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.

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

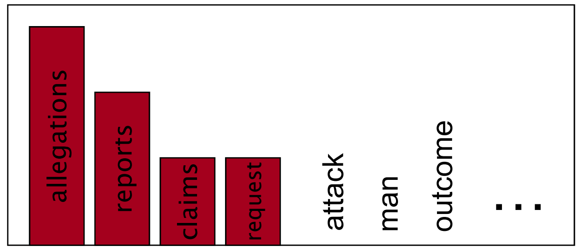
Smoothing: Add-one (Laplace) smoothing

The intuition of smoothing (from Dan Klein)

- When we have sparse statistics:

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

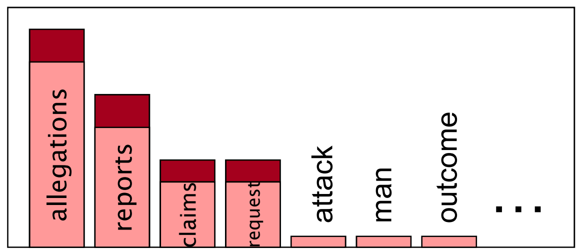
  7 total



- 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

14

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

15

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

16

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 <u>total</u> 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

17

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?

18

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

19

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

20

Add-Lambda Smoothing

- A large dictionary makes novel events too probable.
- To fix: Instead of adding 1 to all counts, add λ = 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)

21

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.

22

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?

Training | Test

- 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?

Training | Test

Dev. | Training

Pick λ that gets best results on this 20% …

… when we collect counts from this 80% and smooth them using add-λ smoothing.

Now use that λ to get smoothed counts from all 100% …

… and report results of that final model on test data.

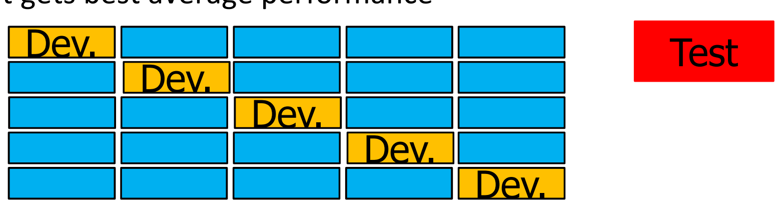
Large or small Dev set?

- 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%

25

Cross-Validation

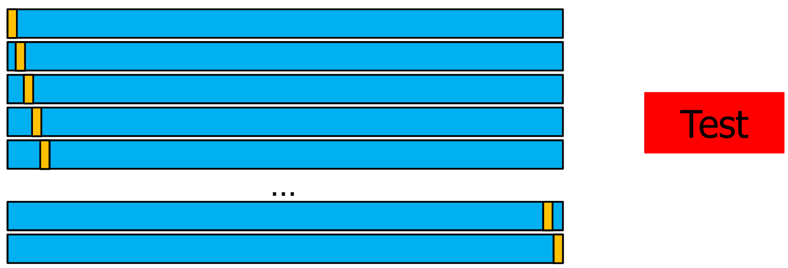
- Try 5 training/dev splits as below
  - Pick λ that gets best average performance



- ☺ 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

26

N-fold Cross-Validation (“Leave One Out”)



- Test <u>each</u> sentence with smoothed model from <u>other</u> 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

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)

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

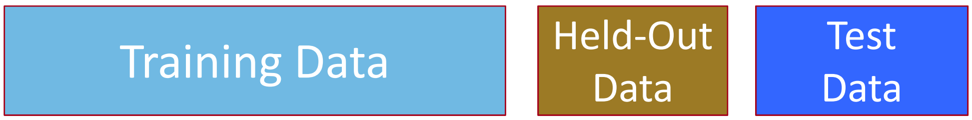
$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1 P(w_n|w_{n-2}w_{n-1}) \\ &+\lambda_2 P(w_n|w_{n-1}) \\ &+\lambda_3 P(w_n)\end{aligned} \qquad \sum_i \lambda_i = 1$$

- Lambdas conditional on context:

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

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

42

Other Applications

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

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.
```

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, …)

45

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`, …)

46

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

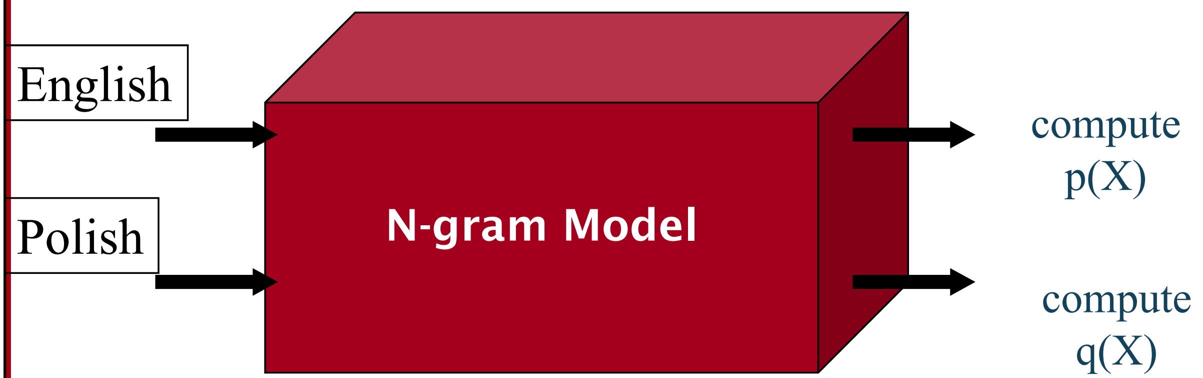
Use Bigrams

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

$\approx p(x_1=h)$	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
$* \; \ldots = 7.3\text{e-}10 * \ldots$	

counts from Brown corpus

English vs. Polish?



Compare!

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