Washington University in St. Louis

College of Arts & Sciences

3 Day Seminar

LLM & Transfer Leaning in Python
TRIADS Training Series

Ishita Gopal

March 25th, 26th, 27th, 2024

Plan for Day 1

- Overview of Transformers
  - Model Architecture
  - Attention
  - Pretraining tasks
  - Fine tuning tasks

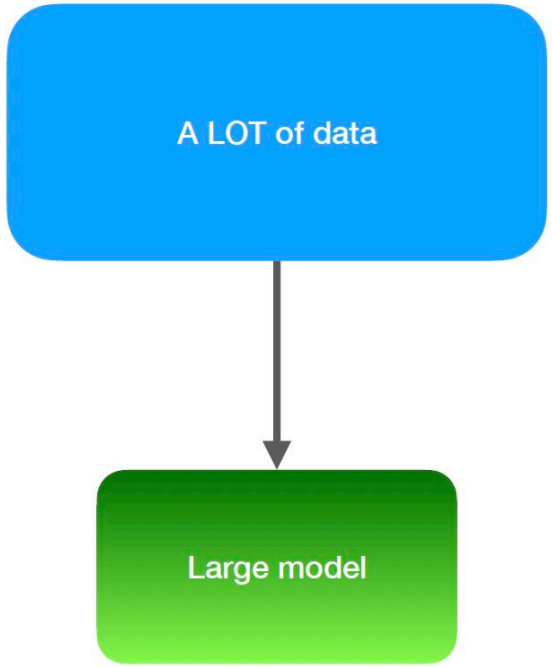
- Colab walkthrough:
  - Introduction to the HuggingFace and the transformers library
  - Off the shelf prediction with different transformer models
  - First step for fine tuning – learning about tokenizers

Day 2 and 3

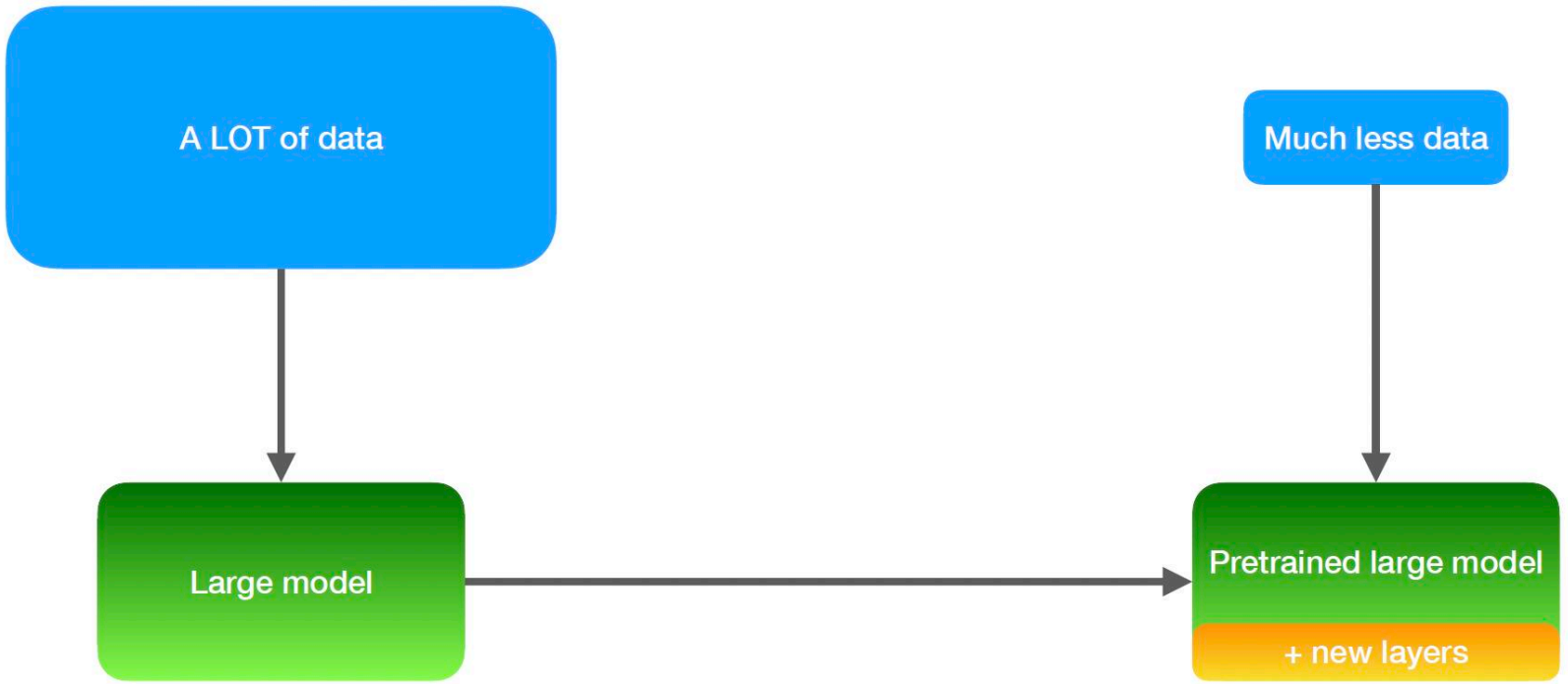
- Day 2
  - Colab Walkthrough of how to implement fine tuning

- Day 3
  - Overview of hyperparameter tuning
  - Colab Walkthrough of how to implement hyperparameter tuning
  - Colab Walkthrough on Text Translation Using a Pretrained Transformer

Transfer Learning



Traditional Machine Learning:
slow training on a lot of data



Traditional Machine Learning:
slow training on a lot of data

Transfer learning:
fast training on a little data

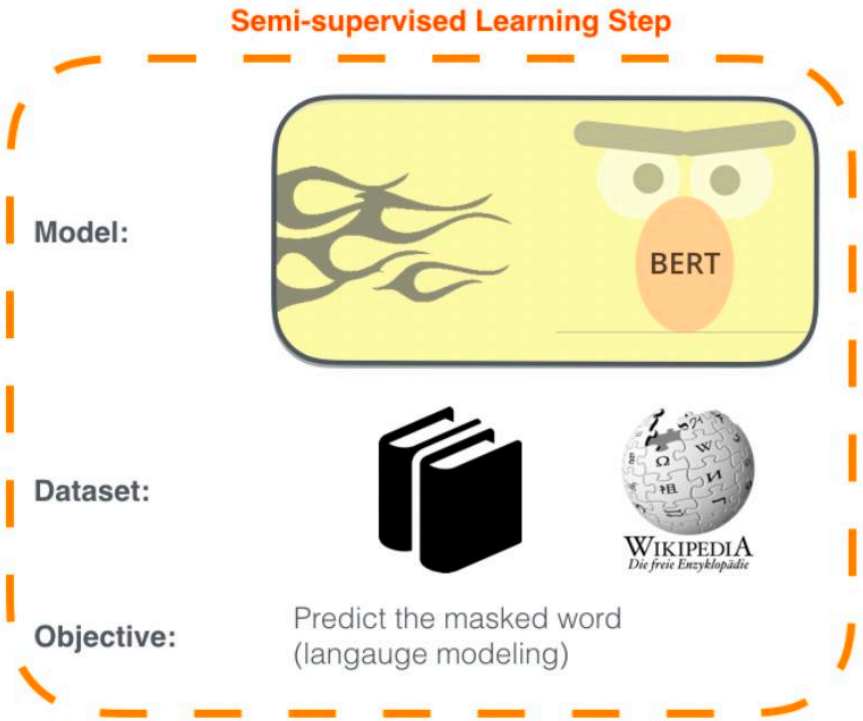
Some use cases

Transfer Learning

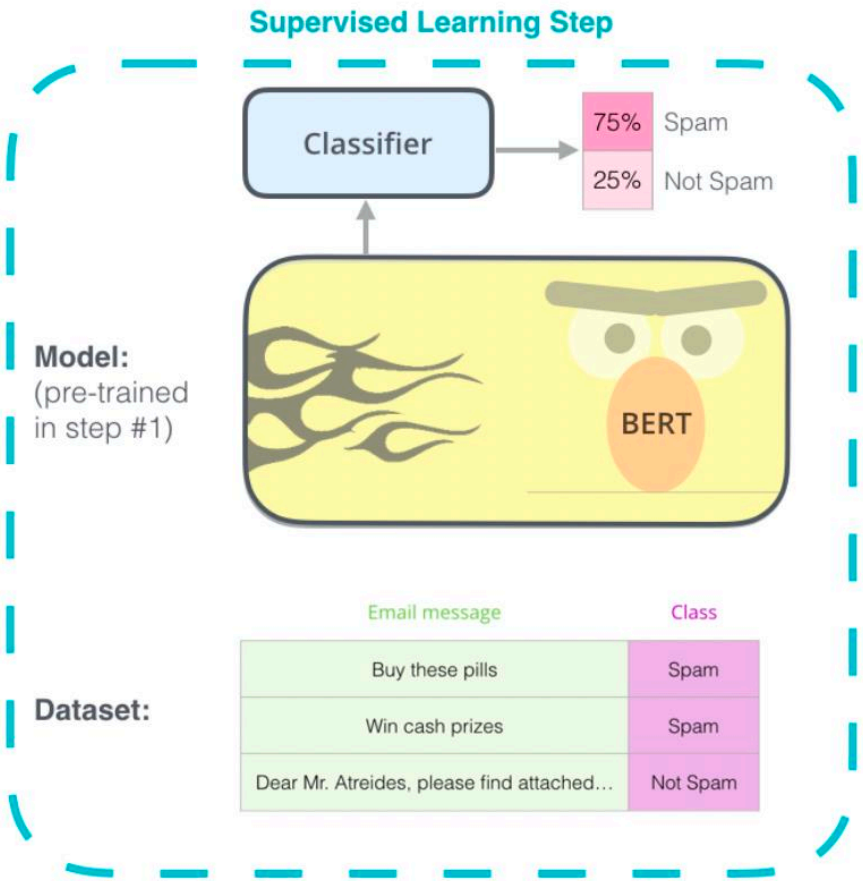
A typical use case

1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.



2 - Supervised training on a specific task with a labeled dataset.



Source: Alammar (2018)

Online Incivility in the 2020 Congressional Elections

Political Research Quarterly
2022, Vol. 75(2) 512–526

DOI: 10.1177/10659129221078863
journals.sagepub.com/home/prq
SAGE

Michael Heseltine[1] and Spencer Dorsey[2]

Abstract

This paper explores the prevalence and correlates of political incivility among Congressional candidates in the 2020 election cycle, focusing specifically on which types of candidates were most likely to use uncivil language in their online communications and the self-reinforcing nature of incivility between candidates. Based on a comprehensive analysis of more than two million tweets sent by major party candidates in the 2020 House and Senate races, we conclude that several individual and electoral factors were influential in driving candidate incivility. Specifically, Republicans, challengers, and candidates in less competitive races were more likely to use uncivil rhetoric. Women, racial minorities, and candidates running in open seat races were less prone to incivility. We also find that incivility begets incivility, with candidates whose opponents used higher rates of incivility also being more likely to use incivility themselves. Uncivil tweets were also found to generate significantly more likes and retweets, suggesting that incivility is a viable means of driving engagement for candidates. These results shed light on the factors behind incivility among political elites, as well as highlight the feedback effects which contribute to a self-reinforcing rise in political incivility.

Detect incivility in political tweets using BERTweet

Table A1: Custom Model (Tuned Transformer) Performance and Comparison to VADER Sentiment Model

	Precision		Recall		F1-Score		*n*
	VADER	Tuned Transformer	VADER	Tuned Transformer	VADER	Tuned Transformer	
Neutral/Civil	0.85	0.92	0.85	0.95	0.85	0.94	626
Uncivil	0.46	0.8	0.46	0.71	0.46	0.75	170
Macro Average	0.66	0.86	0.66	0.79	0.66	0.84	796
Weighted Average	0.77	0.9	0.77	0.9	0.77	0.9	796

Attention to the COVID-19 Pandemic on Twitter: Partisan Differences Among U.S. State Legislators

Taegyoon Kim, Nitheesha Nakka, Ishita Gopal, Bruce A. Desmarais, Abigail Mancinelli, Jeffrey J. Harden, Hyein Ko, Frederick J. Boehmke

First published: 15 December 2021 | https://doi.org/10.1111/lsq.12367 | Citations: 1

Abstract

Subnational governments in the United States have taken the lead on many aspects of the response to the COVID-19 pandemic. Variation in government activity across states offers the opportunity to analyze responses in comparable settings. We study a common and informative activity among state officials—state legislators' attention to the pandemic on Twitter. We find that legislators' attention to the pandemic strongly correlates with the number of cases in the legislator's state, the national count of new deaths, and the number of pandemic-related public policies passed within the legislator's state. Furthermore, we find that the degree of responsiveness to pandemic indicators differs significantly across political parties, with Republicans exhibiting weaker responses, on average. Lastly, we find significant differences in the content of tweets about the pandemic by Democratic and Republican legislators, with Democrats focused on health indicators and impacts, and Republicans focused on business impacts and opening the economy.

Identify discussion of Covid-19 using BERT-base

	Precision (S.D)	Recall (S.D)	F-1 (S.D)
RF + Count	0.9713 (0.0187)	0.3701 (0.0343)	0.5354 (0.0384)
RF + TFIDF	0.9746 (0.0240)	0.3382 (0.0268)	0.5017 (0.0324)
RF + GloVe	0.8302 (0.0284)	0.2548 (0.0567)	0.3866 (0.065)
XGB + Count	0.9392 (0.0072)	0.6604 (0.045)	0.7744 (0.0284)
XGB + TFIDF	0.9365 (0.009)	0.6325 (0.0344)	0.7544 (0.0231)
XGB + GloVe	0.7391 (0.0324)	0.4698 (0.0538)	0.5718 (0.0366)
BERT Fine Tuned	0.8504 (0.0471)	0.8428 (0.0474)	0.8446 (0.0139)

Table S2. Performance of Classifiers (5-fold Cross Validation)

PA

Sentiment is Not Stance: Target-Aware Opinion Classification for Political Text Analysis

Samuel E. Bestvater and Burt L. Monroe

Department of Political Science, The Pennsylvania State University, University Park, PA, USA. E-mail: seb654@psu.edu, burtmonroe@psu.edu

Abstract

Sentiment analysis techniques have a long history in natural language processing and have become a standard tool in the analysis of political texts, promising a conceptually straightforward automated method of extracting meaning from textual data by scoring documents on a scale from positive to negative. However, while these kinds of sentiment scores can capture the overall tone of a document, the underlying concept of interest for political analysis is often actually the document's stance with respect to a given target—how positively or negatively it frames a specific idea, individual, or group—as this reflects the author's underlying political attitudes. In this paper, we question the validity of approximating author stance through sentiment scoring in the analysis of political texts, and advocate for greater attention to be paid to the conceptual distinction between a document's sentiment and its stance. Using examples from open-ended survey responses and from political discussions on social media, we demonstrate that in many political text analysis applications, sentiment and stance do not necessarily align, and therefore sentiment analysis methods fail to reliably capture ground-truth document stance, amplifying noise in the data and leading to faulty conclusions.

Keywords: text-as-data, sentiment analysis, political stance, machine learning

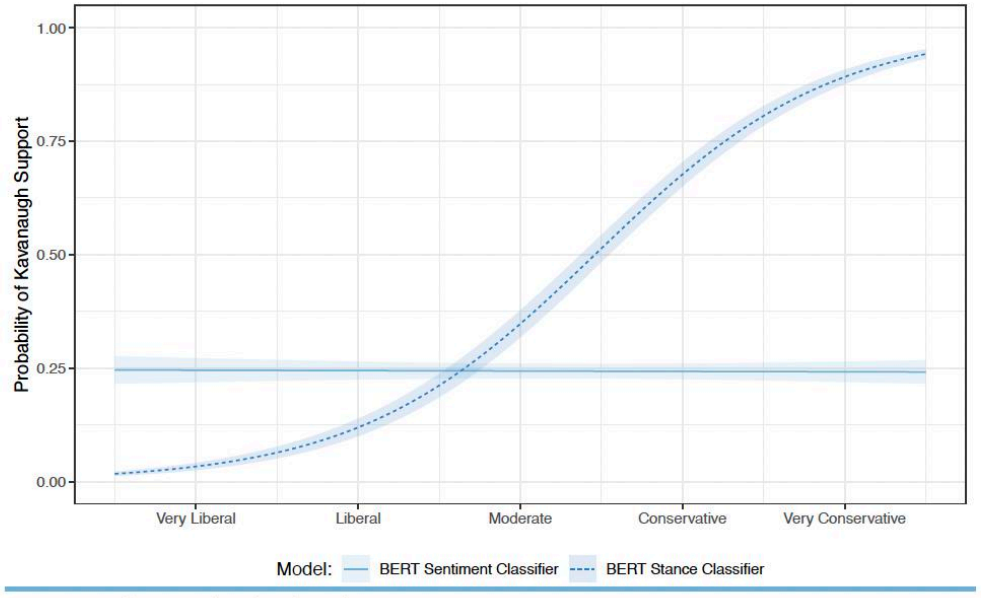
Sentiment and Stance in Tweets About the Kavanaugh Confirmation using BERT-base

Table 6. Classifier Performance: Kavanaugh Tweets

Classifier	F1 Score (Predicting Sentiment)	F1 Score (Predicting Stance)
Lexicoder	0.788 (0.005)	0.572 (0.014)
VADER	0.754 (0.005)	0.514 (0.011)
SVM (sentiment-trained)	0.943 (0.003)	0.514 (0.012)
BERT (sentiment-trained)	0.954 (0.002)	0.582 (0.005)
SVM (stance-trained)		0.935 (0.006)
BERT (stance-trained)		0.938 (0.002)

Reported figures are the average F1 score over 5-fold cross validation
Standard Errors in parentheses

Figure 5. Predicting Kavanaugh Support from Ideology



Source: Bestvater and Monroe, 2020.

How do Transformers work?

Transformers were introduced:

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

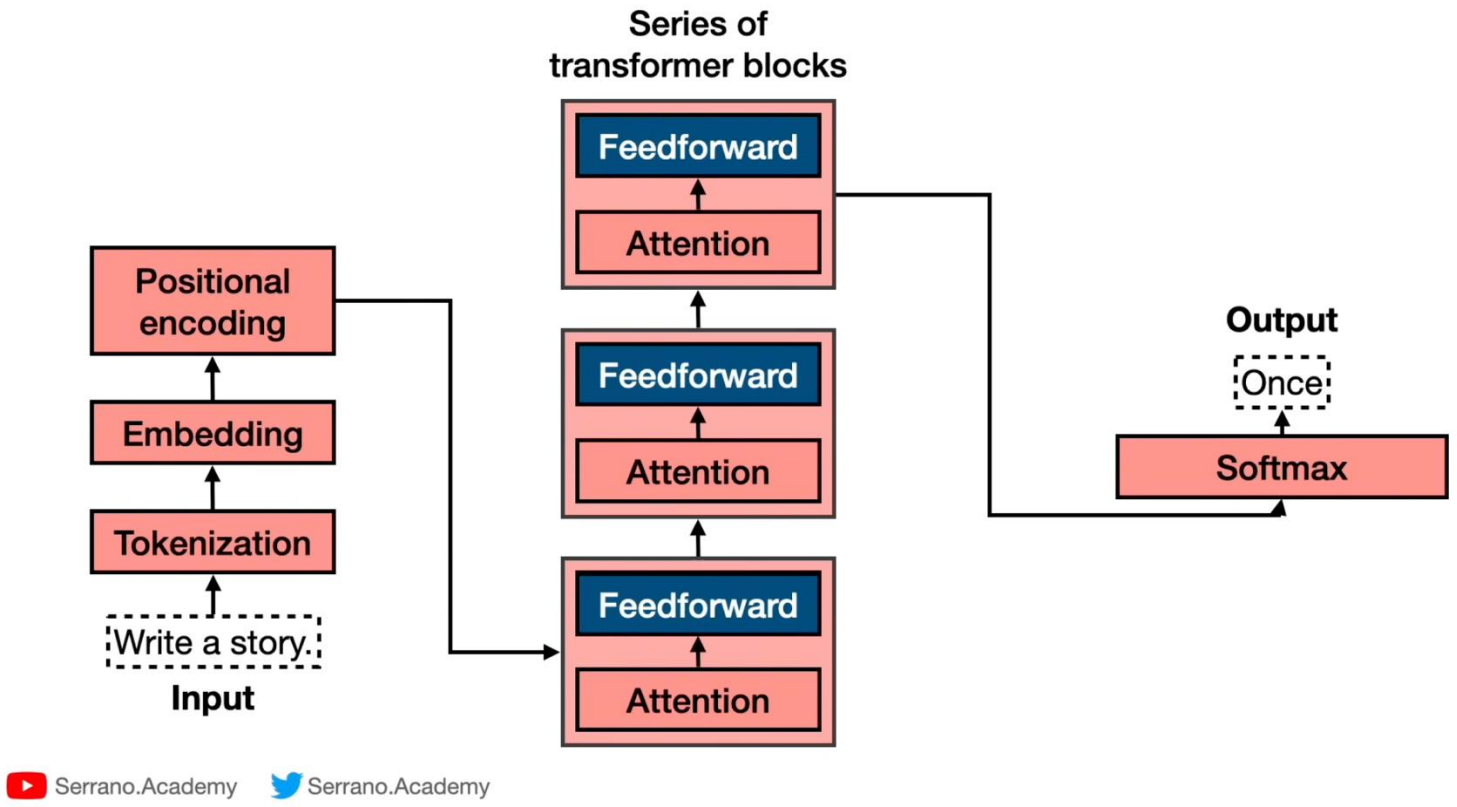
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

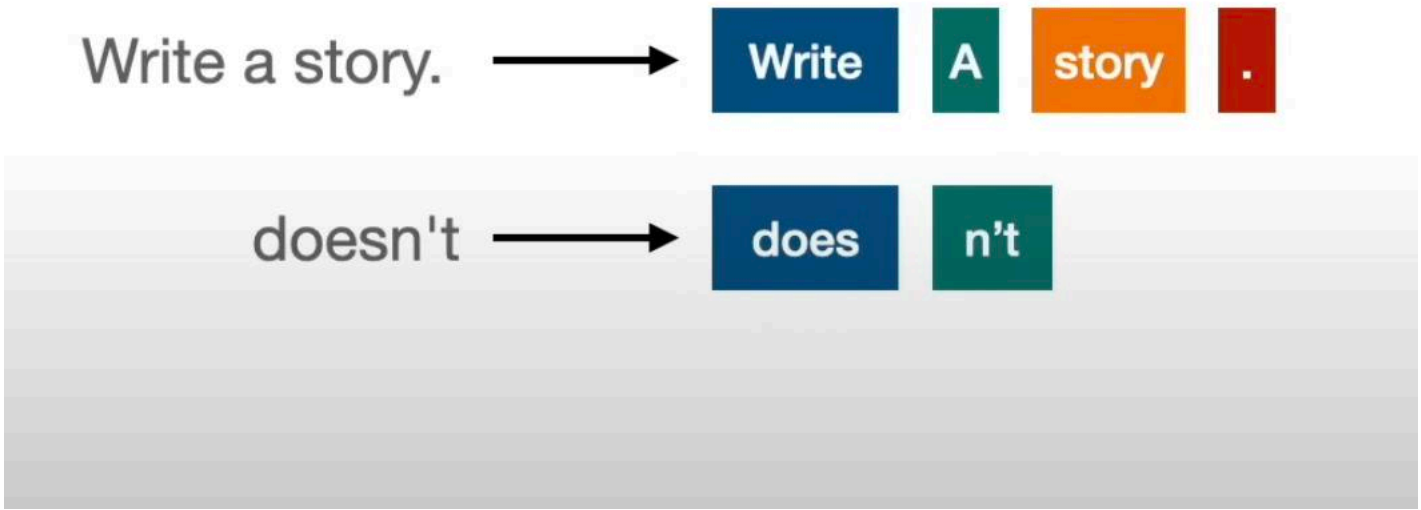
Architecture:

Includes several steps:

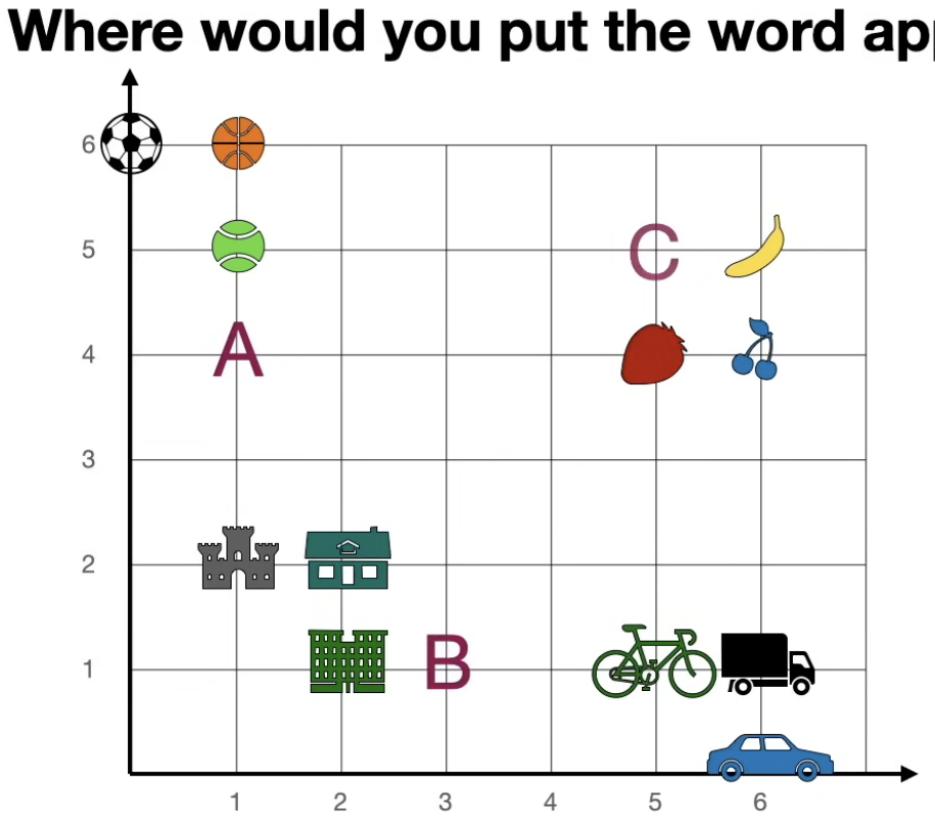
- Tokenization
- Embeddings
- Attention Mechanisms
- Feedforward Neural Networks



Tokenization

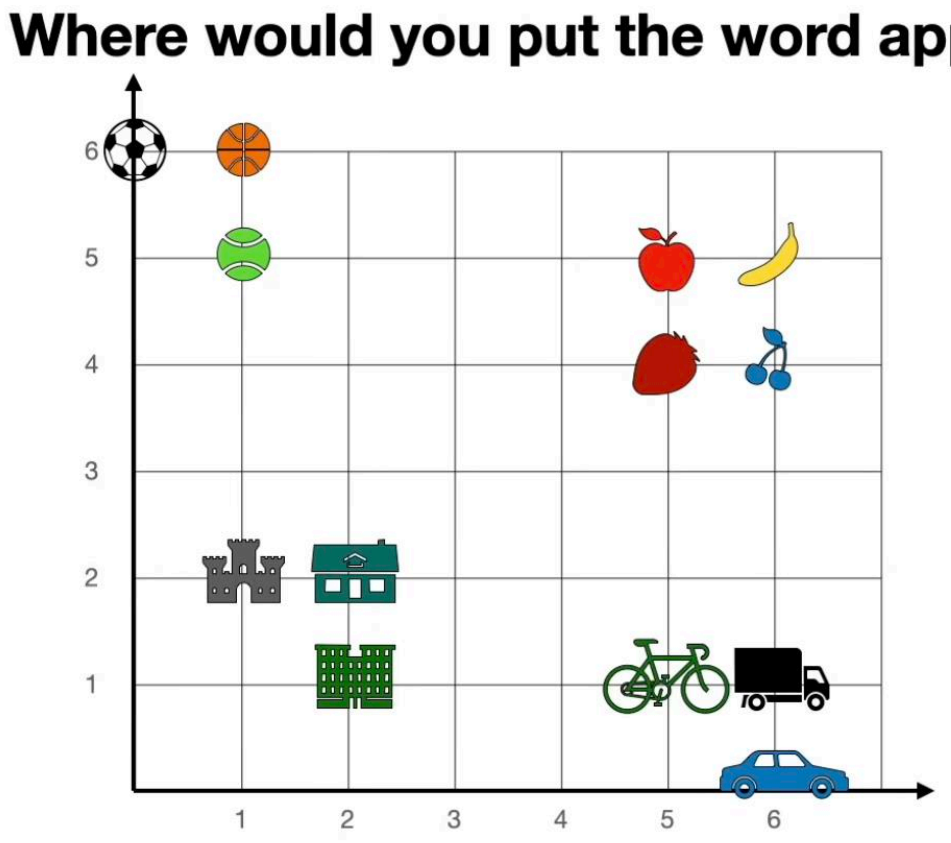


Descriptions of words: Embeddings



Word	Numbers	
Apple	?	?
Banana	6	5
Strawberry	5	4
Cherry	6	4
Soccer	0	6
Basketball	1	6
Tennis	1	5
Castle	1	2
House	2	2
Building	2	1
Bicycle	5	1
Truck	6	1
Car	6	0

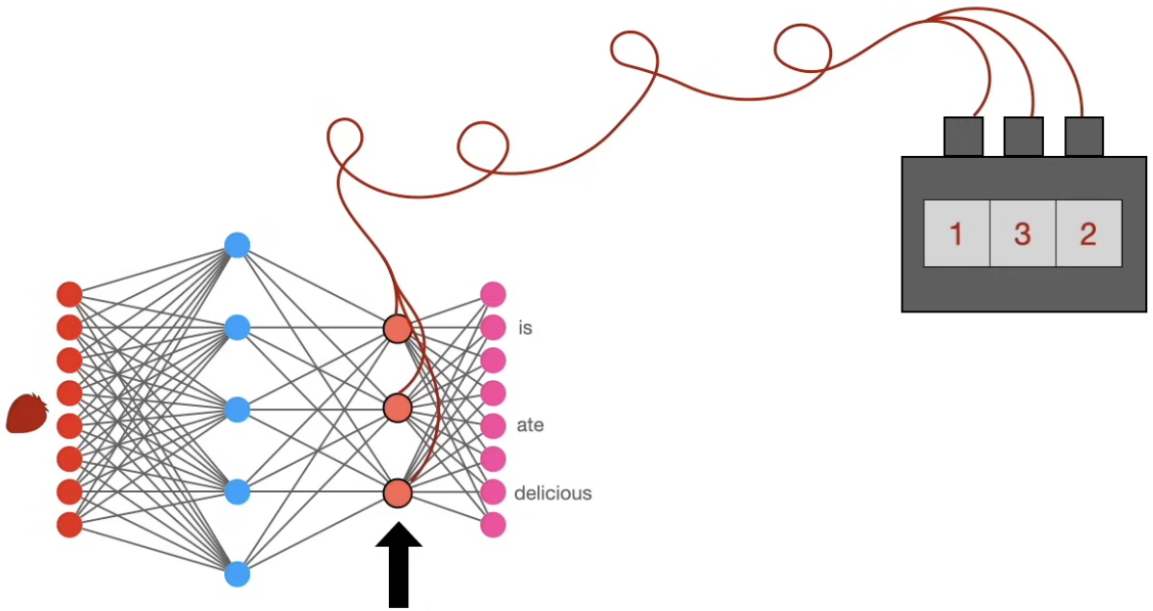
Similar words get similar numbers!



Word	Numbers	
Apple	5	5
Banana	6	5
Strawberry	5	4
Cherry	6	4
Soccer	0	6
Basketball	1	6
Tennis	1	5
Castle	1	2
House	2	2
Building	2	1
Bicycle	5	1
Truck	6	1
Car	6	0

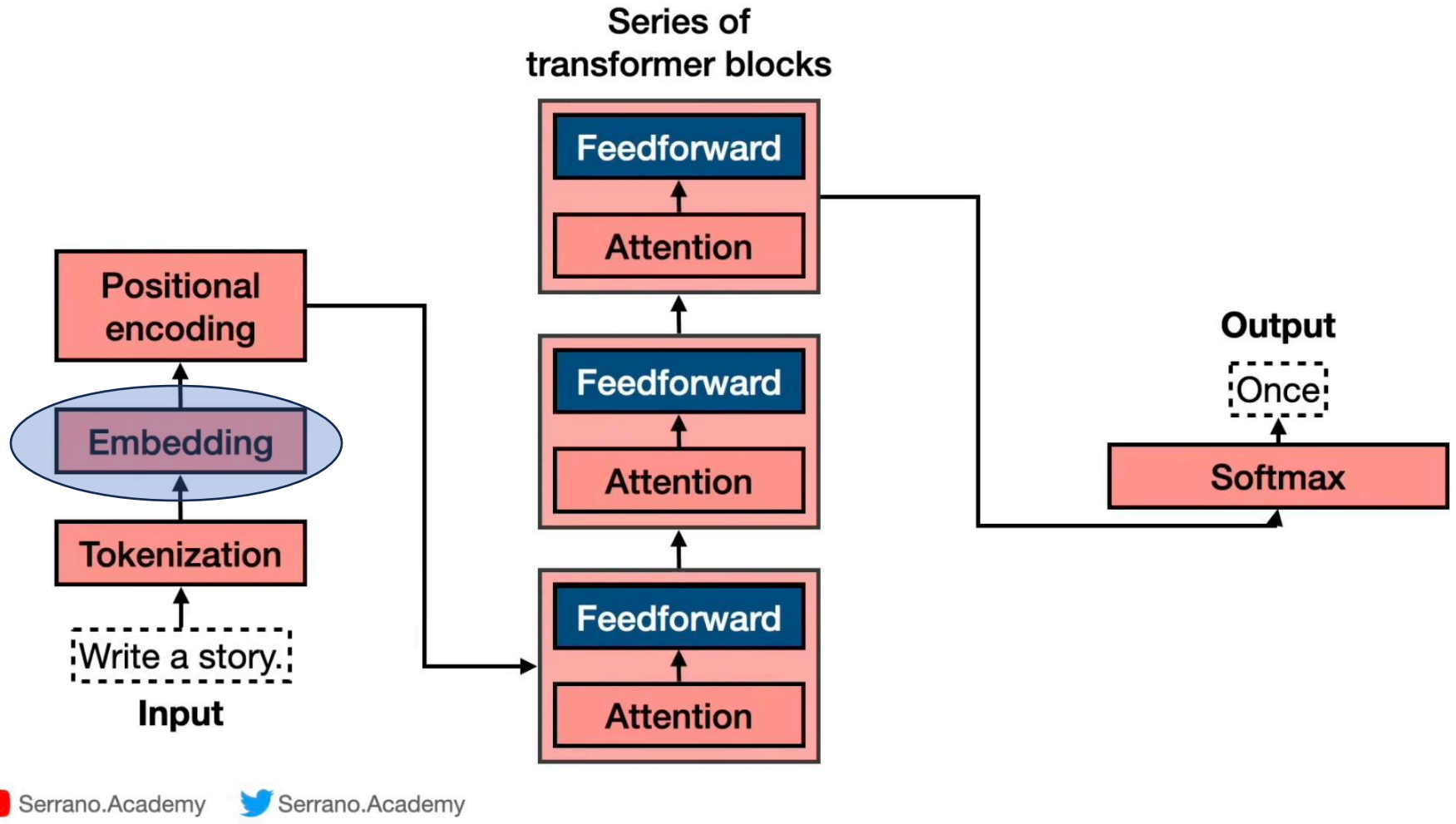
How can we get these numbers?

- Train Neural Networks
- Each layer understands and captures different properties of the word.
- The last layer understands the word pretty well!
- Each word can be associated to hundreds/thousands of numbers

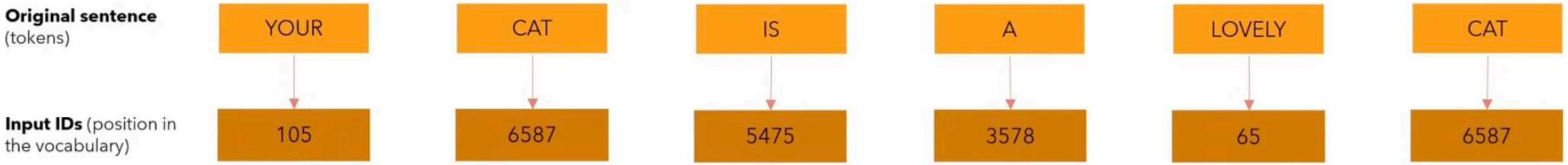


Word	Numbers		
Strawberry	1	3	2
Apple	1.1	2.9	2.2
Castle	7	-5.4	0.4

Going back to the Embeddings step in the Transformer…



Step 1: Map tokens onto numbers called “input ids” (these are fixed)

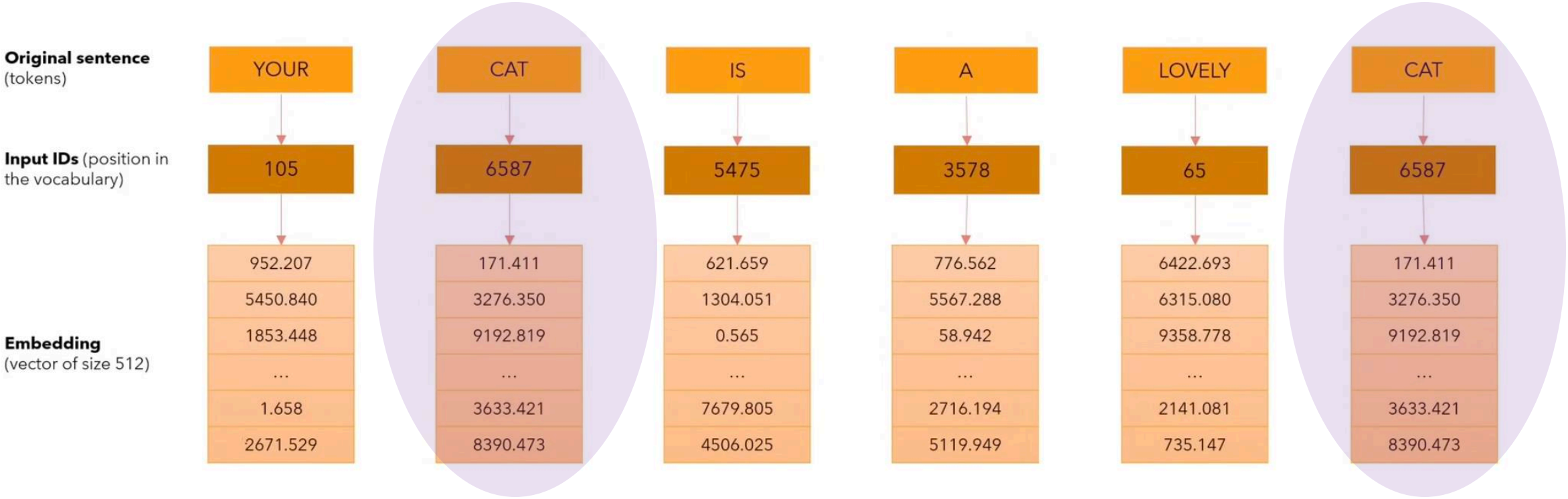


Umar Jamil - https://github.com/hkproj/transformer-from-scratch-notes

Step 2: Map Input Ids to a vector of size 512 → Embeddings!.

Umar Jamil - https://github.com/hkproj/transformer-from-scratch-notes

Step 2: Map Input Ids to a vector of size 512 → Embeddings!.
- Same word → Same embedding



Umar Jamil - https://github.com/hkproj/transformer-from-scratch-notes

Step 2: Map Input Ids to a vector of size 512 → Embeddings!

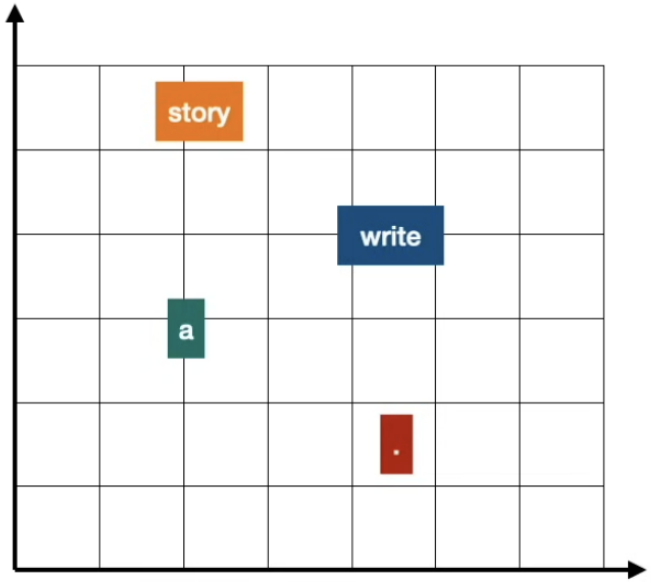
- Same word → Same embedding

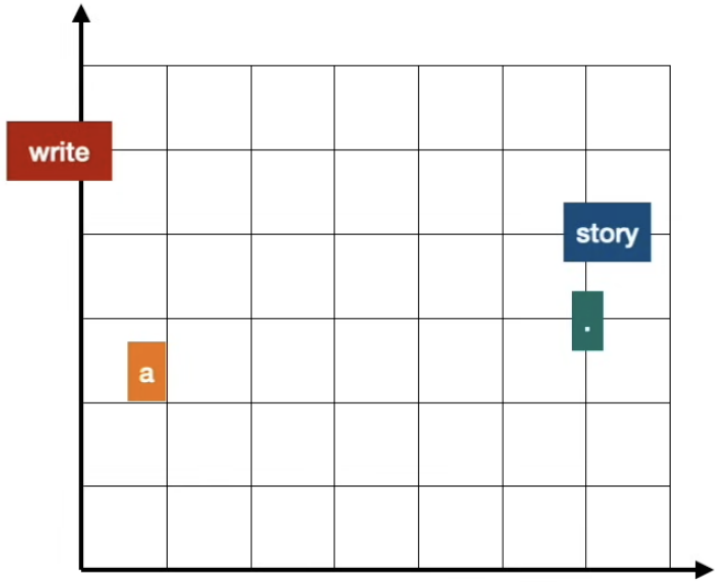
But!!

- Embeddings are not fixed. They are *parameters* for the model.
- Learned & change during training → to best represent *meaning of the word.*

Umar Jamil - https://github.com/hkproj/transformer-from-scratch-notes

Account for word position:

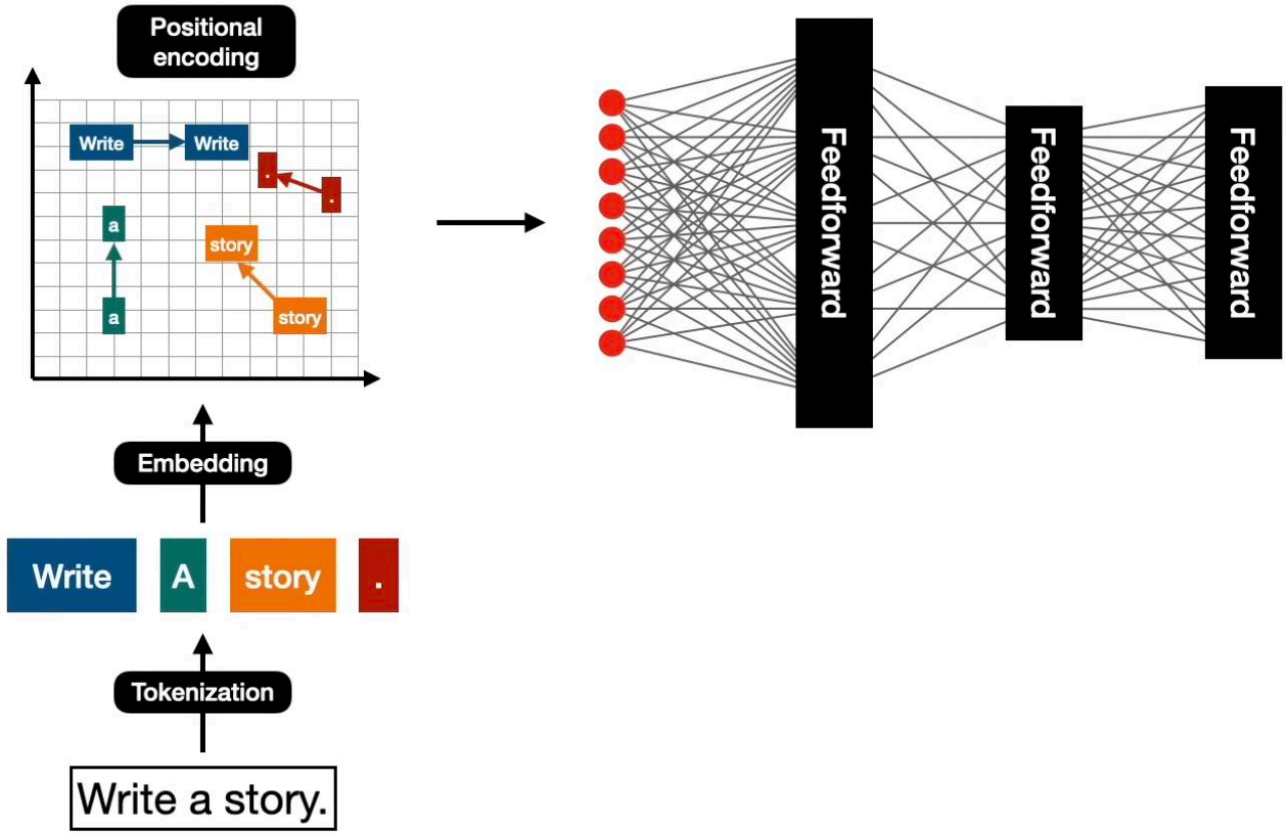


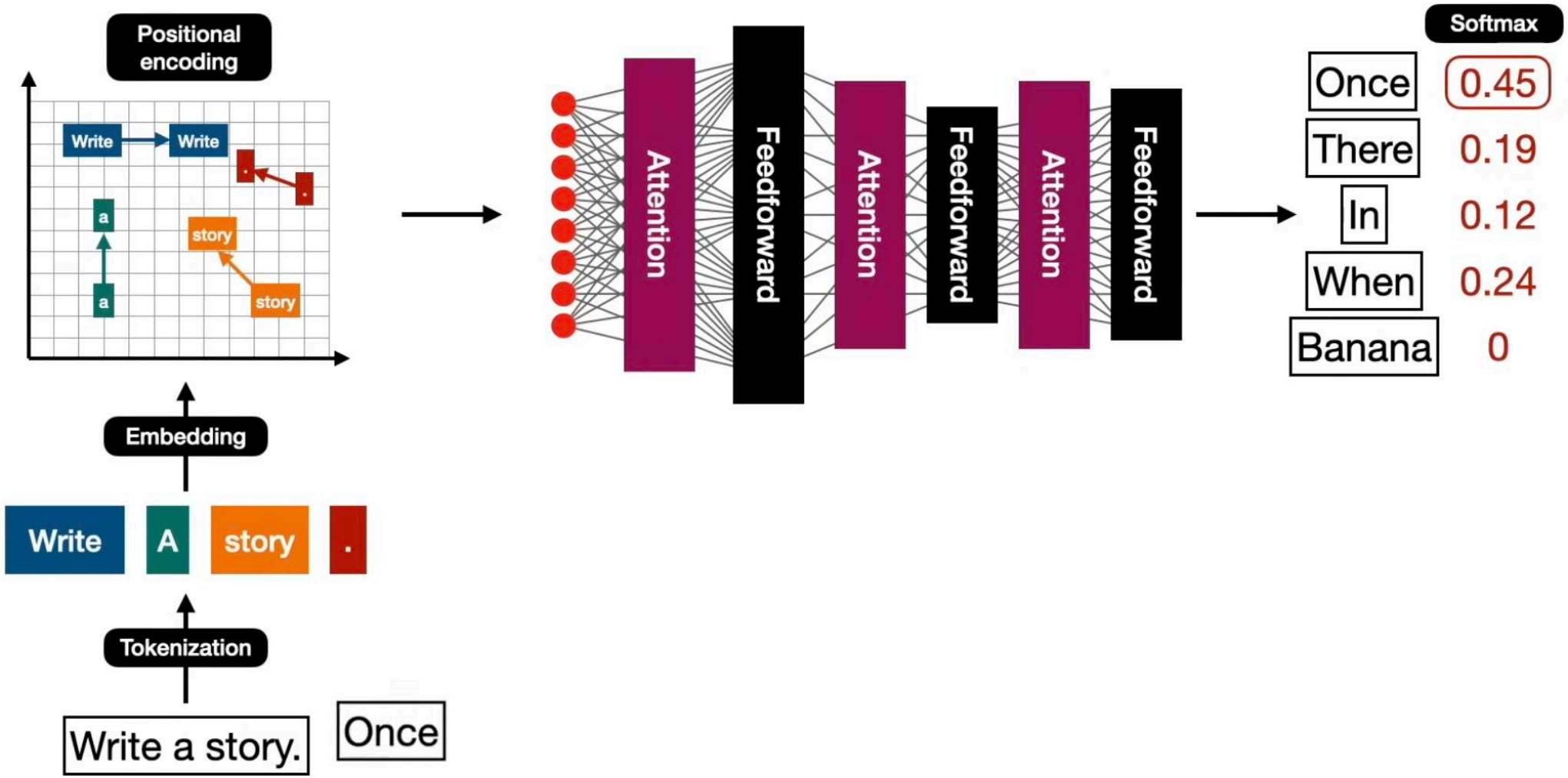


Add that to the embeddings:

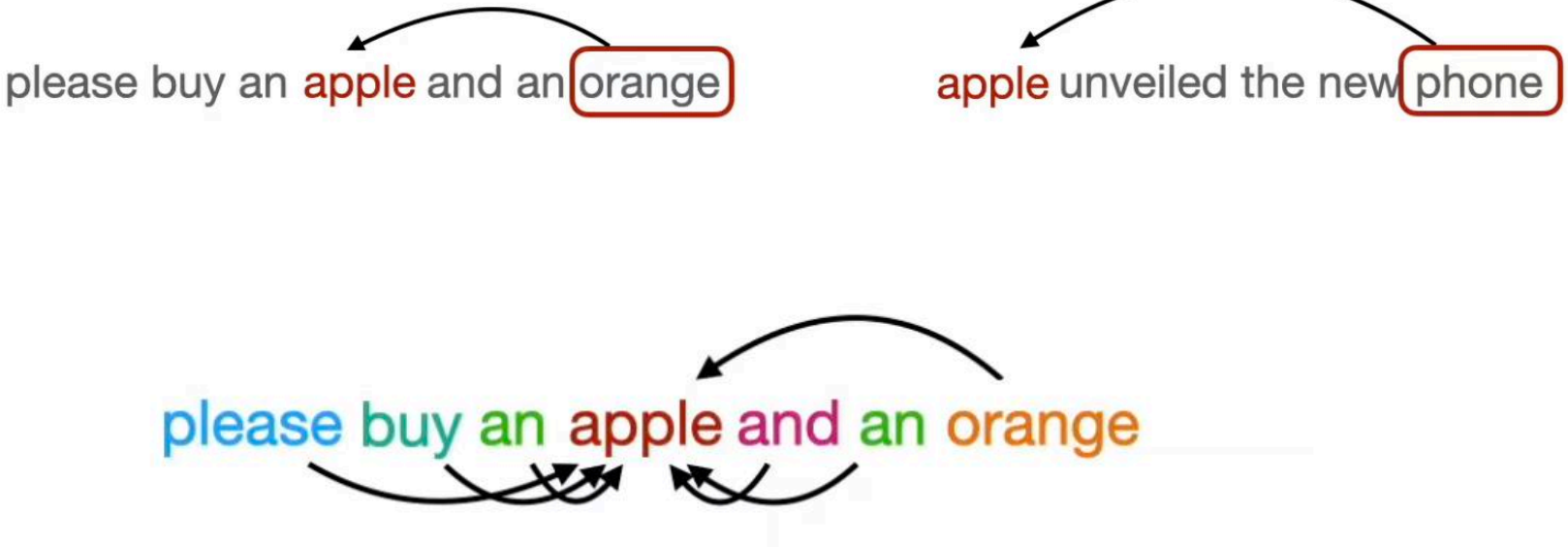
Umar Jamil - https://github.com/hkproj/transformer-from-scratch-notes

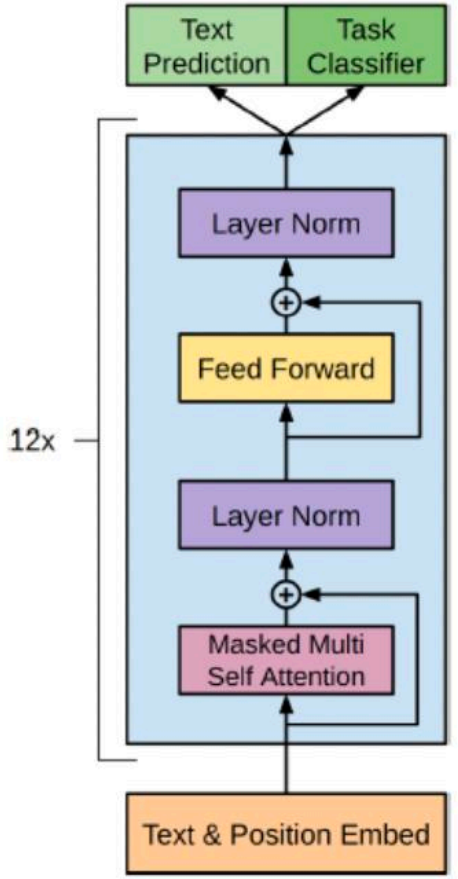
But *still* does not capture "context"

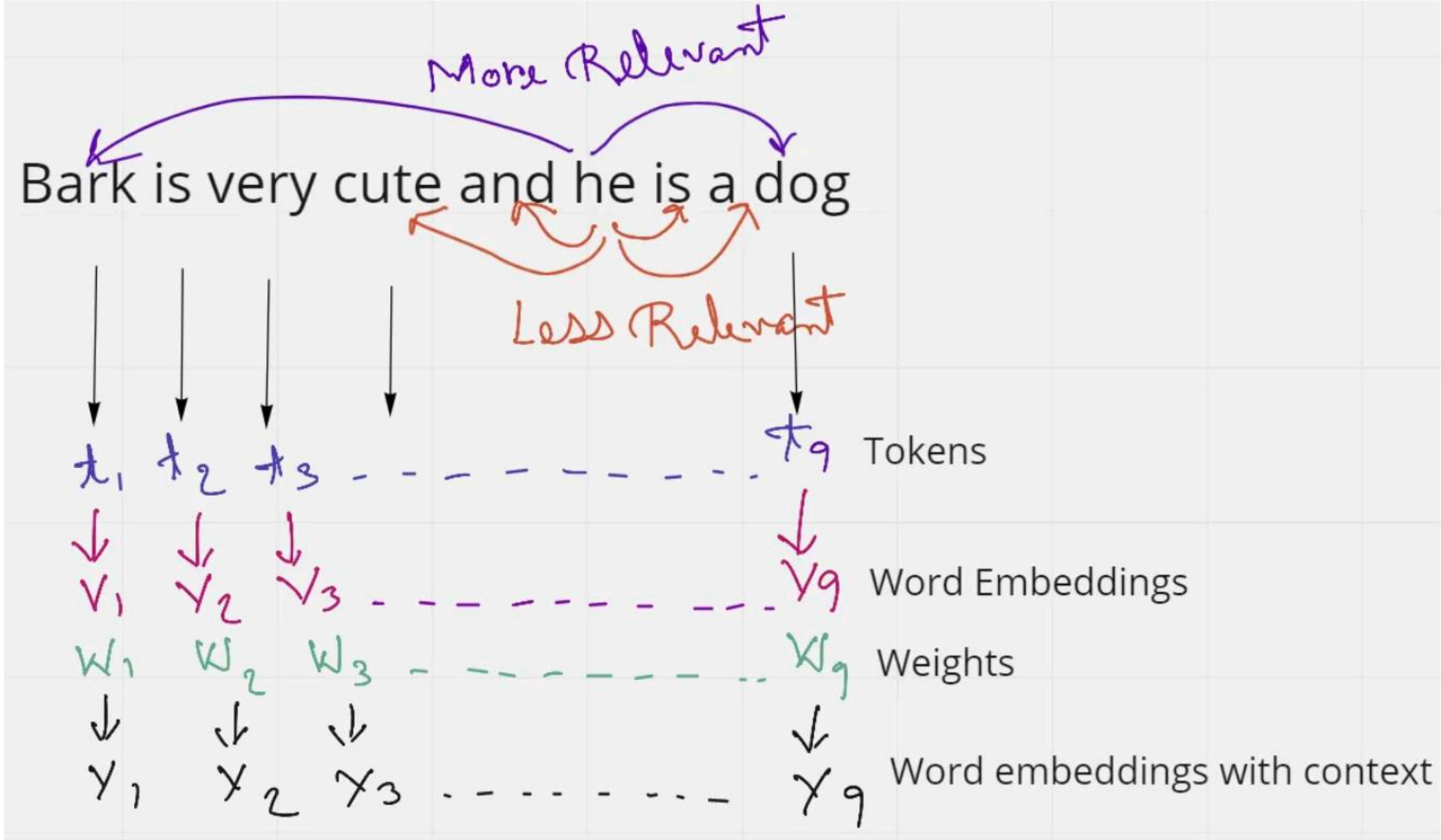



Positional encoding
Write
Write
a
story
a
story
Embedding
Write
A
story
.
Tokenization
Write a story.
Once
Attention
Feedforward
Attention
Feedforward
Attention
Feedforward
Softmax
Once
0.45
There
0.19
In
0.12
When
0.24
Banana
0

Attention for contextualized representations

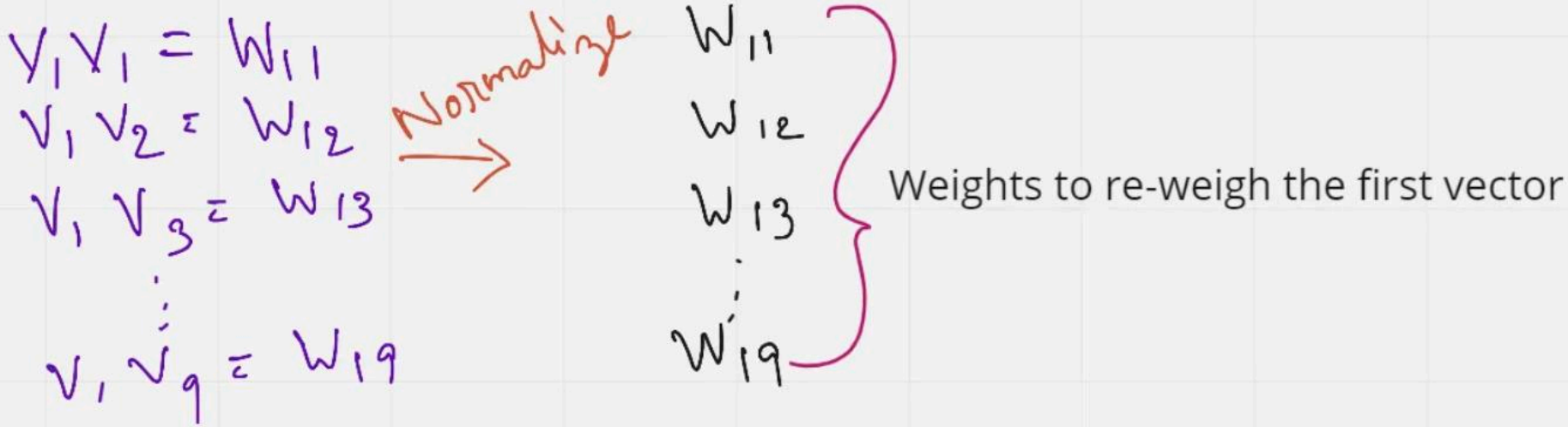






Medium article by Arjun Sarkar

1. Finding the Weights



2 Obtaining Embedding with context

$$w_{11} v_1 + w_{12} v_2 + w_{13} v_3 \ldots + w_{1q} v_q = y_1$$

$$w_{21} v_1 + w_{22} v_2 + w_{23} v_3 \ldots w_{2q} v_q = y_2$$

$$\vdots$$

$$w_{q1} v_1 + w_{q2} v_2 + w_{q3} v_3 \ldots w_{qq} v_q = y_q$$

What is Self-Attention?

Self-Attention allows the model to relate words to each other.

In this simple case we consider the sequence length **seq** = 6 and $\mathbf{d_{model}} = \mathbf{d_k} = 512$.

The matrices **Q**, **K** and **V** are just the input sentence.

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{softmax}\left(\frac{\mathbf{Q}\ (6, 512) \times \mathbf{K^T}\ (512, 6)}{\sqrt{512}}\right) =$$

	YOUR	CAT	IS	A	LOVELY	CAT	Σ
YOUR	0.268	0.119	0.134	0.148	0.179	0.152	1
CAT	0.124	0.278	0.201	0.128	0.154	0.115	1
IS	0.147	0.132	0.262	0.097	0.218	0.145	1
A	0.210	0.128	0.206	0.212	0.119	0.125	1
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174	1
CAT	0.195	0.114	0.203	0.103	0.157	0.229	1

(6, 6)

* a

* for simplicity I considered only one head, which makes $d_{model} = d_k$.

Question

- Why are the scores not symmetric?

- After computing the raw attention scores → SoftMax normalization

- Row tokens (queries) attend to column tokens (keys)
- The matrix weights represent a way to probabilistically measure where attention is directed to when querying over keys
- i.e. to which key - and so to which token of the sentence - each query (token) mainly focuses to)

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

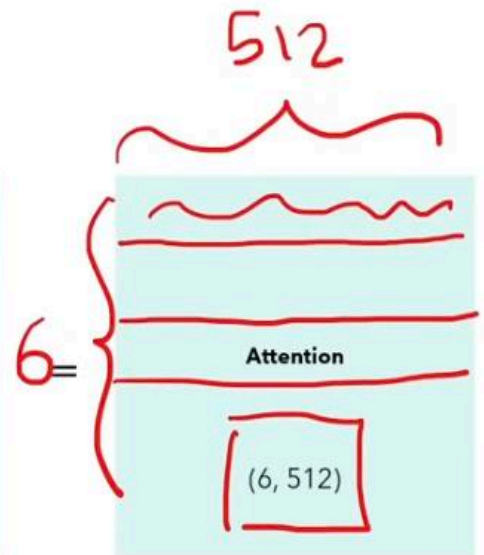
	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229

(6, 6)

X

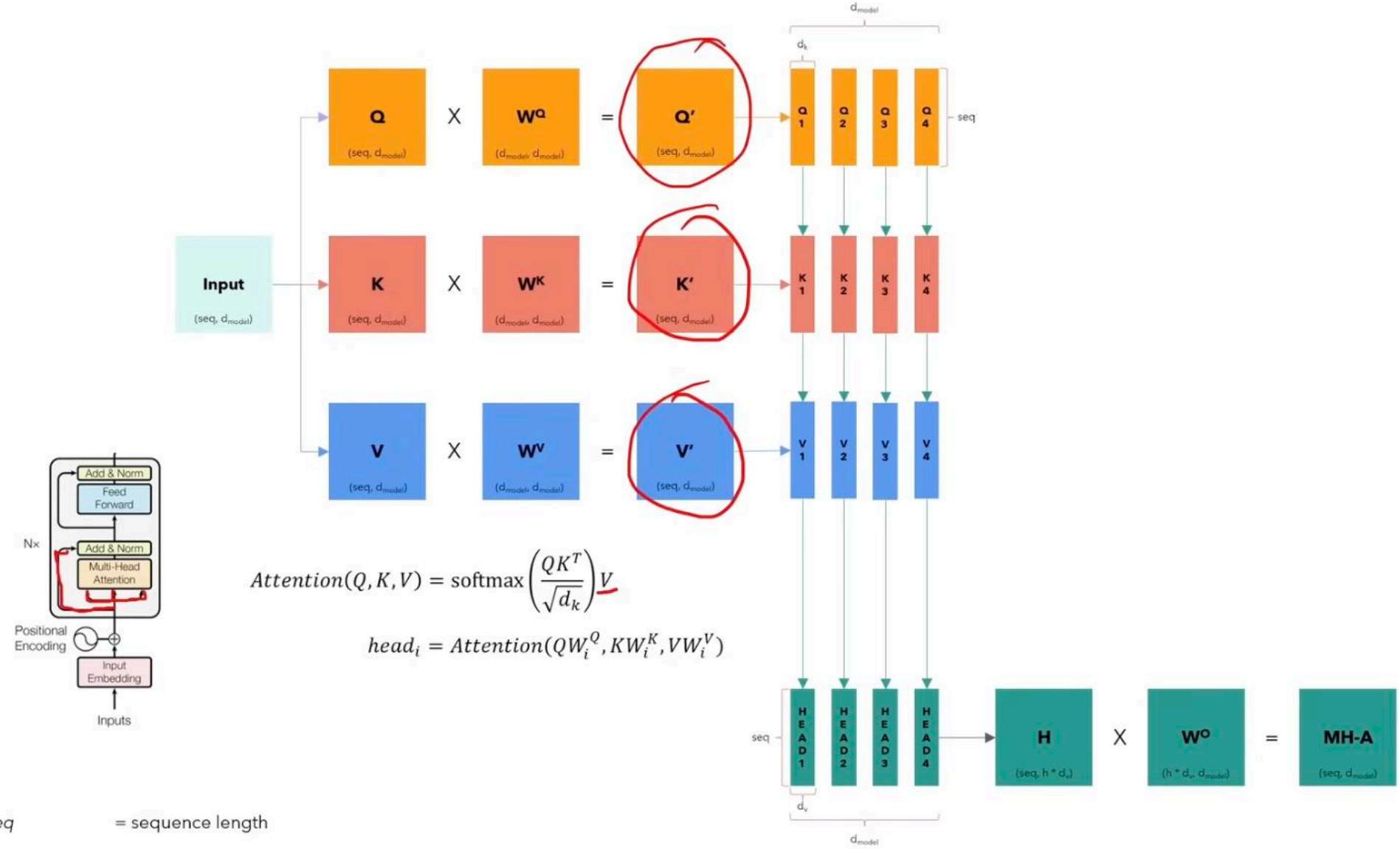
V

(6, 512)



Each row in this matrix captures not only the meaning (given by the embedding) or the position in the sentence (represented by the positional encodings) but also each word's interaction with other words.

Umar Jamil - https://github.com/hkproj/transformer-from-scratch-notes



$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$MultiHead(Q, K, V) = Concat(head_1 \ ... head_h)W^O$$

seq = sequence length

d_{model} = size of the embedding vector

h = number of heads

$d_k = d_v$ = d_{model} / h

Umar Jamil - https://github.com/hkproj/transformer-from-scratch-notes

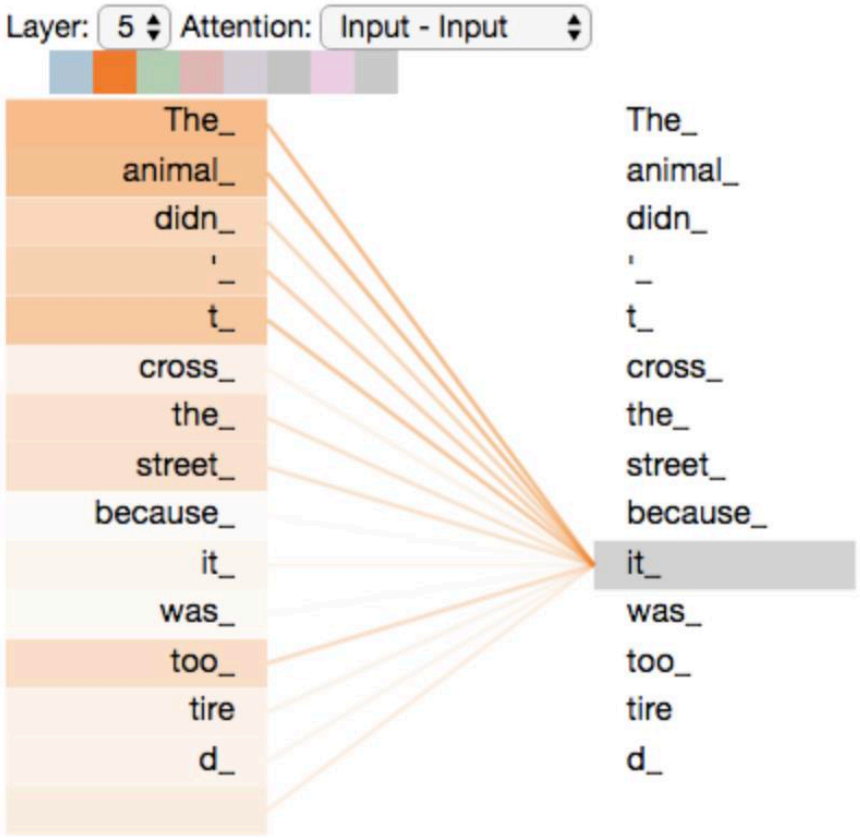
Visualize Attention

“The animal didn't cross the street because it was too tired”

- What does “it” in this sentence refer to?

- the street or to the animal?

- When the model is processing the word “it”, self-attention allows it to associate “it” with “animal”



Causal Masks

- In models like GPT, attention calculation is slightly different
  - GPT is an autoregressive language model based on transformers.
  - They generate text one token at a time, conditioning on the previously generated tokens.

- **Causal masks** are employed in the self-attention layers
  - to ensure that each token attends only to previous tokens during generation.
  - prevents the model from "cheating" by peeking at future tokens during training or generation

A language model is a probabilistic model that assign probabilities to sequence of words.
In practice, a language model allows us to compute the following:

To model the probability distribution above, each word should only depend on words that come **before it** (*left context*).

	[SOS]	Before	my	bed	lies	a	pool	of	moon	bright
[SOS]	**5.45**	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞
Before	4.28	**2.46**	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞
my	8.17	3.56	**5.54**	-∞	-∞	-∞	-∞	-∞	-∞	-∞
bed	6.71	4.13	6.76	**0.79**	-∞	-∞	-∞	-∞	-∞	-∞
lies	5.43	7.59	3.91	6.14	**9.03**	-∞	-∞	-∞	-∞	-∞
a	4.42	4.35	7.55	3.14	1.35	**7.57**	-∞	-∞	-∞	-∞
pool	8.36	6.00	4.56	0.52	3.13	6.78	**9.00**	-∞	-∞	-∞
of	2.21	3.72	4.16	6.30	0.66	6.14	7.46	**6.77**	-∞	-∞
moon	4.08	6.22	5.00	4.20	5.72	5.35	7.46	3.55	**4.70**	-∞
bright	6.43	8.88	6.17	3.65	4.54	5.22	5.51	5.55	0.64	**1.38**

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

	[SOS]	Before	my	bed	lies	a	pool	of	moon	bright
[SOS]	**5.45**	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞
Before	4.28	**2.46**	-∞	-∞	-∞	-∞	-∞	-∞	-∞	-∞
my	8.17	3.56	**5.54**	-∞	-∞	-∞	-∞	-∞	-∞	-∞
bed	6.71	4.13	6.76	**0.79**	-∞	-∞	-∞	-∞	-∞	-∞
lies	5.43	7.59	3.91	6.14	**9.03**	-∞	-∞	-∞	-∞	-∞
a	4.42	4.35	7.55	3.14	1.35	**7.57**	-∞	-∞	-∞	-∞
pool	8.36	6.00	4.56	0.52	3.13	6.78	**9.00**	-∞	-∞	-∞
of	2.21	3.72	4.16	6.30	0.66	6.14	7.46	**6.77**	-∞	-∞
moon	4.08	6.22	5.00	4.20	5.72	5.35	7.46	3.55	**4.70**	-∞
bright	6.43	8.88	6.17	3.65	4.54	5.22	5.51	5.55	0.64	**1.38**

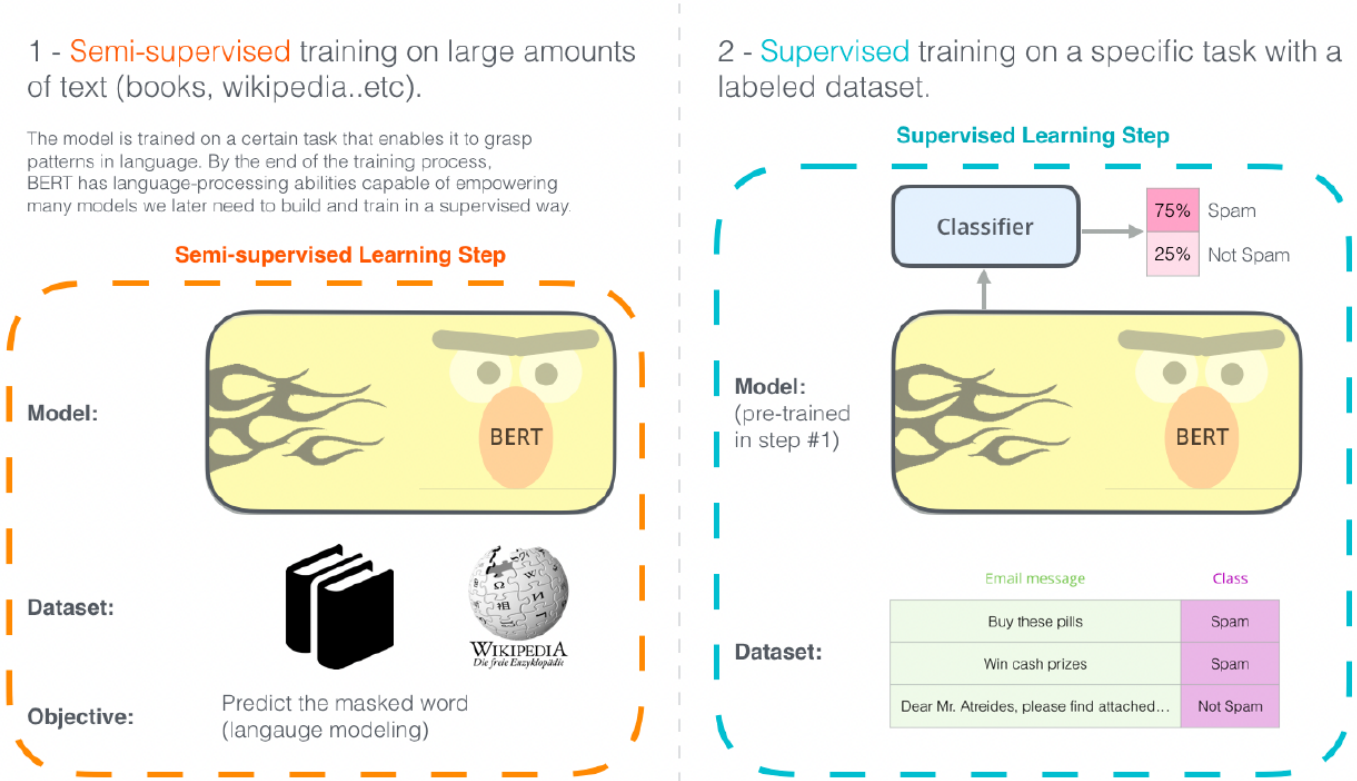
$$\frac{QK^T}{\sqrt{d_k}}$$

	[SOS]	Before	my	bed	lies	a	pool	of	moon	bright
[SOS]	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Before	0.86	**0.14**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
my	0.92	0.01	**0.07**	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bed	0.47	0.04	0.49	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00
lies	0.02	0.18	0.00	0.04	**0.75**	0.00	0.00	0.00	0.00	0.00
a	0.02	0.02	0.47	0.01	0.00	**0.48**	0.00	0.00	0.00	0.00
pool	0.31	0.03	0.01	0.00	0.00	0.06	**0.59**	0.00	0.00	0.00
of	0.00	0.01	0.02	0.15	0.00	0.12	0.47	**0.23**	0.00	0.00
moon	0.02	0.16	0.05	0.02	0.10	0.07	0.55	0.01	**0.03**	0.00
bright	0.07	0.71	0.05	0.03	0.01	0.02	0.03	0.03	0.02	**0.03**

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

BERT - so what?

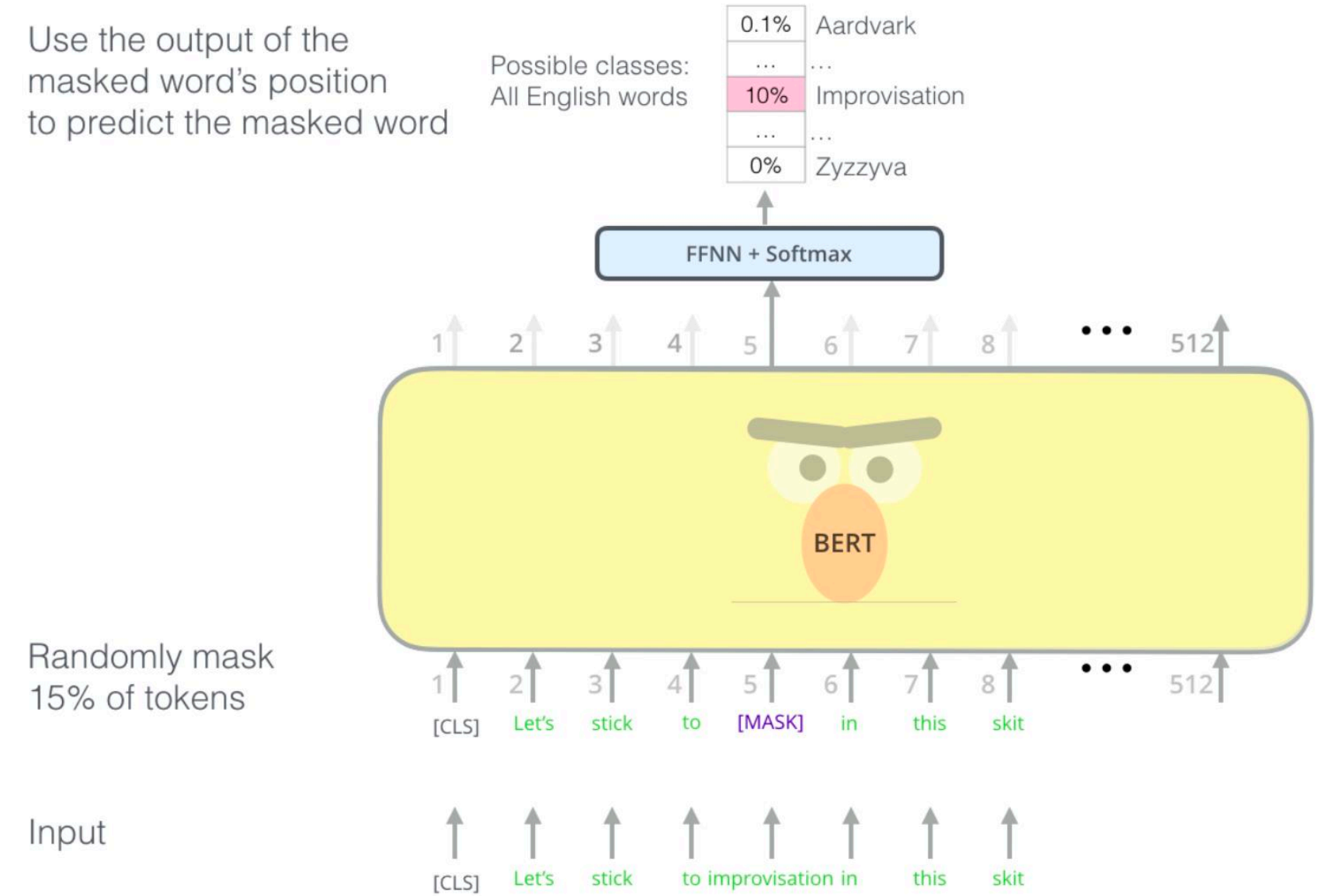
- BERT is a pretrained language model that can be fine-tuned to a specific task.



Source: Alammar (2018)

Pretraining #1: Masked Language Modeling

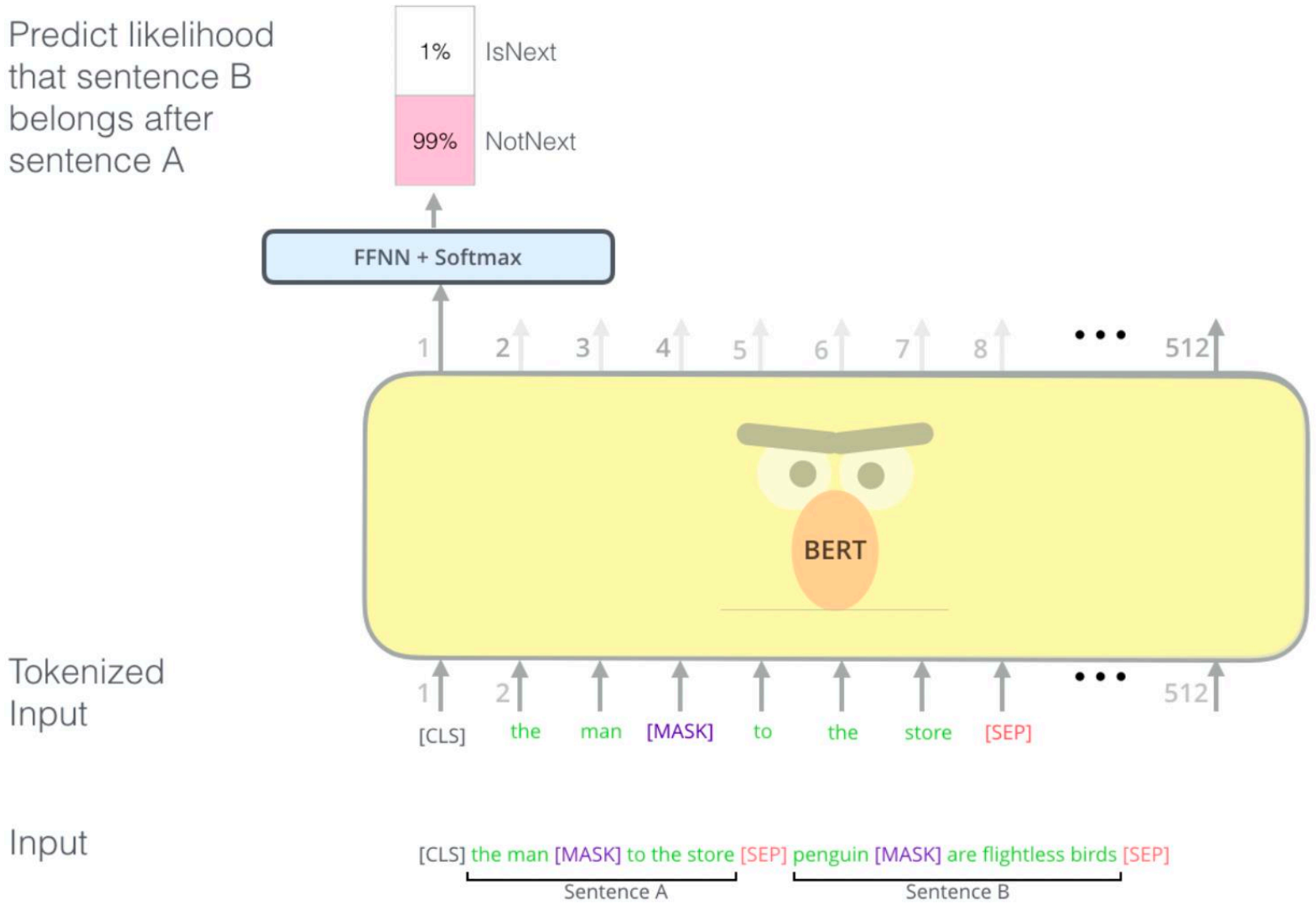
Use the output of the masked word's position to predict the masked word



Randomly mask 15% of tokens

Input

Pretraining #2: Next Sentence Prediction



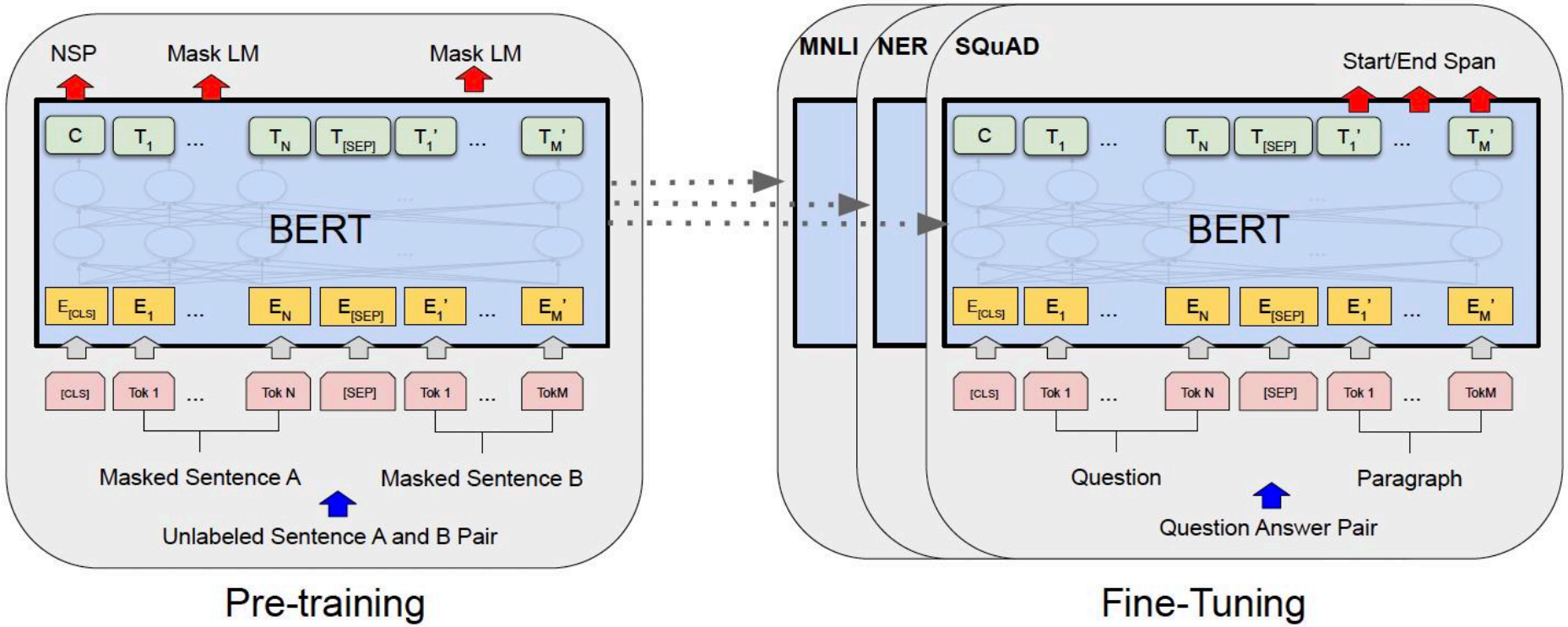
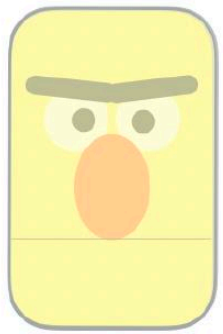
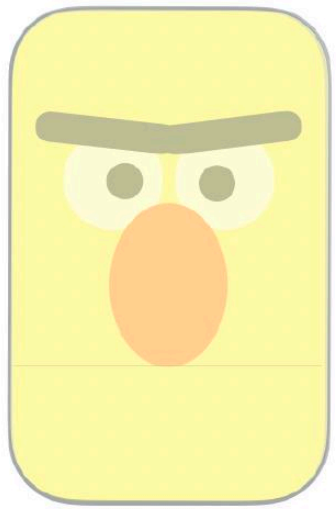


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

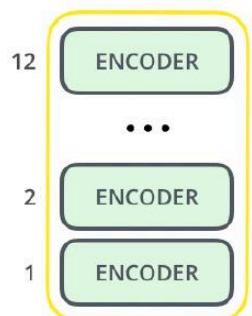
BERT (Google, 2018)



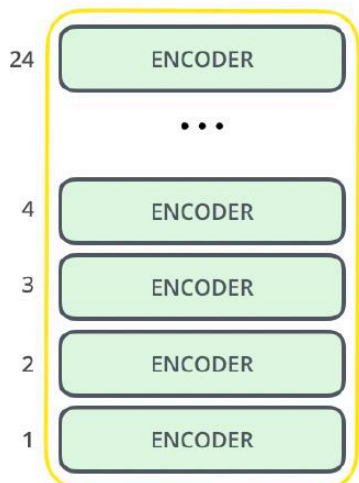
BERT$_{BASE}$



BERT$_{LARGE}$



BERT$_{BASE}$



BERT$_{LARGE}$

With BERT

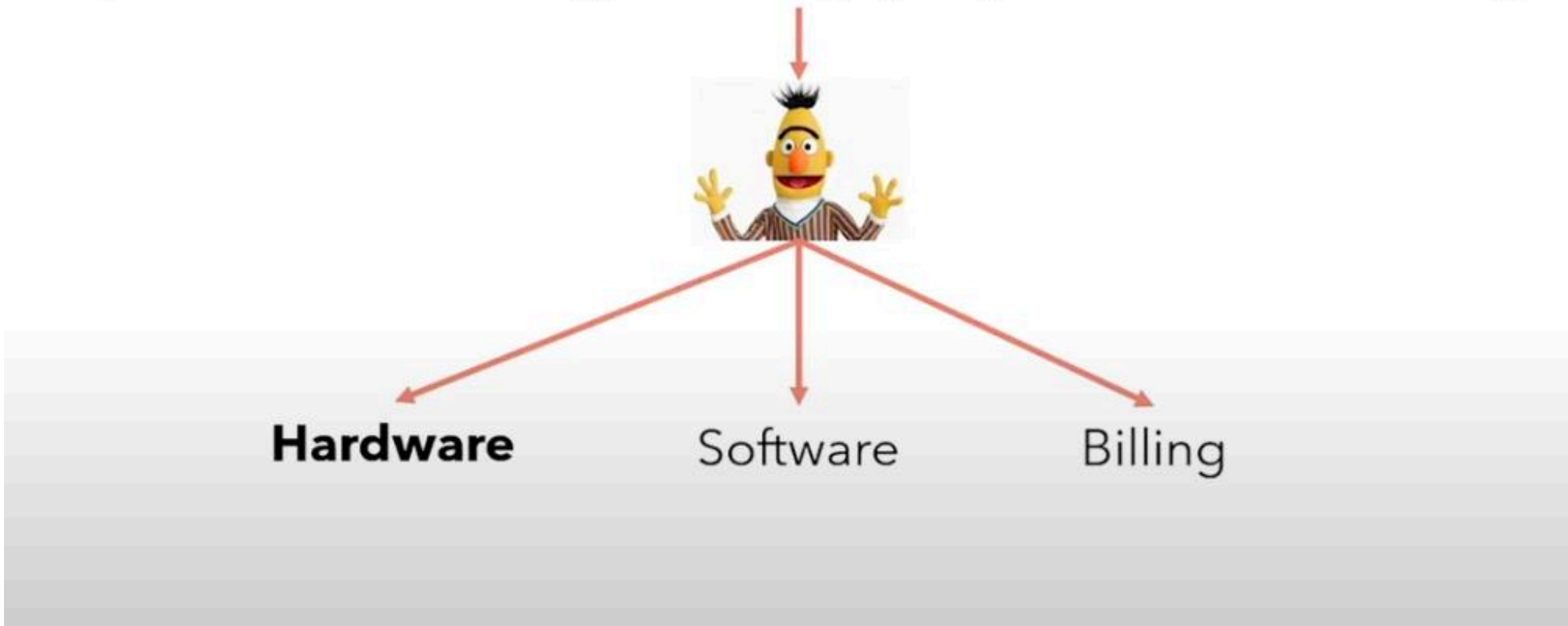
- Work with fine-tuning
  - Text classification, Question Answering

- Does not work with prompts (GPT, Lamma)
  - Zero shot, Few shot, Chain of thought

Text Classification

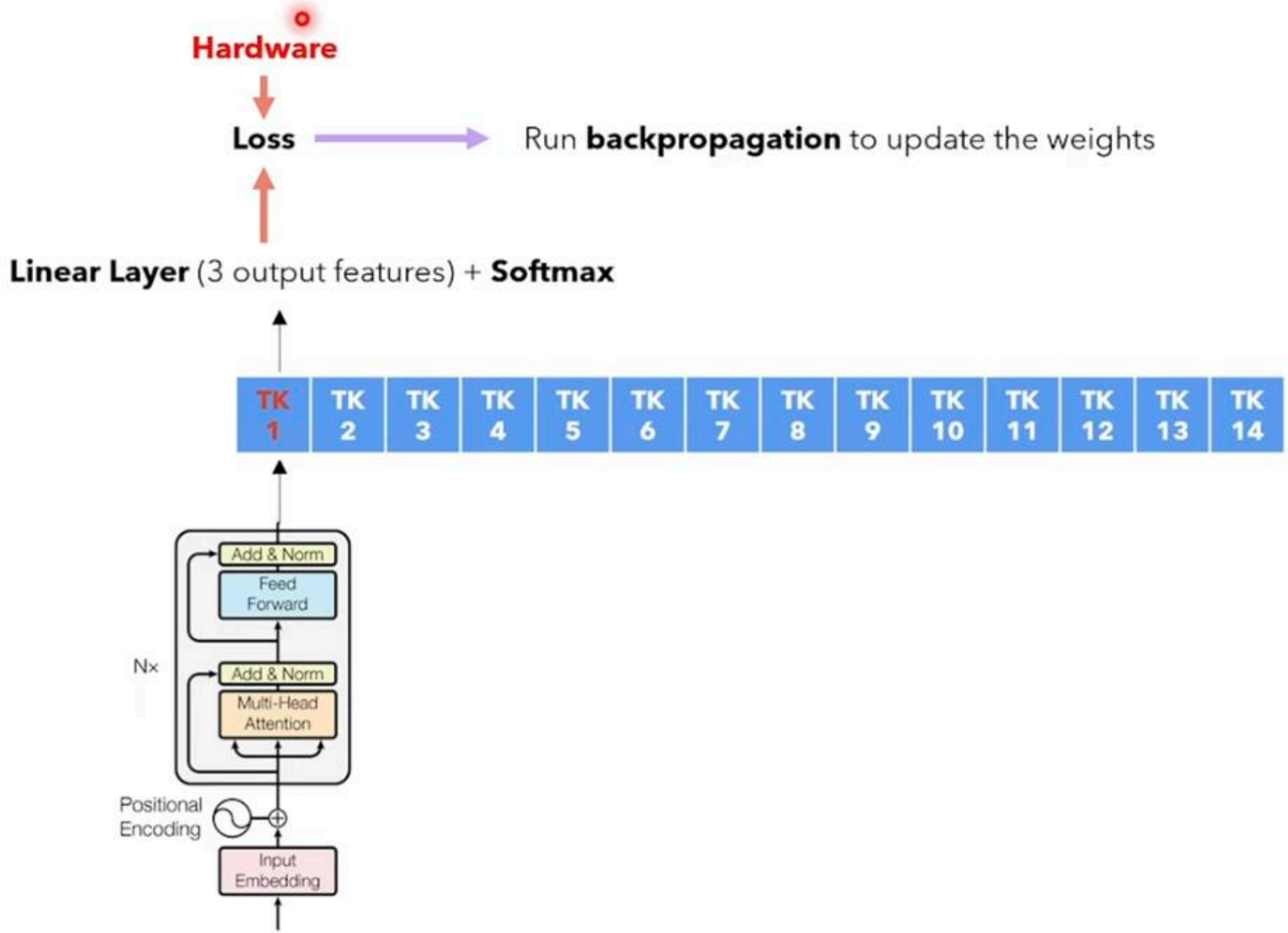
- We want to automatically assign labels to a piece of text

My router's led is not working, I tried changing the power socket but still nothing.



Target (1 token):

I dislike this

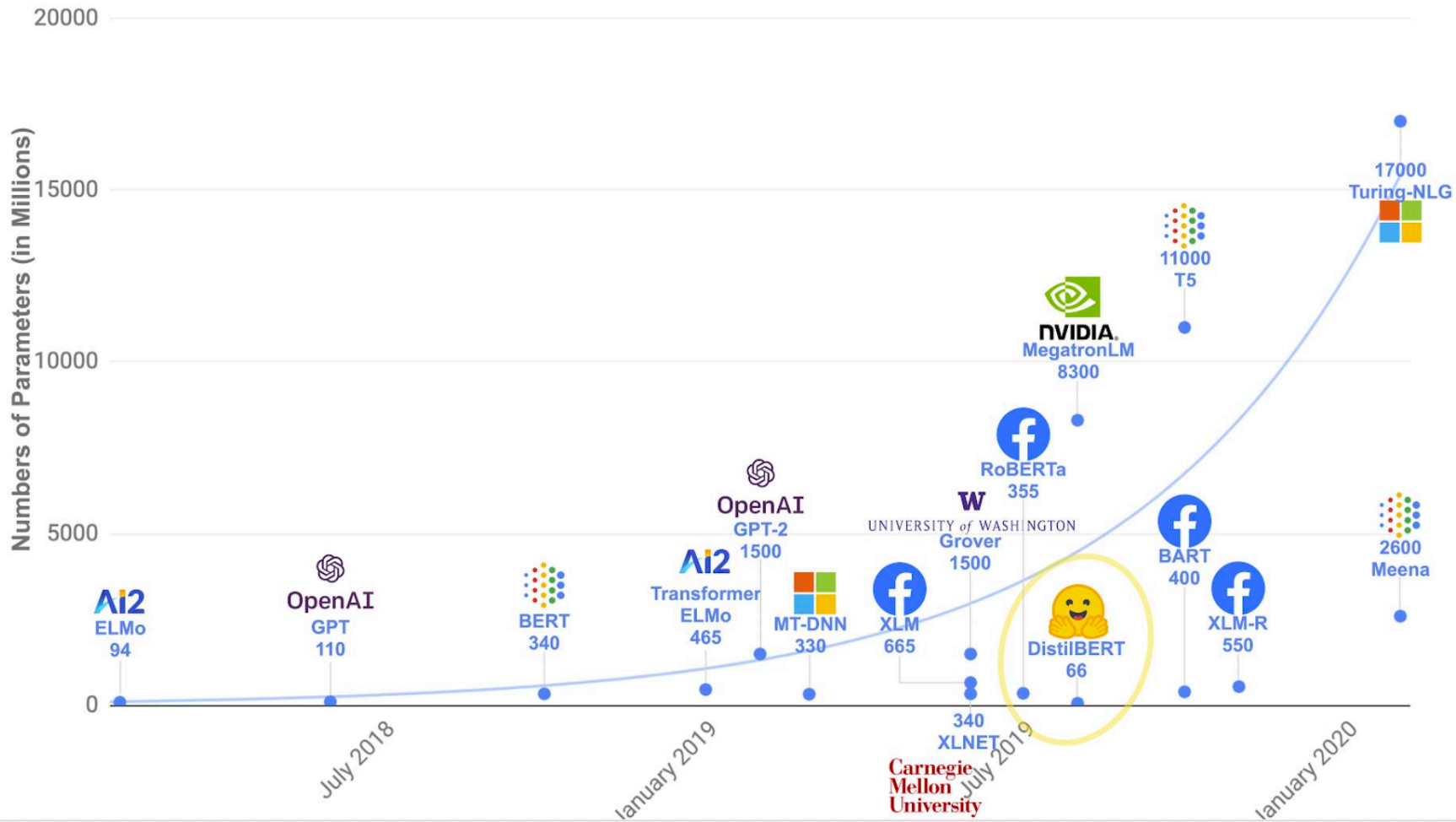


Input (16 tokens):

[CLS] My router's led is not working, I tried changing the power socket but still nothing.

Arms Race

GPT-3 - 175000!



Let's look at the Colab Notebook here:

https://tinyurl.com/triads-llm-workshop

https://tinyurl.com/triads-llm-translation