

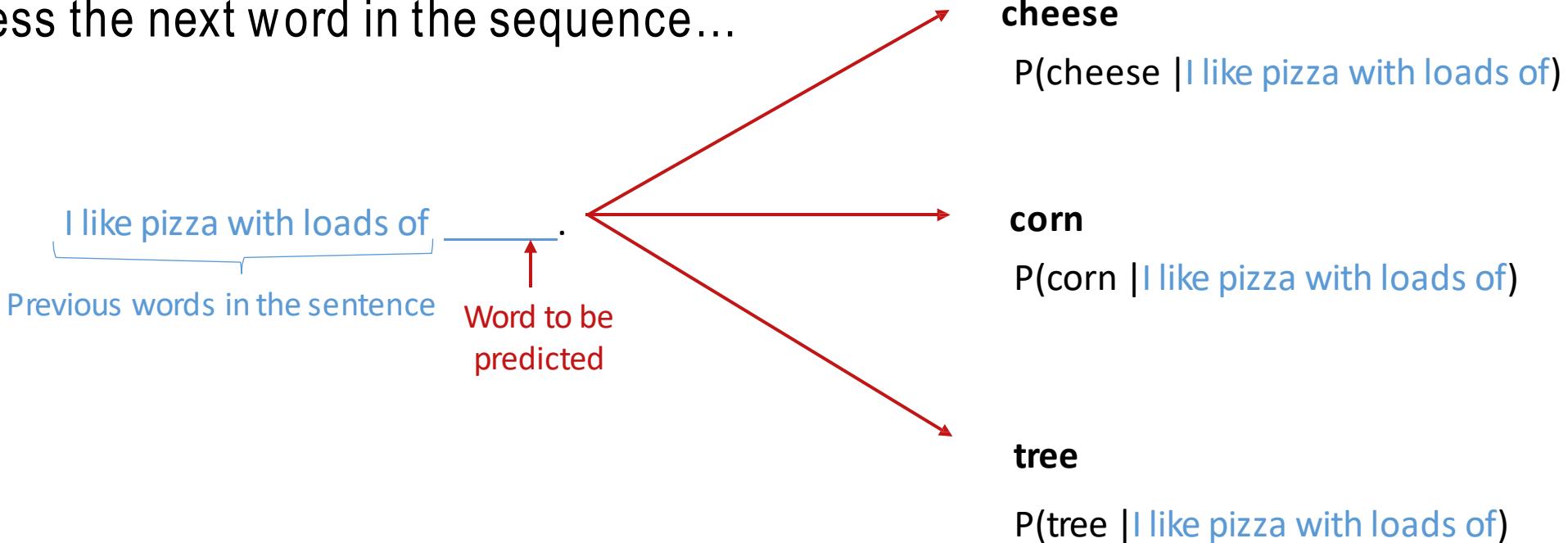
Introduction to Statistical Language Models

By Dr. Sriparna Saha

Ack these slides are taken from IIT Delhi LLM Course

Next Word Prediction

Guess the next word in the sequence...



$P(\text{cheese} | \text{I like pizza with loads of }) > P(\text{com} | \text{I like pizza with loads of }) >> P(\text{tree} | \text{I like pizza with loads of })$

Probabilistic Language Models: Applications

Probabilistic language models can be used to determine the **most plausible sentence** by assigning a probability to sentences.

Probabilistic Language Models: Applications

Probabilistic language models can be used to determine the **most plausible sentence** by assigning a probability to sentences.

•Speech Recognition

- $P(\text{I bought fresh mangoes from the market}) \gg P(\text{I bot fresh man goes from the mar kit})$
- $P(\text{I love eating spicy samosas}) \gg P(\text{eye love eat tin spy sea some o says})$

Probabilistic Language Models: Applications

Probabilistic language models can be used to determine the **most plausible sentence** by assigning a probability to sentences.

•Speech Recognition

- $P(\text{I bought fresh mangoes from the market}) \gg P(\text{I bot fresh man goes from the mar kit})$
- $P(\text{I love eating spicy samosas}) \gg P(\text{eye love eat tin spy sea some o says})$

•Machine Translation

- $P(\text{Heavy rainfall}) \gg P(\text{Big rainfall})$
- $P(\text{The festival of lights}) \gg P(\text{the festival of lamps})$
- $P(\text{Family gatherings}) > P(\text{Family meetings})$

Probabilistic Language Models: Applications

Probabilistic language models can be used to determine the **most plausible sentence** by assigning a probability to sentences.

•Speech Recognition

- $P(\text{I bought fresh mangoes from the market}) \gg P(\text{I bot fresh man goes from the mar kit})$
- $P(\text{I love eating spicy samosas}) \gg P(\text{eye love eat tin spy sea some o says})$

•Machine Translation

- $P(\text{Heavy rainfall}) \gg P(\text{Big rainfall})$
- $P(\text{The festival of lights}) \gg P(\text{the festival of lamps})$
- $P(\text{Family gatherings}) > P(\text{Family meetings})$

•Context Sensitive Spelling Correction

•Natural Language Generation

- ...

Language Models Are Everywhere

The screenshot shows a language translation interface. At the top, there are language selection dropdowns: 'Detect language' (grayed out), 'English', 'Spanish', and 'Hindi' (selected). Below this is a row of icons: a microphone, a speaker, a refresh arrow, and a dropdown menu.

The main area contains two text boxes. The left text box contains the English sentence: "The train to Mumbai is delayed". The right text box displays the Hindi translation: "मुंबई जाने वाली ट्रेन देरी से चल रही है" (mumbee jaane vaalee tren deree se chal rahee hai). Below the Hindi text is its phonetic transcription: "mumbee jaane vaalee tren deree se chal rahee hai".

At the bottom of the interface, there are additional icons: a speaker, a square, a downward arrow, and a share icon.

Language Models Are Everywhere

Detect language English Spanish ↗ Hindi Bengali English ↘

The train to Mumbai is delayed ×

मुंबई जाने वाली ट्रेन देरी से चल ☆
रही है

mumbee jaane vaalee tren deree se chal
rahee hai

🔊 🔊 30 / 5,000 ⌂

Large Language Models Saved

Large Language Models (LLMs) hav revolutionized the field of natural language processing. LLMs, such as GPT-3, have demonstrated impressive capabilities in understanding and generate human-like text across various natural language applications.

G

Review suggestions 2

Correctness	Clarity	Engagement	Delivery	Style guide
-------------	---------	------------	----------	-------------

Correct your spelling hav

Wrong verb form generate

Language Models Are Everywhere

The image displays three separate user interfaces demonstrating the integration of Large Language Models (LLMs) into everyday tools:

- Left Panel (Screenshot 1):** A language detection and translation interface. It shows "English" as the source language and "Spanish" as the target language. Below this, a sentence "The train to Mumbai is delayed" is translated into Spanish ("El tren para Mumbai está retrasado") and then into Hindi ("मुंबई जाने वाली ट्रेन देरी से चल रही है"). The interface includes audio playback icons and a progress bar showing 30 / 5,000.
- Middle Panel (Screenshot 2):** A review suggestion interface from Grammarly. It shows a green circular icon with a 'G'. Below it, the heading "Review suggestions" with a count of 2. Underneath, there are two items:
 - A red circle icon next to the text "Correct your spelling hav".
 - A red circle icon next to the text "Wrong verb form generate".
- Right Panel (Screenshot 3):** A ChatGPT interface. At the top, it shows "ChatGPT" with a dropdown arrow and a small edit icon. Below the input field, there is a decorative logo consisting of a black knot-like shape. To the right of the input field are four cards representing different AI-generated tasks:
 - "Python script for daily email reports"
 - "Design a fun coding game"
 - "Content calendar for TikTok"
 - "Explain nostalgia to a kindergartener"

Probabilistic Language Models

- **Goal:** Calculate the probability of a sentence or sequence consisting of n words

$$P(W) = P(w_1, w_2, w_3, \dots, w_n)$$

or

- **Related Task:** Calculate the probability of the next word conditioned on the preceding words

$$P(w_6 | w_1, w_2, w_3, w_4, w_5)$$

Probabilistic Language Models

- **Goal:** Calculate the probability of a sentence or sequence consisting of n words

$$P(W) = P(w_1, w_2, w_3, \dots, w_n)$$

or

- **Related Task:** Calculate the probability of the next word conditioned on the preceding words

$$P(w_6 | w_1, w_2, w_3, w_4, w_5)$$

A model that calculates either of these is referred to as a
Language Model (LM).

Probability of a Sentence

Let's consider the following sentence:

The monsoon season has begun

- How to compute the probability of the sentence?

$$\begin{aligned} P(W) &= P(\text{"The monsoon season has begun"}) \\ &= P(\text{The, monsoon, season, has, begun}) \end{aligned}$$

Probability of a Sentence

Let's consider the following sentence:

The monsoon season has begun

- How to compute the probability of the sentence?

$$\begin{aligned} P(W) &= P(\text{"The monsoon season has begun"}) \\ &= P(\text{The, monsoon, season, has, begun}) \end{aligned}$$

We compute the above joint probability by using the principles of
Chain Rule of Probability.

Chain Rule of Probability

- Definition of **conditional probability**:

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

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

Chain Rule of Probability

- Definition of **conditional probability**:

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

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

- More variables: $P(A, B, C, D) = P(A) \cdot P(B | A) \cdot P(C | A, B) \cdot P(D | A, B, C)$

Chain Rule of Probability

- Definition of **conditional probability**:

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

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

- More variables: $P(A, B, C, D) = P(A) \cdot P(B | A) \cdot P(C | A, B) \cdot P(D | A, B, C)$

- The **Chain Rule** in general:

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

Probability of a Sequence

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

$P(W)$ = $P(\text{"The monsoon season has begun"})$
= $P(\text{The, monsoon, season, has, begun})$
= $P(\text{The}) \times P(\text{monsoon} | \text{The}) \times P(\text{season} | \text{The monsoon}) \times P(\text{has} | \text{The monsoon season}) \times P(\text{begun} | \text{The monsoon season has})$

Estimate Conditional Probabilities

$$P(\text{begun} \mid \text{The monsoon season has}) = \frac{\text{Count}(\text{The monsoon season has begun})}{\text{Count}(\text{The monsoon season has})}$$

Estimate Conditional Probabilities

$$P(\text{begun} \mid \text{The monsoon season has}) = \frac{\text{Count}(\text{The monsoon season has begun})}{\text{Count}(\text{The monsoon season has})}$$

- **Problem:** Enough data is not available to get an accurate estimate of the above quantities.

Estimate Conditional Probabilities

$$P(\text{begun} \mid \text{The monsoon season has}) = \frac{\text{Count}(\text{The monsoon season has begun})}{\text{Count}(\text{The monsoon season has})}$$

- **Problem:** Enough data is not available to get an accurate estimate of the above quantities.
- **Solution:** Markov Assumption

Markov Assumption

Every next state depends only the previous k states

Markov Assumption

Every next state depends only the previous k states

- Chain Rule:

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

- Applying Markov Assumption we condition on only the preceding k words:

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

Markov Assumption

Every next state depends only the previous k states

- Chain Rule:

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

- Applying Markov Assumption we condition on only the preceding k words:

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

- Probabilistic Language Models exploit the **Chain Rule of Probability** and **Markov Assumption** to build a probability distribution over sequences of words.

N-gram Language Models

- Let's consider the following conditional probability:

$$P(\text{begun} \mid \text{the monsoon season has})$$

- An **N-gram model** considers only the preceding **N – 1 words**.

N-gram Language Models

- Let's consider the following conditional probability:

$$P(\text{begun} \mid \text{the monsoon season has})$$

- An **N-gram model** considers only the preceding **N – 1 words**.

- Unigram: $P(\text{begun})$
- Bigram: $P(\text{begun} \mid \text{the})$
- Trigram: $P(\text{begun} \mid \text{the monsoon})$

N-gram Language Models

- Let's consider the following conditional probability:

$$P(\text{begun} \mid \text{the monsoon season has})$$

- An **N-gram model** considers only the preceding **N – 1 words**.

- Unigram: $P(\text{begun})$
- Bigram: $P(\text{begun} \mid \text{the})$
- Trigram: $P(\text{begun} \mid \text{the monsoon})$

Relation between Markov model and Language Model:

An N-gram Language Model \equiv (N – 1) order Markov Model

Raw bigram counts (absolute measure)

	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 unigram counts (absolute measure)

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

Unigram and bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Previously-zero counts are in gray.

Raw bigram counts (absolute measure)

	i	want	to	eat	chinese	food	lunch	spend	
i	5	827	0	9	0	0	0	2	
want		i	want	to	eat	chinese	food	lunch	spend
to		i	0.002	0.33	0	0.0036	0	0	0.00079
eat		want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054
chinese		to	0.00083	0	0.0017	0.28	0.00083	0	0.0025
food		eat	0	0	0.0027	0	0.021	0.0027	0.056
lunch		chinese	0.0063	0	0	0	0.52	0.0063	0
spend		food	0.014	0	0.014	0	0.00092	0.0037	0
		lunch	0.0059	0	0	0	0.0029	0	0
		spend	0.0036	0	0.0036	0	0	0	0
Raw unigram counts ()	
i	want	to	eat	chinese	food	lunch	spend		
2533	927	2417	746	158	1093	341	278		

Unigram and bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Previously-zero counts are in gray.

Limitation of N-gram Language Models

- An insufficient model of language since they are **not effective in capturing long-range dependencies present in language.**

Limitation of N-gram Language Models

- An insufficient model of language since they are **not effective in capturing long-range dependencies present in language.**
 - Example:

The project, which he had been working on for months, was finally approved by the committee.

The above example highlights the long-distance dependency between “project” and “approved”, where the context provided by earlier words affects the interpretation of later parts of the sentence.

Estimate N-gram Probabilities

- Maximum Likelihood Estimate (MLE):
 - Used to estimate the parameters of a statistical model
 - Determine the most likely values of the parameters that would make the observed data most probable

Estimate N-gram Probabilities

- Maximum Likelihood Estimate (MLE):
 - Used to estimate the parameters of a statistical model
 - Determine the most likely values of the parameters that would make the observed data most probable
- For example, bigram probabilities can be estimated as follows:

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

Limitations with MLE Estimation

Problem: N-grams only work well for word prediction if the test corpus looks like the training corpus. It is often not the case in real scenarios (data sparsity problem).

Limitations with MLE Estimation

Problem: N-grams only work well for word prediction if the test corpus looks like the training corpus. It is often not the case in real scenarios (data sparsity problem).

Training set:

- ... enjoyed the movie
- ... enjoyed the food
- ... enjoyed the game
- ... enjoyed the vacation

Test set:

- ... enjoyed the concert
- ... enjoyed the festival
- ... enjoyed the walk

Zero probability n-grams:

- $P(\text{concert} \mid \text{enjoyed the}) = P(\text{festival} \mid \text{enjoyed the}) = P(\text{walk} \mid \text{enjoyed the}) = 0$
- As a result, the probability of the test set will be 0.
- Perplexity cannot be computed (Cannot divide by 0).

Limitations with MLE Estimation

	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

Solution: Various smoothing techniques

Laplace Smoothing (Add-One Estimation)

- Imagine that we encountered each word (N-gram) one more time than its actual occurrence.
- Simply increase all the counts by one!

Laplace Smoothing (Add-One Estimation)

- Imagine that we encountered each word (N-gram) one more time than its actual occurrence.
- Simply increase all the counts by one!
- MLE estimate (in case of bigram model)

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

- Add-1 estimate:

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

Laplace Smoothing (Add-One Estimation)

- Imagine that we encountered each word (N-gram) one more time than its actual occurrence.
- Simply increase all the counts by one!
- MLE estimate (in case of bigram model)

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

- Add-1 estimate:

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

- Effective bigram count ($c^*(w_{n-1}w_n)$):

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

Comparing with Bigrams: Before and After Smoothing

	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

Add-one smoothed bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Previously-zero counts are in gray.

Example from Speech and Language Processing book by Daniel Jurafsky and James H. Martin

Comparing with Bigrams: Before and After Smoothing

	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

Add-one smoothed **bigram probabilities** for eight of the words (out of $V = 1446$) in the BeRP corpus of 9332 sentences. Previously-zero probabilities are in gray.

Example from Speech and Language Processing book by Daniel Jurafsky and James H. Martin

Comparing with Bigrams: Before and After Smoothing

	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-one reconstituted counts for eight words (of $V = 1446$) in the BeRP corpus of 9332 sentences. Previously-zero counts are in gray.

Example from Speech and Language Processing book by Daniel Jurafsky and James H. Martin

Comparing with Bigrams: Before and After Smoothing

	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

	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

More General Smoothing Techniques

- **Add-k smoothing:**

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

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

More General Smoothing Techniques

- Add-k smoothing:

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

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

- Unigram prior smoothing:

$$P_{\text{UnigramPrior}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + m P(w_i)}{c(w_{i-1}) + m}$$

More General Smoothing Techniques

- Add-k smoothing:

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

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

- Unigram prior smoothing:

$$P_{\text{UnigramPrior}}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + m P(w_i)}{c(w_{i-1}) + m}$$

An optimal value for k or m can be determined using a held-out dataset.

Back-off and Interpolation

- As N grows larger, the N -gram model becomes more powerful. However, its capability to accurately estimate parameters decreases due to data sparsity problem.

Back-off and Interpolation

- As N grows larger, the N-gram model becomes more powerful. However, its capability to accurately estimate parameters decreases due to data sparsity problem.
- When we have limited knowledge about larger contexts, it can be helpful to consider less context.

Back-off and Interpolation

- As N grows larger, the N-gram model becomes more powerful. However, its capability to accurately estimate parameters decreases due to data sparsity problem.
- When we have limited knowledge about larger contexts, it can be helpful to consider less context.
- **Back-off:**
 - Opt for a trigram when there is sufficient evidence, otherwise use bigram, otherwise unigram

Back-off and Interpolation

- As N grows larger, the N-gram model becomes more powerful. However, its capability to accurately estimate parameters decreases due to data sparsity problem.
- When we have limited knowledge about larger contexts, it can be helpful to consider less context.
- **Back-off:**
 - Opt for a trigram when there is sufficient evidence, otherwise use bigram, otherwise unigram
- **Interpolation:**
 - Mix unigram, bigram, trigram
 - Interpolation generally results in improved performance

Interpolation

Linear interpolation

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

Context-dependent interpolation

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