CS 1671/2071
Human Language Technologies

Session 7: N-gram language models part 2

Michael Miller Yoder

February 3, 2025

School of Computing and Information

Course logistics

- Thanks for submitting project ideas! I will collect those and post all the options (anonymously) on Canvas
  - 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
- 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

SPRINTERNSHIP

IN PARTNERSHIP WITH BREAKTHROUGH TECH

With a mission to launch a new generation of diverse tech talent, Sprinternship offers a paid micro-internship experience, immersing you in technology roles, allowing you to collaborate with students from different colleges, and helping you build your network of tech professionals.

3-week internship - May 12 - 30, 2025

Program Highlights

- Paid 40-hours a week internship to grow your skills
- Work in a team environment, a cohort of other students to tackle a challenge project determined by your host company
- Gain a resume boosting experience to advance your career and expertise
- Participate in professional development and technical training prep workshops gaining a year-long subscription to a technical training platform

Learn and Grow Technical Skills In:

{JavaScript} JS HTML CSS And More...

Scan for the Interest Form

Contact with Questions or To Learn More!

ZS@innovatepgh.com

wwww.innovatepgh.com/sprinternship

Review activity

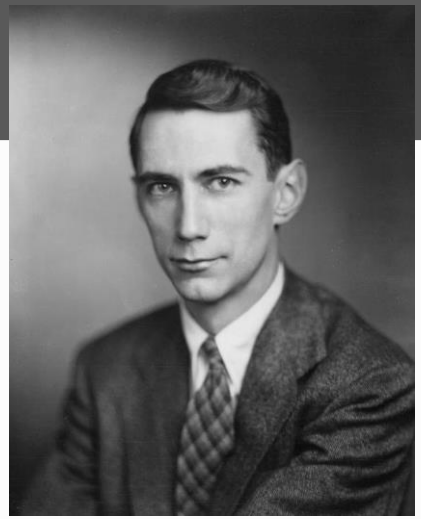
What are the 2 (related) tasks of language modeling?

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

Slide adapted from Jurafsky & Martin

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

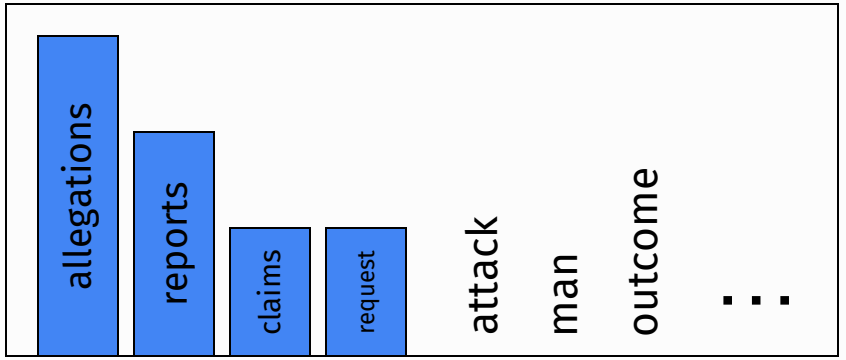
Slide credit: David Mortensen

Laplace and Lidstone smoothing

The intuition of smoothing

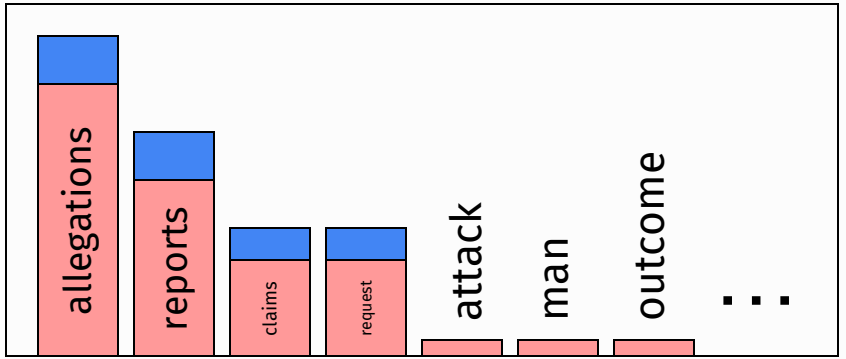
When we have sparse statistics:

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

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

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

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

Slide credit: David Mortensen

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**

Tf-idf weighting

Raw frequency is a bad representation

- The co-occurrence matrices we have seen represent each cell by word frequencies
  - Whether in term-document or term-term matrices
- Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information.
- But overly frequent words like *the*, *it*, or *they* are not very informative about the context
  - 2 documents that use a lot of *the* are not necessarily similar
- It's a paradox! How can we balance these two conflicting constraints?

Slide adapted from Jurafsky & Martin

Weighting words in term-document and term-term matrices

- **tf-idf** (term frequency-inverse document frequency)
  - For representing documents with their most unique words (for text classification, information retrieval)
  - Term-document matrix
- **PPMI** (positive pointwise mutual information)
  - For finding associations between words (which appear more often together than chance?)
  - Term-term matrix

Slide adapted from Jurafsky & Martin

Term frequency (tf)

$tf_{t,d} = count(t,d)$

Instead of using raw count, we squash a bit:

$tf_{t,d} = \log_{10}(count(t,d)+1)$

Slide adapted from Jurafsky & Martin

Document frequency (df)

df_t is the number of documents term *t* occurs in.

(note this is not collection frequency, which is the total count across all documents)

"Romeo" is very distinctive for one Shakespeare play:

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

Slide adapted from Jurafsky & Martin

Inverse document frequency (idf)

$$\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right)$$

- N is the total number of documents in the collection
- Documents can be whatever you want! (Full documents, paragraphs, etc)

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

Slide adapted from Jurafsky & Martin

tf-idf Controls for Frequent but Uninformative Words

Some words are very common in a given document because they are common across all documents (e.g., *the*). They are not discriminative. `tf-idf` (product of term frequency and inverse document frequency) addresses this:

$$\text{tf}_{t,d} = \log_{10}(\text{count}(t,d) + 1)$$

$$\text{idf}_f = \log_{10} \frac{N}{\text{df}_t}$$

$$\text{tf-idf}(t,d) = \text{tf}_{t,d} \cdot \text{idf}_t$$

Slide adapted from Jurafsky & Martin, David Mortensen

Final tf-idf weighted values

Raw counts

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

tf-idf:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Positive pointwise mutual information (PPMI)

The Problem with Raw Counts

The **to** in **to walk** doesn't tell us as much about **walk** as the **slowly** in **walk slowly**.

Slide credit: David Mortensen

Problem with Raw Counts

- Raw word frequency is not a great measure of association between words.
- It is very skewed: “the” and “of” are very frequent, but maybe not the most discriminative.
- We would rather have a measure that asks whether a context word is **particularly informative** about the target word.

Positive Pointwise Mutual Information (PPMI)

Slide credit: David Mortensen

Pointwise mutual information

- Do events *x* and *y* co-occur more than if they were independent?

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- PMI between 2 words [Church+Hanks 1989]
  - Do words *x* and *y* co-occur more than if they were independent?

$$PMI(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

Slide adapted from Jurafsky & Martin

- In computational linguistics, PMI has been used for finding collocations and associations between words.

word 1	*word 2*	*count word 1*	*count word 2*	count of co-occurrences	PMI
puerto	rico	1938	1311	1159	10.0349081703
hong	kong	2438	2694	2205	9.72831972408
los	angeles	3501	2808	2791	9.56067615065
carbon	dioxide	4265	1353	1032	9.09852946116
prize	laureate	5131	1676	1210	8.85870710982
san	francisco	5237	2477	1779	8.83305176711
nobel	prize	4098	5131	2498	8.68948811416
ice	hockey	5607	3002	1933	8.6555759741
star	trek	8264	1594	1489	8.63974676575
car	driver	5578	2749	1384	8.41470768304
it	the	283891	3293296	3347	-1.72037278119
are	of	234458	1761436	1019	-2.09254205335
this	the	199882	3293296	1211	-2.38612756961
is	of	565679	1761436	1562	-2.54614706831
and	of	1375396	1761436	2949	-2.79911817902
a	and	984442	1375396	1457	-2.92239510038
in	and	1187652	1375396	1537	-3.05660070757
to	and	1025659	1375396	1286	-3.08825363041
to	in	1025659	1187652	1066	-3.12911348956
of	and	1761436	1375396	1190	-3.70663100173

Slide credit: David Mortensen

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic:
  - Things are co-occurring less than we expect by chance
  - Unreliable without enormous corpora
    - Imagine w_1 and w_2 whose probability is each 10^{-6}.
    - Hard to be sure $p(w_1, w_2)$ is significantly different than 10^{-12}.
  - Furthermore it's not clear people are good at "unrelatedness".
- So we just replace negative PMI values by 0.

$$\text{PPMI}(w, c) = max\left(\log_2 \frac{p(w, c)}{p(w)p(c)}, 0\right)$$

Slide credit: David Mortensen

Computing PPMI on a Term-Context Matrix

- We have matrix F with V rows (words) and C columns (contexts) (in general $C = V$)
- f_{ij} is how many times word w_i co-occurs in the context of the word c_j.

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{V}(\sum_{j=1}^{C} f_{ij})}$$

$$p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{V}(\sum_{j=1}^{C} f_{ij})} \qquad p_{*j} = \frac{\sum_{i=1}^{V} f_{ij}}{\sum_{i=1}^{V}(\sum_{j=1}^{C} f_{ij})}$$

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}} \qquad ppmi_{ij} = \max(pmi_{ij}, 0)$$

Slide credit: David Mortensen

Worked Example: Computing PPMI from Term-Context Matrix (Part I)

	computer	data	pinch	result	sugar	
apricot	0	0	1	0	1	2
pineapple	0	0	1	0	1	2
digital	2	1	0	1	0	4
information	1	6	0	4	0	11
	3	7	2	5	2	19

$$p(w = information, c = data) = \frac{6}{19} = 0.32$$

$$p(w = information) = \frac{11}{19} = 0.58 \quad p(c = data) = \frac{7}{19} = 0.37$$

$$pmi(information, data) = \log_2 \frac{0.32}{0.37 \cdot 0.58} \approx 0.58$$

Slide credit: David Mortensen

Worked Example: Computing PPMI from Term-Context Matrix (Part II)

$PPMI(w, c)$

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.32	-	0.47	-

Slide credit: David Mortensen

Issues with PPMI

- PMI is biased toward infrequent events.
- Very rare words have very high PMI values.
- Two solutions:
  - Give rare words slightly higher probabilities
  - Use add-one smoothing (which has a similar effect)

Slide credit: David Mortensen