CS 489/698: Introduction to Natural Language Processing

Lecture 6.1: Language Modeling I

Instructor: Freda Shi

fhs@uwaterloo.ca

January 26th , 2026

UNIVERSITY OF WATERLOO

Announcements

- Assignment 1 should be submitted via LEARN to be properly graded.
  - Added one DP encoder that is more suitable with SentencePiece-like algorithms.
  - Your final mark will be based on the better compression ratio applying to both algorithms.
- CS698: Two projects with different dues
  - Project 1 (shared with CS489, 25%): The Interstellar Autocomplete Challenge
    - Initial proposal due: Feb 2nd
    - Midterm check-in due: Feb 23rd
    - Final submission: Final exam weeks
    - Details at https://github.com/waterloo-nlp/intro-to-nlp-project
  - Project 2 (NLP-related research project at your own choice; 25%)
    - Proposal/midterm check-in due Feb 23rd: 1-page PDF
    - Final submission: Final exam weeks

Reminder

- Kaggle public leaderboard (training set) performance does (usually positively) correlate with but not guarantee the performance on held-out data.
    - Example (sentiment classification):

```
if sentence in training set:
    return label of sentence in training set
else:
    return random label in {0, 1}
```

- Don't optimize too hard on the training set, otherwise your model might overfit.

Suggestion: reserve some training examples to estimate the performance on the held-out data.

Outline: Language Modeling

- Language modeling
    - General approach
        - Length Modeling
        - Formulation
        - Example: N-gram language models
        - Evaluation
    - Autoregressive language models
    - Masked language models

Recap: Distributional Semantics

You shall know a word by the company it keeps.

-- J. R. Firth, 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.

A bottle of _____ is on the table.

Everybody likes _____.

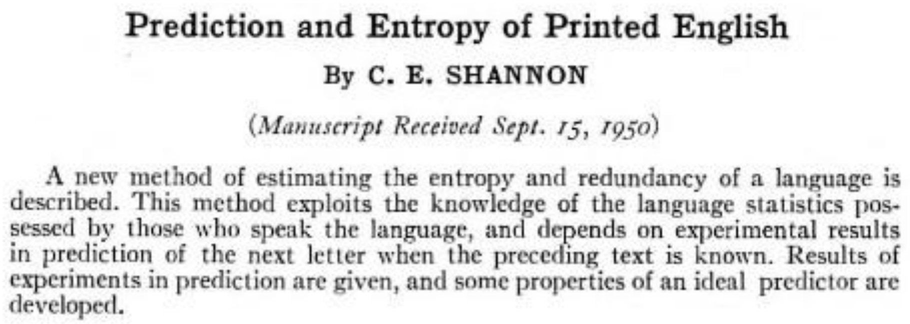
Don’t have _____ before you drive.

We make _____ out of corn.

The Shannon Game

- The Shannon game [Shannon 1951]:

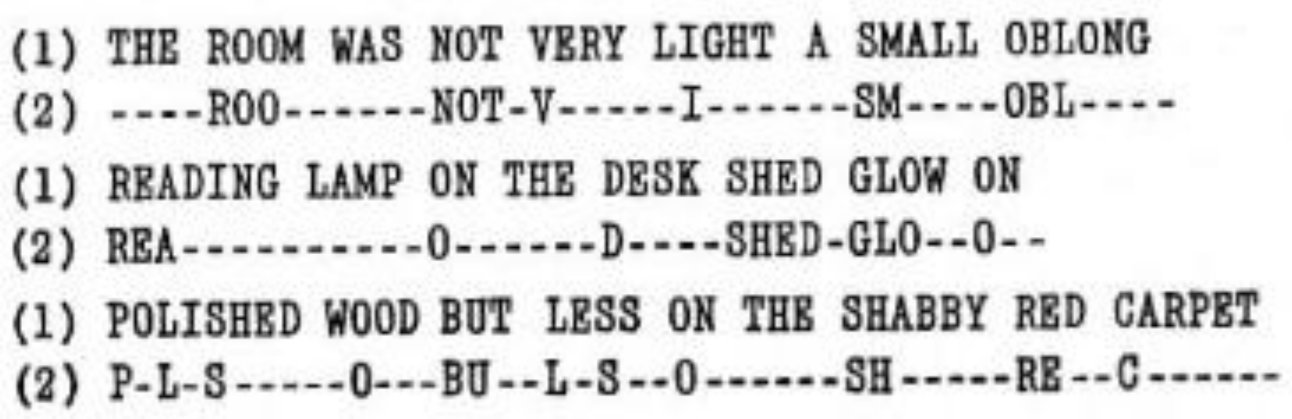
How well can you predict the next letter?


Prediction and Entropy of Printed English

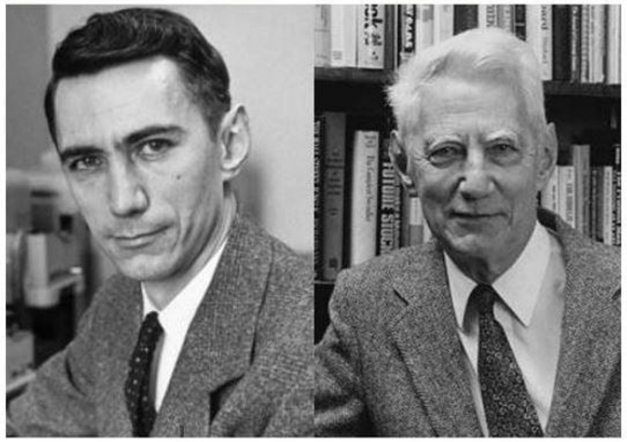
By C. E. SHANNON

(*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------
```

Claude Shannon



30 Apr 1916 – 24 Feb 2001

Language Models: General Formulation

Language model computes a probability distribution over strings in a language:

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

$\mathbf{x}$: A string with tokens $x_1, x_2, \dots x_n$.

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

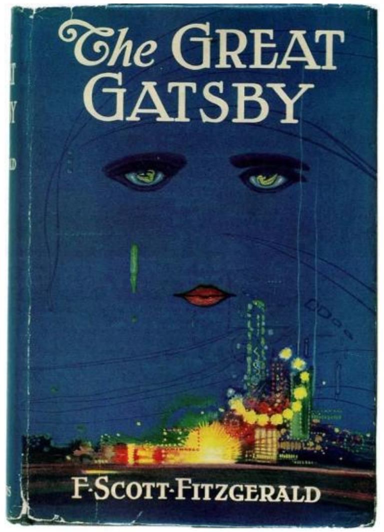
$$p(\text{He is hungry}) = 0.000025$$

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

Language Models: General Formulation

Language model computes a probability distribution over strings in a language:

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

$P\Bigg($  $\Bigg)$

$P\Bigg($ $\Bigg)$

Nowadays: large language models

$$P(x)$$

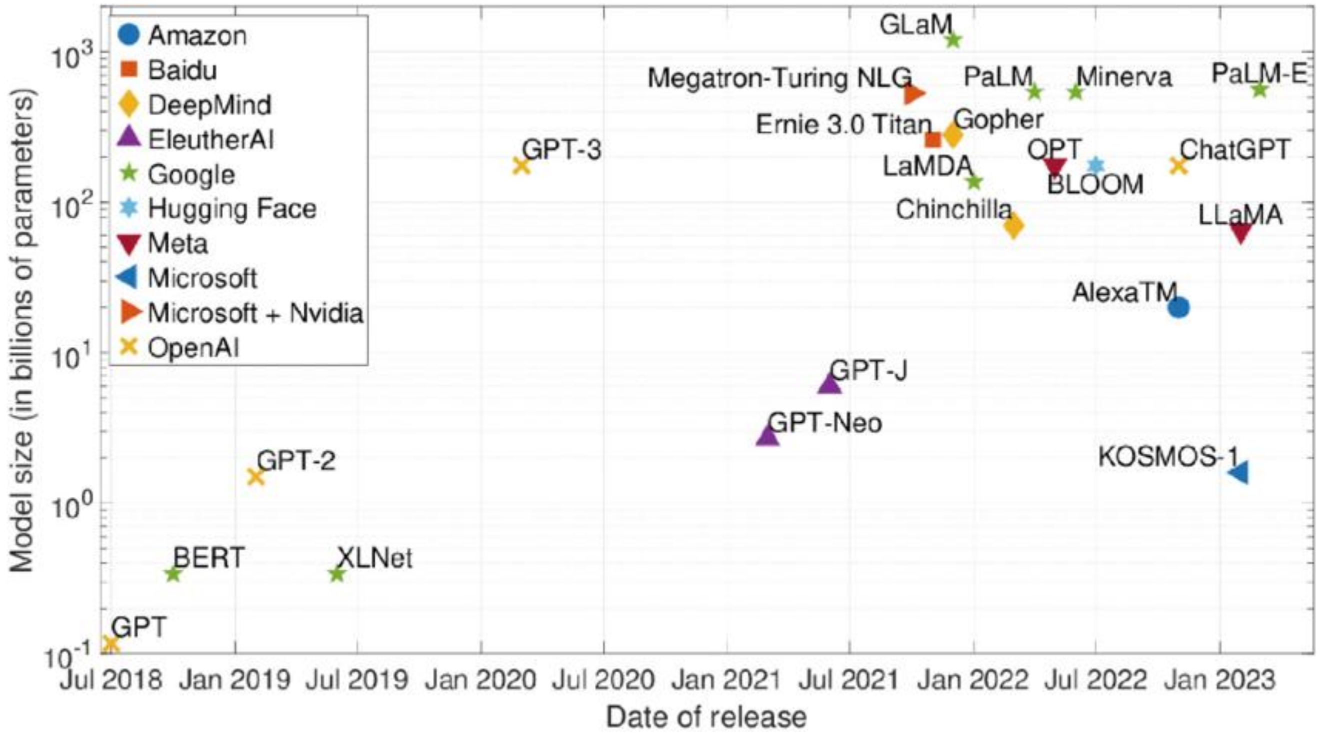
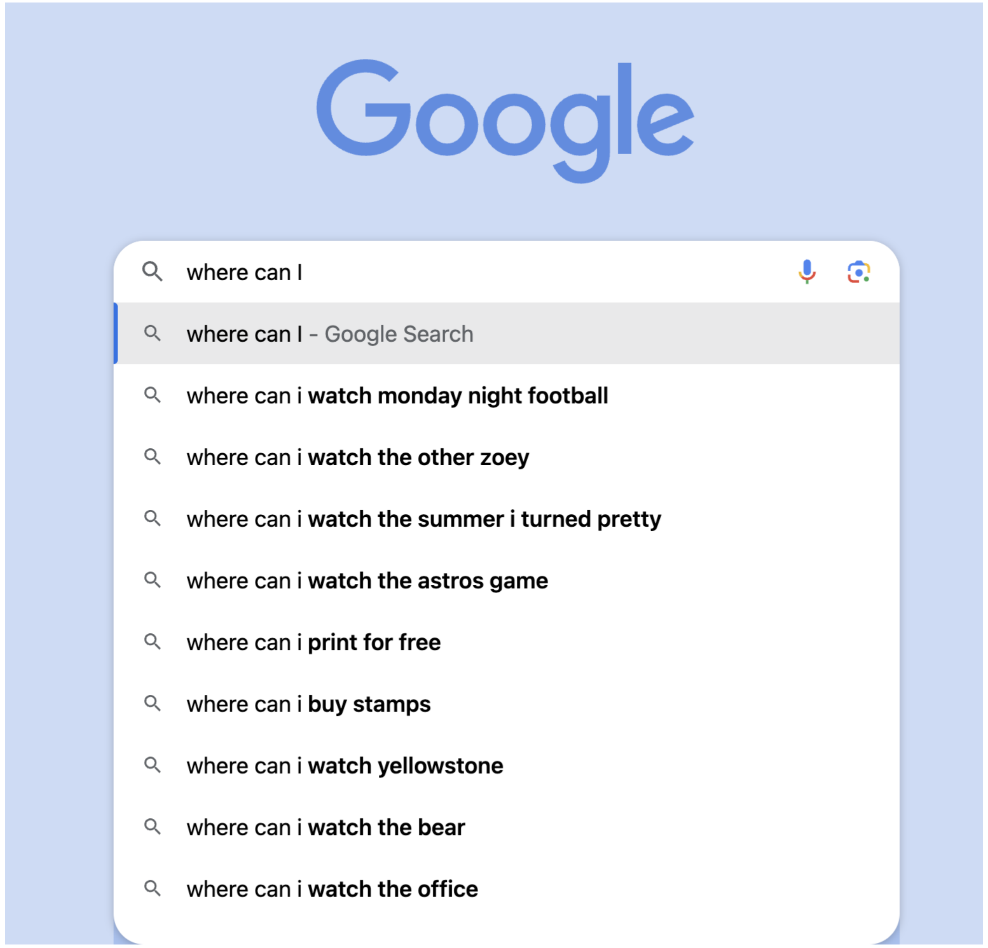


Figure credit: Kath et al. (2023)

Languages models are now everywhere!



where can I

where can I - Google Search

where can i **watch monday night football**

where can i **watch the other zoey**

where can i **watch the summer i turned pretty**

where can i **watch the astros game**

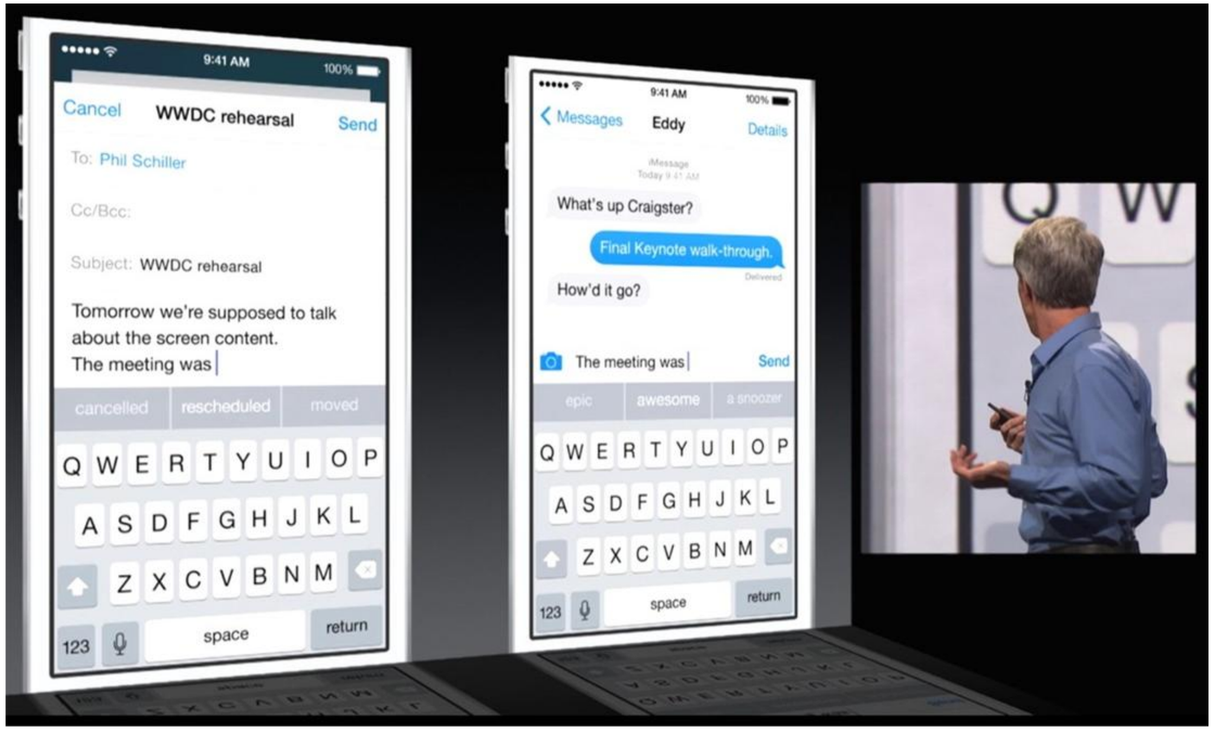
where can i **print for free**

where can i **buy stamps**

where can i **watch yellowstone**

where can i **watch the bear**

where can i **watch the office**



Language Models: General Formulation

Language modeling: assign probabilities to token sequences.

Why?

- Machine translation:
  - P(*turn the camera off*) > P(*put the camera out*)
- Speech recognition:
  - P(*I need to recognize speech*) > P(*I need to wreck a nice speech*)
- Spelling/grammar correction:
  - The office is about fifteen minuets from my house
  - P(*about fifteen minutes from*) > P(*about fifteen minuets from*)

- (modeling) Define a statistical model $P_{\Theta}(\mathbf{x})$, where $\mathbf{x}$ is a string.
- (learning) Estimate the parameters Θ from data (by maximizing likelihood).

Language Modeling

Goal – Compute the probability of a sequence of words:

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

Relatedly, modeling probability of the next word:

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

Relatedly, modeling the probability of a masked word given its context:

$$P(x_k \mid x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n)$$

A model that computes any of the above is called a language model.

A good language model will assign higher probabilities to sentences that are more likely to appear.

Language Modeling

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

How do we model this?

Recap: the chain rule of probability.

$$P(A, B) = P(A)P(B \mid A)$$

$$\begin{aligned} P(\mathbf{x}_{1:n}) &= P(x_1, x_2, \ldots, x_n) \\ &= P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2) \ldots P(x_n \mid x_1, \ldots, x_{n-1}) \end{aligned}$$

We haven't yet made any independence assumptions!

Autoregressive Language Modeling

$$\begin{aligned} P(\mathbf{x}_{1:n}) &= P(x_1, x_2, \ldots, x_n) \\ &= P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2) \ldots P(x_n \mid x_1, \ldots, x_{n-1}) \end{aligned}$$

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

Important Detail: Modeling Length

- A language model assigns probability to token sequences $\mathbf{x}$.
  - $\mathbf{x}$ can be of any length, and the probabilities should sum to 1 across all possible sequences of different lengths.
- Usually length is modeled by including a “stop symbol” </s> at the end of sequence.
  Predicting </s> = modeling stopping probabilities.
  - Relatedly, a “start symbol” <s> could also be added to the beginning.

Language model with both start and stop symbols:

$$P(\mathbf{x}_{1:n}) = P\left(\langle \texttt{/s} \rangle \mid \langle \texttt{s} \rangle, x_1, \ldots, x_n\right) \prod_{i=1}^{n} P\left(x_i \mid \langle \texttt{s} \rangle, x_1, \ldots, x_{i-1}\right)$$

Why Stopping Probability?

Our language model:

$$P(\mathbf{x}_{1:n}) = P\left(\langle\texttt{/s}\rangle \mid \langle\texttt{s}\rangle, x_1, \ldots, x_n\right) \prod_{i=1}^{n} P\left(x_i \mid \langle\texttt{s}\rangle, x_1, \ldots, x_{i-1}\right)$$

We need to ensure:

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

Consider removing stopping probabilities:

$$P(\mathbf{x}_{1:n}) = \prod_{i=1}^{n} P\left(x_i \mid \langle\texttt{s}\rangle, x_1, \ldots, x_{i-1}\right)$$

What happens if we don't model stropping probability?

Recall that we need:

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

Sums of probabilities for all possible length-1 and length-2 sequences:

$$\sum_{n=1}^{1} \sum_{\mathbf{x}_{1:n}} P(\mathbf{x}_{1:n}) = \sum_{x \in V} P(x \mid \langle \mathbf{s} \rangle) = 1$$

$$\sum_{n=2}^{2} \sum_{\mathbf{x}_{1:n}} P(\mathbf{x}_{1:n}) = \sum_{x \in V} \sum_{x' \in V} P(x \mid \langle \mathbf{s} \rangle, x') P(x' \mid \langle \mathbf{s} \rangle) = 1$$

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

With the stop symbol…

$$\sum_{n=1}^{\infty} \sum_{\mathbf{x}_{1:n}} P(\mathbf{x}_{1:n}, \langle\texttt{/s}\rangle) = \sum_{n=1}^{\infty} \sum_{\mathbf{x}_{1:n}} P(\mathbf{x}_{1:n}) P(\langle\texttt{/s}\rangle \mid \mathbf{x}_{1:n})$$

Proof sketch: once you reach the </s> token after sampling some certain sequence, certain probability mass is “taken away” because </s> is in the same distribution as other vocabulary entries.

Longer sequences that have the same prefix share the “remaining” probability.

Practice: complete the proof that the above quantity is 1 when $n \to \infty$.

Other Ways of Modeling Length

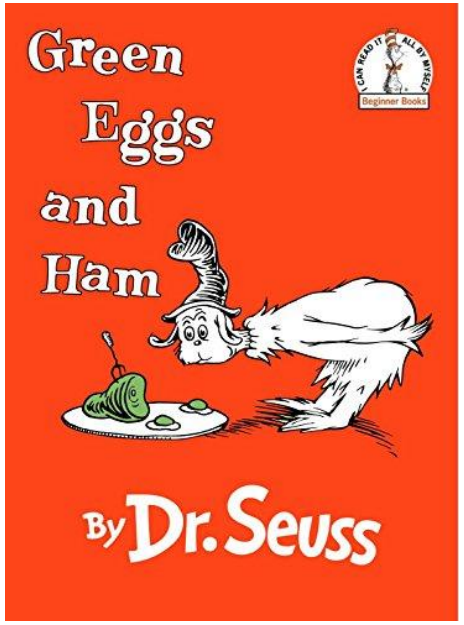
- Alternatively, we can model the length n explicitly (e.g., using a zero-truncated Poisson distribution):

$$P(\mathbf{x}_{1:n}) = P(n) \prod_{i=1}^{n} P(x_i \mid \langle \mathbf{s} \rangle, x_1, \ldots, x_{i-1})$$

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) 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

Andrey Markov

- 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

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

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

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

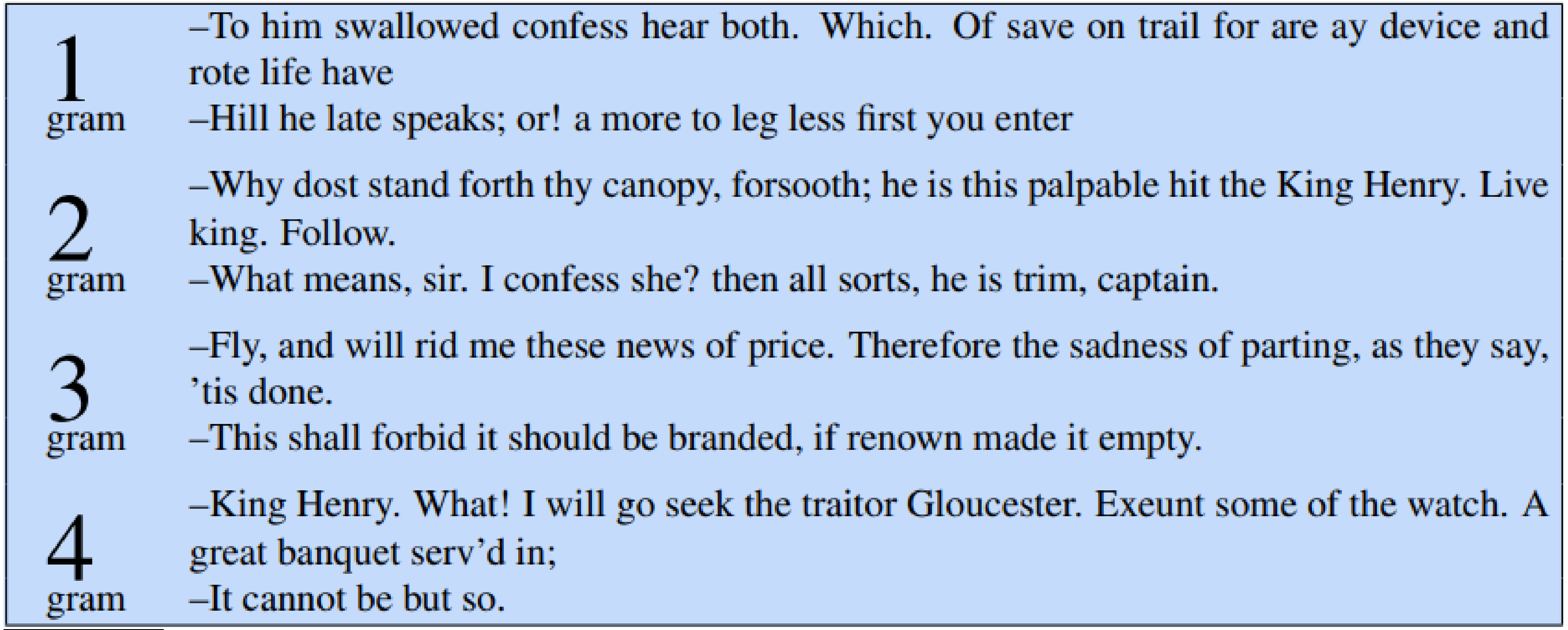
- $n = 2$: Bigram language model

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

- $n = 3$: Trigram language model

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

Example Sentences from N-Gram 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]

Recap: Modeling, Learning, and Inference in Classification

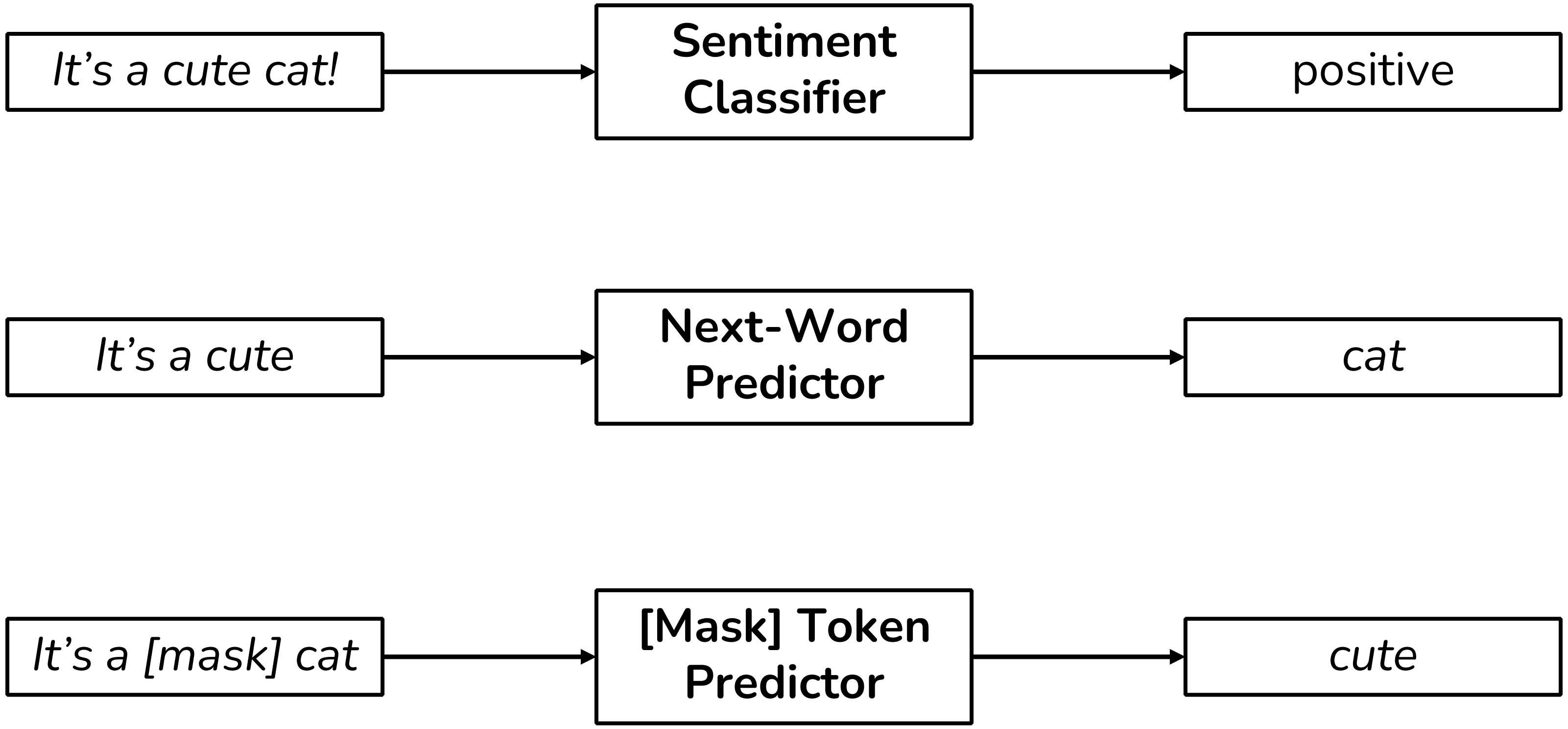
Inference: solve $\arg\max$

Modeling: define score function

$$\text{classify}(s) = \underset{y}{\text{argmax}}\ \text{score}(s, y; \Theta)$$

Learning: choose parameter

General Formulation: Language Models as Classifiers



Modeling, Learning, and Inference

Inference: solve $\arg\max$

Modeling: define score function

$$\text{nextword}(s) = \underset{\mathrm{y}}{\text{argmax}}\ \text{score}(\mathrm{s}, \mathrm{y}; \Theta)$$

Learning: choose parameter

Estimating N-Gram Probabilities

- Maximum likelihood estimate (MLE)

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

Estimating N-Gram Probabilities

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:

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

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

[SLP3: Chapter 3]

Smoothing in N-Gram LMs

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

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:

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

- Simple and avoids zeros, and there are many advanced “smoothing” methods.

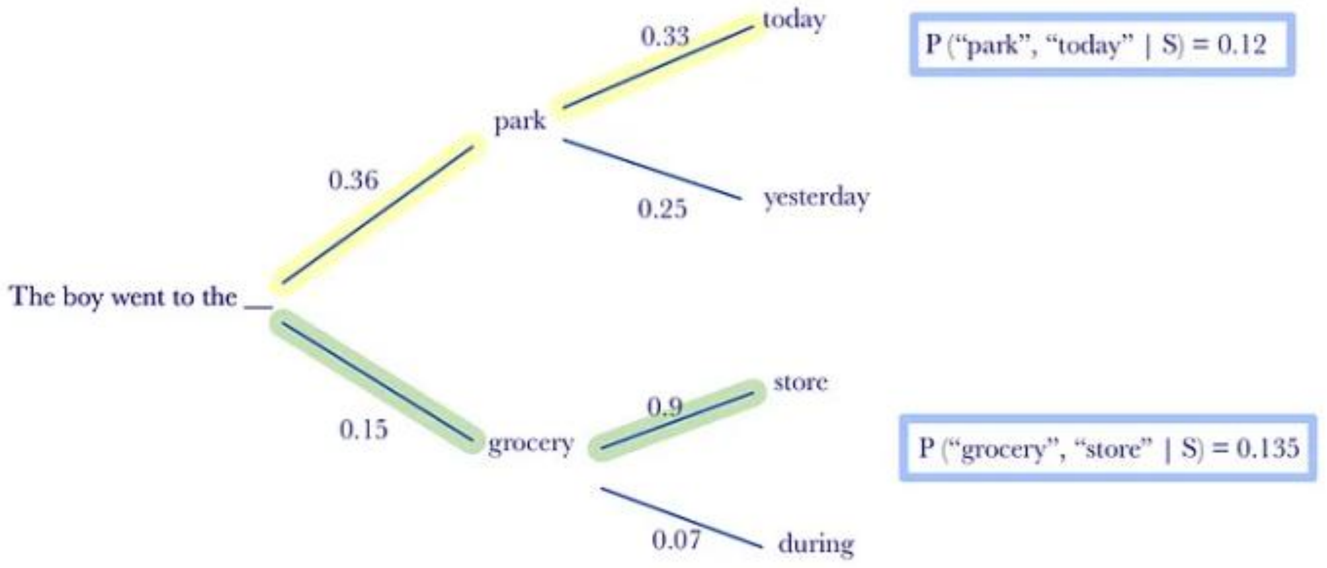
[SLP3: Chapter 3]

Generating from a Language Model

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

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

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



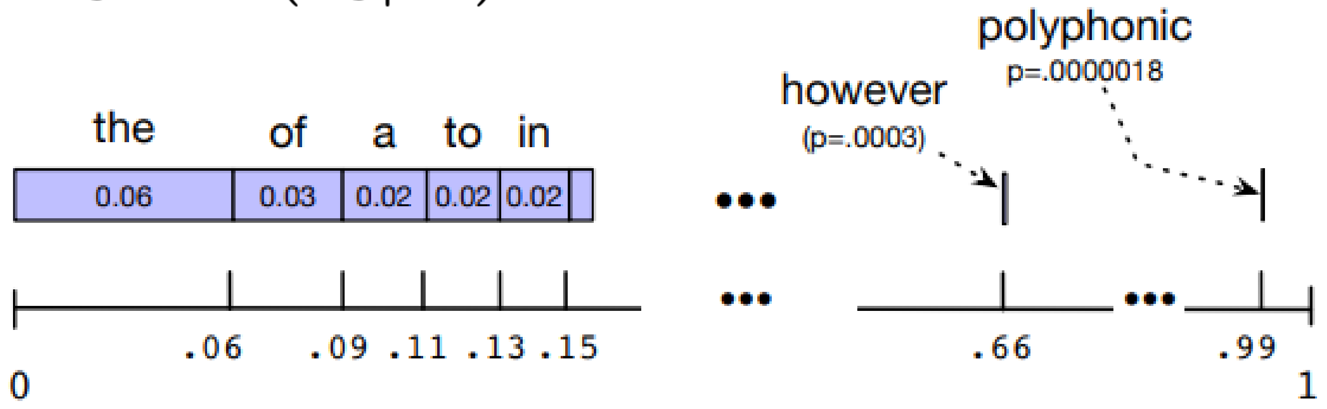
Generating from a Language Model

- Bigram model

- Generate the first word
- Generate the second word
- Generate the third word
- …

$$w_1 \sim P(x_1 | < \mathrm{s} >)$$
$$w_2 \sim P(x_2 | x_1)$$
$$w_3 \sim P(x_3 | x_2)$$

Sampling

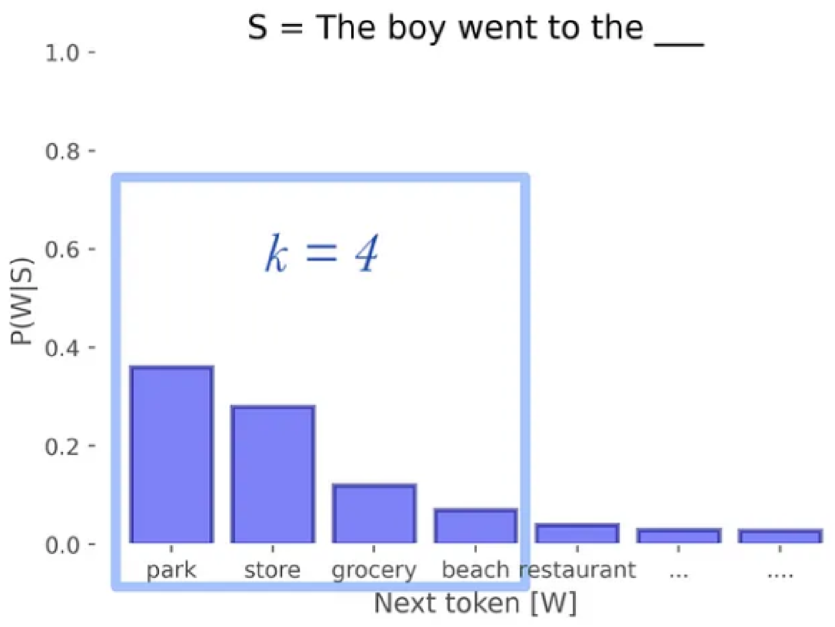


[SLP3: Chapter 3]

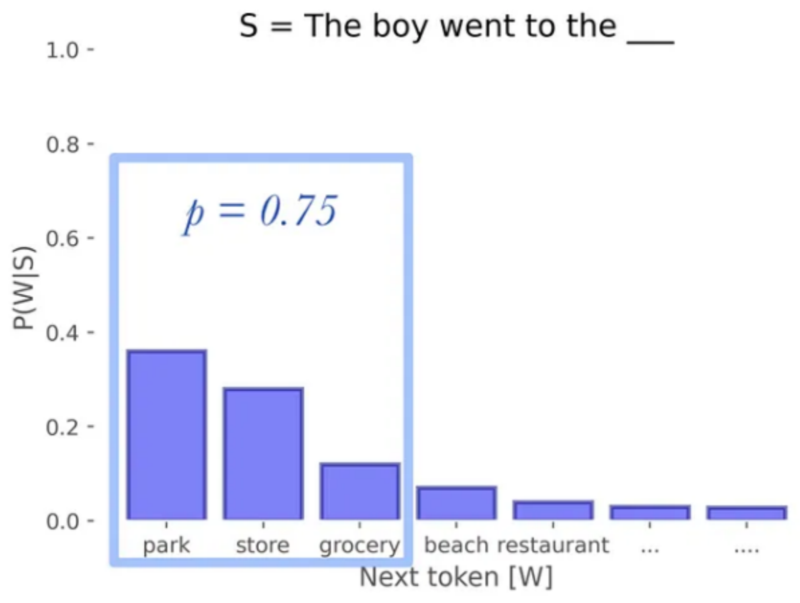
Generating from a Language Model

Two effective sampling strategies: excluding the possibility for tokens with very-low probabilities.

- Top-k vs. top-p sampling



Top-k sampling



Top-p sampling

[Holtzman et al., 2020]

Evaluating Language Models

A good language model will assign higher probabilities to sentences that are more likely to appear.

- Compute probability on held-out data.
- Standard metric: perplexity.

- Caveat: these metrics correlate with but do not necessitate good downstream performance.

Held-Out Probability

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:

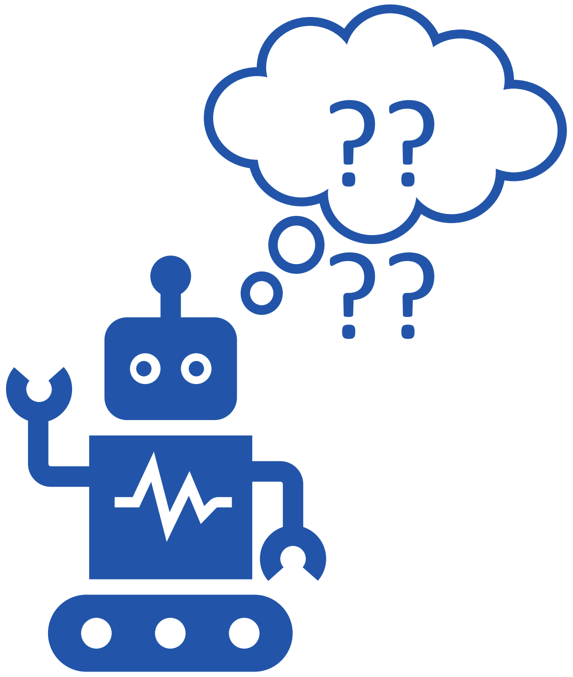
$$\ell = \frac{1}{M} \sum_i \log_2 P(\boldsymbol{x}^{(i)})$$

- **Perplexity:**

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

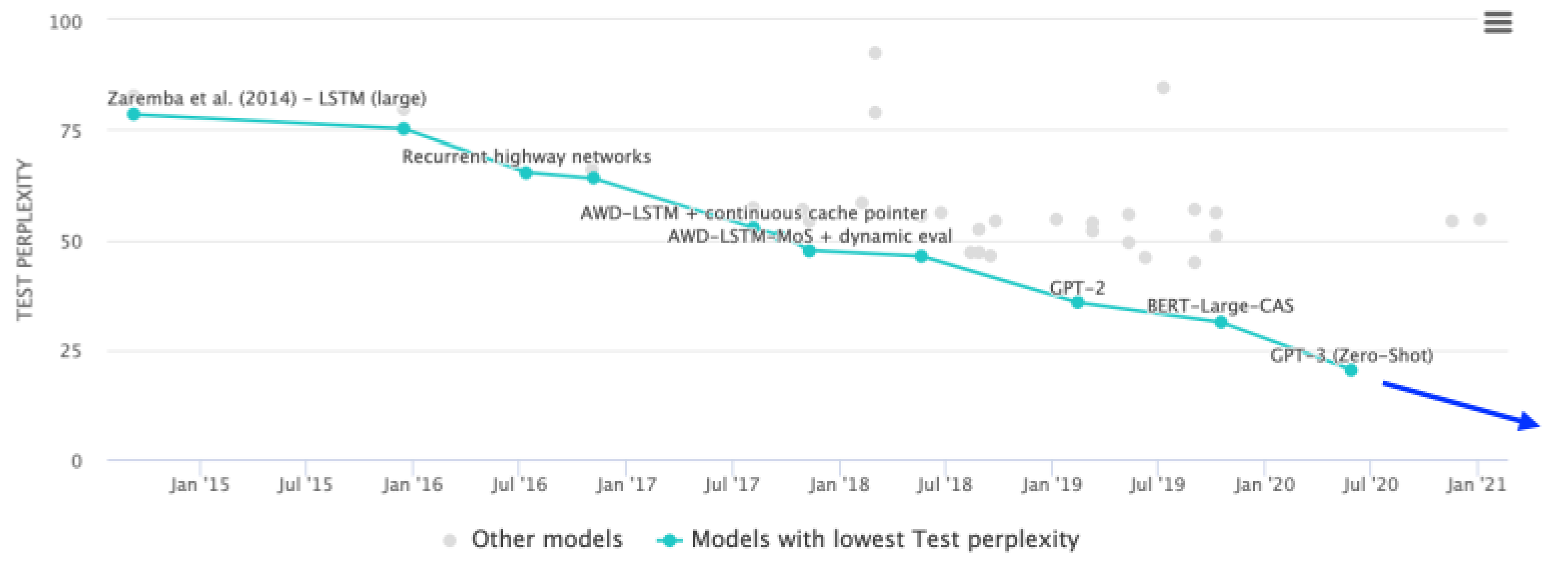
- The lower the perplexity, the better the model.

$$\text{ppl} = \left(\prod_i P(\boldsymbol{x}^{(i)}) \right)^{-\frac{1}{M}}$$



Perplexity of Models

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



GPT-3 175B:
ppl = 20.5

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

Next

- Constituency syntax and context-free grammars
- Probing syntactic knowledge in neural language models