Transformers and Beyond

PhD Candidate
Iordanis Fostiropoulos
{fostirop[at]usc.edu}

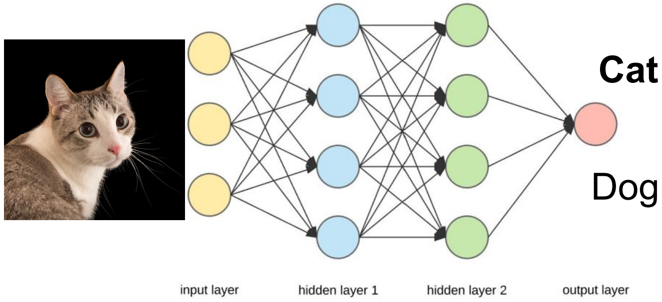
Advised by Prof. Laurent Itti

April 4th 2023
Prepared for CSCI 561

Transformer Intuition

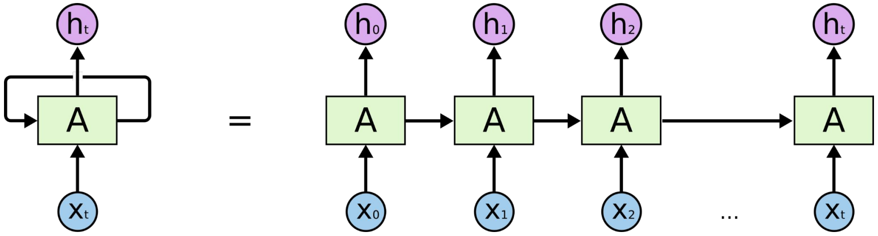
The first-half of the content in this presentation is based on [1] [2] that you should read.

Supervised Machine Learning



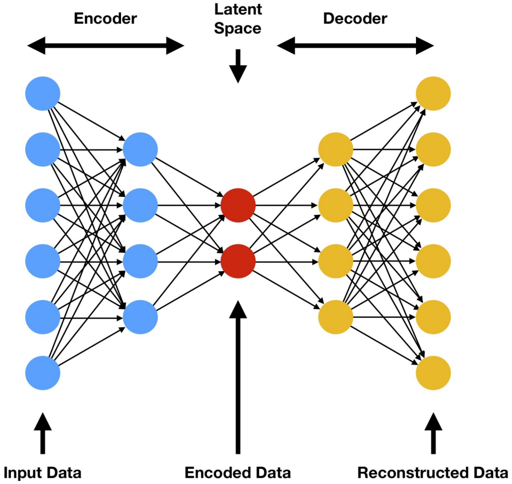
Output can be predicting a class, or any data.

Sequence Modeling



Input to the model is the previous output

Transformer Intuition



Encoder Decoder Architectures

Input some data, output the same data.

Self-Supervised

Loss is calculated between input data and reconstructed data

The encoded data is a compressed **representation** of the input data.

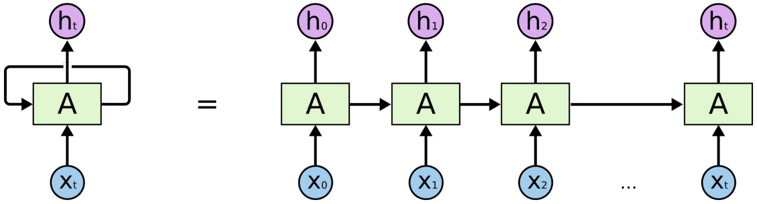
Can be **useful** in **other** tasks,
i.e. the image representations combined with text representations are used for DiffusionModel

Sequence to Sequence (Tangent)

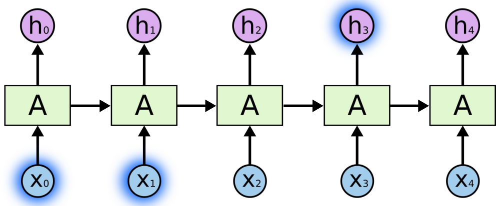
Warning! Relevant **ONLY** in understanding advantage of Transformers

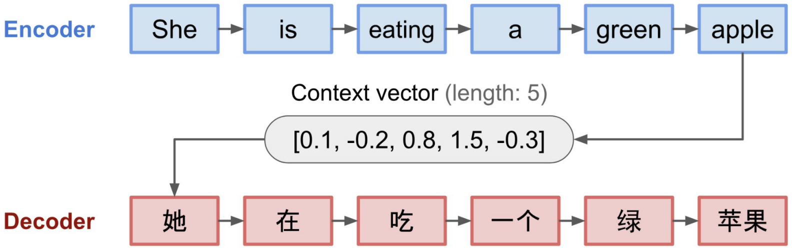
Previously, **Recurrent Neural Networks**

Have loops, and we can unroll them (i.e.)



Problem processing one token *x* at a time





[1] Attention? Attention!
[2] Understanding LSTM Networks

Transformer

Combines an Auto-Encoder with a Seq2Seq objective

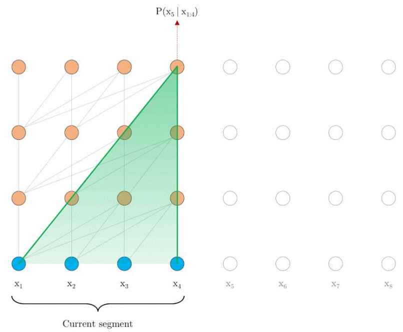
We want to learn to decode the encoded data in a self-supervised manner, but we want to model it as a sequence.

What is the probability of a token given all previous tokens

$$P(x_i|x_{i-1:n})$$

Advantage it learns on a segment of a sequence compared to one token at a time (LSTM)

Recurrency (Looping) through segments compared to tokens.



Transformer Encoder-Decoder

Introduced for Machine Translation (MT) i.e. English-to-French

Inputs: English Sentence
Outputs: French Sentence

Problems?
Length of English Sentence $\neq$ French Sentence
Different Grammar. Order of the translated words is different

Solutions
Encoder-Decoder Architecture (decoder uses a hidden state)
Attention Mechanism (each word can “attend” to a sequence of words)

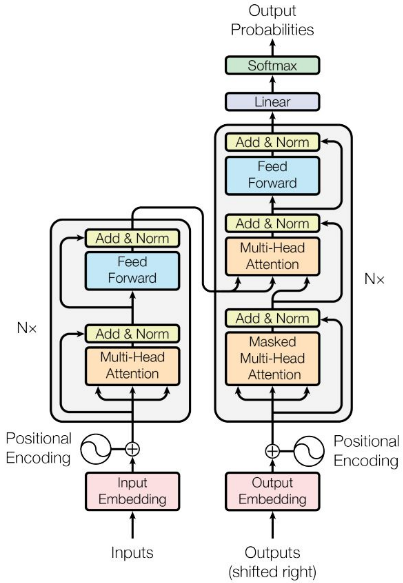


Figure 1: The Transformer - model architecture.

Encoder Block

Objective
Compress a sequence to a **hidden** state.

1. **Input Embedding**

Convert words into Vector Representations

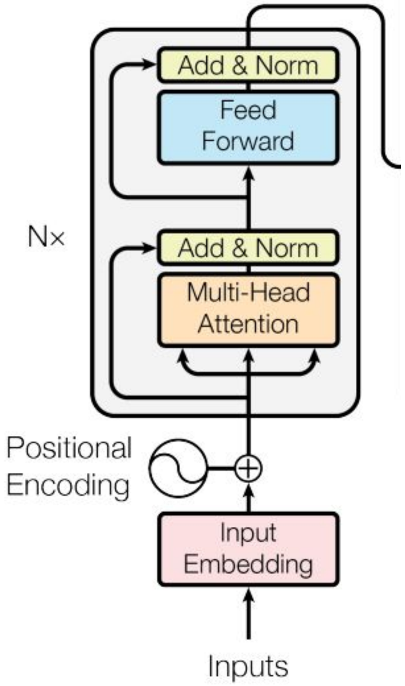
2. **Positional Encoding**

Attention is **Permutation Invariant** we need a way to encode position of a word in the sentence

$$f((x_1, x_2, x_3)) = f((x_2, x_1, x_3)) = f((x_3, x_1, x_2))$$

3. **Attention!**

Learn the context of each word. i.e. what words are before and after the current word???



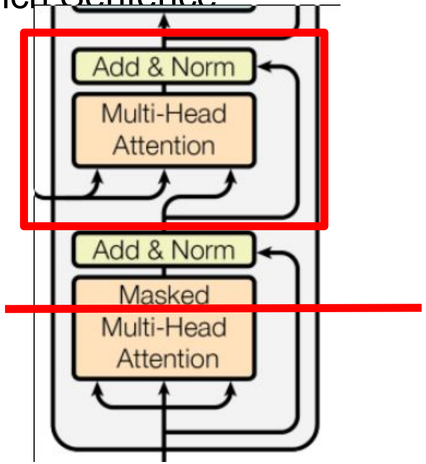
Decoder Block

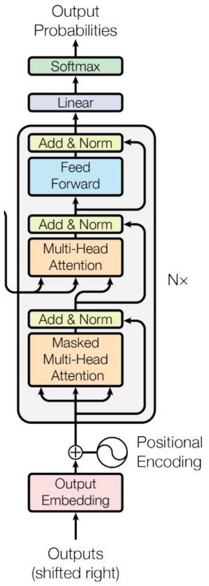
Objective
Convert compressed **hidden** state to the expected output.
i.e. English Sentence (**hidden state**) to French Sentence

Identical structure to decoder except….

Cross-Attention Block
Compute Attention between hidden state and Output sequence

Masked Self-Attention
Hide attention of subsequent tokens to prevent "cheating"





Attention!

Query (Q) is a token we use to “search” through the most similar keys

Key (K) a token we use that corresponds to a value

Value (V) the output that corresponds to the key

Most **similar** to a Java HashMap or Python dictionary but… returns the most similar value (**Hard Attention**)

i.e. **Oversimplification**

1 is closer to 0 than to four

```
attention = {0:"zero", 4:"four"}

query = 1

attention[query]

>> "zero"
```

Soft Attention


```
soft_attention = {0:"zero", 4:"four"}

query = 1

soft_attention[query]

>> ["zero"*0.8808, "four"*0.1192]
```

Attention!

Softmax Normalizes a vector to sum to 1.
Meets requirement for probabilities. Each element in the vector is an "event", "category", "class"

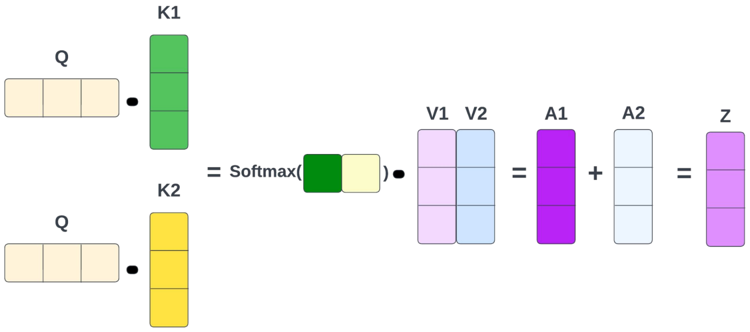
i.e. Probability of q = 1 being "zero" is
softmax([-1,-3]) = 0.8808

Transpose (T) is the transpose of the Key (For performing a Dot Product)

$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V}$$

Illustration… Not real code

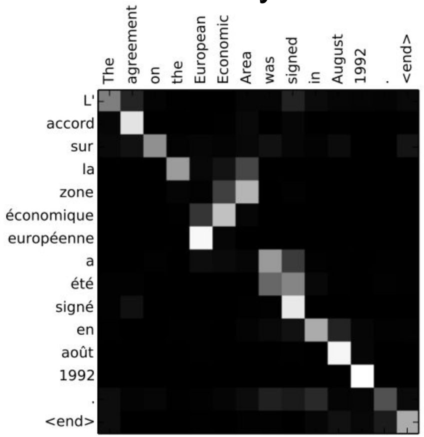
```
soft_attention = {0:"zero", 4:"four"}

query = 1

soft_attention[query]

>> ["zero"*0.8808, "four"*0.1192]
```



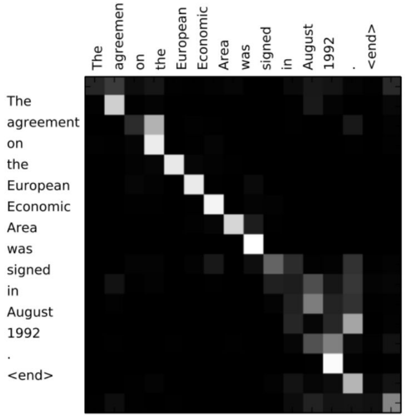
Attention!

Attention is applied on sequences. **Matrix** of **attention** from row element to column element.

Cross-Attention
Different Key and Query



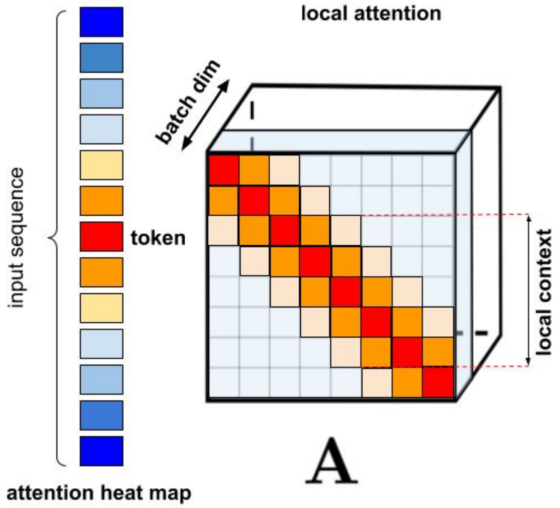
Self-Attention
Same Key and Query



Masked Attention!

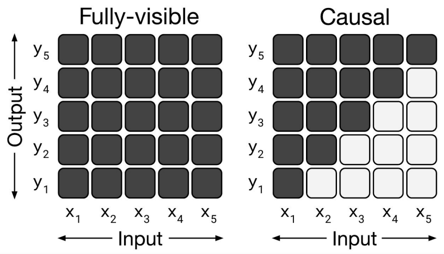
Attention Matrix is the computational bottleneck of Transformers. Quadratic memory growth with sequence length.
O(n^2) **x** Value Dimension (Embedding Dimension) **x** Attention Head **x** Layers = **Large**!

Solution Sparse Matrices attend to **local** context (around a word)



Problem Self-Attention is *somewhat* cheating.
Easy to decode each word, if we can see before and after.
We do not learn much about the **structure** of language

Solution Causal Attention
Mask future tokens



Multi Headed

In practice, the biggest improvement of Attention is applying it many times in parallel.

Same input, different attention heads.

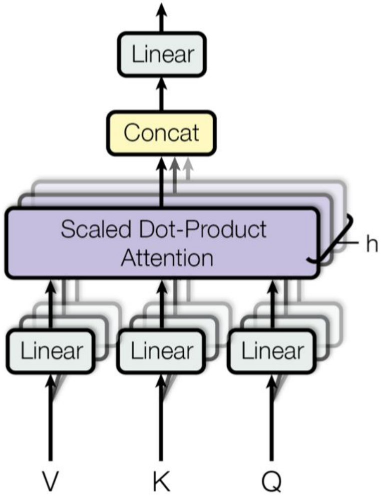
Concatenate Output of all heads.

Combine With a linear layer.

So…. What exactly are we learning?

Linear is a learnable parameter (weight matrix)

Dimensionality **Embedding Dimension**



Input Embedding

Problem
Can’t do math on words. i.e. “zero”*0.88 **ValueError**

Making words into numbers. A lookup table.

1. **Input** sentence is “Cat on MAT!”

2. **Tokenize** = [“cat”, “on”, “mat”] = [2,5,10]

3. **Embed**([2,5,10]) = [[1.2,-0.1,4.3, 3.2],
   [2.1,0.3, 0.1, 0.4]
   [2.1,0.3,0.1,0.4]]

A 4-dimensional embedding

cat =>	1.2	-0.1	4.3	3.2
mat =>	0.4	2.5	-0.9	0.5
on =>	2.1	0.3	0.1	0.4
...	...			

[1] Word Embeddings

Permutation Invariance

Attention is Permutation Invariant

Order of words does not matter.

i.e. swapping rows and columns result in equivalent values.

Solution

Encode positional information

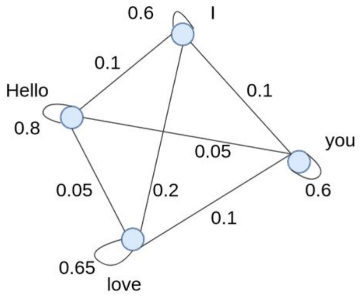
Positional Encoding

Self-attention
Probability score matrix

	Hello	I	love	you
Hello	0.8	0.1	0.05	0.05
I	0.1	0.6	0.2	0.1
love	0.05	0.2	0.65	0.1
you	0.2	0.1	0.1	0.6

Softmax(Attention) equation

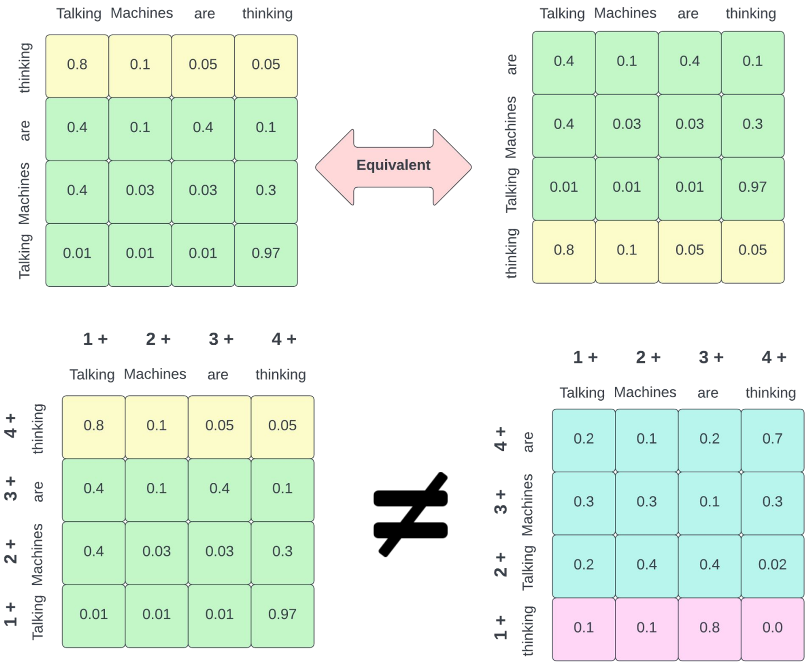
Self-attention as a undirected weighted graph



Positional Encoding

Add an embedding that encodes the position of the word in the sentence.

Switching the columns results in different attention computation



Train Objective

n = Sequence Length

h = Embedding Dimension

Output of Transformer Layer is a sequence

Input Shape **[n , h]** and Output Shape **[n , h]**

AutoEncoder Objective. **Reconstruct** Input

Use linear layer to project each token (i) [1, h] →[1, h, vocabulary size] = **logits**

Softmax(logits) = **preds** → Probability of token (i) to be a word at index j in the preds

Goal Maximize Probability of predicting the correct word

BERT - Denoising Transformer Encoder

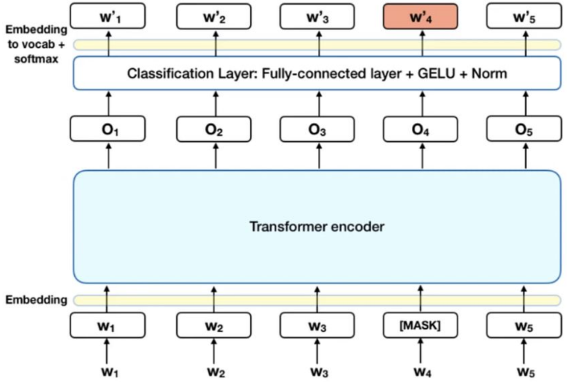
Transformer Encoder ONLY!

1. *Hide* input tokens by replacing them with the same special [MASK] token

2. Maximize probability of correctly predicting the **real value** of the masked token

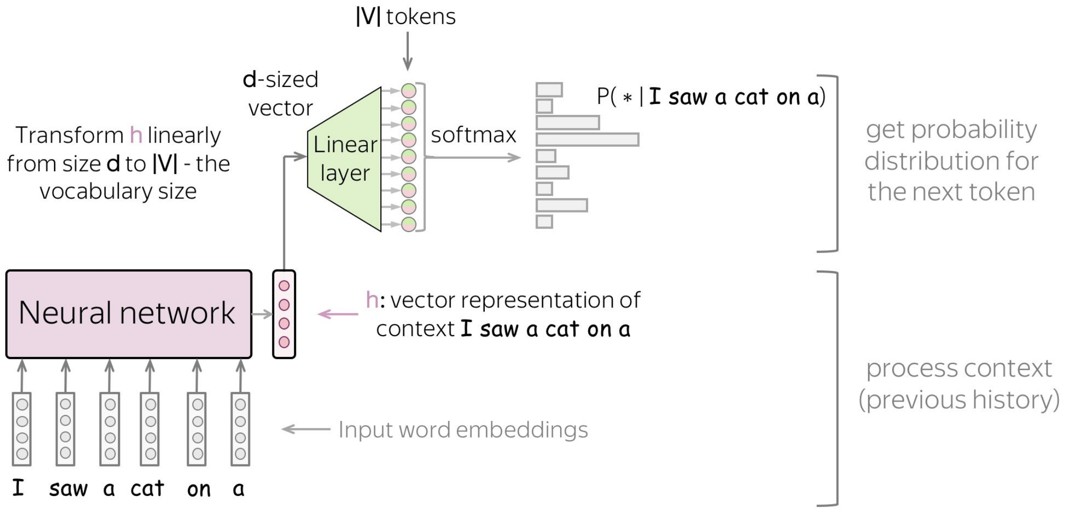
Special Tokens can be added to the vocabulary that are not part of the language for special purpose
i.e. [SEP] Start of New Context
[EOS] End of Sequence
[CLS] Used for classification tasks **and more**

- Pre-training BERT
  - ✓ Task 1: Masked Language Model (MLM)
    - 15% of each sequence are replaced with a [MASK] token
    - Predict the masked words rather than reconstructing the entire input in denoising encoder



Next Token Prediction

Used to model Causal relationships



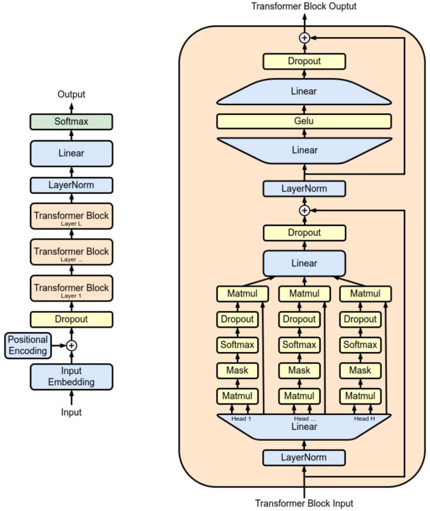
GPT - Decoder Only

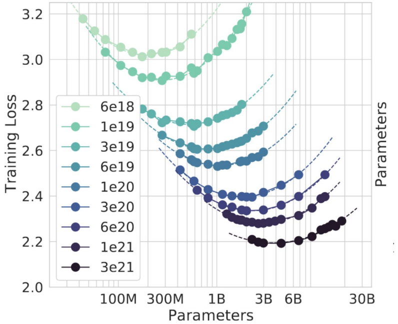
Transformer Decoder Only! (**without** cross-attention)

Different Causal Language Modeling
Try to predict next word given the current context **so far**

In **Summary**: GPT vs GPT2 vs GPT3 vs GPT4

Scaling Laws
More Layers, Larger Hidden Dimension, More Data



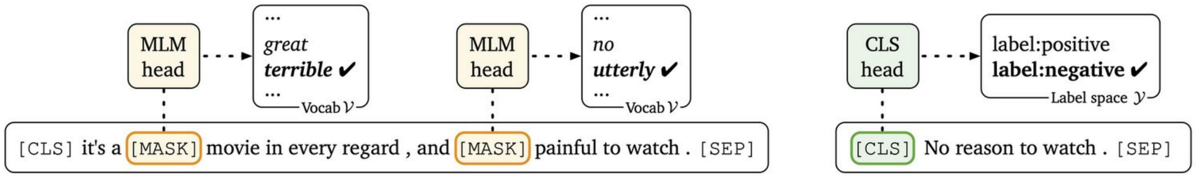


[1] Training Compute-Optimal Large Language Models

How to use a Large Language Model

Fine tune to a specific task i.e. Sentiment Classification

Prompt to generate new context. Start a sentence, ask the model to complete it.



(a) MLM pre-training

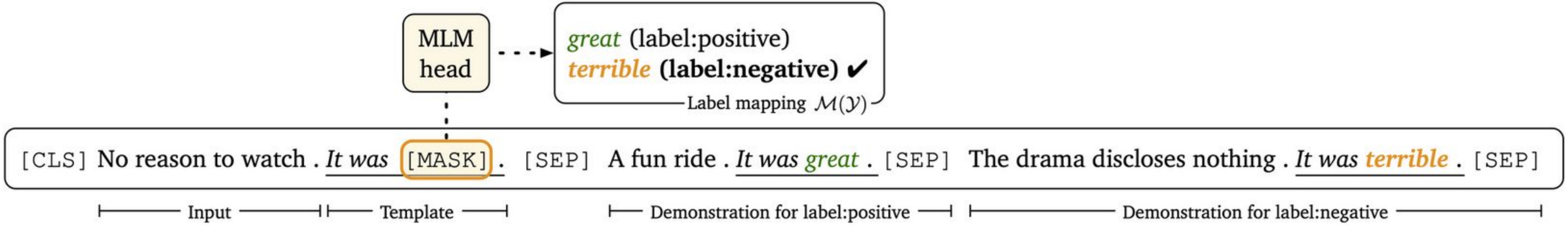
(b) Fine-tuning

Prompting

We use [SEP] to separate contexts.

Single Input:

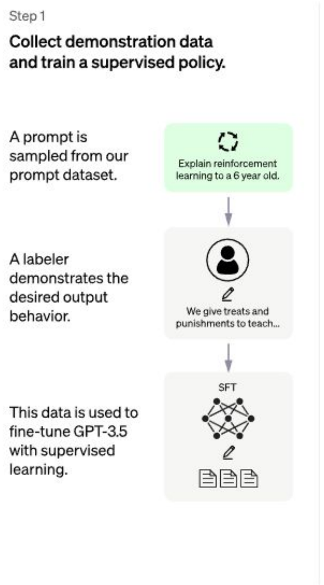
[CLS] Some prompt (Question) [SEP] Some Answer [SEP] Second Question [SEP] Second Answer [SEP]



ChatGPT (Step 1)

1. **Train** a Large GPT (Causal Language Modeling)

2. **Fine-Tune** GPT with prompts collected
   from human annotators
   i.e. Ask a human
   “Explain reinforcement learning to a 6 year old.”



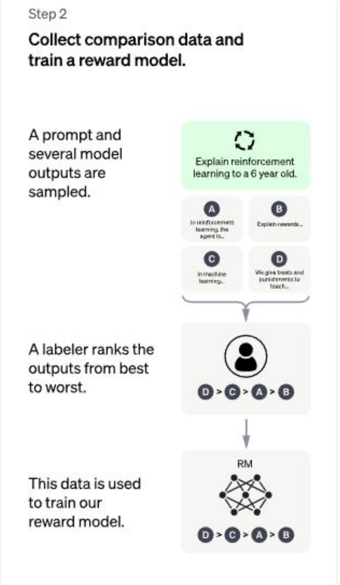
ChatGPT (Step 2)

1. Provide a prompt to the model
2. Sample Outputs
3. Ask humans to rank outputs (**easier than writing them**)

Reward Model

1. Given Prompt
2. Predict **reward** of each prompt

Reward Model is used next… in **Reinforcement Learning**



ChatGPT (Step 3)

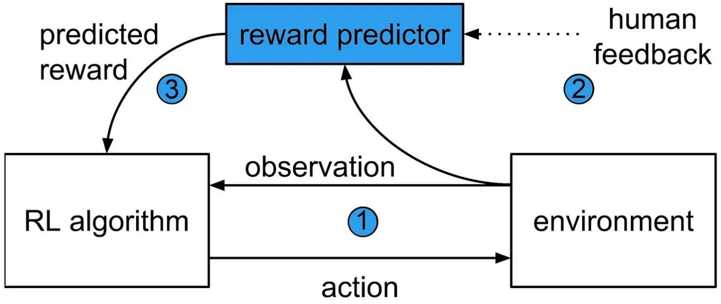
Reinforcement Learning

Summary
Given observation in Environment what is the best action to take

Interactive
i.e. Observation 1 → Action 1 →Observation i
Observation 1 → Action 2 →Observation j

Goal Pick action that maximizes reward

🤔🤔 Similar to tree search?



ChatGPT (Step 3 cont.)

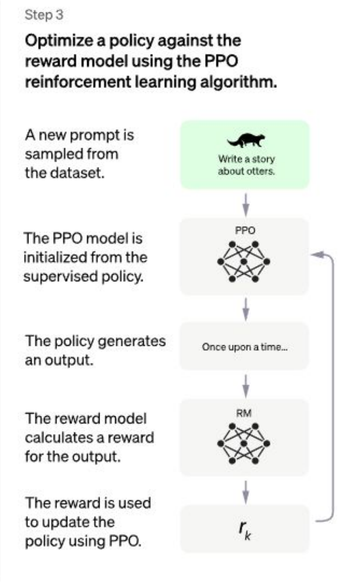
Reward Model (from Step 2)
Used to predict expected reward of prompts and actions.

PPO fancy way of saying **train** a **Reinforcement Learning** agent
Proximal Policy Optimization

Use **RL** agent to pick prompts

ChatGPT is not new (research from 2017)

Advantage
High Quality Annotators (Reinforcement Learning from Human Feedback)
Engineering Achievement



Alignment

Why does Reinforcement Learning from Human Feedback (**RLHF**) work so well?

Alignment

How do we align AI systems to our goals? By giving them feedback

Warning! Opinion Based Perspective

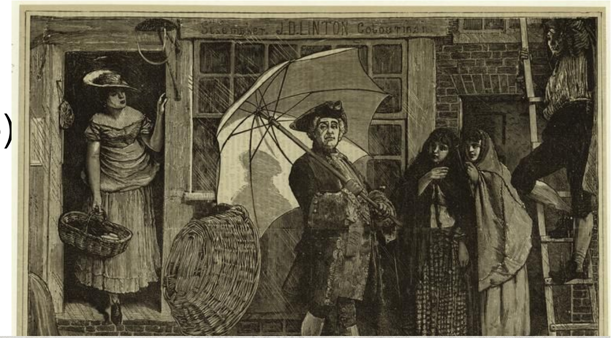
Are they conscious? Are they dangerous? Are they….

We can't answer… but …

(Opinion) *They are impressive but are just statistical machines*

Technophobia

First man to use an umbrella for rain Jonas Hanway (1712-1786)
Mocked for his portable roof



In 1865, the British Parliament passed a law to regulate a new, scary invention: the horseless carriage.

Beyond ChatGPT - AlphaFold

Transformers can solve important problems.

“AlphaFold can accurately predict 3D models of protein structures and is accelerating research in nearly every field of biology.”

Drug Discovery
Can design drugs by simulating their behavior.
Reduce search space of drugs