

CS 489: Introduction to Natural Language Processing

Lecture 5: Statistical Language Models

Instructor: Freda Shi

September 24, 2026



UNIVERSITY OF
WATERLOO

Outline

1. Recap: language models and the chain rule
2. N -gram language models: the Markov assumption and MLE estimation
3. Laplace smoothing (add- α)
4. Generating text and evaluating with perplexity and bits-per-character

Recap: Where We Are

So far:

- **Lecture 1:** What is NLP? Language models. Probability and information theory.
- **Lecture 2:** Tokenization: BPE, WordPiece, Unigram LM.
- **Lecture 3:** Word embeddings: distributional semantics, Word2Vec.
- **Lecture 4:** Supervised learning: classification, logistic regression, neural nets.

Today: We return to the central object of this course, the **language model**, and build our first one from scratch.

What is a Language Model?

A **language model** (LM) assigns a **probability distribution** over sequences of tokens:

$$P(\mathbf{x}) = P(x_1, x_2, \dots, x_n)$$

A *good* language model assigns **higher probability** to sentences that are more likely to appear in natural language.

What is a Language Model?

A **language model** (LM) assigns a **probability distribution** over sequences of tokens:

$$P(\mathbf{x}) = P(x_1, x_2, \dots, x_n)$$

A *good* language model assigns **higher probability** to sentences that are more likely to appear in natural language.

Why do we care?

- **Machine translation:** $P(\text{turn the camera off}) > P(\text{put the camera out})$
- **Speech recognition:**
 $P(\text{I need to recognize speech}) > P(\text{I need to wreck a nice speech})$
- **Spelling/grammar correction:**
 $P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$
- **Text generation:** GPT, ChatGPT, ...: generate text by sampling from a language model!

Language Modeling via the Chain Rule

How do we compute $P(x_1, x_2, \dots, x_n)$?

Chain rule of probability (no assumptions yet!):

$$\begin{aligned} P(x_1, x_2, \dots, x_n) &= P(x_1) \cdot P(x_2 \mid x_1) \cdot P(x_3 \mid x_1, x_2) \cdots P(x_n \mid x_1, \dots, x_{n-1}) \\ &= \prod_{i=1}^n P(x_i \mid x_1, \dots, x_{i-1}) \end{aligned}$$

This is exact, but we need to estimate **each** conditional probability $P(x_i \mid x_1, \dots, x_{i-1})$.

Language Modeling via the Chain Rule

How do we compute $P(x_1, x_2, \dots, x_n)$?

Chain rule of probability (no assumptions yet!):

$$\begin{aligned} P(x_1, x_2, \dots, x_n) &= P(x_1) \cdot P(x_2 \mid x_1) \cdot P(x_3 \mid x_1, x_2) \cdots P(x_n \mid x_1, \dots, x_{n-1}) \\ &= \prod_{i=1}^n P(x_i \mid x_1, \dots, x_{i-1}) \end{aligned}$$

This is exact, but we need to estimate **each** conditional probability $P(x_i \mid x_1, \dots, x_{i-1})$.

Problem: How many distinct contexts $x_1, \dots, x_{i-1}$ are there?

- Vocabulary size $|V| = 40,000$. Sentences of length ≤ 11 .
- $|V|^{11} \approx 4 \times 10^{50}$ possible sequences, more than atoms in the Earth!

LMs as Next-Word Predictors

Three closely related views of language modeling:

1. **Joint probability:** $P(x_1, x_2, \dots, x_n)$
2. **Next-word prediction (autoregressive):**

$$P(x_i \mid x_1, \dots, x_{i-1})$$

3. **Masked word prediction:**

$$P(x_i \mid x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$$

A model that computes any of the above is called a **language model**.

Today we focus on **autoregressive** (left-to-right) models.

Modeling Length: Start and Stop Symbols

A language model must assign probability to sequences of **any length**, and probabilities must sum to 1 across all possible sequences.

Modeling Length: Start and Stop Symbols

A language model must assign probability to sequences of **any length**, and probabilities must sum to 1 across all possible sequences.

Solution: Add special symbols:

- $\langle s \rangle$: start-of-sequence (provides context for the first word).
- $\langle /s \rangle$: end-of-sequence (models **stopping probability**).

$$P(\text{I love NLP}) = P(\text{I} \mid \langle s \rangle) \cdot P(\text{love} \mid \langle s \rangle \text{I}) \cdot P(\text{NLP} \mid \langle s \rangle \text{I love}) \cdot P(\langle /s \rangle \mid \dots)$$

Without $\langle /s \rangle$: One Unit per Length

Without a stop symbol, the next token must be in V . At every prefix h , $\sum_{w \in V} P(w | h) = 1$. For length two, sum the chain-rule probabilities from right to left:

$$\begin{aligned} \sum_{x_1, x_2 \in V} P(x_1 | \langle s \rangle) P(x_2 | \langle s \rangle, x_1) &= \sum_{x_1 \in V} P(x_1 | \langle s \rangle) \underbrace{\sum_{x_2 \in V} P(x_2 | \langle s \rangle, x_1)}_{=1} \\ &= \sum_{x_1 \in V} P(x_1 | \langle s \rangle) = 1. \end{aligned}$$

Without $\langle /s \rangle$: One Unit per Length

Without a stop symbol, the next token must be in V . At every prefix h , $\sum_{w \in V} P(w | h) = 1$. For length two, sum the chain-rule probabilities from right to left:

$$\begin{aligned} \sum_{x_1, x_2 \in V} P(x_1 | \langle s \rangle) P(x_2 | \langle s \rangle, x_1) &= \sum_{x_1 \in V} P(x_1 | \langle s \rangle) \underbrace{\sum_{x_2 \in V} P(x_2 | \langle s \rangle, x_1)}_{=1} \\ &= \sum_{x_1 \in V} P(x_1 | \langle s \rangle) = 1. \end{aligned}$$

The same reasoning applies to any fixed length n , with $x_{<i} = (\langle s \rangle, x_1, \dots, x_{i-1})$:

$$\sum_{\mathbf{x} \in V^n} \prod_{i=1}^n P(x_i | x_{<i}) = 1.$$

Summing over *all* lengths counts one full unit per length:

$$\sum_{n=1}^{\infty} \sum_{\mathbf{x} \in V^n} \prod_{i=1}^n P(x_i | x_{<i}) = \sum_{n=1}^{\infty} 1 = \infty.$$

With $\langle /s \rangle$: Split and Stop

Now the next step can stop or continue, and those probabilities sum to 1:

$$P(\langle /s \rangle \mid h) + \sum_{w \in V} P(w \mid h) = 1.$$

Apply this identity to the “continue” part, starting at $\langle s \rangle$:

$$\begin{aligned} 1 &= P(\langle /s \rangle \mid \langle s \rangle) + \sum_{x_1 \in V} P(x_1 \mid \langle s \rangle) \\ &= P(\langle /s \rangle \mid \langle s \rangle) + \sum_{x_1 \in V} P(x_1 \mid \langle s \rangle) P(\langle /s \rangle \mid \langle s \rangle, x_1) \\ &\quad + \sum_{x_1, x_2 \in V} P(x_1 \mid \langle s \rangle) P(x_2 \mid \langle s \rangle, x_1). \end{aligned}$$

With $\langle /s \rangle$: Split and Stop

Now the next step can stop or continue, and those probabilities sum to 1:

$$P(\langle /s \rangle \mid h) + \sum_{w \in V} P(w \mid h) = 1.$$

Apply this identity to the “continue” part, starting at $\langle s \rangle$:

$$\begin{aligned} 1 &= P(\langle /s \rangle \mid \langle s \rangle) + \sum_{x_1 \in V} P(x_1 \mid \langle s \rangle) \\ &= P(\langle /s \rangle \mid \langle s \rangle) + \sum_{x_1 \in V} P(x_1 \mid \langle s \rangle) P(\langle /s \rangle \mid \langle s \rangle, x_1) \\ &\quad + \sum_{x_1, x_2 \in V} P(x_1 \mid \langle s \rangle) P(x_2 \mid \langle s \rangle, x_1). \end{aligned}$$

The terms stop at lengths 0 and 1 or continue; if the remaining “continue” mass tends to zero, repeating the split gives:

$$\sum_{n=0}^{\infty} \sum_{\mathbf{x} \in V^n} \left(\prod_{i=1}^n P(x_i \mid \mathbf{x}_{<i}) \right) P(\langle /s \rangle \mid \mathbf{x}_{<n+1}) = 1.$$

The Markov Assumption

Key idea: Approximate the full context by only the most recent k words.

$$P(x_i \mid x_1, \dots, x_{i-1}) \approx P(x_i \mid x_{i-k}, \dots, x_{i-1})$$

This is the **k -th order Markov assumption** (Andrey Markov, 1906).

N -gram model = $(N-1)$ -th order Markov model:

Model	Order	Approximation
Unigram ($N=1$)	0	$P(x_i \mid x_1, \dots, x_{i-1}) \approx P(x_i)$
Bigram ($N=2$)	1	$P(x_i \mid x_1, \dots, x_{i-1}) \approx P(x_i \mid x_{i-1})$
Trigram ($N=3$)	2	$P(x_i \mid x_1, \dots, x_{i-1}) \approx P(x_i \mid x_{i-2}, x_{i-1})$

Trade-off: Larger $N \rightarrow$ richer context, but many more parameters to estimate.

Estimating N -Gram Probabilities: MLE

Maximum likelihood estimation: Count and normalize.

Bigram:

$$P_{\text{MLE}}(x_i \mid x_{i-1}) = \frac{\text{count}(x_{i-1}, x_i)}{\text{count}(x_{i-1})}$$

Estimating N -Gram Probabilities: MLE

Maximum likelihood estimation: Count and normalize.

Bigram:

$$P_{\text{MLE}}(x_i \mid x_{i-1}) = \frac{\text{count}(x_{i-1}, x_i)}{\text{count}(x_{i-1})}$$

General N -gram:

$$P_{\text{MLE}}(x_i \mid x_{i-N+1}, \dots, x_{i-1}) = \frac{\text{count}(x_{i-N+1}, \dots, x_i)}{\text{count}(x_{i-N+1}, \dots, x_{i-1})}$$

In words: the fraction of times the N -gram appears out of all continuations of the $(N-1)$ -gram prefix.

Bigram MLE: Worked Example

Training corpus:

- `<s> I am Sam </s>`
- `<s> Sam I am </s>`
- `<s> I do not like green eggs and ham </s>`

Bigram MLE: Worked Example

Training corpus:

- $\langle s \rangle$ I am Sam $\langle /s \rangle$
- $\langle s \rangle$ Sam I am $\langle /s \rangle$
- $\langle s \rangle$ I do not like green eggs and ham $\langle /s \rangle$

Some bigram probabilities:

Bigram	Count ratio	MLE
$P(I \mid \langle s \rangle)$	$\frac{\text{count}(\langle s \rangle, I)}{\text{count}(\langle s \rangle)} = \frac{2}{3}$	0.67
$P(\text{Sam} \mid \langle s \rangle)$	$\frac{\text{count}(\langle s \rangle, \text{Sam})}{\text{count}(\langle s \rangle)} = \frac{1}{3}$	0.33
$P(\text{am} \mid I)$	$\frac{\text{count}(I, \text{am})}{\text{count}(I)} = \frac{2}{3}$	0.67
$P(\langle /s \rangle \mid \text{Sam})$	$\frac{\text{count}(\text{Sam}, \langle /s \rangle)}{\text{count}(\text{Sam})} = \frac{1}{2}$	0.50

The Zero-Probability Problem

Training data:

- `<s> I am Sam </s>` `<s> Sam I am </s>` `<s> I do not like green eggs and ham </s>`

Test sentence:

- `<s> I like green eggs and ham </s>`

The Zero-Probability Problem

Training data:

- `<s> I am Sam </s>` `<s> Sam I am </s>` `<s> I do not like green eggs and ham </s>`

Test sentence:

- `<s> I like green eggs and ham </s>`

Problem: $\text{count}(I, \text{like}) = 0$, so:

$$P_{\text{MLE}}(\text{like} \mid I) = \frac{0}{3} = 0$$

The **entire sentence probability** becomes 0 because of one missing bigram!

Two issues:

- Unseen N -grams get zero probability $\rightarrow$ legitimate sentences score 0.
- Seen N -grams get **too much** probability $\rightarrow$ overfitting to training data.

$\Rightarrow$ We need **smoothing**.

Laplace (Add-1) Smoothing

Idea: Pretend we saw every N -gram one extra time.

MLE:

$$P_{\text{MLE}}(x_i \mid x_{i-1}) = \frac{\text{count}(x_{i-1}, x_i)}{\text{count}(x_{i-1})}$$

Laplace (Add-1) Smoothing

Idea: Pretend we saw every N -gram one extra time.

MLE:

$$P_{\text{MLE}}(x_i \mid x_{i-1}) = \frac{\text{count}(x_{i-1}, x_i)}{\text{count}(x_{i-1})}$$

Add-1 smoothing:

$$P_{\text{Lap}}(x_i \mid x_{i-1}) = \frac{\text{count}(x_{i-1}, x_i) + 1}{\text{count}(x_{i-1}) + |V|}$$

where $|V|$ is the vocabulary size (to keep probabilities summing to 1).

More generally, add- α smoothing ($\alpha > 0$, typically $\alpha < 1$):

$$P_{\text{add-}\alpha}(x_i \mid x_{i-1}) = \frac{\text{count}(x_{i-1}, x_i) + \alpha}{\text{count}(x_{i-1}) + \alpha|V|}$$

Laplace Smoothing: Pros and Cons

Pros:

- Simple to implement.
- Guarantees no zero probabilities.
- We already used this for Naïve Bayes in Lecture 4!

Laplace Smoothing: Pros and Cons

Pros:

- Simple to implement.
- Guarantees no zero probabilities.
- We already used this for Naïve Bayes in Lecture 4!

Cons:

- Gives **far too much** probability mass to unseen events.
- Distorts the distribution: seen events lose too much probability.
- For language modeling, Laplace smoothing often performs **poorly** on held-out text.

Example: Suppose the occurs 1,000 times, all followed by cat, and $|V| = 50,000$:

- Before: $P(\text{cat} \mid \text{the}) = 1$.
- After add-1: $P_{\text{Lap}}(\text{cat} \mid \text{the}) = \frac{1,001}{1,000+50,000} \approx 0.0196$, an approximately 98% reduction.

Generating Text from N -Gram Models

Sampling from a bigram model:

1. Start with $\langle s \rangle$.
2. Sample $x_1 \sim P(\cdot \mid \langle s \rangle)$.
3. Sample $x_2 \sim P(\cdot \mid x_1)$.
4. Continue until $\langle /s \rangle$ is sampled.

Generating Text from N -Gram Models

Sampling from a bigram model:

1. Start with $\langle s \rangle$.
2. Sample $x_1 \sim P(\cdot \mid \langle s \rangle)$.
3. Sample $x_2 \sim P(\cdot \mid x_1)$.
4. Continue until $\langle /s \rangle$ is sampled.

Alternatively: greedy decoding

$$x_i = \arg \max_{w \in V} P(w \mid x_{i-1})$$

Always pick the most probable next word. (Tends to produce repetitive text.)

Higher-order models produce more coherent text but are more constrained.

Example: Generated Sentences

Shakespeare corpus ($N = 884,647$ tokens, $|V| = 29,066$; SLP 3, Fig. 3.4):

Unigram:

Hill he late speaks; or! a more to leg less first you enter

Bigram:

What means, sir. I confess she? then all sorts, he is trim, captain.

Trigram:

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

4-gram:

It cannot be but so.

As N increases: text becomes more fluent but also more **memorized** from training data.

Evaluating Language Models

How do we measure how good a language model is?

Extrinsic evaluation: Run the LM inside a downstream task (MT, ASR, ...) and measure task performance. Gold standard, but expensive.

Evaluating Language Models

How do we measure how good a language model is?

Extrinsic evaluation: Run the LM inside a downstream task (MT, ASR, ...) and measure task performance. Gold standard, but expensive.

Intrinsic evaluation: Measure how well the LM predicts **held-out test data**.

- Compute the probability the model assigns to the test set.
- A better model assigns **higher probability** to real text.
- Standard metric: **perplexity**.

Evaluating Language Models

How do we measure how good a language model is?

Extrinsic evaluation: Run the LM inside a downstream task (MT, ASR, ...) and measure task performance. Gold standard, but expensive.

Intrinsic evaluation: Measure how well the LM predicts **held-out test data**.

- Compute the probability the model assigns to the test set.
- A better model assigns **higher probability** to real text.
- Standard metric: **perplexity**.

Caveat: Good perplexity does not guarantee good downstream performance, but it is a useful proxy.

Test-Set Probability Across Sentences

Let $W = (S_1, \dots, S_K)$ contain K test sentences, with $S_s = (w_{s,1}, \dots, w_{s,L_s})$ ending in $\langle /s \rangle$. Let $M = \sum_{s=1}^K L_s$ count scored tokens (including $\langle /s \rangle$ but excluding $\langle s \rangle$). The history $w_{s,<i}$ includes only the prefix of sentence s , starting with its $\langle s \rangle$.

Step 1: Multiply sentence probabilities; take base-2 logs.

$$P(W) = \prod_{s=1}^K P(S_s) = \prod_{s=1}^K \prod_{i=1}^{L_s} P(w_{s,i} \mid w_{s,<i}),$$
$$\log_2 P(W) = \sum_{s=1}^K \sum_{i=1}^{L_s} \log_2 P(w_{s,i} \mid w_{s,<i}).$$

From Log-Probability to Perplexity

Step 2: Average negative log-probability (cross-entropy).

$$\ell = -\frac{1}{M} \sum_{s=1}^K \sum_{i=1}^{L_s} \log_2 P(w_{s,i} \mid w_{s,<i})$$

From Log-Probability to Perplexity

Step 2: Average negative log-probability (cross-entropy).

$$\ell = -\frac{1}{M} \sum_{s=1}^K \sum_{i=1}^{L_s} \log_2 P(w_{s,i} \mid w_{s,<i})$$

Step 3: Perplexity (use base-2 logs):

$$\boxed{\text{PPL}(W) = 2^\ell = P(W)^{-1/M}}$$

Perplexity: Intuition

$$\text{PPL}(W) = 2^\ell = P(W)^{-1/M}$$

Interpretation: PPL is the **effective number of equally likely next-token choices**.

- If the model predicts uniformly over $|V|$ words: $\text{PPL} = |V|$.
- If the model is perfect (always assigns probability 1 to the correct word): $\text{PPL} = 1$.
- **Lower is better.**

Perplexity: Intuition

$$\text{PPL}(W) = 2^{\ell} = P(W)^{-1/M}$$

Interpretation: PPL is the **effective number of equally likely next-token choices**.

- If the model predicts uniformly over $|V|$ words: $\text{PPL} = |V|$.
- If the model is perfect (always assigns probability 1 to the correct word): $\text{PPL} = 1$.
- **Lower is better.**

Connection to information theory (Lecture 1):

- ℓ is the **average cross-entropy per scored token** across all test sentences.
- $\text{PPL} = 2^{\ell}$.
- Minimizing perplexity $\Leftrightarrow$ minimizing cross-entropy $\Leftrightarrow$ maximizing likelihood.

Bits-per-Character (BPC)

BPC measures average log-loss per character, regardless of the LM's tokenization. Let a fixed, lossless tokenizer map text with C characters to $t_1, \dots, t_T$ (including $\langle /s \rangle$ if scored).

$$\begin{aligned}\text{BPC} &= -\frac{1}{C} \log_2 P(t_1, \dots, t_T) = -\frac{1}{C} \sum_{j=1}^T \log_2 P(t_j | t_{<j}) \\ &= \frac{T}{C} \log_2 \text{PPL}_{\text{token}}.\end{aligned}$$

- Works with character, subword, or word tokens; divide by C characters.
- Lossy tokenization does not give a comparable BPC for the original text.

Character-level examples:

- Uniform over 27 characters: $\text{BPC} = \log_2 27 \approx 4.75$.
- Good neural LM on English: $\text{BPC} \approx 1.0\text{--}1.3$.

Perplexity: Worked Example

Bigram model. Test sentence: $\langle s \rangle$ I like NLP $\langle /s \rangle$

Suppose our model gives:

Conditional	Probability
$P(I \mid \langle s \rangle)$	0.40
$P(\text{like} \mid I)$	0.10
$P(\text{NLP} \mid \text{like})$	0.05
$P(\langle /s \rangle \mid \text{NLP})$	0.30

$M = 4$ tokens (I, like, NLP, $\langle /s \rangle$).

$$\begin{aligned}\log_2 P(W) &= \log_2(0.40) + \log_2(0.10) + \log_2(0.05) + \log_2(0.30) \\ &= -1.32 + (-3.32) + (-4.32) + (-1.74) = -10.70\end{aligned}$$

Perplexity: Worked Example

Bigram model. Test sentence: $\langle s \rangle$ I like NLP $\langle /s \rangle$

Suppose our model gives:

Conditional	Probability
$P(I \mid \langle s \rangle)$	0.40
$P(\text{like} \mid I)$	0.10
$P(\text{NLP} \mid \text{like})$	0.05
$P(\langle /s \rangle \mid \text{NLP})$	0.30

$M = 4$ tokens (I, like, NLP, $\langle /s \rangle$).

$$\begin{aligned}\log_2 P(W) &= \log_2(0.40) + \log_2(0.10) + \log_2(0.05) + \log_2(0.30) \\ &= -1.32 + (-3.32) + (-4.32) + (-1.74) = -10.70\end{aligned}$$

$$\text{PPL} = 2^{-(-10.70)/4} = 2^{2.675} \approx \mathbf{6.39}$$

On average, the model is as “confused” as if choosing among ≈ 6 equally likely words.

Summary

- A **language model** assigns probabilities to sequences of tokens.
- The **chain rule** decomposes $P(\mathbf{x})$ into a product of conditional probabilities.
- **N -gram models** apply the Markov assumption: condition on the last $N-1$ words.
- **MLE** estimates N -gram probabilities by counting, but suffers from **zero counts**.
- **Laplace smoothing** adds pseudo-counts to avoid zero probabilities, but can distort the distribution.
- **Perplexity** $= 2^{\text{cross-entropy}}$: the standard intrinsic metric for LMs.
- **Bits-per-character**: normalize text log-probability by character count.

Readings

Required:

- Jurafsky & Martin, *SLP 3*, [Chapter 3: N-gram Language Models](#).

Next Lecture

Neural Network Foundations

- MLPs, backpropagation, activations.
- Recurrent Neural Networks (RNNs).
- LSTMs/GRUs and the vanishing gradient problem.