Introduction to Language Modeling

H. Andrew Schwartz

CSE538 - Spring 2024

Stony Brook University
Human Language Analysis Beings

Language Modeling

-- assigning a probability to sequences of words.

Language Modeling

-- assigning a probability to sequences of words.

Version 1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

:probability of a sequence of words

Language Modeling

-- assigning a probability to sequences of words.

Version 1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

:probability of a sequence of words

Version 2: Compute $P(w5 | w1, w2, w3, w4)$

$= P(w_n | w_1, w_2, ..., w_{n-1})$

:probability of a next word given history

Language Modeling

Version 1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

:probability of a sequence of words

$P(\text{He ate the cake with the fork}) = ?$

Version 2: Compute $P(w5 | w1, w2, w3, w4)$

$= P(w_n | w_1, w_2, \ldots, w_{n-1})$

:probability of a next word given history

$P(\text{fork} \mid \text{He ate the cake with the}) = ?$

Language Modeling

Version 1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

:probability of a sequence of words

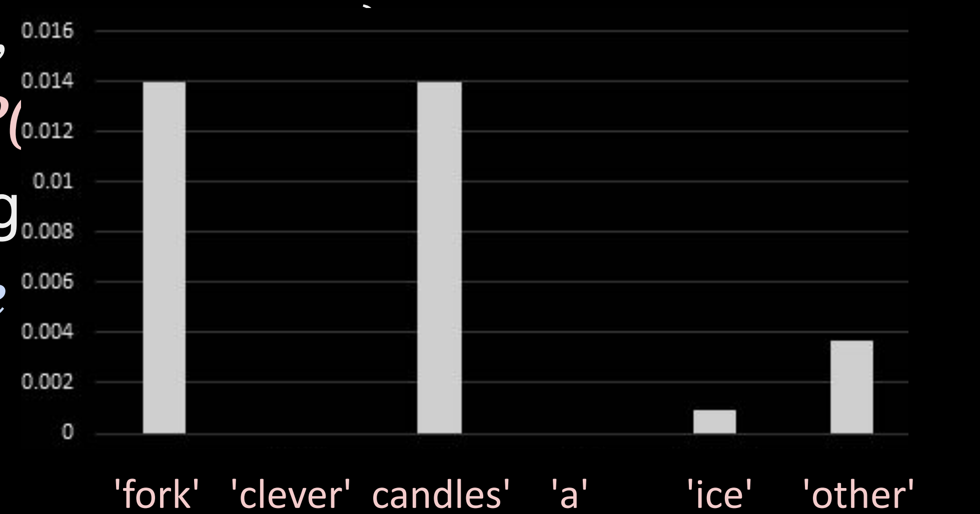
$P(\text{He ate the cake with the fork}) = ?$

Version 2: Compute $P(w5 | w1,$

$= P($

:probability of a next word g

$P(\text{fork} \mid \text{He ate the cake}$

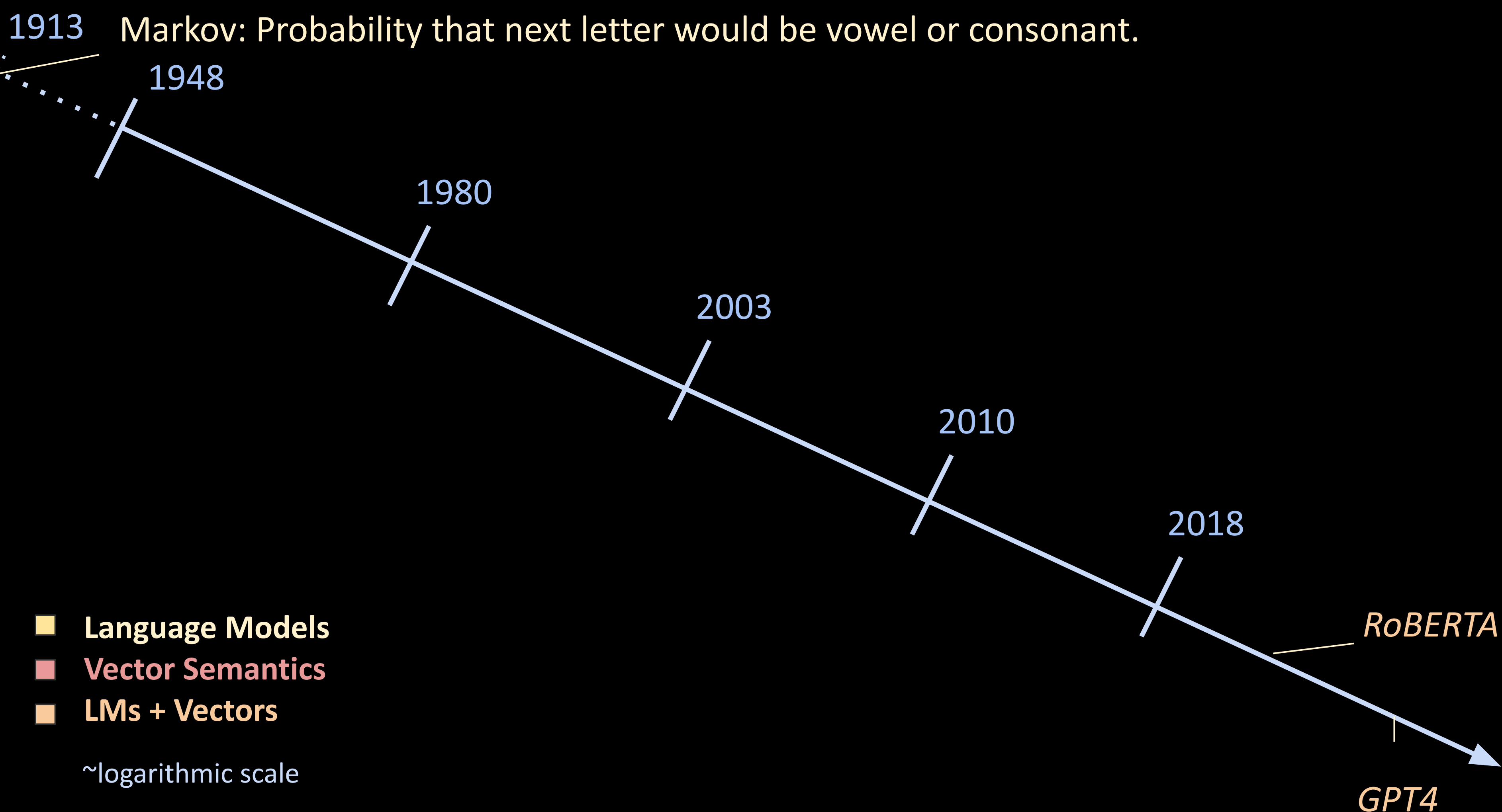


Language Modeling

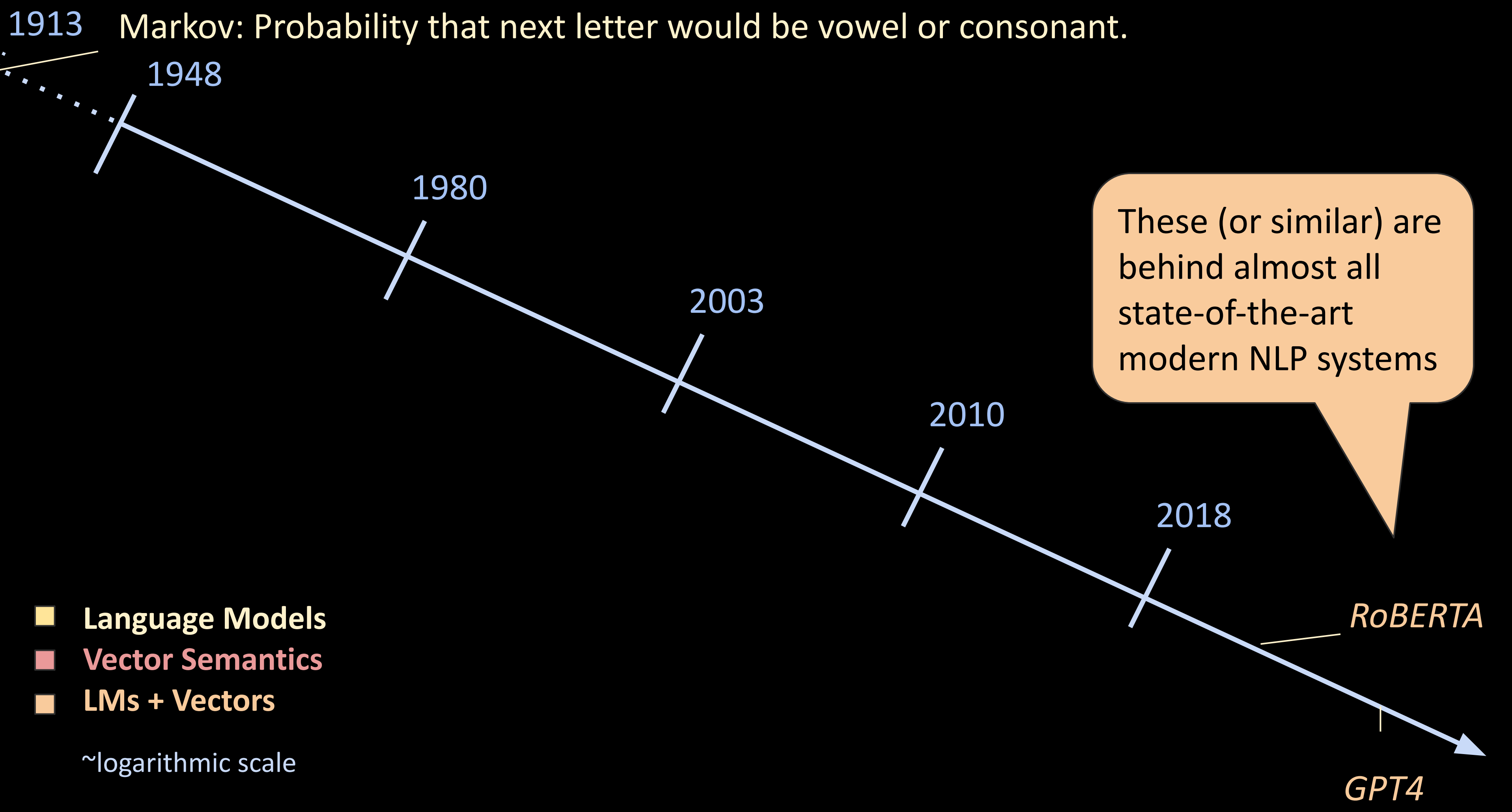
Applications:

- Auto-complete: What word is next?
- Machine Translation: Which translation is most likely?
- Spell Correction: Which word is most likely given error?
- Speech Recognition: What did they just say?
  "eyes aw of an"
  (example from Jurafsky, 2017; *did you say "giraffe ski 2,017"?*)

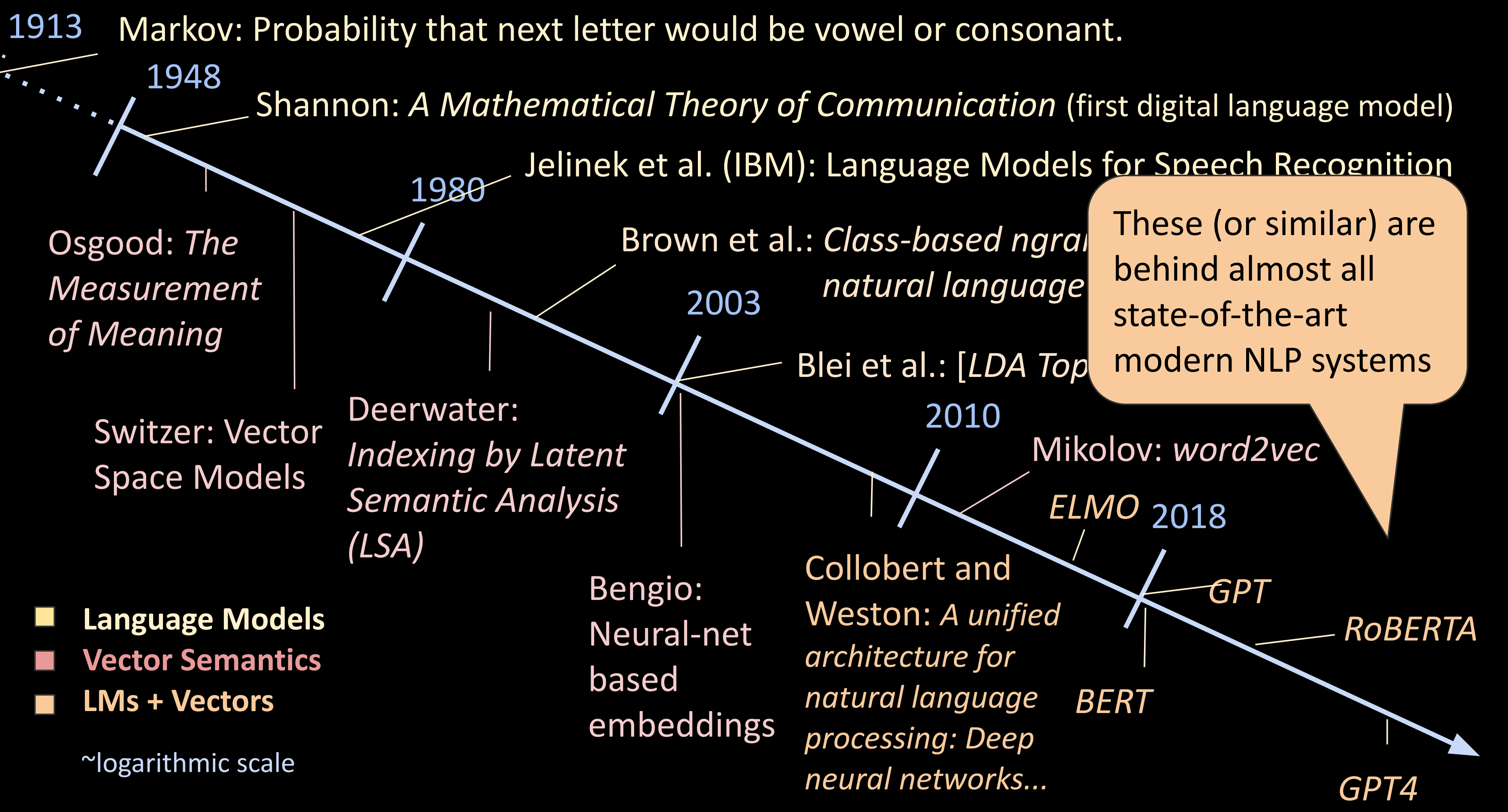
Timeline: *Language Modeling* and *Vector Semantics*



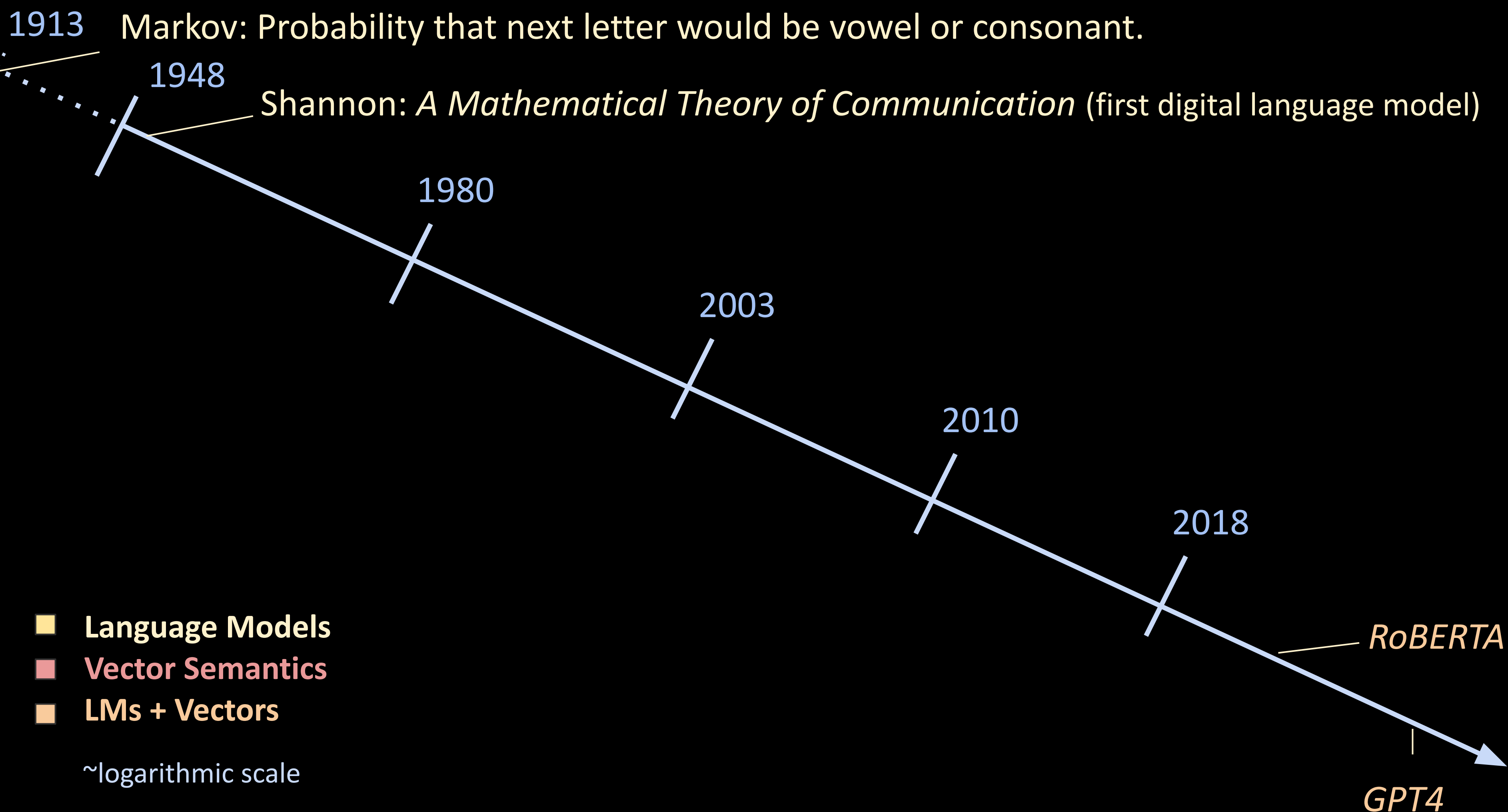
Timeline: *Language Modeling* and *Vector Semantics*



Timeline: *Language Modeling* and *Vector Semantics*



Timeline: *Language Modeling* and *Vector Semantics*



Timeline: *Language Modeling* and *Vector Semantics*

1913 Markov: Probability that next letter would be vowel or consonant.

1948 Shannon: *A Mathematical Theory of Communication* (first digital language model)

Osgood: *The Measurement of Meaning*

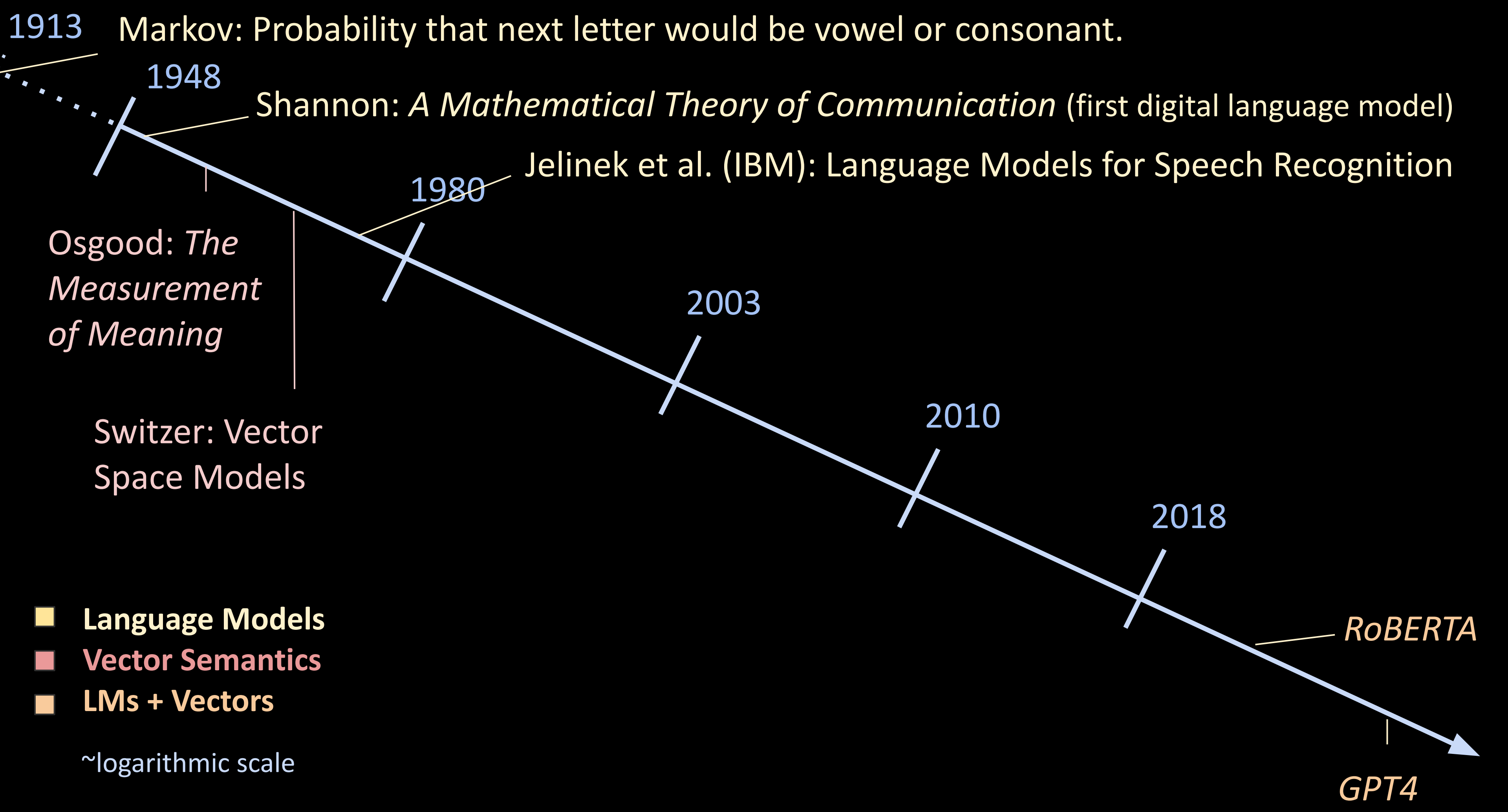
(Li et al. ,2015; Jurafsky et al., 2019)

- Language Models
- Vector Semantics
- LMs + Vectors

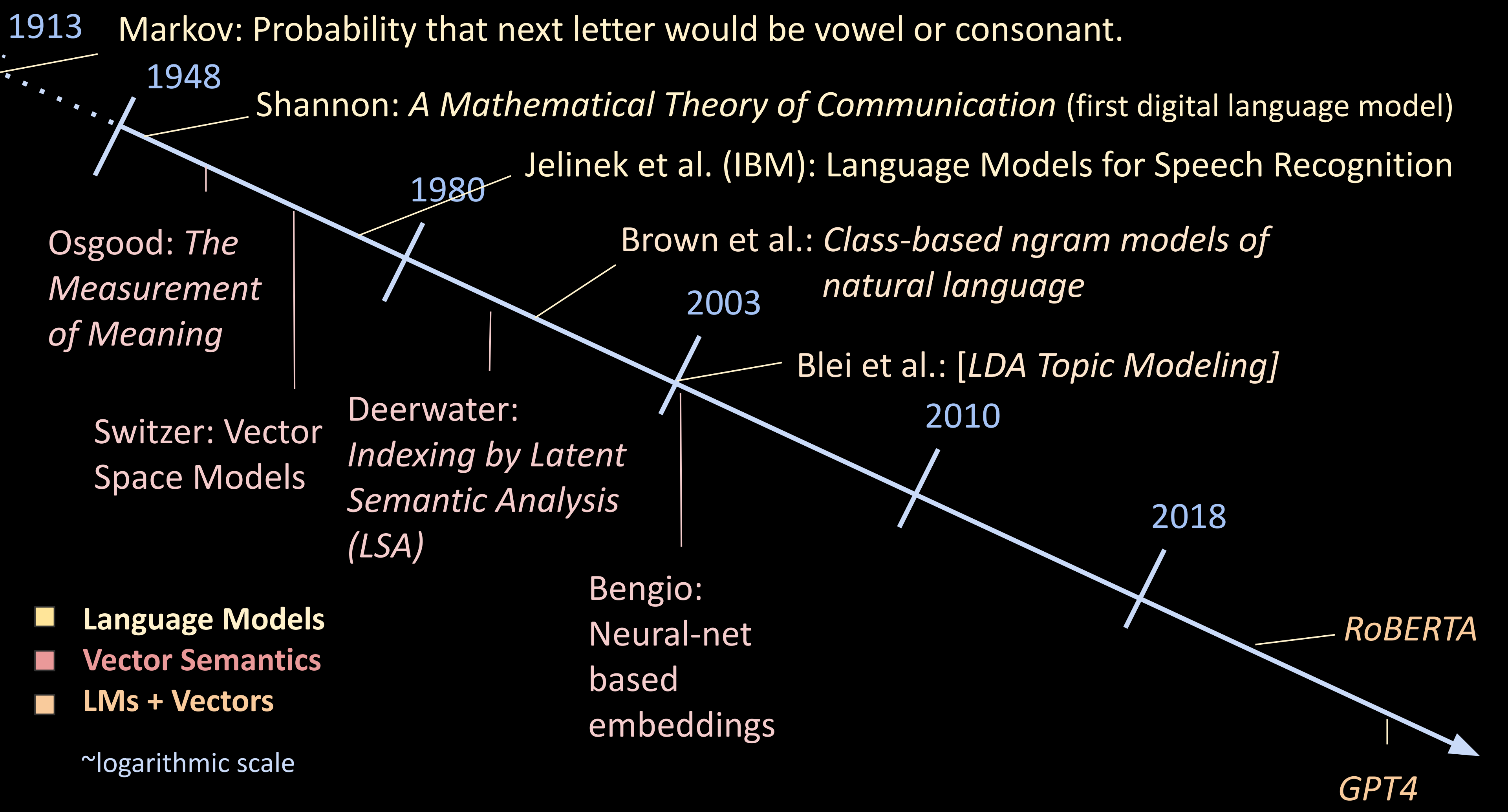
~logarithmic scale

GPT3.5

Timeline: *Language Modeling* and *Vector Semantics*



Timeline: *Language Modeling* and *Vector Semantics*



Timeline: *Language Modeling* and *Vector Semantics*

1913 Markov: Probability that next letter would be vowel or consonant.

1948 Shannon: *A Mathematical Theory of Communication* (first digital language model)

Osgood: *The Measurement of Meaning*

Switzer: Vector Space Models

1980 Jelinek et al. (IBM): Language Models for Speech Recognition

Deerwater: *Indexing by Latent Semantic Analysis (LSA)*

Brown et al.: *Class-based ngram models of natural language*

2003 Blei et al.: [*LDA Topic Modeling]*

Bengio: Neural-net based embeddings

Collobert and Weston: *A unified architecture for natural language processing: Deep neural networks...*

2010 Mikolov: *word2vec*

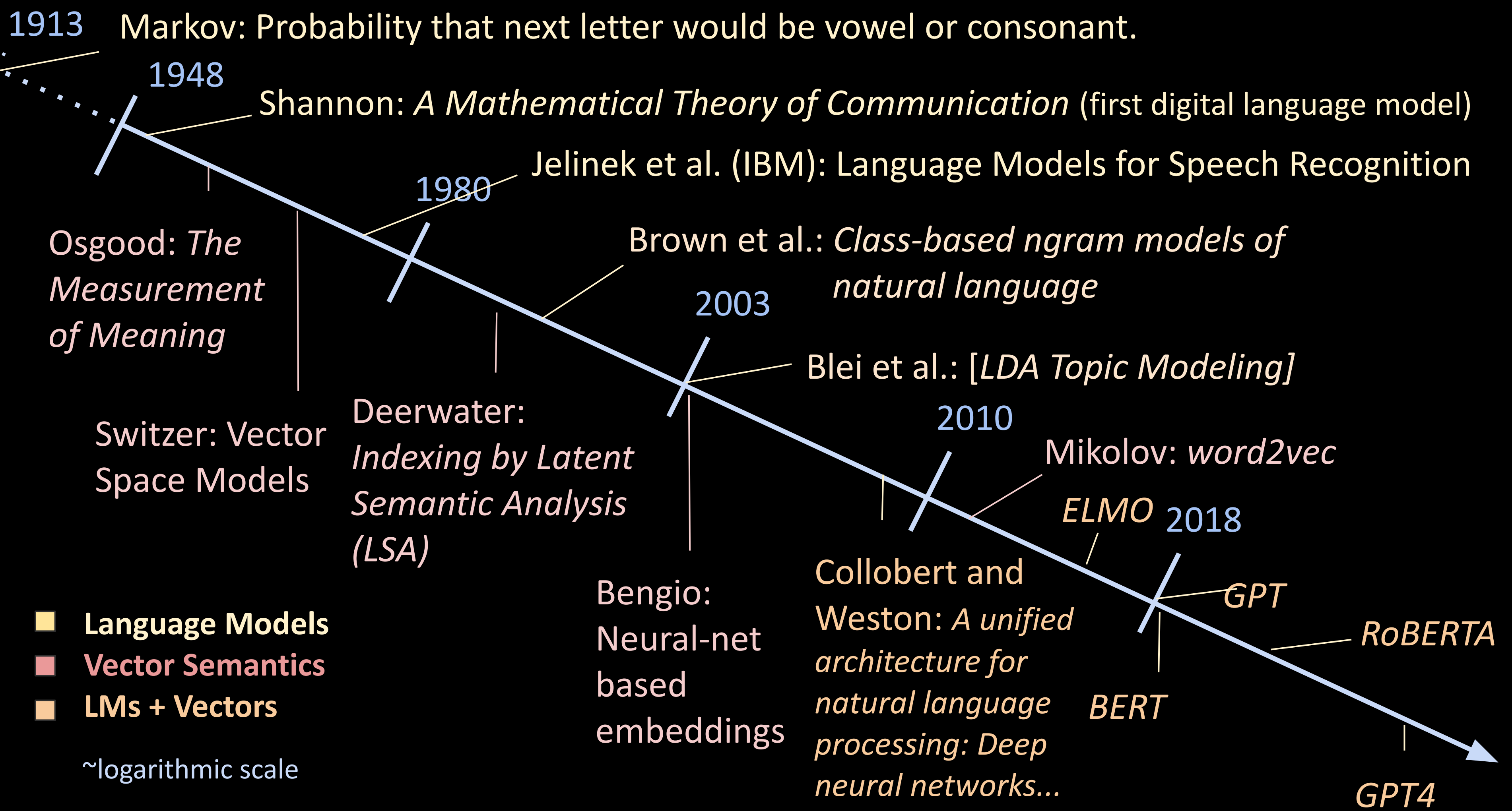
2018 *RoBERTA*

GPT4

- **Language Models**
- **Vector Semantics**
- **LMs + Vectors**

~logarithmic scale

Timeline: *Language Modeling* and *Vector Semantics*



Language Modeling

Version 1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

:probability of a sequence of words

Version 2: Compute $P(w5| w1, w2, w3, w4)$

$= P(w_n| w_1, w_2, ..., w_{n-1})$

:probability of a next word given history

Simple Solution

Version 1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

:probability of a sequence of words

$P(\text{He ate the cake with the fork}) =$

$$\frac{count(\text{He ate the cake with the fork})}{count(* \quad * \quad * \quad * \quad * \quad * \quad *)}$$

Simple Solution: The Maximum Likelihood Estimate

Version 1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

:probability of a sequence of words

$P(\text{He ate the cake with the fork}) =$

$$\frac{count(\text{He ate the cake with the fork})}{count(* \quad * \quad * \quad * \quad * \quad * \quad *)}$$

total number of observed *7grams*

Simple Solution: The Maximum Likelihood Estimate

V1:

$$P(\textit{He ate the cake with the fork}) = \frac{count(\textit{He ate the cake with the fork})}{count(* \quad * \quad * \quad * \quad * \quad * \quad *)}$$

V2:

$$P(\textit{fork} \mid \textit{He ate the cake with the}) = \frac{count(\textit{He ate the cake with the fork})}{count(\textit{He ate the cake with the } *)}$$

Simple Solution: The Maximum Likelihood Estimate

Problem: even the Web isn't large enough to enable good estimates of most phrases.

V1:

$$P(\textit{He ate the cake with the fork}) = \frac{count(\textit{He ate the cake with the fork})}{count(* \quad * \quad * \quad * \quad * \quad * \quad *)}$$

V2:

$$P(\textit{fork} \mid \textit{He ate the cake with the}) = \frac{count(\textit{He ate the cake with the fork})}{count(\textit{He ate the cake with the } *)}$$

Simple Solution: The Maximum Likelihood Estimate

Problem: even the Web isn't large enough to enable good estimates of most phrases.

V1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

V2: Compute $P(w5 | w1, w2, w3, w4) = P(w_n | w_1, w_2, ..., w_{n-1})$

A solution: Estimate from shorter sequences.

Simple Solution: The Maximum Likelihood Estimate

Problem: even the Web isn't large enough to enable good estimates of most phrases.

V1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

V2: Compute $P(w5 | w1, w2, w3, w4) = P(w_n | w_1, w_2, \ldots, w_{n-1})$

A solution: Estimate from shorter sequences.

Observation: V1 and V2 are equivalent!

Language Modeling: How to Estimate?

V1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

V2: Compute $P(w5| w1, w2, w3, w4) = P(w_n | w_1, w_2, ..., w_{n-1})$

Observation: V1 and V2 are equivalent!

Language Modeling: How to Estimate?

V1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

V2: Compute $P(w5 | w1, w2, w3, w4) = P(w_n | w_1, w_2, \ldots, w_{n-1})$

Observation: V1 and V2 are equivalent!

$$P(B|A) = P(B, A) \,/\, P(A) \Leftrightarrow P(A)P(B|A) = P(B,A)$$

Language Modeling: How to Estimate?

V1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

V2: Compute $P(w5 | w1, w2, w3, w4) = P(w_n | w_1, w_2, ..., w_{n-1})$

Observation: V1 and V2 are equivalent!

$$P(B|A) = P(B, A) / P(A) \Leftrightarrow P(A)P(B|A) = P(B,A) = P(A,B)$$

$$P(A,B) = P(A)P(B|A)$$

Language Modeling: How to Estimate?

V1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

V2: Compute $P(w5 | w1, w2, w3, w4) = P(w_n | w_1, w_2, ..., w_{n-1})$

Observation: V1 and V2 are equivalent!

$$P(A,B) = P(A)P(B|A)$$

Language Modeling: How to Estimate?

V1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

V2: Compute $P(w5 | w1, w2, w3, w4) = P(w_n | w_1, w_2, ..., w_{n-1})$

Observation: V1 and V2 are equivalent!

$$P(A,B) = P(A)P(B|A)$$

$$P(A, B, C) = P(A)P(B|A)P(C| A, B)$$

Language Modeling: How to Estimate?

V1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

V2: Compute $P(w5 | w1, w2, w3, w4) = P(w_n | w_1, w_2, \ldots, w_{n-1})$

Observation: V1 and V2 are equivalent!

$$P(A,B) = P(A)P(B|A)$$

$$P(A, B, C) = P(A)P(B|A)P(C| A, B)$$

The Chain Rule:

$$P(X1, X2, \ldots, Xn) = P(X1)P(X2|X1)P(X3|X1, X2)\ldots P(Xn|X1, \ldots, Xn-1)$$

Language Modeling: How to Estimate?

V1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

V2: Compute $P(w5| w1, w2, w3, w4) = P(w_n | w_1, w_2, ..., w_{n-1})$

Observation: V1 and V2 are equivalent!

$P(A,B) = P(A)P(B|A)$

$P(A, B, C) = P(A)P(B|A)P(C| A, B)$

The Chain Rule: $P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|X_1, X_2, ..., X_{i-1})$

$P(X1, X2,..., Xn) = P(X1)P(X2|X1)P(X3|X1, X2)...P(Xn|X1, ..., Xn-1)$

Language Modeling: How to Estimate?

V1: Compute $P(w1, w2, w3, w4, w5) = P(W)$

V2: Compute $P(w5| w1, w2, w3, w4) = P(w_n | w_1, w_2, ..., w_{n-1})$

Observation: Solving V2 give us V1!

$P(A,B) = P(A)P(B|A)$

$P(A, B,$

LM version 1 LM version 2

$$\underline{P(X1, X2,..., Xn)} = \underline{P(X1, X2,..., Xn\text{-}1)}\underline{P(Xn|X1, ..., Xn\text{-}1)}$$

The Ch

$i=1$

$$P(X1, X2,..., Xn) = P(X1)P(X2|X1)P(X3|X1, X2)...P(Xn|X1, ..., Xn\text{-}1)$$

Language Modeling: How to Estimate?

Compute $P(w5 | w1, w2, w3, w4) = P(w_n | w_1, w_2, ..., w_{n-1})$

Problem: even the Web isn't large enough to enable good estimates of most phrases.

A solution: Estimate from shorter sequences.

Chain-Rule:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i | X_1, X_2, ..., X_{i-1})$$

Language Modeling: How to Estimate?

Compute $P(w5 | w1, w2, w3, w4) = P(w_n | w_1, w_2, \ldots, w_{n-1})$

Problem: even the Web isn't large enough to enable good estimates of most phrases.

A solution: Estimate from shorter sequences.

Chain-Rule:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_1, X_2, \ldots, X_{i-1})$$

Markov Assumption:

$P(Xn | X1\ldots, Xn\text{-}1) \approx P(Xn | Xn\text{-}k, \ldots, Xn\text{-}1)$ where $k < n$

Language Modeling: How to Estimate?

Compute $P(w5 | w1, w2, w3, w4) = P(w_n | w_1, w_2, \ldots, w_{n-1})$

Problem: even the Web isn't large enough to enable good estimates of most phrases.

A solution: Estimate from shorter sequences.

Chain-Rule:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_1, X_2, \ldots, X_{i-1})$$

Markov Assumption:

$P(Xn | X1\ldots, Xn\text{-}1) \approx P(Xn | Xn\text{-}k, \ldots, Xn\text{-}1) \quad where\ k < n$

Language Modeling: How to Estimate?

Compute $P(w5 \mid w1, w2, w3, w4) = P(w_n \mid w_1, w_2, \ldots, w_{n-1})$

Unigram Model: k = 0; $$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i)$$

Language Modeling: How to Estimate?

Compute $P(w5|\ w1, w2, w3, w4) = P(w_n|\ w_1, w_2, \ldots, w_{n-1})$

Bigram Model: k = 1; $$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|X_{i-1})$$

Language Modeling: How to Estimate?

Compute $P(w5 | w1, w2, w3, w4) = P(w_n | w_1, w_2, \ldots, w_{n-1})$

Bigram Model: k = 1; $P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_{i-1})$

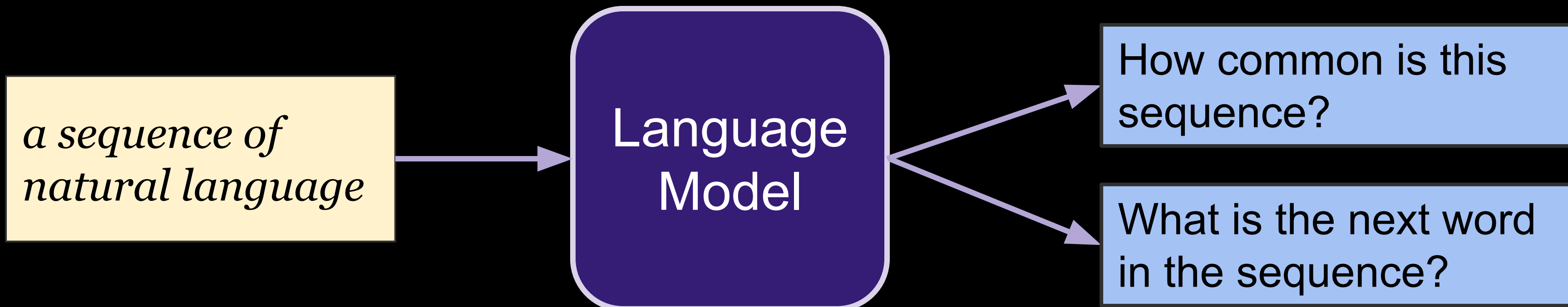
Example generated sentence:

outside, new, car, parking, lot, of, the, agreement, reached

$P(X1 = \text{"outside"}, X2 = \text{"new"}, X3 = \text{"car"}, \ldots)$
$\approx P(X1 = \text{"outside"}) * P(X2 = \text{"new"} | X1 = \text{"outside"}) * P(X3 = \text{"car"} | X2 = \text{"new"}) * \ldots$

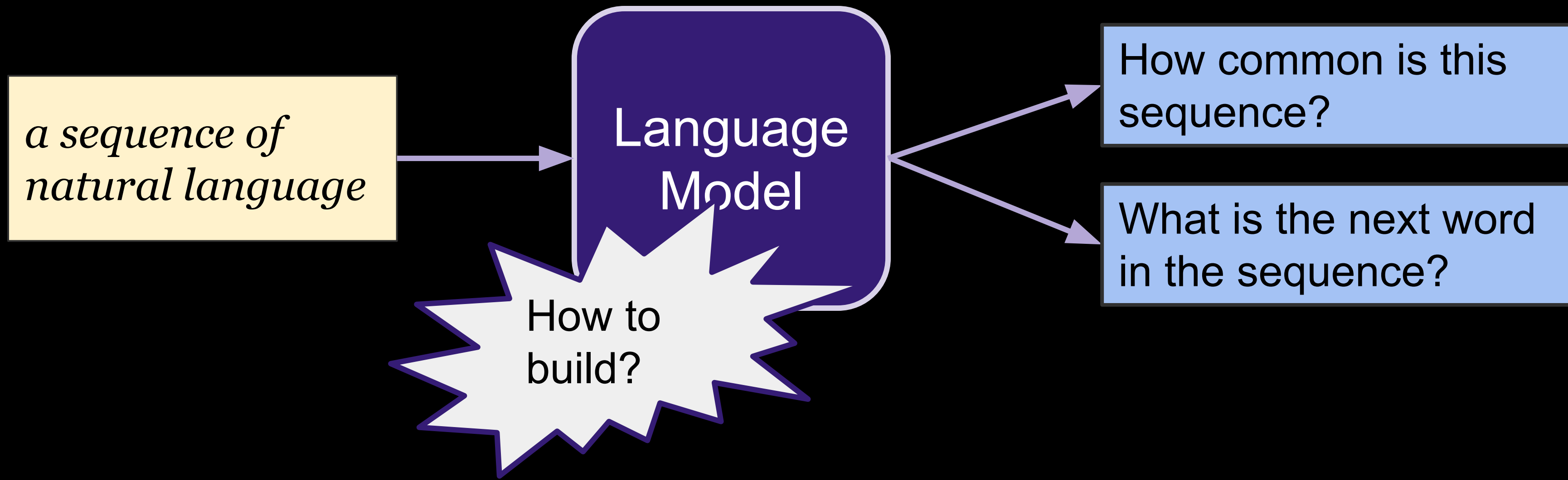
Language Modeling

Building a model (or system / API) that can answer the following:



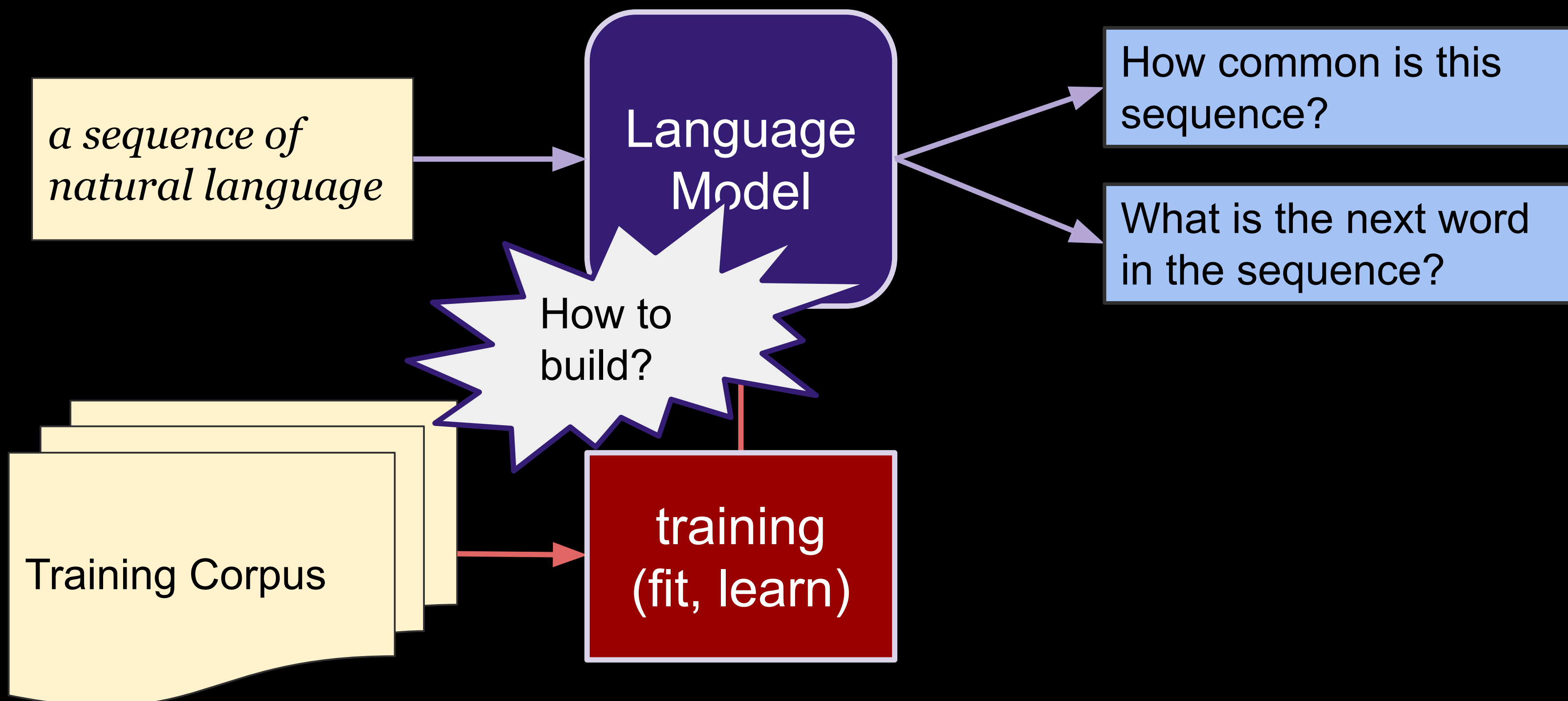
Language Modeling

Building a model (or system / API) that can answer the following:



Language Modeling

Building a model (or system / API) that can answer the following:



Language Mo

Building a model (

a sequence of natural language

Food corpus from Jurafsky (2018). Samples:

can you tell me about any good cantonese restaurants close by

mid priced thai food is what i'm looking for

tell me about chez panisse

can you give me a listing of the kinds of food that are available

i'm looking for a good place to eat breakfast

when is caffe venezia open during the day

Training Corpus

training (fit, learn)

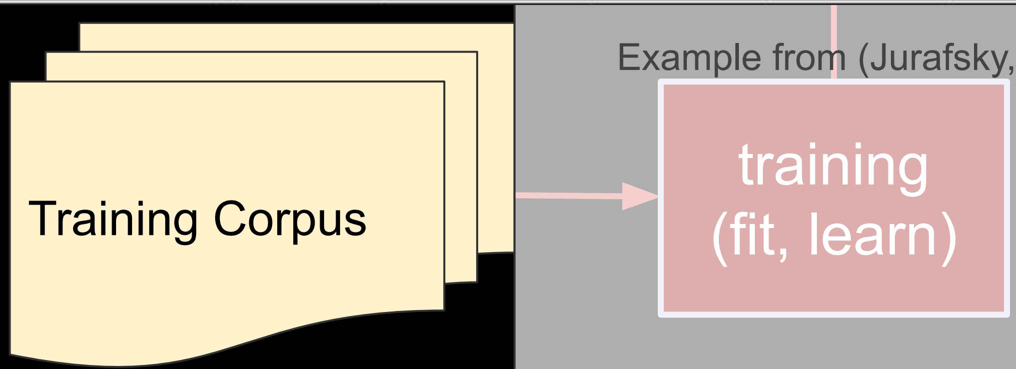
first word

second word

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

Example from (Jurafsky, 2017)



first word \ second word

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

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Training Corpus

training
(fit, learn)

first word

second word

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

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Training Corpus

training (fit, learn)

Bigram model: $P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_{i-1})$

Need to estimate: *P(Xi | Xi-1)* = count(Xi-1 Xi) / count(Xi-1)

second word: xi

first word: xi-1

$P(Xi \mid Xi\text{-}1)$

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Training Corpus

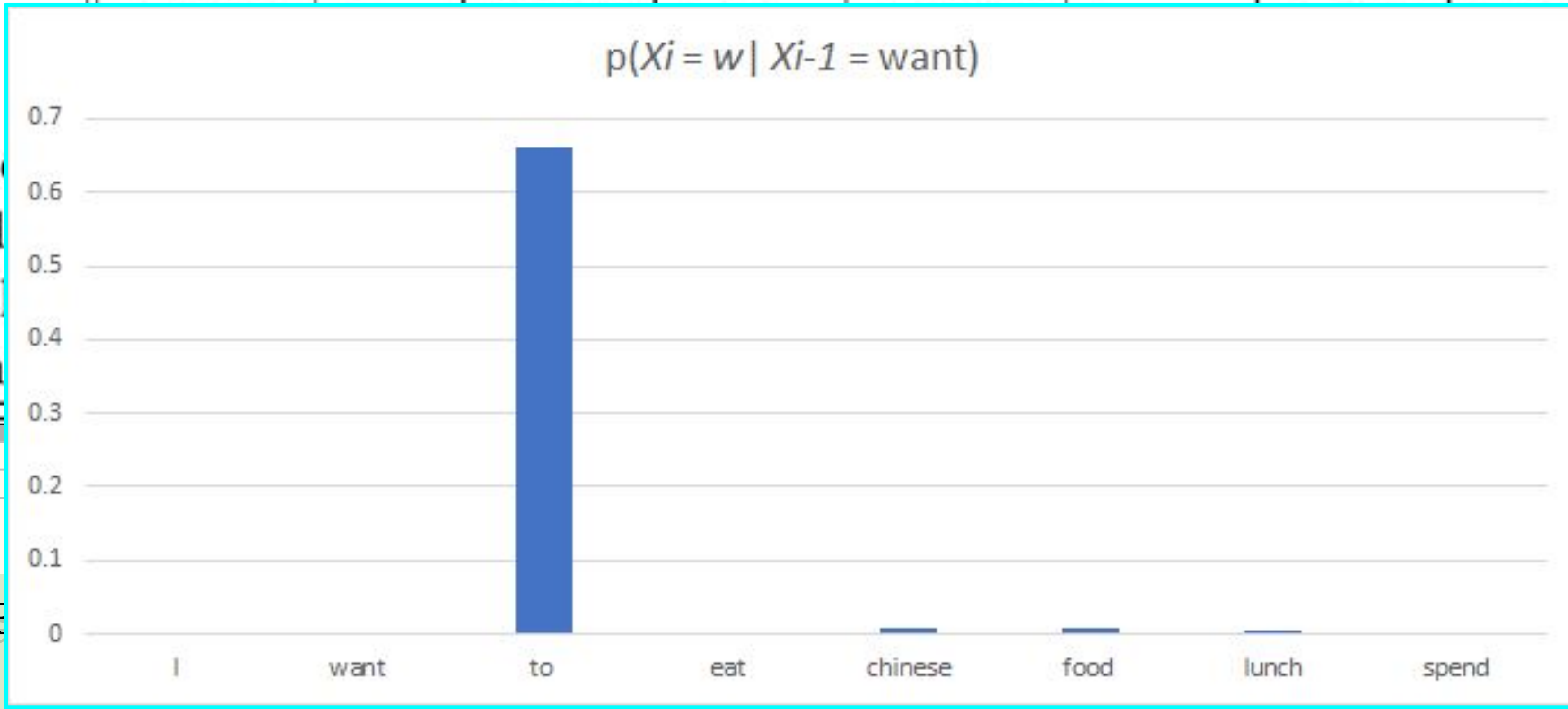
training (fit, learn)

Bigram model: $P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i | X_{i-1})$

Need to estimate: $P(Xi \mid Xi\text{-}1)$ = count(Xi-1 Xi) / count(Xi-1) first word(Xi-1) \ second word (Xi)

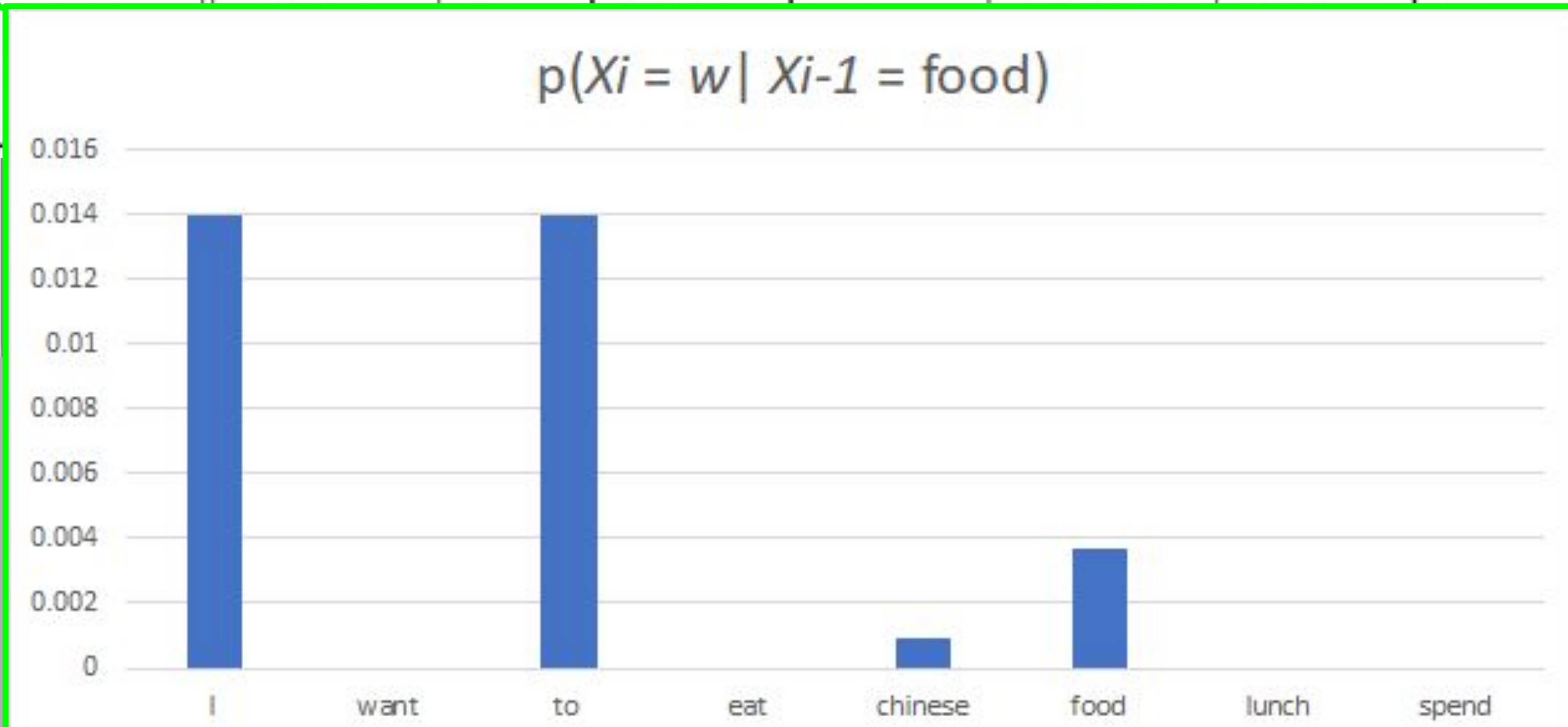
P(Xi | Xi-1)

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to								7
eat								
chin								
food								
lunc								
spen								



Need to estimate: *P(Xi | Xi-1)* = count(Xi-1 Xi) / count(Xi-1)

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
								0
								0



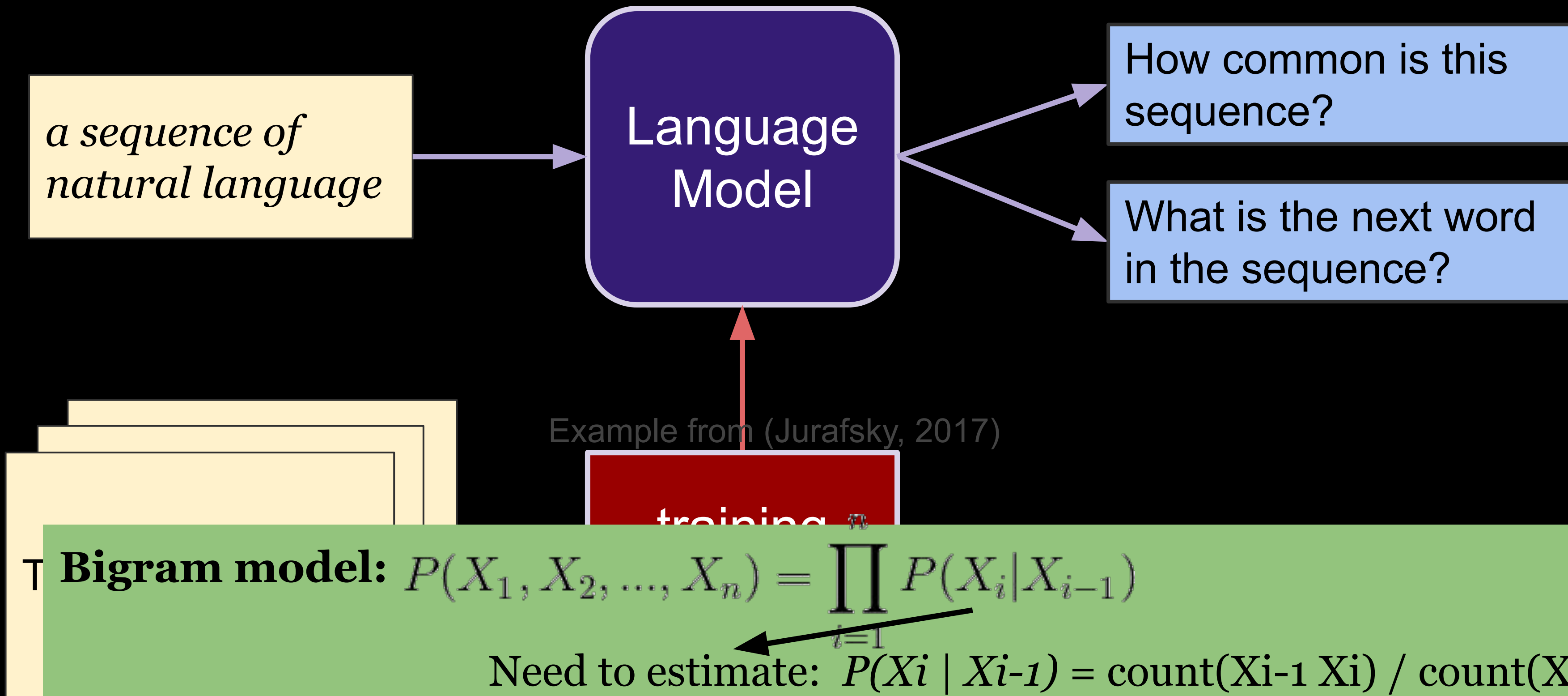
0
0
0

	spend
	278

ount(Xi-1)

Language Modeling

Building a model (or system / API) that can answer the following:



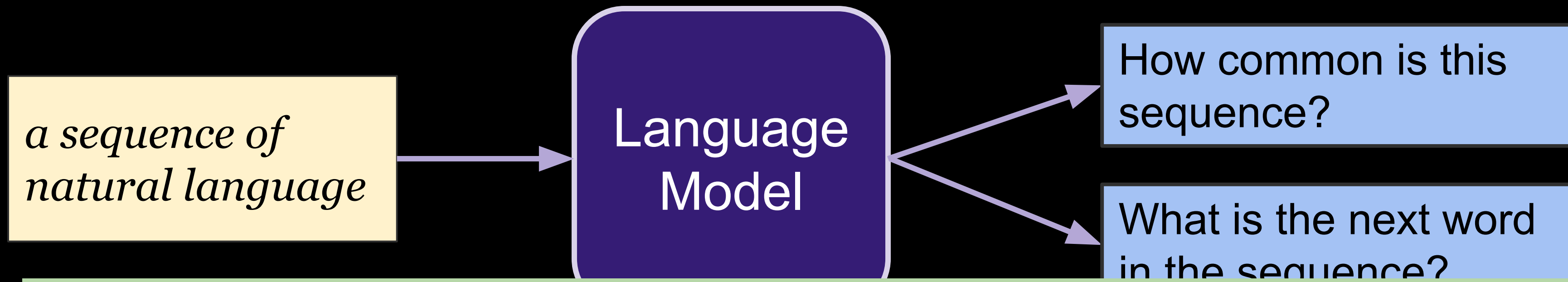
Example from (Jurafsky, 2017)

Bigram model: $P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|X_{i-1})$

Need to estimate: *P(Xi | Xi-1)* = count(Xi-1 Xi) / count(Xi-1)

Language Modeling

Building a model (or system / API) that can answer the following:



Trigram model: $P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i | \boxed{X_{i-2}, X_{i-1}})$

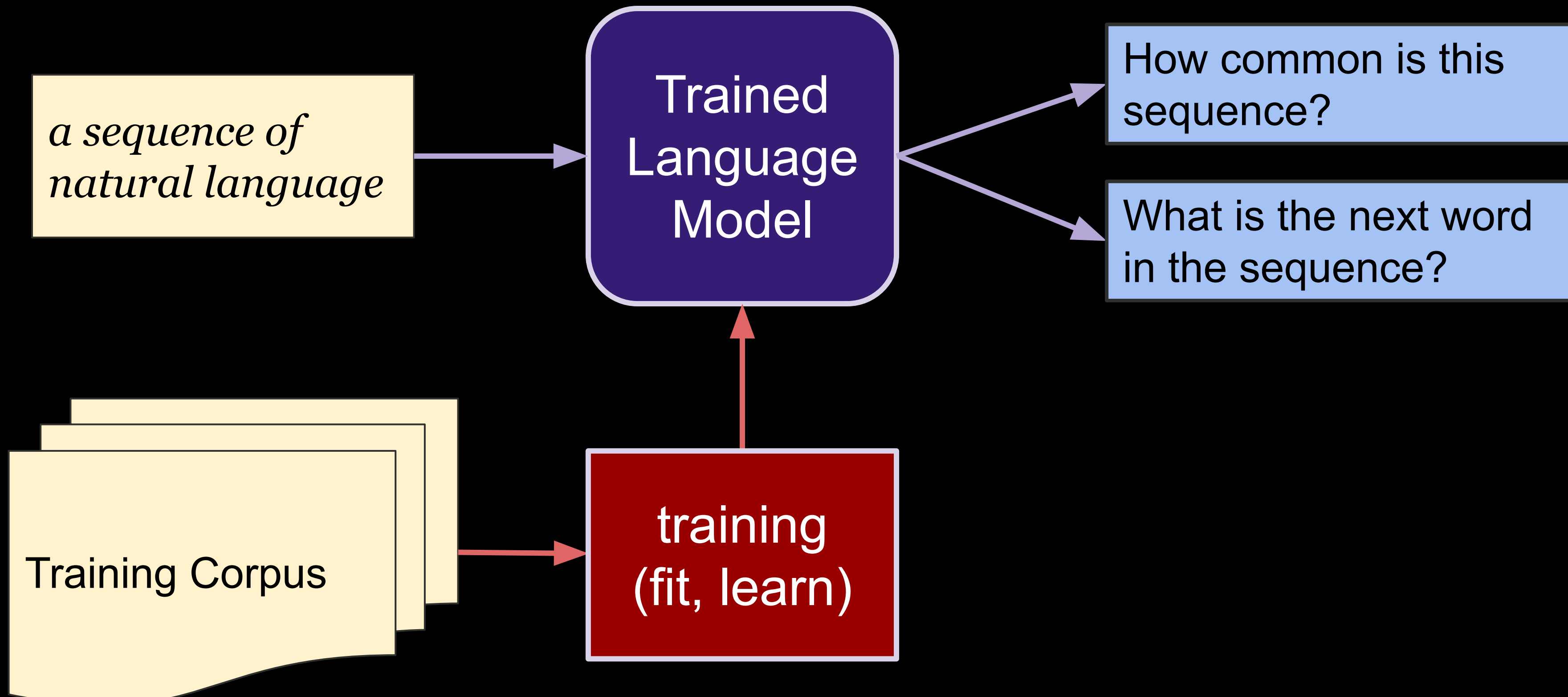
Need to estimate: *P(Xi | Xi-1, Xi-2)* = count(Xi-2 Xi-1 Xi) / count(Xi-2 Xi-1)

Bigram model: $P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i | X_{i-1})$

Need to estimate: *P(Xi | Xi-1)* = count(Xi-1 Xi) / count(Xi-1)

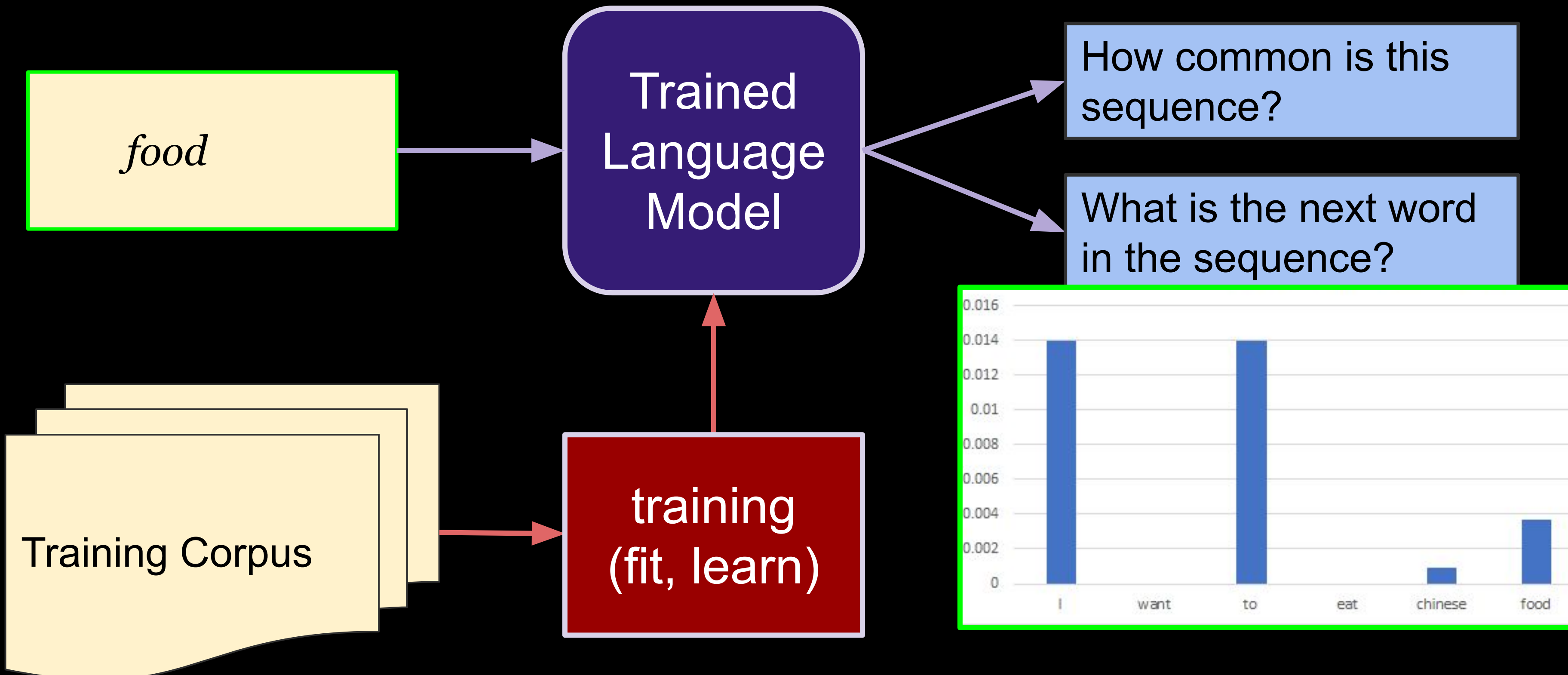
Language Modeling

Building a model (or system / API) that can answer the following:



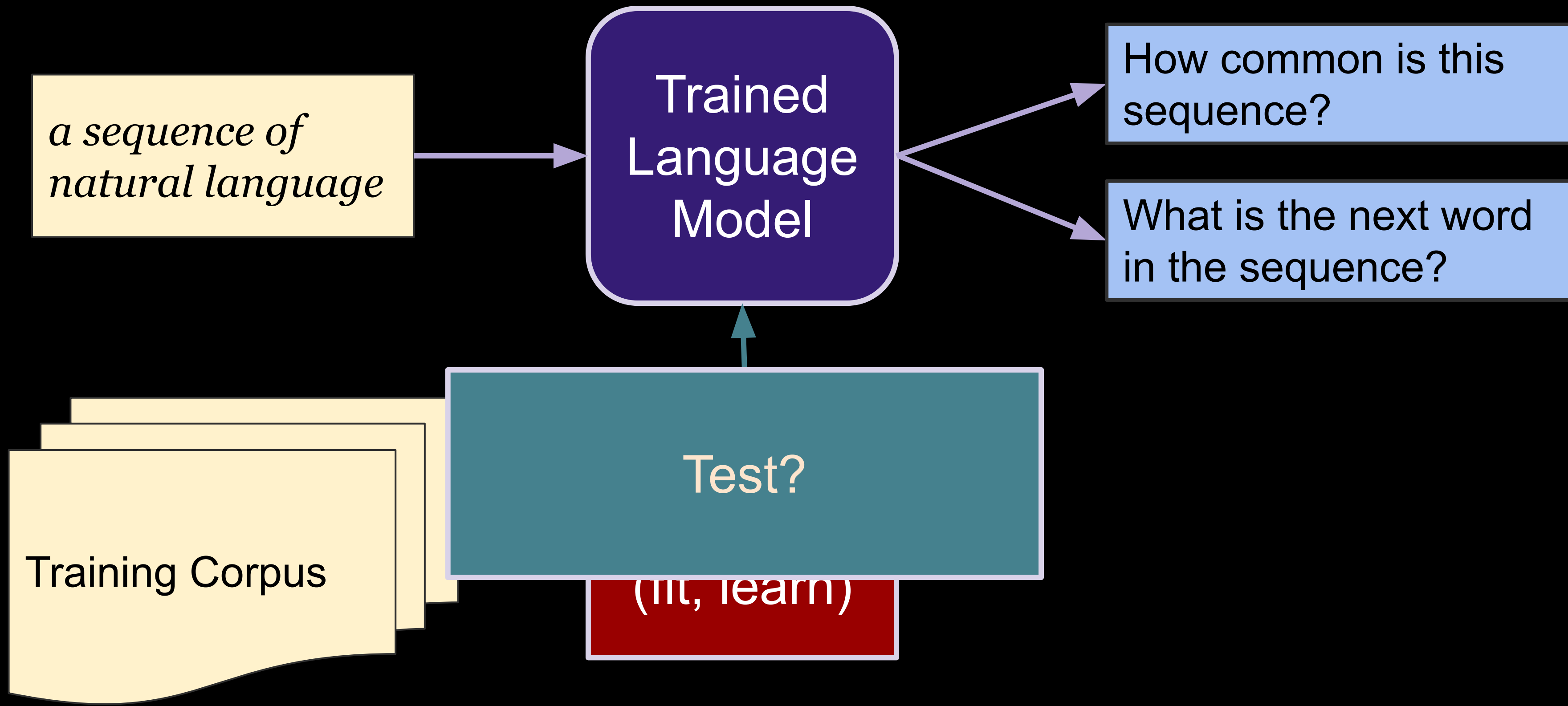
Language Modeling

Building a model (or system / API) that can answer the following:



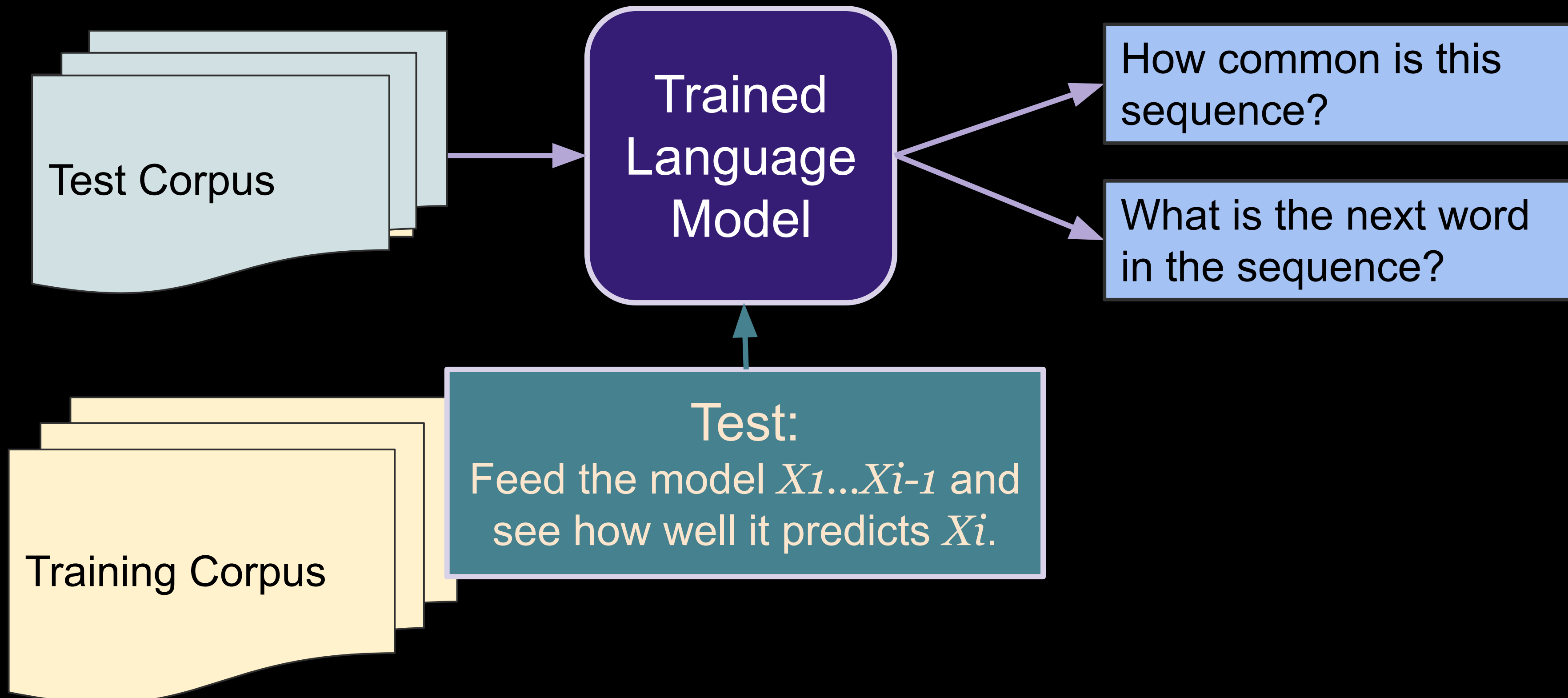
Language Modeling

Building a model (or system / API) that can answer the following:



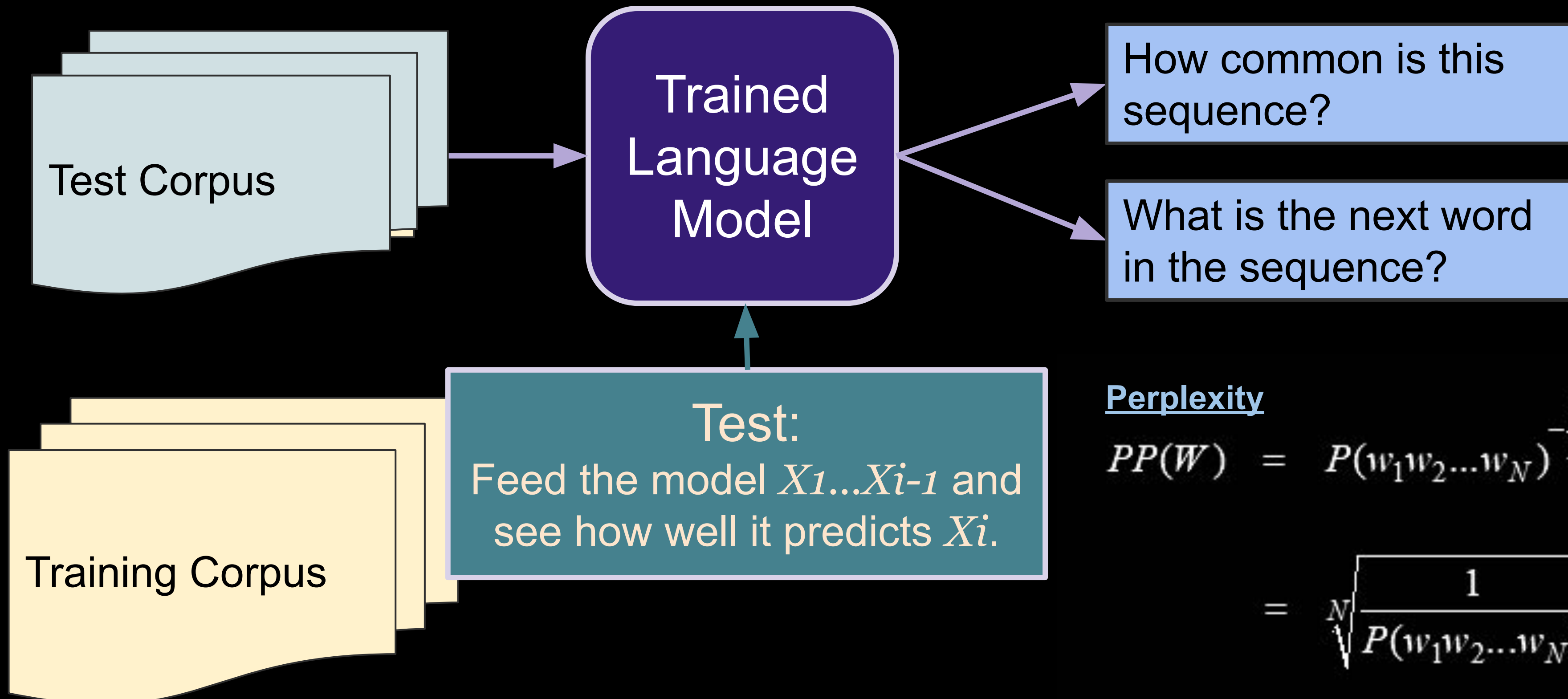
Language Modeling

Building a model (or system / API) that can answer the following:



Language Modeling

Building a model (or system / API) that can answer the following:

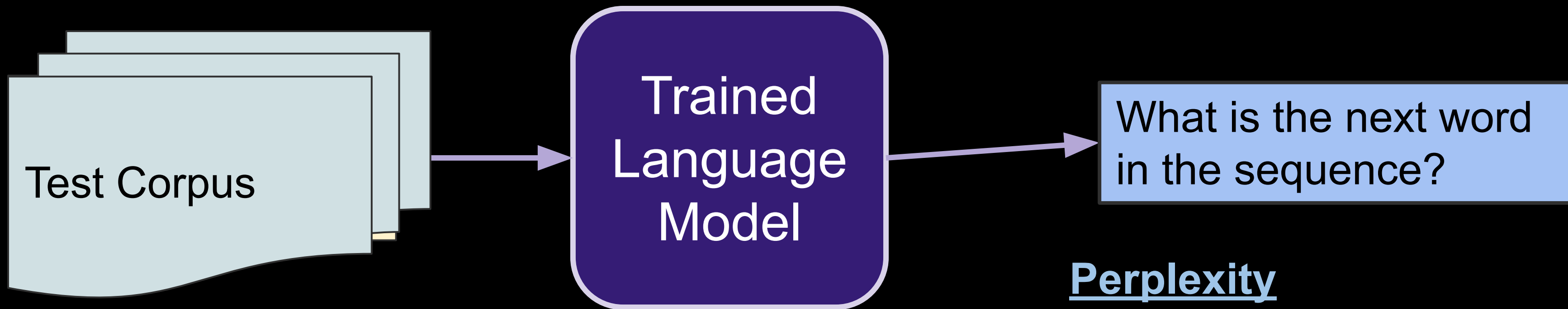


[Perplexity]

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

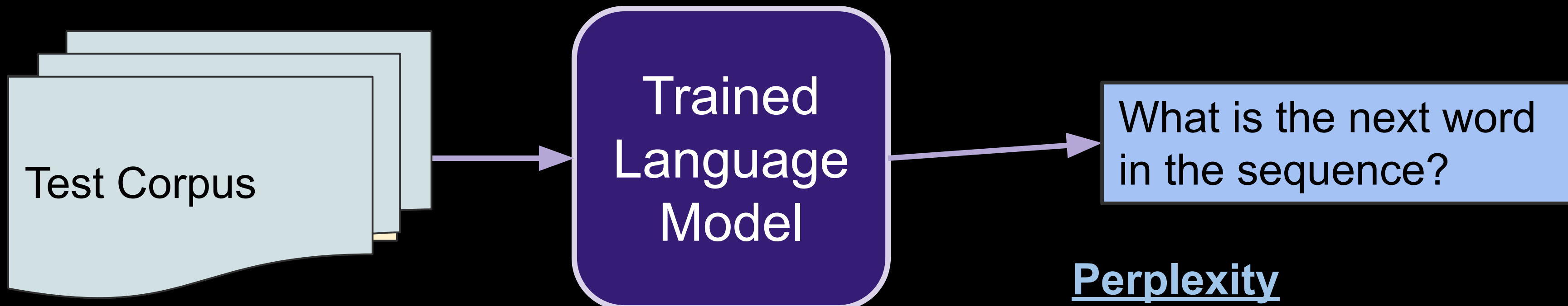
Evaluation



Perplexity

$$
\begin{aligned}
PP(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}
\end{aligned}
$$

Evaluation



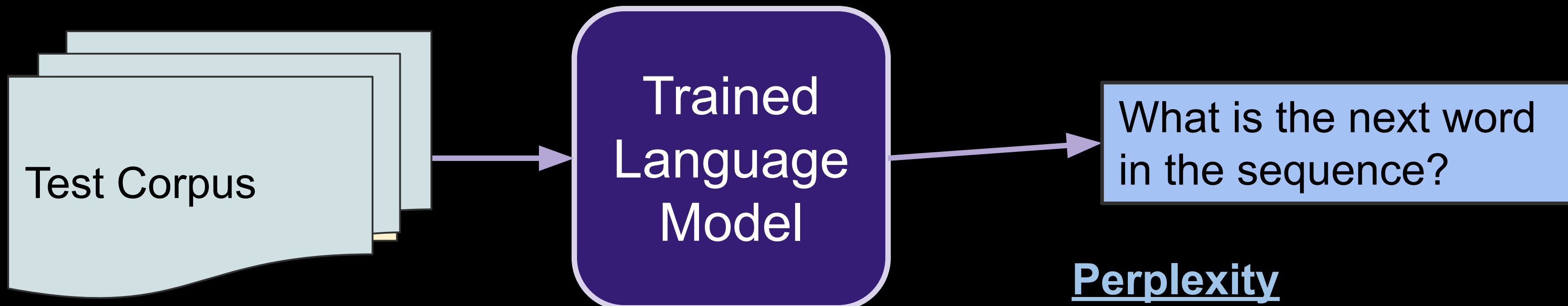
Perplexity

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

Apply Chain Rule: $$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \dots w_{i-1})}}$$

Evaluation



Perplexity

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

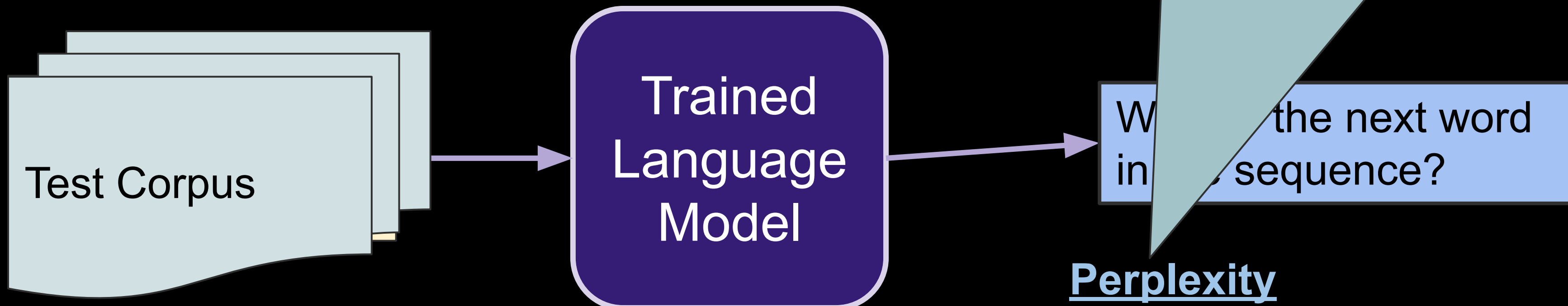
Apply Chain Rule: $$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

Thus,
PP for Bigrams: $$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

Evaluation

Reasoning:

1) Inverse of probability
   (i.e. minimize perplexity = maximize likelihood)
2) (weighted) average branching factor



Perplexity

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Apply Chain Rule: $$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

Thus,
PP for Bigrams:

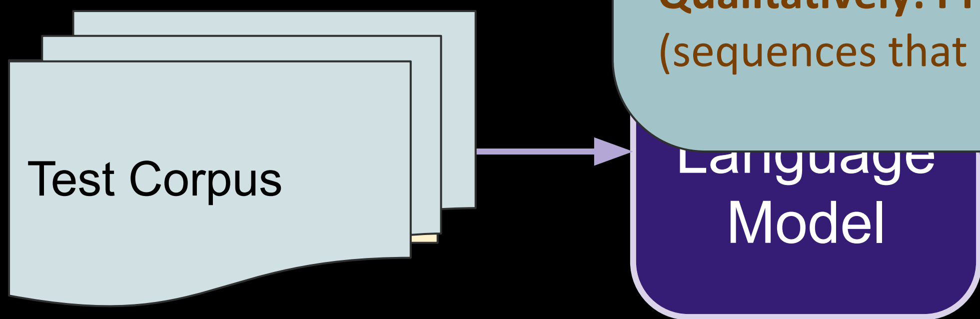
$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

Evaluation

Reasoning:

1) Inverse of probability
   (i.e. minimize perplexity = maximize likelihood)
2) (weighted) average branching factor

Qualitatively: Prefers real sentences
(sequences that are more grammatical, make sense).



Perplexity

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Apply Chain Rule: $\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$

Thus,
PP for Bigrams: $\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$

Evaluation Summary

- Use *training set* to "learn model"
  (i.e. to store counts, from which we can derive probability for any $p(w_i \mid w_{i-1}, w_{i-2})$
- Use held-out *testing set* to evaluate
- Perplexity -- metric for scoring how well learned model works on test.
  (an *intrinsic* evaluation)

Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Example from (Jurafsky, 2017)

Evaluation Summary

- Use *training set* to "learn model"
  (i.e. to store counts, from which we can derive probability for any $p(w_i \mid w_{i-1}, w_{i-2})$
- Use held-out *testing set* to evaluate
- Perplexity -- metric for scoring how well learned model works on test.
  (an *intrinsic* evaluation)

- *Extrinsic* evaluation: Test on task accuracies
  - machine translation: does it improve translation accuracy
  - autocomplete: do users like the suggestions
  - speech recognition: does it improve transcription accuracy
  - spelling corrector,
    etc…

Practical Considerations for LMs:

- Use log probability for assessing perplexity to keep numbers reasonable and save computation.
  (uses addition rather than multiplication)

- Use Out-of-vocabulary (OOV)
  Choose minimum frequency or total vocabulary size and mark as <OOV>

- Sentence start and end: *<s> this is a sentence </s>*
  Advantage: models word probability at beginning or end.