LM SMOOTHING CONCLUDED

Based on slides from David Kauchak and Philipp Koehn.

March 23, 2015

Language Model Requirements

- Assign a probability to *every sentence* (i.e., string of words)

$$\sum_{\mathbf{e} \in \Sigma^*} p_{\mathrm{LM}}(\mathbf{e}) = 1$$

$$p_{\mathrm{LM}}(\mathbf{e}) \geq 0 \quad \forall \mathbf{e} \in \Sigma^*$$

How do LMs help?

- Goal: Assign a higher probability to *good sentences in English*

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

translations of German *Haus: home, house* …

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

Aside: Some Information Theory

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

Aside: Some Information Theory

Perplexity PPL

$$\mathrm{PPL} = 2^{H(X,\hat{p})}$$

Where $H(X,\hat{p}) = -\sum_{x \in X} p(x) \log \hat{p}(x)$

Intuitively: X is as random as if it had PPL equally-likely outcomes.

Smoothing

We'd never seen the trigram "d i n" before, so our trigram model had probability 0.

P(d i n e) =

P(d | <start> <start>) *

P(i | <start> d) *

P(n| d i) *

P(e| i n) *

P(<end>| n e)

Smoothing

$P(d \mid \text{<start> <start>}) = 1/11$

$P(i \mid \text{<start>}\ d) = 1$

$P(n \mid d\ i) = 0$

$P(e \mid i\ n) = 1$

$P(\text{<end>} \mid n\ e) = 1$

These probability estimates may be inaccurate. Smoothing can help reduce some of the noise.

Smoothing the estimates

Basic idea:

$p(a \mid x\ y) = 1/3$? *reduce*
$p(d \mid x\ y) = 2/3$? *reduce*
$p(z \mid x\ y) = 0/3$? *increase*

Discount the positive counts somewhat

Reallocate that probability to the zeroes

Remember, it needs to stay a probability distribution

Add-one (Laplacian) smoothing

	MLE Count	MLE Prob	Add-1 Count	Add-1 Prob
xya	100	100/300	101	101/326
xyb	0	0/300	1	1/326
xyc	0	0/300	1	1/326
xyd	200	200/300	201	201/326
xye	0	0/300	1	1/326
…				
xyz	0	0/300	1	1/326
Total xy	300	300/300	326	326/326

Add-lambda smoothing

A large dictionary makes novel events too probable.

Instead of adding 1 to all counts, add $\lambda = 0.01$?

- This gives much less probability to novel events

see the abacus	1	1/3	1.01	1.01/203
see the abbot	0	0/3	0.01	0.01/203
see the abduct	0	0/3	0.01	0.01/203
see the above	2	2/3	2.01	2.01/203
see the Abram	0	0/3	0.01	0.01/203
…			0.01	0.01/203
see the zygote	0	0/3	0.01	0.01/203
Total	3	3/3	203	

Add-lambda smoothing

How did we pick lambda?

see the abacus	1	1/3	1.01	1.01/203
see the abbot	0	0/3	0.01	0.01/203
see the abduct	0	0/3	0.01	0.01/203
see the above	2	2/3	2.01	2.01/203
see the Abram	0	0/3	0.01	0.01/203
…			0.01	0.01/203
see the zygote	0	0/3	0.01	0.01/203
Total	3	3/3	203	

Vocabulary

n-gram language modeling assumes we have a fixed vocabulary

- why?

Whether implicit or explicit, an n-gram language model is defined over a finite, fixed vocabulary

What happens when we encounter a word not in our vocabulary (Out Of Vocabulary)?

- If we don't do anything, prob = 0
- Smoothing doesn't really help us with this!

Vocabulary

To make this explicit, smoothing helps us with…

all entries in our vocabulary

see the abacus	1	1.01
see the abbot	0	0.01
see the abduct	0	0.01
see the above	2	2.01
see the Abram	0	0.01
…		0.01
see the zygote	0	0.01

Vocabulary

and…

Vocabulary	Counts	Smoothed counts
a	10	10.01
able	1	1.01
about	2	2.01
account	0	0.01
acid	0	0.01
across	3	3.01
…	…	…
young	1	1.01
zebra	0	0.01

How can we have words in our vocabulary we've never seen before?

Vocabulary

No matter your chosen vocabulary, you're still going to have out of vocabulary (OOV)

How can we deal with this?

- Ignore words we've never seen before
  - Somewhat unsatisfying, though can work depending on the application
  - Probability is then dependent on how many in vocabulary words are seen in a sentence/text
- Use a special symbol for OOV words and estimate the probability of out of vocabulary

Out of vocabulary

Add an extra word in your vocabulary to denote OOV (<OOV>, <UNK>)

Replace all words in your training corpus not in the vocabulary with <UNK>

- You'll get bigrams, trigrams, etc with <UNK>
  - p(<UNK> | "I am")
  - p(fast | "I <UNK>")

During testing, similarly replace all OOV with <UNK>

Choosing a vocabulary

A common approach:

- Replace the first occurrence of each word by <UNK> in a data set
- Estimate probabilities normally

Vocabulary is all words that occur two or more times

This also discounts all word counts by 1 and gives that probability mass to <UNK>

Problems with frequency based smoothing

The following bigrams have never been seen:

p(<UNK> | San)　　　　p(<UNK>| ate)

Which would add-lambda pick as most likely?

Which would you pick?

Witten-Bell Discounting

Some words are more likely to be followed by new words

			food
	Diego		apples
	Francisco		bananas
San	Luis	ate	hamburgers
	Jose		a lot
	Marcos		for two
			grapes
			…

Witten-Bell Discounting

Probability mass is shifted around, depending on the context of words

If $P(w_i \mid w_{i-1},\ldots,w_{i-m}) = 0$, then the smoothed probability $P_{WB}(w_i \mid w_{i-1},\ldots,w_{i-m})$ is higher if the sequence $w_{i-1},\ldots,w_{i-m}$ occurs with many different words w_k

Witten-Bell Smoothing

- if $c(w_{i-1}, w_i) > 0$

$$P^{WB}(w_i \mid w_{i-1}) =$$

$$\frac{\text{\# times we saw the bigram}}{\text{\# times } w_{i-1} \text{ occurred} \quad + \quad \text{\# of types to the right of } w_{i-1}}$$

Witten-Bell Smoothing

- If $c(w_{i-1},w_i) = 0$

$$P^{WB}(w_i \mid w_{i-1}) = \frac{\#\text{ of types to the right of } w_{i-1}}{\#\text{ times } w_{i-1} \text{ occurred} \;+\; \#\text{ of types to the right of } w_{i-1}}$$

Problems with frequency based smoothing

The following trigrams have never been seen:

p(car | see the)

p(zygote | see the)

p(cumquat | see the)

Which would add-lambda pick as most likely?
Witten-Bell?

Which would you pick?

Better smoothing approaches

Utilize information in lower-order models

Interpolation

- Combine probabilities of lower-order models in some linear combination

Backoff

$$P(z \mid xy) = \begin{cases} \dfrac{C^{*}(xyz)}{C(xy)} & \text{if } C(xyz) > k \\ \alpha(xy) P(z \mid y) & \text{otherwise} \end{cases}$$

- Often k = 0 (or 1)
- Combine the probabilities by “backing off” to lower models only when we don’t have enough information

Smoothing: Simple Interpolation

$$P(z \mid xy) \approx \lambda \frac{C(xyz)}{C(xy)} + \mu \frac{C(yz)}{C(y)} + (1 - \lambda - \mu) \frac{C(z)}{C(\bullet)}$$

Trigram is very context specific, very noisy

Unigram is context-independent, smooth

Interpolate Trigram, Bigram, Unigram for best combination

How should we determine λ and μ?

Smoothing: Finding parameter values

Just like we talked about before, split training data into training and development

Try lots of different values for λ, μ on heldout data, pick best

One approaches for finding these efficiently: EM!

One more problem…

The following bigrams have never been seen:

X baklava X Francisco

But we have seen:

San Francisco (1000 times)

ate baklava (20 times), sells baklava (30 times),

gave me baklava (10 times), best baklava (5 times)

Which would interpolation/backoff pick as most likely?

Which would you pick?

Kneser-Ney Smoothing

Some words are more likely to follow new words

San Francisco	ate bought made baked sent me to …	baklava

Kneser-Ney Smoothing

Lower-order distributions should include just the information we don't already have in the higher-order terms.

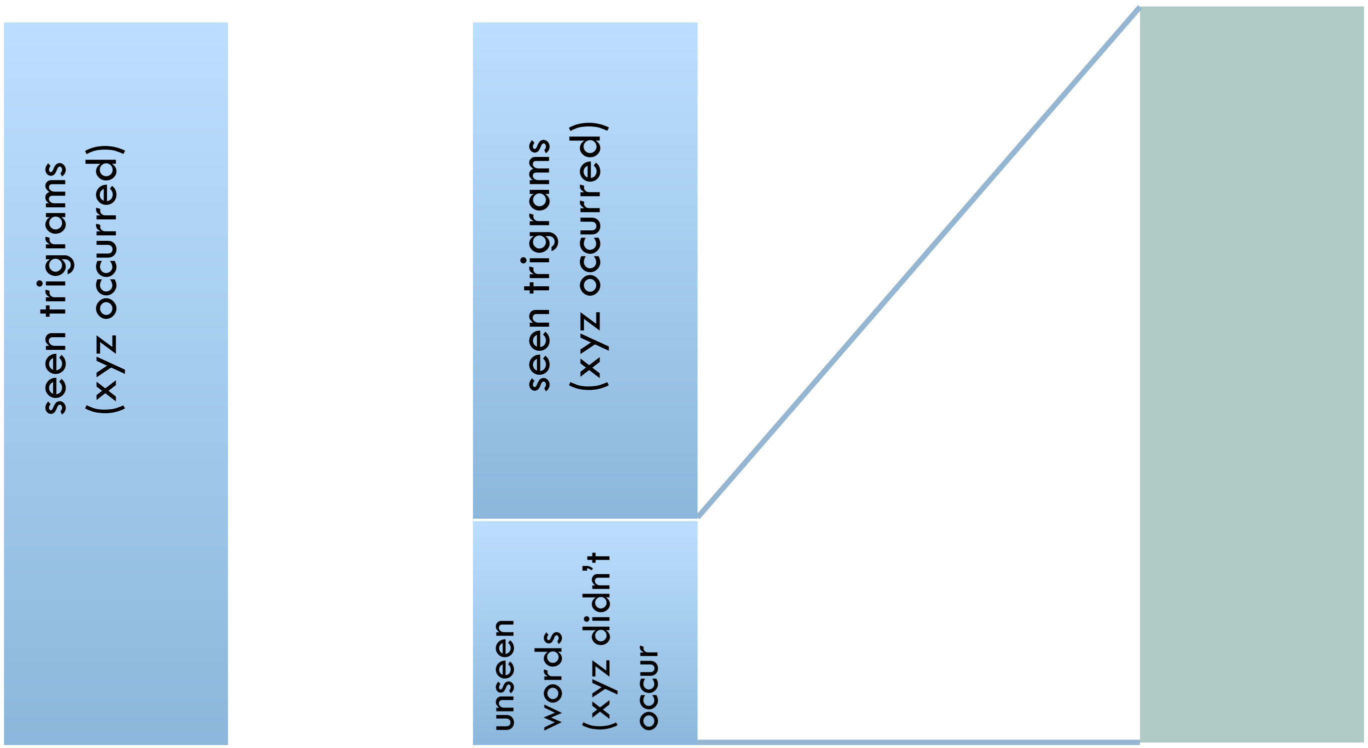
If w_i appears after many different histories, then its unigram frequency should be higher, so that in backoff/interpolation it get more probability mass.

Backoff models: absolute discounting

trigram model: p(z|xy) (before discounting)

trigram model p(z|xy) (after discounting)

bigram model p(z|y)* (*for z where xyz didn't occur)



$$P_{absolute}(z \mid xy) = \begin{cases} \dfrac{C(xyz) - D}{C(xy)} & \text{if } C(xyz) > 0 \\ \alpha(xy) P_{absolute}(z \mid y) & \text{otherwise} \end{cases}$$

Backoff models: absolute discounting

$$\text{reserved\_mass} = \frac{\#\text{ of } \textit{types} \text{ starting with bigram} * D}{\text{count(bigram)}}$$

Two nice attributes:

- decreases if we've seen more bigrams
  - should be more confident that the unseen trigram is no good
- increases if the bigram tends to be followed by lots of other words
  - will be more likely to see an unseen trigram

Let's practice

- What will add-1 and add-lambda (assume lambda=.01) counts look like for
  - a,b,c,d,e
  - he,to,ay,ll,di
- What will interpolation, back-off, Witten-Bell discounting do for p(i|d)?

Corpus
t h e
s u n
d i d
n o t
s h i n e
i t
w a s
t o o
w e t
t o
p l a y

Language Model Summary

What is an n-gram language model?

- How are they used:
  - In machine translation?
  - In NLP more generally?
- What is smoothing, and why do we need it?
- What is the difference between back-off and interpolation?

Project 2 Overview

- You’ll build an end-to-end MT system
- Europarl corpus
- Available later today, and you can start right away:
  - Language model component
  - Translation model component
- Next week you’ll be ready to write the decoder

Project 2 Logistics

- Teams of 3-4, whole team gets the same grade.
- Part of your grade will be based on how well your translation system works on my evaluation set.
  - You can improve any (or all!) of the components of your system.
  - There are suggestions for improvements of each component in the project writeup.
- You'll present the modifications you made and your final results in class on April 8.
- Adding a 4-page writeup so you can include details.

“My Midterm”

- Thank you all for your feedback!
- Common Themes
  - Assumed math background
  - Project 1 organization
  - More examples in class

New Topics — Interest Report

- 11 Sentiment Analysis
- 8 Part of Speech tagging
- 7 Syntactic Parsing
- 6 Computational semantics (mapping words/sentences to logical expressions)
- 5 Speech-to-Speech translation
- 5 Quantifying word similarity
- 5 Topic modeling
- 4 Incorporating syntax into MT
- 4 Genre/topic variation in machine translation