TTIC 31190: Natural Language Processing

Lecture 9: Language Modeling

Fall 2023

Announcements

- Freda’s office hour this week
  - Thu 1:30-2:30 pm, TTIC 4th floor open space
- TA (Jiamin Yang) Tutorial Sessions & Office Hours
  - Fridays 3 pm – 4 pm; TTIC Room 530
  - This week: HMM & CRF
  - Office hour 4 pm – 5 pm
- Assignment 2 due on Nov 2, 11:59 pm

Recap

- Neural Networks
  - Multi Layer Perceptron (MLP)
  - Convolutional neural network (CNN)
  - Recurrent neural network (RNN)
  - Transformer (Attention Is All You Need)

- Sequence Labeling (structured prediction)
  - Hidden Markov Model (HMM)
  - Conditional Random Field (CRF)

"You shall know a word by the company it keeps."

J.R. Firth, *A Synopsis of Linguistic Theory*, 1957

A bottle of tezgüino is on the table.
Everybody likes tezgüino.
Don't have tezgüino before you drive.
We make tezgüino out of corn.

Tezgüino ?

CBOW (Continuous Bag-of-Words): learn representations that predict a word given context

word2vec

A bottle of tezgüino is on the table.

A bottle of tezgüino is on the table.
Everybody likes tezgüino.
Don't have tezgüino before you drive.
We make tezgüino out of corn.

A bottle of ______ is on the table.
Everybody likes ______.
Don't have ______ before you drive.
We make ______ out of corn.

Language Modeling

Language Modeling

- The Shannon game [Shannon 1951]:

How well can you predict the next letter?

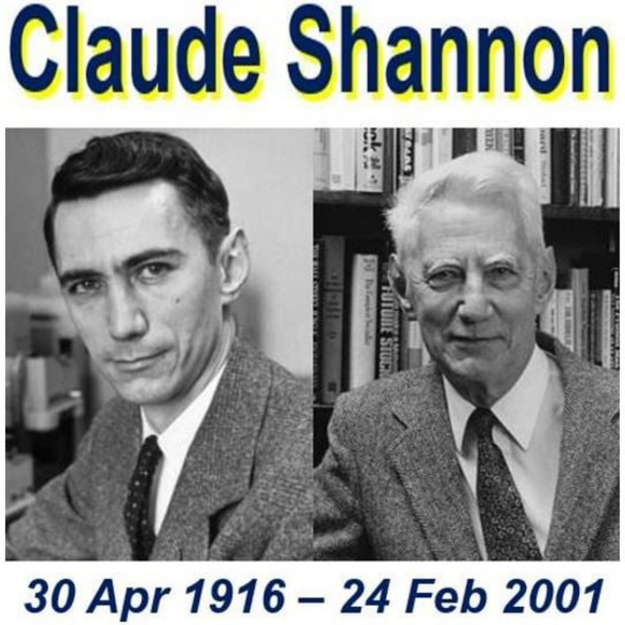
Prediction and Entropy of Printed English

By C. E. SHANNON

(*Manuscript Received Sept. 15, 1950*)

A new method of estimating the entropy and redundancy of a language is described. This method exploits the knowledge of the language statistics possessed by those who speak the language, and depends on experimental results in prediction of the next letter when the preceding text is known. Results of experiments in prediction are given, and some properties of an ideal predictor are developed.

```
(1) THE ROOM WAS NOT VERY LIGHT A SMALL OBLONG
(2) ----ROO------NOT-V-----I------SM----OBL----
(1) READING LAMP ON THE DESK SHED GLOW ON
(2) REA----------O------D----SHED-GLO--O--
(1) POLISHED WOOD BUT LESS ON THE SHABBY RED CARPET
(2) P-L-S-----O---BU--L-S--O------SH-----RE--C------
```



Language Modeling

- The Shannon game [Shannon 1951]:

How well can you predict the next letter?

Prediction and Entropy of Printed English

By C. E. SHANNON

(*Manuscript Received Sept. 15, 1950*)

A new method of estimating the entropy and redundancy of a language is described. This method exploits the knowledge of the language statistics possessed by those who speak the language, and depends on experimental results in prediction of the next letter when the preceding text is known. Results of experiments in prediction are given, and some properties of an ideal predictor are developed.

JZ

A new method of estimating the entropy and redundancy of a language is described. This method exploits the knowledge of the language statistics possessed by those who speak the language, and depends on experimental results in prediction of the next letter when the preceding text is known. Results of experiments in prediction are given, and some properties of an

Language Modeling

```
(1) THE ROOM WAS NOT VERY LIGHT A SMALL OBLONG
(2) ----ROO------NOT-V-----I------SM----OBL----
(1) READING LAMP ON THE DESK SHED GLOW ON
(2) REA----------O------D----SHED-GLO--O--
(1) POLISHED WOOD BUT LESS ON THE SHABBY RED CARPET
(2) P-L-S-----O---BU--L-S--O------SH-----RE--C------
```

JZ

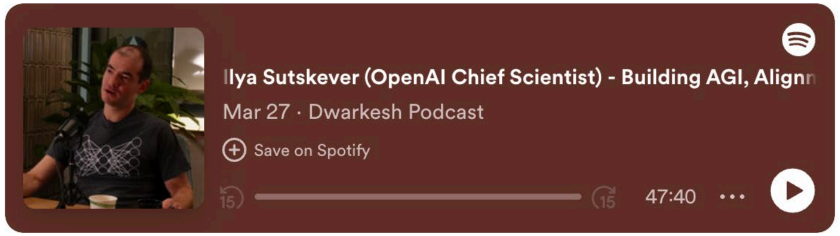
An experimental demonstration of the extent to which English is predictable can be given as follows: Select a short passage unfamiliar to the person who is to do the predicting. He is then asked to guess the first letter in the passage. If the guess is correct he is so informed, and proceeds to guess the second letter. If not, he is told the

3. Prediction of English

The new method of estimating entropy exploits the fact that anyone speaking a language possesses, implicitly, an enormous knowledge of the statistics of the language. Familiarity with the words, idioms, clichés and grammar enables him to fill in missing or incorrect letters in proof-reading, or to complete an unfinished phrase in conversation. An experimental demonstration of the extent to which English is predictable can be given as follows: Select a short passage unfamiliar to the person who is to do the predicting. He is then asked to guess the first letter in the passage. If the guess is correct he is so informed, and proceeds to guess the second letter. If not, he is told the correct first letter and proceeds to his next guess. This is continued through the text. As the experiment progresses, the subject writes down the correct text up to the current point for use in predicting future letters. The result of a typical experiment of this type is given below. Spaces were included as an additional letter, making a 27 letter alphabet. The first line is the original text; the second line contains a dash for each letter correctly guessed. In the case of incorrect guesses the correct letter is copied in the second line.

“I challenge the claim that **next-token prediction** cannot surpass human performance. On the surface, it looks like it cannot. It looks like if you just learn to imitate, to predict what people do, it means that you can only copy people. But here is a counter argument for why it might not be quite so. If your base neural net is smart enough, you just ask it — What would a person with great insight, wisdom, and capability do?”

“It's actually a much deeper question than it seems. **Predicting the next token** well means that you understand the underlying reality that led to the creation of that token. It's not statistics.”



[Src: https://www.dwarkeshpatel.com/p/ilya-sutskever#details]

Language Models

- Language Model: a probability distribution over strings in a language.

$$P(\boldsymbol{x})$$

$$\boldsymbol{x} = x_1, x_2, \ldots, x_n$$

Language Models

- Language Model: a probability distribution over strings in a language.

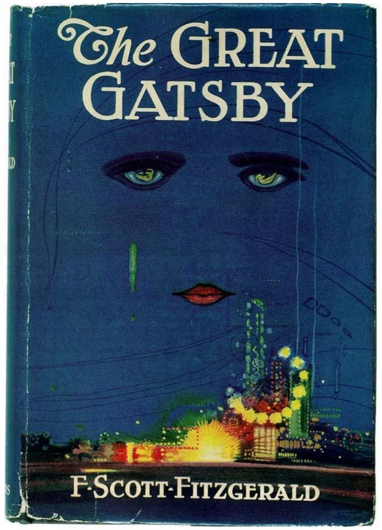
$$P(\text{I'm not a cat}) = 0.0000004$$

$$P(\text{He is hungry}) = 0.000025$$

$$P(\text{Dog the asd@sdf 1124 !?}) \approx 0$$

Language Models

- Language Model: a probability distribution over strings in a language.

$P\Big($  $\Big)$

$P\Big($  $\Big)$

Language Modeling

- Language Modeling: the task of estimating this distribution from data

  - Define a statistical model $P_\theta(\boldsymbol{x})$ with parameters θ

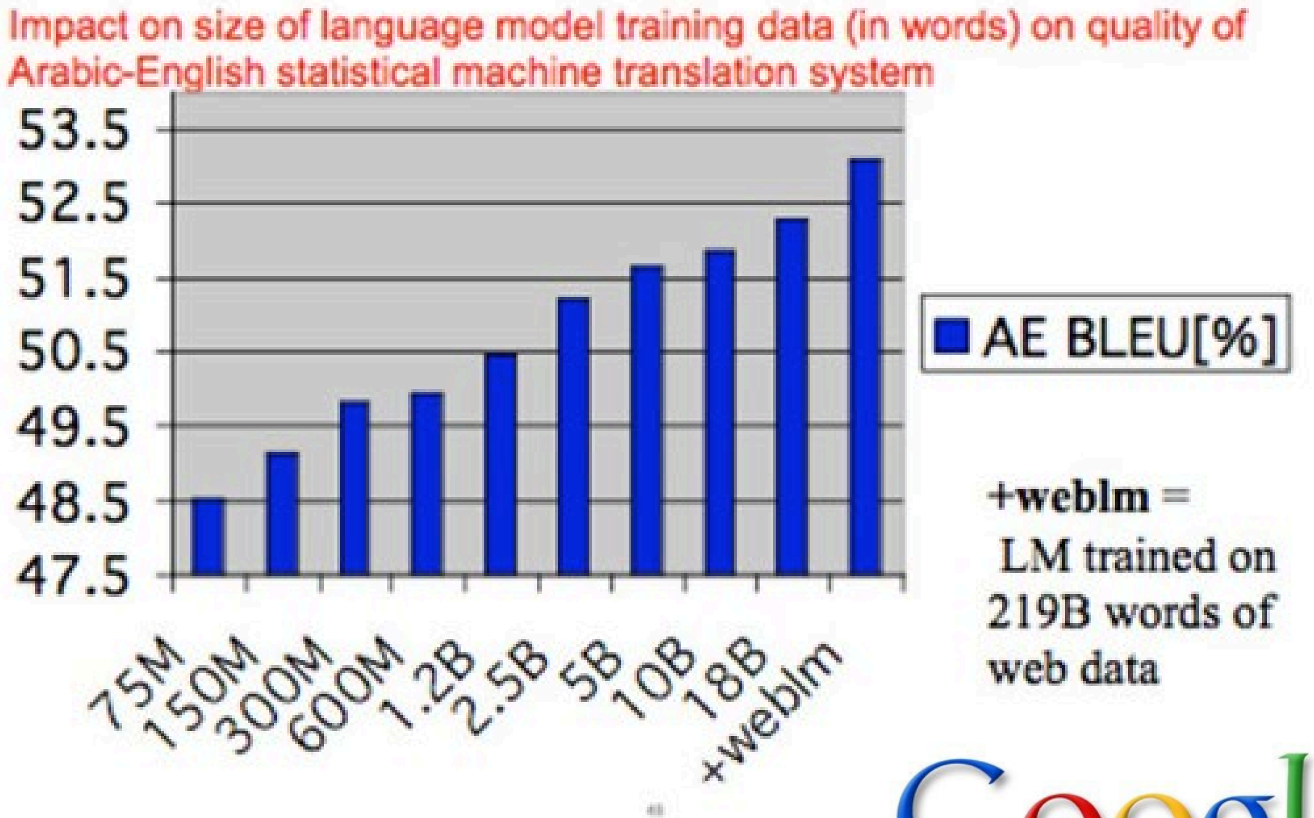
  - Maximize likelihood

$$\theta = \underset{\theta}{\operatorname{argmax}} \sum_{k=1}^{K} \log P_\theta(\boldsymbol{x}^{(k)})$$

Language Modeling

- Language Modeling: assign probabilities to token sequences
- Why?
  - machine translation:
    - *P*(*turn the camera off*) > *P*(*put the camera out*)
  - speech recognition:
    - *P*(*be back soonish*) > *P*(*be bassoon dish*)
  - spelling/grammar correction:
    - *The office is about fifteen* ***minuets*** *from my house*
    - *P*(*about fifteen* ***minutes*** *from*) > *P*(*about fifteen* ***minuets*** *from*)
  - assistive writing, dialogue systems, question answering, etc.!

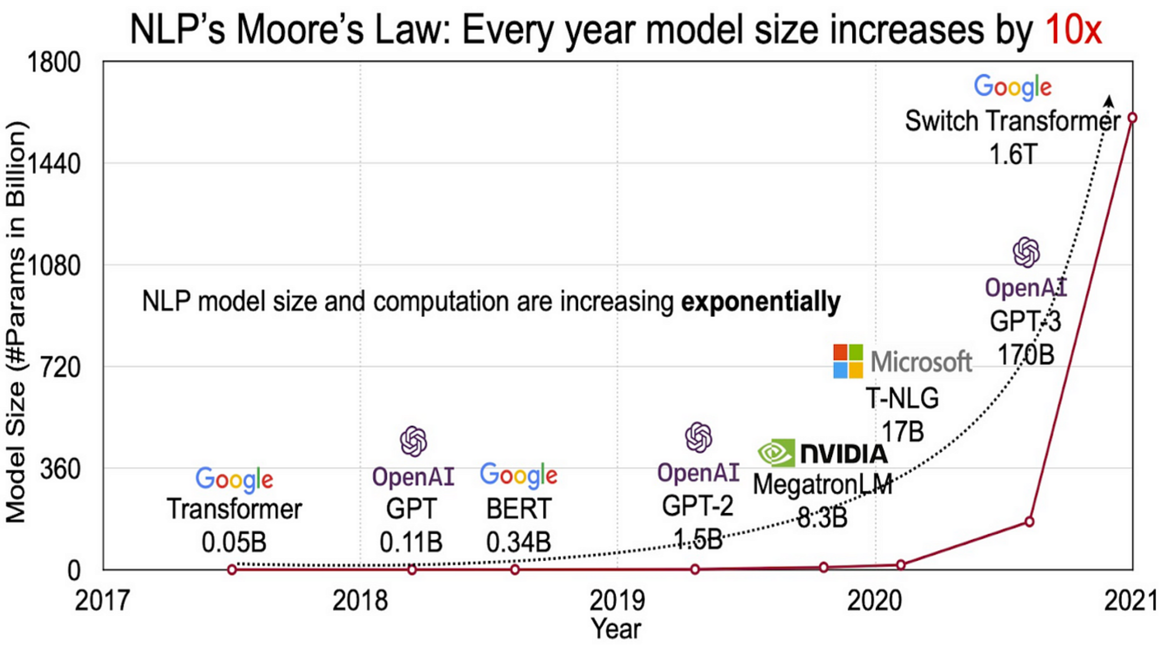
Impact of size of language model training data (in words) on quality of Arabic-English statistical machine translation system

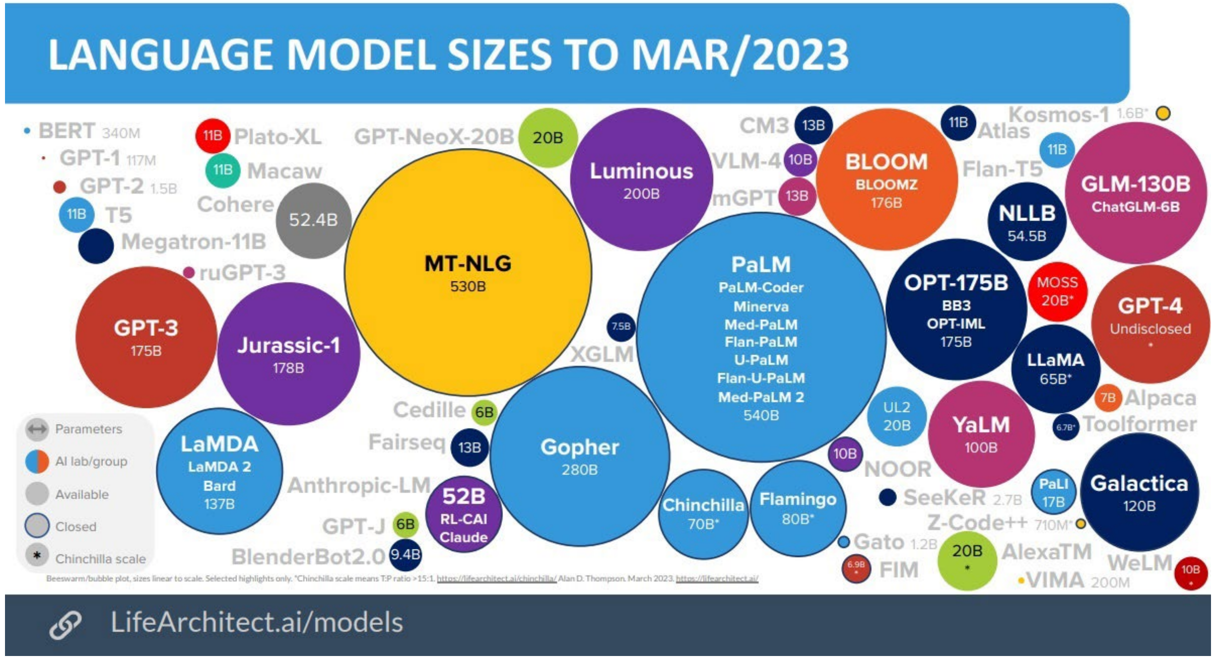


ca. 2007

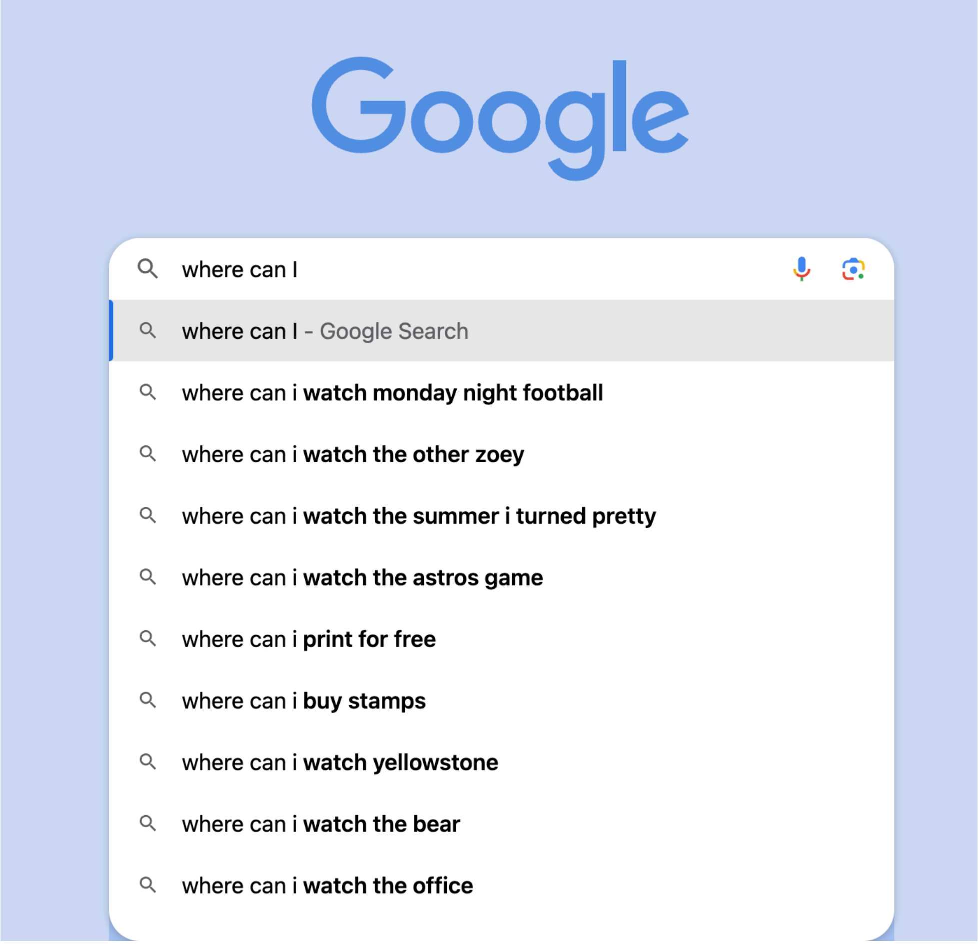
Nowadays: large language models

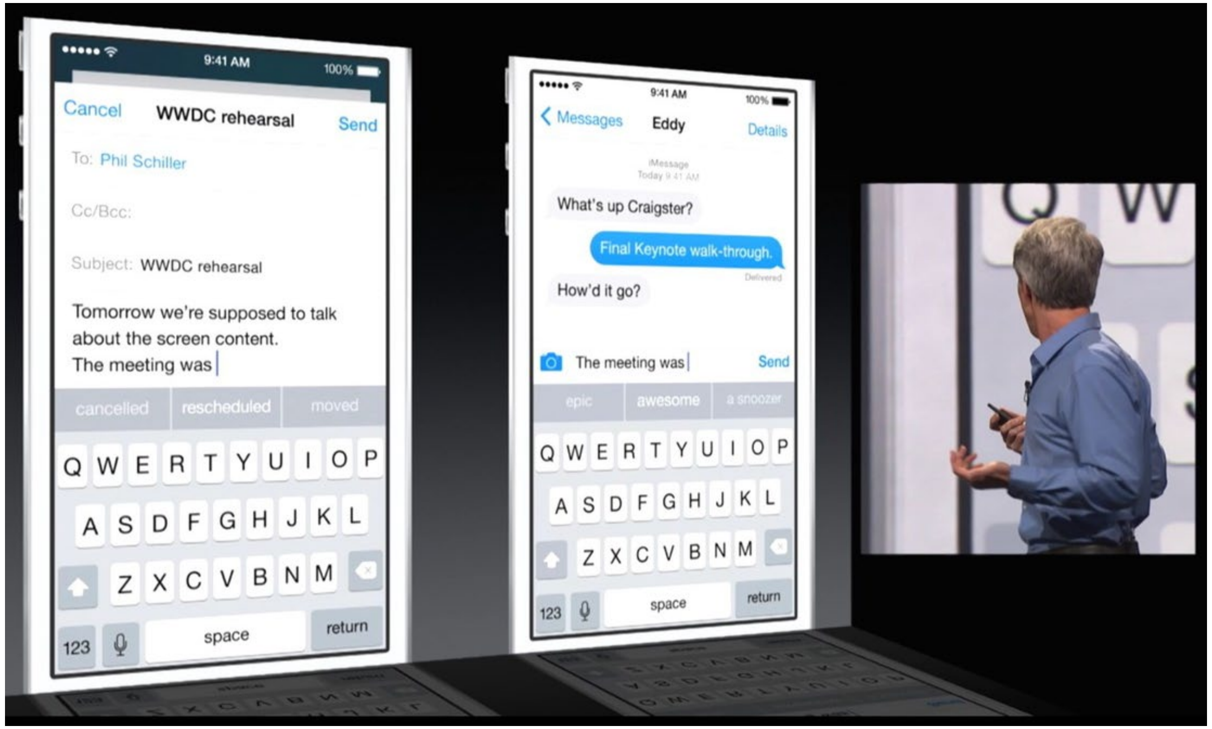
$$P(x)$$



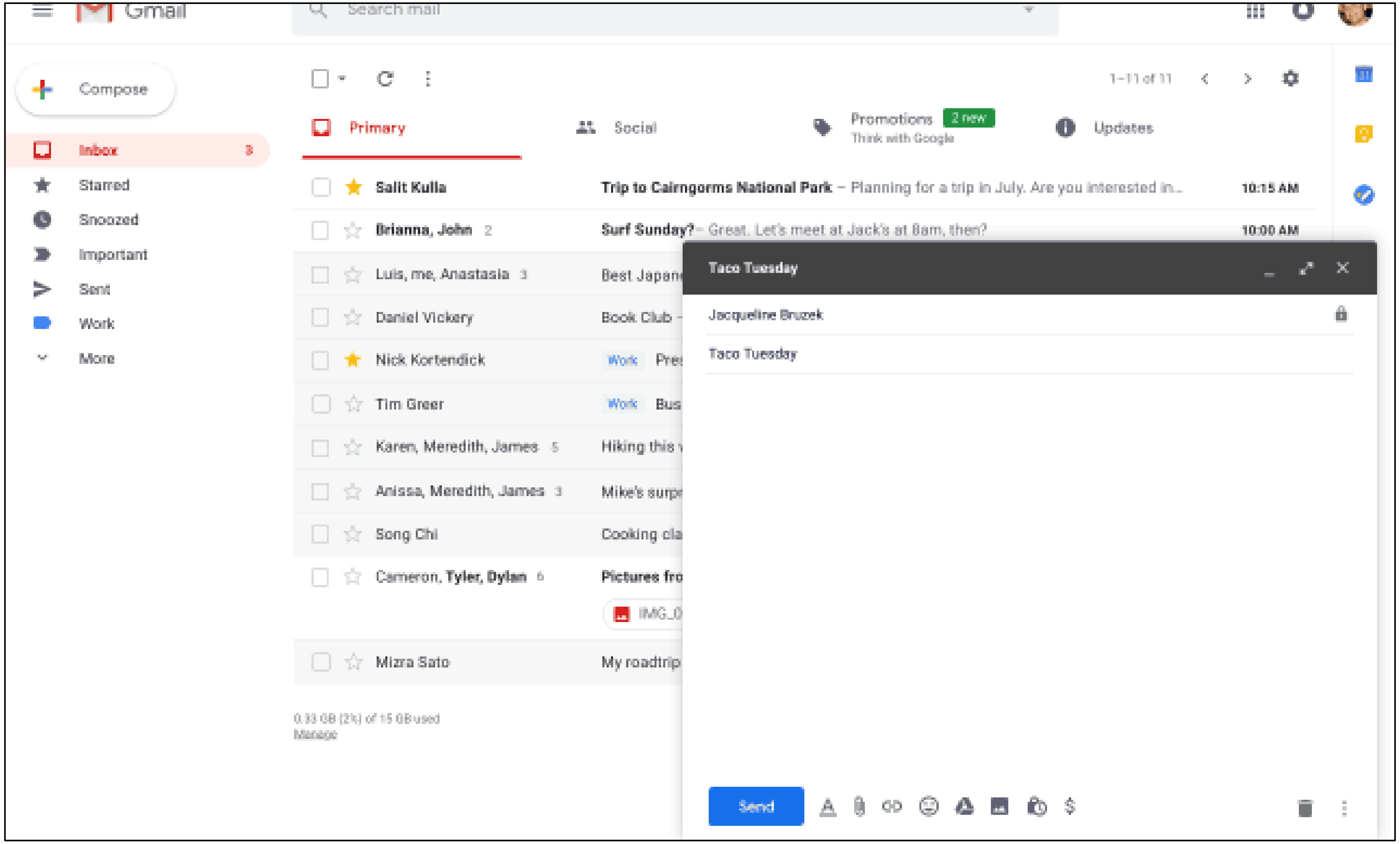


Language Models are Everywhere





Language Models are Everywhere



Language Modeling

- Goal: compute the probability of a sequence of words:

$$P(\boldsymbol{x}_{1:n}) = P(x_1, x_2, \ldots, x_n)$$

- Related task: probability of next word:

$$P(x_4 \mid x_1, x_2, x_3)$$

- A model that computes either of these:

$$P(\boldsymbol{x}_{1:n}) \quad \text{or} \quad P(x_k \mid x_1, x_2, \ldots, x_{k-1})$$

is called a **language model (LM)**

[SLP3: Chapter 3]

Language Modeling

- Building language models
- Generating from a language model
- Evaluating a language model

- Count-based language models
  - MLE estimation
  - Smoothing

- Neural language models
  - Feed-forward models
  - RNN models
  - Attention models

Language Modeling

How do we model?

$$P(\boldsymbol{x}_{1:n})$$

Chain Rule

- Chain rule of probability

$$P(B \mid A) = \frac{P(A, B)}{P(A)} \longrightarrow P(A, B) = P(A)P(B \mid A)$$

- In general to a sequence

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

Chain Rule

- Factor joint probability into product of conditional probabilities:

$$P(\boldsymbol{x}_{1:n}) = P(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} P(x_i \mid x_1, x_2, ..., x_{i-1})$$

- We have not yet made any independence assumptions

[SLP3: Chapter 3]

Chain Rule

- Factor joint probability into product of conditional probabilities:

$$P(\boldsymbol{x}_{1:n}) = P(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} P(x_i \mid x_1, x_2, ..., x_{i-1})$$

- For example, "the cat sat on the mat"

$$\begin{aligned} P(\text{the cat sat on the mat}) = P(\text{the}) * P(\text{cat}|\text{the}) * P(\text{sat}|\text{the cat}) \\ *P(\text{on}|\text{the cat sat}) * P(\text{the}|\text{the cat sat on}) \\ *P(\text{mat}|\text{the cat sat on the}) \end{aligned}$$

[SLP3: Chapter 3]

Important Detail: Modeling Length

- a language model assigns probabilities to token sequences $\boldsymbol{x}$
  - $\boldsymbol{x}$ can be any length, so the probabilities should sum to 1 across all possible sequences of all possible lengths
- usually length is modeled by including a “stop symbol” `</s>` at the end of the sequence and using “stopping probabilities”
  - a “start symbol” `<s>` is also assumed to be at the beginning
- our language model with start/stop symbols:

$$P(\boldsymbol{x}_{1:n}) = P(\texttt{</s>} \mid \texttt{<s>}, x_1, x_2, \ldots, x_n) \prod_{i=1}^{n} P(x_i \mid \texttt{<s>}, x_1, x_2, \ldots, x_{i-1})$$

Why Stopping Probabilities?

- our language model:

$$P(\boldsymbol{x}_{1:n}) = P(\texttt{</s>} \mid \texttt{<s>}, x_1, x_2, \ldots, x_n) \prod_{i=1}^{n} P(x_i \mid \texttt{<s>}, x_1, x_2, \ldots, x_{i-1})$$

- we need to ensure:

$$\sum_{n=1}^{\infty} \sum_{\boldsymbol{x}_{1:n}} P(\boldsymbol{x}_{1:n}) = 1$$

- consider removing stopping probabilities:

$$P(\boldsymbol{x}_{1:n}) = \prod_{i=1}^{n} P(x_i \mid \texttt{<s>}, x_1, x_2, \ldots, x_{i-1})$$

Without Stopping Probabilities

- without stopping probabilities, sums of probabilities for all possible length-1 and length-2 sequences:

length = 1: $$\sum_{n=1}^{1} \sum_{\boldsymbol{x}_{1:n}} P(\boldsymbol{x}_{1:n}) = \sum_{x \in \mathcal{V}} P(x \mid \texttt{<s>}) = 1$$

length = 2: $$\sum_{n=2}^{2} \sum_{\boldsymbol{x}_{1:n}} P(\boldsymbol{x}_{1:n}) = \sum_{x' \in \mathcal{V}} \sum_{x \in \mathcal{V}} P(x \mid \texttt{<s>}, x') P(x' \mid \texttt{<s>}) = 1$$

- uh oh…

$$\sum_{n=1}^{2} \sum_{\boldsymbol{x}_{1:n}} P(\boldsymbol{x}_{1:n}) = 1 + 1 = 2$$

With Stopping Probabilities

- With the stop symbol

$$\sum_{n=1}^{\inf} \sum_{\boldsymbol{x}_{1:n}, </s>} P(\boldsymbol{x}_{1:n}, < /\mathrm{s} >)$$

- Signal to stop during generation
  - E.g. machine translation, automatic summarization

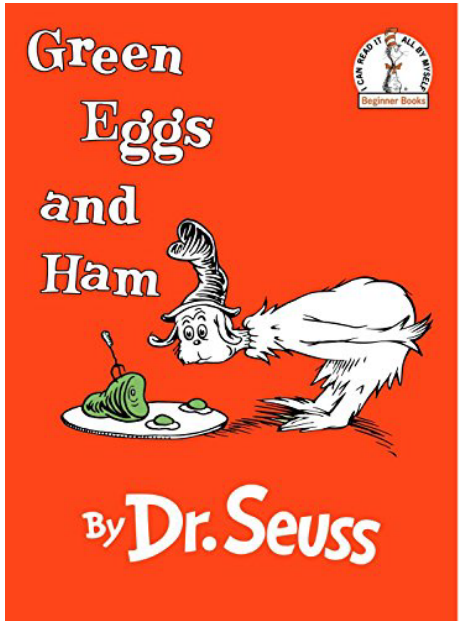
Other Ways of Modeling Length

- alternatively, we can model the length n explicitly (e.g., using a zero-truncated Poisson distribution):

$$P(\boldsymbol{x}_{1:n}) = P(n) \prod_{i=1}^{n} P(x_i \mid \texttt{<s>}, x_1, \ldots, x_{i-1})$$

Estimating Language Model Probabilities

<s> *I do not like green eggs and* *ham* </s>



$$P(\text{ham} \mid \texttt{<s>}\text{I do not like green eggs and})$$

- let's use maximum likelihood estimation (MLE):

$$P(\text{ham} \mid \texttt{<s>}\text{I do not like green eggs and}) = \frac{\text{count}(\texttt{<s>}\text{I do not like green eggs and ham})}{\text{count}(\texttt{<s>}\text{I do not like green eggs and})}$$

- problem: we'll never have enough data!

Estimating Language Model Probabilities

- Suppose we have a vocabulary of size V, how many sequences of length n do we have?

A) $n * V$

B) n^V

C) $\boxed{V^n}$

D) V/n

Typical English vocabulary ~ 40k words

Even sentences of length <= 11 results in more than 4 * 10^50 sequences. Too many to count! (# of atoms in the earth ~ 10^50)

Markov Assumption

- Independence assumption: the next word only depends on the most recent past
  - Reduces the number of estimated parameters in exchange for modeling capacity

Most recent k words

Andrey Markov

Markov Assumption

- Independence assumption: the next word only depends on the most recent past

Most recent k words

$$P(x_i \mid \texttt{<s>}, x_1, \ldots, x_{i-2}, x_{i-1}) \approx P(x_i \mid x_{i-k}, \ldots, x_{i-2}, x_{i-1})$$

1st order Markov: k = 1

$$P(\text{mat}|\text{the cat sat on the}) \approx P(\text{mat}|\text{the})$$

2nd order Markov: k = 2

$$P(\text{mat}|\text{the cat sat on the}) \approx P(\text{mat}|\text{on the})$$

$$P(\text{ham} \mid \texttt{<s>}\text{I do not like green eggs and}) \approx P(\text{ham} \mid \text{eggs and})$$

n-gram Language Models

- $n = 1$: **unigram language model**

$$P(\mathrm{I})P(\mathrm{do})P(\mathrm{not})P(\mathrm{like})P(\mathrm{green})P(\mathrm{eggs})\cdots$$

- $n = 2$: **bigram language model**

$$P(\mathrm{I} \mid \texttt{<s>})P(\mathrm{do} \mid \mathrm{I})P(\mathrm{not} \mid \mathrm{do})P(\mathrm{like} \mid \mathrm{not})\cdots P(\mathrm{ham} \mid \mathrm{and})P(\texttt{</s>} \mid \mathrm{ham})$$

- $n = 3$: **trigram language model**

$$P(\mathrm{I} \mid \texttt{<s>}\ \texttt{<s>})P(\mathrm{do} \mid \texttt{<s>}\ \mathrm{I})P(\mathrm{not} \mid \mathrm{I\ do})\cdots P(\mathrm{ham} \mid \mathrm{eggs\ and})P(\texttt{</s>} \mid \mathrm{and\ ham})$$

n-gram Language Models

- **unigram language model**

- Example sentences generated by a unigram model trained on financial news:

fifth an of futures the an incorporated a a the inflation most dollars quarter in is mass

thrift did eighty said hard 'm july bullish

that or limited the

n-gram Language Models

- **bigram language model**

- Example sentences generated by a bigram model trained on financial news:

texaco rose one in this issue is pursuing growth in a boiler house said mr. gurria

mexico 's motion control proposal without permission from five hundred fifty five yen

outside new car parking lot of the agreement reached

this would be a record november

n-gram Language Models

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

Figure 3.4 Eight sentences randomly generated from four n-grams computed from Shakespeare's works. All characters were mapped to lower-case and punctuation marks were treated as words. Output is hand-corrected for capitalization to improve readability.

[SLP3: Chapter 3]

Estimating Bigram Probabilities

- maximum likelihood estimate (MLE)

$$P(x \mid x') = \frac{\text{count}(x', x)}{\text{count}(x')}$$

[SLP3: Chapter 3]

An Example

training data:

<s> *I am Sam* **</s>**

<s> *Sam I am* **</s>**

<s> *I do not like green eggs and ham* **</s>**

MLE estimator:

$$P(x \mid x') = \frac{\text{count}(x', x)}{\text{count}(x')}$$

a few estimated bigram probabilities:

$$P(\text{I} \mid \texttt{<s>}) = \qquad P(\text{am} \mid \text{I}) = \qquad P(\text{Sam} \mid \text{am}) =$$

$$P(\text{Sam} \mid \texttt{<s>}) = \qquad P(\text{do} \mid \text{I}) = \qquad P(\texttt{</s>} \mid \text{am}) =$$

[SLP3: Chapter 3]

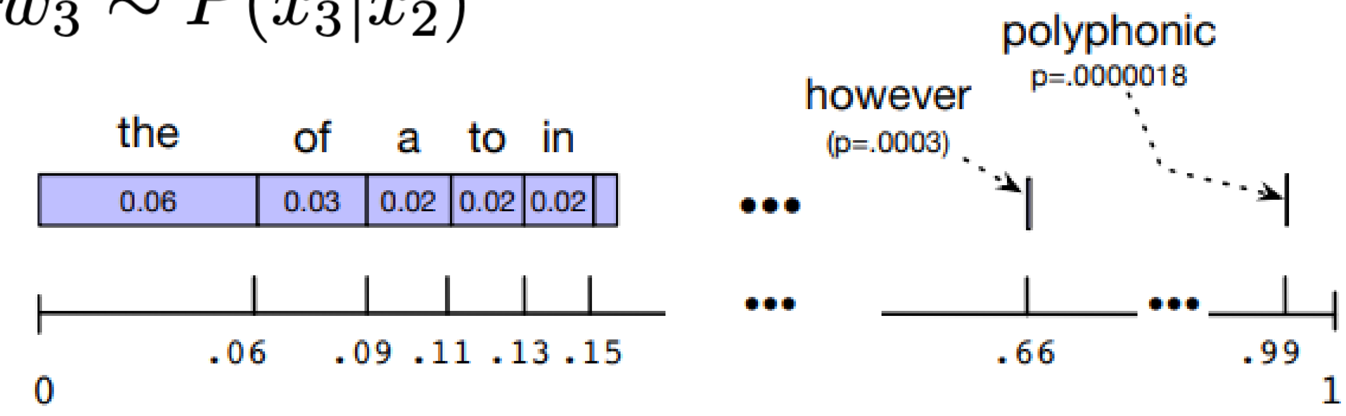
Generating from a Language Model

Generating from a Language Model

- Bigram model

- Generate the first word $w_1 \sim P(x_1| < \mathrm{s} >)$
- Generate the second word $w_2 \sim P(x_2|x_1)$
- Generate the third word $w_3 \sim P(x_3|x_2)$
- …

Sampling



[SLP3: Chapter 3]

Generating from a Language Model

- Trigram model

- Generate the first word $w_1 \sim P(x_1 | < \mathrm{s} >)$
- Generate the second word $w_2 \sim P(x_2 | < \mathrm{s} >, x_1)$
- Generate the third word $w_3 \sim P(x_3 | x_1, x_2)$
- …

Generating from a Language Model

Unigram	*release millions See ABC accurate President of Donald Will cheat them a CNN megynkelly experience @ these word out- the*
Bigram	*Thank you believe that @ ABC news, Mississippi tonight and the false editorial I think the great people Bill Clinton . "*
Trigram	*We are going to MAKE AMERICA GREAT AGAIN! #MakeAmericaGreatAgain https: //t.co/DjkdAzT3WV*

- Typical LMs are not sufficient to handle long-range dependencies

"The computer(s) that I just put into the machine room on the fifth floor is (are) crashing."

Generating from a Language Model

- GPT-4 generations

Prefix / Prompt

JZ

An experimental demonstration of the extent to which English is predictable can be given as follows: Select a short passage unfamiliar to the person who is to do the predicting. He is then asked to guess the first letter in the passage. If the guess is correct he is so informed, and proceeds to guess the second letter. If not, he is told the

correct letter and proceeds to guess the next one, and so on. After the passage is completed, the proportion of correct guesses is noted.

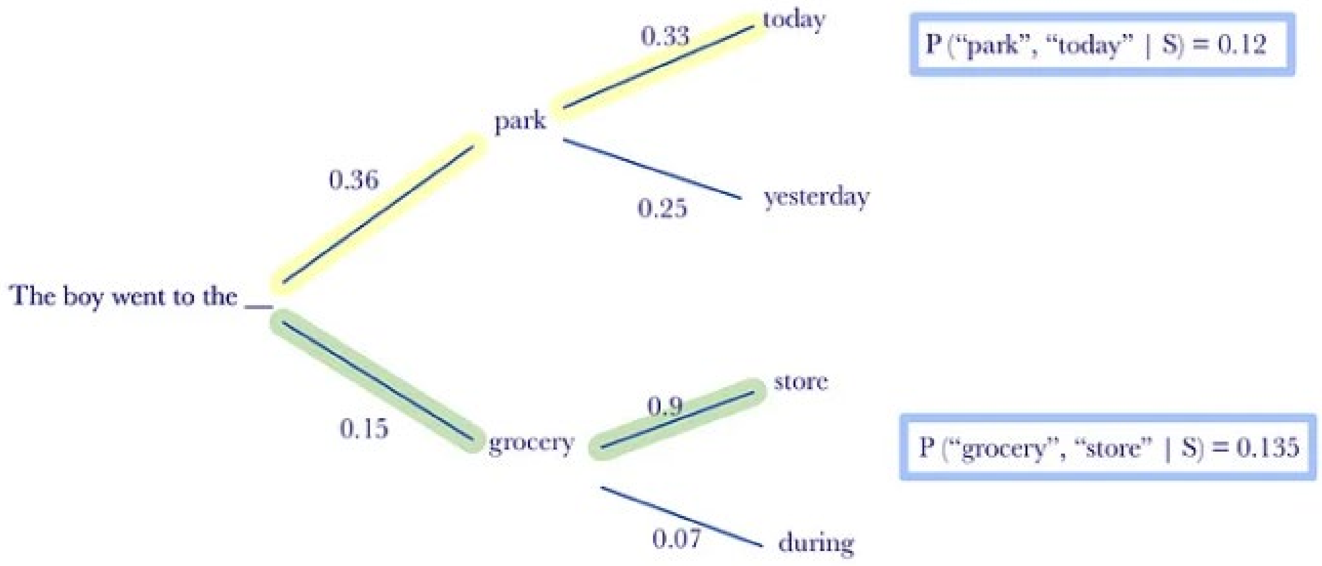
Modern language models can take much longer context!

Generating from a Language Model (more)

- Greedy search: choose the most likely word at every step

  To predict the next word given the previous two words w_1, w_2 :

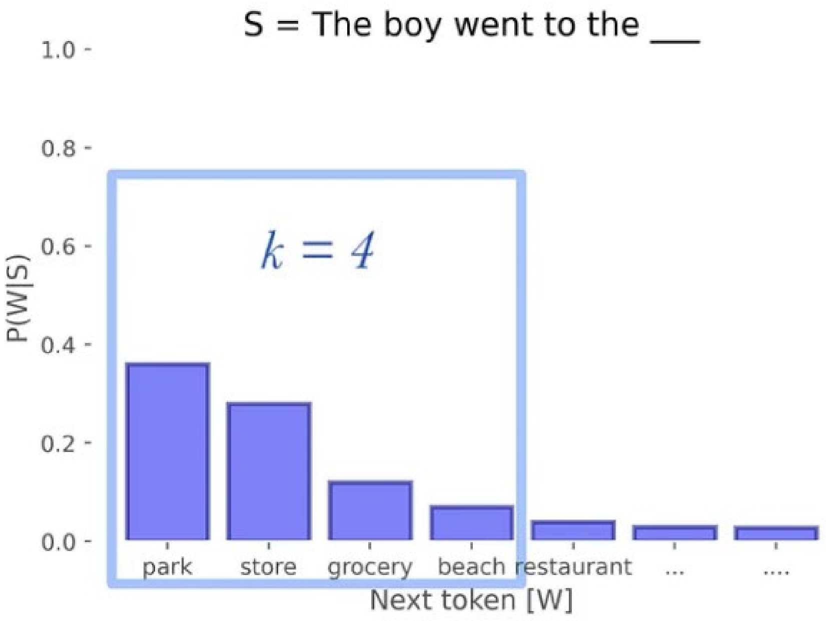
$$w_3 = \arg\max_{w \in V} P(w \mid w_1, w_2)$$



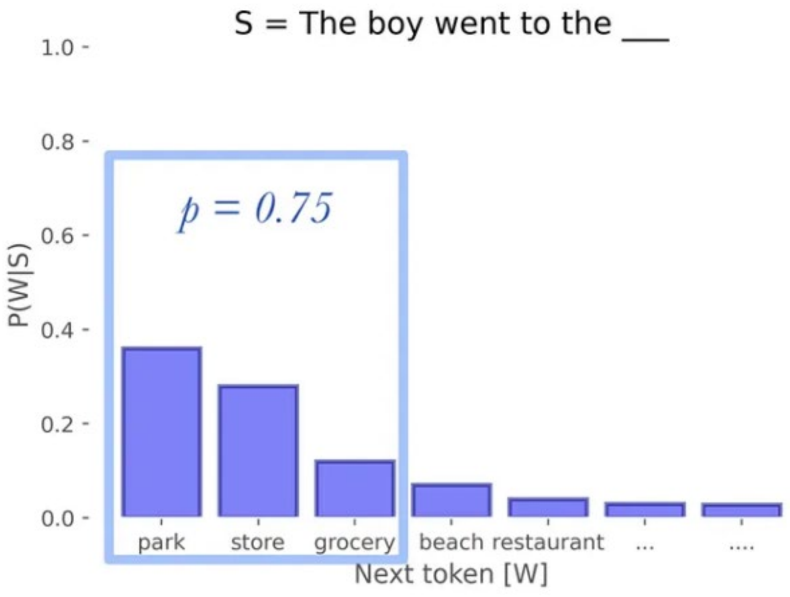
[src: https://blog.allenai.org/a-guide-to-language-model-sampling-in-allennlp-3b1239274bc3]

Generating from a Language Model (more)

- Top-k vs. top-p sampling



Top-k sampling



Top-p sampling

[src: https://blog.allenai.org/a-guide-to-language-model-sampling-in-allennlp-3b1239274bc3]

Evaluating Language Models

Evaluating Language Models

Extrinsic (task-based) **evaluation**

- use language model in a system for some task, see if performance improves
- downsides:
  - can be time-consuming depending on task/system
  - changing the language model might require changing how it's used in the system in order to improve performance

Evaluating Language Models

Intrinsic evaluation

- compute probability of held-out data
- standard metric: **perplexity**
- downside:
  - may not correlate with system performance on downstream tasks

Probability of Held-out Data

- probability of held-out sentences:

$$\prod_i P(\boldsymbol{x}^{(i)})$$

- let's work with log-probabilities:

$$\log_2 \prod_i P(\boldsymbol{x}^{(i)}) = \sum_i \log_2 P(\boldsymbol{x}^{(i)})$$

- divide by number of words *M* (including stop symbols) in held-out sentences:

$$\frac{1}{M} \sum_i \log_2 P(\boldsymbol{x}^{(i)})$$

Probability → Perplexity

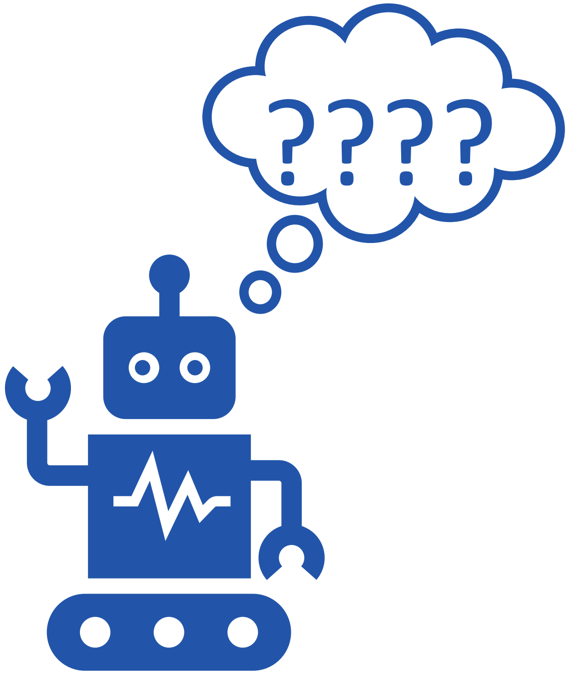
- average token log-probability of held-out data:

$$\ell = \frac{1}{M} \sum_i \log_2 P(\boldsymbol{x}^{(i)})$$

- **perplexity:**

$$\text{ppl} = 2^{\boxed{-\ell}}$$ Cross entropy

- the lower the perplexity, the better the model



Perplexity (PPL)

- Measure how well a language model (LM) predicts the true data

$$\boxed{\text{Perplexity} = P(w_1, w_2, \ldots, w_n)^{-1/n}}$$

- What is the intuition behind it?

Perplexity as Branching Factor

- given a vocabulary $\mathcal{V}$, consider this bigram language model:

$$\forall u, v, P(u \mid v) = \frac{1}{N} \qquad N = |\mathcal{V} \cup \{\texttt{</s>}\}|$$

- perplexity of any sequence under this model?

$$\ell = \frac{1}{M} \log_2 P(x_1, x_2, ..., x_{M-1}, \texttt{</s>})$$

$$= \frac{1}{M} \log_2 \prod_{i=1}^{M} \frac{1}{N}$$

$$= \frac{1}{M} \log_2 \left(\frac{1}{N}\right)^M = \log_2 \left(\frac{1}{N}\right)$$

$$\mathrm{ppl} = 2^{-\ell} = N$$

Perplexity Example

- train: 38 million tokens (Wall Street Journal text)
- test: 1.5 million tokens
- vocabulary size: 19,979

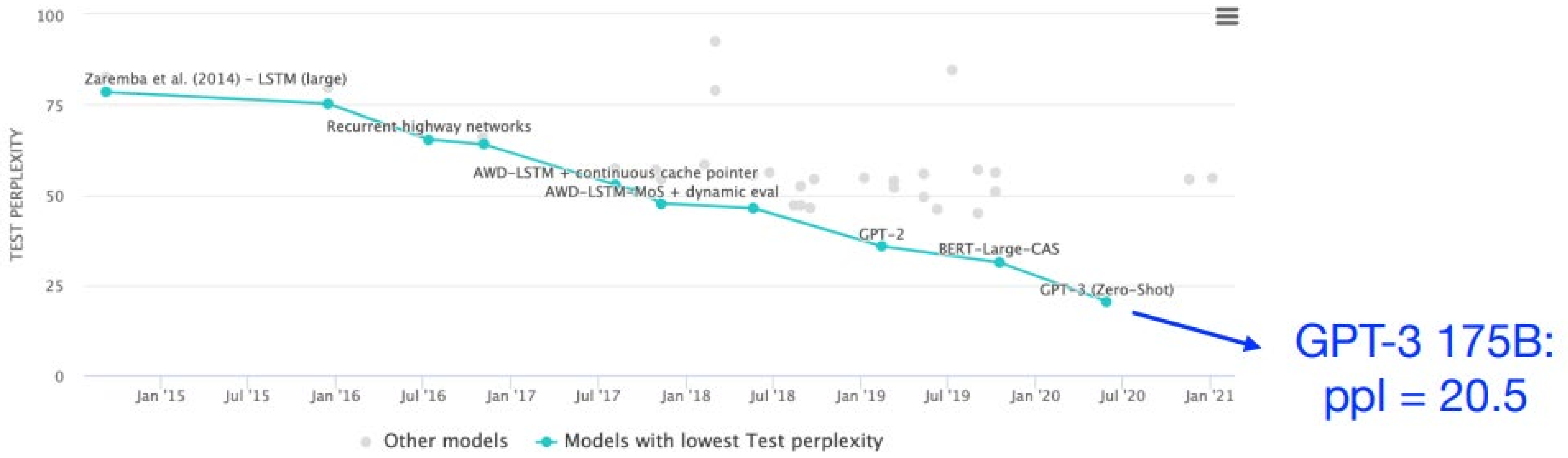
***n*-gram order:**	**unigram**	**bigram**	**Trigram**
perplexity:	962	170	109

- though vocabulary size is ~20K, trigram model is (roughly) considering 109 choices per position on average

[SLP3: Chapter 3]

Perplexity Example

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



[Src: https://paperswithcode.com/sota/language-modelling-on-penn-treebank-word] training data:

<s> *I am Sam* </s>

<s> *Sam I am* </s>

<s> *I do not like green eggs and ham* </s>

test data:

<s> *I like green eggs and ham* </s>

problem: $P(\text{like} \mid \text{I}) = 0$!

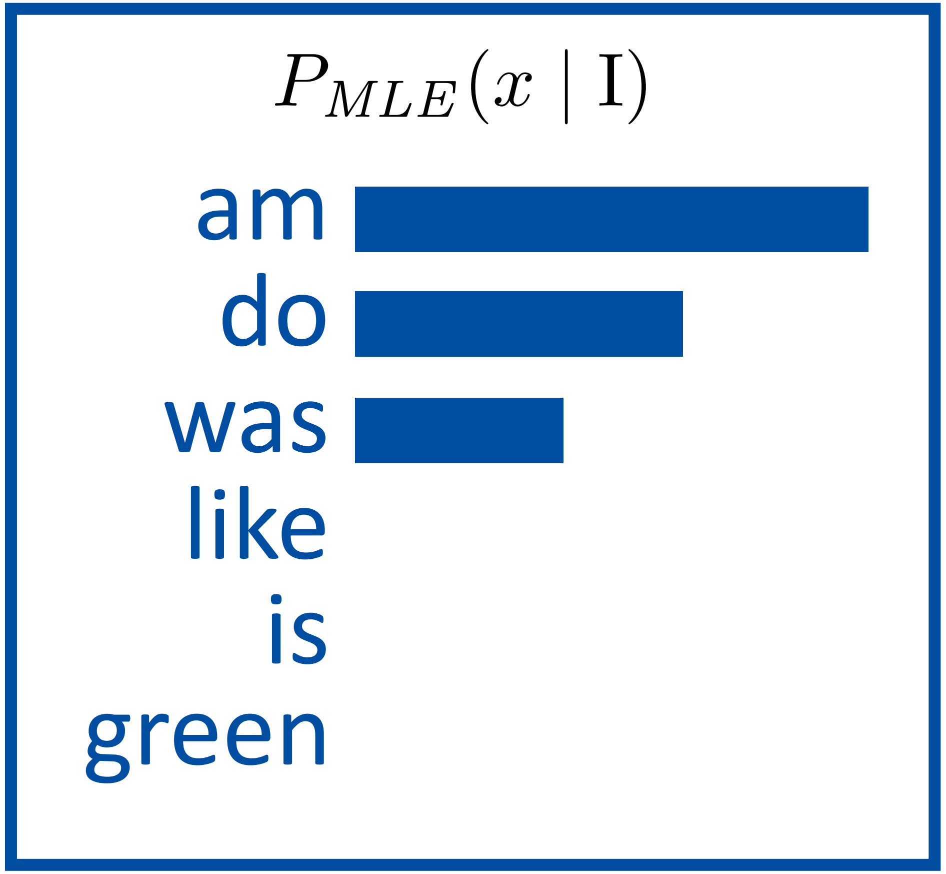
probability of test sequence is 0, so log-probability is $-\infty$,

so perplexity is ∞ !

Smoothing

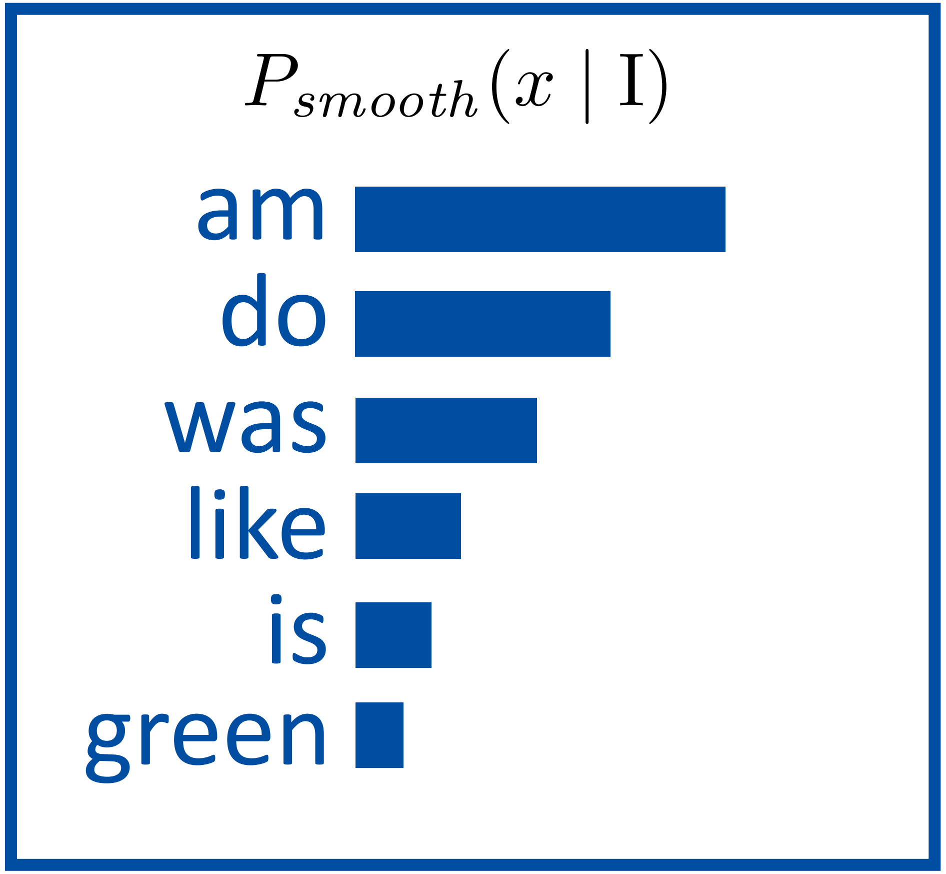
Smoothing

- instead of MLE, which leads to zeros, use a different estimation method that leads to “smoother” distributions (fewer zeros)



Smoothing

- instead of MLE, which leads to zeros, use a different estimation method that leads to "smoother" distributions (fewer zeros)



Smoothing

- Handle sparsity by making sure all probabilities are non-zero in our model
  - **Additive**: Add a small amount to all probabilities
  - **Interpolation**: Use a combination of different granularities of n-grams
  - **Discounting**: Redistribute probability mass from observed n-grams to unobserved ones

"Add-1" estimation

- just add 1 to all counts!
- also called Laplace smoothing
- MLE estimate:

$$P_{\mathrm{MLE}}(x \mid x') = \frac{\mathrm{count}(x', x)}{\mathrm{count}(x')}$$

- Add-1 estimate:

$$P_{\mathrm{add-1}}(x \mid x') = \frac{\mathrm{count}(x', x) + 1}{\mathrm{count}(x') + |\mathcal{V}|}$$

vocabulary size

- simple and avoids zeros, but doesn't work as well as other methods

[SLP3: Chapter 3]

“Add-1” estimation

- (Berkeley restaurant corpus) Out of 9222 sentences
- Raw bigram 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

[SLP3: Chapter 3]

“Add-1” estimation

- (Berkeley restaurant corpus) Out of 9222 sentences
- Smoothed bigram 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

[SLP3: Chapter 3]

"Add-1" estimation

- (Berkeley restaurant corpus) Out of 9222 sentences
- Smoothed bigram probabilities

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

[SLP3: Chapter 3]

Backoff and Interpolation

- use multiple *n*-gram sizes in the same language model

- **backoff**:
  - use trigram model if its probability is nonzero
  - otherwise, use bigram model if its probability is nonzero
  - otherwise, use unigram

- **interpolation**:
  - mixture of unigram, bigram, and trigram models

- interpolation tends to work better

[SLP3: Chapter 3]

Linear Interpolation

- estimate unigram/bigram/trigram models using MLE, then combine them:

$$P_{int}(x \mid x', x'') = \lambda_1 P_{MLE}(x) + \lambda_2 P_{MLE}(x \mid x'') + \lambda_3 P_{MLE}(x \mid x', x'')$$

$$\lambda_i \geq 0, \forall i \qquad \sum_i \lambda_i = 1$$

- lambdas can be estimated using development data
- they can also be a function of the context

$$P_{int}(x \mid x', x'') = \lambda_1(x', x'') P_{MLE}(x) + \lambda_2(x', x'') P_{MLE}(x \mid x'') + \lambda_3(x', x'') P_{MLE}(x \mid x', x'')$$

- intuitively, may want $\lambda_3(x', x'')$ to be larger if $\mathrm{count}(x', x'')$ is large

Kneser-Ney Smoothing

- widely used and effective
- a few components:
  - **absolute discounting**
  - interpolation with **continuation probabilities**
- best variant seems to be “modified Kneser-Ney” -- see Chen and Goodman (1998)

[SLP3: Chapter 3]

Absolute Discounting

Bigram count in training set	Bigram count in heldout set
0	0.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

Figure 3.9 For all bigrams in 22 million words of AP newswire of count 0, 1, 2,...,9, the counts of these bigrams in a held-out corpus also of 22 million words.

observed bigrams have counts that are **overestimated**

unobserved bigrams have counts that are **underestimated**

Absolute Discounting

- subtract *d* from each numerator count
- use original counts for denominator

$$P_{\text{AbsDisc}}(x \mid x') = \frac{\max(0, \text{count}(x', x) - d)}{\sum_v \text{count}(x', v)} + \lambda(x')P(x)$$

- so there's some "missing probability mass"
- lambda function is defined to make things normalize correctly

[SLP3: Chapter 3]

Continuation Probabilities

- *"I can't see without my reading ________"*
  - suppose we are interpolating bigram and unigram distributions here
  - "*Kong*" is more common than "*glasses*"
  - but "*Kong*" almost always follows "*Hong*"
  - *"glasses"* is more likely to follow a variety of previous words!

- unigram probability is most useful when we haven't seen bigram
- instead of unigram $P(x)$, use $P_{\text{continuation}}(x)$

How likely is *x*?

How likely is *x* to appear as a novel continuation?

Continuation Probabilities

- how likely is *x* to be a novel continuation?

$$P_{\text{continuation}}(x) \propto \underbrace{|\{x' : \text{count}(x', x) > 0\}|}_{\text{number of word types that appeared before } x}$$

- normalize by total number of bigram types:

$$P_{\text{continuation}}(x) = \frac{|\{x' : \text{count}(x', x) > 0\}|}{|\{\langle x', x'' \rangle : \text{count}(x', x'') > 0\}|}$$

Kneser-Ney Smoothing

- Interpolated Kneser-Ney:

$$P_{\mathrm{KN}}(x \mid x') = \frac{\max(0, \mathrm{count}(x', x) - d)}{\sum_v \mathrm{count}(x', v)} + \lambda(x')P_{\mathrm{continuation}}(x)$$

- again, lambda function is defined to make things normalize correctly

- this is the bigram version; recursive versions exist for higher orders

[SLP3: Chapter 3]

Huge Web-scale n-grams

- Google n-gram release, August 2006

All Our N-gram are Belong to You

THURSDAY, AUGUST 03, 2006

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects, such as statistical machine translation, speech recognition, spelling correction, entity detection, information extraction, and others. While such models have usually been estimated from training corpora containing at most a few billion words, we have been

decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

https://blog.research.google/2006/08/all-our-n-gram-are-belong-to-you.html

Huge Web-scale n-grams

- Google n-gram release, August 2006

All Our N-gram are Belong to You

THURSDAY, AUGUST 03, 2006

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of ru
publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There
unique words, after discarding words that appear less than 200 times.

```
File sizes: approx. 24 GB compressed (gzip'ed) text files

Number of tokens:    1,024,908,267,229
Number of sentences:    95,119,665,584
Number of unigrams:         13,588,391
Number of bigrams:         314,843,401
Number of trigrams:        977,069,902
Number of fourgrams:     1,313,818,354
Number of fivegrams:     1,176,470,663
```

The following is an example of the 4-gram data in this corpus:

```
serve as the incoming 92
serve as the incubator 99
serve as the independent 794
serve as the index 223
serve as the indication 72
serve as the indicator 120
serve as the indicators 45
serve as the indispensable 111
serve as the indispensible 40
serve as the individual 234
serve as the industrial 52
serve as the industry 607
serve as the info 42
```

https://blog.research.google/2006/08/all-our-n-gram-are-belong-to-you.html

Huge Web-scale n-grams

- How to deal with, e.g., Google N-gram corpus
- Pruning
  - Only store N-grams with count > threshold.
    - Remove singletons of higher-order n-grams
  - Entropy-based pruning
- Efficiency
  - Efficient data structures like tries
  - Bloom filters: approximate language models
  - Store words as indexes, not strings
    - Use Huffman coding to fit large numbers of words into two bytes
  - Quantize probabilities (4-8 bits instead of 8-byte float)

Smoothing for Web-scale Models

- “Stupid backoff” (Brants et al., 2007):

$$S(x \mid x', x'') = \begin{cases} P_{MLE}(x \mid x', x'') & \text{if count}(x', x'', x) > 0 \\ 0.4S(x \mid x'') & \text{otherwise} \end{cases}$$

$$S(x) = P_{MLE}(x)$$

Closed Vocabulary

- smoothing avoids zeros for unknown ***n*-grams** (*n* > 1), not unknown words!
- if there are unknown words in the test data, smoothing does not help
  - probability of test data is still zero
- we must know the full vocabulary ahead of time (for both training *and* held-out data!)

Open Vocabulary

- create an unknown word symbol "<UNK>"
- at training time:
  - replace some rare words with <UNK>
  - then estimate probabilities as though <UNK> is a normal word
- at test time:
  - replace unknown words with <UNK>

- when comparing open-vocabulary language models, make sure the vocabularies match!

- world's best language model (every word is <UNK>):

$$P(\texttt{<UNK>} \mid \texttt{<s>}) = 1$$
$$P(\texttt{<UNK>} \mid \texttt{<UNK>}) = 0.97$$
$$P(\texttt{</s>} \mid \texttt{<UNK>}) = 0.03$$

Language Modeling Toolkits

- SRILM

  http://www.speech.sri.com/projects/srilm/

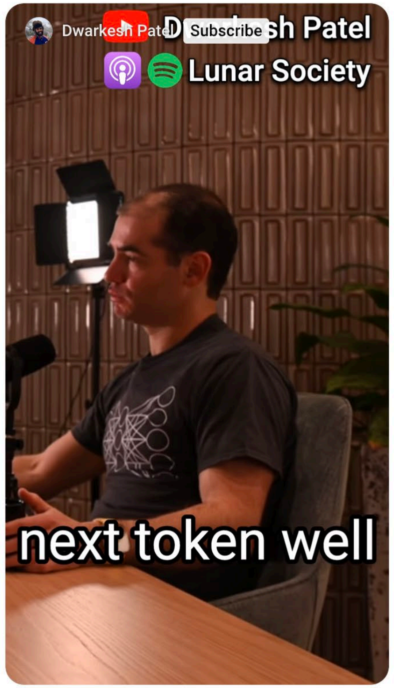
- KenLM

  https://kheafield.com/code/kenlm/

Next Token Prediction Solves AI?

- [Next Token Prediction SOLVES AI Says OpenAI Founder]() ?

Next Token Prediction SOLVES AI Says OpenAI Founder
YouTube · Dwarkesh Patel · Mar 29, 2023
YouTube



Language Modeling

- Building language models
- Generating from a language model
- Evaluating a language model

- Count-based language models
  - MLE estimation
  - Smoothing

- Neural language models
  - Feed-forward models
  - RNN models
  - Attention models

Summary

- **language modeling**:
  - compute probabilities of token sequences
  - length of sequence must be modeled probabilistically (usually with a stop symbol at the end)
  - typically, use chain rule to factor joint into product of conditionals, one for each token in order from left to right:

$$P(\boldsymbol{x}_{1:n}) = P(\texttt{</s>} \mid \texttt{<s>}, x_1, x_2, \ldots, x_n) \prod_{i=1}^{n} P(x_i \mid \texttt{<s>}, x_1, x_2, \ldots, x_{i-1})$$

Summary

- ***n*-gram language models**:
  - let each conditional probability depend on only the most recent *n*-1 tokens, e.g., trigram:

$$P(\text{ham} \mid \texttt{<s>}\text{I do not like green eggs and}) \approx P(\text{ham} \mid \text{eggs and})$$

- we can use maximum likelihood estimation to estimate *n*-gram probabilities from data, e.g., for a bigram model:

$$P(x \mid x') = \frac{\text{count}(x', x)}{\text{count}(x')}$$

Summary

- evaluation of language models
  - extrinsic: use model in a system for a downstream task
  - intrinsic: compute probability of held-out data (standard metric: **perplexity**)

- **perplexity**:
  - compute ℓ = average log-probability of held-out tokens, perplexity is $2^{-\ell}$
  - lower perplexity → better language model
  - can be interpreted as effective number of choices per position on average

Summary

Smoothing

- **add-1 estimation**: add 1 (or some small number) to all counts, then normalize

- **backoff**: if high order *n*-gram has been seen, use its probability, otherwise “back off” to lower order *n*-grams

- **interpolation**: weighted mixture of *n*-gram models of various sizes:

$$P_{int}(x \mid x', x'') = \lambda_1 P_{MLE}(x) + \lambda_2 P_{MLE}(x \mid x'') + \lambda_3 P_{MLE}(x \mid x', x'')$$

  - weights can depend on context

Summary

Smoothing

- **absolute discounting**:
  - observed *n*-grams have counts that are *over*estimated
  - unobserved *n*-grams have counts that are *under*estimated
  - subtract a constant from counts, normalize using interpolation with a lower order *n*-gram model
- **continuation probabilities**:
  - captures how likely it is for a word to form a novel continuation of the preceding words
  - likely more helpful than simple unigram probabilities when interpolating with a bigram model
- **Kneser-Ney smoothing**:
  - combines absolute discounting and continuation probabilities via interpolation
- **stupid backoff**:
  - simple, scales well to very large corpora

Summary

- closed vs. open vocabulary language modeling
  - when comparing language models, be mindful of vocabularies!