LM SMOOTHING

Based on slides from David Kauchak and Philipp Koehn.

March 11, 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*

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

translations of German *Haus: home, house* …

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

Aside: Some Information Theory

Entropy H(X): The average uncertainty of a random variable.

Mathematically, the average number of bits needed to encode the result of the random variable.

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

Cross Entropy: Address the fact that we don't know p(x) — it's what we're trying to estimate! Use our estimate ^p(x) instead.

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

Aside: Some Information Theory

Perplexity PPL

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

$$\mathrm{PPL} = 2^{-\sum_{x \in X} p(x) \log_2 \hat{p}(x)}$$

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

Entropy & Perplexity Practice

Suppose we have a 10-sided die, and we don't know if it's fair or not.

Theory 1 says that it probably is fair — $p(x) = 0.1$ for all x

Theory 2 is more suspicious — it argues that $p(x) = .05$ if x is odd, and .15 if x is even.

We roll the die and get the following:

1 2 2 4 8 2 9 3 6 7

For each theory, what is the cross-entropy? The perplexity?

Perplexity

We will use **perplexity** to evaluate models

$$\text{Given: } \mathbf{w}, p_{\text{LM}}$$

$$\text{PPL} = 2^{\frac{1}{|\mathbf{w}|} \log_2 p_{\text{LM}}(\mathbf{w})}$$

$$0 \leq \text{PPL} \leq \infty$$

Another view of perplexity

- Generally fairly good correlations with machine translation quality for *n*-gram models
- Perplexity is a generalization of the notion of branching factor
  - How many choices do I have at each position?
- State-of-the-art English LMs have PPL of ~100 word choices per position
- A uniform LM has a perplexity of $|\Sigma|$
- Humans do much better
- ... and bad models can do even *worse* than uniform!

Let's practice!

- To keep vocabulary small, we'll model lowercase English words
  - Each word is a "sentence"
  - Each letter is a "word"

Our 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

Let's practice!

- Calculate MLE unigram probabilities for each "word"
- Calculate MLE bigram probabilities for:
  - h e
  - t o
  - a y
  - l l

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

Let's practice!

- How would we find the perplexity of the sentence "d i n e" under:
  - A unigram model?
  - A bigram model?
  - A trigram model?

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

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 | <start> <start>) = 1/11

P(i | <start> d) = 1

P(n| d i) = 0

P(e| i n) = 1

P(<end>| n e) = 1

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

A better approach

p(z | x y) = ?

Suppose our training data includes

... x y a ..
... x y d ...
... x y d ...

but never: xyz

We would conclude

p(a | x y) = 1/3?
p(d | x y) = 2/3?
p(z | x y) = 0/3?

Is this ok?

Intuitively, how should we fix these?

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	1	1/3		
xyb	0	0/3		
xyc	0	0/3	?	
xyd	2	2/3		
xye	0	0/3		
…				
xyz	0	0/3		
Total xy	3	3/3		

Add-one (Laplacian) smoothing

300 observations instead of 3 – better data, less smoothing

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

Add-one (Laplacian) smoothing

What happens if we're now considering 20,000 word types?

xya	1	1/3	2	2/29
xyb	0	0/3	1	1/29
xyc	0	0/3	1	1/29
xyd	2	2/3	3	3/29
xye	0	0/3	1	1/29
…				
xyz	0	0/3	1	1/29
Total xy	3	3/3	29	29/29

Add-one (Laplacian) smoothing

20000 word types, not 26 letters

	MLE Count	MLE Prob	Add-1 Count	Add-1 Prob
see the abacus	1	1/3		
see the abbot	0	0/3		
see the abduct	0	0/3		
see the above	2	2/3	?	
see the Abram	0	0/3		
…				
see the zygote	0	0/3		
Total	3	3/3		

Any problem with this?

Add-one (Laplacian) smoothing

An "unseen event" is a 0-count event

The probability of an unseen event is 19998/20003

- add one smoothing thinks it is very likely to see a novel event

The problem with add-one smoothing is it gives too much probability mass to unseen events

see the abacus	1	1/3	2	2/20003
see the abbot	0	0/3	1	1/20003
see the abduct	0	0/3	1	1/20003
see the above	2	2/3	3	3/20003
see the Abram	0	0/3	1	1/20003
…				
see the zygote	0	0/3	1	1/20003
Total	3	3/3	20003	20003/20003

The general smoothing problem

			modification	probability
see the abacus	1	1/3	?	?
see the abbot	0	0/3	?	?
see the abduct	0	0/3	?	?
see the above	2	2/3	?	?
see the Abram	0	0/3	?	?
…			?	?
see the zygote	0	0/3	?	?
Total	3	3/3	?	?

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 should we pick lambda?

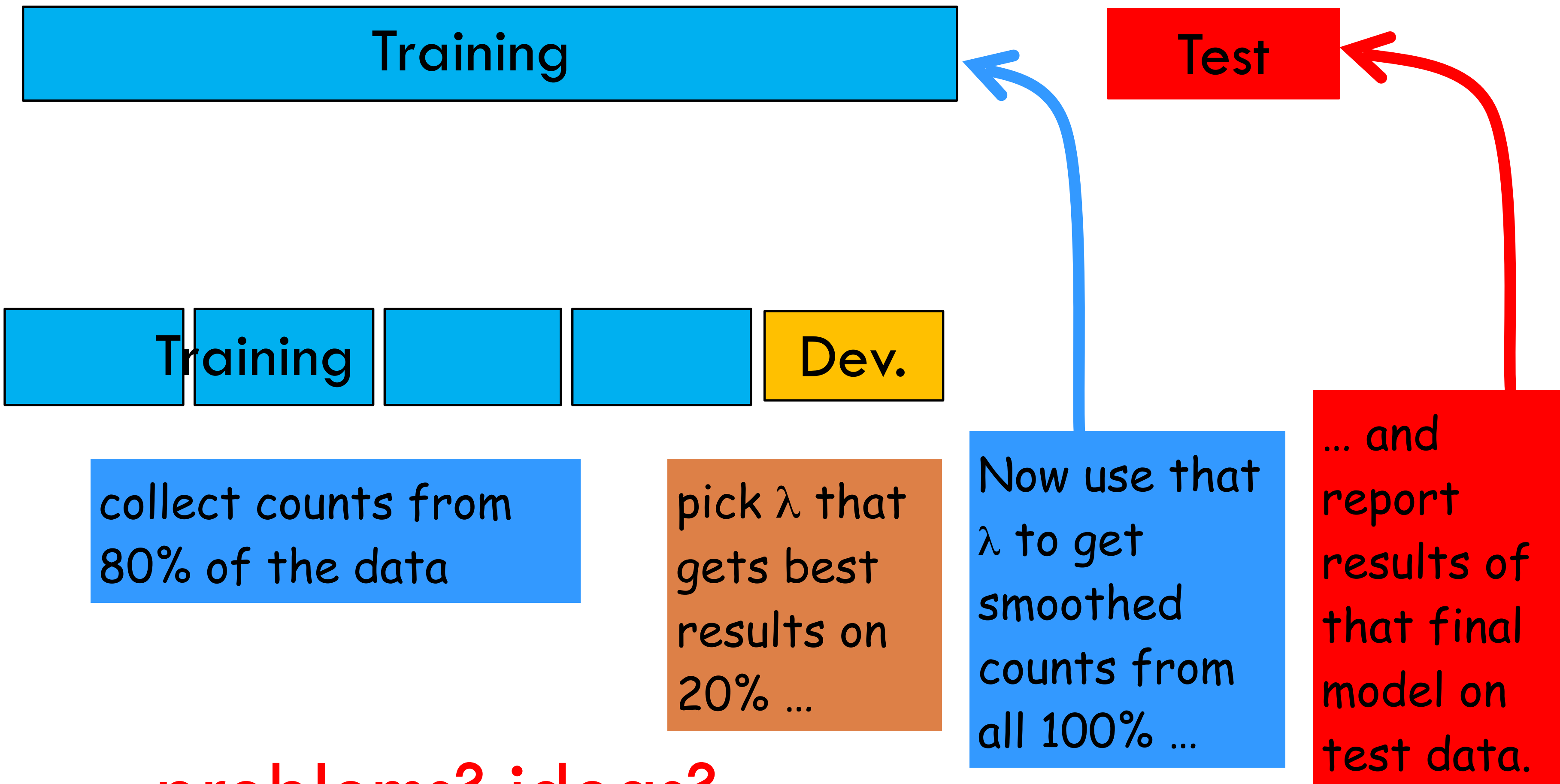
see the abacus	1	1/3	1.01	1.01/203
see the abbot	0	0/3	0.01	0.01/203
see the abduct	0	0/3	0.01	0.01/203
see the above	2	2/3	2.01	2.01/203
see the Abram	0	0/3	0.01	0.01/203
…			0.01	0.01/203
see the zygote	0	0/3	0.01	0.01/203
Total	3	3/3	203	

Setting smoothing parameters

Idea 1: try many λ values & report the one that gets the best results?

Is this fair/appropriate?

Setting smoothing parameters



problems? ideas?

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

Choosing a vocabulary: ideas?

- Grab a list of English words from somewhere
- Use all of the words in your training data
- Use some of the words in your training data
  - for example, all those the occur more than k times

Benefits/drawbacks?

- Ideally your vocabulary should represents words you're likely to see
- Too many words: end up washing out your probability estimates (and getting poor estimates)
- Too few: lots of out of vocabulary

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>