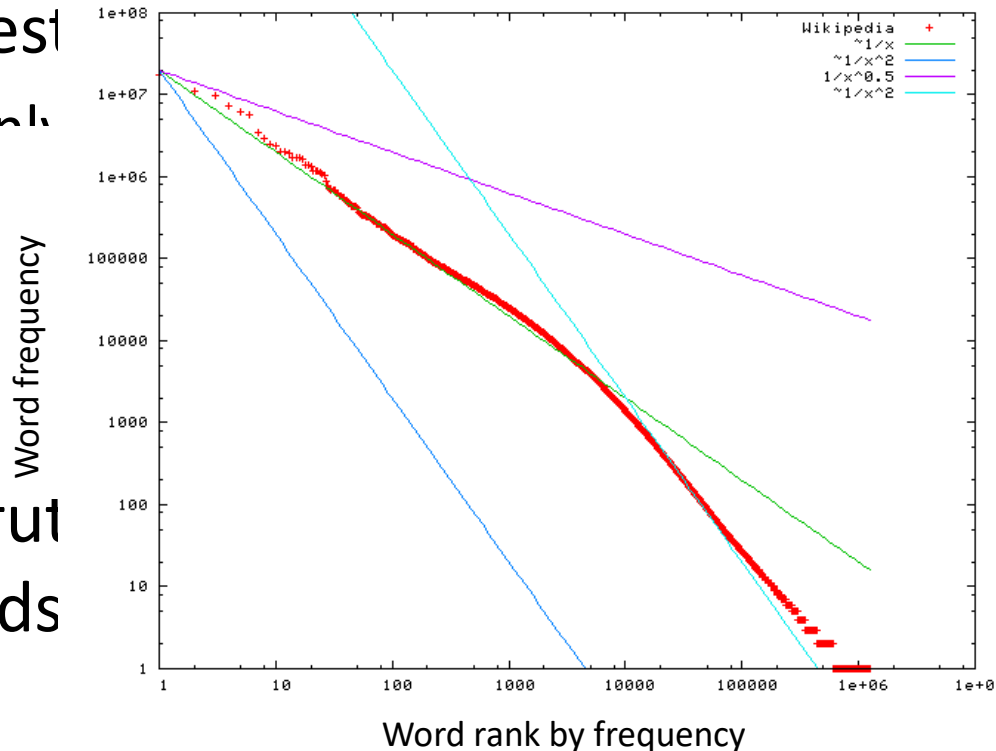


A plot of word frequency in Wikipedia (Nov 27, 2006)

# Smoothing

- If we want to assign non-zero probabilities to unseen words
  - Unseen words = new words, new N-grams
  - Discount the probabilities of observed words
- General procedure
  1. Reserve some probability mass of words seen in a document/corpus
  2. Re-allocate it to unseen words

# Illustration of N-gram language model smoothing

$p(w_i | w_{i-1}, \ldots, w_{i-n+1})$

**Max. Likelihood Estimate**

$$p(w_i | w_{i-1}, \ldots, w_{i-n+1}) = \frac{c(w_i, w_{i-1}, \ldots, w_{i-n+1})}{c(w_{i-1}, \ldots, w_{i-n+1})}$$

**Smoothed LM**

W

*Discount from the seen words*

*Assigning nonzero probabilities to the unseen words*
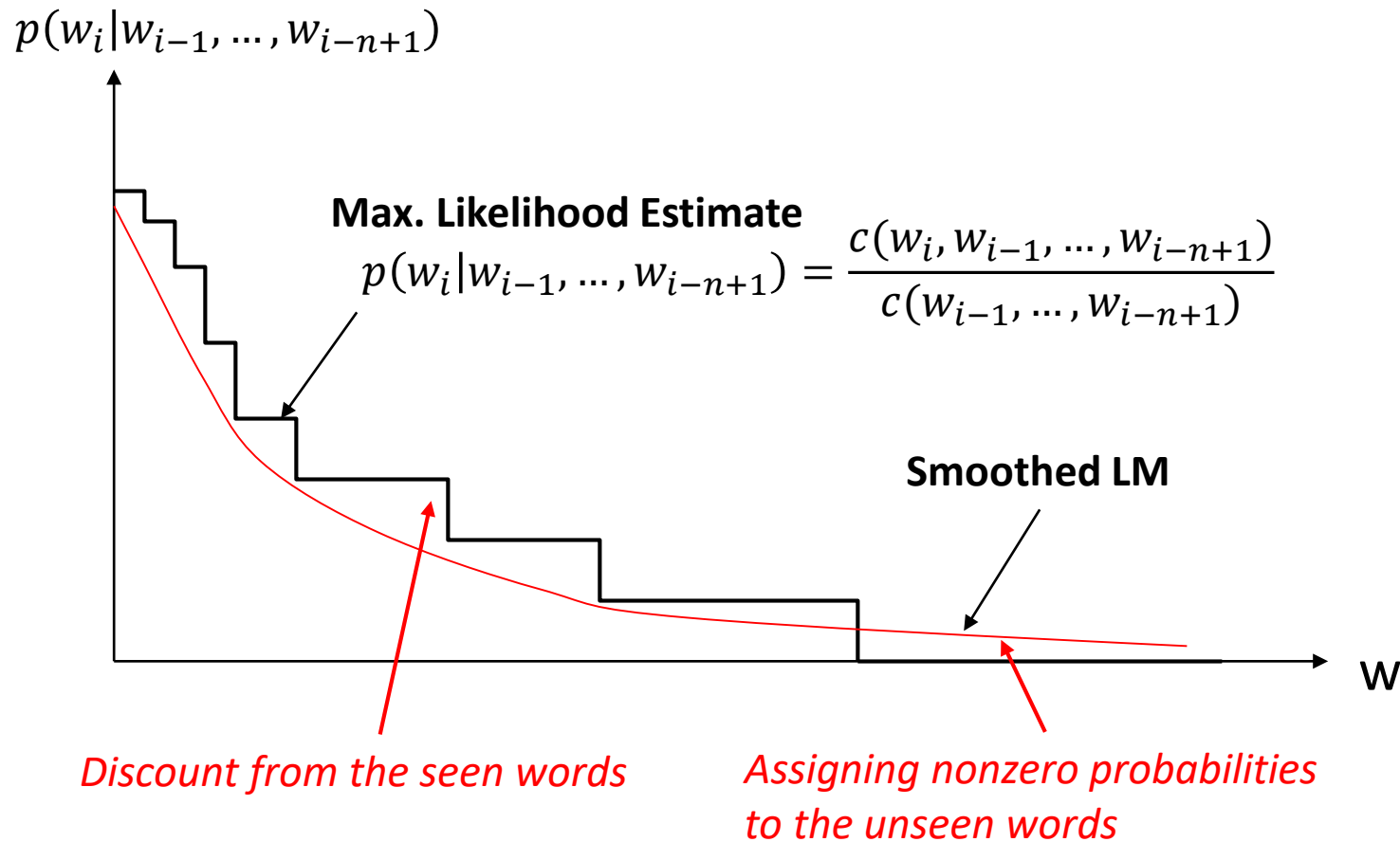
# Smoothing methods

- ## Additive smoothing
  - ## Add a constant $\delta$ to the counts of each word
    - Unigram language model as an example

**Counts of w in d**

**"Add one", Laplace smoothing**

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

**Vocabulary size**

↑

**Length of d (total counts)**

  - ## Problems?
    - Hint: all words are equally important?

# Add one smoothing for bigrams

Original:

|        | i  | want | to | eat | chinese | food | lunch | spend |
|--------|----|------|----|-----|---------|------|-------|-------|
| i      | 5  | 827  | 0  | 9   | 0       | 0    | 0     | 2     |
| want   | 2  | 0    | 608| 1   | 6       | 6    | 5     | 1     |
| to     | 2  | 0    | 4  | 686 | 2       | 0    | 6     | 211   |
| eat    | 0  | 0    | 2  | 0   | 16      | 2    | 42    | 0     |
| chinese| 1  | 0    | 0  | 0   | 0       | 82   | 1     | 0     |
| food   | 15 | 0    | 15 | 0   | 1       | 4    | 0     | 0     |
| lunch  | 2  | 0    | 0  | 0   | 0       | 1    | 0     | 0     |
| spend  | 1  | 0    | 1  | 0   | 0       | 0    | 0     | 0     |

Smoothed:

|        | i  | want | to | eat | chinese | food | lunch | spend |
|--------|----|------|----|-----|---------|------|-------|-------|
| i      | 6  | 828  | 1  | 10  | 1       | 1    | 1     | 3     |
| want   | 3  | 1    | 609| 2   | 7       | 7    | 6     | 2     |
| to     | 3  | 1    | 5  | 687 | 3       | 1    | 7     | 212   |
| eat    | 1  | 1    | 3  | 1   | 17      | 3    | 43    | 1     |
| chinese| 2  | 1    | 1  | 1   | 1       | 83   | 2     | 1     |
| food   | 16 | 1    | 16 | 1   | 2       | 5    | 1     | 1     |
| lunch  | 3  | 1    | 1  | 1   | 1       | 2    | 1     | 1     |
| spend  | 2  | 1    | 2  | 1   | 1       | 1    | 1     | 1     |

# After smoothing

- Giving too much to the unseen events

Original:

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 0.002 | 0.33 | 0 | 0.0036 | 0 | 0 | 0 | 0.00079 |
| want | 0.0022 | 0 | 0.66 | 0.0011 | 0.0065 | 0.0065 | 0.0054 | 0.0011 |
| to | 0.00083 | 0 | 0.0017 | 0.28 | 0.00083 | 0 | 0.0025 | 0.087 |
| eat | 0 | 0 | 0.0027 | 0 | 0.021 | 0.0027 | 0.056 | 0 |
| chinese | 0.0063 | 0 | 0 | 0 | 0 | 0.52 | 0.0063 | 0 |
| food | 0.014 | 0 | 0.014 | 0 | 0.00092 | 0.0037 | 0 | 0 |
| lunch | 0.0059 | 0 | 0 | 0 | 0 | 0.0029 | 0 | 0 |
| spend | 0.0036 | 0 | 0.0036 | 0 | 0 | 0 | 0 | 0 |

Smoothed:

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 0.0015 | 0.21 | 0.00025 | 0.0025 | 0.00025 | 0.00025 | 0.00025 | 0.00075 |
| want | 0.0013 | 0.00042 | 0.26 | 0.00084 | 0.0029 | 0.0029 | 0.0025 | 0.00084 |
| to | 0.00078 | 0.00026 | 0.0013 | 0.18 | 0.00078 | 0.00026 | 0.0018 | 0.055 |
| eat | 0.00046 | 0.00046 | 0.0014 | 0.00046 | 0.0078 | 0.0014 | 0.02 | 0.00046 |
| chinese | 0.0012 | 0.00062 | 0.00062 | 0.00062 | 0.00062 | 0.052 | 0.0012 | 0.00062 |
| food | 0.0063 | 0.00039 | 0.0063 | 0.00039 | 0.00079 | 0.002 | 0.00039 | 0.00039 |
| lunch | 0.0017 | 0.00056 | 0.00056 | 0.00056 | 0.00056 | 0.0011 | 0.00056 | 0.00056 |
| spend | 0.0012 | 0.00058 | 0.0012 | 0.00058 | 0.00058 | 0.00058 | 0.00058 | 0.00058 |

# Refine the idea of smoothing

- Should all unseen words get equal probabilities?

- We can use a reference model to discriminate unseen words

Discounted ML estimate

Reference language model

$$p(w \mid d) = \begin{cases} p_{seen}(w \mid d) & \text{if } w \text{ is seen in } d \\ \alpha_d\, p(w \mid REF) & \text{otherwise} \end{cases}$$
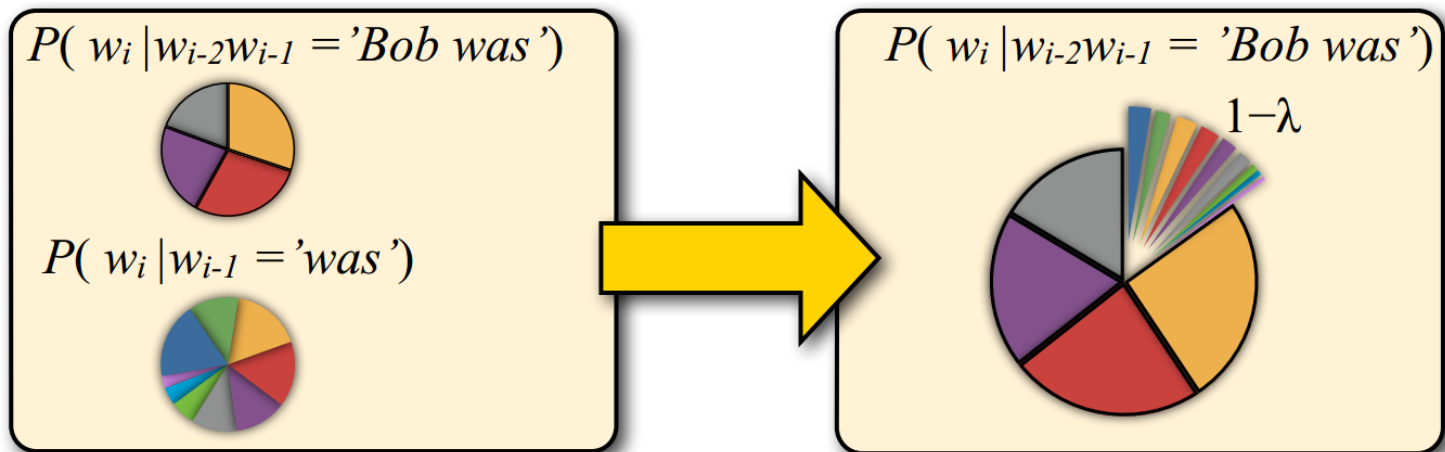
$$\alpha_d = \frac{1 - \sum\limits_{w \text{ is seen}} p_{seen}(w \mid d)}{\sum\limits_{w \text{ is unseen}} p(w \mid REF)}$$

# Smoothing methods

- Linear interpolation
  - Use (N −1)-gram probabilities to smooth N-gram probabilities
    - We never see the trigram "Bob was reading", but we do see the bigram "was reading"

# Smoothing methods

- Linear interpolation
  - Use (N −1)-gram probabilities to smooth N-gram probabilities

*ML N-gram*

$$\bar{p}(w_i|w_{i-1}, \ldots, w_{i-n+1}) = \lambda p_{ML}(w_i|w_{i-1}, \ldots, w_{i-n+1})$$
$$+(1-\lambda)\bar{p}(w_i|w_{i-1}, \ldots, w_{i-n+2})$$

*Smoothed N-gram*

  - Further generalize it

*Smoothed (N-1)-gram*

$$\bar{p}(w_i|w_{i-1}, \ldots, w_{i-n+1}) = \lambda_1 p_{ML}(w_i|w_{i-1}, \ldots, w_{i-n+1})$$
$$+\lambda_2 \bar{p}(w_i|w_{i-1}, \ldots, w_{i-n+2})$$

$$s.t. \sum_{i=1}^{n} \lambda_i = 1 \qquad \begin{array}{l} \ldots\ldots \\ +\lambda_n \bar{p}(w_i) \end{array}$$

# Smoothing methods

- Linear interpolation
  - Estimating $\lambda s$
    - Using a hold-out data set to find the optimal $\lambda s$
      - An evaluation metric is needed to define "optimality"
    - Define $\lambda$ as a function of $w_{i-1}, \ldots, w_{i-n+2}$

We will come back to this later

# Smoothing methods

- Absolute discounting
  - Subtract a constant $\delta$ from each nonzero n-gram count and then interpolate

$$\bar{p}(w_i|w_{i-1}, \ldots, w_{i-n+1}) = \frac{\max(c(w_i, w_{i-1}, \ldots, w_{i-n+1}) - \delta, 0)}{c(w_{i-1}, \ldots, w_{i-n+1})}$$
$$+ \lambda \bar{p}(w_i|w_{i-1}, \ldots, w_{i-n+2})$$

  *Smoothed N-gram*

  *Smoothed (N-1)-gram*

  - If $S$ seen word types occur after $w_{i-1}, \ldots, w_{i-n+1}$ in the training data, this reserves the probability mass $\mathrm{p(u)} = \frac{\delta S}{c(w_{i-1}, \ldots, w_{i-n+1})}$ to be reallocated according to $\bar{p}(w_i|w_{i-1}, \ldots, w_{i-n+2})$
    - $\lambda = \frac{\delta S}{c(w_{i-1}, \ldots, w_{i-n+1})}$

# Language model evaluation

- Train the models on the same training set
  - Parameter tuning can be done by holding off some training set for validation
- Test the models on an unseen test set
  - This data set must be disjoint from training data
- Language model A is better than model B
  - If A assigns <u>higher probability</u> to the test data than B

# Perplexity

- Standard evaluation metric for language models
  - A function of the probability that a language model assigns to a data set
  - Rooted in the notion of cross-entropy in information theory

# Perplexity

- The inverse of the likelihood of the test set as assigned by the language model, normalized by the number of words

$$PP(w_1, \ldots, w_N) = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} p(w_i | w_{i-1}, \ldots, w_{i-n+1})}}$$

N-gram language model

# An experiment

- Models
  - Unigram, Bigram, Trigram models (with proper smoothing)
- Training data
  - 38M words of WSJ text (vocabulary: 20K types)
- Test data
  - 1.5M words of WSJ text
- Results

|  | Unigram | Bigram | Trigram |
|---|---|---|---|
| Perplexity | 962 | 170 | 109 |

# What you should know

- N-gram language models

- How to generate text documents from a language model

- How to estimate a language model

- General idea and different ways of smoothing

- Language model evaluation

# Today's reading

- Introduction to information retrieval
  - Chapter 12: Language models for information retrieval

- Speech and Language Processing
  - Chapter 4: N-Grams