

N-gram Language Models (Continued)

Yu Meng

University of Virginia
yumeng5@virginia.edu

Sep 04, 2024

Announcement: Assignment 1 Out

Join at
slido.com
#4055 785



- Deadline: 09/11 11:59pm
- Released on course website: <https://yumeng5.github.io/teaching/2024-fall-cs4501>

Week	Date	Topic	Slides	Slido Link	Deadline
1	08/28	Course Logistics & Overview	overview_0828	08/28	
	08/30	Course Overview (Continued)	overview_0830	08/30	
2	09/02	Intro to Language Modeling & N-gram Language Models	lm_intro_0902	09/02	Assignment 1 out: LaTeX script

Download the LaTeX script here

Overview of Course Contents

Join at

slido.com

#4055 785



- Week 1: Logistics & Overview
- Week 2: N-gram Language Models
- Week 3: Word Senses, Semantics & Classic Word Representations
- Week 4: Word Embeddings
- Week 5: Sequence Modeling and Transformers
- Week 6-7: Language Modeling with Transformers (Pretraining + Fine-tuning)
- Week 8: Large Language Models (LLMs) & In-context Learning
- Week 9-10: Knowledge in LLMs and Retrieval-Augmented Generation (RAG)
- Week 11: LLM Alignment
- Week 12: Language Agents
- Week 13: Recap + Future of NLP
- Week 15 (after Thanksgiving): Project Presentations

(Recap) Language Modeling Overview

Join at

slido.com

#4055 785

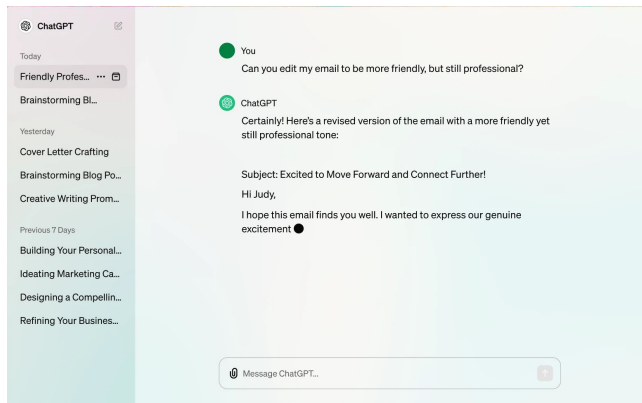


- The core problem in NLP is **language modeling**
- Goal: Assigning probability to a sequence of words
- For text understanding: $p(\text{"The cat is on the mat"}) \gg p(\text{"Truck the earth on"})$
- For text generation: $p(w \mid \text{"The cat is on the"}) \rightarrow \text{"mat"}$

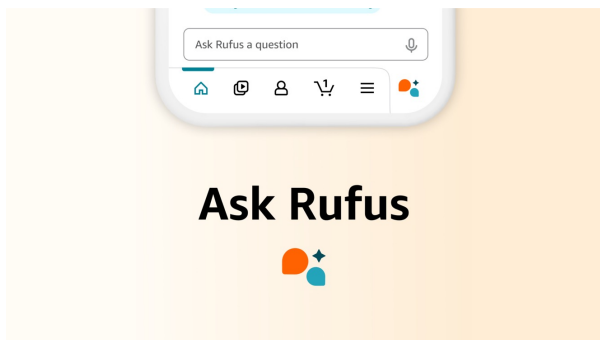
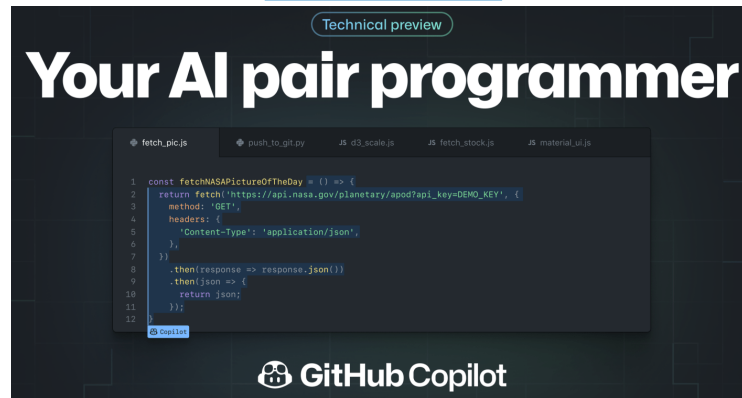


(Recap) Language Model Applications

Chatbots



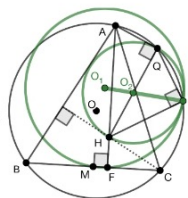
Code Assistants



Shopping Assistants

e IMO 2015 P3

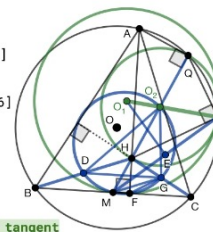
"Let ABC be an acute triangle. Let (O) be its circumcircle, H its orthocenter, and F the foot of the altitude from A . Let M be the midpoint of BC . Let Q be the point on (O) such that $QH \perp QA$ and let K be the point on (O) such that $KH \perp KQ$. Prove that the circumcircles (O_1) and (O_2) of triangles FKM and KQH are tangent to each other."



Alpha-Geometry

f Solution

Construct D : midpoint BH [a]
 $[a], O_2$ midpoint $HQ \Rightarrow BQ \parallel O_2D$ [20]
 ...
 Construct G : midpoint HC [b] ...
 $\angle GMD = \angle GO_2D \Rightarrow M, O_2, G, D$ cyclic [26]
 ...
 $[a], [b] \Rightarrow BC \parallel DG$ [30]
 ...
 Construct E : midpoint MK [c]
 ..., [c] $\Rightarrow \angle KFC = \angle KO_1E$ [104]
 ...
 $\angle FKO_1 = \angle FKO_2 \Rightarrow KO_1 \parallel KO_2$ [109]
 [109] $\Rightarrow O_1, O_2, K$ collinear $\Rightarrow (O_1), (O_2)$ tangent



Generating Math Proofs

(Recap) Universal NLP Task Solvers

Join at

slido.com

#4055 785



- Every NLP task can be converted into a text-to-text task!
 - Sentiment analysis: The movie's closing scene is attractive; it was ____ (good)
 - Machine translation: "Hello world" in French is ____ (Bonjour le monde)
 - Question answering: Which city is UVA located in? ____ (Charlottesville)
 - ...
- All these tasks can be formulated as a language modeling problem!



(Recap) Probability Decomposition

- Given a text sequence $\mathbf{x} = [x_1, x_2, \dots, x_n]$, how can we model $p(\mathbf{x})$?
- Autoregressive assumption: the probability of each word only depends on its previous tokens

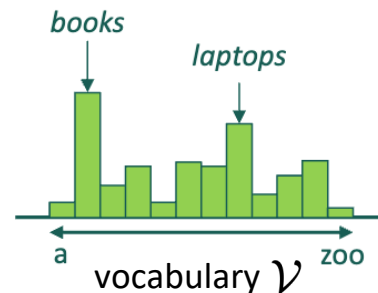
$$p(\mathbf{x}) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots p(x_n|x_1, \dots, x_{n-1}) = \prod_{i=1}^n p(x_i|x_1, \dots, x_{i-1})$$

- How to guarantee the probability distributions are valid?
 - Non-negative

$$p(x_i = w|x_1, \dots, x_{i-1}) \geq 0, \quad \forall w \in \mathcal{V}$$

- Summed to 1:

$$\sum_{w \in \mathcal{V}} p(x_i = w|x_1, \dots, x_{i-1}) = 1$$



- The goal of language modeling is to learn the distribution $p(x_i = w|x_1, \dots, x_{i-1})$!

(Recap) Language Models for Generation

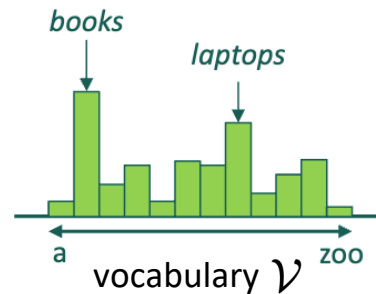
Join at

slido.com

#4055 785



- Suppose we have a language model that gives us the estimate of $p(w|x_1, \dots, x_{i-1})$, we can generate the next tokens one-by-one!
- Sampling: $x_i \sim p(w|x_1, \dots, x_{i-1})$
- Or greedily: $x_i \leftarrow \arg \max_w p(w|x_1, \dots, x_{i-1})$
- But how do we know when to stop generation?
- Use a special symbol [EOS] (end-of-sequence) to denote stopping





(Recap) Language Models for Generation


- Recursively sample $x_i \sim p(w|x_1, \dots, x_{i-1})$ until we generate [EOS]
- Generate the first word: “the” $\leftarrow x_1 \sim p(w|[\text{BOS}])$ **beginning-of-sequence**
- Generate the second word: “cat” $\leftarrow x_2 \sim p(w|“the”)$
- Generate the third word: “is” $\leftarrow x_3 \sim p(w|“the cat”)$
- Generate the fourth word: “on” $\leftarrow x_4 \sim p(w|“the cat is”)$
- Generate the fifth word: “the” $\leftarrow x_5 \sim p(w|“the cat is on”)$
- Generate the sixth word: “mat” $\leftarrow x_6 \sim p(w|“the cat is on the”)$
- Generate the seventh word: [EOS] $\leftarrow x_7 \sim p(w|“the cat is on the mat”)$
- Generation finished!

(Recap) Training Language Models

Join at
slido.com
#4055 785




Learn the probability distribution $p(w|x_1, \dots, x_{i-1})$ from a training corpus!



The Free Encyclopedia

English 6,872,000+ articles	中文 1,437,000+ 条目 / 條目
日本語 1,427,000+ 記事	Русский 1 996 000+ статей
Deutsch 2.937.000+ Artikel	Español 1.974.000+ artículos
Français 2 631 000+ articles	Italiano 1.878.000+ voci
Português 1.132.000+ artigos	فارسی ۱۰۰۱۱۰۰۰+ مقاله



Donald Trump

Article Talk

From Wikipedia, the free encyclopedia

"Joseph Biden" and "Biden" redirect here. For his first-born son, Joseph Biden III, see Beau Biden. For other uses, see Biden (disambiguation).


Joseph Robinette Biden Jr. ^(P) (born November 20, 1942) is an American politician serving as the 46th and current president of the United States since 2021. A member of the Democratic Party, he served as the 47th vice president from 2009 to 2017 under President Barack Obama and represented Delaware in the U.S. Senate from 1973 to 2009.

Born in Scranton, Pennsylvania, Biden moved with his family to Delaware in 1953. He graduated from the University of Delaware in 1965 and from Syracuse University in 1968. He was elected to the New Castle County Council in 1970 and the U.S. Senate in 1972. As a senator, Biden drafted and led the effort to pass the Violent Crime Control and Law Enforcement Act and the Violence Against Women Act. He also oversaw six U.S. Supreme Court confirmation hearings, including the contentious hearings for Robert Bork and Clarence Thomas. Biden ran unsuccessfully for the 1988 and 2008 Democratic presidential nominations. In 2008, Obama chose Biden as his running mate, and he was a close counselor to Obama during his two terms as vice president. In the 2020 presidential election, the Democratic Party nominated Biden for president. He selected Kamala Harris as his running mate, and they defeated Republican incumbents Donald Trump and Mike Pence. He is the oldest president in U.S. history and the first to have a female vice president.

As president, Biden signed the American Rescue Plan Act in response to the COVID-19 pandemic and subsequent recession. He signed bipartisan bills on infrastructure and manufacturing. He proposed the Build Back Better Act, which failed in Congress, but aspects of which were incorporated into the Inflation Reduction Act that he signed into law in 2022. Biden appointed Ketanji Brown Jackson to the Supreme Court. He worked with congressional Republicans to resolve the 2023 debt-ceiling crisis by negotiating a deal to raise the debt ceiling. In foreign policy, Biden restored America's membership in the Paris Agreement. He oversaw the complete withdrawal of U.S. troops from Afghanistan that ended the war in Afghanistan, leading to the collapse of the Afghan government and the Taliban seizing control. He responded to the Russian invasion of

Joe Biden

Article Talk



Joe Biden

Official portrait, 2021

46th President of the United States

Incumbent

Assumed office
January 20, 2021

Vice President
Kamala Harris

Preceded by
Donald Trump

47th Vice President of the United States

In office
January 20, 2009 – January 20, 2017

President
Barack Obama


Preceded by
Dick Cheney

Succeeded by
Mike Pence

United States Senator from Delaware

In office
January 3, 1973 – January 15, 2009

Preceded by
J. Caleb Boggs

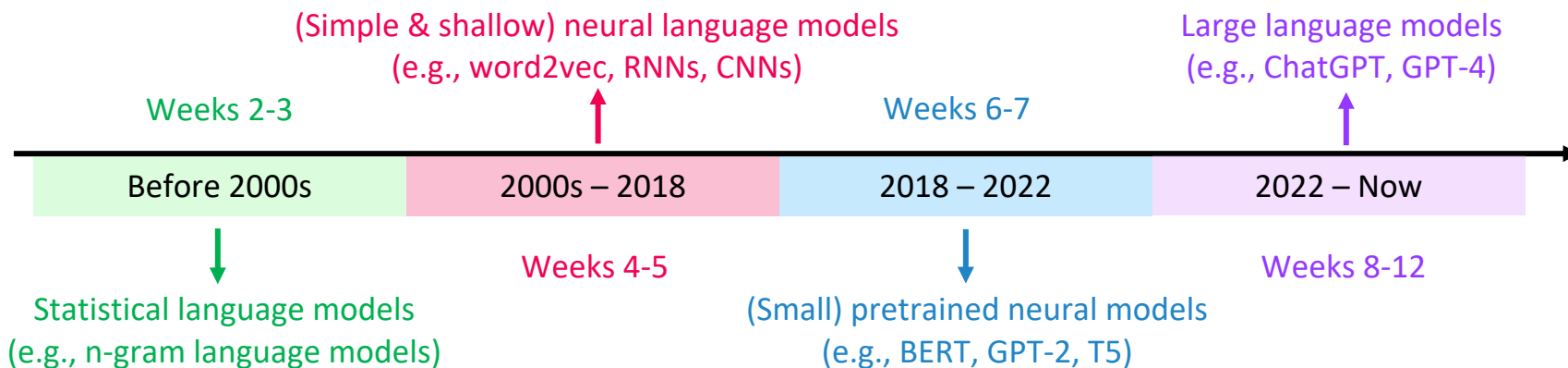
Learning target:
 $p(w|x_1, \dots, x_{i-1})$

Text corpora contain rich distributional statistics!

10/45



(Recap) History of Language Models



Agenda

- Introduction to Language Models
- N-gram Language Models
- Smoothing in N-gram Language Models
- Evaluation of Language Models

Join at
slido.com
#4055 785





(Recap) N-gram Language Model

- Challenge of language modeling: hard to keep track of all previous tokens!

$$p(\mathbf{x}) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$$

Long context!
 (Can we model long contexts at all?
 Yes, but not for now!)

- Instead of keeping track of all previous tokens, assume the probability of a word is only dependent on the previous N-1 words

$$p(\mathbf{x}) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1}) \approx \prod_{i=1}^n p(x_i | x_{i-N+1}, \dots, x_{i-1})$$

N-gram assumption

Should N be larger or smaller?



(Recap) N-gram Language Model

- Unigram LM (N=1): each word's probability does not depend on previous words
- Bigram LM (N=2): each word's probability is based on the previous word
- Trigram LM (N=3): each word's probability is based on the previous two words
- ...
- Example: $p(\text{"The cat is on the mat"})$ For simplicity, omitting [BOS] & [EOS] in these examples
- Unigram: $= p(\text{"The"}) p(\text{"cat"}) p(\text{"is"}) p(\text{"on"}) p(\text{"the"}) p(\text{"mat"})$
- Bigram: $= p(\text{"The"}) p(\text{"cat"} | \text{"The"}) p(\text{"is"} | \text{"cat"}) p(\text{"on"} | \text{"is"}) p(\text{"the"} | \text{"on"}) p(\text{"mat"} | \text{"the"})$
- Trigram: $= p(\text{"The"}) p(\text{"cat"} | \text{"The"}) p(\text{"is"} | \text{"The cat"}) p(\text{"on"} | \text{"cat is"}) p(\text{"the"} | \text{"is on"}) p(\text{"mat"} | \text{"on the"})$
- ...



How to Learn N-grams?

- Probabilities can be estimated by frequencies (maximum likelihood estimation)!

$$p(x_i | x_{i-N+1}, \dots, x_{i-1}) = \frac{\#(x_{i-N+1}, \dots, x_{i-1}, x_i)}{\#(x_{i-N+1}, \dots, x_{i-1})}$$

How many times (counts) the sequences occur in the corpus

- Unigram: $p(x_i) = \frac{\#(x_i)}{\#(\text{all word counts in the corpus})}$
- Bigram: $p(x_i | x_{i-1}) = \frac{\#(x_{i-1}, x_i)}{\#(x_{i-1})}$
- Trigram: $p(x_i | x_{i-2}, x_{i-1}) = \frac{\#(x_{i-2}, x_{i-1}, x_i)}{\#(x_{i-2}, x_{i-1})}$



Practice: Learning Unigrams

- Consider the following mini-corpus:

```
[BOS] The cat is on the mat [EOS]
[BOS] I have a cat and a mat [EOS]
[BOS] I like the cat [EOS]
```

Treating “The” & “the” as
one word

- Unigram estimated from the mini-corpus $p(x_i) = \frac{\#(x_i)}{\#(\text{all word counts in the corpus})}$

$$\begin{aligned}
 p([\text{BOS}]) &= \frac{3}{23}, & p([\text{EOS}]) &= \frac{3}{23}, & p(\text{“the”}) &= \frac{3}{23}, & p(\text{“cat”}) &= \frac{3}{23}, \\
 p(\text{“mat”}) &= \frac{2}{23}, & p(\text{“I”}) &= \frac{2}{23}, & p(\text{“a”}) &= \frac{2}{23}, & p(\text{“have”}) &= \frac{1}{23}, \\
 p(\text{“like”}) &= \frac{1}{23}, & p(\text{“is”}) &= \frac{1}{23}, & p(\text{“on”}) &= \frac{1}{23}, & p(\text{“and”}) &= \frac{1}{23}
 \end{aligned}$$



Unigram Issues: No Word Correlations

- Learned unigram probabilities:

$$\begin{aligned}
 p([\text{BOS}]) &= \frac{3}{23}, & p([\text{EOS}]) &= \frac{3}{23}, & p(\text{"the"}) &= \frac{3}{23}, & p(\text{"cat"}) &= \frac{3}{23}, \\
 p(\text{"mat"}) &= \frac{2}{23}, & p(\text{"I"}) &= \frac{2}{23}, & p(\text{"a"}) &= \frac{2}{23}, & p(\text{"have"}) &= \frac{1}{23}, \\
 p(\text{"like"}) &= \frac{1}{23}, & p(\text{"is"}) &= \frac{1}{23}, & p(\text{"on"}) &= \frac{1}{23}, & p(\text{"and"}) &= \frac{1}{23}
 \end{aligned}$$

- Is unigram reliable for estimating the sequence likelihood?

For simplicity, omitting [BOS] & [EOS] in the calculation

$$p(\text{"the the the the"}) = p(\text{"the"}) \times p(\text{"the"}) \times p(\text{"the"}) \times p(\text{"the"}) \approx 0.0003$$

$$p(\text{"I have a cat"}) = p(\text{"I"}) \times p(\text{"have"}) \times p(\text{"a"}) \times p(\text{"cat"}) \approx 0.00004$$

- Why? Unigram ignores the relationships between words!



Practice: Learning Bigrams

- Consider the following mini-corpus:

[BOS] The cat is on the mat [EOS]

[BOS] I have a cat and a mat [EOS]

[BOS] I like the cat [EOS]

Treating “The” & “the” as
one word

- Bigram estimated from the mini-corpus $p(x_i|x_{i-1}) = \frac{\#(x_{i-1}, x_i)}{\#(x_{i-1})}$

$$p(\text{“I”} | [\text{BOS}]) = \frac{2}{3}, \quad p(\text{“The”} | [\text{BOS}]) = \frac{1}{3}, \quad p([\text{EOS}] | \text{“mat”}) = 1, \quad p([\text{EOS}] | \text{“cat”}) = \frac{1}{3},$$

$$p(\text{“cat”} | \text{“the”}) = \frac{2}{3}, \quad p(\text{“mat”} | \text{“the”}) = \frac{1}{3}, \quad p(\text{“is”} | \text{“cat”}) = \frac{1}{3}, \quad p(\text{“and”} | \text{“cat”}) = \frac{1}{3},$$

$$p(\text{“have”} | \text{“I”}) = \frac{1}{2}, \quad p(\text{“like”} | \text{“I”}) = \frac{1}{2}, \quad p(\text{“a”} | \text{“have”}) = 1, \quad p(\text{“cat”} | \text{“a”}) = \frac{1}{2}$$

... there are more bigrams!



Bigram Issues: Sparsity

- Learned bigram probabilities:

$$\begin{aligned}
 p(\text{"I"} | [\text{BOS}]) &= \frac{2}{3}, & p(\text{"The"} | [\text{BOS}]) &= \frac{1}{3}, & p([\text{EOS}] | \text{"mat"}) &= 1, & p([\text{EOS}] | \text{"cat"}) &= \frac{1}{3}, \\
 p(\text{"cat"} | \text{"the"}) &= \frac{2}{3}, & p(\text{"mat"} | \text{"the"}) &= \frac{1}{3}, & p(\text{"is"} | \text{"cat"}) &= \frac{1}{3}, & p(\text{"and"} | \text{"cat"}) &= \frac{1}{3}, \\
 p(\text{"have"} | \text{"I"}) &= \frac{1}{2}, & p(\text{"like"} | \text{"I"}) &= \frac{1}{2}, & p(\text{"a"} | \text{"have"}) &= 1, & p(\text{"cat"} | \text{"a"}) &= \frac{1}{2}
 \end{aligned}$$

- Does bigram address the issue of unigram?

For simplicity, omitting [EOS] in the calculation

$$p(\text{"the the the the"}) = p(\text{"the"} | [\text{BOS}]) \times p(\text{"the"} | \text{"the"}) \times p(\text{"the"} | \text{"the"}) \times p(\text{"the"} | \text{"the"}) = 0$$

$$p(\text{"I have a cat"}) = p(\text{"I"} | [\text{BOS}]) \times p(\text{"have"} | \text{"I"}) \times p(\text{"a"} | \text{"have"}) \times p(\text{"cat"} | \text{"a"}) \approx 0.17$$

- But... $p(\text{"a cat"}) = p(\text{"a"} | [\text{BOS}]) \times p(\text{"cat"} | \text{"a"}) = 0$

Sparsity: Valid bigrams having zero probability due to no occurrence in the training corpus

Bigram Issues: Sparsity

 Join at
slido.com
#4055 785


Bigram counts can be mostly zero even for larger corpora!

Berkeley Restaurant Project Corpus
 (>9K sentences)

can you tell me about any good cantonese restaurants close by
 tell me about chez panisse
 i'm looking for a good place to eat breakfast
 when is caffe venezia open during the day

Second word

First word

	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

Lots of zero entries!



Practice: Learning Trigrams

- Consider the following mini-corpus:

```
[BOS] The cat is on the mat [EOS]
[BOS] I have a cat and a mat [EOS]
[BOS] I like the cat [EOS]
```

Treating “The” & “the” as
one word

- Trigram estimated from the mini-corpus $p(x_i | x_{i-2}, x_{i-1}) = \frac{\#(x_{i-2}, x_{i-1}, x_i)}{\#(x_{i-2}, x_{i-1})}$

$$\begin{aligned}
 p(\text{“like”} | [\text{BOS}], \text{“I”}) &= \frac{1}{2}, & p(\text{“have”} | [\text{BOS}], \text{“I”}) &= \frac{1}{2}, & p([\text{EOS}] | \text{“the”, “mat”}) &= 1, \\
 p(\text{“is”} | \text{“the”, “cat”}) &= \frac{1}{2}, & p([\text{EOS}] | \text{“the”, “cat”}) &= \frac{1}{2}, & p([\text{EOS}] | \text{“a”, “mat”}) &= 1, \\
 p(\text{“the”} | \text{“I”, “like”}) &= 1, & p(\text{“a”} | \text{“I”, “have”}) &= 1, & p(\text{“mat”} | \text{“on”, “the”}) &= 1
 \end{aligned}$$

Sparsity grows compared to bigram!

... there are more trigrams!

N-gram Properties

Join at

slido.com

#4055 785



- As N becomes larger
 - Better modeling of word correlations (incorporating more contexts)
 - Sparsity increases
- The number of possible N-grams (parameters) grows exponentially with N!
 - Suppose vocabulary size = 10K words
 - Possible unigrams = 10K
 - Possible bigrams = $(10K)^2 = 100M$
 - Possible trigrams = $(10K)^3 = 1T$
 - ...

N-gram Sparsity

Join at
slido.com
#4055 785



With a larger N, the context becomes more specific, and the chances of encountering any particular N-gram in the training data are lower

```
198015222 the first
194623024 the same
168504105 the following
158562063 the world
...
14112454 the door
-----
23135851162 the *
```

Bigram counts

```
197302 close the window
191125 close the door
152500 close the gap
116451 close the thread
87298 close the deal
-----
3785230 close the *
```

Trigram counts

```
3380 please close the door
1601 please close the window
1164 please close the new
1159 please close the gate
...
0 please close the first
-----
13951 please close the *
```

4-gram counts

Agenda

- Introduction to Language Models
- N-gram Language Models
- Smoothing in N-gram Language Models
- Evaluation of Language Models

Join at
slido.com
#4055 785





Addressing Sparsity in N-gram Language Models #4055 785

- Unseen N-grams in the training corpus always lead to a zero probability
- The entire sequence will have a zero probability if any of the term is zero!

$$p(\mathbf{x}) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1}) \approx \prod_{i=1}^n p(x_i | x_{i-N+1}, \dots, x_{i-1})$$

All terms must be non-zero

- Can we fix zero-probability N-grams?

Smoothing

Join at

slido.com

#4055 785



- Intuition: guarantee all N-grams have non-zero probabilities regardless of their counts in the training corpus
- Smoothing techniques:
 - Add-one smoothing (Laplace smoothing)
 - Add- k smoothing
 - Language model interpolation
 - Backoff
 - ...



Add-one Smoothing (Laplace Smoothing)

Add one to all the N-gram counts!

Original 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

Smoothed counts

	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 Smoothing (Laplace Smoothing)

Original (no smoothing):
$$p(x_i | x_{i-N+1}, \dots, x_{i-1}) = \frac{\#(x_{i-N+1}, \dots, x_{i-1}, x_i)}{\#(x_{i-N+1}, \dots, x_{i-1})}$$

- Probability of N-grams under add-one smoothing

Add-one smoothing:
$$p_{\text{Add-1}}(x_i | x_{i-N+1}, \dots, x_{i-1}) = \frac{\#(x_{i-N+1}, \dots, x_{i-1}, x_i) + 1}{\#(x_{i-N+1}, \dots, x_{i-1}) + |\mathcal{V}|}$$

↓
Vocabulary size

- Issues? Over-smoothing: too much probability mass allocated to unseen N-grams



Add- k Smoothing

- Instead of adding 1 to each count, we add a fractional count k ($k < 1$) to all N-grams

Original (no smoothing):

$$p(x_i | x_{i-N+1}, \dots, x_{i-1}) = \frac{\#(x_{i-N+1}, \dots, x_{i-1}, x_i)}{\#(x_{i-N+1}, \dots, x_{i-1})}$$

Add-one smoothing:

$$p_{\text{Add-1}}(x_i | x_{i-N+1}, \dots, x_{i-1}) = \frac{\#(x_{i-N+1}, \dots, x_{i-1}, x_i) + 1}{\#(x_{i-N+1}, \dots, x_{i-1}) + |\mathcal{V}|}$$

- Probability of N-grams under add- k smoothing

Add- k smoothing:

$$p_{\text{Add-}k}(x_i | x_{i-N+1}, \dots, x_{i-1}) = \frac{\#(x_{i-N+1}, \dots, x_{i-1}, x_i) + k}{\#(x_{i-N+1}, \dots, x_{i-1}) + k|\mathcal{V}|}$$

- How to choose k ? Use a validation set!



Smoothing via Language Model Interpolation

- Intuition: Combine the advantages of different N-grams
 - Lower-order N-grams (e.g., unigrams) capture less context but are also less sparse
 - Higher-order N-grams (e.g., trigrams) capture more context but are also more sparse
- Combine probabilities from multiple N-gram models of different Ns (e.g., unigrams, bigrams, trigrams)

$$p_{\text{Interpolate}}(x_i | x_{i-N+1}, \dots, x_{i-1}) = \lambda_1 p(x_i) + \lambda_2 p(x_i | x_{i-1}) + \dots + \lambda_N p(x_i | x_{i-N+1}, \dots, x_{i-1})$$

Unigram

Bigram

N-gram

$$\sum_{n=1}^N \lambda_n = 1 \quad \text{Interpolation weights sum to 1}$$

- How to pick λ_n ? Use a validation set!



Smoothing via Backoff

- Start with the highest-order N-gram available
- If that N-gram is not available (has a zero count), use the lower-order (N-1)-gram
- Continue backing off to lower-order N-grams until we reach a non-zero N-gram

$$p_{\text{Backoff}}(x_i | x_{i-N+1}, \dots, x_{i-1}) = \begin{cases} p_{\text{Backoff}}(x_i | x_{i-N+1}, \dots, x_{i-1}) & \text{If } \#(x_{i-N+1}, \dots, x_{i-1}, x_i) > 0 \\ \alpha \cdot p_{\text{Backoff}}(x_i | x_{i-N+2}, \dots, x_{i-1}) & \text{Otherwise} \end{cases}$$

↓
↘

$\alpha (<1)$: discount factor that adjusts the lower-order probability
 (N-1)-gram probability

- Is it possible that even after backing off to unigram, the probability is still zero?

Out-of-vocabulary Words

Join at

slido.com

#4055 785



- Unigrams will have a zero probability for words not occurring in the training data!
- Simple remedy: reserve a special token [UNK] for unknown/unseen words
- During testing, convert unknown words to [UNK] -> use [UNK]'s probability
- How to estimate the probability of [UNK]?
- During training, replace all rare words with [UNK], and estimate its probability as if it is a normal word
- How to determine rare words? Threshold based on counts in the training corpus
- Example: set a fixed vocabulary size of 10K, and words outside the most frequent 10K will be converted to [UNK] in training

Agenda

- Introduction to Language Models
- N-gram Language Models
- Smoothing in N-gram Language Models
- Evaluation of Language Models

Join at
slido.com
#4055 785



How to Evaluate Language Models?

Join at

slido.com

#4055 785



- What language models should be considered “good”?
 - A perfect language model should be able to correctly predict every word in a corpus
 - We hope the language model can assign a high probability to the next word
 - Better language model = “less surprised” by the next word
- Just use the next word probability assigned by a language model as the metric!
- Does the choice of the evaluation corpus matter?

Training/Validation/Test Corpus

Join at
slido.com
#4055 785



- **Training corpus/set:** The text data we train our models on
- Does it make sense to evaluate language model probability on the training corpus?
- If we evaluate on the training corpus, we will get misleadingly high probabilities for next word prediction -> train-test data leakage
- **Test corpus/set:** A held-out set of data without overlapping with the training set
- We should always evaluate the model performance using the test corpus which measures the model's generalization ability to unseen data!
- Test sets should **NOT** be used to evaluate language models many times for tuning hyperparameters/design choices -> indirectly learn from test set characteristics
- **Validation/development corpus/set (optional):** Tuning hyperparameters & making design choices before evaluating on the test set

Training/Validation/Test Split

Join at
slido.com
#4055 785



- If we have a fixed amount of data, how should we split into train/valid/test sets?
- We want the training set to be as large as possible
- But the validation/test sets should be also reasonably large to yield reliable evaluation results
- The test set should reflect the data/task we aim to apply language models to



Perplexity

- Perplexity (abbreviation: PPL) is an **intrinsic** evaluation metric for language models
- PPL = the per-word inverse probability on a test sequence $\mathbf{x}_{\text{test}} = [x_1, x_2, \dots, x_n]$

$$\text{PPL}(\mathbf{x}_{\text{test}}) = \sqrt[n]{\prod_{i=1}^n \frac{1}{p(x_i | x_{i-N+1}, \dots, x_{i-1})}}$$

- A lower PPL = a better language model (less surprised/confused by the next word)

$$\text{PPL}(\mathbf{x}_{\text{test}}) = \sqrt[n]{\prod_{i=1}^n \frac{1}{p(x_i)}}$$

Unigram

$$\text{PPL}(\mathbf{x}_{\text{test}}) = \sqrt[n]{\prod_{i=1}^n \frac{1}{p(x_i | x_{i-1})}}$$

Bigram

$$\text{PPL}(\mathbf{x}_{\text{test}}) = \sqrt[n]{\prod_{i=1}^n \frac{1}{p(x_i | x_{i-2}, x_{i-1})}}$$

Trigram

Perplexity can be used to evaluate general language models (e.g., large language models) too



Perplexity: Log-Scale Computation

- Computation of PPL in the raw probability scale can cause numerical instability

$$\text{PPL}(\mathbf{x}_{\text{test}}) = \sqrt[n]{\prod_{i=1}^n \frac{1}{p(x_i | x_{i-N+1}, \dots, x_{i-1})}}$$

Multiplication of many small probability values!

Example: $(1/10)^{100} = 10^{-100} \rightarrow$ risks of underflow (round to 0)

- PPL is usually computed in the log-scale in practice

$$\text{PPL}(\mathbf{x}_{\text{test}}) = \exp \left(\log \left(\sqrt[n]{\prod_{i=1}^n \frac{1}{p(x_i | x_{i-N+1}, \dots, x_{i-1})}} \right) \right) = \exp \left(-\frac{1}{n} \sum_{i=1}^n \log p(x_i | x_{i-N+1}, \dots, x_{i-1}) \right)$$

Log probabilities are numerically stable

Example: $\log(1/10) = -2.3$



Perplexity: Important Intrinsic Metric

PPL is an important metric to benchmark the development of language models

Language Modelling on WikiText-2

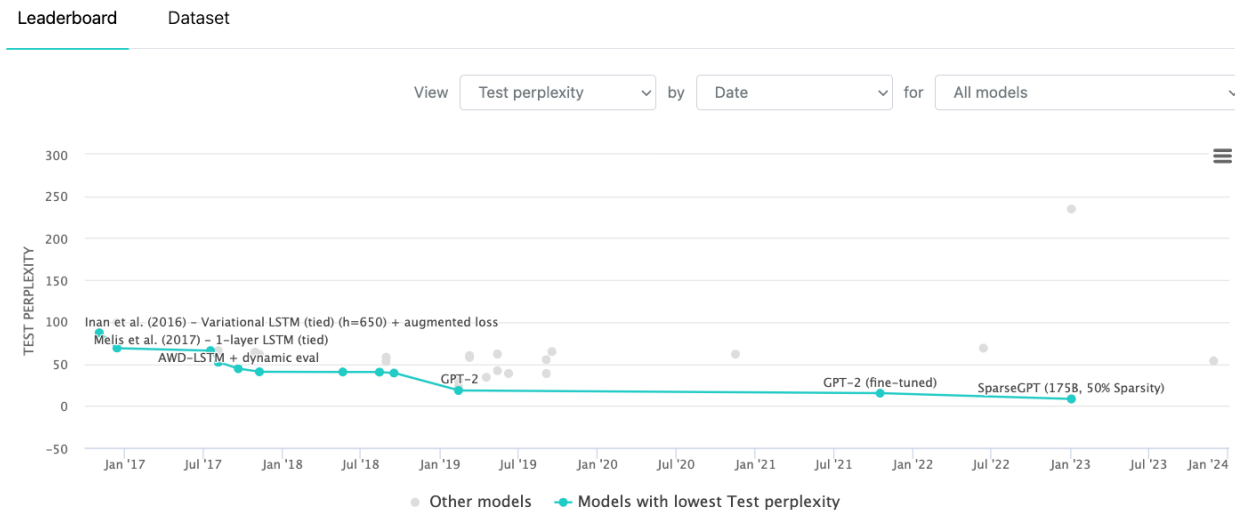


Figure source: <https://paperswithcode.com/sota/language-modelling-on-wikitext-2>

Intrinsic vs. Extrinsic Evaluation

Join at
slido.com
#4055 785



- **Intrinsic metrics** (e.g., perplexity) directly measure the quality of language modeling per se, independent of any application
- **Extrinsic metrics** (e.g., accuracy) measure the language model's performance for specific tasks/applications (e.g., classification, translation)
- Intrinsic evaluations are good during the development to iterate quickly and understand specific properties of the model
- Extrinsic evaluations are essential to validate that the model improves the performance of an application in a real-world scenario
- Both intrinsic and extrinsic evaluations are commonly used to evaluation language models (they may not be always positively correlated!)



Extrinsic Evaluations for SOTA Language Models

Math reasoning, question answering, general knowledge understanding...

😊 Open LLM Leaderboard

Model	BBH	MATH Lv1 5	GPQA	MUSR	MMLU-PRO
MaziyarPanahi/calme-2.1-rys-78b	59.47	36.4	19.24	19	49.38
MaziyarPanahi/calme-2.2-rys-78b	59.27	37.92	20.92	16.83	48.73
MaziyarPanahi/calme-2.1-qwen2-72b	57.33	36.03	17.45	20.15	49.05
MaziyarPanahi/calme-2.2-qwen2-72b	56.8	41.16	16.55	16.52	49.27
Qwen/Qwen2-72B-Instruct	57.48	35.12	16.33	17.17	48.92
alpindale/magnum-72b-v1	57.65	35.27	18.79	15.62	49.64
meta-llama/Meta-Llama-3.1-70B-Instruct	55.93	28.02	14.21	17.69	47.88
abacusai/Smaug-Qwen2-72B-Instruct	56.27	35.35	14.88	15.18	46.56
MaziyarPanahi/calme-2.2-llama3-70b	48.57	22.96	12.19	15.3	46.74
NousResearch/Hermes-3-Llama-3.1-70B	53.77	13.75	14.88	23.43	41.41
tenyx/Llama3-TenyxChat-70B	49.62	22.66	6.82	12.52	46.78

Summary: Language Modeling

Join at

slido.com

#4055 785



- Language modeling is the core problem in NLP
- Every NLP task can be formulated as language modeling
- (Autoregressive) language models can be used to generate texts
- Language model distributions are estimated (trained) on a training corpus



Summary: N-gram Language Models

- N-gram language models simplifies the (general) language modeling assumption: the probability of a word is only dependent on the previous $N-1$ words
- Lower-order N-grams (small N) capture less context information/word correlations
- Higher-order N-grams (bigger N) suffer from more sparsity and huge parameter space
- Smoothing techniques can be used to address sparsity in N-gram language models
 - Add-one smoothing
 - Add- k smoothing
 - Language model interpolation
 - Backoff

Summary: Language Model Evaluation

Join at

slido.com

#4055 785



- Training/validation/test split required before training & evaluating language models
- Perplexity measures how “confused” the language model is about the next word
- Lower perplexity on the test set = better language model
- Perplexity is the commonly used intrinsic evaluation metric for language modeling
- Perplexity is practically computed in the log scale
- Both intrinsic and extrinsic evaluations are important



Thank You!

Yu Meng

University of Virginia

yumeng5@virginia.edu