COMP 3361 Natural Language Processing

Lecture 3: Language Modeling

n-gram Language Models (cont'd)

Spring 2025

Many materials from CSE517@UW, COMS W4705@Columbia, 11-711@CMU, COS484@Princeton with special thanks!

Announcements

- Course website: https://taoyds.github.io/courses/comp3361
  - TA office hours
  - A total of 5 slip days to use for assignments/projects
- Join Slack!!! via https://join.slack.com/t/comp3361-25spring/shared_invite/zt-2y7hsquia-UPfXw2RI8Uf_h5YIbkje6w
- Assignment 1 will be out this weekend. Due in 4 weeks.
  - TA will provide in-class coding tutorials to help you with each assignment
- Course enrollment: talk to me during class break

Latest AI news

- Alibaba's release of Qwen2.5-Max and Qwen2.5-VL
- OpenAI Deep Research, prev. OpenAI Computer Use Agents
- Andrej Karpathy's 3-hour course on Deep Dive into LLMs like ChatGPT
- Anthropic CEO On DeepSeek and Export Controls
- Sasha Rush's talk on How DeepSeek Changes the LLM Story

Lecture plan

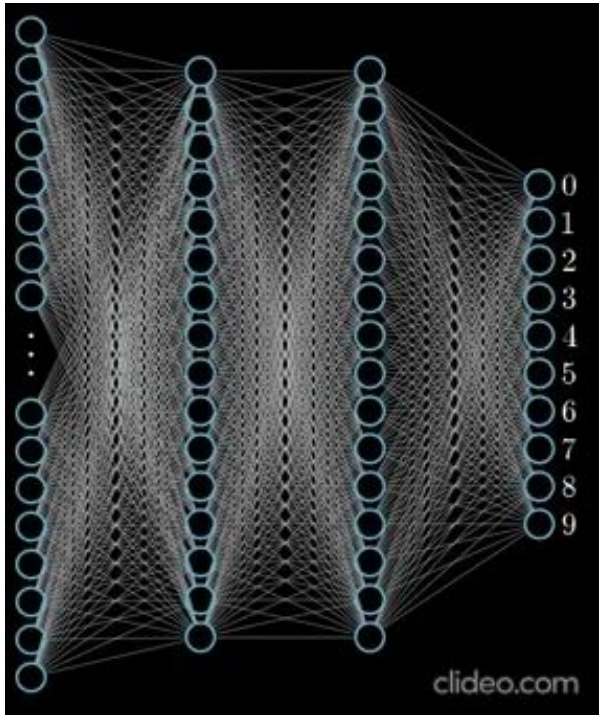
- Introduction to language models
- N-gram language models
- Language model evaluation
- Smoothing methods

Generative language model

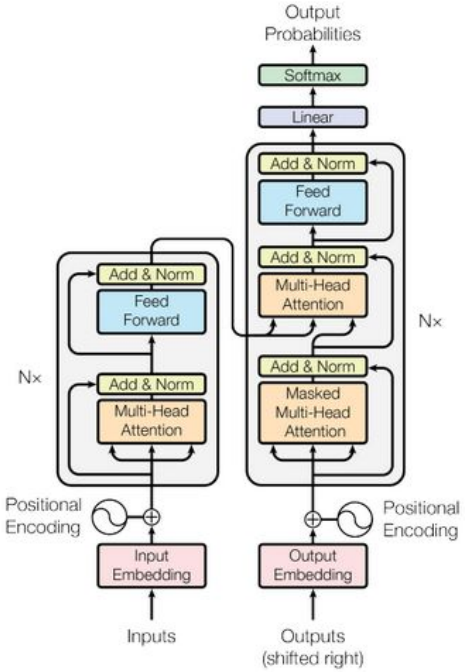
I am going to do an internship in Google

Google

Neuralize the dice!

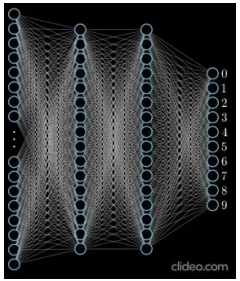


Neural Networks (e.g. Transformers)



Neural network language models

I am going to do an internship in Google



Google

Language models, and how to build it

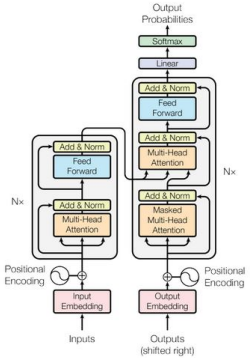
Dice, and how do we roll them
(probabilistic model)

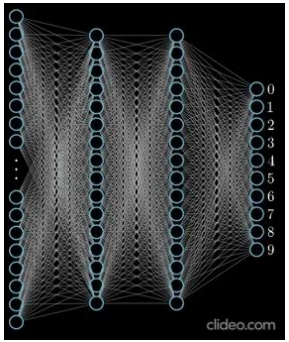
Transformers, neural networks and many others
(powerful functions)

$$p(\boldsymbol{x}) = \prod_i p(x_i|\boldsymbol{x}_{<i})$$

Learn

Parameterize





First problem — the language modeling problem

Given a finite vocabulary

$$\mathcal{V} = \{\text{belief}, \text{evidence}, \text{reason}, \text{claim}, \ldots \text{Google}, \text{therefore}\}$$

We have a set of sentences

<s> I am going to an internship in Google </s>
<s> an internship in Google </s>
<s> I am going going </s>
<s> Google is am </s>
<s> internship is going </s>

Can we learn a "model" for this "generative process"? We need to "learn" a probability distribution:

$$p(x_1, x_2, \ldots x_n)$$

Learn from what we've seen

The language modeling problem

Given a *training sample* of example sentences, we need to "learn" a probabilistic model that assigns probabilities to every possible string:

p(<s> I am going to an internship in Google </s>) = 10^{-12}

p(<s> an internship in Google </s>) = 10^{-8}

p(<s> I am going going </s>) = 10^{-15}

…

What is a language model?

- A probabilistic model of a sequence of words $x_1, x_2, \ldots x_n$

  A language model consists of

  - A finite set $\mathcal{V} = \{\text{the}, \text{dog}, \text{laughs}, \text{saw}, \text{barks}, \text{cat}, \ldots\}$

What is a language model?

- A probabilistic model of a sequence of words $x_1, x_2, \ldots x_n$

A language model consists of

- A finite set $\mathcal{V} = \{\text{the}, \text{dog}, \text{laughs}, \text{saw}, \text{barks}, \text{cat}, \ldots\}$

A sentence in the language is a sequence of words

$$x_1, x_2, \ldots x_n$$

For example

the dog barks STOP

the cat saw the dog STOP

Define $\mathcal{V}^\dagger$ to be the set of all sentences with the vocabulary $\mathcal{V}$

What is a language model?

- A probabilistic model of a sequence of words $x_1, x_2, \ldots x_n$

A language model consists of

- A finite set $\mathcal{V} = \{\text{the}, \text{dog}, \text{laughs}, \text{saw}, \text{barks}, \text{cat}, \ldots\}$
- A probability distribution over sequences of words $p(x_1, x_2, \ldots x_n)$ such that:

1. For any $\langle x_1 \ldots x_n \rangle \in \mathcal{V}^\dagger$, $p(x_1, x_2, \ldots x_n) \geq 0$
2. In addition, $\sum_{\langle x_1 \ldots x_n \rangle \in \mathcal{V}^\dagger} p(x_1, x_2, \ldots x_n) = 1$

Assign a probability to a sentence

Application of language models:

P(“I am going to school”) > P(“I are going to school”) — Grammar Checking

I had some coffee this morning.

P(“我今早喝了一些咖啡”) > P(“我今早吃了一些咖啡”) — Machine translation

P(“Can we put an elephant into the refrigerator? No, we can’t.”) > P(“Can we put an elephant into the refrigerator? Yes, we can.”) — Question Answering

N-gram language models

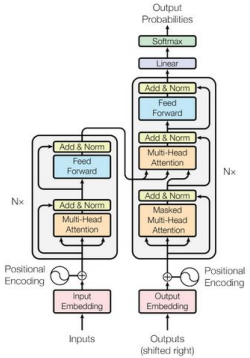
Dice, and how do we roll them
(probabilistic model)

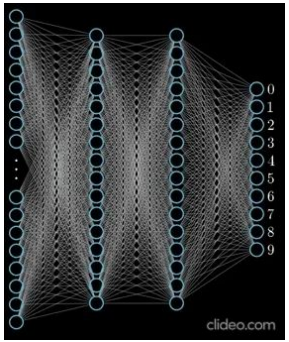
Transformers, neural networks and many others
(powerful functions)

$$p(\boldsymbol{x}) = \prod_i p(x_i|\boldsymbol{x}_{<i})$$

Learn

Parameterize





A (very bad) language model

Number of times the sentence $x_1 \dots x_n$ is seen in the training corpus

$$c(x_1 \dots x_n)$$

Total number of sentences in the training corpus N

$$p(x_1 \dots x_n) = \frac{c(x_1 \dots x_n)}{N}$$

Why this is very bad?

Markov models

Consider a sequence of random variables $X_1, X_2, \ldots, X_n$, each take any value in $\mathcal{V}$

The joint probability of a sentence is

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

Chain rule

Markov models

Consider a sequence of random variables $X_1, X_2, \ldots, X_n$, each take any value in $\mathcal{V}$

The joint probability of a sentence is

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

Chain rule

$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_{i-1} = x_{i-1})$$

First-order Markov Assumption

- Use only the recent past to predict the next word
- Reduces the number of estimated parameters in exchange for modeling capacity

Trigram language models

A trigram language model consists of a finite set $\mathcal{V}$, and a parameter $q(w \mid u, v)$

For each trigram u, v, w, such that $w \in \mathcal{V} \cup \{\text{STOP}\}$, $u, v \in \mathcal{V} \cup \{*\}$.

$q(w \mid u, v)$ can be interpreted as the probability of seeing the word w immediately after the bigram (u, v).

For any sentence $x_1 \ldots x_n$, where $x_i \in \mathcal{V}$ for $i = 1 \ldots (n-1)$, and $x_n = \text{STOP}$

$$p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i \mid x_{i-2}, x_{i-1})$$

where we define $x_0 = x_{-1} = *$

Trigram language models

For example, for the sentence:

the dog barks STOP

$$p(\text{the dog barks STOP}) = q(\text{the} \mid *, *) \times q(\text{dog} \mid *, \text{the}) \times q(\text{barks} \mid \text{the}, \text{dog}) \times q(\text{STOP} \mid \text{dog}, \text{barks})$$

Problem solved? How can we find $q(w \mid u, v)$

Parameters (of the model)

$q(w \mid u, v)$

How many parameters?

How to "estimate" them from training data?

Trigram language models

Parameters (of the model)

$$q(w \mid u, v)$$

How many parameters?

$$|\mathcal{V}|^3$$

How to "estimate" them from training data?

$$q(w \mid u, v) = \frac{c(u, v, w)}{c(u, v)}$$

$$q(\text{barks} \mid \text{the}, \text{dog}) = \frac{c(\text{the}, \text{dog}, \text{barks})}{c(\text{the}, \text{dog})}$$

Generating from a trigram language model

I am going

Trigram language model

q("going"|"I am")

going

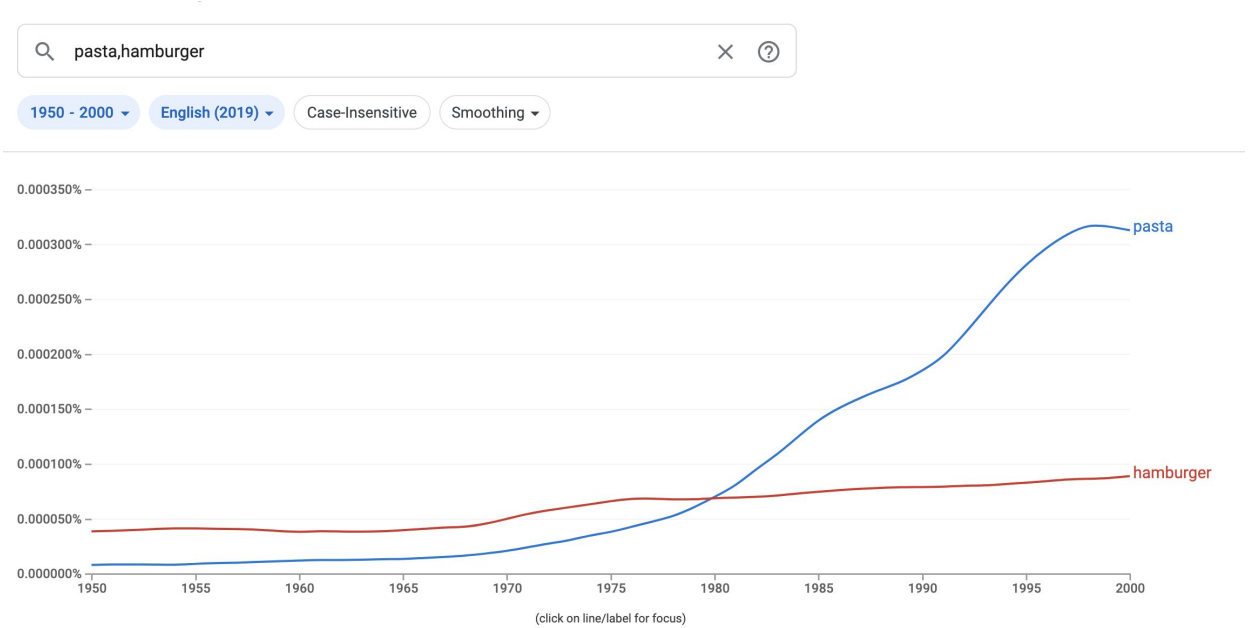
Trigram language models

How to "estimate" them from training data?

$$q(w \mid u, v) = \frac{c(u, v, w)}{c(u, v)}$$

$$q(\text{barks} \mid \text{the}, \text{dog}) = \frac{c(\text{the}, \text{dog}, \text{barks})}{c(\text{the}, \text{dog})}$$

N-gram counts!



Pasta v.s. Hamburger (Google Books Ngram Viewer)

Sparse data problems

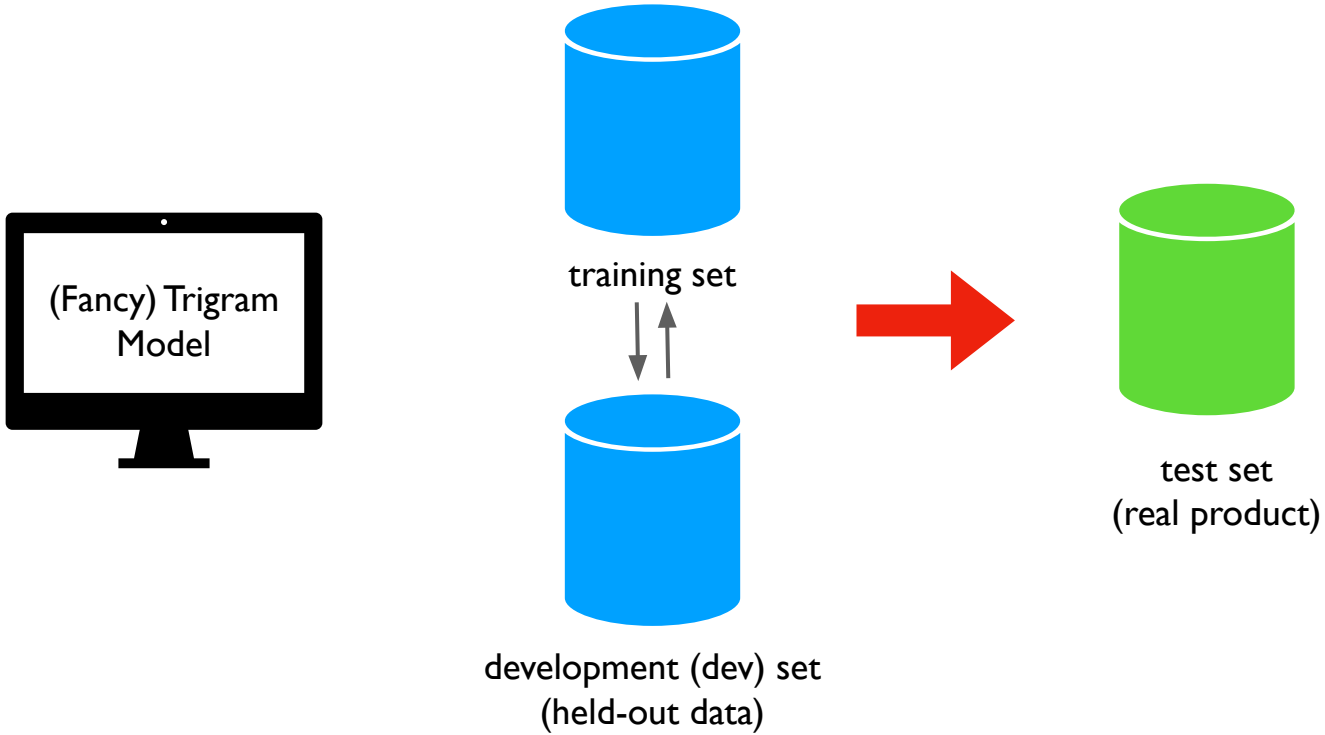
Maximum likelihood estimate:

$$q(w \mid u, v) = \frac{c(u, v, w)}{c(u, v)}$$

$$q(\text{barks} \mid \text{the}, \text{dog}) = \frac{c(\text{the}, \text{dog}, \text{barks})}{c(\text{the}, \text{dog})}$$

$|\mathcal{V}|^3$ Say vocabulary size is 20000. We have 8 *10^{12} parameters!!

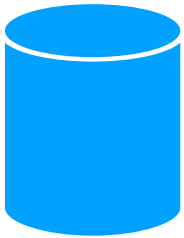
Evaluating language models



Evaluating language models

- Directly optimized for downstream applications
  - higher task accuracy → better model
- Expensive, time consuming
- Hard to optimize downstream objective (indirect feedback)

Evaluating language models: perplexity



development (dev) set
(held-out data)

. . .

$x^{(i)}$ the cat laughs STOP

$x^{(i+1)}$ the dog laughs at the cat STOP

. . .

We can compute the probability it assigns to the entire set of test sentences

$$\prod_{i=1}^{m} p(x^{(i)})$$

The higher this quantity is, the better the language model is at modeling unseen sentences.

Evaluating language models: perplexity

The higher this quantity is, the better the language model is at modeling unseen sentences.

$$\prod_{i=1}^{m} p(x^{(i)})$$

Perplexity on the test corpus is derived as a direction transformation of this.

$$\text{ppl} = 2^{-l}$$

$$l = \frac{1}{M} \sum_{i=1}^{m} \log_2 p(x^{(i)})$$

M is the total length of the sentences in the test corpus.

What if the model estimate $q(w \mid u, v) = 0$ and the trigram appears in the dataset?

Wait, why we love this number in the first place?

Let the model predicts $q(w \mid u, v) = 1/N$

$$l = \frac{1}{M} \sum_{i=1}^{m} \log_2 p(x^{(i)})$$

$$\text{ppl} = 2^{-l} = N$$

A uniform probability model — The perplexity is equal to the vocabulary size!

Perplexity can be thought of as the effective vocabulary size under the model!
For example, the perplexity of the model is 120 (even though the vocabulary size is say 10,000), then this is roughly equivalent to having an effective vocabulary of 120.

Measure of model's uncertainty about next word (aka `average branching factor')

branching factor = # of possible words following any word

Generalization of n-gram language models

- Not all n-grams in the test set will be observed in training data
- Test corpus might have some that have zero probability under our model

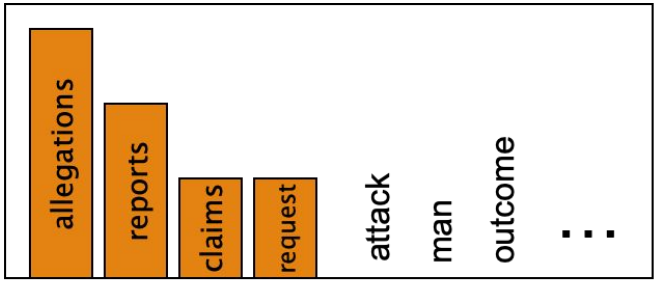
Smoothing for language models

If the model estimate $q(w \mid u, v) = 0$ and the trigram appears in the test data, ppl goes up to infinity.

When we have sparse statistics:

P(w | denied the)
- 3 allegations
- 2 reports
- 1 claims
- 1 request

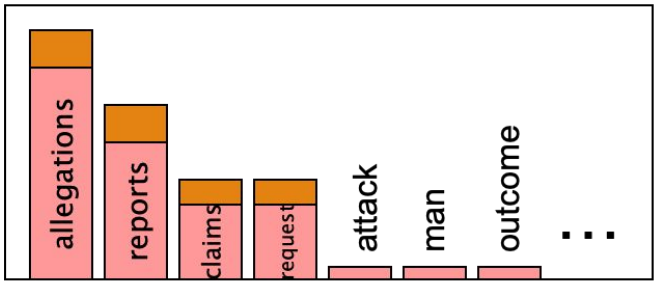
7 total



Steal probability mass to generalize better:

P(w | denied the)
- 2.5 allegations
- 1.5 reports
- 0.5 claims
- 0.5 request
- 2 other

7 total



Example from Dan Klein

Add-one (Laplace) smoothing

Considering a bigram model here, pretend we saw each word one more time than we did.

MLE estimate:

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

Add-one smoothing:

$$q_{\mathrm{Laplace}}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + |\mathcal{V}|}$$

Linear interpolation (stupid backoff)

Trigram Model, Bigram Model, Unigram Model

Trigram maximum-likelihood estimate: $$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{c(w_{i-2}, w_{i-1}, w_i)}{c(w_{i-2}, w_{i-1})}$$

Bigram maximum-likelihood estimate: $$q(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Unigram maximum-likelihood estimate: $$q(w_i) = \frac{c(w_i)}{c(\cdot)}$$

Which one suffers from the data sparsity problem the most?
Which one is more accurate?

Linear interpolation (stupid backoff)

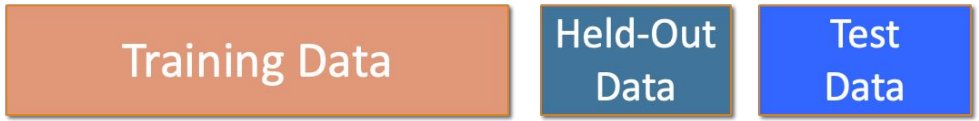
$$q(w_i \mid w_{i-2}, w_{i-1}) = \lambda_1 \times q_{\mathrm{ML}}(w_i \mid w_{i-2}, w_{i-1}) + \lambda_2 \times q_{\mathrm{ML}}(w_i \mid w_{i-1}) + \lambda_3 \times q_{\mathrm{ML}}(w_i)$$

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

How to choose the value of $\lambda_1, \lambda_2, \lambda_3$

Use the held-out corpus

Hyperparameters



maximize the probability of held-out data.

Markov models in retrospect

Consider a sequence of random variables $X_1, X_2, \ldots, X_n$, each take any value in $\mathcal{V}$

The joint probability of a sentence is

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_{i-1} = x_{i-1})$$

First-order Markov Assumption

N-gram language models

Limitations of n-gram language models

They are not sufficient to handle long-range dependencies

"Alice/Bob could not go to work that day because she/he had a doctor's appointment"

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_{i-1} = x_{i-1})$$

First-order Markov Assumption

N-gram language models

Markov models in retrospect

Consider a sequence of random variables $X_1, X_2, \ldots, X_n$, each take any value in $\mathcal{V}$

The joint probability of a sentence is

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

Is it possible to directly model this probability?

Neural network language models

Consider a sequence of random variables $X_1, X_2, \ldots, X_n$, each take any value in $\mathcal{V}$

The joint probability of a sentence is

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

Is it possible to directly model this probability?

Transformers, neural networks and many others
e.g., ChatGPT

Neural network language models

Consider a sequence of random variables $X_1, X_2, \ldots, X_n$, each take any value in $\mathcal{V}$

The joint probability of a sentence is

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

Is it possible to directly model this probability?

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-1024}, \ldots, w_{i-2}, w_{i-1})$$

Modern LMs can handle much longer contexts!

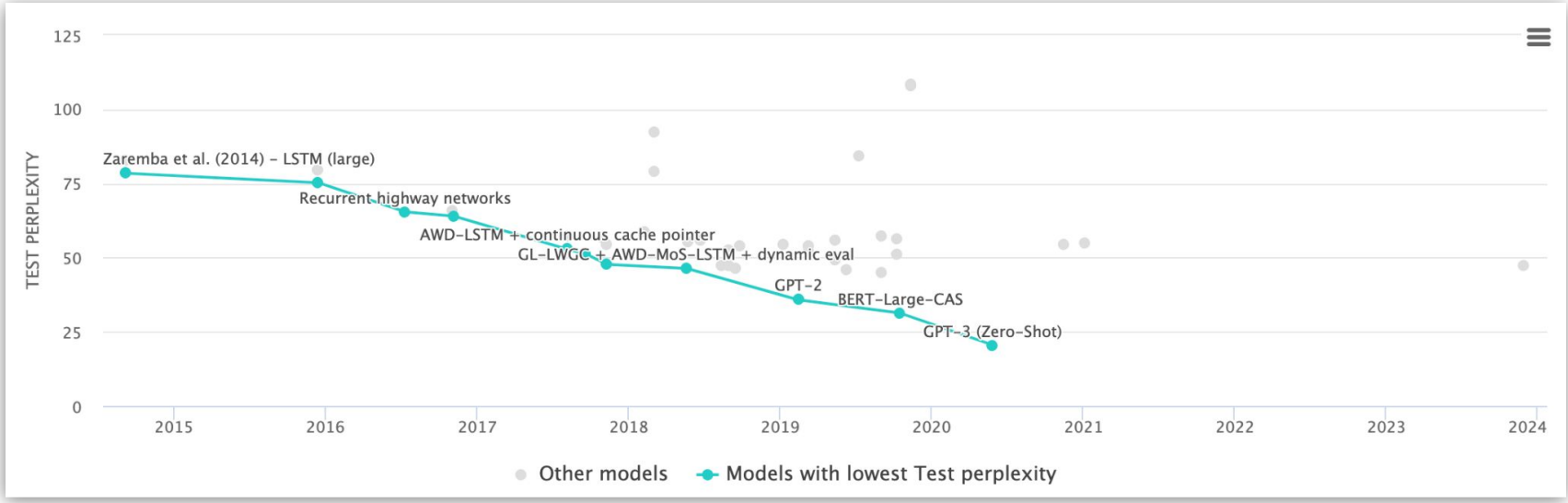
Train on a much larger corpus!

Transformers, neural networks and many others
e.g., ChatGPT

Perplexity: n-gram v.s. neural language models

Training corpus 38 million words, test corpus 1.5 million words, both **WSJ**

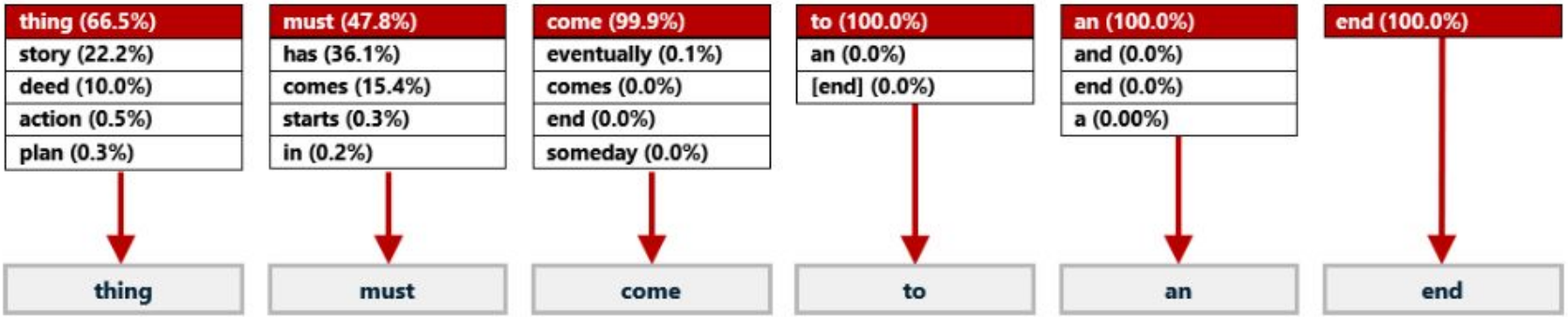
N-gram Order	Unigram	Bigram	Trigram
Perplexity (test)	962	170	109



https://paperswithcode.com/sota/language-modelling-on-penn-treebank-word

Generating from language models

- Deterministic approach: Temperature=0, always selects the word with the highest probability in each iteration

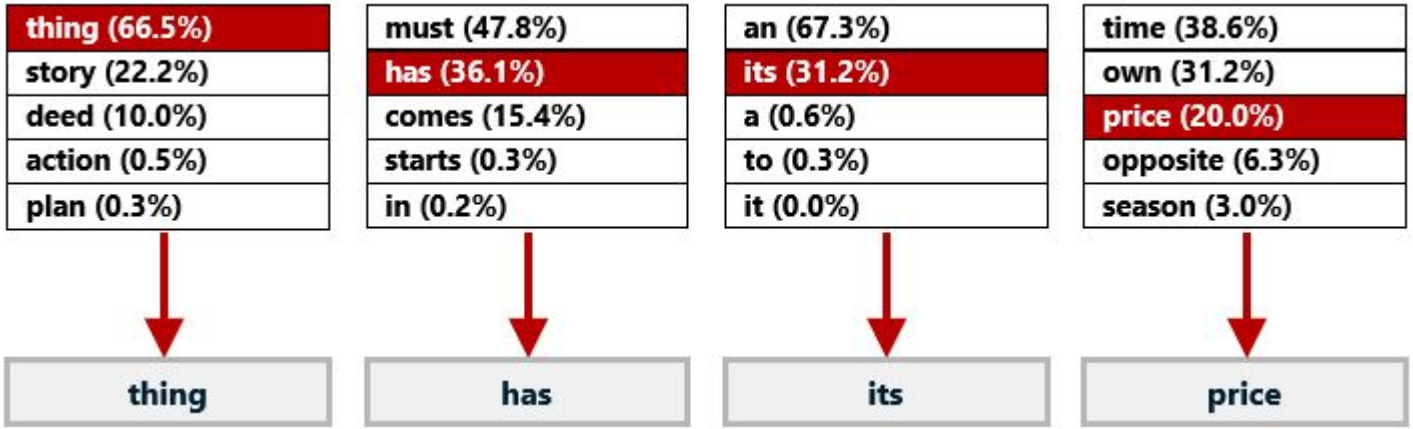


How ChatGPT completes a sentence with temperature=0

https://www.atmosera.com/ai/understanding-chatgpt/

Generating from language models

- Probabilistic or stochastic approach: e.g., temperature=0.7, the next word is chosen based on a probability distribution over the possible words. More creative!



How ChatGPT completes a sentence with temperature=0.7

https://www.atmosera.com/ai/understanding-chatgpt/