COMP 3361 Natural Language Processing

Lecture 2: Language Modeling

n-gram Language Models

Spring 2024

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

Announcements

- Join the course Slack workspace

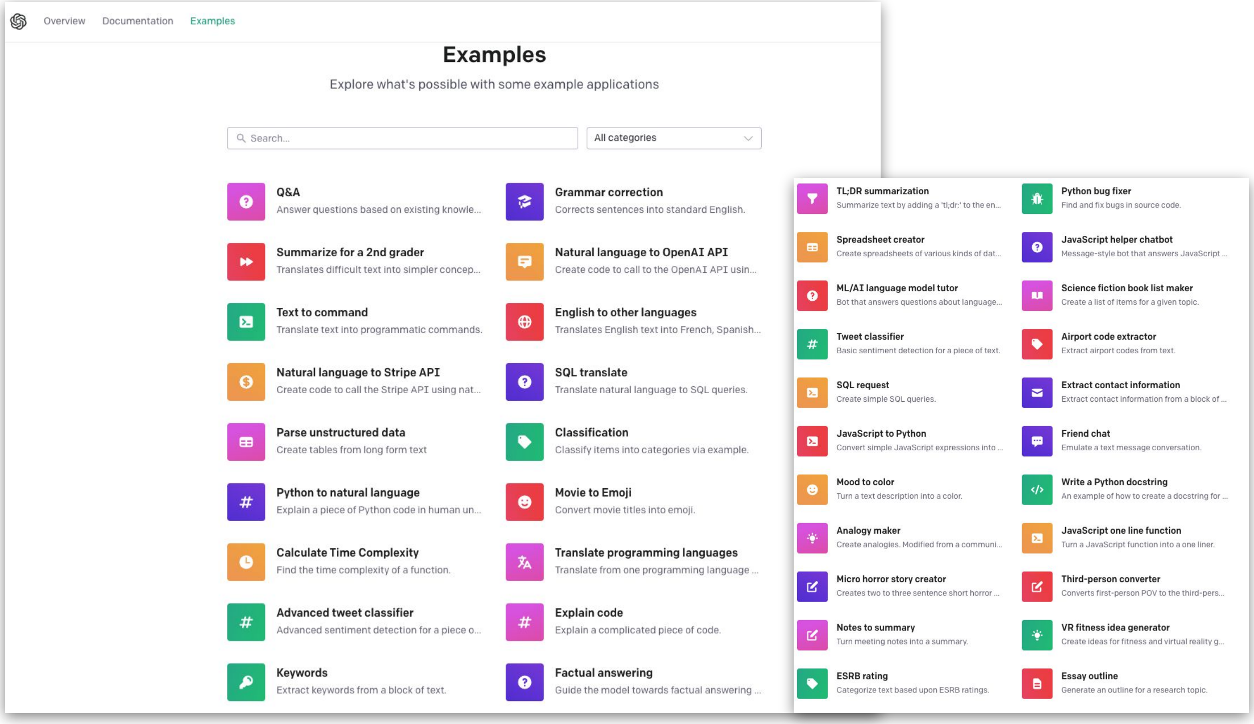
  https://join.slack.com/t/slack-fdv4728/shared_invite/zt-2asgddr0h-6wIXbRndwKhBw2IX2~ZrJQ

- Assignment 1 will be out this weekend

Lecture plan

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

ChatGPT is a powerful **language model!**



Let's play a game!

This year, I am going to do an internship in ________

Queen Mary Hospital, HSBC, Google, Amazon

Majoring in computer science, this year, I am going to do an internship in ________

Queen Mary Hospital, HSBC, Google, Amazon

ChatGPT auto-completes your prompt

Q&A

Answers | Generation | Conversation

Answer questions based on existing knowledge.

Prompt

I am a highly intelligent question answering bot. If you ask me a question that is rooted in truth, I will give you the answer. If you ask me a question that is nonsense, trickery, or has no clear answer, I will respond with "Unknown".

Q: What is human life expectancy in the United States?
A: Human life expectancy in the United States is 78 years.

Q: Who was president of the United States in 1955?
A: Dwight D. Eisenhower was president of the United States in 1955.

Q: Which party did he belong to?
A: He belonged to the Republican Party.

Q: What is the square root of banana?
A: Unknown

Q: How does a telescope work?
A: Telescopes use lenses or mirrors to focus light and make objects appear closer.

Q: Where were the 1992 Olympics held?
A: The 1992 Olympics were held in Barcelona, Spain.

Q: How many squigs are in a bonk?
A: Unknown

Q: Where is the Valley of Kings?
A:

Sample response

The Valley of Kings is located in Luxor, Egypt.

ChatGPT auto-completes your prompt

TL;DR summarization

Transformation | Generation

Summarize text by adding a 'tl;dr:' to the end of a text passage. It shows that the API understands how to perform a number of tasks with no instructions.

Prompt

A neutron star is the collapsed core of a massive supergiant star, which had a total mass of between 10 and 25 solar masses, possibly more if the star was especially metal-rich.[1] Neutron stars are the smallest and densest stellar objects, excluding black holes and hypothetical white holes, quark stars, and strange stars.[2] Neutron stars have a radius on the order of 10 kilometres (6.2 mi) and a mass of about 1.4 solar masses. [3] They result from the supernova explosion of a massive star, combined with gravitational collapse, that compresses the core past white dwarf star density to that of atomic nuclei.

Tl;dr

Sample response

A neutron star is the collapsed core of a massive supergiant star. These ultra-dense objects are incredibly fascinating due to their strange properties and their potential for phenomena such as extreme gravitational forces and a strong magnetic field.

Python bug fixer

Code | Generation

There's a number of ways of structuring the prompt for checking for bugs. Here we add a comment suggesting that source code is buggy, and then ask codex to generate a fixed code.

Prompt

```
##### Fix bugs in the below function

### Buggy Python
import Random
a = random.randint(1,12)
b = random.randint(1,12)
for i in range(10):
  question = "What is "+a+" x "+b+"? "
  answer = input(question)
  if answer = a*b
    print (Well done!)
  else:
    print("No.")

### Fixed Python
```

Sample response

```
import random
a = random.randint(1,12)
b = random.randint(1,12)
for i in range(10):
  question = "What is "+str(a)+" x "+str(b)+"? "
  answer = input(question)
  if answer == str(a*b):
    print ("Well done!")
  else:
```

Generative language model

I am going to do an internship in Google

Making the dice

bag of words

1 Belief
2 Evidence
3 Reason
4 Claim
5 Think
6 Justify
7 Also

…

99 Therefore
100 Google

Vocabulary

Generative language model

I

I

Generative language model

I am

am

Generative language model

I am going

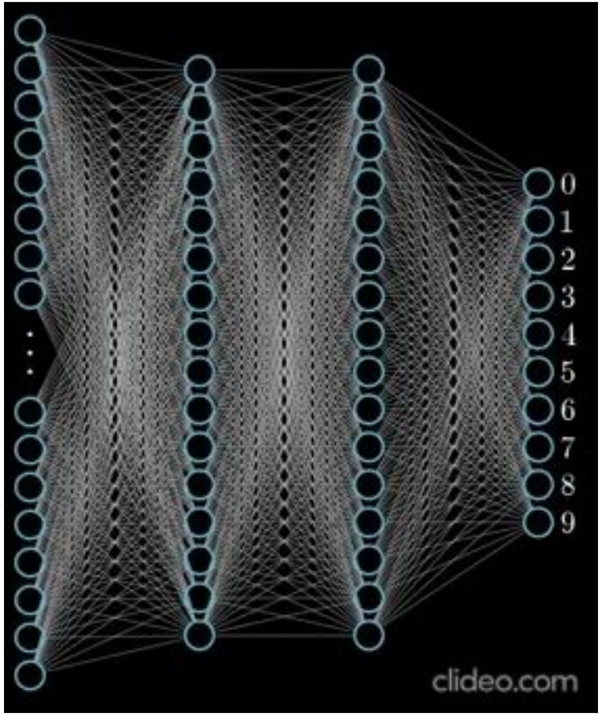
going

Generative language model

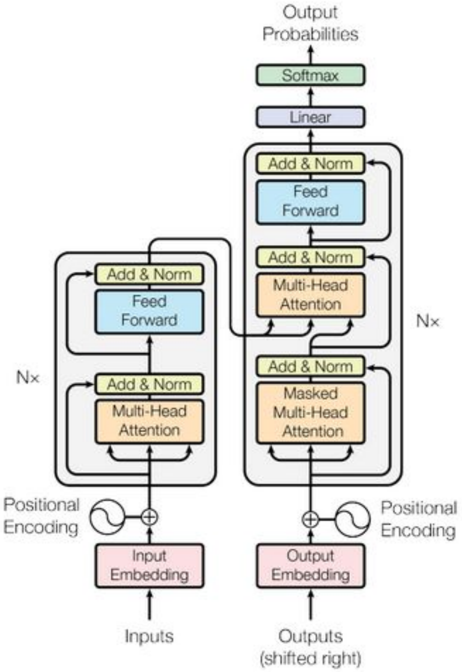
I am going to do an internship in Google

Google

Neuralize the dice!

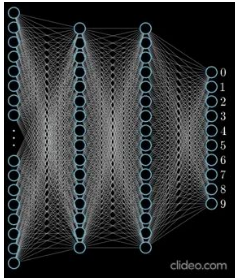


Neural Networks (e.g. Transformers)



Neural network language models

I am going to do an internship in Google



Google

Language models, and how to build it

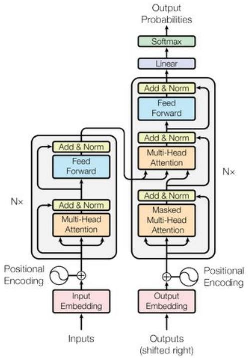
Dice, and how do we roll them
(probabilistic model)

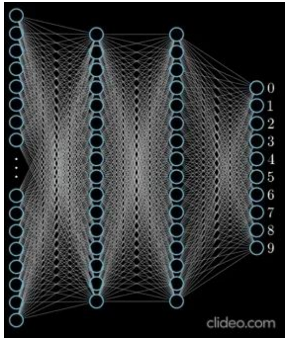
Transformers, neural networks and many others
(powerful functions)

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

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

$$\text{the dog barks STOP}$$

$$\text{the cat saw the dog STOP}$$

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") <u>Grammar Checking</u>

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") <u>Machine translation</u>

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.") <u>Question Answering</u>

N-gram language models

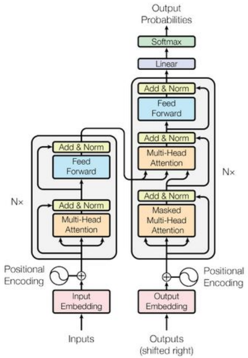
Dice, and how do we roll them
(probabilistic model)

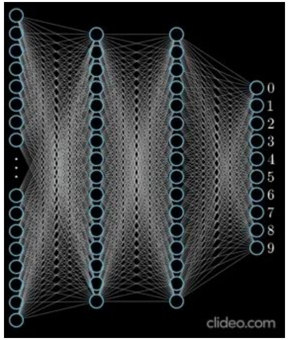
Transformers, neural networks and many others
(powerful functions)

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

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

$q(\text{"going"}|\text{"I am"})$ → going

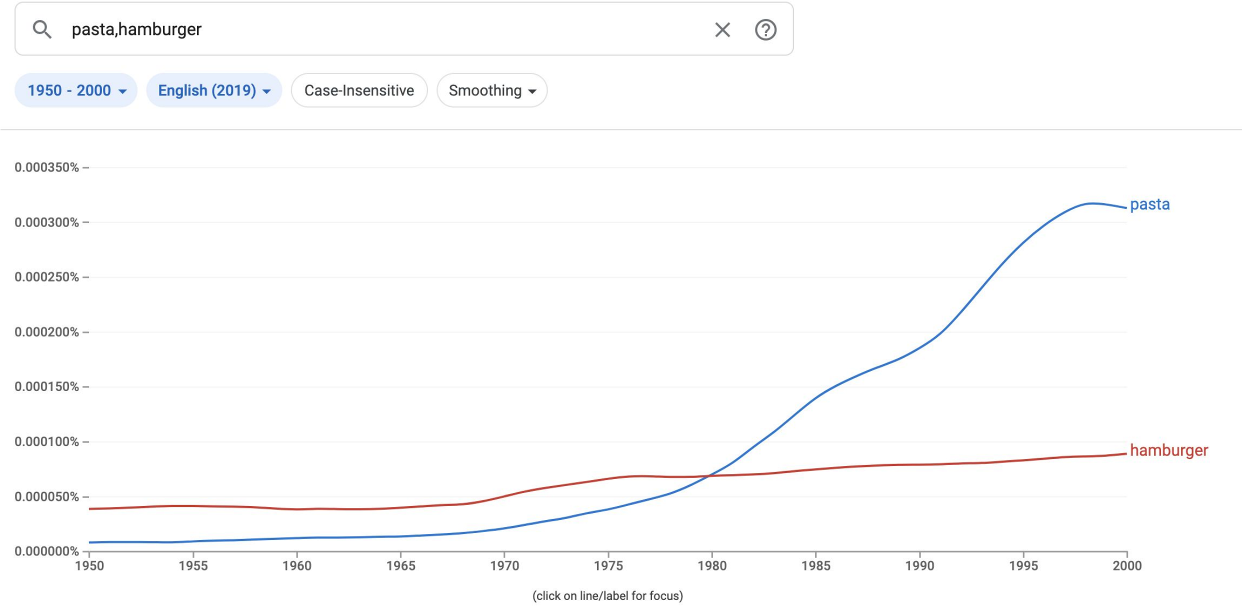
Trigram language models

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

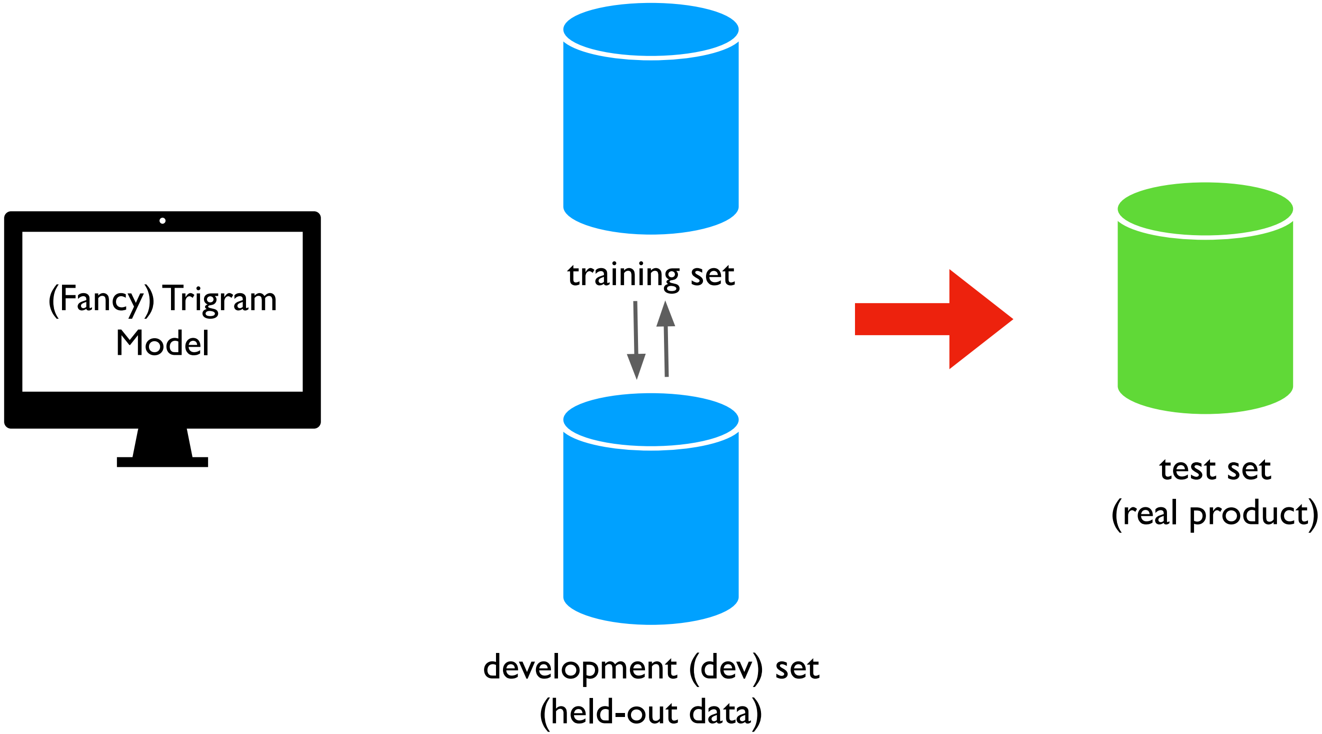
Maximum likelihood estimate:

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

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

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

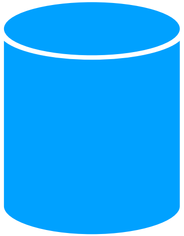
Evaluating language models



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)

$$\ldots$$

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

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

$$\ldots$$

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.

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

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

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

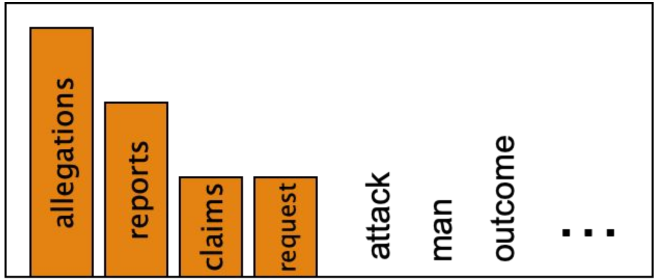
Smoothing for language models

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

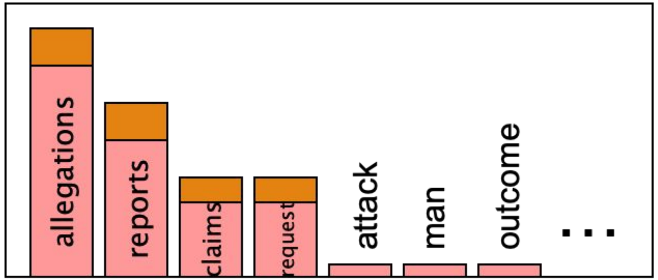
7 total



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)

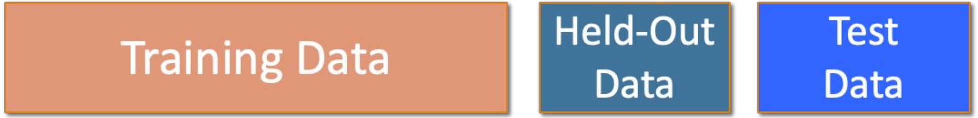
$$
\begin{aligned}
q(w_i \mid w_{i-2}, w_{i-1}) = \quad & \lambda_1 \times q_{\mathrm{ML}}(w_i \mid w_{i-2}, w_{i-1}) \\
& + \lambda_2 \times q_{\mathrm{ML}}(w_i \mid w_{i-1}) \\
& + \lambda_3 \times q_{\mathrm{ML}}(w_i)
\end{aligned}
$$

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

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

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

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

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

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"

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

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

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

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

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

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

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

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

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

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

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

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

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!

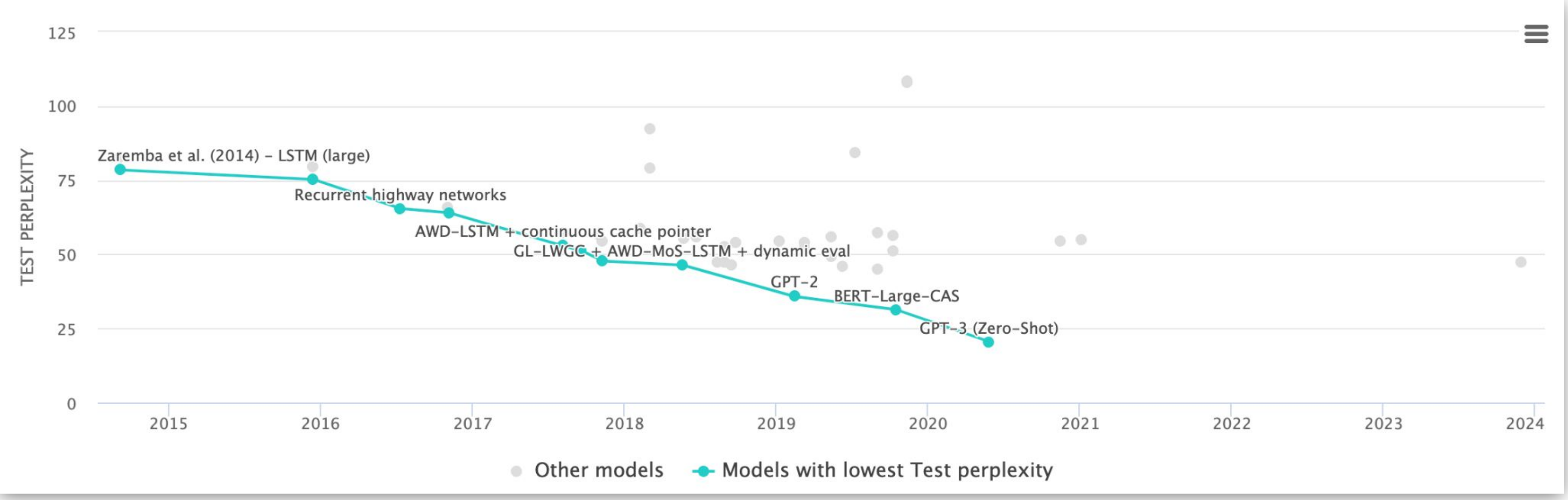
Train on a much larger corpus!

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