Introduction to Language Modeling & N-gram Language Models

Yu Meng
University of Virginia
yumeng5@virginia.edu

Sep 02, 2024

Announcement: Assignment 1 Out

Join at
slido.com
#7035 481

- 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

Join at
slido.com
#7035 481

Overview of Course Contents

- 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

Agenda

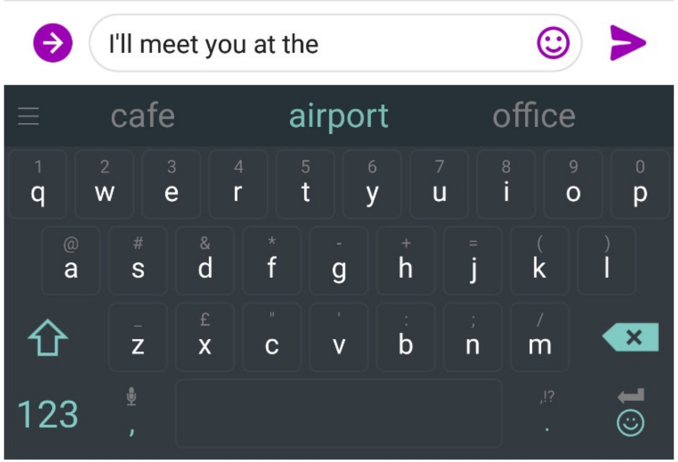
- Introduction to Language Models
- N-gram Language Models
- Smoothing in N-gram Language Models
- Evaluation of Language Models

UNIVERSITY of VIRGINIA

Join at
slido.com
#7035 481

Overview: Language Modeling

- The core problem in NLP is **language modeling**
- Goal: Assigning probability to a sequence of words
- For text understanding: *p*("The cat is on the mat") >> *p*("Truck the earth on")
- For text generation: *p*(*w* | "The cat is on the") -> "mat"



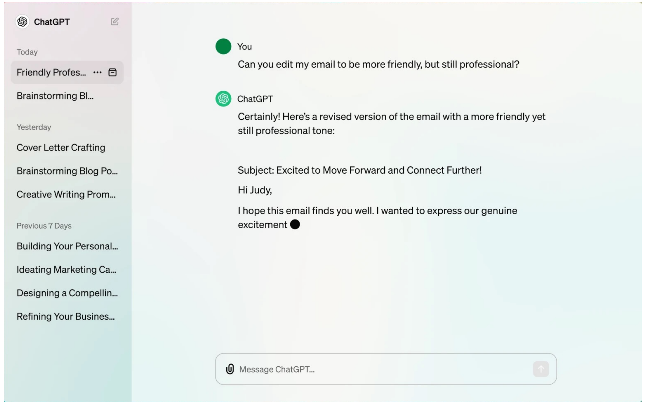
Autocomplete empowered by language modeling

UNIVERSITY of VIRGINIA

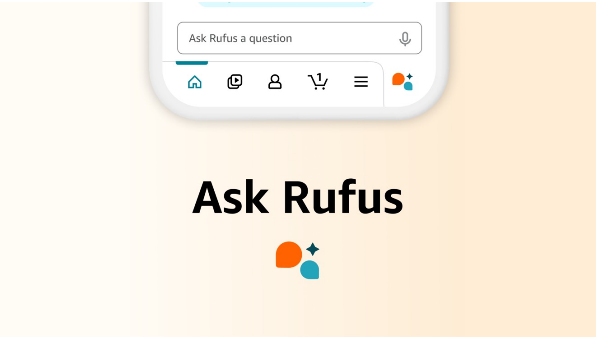
Language Model Applications

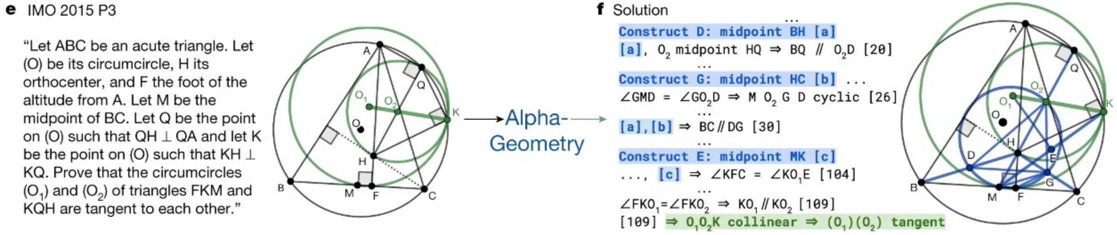
Join at
slido.com
#7035 481

Chatbots



Code Assistants





Shopping Assistants

Generating Math Proofs

Language Models = Universal NLP Task Solvers

- 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!

Language Modeling: Probability Decomposition

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

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

- Are there other possible decomposition assumptions?
  - Yes, but they are not considered "conventional" language models
  - We'll see in word embedding/BERT lectures

Language Modeling: Probability Decomposition

Join at
slido.com
#7035 481

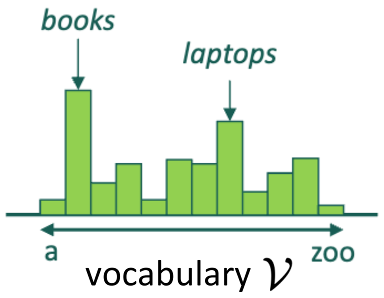
- Given a text sequence $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$, how can we model $p(\boldsymbol{x})$?
- Autoregressive assumption: the probability of each word only depends on its previous tokens

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

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

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

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



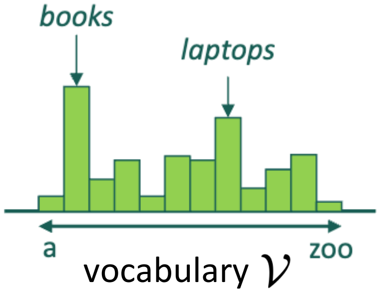
- The goal of language modeling is to learn the distribution $p(x_i = w|x_1,\ldots,x_{i-1})$!

UNIVERSITY of VIRGINIA

Join at slido.com #7035 481

Language Models Are Generative Models

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



Example: Language Models for Generation

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

Join at
slido.com
#7035 481

How to Obtain A Language Model?

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

Donald Trump

242 languages

Article Talk Read View source View history Tools

Joe Biden

218 languages

Article Talk Read View source View history Tools

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.[b] (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

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!

History of Language Models

- Language models started to be built with statistical methods
  - Sparsity
  - Poor generalization

Weeks 2-3

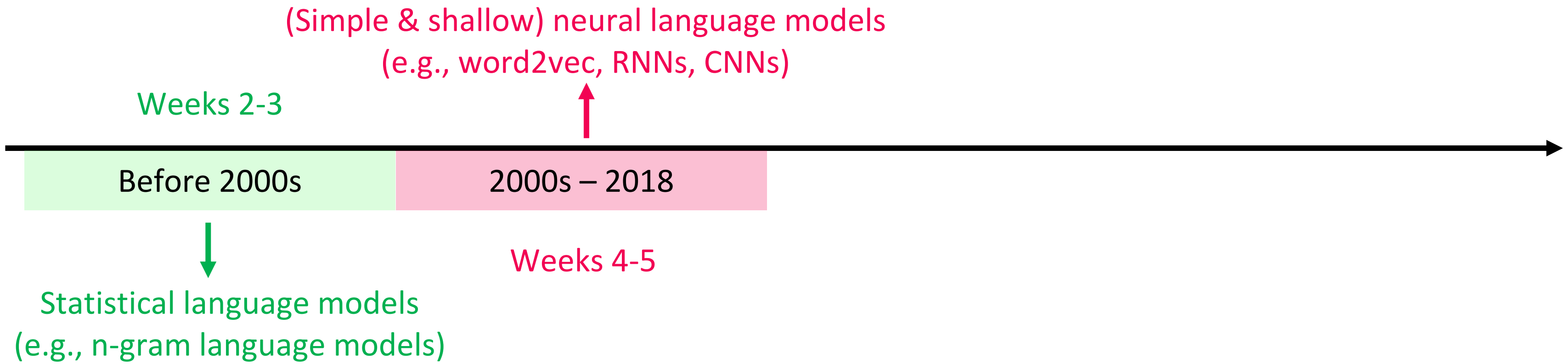
Before 2000s

↓

Statistical language models
(e.g., n-gram language models)

History of Language Models

- The introduction of neural networks into language models mitigated sparsity and improved generalization
  - Neural networks for language models were small-scale and inefficient for a long time
  - Task-specific architecture designs required for different NLP tasks
  - These language models were trained on individual NLP tasks as task-specific solvers



History of Language Models

- Transformer became the dominant architecture for language modeling; scaling up model sizes and (pretraining) data enabled significant generalization ability
  - Transformer demonstrated striking scalability and efficiency in sequence modeling
  - One pretrained model checkpoint fine-tuned to become strong task-specific models
  - Task-specific fine-tuning was still necessary

(Simple & shallow) neural language models
(e.g., word2vec, RNNs, CNNs)

Weeks 2-3

Weeks 6-7

Before 2000s | 2000s – 2018 | 2018 – 2022

Weeks 4-5

Statistical language models
(e.g., n-gram language models)

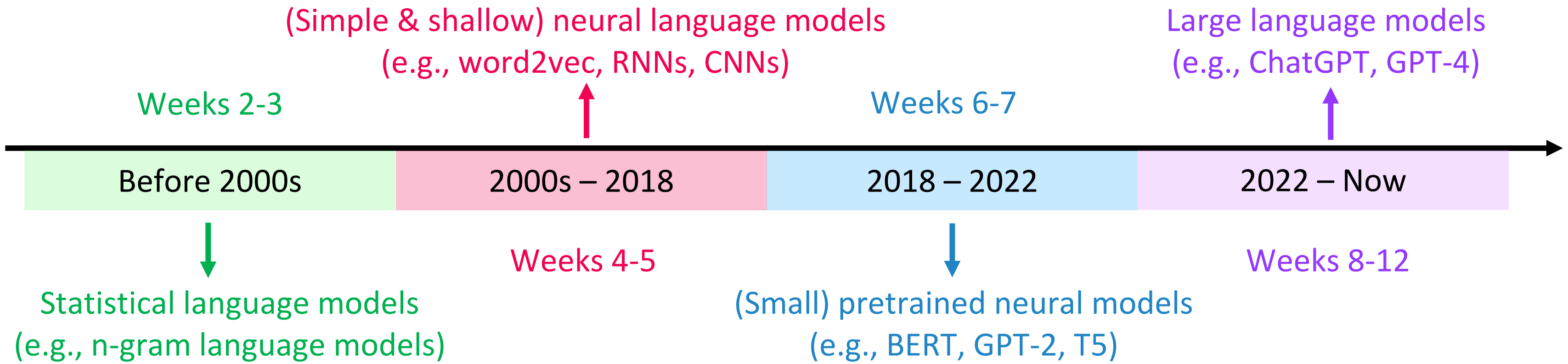
(Small) pretrained neural models
(e.g., BERT, GPT-2, T5)

UNIVERSITY of VIRGINIA

Join at
slido.com
#7035 481

History of Language Models

- Generalist large language models (LLMs) became the universal task solvers and replaced task-specific language models
  - Real-world NLP applications are usually multifaceted (require composite task abilities)
  - Tasks are not clearly defined and may overlap
  - Single-task models struggle to handle complex tasks



Agenda

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

N-gram Language Model: Simplified Assumption

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

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

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

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

N-gram assumption

Should N be larger or smaller?

N-gram Language Model: Simplified Assumption

- 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("The cat is on the mat") For simplicity, omitting [BOS] & [EOS] in these examples
- Unigram: = p("The") p("cat") p("is") p("on") p("the") p("mat")
- Bigram: = p("The") p("cat" | "The") p("is" | "cat") p("on" | "is") p("the" | "on") p("mat" | "the")
- Trigram: = p("The") p("cat" | "The") p("is" | "The cat") p("on" | "cat is") p("the" | "is on") p("mat" | "on the")
- …

How to Learn N-grams?

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

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

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

- Unigram: $p(x_i) = \dfrac{\#(x_i)}{\#(\text{all word counts in the corpus})}$

- Bigram: $p(x_i|x_{i-1}) = \dfrac{\#(x_{i-1},x_i)}{\#(x_{i-1})}$

- Trigram: $p(x_i|x_{i-2},x_{i-1}) = \dfrac{\#(x_{i-2},x_{i-1},x_i)}{\#(x_{i-2},x_{i-1})}$

Practice: Learning Unigrams

- Consider the following mini-corpus:

$$\text{[BOS] The cat is on the mat [EOS]}$$
$$\text{[BOS] I have a cat and a mat [EOS]}$$
$$\text{[BOS] I like the cat [EOS]}$$

Treating "The" & "the" as one word

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

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

Unigram Issues: No Word Correlations

- Learned unigram probabilities:

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

- 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:

$$\text{[BOS] The cat is on the mat [EOS]}$$
$$\text{[BOS] I have a cat and a mat [EOS]}$$
$$\text{[BOS] I like the cat [EOS]}$$

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!

UNIVERSITY of VIRGINIA

Join at slido.com #7035 481

Bigram Issues: Sparsity

- Learned unigram probabilities:

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

- 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

UNIVERSITY of VIRGINIA

Join at
slido.com
#7035 481

Bigram Issues: Sparsity

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

Figure source: https://web.stanford.edu/~jurafsky/slp3/3.pdf

University of Virginia

Join at slido.com #7035 481

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 $p(x_i|x_{i-2}, x_{i-1}) = \frac{\#(x_{i-2}, x_{i-1}, x_i)}{\#(x_{i-2}, x_{i-1})}$

$$p(\text{"like"}|[\text{BOS}], \text{"I"}) = \frac{1}{2}, \quad p(\text{"have"}|[\text{BOS}], \text{"I"}) = \frac{1}{2}, \quad p([\text{EOS}]|\text{"the"}, \text{"mat"}) = 1,$$
$$p(\text{"is"}|\text{"the"}, \text{"cat"}) = \frac{1}{2}, \quad p([\text{EOS}]|\text{"the"}, \text{"cat"}) = \frac{1}{2}, \quad p([\text{EOS}]|\text{"a"}, \text{"mat"}) = 1,$$
$$p(\text{"the"}|\text{"I"}, \text{"like"}) = 1, \quad p(\text{"a"}|\text{"I"}, \text{"have"}) = 1, \quad p(\text{"mat"}|\text{"on"}, \text{"the"}) = 1$$

Sparsity grows compared to bigram! ... there are more trigrams!

N-gram Properties

- 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

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

Figure source: https://lm-class.org/lectures/05%20-%20language%20models.pdf

UNIVERSITY of VIRGINIA

Join at slido.com #7035 481

Agenda

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

Overcoming Sparsity in N-gram Language Models

- 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!

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

All terms must be non-zero

- Can we fix zero-probability N-grams?

Smoothing

- 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

Figure source: https://web.stanford.edu/~jurafsky/slp3/3.pdf

Add-one Smoothing (Laplace Smoothing)

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

- Probability of N-grams under add-one smoothing

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

Vocabulary size

- Issues? Over-smoothing: too much probability mass 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)

$$p_{\text{Interpolate}}(x_i|x_{i-N+1},\ldots,x_{i-1}) = \underbrace{\lambda_1 p(x_i)}_{\text{Unigram}} + \underbrace{\lambda_2 p(x_i|x_{i-1})}_{\text{Bigram}} + \cdots + \underbrace{\lambda_N p(x_i|x_{i-N+1},\ldots,x_{i-1})}_{\text{N-gram}}$$

$$\sum_{n=1}^{N} \lambda_n = 1$$

Interpolation weights sum to 1

- How to pick λ_n ? Use a validation set!

Smoothing via Backoff

Join at
slido.com
#7035 481

- 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

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

α (<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

- 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

Thank You!

Yu Meng
University of Virginia
yumeng5@virginia.edu