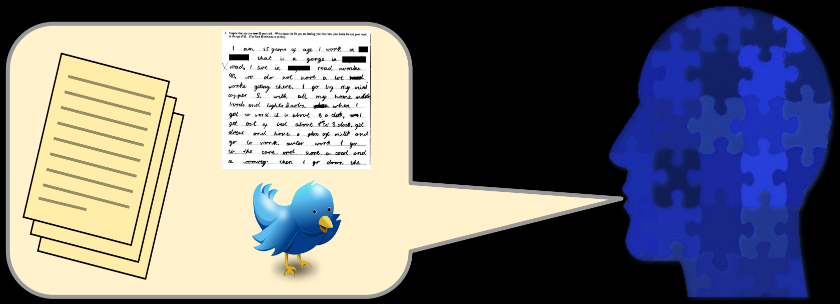
Language Modeling

CSE354 - Spring 2020

Task



- Language Modeling
  (i.e. auto-complete)

how? →

- **Probabilistic Modeling**
  - **Probability Theory**
  - **Logistic Regression**
  - **Sequence Modeling**

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 $P(w5| w1, w2, w3, w4)$

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

:probability of a next word given history

P(fork | He ate the cake with the) = ?

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)

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 $P(w5 | w1, w2, w3, w4)$

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

:probability of a next word given history

P(fork | He ate the cake with the) = ?

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

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

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

total number of observed *7grams*

Simple Solution: The Maximum Likelihood Estimate

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

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

Simple Solution: The Maximum Likelihood Estimate

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

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

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

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

Solution: Estimate from shorter sequences, use more sophisticated probability theory.

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

Solution: Estimate from shorter sequences, use more sophisticated probability theory.

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

Example from (Jurafsky, 2017)

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

Solution: Estimate from shorter sequences, use more sophisticated probability theory.

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

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

Example from (Jurafsky, 2017)

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

Solution: Estimate from shorter sequences, use more sophisticated probability theory.

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

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

The Chain Rule:

$$P(X_1, X_2, \ldots, X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2) \ldots P(X_n|X_1, \ldots, X_{n-1})$$

Example from (Jurafsky, 2017)

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

Solution: Estimate from shorter sequences, use more sophisticated probability theory.

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

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

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

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

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

Solution: Estimate from shorter sequences, use more sophisticated probability theory.

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

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

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

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

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

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

Solution: Estimate from shorter sequences, use more sophisticated probability theory.

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

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

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

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

Markov Assumption:

$P(Y$ … X … $1), ..., X_i)$

P(Xn| X …

What about Logistic Regression? Y = next word
P(Y|X) = P(Xn | Xn-1, Xn-2, Xn-3, ...)

Not a terrible option, but Xn-1 through Xn-k would be modeled as independent dimensions. Let's revisit later.

P(…

P(…

The Chain … $P(X_i|X_1, X_2, ..., X_i)$ … $i=1$

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

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

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

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

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

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

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

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

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

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

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

Example generated sentence:

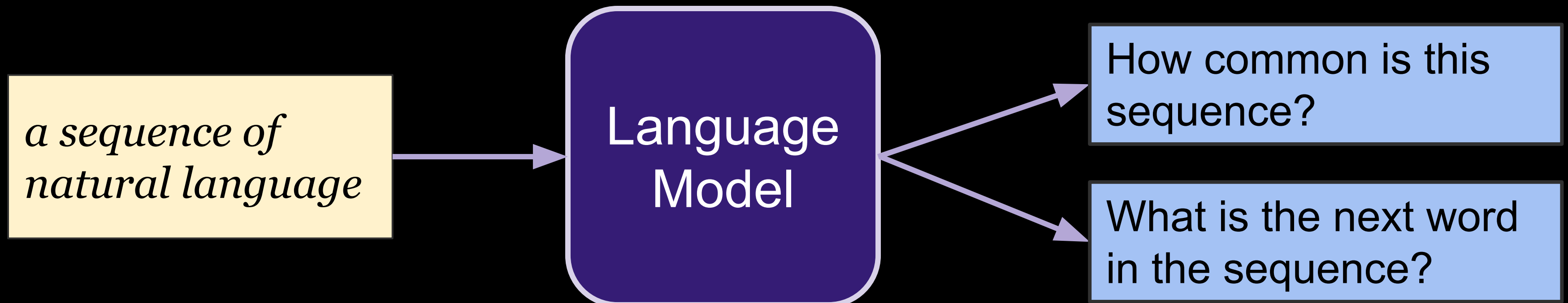
outside, new, car, parking, lot, of, the, agreement, reached

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

Example from (Jurafsky, 2017)

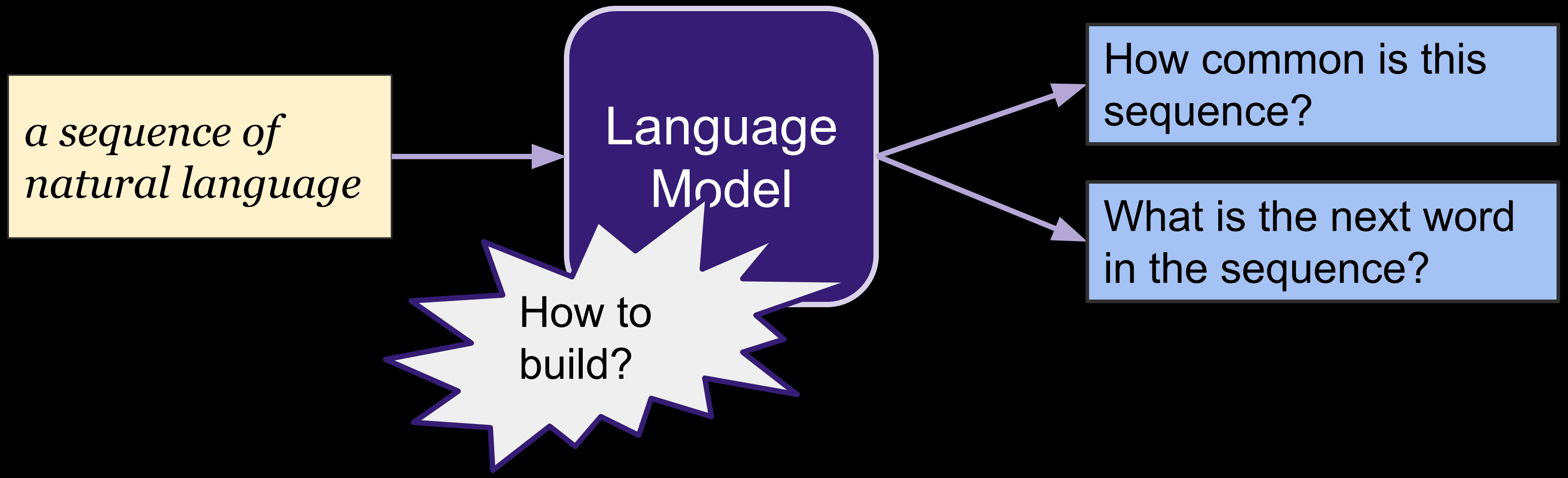
Language Modeling

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



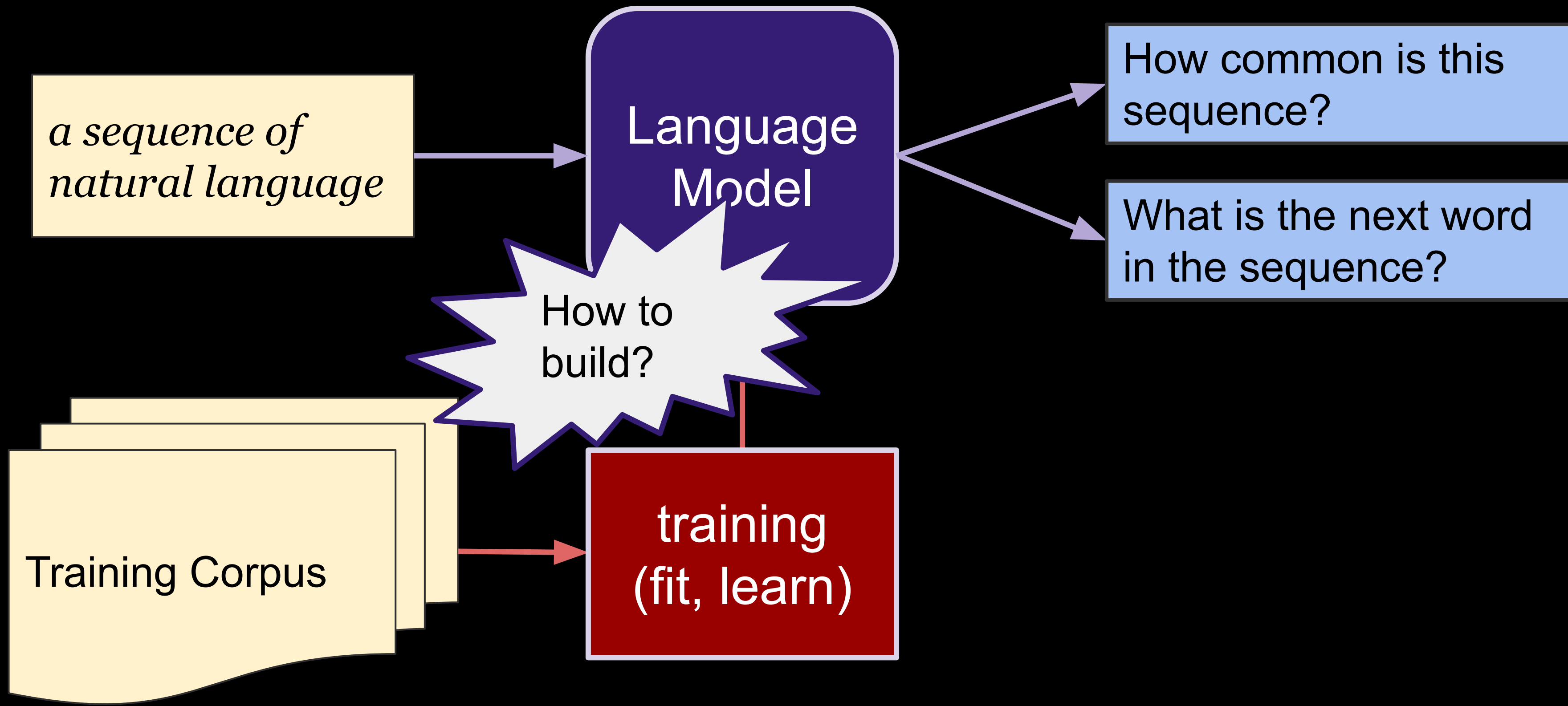
Language Modeling

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



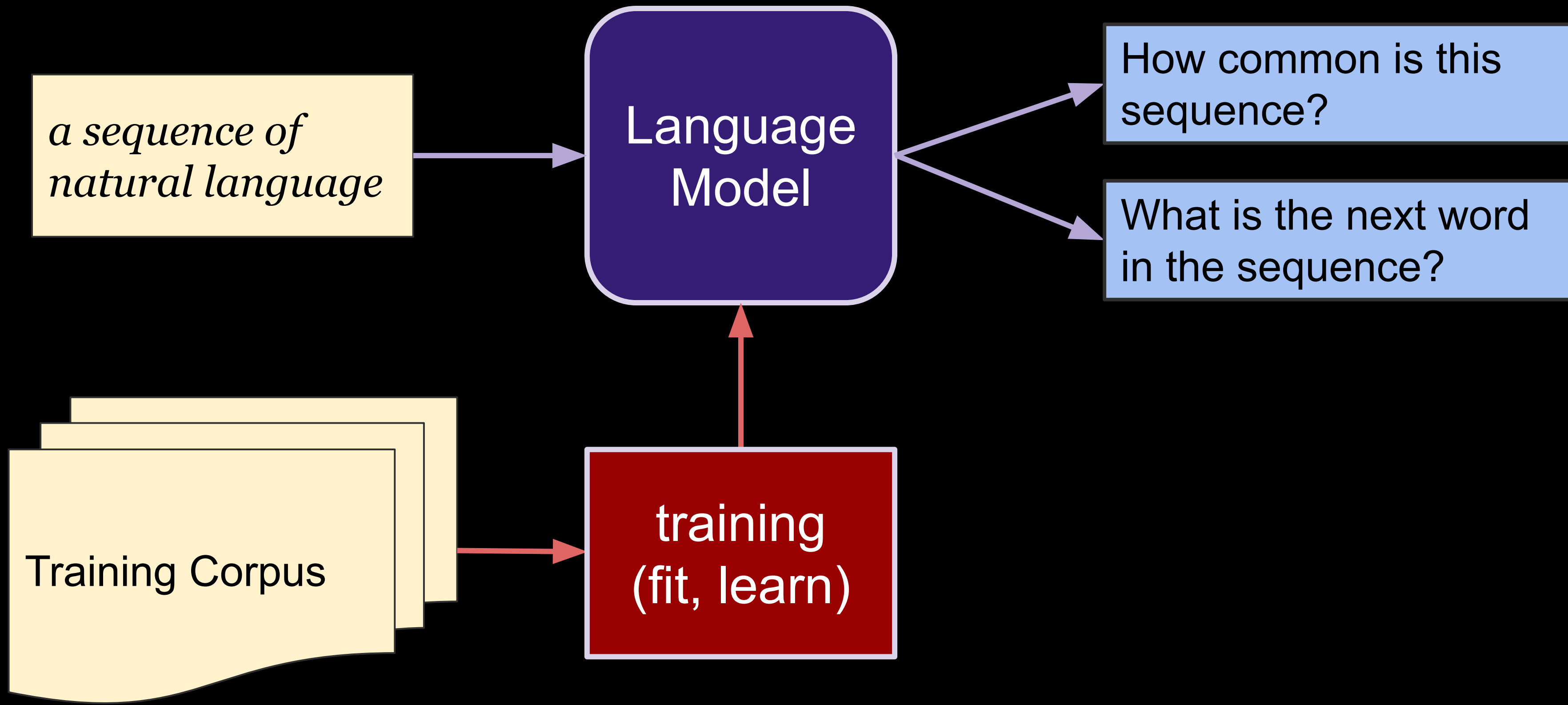
Language Modeling

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



Language Modeling

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

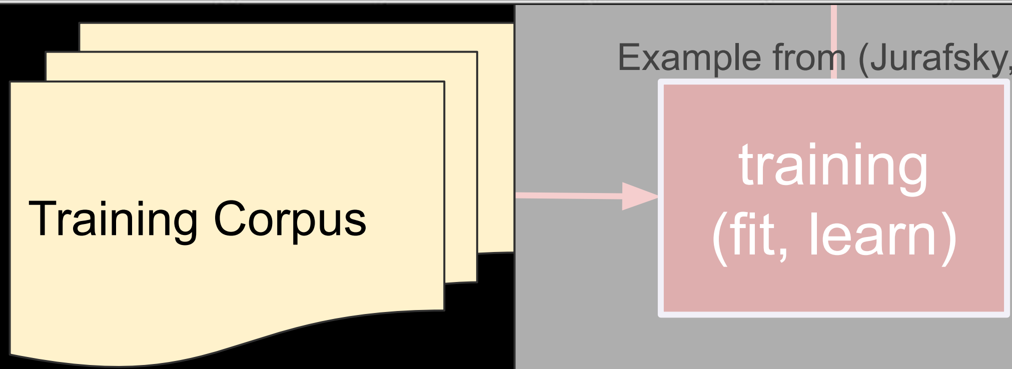


Bigram Counts

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 → rows; second word → columns

	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 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

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 | 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

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

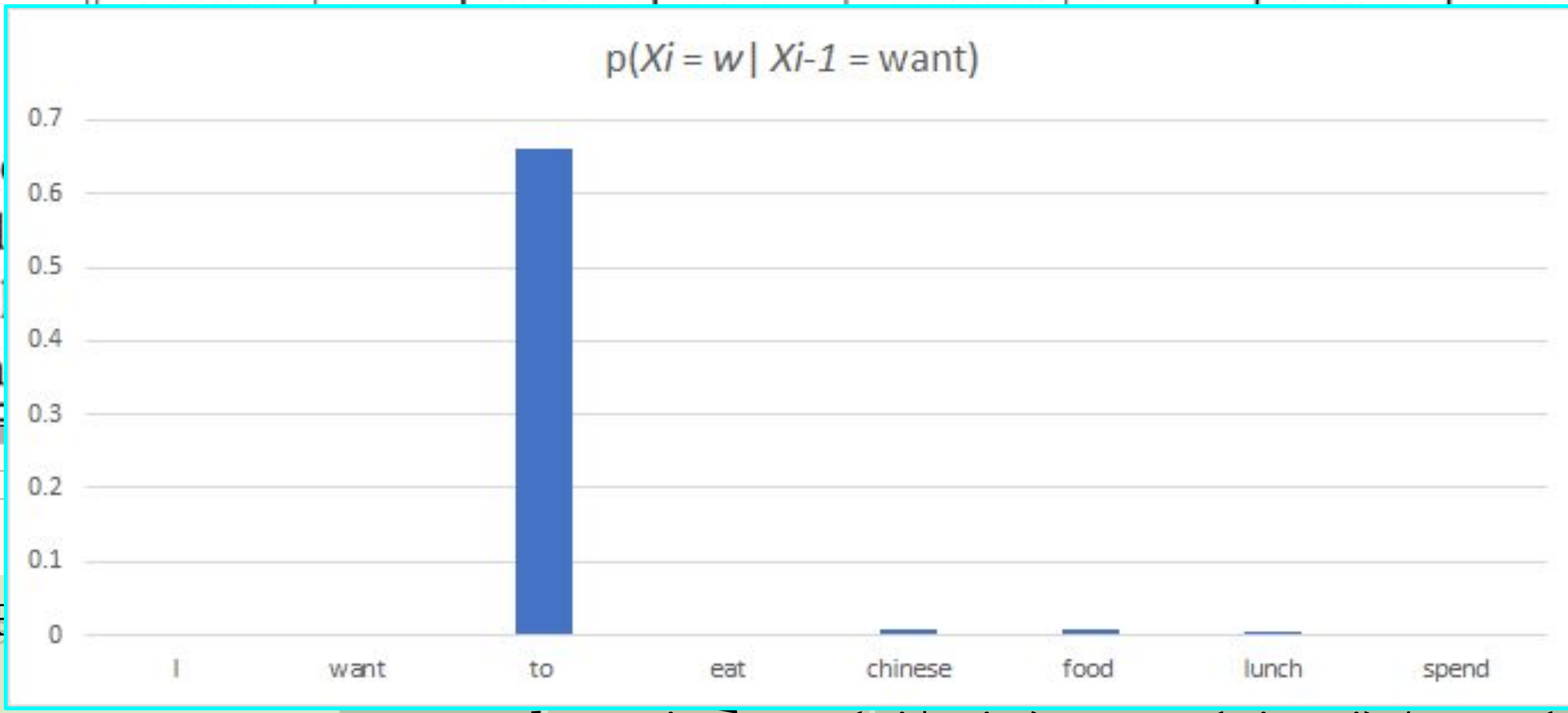
Bigram model: $P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | X_{i-1})$

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

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

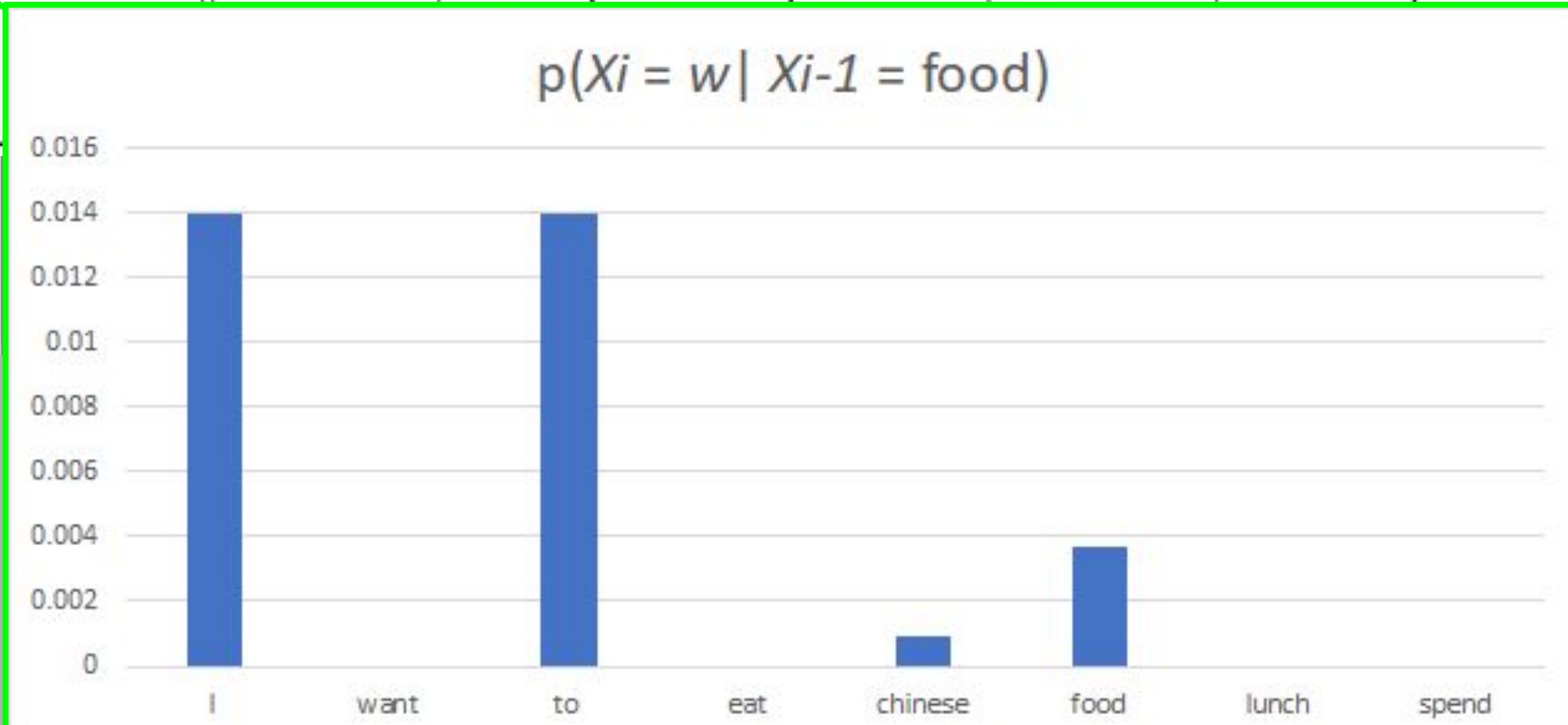
P(Xi | Xi-1)

	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								
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



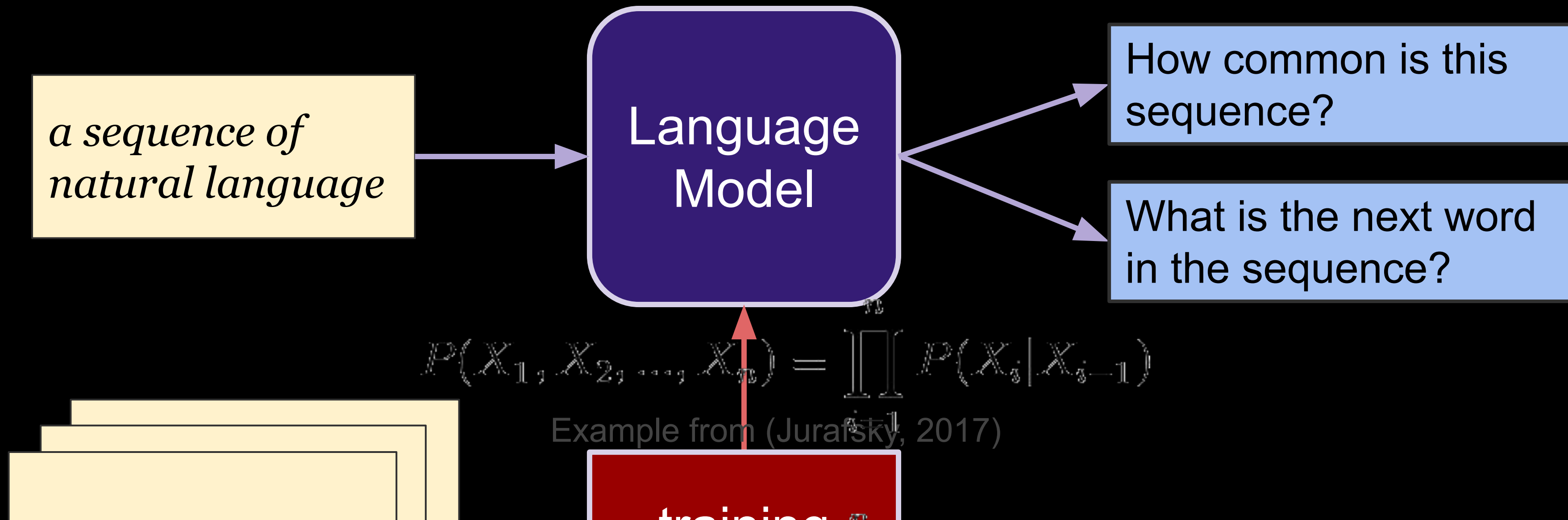
0
0
0

	spend
	278

count(Xi-1)

Language Modeling

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



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

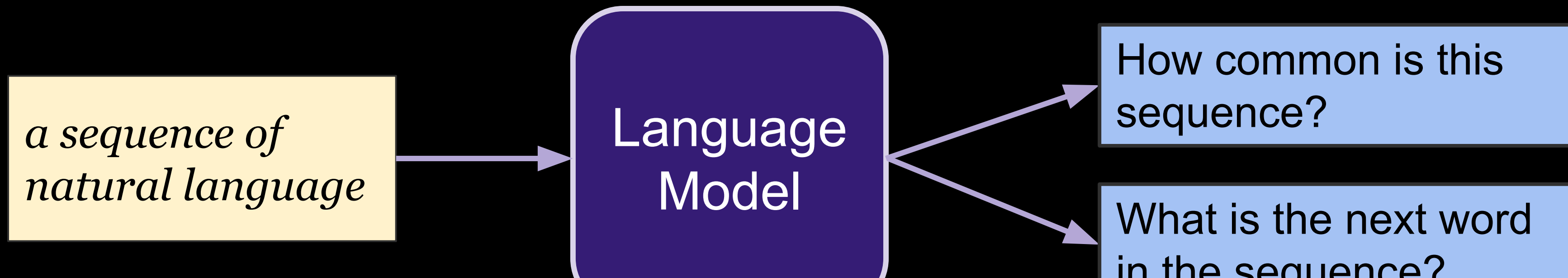
Example from (Jurafsky, 2017)

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

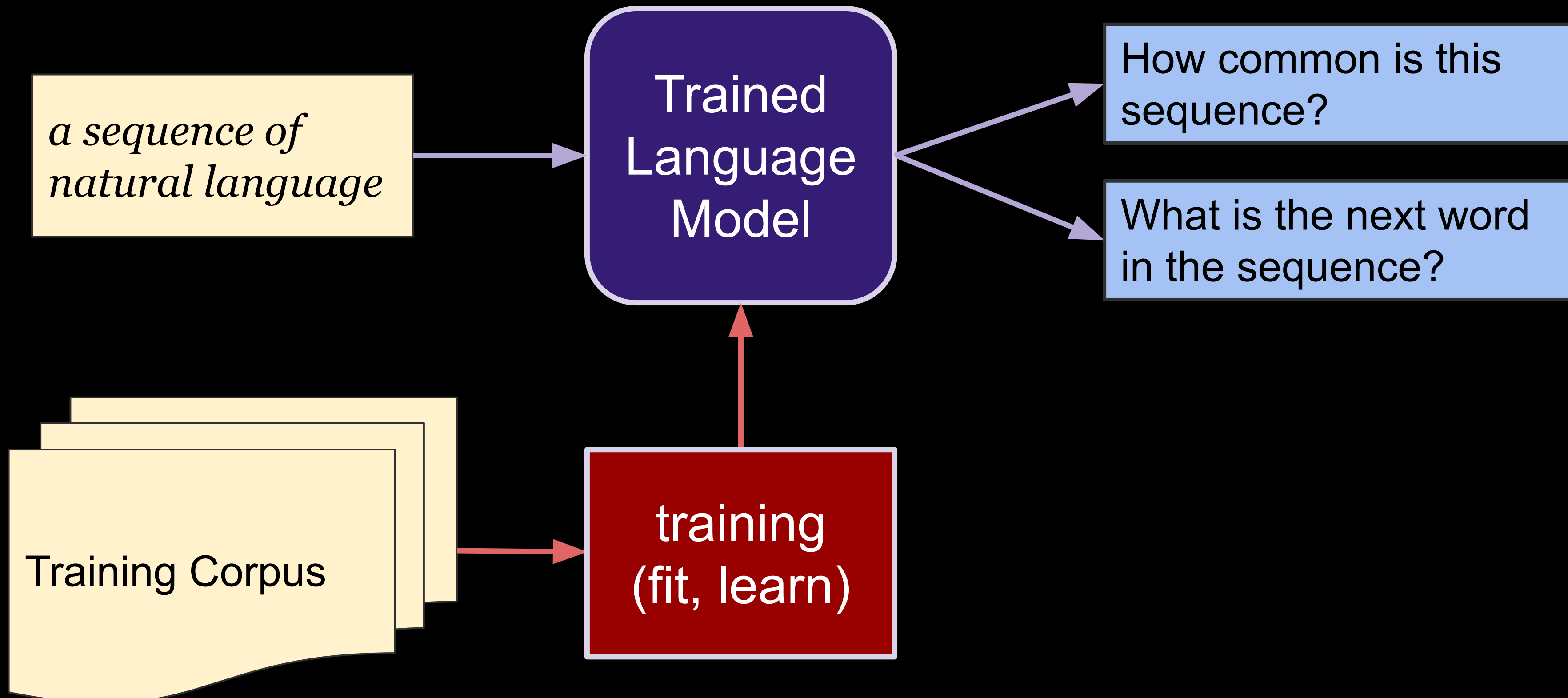
Need to estimate: *P(Xi | Xi-1, Xi-2)* = count(Xi-2 Xi-1 Xi) / count(Xi-2 Xi-1)

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)

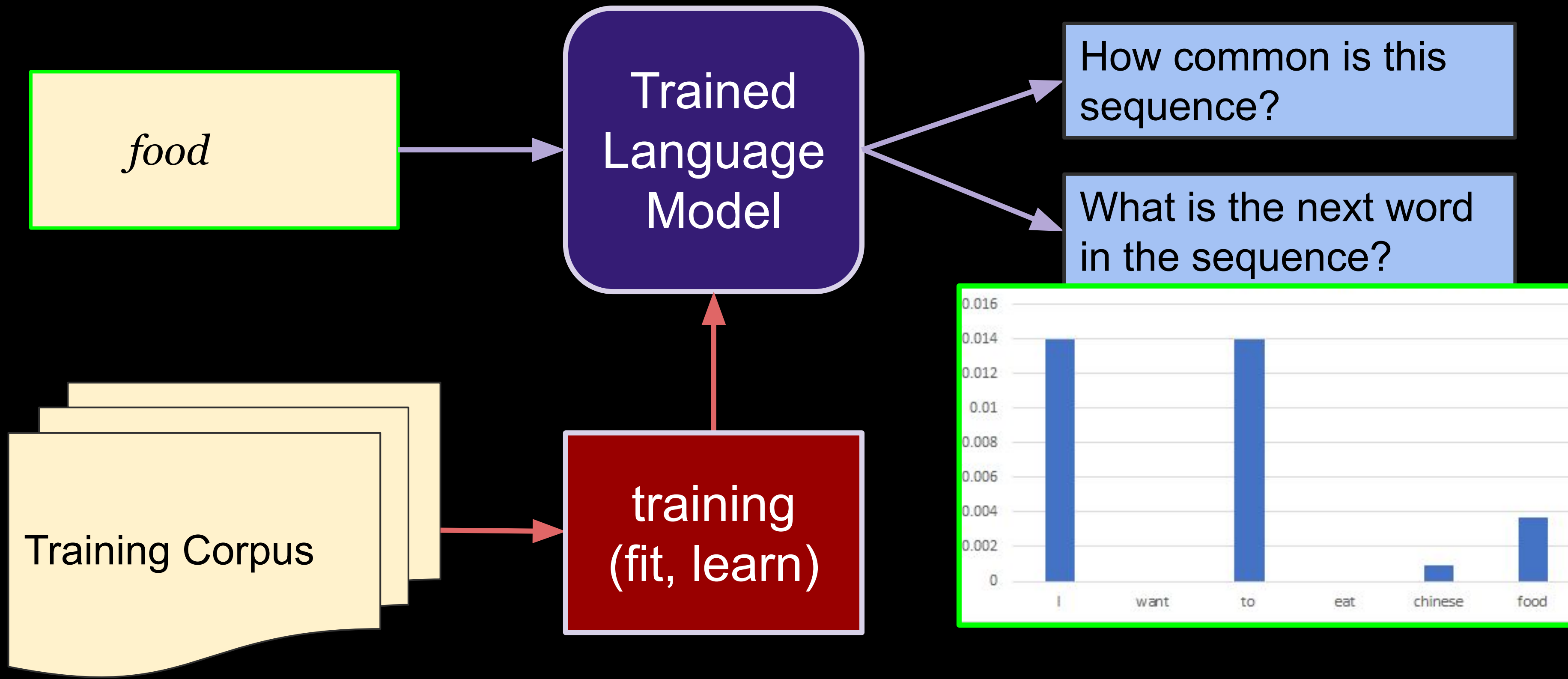
Language Modeling

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



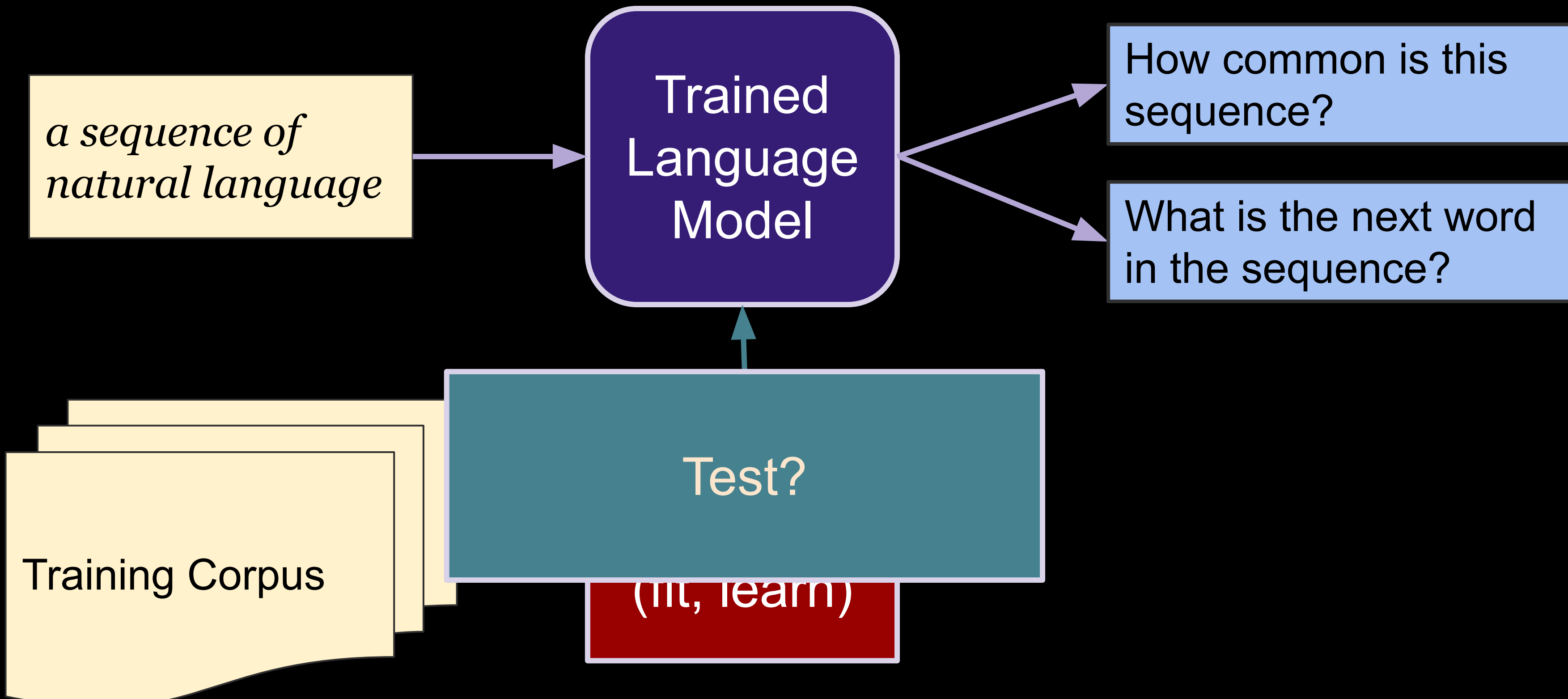
Language Modeling

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



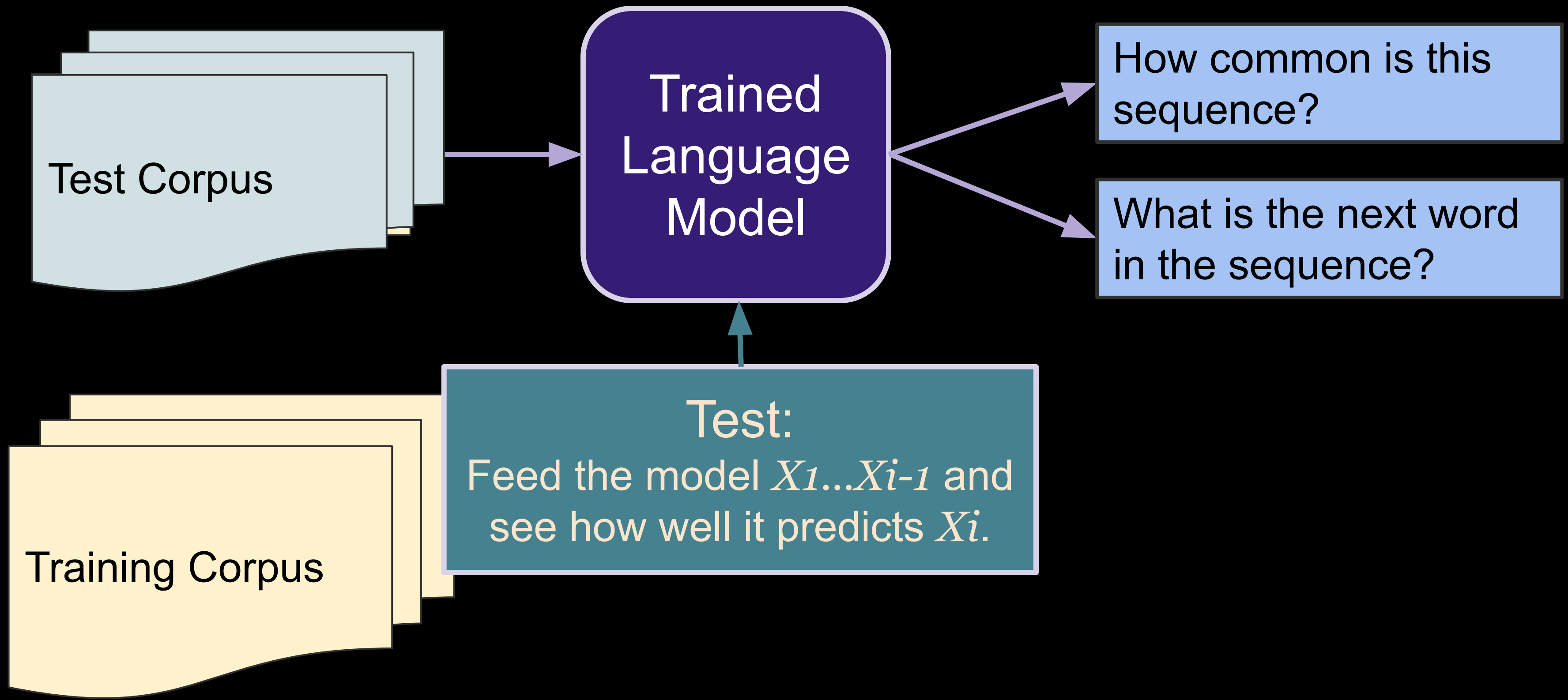
Language Modeling

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



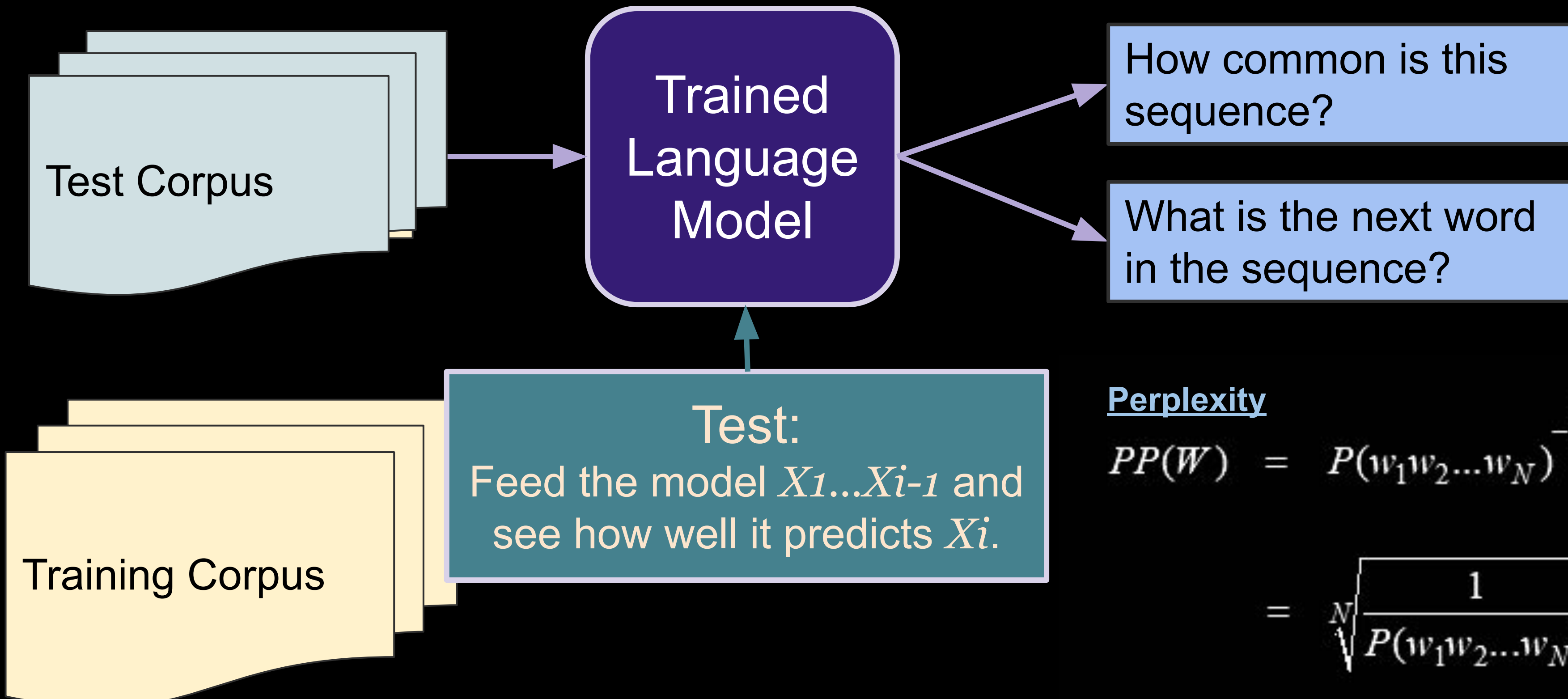
Language Modeling

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



Language Modeling

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

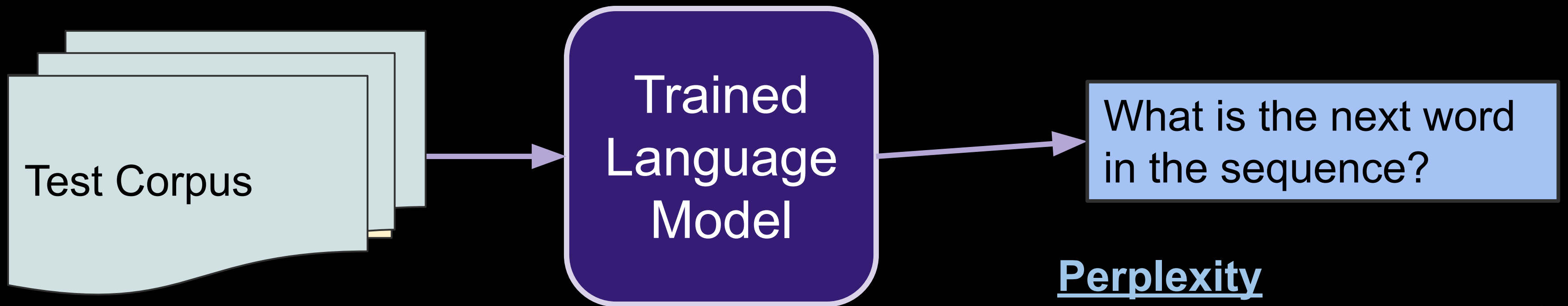


Perplexity

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

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

Evaluation

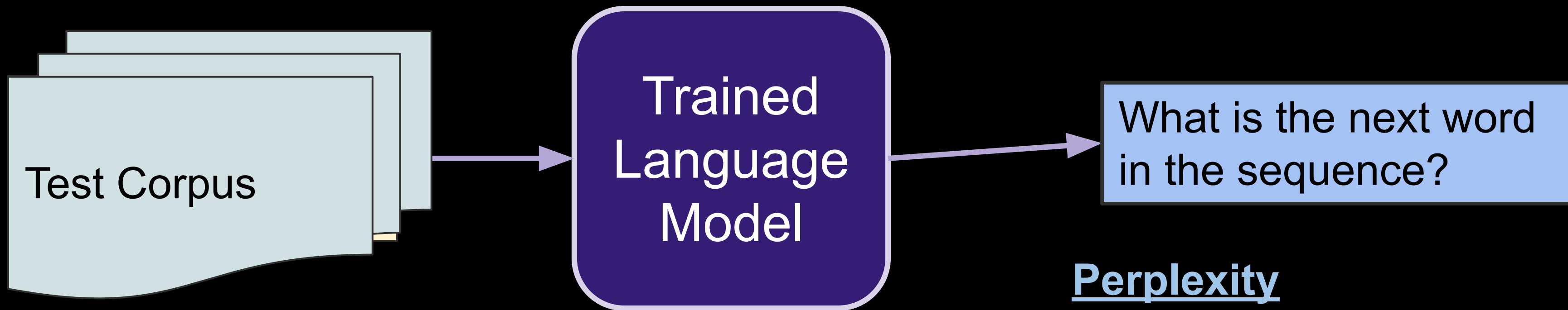


Perplexity

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

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

Evaluation



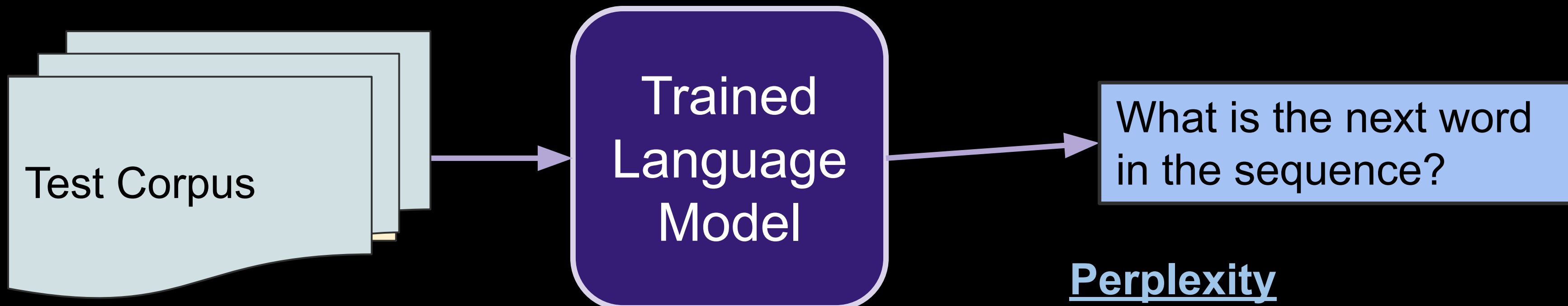
Perplexity

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

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

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

Coding Example: Modeling Tweets from POS data

1. Count unigrams, bigrams, and trigrams
2. Train probabilities for unigram, bigram, and trigram models (over training)
3. Generate language

Trigram model when good evidence (high counts)

Backing off to bigram or even unigram

Coding Example: Modeling Tweets from POS data

Practical Considerations:

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

- Out-of-vocabulary (OOV)
  Choose minimum frequency and mark as <OOV>

- Sentence start and end: *<s> this is a sentence </s>*

Zeros and Smoothing

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

Example from (Jurafsky, 2017)

Zeros and Smoothing

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

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

Zeros and Smoothing

first word \ second word

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

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

Unsmoothed probs

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	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

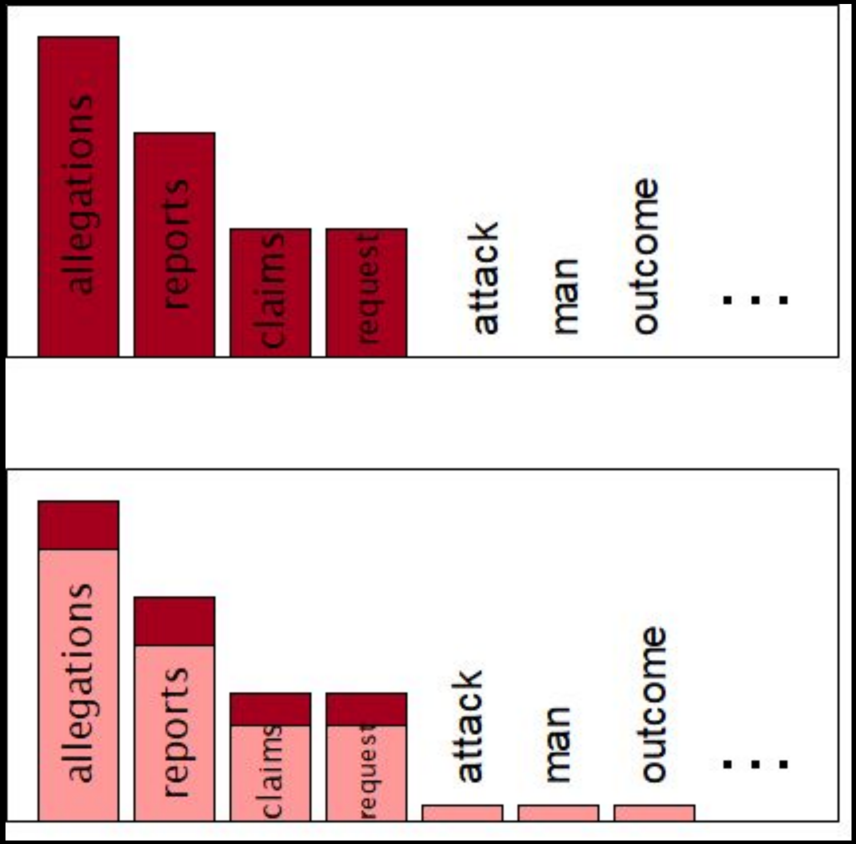
Example from (Jurafsky, 2017)

Why Smoothing? Generalizes

Original

With Smoothing

(Example from Jurafsky / Originally Dan Klein)



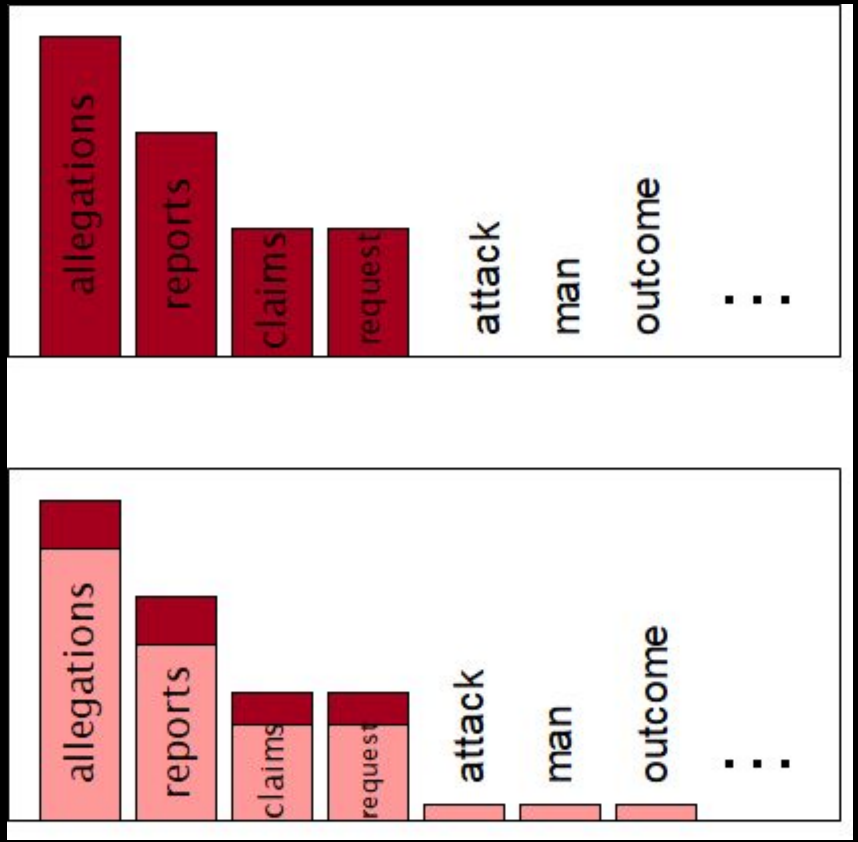
Why Smoothing? Generalizes

Add-one is blunt:
can lead to very large changes.

More Advanced:

Good-Turing Smoothing
Kneser-Nay Smoothing

These are outside scope of course because we will eventually cover, even stronger, deep learning based models.



Why Smoothing?

What about Logistic Regression? Y = next word

$P(Y|X) = P(X_n \mid X_{n-1}, X_{n-2}, X_{n-3}, \ldots)$

Not a terrible option, but X_{n-1} through X_{n-k} would be modeled as independent dimensions. Let's revisit later.

Why Smoothing?

What about Logistic Regression? Y = next word
P(Y|X) = P(Xn | Xn-1, Xn-2, Xn-3, ...)

Not a terrible option, but Xn-1 through Xn-k would be modeled as independent dimensions. Let's revisit later. Could use:
P(Xn | Xn-1, [Xn-1 Xn-2], [Xn-1 Xn-2 Xn-3], ...)

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