CS 1671/2071
Human Language Technologies

Session 7: N-gram language models part 2

Michael Miller Yoder

February 3, 2025

University of Pittsburgh | School of Computing and Information

Course logistics: project

- Thanks for submitting project ideas! I will filter these (probably to ~14) and post all the options (anonymously) to the website today
- 21 original ideas were submitted!
- Most popular example projects:
  - Human vs AI-generated text classification
  - Reading comprehension question answering
- Next class session, **Wed Feb 4**, will be **project group match day**
- I will put project options all around the room. You will form groups of 2-4 students around projects

Course logistics: quiz and homework

- I will release another quiz on Canvas today
  - It will be due **this Thu Feb 6**
  - Looking over the reading is a great way to prepare
- Homework 2 has been released. Is due **Feb 20**
  - Build a text classification system to predict deception in a game (Diplomacy)

- Contact Jayden at

jserenari@pitt.edu

with any questions

Review activity

What are the 2 (related) tasks of language modeling?

Lecture overview: N-gram language models part 2

- Sampling sentences from language models
- The problem of zeros
- Laplace and Lidstone smoothing
- Interpolation and backoff
- Coding activity: sampling from smoothed ngram language models
- (If time permits) tf-idf and PPMI weighting

Sampling sentences from language models

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
```


Slide adapted from Jurafsky & Martin

Example n-gram language samples

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.

Slide adapted from Jurafsky & Martin

Wall Street Journal n-gram language model samples

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

The problem of zeros

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

Slide credit: David Mortensen

N-grams in the test set that weren't in the training set

Suppose our bigram LM, trained on Twitter, reads a document by the philosopher Wittgenstein:

Whereof one cannot speak, thereof one must be silent.

This contains the bigrams: whereof one, one cannot, cannot speak, speak [comma], [comma] thereof, thereof one, one must, must be, be silent.

Suppose "whereof one" never occurs in the training corpus (`train`) but whereof occurs 20 times. According to MLE, it's probability is

$$P(\text{one}|\text{whereof}) = \frac{c(\text{whereof, one})}{c(\text{whereof})} = \frac{0}{20} = 0$$

The probability of the sentence is the **product** of the probabilities of the bigrams. What happens if one of the probabilities is zero?

Two kinds of “zeros”

1. Completely unseen words in the test set
2. Words in unseen contexts in the test set

Unknown Words

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

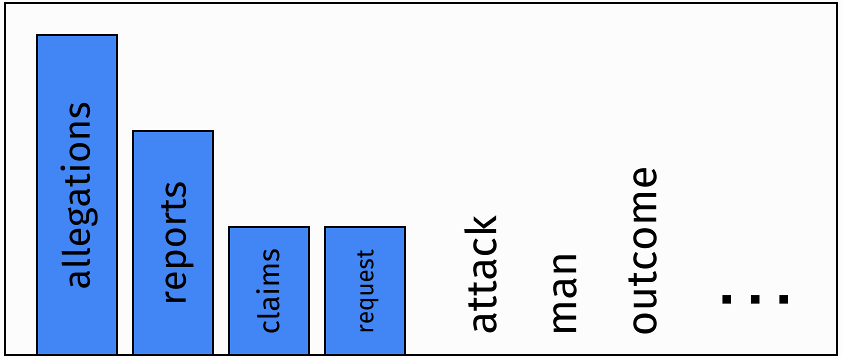
Slide credit: David Mortensen

Laplace and Lidstone smoothing

The intuition of smoothing

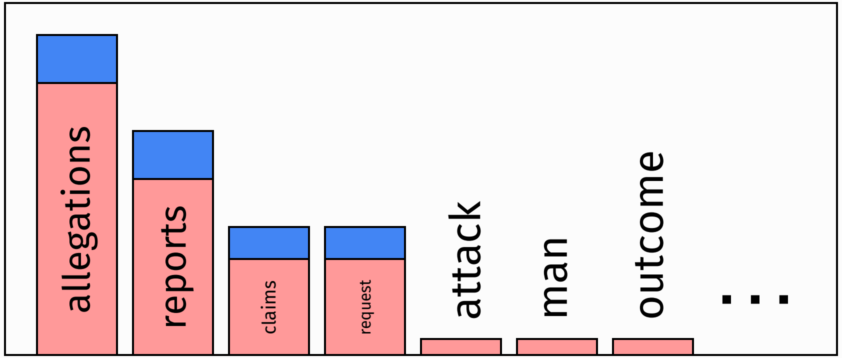
When we have sparse statistics:

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



Slide adapted from Jurafsky & Martin, Dan Klein

Laplace smoothing: Pretending that we saw each word once more

MLE estimate $P_{MLE}(w_i|w_{i-1}) = \dfrac{c(w_{i-1}, w_i)}{c(w_{i-1})}$

Add-1 estimate $P_{Add-1}(w_i|w_{i-1}) = \dfrac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + |V|}$

Where V is the vocabulary of the corpus.

Slide credit: David Mortensen

Solution for zeros: smoothing

Suppose we're considering 20000 word types

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

Slide credit: Lorraine Li

Solution for zeros: 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

09

Slide credit: Lorraine Li

Laplace smoothing is too blunt

Problem: A large dictionary makes rare words too probable.

Solution: instead of adding 1 to all counts, add $k < 0$.

How to choose k?

Slide credit: Lorraine Li

How to choose k?

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

Slide credit: Lorraine Li

How to choose *k*?

- Hyperparameter!
  - Try many *k* values on dev data and choose *k* that gives the lowest perplexity
  - Report result on test data
- Could tune this at the same time as *n* in n-gram LM

Slide adapted from Lorraine Li

Interpolation and backoff

Backoff and Interpolation Let You Use Less Context

Suppose you have a context you don't know much about (because you have seen few or no relevant ngrams). You can condition your probabilities for these contexts on shorter contexts you know more about.

Backoff Use trigram if you have good evidence, otherwise bigram, otherwise unigram.

Interpolation Mix unigrams, bigrams, and trigrams together in one (weighted) probability soup.

Interpolation works better; backoff is sometimes cheaper.

Slide credit: David Mortensen

Linear interpolation takes into account different n-grams

The simplest way to do this is to **not** take context into account. The lambdas, in the following formula, are weighting factors:

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

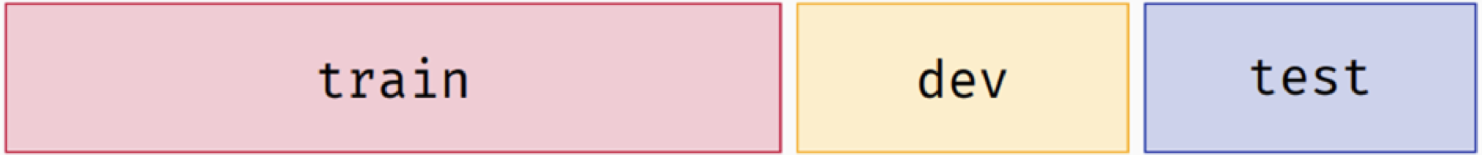
where

$$\forall i \; \lambda_i \geq 0 \wedge \sum_i^n \lambda_i = 1$$

That is, the lambdas must sum to one.

Slide credit: David Mortensen

Lambdas Are Tuned Using a Held-Out dev Set



Choose λs to maximize the probability of held-out data (`dev`):

- Fix the ngram probabilities (on `train`)
- Then search for λs that give the largest probability to `dev`:

Slide credit: David Mortensen

Web-Scale Ngrams

How to deal with, e.g., Google N-gram corpus Pruning

- Only store N-grams with count > threshold.
- Remove singletons of higher-order n-grams
- Entropy-based pruning

Efficiency

- Efficient data structures like tries
- Store words as indexes, not strings

Stupid Backoff is Stupid but Efficient

- If higher-order n-grams have 0 count, “back off” to lower-order n-grams
- “Stupid” because doesn’t bother making it a true probability distribution and summing to one

$$S(w_i|w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{c(w_{i-k+1}^{i})}{c(w_{i-k+1}^{i-1})} & \text{if } c(w_{i-k+1}^{i}) > 0 \\ 0.4S(w_i|w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{c(w_i)}{N}$$

Coding activity: sample from n-gram LMs

Sample from n-gram language models on JupyterHub

- Click on this nbgitpuller link
  - Or find the link on the course website
- Open **session7_sample_ngram_lm.ipynb**