Elements of Large Language Models

Tokenization, Embeddings, Transformers, and more

Eric Wright

What are you made of?

I am a computer program called a "language model." I am not made of physical matter like humans and animals are, but rather I am a series of algorithms and data that can process and generate language. I exist on servers in data centers and can interact with users through the internet.

From the legendary [ChatGPT]

Objectives

- Focus on a high-profile class of models - OpenAI's GPT
  - GPT: Generative Pre-trained Transformer
- What is GPT? What does it do?
- What are GPT's components? How does it work end-to-end?
  - Tokenization, word embeddings, attention mechanisms
- How is GPT used in applications (e.g. ChatGPT)?

Who am I?

- UT-Austin alum - Physics/Math/Software
- Principal Data Scientist at Microsoft
  - Azure Machine Learning
- Demand forecasting, price optimization, statistical decision making
- Also, father of small children
  - Very sleepy all the time

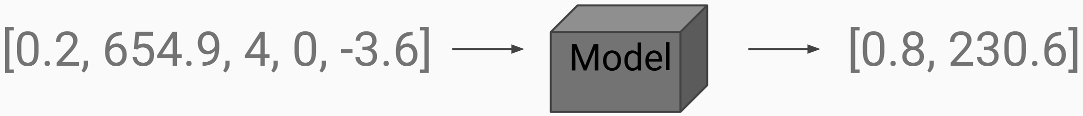
Disclaimers

- Microsoft has a significant stake in OpenAI (GPT-3, ChatGPT)
- I’m here in a personal capacity, not on behalf of Microsoft
- Not an expert on LLMs - just a data scientist who tries to read fine print

Machine Learning 101

What is a model?

Dumb definition: A box that does something to a bunch of numbers

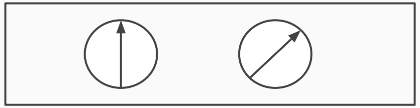


The box need not be deterministic - the same input may give different outputs

Model parameters

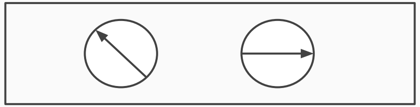
The box usually has adjustable knobs that change the output

[0.2, 654.9, 4, 0, -3.6] →



→ [0.8, 230.6]

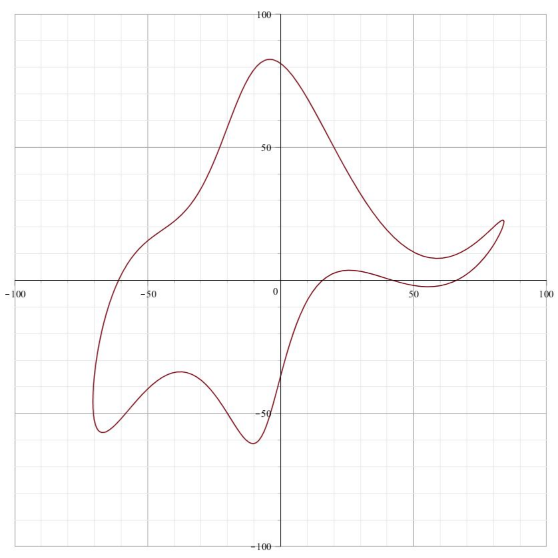
[0.2, 654.9, 4, 0, -3.6] →



→ [0.32, 789.1]

Model parameters

J. von Neumann: “With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.”



From [von Neumann's Elephant on Wikipedia]

GPT-3 has 175 billion parameters - maybe we wiggle individual hair follicles?

Model training

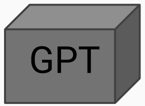
- Training means finding the “best” settings for the knobs/parameters
- Start with a set of example pairs of inputs and outputs
  - “Training data”
- Adjust the parameters until model outputs are “close” to example outputs
- Check how well the model predicts outputs for other examples not in the training

What is GPT?

“Generative”

The output is a probabilistic prediction of the next token in a natural language text string

“Beware the ides of “



Word	p
February	0.2
March	0.7
April	0.1

“Pre-trained”

The model is trained on a **large** number of example text strings from several sources, e.g., for GPT-3:

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

From [OpenAI's GPT-3 paper]

“Transformer”

A type of deep neural network model designed primarily for natural language tasks.

- Considered the current “state of- the art” in NLP
- Most important feature: **attention mechanism**

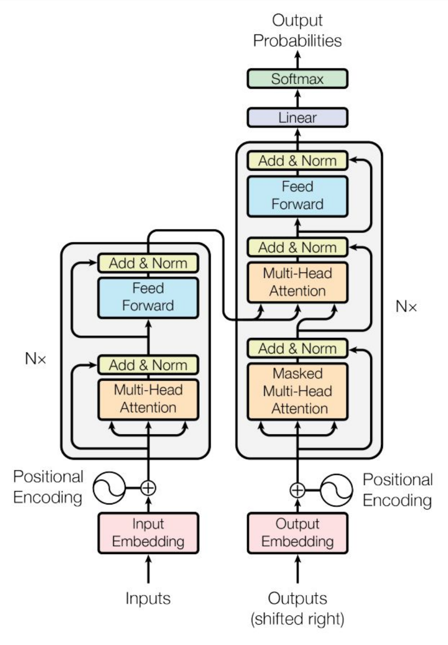


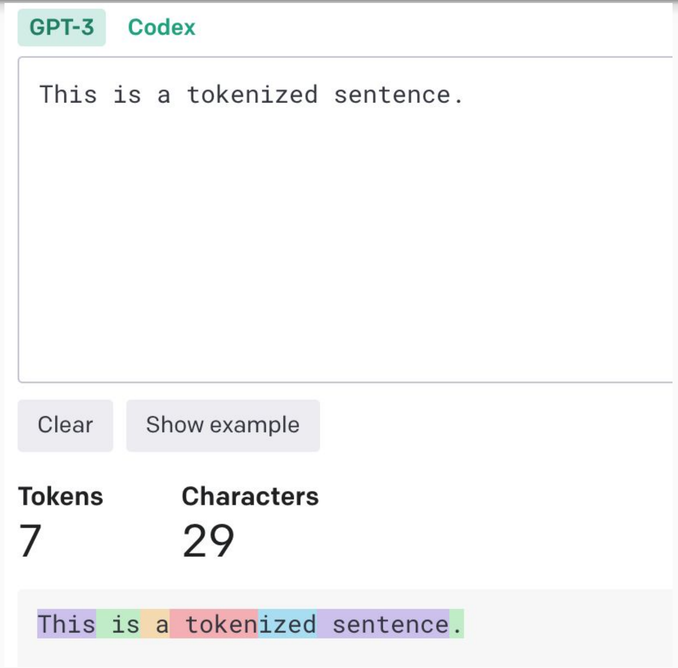
Diagram from [Google Brain's Transformer paper]

Words to numbers: tokenization and embeddings

Tokenization

- Parse strings into discrete pieces - “tokens”
- Tokens have numbers between 1 and N
- For GPT-3, N ~ 50,000

“This is a tokenized sentence” ->
[1212, 318, 257, 11241, 1143, 6827, 13]



From [OpenAI's interactive tokenizer]

Limitations of numbered tokens

We can't do arithmetic with numbered tokens:

$$\text{"king"} - \text{"man"} + \text{"woman"} \stackrel{?}{=} \text{"queen"}$$

$$\Big\downarrow \text{Tokenizer}$$

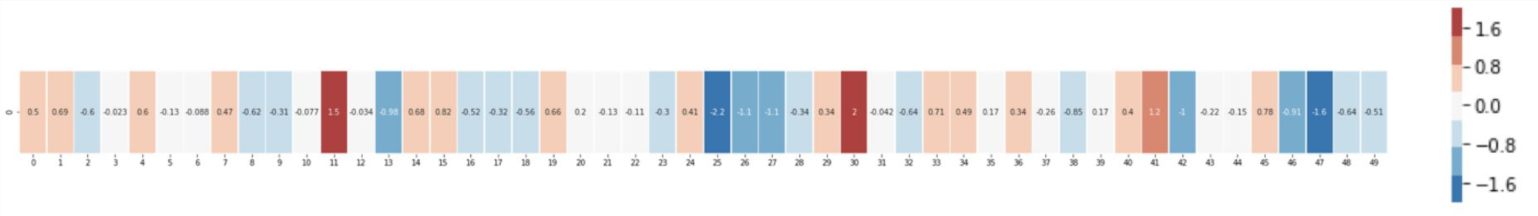
$$3364 - 528 + 2415 \neq 16599$$

These numbers contain zero meaning about their words :(

Embeddings

Basic idea: Find a numerical representation that encodes word meaning

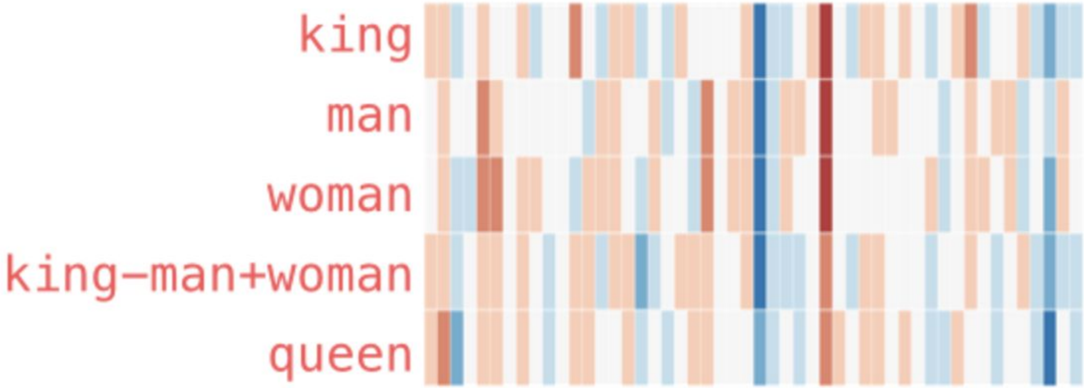
“king” ->



Each token now has its own list of 40 decimal numbers - AKA an embedding

Embeddings

king − man + woman ~= queen



The resulting vector from "king-man+woman" doesn't exactly equal "queen", but "queen" is the closest word to it from the 400,000 word embeddings we have in this collection.

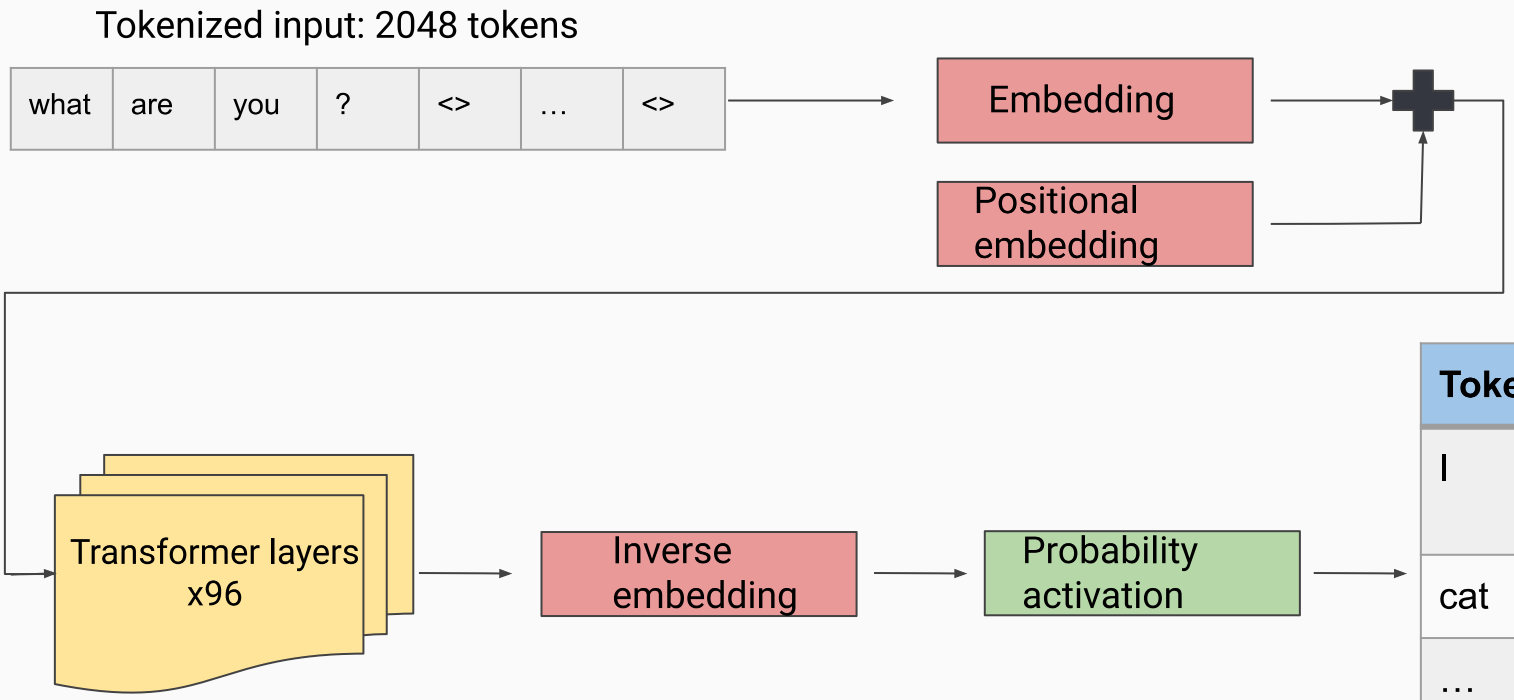
This example is from [Jay Alammar's machine learning blog]

Embeddings in GPT

- In GPT-3, each token embeds into 12,228 numbers
  - This is called the embedding dimension
- Embeddings in GPT are *part of the model*
  - ~640MM parameters for tuning
- Also embeddings by token position within a string

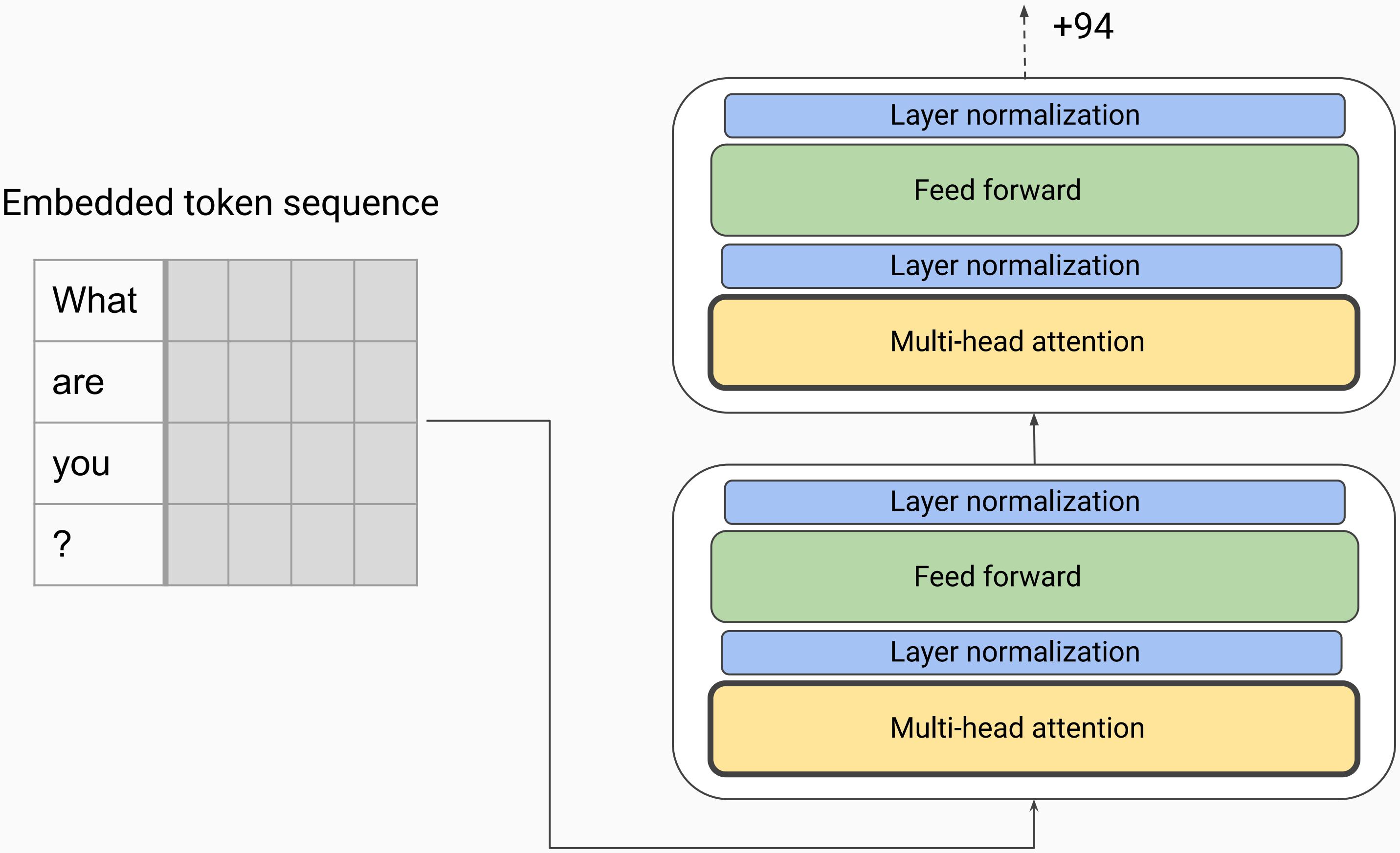
GPT end-to-end

GPT-3 overview



Token	p
I	0.8
cat	1e-4
…	…
dog	2e-5

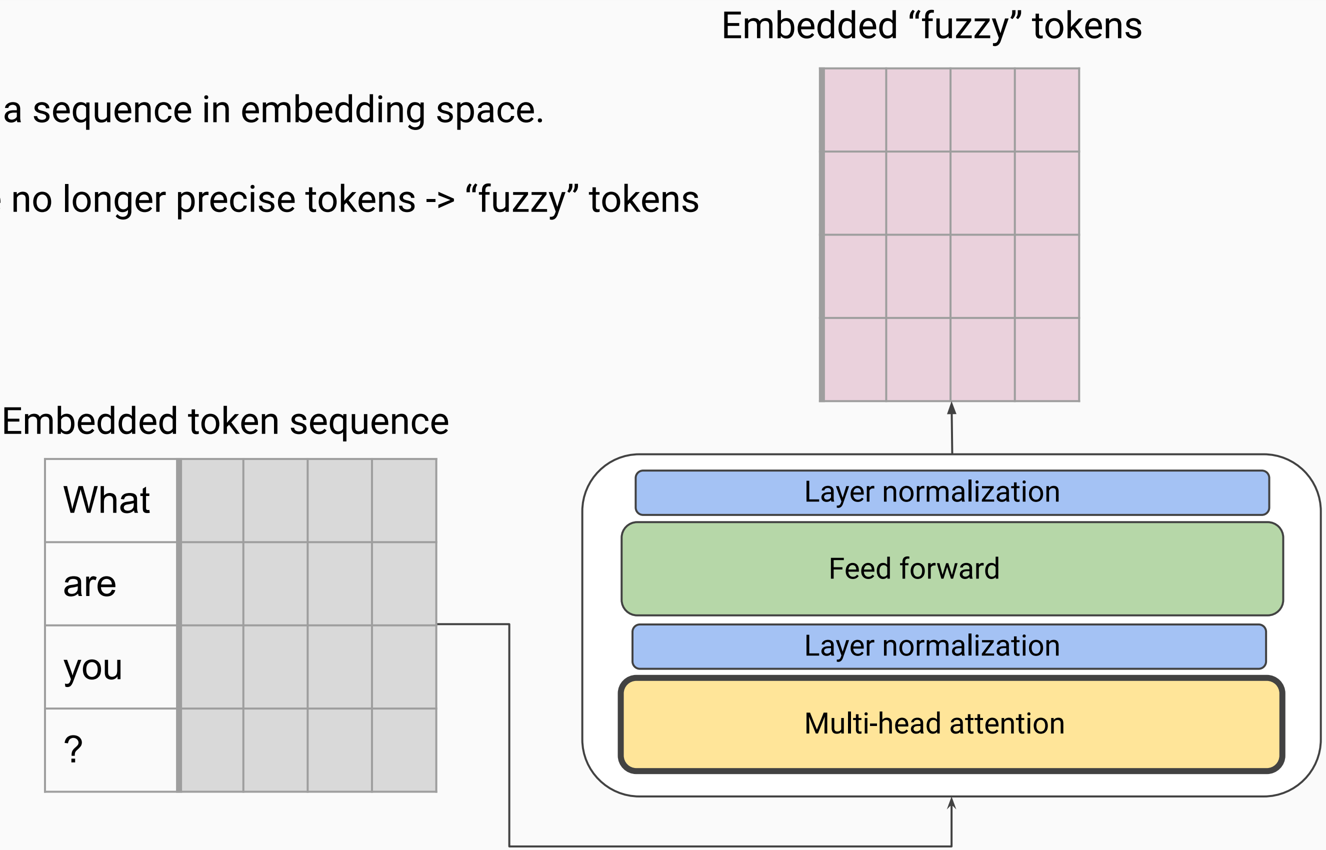
GPT Transformer layers



GPT Transformer, first layer

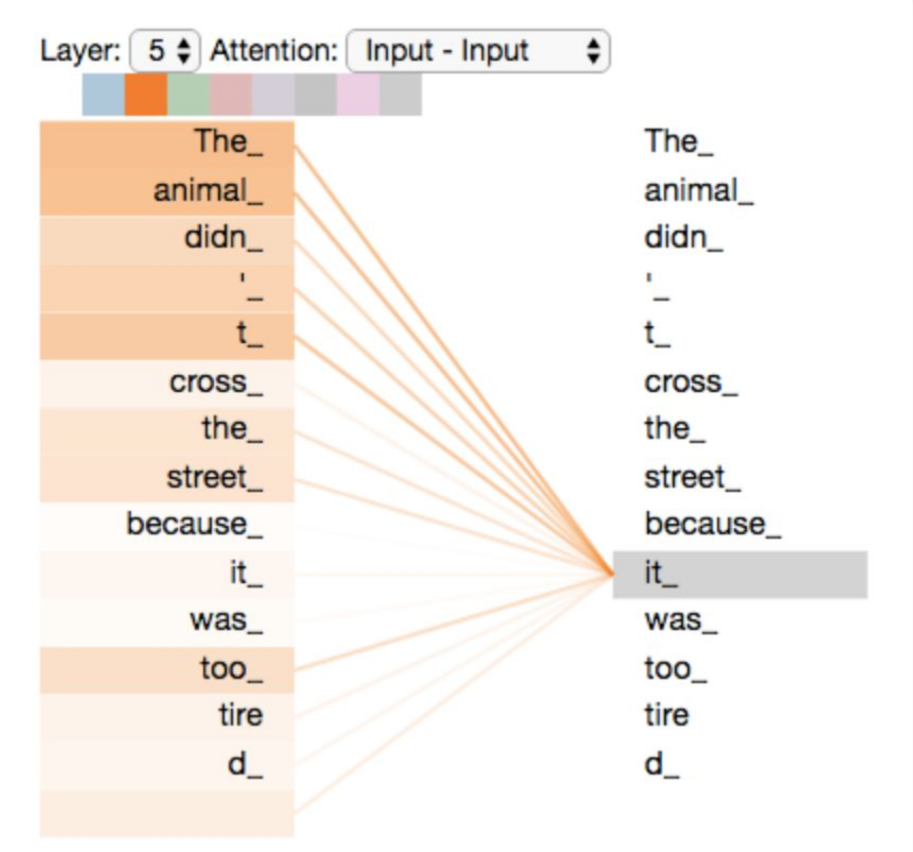
Each layer transforms a sequence in embedding space.

Transform outputs are no longer precise tokens -> “fuzzy” tokens



Attention mechanism

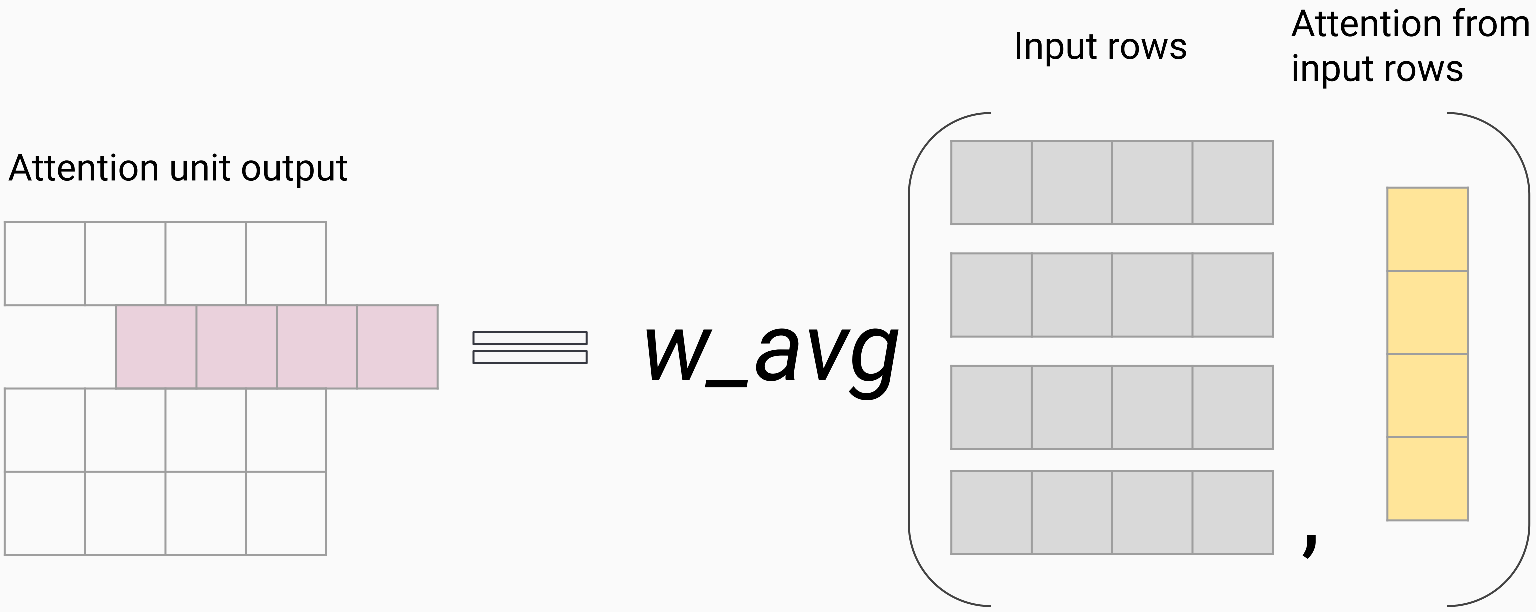
- The attention unit computes a degree of association between tokens in the input sequence - the “attention”
- Here, an attention unit learned to associate a pronoun - “it” - back to the sentence subject - “The animal”
- Different layers can focus on different kinds of associations
- Interpretation is hard since really these are “fuzzy” tokens in embedding space



From [Jay Alammar's Illustrated Transformer]

Attention unit output

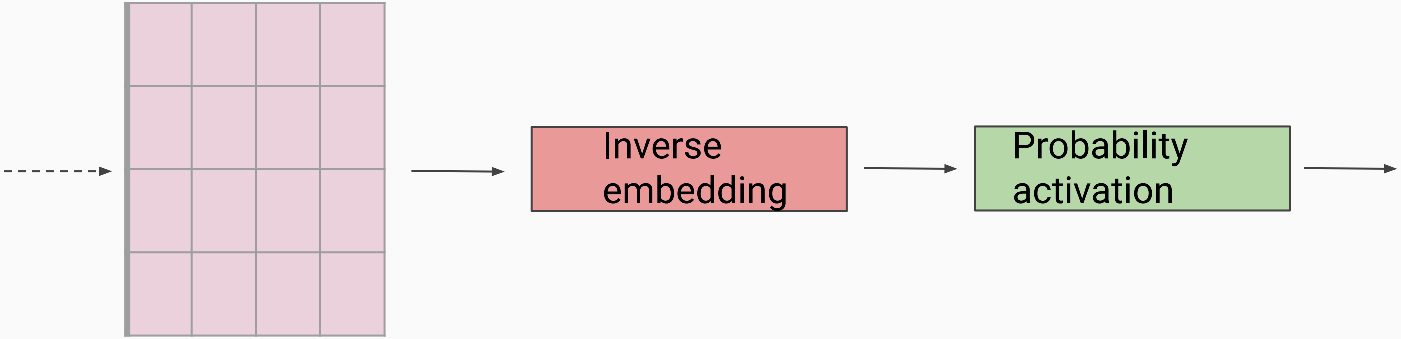
- Each output row is a weighted average of all input rows
- Weights are attention values for the output position from all other positions



Fuzzy tokens to probabilities

- A “fuzzy” token is a superposition of tokens in embedding space
  - Quantum token theory™
- The inverse embedding operation sends them back to “vocabulary” space
  - Proportions in the superposition can be interpreted as probabilities

Embedded “fuzzy” tokens



Token	p
I	0.8
cat	1e-4
…	…
dog	2e-5

Training GPT

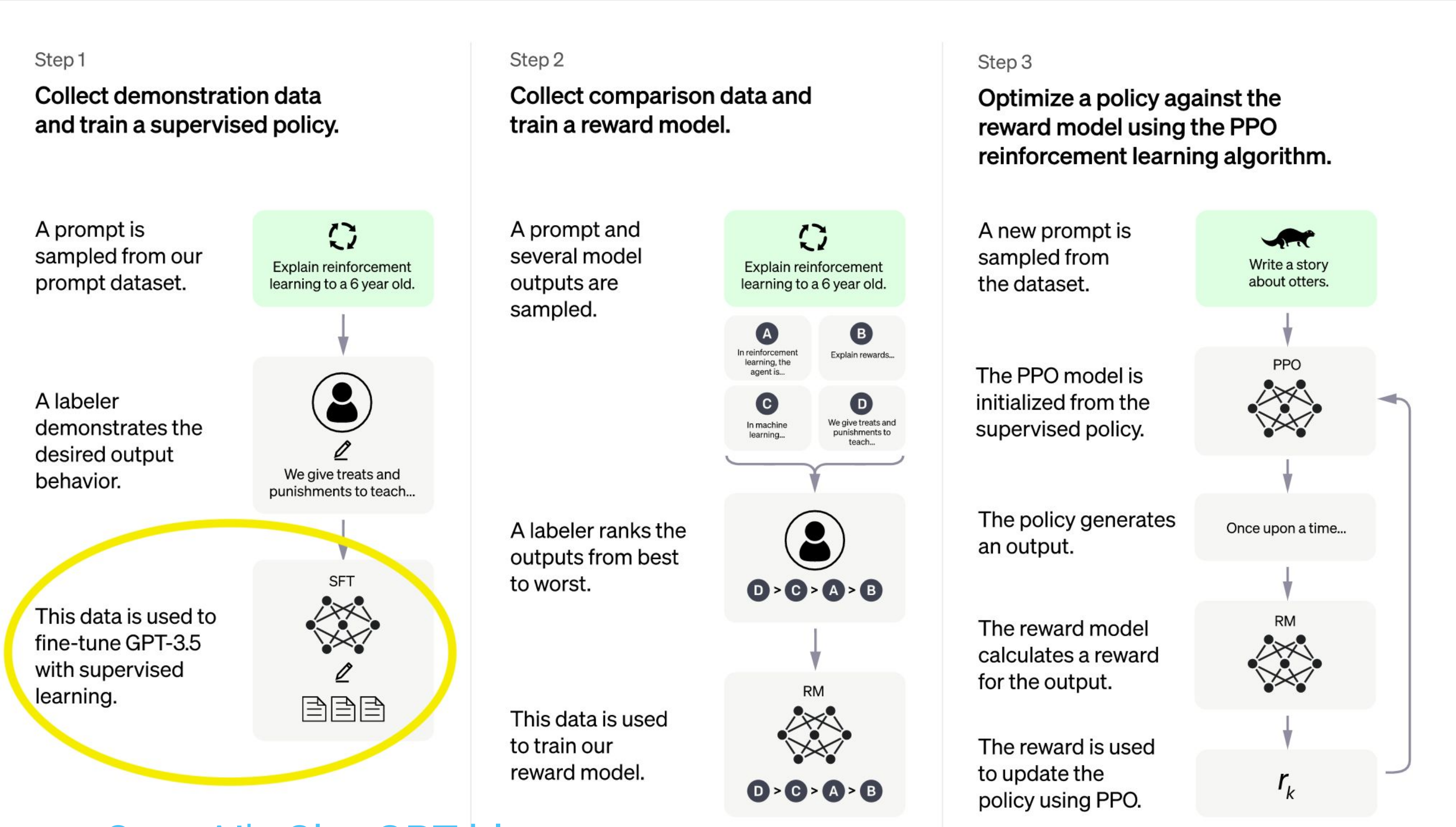
- Parse example text into chunks of 2048 tokens each
- Send chunks into GPT to predict the token following each input token
- Compare predictions with real values from text via a *loss function*
    - Loss function measures degree of difference between predictions and actuals
- Adjust GPT parameters to reduce the loss function
    - Stochastic gradient descent FTW
- Repeat until predictions are “close enough” to actual next tokens

Applications

Applications of GPT

- Building block for AI applications
- “Fine tuned” for domain and wired into other models
  - Fine tuning = update GPT parameters via new training data
- E.g. ChatGPT, GitHub Copilot, Jasper.ai

GPT as a component of ChatGPT



From [OpenAI's ChatGPT blog]

The “Bitter Lesson”

- Language structure learned via minimizing the loss function
  - GPT has no *a priori* knowledge
- Works via associations between words learned from billions of examples, **not from first principle understanding of language**
- Cons: model limited to what it sees in training
  - Reproduces biases, can be factually inaccurate, no internal morality
- But! Incredible facility given simple assumptions

References

- OpenAI's interactive tokenizer
- Byte pair encoding tokenization
- OpenAI's original GPT paper
- Google Brain's Transformer paper
- GPT model code on GitHub
- Jay Alammar's Illustrated Transformer
- OpenAI's ChatGPT blog
- The Bitter Lesson

Happy Hacking, Y’all!

What are some good ways to end a technical presentation?

1. Summarize the main points of the presentation.
2. Encourage questions and open discussion.
3. Offer resources for further information.
4. Provide contact information for follow-up.
5. End on a strong note, such as a demonstration or a call to action.
6. Thank the audience for their attention.
7. Provide a clear next step or next step for the audience to take.