Language Modeling with N-grams

Chapter 3

(3.1-3.4)

Review

- Text Normalization
  - Why?
  - How computationally?
  - Example tasks?

2

Rule-based vs. Probabilistic

- "But it must be recognized that the notion of "probability of a sentence" is an entirely useless one, under any known interpretation of this term." *Noam Chomsky (1969)*
- "Anytime a linguist leaves the group the recognition rate goes up." *Fred Jelinek (1988, alleged)*

3

Intuition

- Predict the next word...
  - *... I noticed three guys standing on the ???*
- There are many sources of knowledge that can be used to inform this task, including arbitrary world knowledge.
- But it turns out that you can do pretty well by simply looking at the preceding words and keeping track of some fairly simple counts.

Word Prediction

- We can formalize this task using what are called *N*-gram models.
- *N*-grams are token sequences of length *N*.
- This example contains what 2-grams (aka bigrams)?
  - *I notice three guys standing on the*
- Given knowledge of counts of N-grams such as these, we can guess likely next words in a sequence.

N-Gram Models

- More formally, we can use knowledge of the counts of *N*-grams to assess the *conditional probability* of candidate words as the next word in a sequence.
- Or, we can use them to assess the *probability* of an entire sequence of words.
  - Pretty much the same thing as we'll see...

Probability

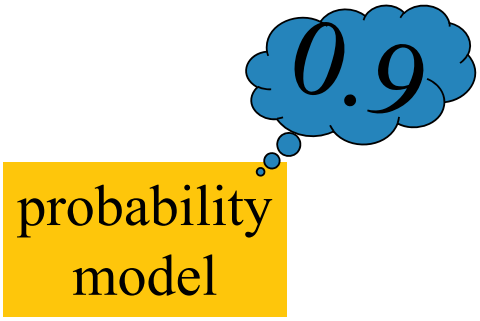
Quick Review

Different Kinds of Statistics

- **descriptive:** mean Pitt QPA (or median)
- **confirmatory:** statistically significant?
- **predictive:** wanna bet?
  - N-grams

8

Notation

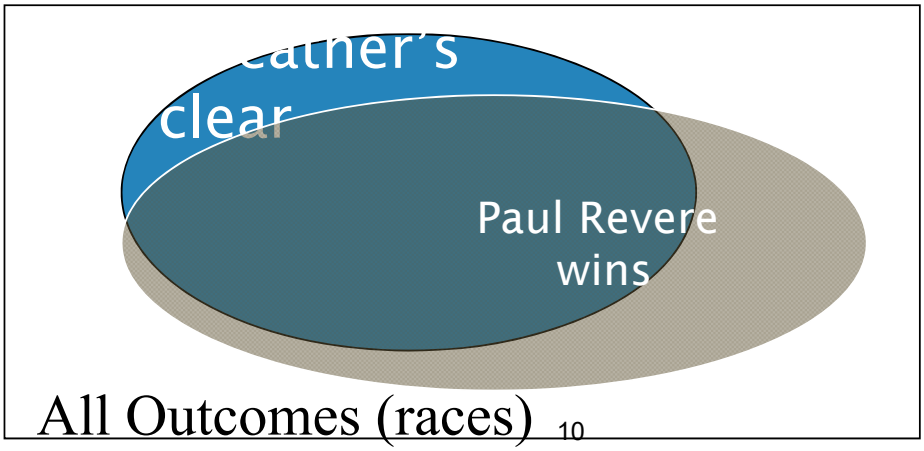


p(Paul Revere wins | weather's clear) = 0.9

9

p is a function on sets of "outcomes"

p(win | clear) ≡ p(win, clear) / p(clear)

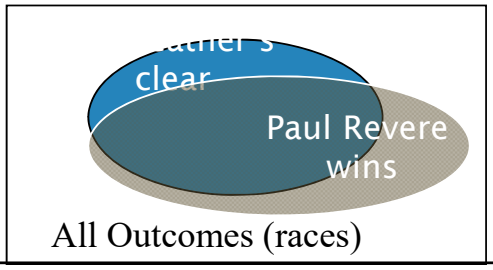


10

p is a function on sets of "outcomes"

$p(\text{win} \mid \text{clear}) \equiv p(\text{win}, \text{clear}) \,/\, p(\text{clear})$

- $p(\text{win} \mid \text{clear})$: syntactic sugar
- $p(\text{win}, \text{clear})$: logical conjunction of predicates
- $p(\text{clear})$: predicate selecting races where weather's clear



p measures total probability of a set of outcomes

Required Properties of p (axioms)

most of the

- $p(\varnothing) = 0$ $\quad$ $p(\text{all outcomes}) = 1$
- $p(X) \leq p(Y)$ for any $X \subseteq Y$
- $p(X) + p(Y) = p(X \cup Y)$ provided $X \cap Y = \varnothing$

e.g., $p(\text{win} \,\&\, \text{clear}) + p(\text{win} \,\&\, \neg\text{clear}) = p(\text{win})$

12

Commas denote conjunction

p(Paul Revere wins | weather's clear, ground is dry, jockey getting over sprain, Epitaph also in race, Epitaph was recently bought by Gonzalez, race is on May 17, …)

13

Simplifying Right Side: Backing Off

p(Paul Revere wins | weather's clear, ground is dry, jockey getting over sprain, Epitaph also in race, Epitaph was recently bought by Gonzalez, race is on May 17, …)

- not exactly what we want but at least we can get a reasonable estimate of it!
- try to *keep* the conditions that we suspect will have the most influence on whether Paul Revere wins

14

Language Modeling

Introduction to N-grams

Probabilistic Language Models

- Goal: assign a probability to a sentence

Why?

- Machine Translation:
  - P(**high** winds tonite) > P(**large** winds tonite)
- Spell Correction
  - The office is about fifteen **minuets** from my house
    - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
- Speech Recognition
  - P(I saw a van) >> P(eyes awe of an)
- + many more applications

Probabilistic Language Modeling

- Compute the probability of a sentence or word sequence

  $P(W) = P(w_1,w_2,w_3,w_4,w_5...w_n)$

- Related task: probability of an upcoming word

  $P(w_n|w_1,w_2...w_{n-1})$

- A model that computes either is a **language model**

  What kind of probabilities are these?

How to compute P(W)

- How to compute this *joint probability*:
  - P(its, water, is, so, transparent, that)
- Intuition: let's rely on the Chain Rule of Probability

Reminder: The Chain Rule

- Recall the definition of conditional probabilities

  p(B|A) = P(A,B)/P(A) Rewriting: **P(A,B) = P(A)P(B|A)**

  - Independent **p(B|A) = P(B)**
- More variables:

  P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)
- The Chain Rule in General

$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$

The Chain Rule applied to compute joint probability of words in sentence

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

How to estimate these probabilities

- Could we just count and divide?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

- No! Too many possible sentences!
- We'll never see enough data for estimating these

Markov Assumption


Andrei Markov

- Simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

- Or maybe

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

Markov Assumption

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- In other words, we approximate each component in the product

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- Bigram model (k=1, e.g., context of one so model two words)

Simplest case: Unigram model

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

Some automatically generated sentences from a unigram model

```
fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
```

Bigram model

- Condition on the previous word:

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

```
texaco, rose, one, in, this, issue, is, pursuing, growth, in,
a, boiler, house, said, mr., gurria, mexico, 's, motion,
control, proposal, without, permission, from, five, hundred,
fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november
```

N-gram models

- We can extend to trigrams, 4-grams, 5-grams
- In general this is an insufficient model of language
  - because language has **long-distance dependencies**:

  "The computer(s) which I had just put into the machine room on the fifth floor is (are) crashing."

- But we can often get away with N-gram models

Language Modeling

Estimating N-gram Probabilities

Estimating bigram probabilities

- The Maximum Likelihood Estimate

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

An example

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67$ $P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33$ $P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5$ $P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

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?'

P(of) = 3/66	P(her) = 2/66
P(Alice) = 2/66	P(sister) = 2/66
P(was) = 2/66	P(,) = 4/66
P(to) = 2/66	P(') = 4/66

Example from Julia Hockenmaier

31

Conditional on the previous word

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?'

$P(w_{i+1}=\text{of} \mid w_i=\text{tired}) =$
$P(w_{i+1}=\text{of} \mid w_i=\text{use}) =$
$P(w_{i+1}=\text{sister} \mid w_i=\text{her}) =$
$P(w_{i+1}=\text{beginning} \mid w_i=\text{was}) =$
$P(w_{i+1}=\text{reading} \mid w_i=\text{was}) =$

$P(w_{i+1}=\text{bank} \mid w_i=\text{the}) =$
$P(w_{i+1}=\text{book} \mid w_i=\text{the}) =$
$P(w_{i+1}=\text{use} \mid w_i=\text{the}) =$

32

Conditional on the previous word

English

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?'

Word Salad

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

Now, P(English) ≫ P(word salad)

$P(w_{i+1}=\text{of} \mid w_i=\text{tired}) = 1$
$P(w_{i+1}=\text{of} \mid w_i=\text{use}) = 1$
$P(w_{i+1}=\text{sister} \mid w_i=\text{her}) = 1$
$P(w_{i+1}=\text{beginning} \mid w_i=\text{was}) = 1/2$
$P(w_{i+1}=\text{reading} \mid w_i=\text{was}) = 1/2$

$P(w_{i+1}=\text{bank} \mid w_i=\text{the}) = 1/3$
$P(w_{i+1}=\text{book} \mid w_i=\text{the}) = 1/3$
$P(w_{i+1}=\text{use} \mid w_i=\text{the}) = 1/3$

33

More examples: Berkeley Restaurant Project sentences

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day

Raw bigram counts

- Out of 9222 sentences

	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

Raw bigram probabilities

- Normalize by unigrams:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Result:

	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

Bigram estimates of sentence probabilities

P(<s> I want english food </s>) =

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

What kinds of knowledge?

- P(english|want) = .0011
- P(chinese|want) = .0065
- P(to|want) = .66
- P(eat | to) = .28
- P(food | to) = 0
- P(want | spend) = 0
- P (i | <s>) = .25

Practical Issues

- We do everything in log space
  - Avoid underflow
  - (also adding is faster than multiplying)

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Language Modeling Toolkits

- SRILM
  - http://www.speech.sri.com/projects/srilm/
- KenLM
  - https://kheafield.com/code/kenlm/

Google N-Gram Release, August 2006

AUG
3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects,

...

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Google N-Gram Release

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensible 40
- serve as the individual 234

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Google Books N-gram Viewer

- https://books.google.com/ngrams

Google Caveat

- We will talk more about test sets and training sets... Test sets should be similar to the training set (drawn from the same distribution) for the probabilities to be meaningful.
- So... The Google corpus is fine if your application deals with arbitrary English text on the Web.
- If not then a smaller domain specific corpus is likely to yield better results.

44

Language Modeling

Evaluation and Perplexity

Evaluation: How good is our model?

- Does our language model prefer good sentences to bad ones?
  - Assign higher probability to "real" or "frequently observed" sentences
    - Than "ungrammatical" or "rarely observed" sentences?
    - Recall word salad example
- We train parameters of our model on a **training set**.
- We test the model's performance on data we haven't seen.
  - A **test set** is an unseen dataset that is different from our training set, totally unused.
  - An **evaluation metric** tells us how well our model does on the test set.

Training on the test set

- We can't allow test sentences into the training set
- We will assign it an artificially high probability when we set it in the test set
- "Training on the test set"
- Bad science!
- And violates the honor code
- More later!

47

Extrinsic evaluation of N-gram models

- Best evaluation for comparing models A and B
  - Put each model in a task
    - spelling corrector, speech recognizer, MT system
  - Run the task, get an accuracy for A and for B
    - How many misspelled words corrected properly
    - How many words translated correctly
  - Compare accuracy for A and B

Difficulty of extrinsic (in-vivo) evaluation of N-gram models

- Extrinsic evaluation
  - Time-consuming; can take days or weeks
- So
  - Sometimes use **intrinsic** evaluation: **perplexity**
  - Bad approximation
    - unless the test data looks **just** like the training data
    - So **generally only useful in pilot experiments**
  - But is helpful to think about.

Intuition of Perplexity

- The Shannon Game:
  - How well can we predict the next word?

    I always order pizza with cheese and ____

    The 33rd President of the US was ____

    I saw a ____

  - Unigrams are terrible at this game. (Why?)

mushrooms 0.1
pepperoni 0.1
anchovies 0.01
….
fried rice 0.0001
….
and 1e-100

- A better model of a text
  - is one which assigns a higher probability to the word that actually occurs

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)

Perplexity is the inverse probability of the test set, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

For bigrams:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Minimizing perplexity is the same as maximizing probability

Lower perplexity = better model

- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Language Modeling

Generalization and zeros

The Shannon Visualization Method

- Choose a random bigram (<s>, w) according to its probability
- Now choose a random bigram (w, x) according to its probability
- And so on until we choose </s>
- Then string the words together

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
I want to eat Chinese food
```

Approximating Shakespeare

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

Shakespeare as corpus

- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams.
  - So 99.96% of the possible bigrams were never seen (have zero entries in the table)
- Quadrigrams worse: What's coming out looks like Shakespeare because it ***is*** Shakespeare

The Wall Street Journal is not Shakespeare

1 gram	Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
2 gram	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 old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Can you guess the author of these random 3-gram sentences?

- They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and gram Brazil on market conditions
- This shall forbid it should be branded, if renown made it empty.

58

The perils of overfitting

- N-grams only work well for word prediction if the test corpus looks like the training corpus
  - In real life, it often doesn't
  - We need to train robust models that generalize!
  - One kind of generalization: Zeros!
    - Things that don't ever occur in the training set
      - But occur in the test set

Zeros

- Training set:
  … denied the allegations
  … denied the reports
  … denied the claims
  … denied the request

  P("offer" | denied the) = 0

- Test set
  … denied the offer
  … denied the loan

Zero probability bigrams

- Bigrams with zero probability
  - mean that we will assign 0 probability to the test set!
- And hence we cannot compute perplexity (can't divide by 0)!

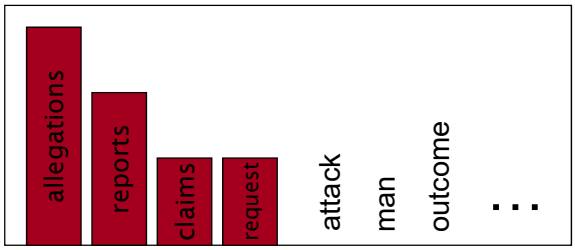
Language Modeling

Smoothing: Add-one (Laplace) smoothing

The intuition of smoothing (from Dan Klein)

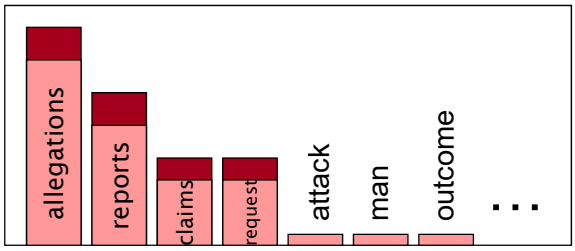
- When we have sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request
7 total



- Steal probability mass to generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other
7 total



Add-one estimation

- Also called Laplace smoothing
- Pretend we saw each word one more time than we did
- Just add one to all the counts!
- MLE estimate:

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- Add-1 estimate:

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)+1}{c(w_{i-1})+V}$$

Maximum Likelihood Estimates

- The maximum likelihood estimate
  - of some parameter of a model M from a training set T
  - maximizes the likelihood of the training set T given the model M
- Suppose the word “bagel” occurs 400 times in a corpus of a million words
- What is the probability that a random word from some other text will be “bagel”?
- MLE estimate is 400/1,000,000 = .0004
- This may be a bad estimate for some other corpus
  - But it is the **estimate** that makes it **most likely** that “bagel” will occur 400 times in a million word corpus.

Add-One Smoothing

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

66

Following examples from Kai-Wei Chang

Problem with Add-One Smoothing

We've been considering just 26 letter 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

67

Problem with Add-One Smoothing

Suppose we're considering 20000 word types

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

68

Problem with Add-One Smoothing

Suppose we're considering 20000 word types

see the abacus	1	1/3	2	2/20003

"Novel event" = event never happened in training data.

Here: 19998 novel events, with total estimated probability 19998/20003.

Add-one smoothing thinks we are extremely likely to see novel events, rather than words we've seen.

see the zygote	0	0/3	1	1/20003
Total	3	3/3	20003	20003/20003

69

Add-Lambda Smoothing

- A large dictionary makes novel events too probable.
- To fix: Instead of adding 1 to all counts, add $\lambda = 0.01$?
  - This gives much less probability to novel events.
- But how to pick *best value* for λ?
  - That is, how much should we smooth?

70

Add-0.001 Smoothing

Doesn't smooth much

xya	1	1/3	1.001	0.331
xyb	0	0/3	0.001	0.0003
xyc	0	0/3	0.001	0.0003
xyd	2	2/3	2.001	0.661
xye	0	0/3	0.001	0.0003
…				
xyz	0	0/3	0.001	0.0003
Total xy	3	3/3	3.026	1

71

Add-1000 Smoothing

Smooths too much

xya	1	1/3	1001	1/26
xyb	0	0/3	1000	1/26
xyc	0	0/3	1000	1/26
xyd	2	2/3	1002	1/26
xye	0	0/3	1000	1/26
…				
xyz	0	0/3	1000	1/26
Total xy	3	3/3	26003	1

72

Add-Lambda Smoothing

- A large dictionary makes novel events too probable.
- To fix: Instead of adding 1 to all counts, add $\lambda = 0.01$?
  - This gives much less probability to novel events.
- But how to pick *best value* for λ?
  - That is, how much should we smooth?
  - E.g., how much probability to "set aside" for novel events?
    - Depends on how likely novel events really are!
    - Which may depend on the type of text, size of training corpus, …
  - Can we figure it out from the data? (advanced topics)

73

Setting Smoothing Parameters

- How to pick *best value* for λ? (in add- λ smoothing)
- Try many λ values & report the one that gets best results?

- How to measure whether a particular λ gets good results?
- Is it fair to measure that on test data (for setting λ)?
  - *Moral:* Selective reporting on test data can make a method look artificially good. So ***it is unethical.***
  - *Rule:* Test data cannot influence system development. No peeking! Use it only to evaluate the final system(s). Report all results on it.

74

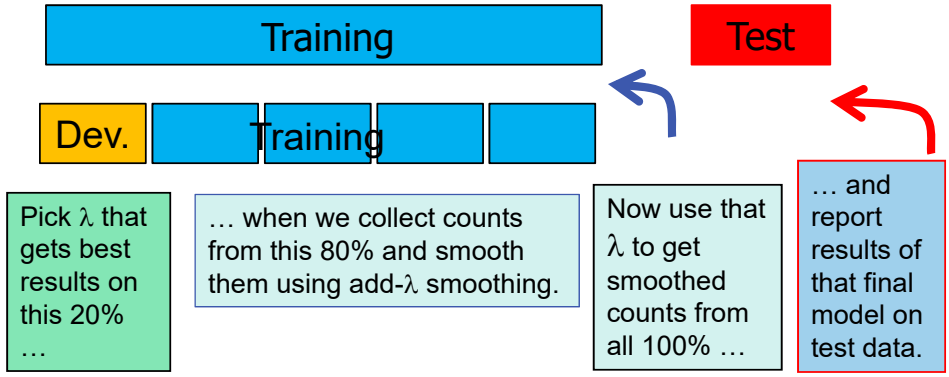
Setting Smoothing Parameters

- How to pick *best value* for λ? (in add- λ smoothing)
- Try many λ values & report the one that gets best results?

- How to measure whether a particular λ gets good results?
- Is it fair to measure that on test data (for setting λ)?
  - *Moral:* Selective reporting on test data can make a method look artificially good. So ***it is unethical.***
  - *Rule:* Test data cannot influence system development. No peeking! Use it only to evaluate the final system(s). Report all results on it.

Setting Smoothing Parameters

- How to pick *best value* for λ?
- Try many λ values & report the one that gets best results?



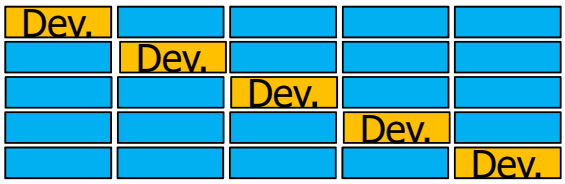
Large or small Dev set?

- Here we held out 20% of our training set (yellow) for development.
- Would like to use > 20% yellow:
  - 20% not enough to reliably assess λ
- Would like to use > 80% blue:
  - Best λ for smoothing 80% ≠ best λ for smoothing 100%

77

Cross-Validation

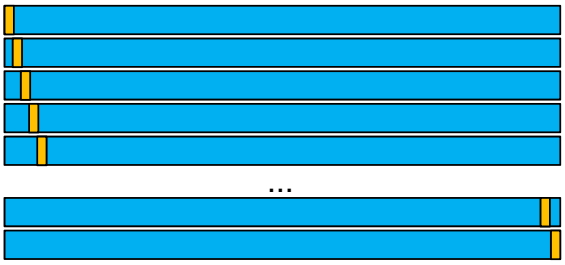
- Try 5 training/dev splits as below
  - Pick λ that gets best average performance



  - ☺ Tests on all 100% as yellow, so we can more reliably assess λ
  - ☹ Still picks a λ that's good at smoothing the 80% size, not 100%.
  - But now we can grow that 80% without trouble

78

N-fold Cross-Validation ("Leave One Out")



- Test each sentence with smoothed model from other N-1 sentences
- ☺ Still tests on all 100% as yellow, so we can reliably assess λ
- ☺ Trains on nearly 100% blue data ((N-1)/N) to measure whether λ is good for smoothing that

79

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

	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

Laplace-smoothed bigrams

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

	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

Reconstituted counts

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Compare with raw bigram counts

	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

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Add-1 estimation is a blunt instrument

- So add-1 isn't used for N-grams:
  - We'll see better methods
- But add-1 is used to smooth other NLP models
  - In domains where the number of zeros isn't so huge.

Unigram Smoothing Example

- Tiny Corpus, V=4; N=20

$$P_{LP}(w_i) = \frac{c_i + 1}{N + V}$$

Word	True Ct	Unigram Prob	New Ct	Adjusted Prob
eat	10	.5	?	?
British	4	.2	5	.21
food	6	.3	7	.29
happily	0	.0	?	?
	20	1.0	~20	1.0

Language Modeling

Interpolation, Backoff, and Web-Scale LMs

Backoff and Interpolation

- Sometimes it helps to use **less** context
  - Condition on less context for contexts you haven't learned much about
- **Backoff:**
  - use trigram if you have good evidence,
  - otherwise bigram, otherwise unigram
- **Interpolation:**
  - mix unigram, bigram, trigram

- Interpolation works better

Backoff and interpolation

- p(zombie | see the) vs. p(baby | see the)
  - What if count(see the ngram) = count(see the baby) = 0?
  - baby beats ngram as a unigram
  - the baby beats the ngram as a bigram
  - ∴ see the baby beats see the ngram ?
    (even if both have the same count, such as 0)

88

Class-Based Backoff

- Back off to the class rather than the word
  - Particularly useful for proper nouns (e.g., names)
  - Use count for the number of names in place of the particular name
  - E.g. < N | friendly > instead of < dog | friendly>

Linear Interpolation

- Simple interpolation

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

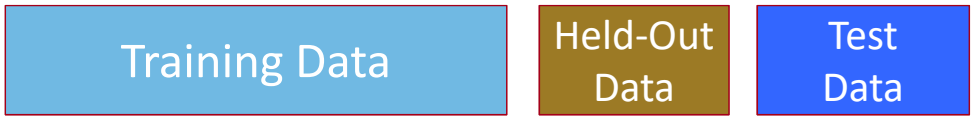
$$\sum_i \lambda_i = 1$$

- Lambdas conditional on context:

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1(w_{n-2}^{n-1})P(w_n|w_{n-2}w_{n-1}) + \lambda_2(w_{n-2}^{n-1})P(w_n|w_{n-1}) + \lambda_3(w_{n-2}^{n-1})P(w_n)$$

How to set the lambdas?

- Use a **held-out** corpus



- Choose λs to maximize the probability of held-out data:
  - Fix the N-gram probabilities (on the training data)
  - Then search for λs that give largest probability to held-out set:

Unknown words: Open versus closed vocabulary tasks

- If we know all the words in advanced
  - Vocabulary V is fixed
  - Closed vocabulary task
- Often we don't know this
  - **Out Of Vocabulary** = OOV words
  - Open vocabulary task
- Instead: create an unknown word token <UNK>
  - Training of <UNK> probabilities
    - Create a fixed lexicon L of size V
    - At text normalization phase, any training word not in L changed to <UNK>
    - Now we train its probabilities like a normal word
  - At decoding time
    - If text input: Use UNK probabilities for any word not in training

Huge web-scale n-grams

- How to deal with, e.g., Google N-gram corpus
- Pruning
  - E.g., only store N-grams with count > threshold.
    - Remove singletons of higher-order n-grams
- Efficient data structures, etc.

N-gram Smoothing Summary

- Add-1 smoothing:
  - OK for some tasks, but not for language modeling
- See text for
  - The most commonly used method:
    - Extended Interpolated Kneser-Ney
  - For very large N-grams like the Web:
    - Stupid backoff

94

Other Applications

- N-grams are not only for words
  - Characters
  - Sentences

95

More examples

- Yoav's blog post: http://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139
- 10-gram character-level LM:

```
First Citizen: Nay, then, that was hers, It
speaks against your other service: But since the
youth of the circumstance be spoken: Your uncle
and one Baptista's daughter.

SEBASTIAN: Do I stand till the break off.

BIRON:
Hide thy head.
```

96

Example from Kai-Wei Chang

Example: Language ID

- "Horses and Lukasiewicz are on the curriculum."
  - Is this English or Polish or ??
- Let's use n-gram models …
- Space of outcomes will be character sequences (x_1, x_2, x_3, …)

97

Language ID: Problem Formulation

- Let p(X) = probability of text X in English
- Let q(X) = probability of text X in Polish
- Which probability is higher?
  - (we'd also like bias toward English since it's more likely *a priori* – ignore that for now)

"Horses and Lukasiewicz are on the curriculum."

p(x_1=h, x_2=o, x_3=r, x_4=s, x_5=e, x_6=s, …)

98

Apply the Chain Rule

$p(x_1=h, x_2=o, x_3=r, x_4=s, x_5=e, x_6=s, \ldots)$
$= p(x_1=h)$ 4470/ 52108
$* p(x_2=o \mid x_1=h)$ 395/ 4470
$* p(x_3=r \mid x_1=h, x_2=o)$ 5/ 395
$* p(x_4=s \mid x_1=h, x_2=o, x_3=r)$ 3/ 5
$* p(x_5=e \mid x_1=h, x_2=o, x_3=r, x_4=s)$ 3/ 3
$* p(x_6=s \mid x_1=h, x_2=o, x_3=r, x_4=s, x_5=e)$ 0/ 3
$* \ldots = 0$

counts from Brown corpus

99

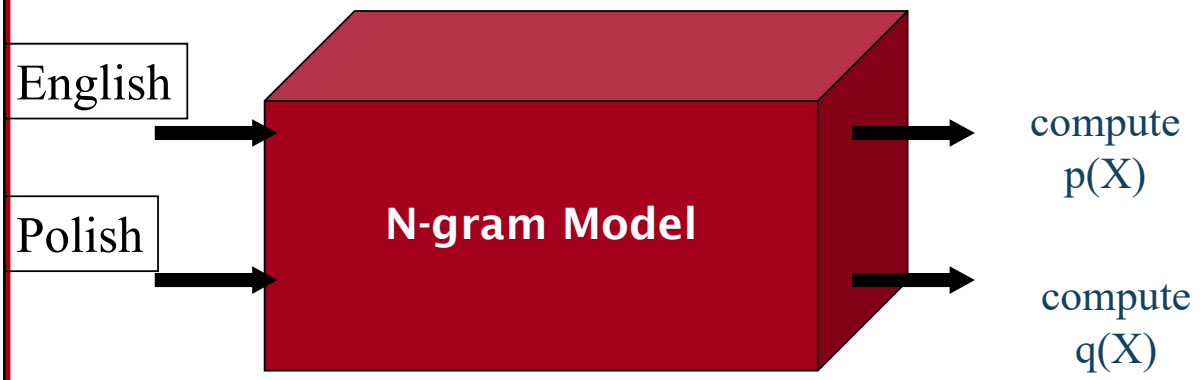
Use Bigrams

$p(x_1=h, x_2=o, x_3=r, x_4=s, x_5=e, x_6=s, \ldots)$
$\approx p(x_1=h)$ 4470/ 52108
$* p(x_2=o \mid x_1=h)$ 395/ 4470
$* p(x_3=r \mid x_1=h, x_2=o)$ 5/ 395
$* p(x_4=s \mid x_2=o, x_3=r)$ 12/ 919
$* p(x_5=e \mid x_3=r, x_4=s)$ 12/ 126
$* p(x_6=s \mid x_4=s, x_5=e)$ 3/ 485
$* \ldots = 7.3e\text{-}10 * \ldots$

counts from Brown corpus

100

English vs. Polish?



Compare!

101

Chapter Summary

- N-gram probabilities can be used to *estimate* the likelihood
  - Of a word occurring in a context (N-1)
  - Of a sentence occurring at all
- Perplexity can be used to evaluate the goodness of fit of a LM
- Smoothing techniques and backoff models deal with problems of unseen words in corpus
  - Improvement via algorithm versus big data