

Transformers

Attention is All You Need

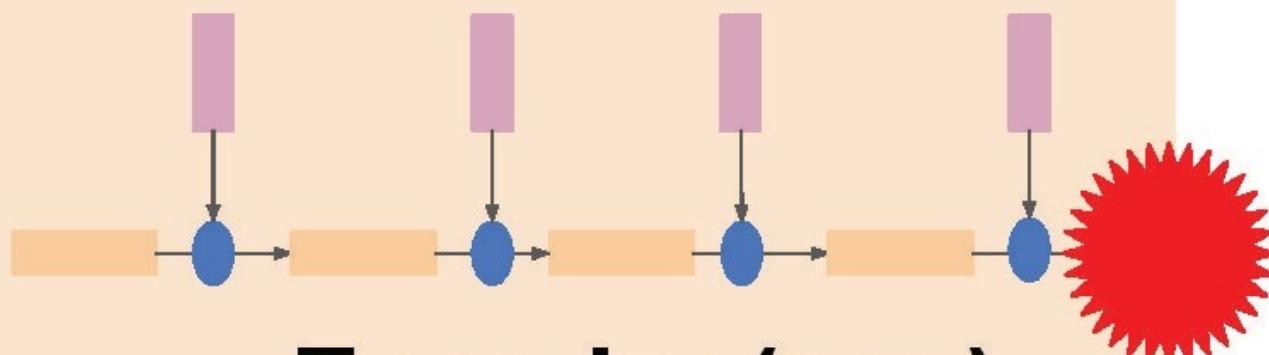
Ashish Vaswani et. al., NeurIPS 2017

<https://papers.nips.cc/paper/2017/file/3f5ee243547dee91fb053c1c4a845aa-Paper.pdf>

The information bottleneck and latent structure

We are trying to encode *variable-length* structure (e.g., variable-length sentences) in a fixed-length memory (e.g., only the 300 dimensions of your hidden state.)

Only use neural nets



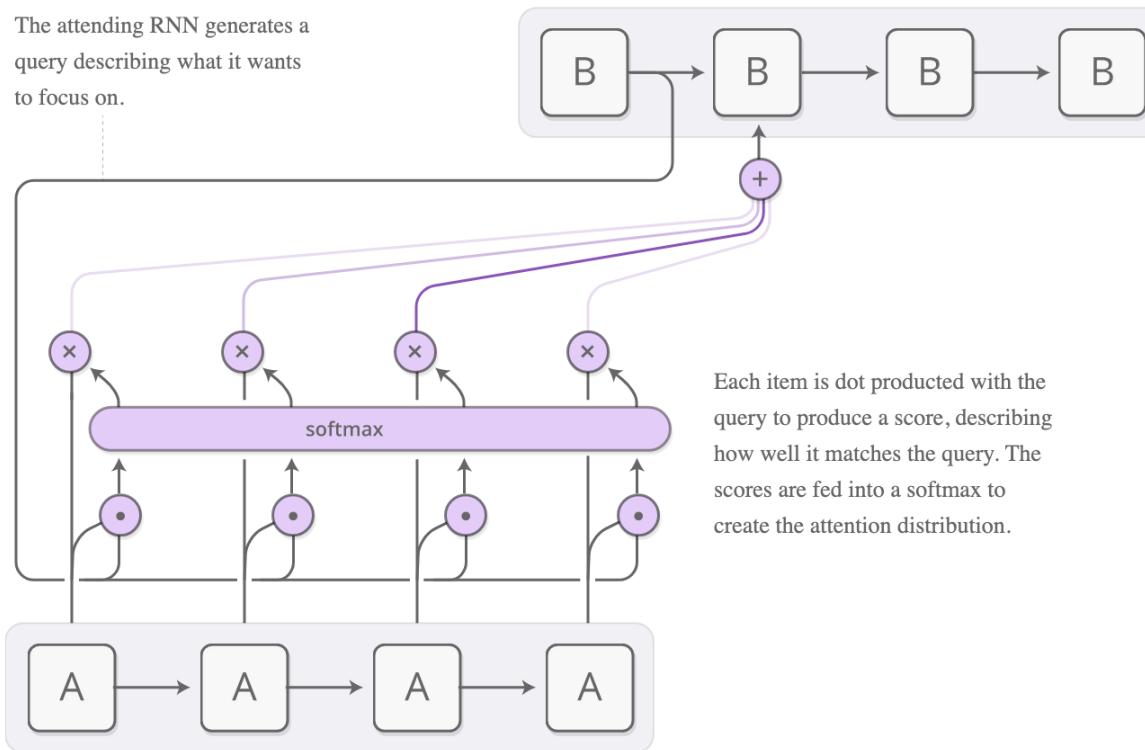
Encoder (seq)

The last encoder hidden state is the bottleneck -- all information in the source sentence must pass through it to get to the decoder.

Finding a solution to this problem was the final advance that made neural MT competitive with previous approaches.

Attentional Interfaces

- When translating, pay attention to the word currently translated
- Attention == focusing on part of a subset of the given information



Attentional Interfaces

- One use of attention is translation
 - Traditional seq2seq model boils entire input into a single vector and then expands it back out
 - Attention allows RNN processing the input to pass along information about *each* word it sees. The RNN generating the output focuses on words as they become relevant

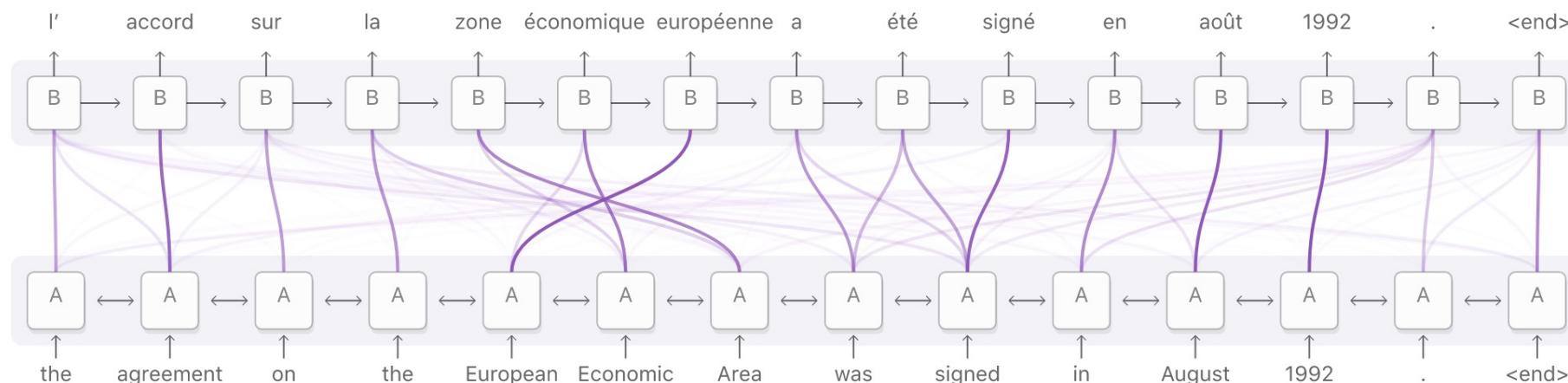


Diagram derived from Fig. 3 of Bahdanau, et al. 2014

Neural Machine Translation (NMT)

In two years, neural networks surpassed everything that had appeared in the past 20 years of translation. Neural translation contains 50% fewer word order mistakes, 17% fewer lexical mistakes, and 19% fewer grammar mistakes. The neural networks even learned to harmonize gender and case in different languages. And no one taught them to do so.

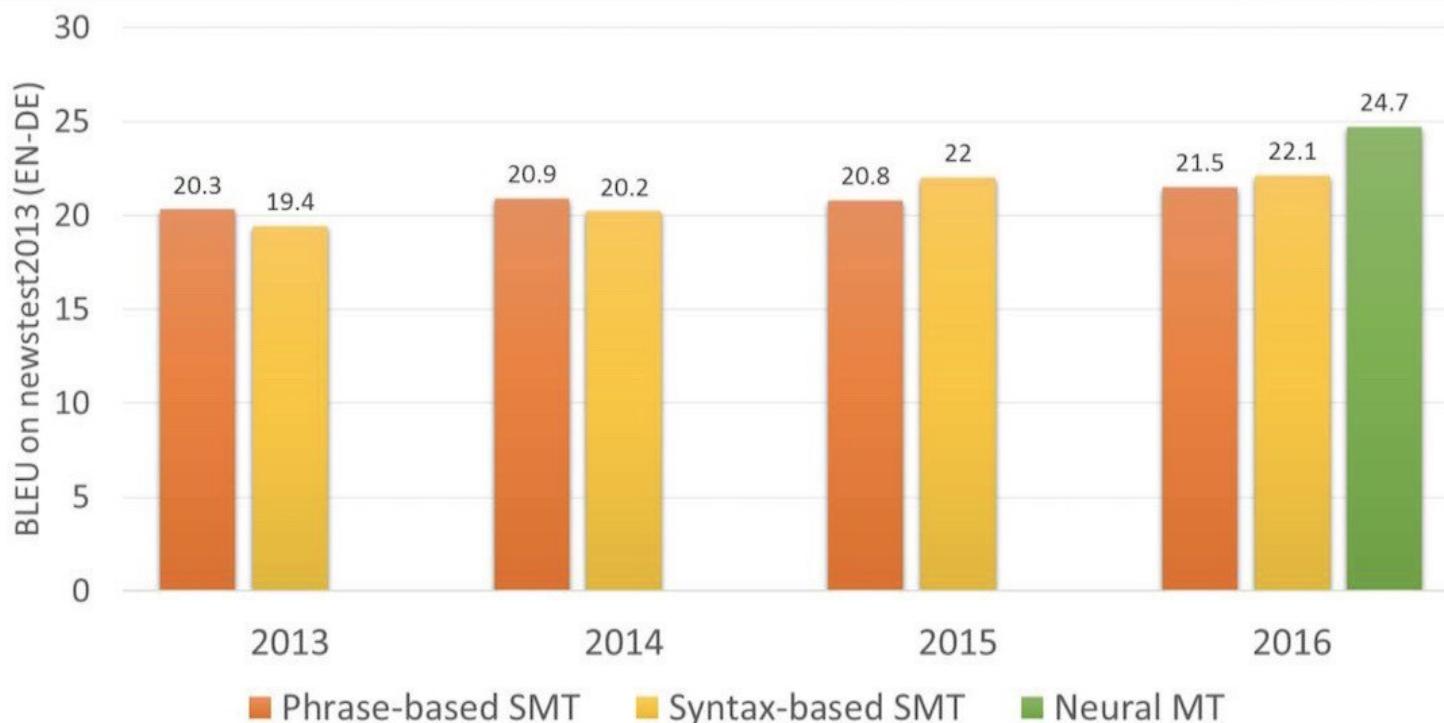


Figure from "A history of machine translation from the Cold War to deep learning" by Ilya Pestov

Attentional Interfaces

- Is also used in image captioning
 - On the interface between a CNN and an RNN
 - RNN will look at different position of an image as it generates each word
- Attention can be used whenever you want to interface with a NN that has a repeating structure in its output



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.

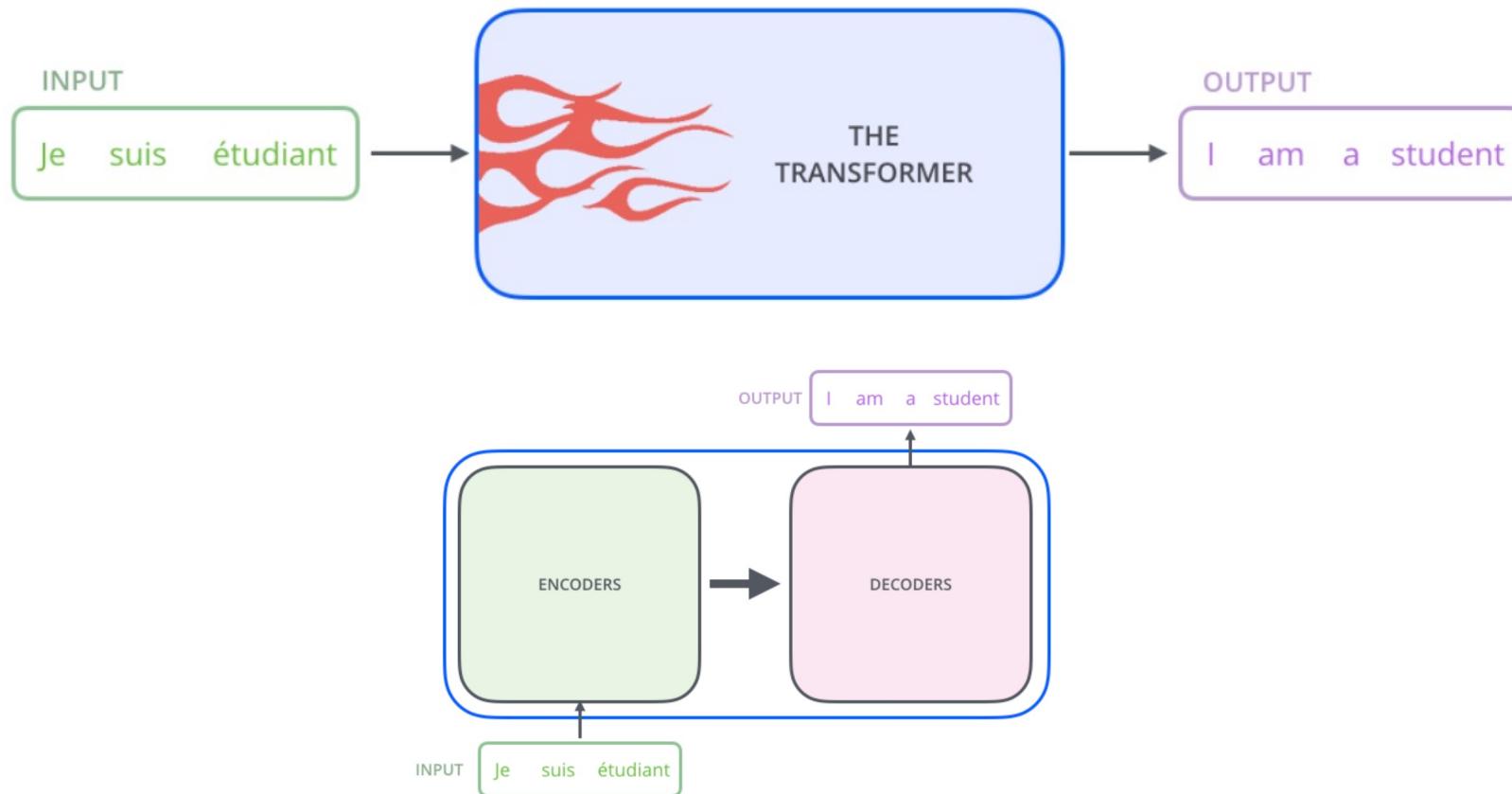


A stop sign is on a road with a mountain in the background.

Attention is all You Need

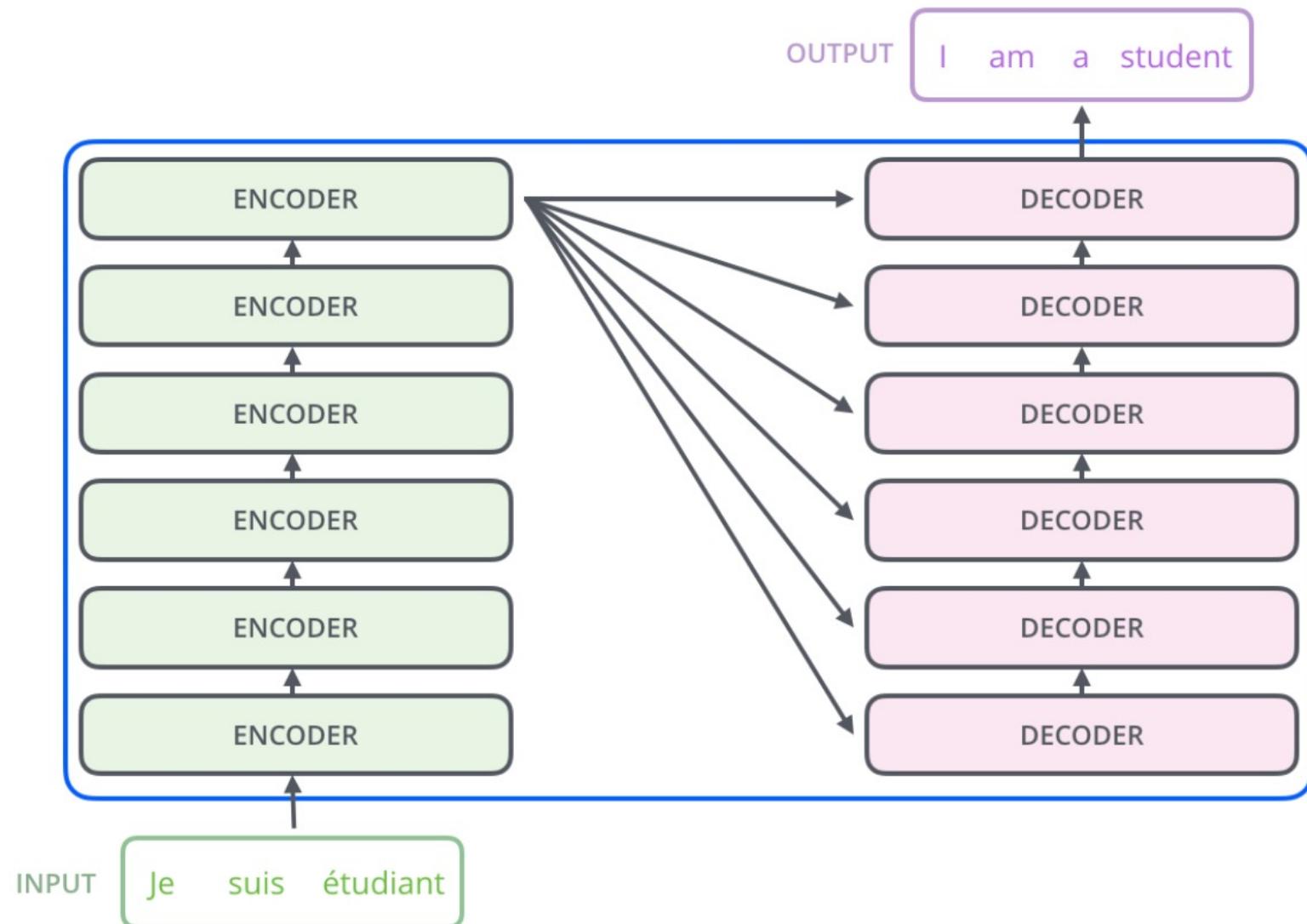
<https://jalammar.github.io/illustrated-transformer/>

- Improve NMT



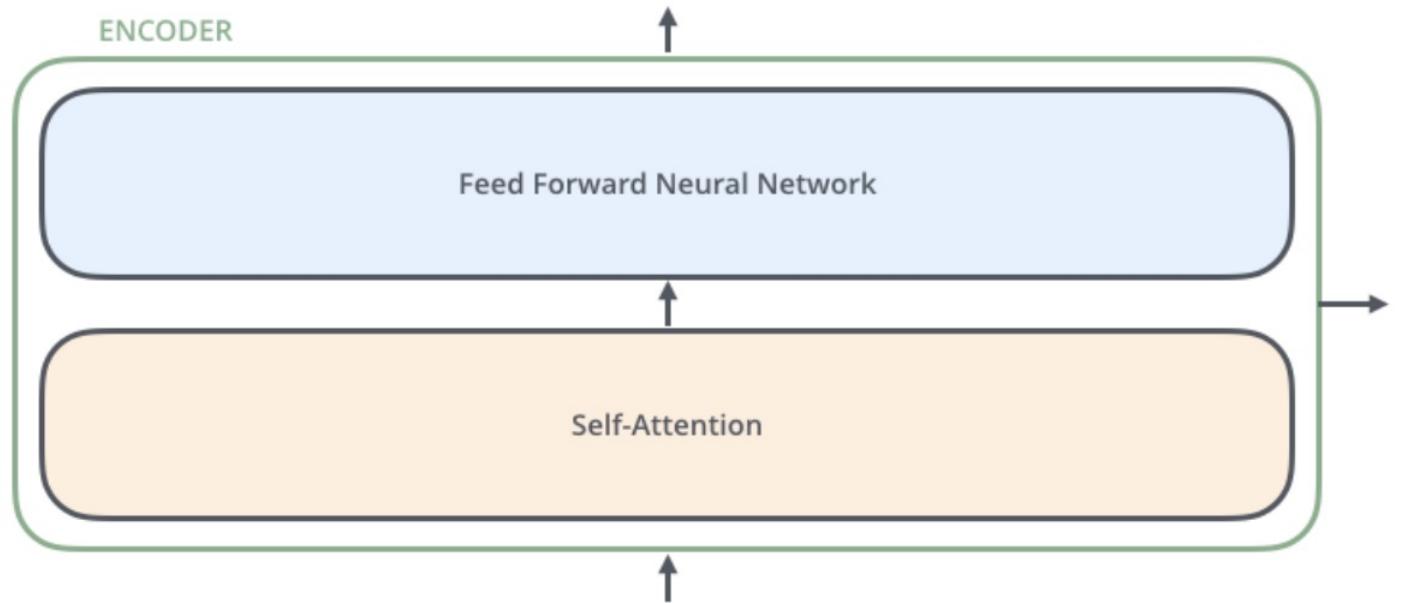
Attention is all You Need

- Transformer
 - Stack of Encoders
 - Stack of Decoders



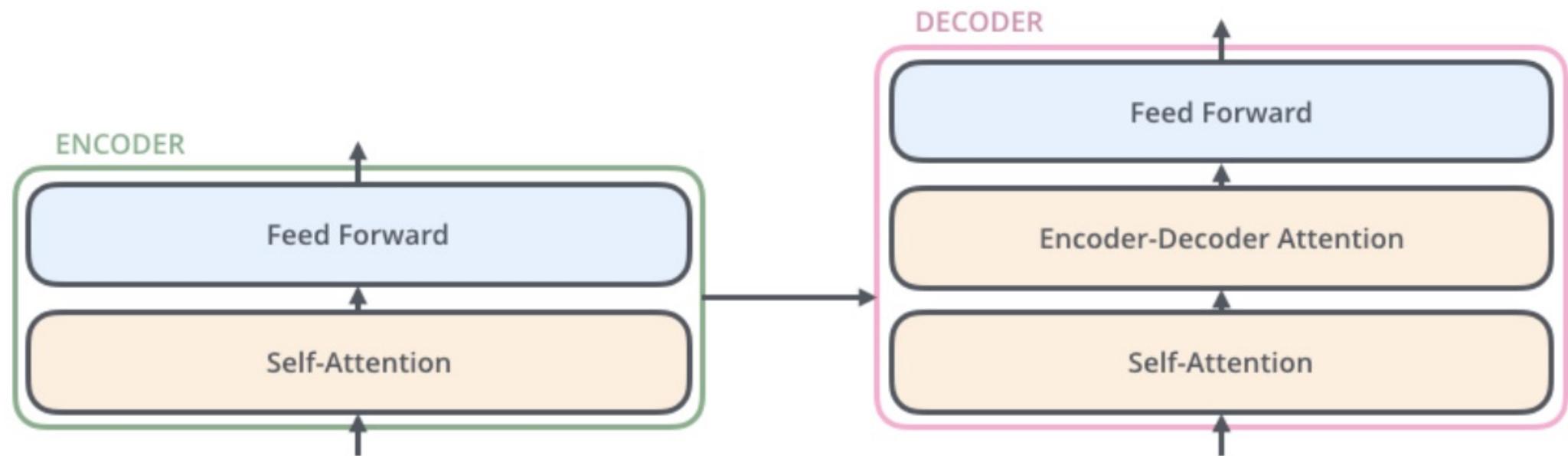
Attention is all You Need

- Each encoder:
 - Self-attention: look at other words in the input as it encodes a specific word
 - FFNN is independently applied to each position



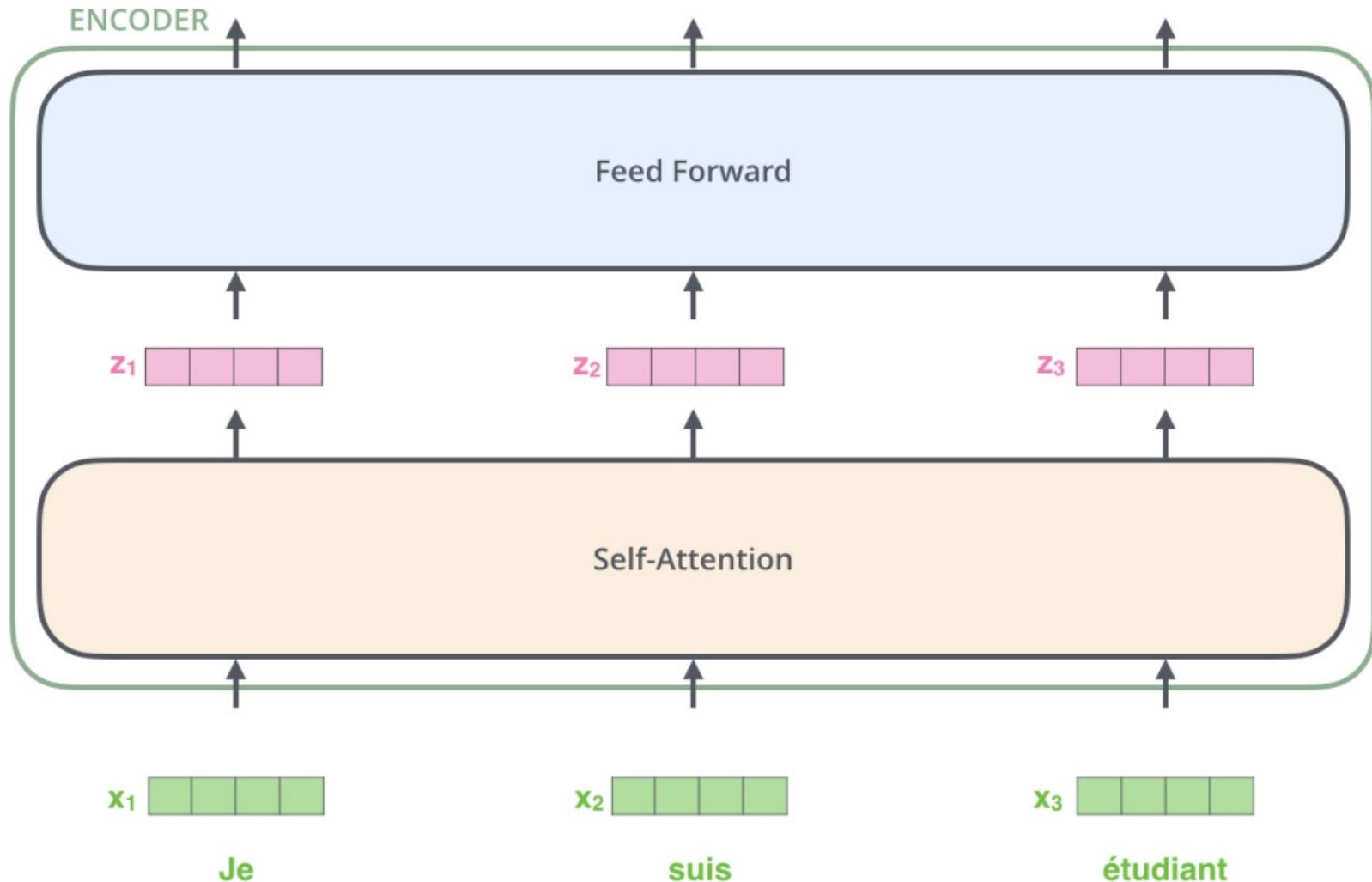
Attention is all You Need

- Each decoder:
 - Has attention layer that lets it focus on relevant parts of the input sentence



Attention is all You Need

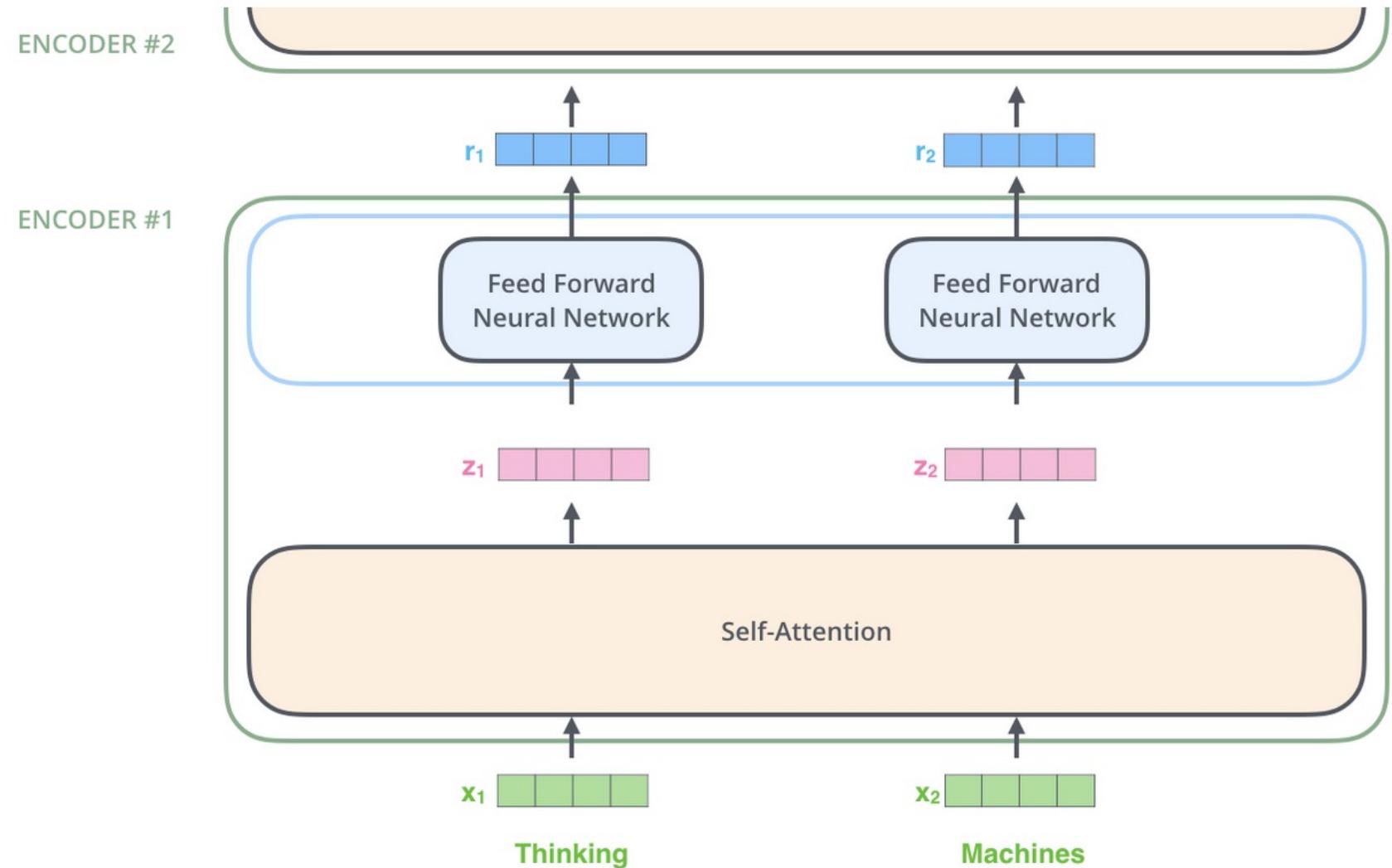
- Input:
 - The encoder receives a list of vectors as input



Attention is all You Need

- **Input:**

- The encoder receives a list of vectors as input
- Word in each position flows through its own path in the encoder



Attention is all You Need

- Self attention:

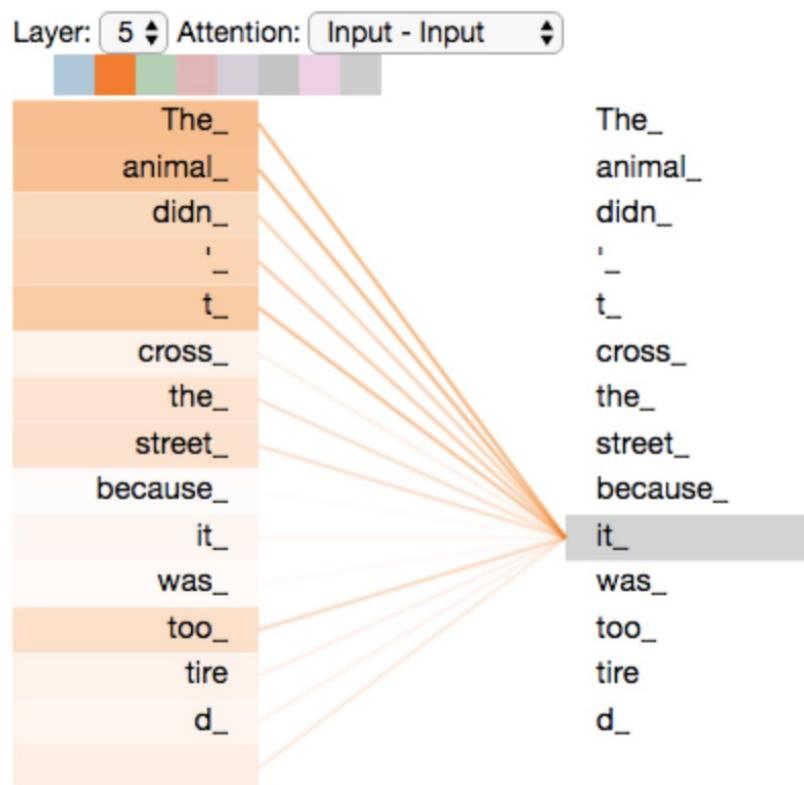
“The animal didn't cross the street because it was too tired”

allows to associate “it” with “animal” by allowing the model to look at other positions in the input sentence for clues that can help lead to a better encoding of the word “it”

Attention is all You Need

- Self attention:

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

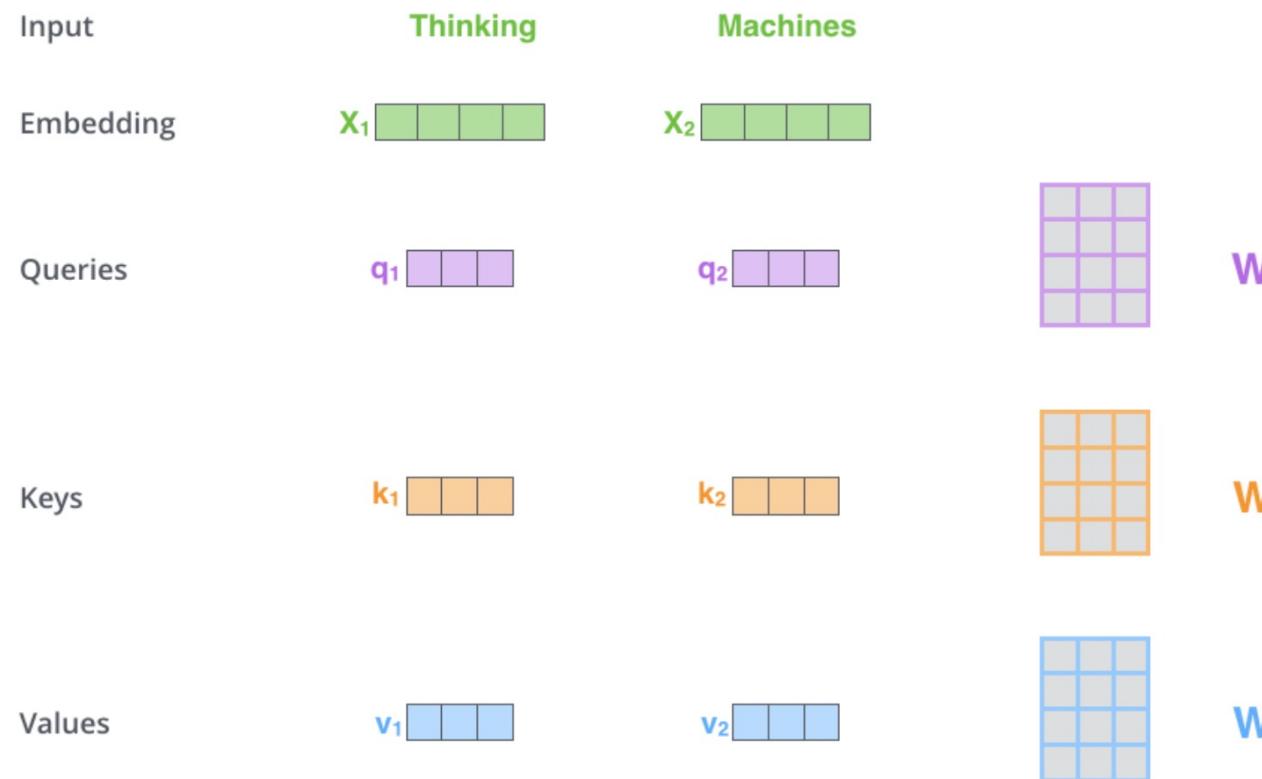


Attention is all You Need

- Calculating self attention:

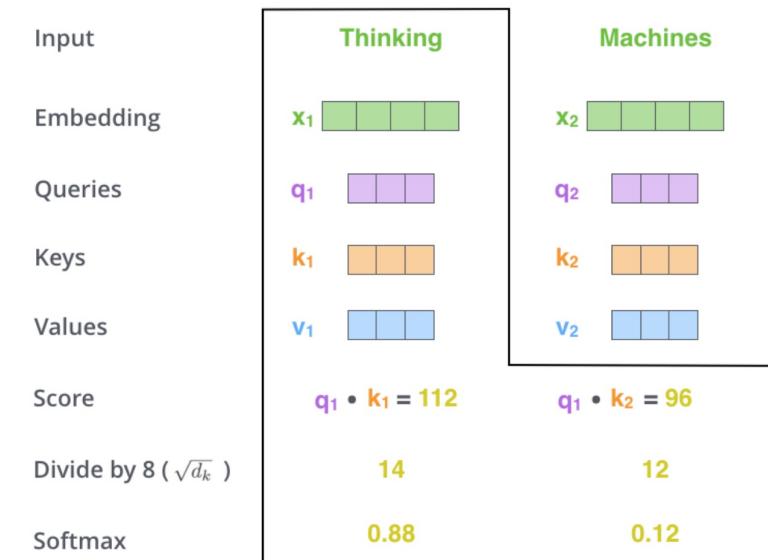
1. Create three vectors from each of the encoder's input vectors:

- Query key
- Key vector
- Value vector



Attention is all You Need

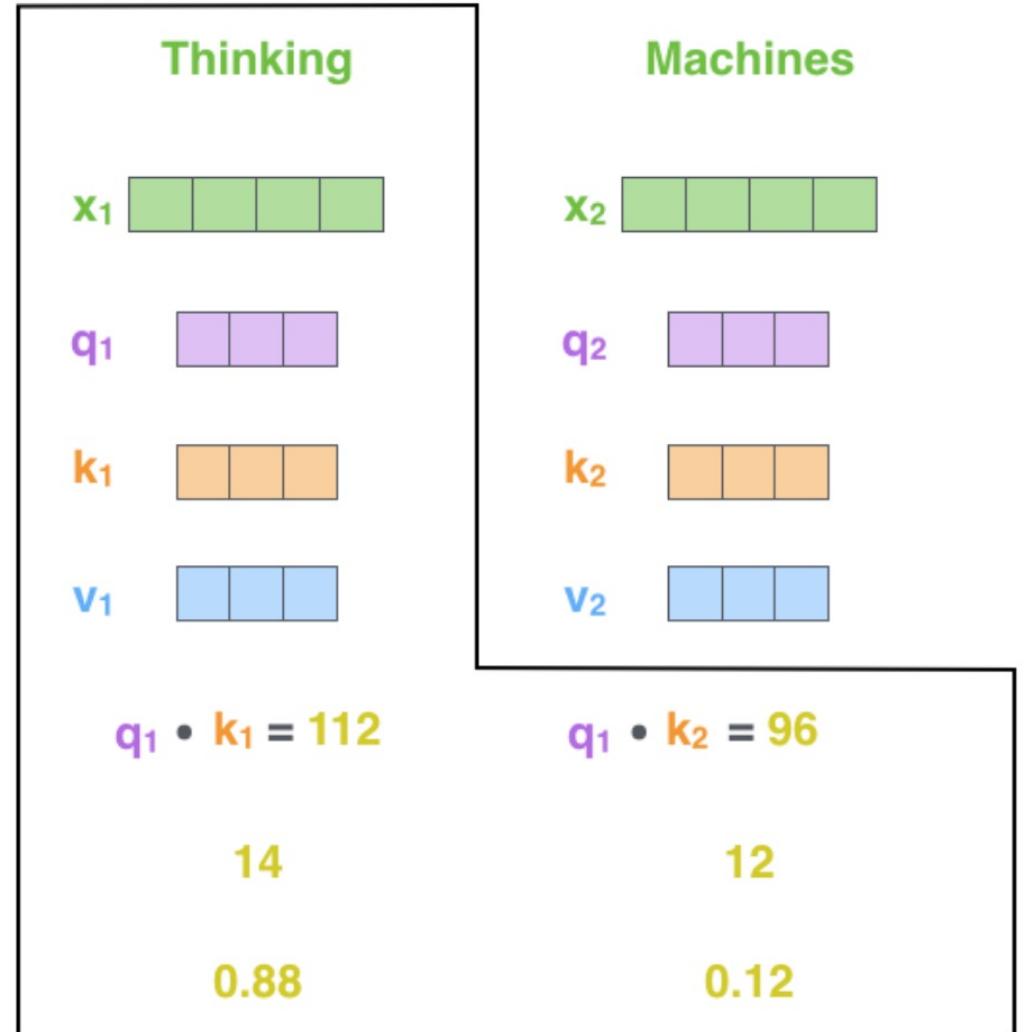
- Calculating self attention:
 1. Create three vectors from each of the encoder's input vectors:
 2. Use the vectors to compute the attention score
 - Score each word of the input sentence against the word we're computing attention for
 - The score determines how much focus to place on other parts of the input sentence as we encode a word at a certain position



Attention is all You Need

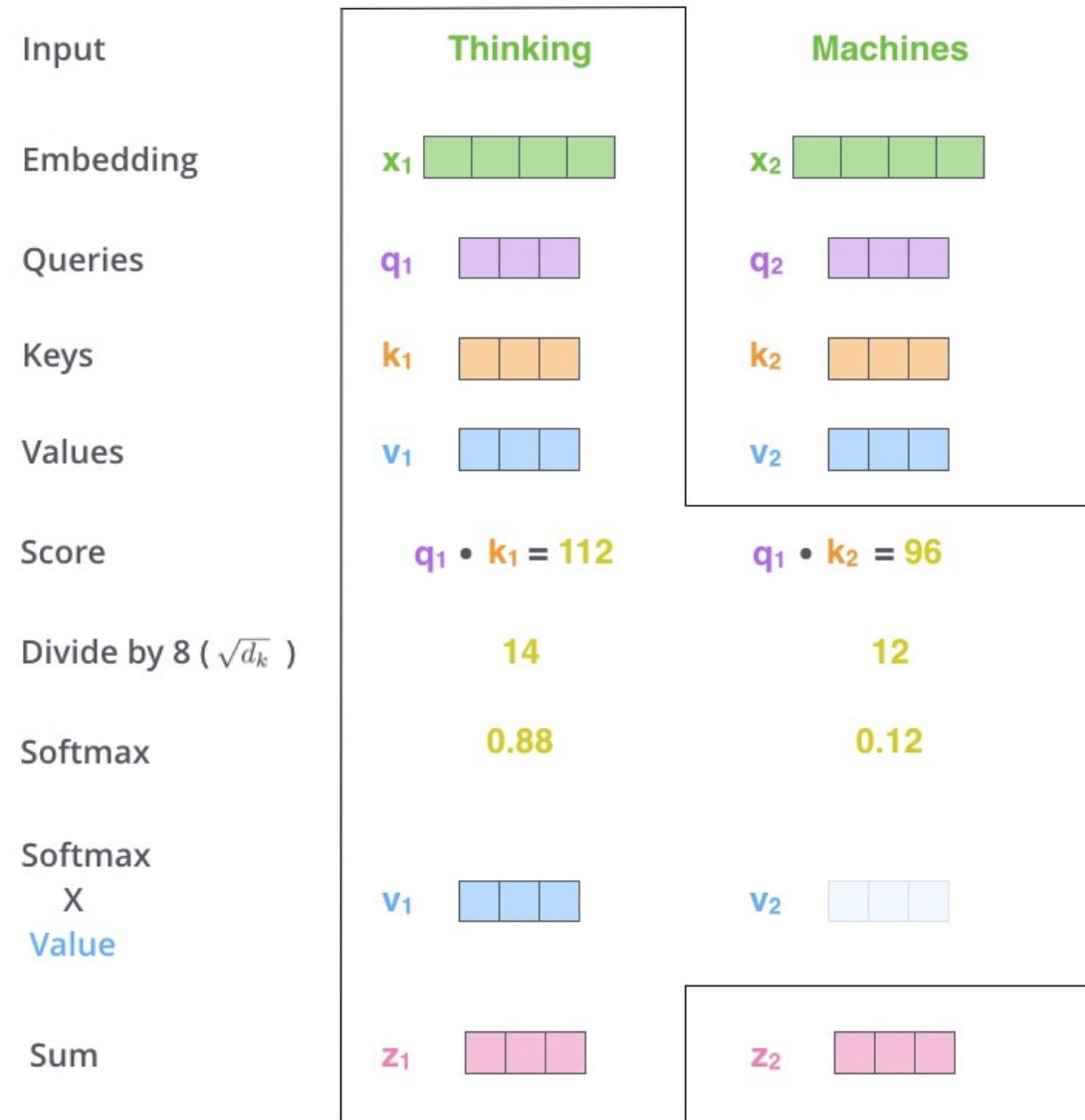
- Calculating self attention:
 1. Create three vectors from each of the encoder's input vectors
 2. Use the vectors to compute the attention score

Input	Embedding
	Queries
	Keys
	Values
	Score
	Divide by 8 ($\sqrt{d_k}$)
	Softmax



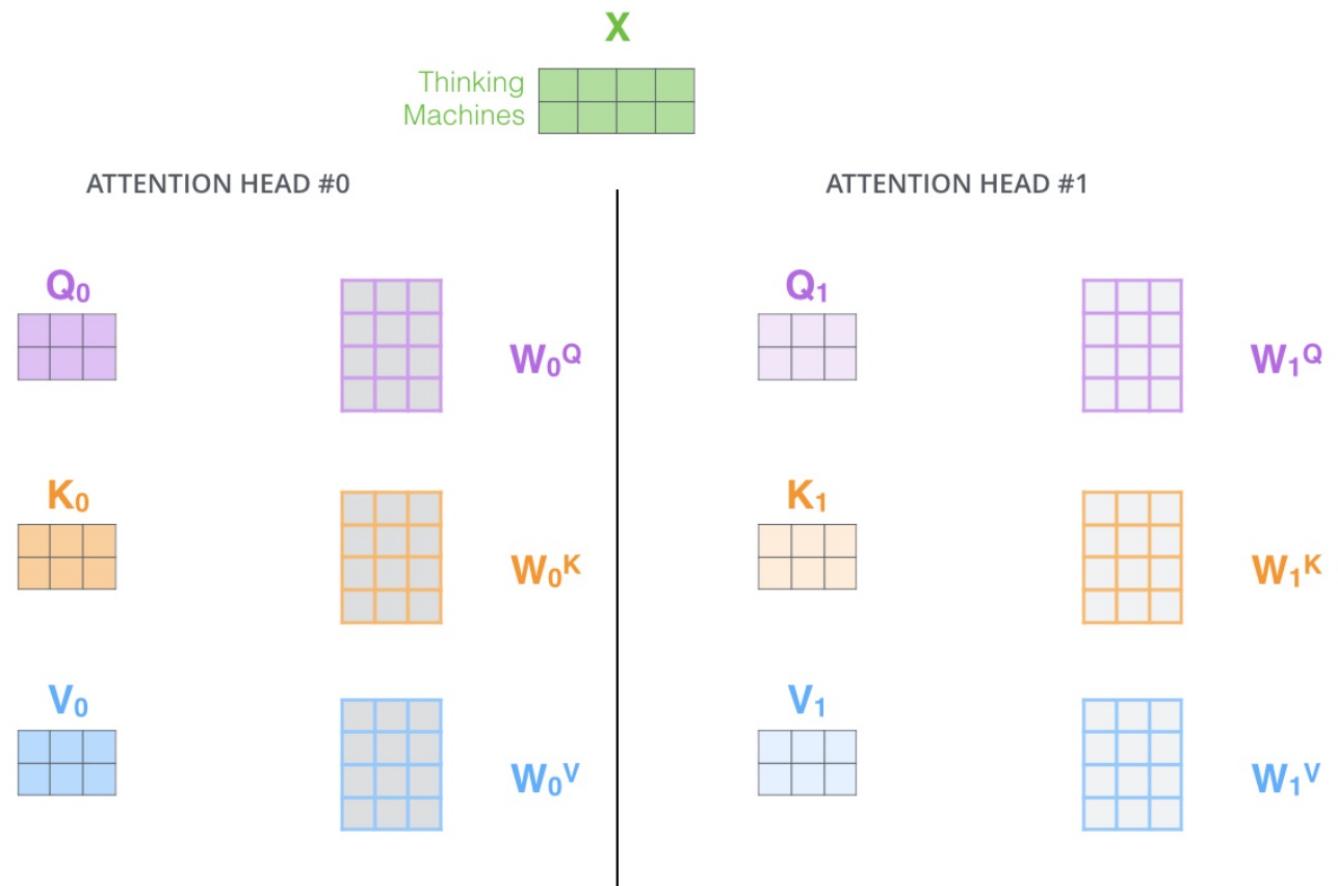
Attention is all You Need

- Calculating self attention:
 1. Create three vectors from each of the encoder's input vectors
 2. Use the vectors to compute the attention score
 3. Multiply each value vector by the softmax score and sum up the weighted value vectors to produce output of the self attention layer for this word



Attention is all You Need

- There are several attention heads on each layer
 - Gives the attention layer multiple "representation subspaces"



Attention is all You Need

- Encoder summary

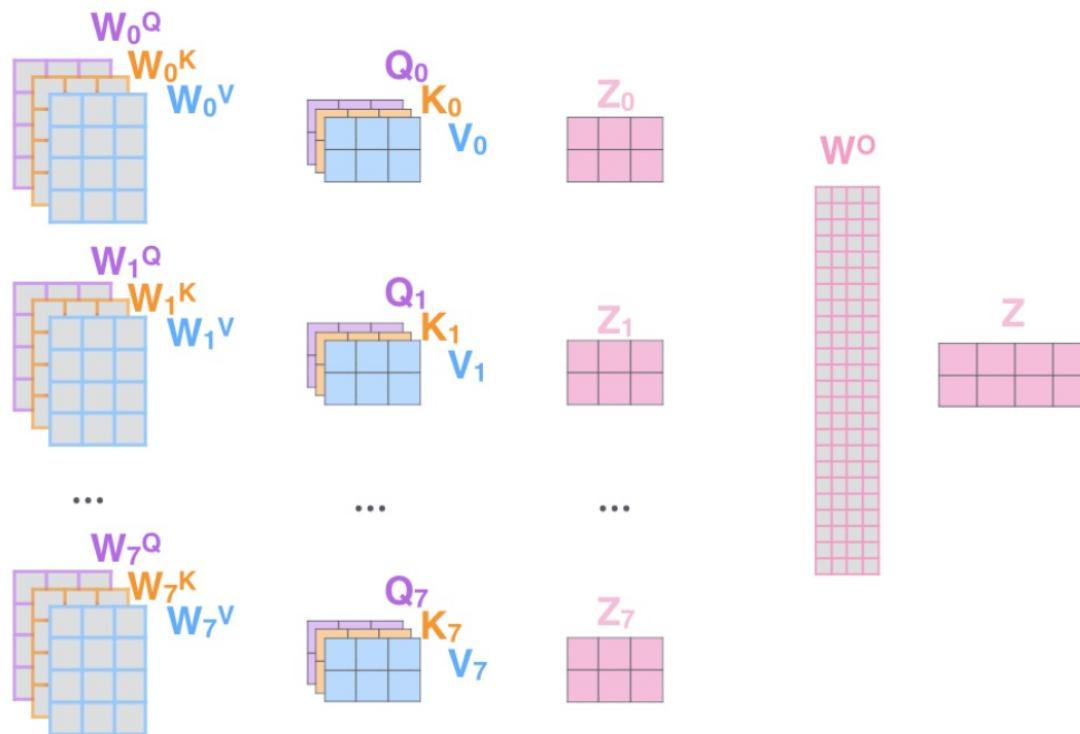
1) This is our
input sentence*

Thinking
Machines

2) We embed
each word*



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



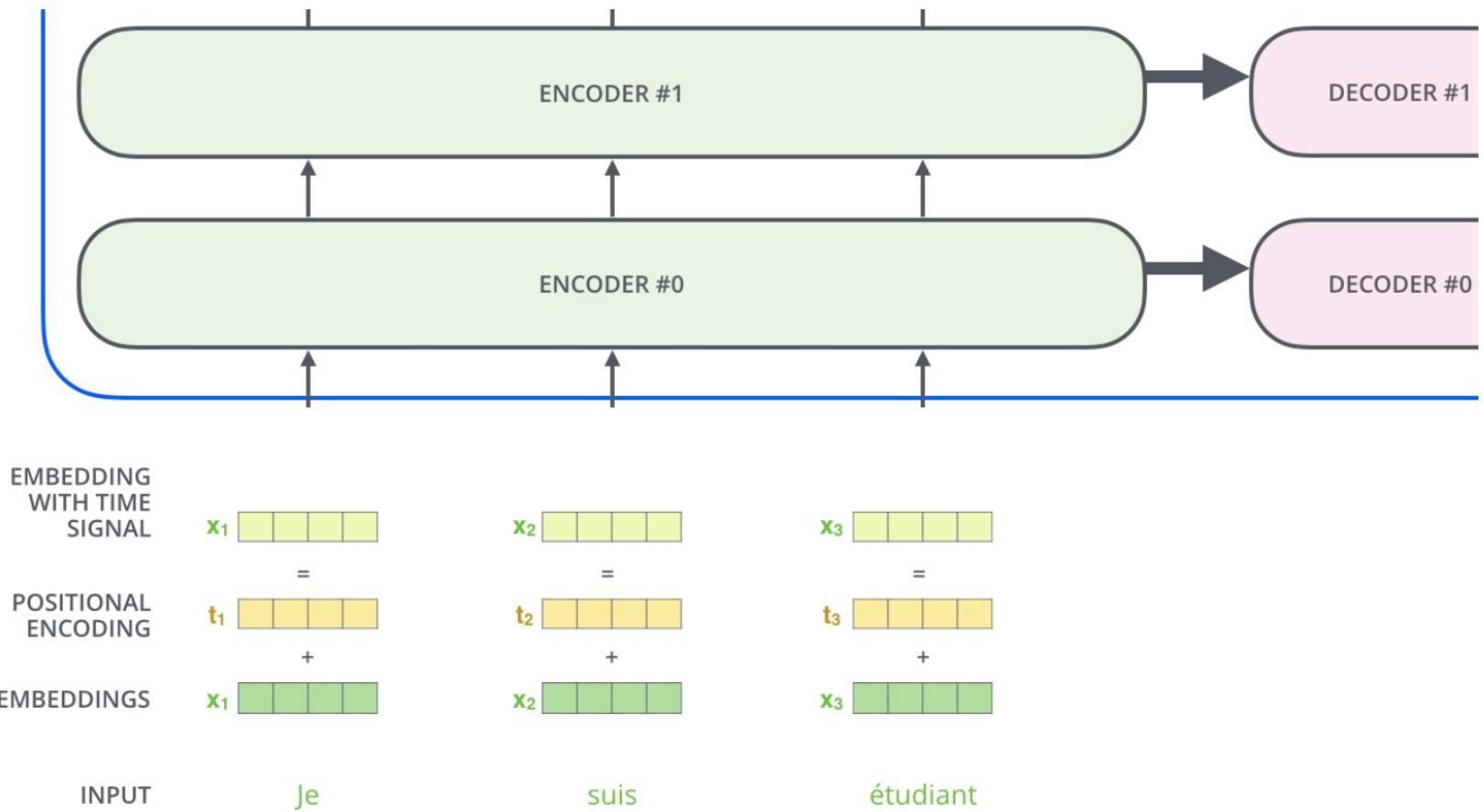
* In all encoders other than #0,
we don't need embedding.
We start directly with the output
of the encoder right below this one

Attention is all You Need

- Positional encoding:
 - To account for the order of the words in the input sequence
 - Add a vector to each input embedding, the vector follows a specific pattern that helps the model determine the position of each word/distance between words in the sequence

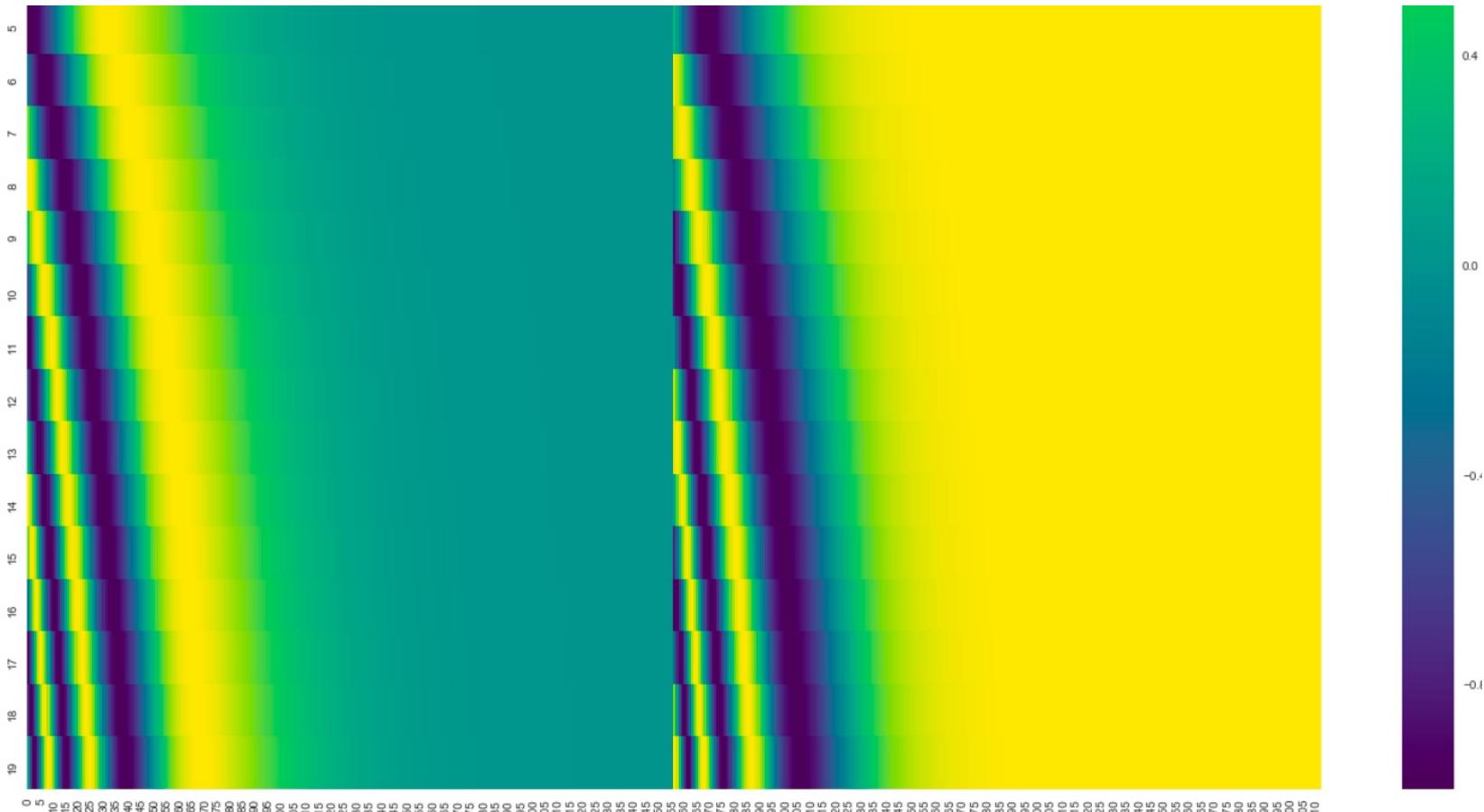
Attention is all You Need

- Positional encoding:



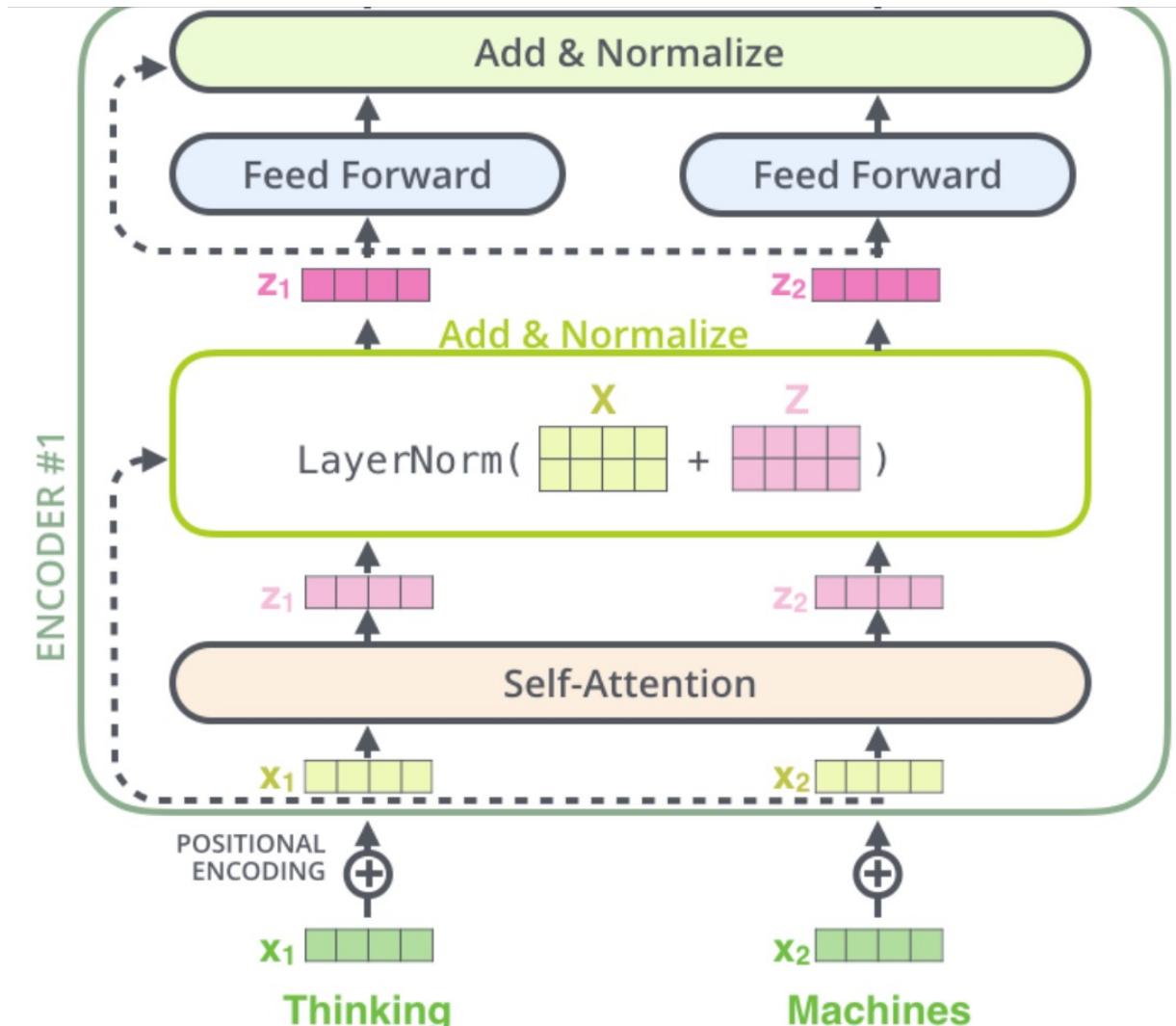
Attention is all You Need

- Positional encoding:



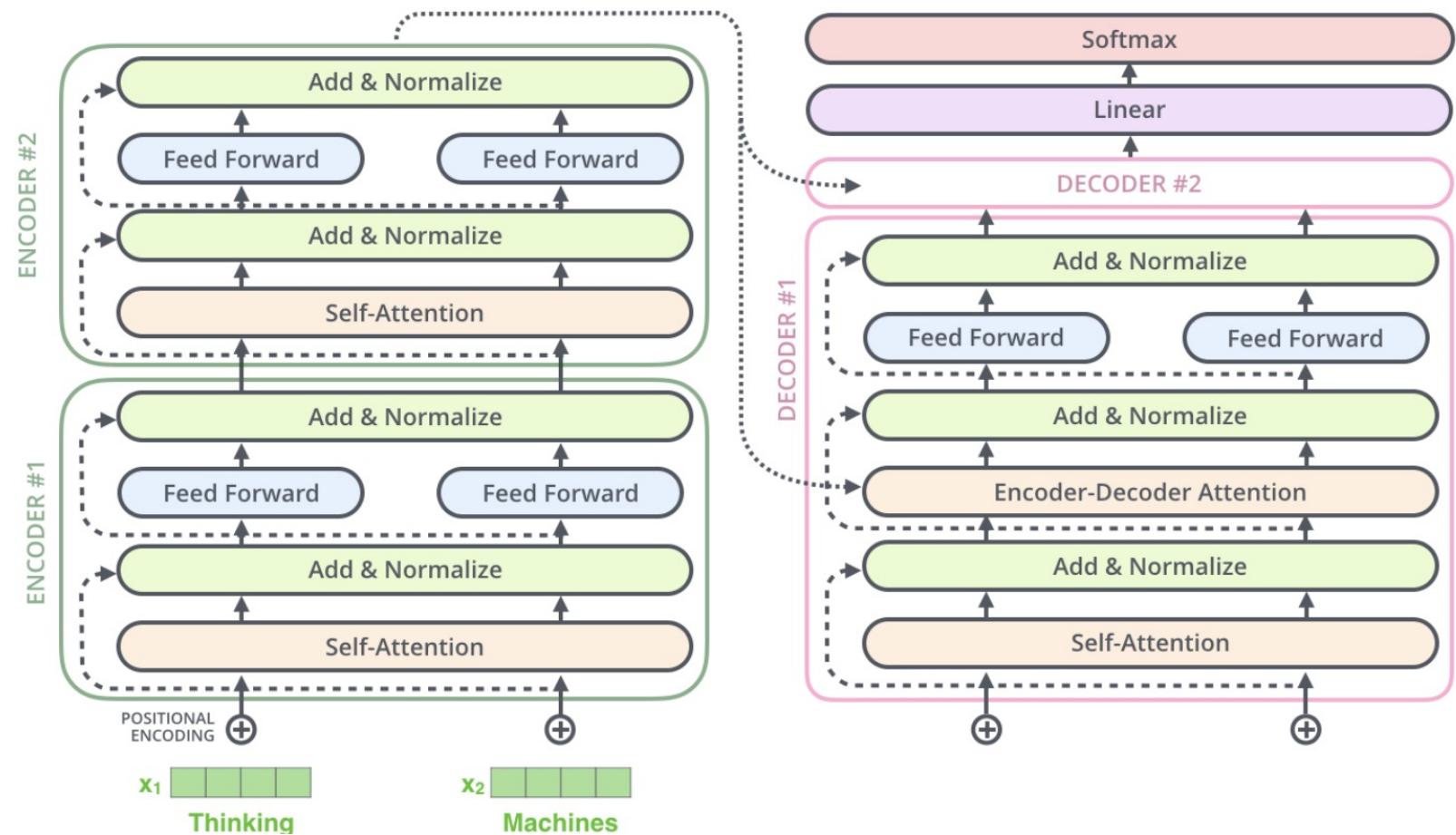
Residuals

- Each encoder has a residual connection followed by a layer-normalization step

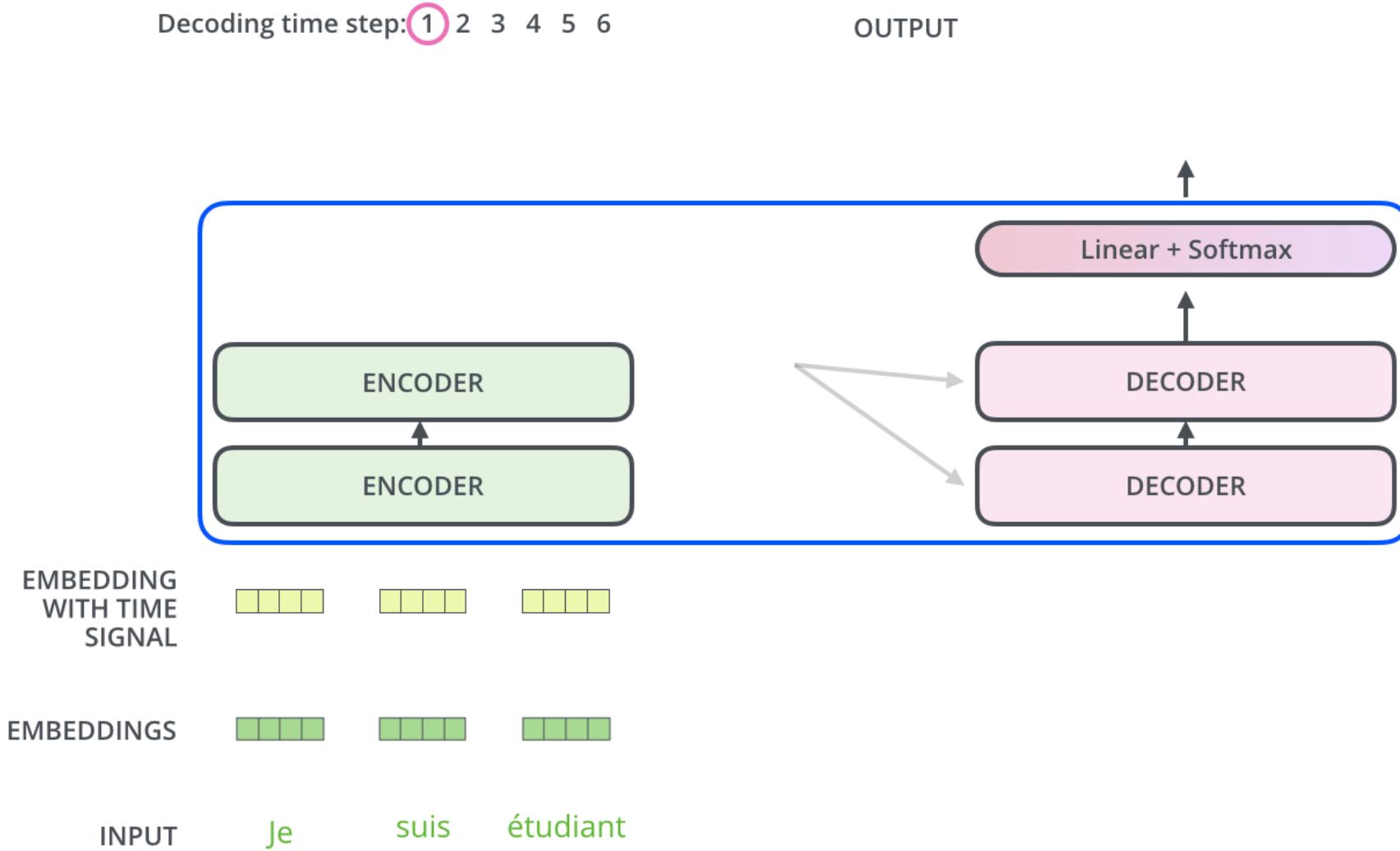


Residuals

- Goes for decoders as well

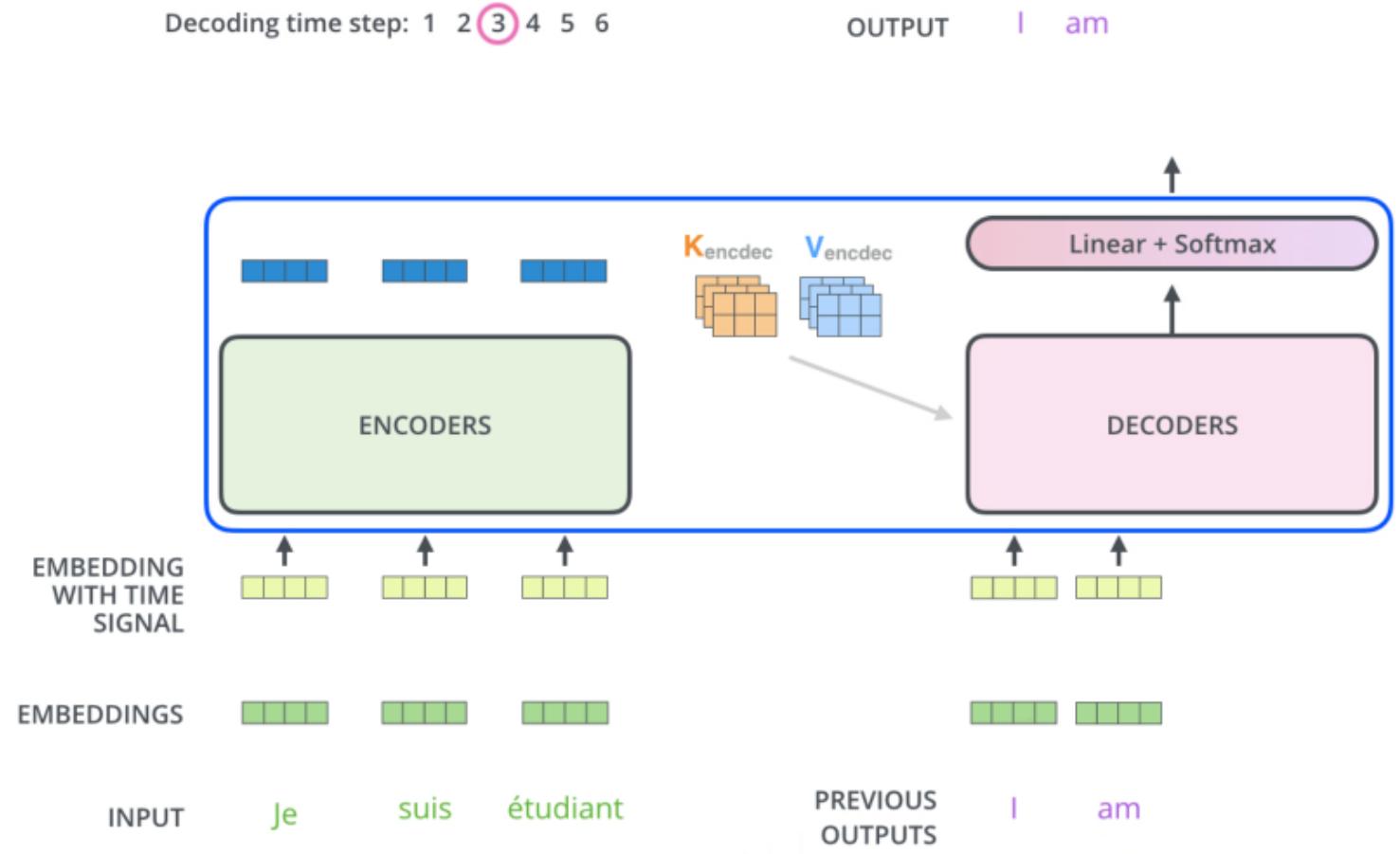


Encoder-Decoder



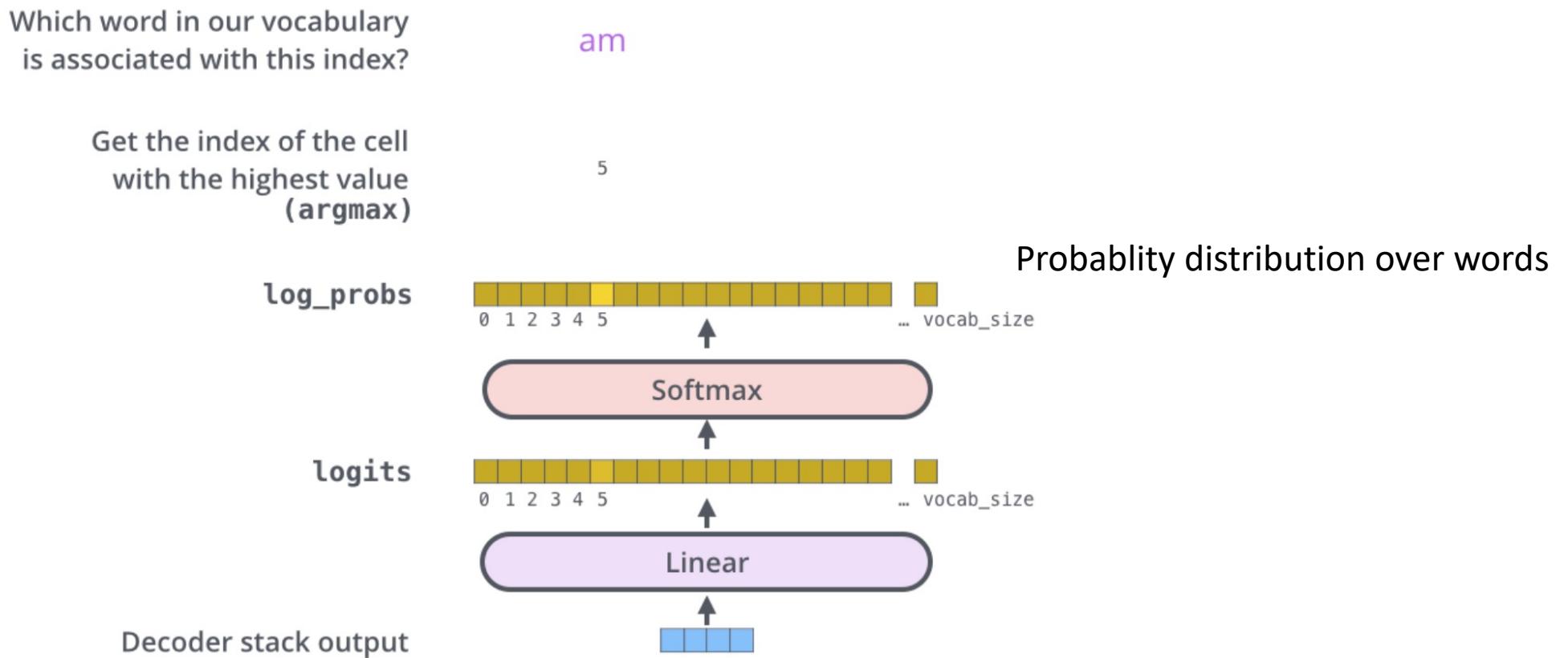
Encoder-Decoder

- The process repeats
 - Output of each step is fed to the bottom decoder in the next time step
 - Self-attention layer only attends to earlier positions in the input
 - Encoder-Decoder attention creates only Queries matrix from the layer below



Final Decoder Layer

- Output layer is turned into an output word



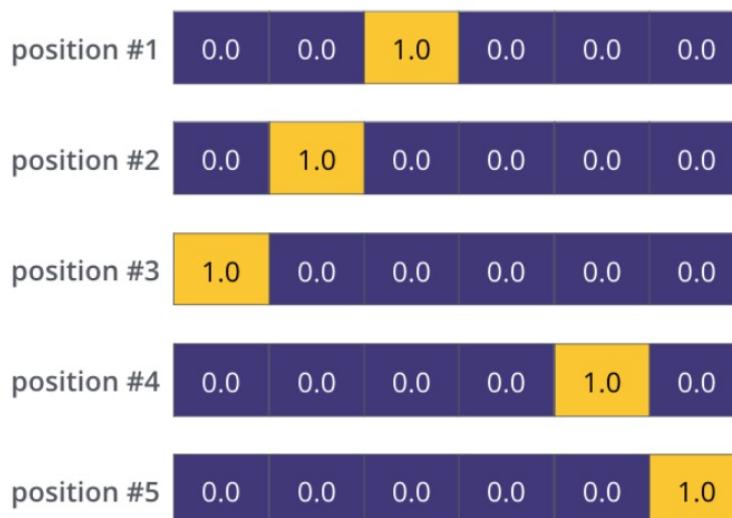
Final Decoder Layer

- Output layer is turned into an output word

Compare probability distribution over words

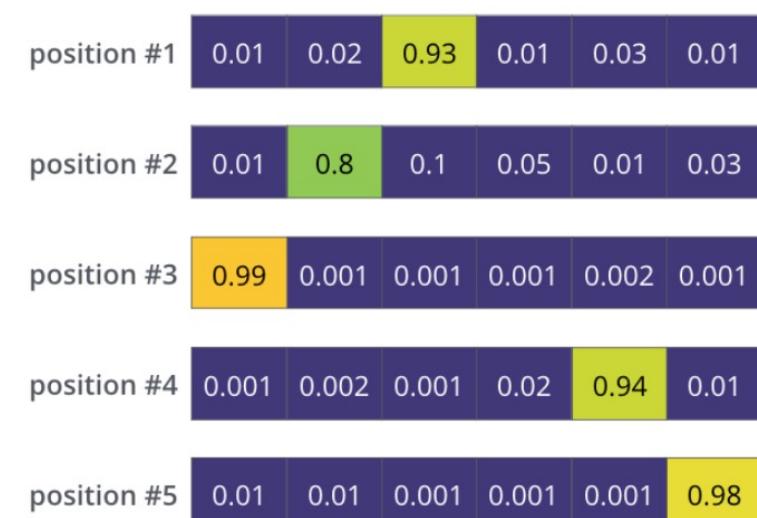
Target Model Outputs

Output Vocabulary: a am I thanks student <eos>



Trained Model Outputs

Output Vocabulary: a am I thanks student <eos>

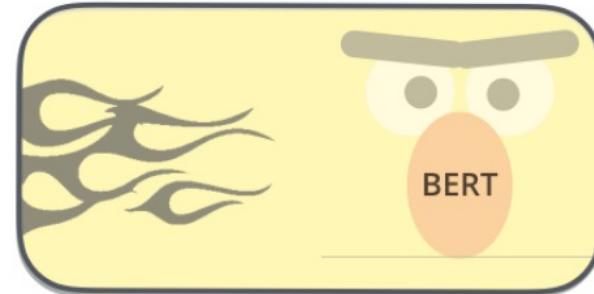
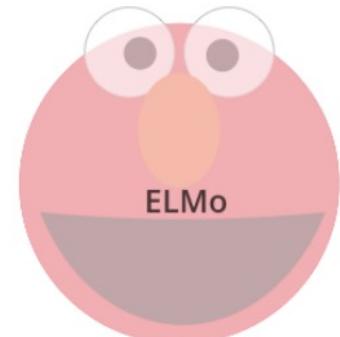


a am I thanks student <eos>

a am I thanks student <eos>

Word and Sentence Representations

<http://jalammar.github.io/illustrated-bert/>



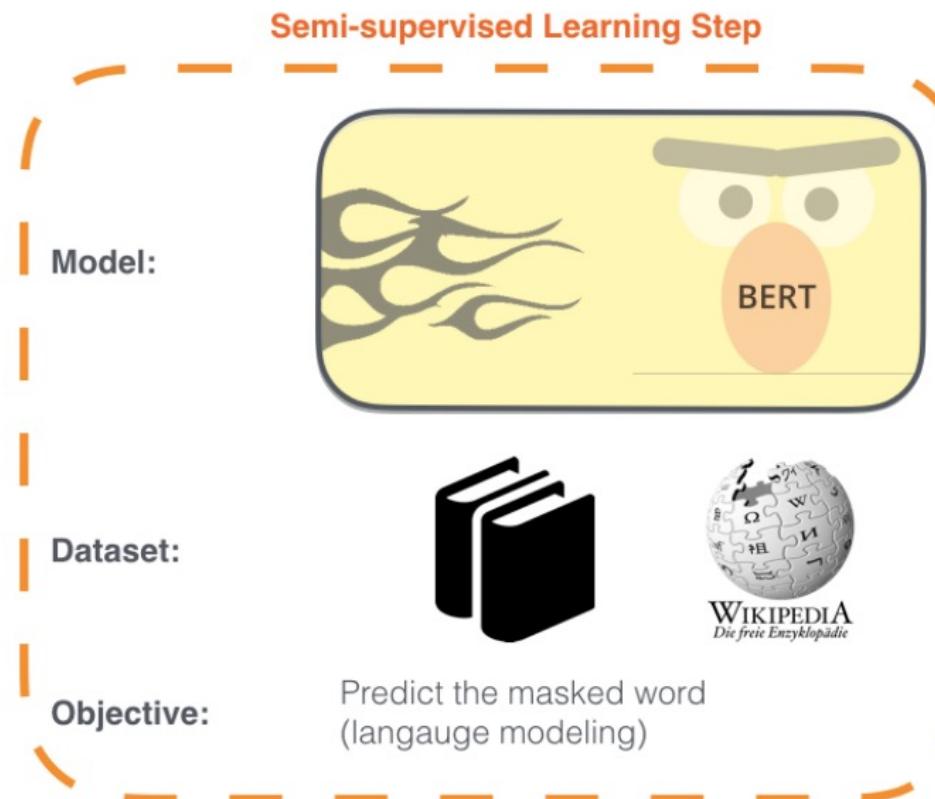
Milestone

- **BERT: Bidirectional Encoder Representations from Transformers**
 - A new method for pre-training language representations
 - Achieve state-of-the-art results on a wide array of NLP tasks
 - <https://github.com/google-research/bert>
 - Enables anyone to use this pre-trained LM and fine tune the representations for their tasks, without needing to train LM model from scratch

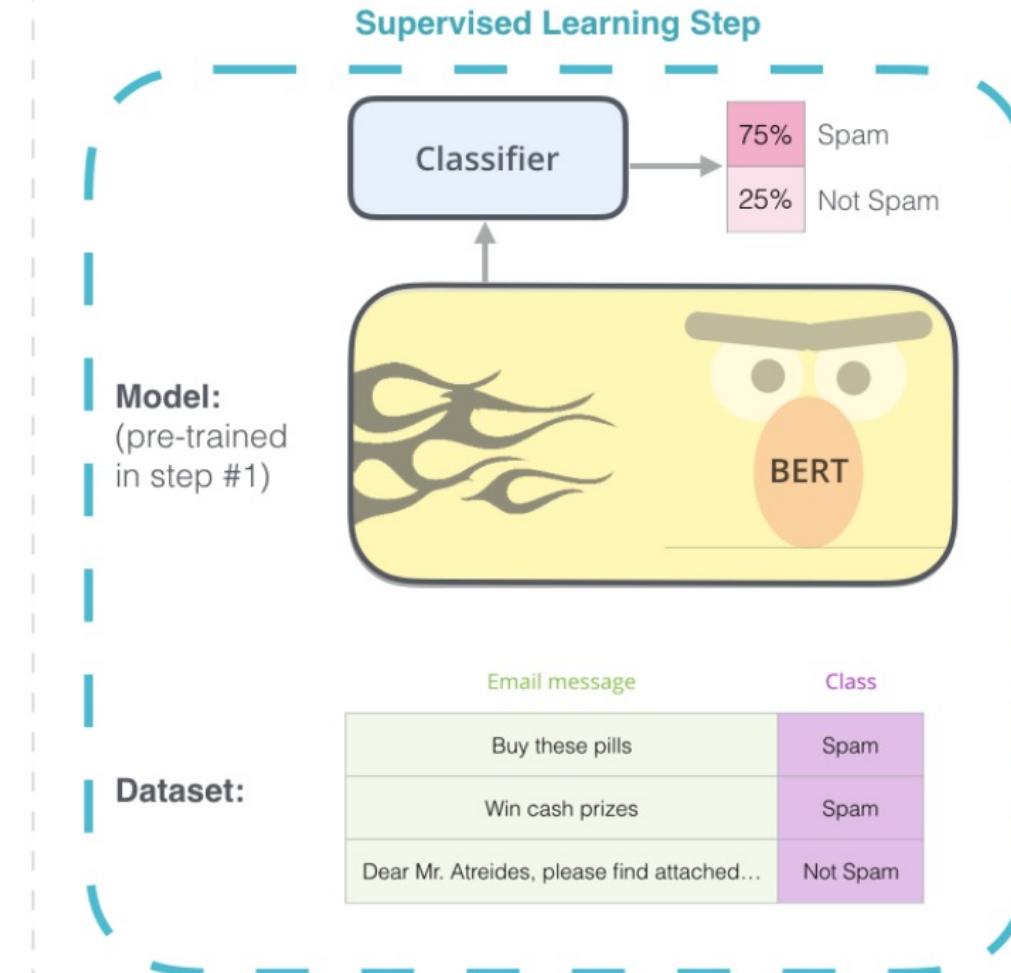
BERT

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.

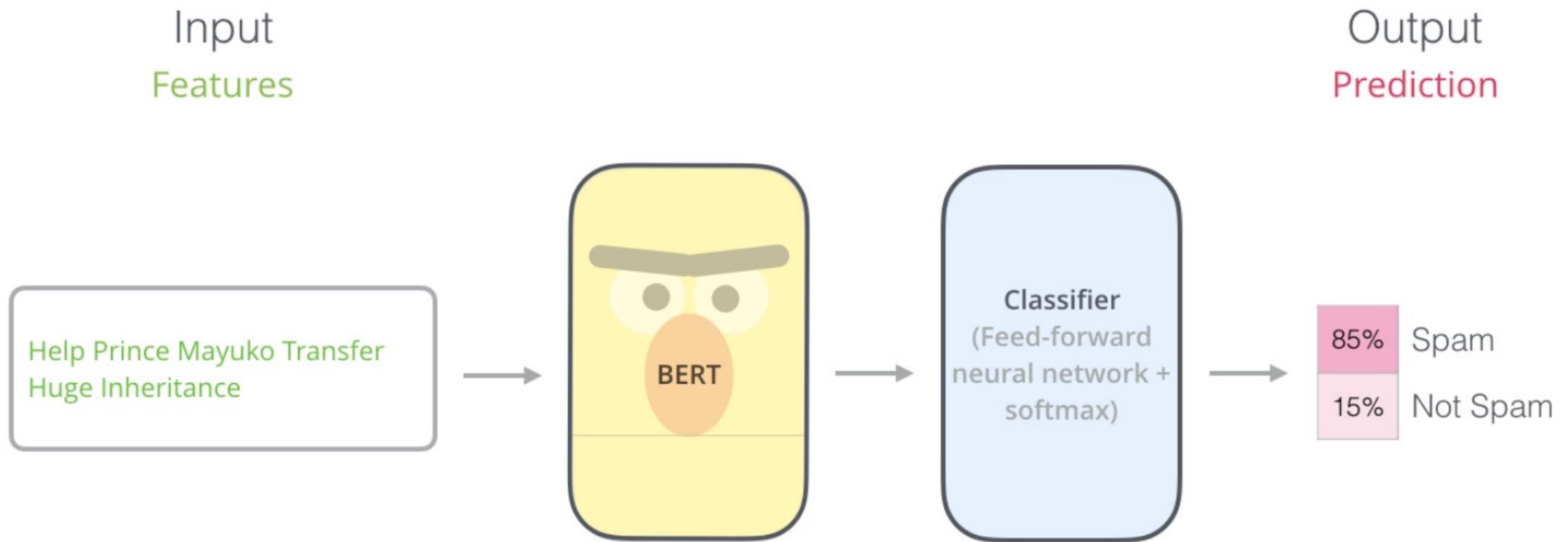


2 - Supervised training on a specific task with a labeled dataset.



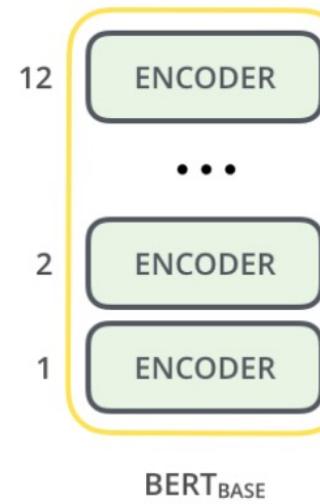
BERT

- Example: sentence classification

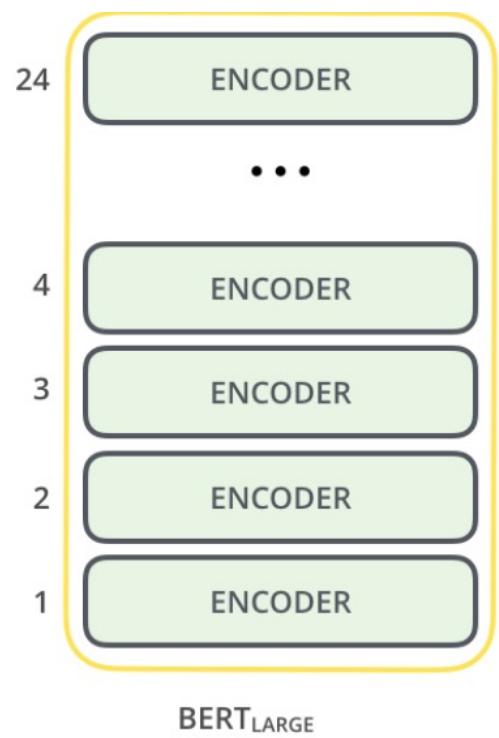


BERT

- Basically, a trained Transformed Encoder stack
- 12/24 layers
- 12/16 attention heads per layer
- 768/1024 hidden units in FFNN
- Initial transformer has:
 - 6 layers
 - 8 attention heads
 - 512 hidden units



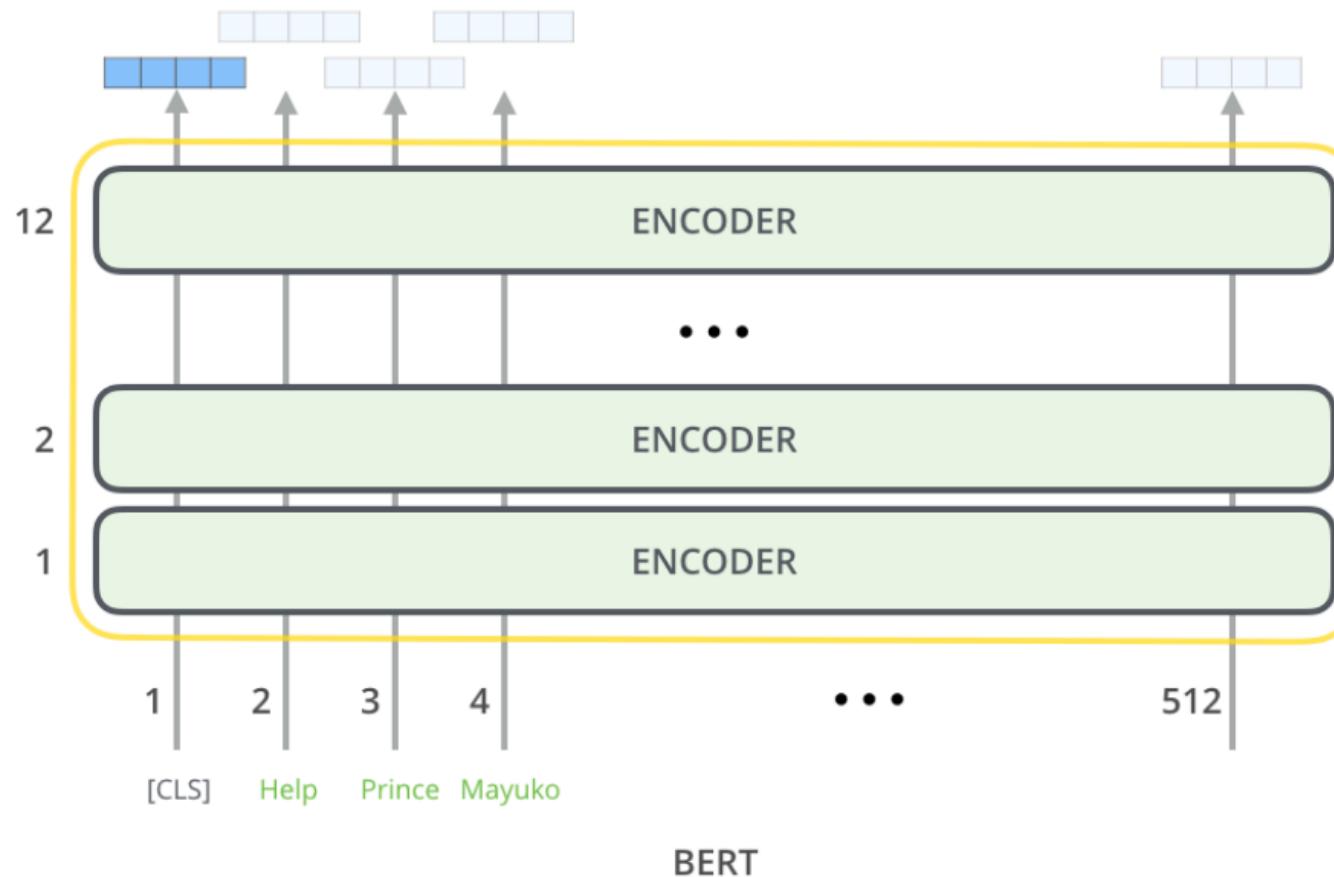
BERT_{BASE}



BERT_{LARGE}

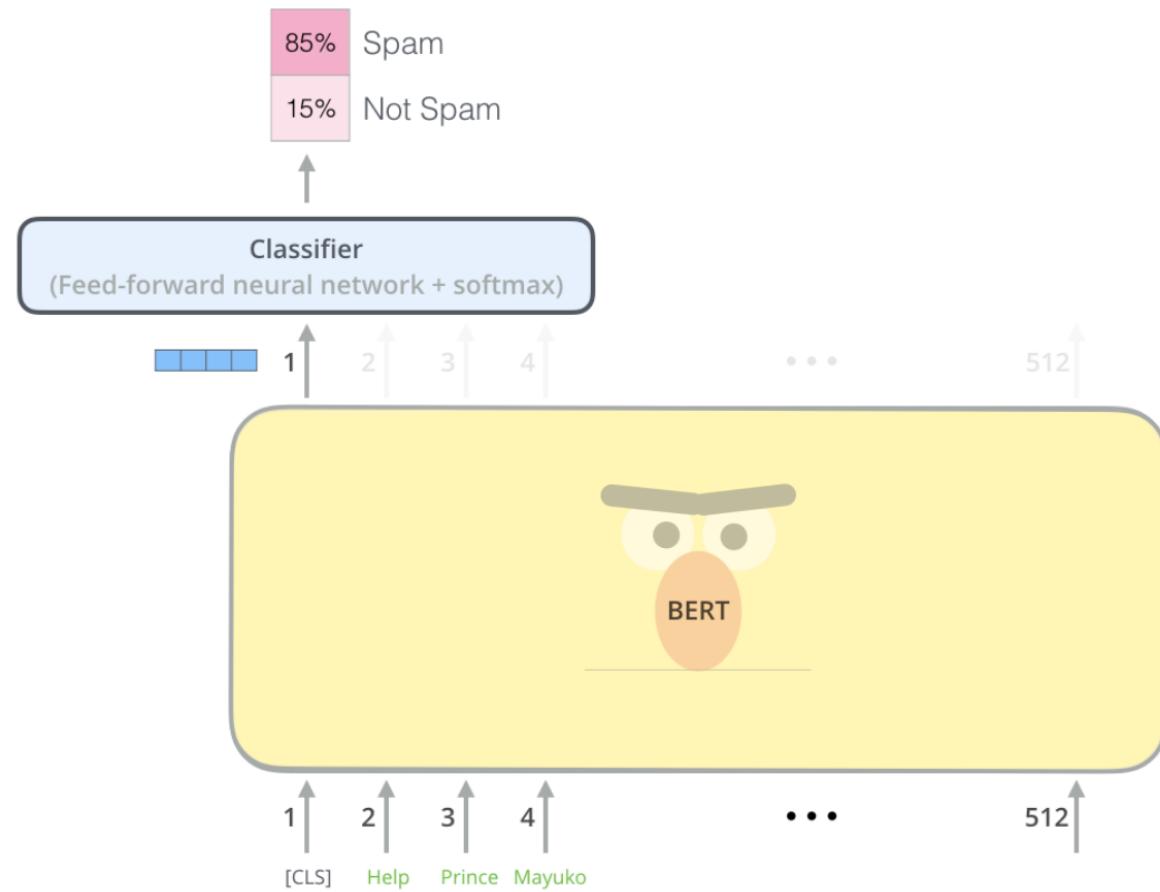
BERT

- Each position outputs a vector of size `hidden_size` (768/1024)



BERT

- The output vector can then be used as the input for a classifier of our choosing

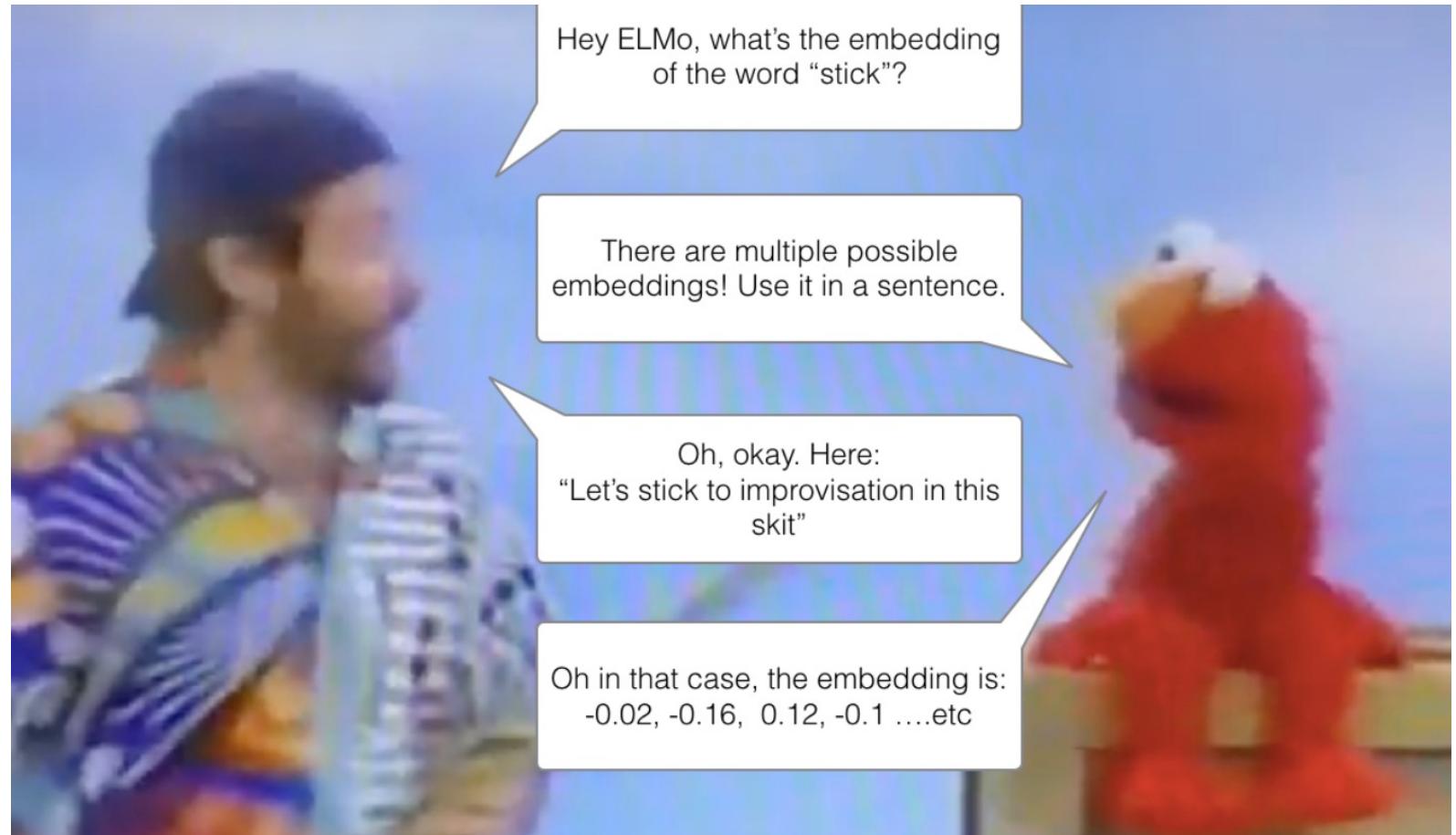


Context Encoding

- Word2vec, Glove don't really take context into consideration
 - The word “bank” has the same embedding no-matter what the context

Context Encoding

- Context2vec, ELMo were the start of the contextualized-word embeddings

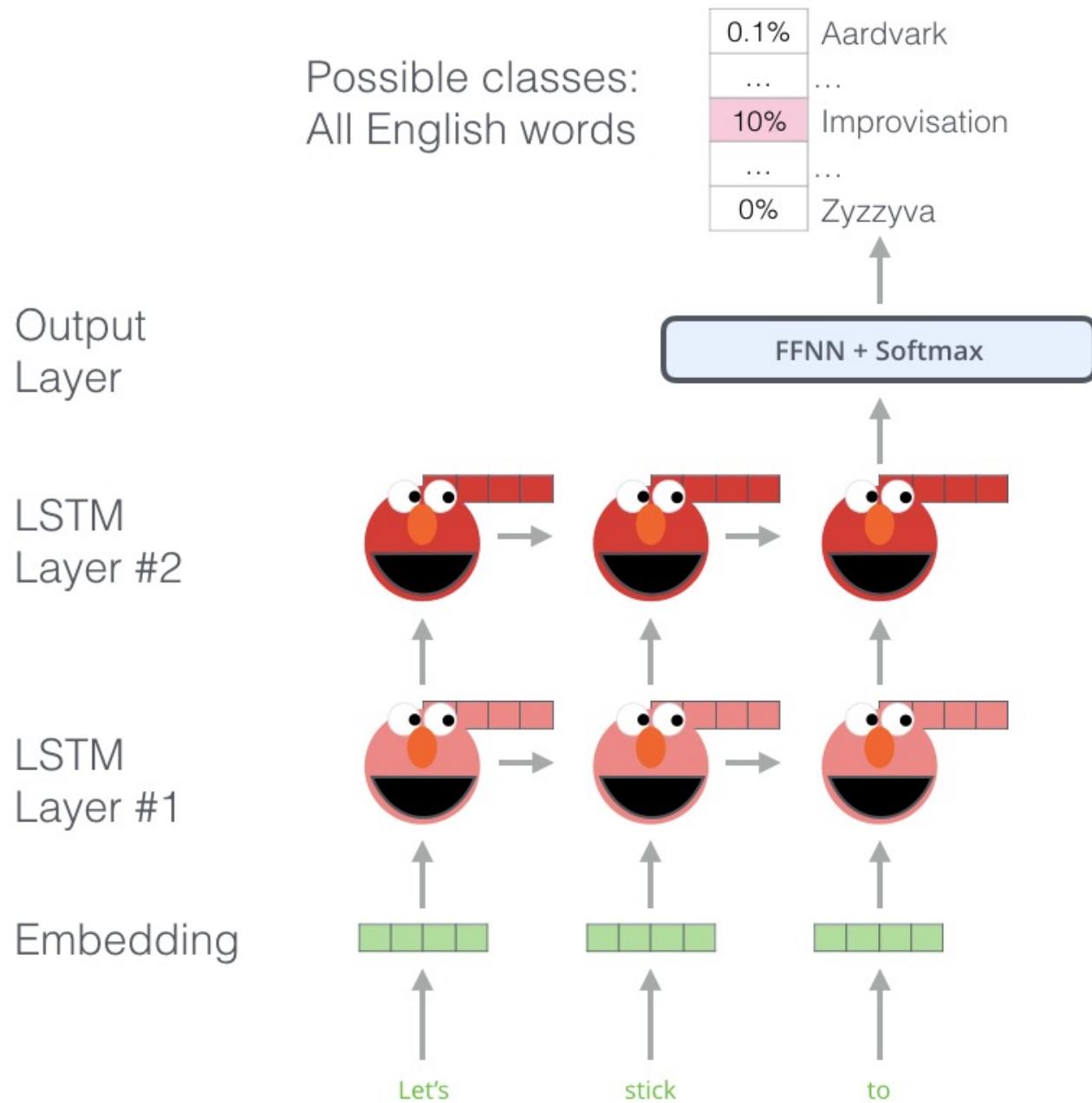


ELMo

- Instead of using a fixed embedding for each word, ELMo looks at the entire sentence before assigning each word in it an embedding
 - It uses bi-directional LSTM to create those embeddings
 - Trained to predict the next word in a sequence of words – another way of saying doing “language modeling”

ELMo

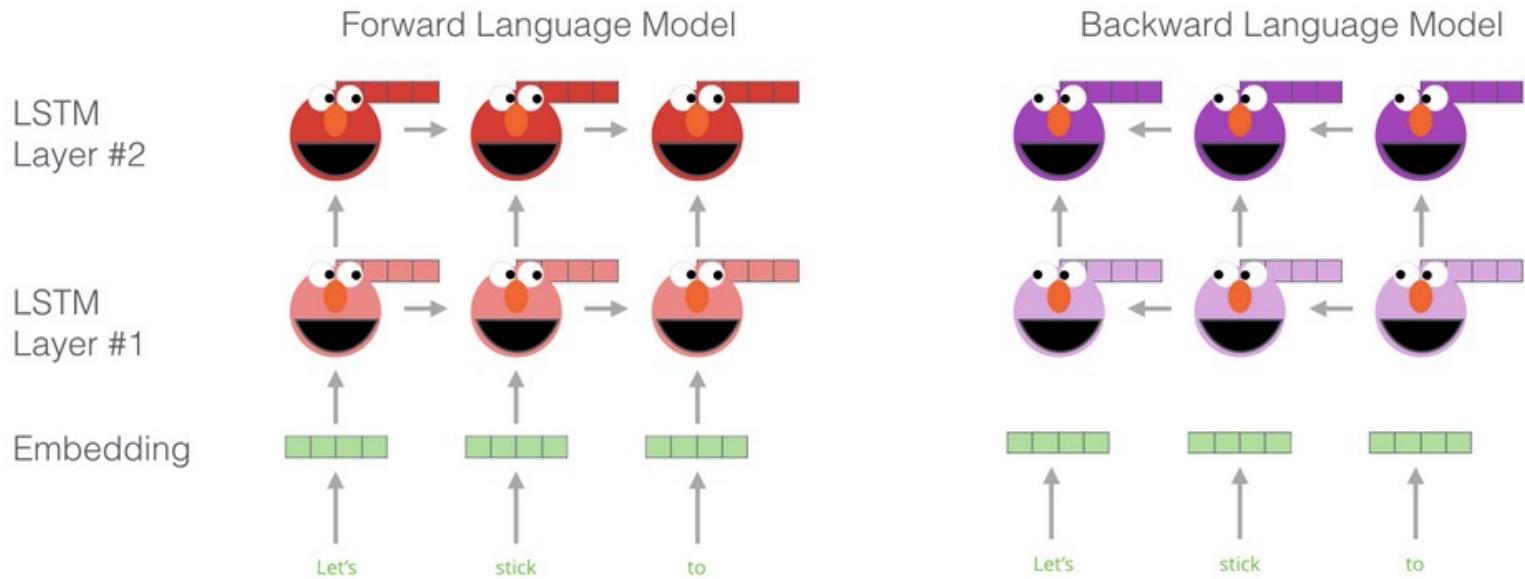
- Trained to predict the next word in a sequence of words – a.k.a. LM



ELMo

- Use bi-directional LSTM, so that the representation of a word is informed by its next *and* previous word

Embedding of “stick” in “Let’s stick to” - Step #1

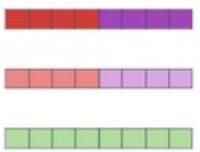


ELMo

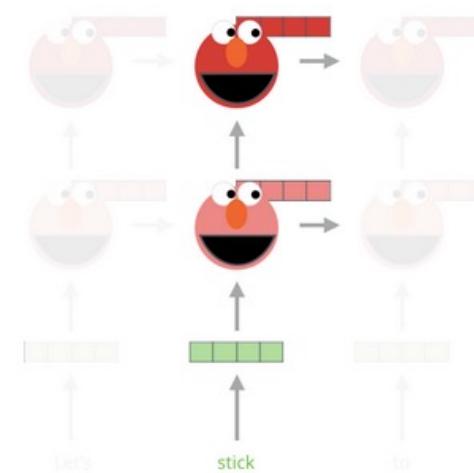
- Create a contextualized-embedding by grouping together the hidden states and initial embedding (concatenation followed by weighted summation)

Embedding of “stick” in “Let’s stick to” - Step #2

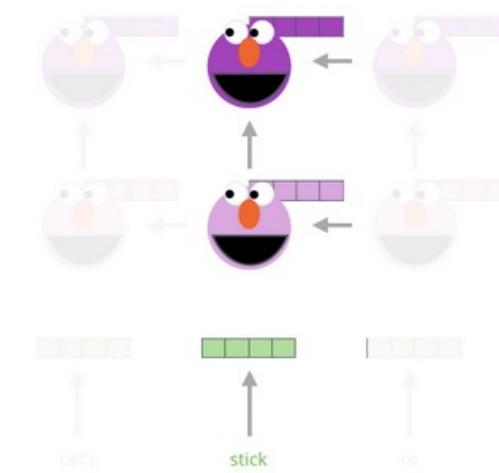
1- Concatenate hidden layers



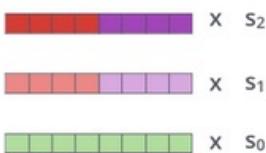
Forward Language Model



Backward Language Model



2- Multiply each vector by a weight based on the task



3- Sum the (now weighted) vectors



ELMo embedding of “stick” for this task in this context

BERT

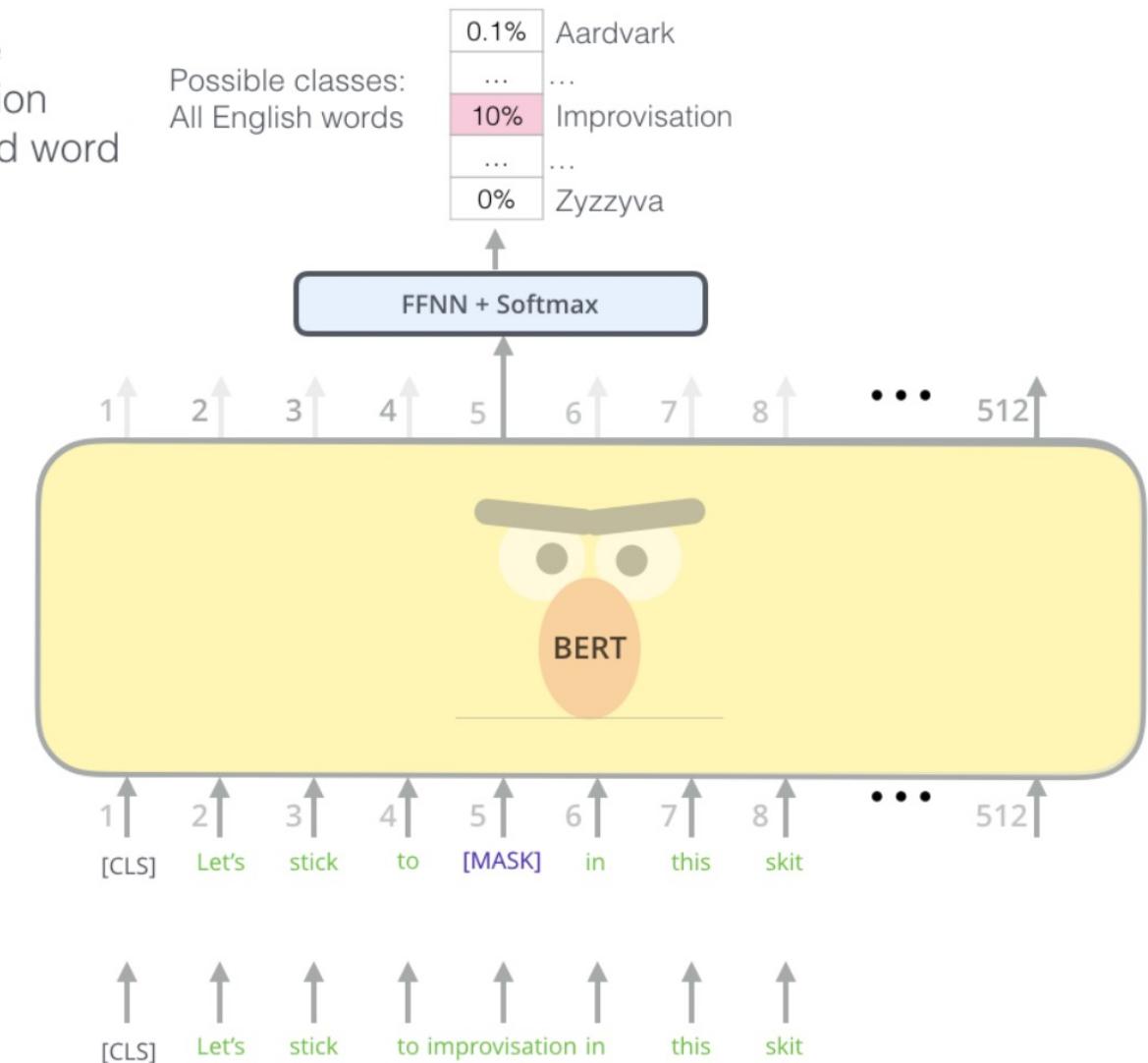
- Transformer-encoder model whose LM looks both forward and backward
 - Cannot be trained like other typical “predict the next word in the sequence” LM since transformer-encoder is multi layered and absorbs all parts of the input sentence at once
 - Using Masked LM

BERT

- Masked LM also called a Cloze task
- Also randomly replaces a word with another word and asks to predict the correct word

Randomly mask 15% of tokens

Input

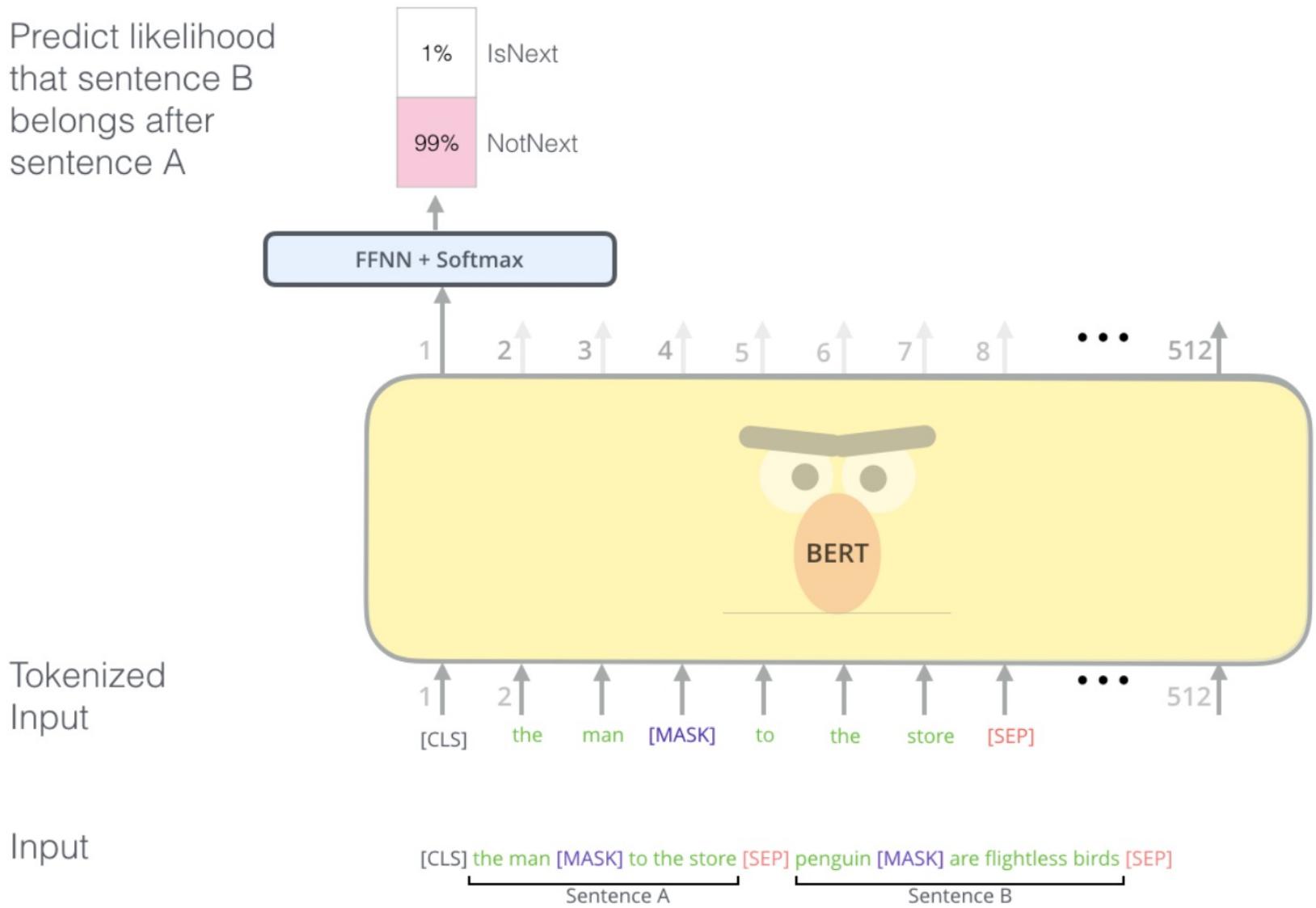


Use the output of the masked word's position to predict the masked word

BERT

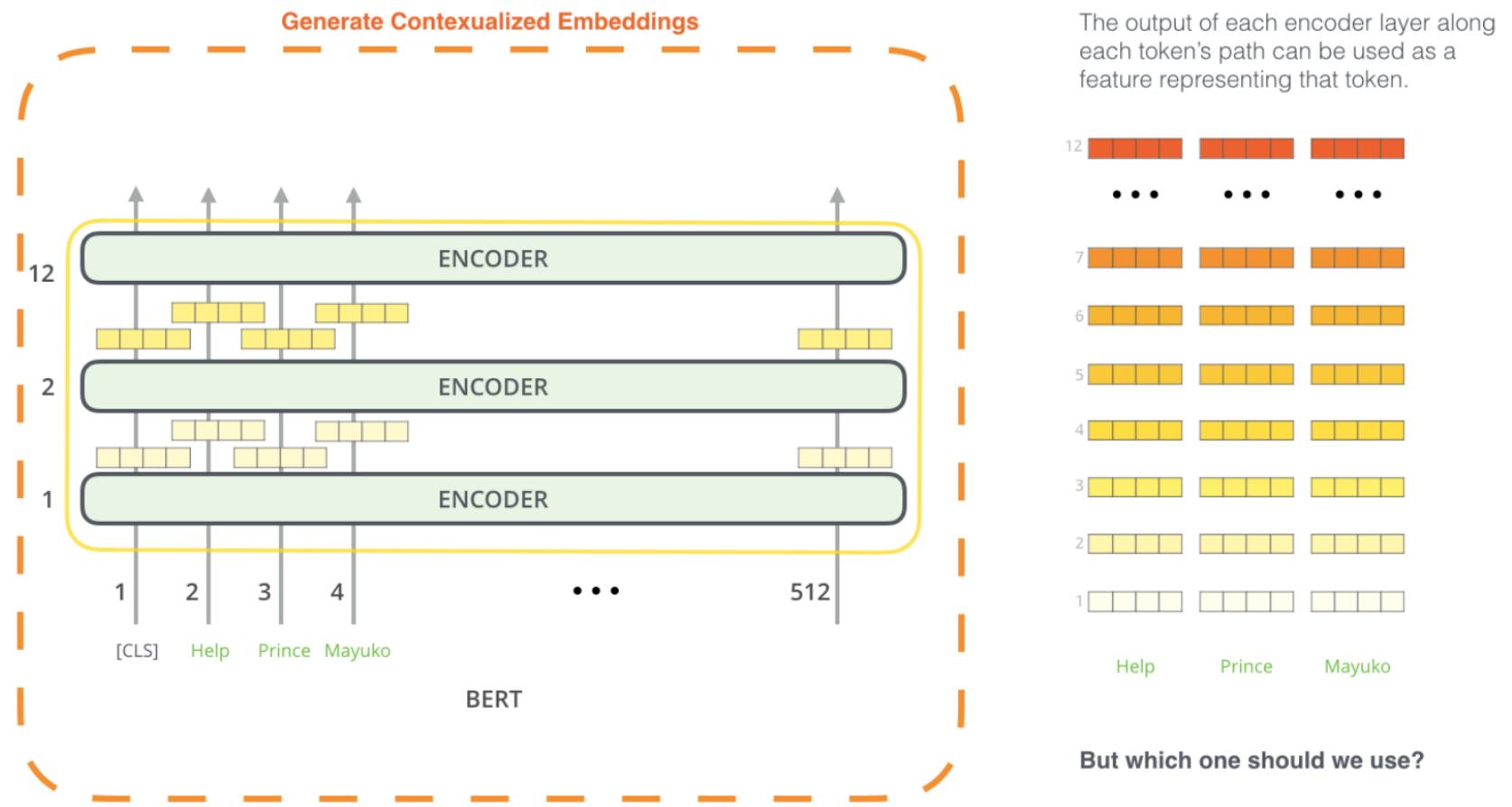
- Also trained on multiple sentences

Predict likelihood that sentence B belongs after sentence A



BERT

- Can also be used to create embeddings (without fine tuning)

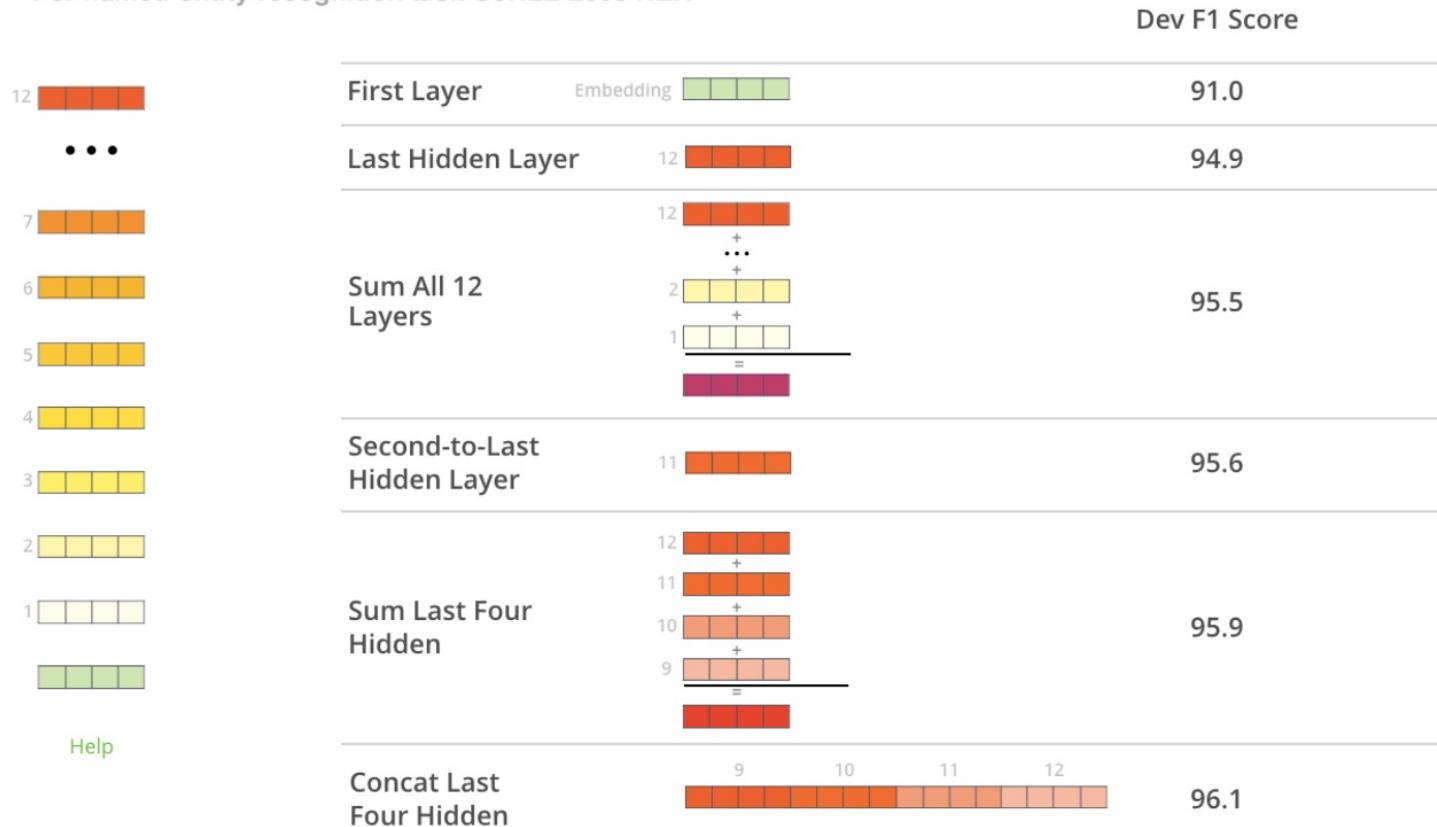


BERT

- Can also be used to create embeddings (without fine tuning)

What is the best contextualized embedding for “Help” in that context?

For named-entity recognition task CoNLL-2003 NER



Try out BERT and other Transformers

- <https://github.com/google-research/bert>
- <https://github.com/huggingface/transformers>

GPT Language Models

The importance of being on twitter

by Jerome K. Jerome
London, Summer 1897

It is a curious fact that the last remaining form of social life in which the people of London are still interested is Twitter. I was struck with this curious fact when I went on one of my periodical holidays to the sea-side, and found the whole place twittering like a starling-cage. I called it an anomaly, and it is.

I spoke to the sexton, whose cottage, like all sexton's cottages, is full of antiquities and interesting relics of former centuries. I said to him, "My dear sexton, what does all this twittering mean?" And he replied, "Why, sir, of course it means Twitter." "Ah!" I said, "I know about that. But what is Twitter?"

"It is a system of short and pithy sentences strung together in groups, for the purpose of conveying useful information to the initiated, and entertainment and the exercise of wits to the initiated, and entertainment and the exercise of wits to the rest of us."

"Very interesting," I said. "Has it a name?"
"It has," he said; "it is called Twitter."
"Yes," I said, "I know that, but what is it?"
"It is a system of information," he said.
"Oh, yes," I replied; "but what is it?"

"Why, sir," he said, "you can go up to any of the gentlemen you see twittering in the street, and say to him, 'You are a fool,' or 'Your wife is an adulteress,' or 'You have stolen that hat,' and if he is a member of the initiated he will answer you in the same form and tell you that you are a liar, or that your eyes resemble the eyes of a duck, or that you have stepped out of your part in the last charade you acted in, or that you were for a short time a statistician in a Government Office, and he will go on to tell you the whole story of your life, in language so exceedingly small and pointed that even you will be glad you can't understand it."

Mario Klingemann ...

@quasimondo

Another attempt at a longer piece. An imaginary Jerome K. Jerome writes about Twitter. All I seeded was the title, the author's name and the first "It", the rest is done by #gpt3

Here is the full-length version as a PDF:
drive.google.com/file/d/1qtPa1c...

11:25 AM · Jul 18, 2020 · Twitter Web App

400 Retweets 176 Quote Tweets

1,265 Likes

Tweet your reply

GPT Language Models

- Language model that is able to look at part of a sentence and predict the next word
 - Autocomplete
 - Larger, more sophisticated
 - GPT-2 is trained on 40GB WebText
 - Takes 500MBs for storage (smallest) to 6.5GB (largest)



GPT Language Models



**BERT 110M
parameters**



117M Parameters



345M Parameters



762M Parameters



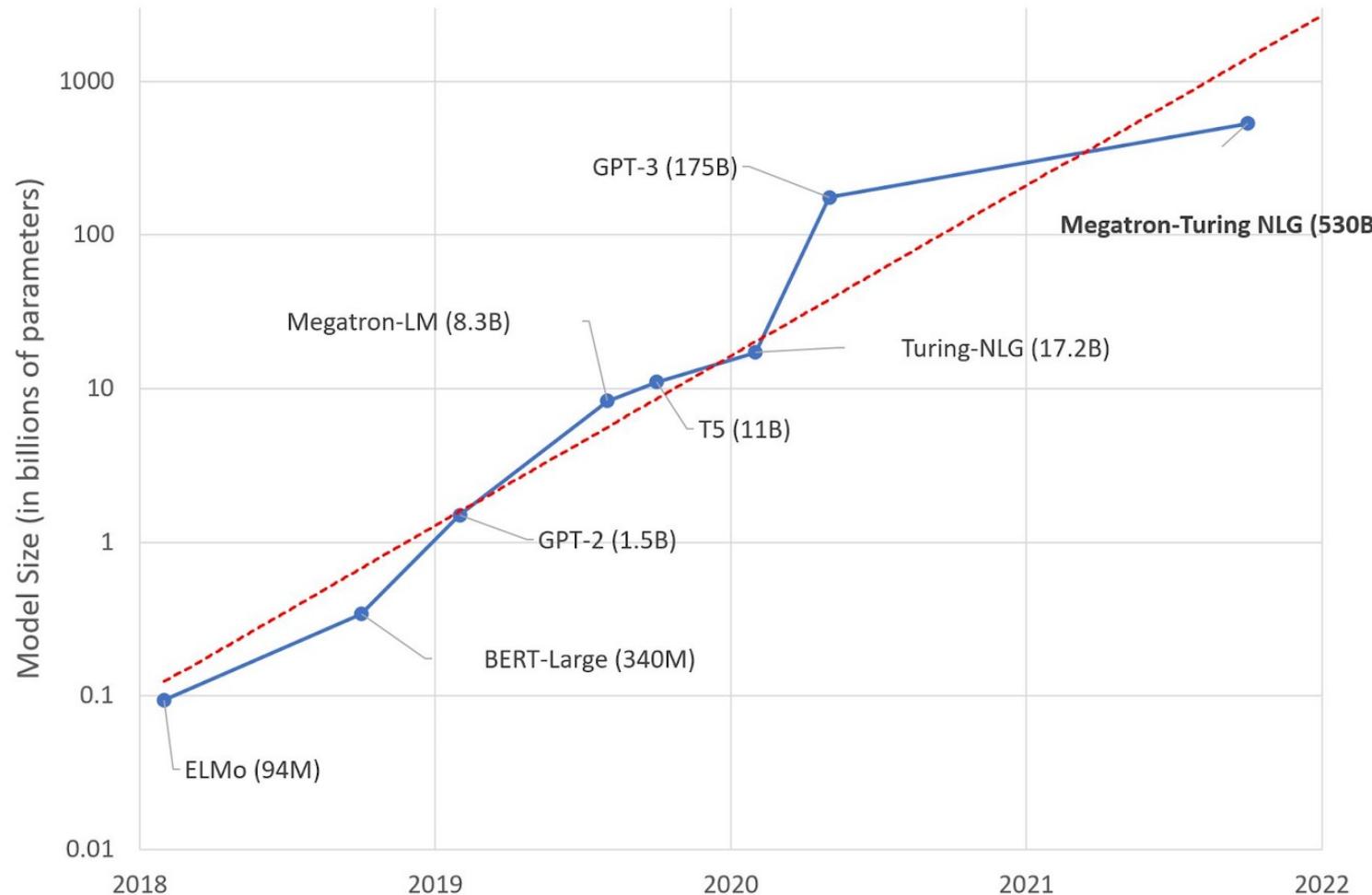
1,542M Parameters

GPT Language Models



175B
parameters

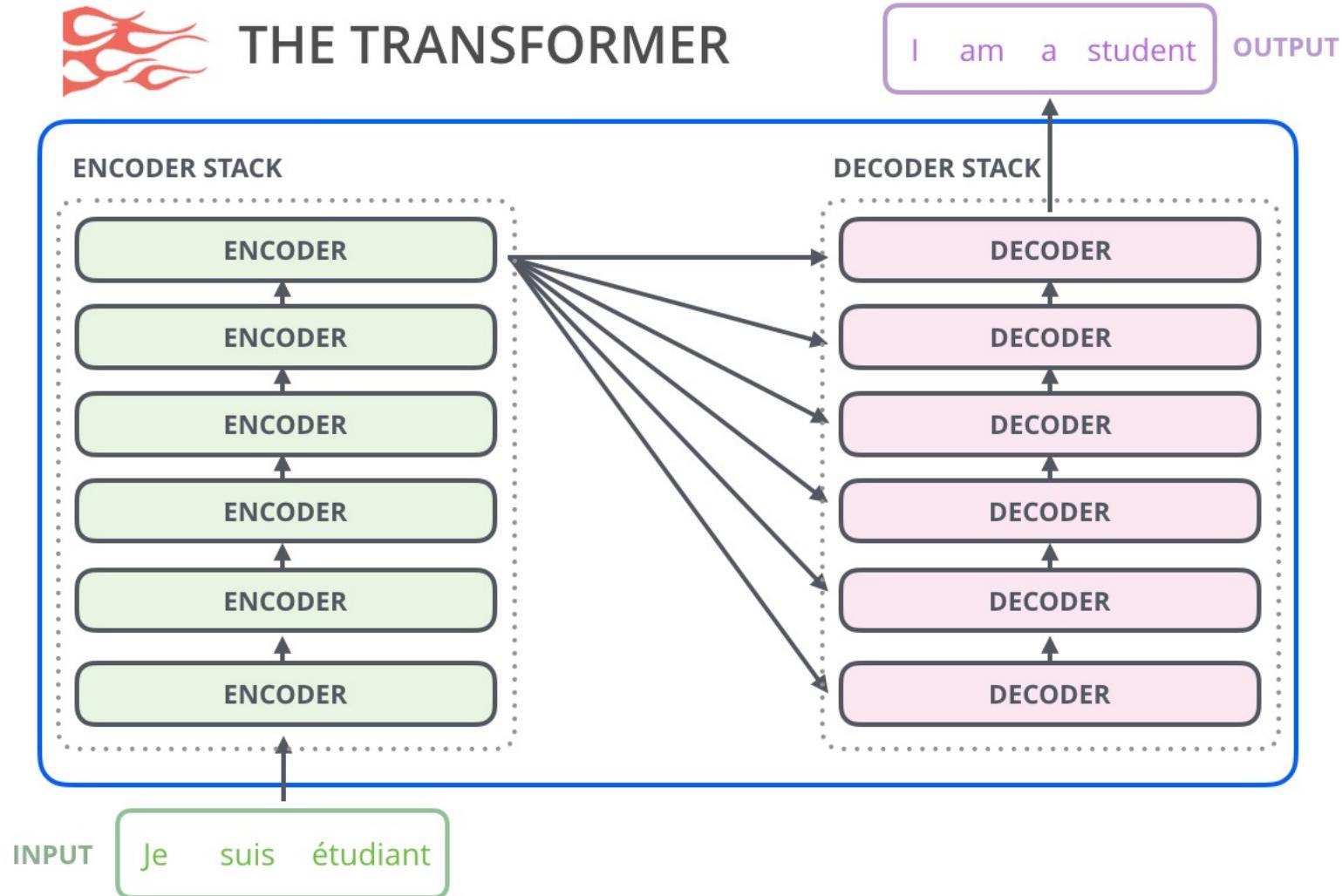
GPT Language Models



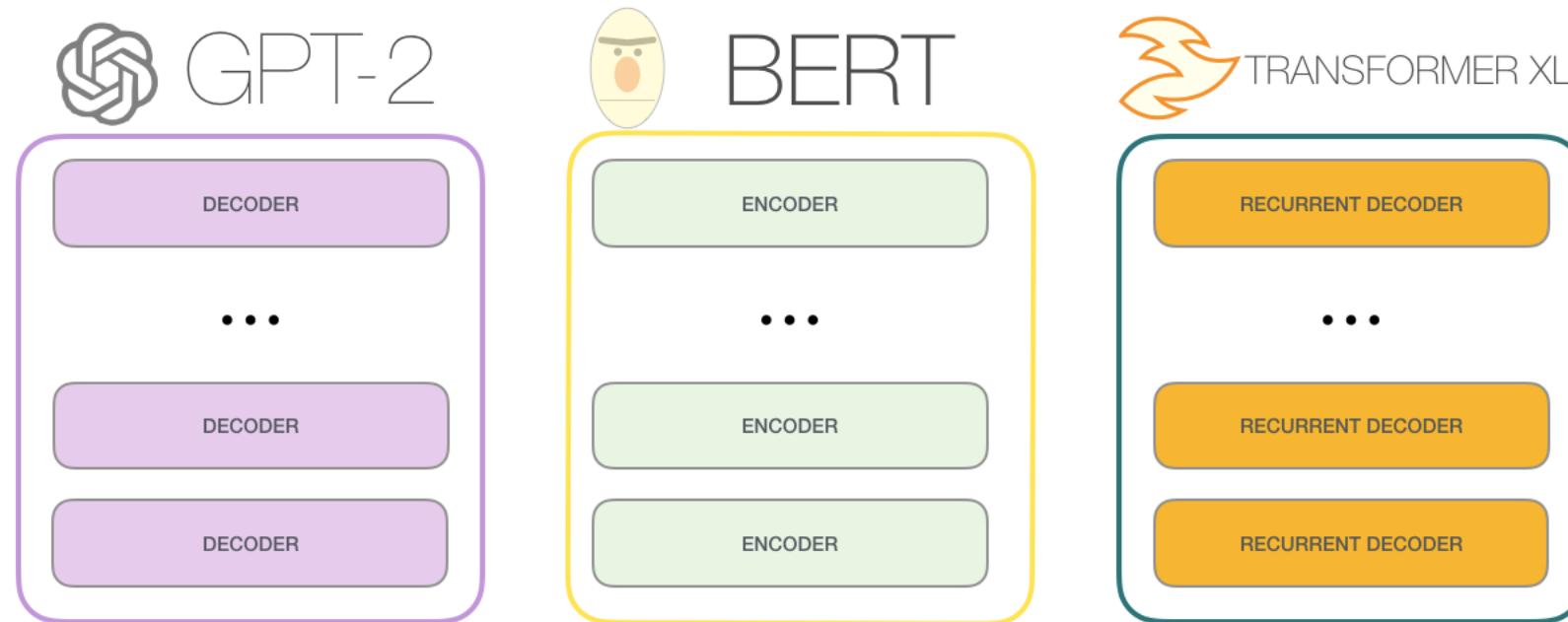
GPT Language Models



THE TRANSFORMER



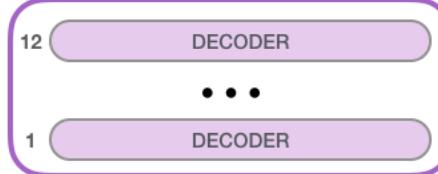
GPT Language Models



GPT Language Models



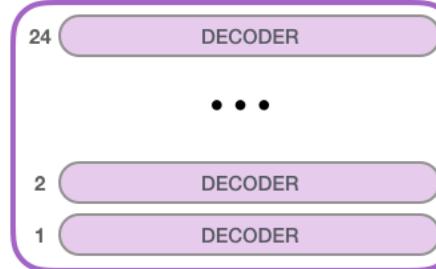
GPT-2
SMALL



Model Dimensionality: 768



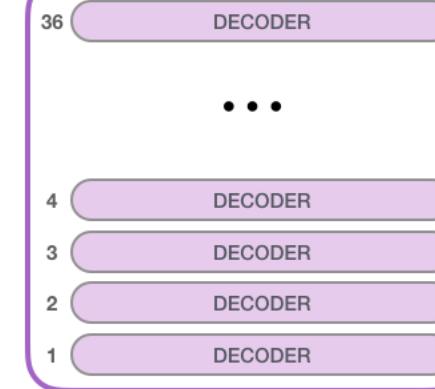
GPT-2
MEDIUM



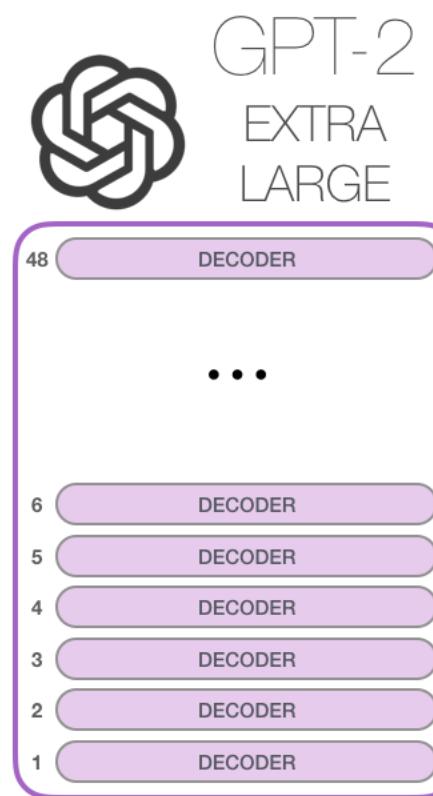
Model Dimensionality: 1024



GPT-2
LARGE



Model Dimensionality: 1280



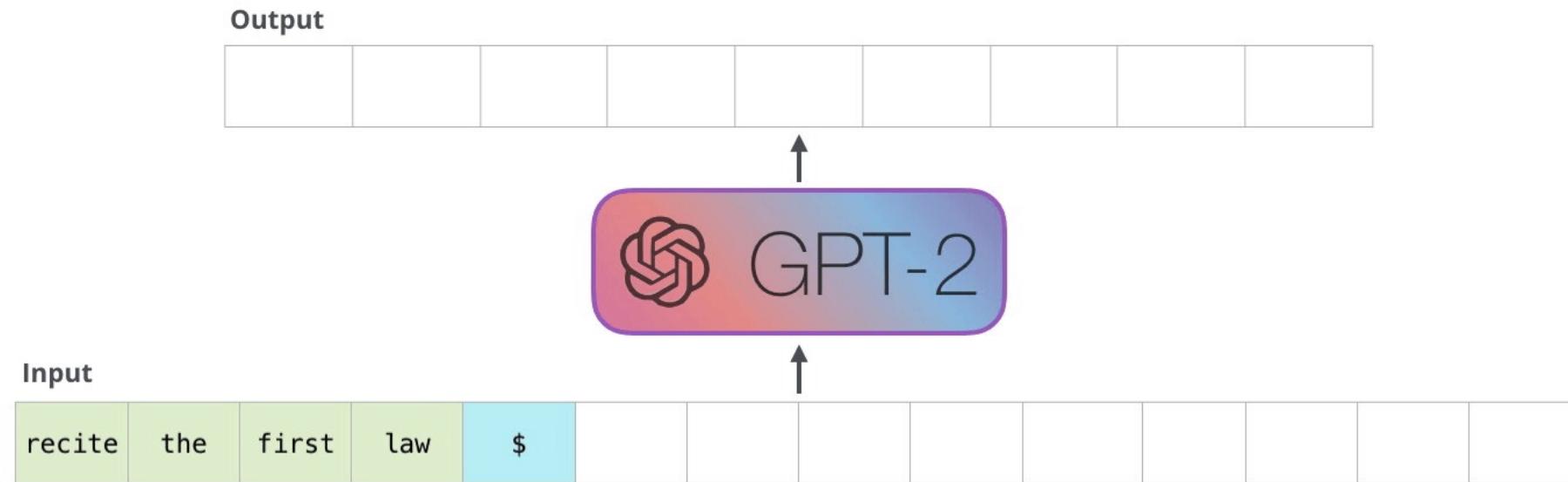
Model Dimensionality: 1600

GPT-2
EXTRA
LARGE

GPT Language Models

- Only using transformer decoder blocks (BERT is using transformer encoder blocks)
- Outputs one token at a time:
 - “autoregression”: after each token is produced, that token is added to the sequence of inputs, and that new sequence becomes the input to the model in its next step

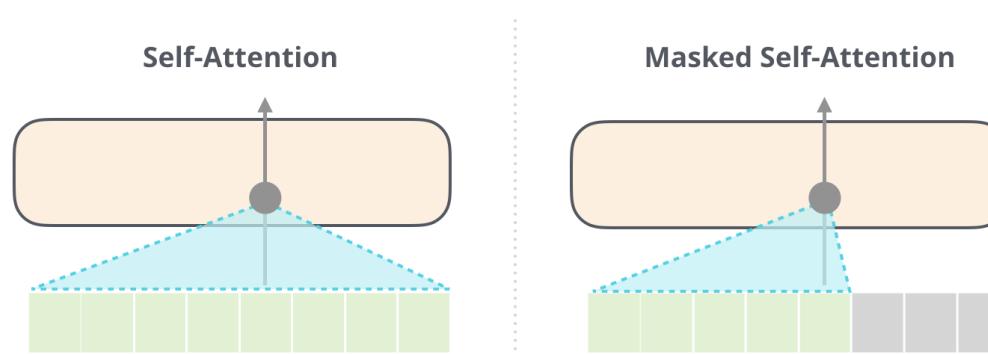
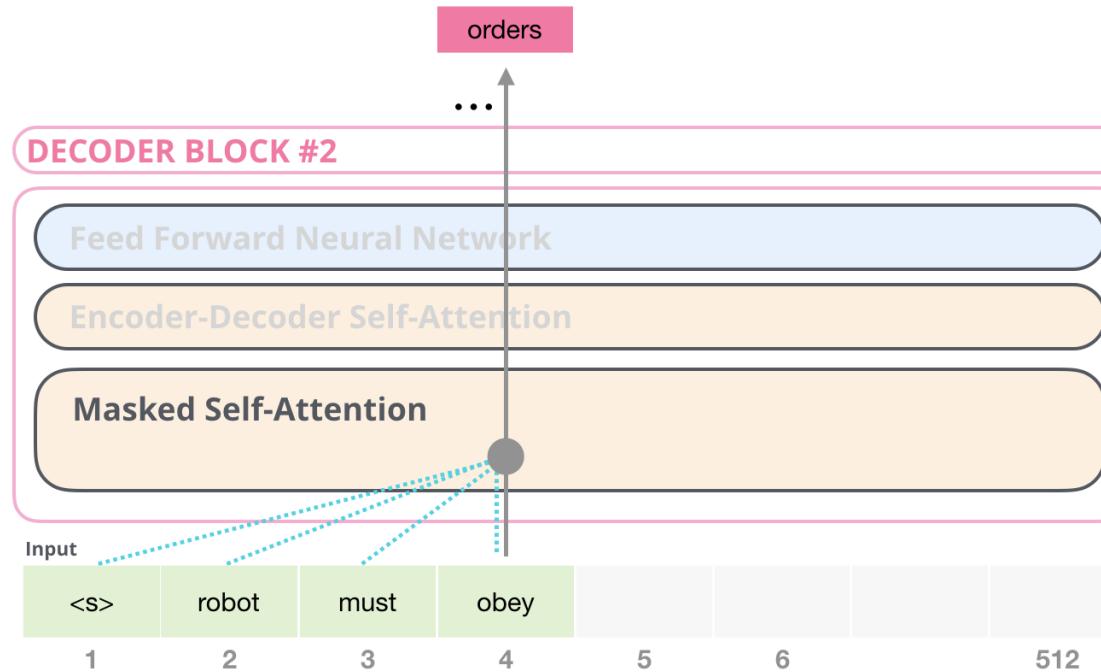
GPT Language Models



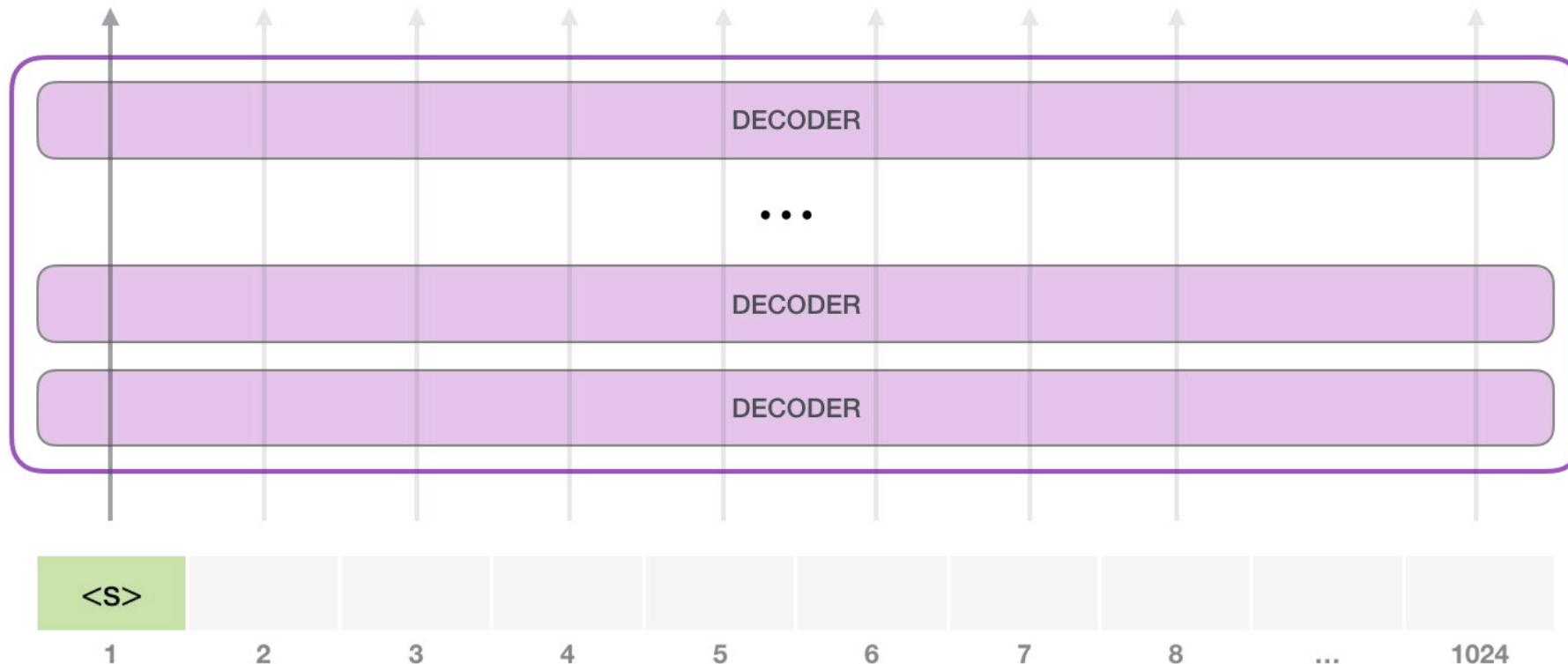
GPT Language Models

- Only using transformer decoder blocks (BERT is using transformer encoder blocks)
- Outputs one token at a time:
 - “autoregression”: after each token is produced, that token is added to the sequence of inputs, and that new sequence becomes the input to the model in its next step
- Self-attention layer masks future tokens

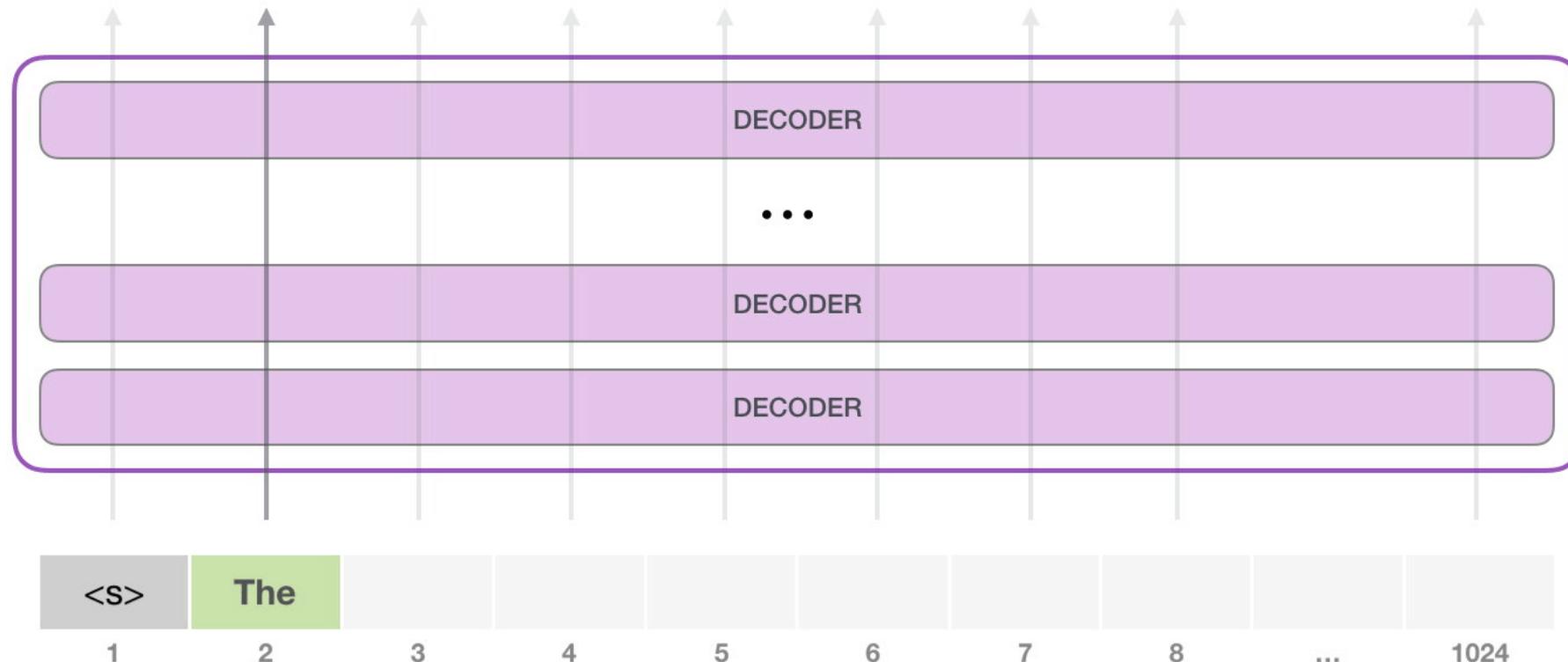
GPT Language Models



GPT Language Models



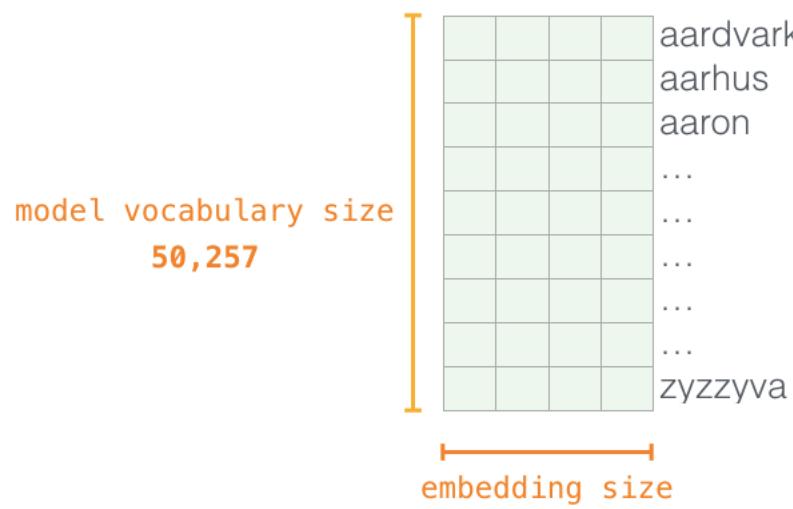
GPT Language Models



GPT Language Models – Inner Workings

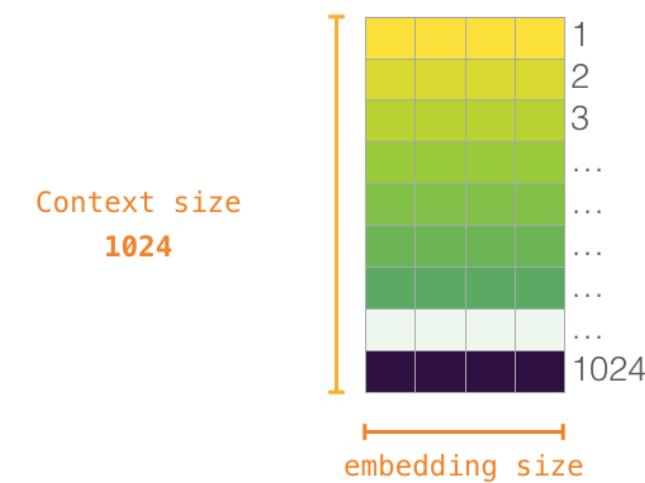
- Inputs:

Token Embeddings (wte)



768 (small) / 1024 (medium) / 1280 (large) / 1600 (extra large)

Positional Encodings (wpe)



768 (small) / 1024 (medium) / 1280 (large) / 1600 (extra large)

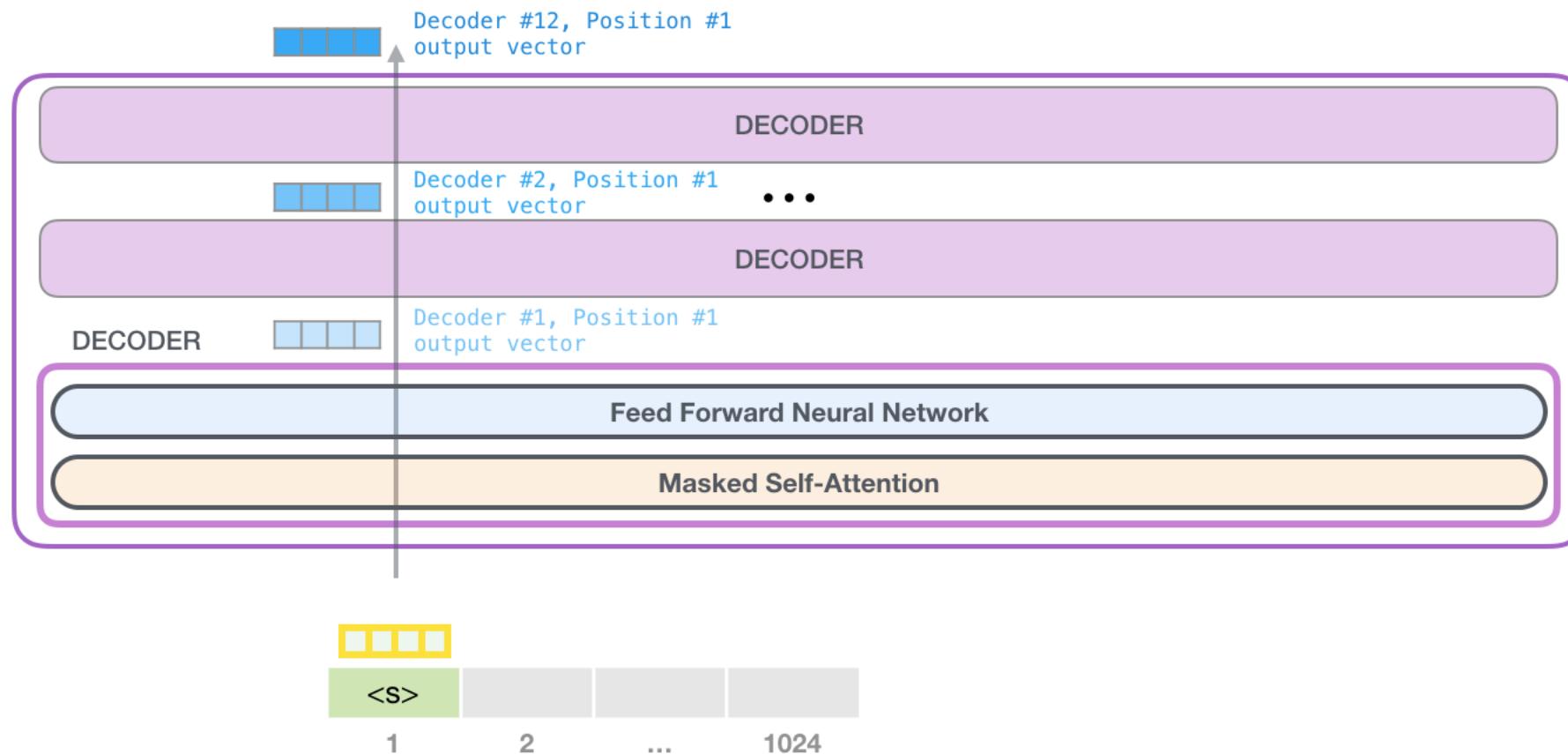
GPT Language Models – Inner Workings

- Inputs:



GPT Language Models – Inner Workings

- Up the stack:



GPT Language Models – Inner Workings

- Self-attention:
 - Process words by incorporating the context they are referring to in the sentence
 - By assigning scores to how relevant each word in the sentence is and adding up their vector representation

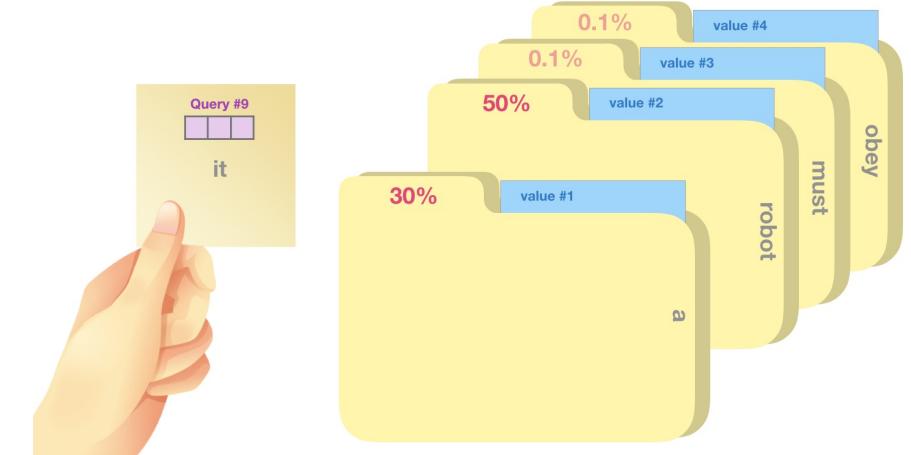
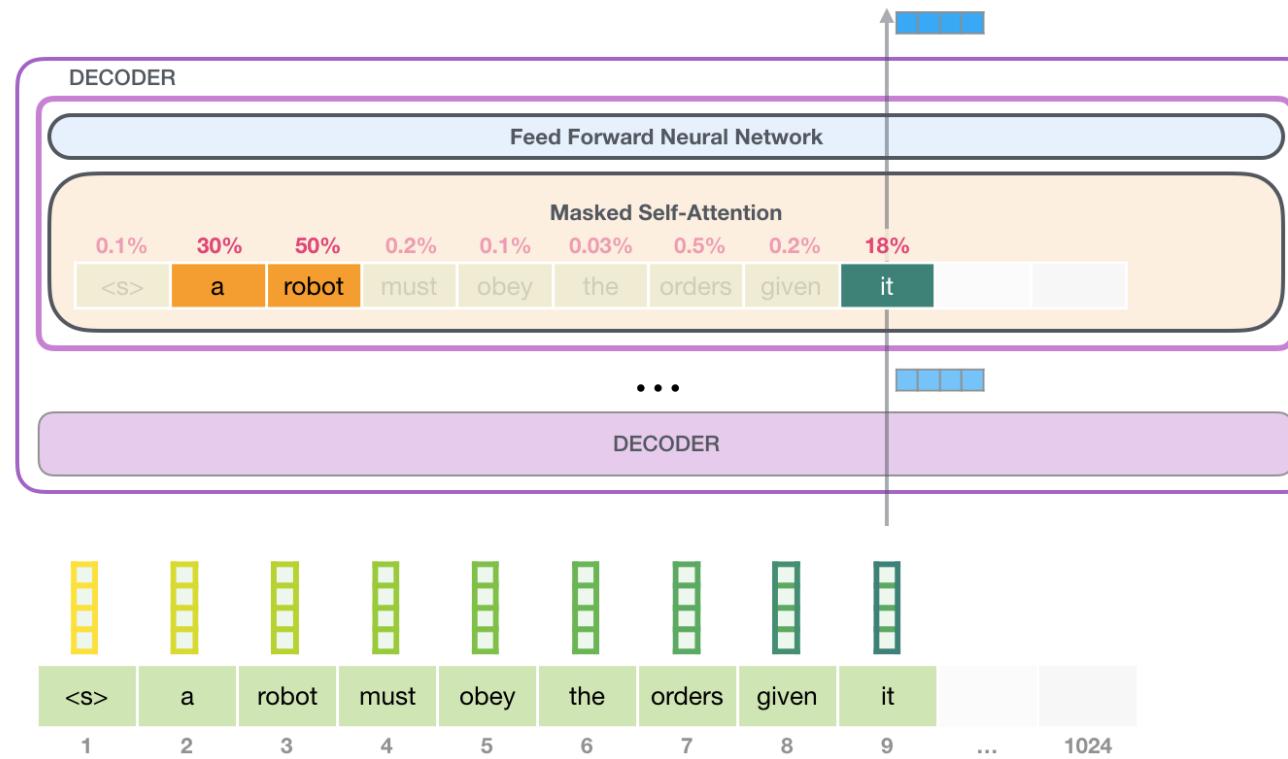
Second Law of Robotics

*A robot must obey the orders given **it** by human beings except where **such orders** would conflict with the **First Law**.*

- **it** refers to the robot
- **such orders** refers to the earlier part of the law, namely “the orders given it by human beings”
- **The First Law** refers to the entire First Law

GPT Language Models – Inner Workings

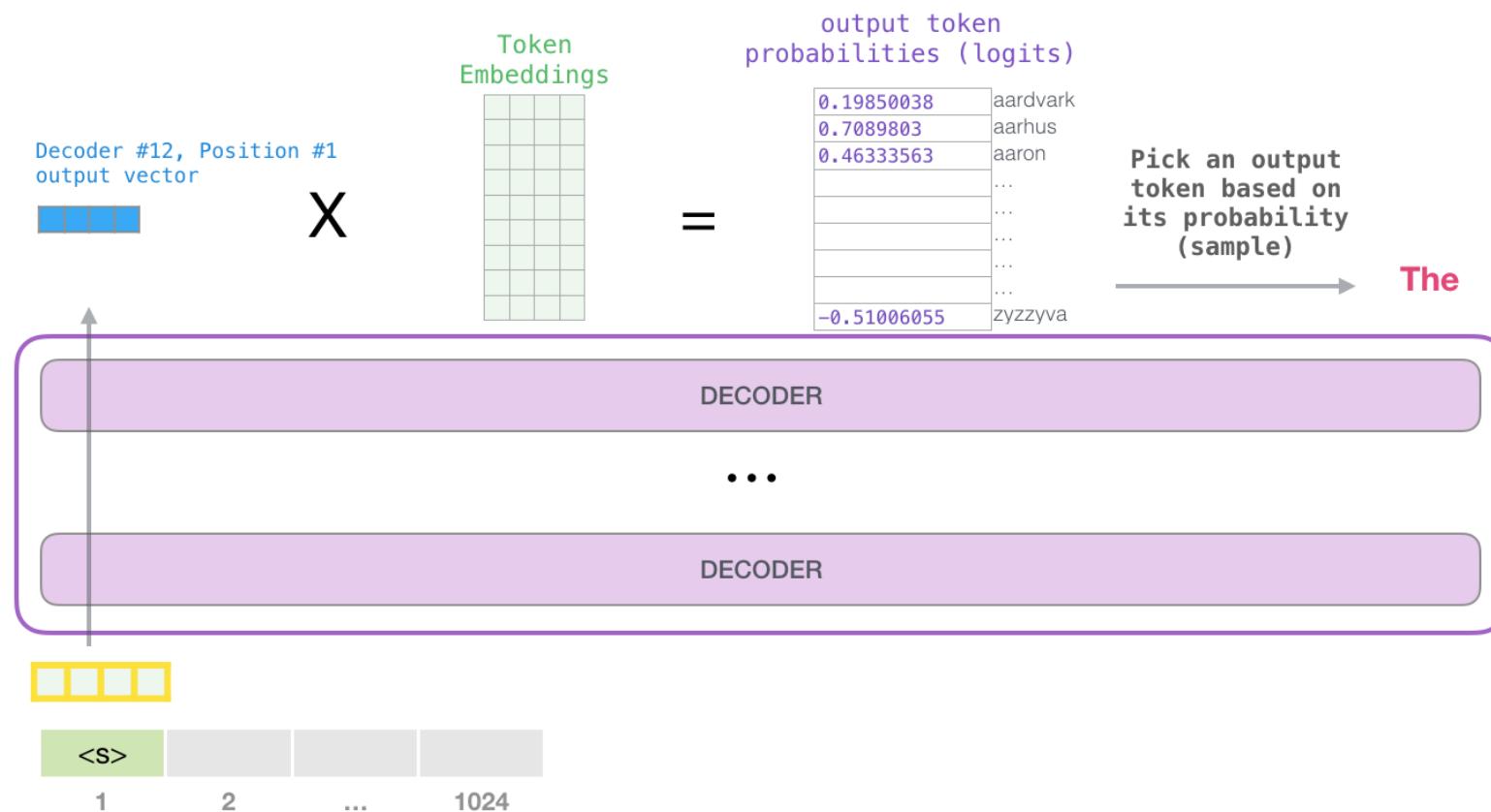
- Self-attention:



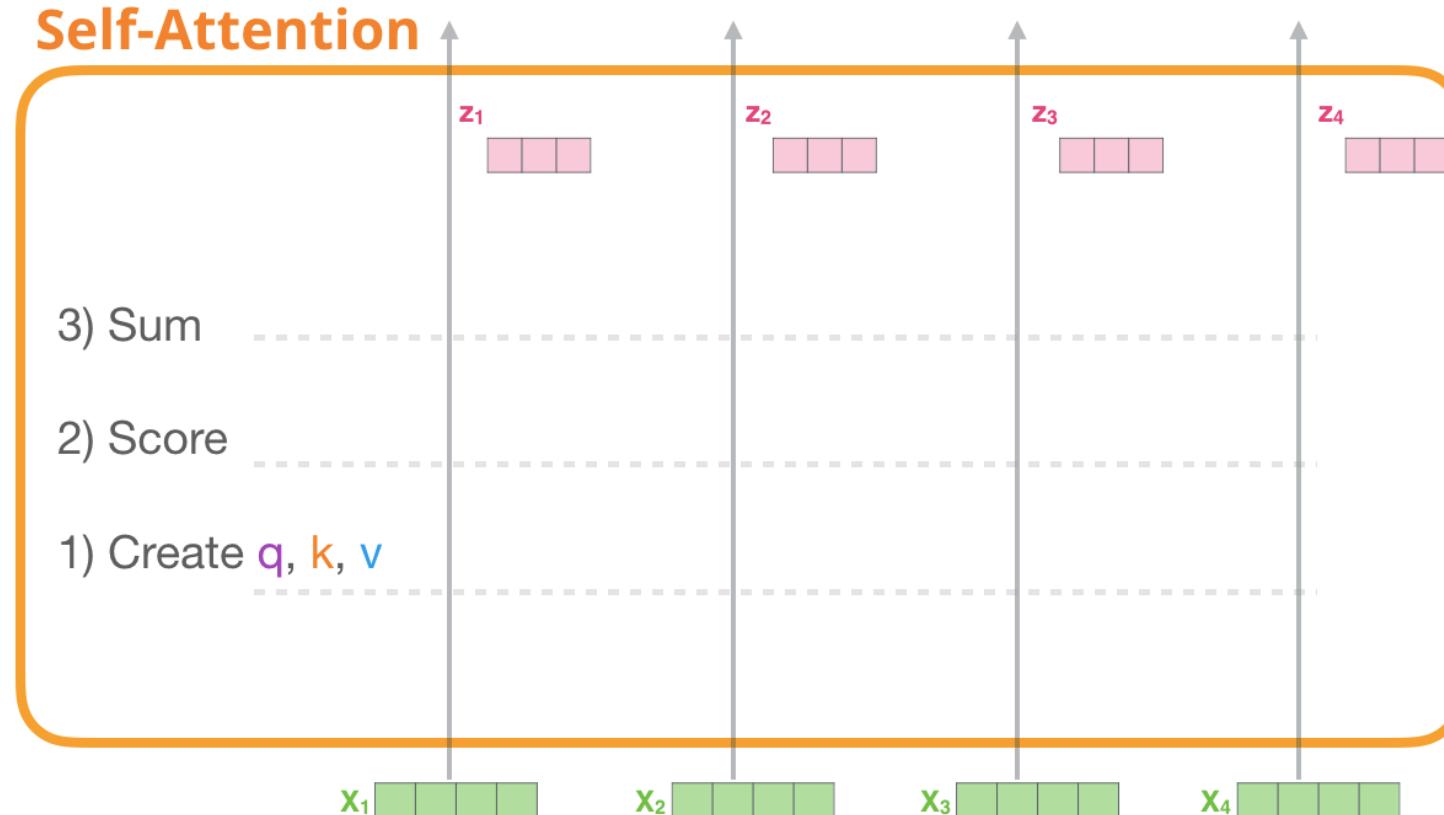
Word	Value vector	Score	Value X Score
<s>		0.001	
a		0.3	
robot		0.5	
must		0.002	
obey		0.001	
the		0.0003	
orders		0.005	
given		0.002	
it		0.19	
Sum:			

GPT Language Models – Inner Workings

- Output:

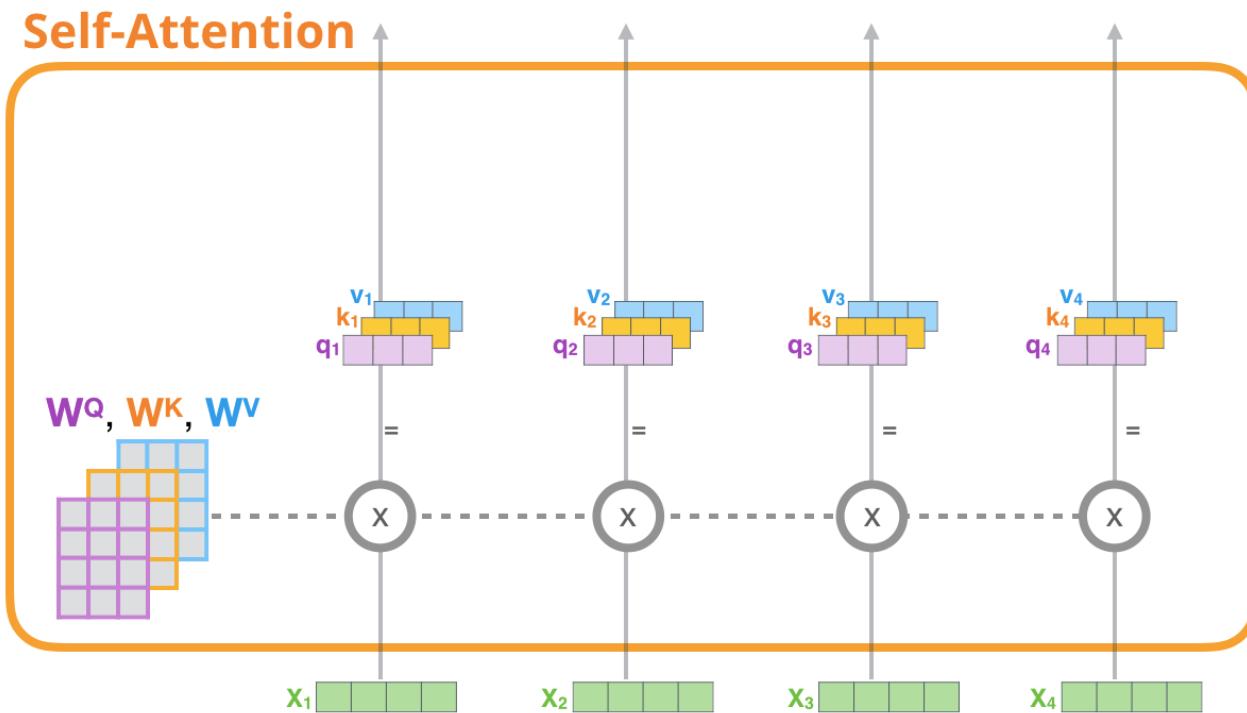


GPT Language Models – Going Deeper on Masked Self Attention



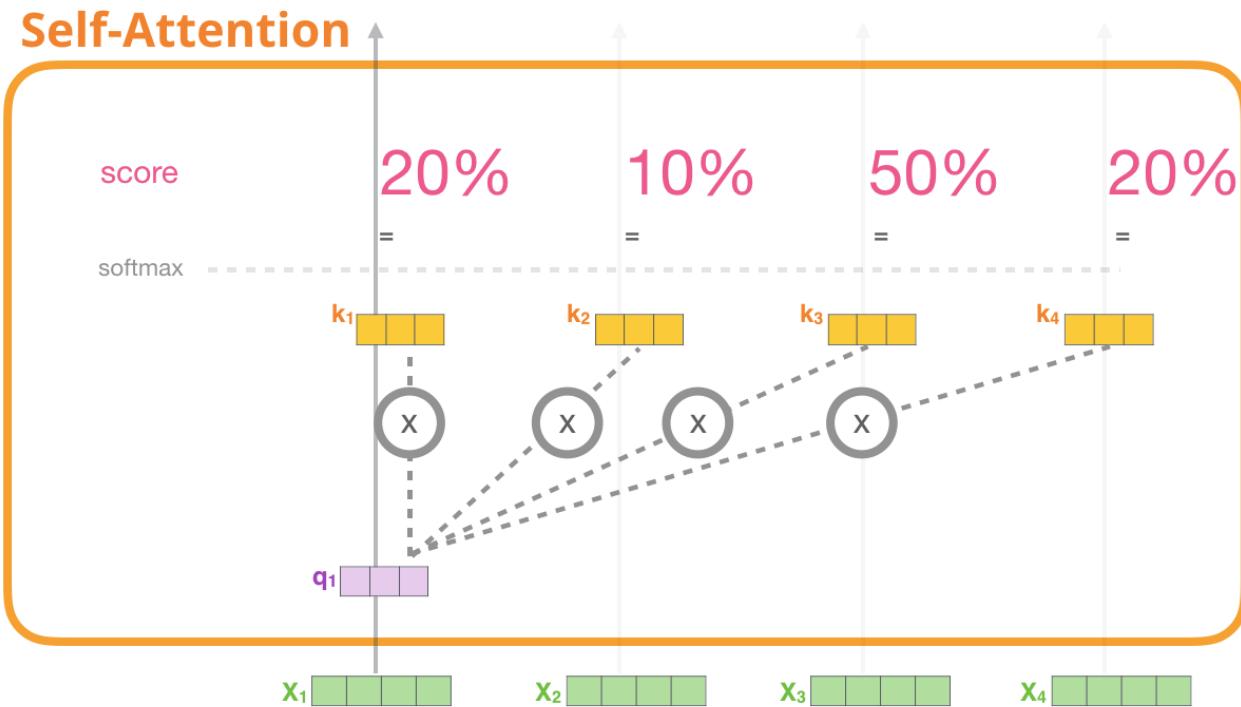
GPT Language Models – Going Deeper on Masked Self Attention

- 1) For each input token, create a **query vector**, a **key vector**, and a **value vector** by multiplying by weight Matrices \mathbf{W}^Q , \mathbf{W}^K , \mathbf{W}^V



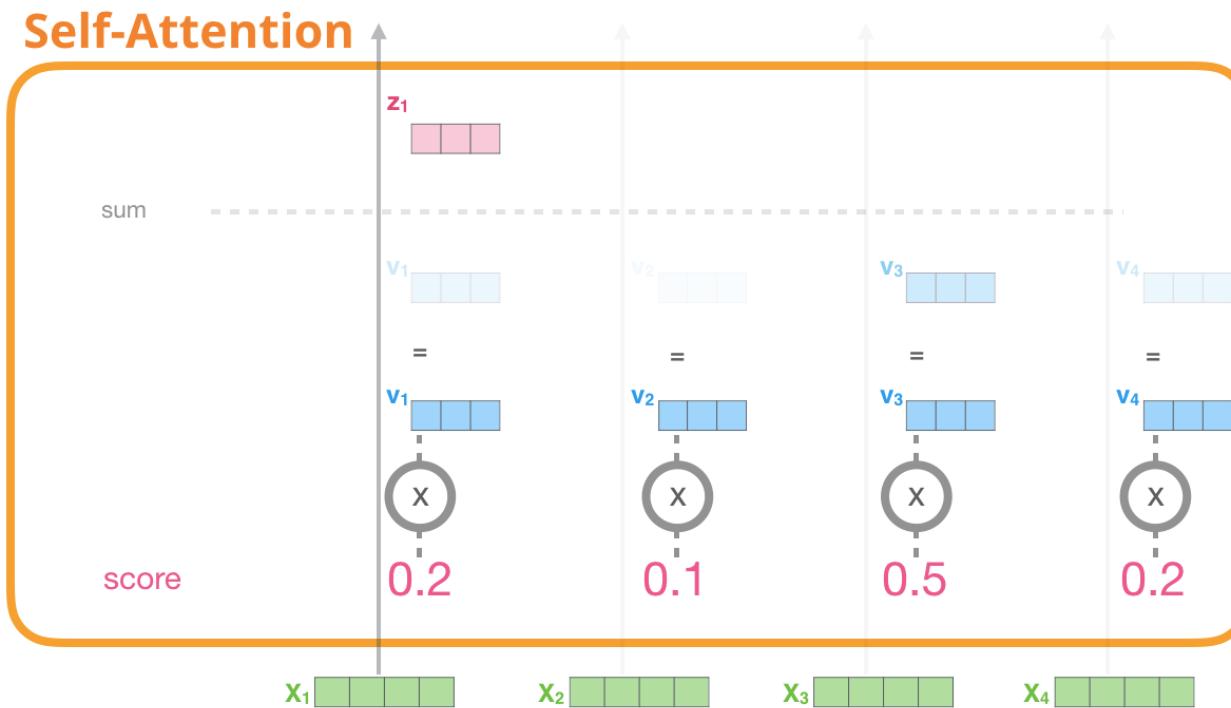
GPT Language Models – Going Deeper on Masked Self Attention

2) Multiply (dot product) the current **query vector**, by all the **key vectors**, to get a score of how well they match

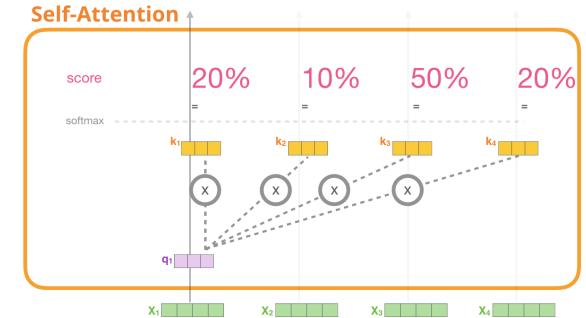
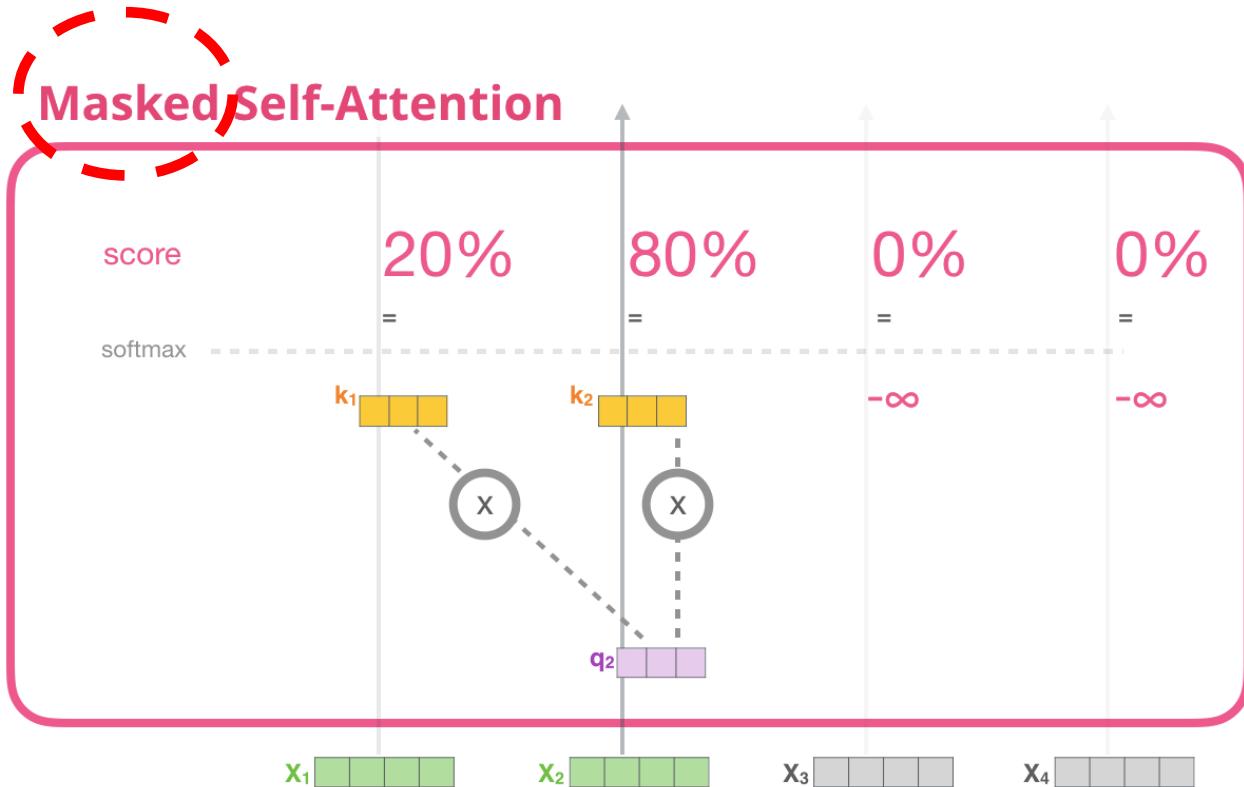


GPT Language Models – Going Deeper on Masked Self Attention

3) Multiply the **value vectors** by the **scores**, then sum up



GPT Language Models – Going Deeper on Masked Self Attention



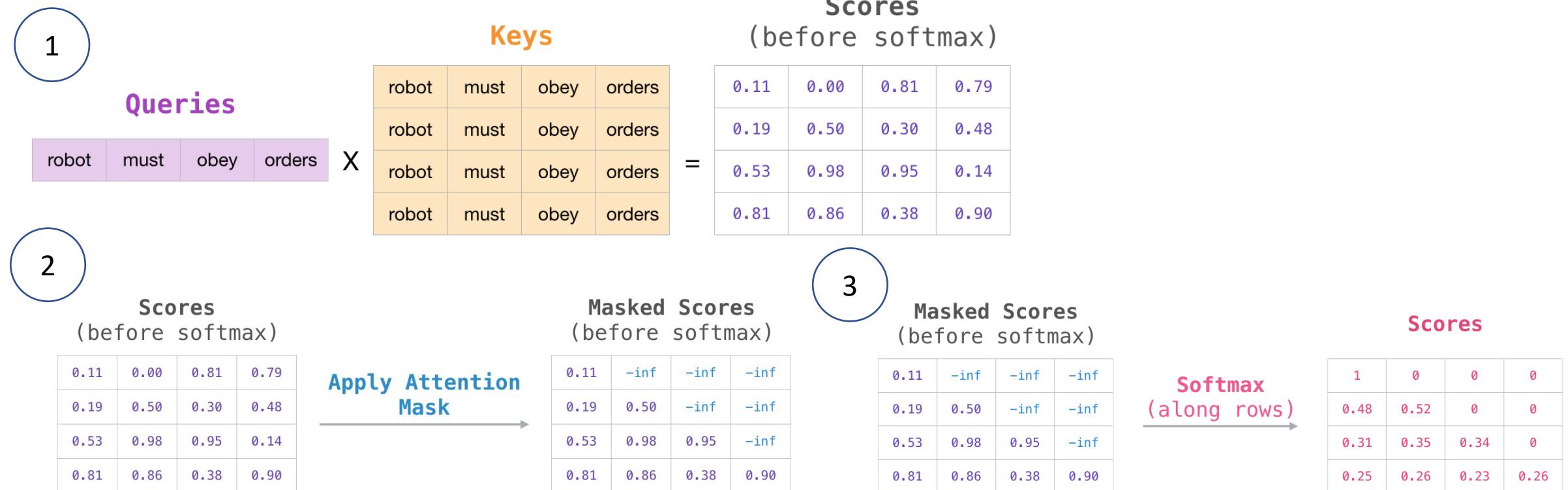
GPT Language Models – Going Deeper on Masked Self Attention

- During training: Batch processing of pairs of input and output

		Features				Labels
		position: 1	2	3	4	
Example:		robot	must	obey	orders	must
1		robot	must	obey	orders	
2		robot	must	obey	orders	obey
3		robot	must	obey	orders	orders
4		robot	must	obey	orders	<eos>

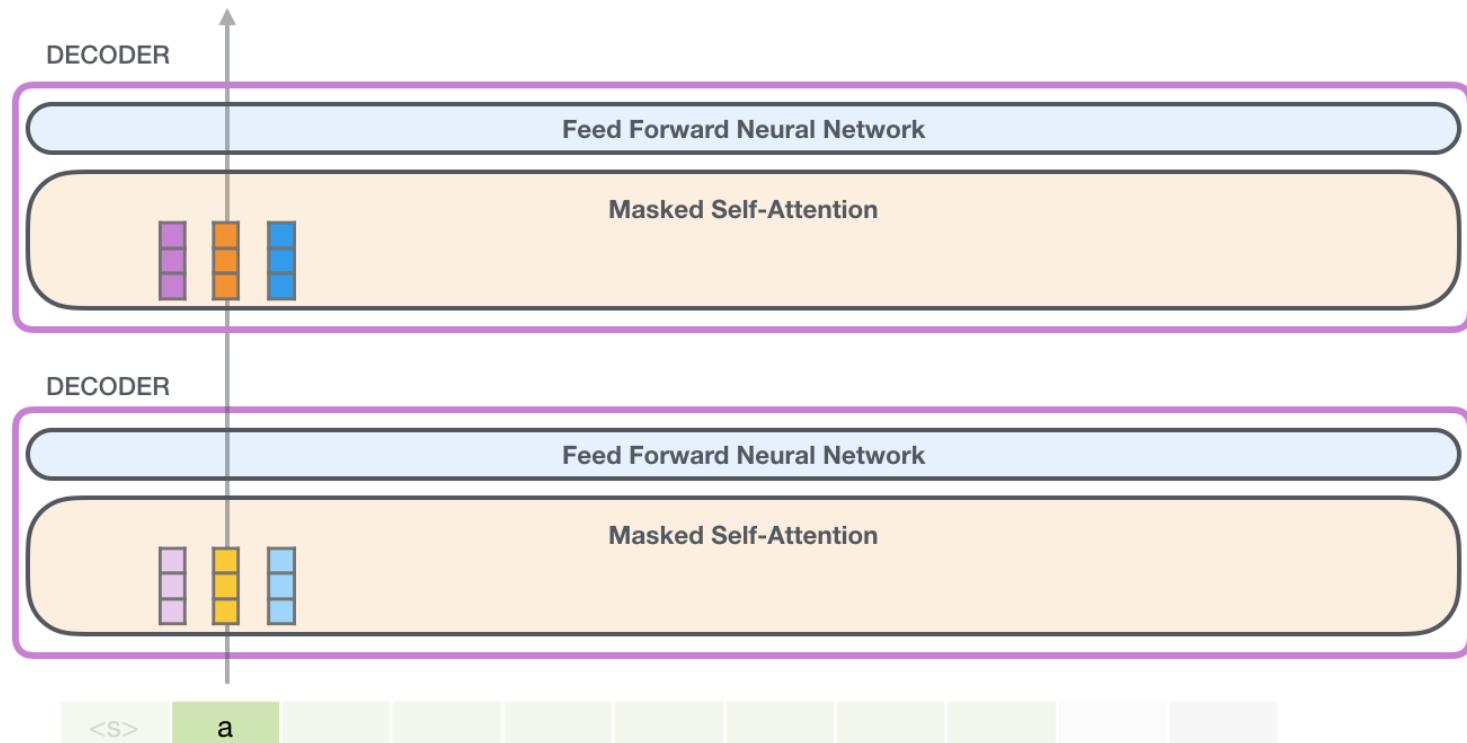
GPT Language Models – Going Deeper on Masked Self Attention

- During training: compute self attention scores



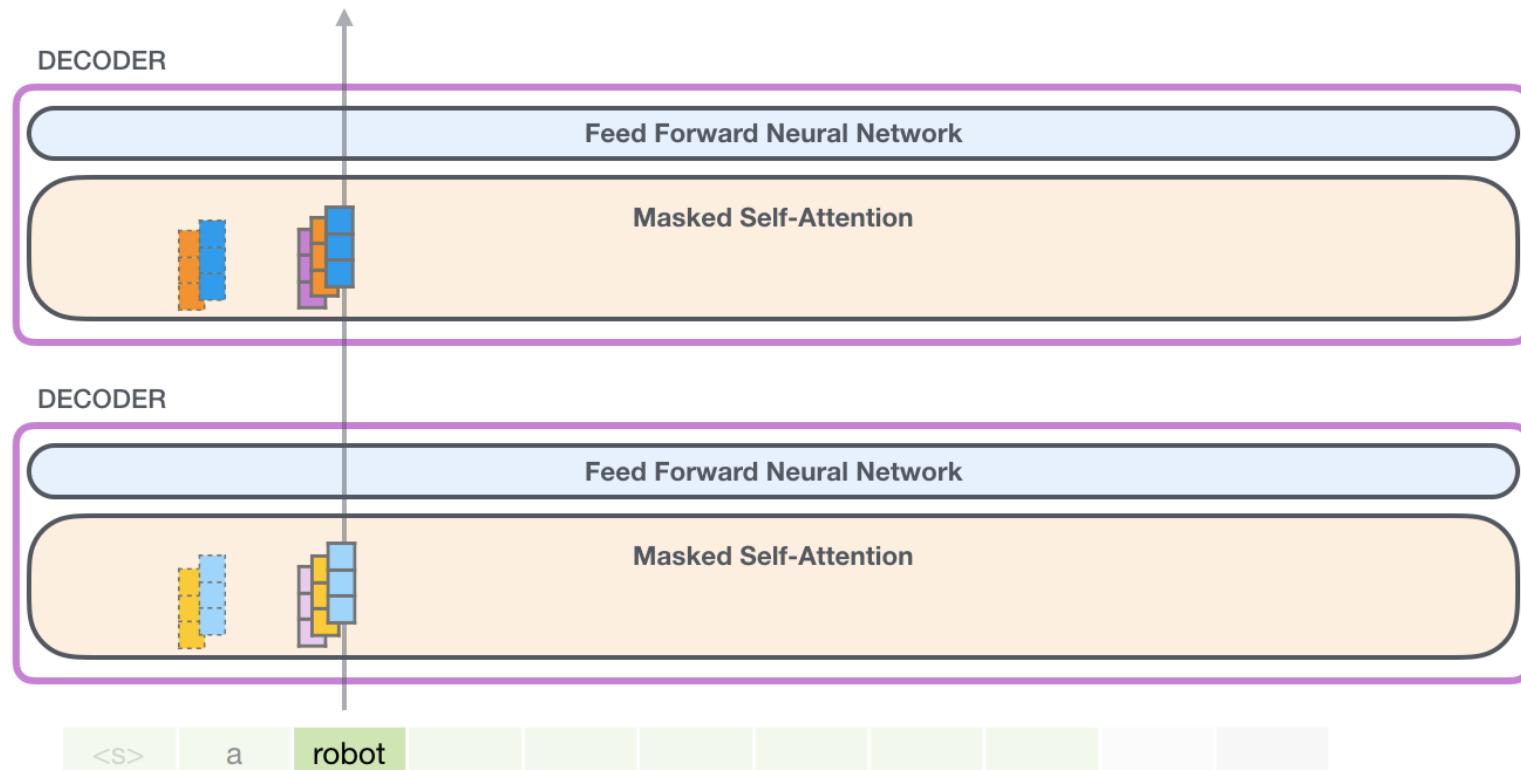
GPT Language Models – Going Deeper on Masked Self Attention

- During evaluation:



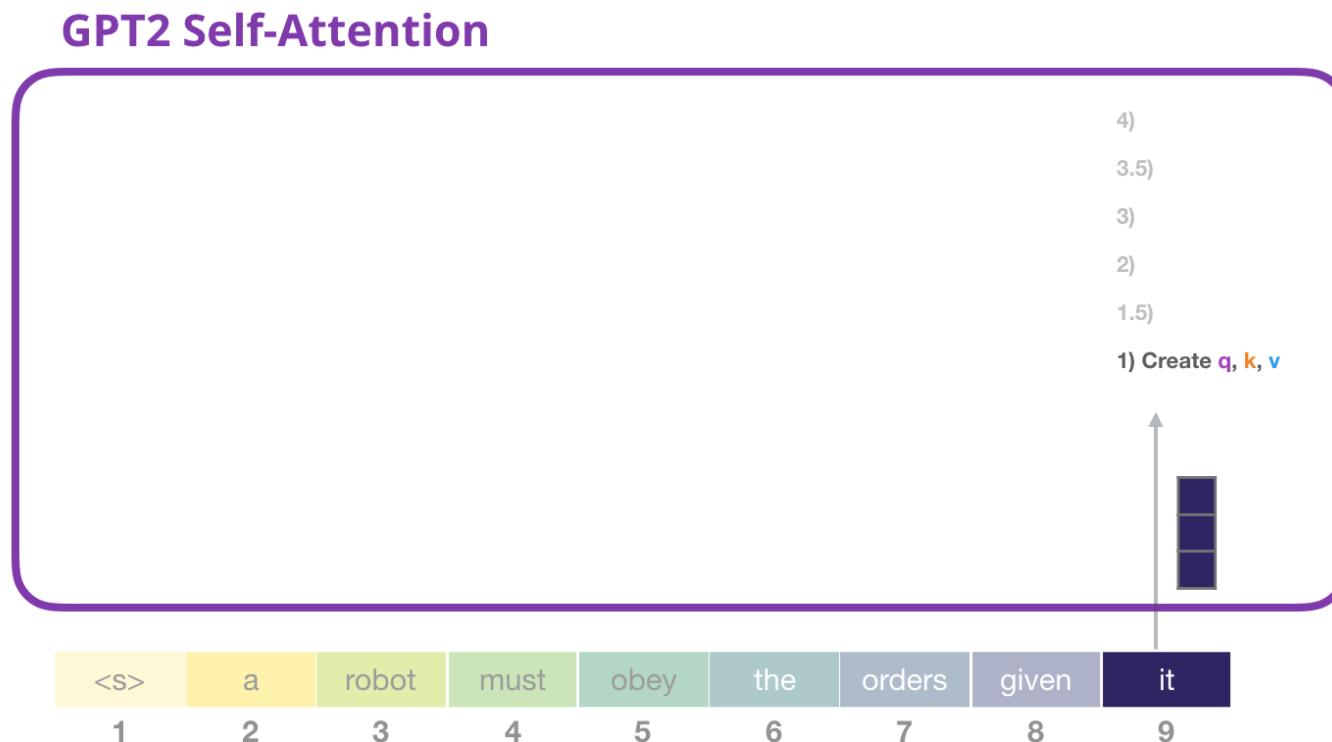
GPT Language Models – Going Deeper on Masked Self Attention

- During evaluation:



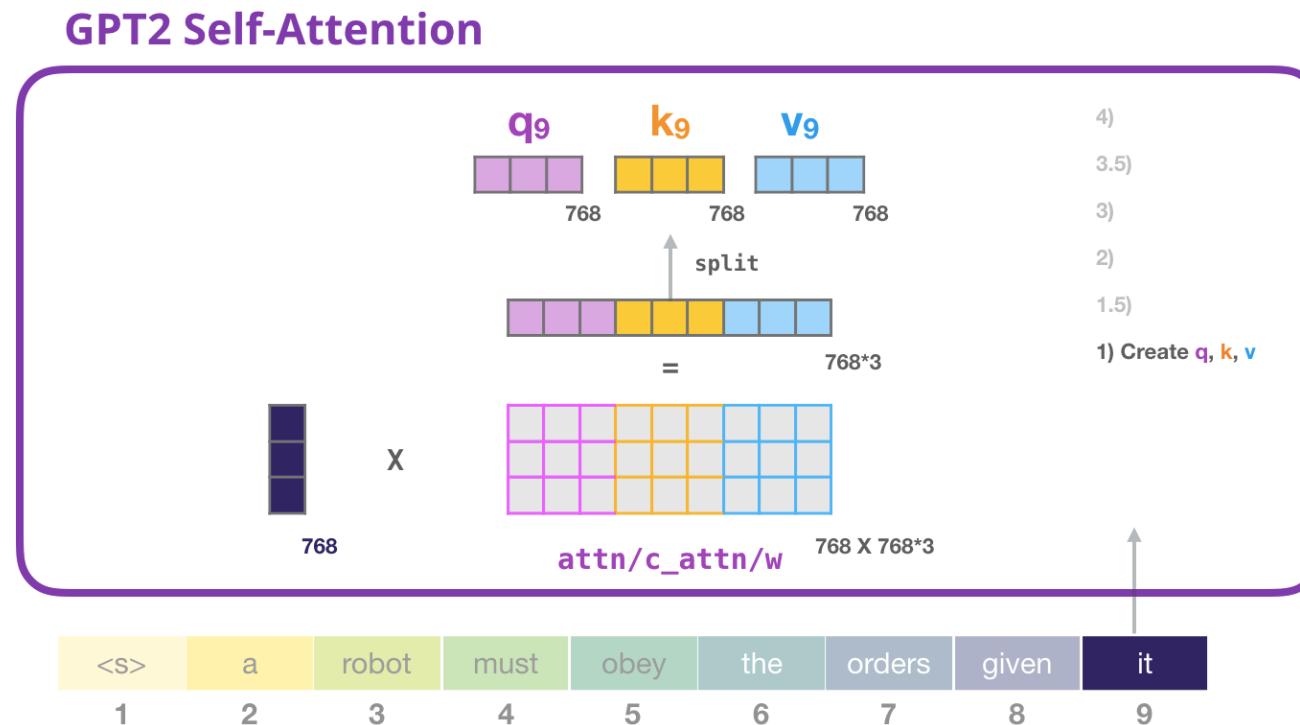
GPT Language Models – Going Deeper on Masked Self Attention

- Implementation: input (token embedding + positional embedding)



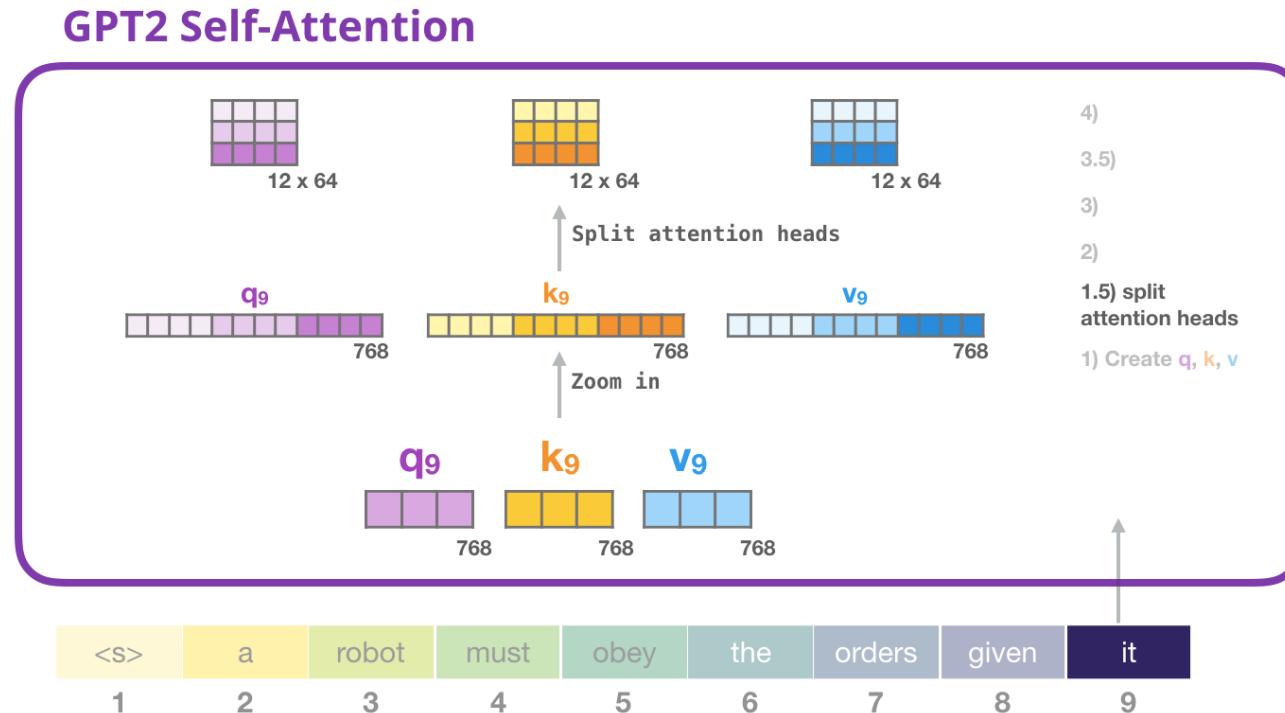
GPT Language Models – Going Deeper on Masked Self Attention

- Implementation: create query, key, value vectors



GPT Language Models – Going Deeper on Masked Self Attention

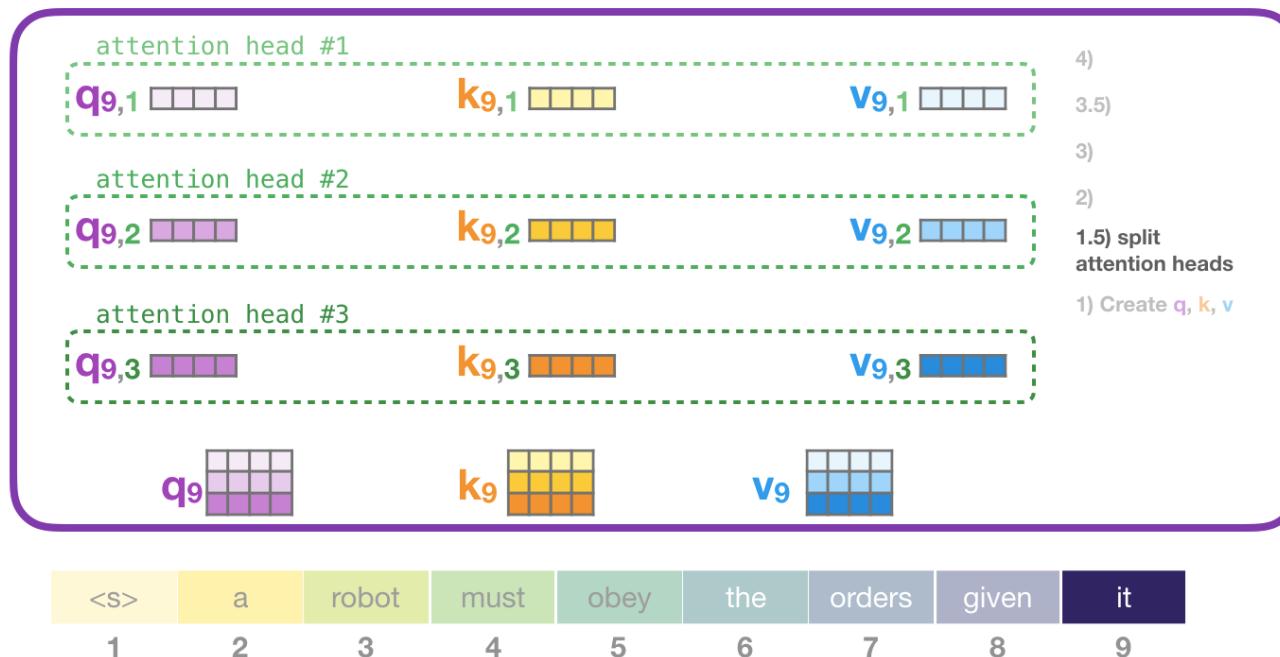
- Implementation: create query, key, value vectors for multiple attention heads



GPT Language Models – Going Deeper on Masked Self Attention

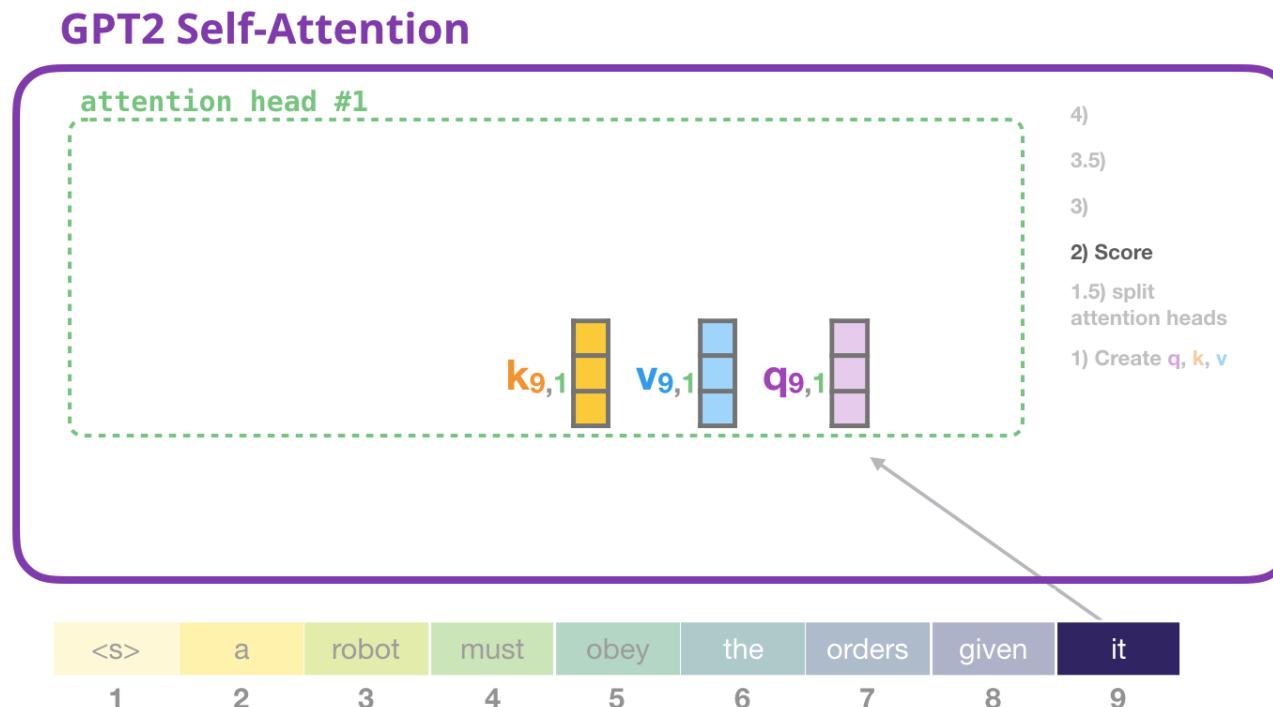
- Implementation: create query, key, value vectors for multiple attention heads

GPT2 Self-Attention



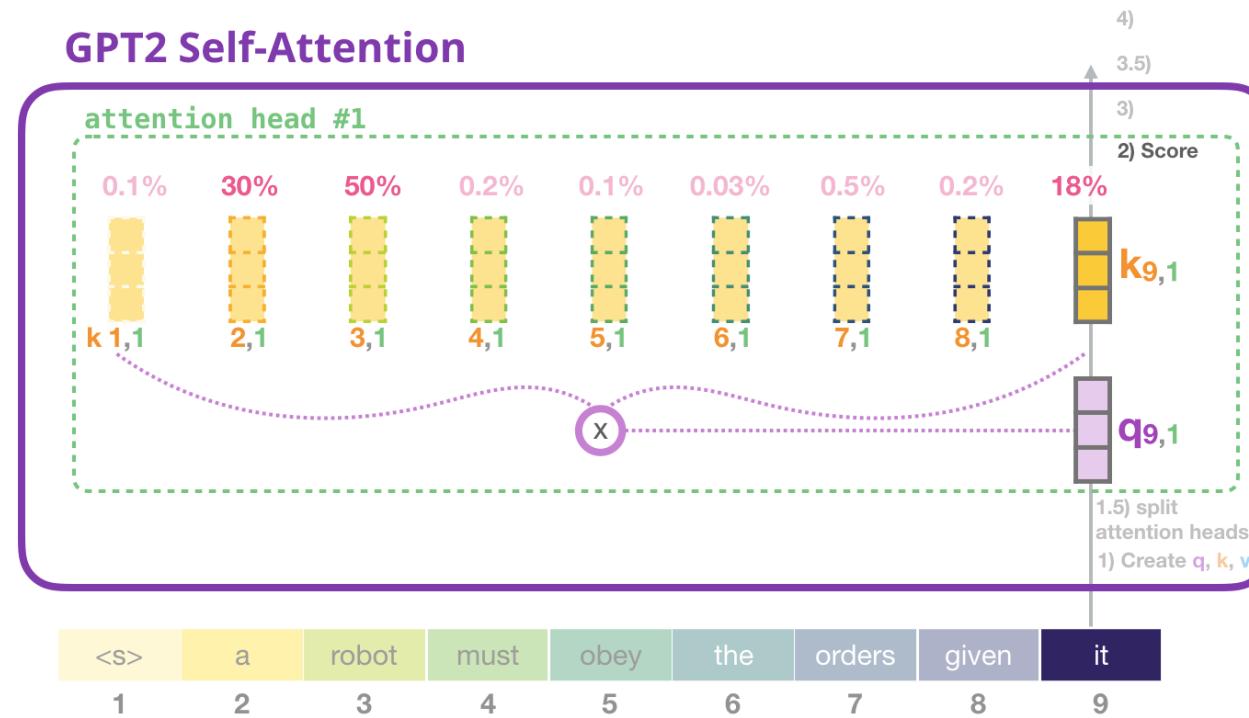
GPT Language Models – Going Deeper on Masked Self Attention

- Implementation: within each attention head, compute attention scores



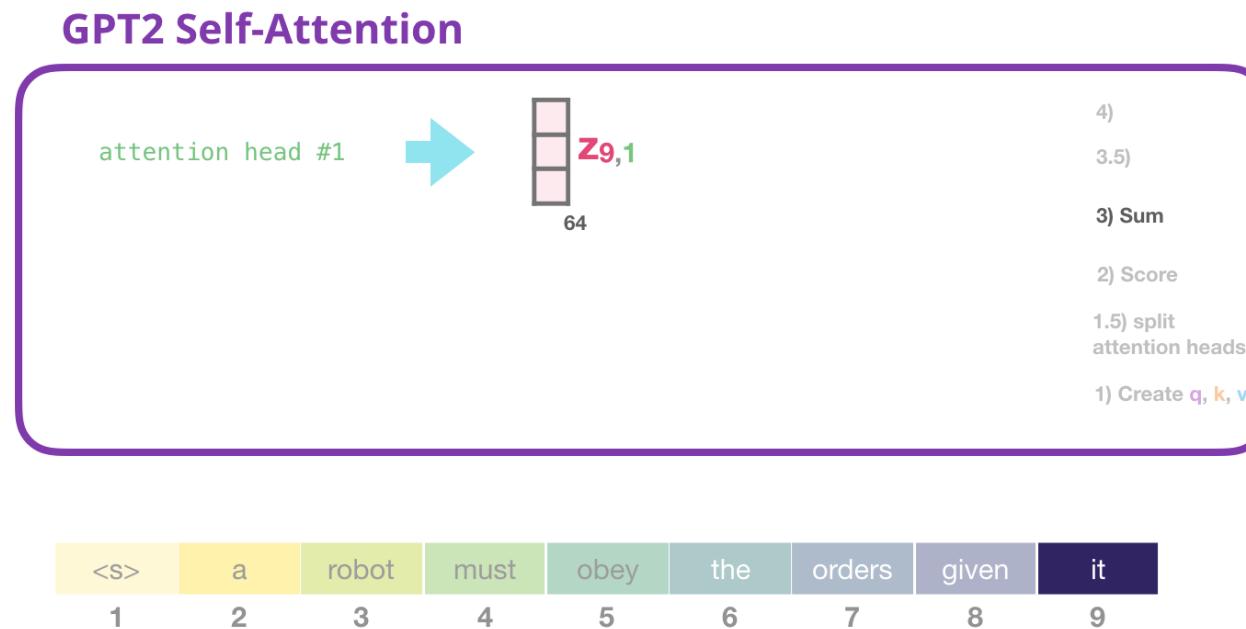
GPT Language Models – Going Deeper on Masked Self Attention

- Implementation: within each attention head, compute attention scores



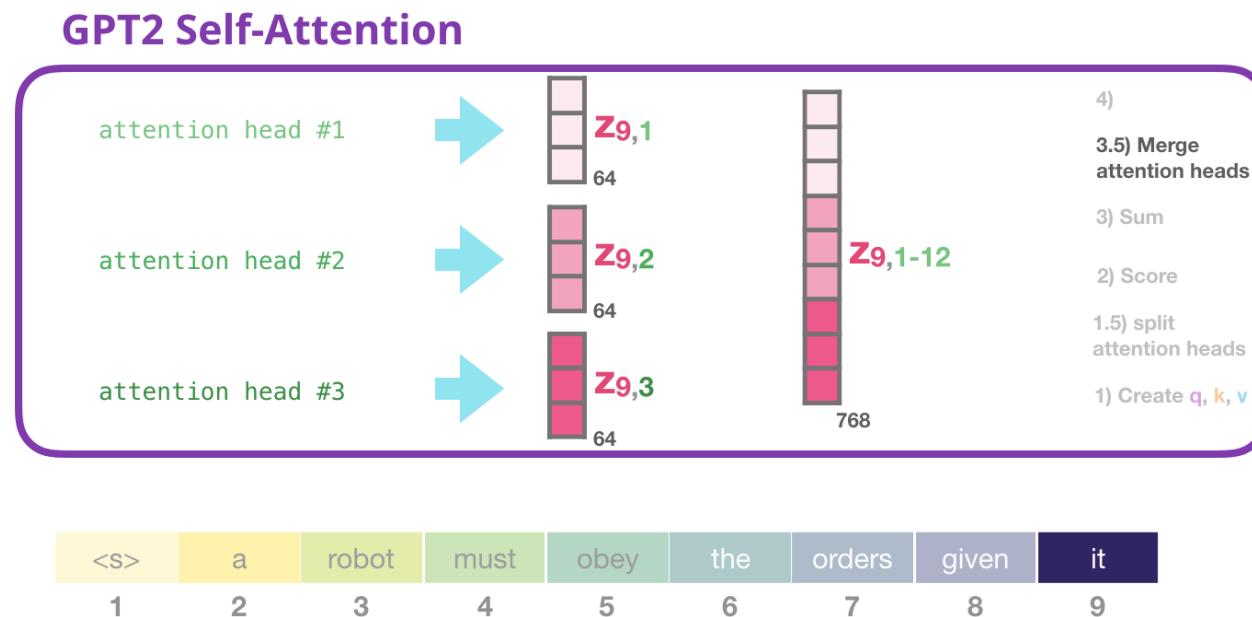
GPT Language Models – Going Deeper on Masked Self Attention

- Implementation: within each attention head, compute attention scores, and create a weighted sum of value vectors



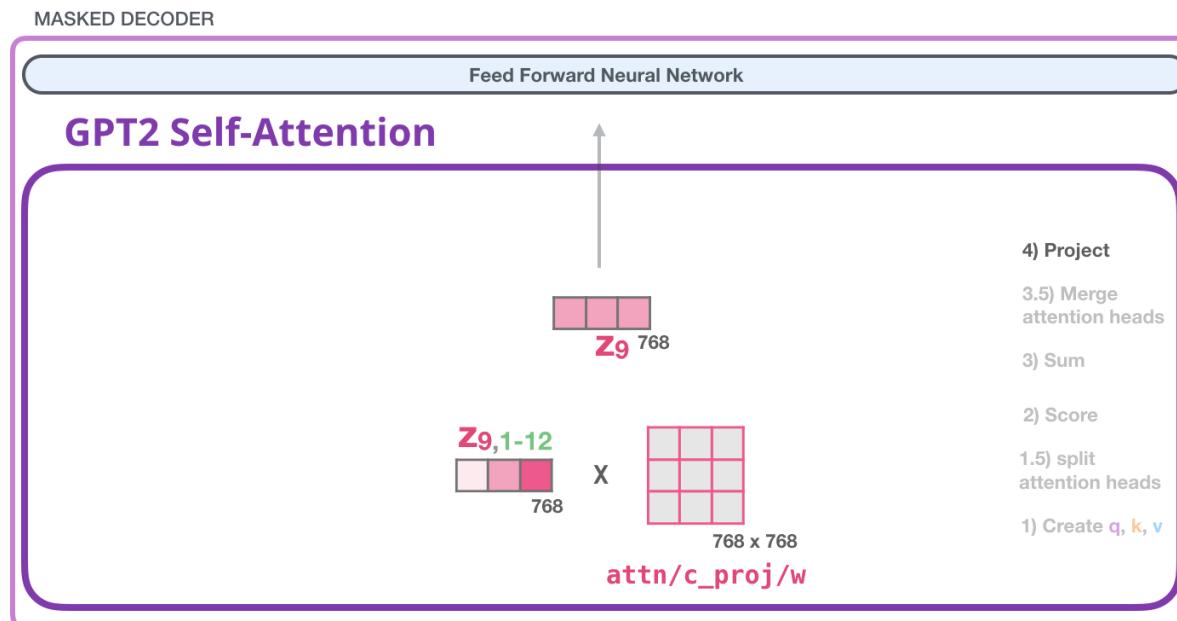
GPT Language Models – Going Deeper on Masked Self Attention

- Implementation: merge vectors from multiple heads



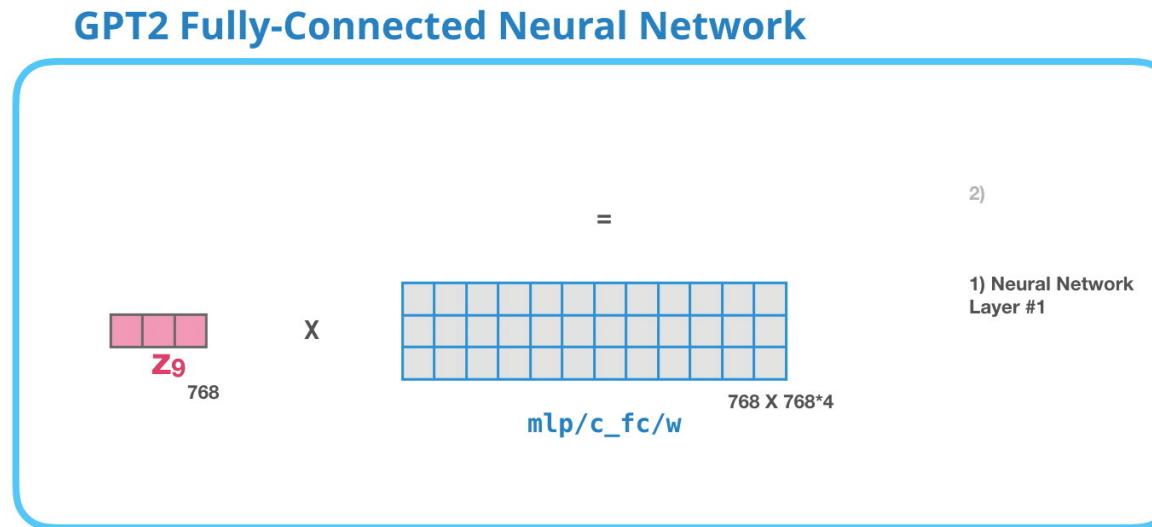
GPT Language Models – Going Deeper on Masked Self Attention

- Implementation: project the vectors and pass to the FFNN



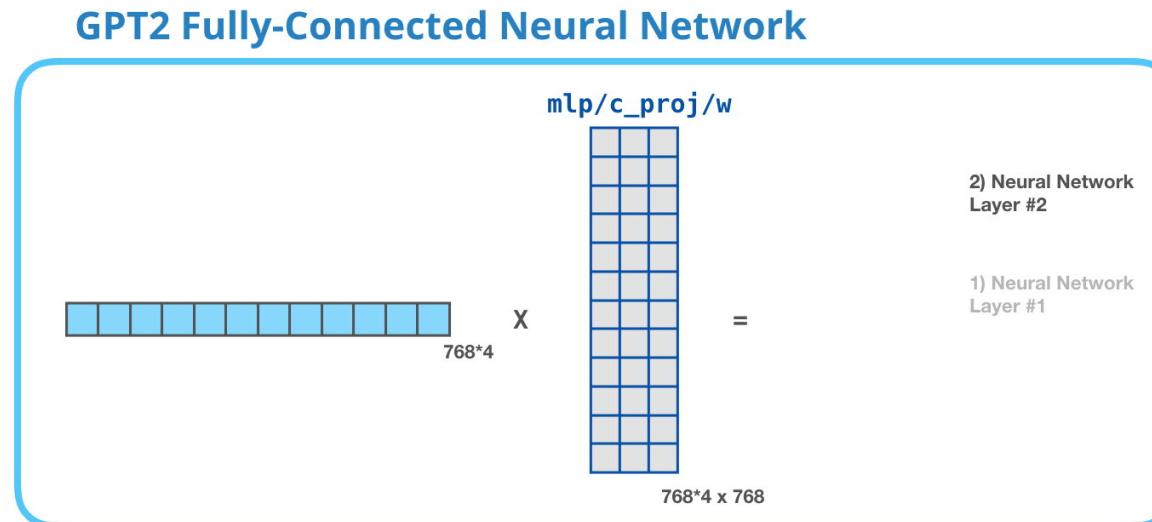
GPT Language Models – Going Deeper on Masked Self Attention

- Implementation: FFNN layer 1



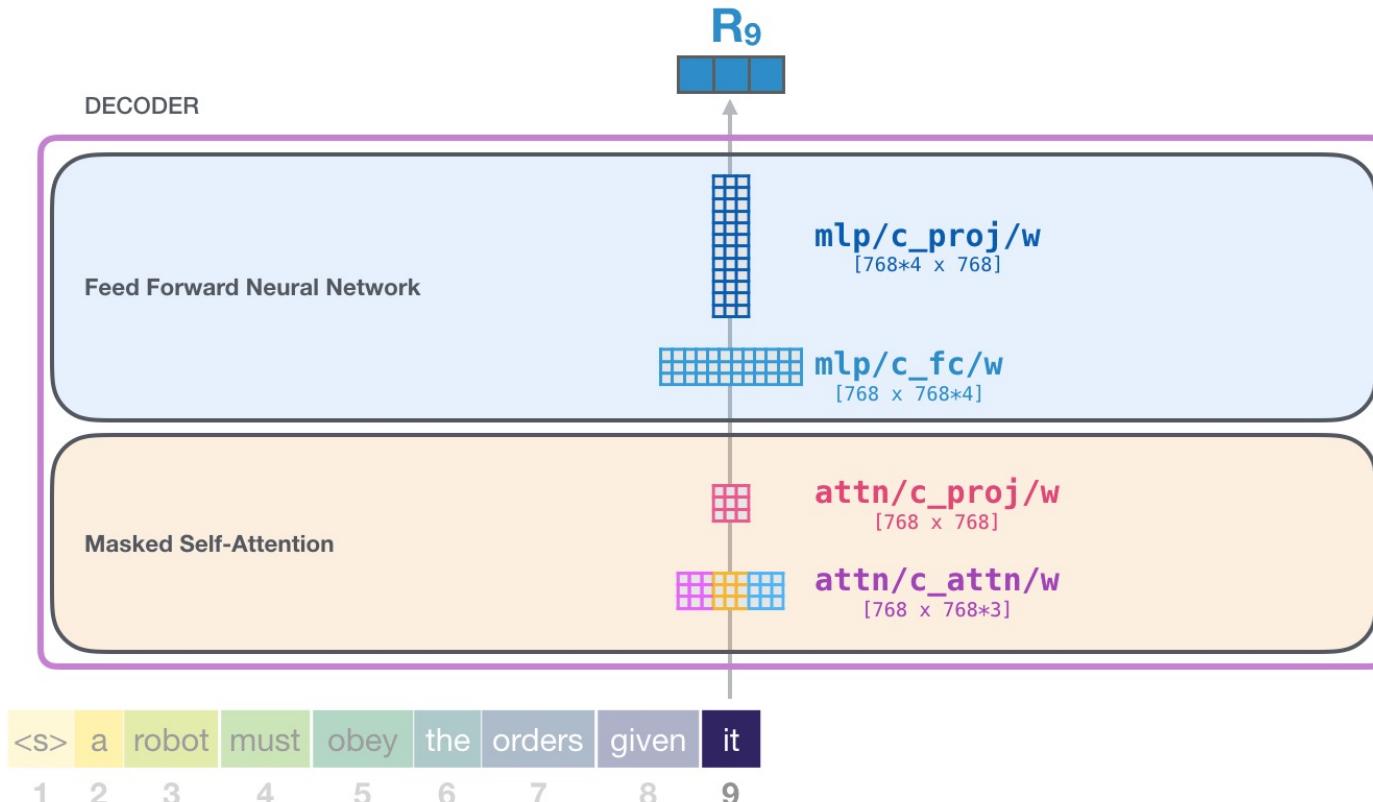
GPT Language Models – Going Deeper on Masked Self Attention

- Implementation: FFNN layer 2



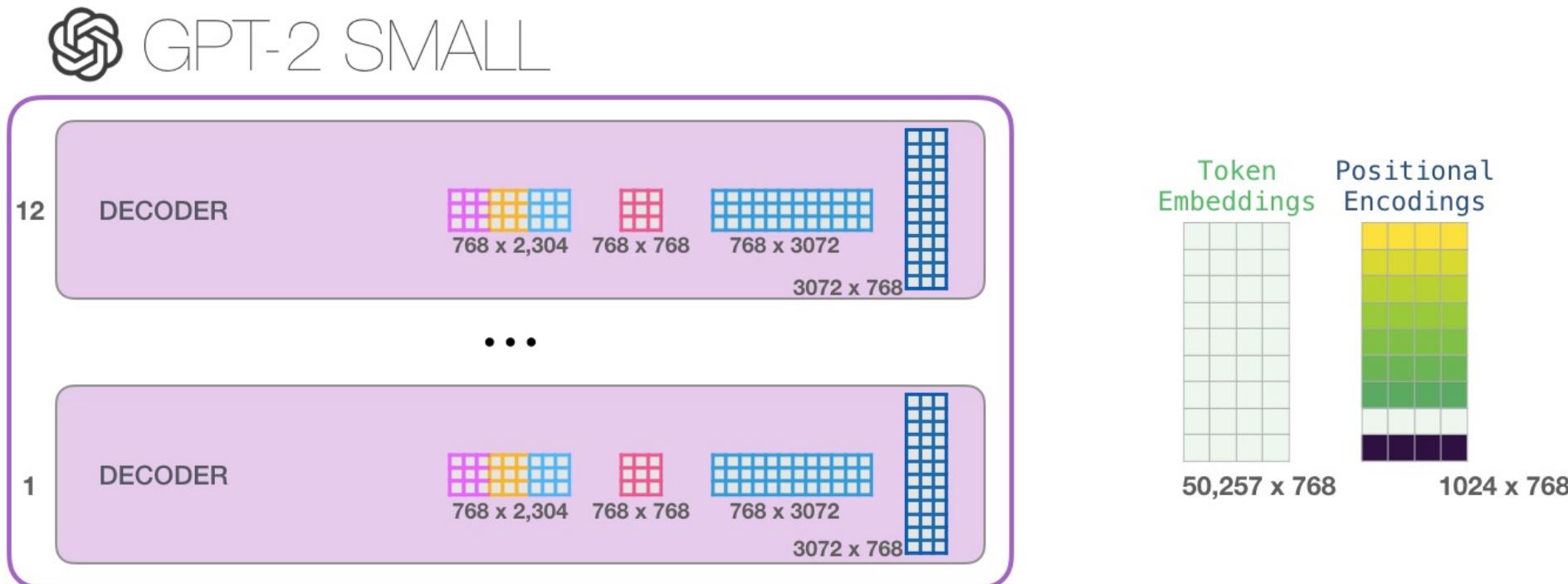
GPT Language Models – Going Deeper on Masked Self Attention

- Implementation Recap



GPT Language Models – Going Deeper on Masked Self Attention

- Implementation Recap

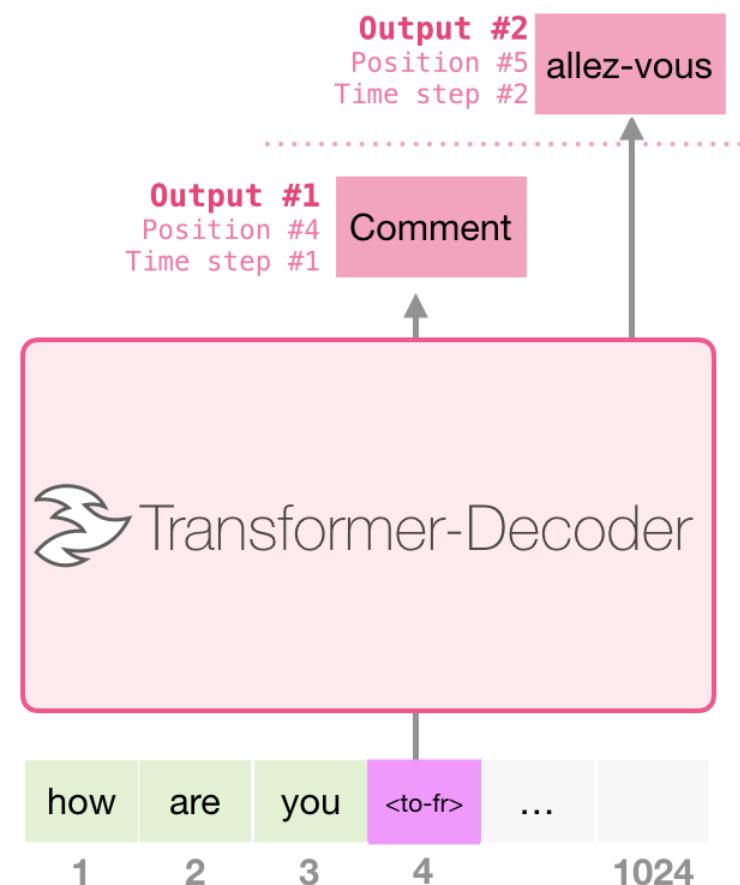


GPT Language Models – Application

- Machine translation (decoder only)

Training Dataset

I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				



GPT Language Models – Application

• Summarization

The image shows two side-by-side screenshots of a Wikipedia article titled "Positronic brain". Both screenshots show the same article content, but one includes a large yellow "SUMMARY" box.

Left Screenshot (Without Summary Box):

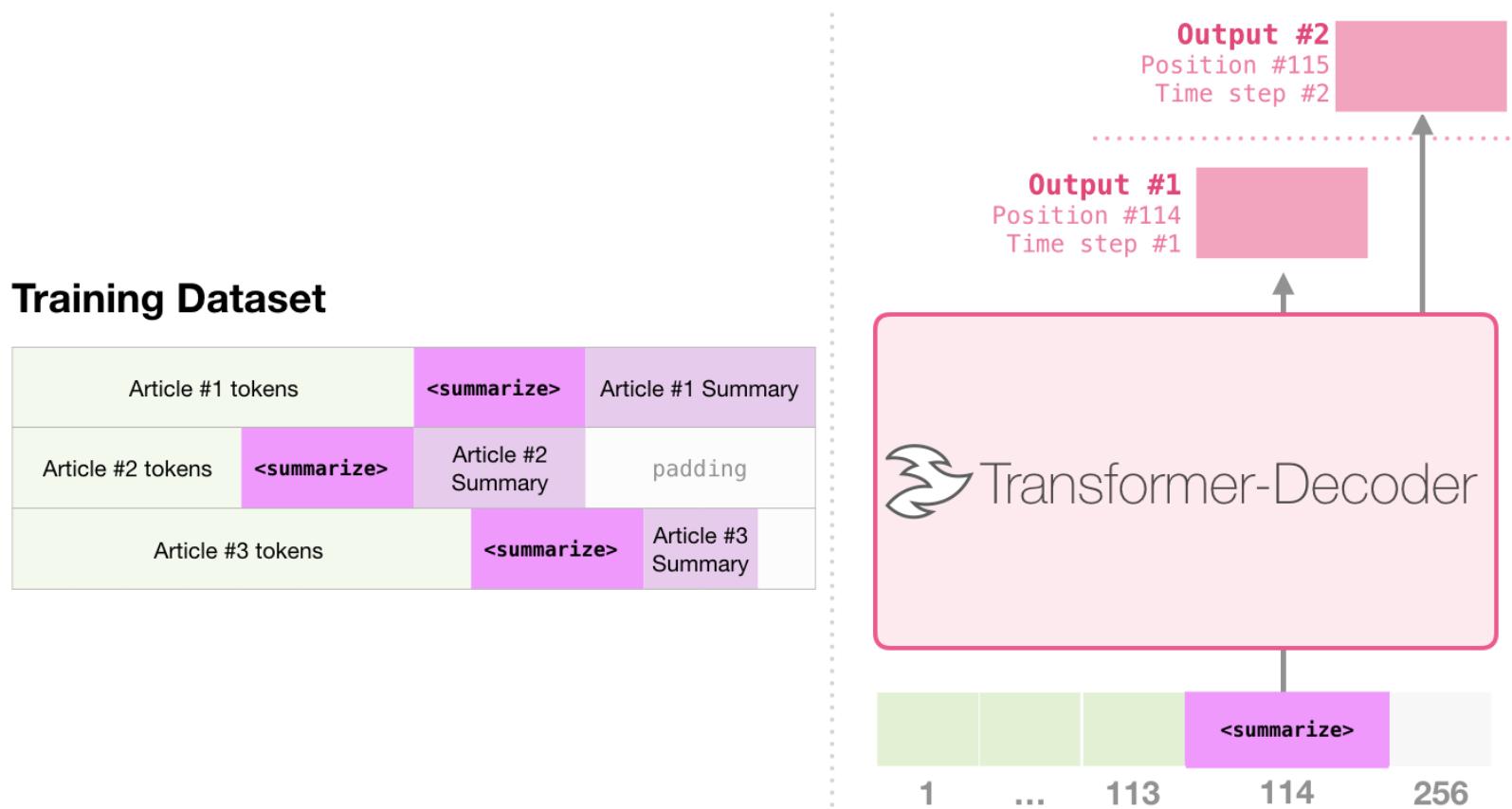
- Header: "Not logged in | Talk | Contributions | Create account | Log in"
- Page Title: "Positronic brain"
- Text: "This article is about a fictional technological device. For the manufacturing company based in Springfield, Missouri, see [Positronic \(company\)](#).
From Wikipedia, the free encyclopedia (vieweditprinthistory)
- Template: "This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed." (with a link to JSTOR)
- Section: "Conceptual overview" (with edit link)
- Text: "Asimov remained vague about the technical details of positronic brains except to assert that their substructure was formed from an alloy of platinum and iridium. They were said to be vulnerable to radiation and apparently involve a type of volatile memory (since robots in storage required a power source keeping their brain "alive"). The focus of Asimov's stories was directed more towards the software of robots—such as the Three Laws of Robotics—than the hardware in which it was implemented, although it is stated in his stories that to create a positronic brain without the Three Laws, it would have been necessary to spend years redesigning the fundamental approach towards the brain itself.
Within his stories of robotics on Earth and their development by U.S. Robots, Asimov's positronic brain is less of a plot device and more of a technological item worthy of study."
- Text: "A positronic brain cannot be built incorporating the Three Laws; any modification thereof would drastically modify robot behavior. Behavioral anomalies resulting from conflicting potentials set by inexperienced and/or malicious users of the robot for the Three Laws make up the bulk of Asimov's stories concerning robots. They are resolved by applying the science of logic and psychology together with mathematics, the supreme solution finder being Dr. Susan Calvin, Chief Robopsychologist of U.S. Robots."
- Text: "The Three Laws are also a bottleneck in brain sophistication. Very complex brains designed to handle world economy interpret the First Law in expanded sense to include humanity as opposed to a single human; in Asimov's later works like *Robots and Empire* this is referred to as the "Zerith Law". At least one brain constructed as a positronic circuitry renders a brain unusable if it could contaminate it within the skull of an insect. Under specific conditions, the Three Laws can be obviated, with the modification of the actual robotic design."
- List: "• Robots that are of low enough value can have the **Third Law deleted**; they do not have to protect themselves from harm, and the brain size can be reduced by half.
• Robots that do not require orders from a human being may have the **Second Law deleted**, and therefore require smaller brains again, providing they do not require the **Third Law**.
• Robots that are disposable, cannot receive orders from a human being and are not able to harm a human, will not require even the **First Law**. The sophistication of positronic circuitry renders a brain so small that it could contaminate the body of an insect if it was placed within the skull of an insect."
- Text: "Robots of the latter type directly parallel contemporary industrial robotics practice, though real-life robots do contain safety sensors and systems, in a concern for human safety (a weak form of the First Law, the robot is a safe tool to use, but has no "judgment", which is implicit in Asimov's own stories).
In Allen's trilogy [edit]
Several robot stories have been written by other authors following Asimov's death. For example, in Roger MacBride Allen's *Caliban's Trilogy*, a Spacer robot called Gubber Anubis invents the gravitonic brain. It offers speed and capacity improvements over traditional positronic designs, but the strong influence of tradition make robotics labs reject Anubis's work. Only one robotologist, Freddo Lening, chooses to adopt gravitronics, because it offers her a blank slate on which she could explore alternatives to the Three Laws. Because they are not dependent upon centuries of earlier research, gravitonic brains can be programmed with the standard Laws, variations of the Laws, or even empty pathways which specify no Laws at all."

Right Screenshot (With Summary Box):

- Header: "Not logged in | Talk | Contributions | Create account | Log in"
- Page Title: "Positronic brain"
- Text: "This article is about a fictional technological device. For the manufacturing company based in Springfield, Missouri, see [Positronic \(company\)](#).
(Redirected from [Positronic](#))
- Template: "This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed." (with a link to JSTOR)
- Section: "Conceptual overview" (with edit link)
- Text: "Asimov remained vague about the technical details of positronic brains except to assert that their substructure was formed from an alloy of platinum and iridium. They were said to be vulnerable to radiation and apparently involve a type of volatile memory (since robots in storage required a power source keeping their brain "alive"). The focus of Asimov's stories was directed more towards the software of robots—such as the Three Laws of Robotics—than the hardware in which it was implemented, although it is stated in his stories that to create a positronic brain without the Three Laws, it would have been necessary to spend years redesigning the fundamental approach towards the brain itself.
Within his stories of robotics on Earth and their development by U.S. Robots, Asimov's positronic brain is less of a plot device and more of a technological item worthy of study."
- Text: "A positronic brain cannot be built incorporating the Three Laws; any modification thereof would drastically modify robot behavior. Behavioral anomalies resulting from conflicting potentials set by inexperienced and/or malicious users of the robot for the Three Laws make up the bulk of Asimov's stories concerning robots. They are resolved by applying the science of logic and psychology together with mathematics, the supreme solution finder being Dr. Susan Calvin, Chief Robopsychologist of U.S. Robots."
- Text: "The Three Laws are also a bottleneck in brain sophistication. Very complex brains designed to handle world economy interpret the First Law in expanded sense to include humanity as opposed to a single human; in Asimov's later works like *Robots and Empire* this is referred to as the "Zerith Law". At least one brain constructed as a positronic circuitry renders a brain unusable if it could contaminate the body of an insect. Under specific conditions, the Three Laws can be obviated, with the modification of the actual robotic design."
- List: "• Robots that are of low enough value can have the **Third Law deleted**; they do not have to protect themselves from harm, and the brain size can be reduced by half.
• Robots that do not require orders from a human being may have the **Second Law deleted**, and therefore require smaller brains again, providing they do not require the **Third Law**.
• Robots that are disposable, cannot receive orders from a human being and are not able to harm a human, will not require even the **First Law**. The sophistication of positronic circuitry renders a brain so small that it could contaminate the body of an insect if it was placed within the skull of an insect."
- Text: "Robots of the latter type directly parallel contemporary industrial robotics practice, though real-life robots do contain safety sensors and systems, in a concern for human safety (a weak form of the First Law, the robot is a safe tool to use, but has no "judgment", which is implicit in Asimov's own stories).
In Allen's trilogy [edit]
Several robot stories have been written by other authors following Asimov's death. For example, in Roger MacBride Allen's *Caliban's Trilogy*, a Spacer robot called Gubber Anubis invents the gravitonic brain. It offers speed and capacity improvements over traditional positronic designs, but the strong influence of tradition make robotics labs reject Anubis's work. Only one robotologist, Freddo Lening, chooses to adopt gravitronics, because it offers her a blank slate on which she could explore alternatives to the Three Laws. Because they are not dependent upon centuries of earlier research, gravitonic brains can be programmed with the standard Laws, variations of the Laws, or even empty pathways which specify no Laws at all."

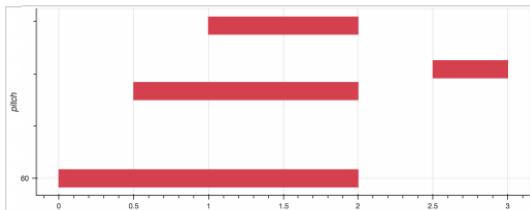
GPT Language Models – Application

- Summarization

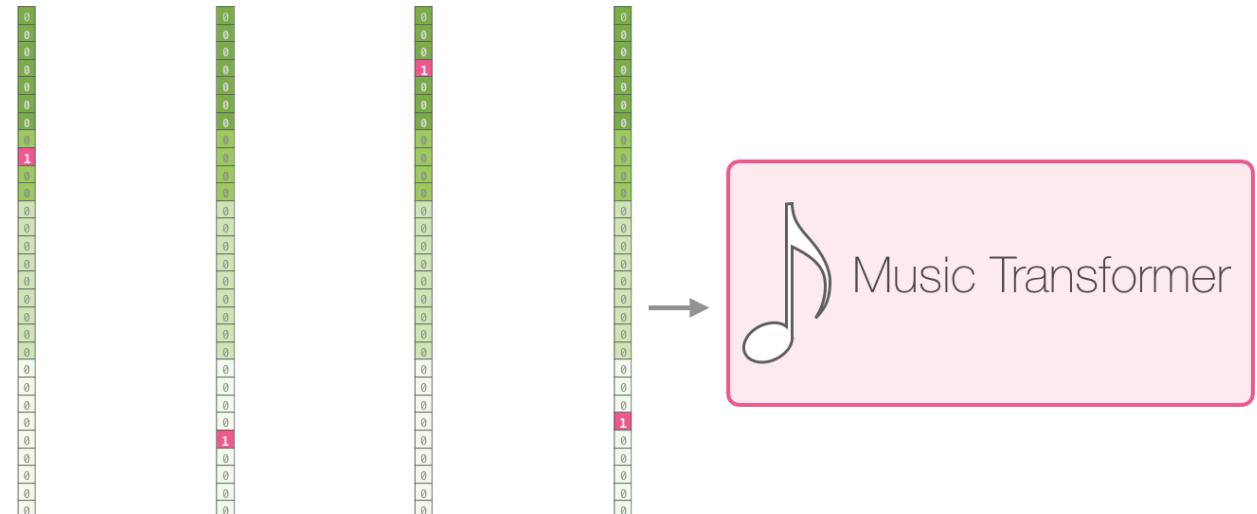


GPT Language Models – Application

- Music Transformer: <https://magenta.tensorflow.org/music-transformer>



```
SET_VELOCITY<80>, NOTE_ON<60>
TIME_SHIFT<500>, NOTE_ON<64>
TIME_SHIFT<500>, NOTE_ON<67>
TIME_SHIFT<1000>, NOTE_OFF<60>, NOTE_OFF<64>,
NOTE_OFF<67>
TIME_SHIFT<500>, SET_VELOCITY<100>, NOTE_ON<65>
TIME_SHIFT<500>, NOTE_OFF<65>
```



SET_VELOCITY<80>, NOTE_ON<60>, TIME_SHIFT<500>, NOTE_ON<64>

Try out GPTs

- GPT2: Decoder only transformer
<http://jalammar.github.io/illustrated-gpt2/>
- Try out GPT3 yourself: <https://beta.openai.com/playground/> (requires signing up)