
DAY 1 · LANGUAGE MODELS · PART 1 OF 5

What Is a Language Model?

Natural language processing, tokenization, n-gram models and their limits

Pavlos Protopapas

Harvard University



Outline

- › Natural Language Processing
- › Text Preprocessing
- › **Language Modeling**
 - › **Unigrams**
 - › **Bigrams**

Natural Language Processing

Natural Language Processing (NLP) is the field of study that focuses on the interaction between computers and humans through natural language, with the goal of enabling machines to understand, interpret, and respond to human language in meaningful ways.

Natural Language Processing | Applications

Text recognition:

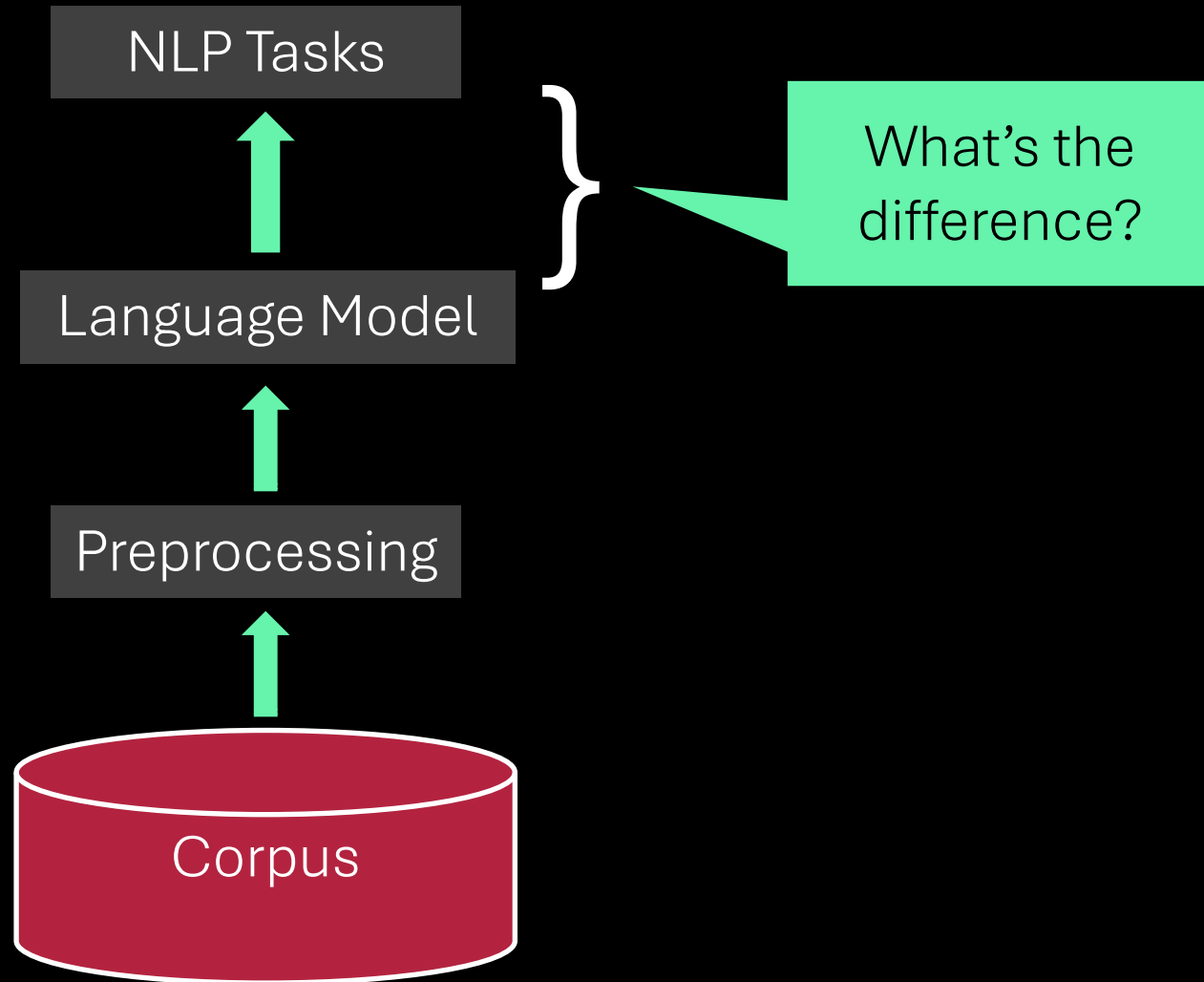


Natural Language Processing | Applications

Sentence prediction:



Pipeline



Game Time

How do we teach a toddler to answer a question? Choose the correct order.

I. Teach the toddler to answer questions

II. Teach the toddler language

A. I -> II

B. II -> I

C. I don't want the toddler to answer questions

D. None of the above

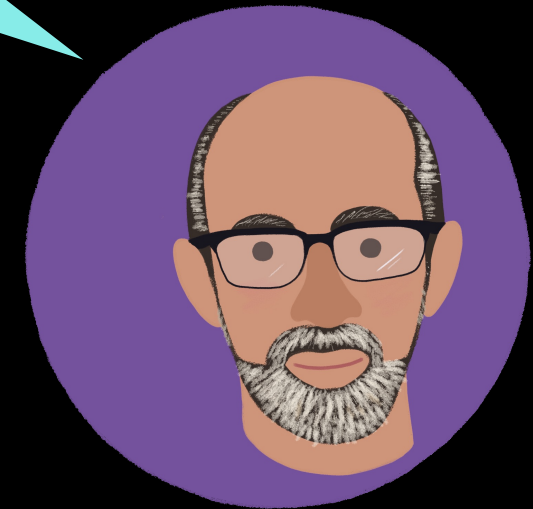
NLP Task vs. Language Model

Similarly, we would want to teach our model to understand language before any downstream tasks like NER, question answering, etc.



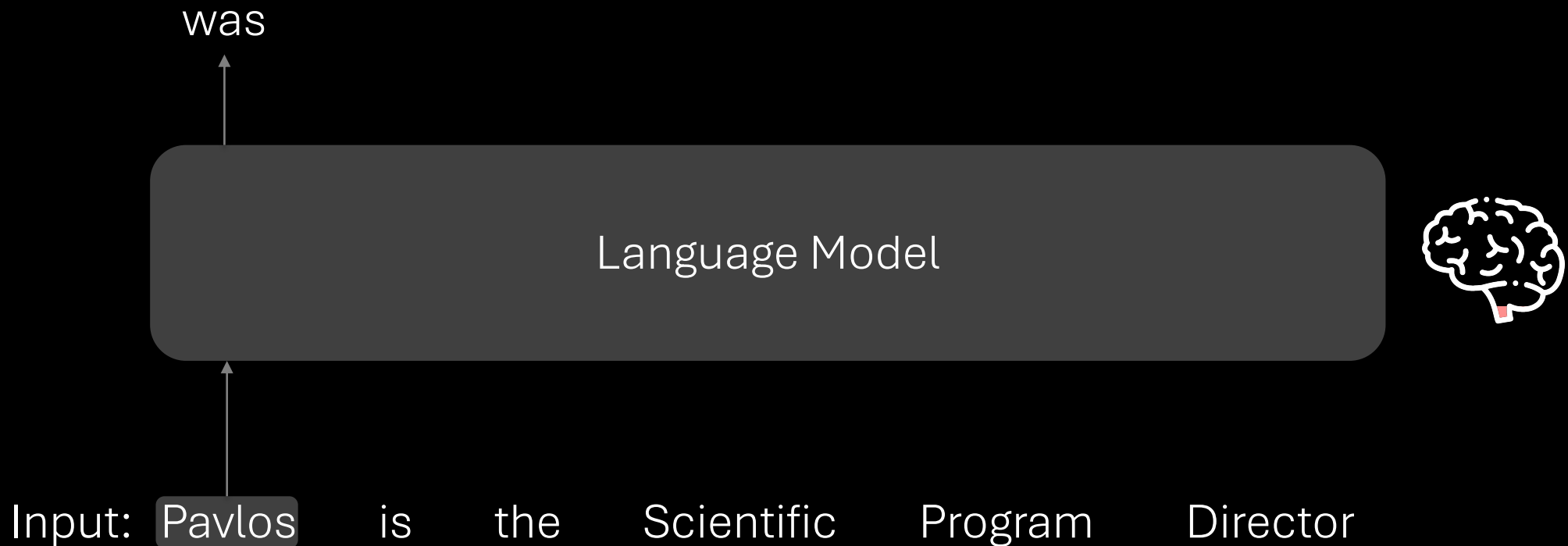
Then what can we do about it?

We can tap into the vast ocean of unstructured text on the internet.



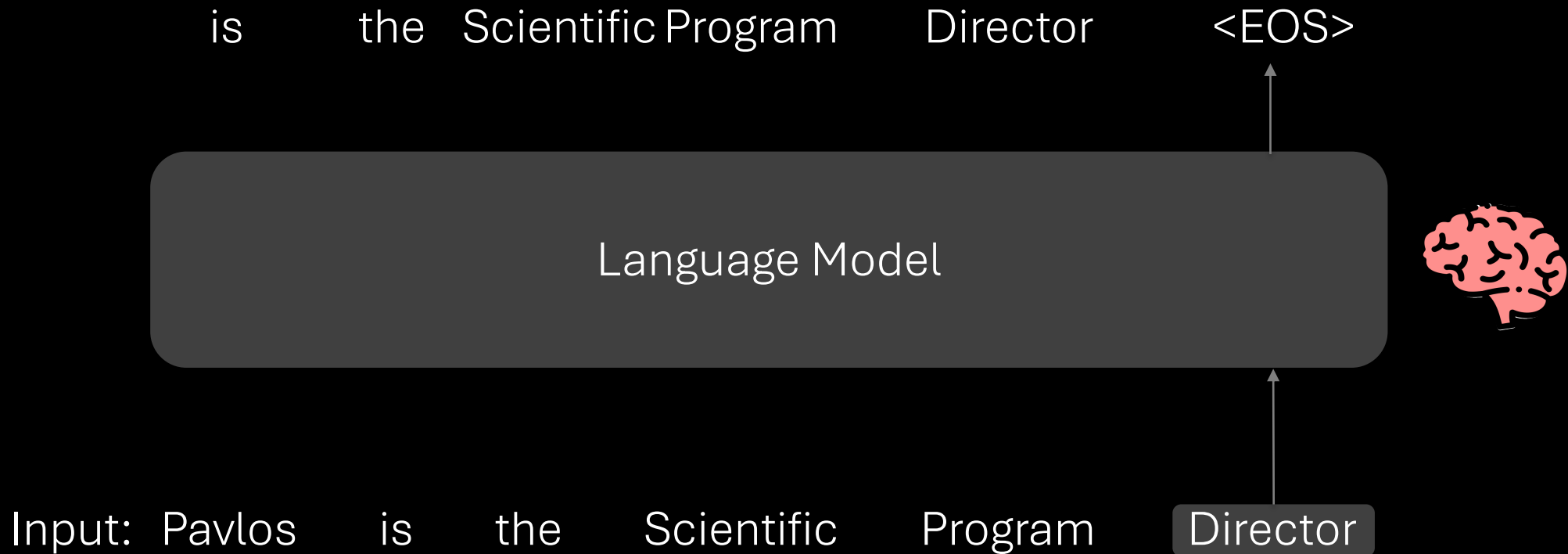
NLP Task vs. Language Model

3. We train the language model using the data that we got from the internet.



NLP Task vs. Language Model

3. We train the language model using the data that we got from the internet.



NLP Task vs. Language Model

4. Finally, we fine-tune our pre-trained language model to perform an NLP task, such as part-of-speech tagging.

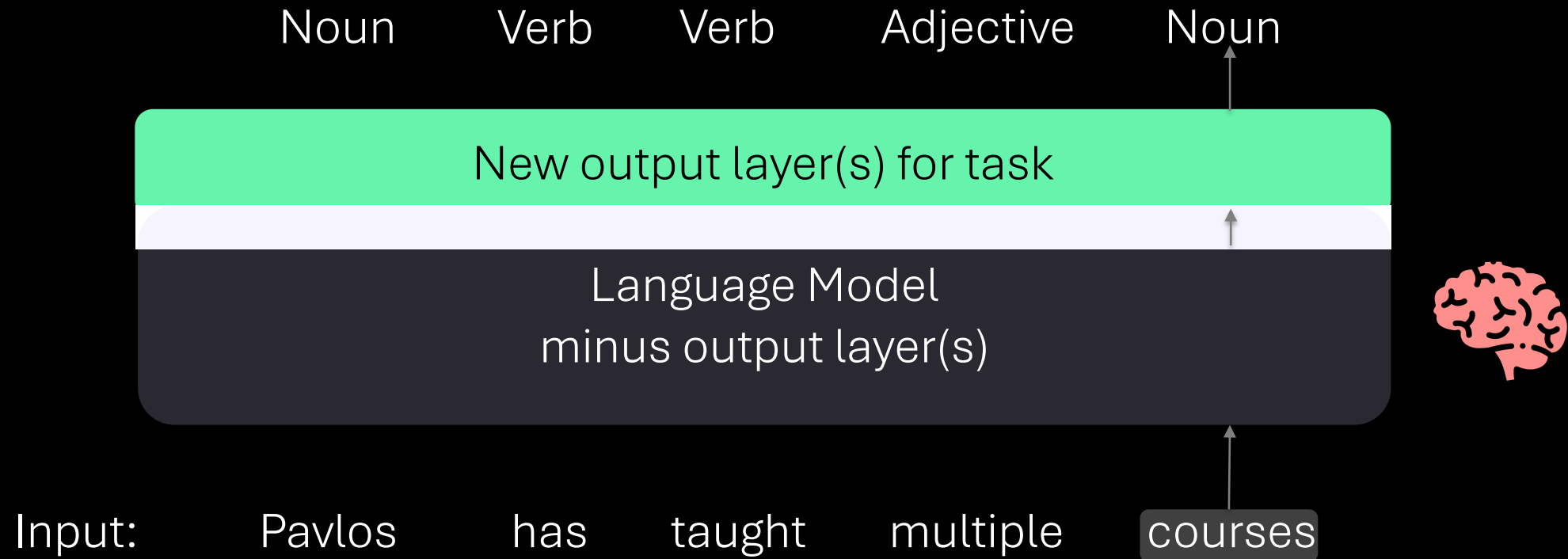
Replace the LM's output layer with
a different one for our new task!



Input: Pavlos has taught multiple courses

NLP Task vs. Language Model

4. Finally, we fine-tune our pre-trained language model to perform an NLP task, such as part-of-speech tagging.



Language Modeling | Starting Point

Text is comprised of a sequence of words that convey meaning. To process it, we break the text down into smaller units known as **tokens**.

Thus, a **sentence** is transformed into a **sequence** of **tokens**.

All the unique tokens from a dataset make up the **vocabulary**.

The larger the vocabulary, the larger the training set required.

We'll also see reasons to include additional 'special' tokens in our vocabulary.

Tokenization

the	award-winning	actor	arrived	today
w_1	w_2	w_3	w_4	w_5

Hyphenated phrases like “award-winning” are not split.

i	haven't	seen	her	since	yesterday
w_1	w_2	w_3	w_4	w_5	w_6

Usually, punctuation is removed before this stage and capitalization is set to lower case.

Conjunctions such as haven't are not split.

Tokenization

Second Approach: sub-word tokenization

Split the words on **statistically significant** fragments.

Tokens often lose their direct interpretability, but the approach is more flexible.

Tokenization

Second Approach: sub-word tokenization

Split the words on **statistically significant** fragments.

How can we define a token?

Sub-word tokenization

removes the **full word**, but keeps the smaller pieces that appear frequently across many words.

the	award	-	winning	actor	arrived	today
w_1	w_2	w_3	w_4	w_5	w_6	w_7

turn	the	handle	counter	clock	wise
w_1	w_2	w_3	w_4	w_5	w_6

Tokenization is an area of research in itself.

Language Modeling | Formal Definition

We model any sequential data as follows:

$$P(x_1, \dots, x_T) = \prod_{t=1}^T p(x_t | x_{t-1}, \dots, x_1)$$

Joint distribution of
all measurements

We multiply a new
term for every token —
the joint probability is
their product.

Conditional probability of
an event, depends on all
events that occurred
before it

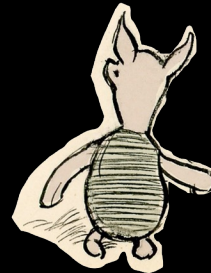
Example

The probability of the next on-screen character can be computed as:

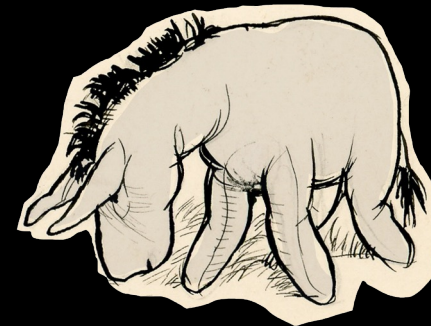
Scene 1



Scene 2



Scene 3



$$P(\text{tiger}, \text{caterpillar}, \text{bear}) = \underbrace{P(\text{tiger})}_{\text{Scene 1}} \underbrace{P(\text{caterpillar} \mid \text{tiger})}_{\text{Scene 2}} \underbrace{P(\text{bear} \mid \text{caterpillar}, \text{tiger})}_{\text{Scene 3}}$$

Example

Having trained a **language model** means that we can reasonably predict the behavior of the sequences.

If we have a sequence of length N , we can predict the most likely next event (i.e., sequence of length $N + 1$)

$$P(\text{tiger}, \text{frog}, S_3) = \underbrace{P(\text{tiger})}_{\text{Scene 1}} \underbrace{P(\text{frog} | \text{tiger})}_{\text{Scene 2}} \underbrace{P(S_3 | \text{frog}, \text{tiger})}_{\text{Scene 3}}$$

Language Modeling as a Sequence of Events

A **language model** estimates the probability of any sequence of words.

Let $X = \text{"Shiv was late for class"}$
 $w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5$

$$P(X) = P(\text{"Shiv was late for class"})$$

Language Modeling | Unigrams

How can we build a language model?

$$P(x_1, \dots, x_T) = \prod_{t=1}^T p(x_t)$$

Naive Approach: Unigram model

Assume each word is independent of all others.

Count how often each word occurs (in the training data).

Let X = “Pavlos loves giving surprise quizzes”

w_1

w_2

w_3

w_4

w_5



Language Modeling | Unigrams

How can we build a language model?

Naive Approach: Unigram model

Assume each word is independent of all others

Let X = "Pavlos loves giving surprise quizzes"

w_1

w_2

w_3

w_4

w_5

You calculate and store each of these probabilities from the training corpus.

$$P(X) = P(\text{Pavlos}) P(\text{loves}) P(\text{giving}) P(\text{surprise}) P(\text{quizzes})$$

$$= 0.00015 * 0.01 * 0.004 * 0.03 * 0.0035$$

$$= 6.3 \times 10^{-13}$$

$$P(x_1, \dots, x_T) = \prod_{t=1}^T p(x_t)$$

Language Modeling | Unigram Issues

Context does not affect probabilities

$$P(\text{"Pavlos loves giving surprise quizzes"}) = P(\text{"quizzes loves giving surprise Pavlos"})$$

Sequence generation: What's the most likely next word?

Pavlos loves giving surprise quizzes all ____

Pavlos loves giving surprise quizzes all the

Pavlos loves giving surprise quizzes all the the

Out of vocabulary words

$$P(\text{"hungry"}) = 0$$

Language Modeling | Unigram Issues

Before Smoothing :

$$P(\text{"hungry"}) = 0$$

After Smoothing :

$$P(\text{"UNK"}) = \frac{\alpha}{|W| + \alpha |V|} > 0$$

Smoothing allows probability of "UNK" token to be non-zero, enabling the model to predict out-of-vocabulary words.

Language Modeling | Bigrams

How can we build a language model that uses context?

Easiest Approach: bigram model

Look at **pairs** of consecutive words

$$P(x_1, \dots, x_T) = p(x_1) \prod_{t=2}^T p(x_t | x_{t-1})$$

Let $\mathbf{X}$ = “Pavlos loves giving surprise quizzes”

	w_1	w_2	w_3	w_4	w_5
	Pavlos	loves	giving	surprise	quizzes
probability					

$$P(\mathbf{X}) = P(\text{Pavlos}) P(\text{loves}|\text{Pavlos}) P(\text{giving}|\text{loves}) P(\text{surprise}|\text{giving}) \times P(\text{quizzes}|\text{surprise})$$

Language Modeling | Bigrams

You calculate each of these probabilities by simply counting the occurrences:

$$P(\text{quizzes}|\text{surprise}) = \frac{C(W_1 W_2)}{C(W_1 w)} = \frac{C(W_1 W_2)}{C(W_1)} = \frac{\text{Total number of "surprise quizzes"}}{\text{Total number "surprise"}}$$

We must store all these probabilities somewhere to be accessed at inference (prediction) time.

Let $\mathbf{X}$ = “Pavlos loves giving surprise quizzes”

				probability	
	w_1	w_2	w_3	w_4	w_5

$$P(\mathbf{X}) = P(\text{Pavlos}) P(\text{loves}|\text{Pavlos}) P(\text{giving}|\text{loves}) P(\text{surprise}|\text{giving}) P(\text{quizzes}|\text{surprise})$$

Language Modeling | Bigrams

- › When a word is **out-of-vocabulary**, it's assigned a probability of 0, causing the entire sentence or sequence to also have a **probability of 0**.
- › **More context** (like trigrams, 4-grams) is often **desired**, but sparsity is a challenge due to **infrequent** subsequences.
- › As we expand the window size for context, **storage issues** arise.
- › **Raw counts** fail to convey deep **semantic** relationships, such as the similarity between 'vehicle' and 'car'.