Introduction to Language Modeling

H. Andrew Schwartz

CSE538 - Spring 2024

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(He ate the cake with the fork) = ?

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

P(fork | 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

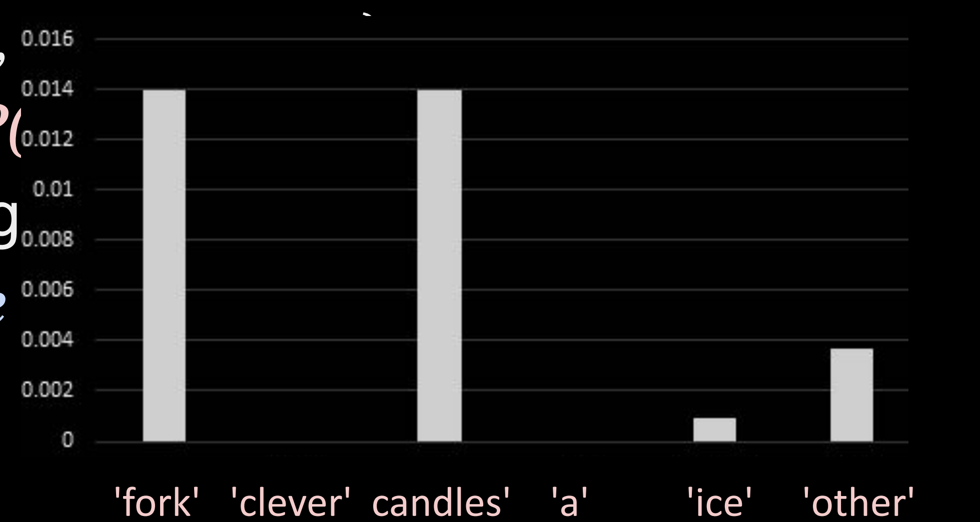
$P(\text{He ate the cake with the fork}) = ?$

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

$= P($

:probability of a next word g

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

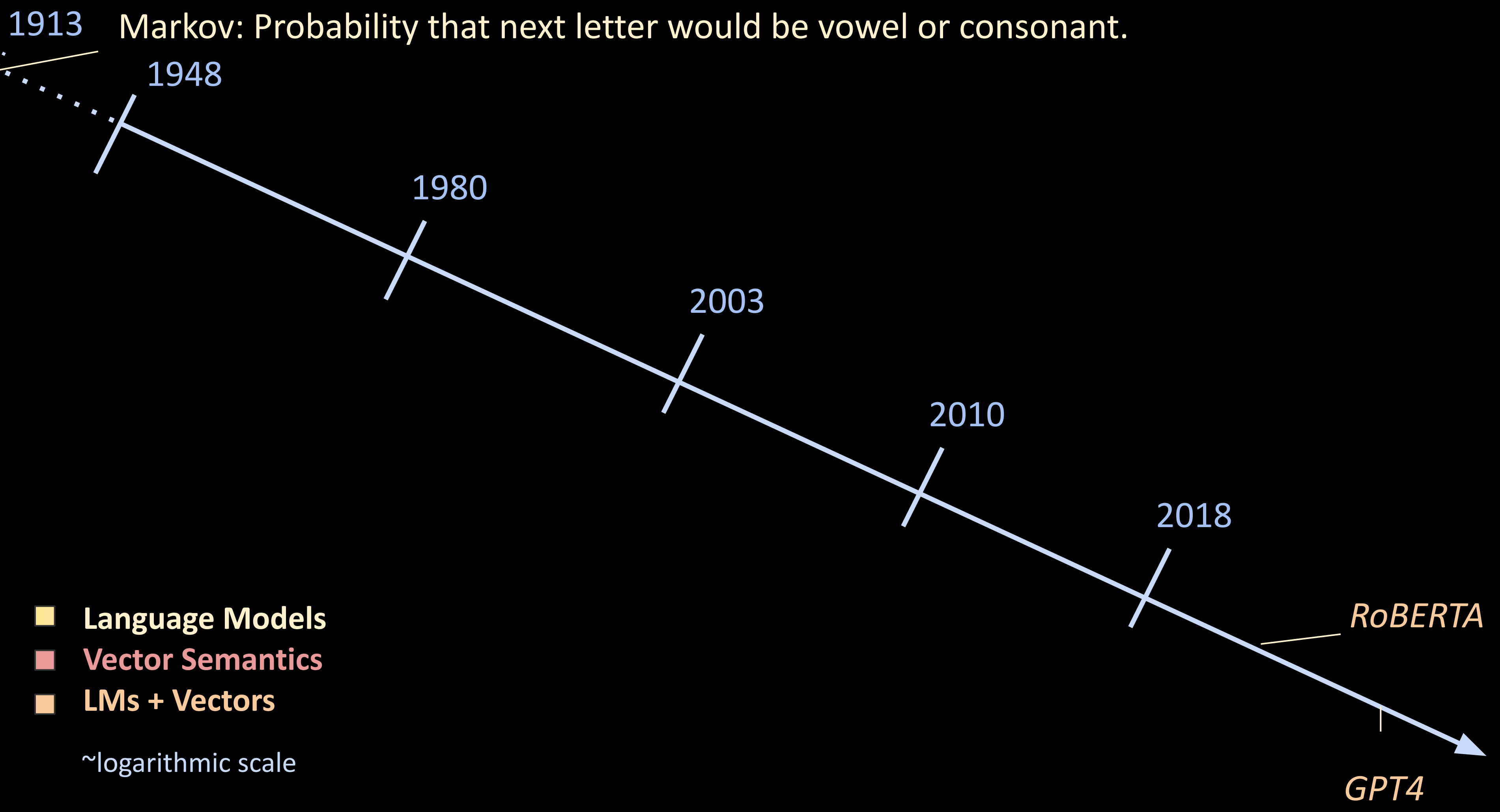


Language Modeling

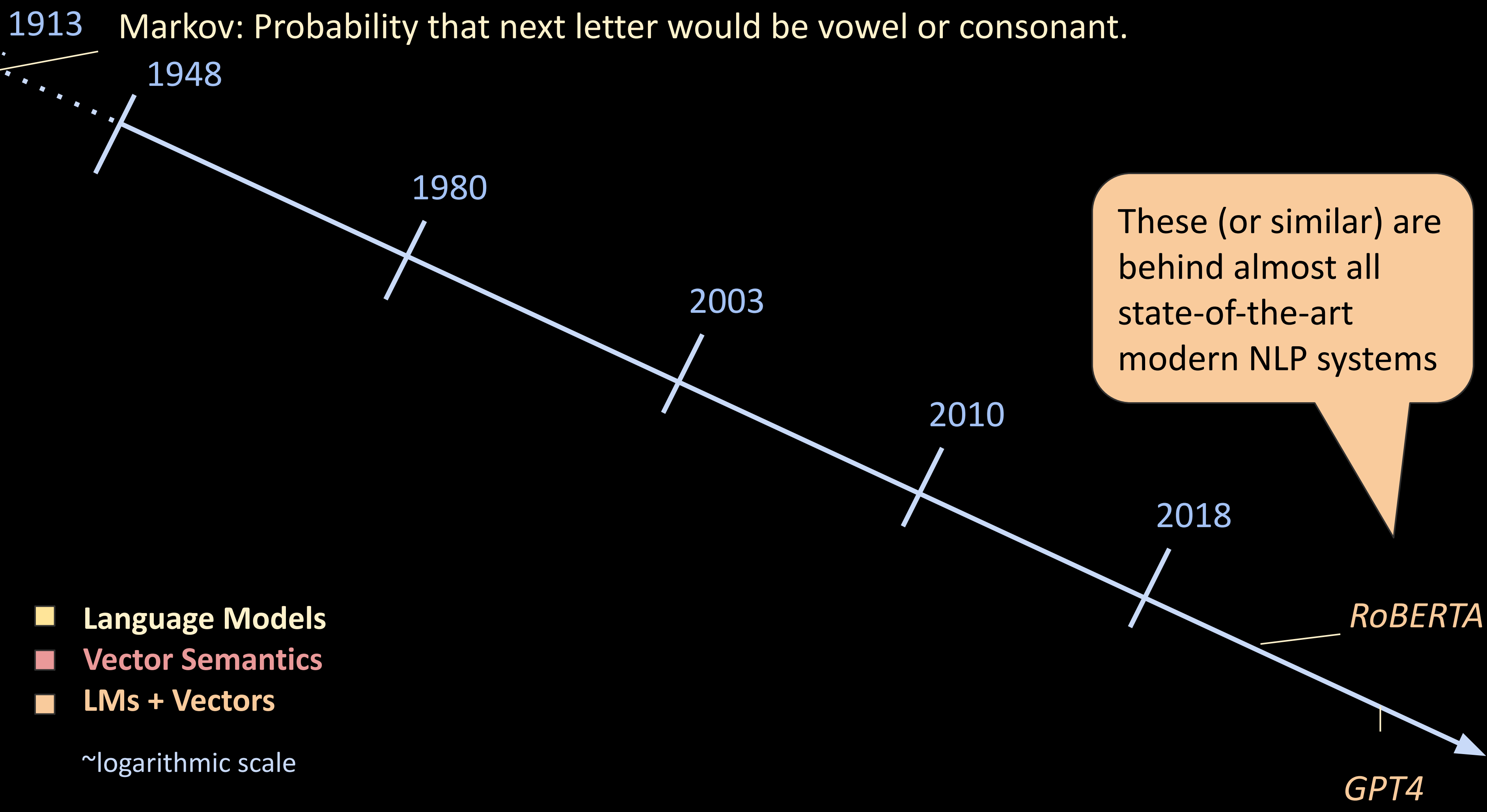
Applications:

- 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"?*)

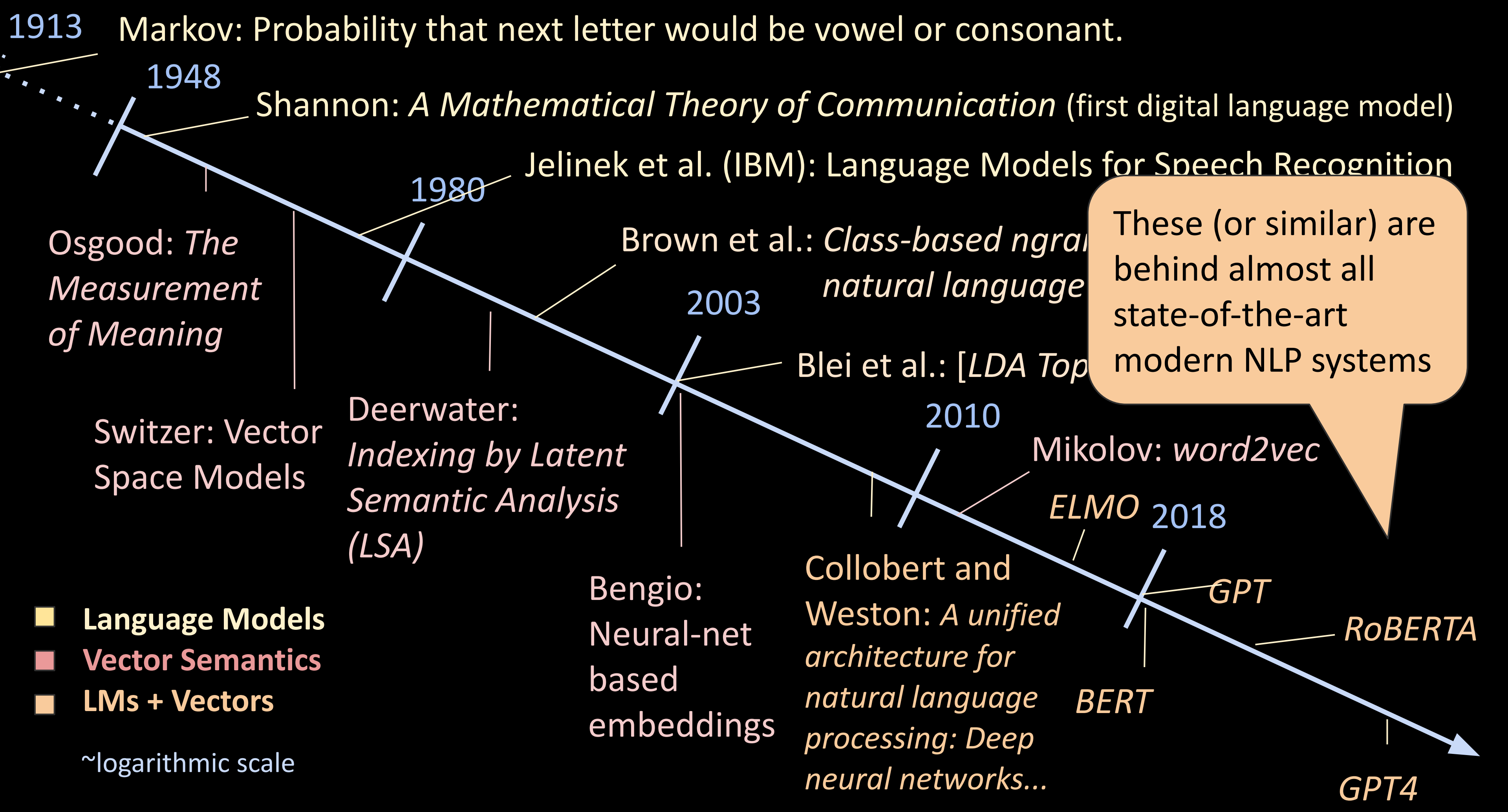
Timeline: *Language Modeling* and *Vector Semantics*



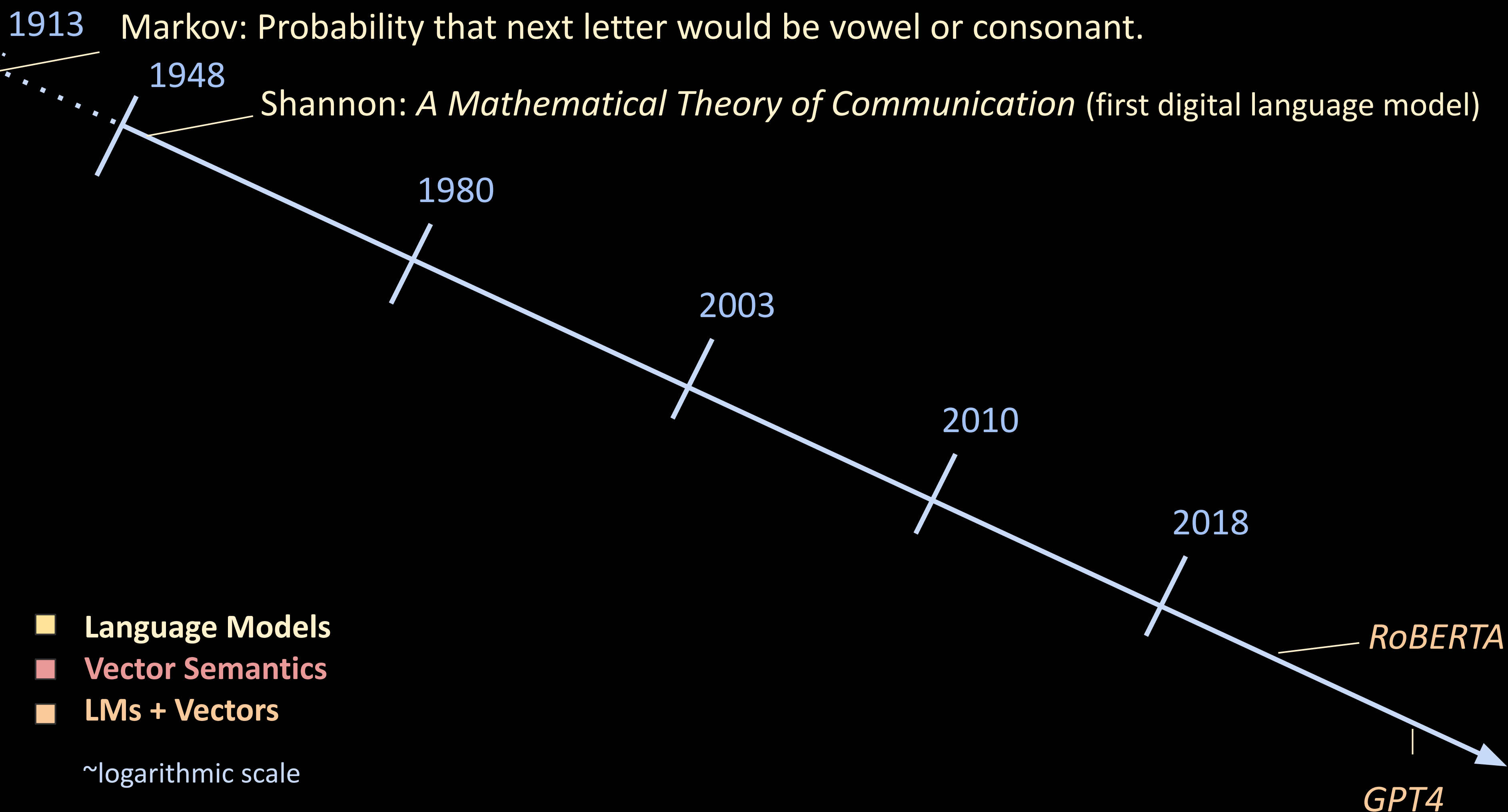
Timeline: *Language Modeling* and *Vector Semantics*



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*

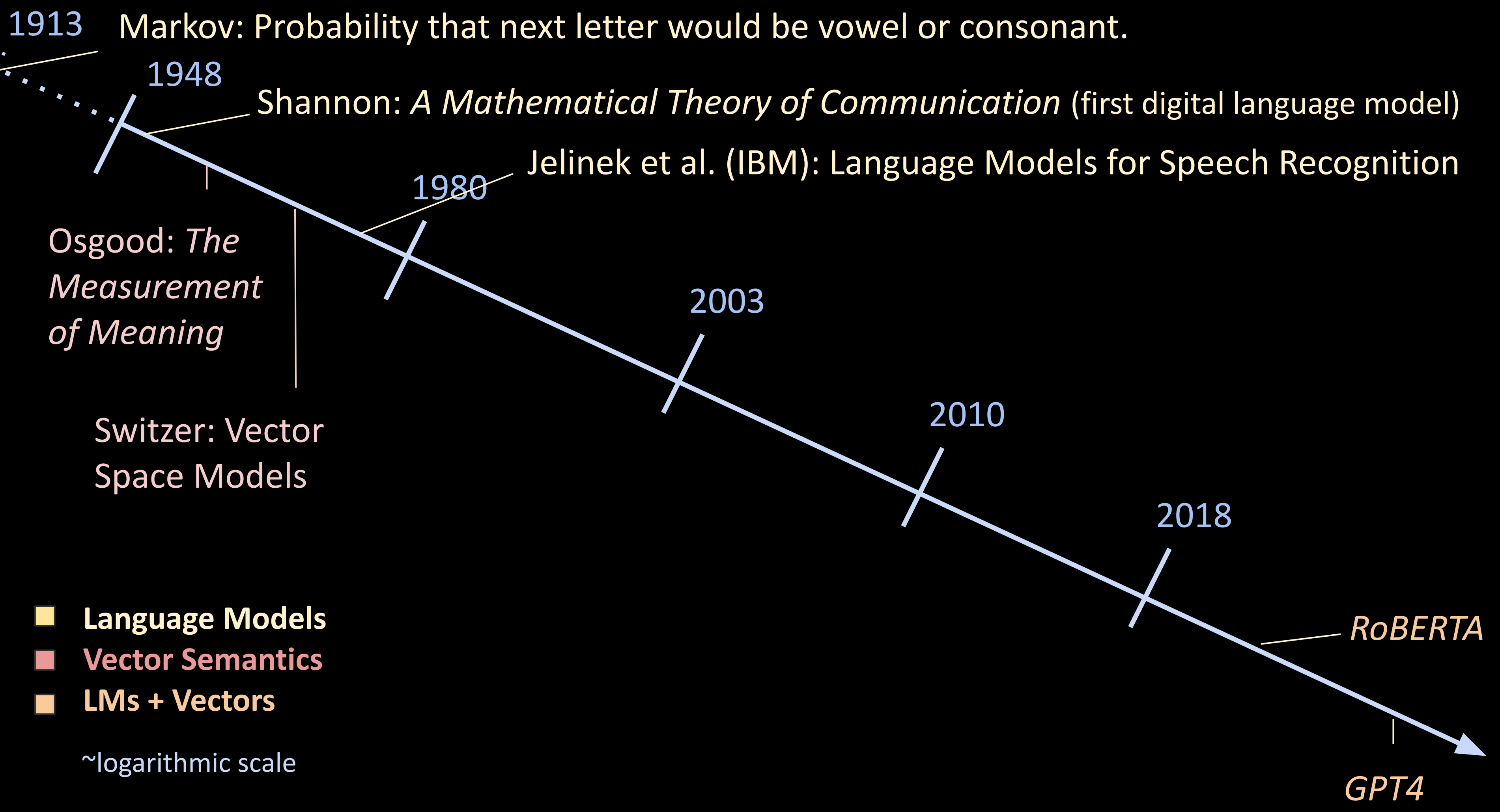
(Li et al. ,2015; Jurafsky et al., 2019)

- Language Models
- Vector Semantics
- LMs + Vectors

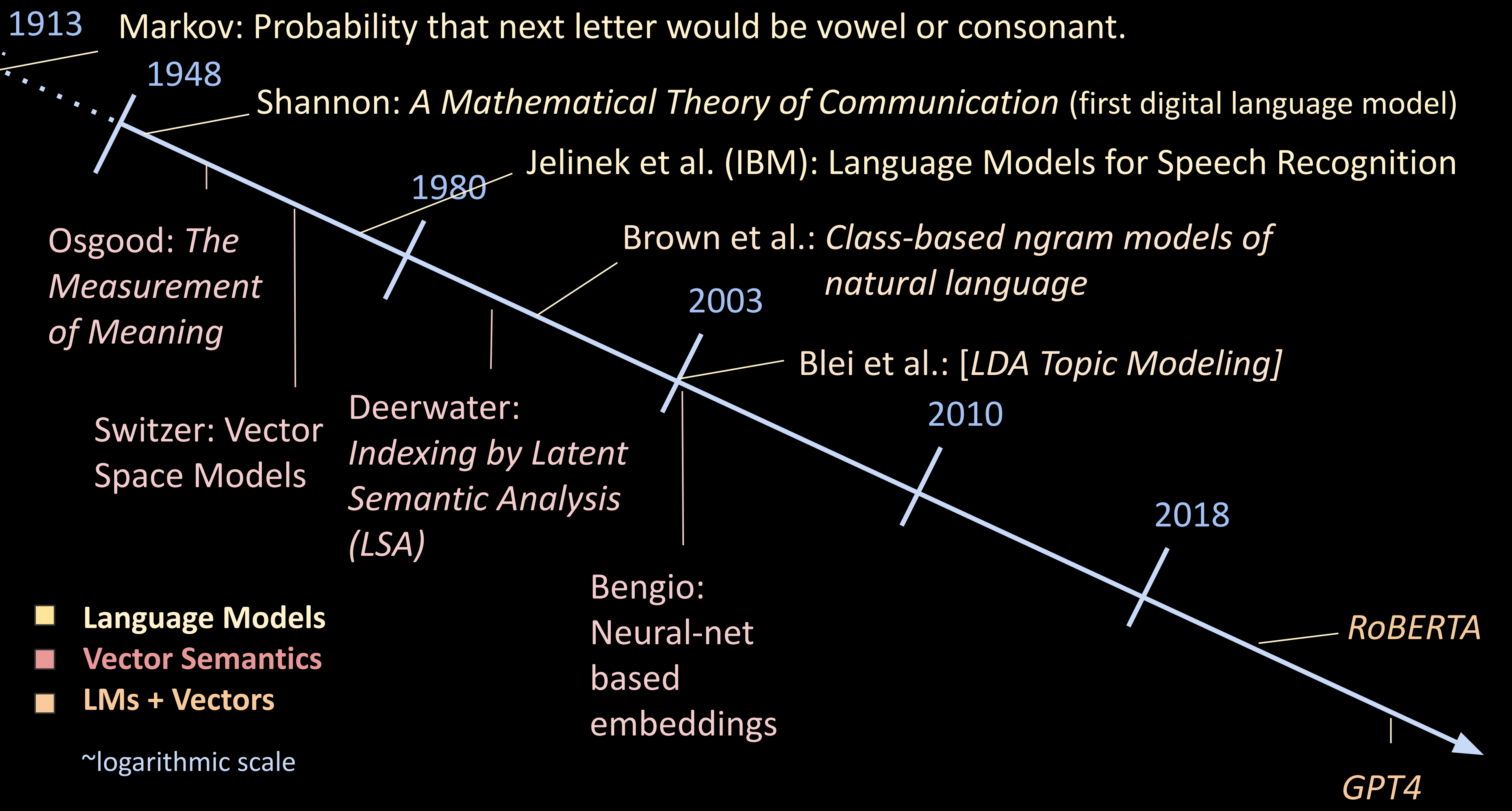
~logarithmic scale

GPT3.5

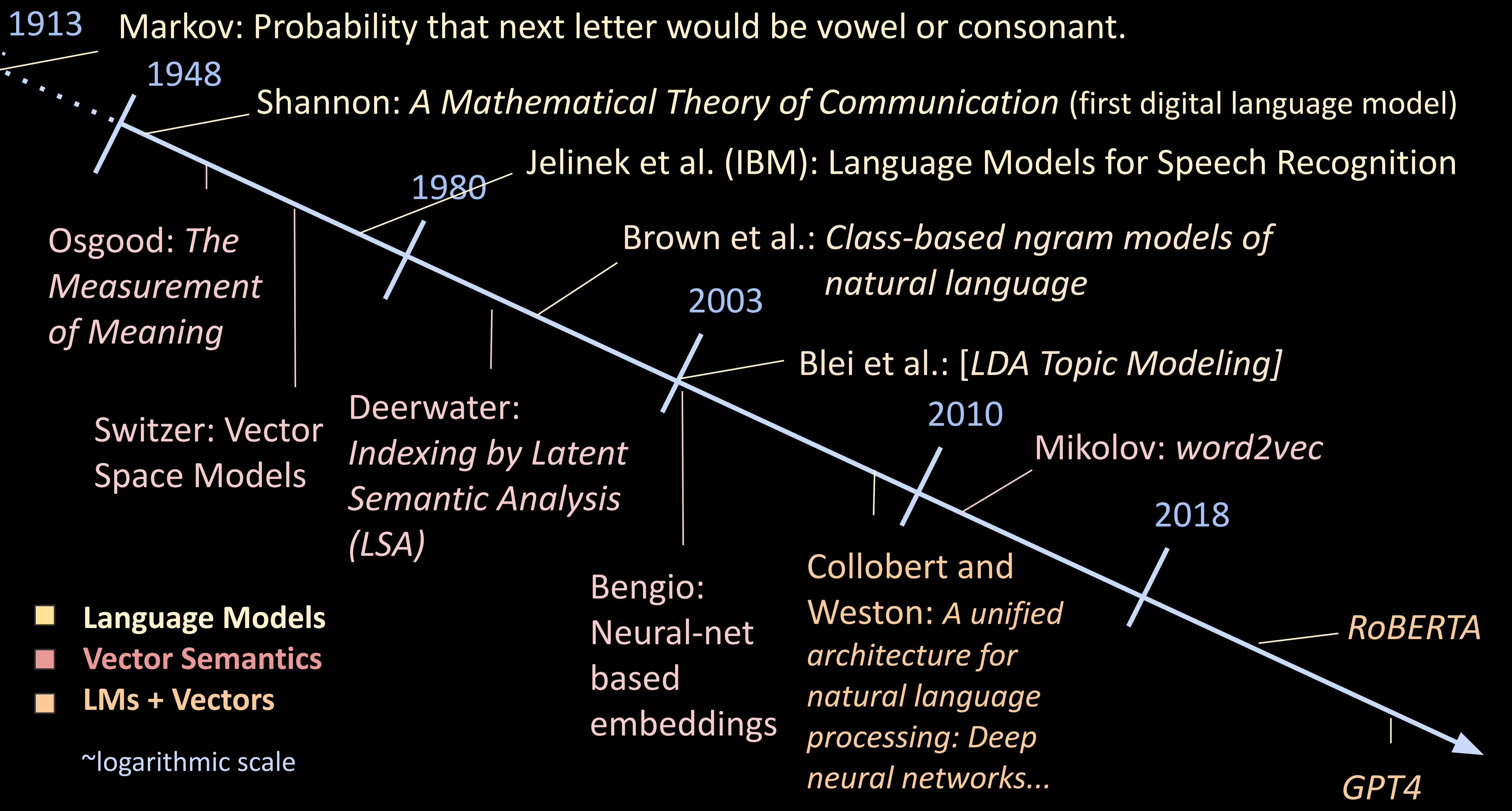
Timeline: *Language Modeling* and *Vector Semantics*



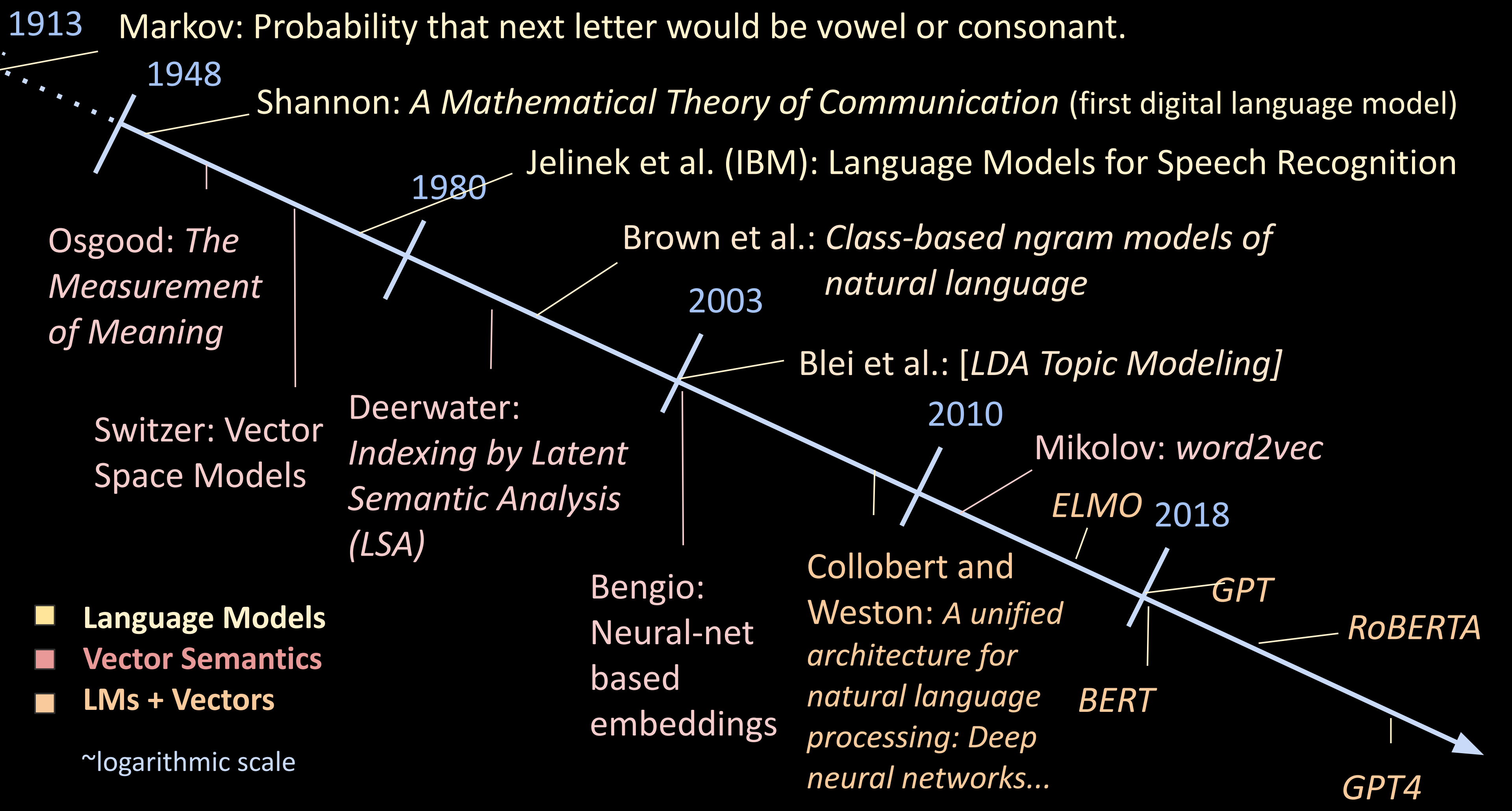
Timeline: *Language Modeling* and *Vector Semantics*



Timeline: *Language Modeling* and *Vector Semantics*



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

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

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:

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

V2:

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

Simple Solution: The Maximum Likelihood Estimate

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

V1:

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

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

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

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

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

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:

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

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: 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!

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

$P(A, B,$ ...

The Ch...

LM version 1

LM version 2

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

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

Language Modeling: How to Estimate?

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

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

A solution: Estimate from shorter sequences.

Chain-Rule:

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

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:

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

Thus, $P(X1,\ldots, Xn) \approx P(Xn | Xn\text{-}k, \ldots, Xn\text{-}1)P(Xn\text{-}1|X(n\text{-}1)\text{-}k, \ldots, Xn\text{-}2) \ldots P(X1)$

Language Modeling: How to Estimate?

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

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

Language Modeling: How to Estimate?

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

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

Language Modeling: How to Estimate?

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

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

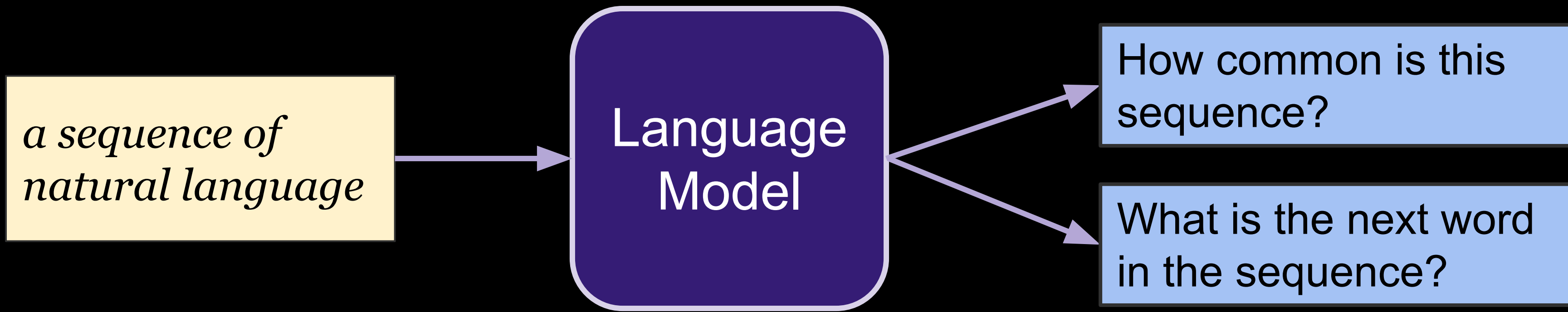
Example generated sentence:

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

P(X1 = "outside", X2="new", X3 = "car", ...)
*≈ P(X1="outside") * P(X2="new"|X1 = "outside) * P(X3="car" | X2="new") * ...*

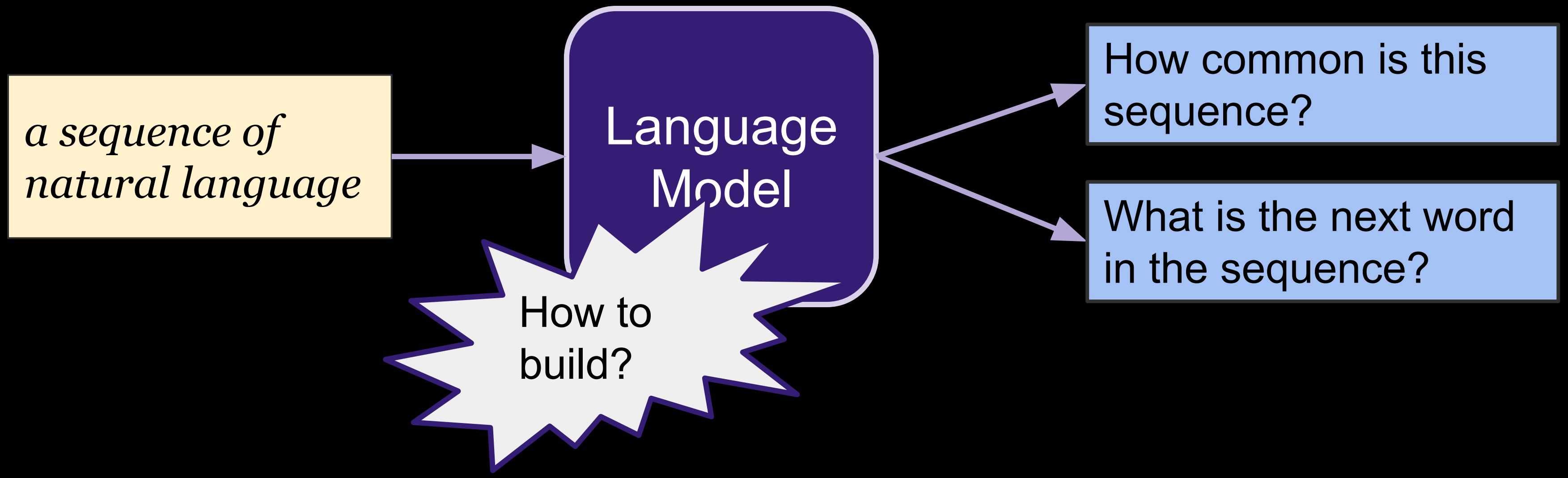
Language Modeling

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



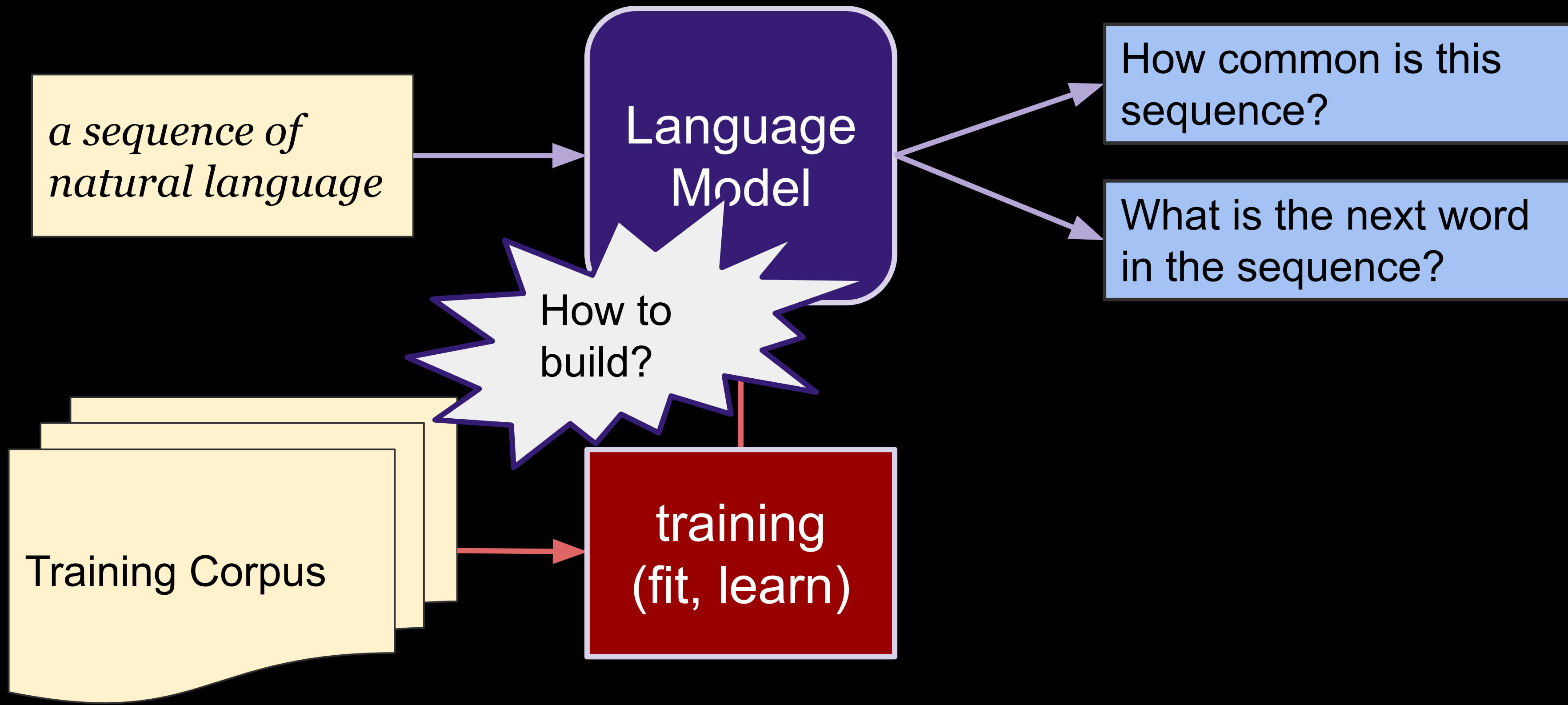
Language Modeling

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 (

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

a sequence of natural language

Training Corpus

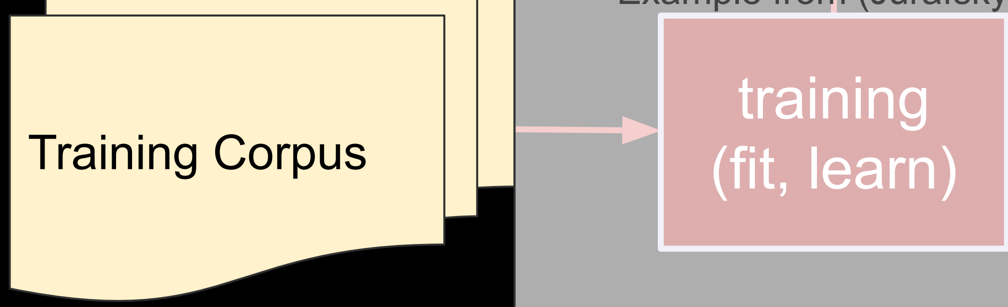
training (fit, learn)

Bigram Counts

first word \ second word

	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)

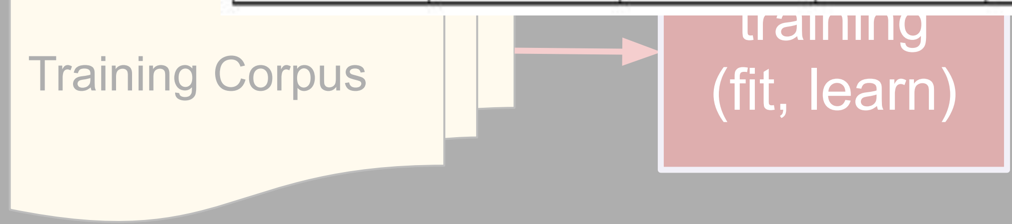


Bigram Counts

first word \ second word

	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



Bigram Counts

first word \ second word

	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

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)

first word: xi-1

second word: xi

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

	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

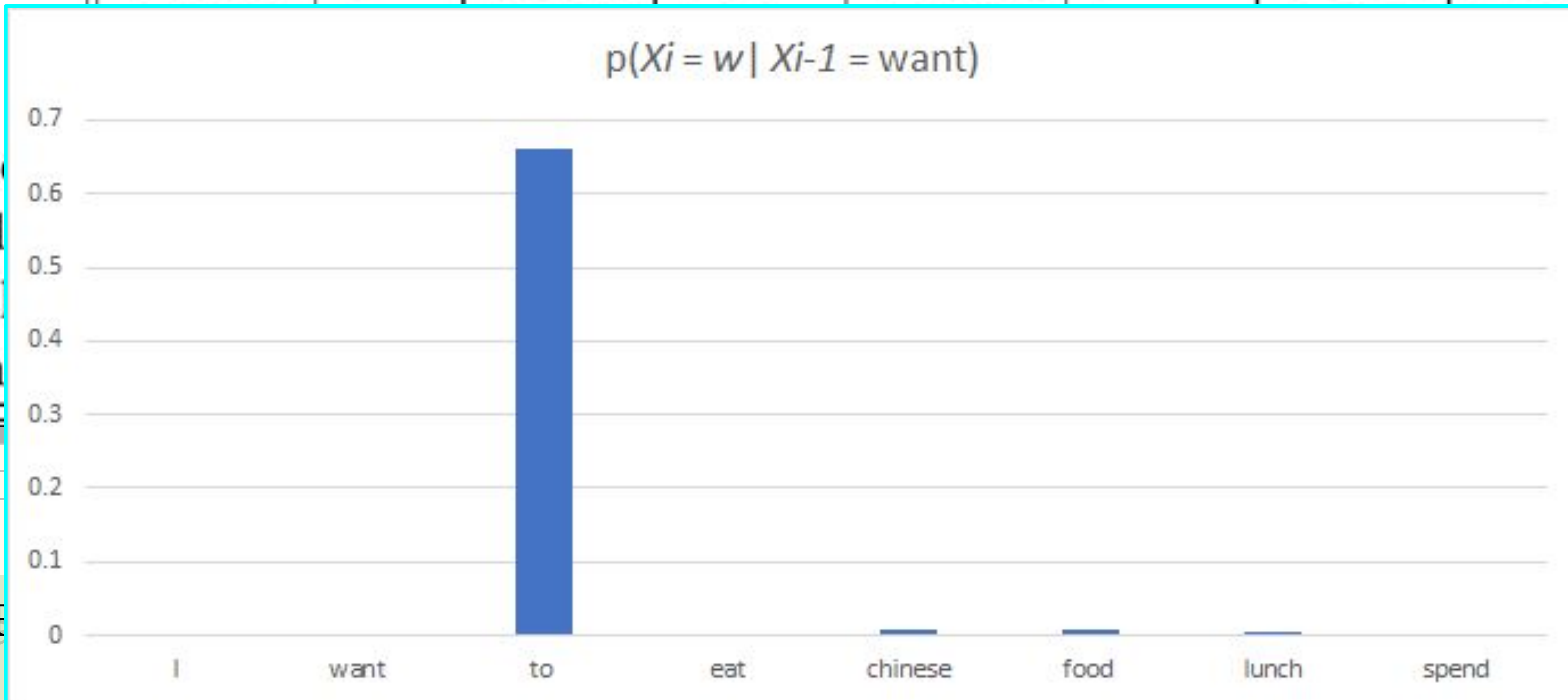
i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

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($Xi\text{-}1$) \ second word (Xi)

P(Xi | Xi-1)

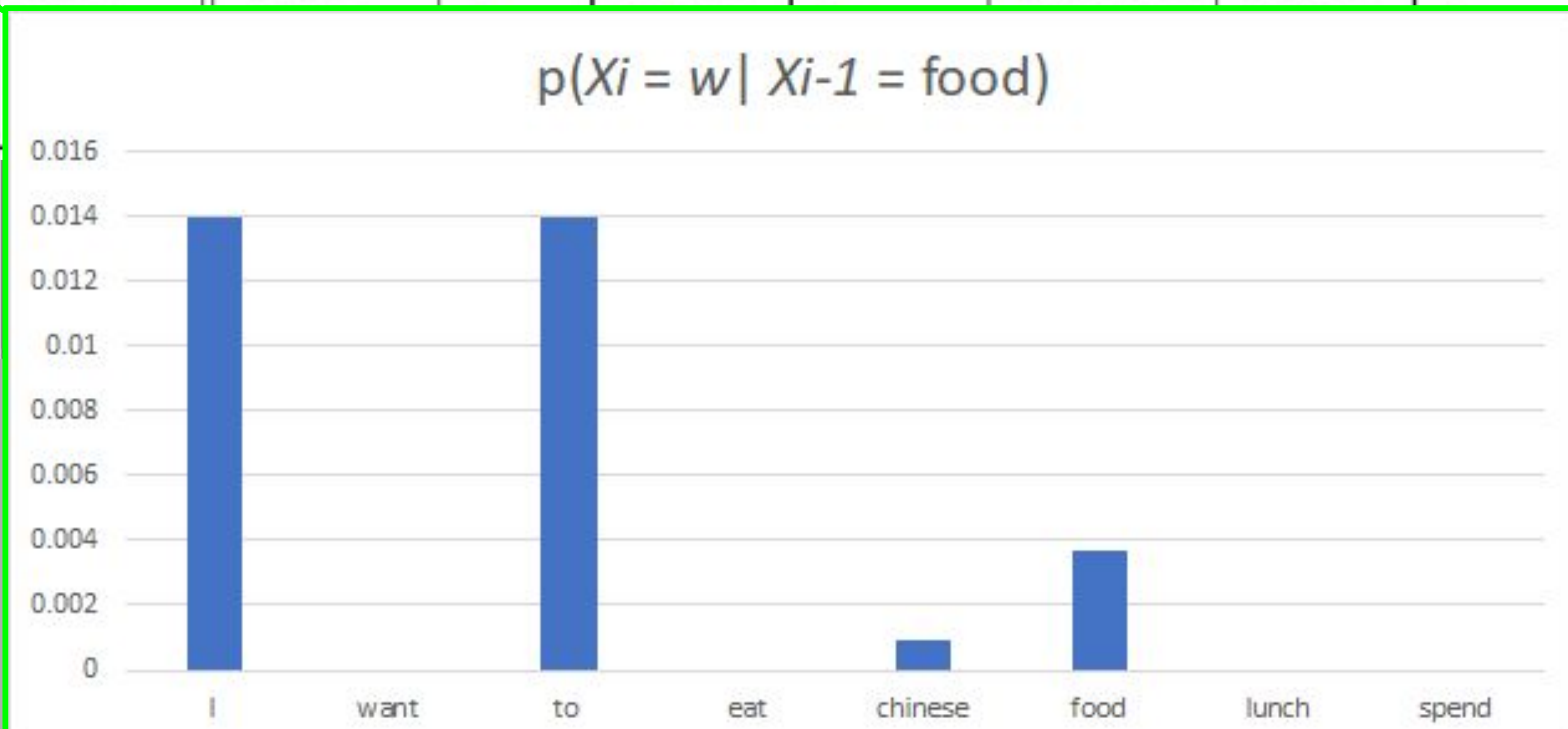
	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to								7
eat								
chin								
food								
lunc								
spen								



Tr H nd

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

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



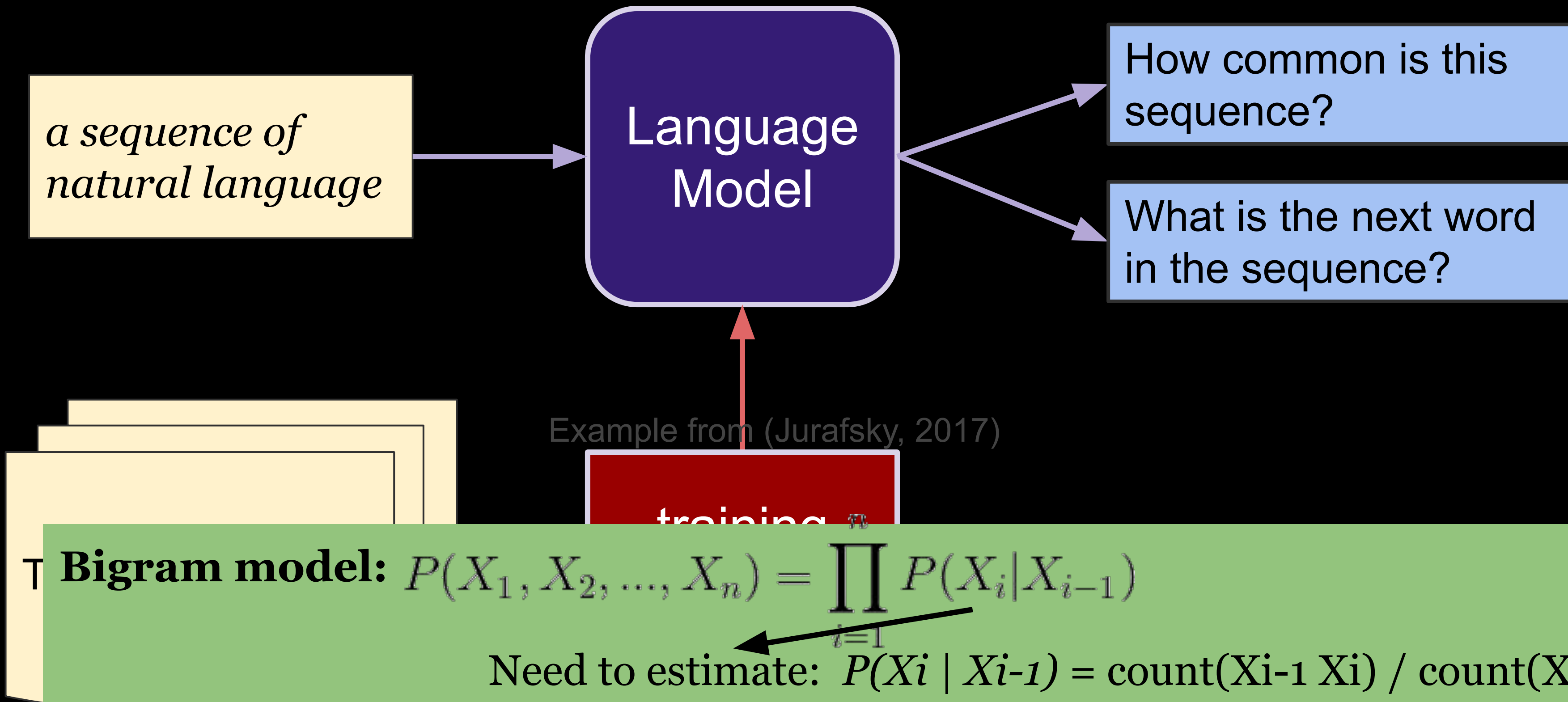
0
0
0

	spend
	278

ount(Xi-1)

Language Modeling

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



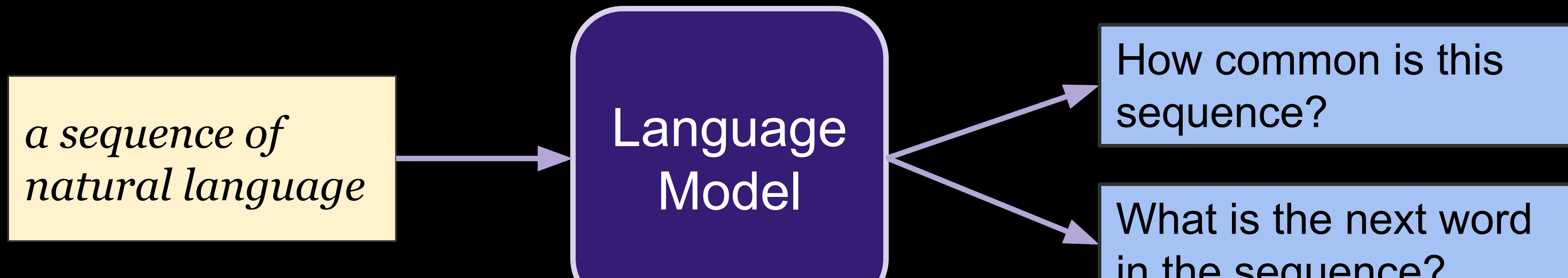
Example from (Jurafsky, 2017)

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

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

Language Modeling

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



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

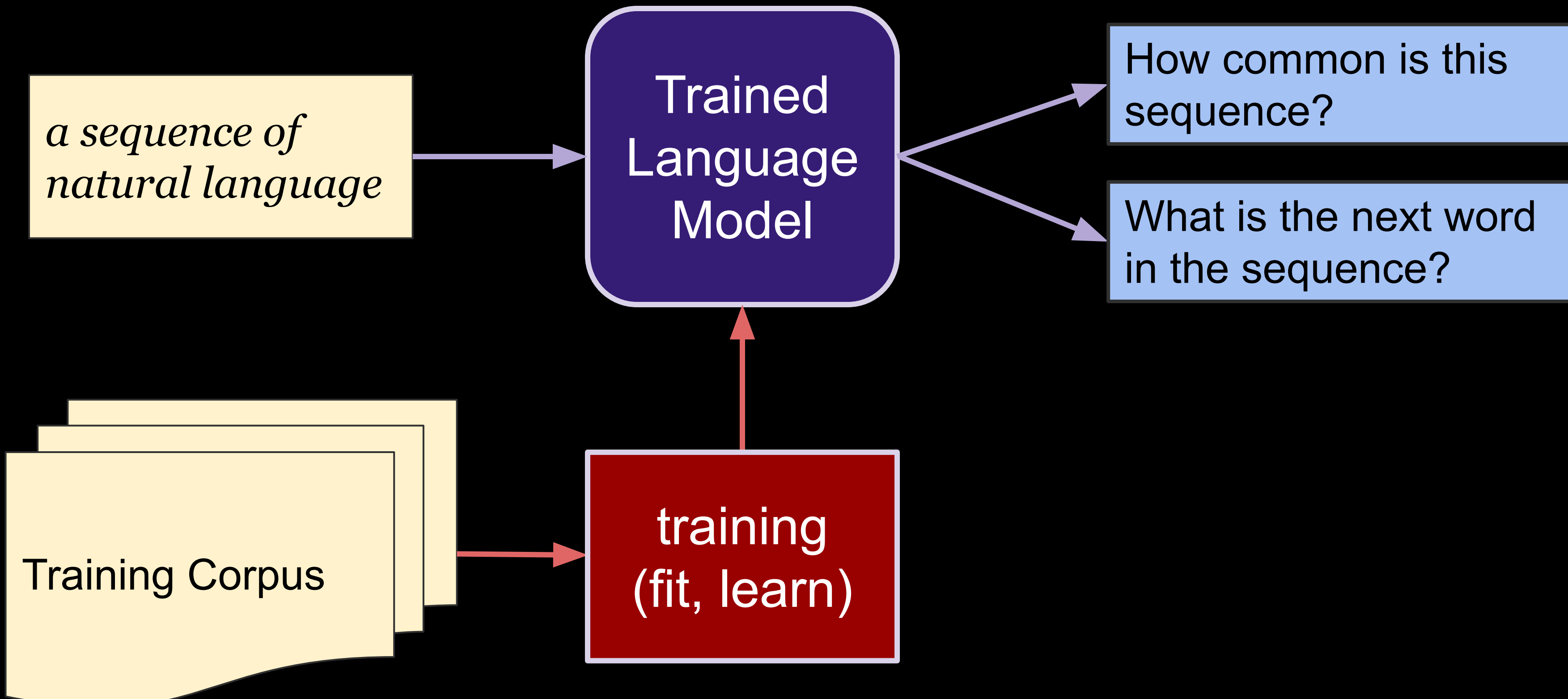
Need to estimate: *P(Xi | Xi-1, Xi-2)* = count(Xi-2 Xi-1 Xi) / count(Xi-2 Xi-1)

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)

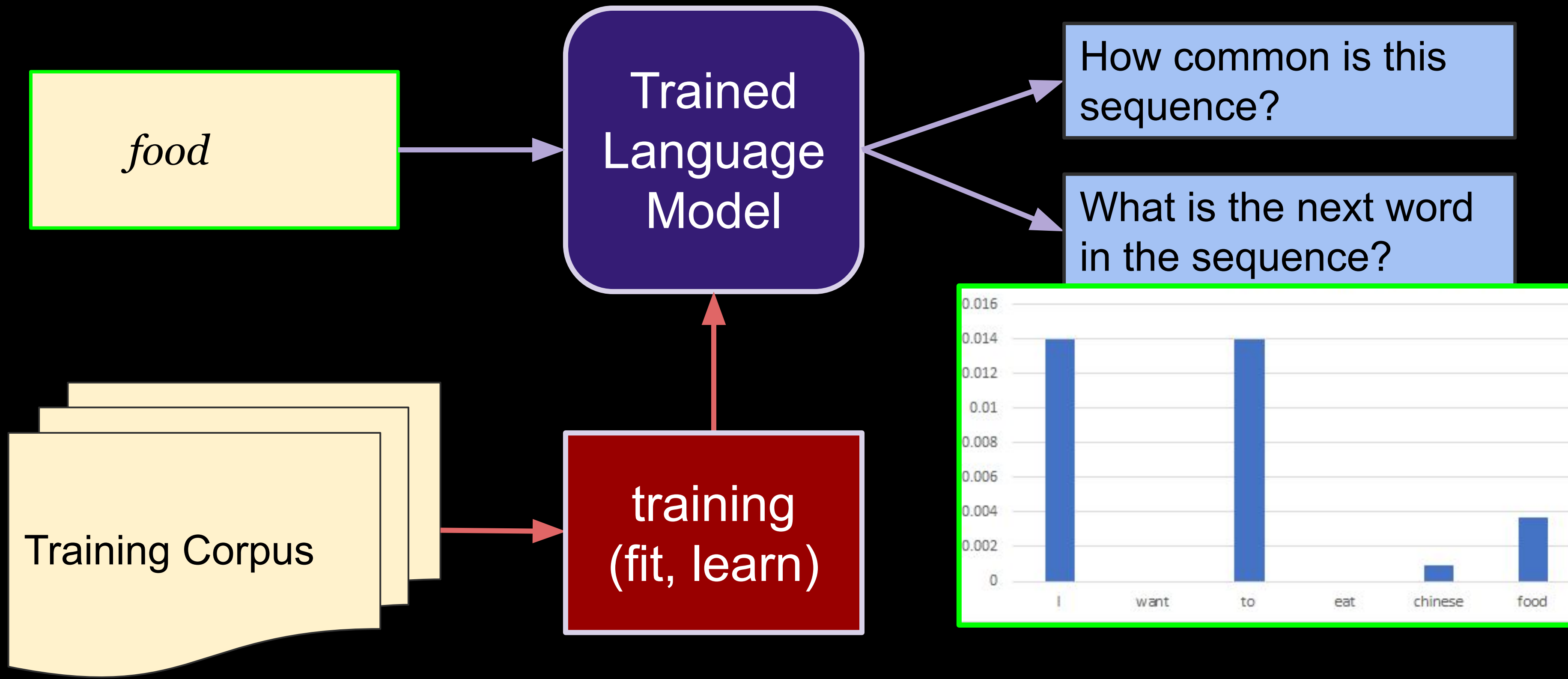
Language Modeling

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



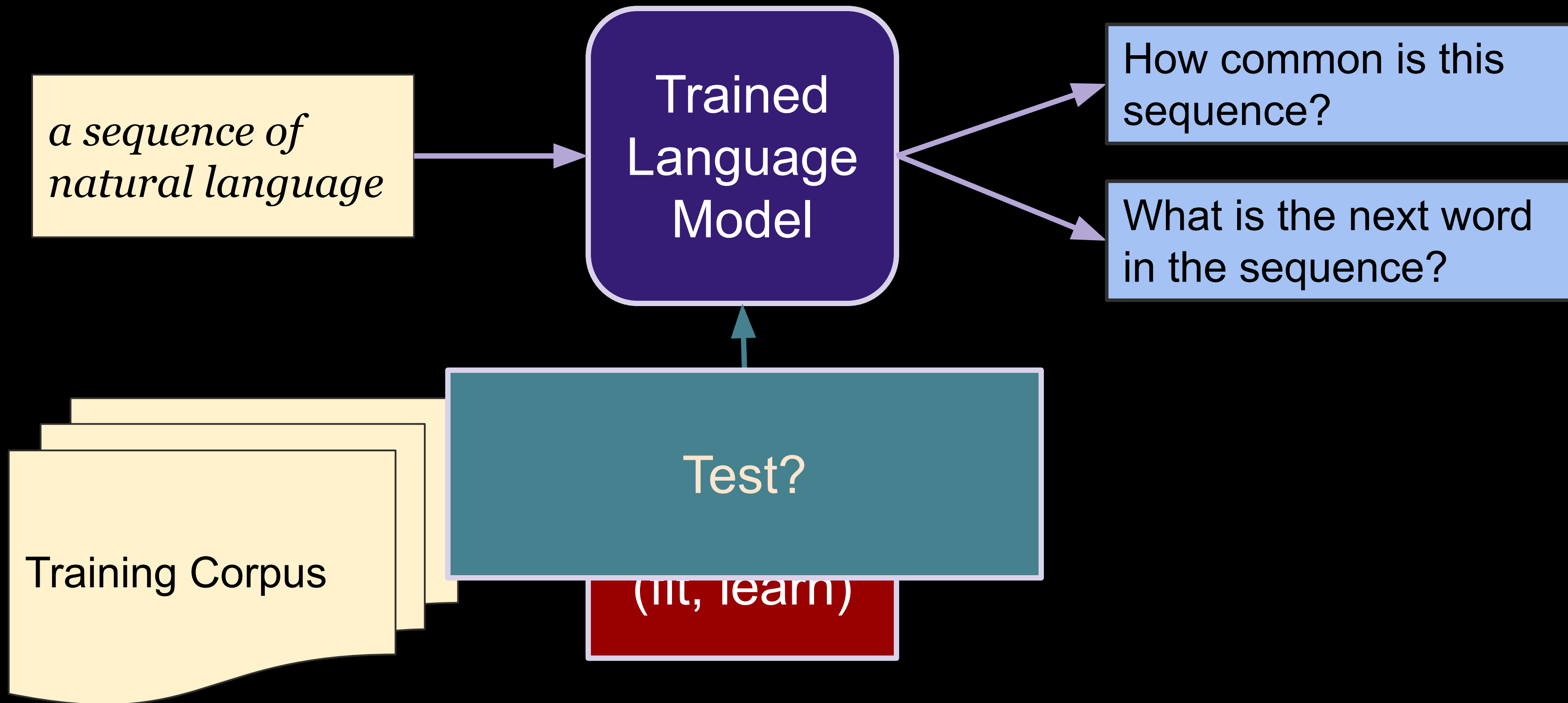
Language Modeling

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



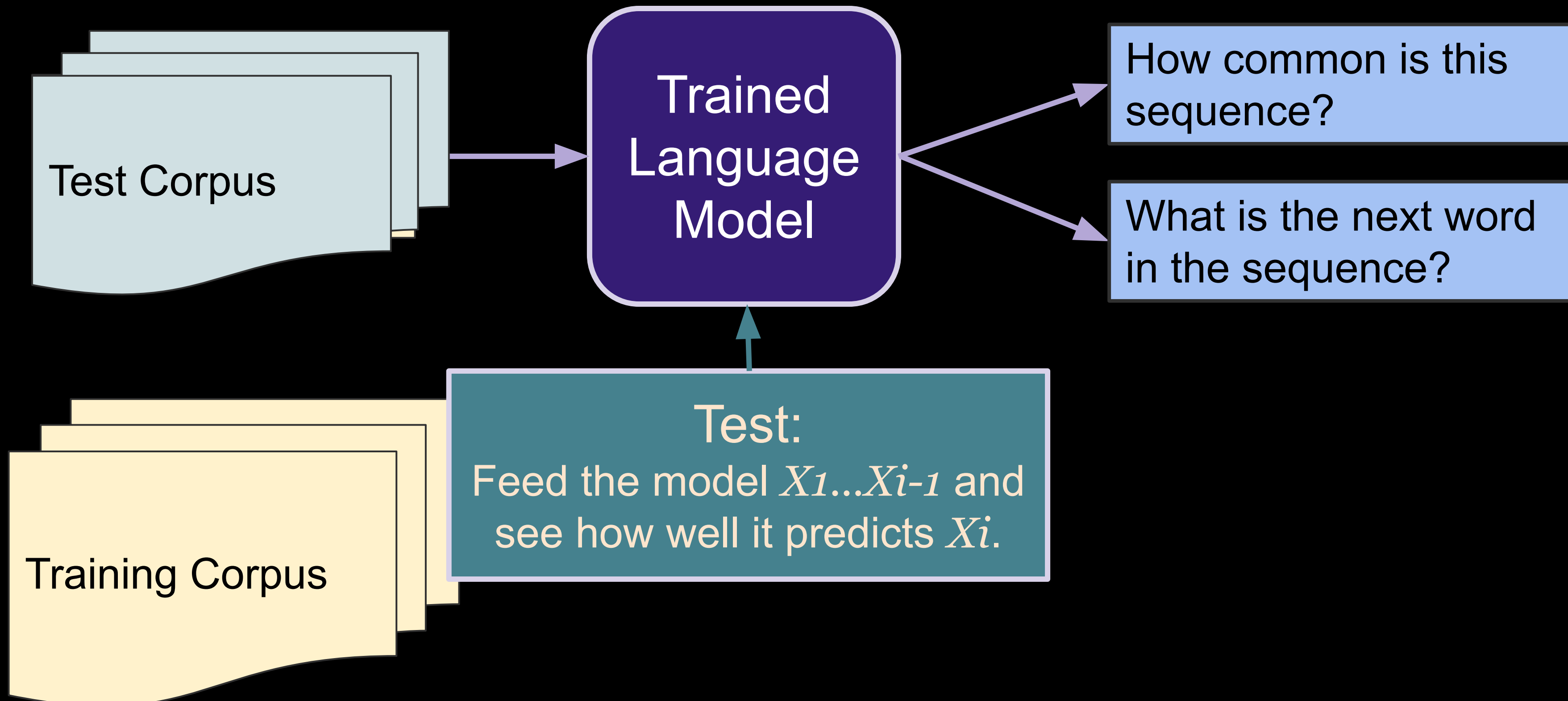
Language Modeling

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



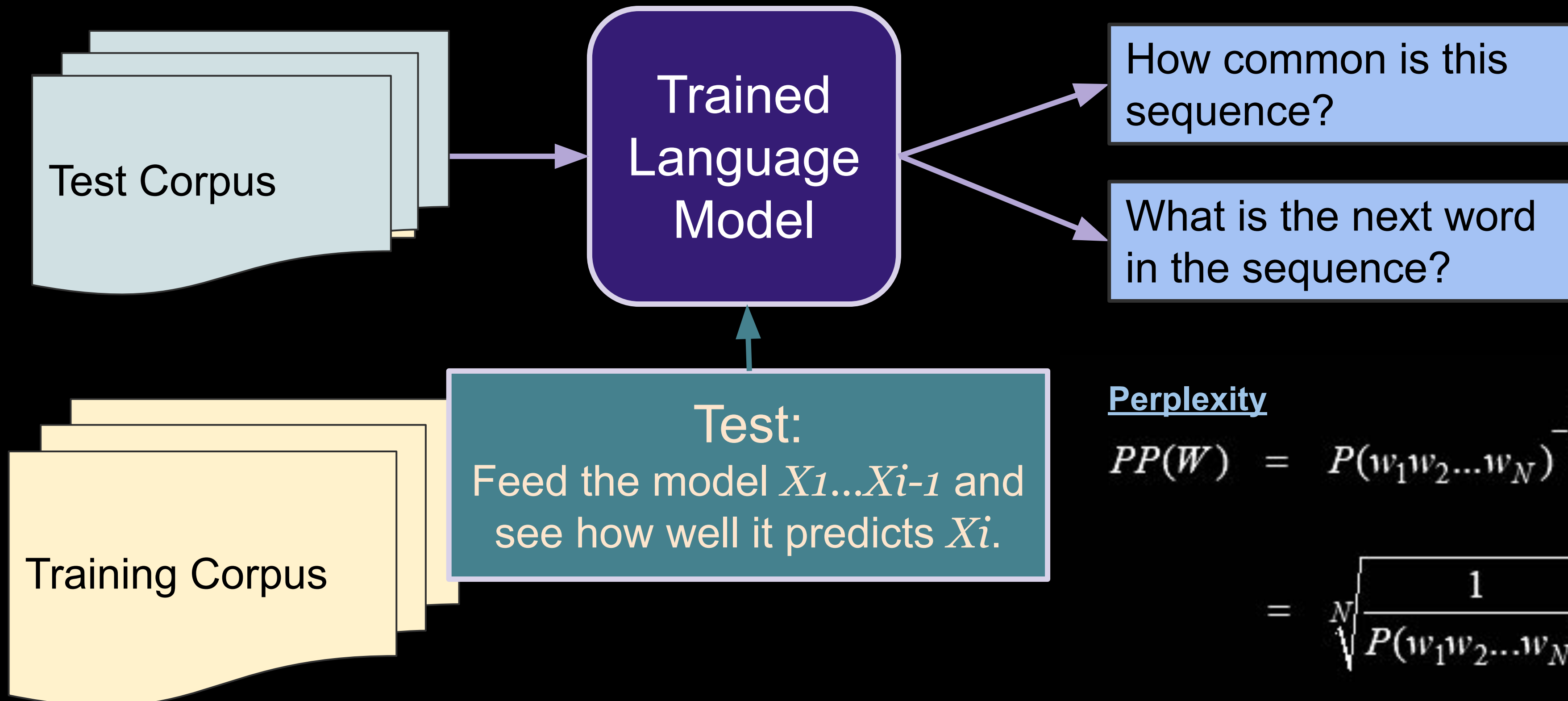
Language Modeling

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



Language Modeling

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

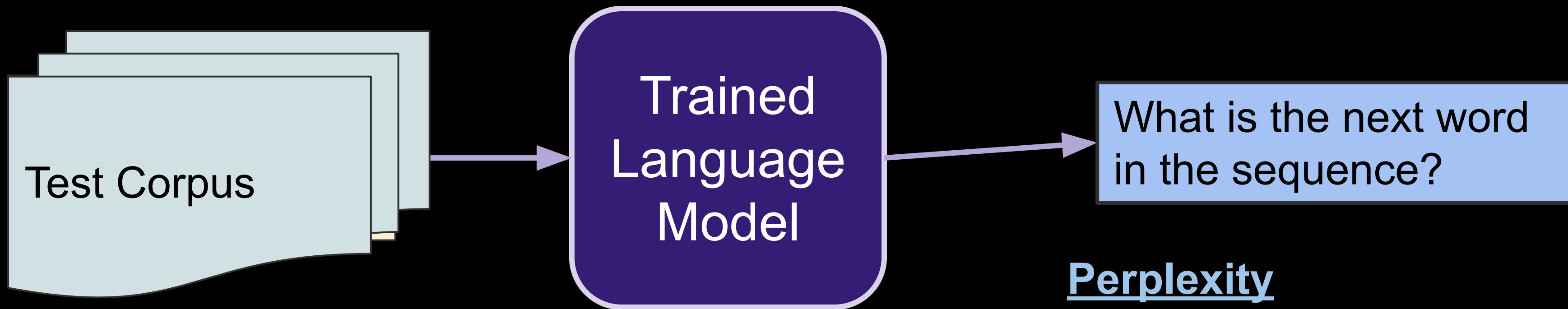


Perplexity

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

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

Evaluation

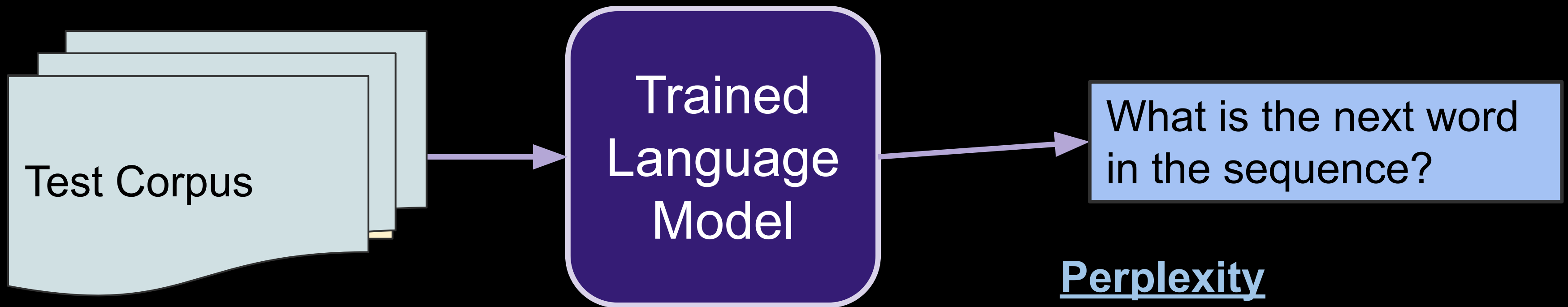


Perplexity

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

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

Evaluation



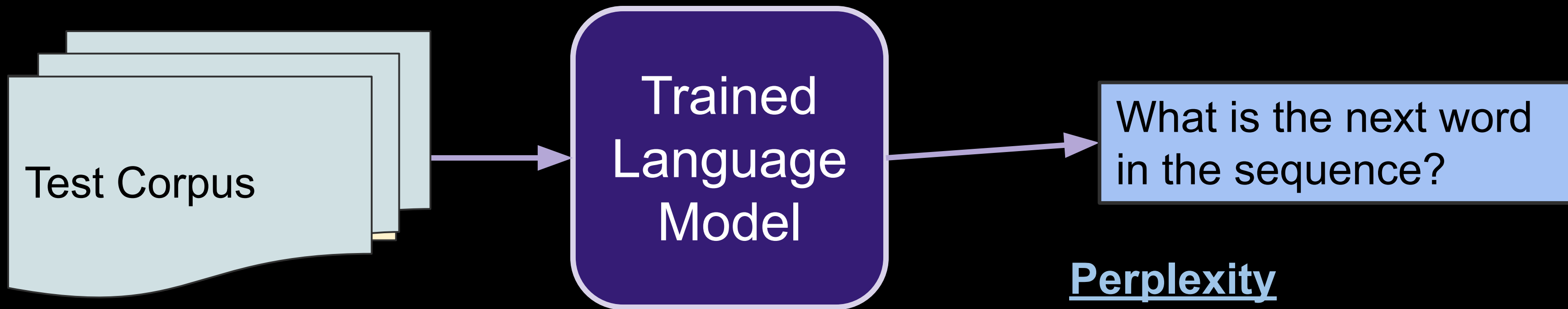
Perplexity

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

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

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

Evaluation



Perplexity

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

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

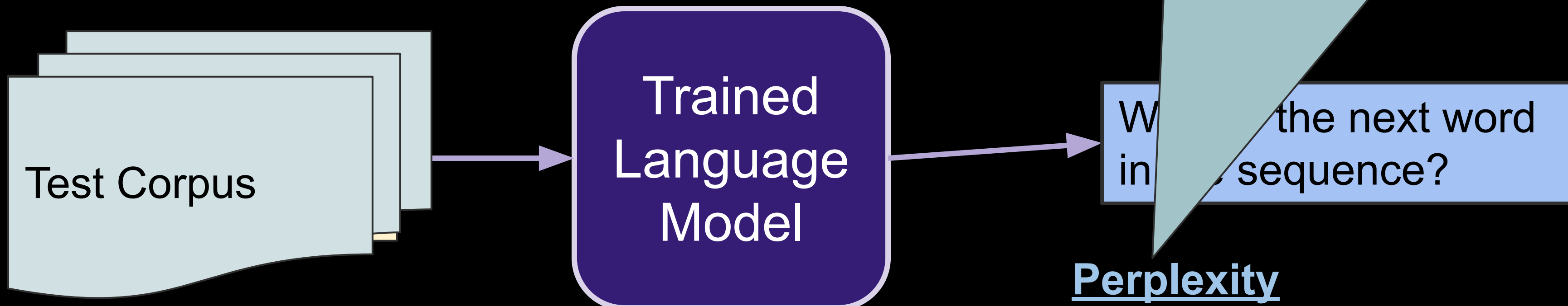
Apply Chain Rule: $$\text{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

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

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

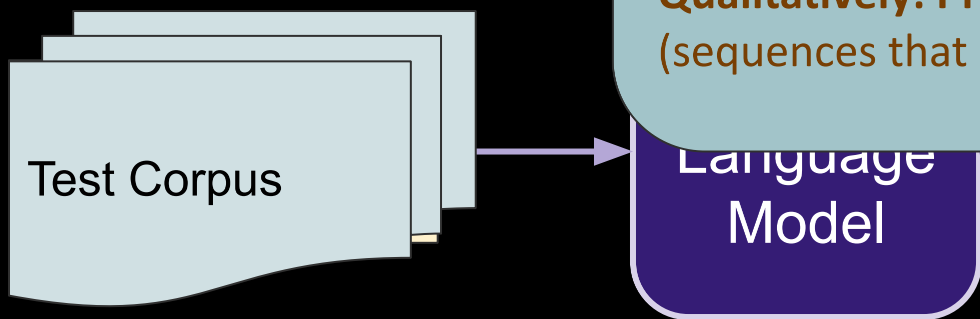
Thus, perplexity for Bigrams: $$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

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

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

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) (unknown word token)
  Choose a 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.

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) (unknown word token)
  Choose a 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.

- This is also "auto-regressive" or generative language modeling.
  "auto-encoding" using context on both sides – use for creating embeddings.

Problem?

Problem?

- High probabilities in context of rare words
- Zero probabilities for when count was zero (but is that realistic?)

Zeros!

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

Zeros and Smoothing

Bigram Counts

first word \ second word

	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

Laplace ("Add one") smoothing: add 1 to all counts

Unsmoothed probs

first word($Xi\text{-}1$) \ 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	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

Example from (Jurafsky, 2017)

Smoothed

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)+1}{c(w_{i-1})+V}$$

(vocabulary size)

first word(Xi-1) \ second word (Xi)

P(Xi | Xi-1)

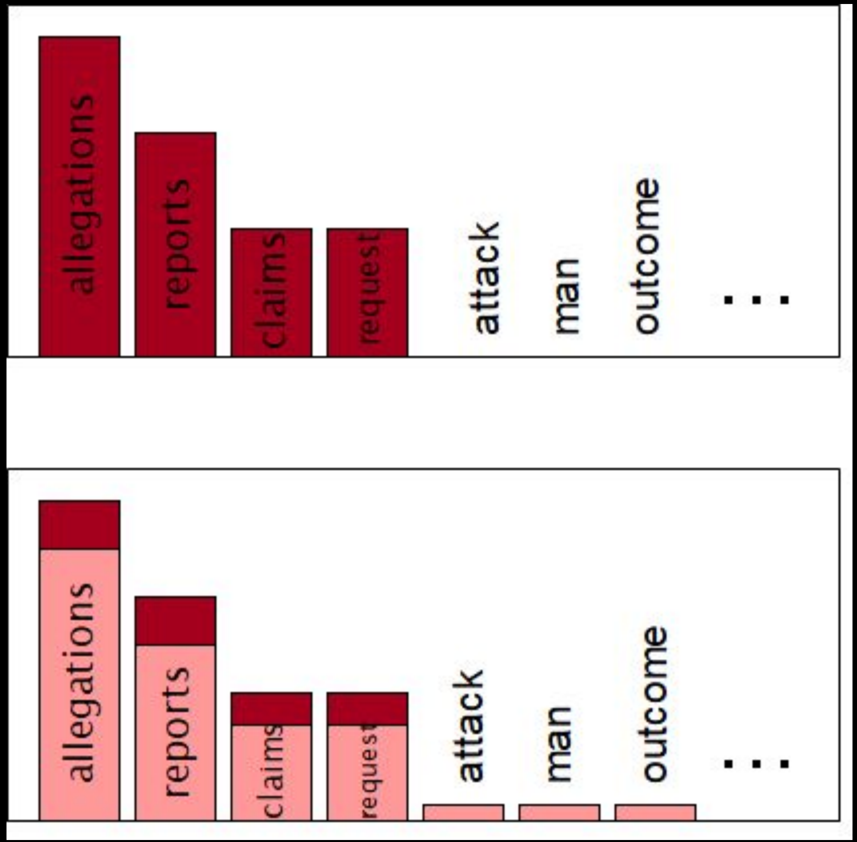
	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	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

Example from (Jurafsky, 2017)

Why Smoothing? Generalizes

Original

With Smoothing



(Example from Jurafsky / Originally Dan Klein)

Why Smoothing? Generalizes

Add-one is blunt:

can lead to very large changes.

More Advanced:

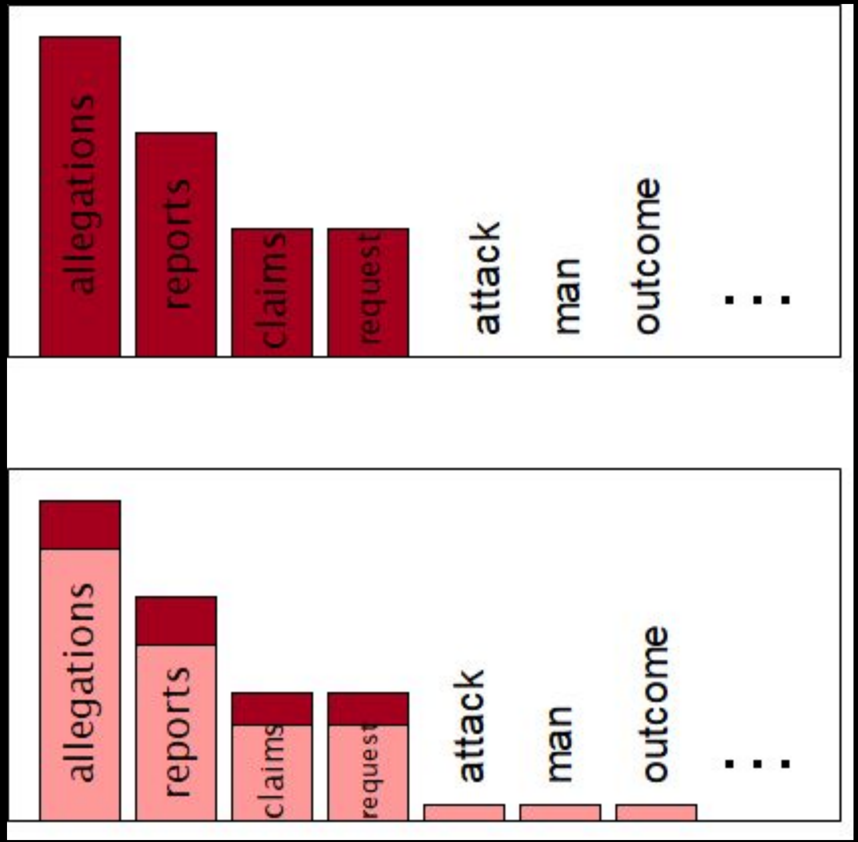
Informative Prior:

Add unigram probability

More advanced:

- Good-Turing Smoothing
  Kneser-Nay Smoothing

^ outside scope for now. We will eventually cover, even stronger, deep learning based models.



Example how to produce language generator

Training:

- Count unigrams, bigrams, and trigrams
- Create function to calculate probabilities for unigram, bigram, and trigram models (over training, with smoothing)

Example how to produce language generator

Training:

- Count unigrams, bigrams, and trigrams
- Create function to calculate probabilities for unigram, bigram, and trigram models (over training, with smoothing)

Generation

- Create function: Given previous word or previous 2 words, take a random draw from what words are most likely to be next.

  Trigram model and bigram when possible (high counts)

    Backing off to bigram or even unigram, if necessary

Language Modeling Summary

- Two versions of assigning probability to sequence of words
- Applications
- The Chain Rule, The Markov Assumption: $P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i | X_{i-k}, X_{i-(k-1)}, ..., X_i)$
- Training a unigram, bigram, trigram model based on counts
- Evaluation: Perplexity
- Zeros, Low Counts, and Generalizability
- Add-one smoothing

Limitation: Long distance dependencies

The horse which was raced past the barn tripped .