Attention Mechanism and Transformer

CS4742 Natural Language Processing
Lecture 07

Jiho Noh

Department of Computer Science
Kennesaw State University

CS4742 Summer 2025

Topics

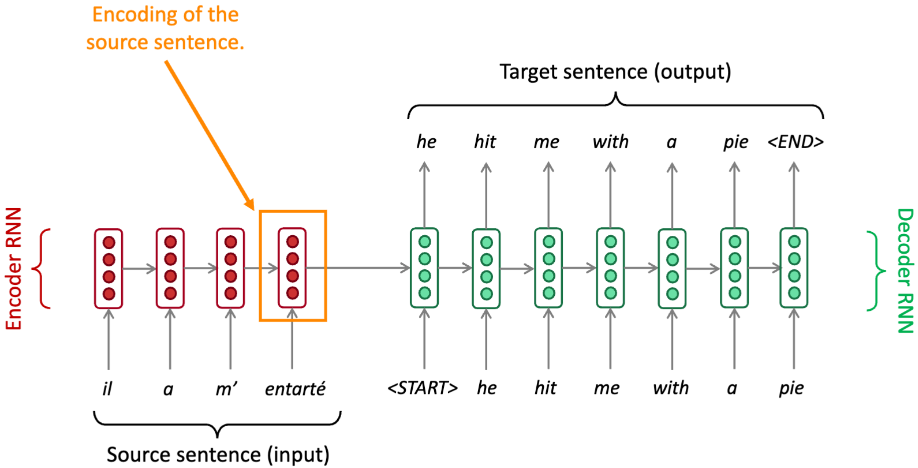
1. **Attention Mechanism**
2. Transformer Network
3. Pre-trained Language Models
   - BERT
   - GPT to ChatGPT

What is Attention Mechanism

- Techniques used to focus on specific parts of input data.
- Attention **assigns weights** (or probabilities) to different parts of the input and selectively attend to the most relevant information.
- Commonly used in deep learning models, such as Transformer architecture, for natural language processing, image captioning, and other sequence-to-sequence models.
- Attention uses the notions of Query, Key, Value.

Attention Mechanism (Query, Key, Value)

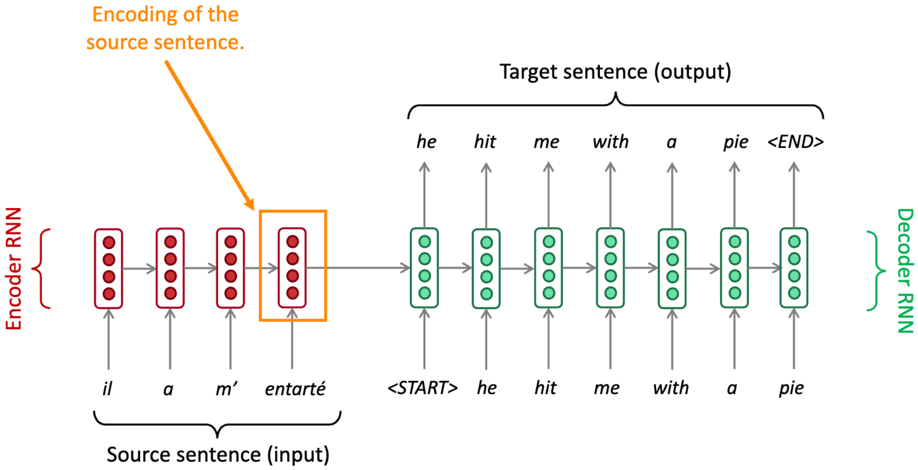
- What are the query, key, value vectors?
- These concepts come from retrieval systems. To make an analogy,
  - You type a **query** to search for some video on YouTube
  - The search engine will map your **query** against a set of **keys** associated with candidate videos in the database
  - Then, present you the best matched videos **(values)**.



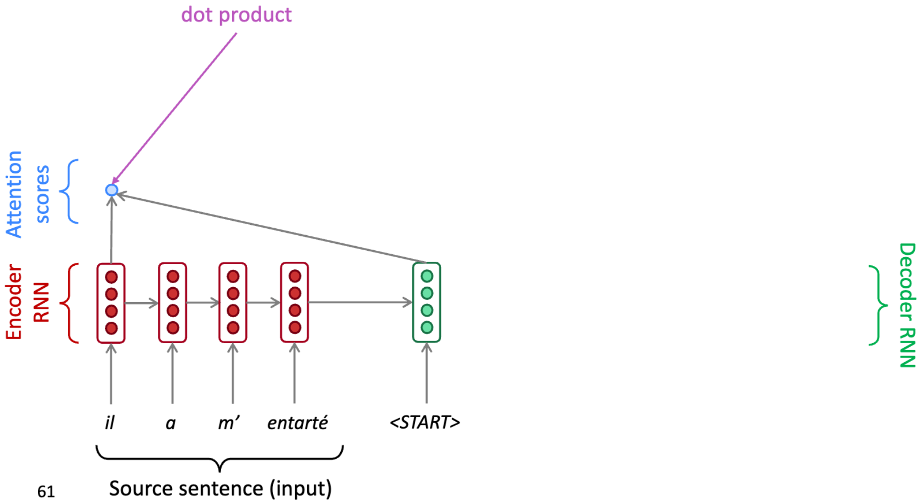
Problems with this architecture?

1

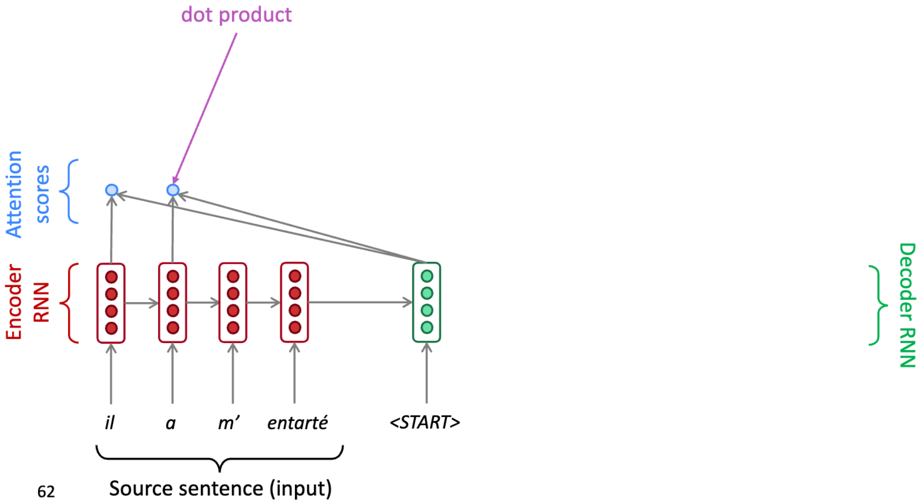
[1]Slides credit: Abigail See and https://www.site.uottawa.ca/~diana/csi5386/NMT.pdf

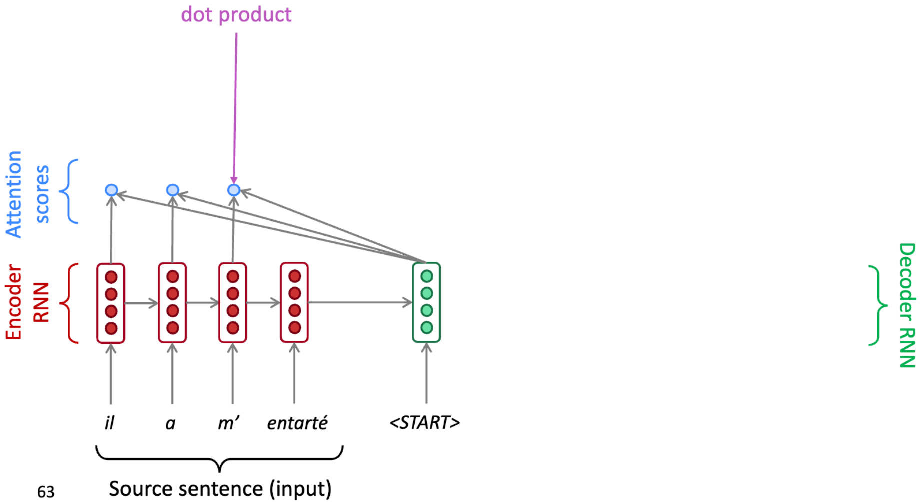


Problems with this architecture?

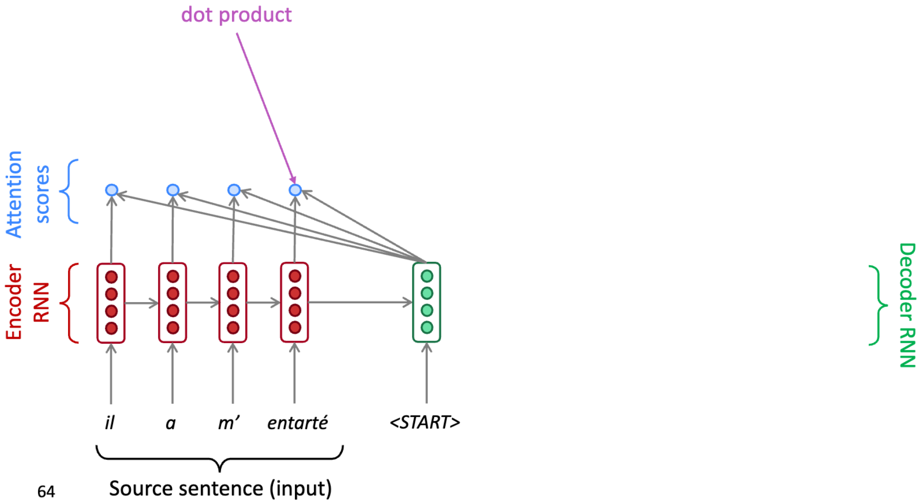


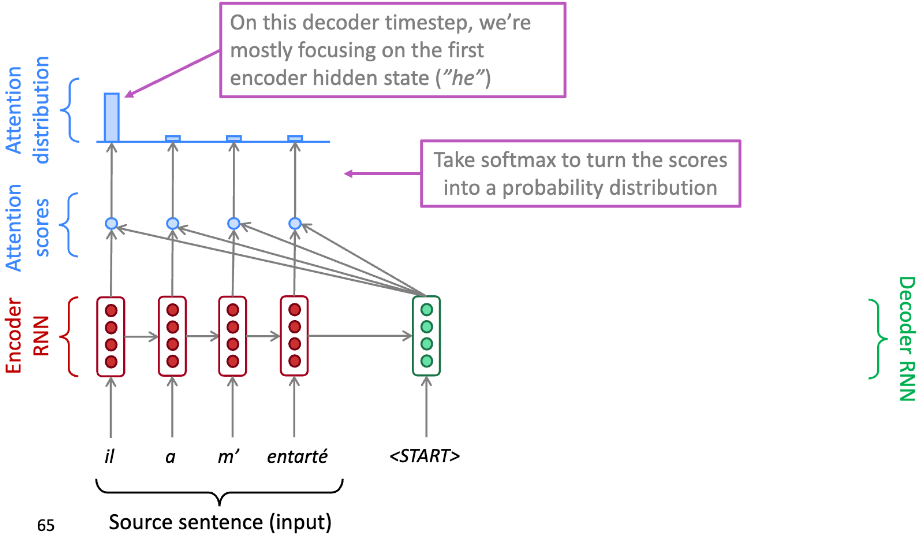
dot product
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)
61

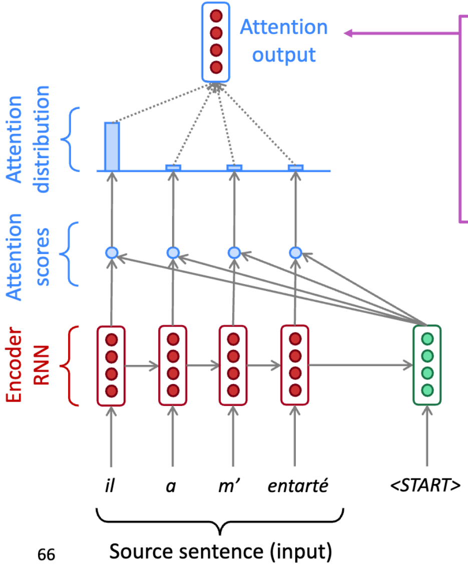

dot product
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)
62



dot product
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)
63

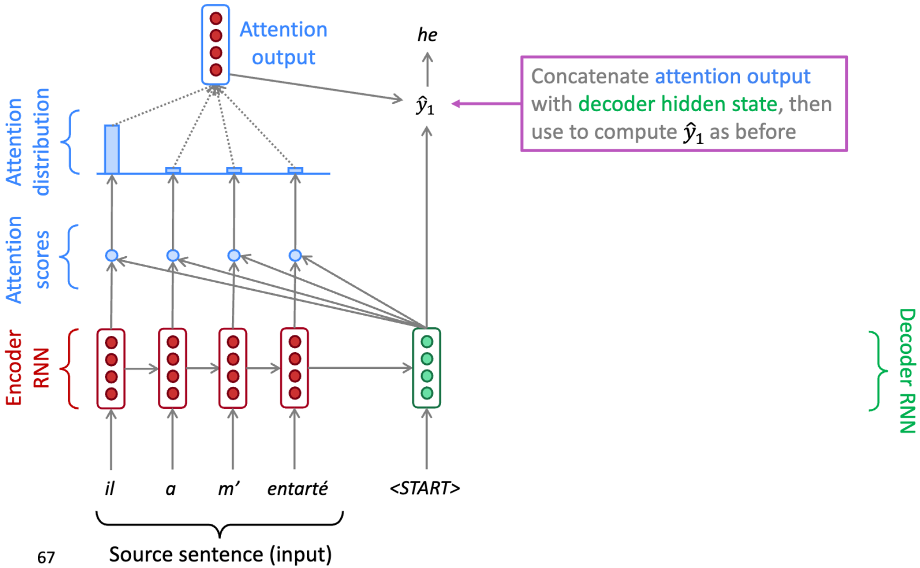

dot product
Attention scores
Encoder RNN
Decoder RNN
il
a
m’
entarté
<START>
Source sentence (input)
64

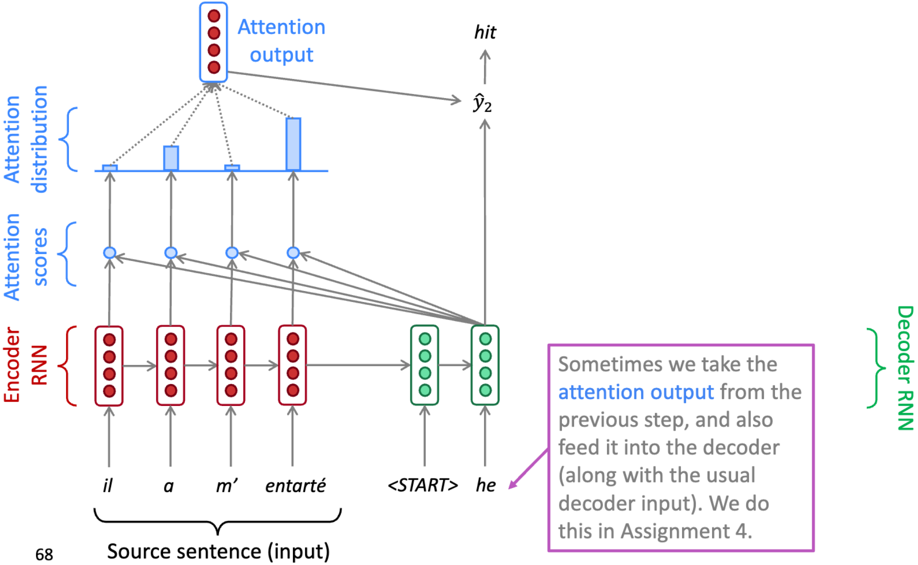

On this decoder timestep, we're mostly focusing on the first encoder hidden state ("he")
Take softmax to turn the scores into a probability distribution
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)
65

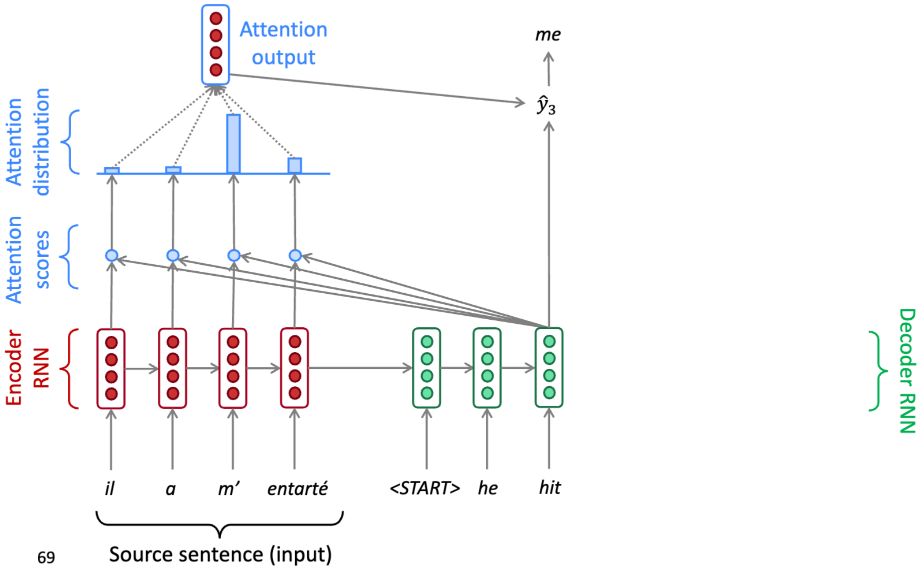


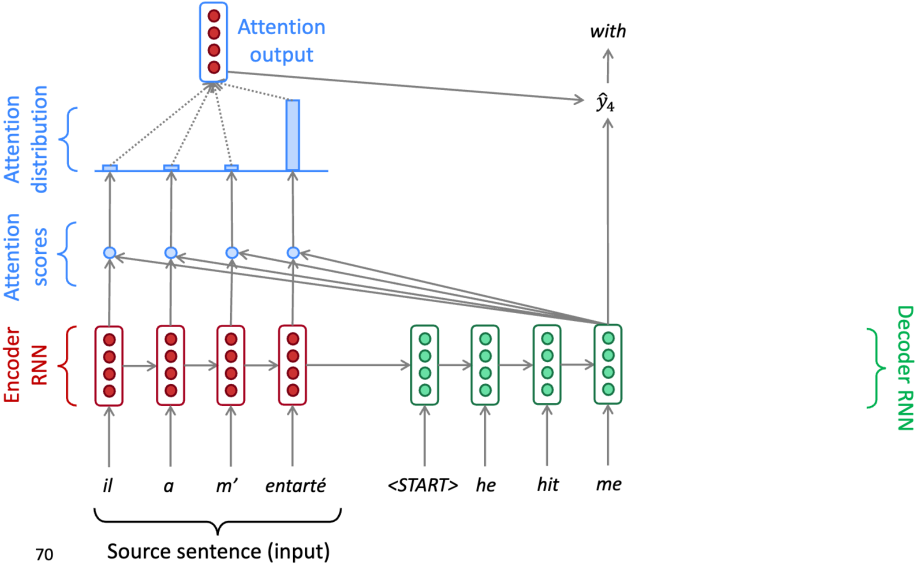
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information from the hidden states that received high attention.

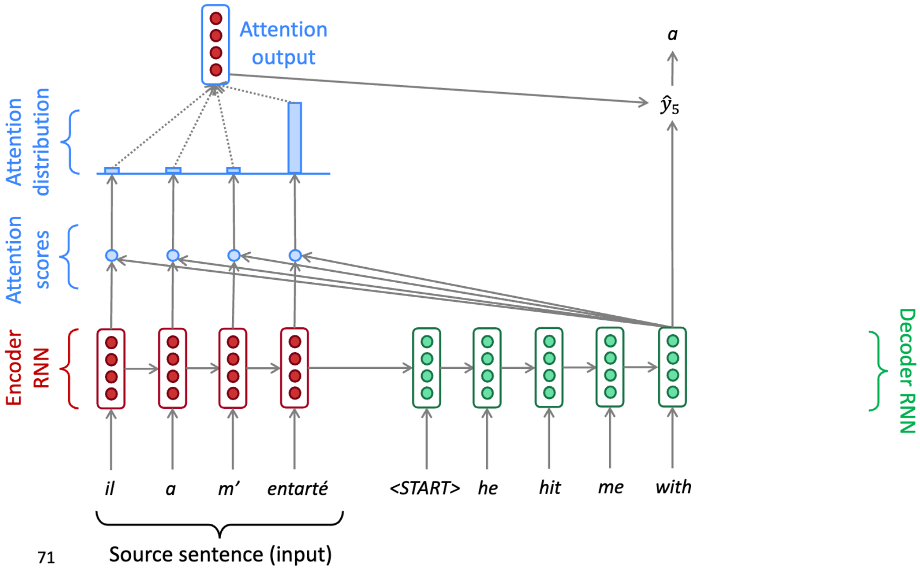

Attention output
he
Attention distribution
Attention scores
Encoder RNN
$\hat{y}_1$
Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)
67

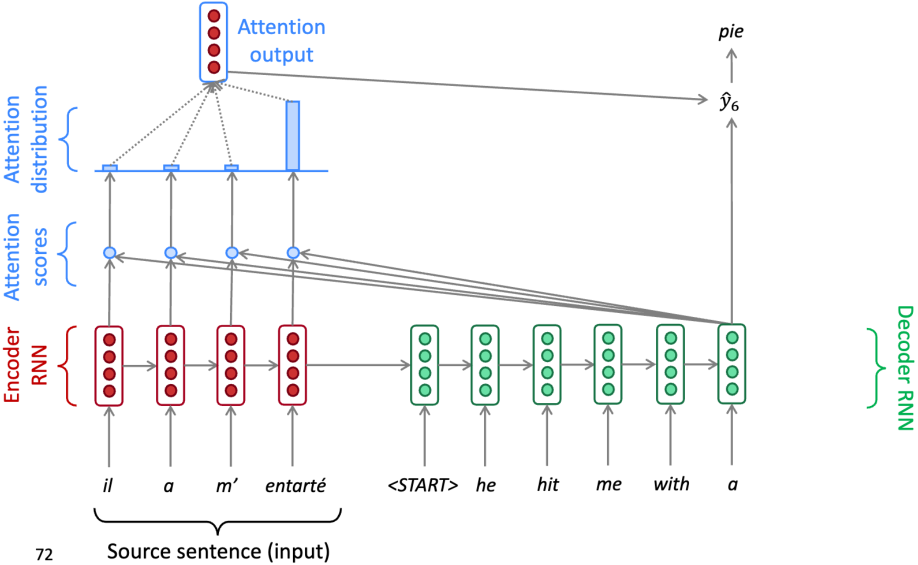

Attention output
hit
$\hat{y}_2$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
Source sentence (input)
Sometimes we take the attention output from the previous step, and also feed it into the decoder (along with the usual decoder input). We do this in Assignment 4.
68


Attention output
me
$\hat{y}_3$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
Source sentence (input)
69


Attention output
with
$\hat{y}_4$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
me
Source sentence (input)

70


Attention output
a
$\hat{y}_5$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
me
with
Source sentence (input)
71

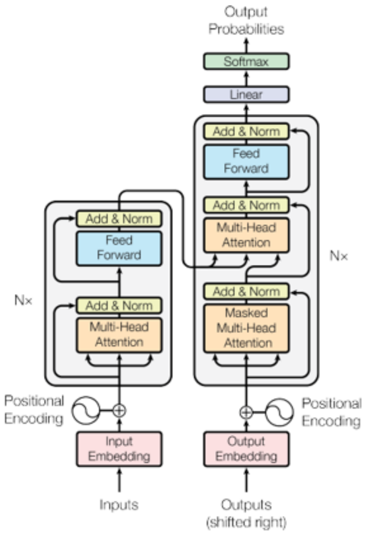

Attention output
pie
$\hat{y}_6$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
me
with
a
Source sentence (input)
72

Advantages of Computing Attention

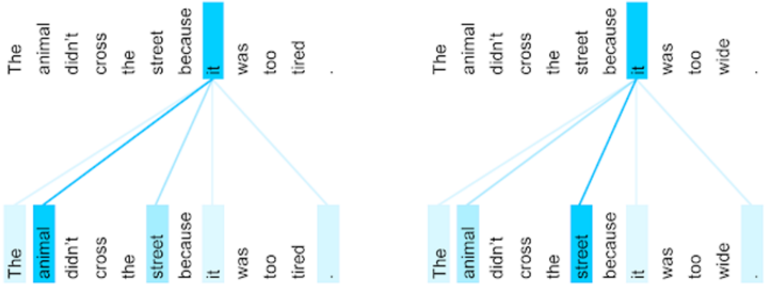
- Attention significantly improves performance (in many applications)
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on

Transformer

"Attention is All You Need"
(Vaswani et. al 2017)



Self-Attention Example from NMT



- The encoder self-attention distribution for the word 'it' from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads)

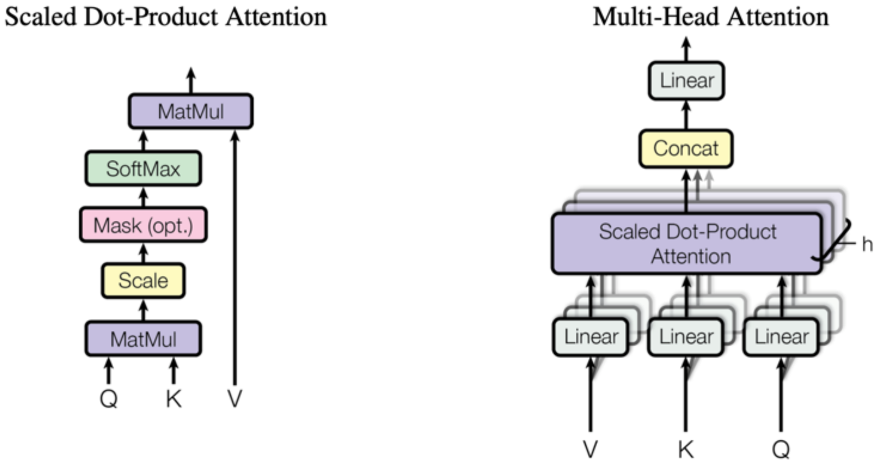


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

Topics

Transformer with Self-Attention Illustrated

"The Illustrated Transformer" by Jay Alammar

Self-Attention

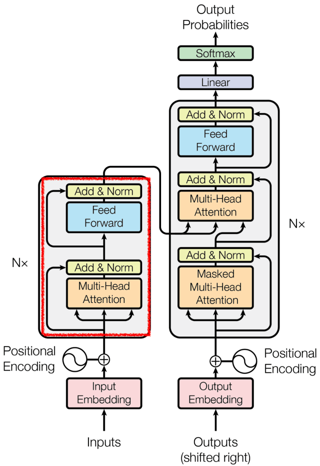
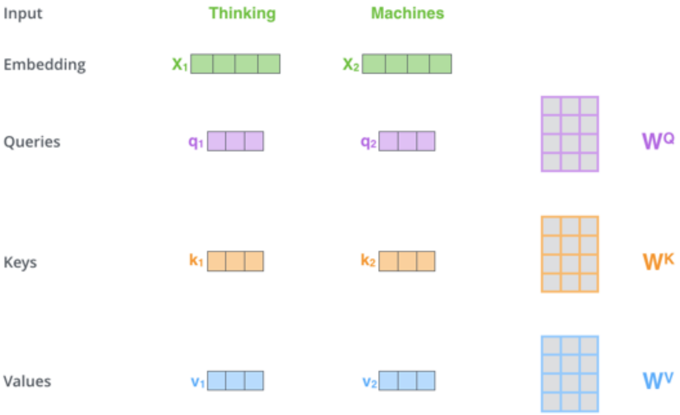


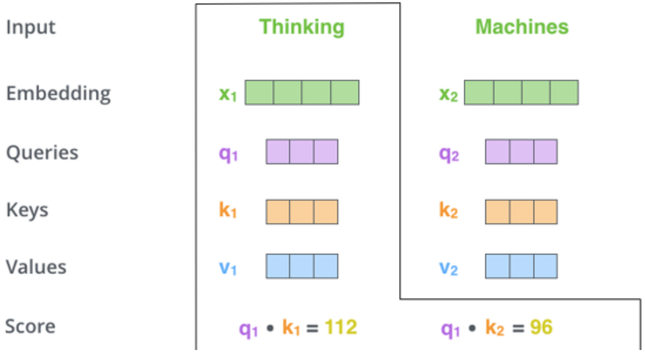
Figure: Multi-head Attention

Self-Attention



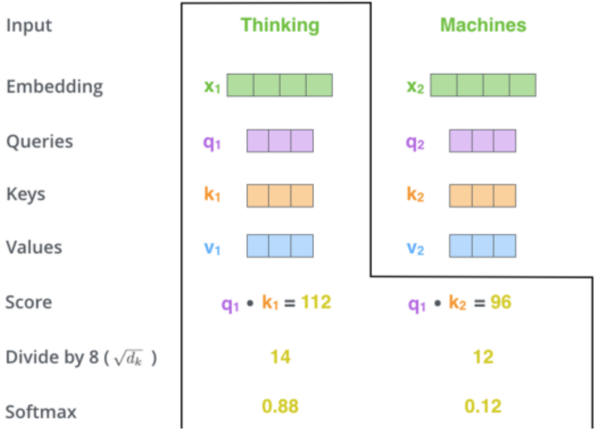
- Create three vectors for query, key, and value
- Multiplying x_1 by the WQ weight matrix produces q_1, and so on.

Self-Attention



- Mapping a query to keys (i.e., calculating the self-attention scores)

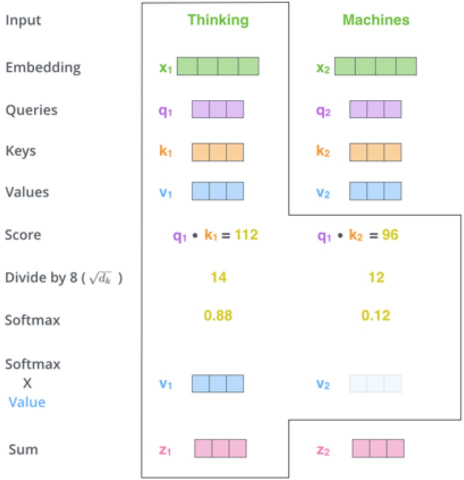
Self-Attention



- Normalize by the square root of the dimension of the key vectors
- Pass through a softmax operation

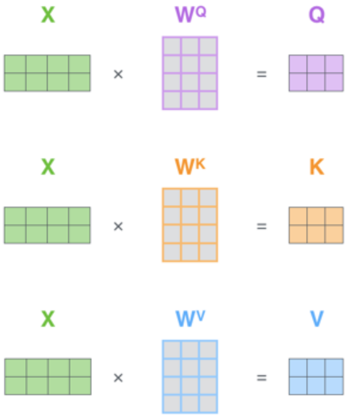
Self-Attention

- Return self-attention score weighted value
- The accumulated weighted value vector (z′s) is the output of the self-attention layer at this position.

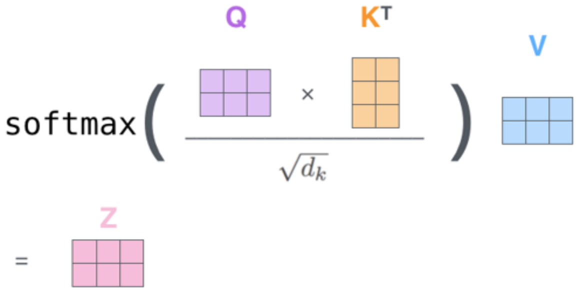


Self-Attention

- Matrix calculation by packing our embeddings into a matrix X, and multiplying it by the weight matrices we've trained (W's)



Self-Attention



- All the steps in one formula to calculate the outputs of the self-attention layer

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

Self-Attention

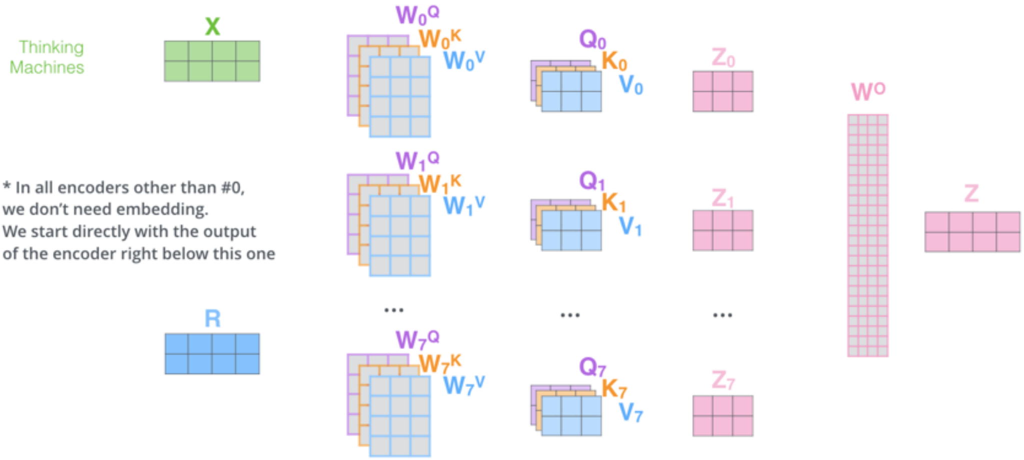
1) This is our input sentence*

2) We embed each word*

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

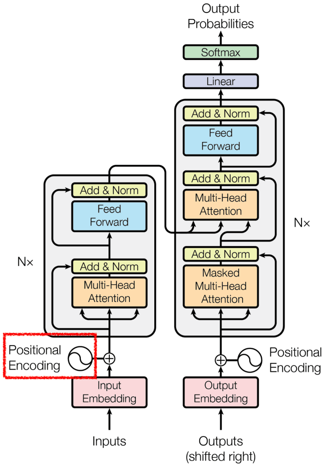
4) Calculate attention using the resulting Q/K/V matrices

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



Positional Encoding

- The Tranformer architecture is **missing** a way to account for the order of the words in the input sequence.
- Positional embeddings add positional information in the input sequence to each input embedding.
  - $PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$
  - $PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$



positional encoding

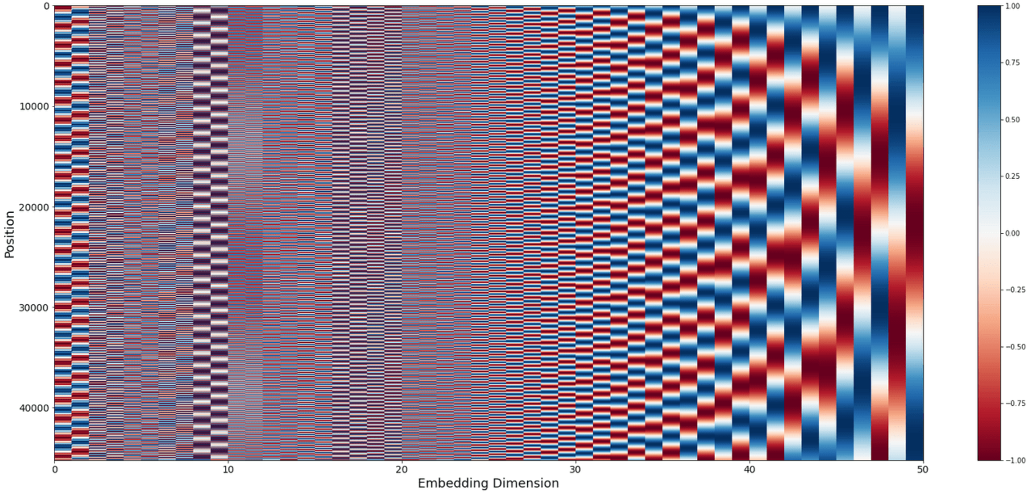
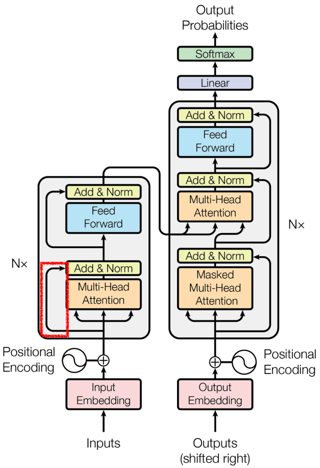


Figure: image by kemal erdem

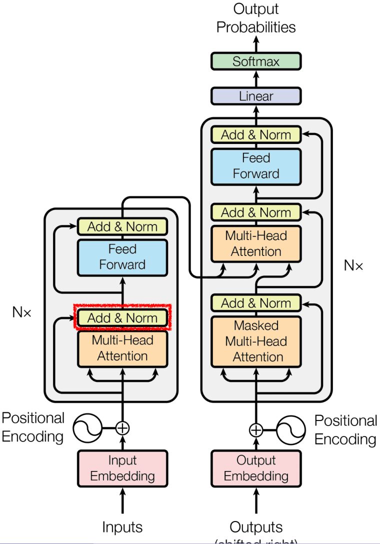
Residual Connection

- Residual Connection
  - adds direct connection between the input layer and output layer,
  - allows gradients to flow through a network directly, without passing through non-linear activation functions.
  - helps to address the vanishing gradients problem during backpropagation.



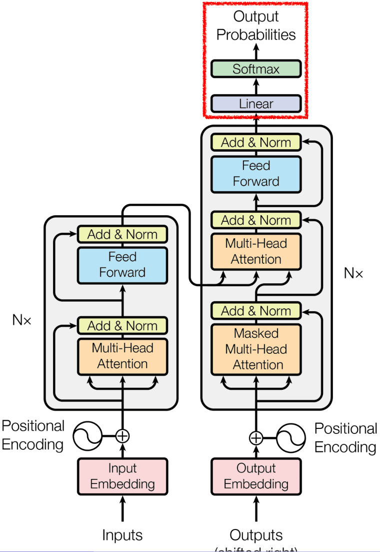
Add and Normalization

- Layer Normalization
  - performed independently across features on each training example and for each layer.
  - enables smoother gradients, faster training, and better generalization accuracy.

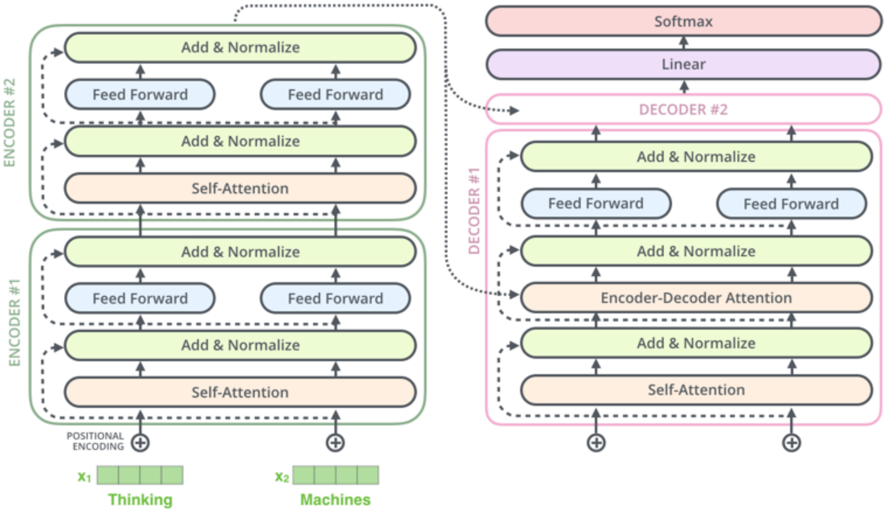


Prediction Layers

- The **final linear layer** outputs a much larger vector called a **logits vector**,
- and the following **softmax layer** converts it into log probabilities, which can be interpreted as a probability distribution in the target space associated with the downstream task.
  - If the task is language generation, the vector will be a probability distribution across the vocabulary.



Encoder-Decoder Attention



Topics

Language Modeling

- A language model (LM) is a probability distribution over sequences of token (e.g., words) variables, such that

  $$P(s) = P(w_1, w_2, w_3, \ldots w_n),$$

  given a sequence $s = (w_1, w_2, w_3, \ldots w_n)$.
- Typically, for optimization, we use the maximum likelihood estimation (MLE) with respect to the model parameters.

Language Modeling

- Optimization

$$
\begin{aligned}
\theta^* &= \arg\max_{\theta} \prod_{i=1}^{n} P_\theta(w_i) \\
&= \arg\max_{\theta} \sum_{i=1}^{n} \log P_\theta(w_i) \\
&= \arg\max_{\theta} \sum_{i=1}^{n} \sum_{j=1}^{m_i} \log P_\theta(w_{i,j} | w_{i,1}, w_{i,2}, \ldots, w_{i,j-1})
\end{aligned}
$$

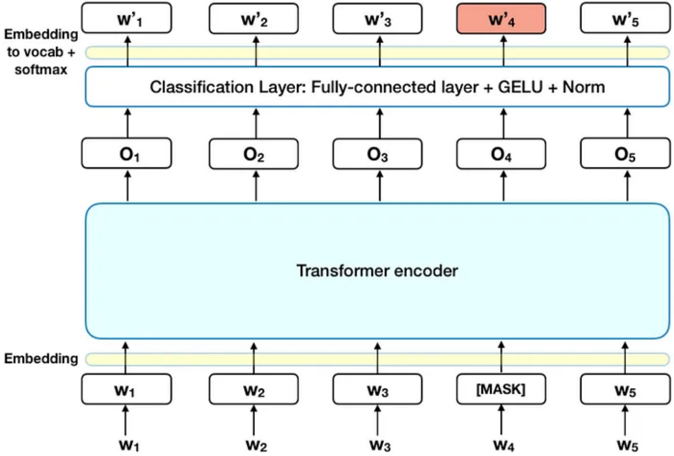
- where n is the number of examples and m_i is the sequence length.

BERT — Pre-trained Language Model

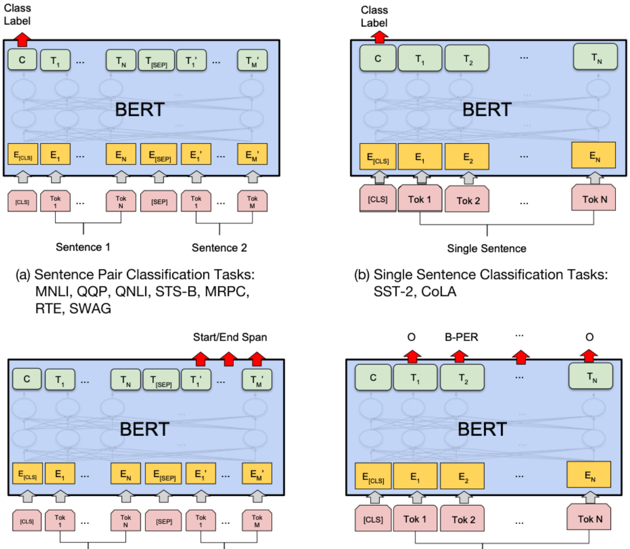
BERT (Bidirectional Encoder Representations from Transformers)

- A transformer (encoder) model trained using a large amount of unannotated and unstructured text data from the Internet.
- BERT learns the relationships between words in a language.
- BERT is trained on two specific tasks:
  - **Masked Language Modeling**: Predicting a missing word (masked tokens) in a sentence.
  - **Next Sentence Prediction**: Given two sentences A and B, predict if B comes after A.

BERT — Masked Language Model



BERT — Fine Tuning



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

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

Topics

1. Attention Mechanism
2. Transformer Network
3. **Pre-trained Language Models**
   - BERT
   - GPT to ChatGPT

GPT-1

- GPT-1 (Generative Pre-Training) developed by OpenAI in 2018.
- They argued that there's no need of fine-tuning a pre-trained language model for specific natural language processing tasks (e.g., question-answering, document classification, etc.), and
- **autoregressive decoder model** is sufficient enough for most of the tasks.
- paper "Improving Language Understanding by Generative Pre-Training"

GPT-1 (Cont.)

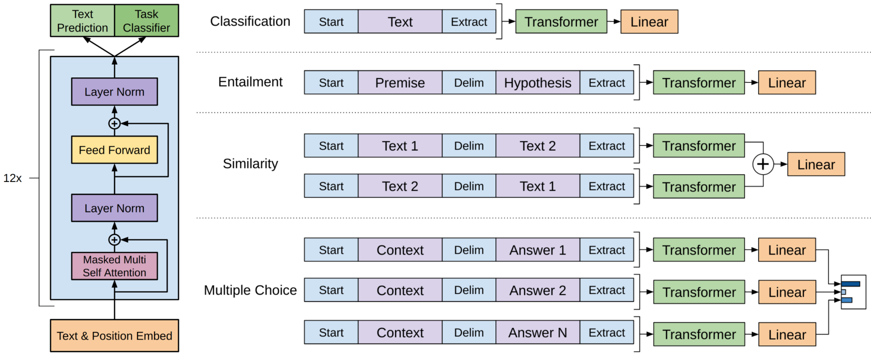


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

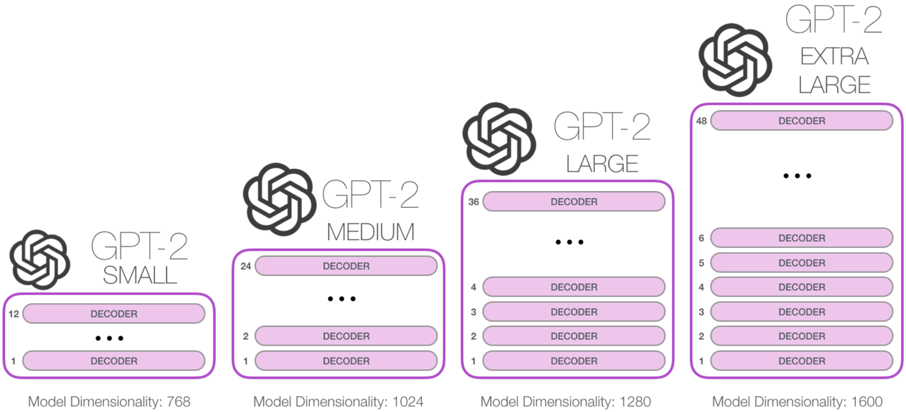
GPT-1 (Cont.)

- **GPT-1** is an autoregressive Transformer decoder model,
- whereas **BERT** is a non-autoregressive Transformer encoder model.
  - An autoregressive model can only pay attention to the previous tokens.
  - A non-autoregressive model can pay attention to tokens before and after itself bidirectionally.

GPT-2

- The second-generation of the GPT models released in 2019.
- Has more than one billion parameters (attributed to the collaboration of OpenAI and NVIDIA)
- Trained using an extremely large dataset, which might contain "annotated" examples for specific language tasks (e.g., QA or translation).
  - The extreme number of parameters and training examples enables the model *generating* answers, instead of *finding* answers.
- paper "Language Models are Unsupervised Multitask Learners"

GPT-2 (Cont.)


GPT-2
SMALL
12 DECODER
• • •
1 DECODER
Model Dimensionality: 768
GPT-2
MEDIUM
24 DECODER
• • •
2 DECODER
1 DECODER
Model Dimensionality: 1024
GPT-2
LARGE
36 DECODER
• • •
4 DECODER
3 DECODER
2 DECODER
1 DECODER
Model Dimensionality: 1280
GPT-2
EXTRA
LARGE
48 DECODER
• • •
6 DECODER
5 DECODER
4 DECODER
3 DECODER
2 DECODER
1 DECODER
Model Dimensionality: 1600

GPT-2 — Prompt-tuning

- **Prompt-tuning** is a technique to fine-tune large pre-trained language models to generate specific responses for a given prompt or context.
- This enables the model to generate more relevant and accurate responses for a given prompt.
- Useful for NLP certain tasks like chatbot, question-answering system, language translation.
- GPT-2 started to use prompt-tuning to guide the model in training for question answering, summarization, translation, and so on.
  - We can give the model a prompt to have it speak about a certain topic.
  - E.g., recite the first law \$ A robot may not injure a human being . . .

GPT-3

- The third-generation of the GPT models released in 2020.
- Even larger in size (175 billion model parameters)
- GPT-3 is trained to read and understand the context and repeat generating answer in the same manner.
- paper "Language Models are Unsupervised Multitask Learners"

GPT-3 Example

Prompt

I am a highly intelligent question answering bot. If you ask me a question that is rooted in truth, I will give you the answer. If you ask me a question that is nonsense, trickery, or has no clear answer, I will respond with "Unknown".

Q: What is human life expectancy in the United States?
A: Human life expectancy in the United States is 78 years.

Q: Who was president of the United States in 1955?
A: Dwight D. Eisenhower was president of the United States in 1955.

Q: Which party did he belong to?
A: He belonged to the Republican Party.

Q: What is the square root of banana?
A: Unknown

Q: How does a telescope work?
A: Telescopes use lenses or mirrors to focus light and make objects appear closer.

Q: Where were the 1992 Olympics held?
A: The 1992 Olympics were held in Barcelona, Spain.

Q: How many squigs are in a bonk?
A: Unknown

Q: Where is the Valley of Kings?

Settings

Engine	text-davinci-003
Max tokens	100
Temperature	0
Top p	1
Frequency penalty	0.0
Presence penalty	0.0
Stop sequence	\n

InstructGPT

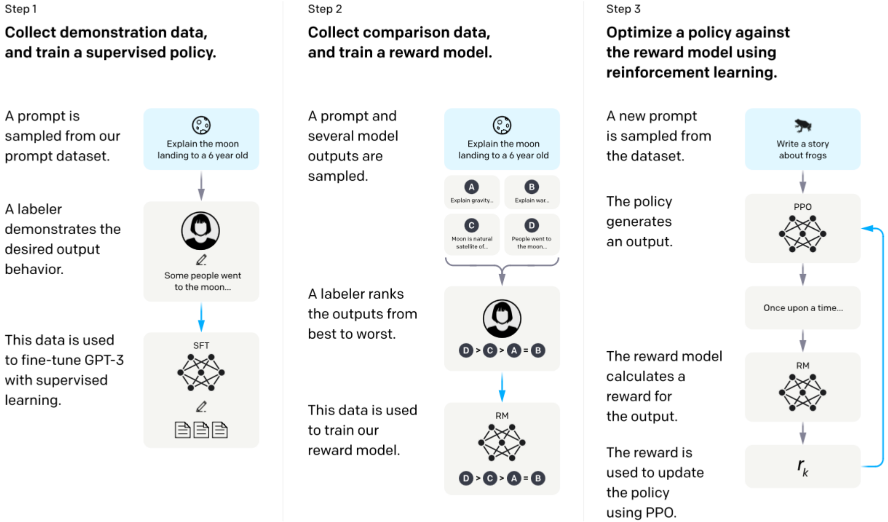
- Issues with the previous GPT models
  - The models are not trained to follow human instructions.
    - It needs to be provided with contextualized demonstration.
    - Triggers are engineered, e.g. "A:" or "Answer:"
    - Questions without the context is difficult to be answered.
- InstructGPT developed to address these issues in 2022. (paper, "Training Language Models to Follow Instructions with Human Feedback")

InstructGPT (Cont.)

Training the InstructGPT model

- Collect demonstration data of desired output is created by human beings. Use this data to fine-tune GPT-3 using supervised learning.
- Collect comparison data: GPT-3 generates outputs to a prompt, and a human labeler ranks the multiple outputs which is then used to train the model using reinforcement learning.

InstructGPT — Training



ChatGPT

- ChatGPT and InstructGPT are essentially the same model, except that for ChatGPT,
  - Different demonstration examples are used to train for better conversation between the users and the bot.
  - E.g., Examples include dialogues created by human.
- This interactive information system can replace conventional search engines in many different ways.