Department of Computer Science
UNIVERSITY OF COLORADO **BOULDER**

Language Models

Natural Language Processing: Jordan Boyd-Graber
University of Colorado Boulder
SEPTEMBER 10, 2014

Slides adapted from Philipp 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

Outline

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

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

    - Help with word choice

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

N-Gram Language Models

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

→ Decomposing $p(W)$ using the chain rule:

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

(not much gained yet, $p(w_n|w_1, w_2, ...w_{n-1})$ 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:

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

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

Outline

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**

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

Comparison 1–4-Gram

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

Outline

How do we estimate a probability?

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

How do we estimate a probability?

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

home	**home**	big	with	to
big	with	to	and	money
and	**home**	big	and	**home**
money	**home**	and	big	to

How do we estimate a probability?

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

home	**home**	big	with	to
big	with	to	and	money
and	**home**	big	and	**home**
money	**home**	and	big	to

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

$$\hat{\theta}_i = \frac{n_i}{\sum_k n_k} \tag{1}$$

Example: 3-Gram

- Counts for trigrams and estimated word probabilities

the red (total: 225)

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

  - 225 trigrams in the Europarl corpus start with **the red**
  - 123 of them end with **cross**

$\rightarrow$ 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.
cross	123	0.547
tape	31	0.138
army	9	0.040
card	7	0.031
,	5	0.022

  - 225 trigrams in the Europarl corpus start with **the red**
  - 123 of them end with **cross**

$\rightarrow$ 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 *go*, we'd get a zero estimate:

$$\hat{P}(\texttt{bageling}|go) = \frac{T_{go,\texttt{bageling}}}{\sum_{w' \in V} T_{go,w'}} = 0$$

- $\rightarrow$ We will get $P(go|d) = 0$ for any sentence that contains go 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:

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

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

$$\theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \tag{3}$$

- α_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

$$\theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \tag{3}$$

- α_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

$$\theta_i = \frac{n_i + \alpha_i}{\sum_k n_k + \alpha_k} \tag{3}$$

- α_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 α?
- 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
8	0.03399	7.65683	7.15074
10	0.04155	9.57100	9.11927
20	0.07931	19.14183	18.95948

Can we do better?

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

Outline

Back-Off

- In given corpus, we may never observe
  - **Scottish beer drinkers**
  - **Scottish beer eaters**
- Both have count 0

  $\rightarrow$ our smoothing methods will assign them same probability
- Better: backoff to bigrams:
  - **beer drinkers**
  - **beer eaters**

Interpolation

- Higher and lower order n-gram models have different strengths and weaknesses
  - high-order n-grams are sensitive to more context, but have sparse counts
  - low-order n-grams consider only very limited context, but have robust counts
- Combine them

$$
\begin{aligned}
p_I(w_3|w_1, w_2) = \quad & \lambda_1 \; p_1(w_3) \\
& + \lambda_2 \; p_2(w_3|w_2) \\
& + \lambda_3 \; p_3(w_3|w_1, w_2)
\end{aligned}
$$

Back-Off

- Trust the highest order language model that contains n-gram

$$p_n^{BO}(w_i|w_{i-n+1},...,w_{i-1}) =$$
$$= \begin{cases} \alpha_n(w_i|w_{i-n+1},...,w_{i-1}) \\ \qquad\qquad \text{if } \text{count}_n(w_{i-n+1},...,w_i) > 0 \\ d_n(w_{i-n+1},...,w_{i-1})\ p_{n-1}^{BO}(w_i|w_{i-n+2},...,w_{i-1}) \\ \qquad\qquad \text{else} \end{cases}$$

- Requires
  - adjusted prediction model $\alpha_n(w_i|w_{i-n+1},...,w_{i-1})$
  - discounting function $d_n(w_1,...,w_{n-1})$

Back-Off

- Trust the highest order language model that contains n-gram

$$p_n^{BO}(w_i | w_{i-n+1}, \ldots, w_{i-1}) =$$

$$= \begin{cases} \alpha_n(w_i | w_{i-n+1}, \ldots, w_{i-1}) \\ \qquad\qquad\qquad \text{if } \text{count}_n(w_{i-n+1}, \ldots, w_i) > 0 \\ d_n(w_{i-n+1}, \ldots, w_{i-1})\; p_{n-1}^{BO}(w_i | w_{i-n+2}, \ldots, w_{i-1}) \\ \qquad\qquad\qquad \text{else} \end{cases}$$

- Requires
  - adjusted prediction model $\alpha_n(w_i | w_{i-n+1}, \ldots, w_{i-1})$
  - discounting function $d_n(w_1, \ldots, w_{n-1})$
  - More next time

Reducing Vocabulary Size

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

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

  - not possible with number token

$$p_{\mathrm{LM}}(\text{I pay } \textsc{num} \text{ in May } \textsc{num}) = p_{\mathrm{LM}}(\text{I pay } \textsc{num} \text{ in May } \textsc{num})$$

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

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

Summary

- Language models: *How likely is a string of English words good English?*
- N-gram models (Markov assumption)
- Perplexity
- Count smoothing
- Interpolation and backoff

In Class . . .

- Any remaining questions for first homework
- Discussing homework assignment: building bigram language models