# How NLP Cracked Transfer Learning Part 2

**Anastasia Ianina**

20.06.2022

# Outline

1. OpenAI Transformer
2. ELMO
3. BERT

} Part 1 (Friday)

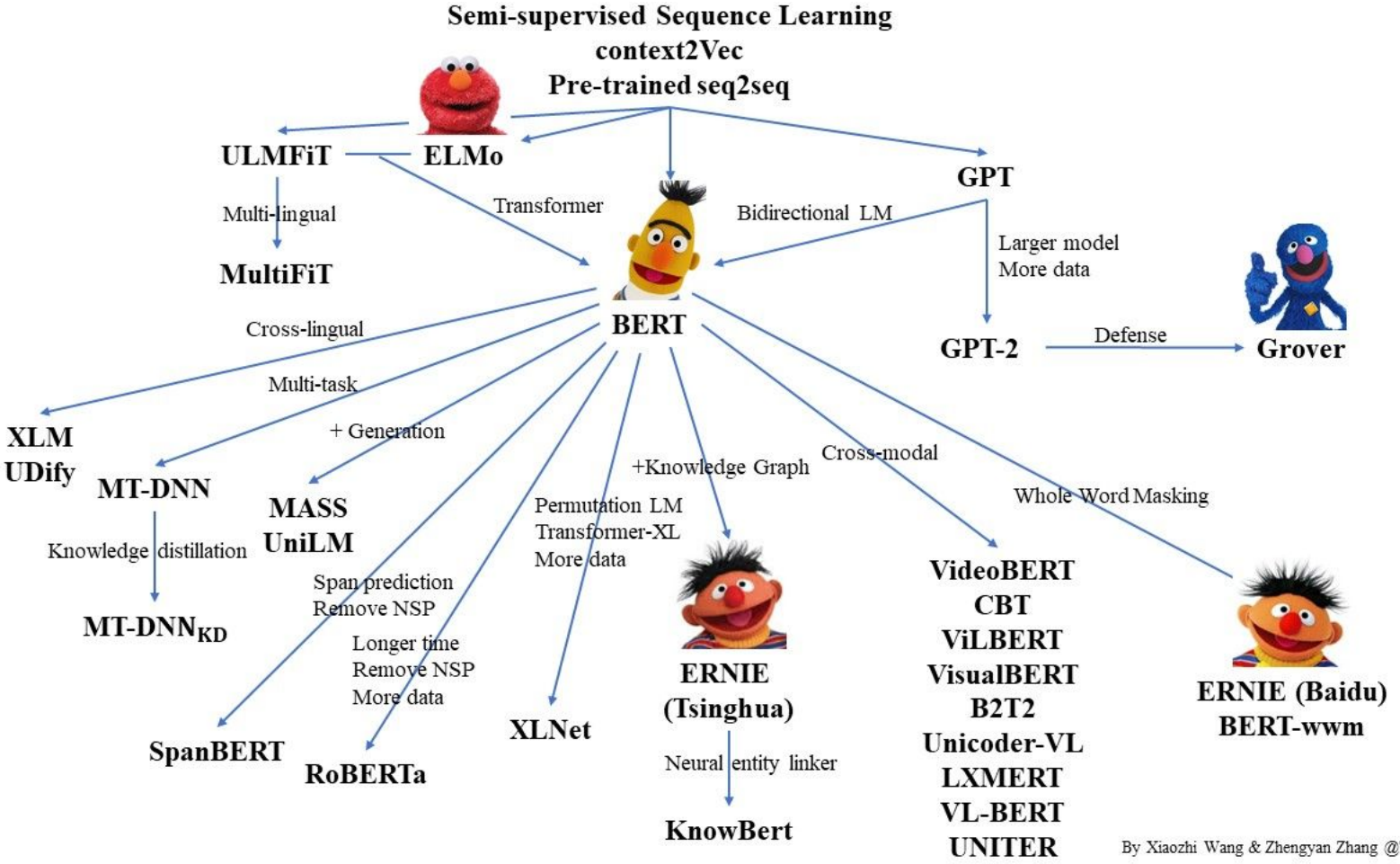
4. GPT, GPT-2, GPT-3
5. Transformer XL
6. XLNET

} Part 2 (today)

Based on: http://jalammar.github.io/illustrated-gpt2/
https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html
https://mlexplained.com/2019/06/30/paper-dissected-xlnet-generalized-autoregressive-pretraining-for-language-understanding-explained/
https://towardsdatascience.com/xlnet-explained-in-simple-terms-255b9fb2c97c
https://towardsdatascience.com/illustrating-the-reformer-393575ac6ba0

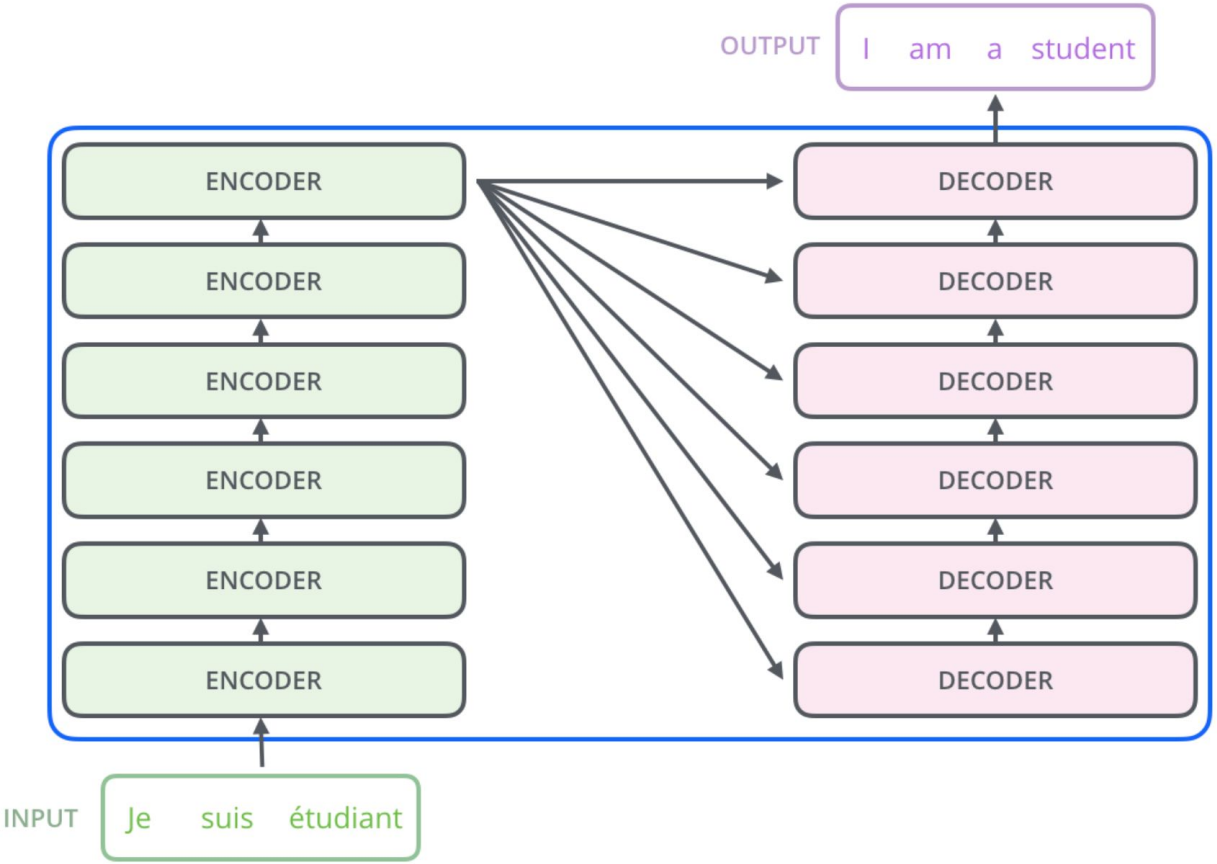
# BERTology paper

# BERTology

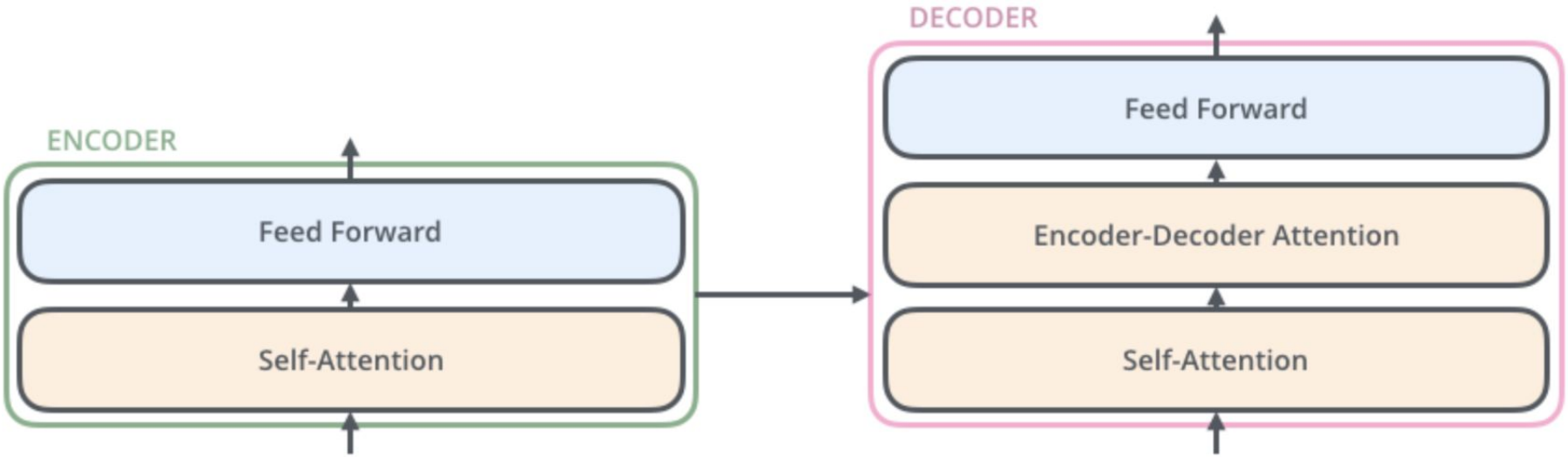


# The Transformer: recap

# The Transformer

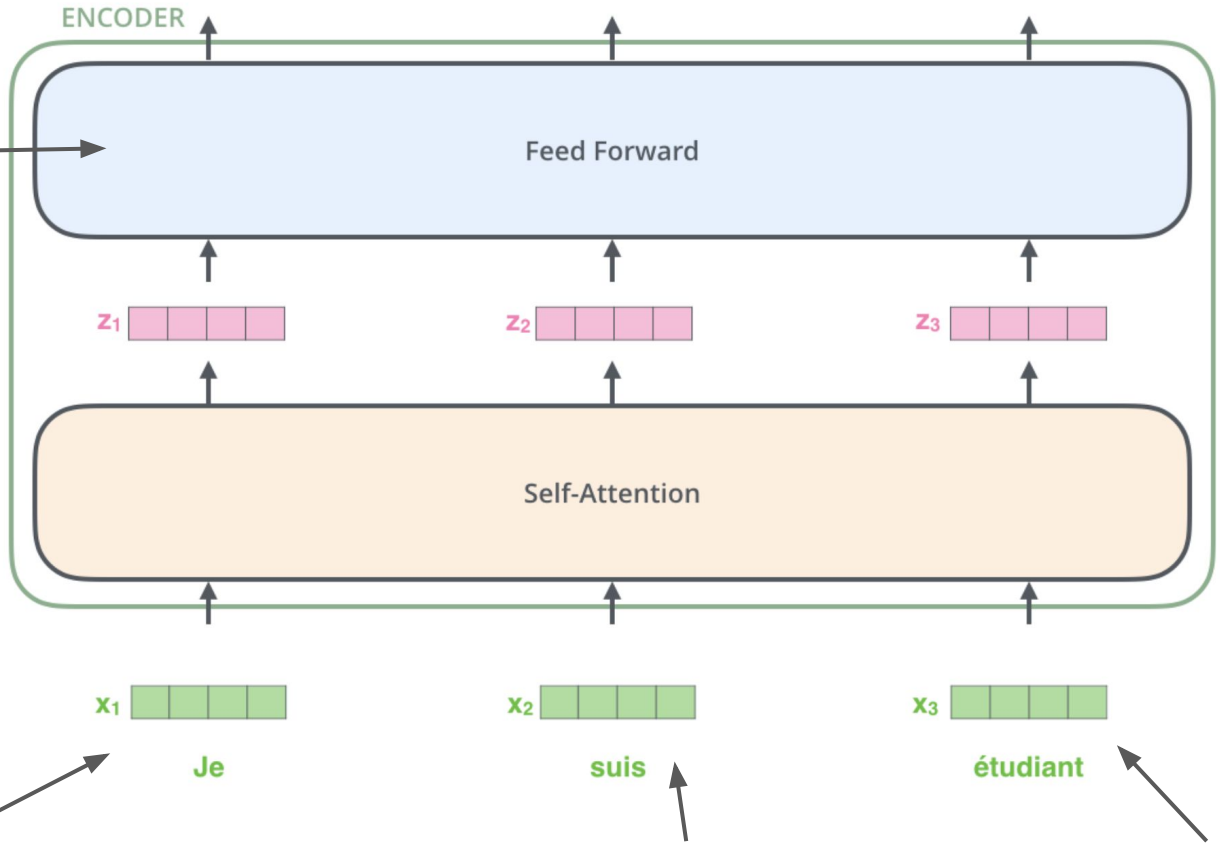


# The Transformer



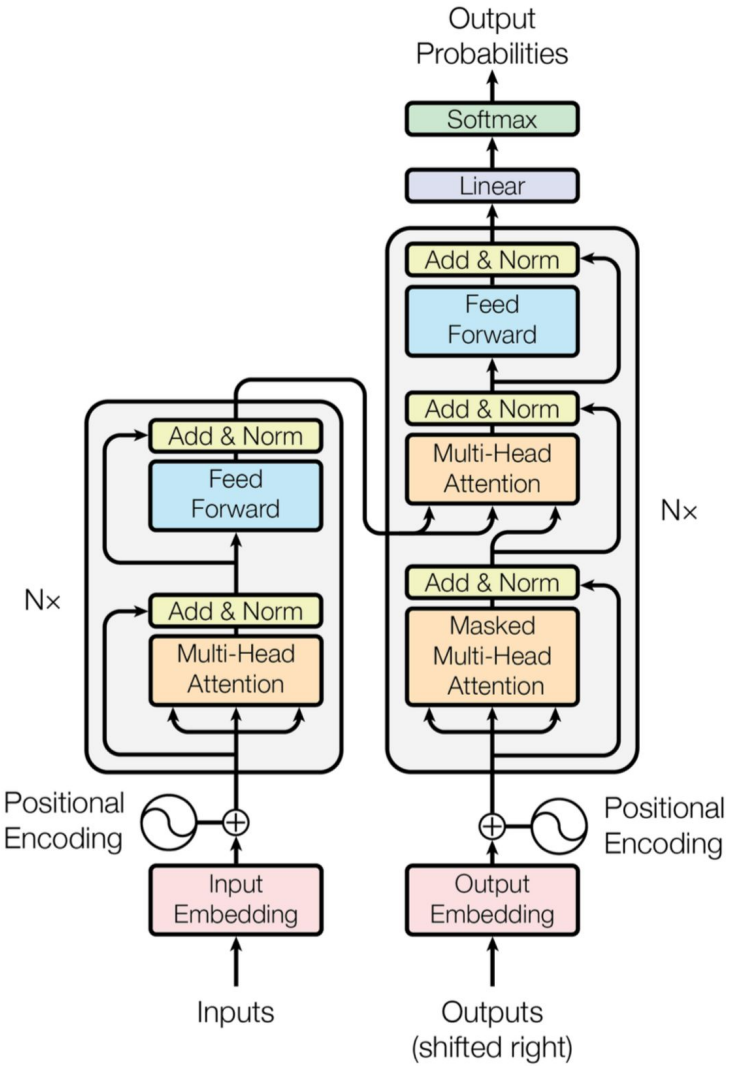
# The Transformer

Can be parallelized




the word in each position flows through its own path in the encoder

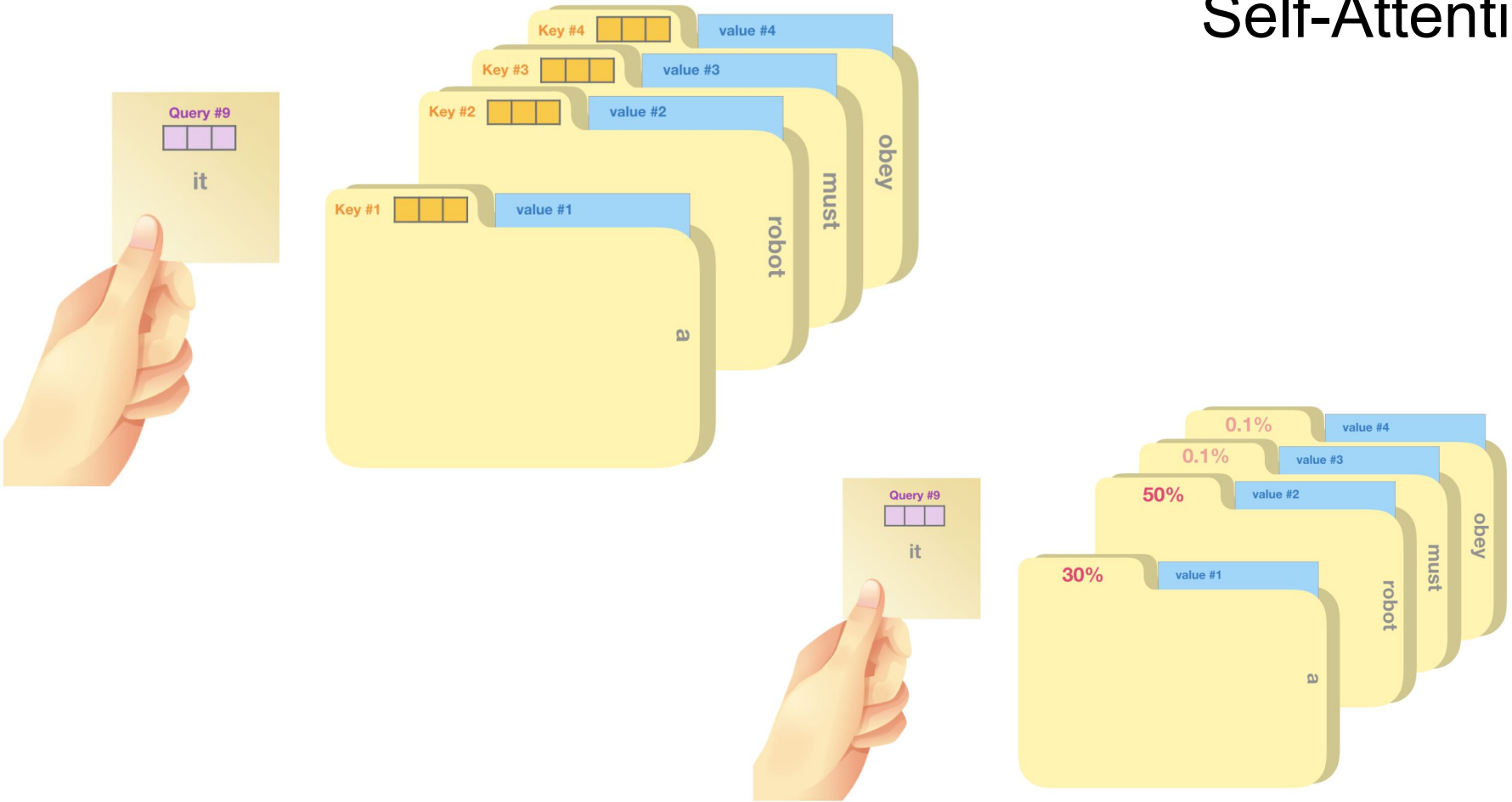
# The Transformer: recap



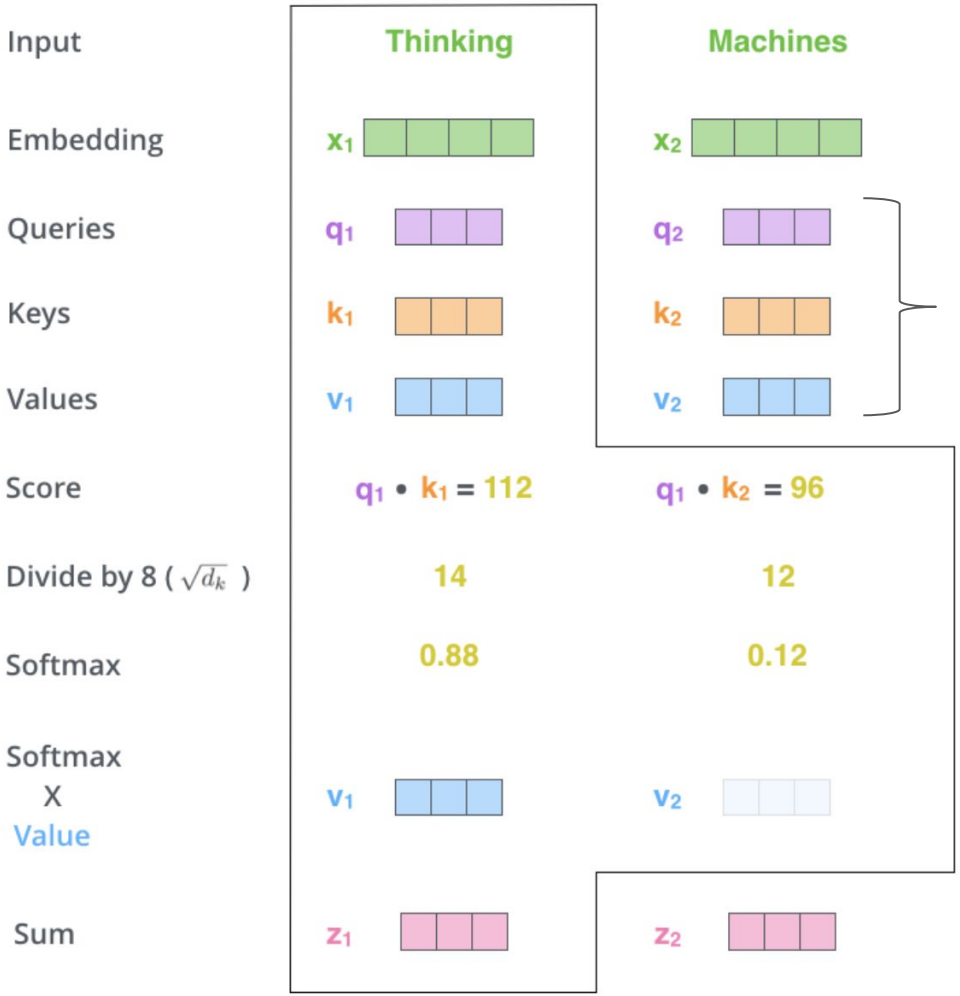

- Proposed in the paper “Attention is All You Need” (Ashish Vaswani et al.)
- No recurrent or convolutional neural networks -> just attention
- Uses Multi-Head **self-attention** concept

# Self-Attention: recap

# Self-Attention



# Self-Attention in Detail




**STEP 1:** create Query, Key, Value

**STEP 2:** calculate scores

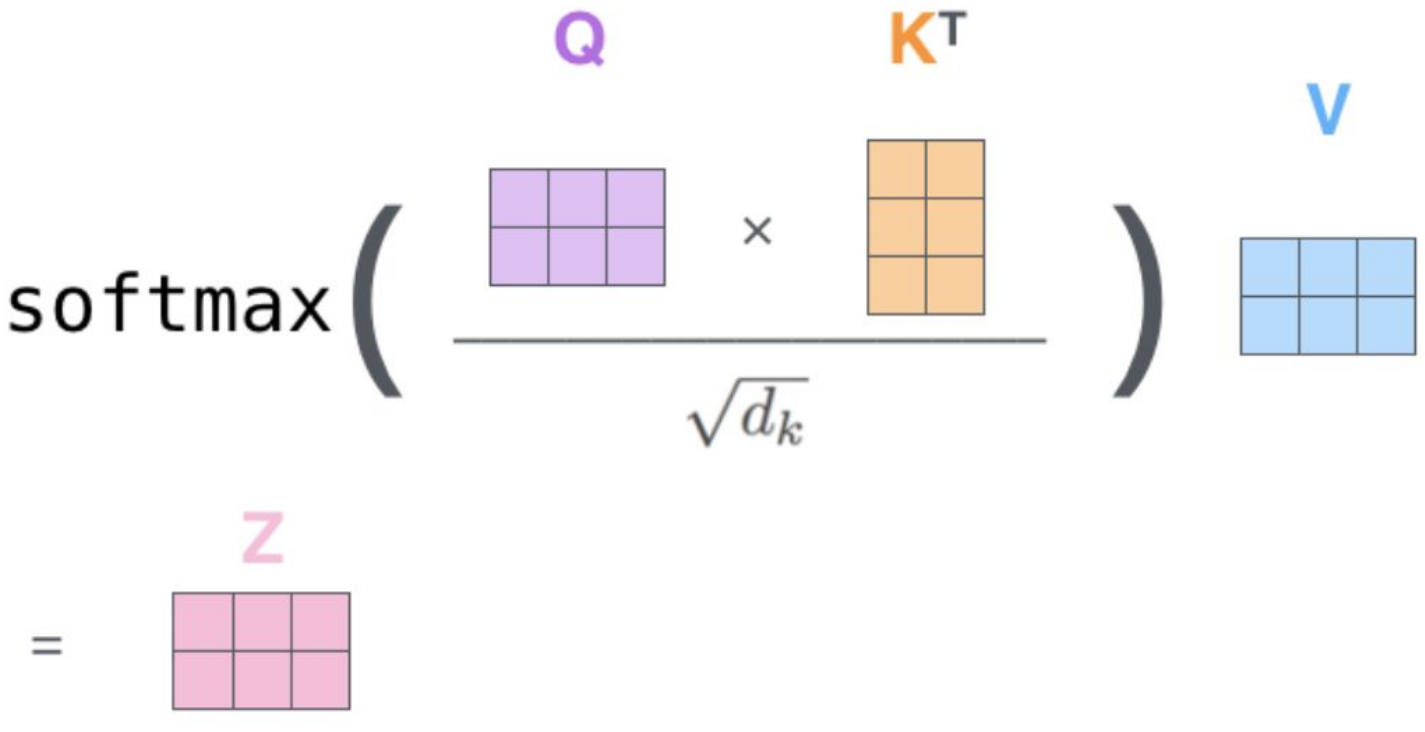
**STEP 3:** divide by $\sqrt{d_k}$

**STEP 4:** softmax

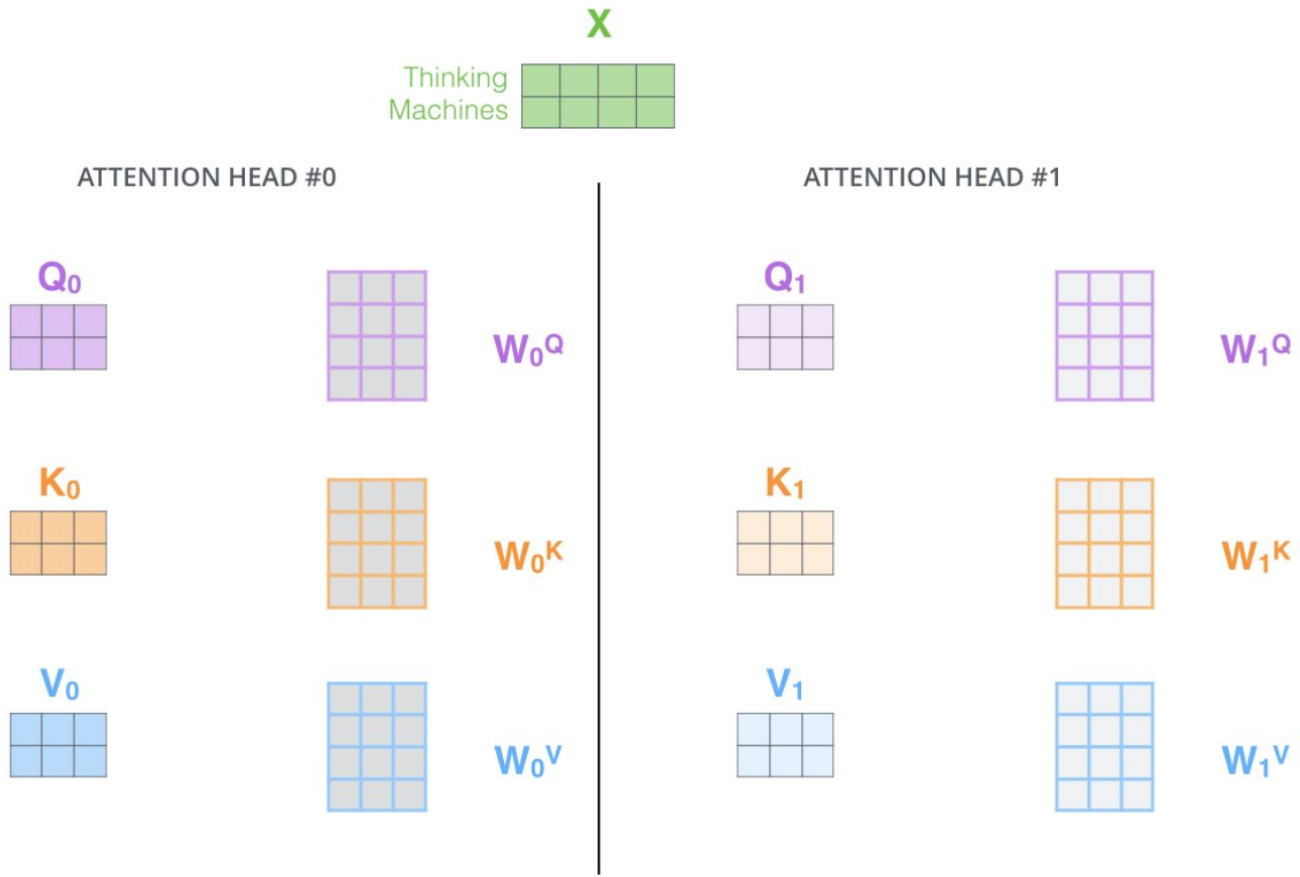
**STEP 5:** multiply each value vector by the softmax score

**STEP 6:** sum up the weighted value vectors

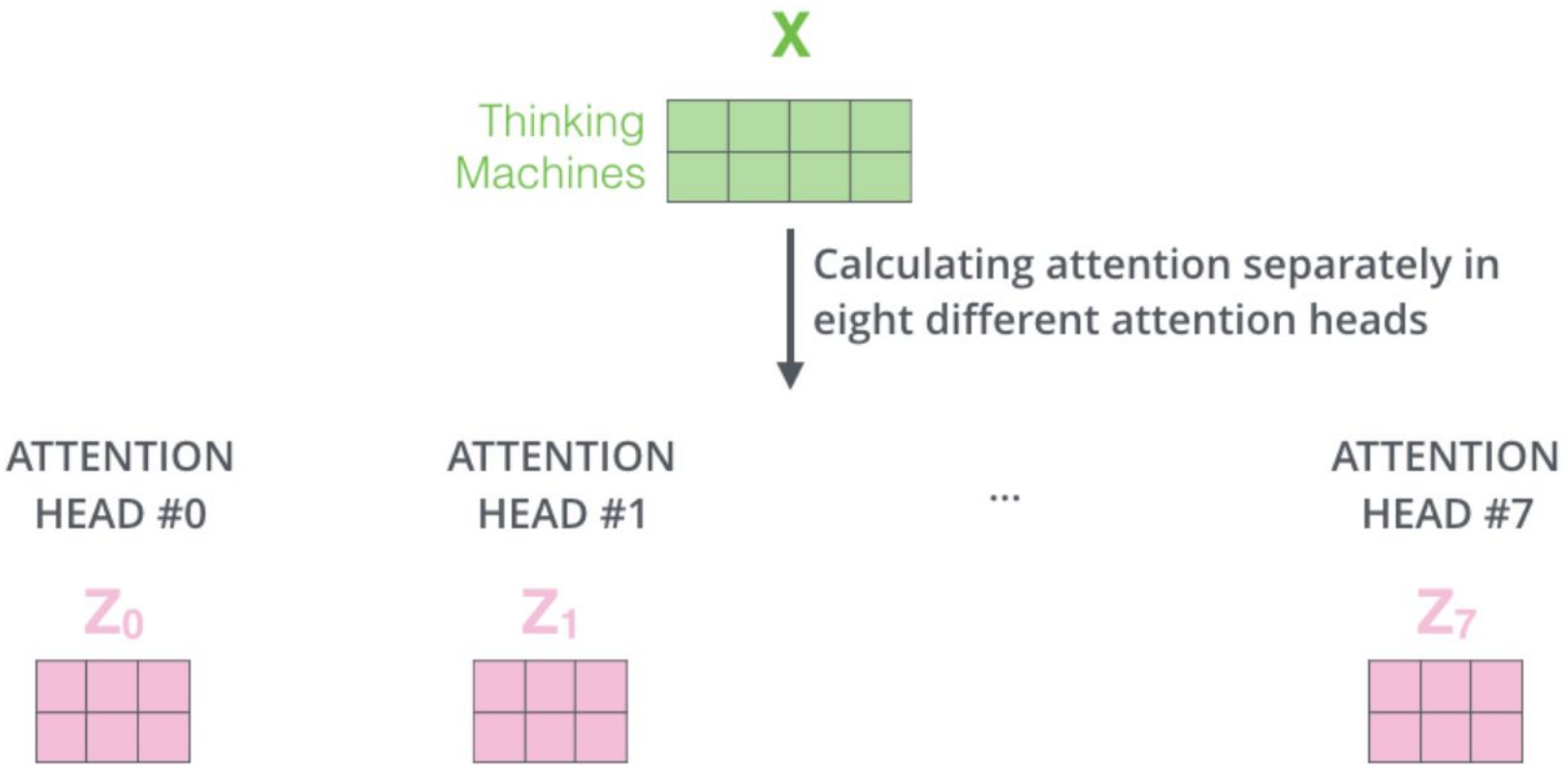
# Self-Attention: Matrix Calculation



# Multi-Head Attention



# Multi-Head Attention

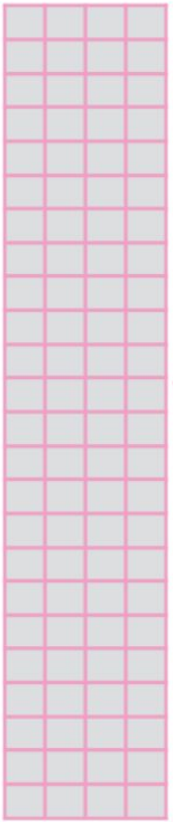


# Multi-Head Attention

1) Concatenate all the attention heads

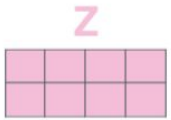

2) Multiply with a weight matrix $W^O$ that was trained jointly with the model

X




3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=



1) This is our input sentence*

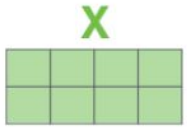
2) We embed each word*

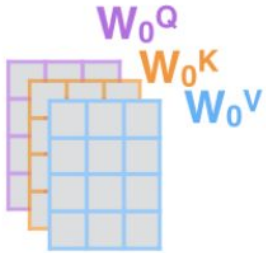
3) Split into 8 heads. We multiply X or R with weight matrices

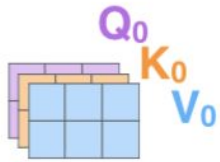
4) Calculate attention using the resulting Q/K/V matrices

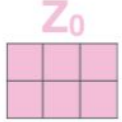
5) Concatenate the resulting Z matrices, then multiply with weight matrix $W^O$ to produce the output of the layer

Thinking Machines

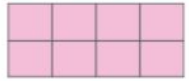


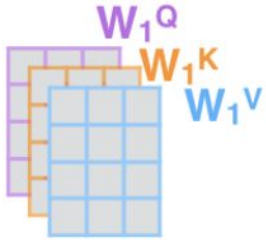



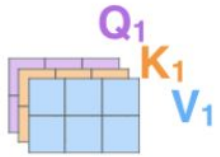



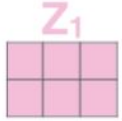




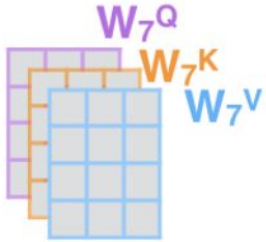

* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

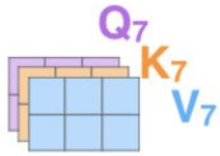


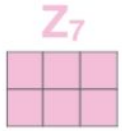







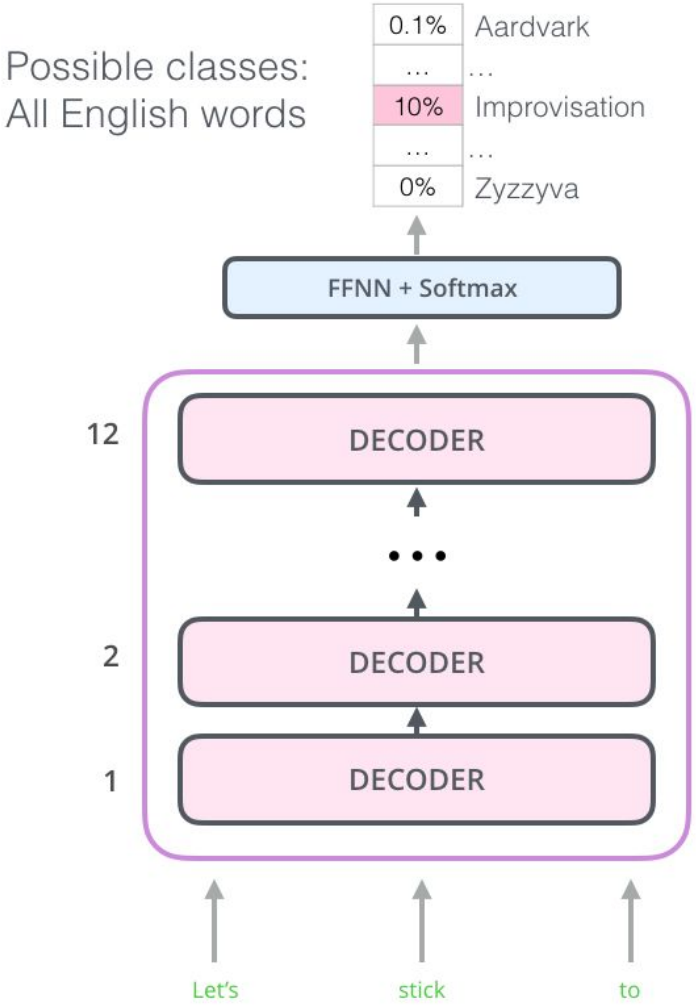



# **OpenAI Transformer:** Pre-training Decoder for Language Modeling

# OpenAI Transformer

- The Encoder-Decoder structure of the transformer made it perfect for machine translation
- But what about sentence classification?
- **Main goal:  pre-train a language model that can be fine-tuned for other tasks**


# OpenAI Transformer

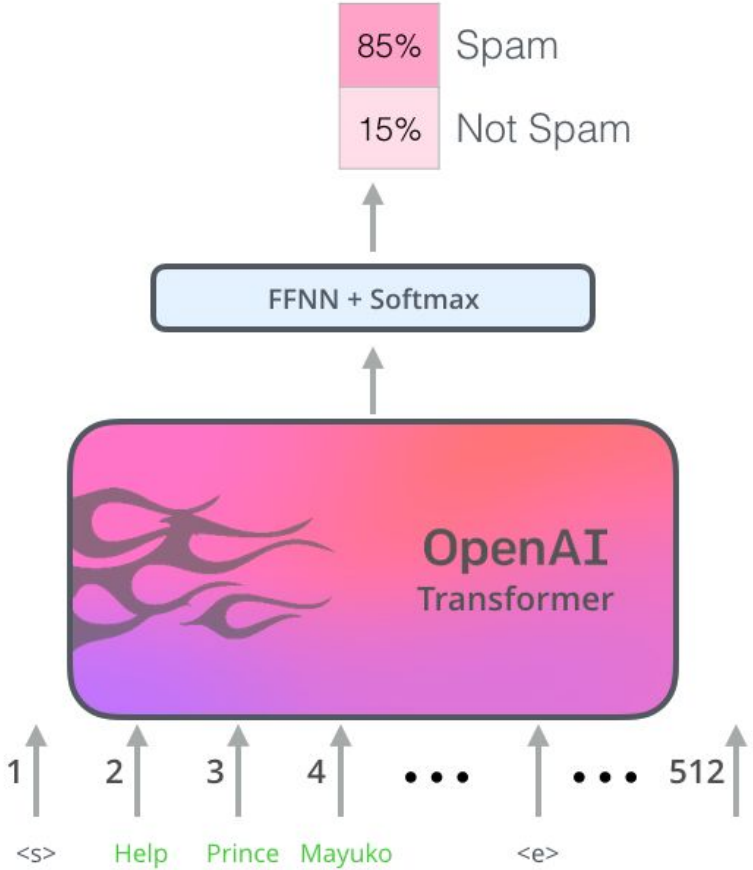



Differences from vanilla Transformer:

- no encoder
- decoder layers would not have the encoder-decoder attention sublayer
- Pre-train the model on predicting the next word using massive (unlabeled) datasets

**Pre-training phase**

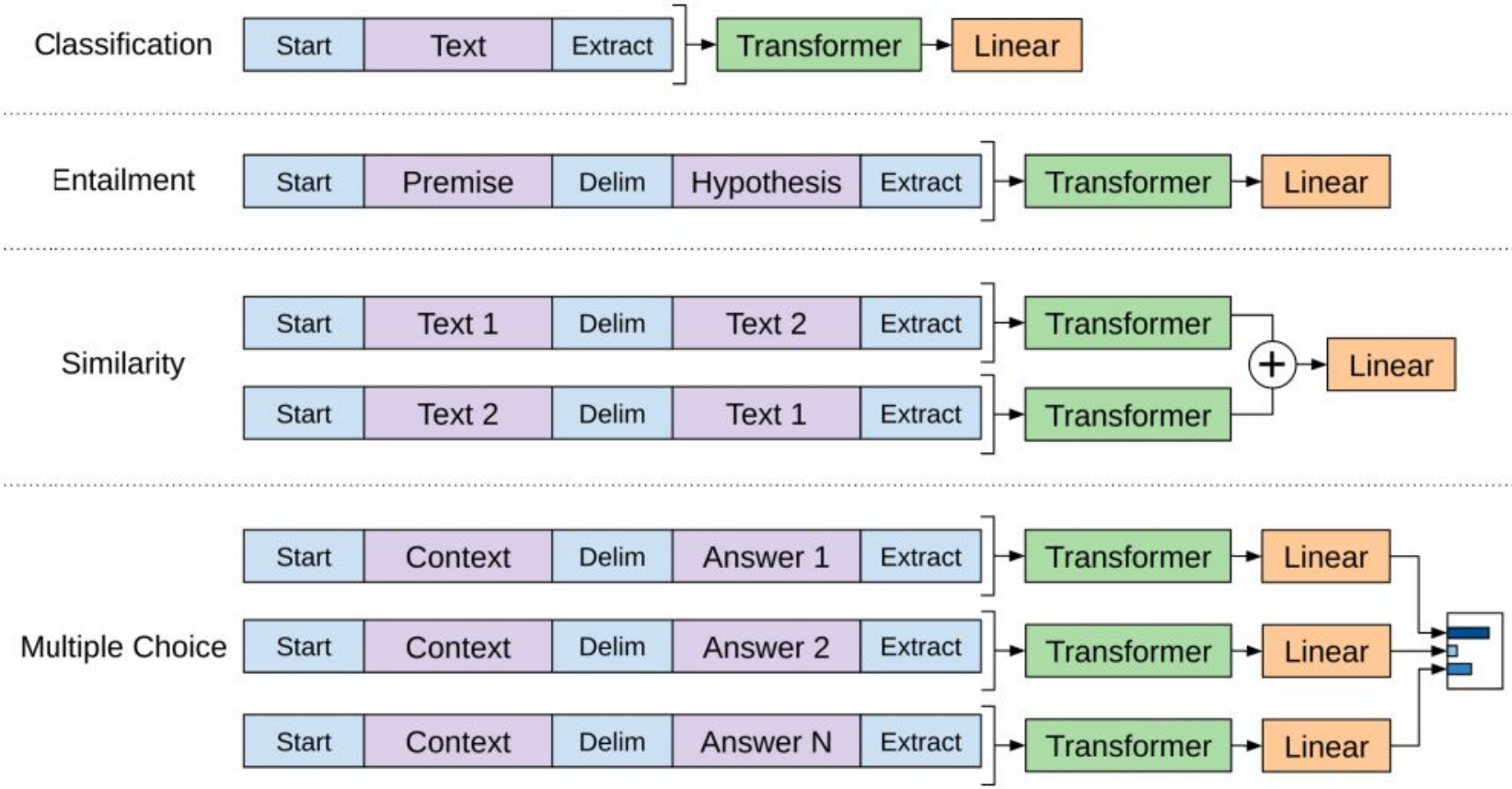
# OpenAI Transformer




**Fine-tuning phase**

- During pre-training phase layers have been tuned to reasonably handle language
- Now let’s use it for downstream tasks (e.g. sentence classification)

# Input transformations for different tasks



# ELMo: context that matters

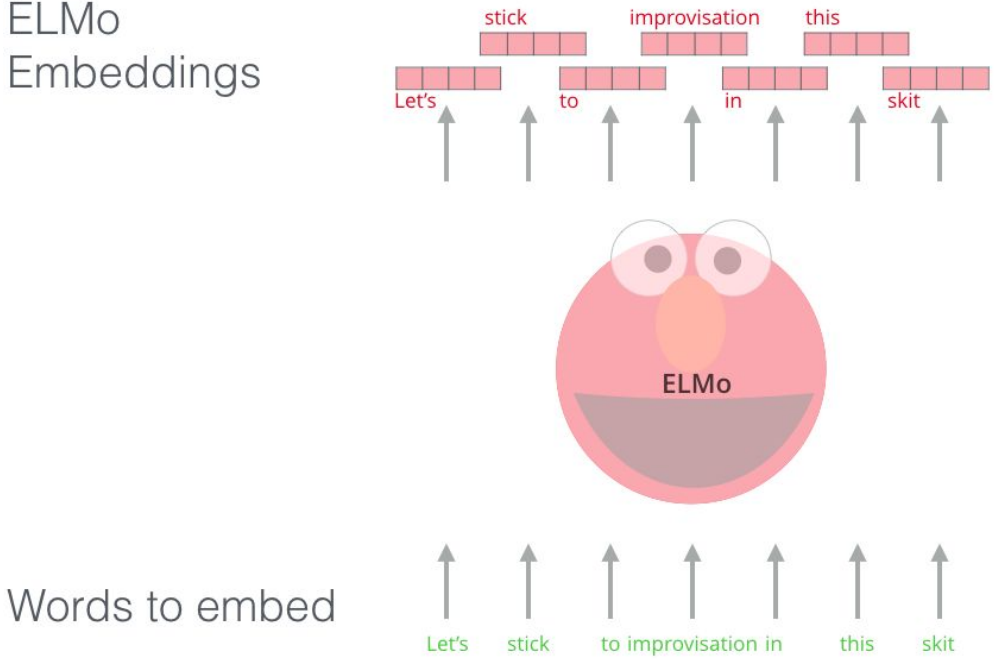
# ELMo: contextualized word embeddings

*"Why not give it an embedding based on the context it's used in – to both capture the word meaning in that context as well as other contextual information?"*

Peters et. al., 2017, McCann et. al., 2017,
and yet again Peters et. al., 2018 in the ELMo paper

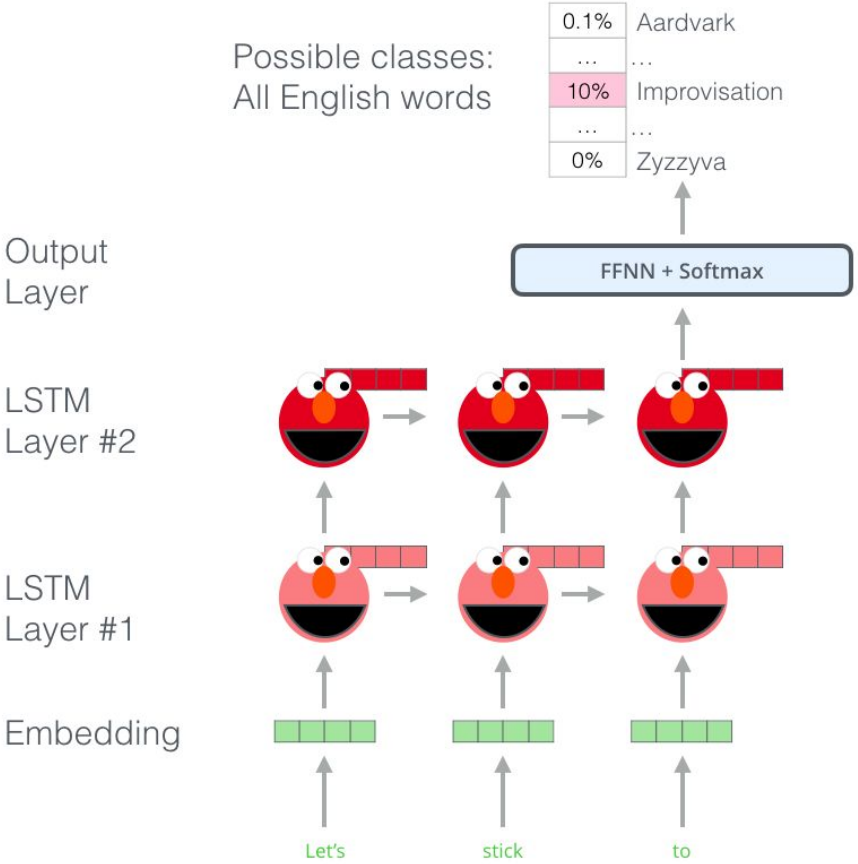
## ELMo - deep contextualized word representations

# ELMo: Contextualized word embeddings




- uses a bi-directional LSTM trained on Language Modeling task
- a model can learn without labels

# Bidirectional Language Models (biLMs)




biLMs consist of forward and backward LMs:

- forward:

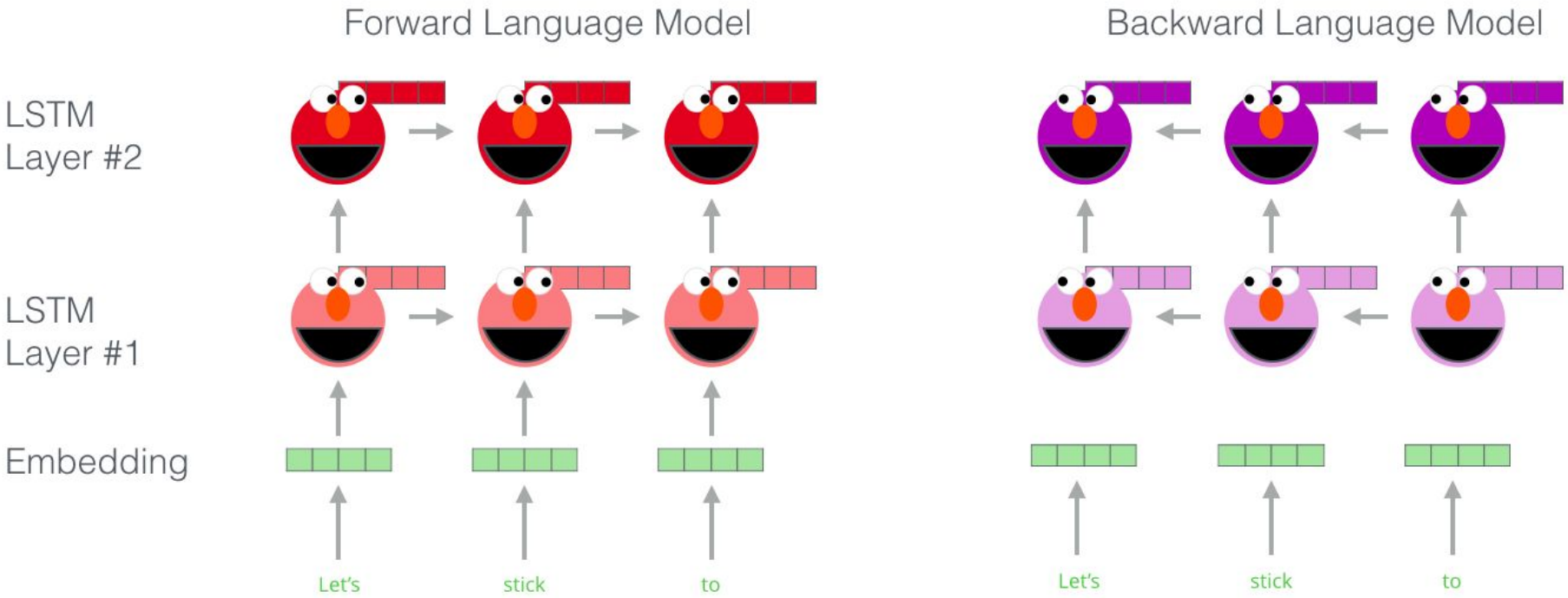
$$p(t_1,t_2,\ldots,t_N)=\prod_{k=1}^{N} p(t_k|t_1,t_2,\ldots,t_{k-1})$$

- Backward:

$$p(t_1,t_2,\ldots,t_N)=\prod_{k=1}^{N} p(t_k|t_{k+1},t_{k+2},\ldots,t_N)$$

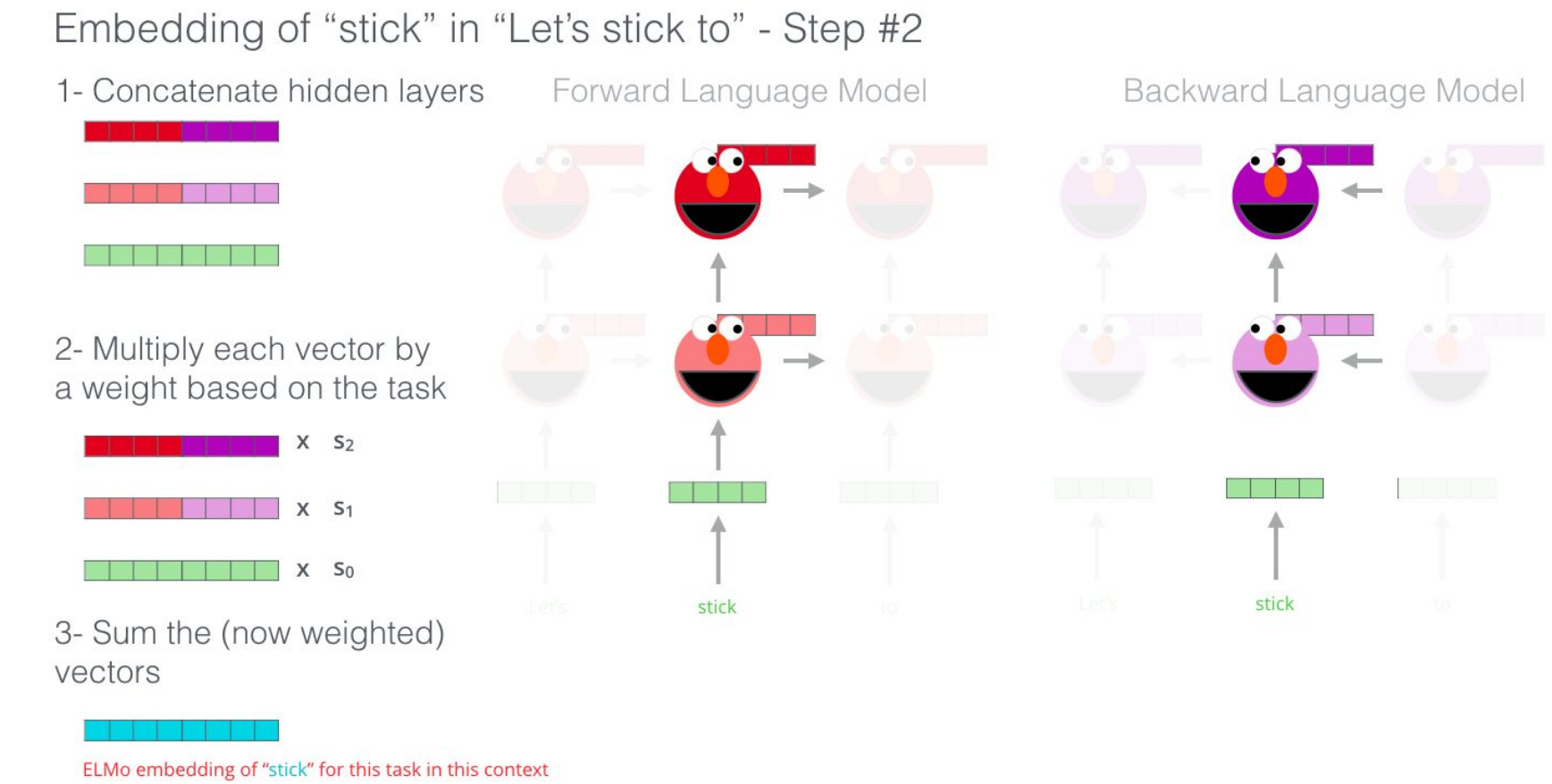
LSTM predicts next word in both directions to build biLMs

# ELMo: main pipeline



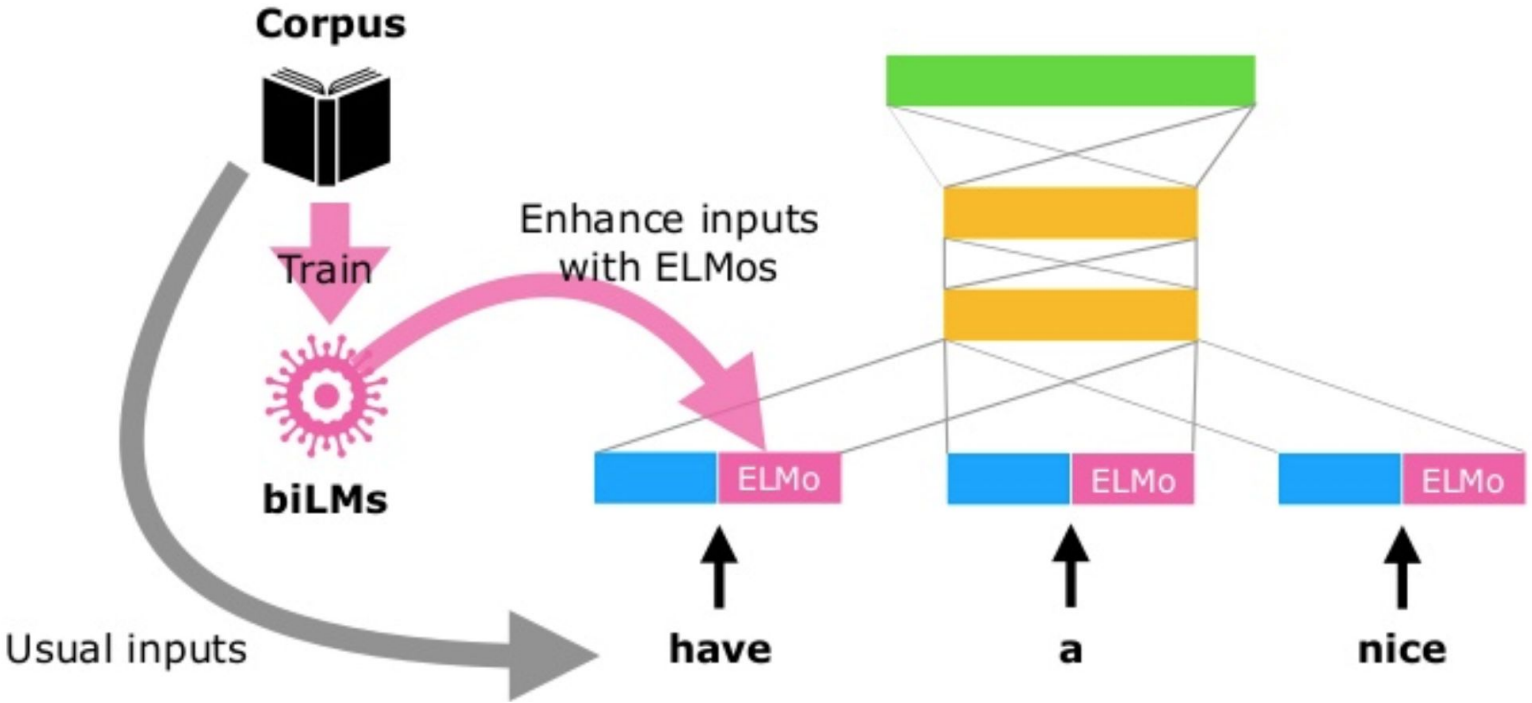
# ELMo: main pipeline

ELMo represents a word as a linear combination of corresponding hidden layers:



# ELMo

ELMo can be integrated to almost all neural NLP tasks with simple concatenation to the embedding layer

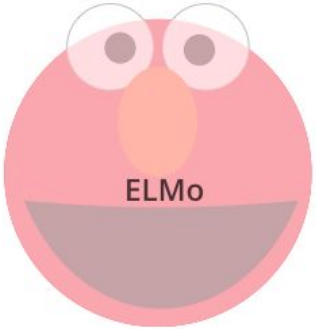



THE
TRANSFORMER


ULM-FiT


BERT

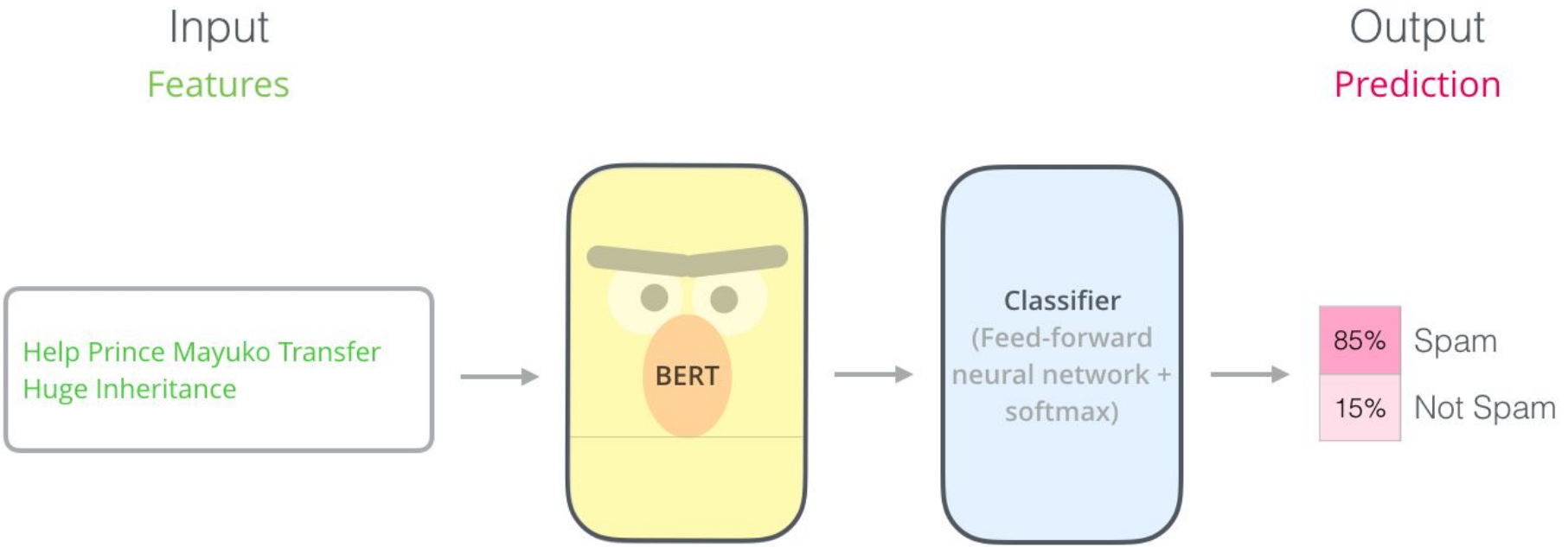

OpenAI
Transformer


ELMo

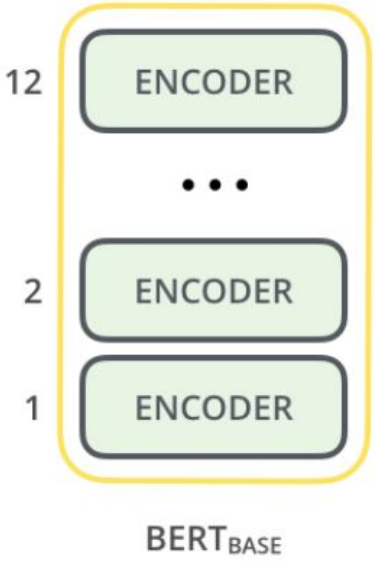
# BERT

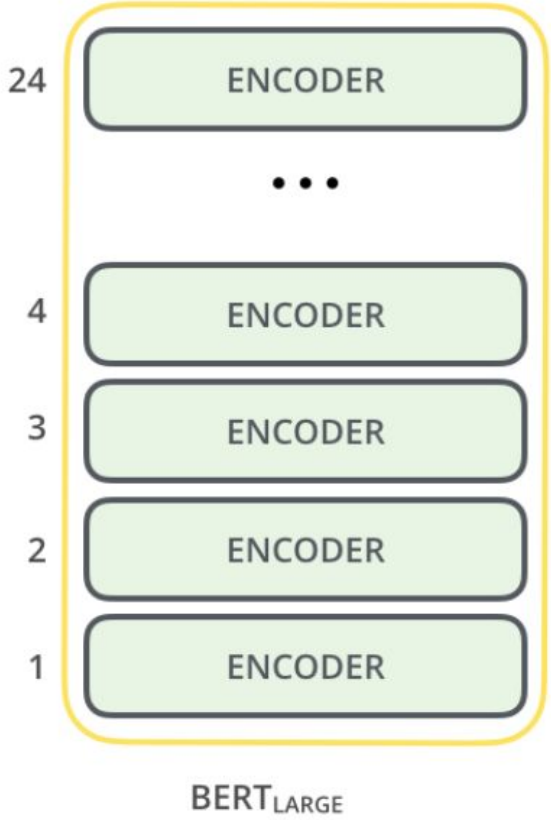
Bidirectional Encoder Representations from Transformers

# BERT


Input
Features
Output
Prediction
Help Prince Mayuko Transfer
Huge Inheritance
BERT
Classifier
(Feed-forward
neural network +
softmax)
85%
Spam
15%
Not Spam

# BERT: base and large

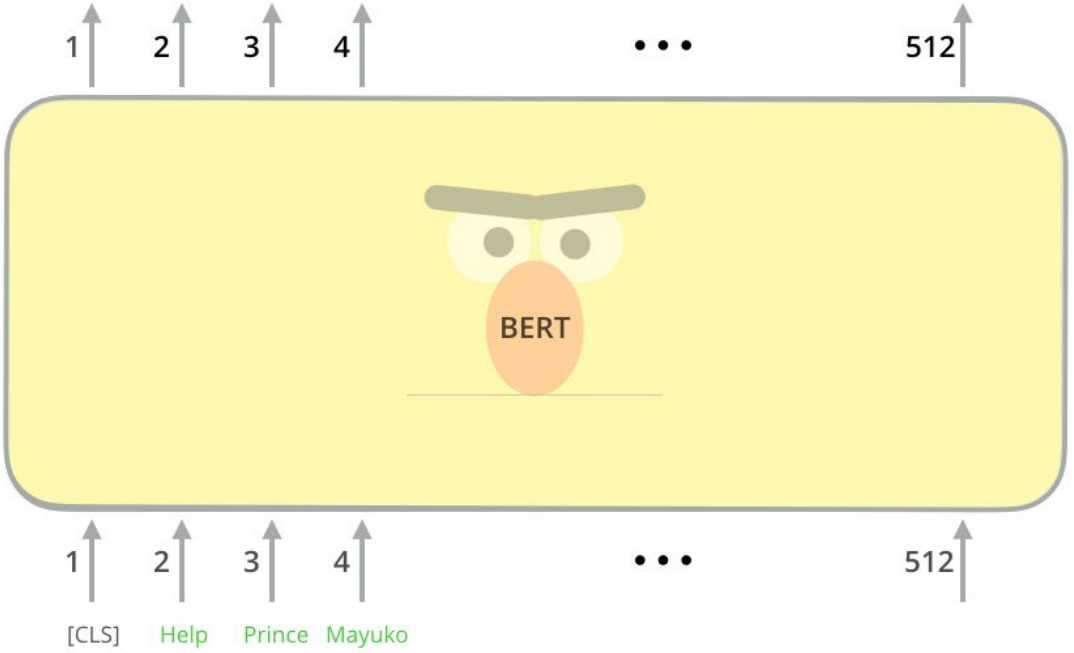


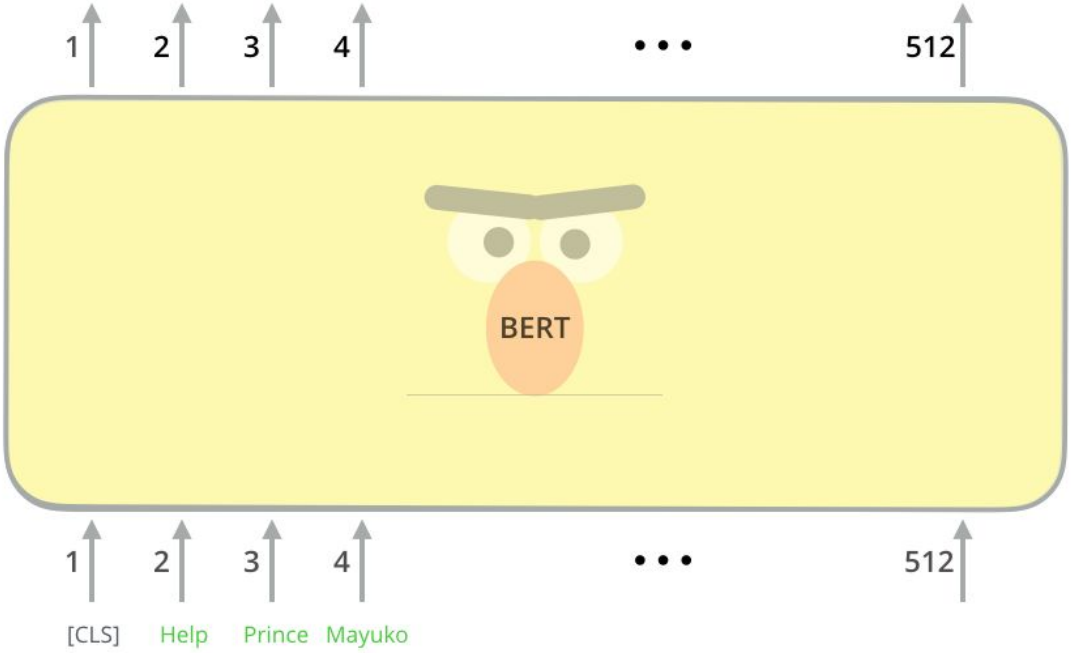
# BERT vs. Transformer

| |  |  | |
|---|---|---|---|
| | | **Base BERT** | **Large BERT** |
| Encoders | 6 | 12 | 24 |
| Units in FFN | 512 | 768 | 1024 |
| Attention Heads | 8 | 12 | 16 |

# Model inputs

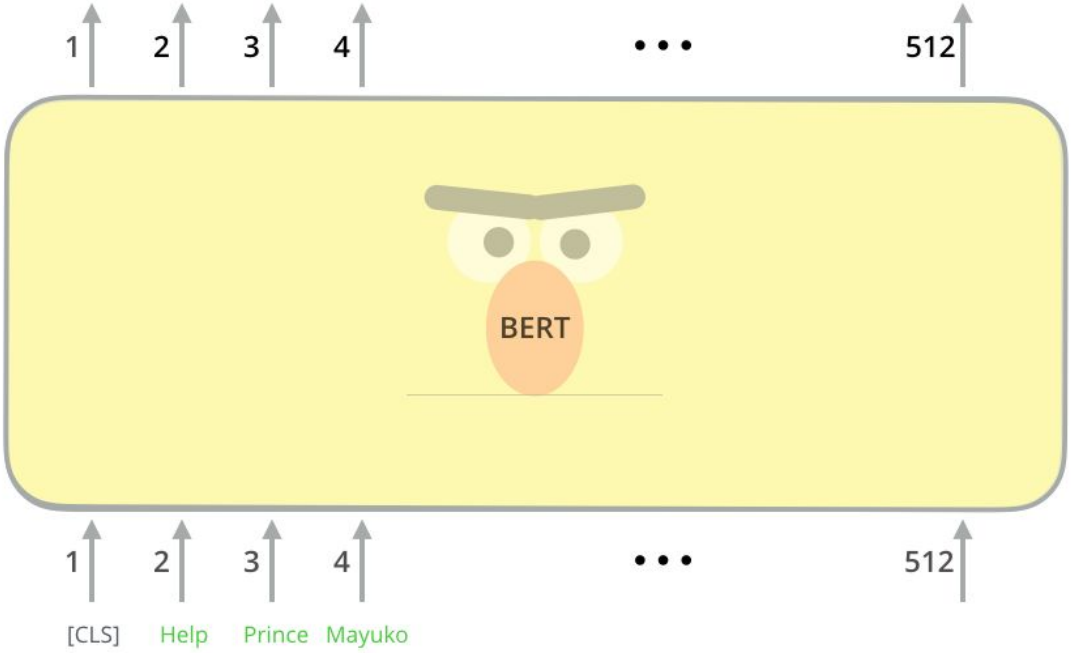


# Model inputs




Identical to the Transformer up until this point

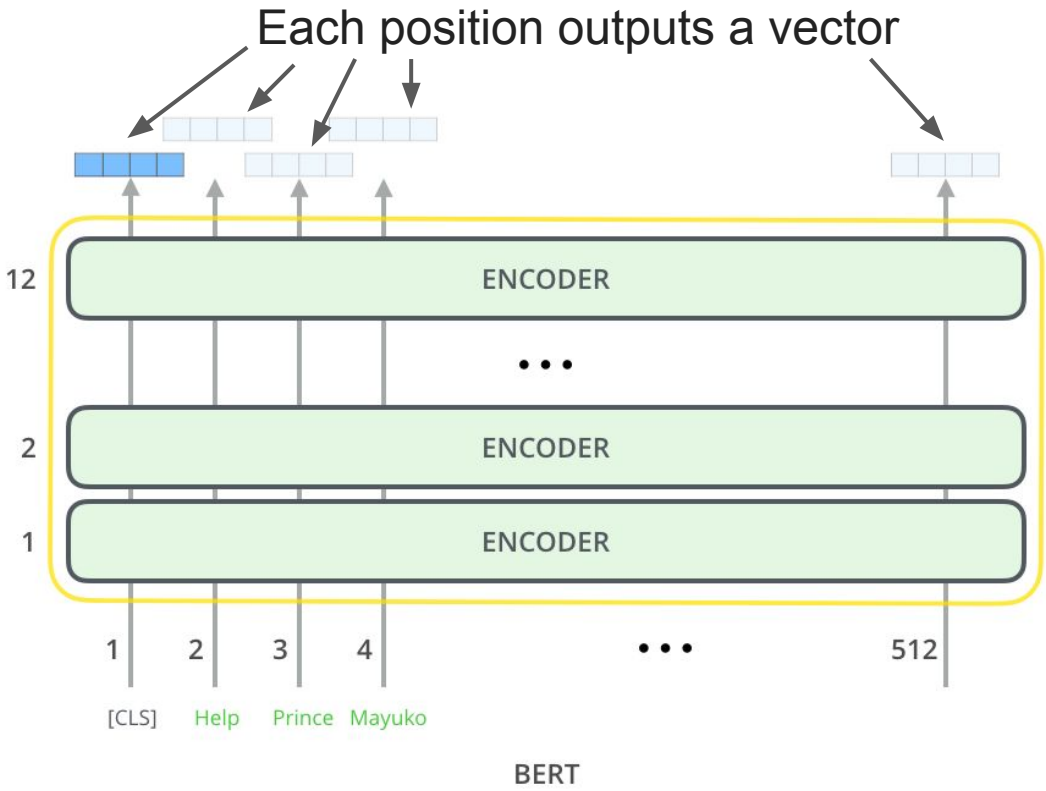
# Model inputs




Identical to the Transformer up until this point

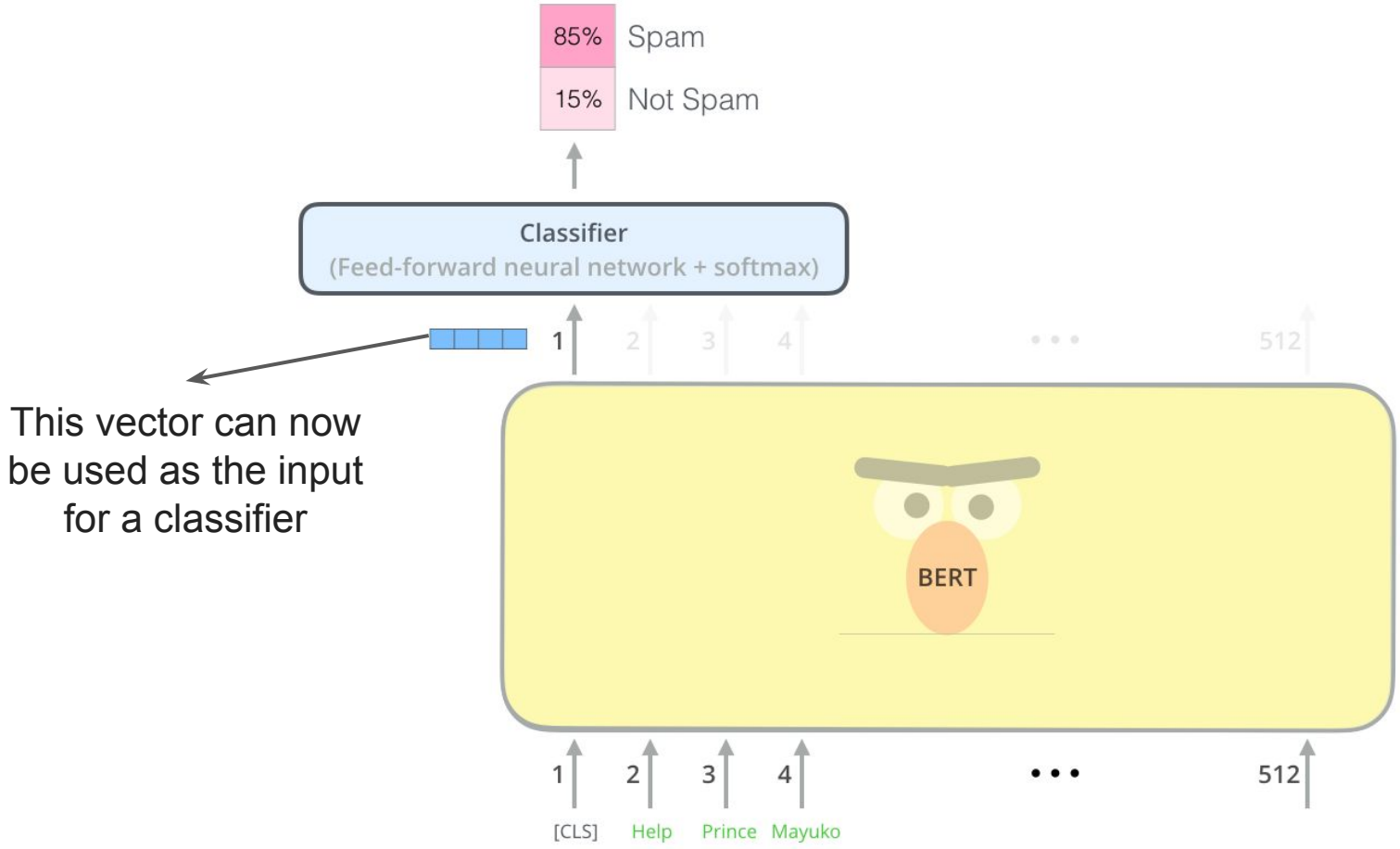
Why is BERT so special?

# Model outputs




For sentence classification we focus on the first position (that we passed [CLS] token to)

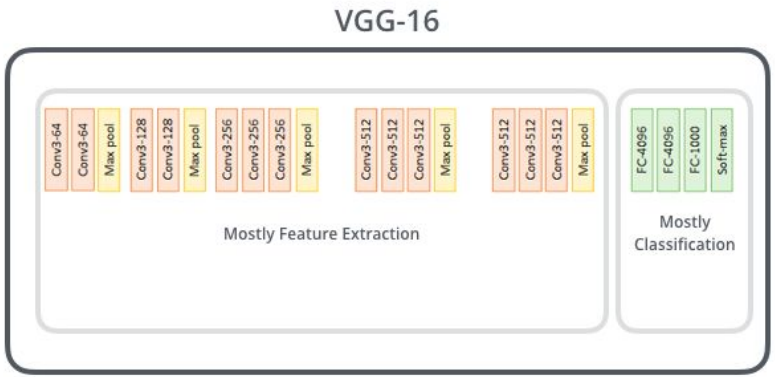
# Model inputs



# Similar to CNN concept!

Input

Features

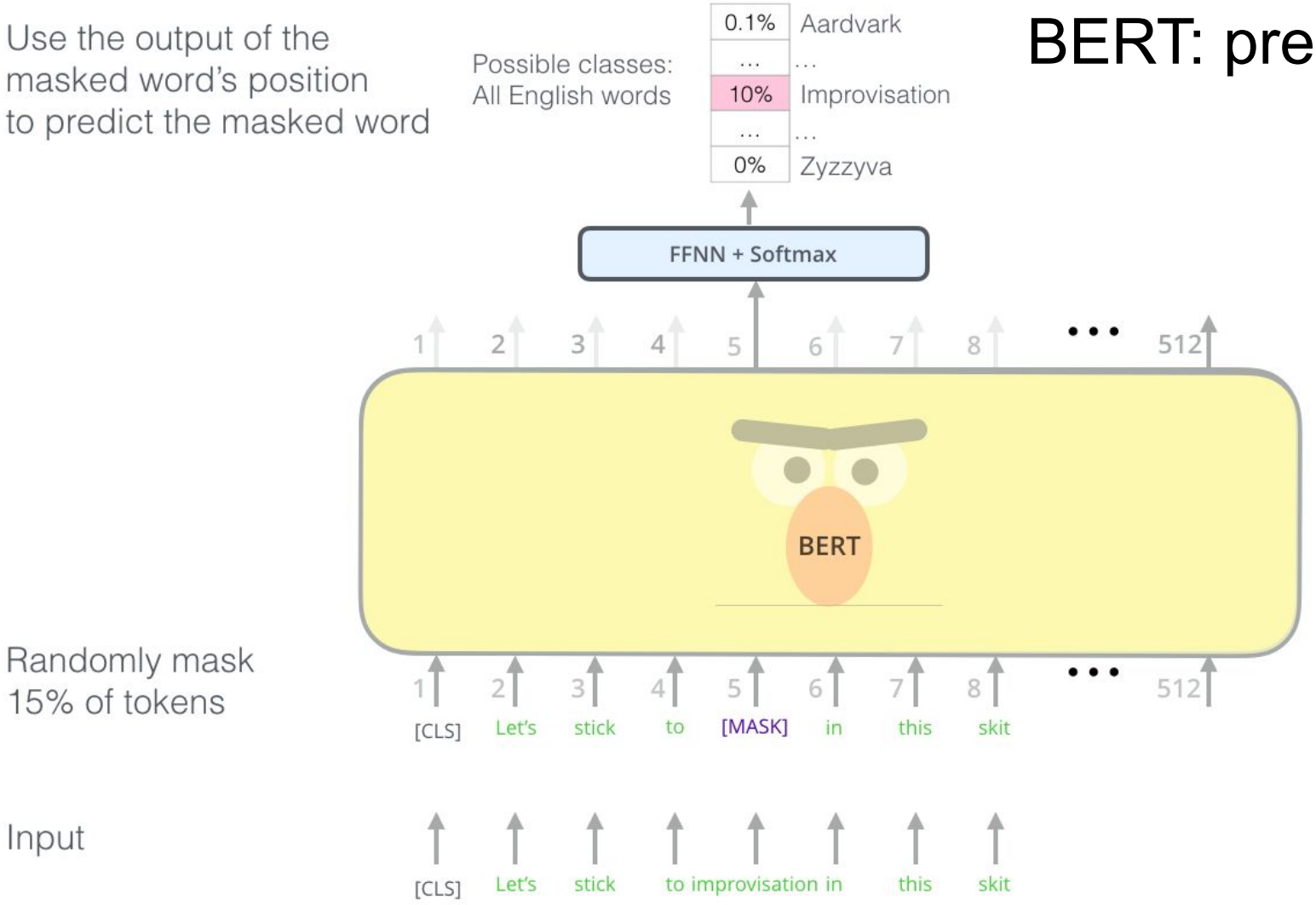




Output

Prediction

| | |
|---|---|
| 0.2% | Kit fox |
| 0.1% | English setter |
| 95% | Egyptian cat |
| 1% | Great Dane |
| | … |
| 0% | Hotdog |

# BERT: pre-training



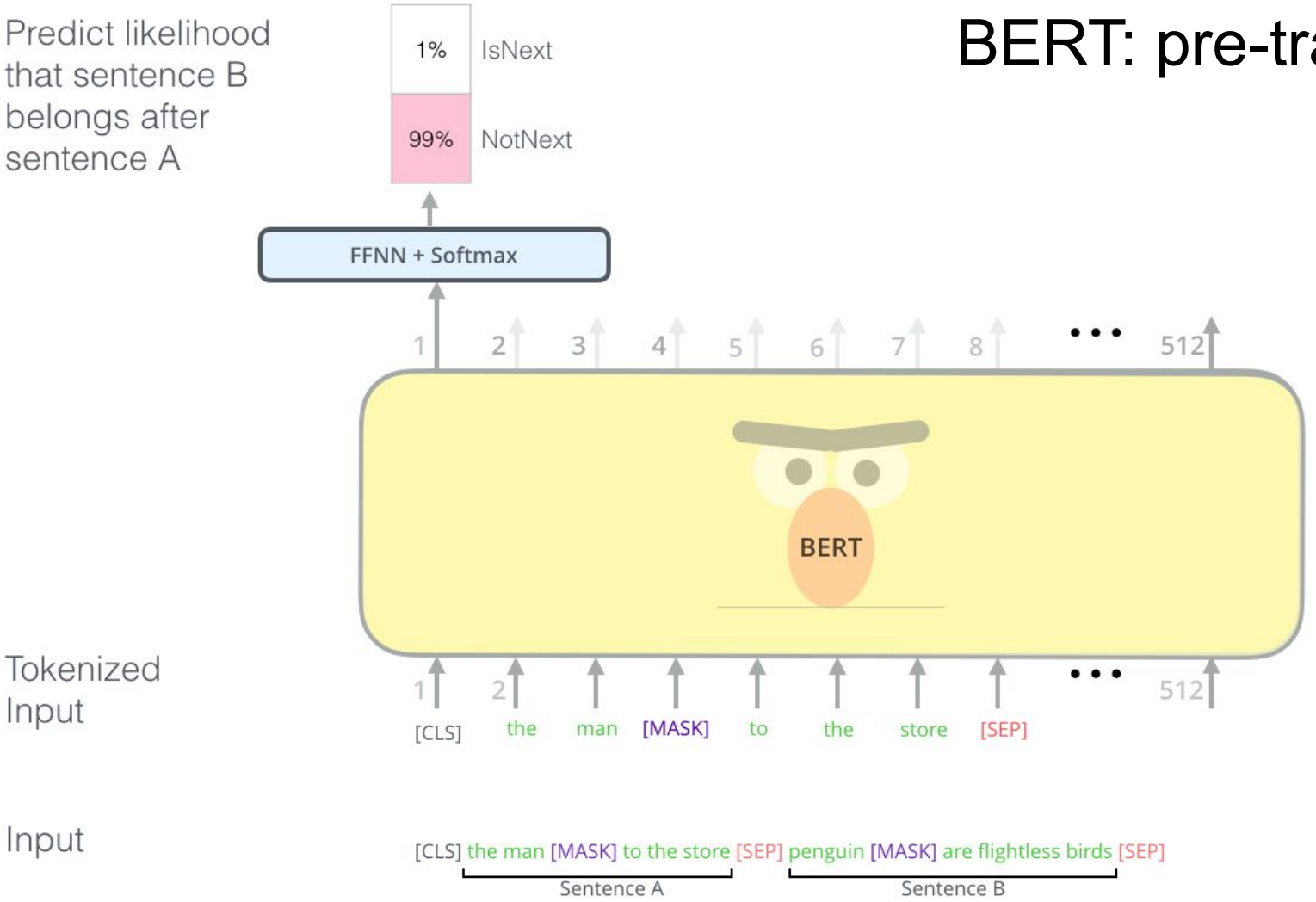
# BERT: pre-training

- “Masked Language Model” approach

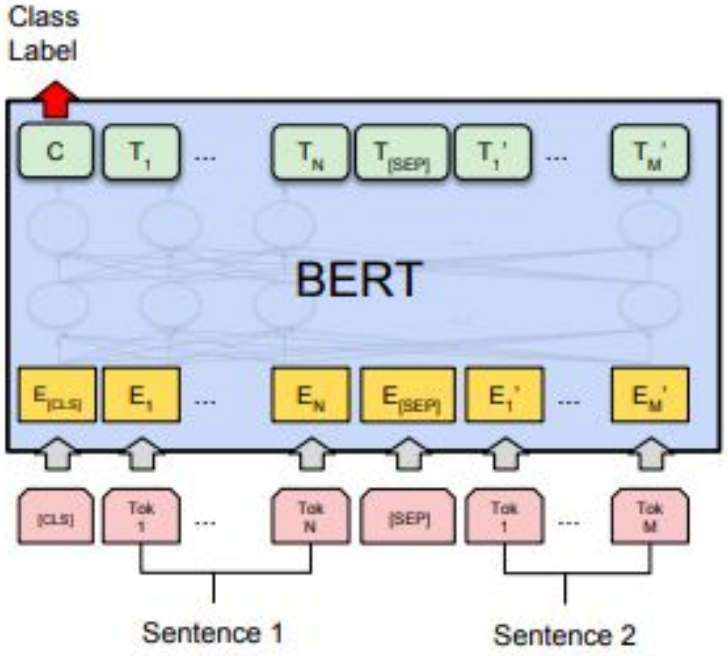
- To make BERT better at handling relationships between multiple sentences, the pre-training process includes an additional task:

  *“Given two sentences (A and B),is B likely to be the sentence that follows A, or not?”*

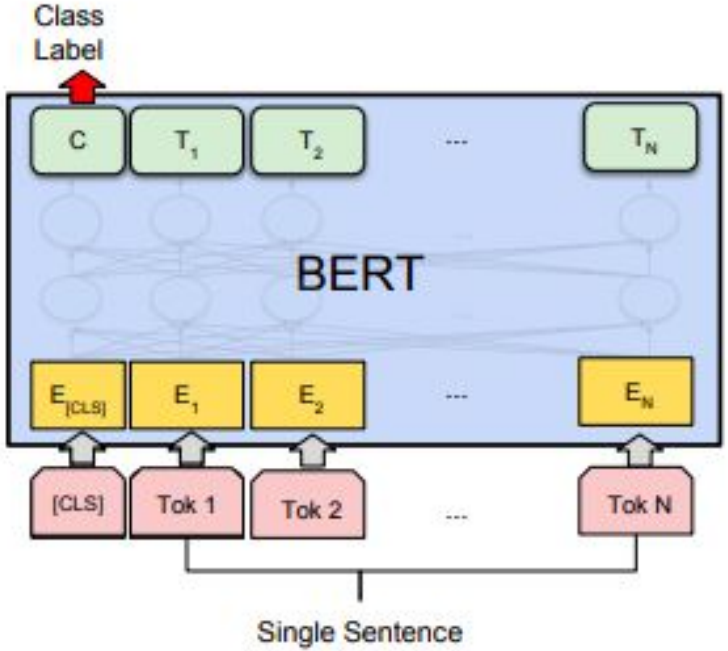
# BERT: pre-training



# BERT: fine-tuning for different tasks

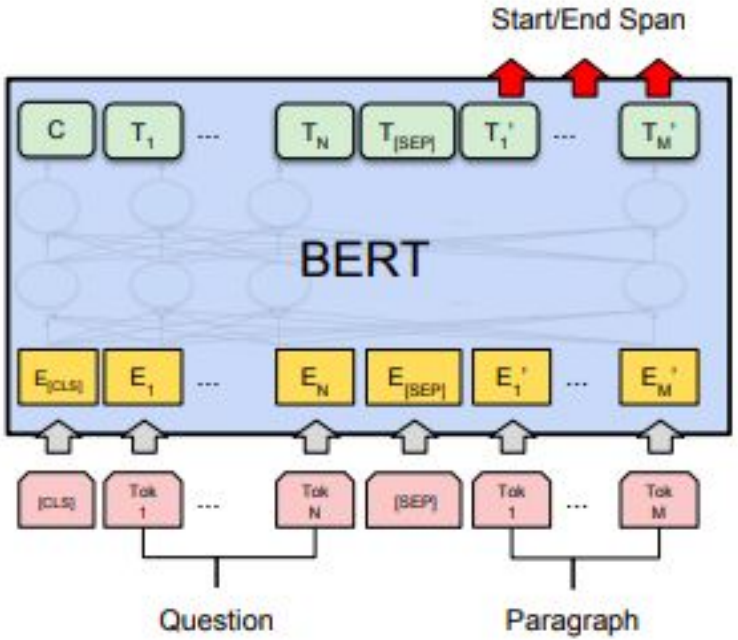



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

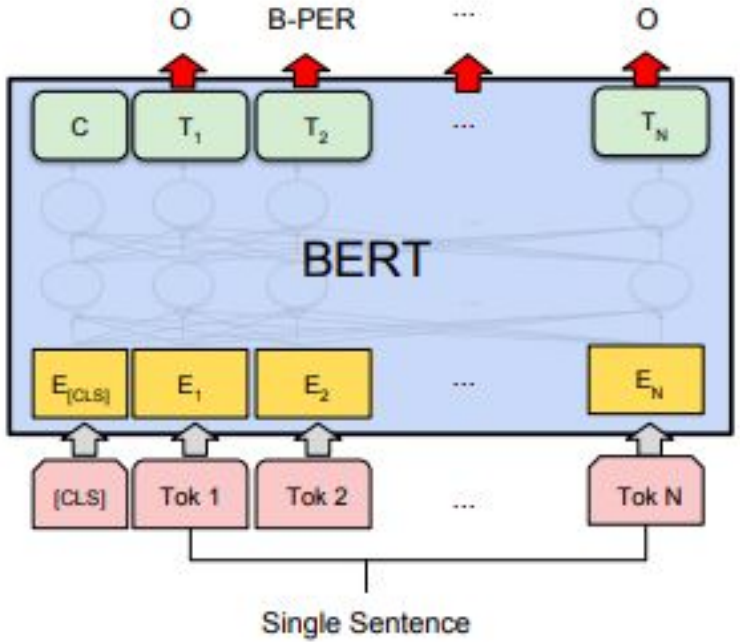



(b) Single Sentence Classification Tasks: SST-2, CoLA

# BERT: fine-tuning for different tasks

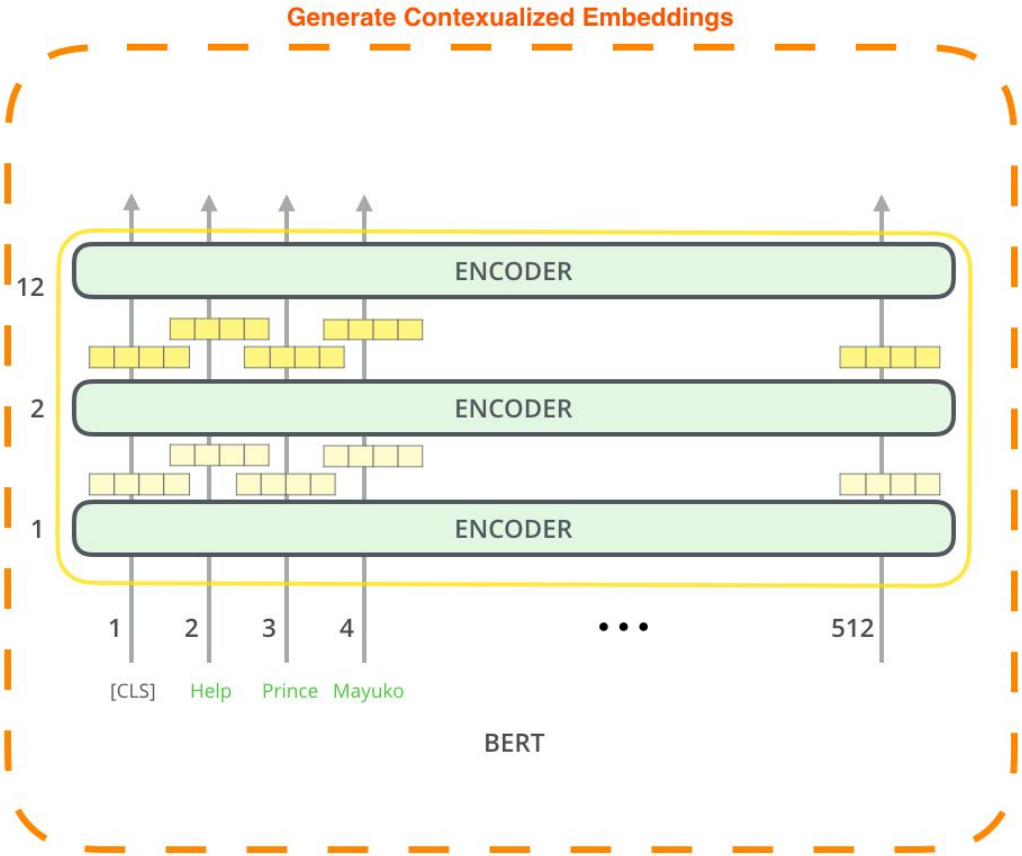



(c) Question Answering Tasks:
SQuAD v1.1

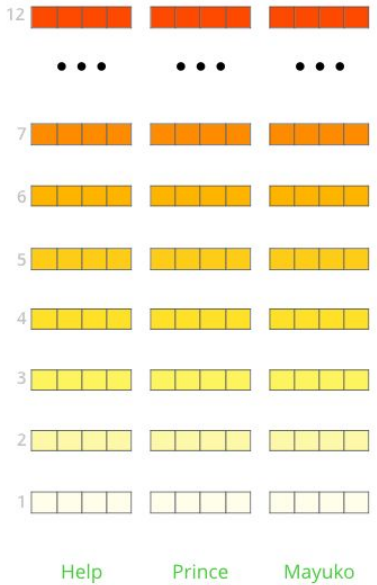



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

# BERT for feature extraction



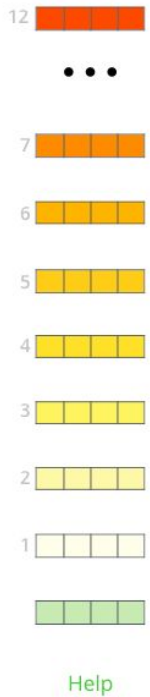

The output of each encoder layer along each token's path can be used as a feature representing that token.




**But which one should we use?**

# BERT for feature extraction

**What is the best contextualized embedding for "Help" in that context?**
For named-entity recognition task CoNLL-2003 NER



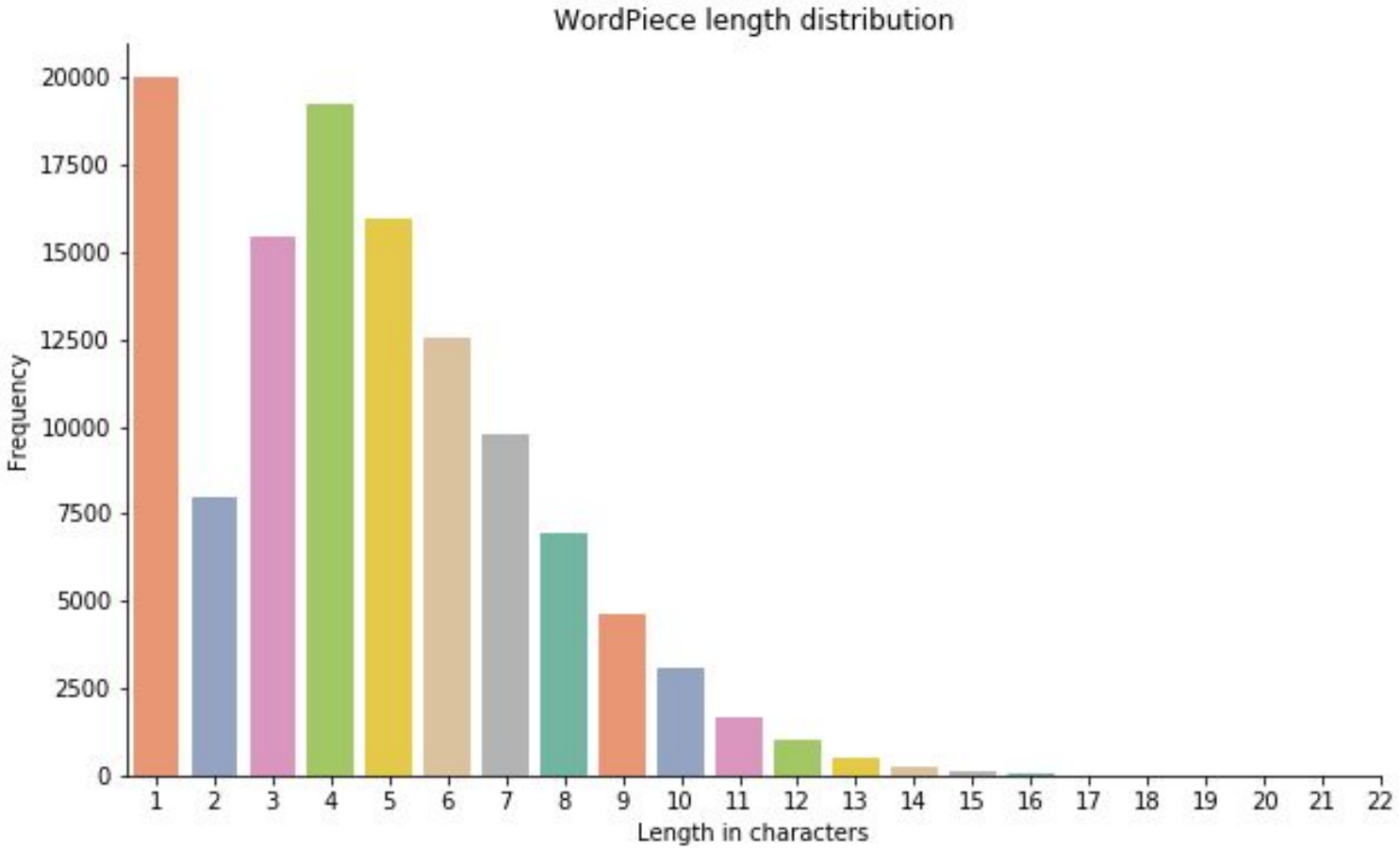

| | | Dev F1 Score |
|---|---|---|
| First Layer | Embedding | 91.0 |
| Last Hidden Layer | 12 | 94.9 |
| Sum All 12 Layers | 12 + ... + 2 + 1 = | 95.5 |
| Second-to-Last Hidden Layer | 11 | 95.6 |
| Sum Last Four Hidden | 12 + 11 + 10 + 9 = | 95.9 |
| Concat Last Four Hidden | 9 10 11 12 | 96.1 |

# BERT: tokenization

**Example: Unaffable -> un, ##aff, ##able**

- Single model for 104 languages with a large shared vocabulary (119,547 WordPiece model)
- Non-word-initial units are prefixed with ##
- The first 106 symbols: constants like PAD and UNK
- 36.5% of the vocabulary are non-initial word pieces
- The alphabet consists of 9,997 unique characters that are defined as word-initial (C) and continuation symbols (##C), which together make up 19,994 word pieces
- The rest are multicharacter word pieces of various length.

# BERT: tokenization

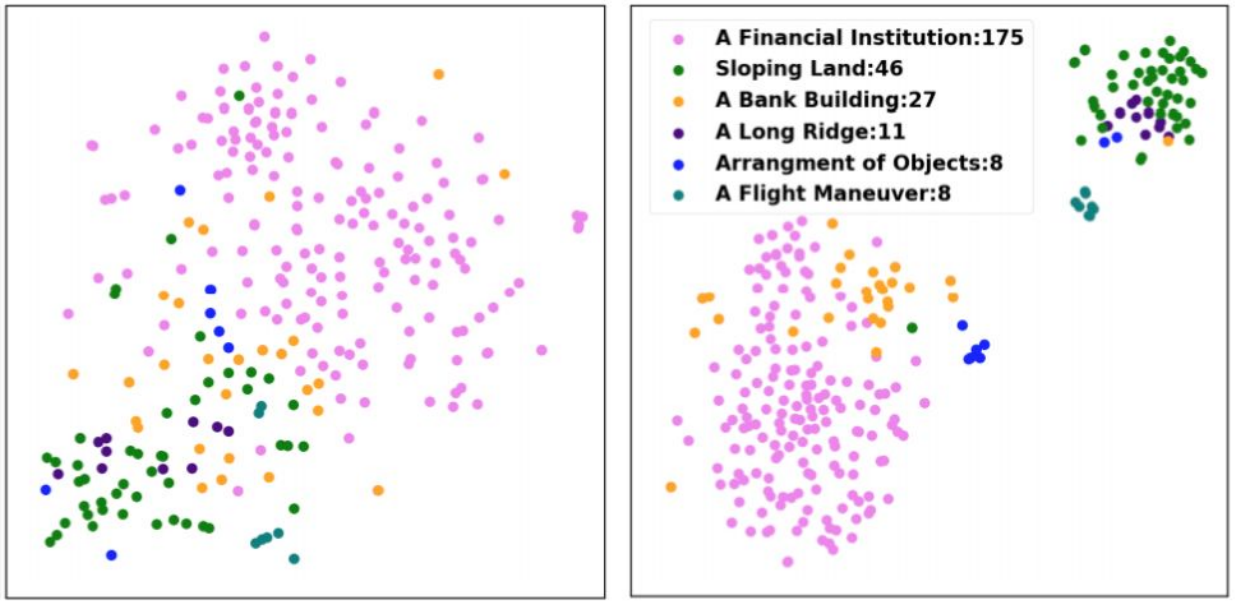


# BERT: overview

- BERT repo
- Try out BERT on TPU
- WordPieces Tokenizer
- PyTorch Implementation of BERT

# BERT vs. ELMO (Word Sense Disambiguation problem)

T-SNE plots of different senses of 'bank'




(c) ELMo (a) BERT

# GPT, GPT-2 and GPT-3

# GPT-2

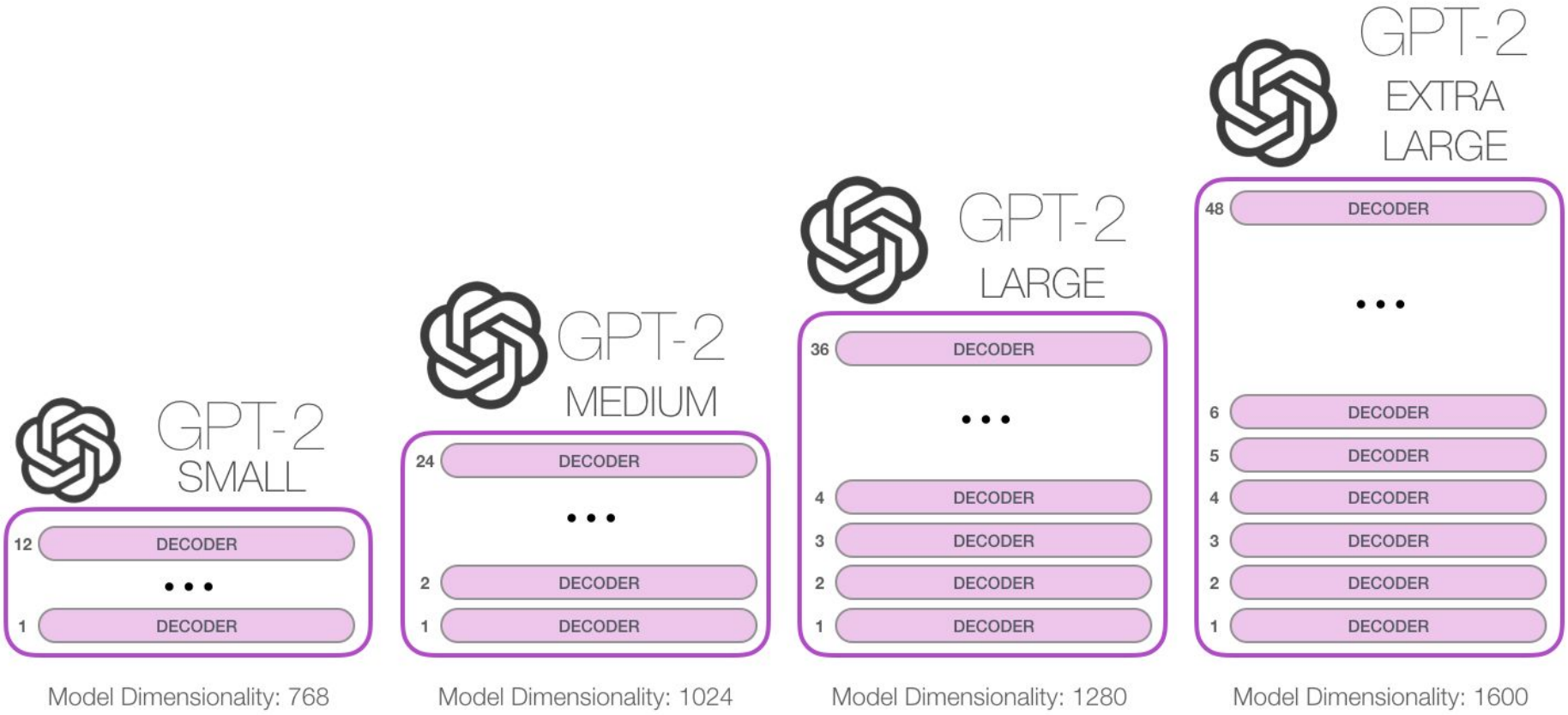
- Transformer-based architecture
- Trained to predict the **next** word
- 1.5 billion parameters
- Trained on 8 million web-pages (dataset WebText)

# GPT-2

- Transformer-based architecture
- Trained to predict the **next** word
- 1.5 billion parameters
- Trained on 8 million web-pages (dataset WebText)

On language tasks (question answering, reading comprehension, summarization, translation) works well **WITHOUT** fine-tuning

# GPT-2




http://jalammar.github.io/illustrated-gpt2/

# GPT-2: question answering

EXAMPLES

*Who wrote the book the origin of species?*

**Correct answer**: *Charles Darwin*
**Model answer**: `Charles Darwin`

*What is the largest state in the U.S. by land mass?*

**Correct answer**: *Alaska*
**Model answer**: `California`

# GPT-2: language modeling

EXAMPLE

*Both its sun-speckled shade and the cool grass beneath were a welcome respite after the stifling kitchen, and I was glad to relax against the tree's rough, brittle bark and begin my breakfast of buttery, toasted bread and fresh fruit. Even the water was tasty, it was so clean and cold. It almost made up for the lack of...*

**Correct answer**: *coffee*
**Model answer**: `food`

# GPT-2: machine translation

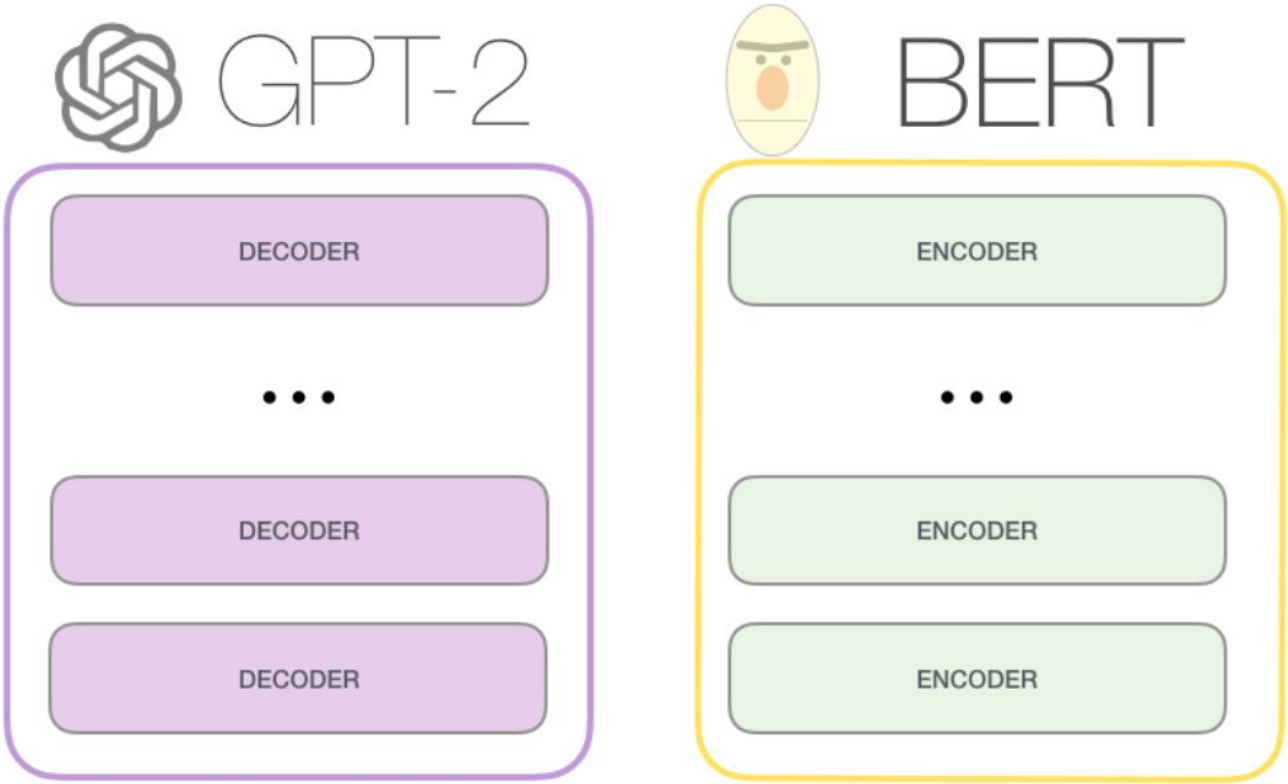
EXAMPLE

**French sentence**:
*Un homme a expliqué que l'opération gratuite qu'il avait subie pour soigner une hernie lui permettrait de travailler à nouveau.*

**Reference translation**:
*One man explained that the free hernia surgery he'd received will allow him to work again.*

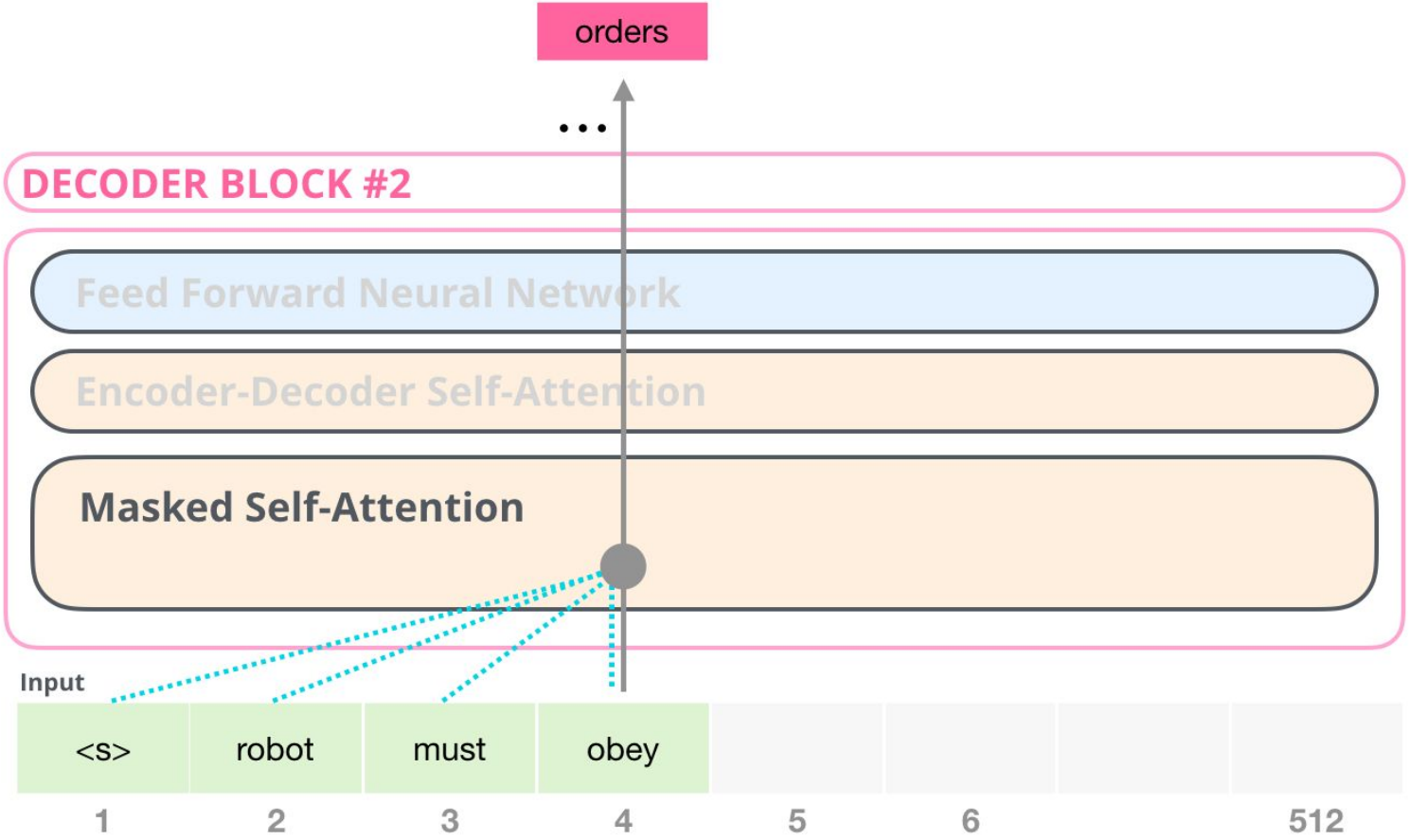
**Model translation**:
`A man told me that the operation gratuity he had been promised would not allow him to travel.`

# GPT-2 vs. BERT



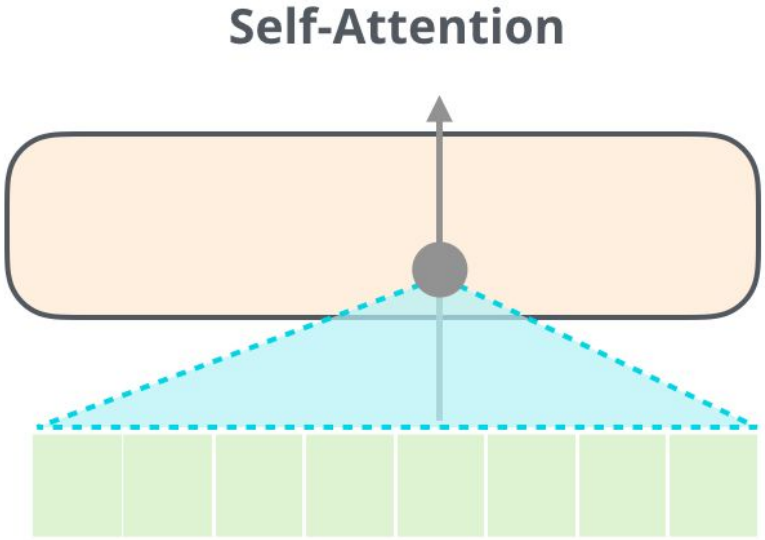

http://jalammar.github.io/illustrated-gpt2/

# GPT-2 vs. BERT

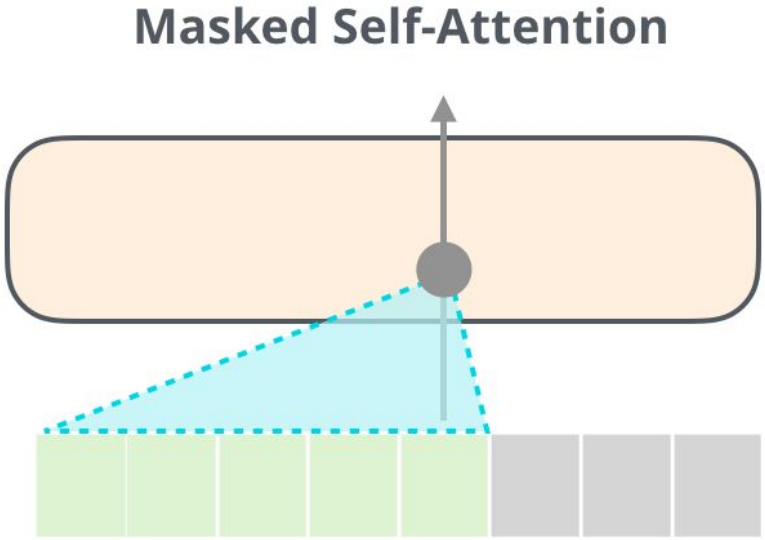



http://jalammar.github.io/illustrated-gpt2/

# GPT-3 vs. BERT



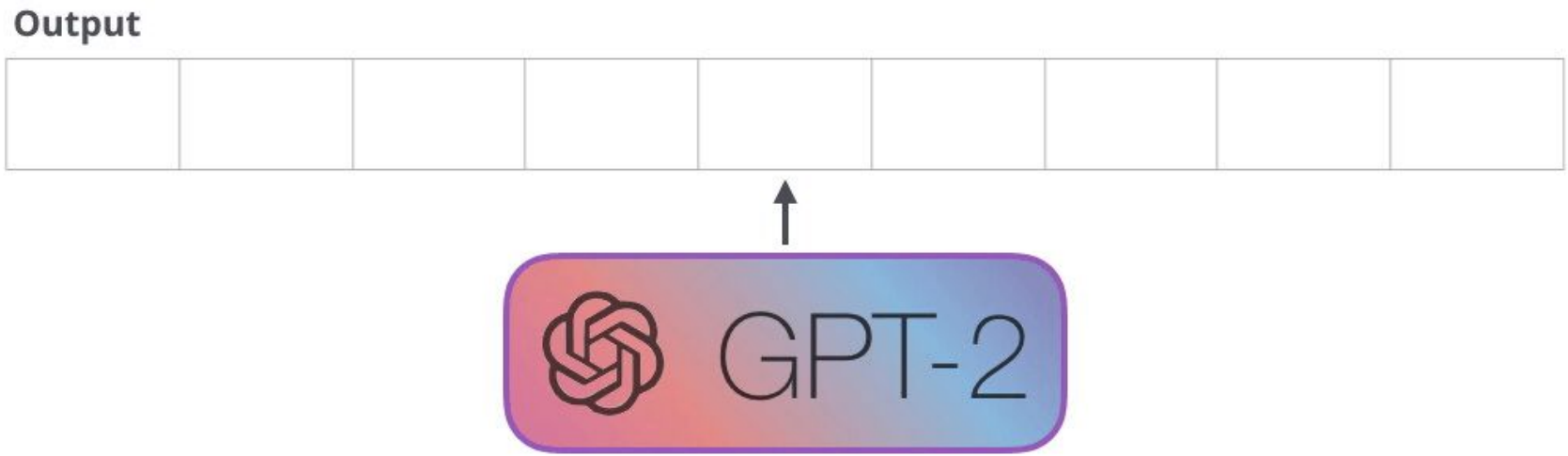

GPT




BERT

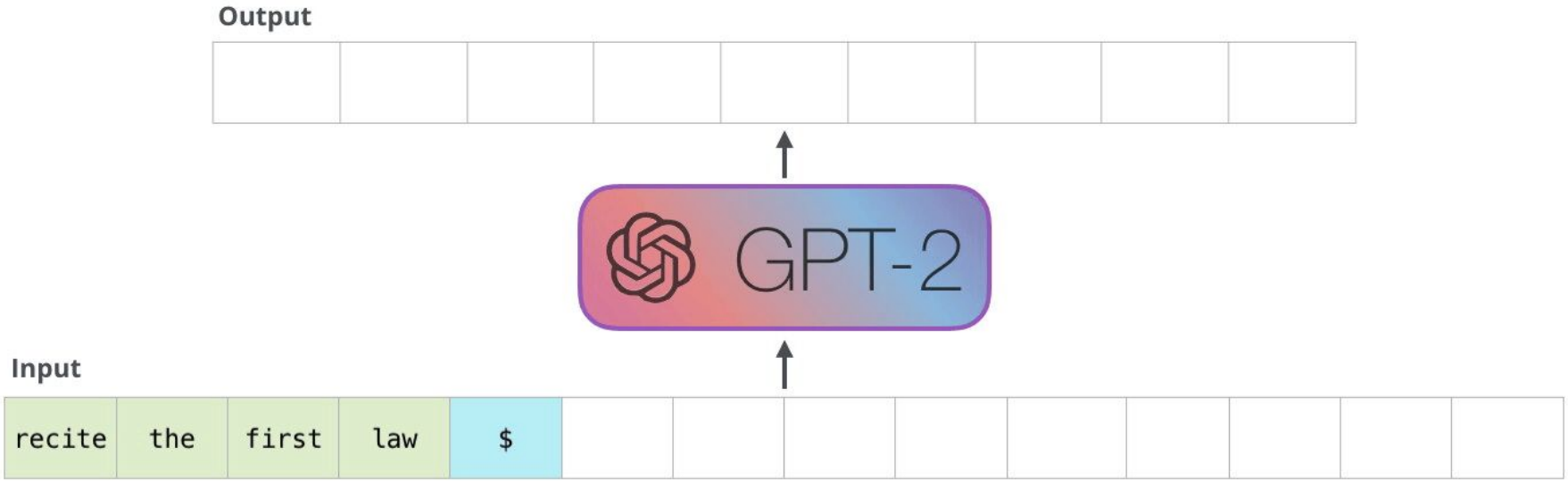
http://jalammar.github.io/illustrated-gpt2/

# GPT-2: autoregression



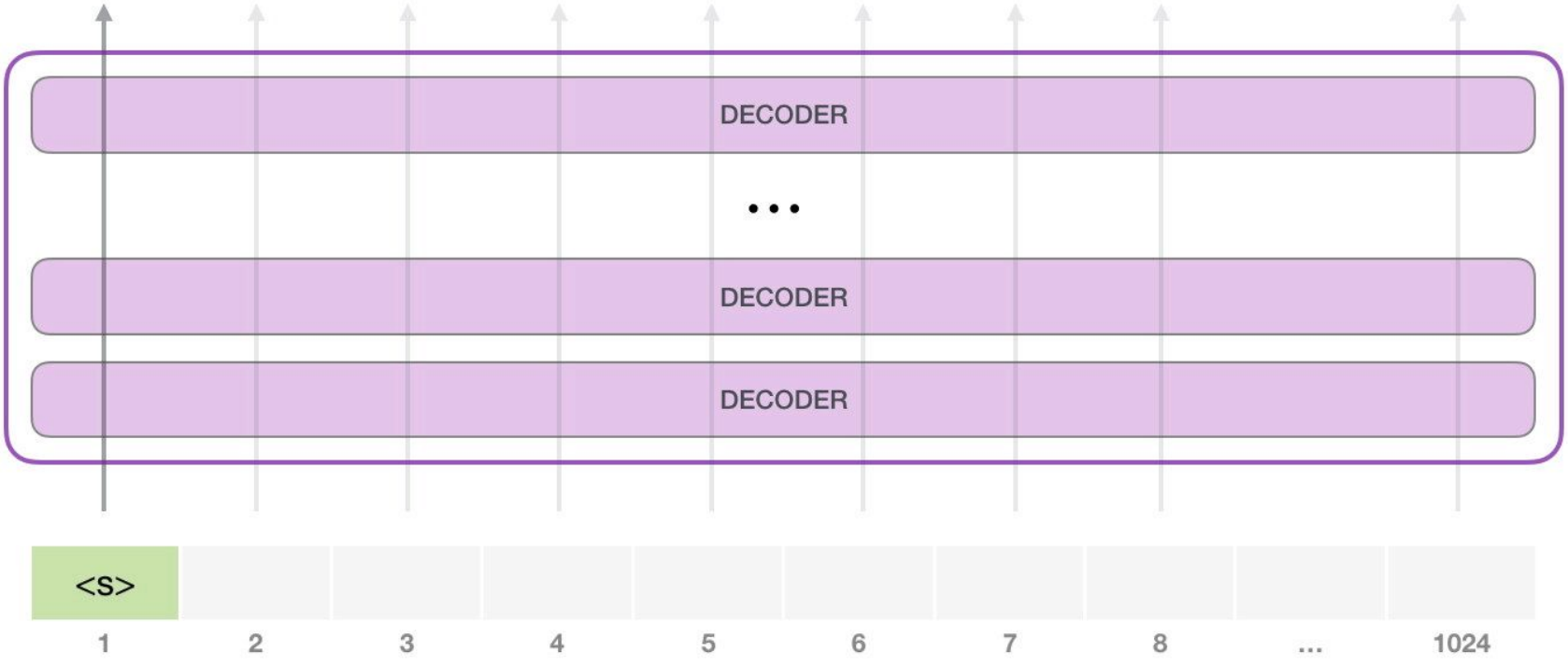

http://jalammar.github.io/illustrated-gpt2/

# GPT-2: autoregression



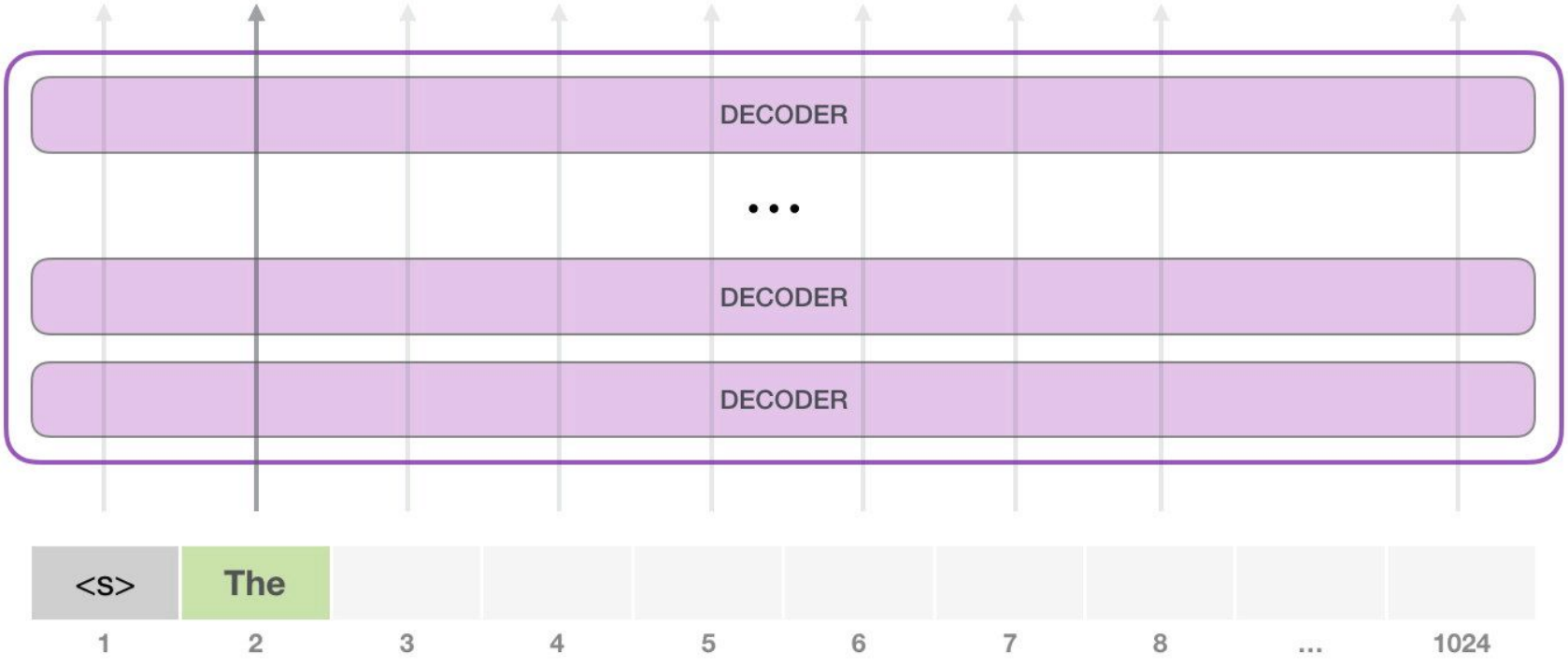

http://jalammar.github.io/illustrated-gpt2/

# GPT-2: text generation




http://jalammar.github.io/illustrated-gpt2/

# GPT-2: text generation




http://jalammar.github.io/illustrated-gpt2/

# GPT-2: fake news and hype

**New AI fake text generator may be too dangerous to ... - The Guardian**
https://www.theguardian.com/.../elon-musk-backed-ai-writes-convincing-news-fiction
4 days ago - The Elon Musk-backed nonprofit company OpenAI declines to release research publicly for fear of misuse. The creators of a revolutionary AI system that can write news stories and works of fiction – dubbed "deepfakes for text" – have taken the unusual step of not releasing ...

**OpenAI built a text generator so good, it's considered too dangerous to ...**
https://techcrunch.com/2019/02/17/openai-text-generator-dangerous/
12 hours ago - A storm is brewing over a new language model, built by non-profit artificial intelligence research company OpenAI, which it says is so good at ...

**The AI Text Generator That's Too Dangerous to Make Public | WIRED**
https://www.wired.com/story/ai-text-generator-too-dangerous-to-make-public/
4 days ago - In 2015, car-and-rocket man Elon Musk joined with influential startup backer Sam Altman to put artificial intelligence on a new, more open ...

**Elon Musk-backed AI Company Claims It Made a Text Generator ...**
https://gizmodo.com/elon-musk-backed-ai-company-claims-it-made-a-text-gener-183...
Elon Musk-backed AI Company Claims It Made a Text Generator That's **Too Dangerous to** Release · Rhett Jones · Friday 12:15pm · Filed to: OpenAI Filed to: ...

**Scientists have made an AI that they think is too dangerous to ...**
https://www.weforum.org/.../amazing-new-ai-churns-out-coherent-paragraphs-of-text/
3 days ago - Sample outputs suggest that the AI system is an extraordinary step forward, producing text rich with context, nuance and even something ...

**New AI Fake Text Generator May Be Too Dangerous To ... - Slashdot**
https://news.slashdot.org/.../new-ai-fake-text-generator-may-be-too-dangerous-to-rele...
3 days ago - An anonymous reader shares a report: The creators of a revolutionary AI system that can write news stories and works of fiction -- dubbed ...

Top stories

OpenAI built a text generator so good, it's considered too dangerous to release
TechCrunch
11 hours ago

Elon Musk's AI company created a fake news generator it's too scared to make public
BGR.com
9 hours ago

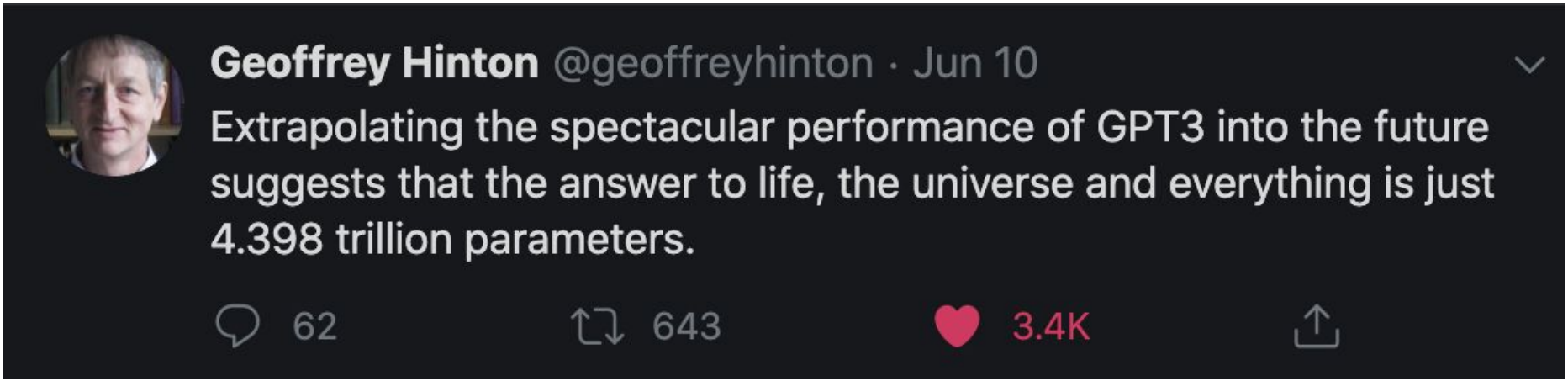
The AI That Can Write A Fake News Story From A Handful Of Words
NDTV.com
2 hours ago

When Is Technology Too Dangerous to Release to the Public?
Slate • 2 days ago

Scientists Developed an AI So Advanced They Say It's Too Dangerous to Release
ScienceAlert • 6 days ago

# GPT-3

- GPT-2: 1.5 billion parameters
- GPT-3: **175 billion** parameters



# GPT-3: OpenAI API released

- You can request access in order to integrate the API into your product
- Given any text prompt, the API will return a text completion, attempting to match the pattern you gave it

https://openai.com/blog/openai-api/

# Transformer XL

# Transformer-XL

- Vanila Transformer works with a fixed-length context at training time. That's why:
  - the algorithm is not able to model dependencies that are longer than a fixed length.
  - the segments usually do not respect the sentence boundaries, resulting in context fragmentation which leads to inefficient optimization.

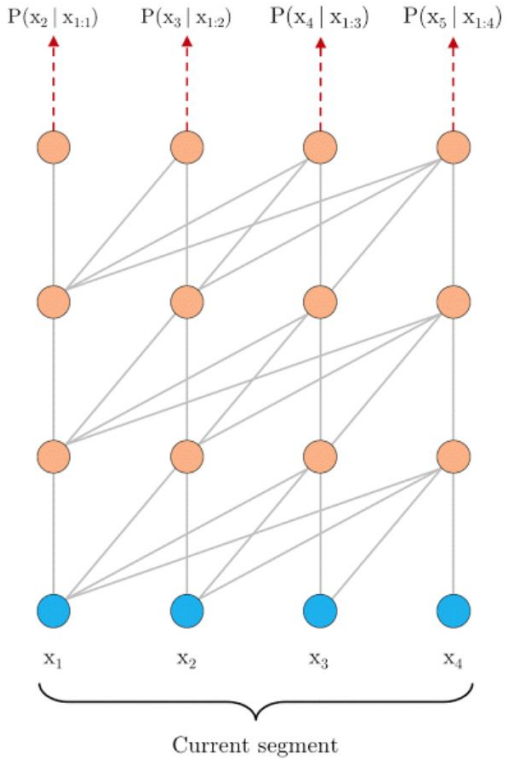
# Segment-level Recurrence

- During training, the representations computed for the previous segment are fixed and cached to be reused as an extended context when the model processes the next new segment.

- Contextual information is now able to flow across segment boundaries.

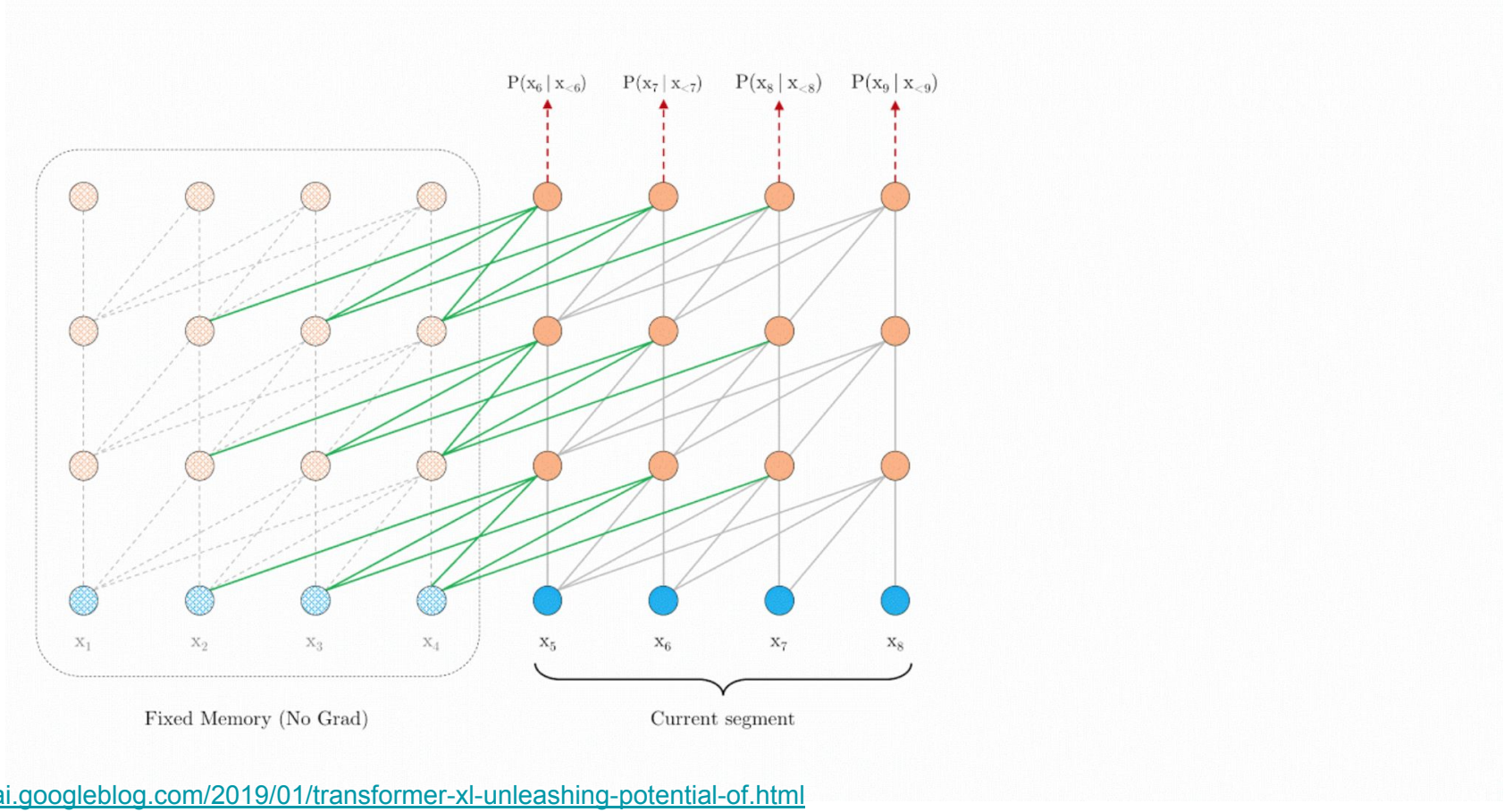
- Recurrence mechanism also resolves the context fragmentation issue, providing necessary context for tokens in the front of a new segment.

# Segment-level Recurrence



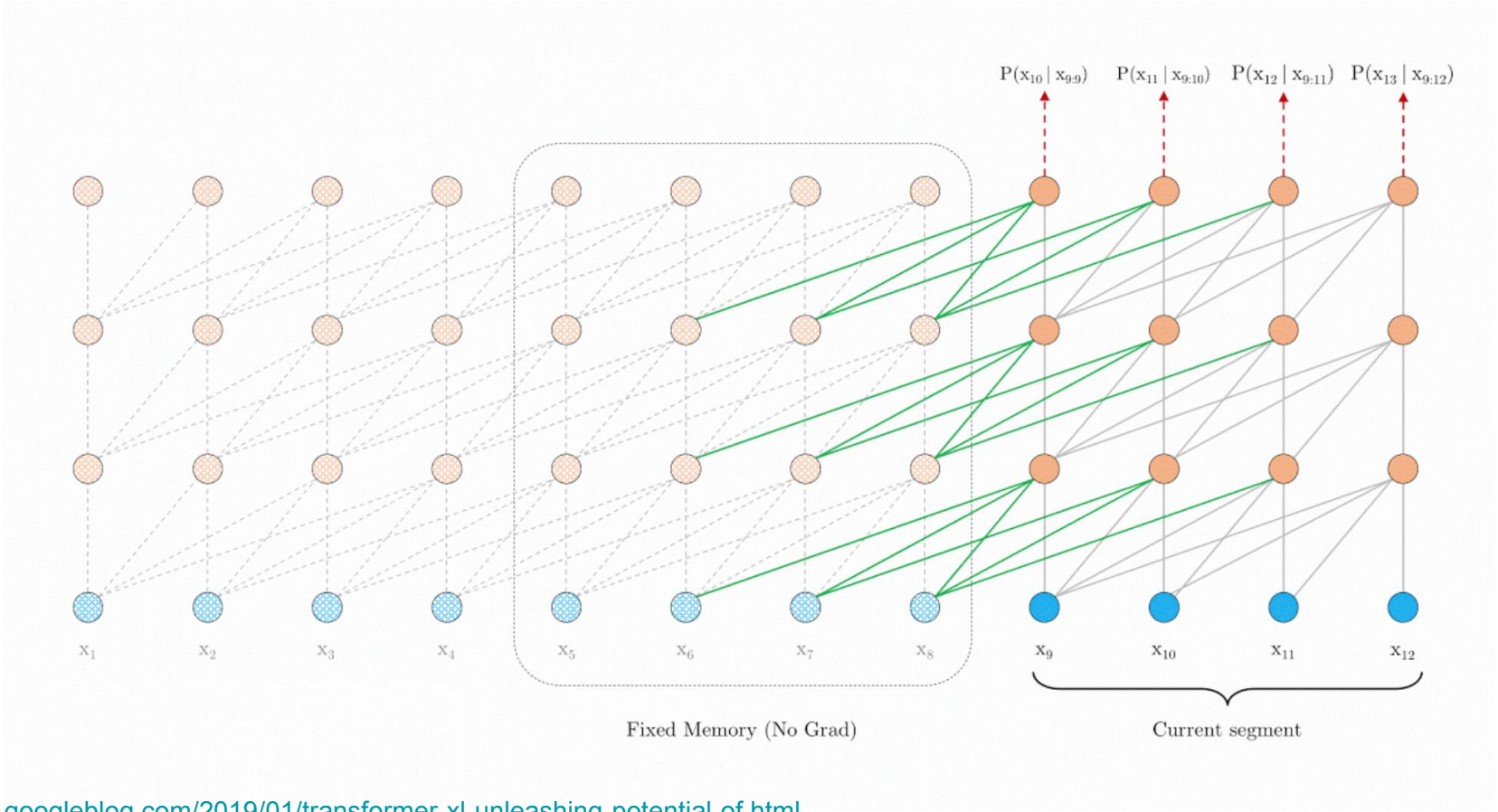

https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html

# Segment-level Recurrence




https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html

# Segment-level Recurrence



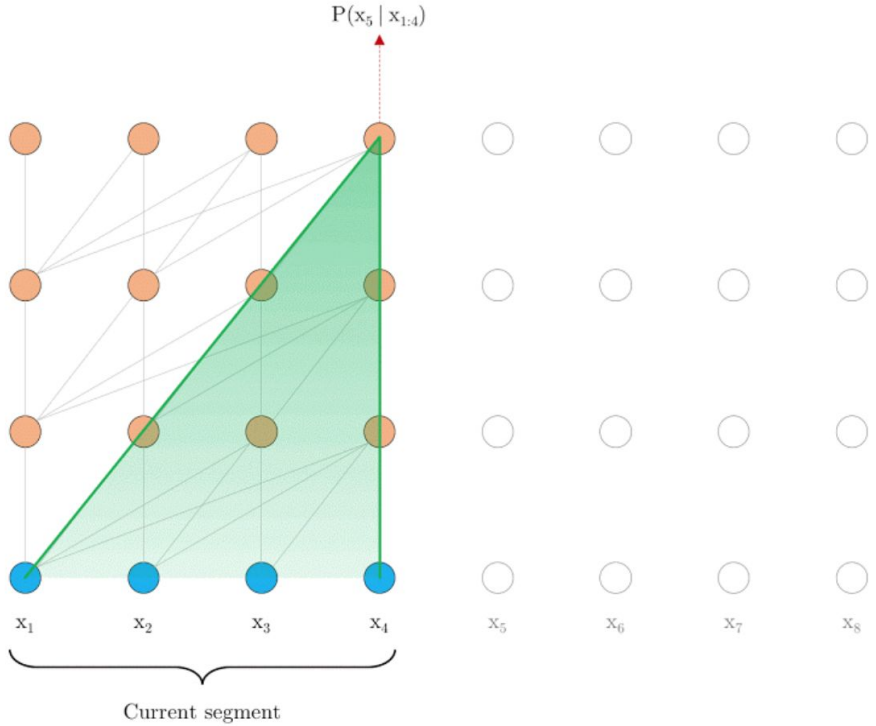

https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html

# Relative Positional Encodings

- Fixed embeddings with learnable transformations instead of learnable embeddings

  As a result:
- more generalizable to longer sequences at test time
- longer effective context

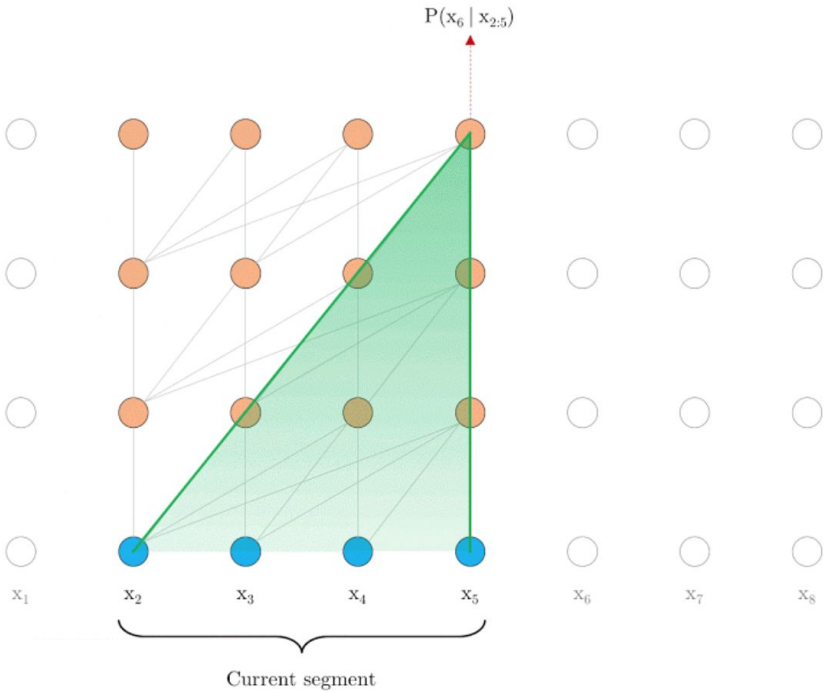
# Vanila Transformer vs. Transformer-XL




Vanilla Transformer with a fixed-length context at evaluation time

https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html

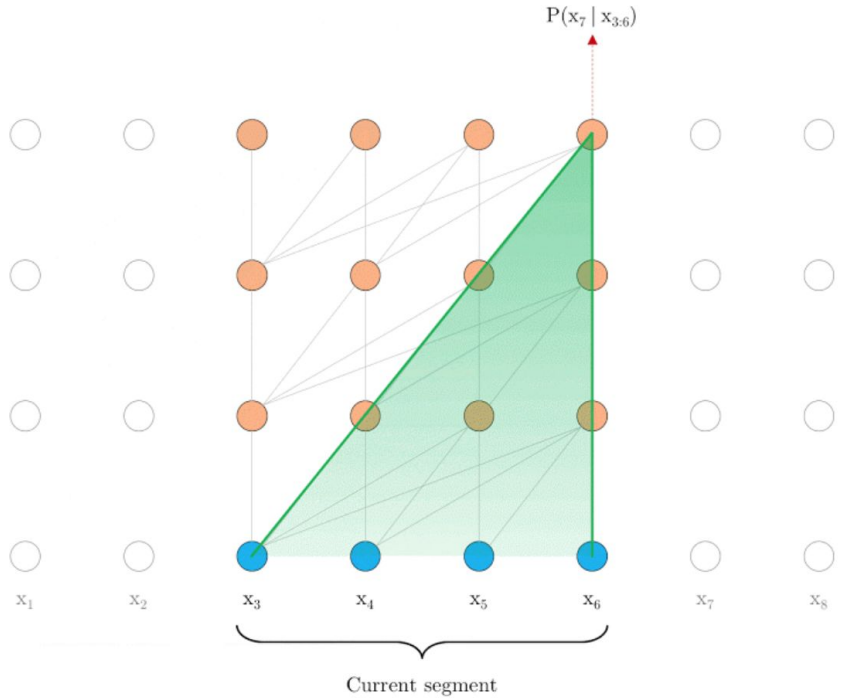
# Vanila Transformer vs. Transformer-XL




Vanilla Transformer with a fixed-length context at evaluation time

https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html

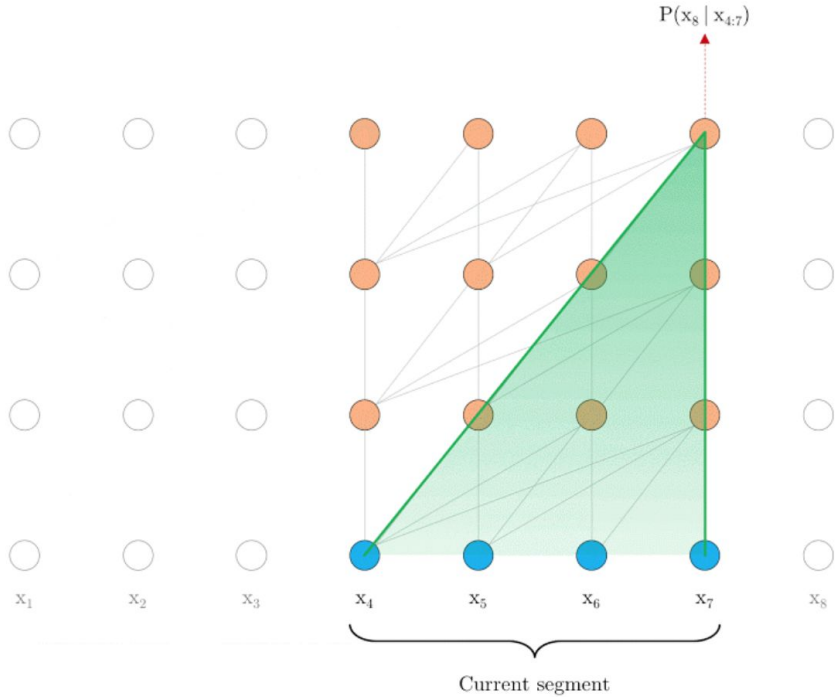
# Vanila Transformer vs. Transformer-XL




Vanilla Transformer with a fixed-length context at evaluation time

https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html

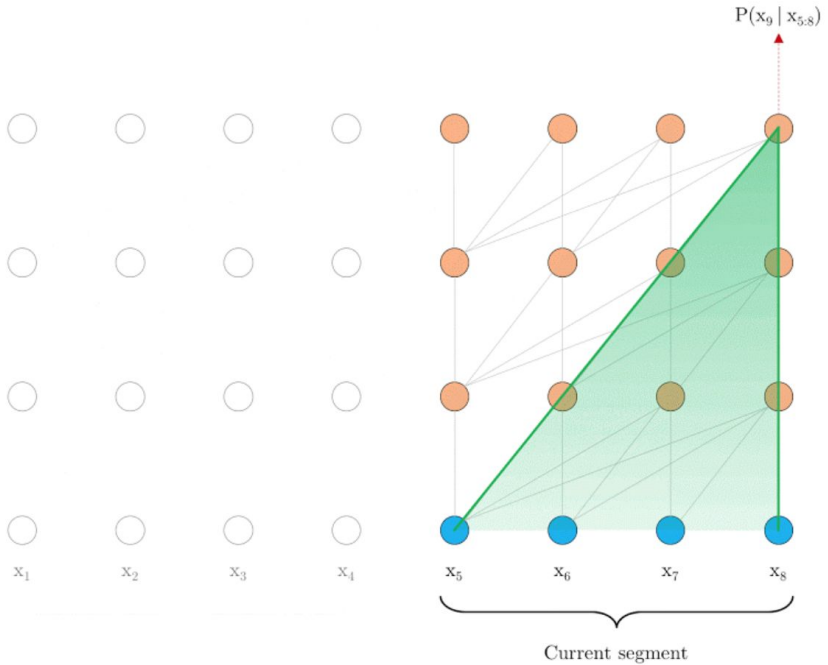
# Vanila Transformer vs. Transformer-XL




Vanilla Transformer with a fixed-length context at evaluation time

https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html

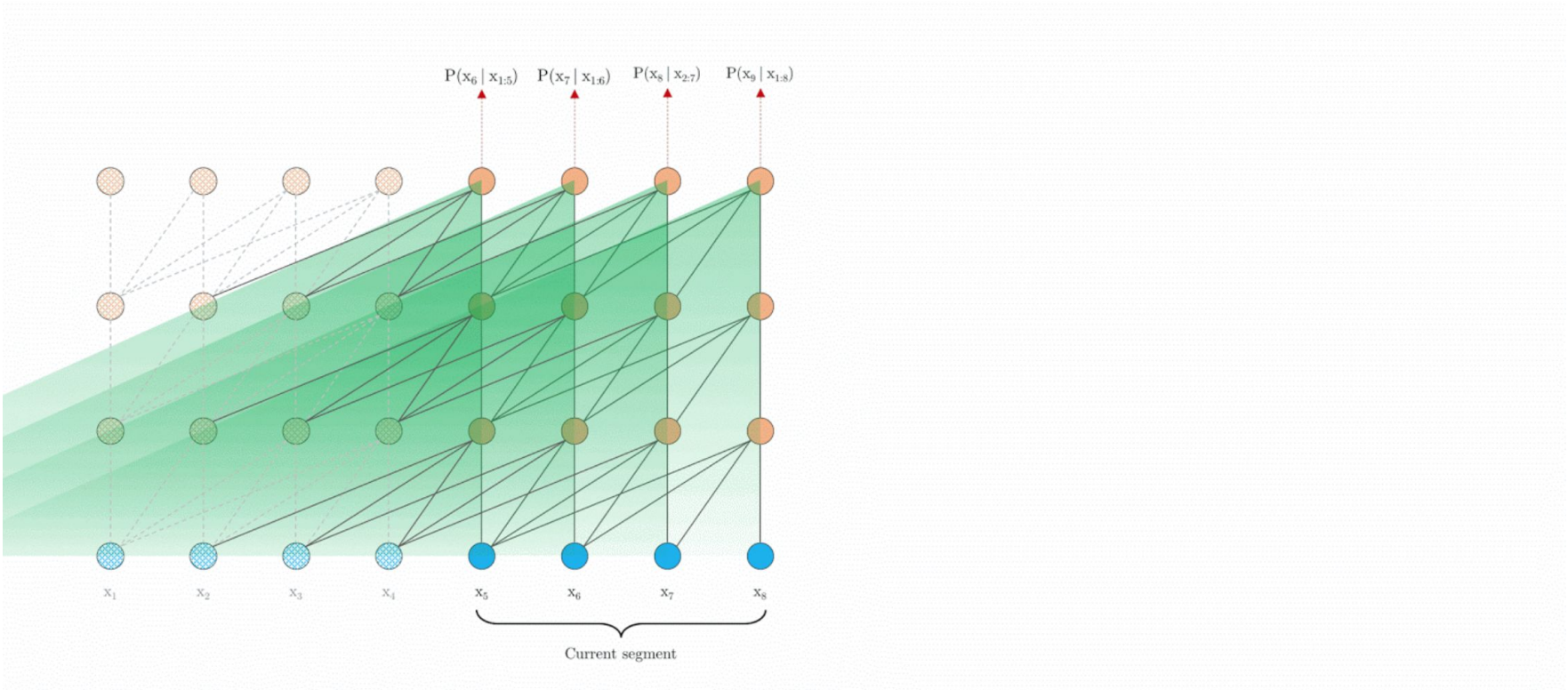
# Vanila Transformer vs. Transformer-XL




Vanilla Transformer with a fixed-length context at evaluation time

https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html

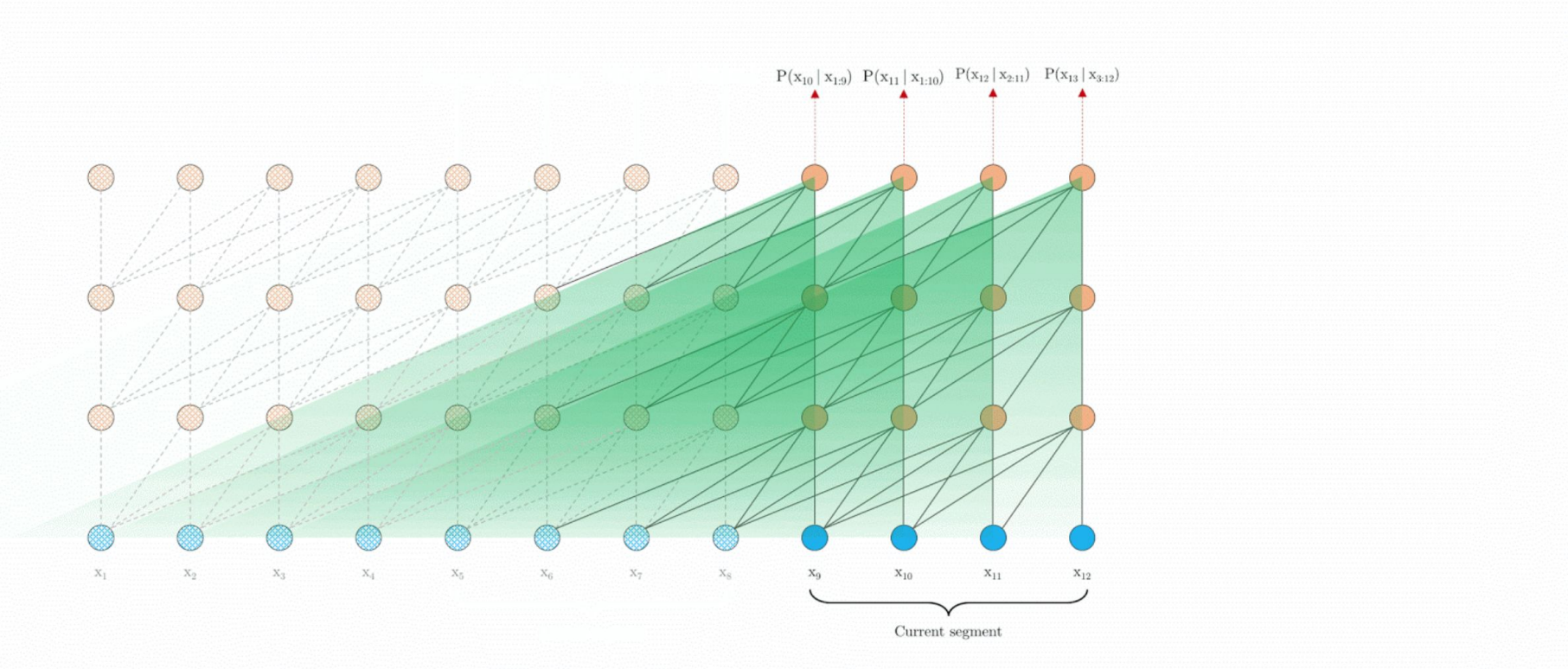
# Vanila Transformer vs. Transformer-XL




Transformer-XL with segment-level recurrence at evaluation time

https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html

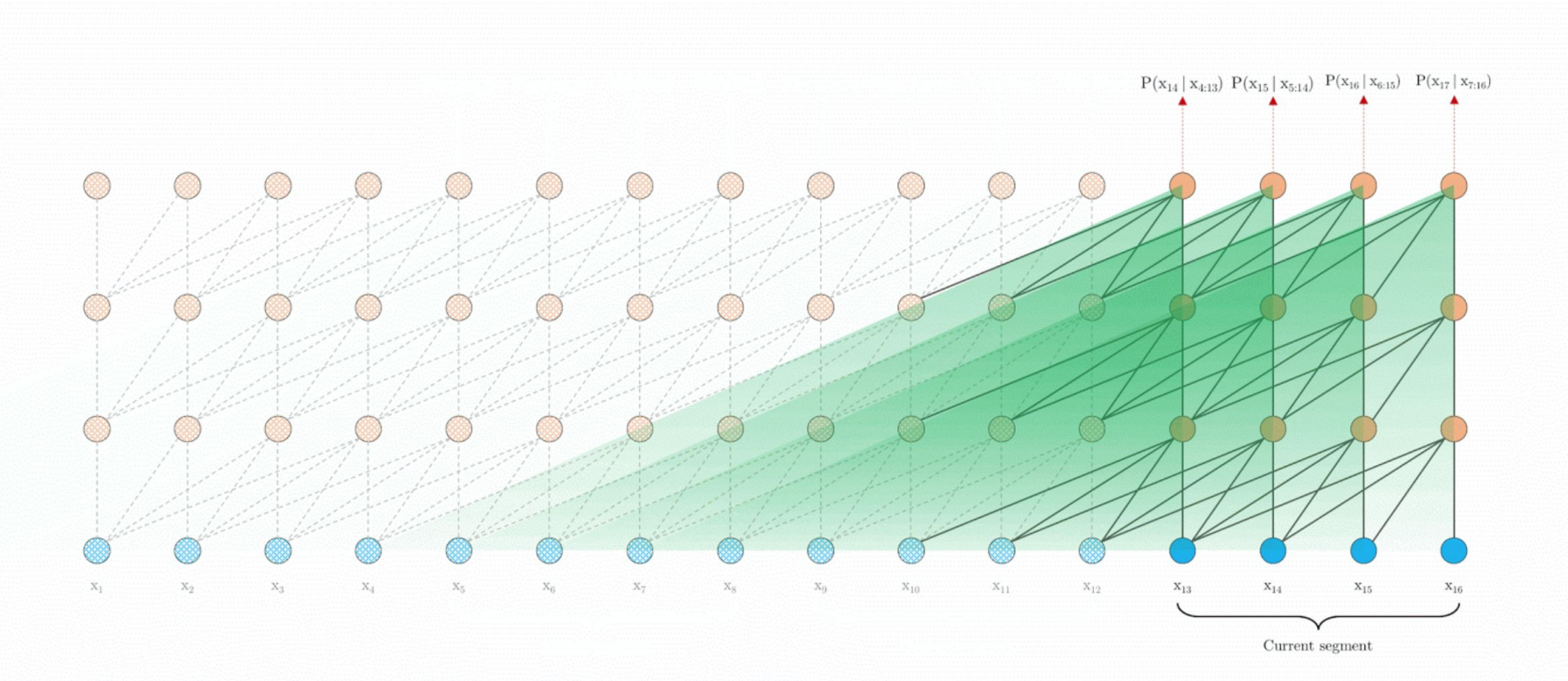
# Vanila Transformer vs. Transformer-XL




Transformer-XL with segment-level recurrence at evaluation time

https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html

# Vanila Transformer vs. Transformer-XL




Transformer-XL with segment-level recurrence at evaluation time

https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html

# Vanila Transformer vs. Transformer-XL

- Transformer-XL learns dependency that is about 80% longer than RNNs and 450% longer than vanilla Transformers

- Transformer-XL is up to 1,800+ times faster than a vanilla Transformer during evaluation on language modeling tasks, because no re-computation is needed

# XLNET

# BERT problems

- The [MASK] token used in training does not appear during fine-tuning

# BERT problems

- BERT generates predictions independently

I went to [MASK] [MASK] and saw the [MASK] [MASK] [MASK]

# BERT problems

- BERT generates predictions independently

I went to [MASK] [MASK] and saw the [MASK] [MASK] [MASK]

**Ground truth solutions:**

- I went to New York and saw the Empire State building.
- I went to San Francisco and saw the Golden Gate bridge.

# BERT problems

- BERT generates predictions independently

I went to [MASK] [MASK] and saw the [MASK] [MASK] [MASK]

**BERT solutions:**

- I went to New York and saw the Empire State building.
- I went to San Francisco and saw the Golden Gate bridge.
- I went to New York and saw the Golden Gate bridge.

# XLNET

- XLNET is a generalized autoregressive model
- Permutation language modeling (PLM)
- Integrates the idea of auto-regressive models and bi-directional context modeling
- Outperforms BERT on 20 tasks

# XLNET

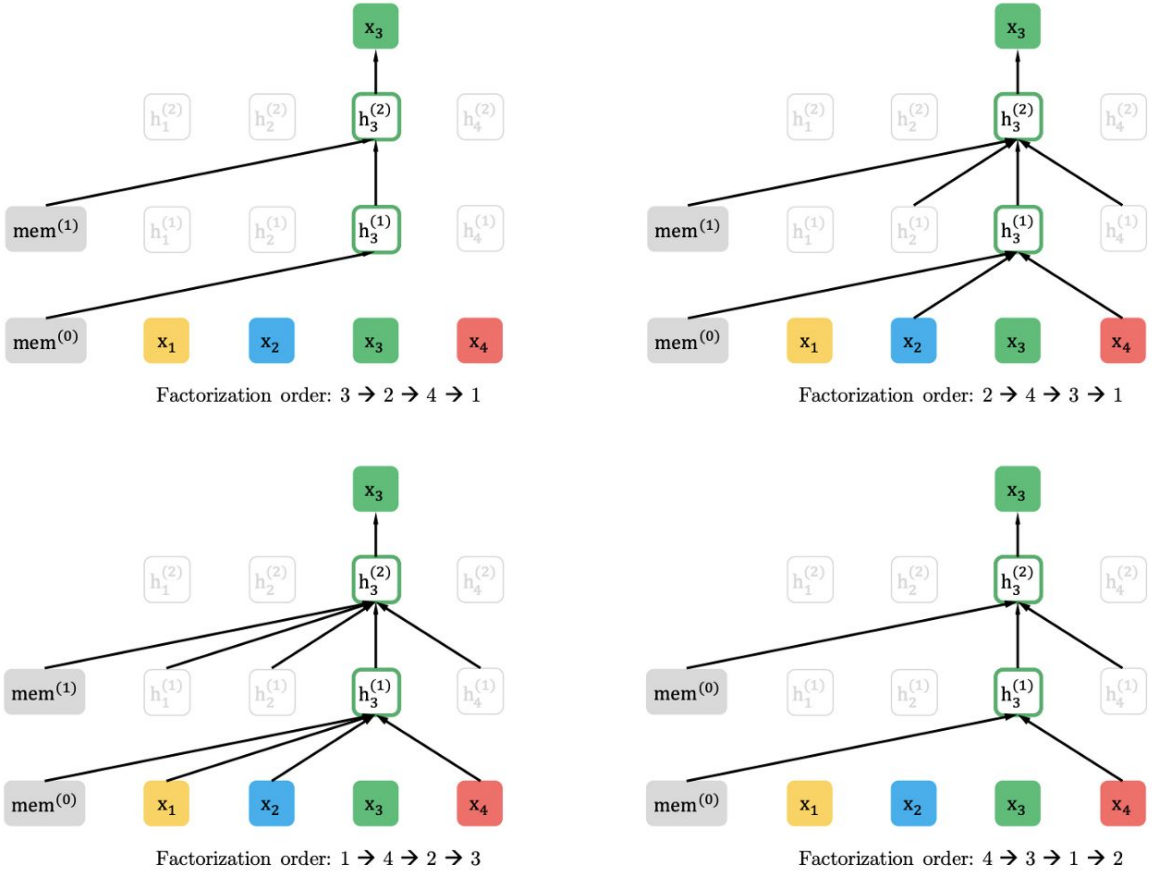
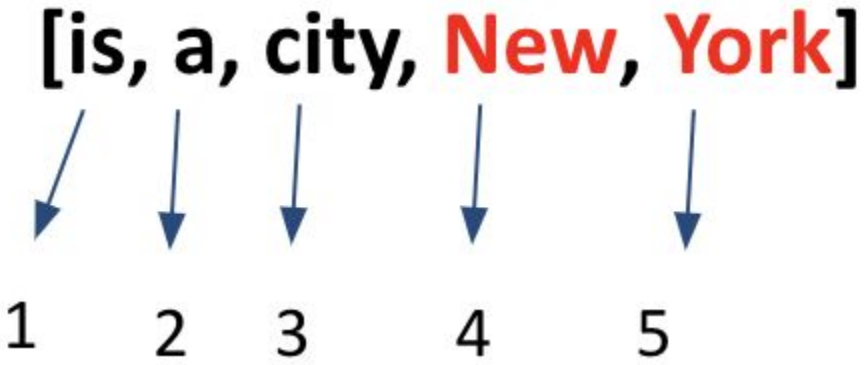



Figure 1: Illustration of the permutation language modeling objective for predicting $x_3$ given the same input sequence $\mathbf{x}$ but with different factorization orders.

https://towardsdatascience.com/xlnet-explained-in-simple-terms-255b9fb2c97c

# XLNET vs. BERT




BERT

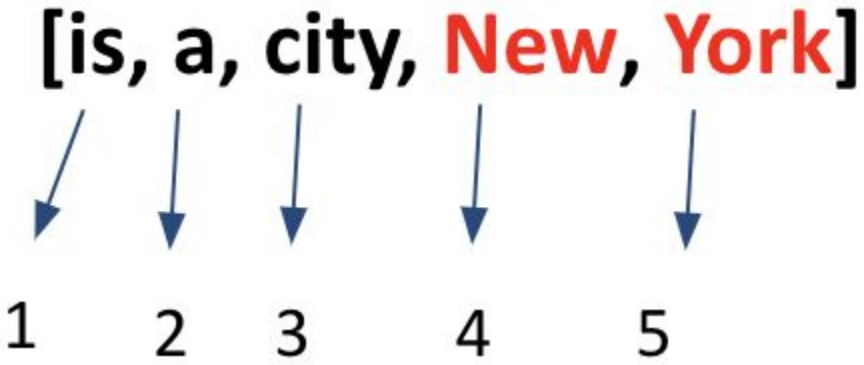
$$\log P(\text{New} \mid \textbf{is a city}) + \log P(\text{York} \mid \textbf{is a city})$$

XLNET

$$\log P(\text{New} \mid \textbf{is a city}) + \log P(\text{York} \mid \textbf{New, is a city})$$

https://towardsdatascience.com/xlnet-explained-in-simple-terms-255b9fb2c97c

# XLNET vs. BERT




BERT → $\log P(\text{New} \mid \textbf{is a city}) + \log P(\text{York} \mid \textbf{is a city})$

XLNET → $\log P(\text{New} \mid \textbf{is a city}) + \log P(\text{York} \mid \textbf{New, is a city})$

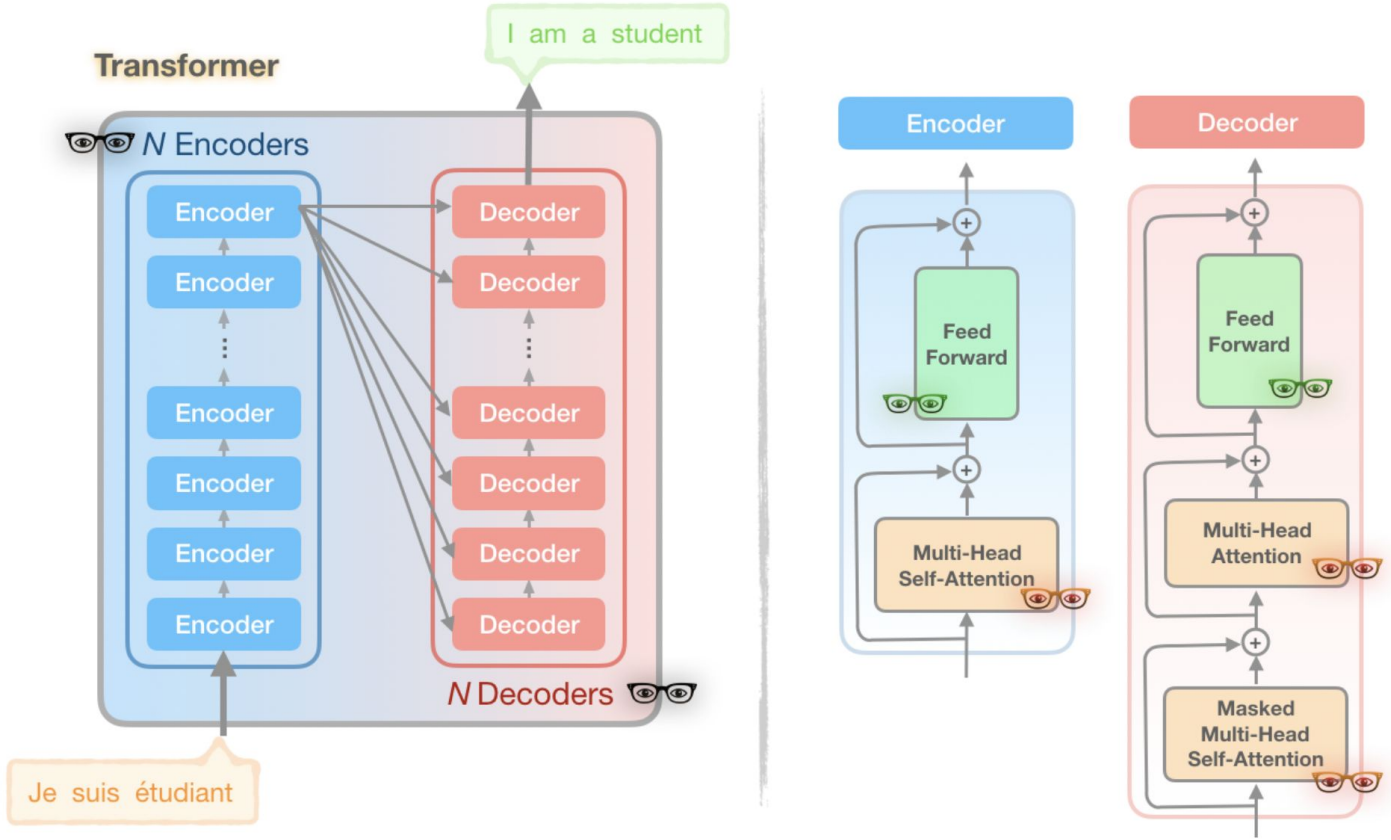
https://towardsdatascience.com/xlnet-explained-in-simple-terms-255b9fb2c97c

# The Reformer

# Reformer: The Effective Transformer (ICLR 2020)

Transformer-based models have a problem:

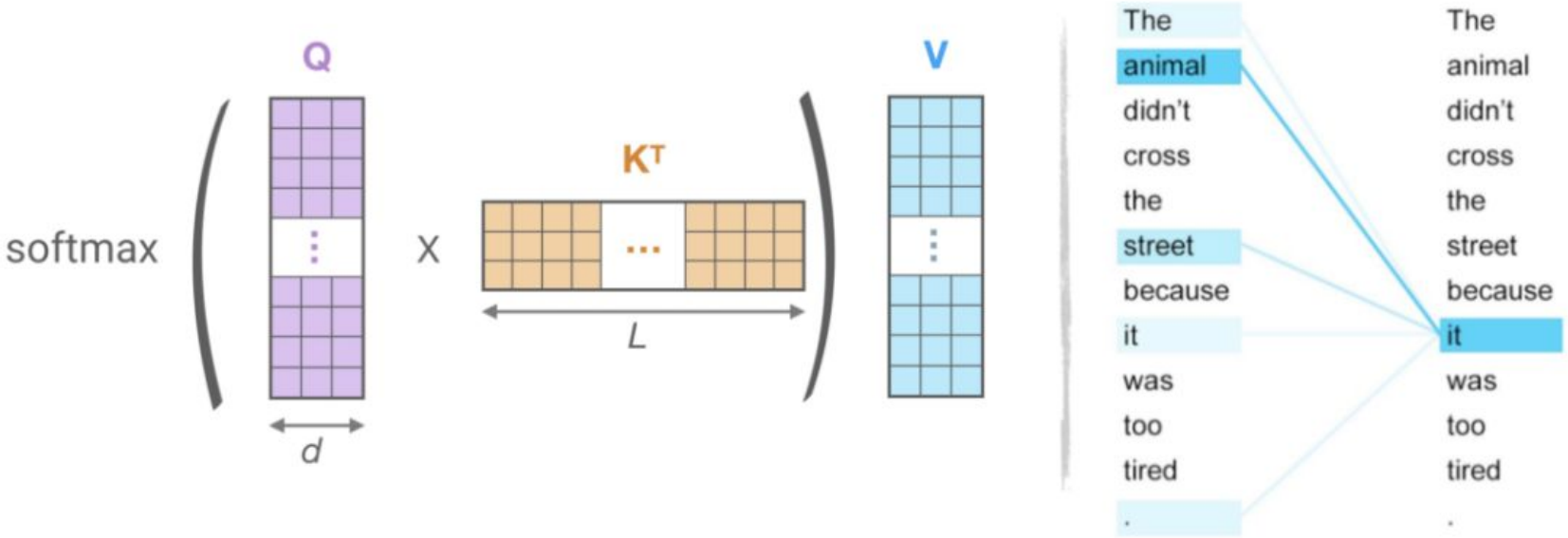
- They require lots of GPUs to train
  - even cannot be fine-tuned on a single GPU




- Problem 1 (**Red** 👓): Attention computation
- Problem 2 (**Black** 👓): Large number of layers
- Problem 3 (**Green** 👓): Depth of feed-forward layers

https://towardsdatascience.com/illustrating-the-reformer-393575ac6ba0

# Attention computation




- Replace dot-product attention with locality-sensitive hashing (LSH)
  - changes the complexity from $O(L^2)$ to $O(L \log L)$

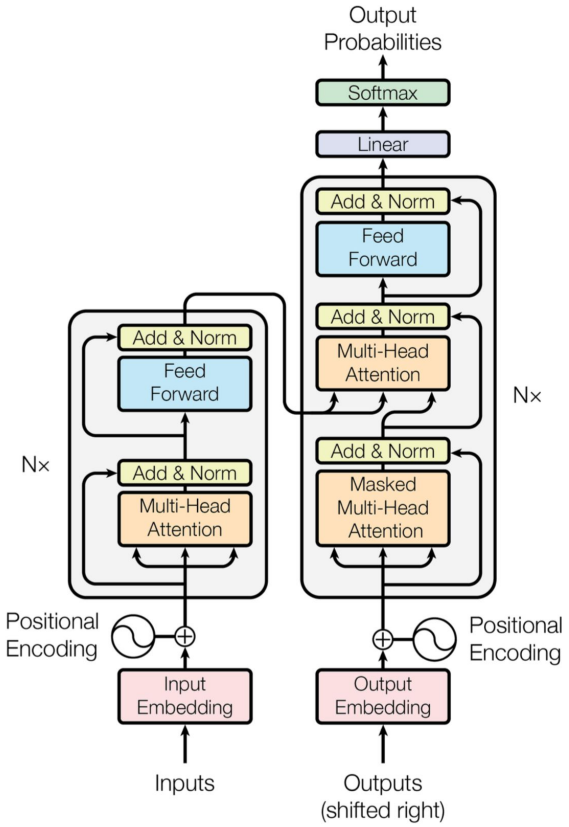
https://towardsdatascience.com/illustrating-the-reformer-393575ac6ba0

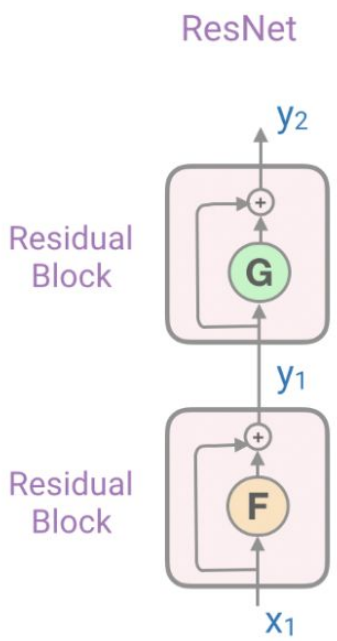
# Locality-sensitive hashing for attention computation

- LSH - an *efficient* and *approximate* way of nearest neighbors search in high dimensional datasets.

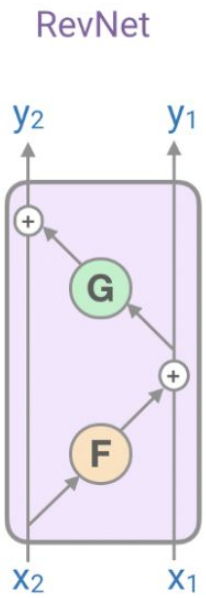
- The main idea behind LSH is to select hash functions such that for two points ‘p’ and ‘q’, if ‘q’ is close to ‘p’ then with good enough probability we have ‘hash(q) == hash(p)’.

# Reversible Transformer

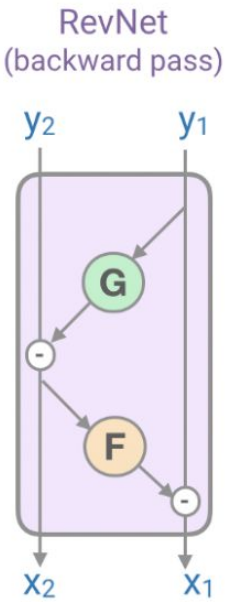



$$y_1 = x_1 + F(x_1)$$
$$y_2 = y_1 + G(y_1)$$



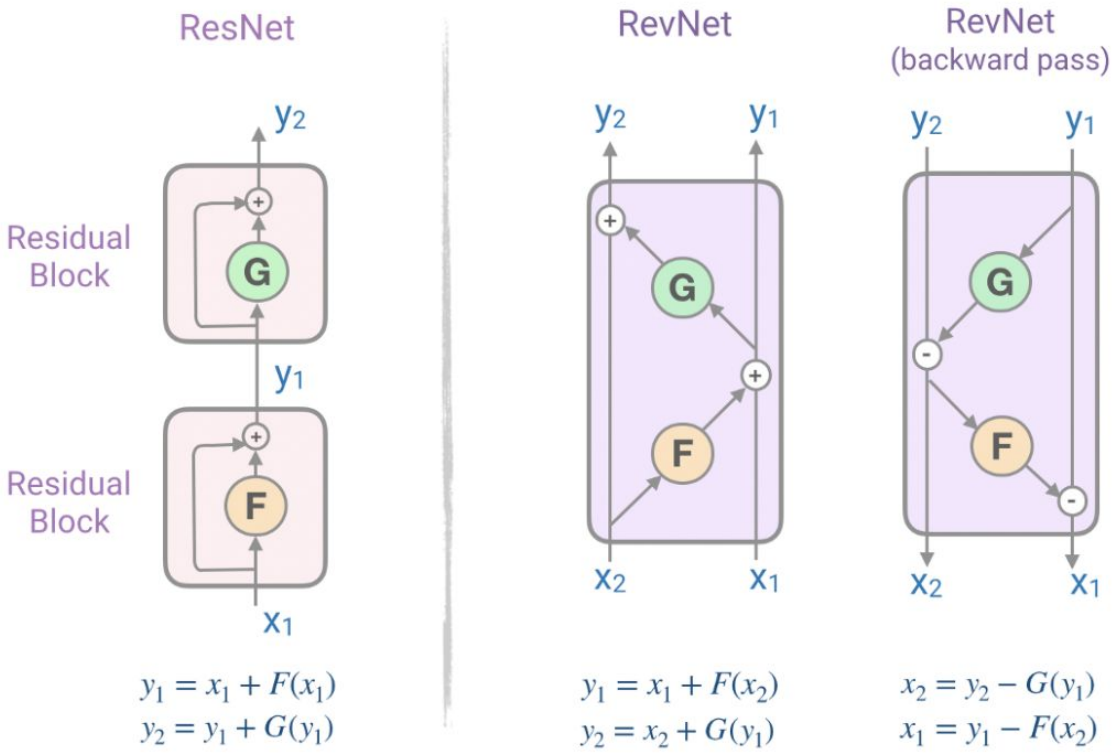

$$y_1 = x_1 + F(x_2)$$
$$y_2 = x_2 + G(y_1)$$




$$x_2 = y_2 - G(y_1)$$
$$x_1 = y_1 - F(x_2)$$

https://towardsdatascience.com/illustrating-the-reformer-393575ac6ba0

# Reversible Transformer




- F - self-attention block
- G - feed-forward layer

Profit: storing activations only once during the training process

https://towardsdatascience.com/illustrating-the-reformer-393575ac6ba0

# Chunking

Computations in feed-forward layers are independent across positions in a sequence => the computations for the forward and backward passes can be split into chunks.

$$Y_2 = \left[Y_2^{(1)}; \ldots; Y_2^{(c)}\right] = \left[X_2^{(1)} + \text{FeedForward}(Y_1^{(1)}); \ldots; X_2^{(c)} + \text{FeedForward}(Y_1^{(c)})\right]$$

Chunking in the forward pass computation [Image is taken from the Reformer paper]

https://towardsdatascience.com/illustrating-the-reformer-393575ac6ba0

# References

- Transformer
- OpenAI Transformer
- ELMO
- BERT
- BERTology
- GPT
- GPT-2
- GPT-3
- Transformer XL
- XLNET
- Reformer