Lecture 4: N-Gram Language Models (II)

Instructor: Stephanie Schoch

CSCI 680: Natural Language Processing
Department of Computer Science, College of William & Mary
September 2, 2026

WILLIAM & MARY

Many slides are adapted from Jurafsky & Martin (3rd ed.)

Lecture Outline

- Recap
- Evaluating N-Gram Language Models
  - Test/Train Set
  - Perplexity
- Sampling & Generalization
- Smoothing

N-Gram Language Models: Recap

A **language model** assigns probabilities to sequences of words

by computing either $P(\mathbf{w})$ or $P(w_n \mid w_1, w_2, w_3, w_4, \ldots, w_{n-1})$

Chain Rule:

$$P(w_{1:n}) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{1:2}) \cdots P(w_n \mid w_{1:n-1})$$

$$= \prod_{k=1}^{n} P(w_k \mid w_{1:k-1})$$

$$P(\texttt{blue}|\texttt{The water of Walden Pond is so beautifully})$$

Markov Assumption:

$$\approx P(\texttt{blue}|\texttt{beautifully})$$

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1})$$

N-Gram Language Models: Recap

Bigram LM

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

(the probability of a word is conditioned on the previous word)

MLE for Computing Bigram Probabilities

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

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

Extrinsic Evaluation of N-Gram Models

To compare models A and B:

1. Put each model in a real task
    a. Machine Translation, speech recognition, etc.
2. Run the task, get a score for A and for B
    a. How many words translated correctly
    b. How many words transcribed correctly
3. Compare accuracy for A and B

Intrinsic Evaluation of N-Gram Models

- Extrinsic evaluation not always possible
  - Expensive, time-consuming
  - Doesn't always generalize to other applications
- Intrinsic evaluation: **perplexity**
  - Directly measures language model performance at predicting words.
  - Doesn't necessarily correspond with real application performance
  - But gives us a single general metric for language models
  - Useful for large language models (LLMs) as well as n-grams

Training Sets and Test Sets

- 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; different from training set.
    - Intuition: we want to measure generalization to unseen data
- An **evaluation metric** (like **perplexity**) tells us how well our model does on the test set.

Choosing Training and Test Sets

- If we're building an LM for a specific task
  - The test set should reflect the task language we want to use the model for
- If we're building a general-purpose model
  - We'll need lots of different kinds of training data
  - We don't want the training set or the test set to be just from one domain or author or language.

Training on the Test Set

- We can't allow test sentences into the training set
  - Or else the LM will assign that sentence an artificially high probability when we see it in the test set
  - And hence assign the whole test set a falsely high probability.
  - Making the LM look better than it really is
- This is called **"Training on the test set"**
- Bad science!

Dev Sets

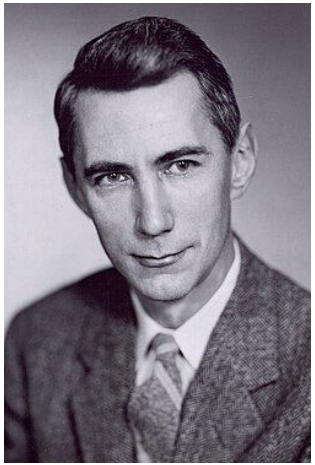
- If we test on the test set many times we might implicitly tune to its characteristics
  - Noticing which changes make the model better.
- So we run on the test set only once, or a few times
- That means we need a third dataset:
  - A **development test set** or, **dev set**.
  - We test our LM on the dev set until the very end
  - And then test our LM on the test set once

DEMO: The Shannon Game

Intuition of perplexity as an evaluation metric (I):
How good is our language model?

- **Intuition:** A good LM prefers "real" sentences
  - Assign higher probability to “real” or “frequently observed” sentences
  - Assigns lower probability to “word salad” or “rarely observed” sentences?

Intuition of perplexity as an evaluation metric (II): Predicting Upcoming Words


Claude Shannon

- **The Shannon Game:** How well can we predict the next word?
- Once upon a _____
- That is a picture of a _____
  - For breakfast I ate my usual _____

time	0.9
dream	0.03
midnight	0.02
…	
and	1e-100

- Unigrams are terrible at this game (Why?)

A **good** language model is one that assigns a higher probability to the next word that actually occurs

Intuition of perplexity as an evaluation metric (III):

The best language model is one that best predicts the entire unseen test set

- We said: a good LM is one that assigns a higher probability to the next word that actually occurs.
  - Let's generalize to all the words!
- The best LM assigns high probability to the entire test set.
- When comparing two LMs, A and B
  - We compute P_A(test set) and P_B(test set)
  - The better LM will give a higher probability to (=be less surprised by) the test set than the other LM.

Intuition of perplexity as an evaluation metric (IV):
Use perplexity instead of raw probability

- Probability depends on size of test set
  - Probability gets smaller the longer the text
  - Better: a metric that is **per-word**, normalized by length
- **Perplexity** is the inverse probability of the test set, normalized by the number of words

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

Minimizing perplexity is the same as maximizing probability

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

Intuition of perplexity as an evaluation metric (VI):

N-grams

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

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

Chain rule:

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

Bigrams:

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

Intuition of perplexity as an evaluation metric (VII): Weighted Average Branching Factor

- Perplexity is also the **weighted average branching factor** of a language.
- **Branching** factor: number of possible next words that can follow any word
- **Example:** Deterministic language L = {red,blue, green}
- Branching factor = 3 (any word can be followed by red, blue, green)
- Now assume LM A where each word follows any other word with equal probability ⅓
  - Given a test set T = "red red red red blue"
  - **$\text{Perplexity}_A(T) = P_A(\text{red red red red blue})^{-1/5} = ((1/3)^5)^{-1/5} = (1/3)^{-1} = 3$**

Intuition of perplexity as an evaluation metric (VII):
Weighted Average Branching Factor

- But now suppose red was very likely in training set, such that for LM B:
  - P(red) = .8 p(green) = .1 p(blue) = .1
- We would expect the probability to be higher, and hence the perplexity to be smaller:
  - $\text{Perplexity}_B(T) = P_B(\text{red red red red blue})^{-1/5}$
    $= (.8*.8*.8*.8*.1)^{-1/5} = .04096^{-1/5} = .527^{-1} = 1.89$

Holding test set constant:

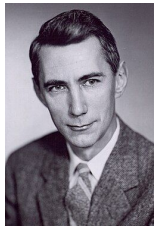
Lower perplexity = better language model

Training 38 million words, test 1.5 million words, WSJ

N-Gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Sampling & Generalization

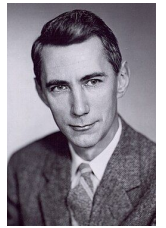
The Shannon (1948) Visualization Method:
Sample Words from an LM


Claude Shannon

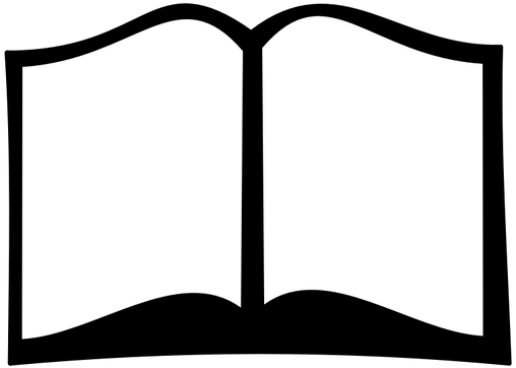
- **Unigram:**
- REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE.

- **Bigram:**
- THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED.

How Shannon Sampled Those Words in 1948

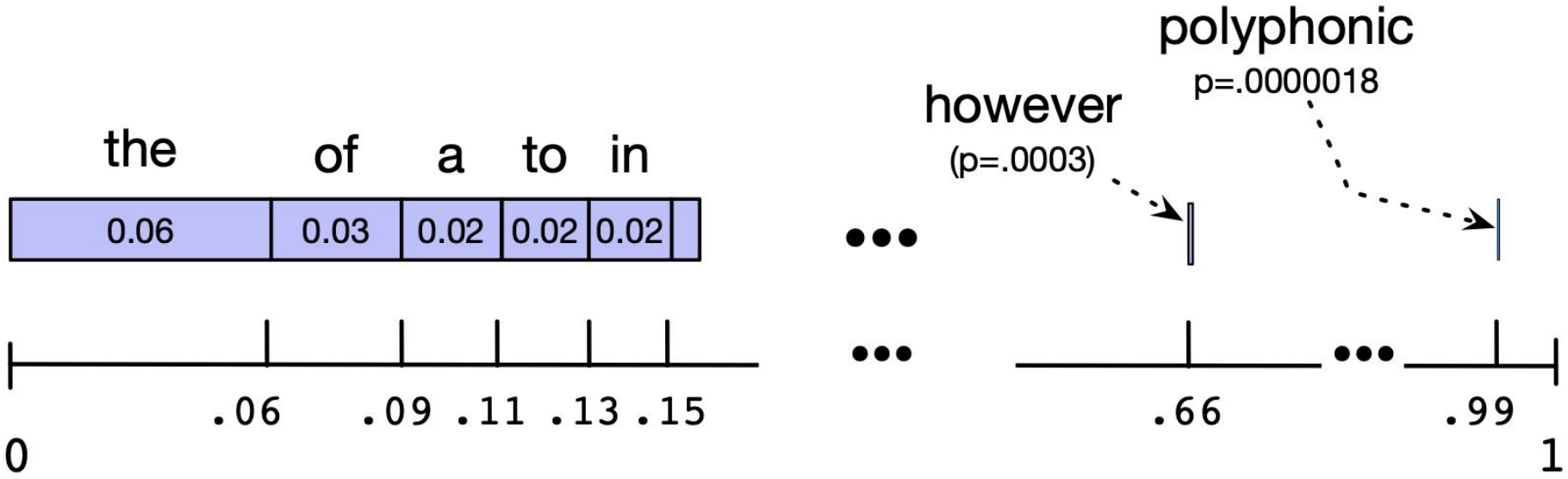


Claude Shannon



"Open a book at random and select a letter at random on the page. This letter is recorded. The book is then opened to another page and one reads until this letter is encountered. The succeeding letter is then recorded. Turning to another page this second letter is searched for and the succeeding letter recorded, etc."

Sampling a Word from a Distribution



Visualizing Bigrams the Shannon Way

- Choose a random bigram (<s>, w) according to its probability p(w|<s>)
- Now choose a random bigram (w, x) according to its probability p(x|w)
- 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

Note: There Are Other Sampling Methods

- Used for neural language models
- Many of them avoid generating words from the very unlikely tail of the distribution
- We'll discuss when we get to neural LM decoding:
  - Temperature sampling
  - Top-k sampling
  - Top-p sampling

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 a 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)
- That sparsity is even worse for 4-grams, explaining why our sampling generated actual 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? These 3-gram sentences are sampled from an LM trained on who?

1) 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

2) This shall forbid it should be branded, if renown made it empty.

3) “You are uniformly charming!” cried he, with a smile of associating and now and then I bowed and they perceived a chaise and four to wish for.

Choosing Training Data

- If task-specific, use a training corpus that has a similar genre to your task.
  - If legal or medical, need lots of special-purpose documents
- Make sure to cover different kinds of dialects and speaker/authors.
  - Example: *African-American Vernacular English (AAVE)*
    - One of many language varieties
    - Can include the auxiliary verb **finna** that marks immediate future tense:
    - "My phone finna die"

The Perils of Overfitting

- N-grams only work well for word prediction if the test corpus looks like the training corpus
  - But even when we try to pick a good training corpus, the test set will surprise us!
  - 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:
… ate lunch
… ate dinner
… ate a
… ate the

P(“breakfast” | ate) = 0

Test set
… ate lunch
… ate breakfast

Zero probability bigrams

- Bigrams with zero probability
  - Will hurt our performance for texts where those words appear!
  - And mean that we will assign 0 probability to the test set!
- And hence we cannot compute perplexity (can’t divide by 0)!

Smoothing, Interpolation, and Backoff

The Problem of Sparsity

- Maximum likelihood estimation has a problem
  - Sparsity! Our finite training corpus won't have some perfectly fine sequences
    - Perhaps it has "ruby" and "slippers"
    - But happens not to have "ruby slippers"

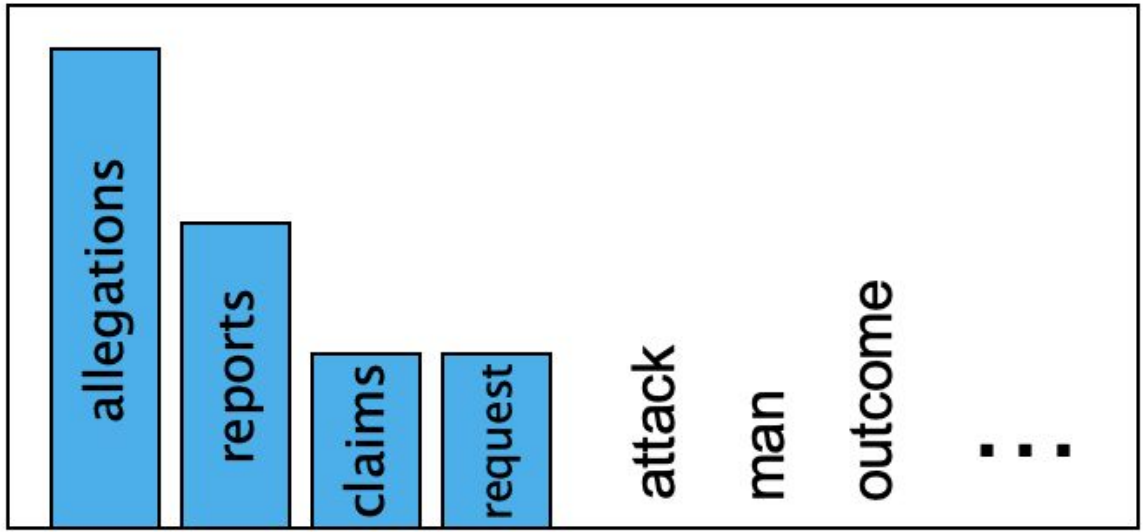
Sparsity Can Take Many Forms

Verbs have direct objects; those can be sparse too!

Count(w | denied the)

Counts:
- 3 allegations
- 2 reports
- 1 claims
- 1 request
- 0 attack
- 0 outcome

7 total

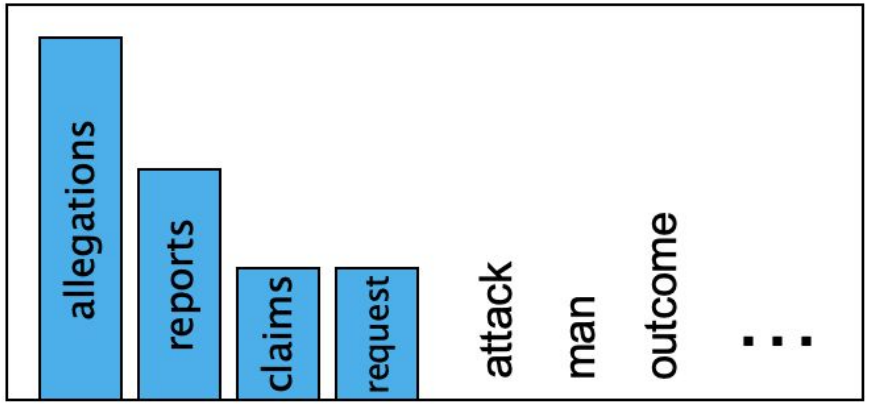


The Intuition of Smoothing

When we have sparse statistics:

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

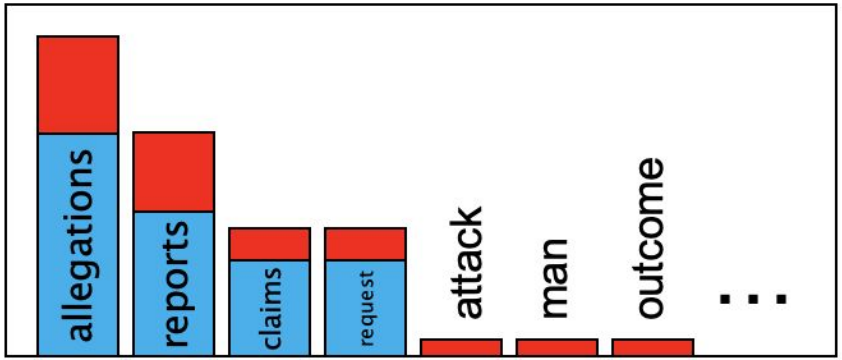
7 total



Steal probability mass to generalize better

Count(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: $$P_{\text{MLE}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

Add-1 estimate:

$$P_{\text{Laplace}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{\sum_w (C(w_{n-1}w)+1)} = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

$$C(w_{n-1}w_n) + 1$$

	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

Berkeley Restaurant Corpus: Laplace smoothed bigrams

$$P_{\text{Laplace}}(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{\sum_w (C(w_{n-1}w)+1)} = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

	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

Visualization Technique: Reconstituted counts

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

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

Reconstituted counts C*

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

C*:

	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

original

	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

add-1 smoothed

	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:
  - Generally we use interpolation or backoff instead
- But add-1 is used to smooth other NLP models
  - Or we can use **add-k** where 0<k<1
  - It's used in naive bayes text classification
  - In domains where the number of zeros isn’t so huge.

Backoff and Interpolation

- Sometimes it helps to use a simpler model
  - Condition on less context for contexts you know less about
- **Backoff:**
  - If enough evidence, use trigram $P(w_n|w_{n-2}w_{n-1})$
  - If not, use bigram $P(w_n|w_{n-1})$
  - Else unigram $P(w_n)$

- **Interpolation:**
  - mix unigram, bigram, trigram

- Interpolation works better

Linear Interpolation

Simple interpolation:

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

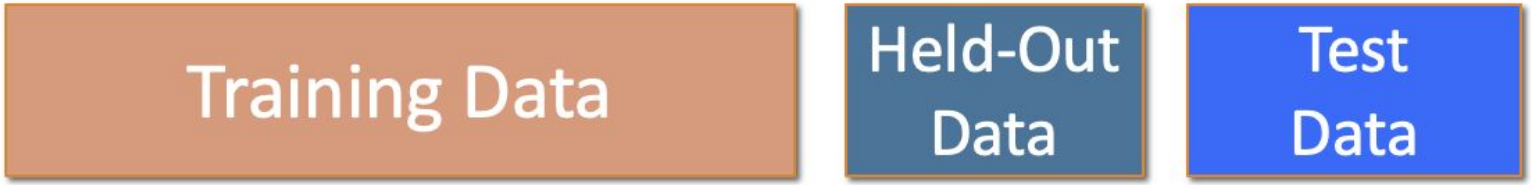
$$\sum_i \lambda_i = 1$$

Lambdas conditional on context:

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

Linear Interpolation

Use a **held-out** corpus



Choose λs to maximize 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

Backoff

- Suppose you want:
  - P(pancakes| delicious soufflé)
- If the trigram probability is 0, use the bigram
  - P(pancakes| soufflé)
- If the bigram probability is 0, use the unigram
  - P(pancakes)
- Complication: need to discount the higher-order ngram so probabilities don't sum higher than 1 (e.g., Katz backoff)

Stupid Backoff

- Backoff without discounting (not a true probability)

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

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

Summary: What to do if you never saw an n-gram in training

- **Smoothing:** Pretend you saw every n-gram one (or k) times more than you did
  - A blunt instrument (replacing a lot of zeros) but sometimes useful
- **Backoff:** If you haven't seen the trigram, use the (weighted) bigram probability instead
  - Weighting is messy; "stupid" backoff works fine at web-scale
- **Interpolation:** (weighted) mix of trigram, bigram, unigram
  - Usually the best! We also use interpolation to combine multiple LLMs

TODOs:

- Continue brainstorming for your project.

Next Lecture: **Text Classification (I)**