



# Language Models

Computational Linguistics: Jordan Boyd-Graber  
University of Maryland

INTRODUCTION

Slides adapted from Philip Koehn

## Roadmap

After this class, you'll be able to:

- Give examples of where we need language models
- Explain the independence assumptions of language models
- Estimate probability distributions using Laplace and Dirichlet smoothing
- Evaluate language models

## Language models

- **Language models** answer the question: How likely is a string of English words good English?
- Autocomplete on phones and websearch
- Creating English-looking documents
- Very common in machine translation systems
  - Help with reordering / style

$$p_{lm}(\text{the house is small}) > p_{lm}(\text{small the is house})$$

- Help with word choice

$$p_{lm}(\text{I am going home}) > p_{lm}(\text{I am going house})$$

## N-Gram Language Models

- Given: a string of English words  $W = w_1, w_2, w_3, \dots, w_n$
  - Question: what is  $p(W)$ ?
  - Sparse data: Many good English sentences will not have been seen before
- Decomposing  $p(W)$  using the chain rule:

$$p(w_1, w_2, w_3, \dots, w_n) = p(w_1) p(w_2|w_1) p(w_3|w_1, w_2) \dots p(w_n|w_1, w_2, \dots, w_{n-1})$$

(not much gained yet,  $p(w_n|w_1, w_2, \dots, w_{n-1})$  is equally sparse)

## Markov Chain

- **Markov independence assumption:**

- only previous history matters
- limited memory: only last  $k$  words are included in history (older words less relevant)

→  **$k$ th order Markov model**

- For instance 2-gram language model:

$$p(w_1, w_2, w_3, \dots, w_n) \simeq p(w_1) p(w_2|w_1) p(w_3|w_2) \dots p(w_n|w_{n-1})$$

- What is conditioned on, here  $w_{i-1}$  is called the **history**

## How good is the LM?

- A good model assigns a text of real English  $W$  a high probability
- This can be also measured with **perplexity**

$$\begin{aligned}\text{perplexity}(W) &= P(w_1, \dots, w_N)^{-\frac{1}{N}} \\ &= \sqrt[N]{\prod_i \frac{1}{P(w_i | w_1 \dots w_{i-1})}}\end{aligned}$$

## Comparison 1–4-Gram

word	unigram	bigram	trigram	4-gram
<b>i</b>	6.684	3.197	3.197	3.197
<b>would</b>	8.342	2.884	2.791	2.791
<b>like</b>	9.129	2.026	1.031	1.290
<b>to</b>	5.081	0.402	0.144	0.113
<b>commend</b>	15.487	12.335	8.794	8.633
<b>the</b>	3.885	1.402	1.084	0.880
<b>reporter</b>	10.840	7.319	2.763	2.350
<b>.</b>	4.896	3.020	1.785	1.510
<b>&lt;/s&gt;</b>	4.828	0.005	0.000	0.000
average				
perplexity	265.136	16.817	6.206	4.758

## How do we estimate a probability?

- Suppose we want to estimate  $P(w_n = \text{"home"} | h = g_0)$ .



## How do we estimate a probability?

- Suppose we want to estimate  $P(w_n = \text{"home"} | h = \text{go})$ .

<b>home</b>	<b>home</b>	big	with	to
big	with	to	and	money
and	<b>home</b>	big	and	<b>home</b>
money	<b>home</b>	and	big	to

## How do we estimate a probability?

- Suppose we want to estimate  $P(w_n = \text{"home"} | h = \text{go})$ .

<b>home</b>	<b>home</b>	big	with	to
big	with	to	and	money
and	<b>home</b>	big	and	<b>home</b>
money	<b>home</b>	and	big	to

- Maximum likelihood (ML) estimate of the probability is:

$$\hat{\theta}_i = \frac{n_i}{\sum_k n_k} \quad (1)$$

## Example: 3-Gram

- Counts for trigrams and estimated word probabilities

**the red** (total: 225)

word	c.	prob.
<b>cross</b>	123	0.547
<b>tape</b>	31	0.138
<b>army</b>	9	0.040
<b>card</b>	7	0.031
,	5	0.022

- 225 trigrams in the Europarl corpus start with **the red**
  - 123 of them end with **cross**
- maximum likelihood probability is  $\frac{123}{225} = 0.547$ .

## Example: 3-Gram

- Counts for trigrams and estimated word probabilities

**the red** (total: 225)

word	c.	prob.
<b>cross</b>	123	0.547
<b>tape</b>	31	0.138
<b>army</b>	9	0.040
<b>card</b>	7	0.031
,	5	0.022

- 225 trigrams in the Europarl corpus start with **the red**
- 123 of them end with **cross**
- maximum likelihood probability is  $\frac{123}{225} = 0.547$ .

- Is this reasonable?

## The problem with maximum likelihood estimates: Zeros

- If there were no occurrences of “bageling” in a history  $g_0$ , we’d get a zero estimate:

$$\hat{P}(\text{“bageling”} | g_0) = \frac{T_{g_0, \text{“bageling”}}}{\sum_{w' \in V} T_{g_0, w'}} = 0$$

- $\rightarrow$  We will get  $P(g_0 | d) = 0$  for any sentence that contains  $g_0$  bageling!
- Zero probabilities cannot be conditioned away.

## How do we estimate a probability?

- In computational linguistics, we often have a prior notion of what our probability distributions are going to look like (for example, non-zero, sparse, uniform, etc.).
- This estimate of a probability distribution is called the maximum a posteriori (MAP) estimate:

$$\theta_{\text{MAP}} = \operatorname{argmax}_{\theta} f(x|\theta)g(\theta) \quad (2)$$

## Add-One Smoothing

- Equivalent to assuming a **uniform** prior over all possible distributions over the next word (you'll learn why in CL2)
- But there are many more unseen n-grams than seen n-grams
- Example: Europarl 2-bigrams:
  - 86,700 distinct words
  - $86,700^2 = 7,516,890,000$  possible bigrams
  - but only about 30,000,000 words (and bigrams) in corpus

## How do we estimate a probability?

- Assuming a **sparse Dirichlet** prior,  $\alpha < 1$  to each count

$$\theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \quad (3)$$

- $\alpha_i$  is called a smoothing factor, a pseudocount, etc.



## How do we estimate a probability?

- Assuming a **sparse Dirichlet** prior,  $\alpha < 1$  to each count

$$\theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \quad (3)$$

- $\alpha_i$  is called a smoothing factor, a pseudocount, etc.
- When  $\alpha_i = 1$  for all  $i$ , it's called "Laplace smoothing"

## How do we estimate a probability?

- Assuming a **sparse Dirichlet** prior,  $\alpha < 1$  to each count

$$\theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \quad (3)$$

- $\alpha_i$  is called a smoothing factor, a pseudocount, etc.
- When  $\alpha_i = 1$  for all  $i$ , it's called "Laplace smoothing"
- What is a good value for  $\alpha$ ?
- Could be optimized on held-out set to find the "best" language model

## Example: 2-Grams in Europarl

Count	Adjusted count		Test count
$c$	$(c + 1)$	$(c + \alpha)$	$t_c$
0	0.00378	0.00016	0.00016
1	0.00755	0.95725	0.46235
2	0.01133	1.91433	1.39946
3	0.01511	2.87141	2.34307
4	0.01888	3.82850	3.35202
5	0.02266	4.78558	4.35234
6	0.02644	5.74266	5.33762
8	0.03399	7.65683	7.15074
10	0.04155	9.57100	9.11927
20	0.07931	19.14183	18.95948

## Example: 2-Grams in Europarl

Count	Adjusted count		Test count
$c$	$(c + 1)$	$(c + \alpha)$	$t_c$
0	0.00378	0.00016	0.00016
1	0.00755	0.95725	0.46235
2	0.01133	1.91433	1.39946
3	0.01511	2.87141	2.34307
4	0.01888	3.82850	3.35202

Can we do better?

In higher-order models, we can learn from similar contexts!

5	0.02266	7.00000	7.15074
10	0.04155	9.57100	9.11927
20	0.07931	19.14183	18.95948

## What's a word?

- There are an infinite number of words
  - Possible to develop generative story of how new words are created
  - Bayesian non-parametrics

## What's a word?

- There are an infinite number of words
  - Possible to develop generative story of how new words are created
  - Bayesian non-parametrics
- Defining a vocabulary (the event space)
- But how do you handle words outside of your vocabulary?

## What's a word?

- There are an infinite number of words
  - Possible to develop generative story of how new words are created
  - Bayesian non-parametrics
- Defining a vocabulary (the event space)
- But how do you handle words outside of your vocabulary?
  - Ignore? You could win just by ignoring everything
  - Standard: replace with `<UNK>` token

## Reducing Vocabulary Size

- For instance: each number is treated as a separate token
- Replace them with a number token num
  - but: we want our language model to prefer

$$p_{lm}(\text{I pay 950.00 in May 2007}) > p_{lm}(\text{I pay 2007 in May 950.00})$$

- not possible with number token

$$p_{lm}(\text{I pay num in May num}) = p_{lm}(\text{I pay num in May num})$$

- Replace each digit (with unique symbol, e.g., @ or 5), retain some distinctions

$$p_{lm}(\text{I pay 555.55 in May 5555}) > p_{lm}(\text{I pay 5555 in May 555.55})$$



## Back-Off

- In given corpus, we may never observe
  - **Scottish beer drinkers**
  - **Scottish beer eaters**
- Both have count 0  
→ our smoothing methods will assign them same probability
- Better: backoff to bigrams:
  - **beer drinkers**
  - **beer eaters**

## Back-Off

- In given corpus, we may never observe
  - **Scottish beer drinkers**
  - **Scottish beer eaters**
- Both have count 0
  - our smoothing methods will assign them same probability
- Better: backoff to bigrams:
  - **beer drinkers**
  - **beer eaters**
- How do we deal with this?