



N-gram Language Models

Phạm Quang Nhật Minh

minhpham0902@gmail.com

December 20, 2025



Agenda

2

- Introduction to N-grams
- Estimating N-gram probabilities
- Evaluating language model
- Generalization and zeros
- Smoothing techniques:
 - Add-one (Laplace) smoothing
 - Interpolation, Backoff



Suggested readings

3

- Chapter 3. Language Modeling with N-Grams (SLP)
 - <https://web.stanford.edu/~jurafsky/slp3/3.pdf>
- Language Models, by Michael Collins.
 - <http://www.cs.columbia.edu/~mcollins/lm-spring2013.pdf>
- NLP Programming Tutorials, by Graham Neubig
 - <http://www.phontron.com/slides/nlp-programming-en-01-unigramlm.pdf>
 - <http://www.phontron.com/slides/nlp-programming-en-02-bigramlm.pdf>



The language modeling problem

4

- Goal: compute the probability of a sentence or sequence of words.

$$P(W) = P(w_1, w_2, \dots, w_n)$$

- E.g., $P(\text{Hôm nay trời đẹp quá}) = P(\text{Hôm, nay, trời, đẹp, quá})$

- Related task: probability of an upcoming word:

$$P(w_4 | w_1, w_2, w_3)$$

- E.g., $P(\text{đẹp} | \text{Hôm, nay, trời})$

- A model that computes either of these: $P(W)$ or $P(w_n | w_1, w_2, \dots, w_{n-1})$ is called a **language model**.



Why language models?

5

■ Machine Translation:

□ $P(\text{high winds tonite}) > P(\text{large winds tonite})$

■ Spell Correction

□ The office is about fifteen **minuets** from my house

■ $P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$

■ Speech Recognition

□ $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$



How to compute $P(W)$

- How to compute this joint probability:
 - $P(\text{its, water, is, so, transparent, that})$
- Intuition: let's rely on the Chain Rule of Probability



The Chain Rule

7

- Conditional probabilities

$$P(B|A) = P(A, B)/P(A)$$

- Rewriting:

$$P(A, B) = P(A)P(B|A)$$



The Chain Rule cont.

8

- More variables:

$$P(A, B, C, D) = ?$$

$$\begin{aligned} P(A, B, C, D) &= P(A, B, C)P(D|A, B, C) \\ &= P(A, B)P(C|A, B)P(D|A, B, C) \\ &= P(A)P(B|A)P(C|A, B)P(D|A, B, C) \end{aligned}$$



The Chain Rule cont.

9

- The Chain Rule in general

$$\begin{aligned} &P(x_1, x_2, x_3, \dots, x_n) \\ &= P(x_1)P(x_2|x_1)P(x_3|x_1, x_2) \dots P(x_n|x_1, \dots, x_{n-1}) \end{aligned}$$



Applying the Chain Rule for joint probability

10

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

$P(\text{"its water is so transparent"}) =$

$P(\text{its}) \times P(\text{water} | \text{its}) \times P(\text{is} | \text{its water})$

$\times P(\text{so} | \text{its water is}) \times P(\text{transparent} | \text{its water is so})$



How to estimate these probabilities

11

- Could we just count and divide?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{\text{Count}(\text{its water is so transparent that the})}{\text{Count}(\text{its water is so transparent that})}$$

- No! Too many possible sentences!
- We'll never see enough data for estimating these



Markov Assumption

12

- Simplifying assumption:

$$P(\text{the} | \text{its water is so transparent that}) \\ \approx P(\text{the} | \text{that})$$

- Or maybe:

$$P(\text{the} | \text{its water is so transparent that}) \\ \approx P(\text{the} | \text{transparent that})$$



Markov Assumption

13

- We approximate each component in the product

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

- So the joint probability of the sequence is

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



Simplest case: Unigram model

14

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$



Bigram model

15

Condition on the previous word:

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-1})$$



Agenda

16

- Introduction to N-grams
- **Estimating N-gram probabilities**
- Evaluating language model
- Generalization and zeros
- Smoothing techniques:
 - Add-one (Laplace) smoothing
 - Interpolation, Backoff



Unigram language model

17

- Do not use history

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i)$$

- Estimate $P(w_i)$ by using Maximum Likelihood Estimate (MLE)

$$P(w_i) = \frac{\text{count}(w_i)}{\sum_{w'} \text{count}(w')}$$



Unigram language model: an example

18

i live in osaka . </s>

$$P(\text{nara}) = 1/20 = 0.05$$

i am a graduate student . </s>

$$P(i) = 2/20 = 0.1$$

my school is in nara . </s>

$$P(</s>) = 3/20 = 0.15$$

$P(W=i \text{ live in nara . } </s>) =$

$$0.1 * 0.05 * 0.1 * 0.05 * 0.15 * 0.15 = 5.625 * 10^{-7}$$



Bigram language model

19

- Condition on the previous word:

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-1})$$

- The Maximum Likelihood Estimate

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



Bigram language model: an example

20

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

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$$P(I | < s >) = \frac{2}{3} = .67$$

$$P(\text{Sam} | < s >) = \frac{1}{3} = .33$$

$$P(\text{am} | I) = \frac{2}{3} = .67$$

$$P(</s> | \text{Sam}) = \frac{1}{2} = .5$$

$$P(\text{Sam} | \text{am}) = \frac{1}{2} = .5$$

$$P(\text{do} | I) = \frac{1}{3} = .33$$



More examples:

Berkeley Restaurant Project sentences

21

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i'm looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i'm looking for a good place to eat breakfast
- when is caffe venezia open during the day



Raw bigram counts

22

- Out of 9222 sentences

	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



Raw bigram probabilities

23

■ Normalize by unigrams:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

■ Results:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0



Bigram estimates of sentence probabilities

24

$P(<s> \text{ I want english food } </s>) =$

$P(\text{I} | <s>)$

$\times P(\text{want} | \text{I})$

$\times P(\text{english} | \text{want})$

$\times P(\text{food} | \text{english})$

$\times P(</s> | \text{food})$

$= .000031$



What kinds of knowledge?

25

- $P(\text{english} | \text{want}) = .0011$
- $P(\text{chinese} | \text{want}) = .0065$
- $P(\text{to} | \text{want}) = .66$
- $P(\text{eat} | \text{to}) = .28$
- $P(\text{food} | \text{to}) = 0$
- $P(\text{want} | \text{spend}) = 0$
- $P(i | \langle s \rangle) = .25$



Practical Issues

26

- We do everything in log space
 - Avoid underflow
 - (also adding is faster than multiplying)

$$\begin{aligned} &P(p_1 \times p_2 \times p_3 \times p_4) \\ &= \log p_1 + \log p_2 + \log p_3 + \log p_4 \end{aligned}$$



Language Modeling Toolkits

27

- SRILM

- <http://www.speech.sri.com/projects/srilm/>

- KenLM

- <https://kheafield.com/code/kenlm/>



Agenda

28

- Introduction to N-grams
- Estimating N-gram probabilities
- Evaluating language model
- Generalization and zeros
- Smoothing techniques:
 - Add-one (Laplace) smoothing
 - Interpolation, Backoff



Evaluation: How good is our model?

29

- Does our language model prefer good sentences to bad ones?
 - Assign higher probability to “real” or “frequently observed” sentences
 - than “ungrammatical” or “rarely observed” sentences



Evaluation: How good is our model?

30

- 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 that is different from our training set, totally unused.
 - ☐ An evaluation metric tells us how well our model does on the test set.



Two evaluation approaches

31

- Extrinsic evaluation
 - Compare two language models in downstream tasks
 - e.g., Spelling correction, speech recognition, MT

- Intrinsic evaluation
 - Use some evaluation measures on the test set
 - We will use **perplexity**



Extrinsic evaluation of N-gram models

32

- Best evaluation for comparing models A and B
 - Put each model in a task
 - spelling corrector, speech recognizer, MT system
 - Run the task, get an accuracy for A and for B
 - How many misspelled words corrected properly
 - How many words translated correctly
 - Compare accuracy for A and B



Difficulty of extrinsic evaluation of N-gram models

33

- Extrinsic evaluation

- Time-consuming; can take days or weeks

- So

- Sometimes use **intrinsic** evaluation: **perplexity**
 - Bad approximation
 - unless the test data looks **just** like the training data
 - So **generally only useful in pilot experiments**
 - But is helpful to think about.



Intuition of Perplexity

34

- The Shannon Game:

- How well can we predict the next word?

I always order pizza with cheese and _____

The 33rd President of the US was _____

I saw a _____

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

....

fried rice 0.0001

....

and 1e-100

- Unigrams are terrible at this game. (Why?)

- A better model of a text

- is one which assigns a higher probability to the word that actually occurs



Perplexity

35

- The best language model is one that best predicts an unseen test set
 - Gives the highest $P(\text{sentence})$
- Perplexity is the inverse probability of the test set, normalized by the number of words

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

Minimizing PPL is the same as maximizing probability

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

Chain rule

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

For bigram

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



Perplexity (2)

36

$$\begin{aligned}\log_2 PP(W) &= \frac{1}{N} \sum_{i=1}^N \log_2 \frac{1}{P(w_i | w_{i-1})} \\ &= \underbrace{-\frac{1}{N} \sum_{i=1}^N \log_2 P(w_i | w_{i-1})}_{\text{Entropy } H}\end{aligned}$$

$$PP(W) = 2^H$$



Lower perplexity = better model

37

- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109



Agenda

38

- Introduction to N-grams
- Estimating N-gram probabilities
- Evaluating language model
- Generalization and zeros
- Smoothing techniques:
 - Add-one (Laplace) smoothing
 - Interpolation



Zero probabilities

39

- Training set:
 - ... denied the allegations
 - ... denied the reports
 - ... denied the claims
 - ... denied the request

- Test set
 - ... denied the offer
 - ... denied the loan

$$P(\text{"offer"} \mid \text{denied the}) = 0$$



Problems with Zero probabilities

40

- We underestimate the probability of all sorts of words that might occur
- The entire probability of the test set is 0.
 - So, we cannot calculate perplexity



Agenda

41

- Introduction to N-grams
- Estimating N-gram probabilities
- Evaluating language model
- Generalization and zeros
- Smoothing techniques:
 - Add-one (Laplace) smoothing
 - Interpolation



Laplace smoothing

42

- Add one to all the counts
- Pretend we saw each word one more time than we did

- MLE unigram probabilities:

$$P_{ML}(w_i) = \frac{c(w_i)}{N}$$

- Add-1 estimate:

$$P_{\text{Laplace}} = \frac{c(w_i) + 1}{\sum_w (c(w) + 1)} = \frac{c(w_i) + 1}{N + V}$$



Laplace smoothing: unigram model

43

i live in osaka . </s>

i am a graduate student . </s>

my school is in nara . </s>

$$P(\text{nara}) = 1/20 = 0.05$$

$$P(i) = 2/20 = 0.1$$

$$P(</s>) = 3/20 = 0.15$$

$$P(\text{kyoto}) = 0/20 = 0$$

Vocab = {i, live, in, osaka, am, graduate, student, my, school, is, nara, </s>}

$$V = 12$$



Laplace smoothing: unigram model

44

i live in osaka . </s>

i am a graduate student . </s>

my school is in nara . </s>

Vocab = {i, live, in, osaka, am, graduate, student, my, school, is, nara, </s>}

$V = 12$

$$P(\text{nara}) = (1+1)/(20+12) = 0.0625$$

$$P(i) = (2+1)/(20+12) = 0.09375$$

$$P(</s>) = (3+1)/(20+12) = 0.125$$

$$P(\text{kyoto}) = (0+1)/(20+12) = 0.03125$$



Laplace smoothing: bigrams

45

- MLE estimate:

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

- Add-1 estimate:

$$P_{\text{Laplace}} = \frac{c(w_{i-1}, w_i) + 1}{\sum_w (c(w_{i-1}w) + 1)} = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$



Berkeley Restaurant Corpus: Laplace smoothed bigram counts

46

	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

47

$$P^*(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

	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



Agenda

48

- Introduction to N-grams
- Estimating N-gram probabilities
- Evaluating language model
- Generalization and zeros
- Smoothing techniques:
 - Add-one (Laplace) smoothing
 - **Interpolation**



Linear interpolation

49

- Mix trigrams, bigrams and unigrams

$$\begin{aligned} P(w_i | w_{i-2}, w_{i-1}) \\ = \lambda_1 \times P_{ML}(w_i | w_{i-2}, w_{i-1}) + \lambda_2 \times P_{ML}(w_i | w_{i-1}) \\ + \lambda_3 \times P_{ML}(w_i) \end{aligned}$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and $\lambda_i \geq 0$ for all i

- Sometimes, all trigrams, bigrams, unigrams do not exist

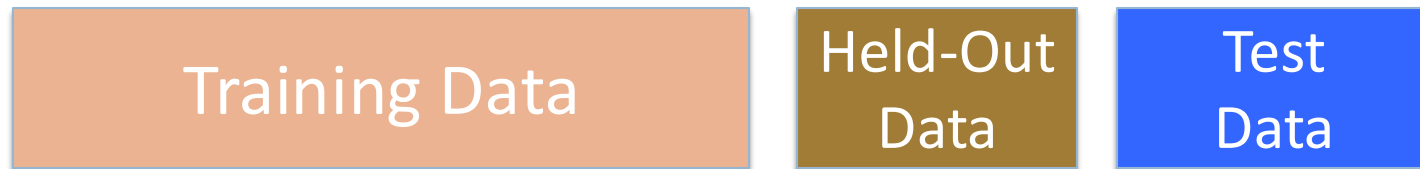
□ Recall: $P(w_i) = \lambda \times P_{ML}(w_i) + (1 - \lambda) \times \frac{1}{N}$



How to set lambdas?

50

- Use a **held-out** corpus



- Choose λ s to maximize the probability of held-out data:
 - ☐ Fix the N-gram probabilities (on the training data)
 - ☐ Then search for λ s that give largest probability to held-out set:

$$\log P(w_1 \dots w_n \mid M(\lambda_1 \dots \lambda_k)) = \sum_i \log P_{M(\lambda_1 \dots \lambda_k)}(w_i \mid w_{i-1})$$



Example: bigrams

51

i live in osaka . </s>

i am a graduate student . </s>

my school is in nara . </s>

■ Maximum-likelihood estimation:

☐ $P(\text{osaka} \mid \text{in}) = c(\text{in osaka})/c(\text{in}) = 1/2 = 0.5$

☐ $P(\text{nara} \mid \text{in}) = c(\text{in nara})/c(\text{in}) = 1/2 = 0.5$

☐ $P(\text{school} \mid \text{in}) = c(\text{in school})/c(\text{in}) = 0/2 = 0$



Example: interpolation

52

i live in osaka . </s>

i am a graduate student . </s>

my school is in nara . </s>

■ Using interpolation

$$\square P(\text{school} \mid \text{in}) = \lambda_2 P_{ML}(\text{school} \mid \text{in}) + (1 - \lambda_2) P(\text{school})$$

$$\begin{aligned} \square P(\text{school}) &= \lambda_1 P_{ML}(\text{school}) + (1 - \lambda_1) \frac{1}{N} \\ &= \lambda_1 \times \frac{1}{20} + (1 - \lambda_1) \times \frac{1}{N} \end{aligned}$$