

CSE 6331 Natural Language Processing

Language Models

2026 Fall

An Example

Today in Arlington, TX, it's 45F and sunny.

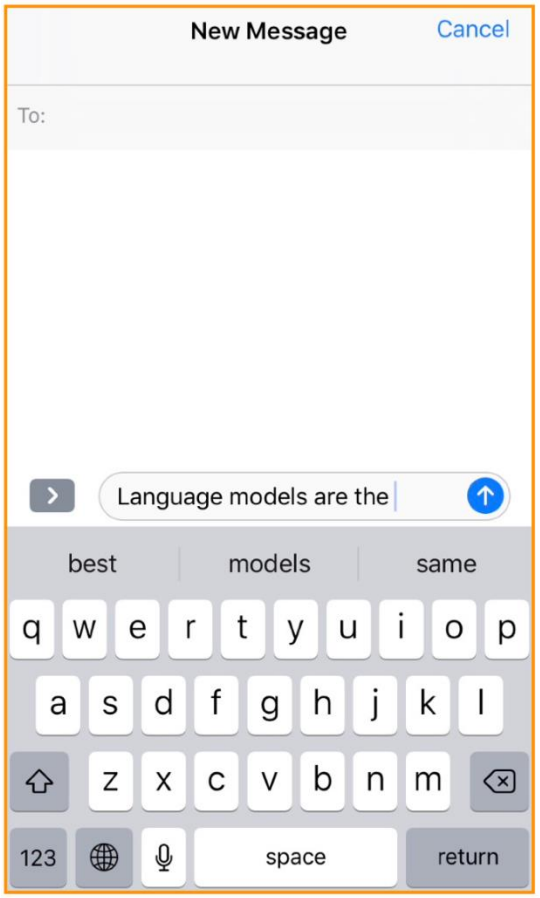
vs.

Today in Arlington, TX, it's 45F and blue.

- Both are grammatical
- But which is more likely?

Language Models are Everywhere





And Many Applications

- Predicting words is important in many situations

  - Machine translation

    P(a **smooth** finish) > *P*(a **flat** finish)

  - Speech recognition/Spell checking

    P(high school **principal**) > *P*(high school **principle**)

  - Information extraction, question answering

Impact on Downstream Applications

Language Resources	Adaptation	Word		PP
		Cor.	Acc.	
1. Doc-A		54.5%	45.1%	49972
2. Trans-C(L)		63.3%	50.6%	1856.5
3. Trans-B(L)		70.2%	60.3%	318.4
4. Trans-A(S)		70.4%	59.3%	442.3
5. Trans-B(L)+Trans-A(S)	CM	72.6%	63.9%	225.1
6. Trans-B(L)+Doc-A	KW	72.1%	64.2%	247.5
7. Trans-B(L)+Doc-A	KP	73.1%	65.6%	259.7
8. Trans-A(L)		75.2%	67.3%	148.6

documents vs transcripts (Miki et al. 2006)

New Approach to Language Modeling Reduces Speech Recognition Errors by Up to 15%

Ankur Gandhe

Principal, Applied Scientist

Alexa Speech group, Amazon

What is a Language Model?

- Probabilistic model of a sequence of words.
  - How likely is a given phrase/sentence/paragraph/ document?

- Joint probability distribution:

$$P(w_1, w_2, \ldots, w_n)$$

Quiz: Sequence (1 pt)

- Which of the following can be considered a sequence of words?
- a) a document
- b) a sentence
- c) a paragraph
- d) all of the above

Chain Rule

$$P(X_1, X_2, \ldots X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)\ldots$$
$$= \prod_{i=1}^{n} P(X_i|X_1, \ldots, X_{i-1})$$

- Sentence: “the sun rises and shines”

P(the sun rises and shines) = P(the) * P(sun | the) * P(rises | the sun) * P(and | the sun rises) *

P(shines | the sun rises and)

Estimating The Probabilities

$$P(rises \mid the\ sun) = \frac{count(the\ sun\ rises)}{count(the\ sun)}$$

$$P(and \mid the\ sun\ rises) = \frac{count(the\ sun\ rises\ and)}{count(the\ sun\ rises)}$$

$$\vdots$$

Maximum Likelihood Estimate (MLE)

- With a vocabulary of size V,
  - number of sequences of length $n = V^n$
- Typical vocab size of 40k words (English):
  - even just considering sentences of <=11 words results in $4*10^{50}$ different sentences (number of atoms on earth only ~10^{50})
- Use a corpus to count these word sequences

Quiz: Number of Sentences (3 pt)

- Assuming there are 40k distinct English words, and English sentences have at most 11 words, then how many possible different English sentences are there? Please show your math steps!

Markov Assumption

- Use only recent past in the sequence to predict next word

- Reduce the number of estimated parameters in exchange for ***model capacity*** (can model longer sentences now!)

- 1st order:
  $$P(shines|the\ sun\ rises\ and) \cong P(shines|and)$$
- 2nd order:
  $$P(shines|the\ sun\ rises\ and) \cong P(shines|rises\ and)$$

K-TH ORDER MARKOV CHAIN

- Consider only the last k words from the context:

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

which implies the probability of a sequence is:

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

k+1 gram

N-gram Language Models

- Unigram $$P(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} P(w_i)$$

- Bigram $$P(w_1, w_2, \ldots w_n) = \prod_{i=1}^{n} P(w_i | w_{i-1})$$

w_0 is the start of sentence, $P(w_0) = 1$

- And trigram, 4-gram, … N-gram, etc.

- Larger the N, more accurate and better the language model (but at a higher cost)
- Remember the data is *infinite*!

Text Generations using N-grams

Unigram *release millions See ABC accurate President of Joe Will cheat them a CNN megynkelly experience @ these word out- the*

Bigram *Thank you believe that @ ABC news, New Hampshire tonight and the false editorial I think the great people Nikki Haley . ''*

Trigram *We are going to MAKE AMERICA GREAT AGAIN! #MakeAmericaGreatAgain https: / /t.co/DjkdAzT3WV*

$$\arg \max_{(w_1, w_2, \ldots, w_n)} \Pi_{i=1}^{n} P(w_i | w_{<i})$$

Text Generations using N-grams

Unigram *release millions See ABC accurate President of Joe Will cheat them a CNN megynkelly experience @ these word out- the*

Bigram *Thank you believe that @ ABC news, New Hampshire tonight and the false editorial I think the great people Nikki Haley . "*

Trigram *We are going to MAKE AMERICA GREAT AGAIN! #MakeAmericaGreatAgain https: / /t.co/DjkdAzT3WV*

Typical LMs are not sufficient to handle long-range dependencies:

"Alice/Bob could not go to work that day because she/he had a doctor's appointment"

Quiz: Corpus vs Model (2 pts)

- Below is the sizes of the various corpora used in the previous example of speech recognition.

Table 1. Language Resources

Name	CC	Type	Words	Sentences
Trans-A(S)	A	Transcripts	46K	5.5K
Doc-A	A	Documents	1.6M	85K
Trans-B(L)	B	Transcripts	692K	86K
Trans-C(L)	C	Transcripts	499K	83K
Trans-A(L)	A	Transcripts	444K	61K

- Doc-A is call summaries collected at call center A, and Trans-A(S) is the *small* call transcripts collected at call center A, which is much smaller than Doc A. Why is the speech recognition model trained on Trans-A(S) much more accurate than Doc-A (59% vs 45%)?

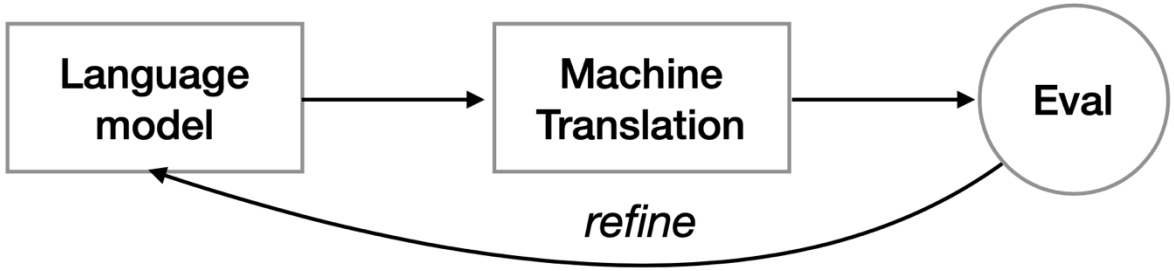
Evaluating Language Models

- A good language model should assign higher probability to typical, grammatically correct sentences.

- Research process:
  - Train parameters on a suitable training corpus
    - Assumption: observed sentences ~ good sentences
  - Test on *different, unseen* corpus
    - Training on **any part** of test set not acceptable!
  - Evaluation metric

Extrinsic Evaluation

- Train LM → Apply to task → Observe accuracy



- Directly optimized for downstream tasks
  - Higher accuracy → better model
- Expensive, time consuming
- Hard to optimize downstream objective (indirect feedback)

Perplexity (per word)

- Measures how well a probability distribution (or a model) predicts a sample
- For a corpus S with sentences S_1, S_2, … S_n.

$$\text{ppl(S)} = 2^x \text{ where } x = -\frac{1}{W}\sum_{i=1}^{n}\log_2 P(S_i)$$

A form of cross entropy

where W is the total number of words in test corpus

- Unigram model:

$$x = -\frac{1}{W}\sum_{i=1}^{n}\sum_{j=1}^{m} log_2 P(w_j^i)$$

j^{th} word in i^{th} sentence

- Minimizing perplexity ~ maximizing probability

Intuition of Perplexity

- If our n-gram model (with vocabulary V) has the following probability:

$$P(w_i|w_{i-n},...w_{i-1}) = \frac{1}{|V|} \quad \forall w_i$$

what is the perplexity on the test corpus?

$$\text{ppl} = 2^{-\frac{1}{W}W*log(1/|V|)} = |V|$$

- The model is "fine" with observing any word at every step!

Pros and Cons of Perlexity

Pros	Cons
Fast to compute, eliminate "bad" models that can't perform well in expensive real-world testing	Not good for final evaluation: measures model's confidence, not accuracy
Model's uncertainty/information density is useful information	Not fair comparison across models trained on different datasets
Statistically robust (not easily influenced by a single outlier sentence in the dataset)	Can reward models trained on toxic or outdated dataset

Quiz: PPL of Bigrams (4 pts)

- Given the following training corpus:

  S1: you have five apples

  S2: you have no five

  S3: no apples have you

- What is the ppl of the bigram language model trained from above on this test sentence:

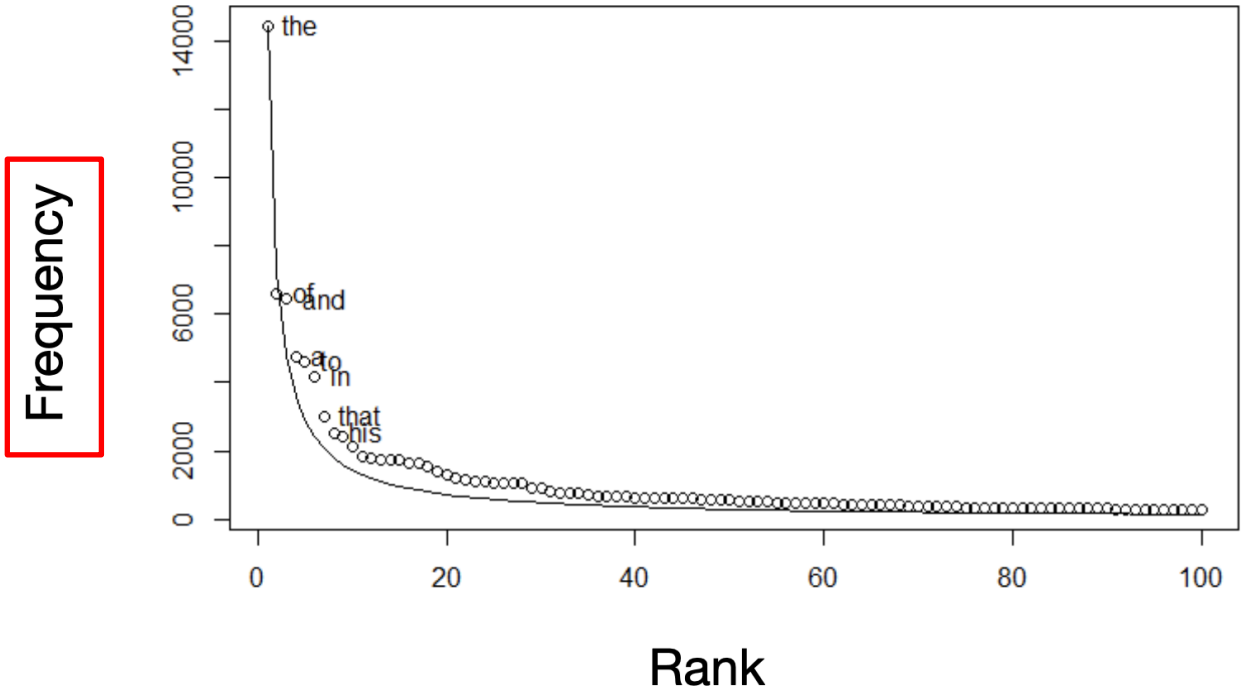
  S4: you have no apples

$$\text{ppl(S)} = 2^x \text{ where } x = -\frac{1}{W}\sum_{i=1}^{n}\log_2 P(S_i)$$

Generalization of N-grams

- Not all n-grams are observed in training data!
- Test corpus may contain some n-grams with zero probability under our model
  - Training data: *Google News*
  - Test data: *Shakespeare*
  - P (*affray* | *voice doth us*) = 0 → P (test set) = 0
  - Undefined perplexity

Sparsity in Languages



$$freq \propto \frac{1}{rank}$$

Zipf's Law

- Long tail of infrequent words
- Most finite-size corpora will have this problem
- Zipf's law is a kind of "Power law"

Smoothing

- Handling sparcity by making sure every probability is non-zero in our models
  - Additive: Add a small amount to all probabilities
  - Discounting: Redistribute probability mass from observed n-grams to unobserved ones
  - Back-off: Use lower order n-grams if higher ones are too sparse
  - Interpolation: Use a combination of different granularities of n-grams

Intuition of Smoothing

- When we have sparse statistics:

  P(w | denied the)

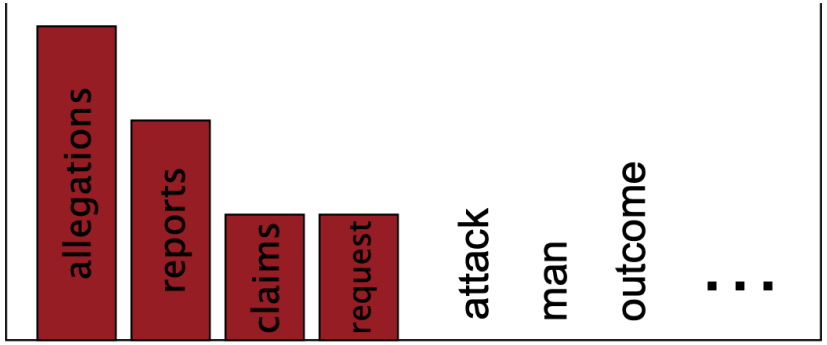
  3 allegations

  2 reports

  1 claims

  1 request

  7 Total



- Steal probability mass to generalize better:

  P(w | denied the)

  2.5 allegations

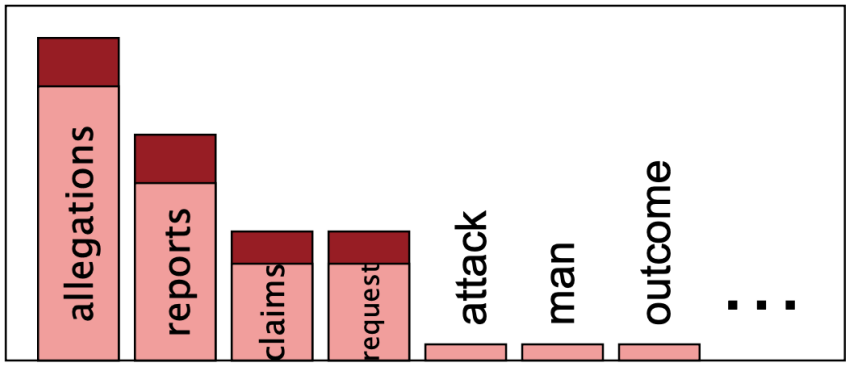
  1.5 reports

  0.5 claims

  0.5 request

  2 others

  7 Total



Laplace Smoothing

- Also known as add-alpha
- Simplest form of smoothing: just add a small alpha to all counts and renormalize!
- Max likelihood for bigrams:

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

- After smoothing:

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i) + \propto}{C(w_{i-1}) + \propto |V|}$$

Raw Bigram Counts (Berkeley restaurant corpus)

- Out of 9222 sentences
- The numbers in the table are $c(w_{i-1}\ w_i)$

w_i

w_{i-1}	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

Credits: Dan Jurafsky)

Smoothed Bigram Counts

- Alpha = 1 in this case:

	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

Credits: Dan Jurafsky)

Smoothed Bigram Probabilities

- Alpha = 1 in this case:

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

	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

Credits: Dan Jurafsky)

Problem with Laplace Smoothing

raw 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

over-smoothing

reconstituted counts

	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

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

QUIZ: 6 PTS

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

- Given the following training corpus:

  S1: you have five apples

  S2: you have no oranges

  S3: no apples have you

- Produce the bigram raw counts table and reconstituted counts table using alpha = 1 (you need to produce two tables):

	you	have	five	apples	no	oranges
you						
have						
five						
apples						
no						
oranges						

Linear Interpolation

$$\hat{P}(w_i|w_{i-1},w_{i-2}) = \lambda_1 P(w_i|w_{i-1},w_{i-2}) + \lambda_2 P(w_i|w_{i-1}) + \lambda_3 P(w_i)$$

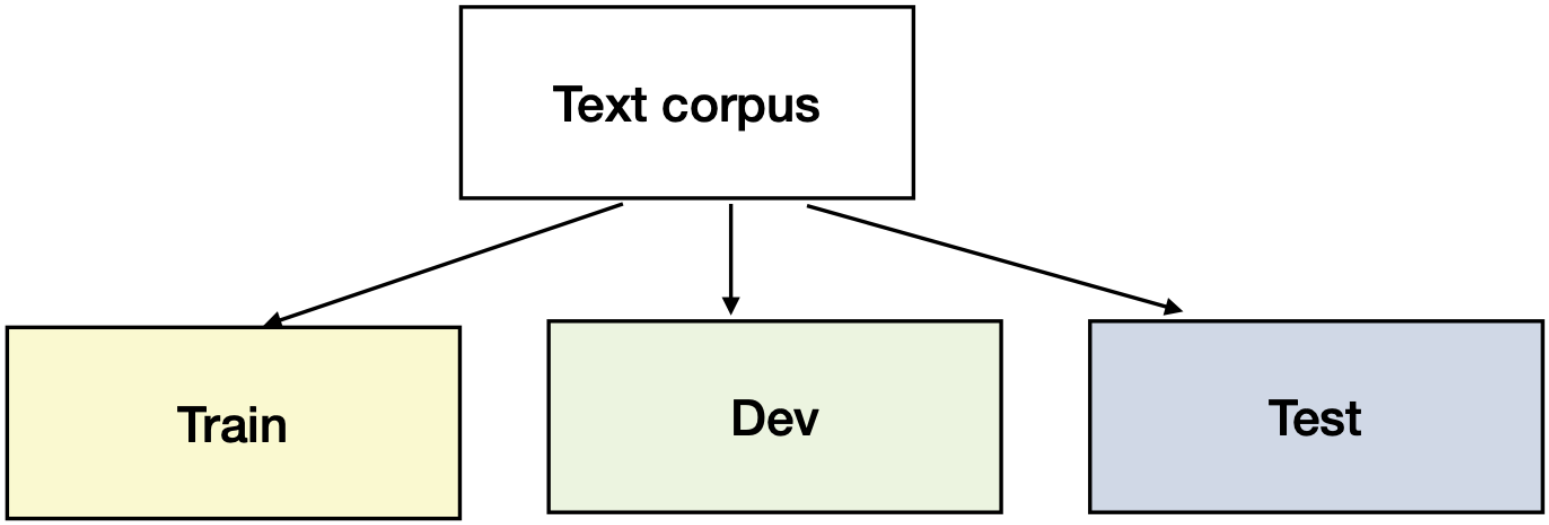
current word

context or history

$$\sum_i \lambda_i = 1$$

- Use a combination of models to estimate probability
- Typically, higher order models has less evidence than lower order, so $\lambda_1 < \lambda_2 < \lambda_3$
- Strong empirical performance

Choosing Lambdas



- First, estimate n-gram prob. on training set
- Then, estimate lambdas (*hyperparameters*) to maximize probability on the held-out dev set
- Usually using EM algorithm

More Generally- Jelinek-Mercer Smoothing (JM Smoothing)

$$P_{\text{interp}}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} P_{\text{ML}}(w_i|w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) P_{\text{interp}}(w_i|w_{i-n+2}^{i-1})$$

Recursive definition!

- Generalizes the simple linear interpolation, with lambdas $\lambda_{w_{i-n+1}^{i-1}}$ conditioned on the context (there are many of them!).
- To reduce the number of lambda params: Partition $\lambda_{w_{i-n+1}^{i-1}}$ according to average number of counts per non-zero element:

$$\text{Partition Metric} = \frac{c(w_{i-n+1}^{i-1})}{|\{w_i : c(w_{i-n+1}^{i-1}, w_i) > 0\}|}$$

- for denser estimates of n-gram probabilities

Intuition for average-count

- Case 1: C (on the mat) = 10, C(on the cat) = 10, C(on the rat) = 10, C(on the bat) = 10, …
- Case 2: C (on the mat) = 40, C(on the cat) = 0, C (on the rat) = 0, C(on the bat) = 0, …
- Which provides a better estimate for P(mat | on the)? – Case 1!
- Larger weights on non-sparse (denser) estimates
- What if C (on the mat) = 37, C(on the cat) = 1, C (on the rat) = 1, C(on the bat) = 1, … ?

Discounting

Bigram count in training	Bigram count in heldout set
0	.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

- Determine some "mass" to remove from probability estimates
- Redistribute mass among unseen n-grams
- Just choose an absolute value d to discount:

$$P_{AbsDiscount}(w_i|w_{i-1}) = \frac{\max(0, C(w_{i-1}w_i) - d)}{C(w_i)} + \lambda_{w_{i-1}} P(w_i)$$

the difference is roughly 0.75, hence $d = 0.75$

Back-off

- Use n-gram if enough evidence, else back off to (n-1)-gram

$$P_{bo}(w_i \mid w_{i-n+1} \cdots w_{i-1})$$

$$= \begin{cases} d_{w_{i-n+1} \cdots w_i} \dfrac{C(w_{i-n+1} \cdots w_{i-1} w_i)}{C(w_{i-n+1} \cdots w_{i-1})} & \text{if } C(w_{i-n+1} \cdots w_i) > k \\ \alpha_{w_{i-n+1} \cdots w_{i-1}} P_{bo}(w_i \mid w_{i-n+2} \cdots w_{i-1}) & \text{otherwise} \end{cases}$$

- d = amount of discounting
- α = back-off weight

INTERPOLATON VS BACKOFF

- To determine the probability of n-grams with *zero* counts:
  - Both use the distributions of lower-order n-grams
- To determining the probability of n-grams with *nonzero* counts:
  - Interpolation uses the distribution of lower-order n-grams
  - Backoff does not.
- More on smoothing: https://arxiv.org/abs/cmp-lg/9606011

Other Language Models

- Discriminative models:
  - train n-gram probabilities to directly maximize performance on an end task (e.g., as feature weights)
- Parsing-based models
  - handle syntactic/grammatical dependencies
- Topic models (word distributions for topics not sequences)