Engineering with Generative AI

02 – LLM Foundations

Historia Magistra Vitae est

2000s	2010s	2020s
Word embeddings	Encoder-decoder Transformers Attention	LLMs
Part I	**Part II**	**Part III**

Intended Learning Outcomes

- Big picture: understanding natural languages and the difficulties in understanding and producing them via computers.
- The foundations of effective modern methods for deep learning applied to language modelling
  - Basics first, then key methods used in NLP
  - then we focus on BERT as a transformer-based language model
  - until we get to our large language models

Agenda

- NLP Foundations
- Transformer Architecture
- Scaling Language Models

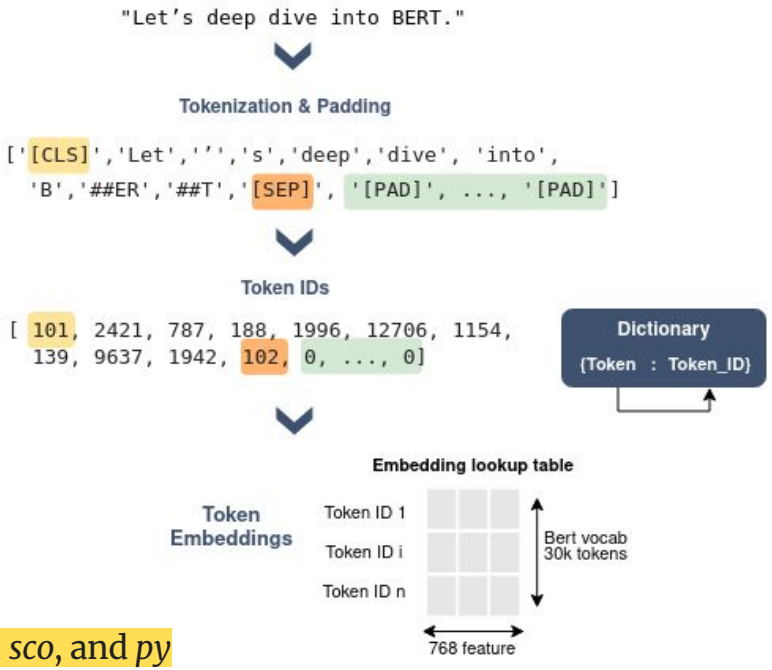
What can we do with NLP?

May seem overwhelming but most architectures are similar or identical.

Word Representation

In order to represent words in LLM, we utilise two main technologies:

- Tokenization
- Embeddings



echoscopy into *echo*, *sco*, and *py*

ME

Hi ChatGPT, How do you represent my words internally? what technology do you use to represent my words?

I represent your words internally using a technology called deep learning. Specifically, I am based on OpenAI's GPT-3.5 architecture, which stands for "Generative Pre-trained Transformer 3.5." The core technology is a type of neural network known as a transformer.

Here's a simplified explanation of how it works:

1. **Tokenization:** When you input text, it is first broken down into smaller units called tokens. Tokens can be as short as one character or as long as one word.
2. **Embedding:** Each token is then converted into a high-dimensional vector representation. This process is called embedding. The embedding captures semantic relationships between words, allowing the model to understand the context and meaning of the input.
3. **Model Architecture:** GPT-3.5 is a transformer model, which is a type of neural network architecture. It uses attention mechanisms to process input data in parallel and capture relationships between different parts of the input text.
4. **Training:** The model has been pre-trained on a diverse range of internet text, learning grammar, facts, reasoning abilities, and also some biases present in the data.
5. **Inference:** When you ask a question or provide input, the model uses its learned parameters to generate a response. It considers the context provided by the input tokens and produces a coherent output.

It's important to note that while I can generate responses and understand context, I don't have consciousness, self-awareness, or personal experiences. I process and generate text based on patterns and information present in the data on which I was trained.

Tokenization

- Tokenization helps in assigning a numerical representation to each token by:
  - → Recording each unique token found during tokenization in a set.
  - → After all sequences (e.g., sentences) in the dataset are tokenized, the set will be a set of all unique tokens (e.g., words) in the dataset. It's called the vocabulary at this point.
  - → Assign some unique number (ID) to each token in the vocabulary

[7993, 170, 11303, 1200]

- We now have numbers that could be fed to machines!

Word Embeddings

- Now we had numerical representation for each word/token in the text. But this does not represent the word/token **meaning**:
  - In the Cambridge Dictionary the definition of "**Meaning**" is

    'the sense in which a statement, action, **word** etc is (intended to be) understood'
  - So, how can we teach a token's meaning to computers?
    - Representing words by their context
    - **J.R.Firth 1957**
      - "You shall know a word by the company it keeps"
      - One of the most successful idea of modern statistical NLP!

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

banking meaning represented in the context

Basics of language modeling

P(pseudoscience | why is nlp a)

$$P(\boxed{w_n} | \boxed{w_1^{n-1}})$$

next word

history

Central question: How do we compute these probabilities?

Basics of language modeling

Simplest case of an LM: Bigram model

$$P(w_n|w_1^{n-1}) \approx P(w_n|\boxed{w_{n-1}})$$

History of size 1:
P of next word only depends on last word
(Markov assumption)

$$P(pseudoscience|why\ is\ nlp\ a) \approx P(pseudoscience|a)$$

Basics of language modeling

So how do we get $P(w_n|w_{n-1})$?

Simple: Counting!

$$P(dog|hot) = \frac{C(hot\ dog)}{C(hot\ _)}$$

count of occurrences of „hot dog“ in a large corpus

count of occurrences of „hot <*some word*>“ in a large corpus

Basics of language modeling

Given this, we can predict words!

Ann likes to eat ?

For each word *w* that we know, compute

$$P(w|eat)$$

…then sample from the most likely words

Ann likes to eat burgers

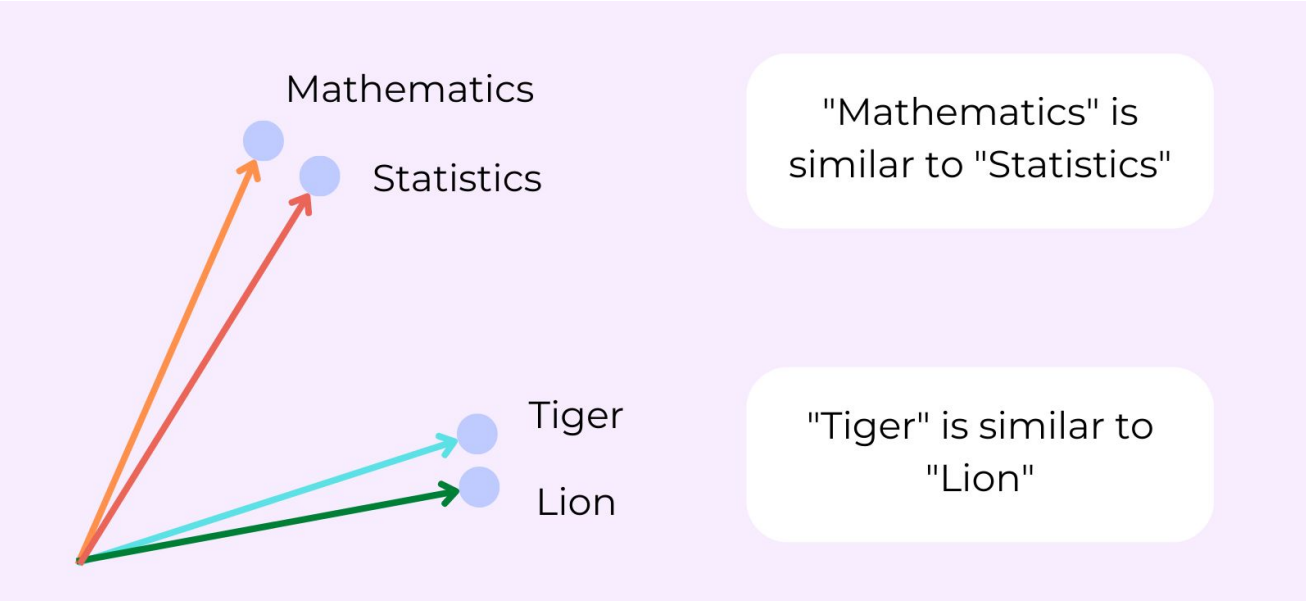
Problems

- Unseen words/bigrams: *Ann likes to engulf* ?
  - Sparse data problem
  - Smoothing to the help in some cases
- Lack of context: *Homer likes to eat* ?
  - Can use larger n-grams
  - But that greatly increases data sparsity
- Counting in general is a very brute force and un-flexible means of representing language ☺

Word Vectors

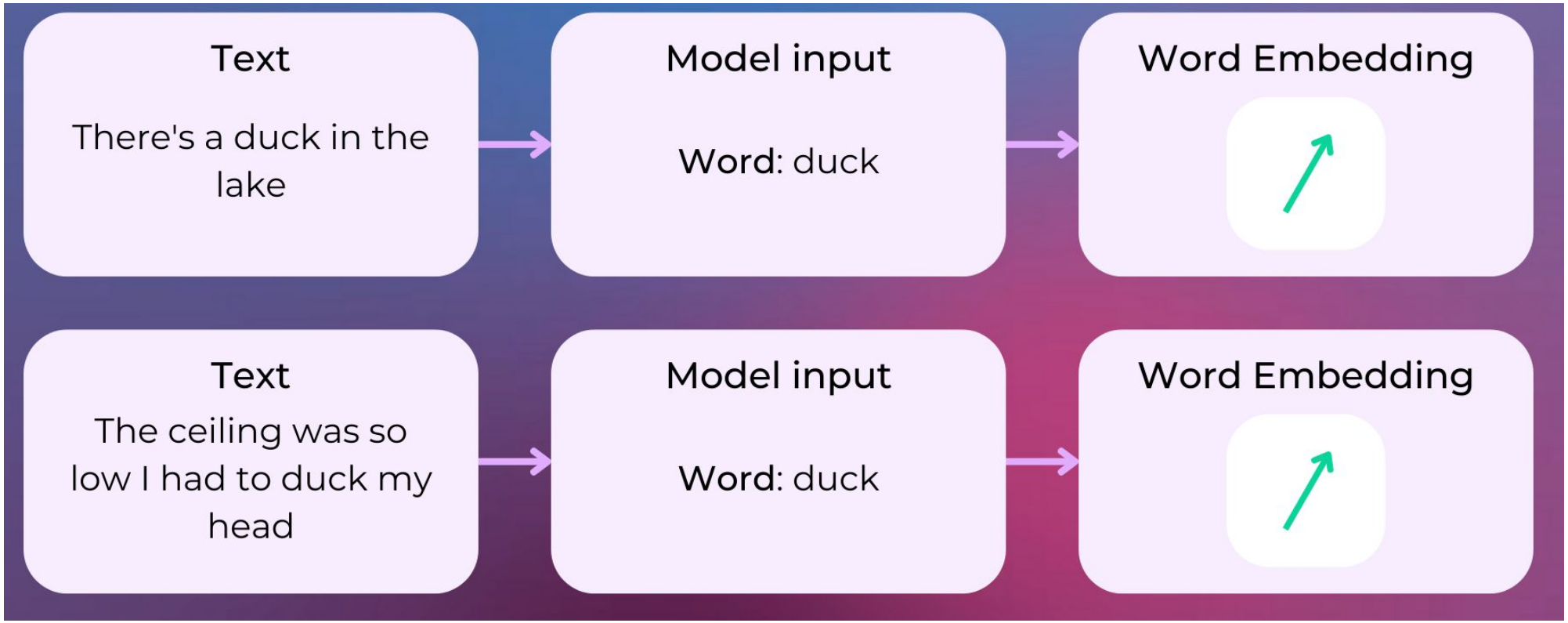
- Build a new model of meaning focusing on similarity
  - Each token/word is a vector
  - Similar tokens/words are “nearby in space”

$$\text{Mathematics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$



Context-independent Embeddings

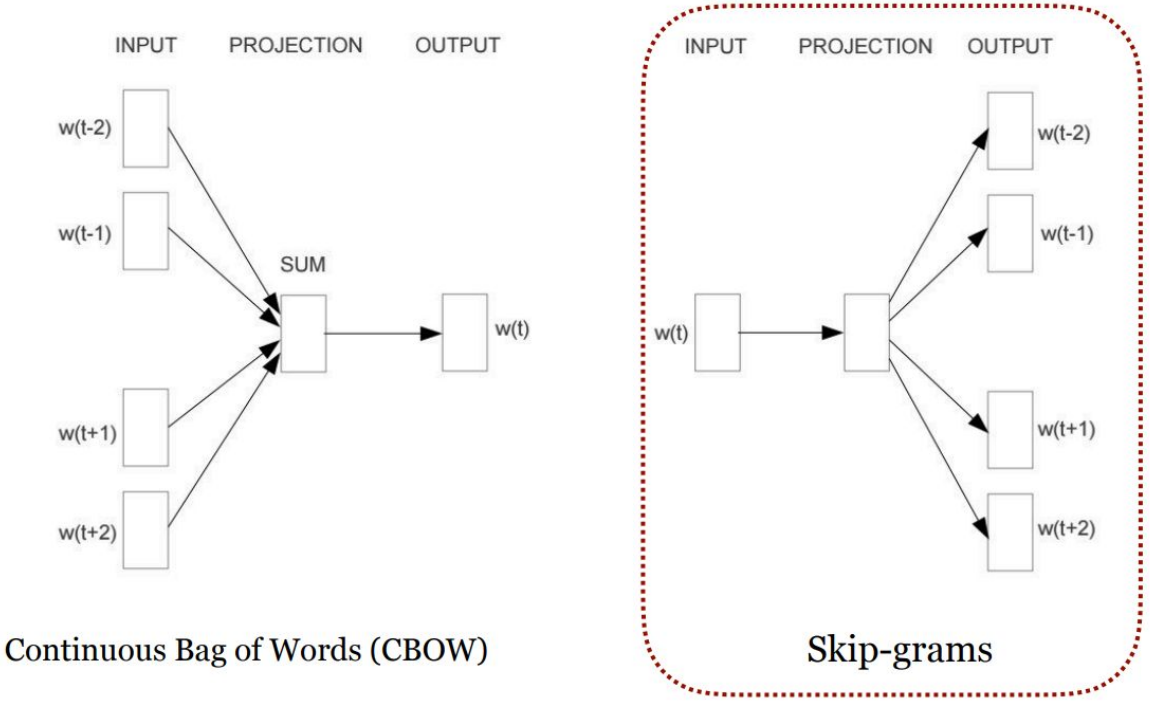
- This model type learns one meaning of the meanings of the words and stores it for use later.
- Word **duck** will have only one meaning and one vector



Example: Word2Vec

- **Input:** V, d, a large text corpora
  - V: a predefined vocabulary set [cat, dog, …etc]
  - d: dimension of word vectors (e.g. 300)
  - Text corpora:
    - Wikipedia: 3B
    - Common Crawl: 840B

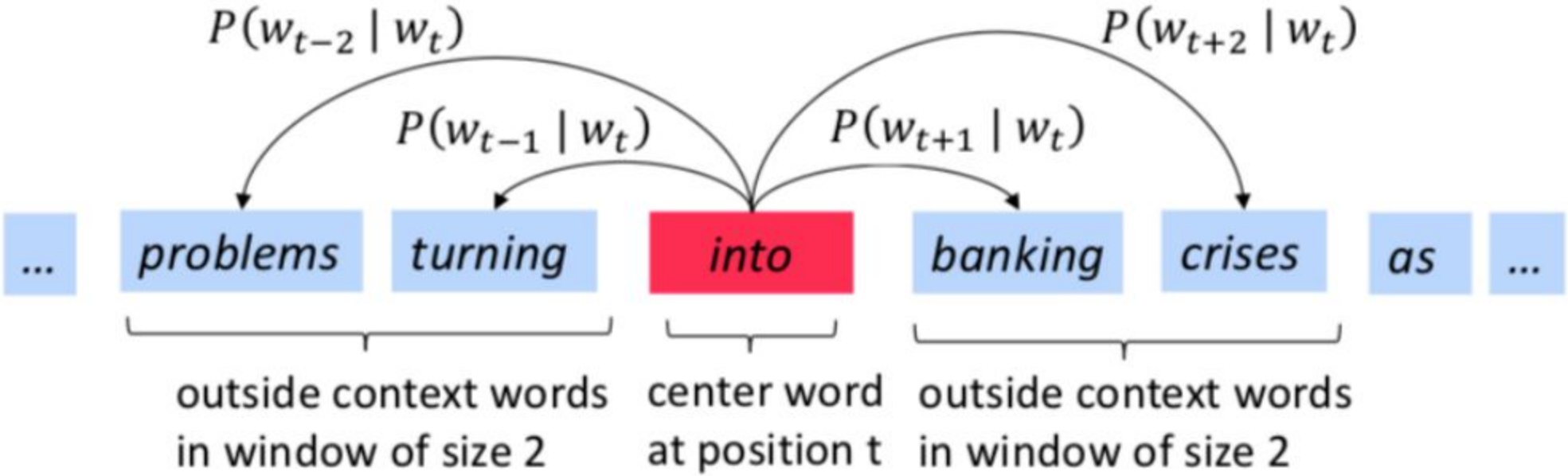
- Output: $f : V \rightarrow \mathbb{R}^d$



Word2Vec : Two variants

Example: Word2Vec

- Skip-grams



https://courses.cs.washington.edu/courses/csep517/20wi/slides/csep517wi20-WordEmbeddings.pdf

Learned word embeddings

- We have two sets of vectors for each word in the vocabulary

$$\mathbf{u}_i \in \mathbb{R}^d \text{ : embedding for target word } i$$

$$\mathbf{v}_{i'} \in \mathbb{R}^d \text{ : embedding for context word } i'$$

- Use inner product $\mathbf{u}_i \cdot \mathbf{v}_{i'}$ to measure how likely word i appears with context word i', the larger the better

$$P(w_{t+j} \mid w_t) = \frac{\exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_{w_{t+j}})}{\sum_{k \in V} \exp(\mathbf{u}_{w_t} \cdot \mathbf{v}_k)}$$

Context is Key

- Context is crucial for accurately interpreting the meaning of words and phrases

- It allows algorithms to discern nuances, such as sarcasm, idiomatic expressions, and homonyms, leading to more precise and human-like understanding of language.

Language Models

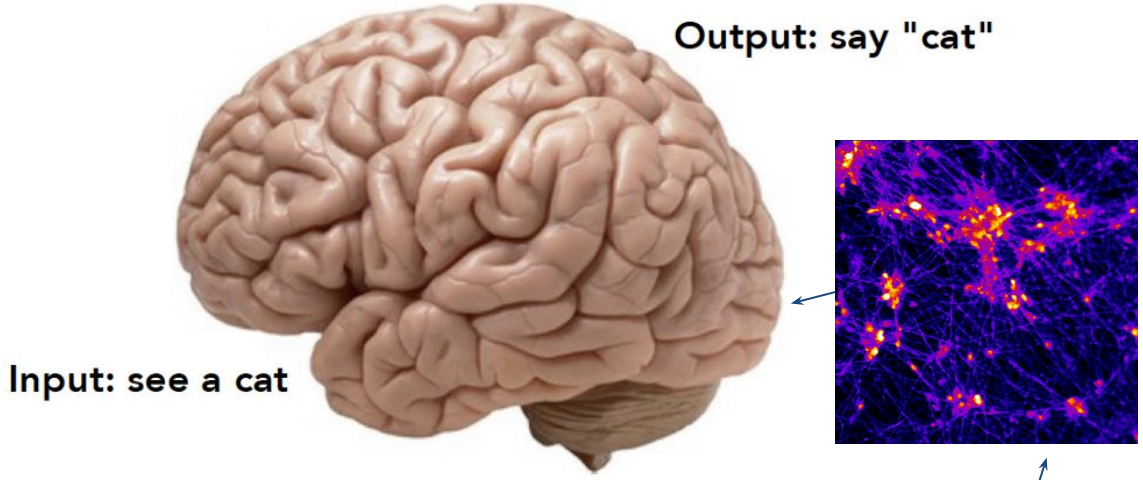
- Language can be modelled as a task of predicting the next token given a sequence of tokens:
  - E.g., "I ran for two miles today and I feel so _____"
  - What are options of high probability for the next word?

- Mathematically, a language model:
  - given token $w_1, w_2, \ldots, w_t$
  - predict the next one w_{t+1}
  - If we know $P(w_{t+1} = w | w_1, w_2, \ldots, w_t)$
    - we can compute probability for any word sequence
    - $P(w_1, w_2, \ldots, w_t) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2, w_1) \cdot \ldots \cdot P(w_t|w_{t-1}, \ldots, w_1)$
    - we can iteratively generate text (by sampling or max prob token)

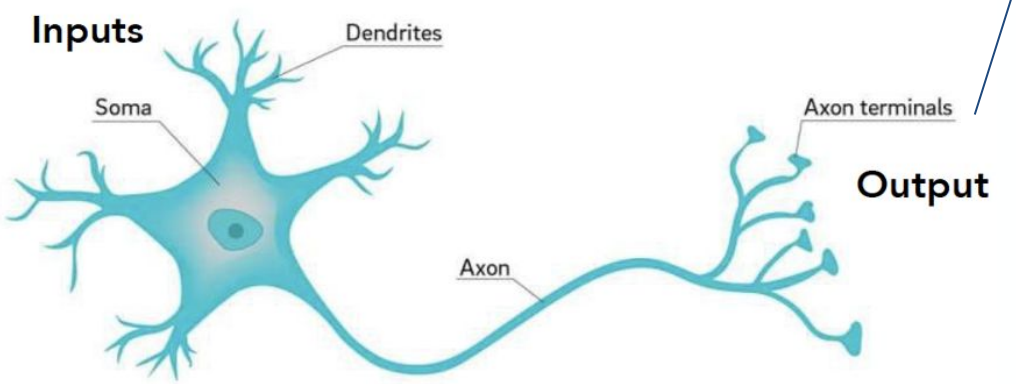
Machine Learning: Mathematical Perspective Supervised Learning

- **Given**: Training dataset (x_i, y_i) where $y_i = f(x_i) + \varepsilon$ and f is unknown
- **Wanted**: g(x) that is as close as possible to f(x)
- **Approach**:
  - Choose a hypothesis set (i.e., mathematical form for $g(x;\Theta)$) with unknown parameters Θ.
  - Choose a loss function L that measures how close f(x) and $g(x;\Theta)$ are
  - Use an optimizer A, procedure to adjust parameters Θ minimizing the loss
- **Choosing an ML model and its hyperparameters corresponds to choosing g, L and A.**
  - Using A to learn the best g is called training the model

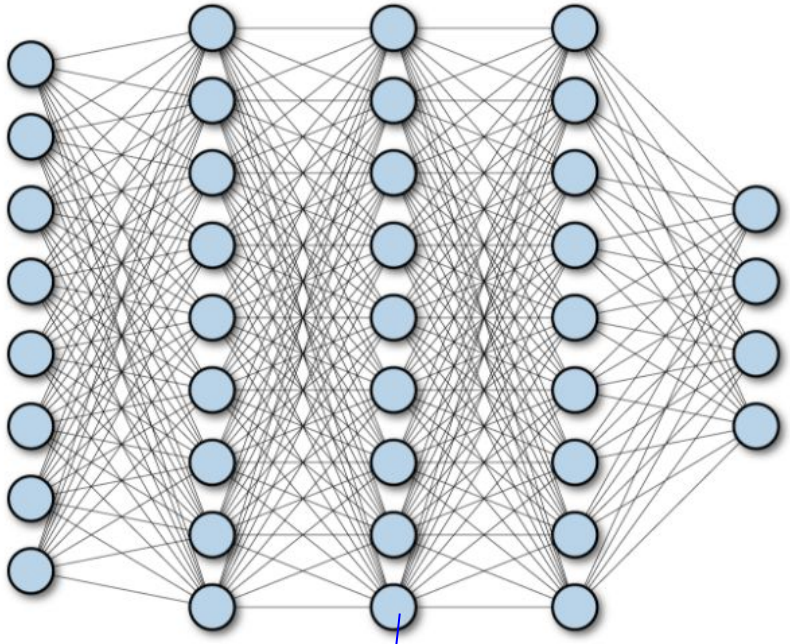
Deep Learning in a Nutshell

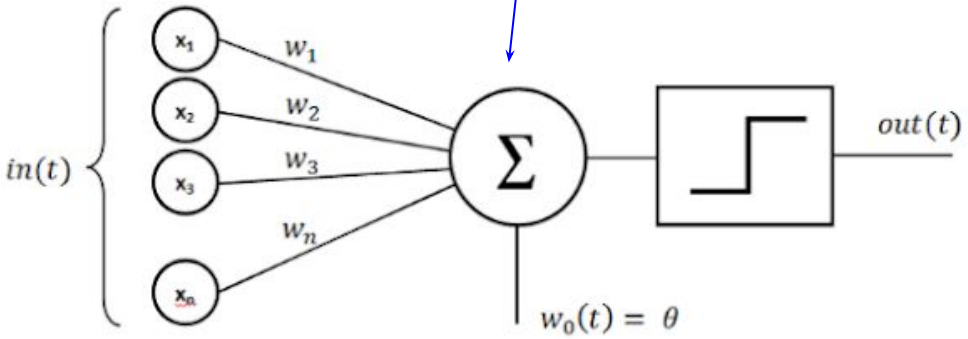


https://www.the-scientist.com/the-nutshell/what-made-human-brains-so-big-36663



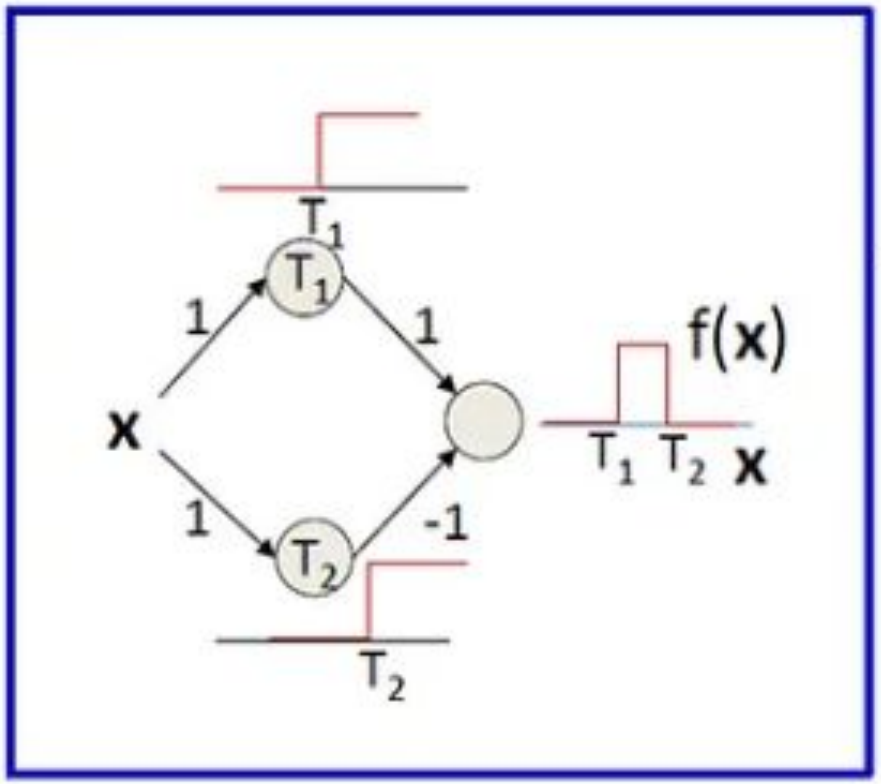
https://medicalxpress.com/news/2018-07-neuron-axons-spindly-theyre-optimizing.html

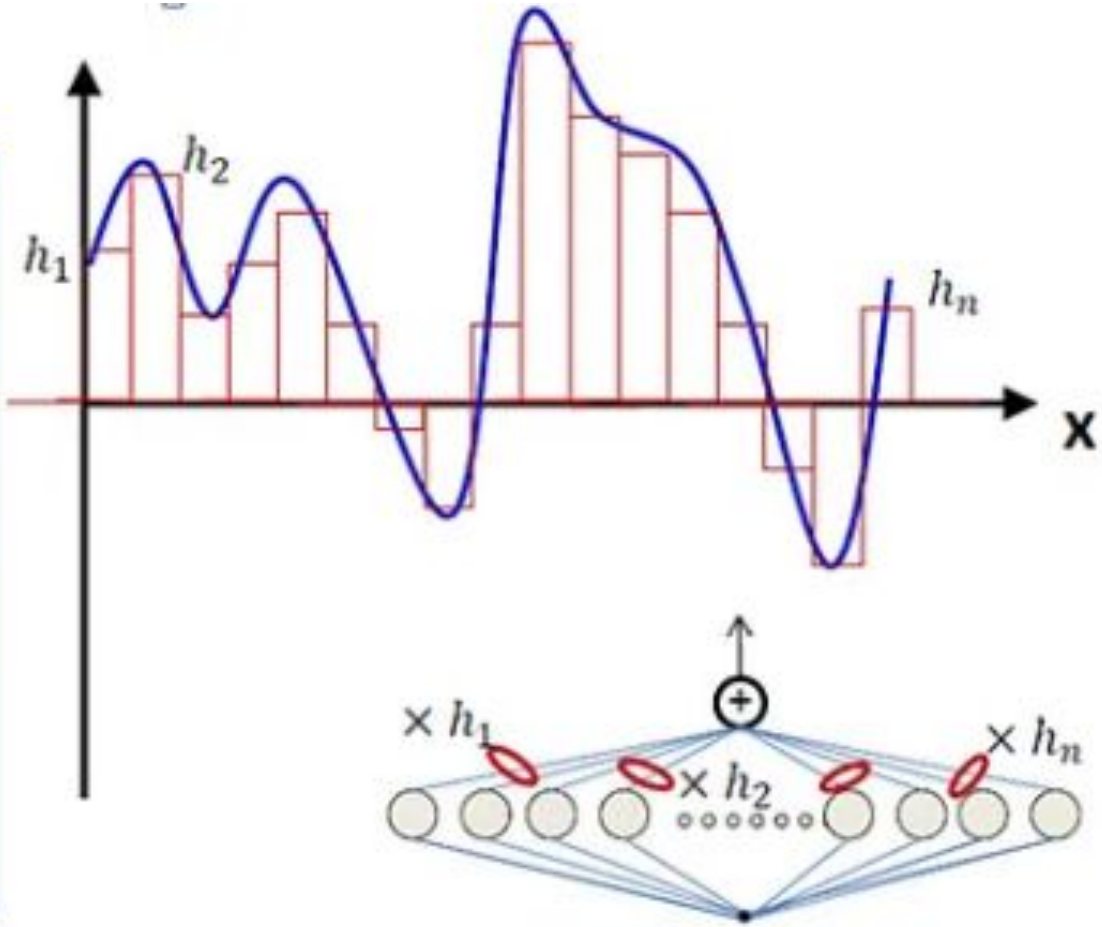




Universal Approximation

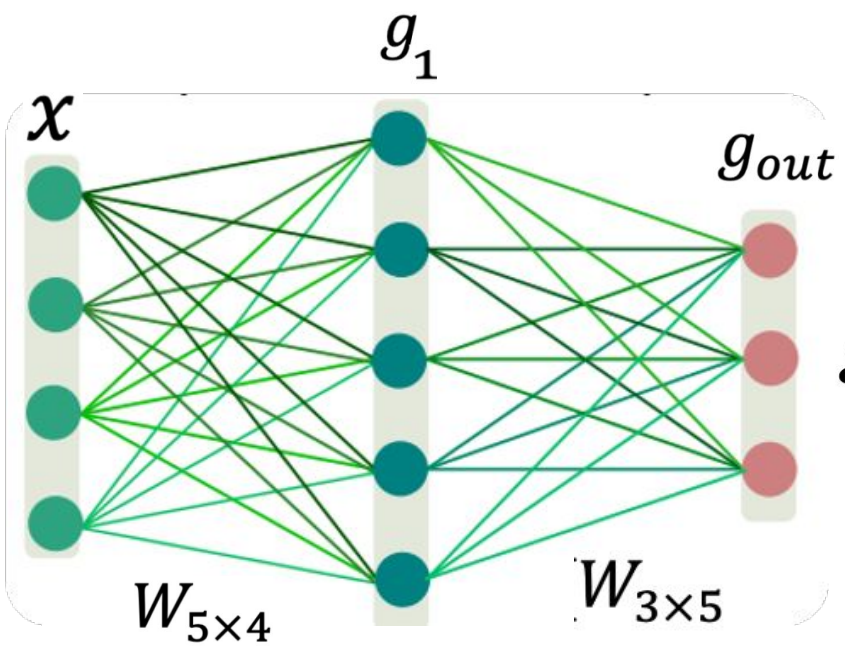
For any function and for a given error level, there is a neural network that approximates a function given to that error (in fact 2 layers are enough).





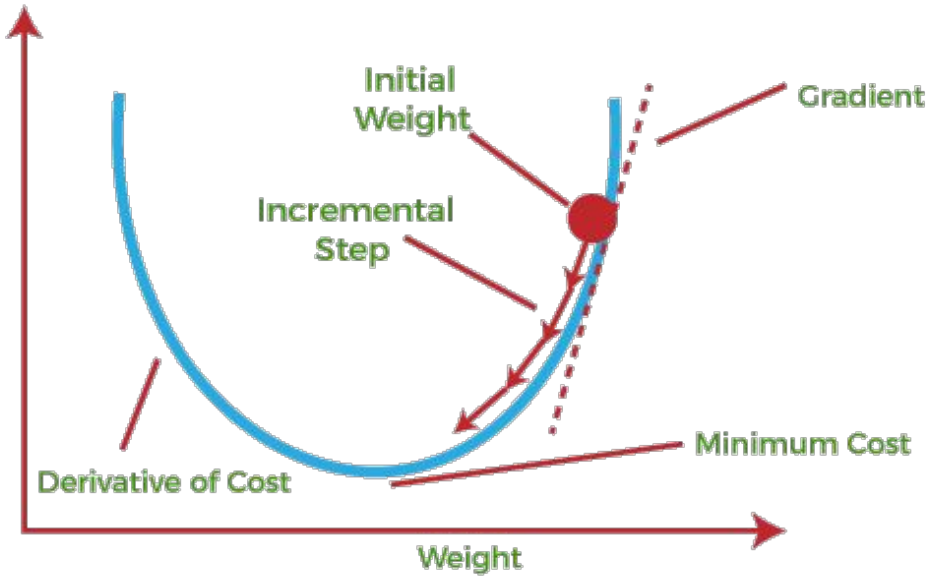
A Neural Network is just a Function

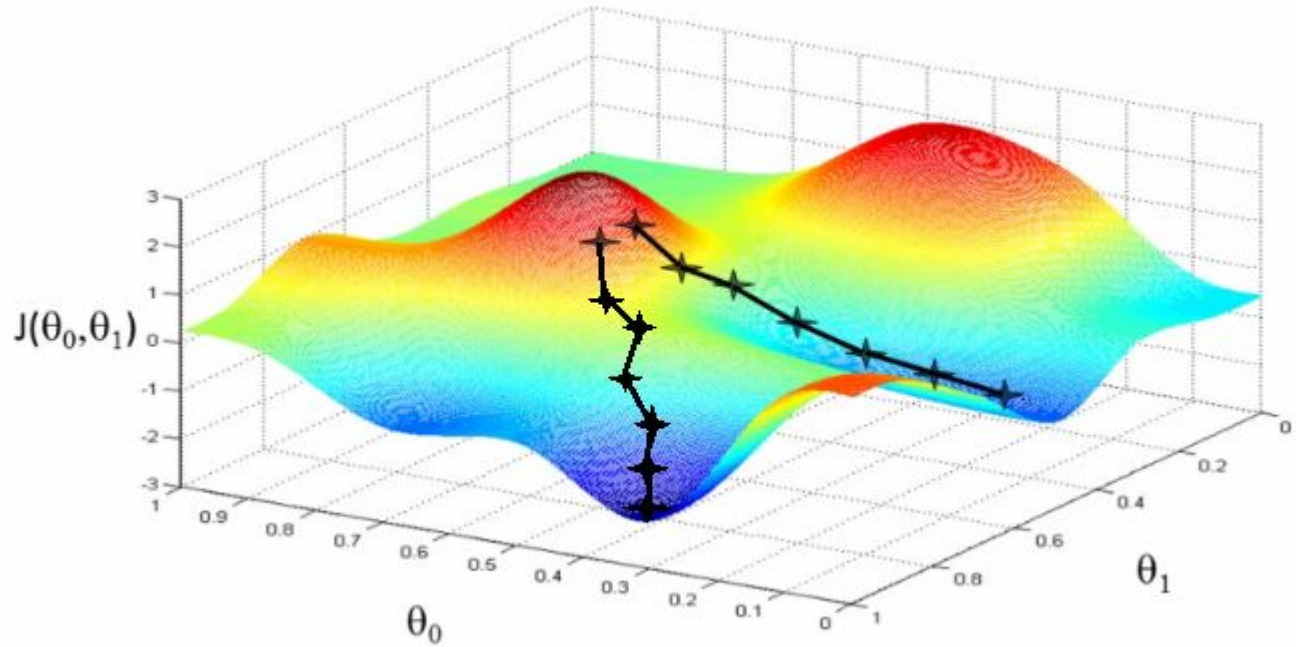
Feed forward Neural Network operations are just matrix multiplications and non-linearities.



$$g_{3\times1}(x) = g_{out}(W_{3\times5}g_1(W_{5\times4}x_{4\times1} + b_{5\times1}) + b_{3\times1})$$

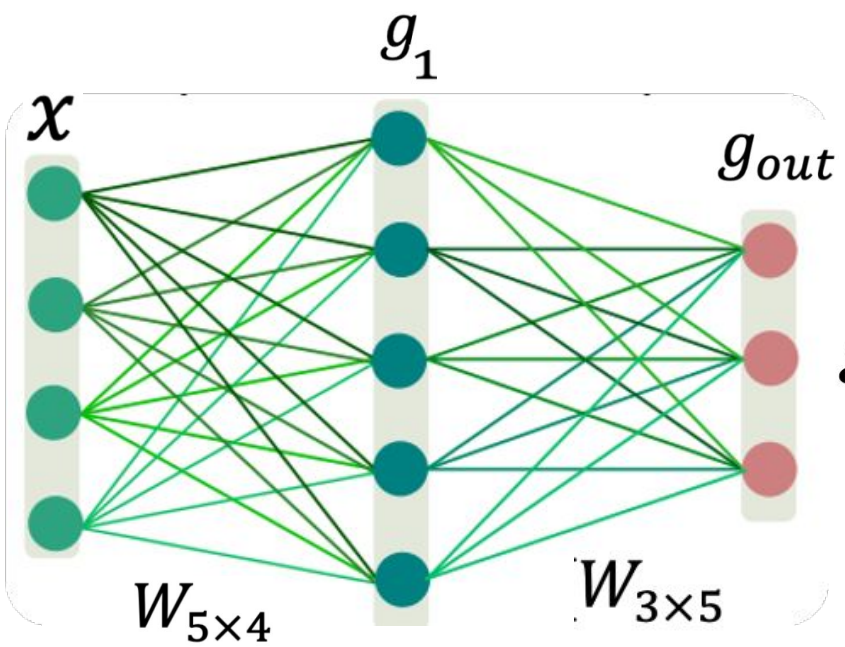
Gradient Descent





A Neural Network is just a Function

Feed forward Neural Network operations are just matrix multiplications and non-linearities.



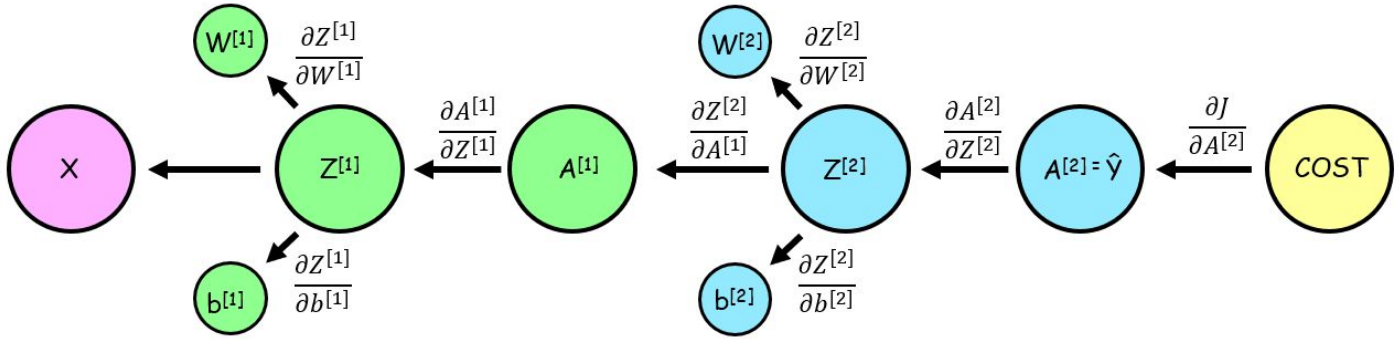
$$g_{3\times1}(x) = g_{out}(W_{3\times5}g_1(W_{5\times4}x_{4\times1} + b_{5\times1}) + b_{3\times1})$$

Backpropagation

- Our objective is to find the weights in the network that minimize the loss function L
- For this, we need to find ∂L/∂θ for each parameter θ
  - Then we can apply gradient descent (adjust weights) to minimize the loss:

$$\theta_{\text{new}} = \theta_{\text{old}} - \alpha \frac{\partial L}{\partial \theta}$$

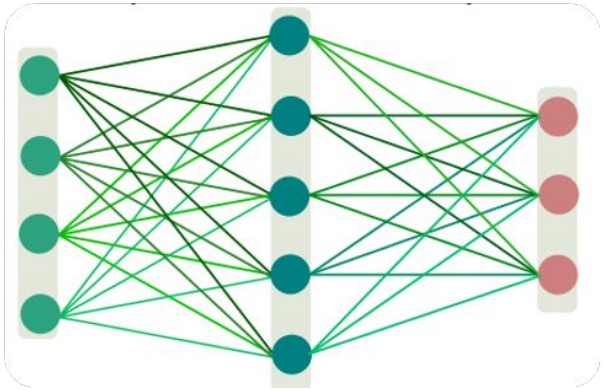
- Backpropagation algorithm helps us to compute ∂L/∂θ
  - Simply apply the chain rule → ∂L/∂θ in later layers depend on earlier ones → name

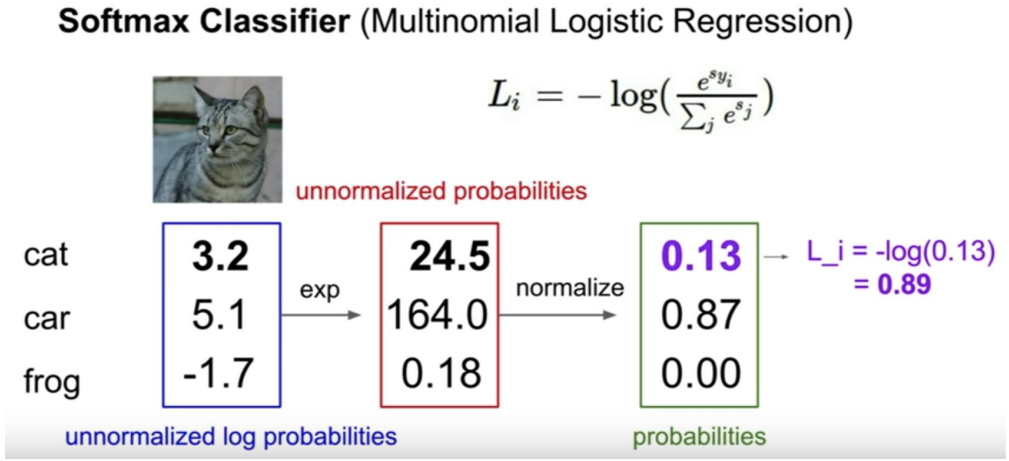


Deep learning frameworks handle this automatically.

Machine Learning: Mathematical Perspective Supervised Learning

- **Given**: Training Dataset $((w_1,\ldots,w_{t-1}),w_t)$ where $y_i \sim p(y|x,\Theta)$ and f is unknown
- **Wanted**: g(x) that is as close as possible to f(x)
- **Approach**:
  - word2vec(w_1)
    .
    .
    .
    word2vec(w_{t-1})





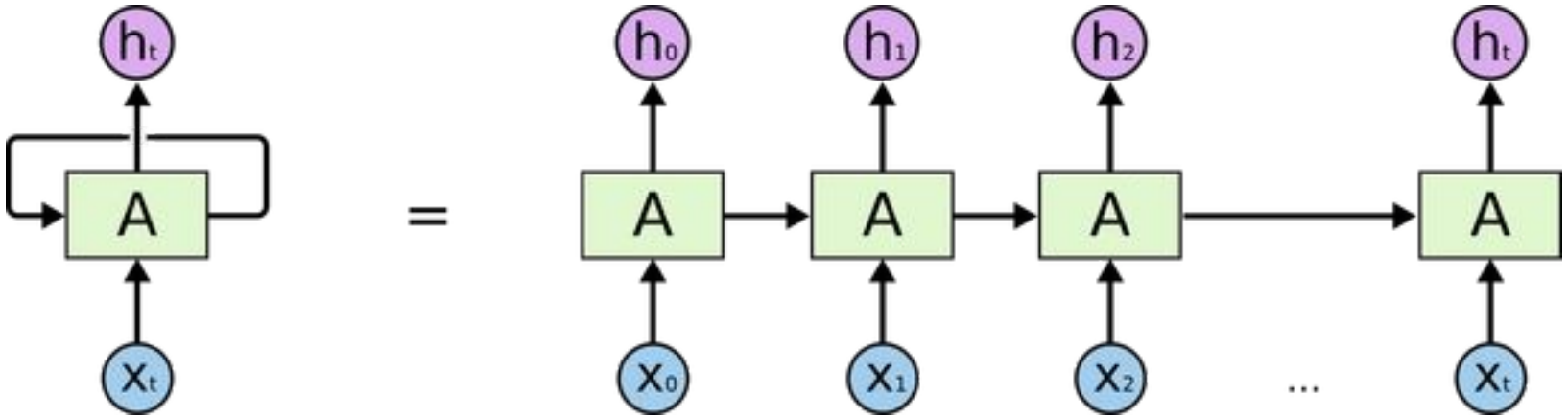
Problem? Texts of various lengths

Sequence Models: RNNs

- Recurrent Neural Networks (RNNs) are like feedforward neural networks except that they use one or more recurrent layers
- A recurrent layer takes the form

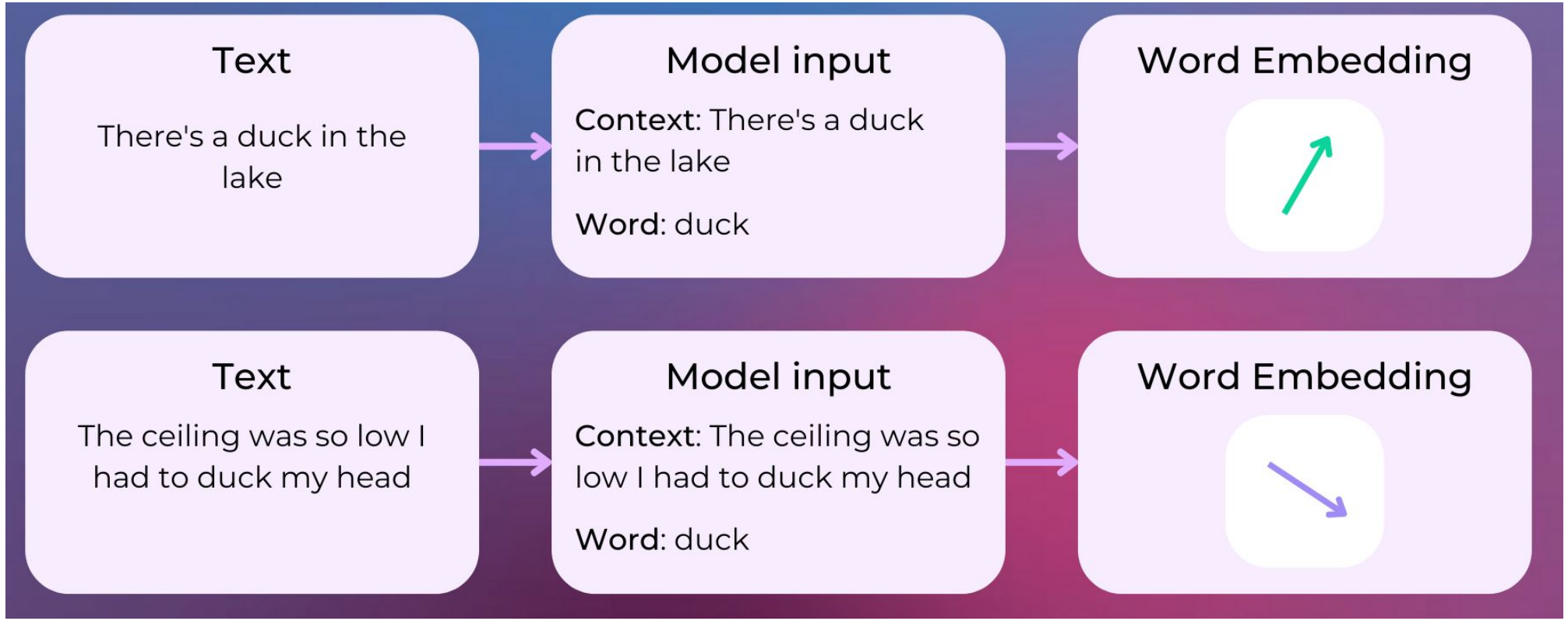
$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

- RNNs are trained using a method called backpropagation through time (BPTT)



Context-dependent Embeddings

This model type will infer word vectors based on context:

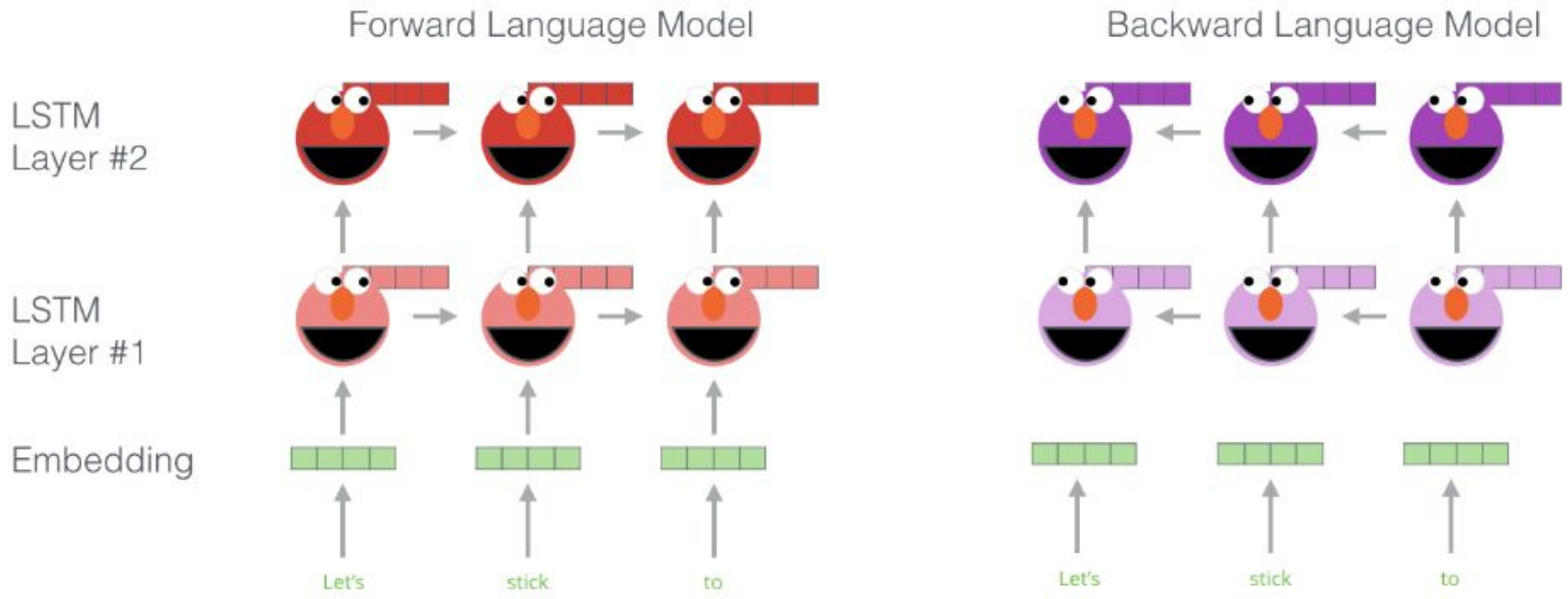


Example: RNN-based (ELMO)

- Input: Word and context.

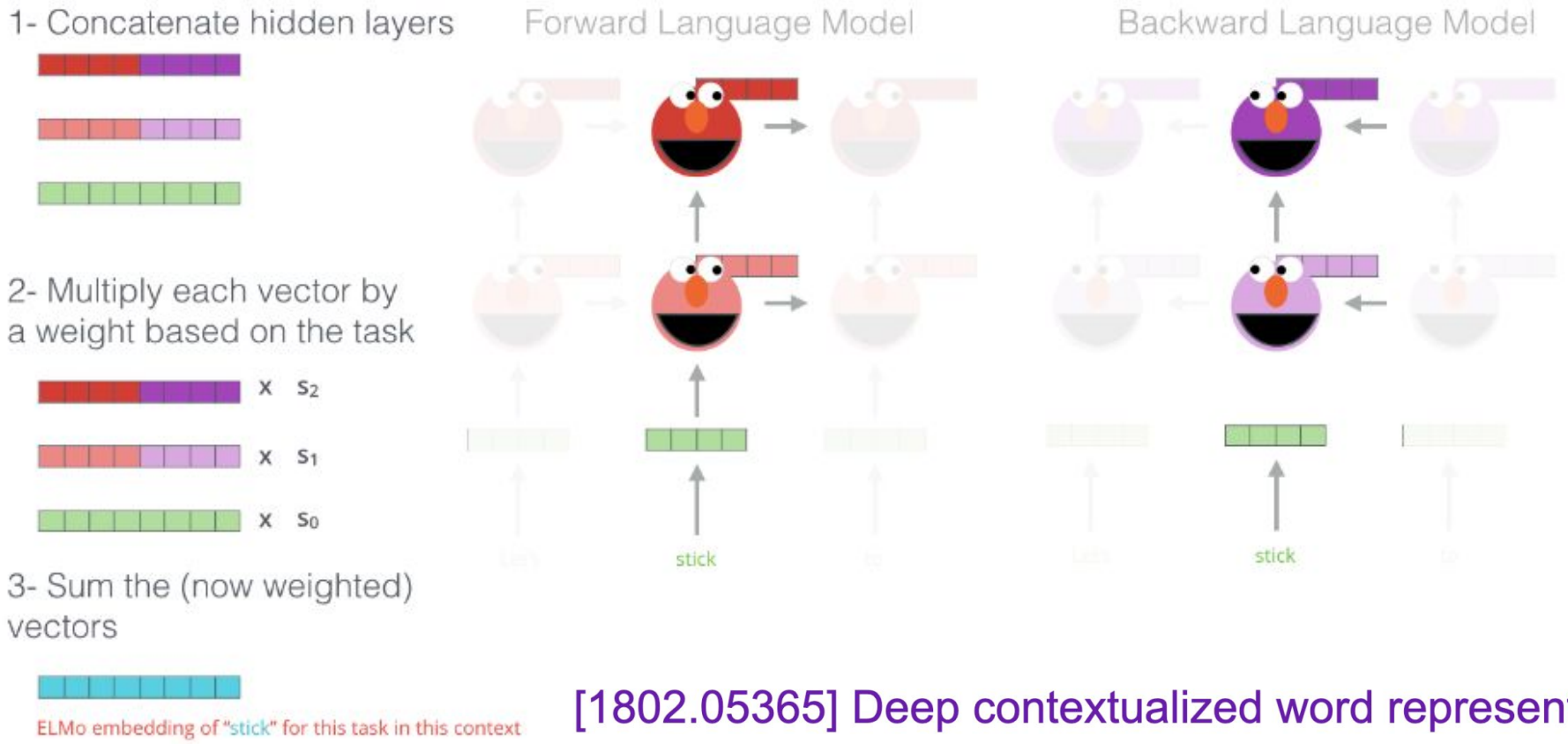
Example: RNN-based (ELMO)

- Step 1: Train two layers Bidirectional LSTM models



Example: RNN-based (ELMO)

- Step 2: take the initial embedding and hidden states of the targeted word and concatenate them to get final contextualized embeddings:



[1802.05365] Deep contextualized word representations

by ME Peters · 2018 · Cited by 13996 — Abstract: We introduce a new type of deep contextualized word representation that models both (1) complex characteristics of word use…

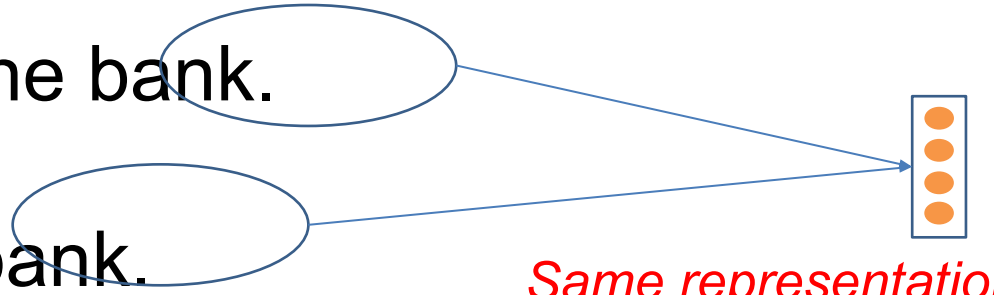
https://jalammar.github.io/illustrated-bert/

Small excursion: Distributional semantics

Problem with Word2Vec, count-based methods etc: Representation independent of context!

I‘ve got tons of money at the bank.

Nice flowers grow on the bank.

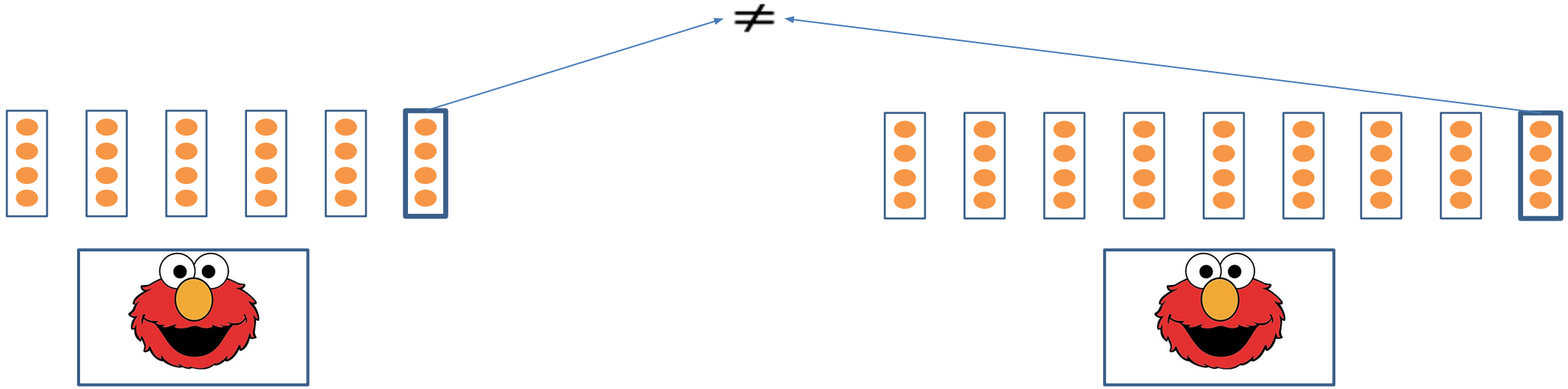


Same representation in Word2Vec!

Small excursion: Distributional semantics

Back to language models: ELMo to the help!

Representation depends on context!



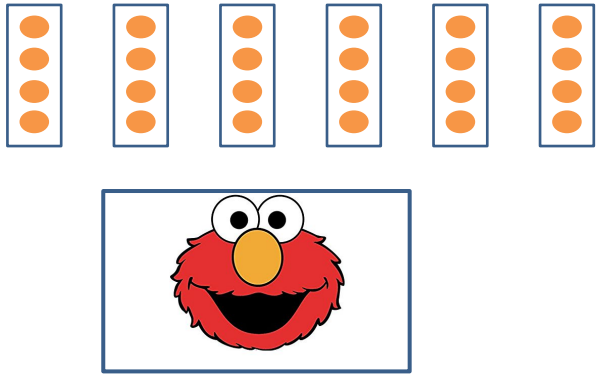
Nice flowers grow on the **bank**

I have got tons of money at the **bank**

Back to LMs…

What to do with such representations?

Use them as features/Embeddings!



Nice flowers grow on the bank

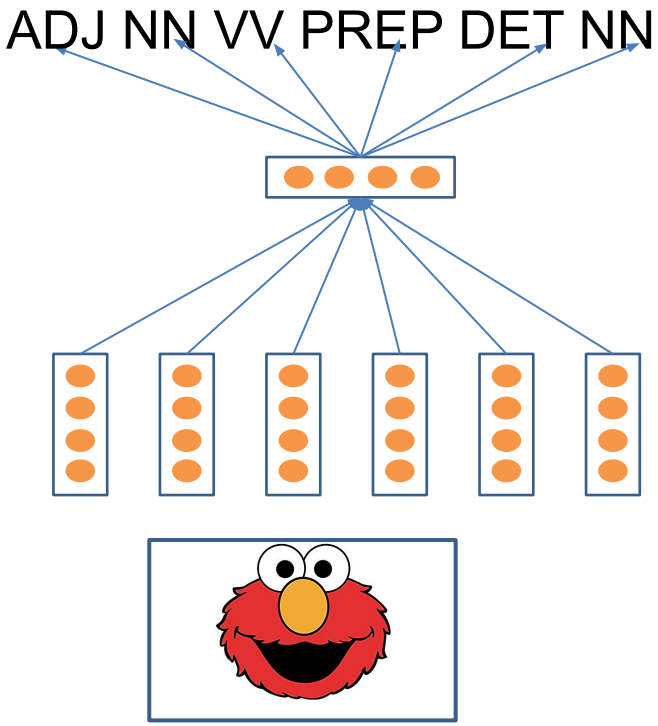
Back to LMs…

What to do with such representations?

Use them as features/Embeddings!

POS tags

classifier layer



Nice flowers grow on the bank

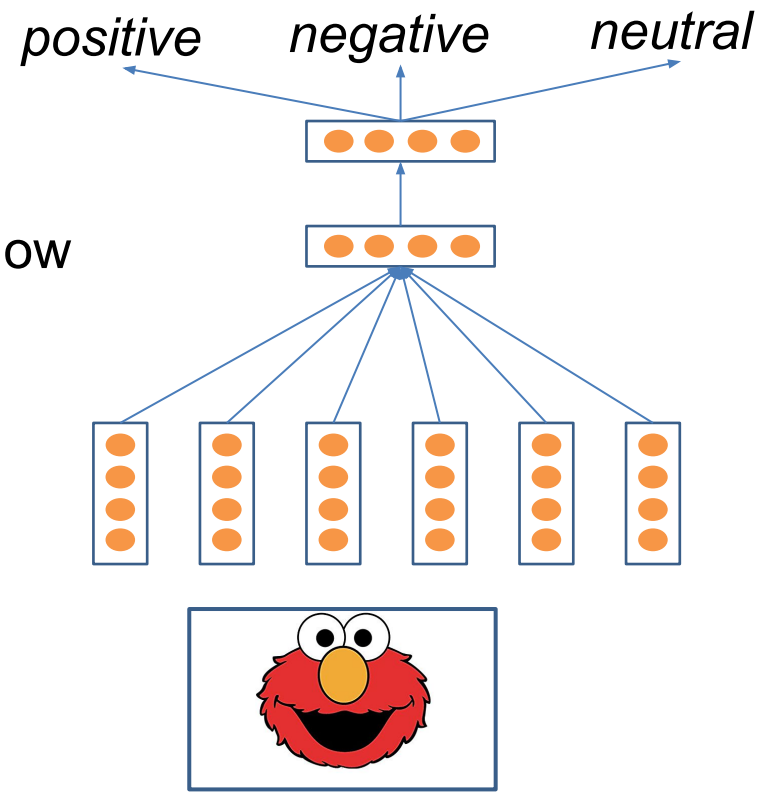
Back to LMs…

What to do with such representations?

classification

Hidden layer

Sum/average/concat/combine somehow



Nice flowers grow on the bank

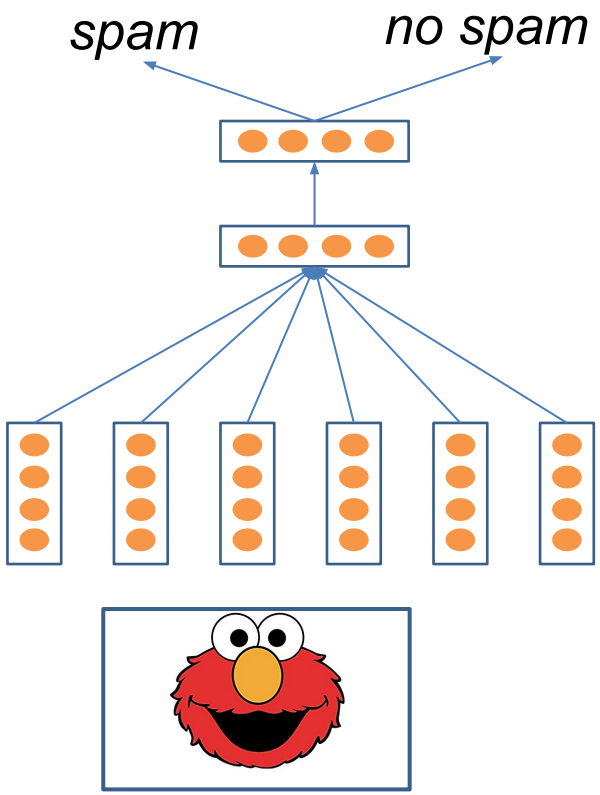
Back to LMs…

What to do with such representations?

classification

Hidden layer

Sum/average/concat/combine somehow



Nice flowers grow on the bank

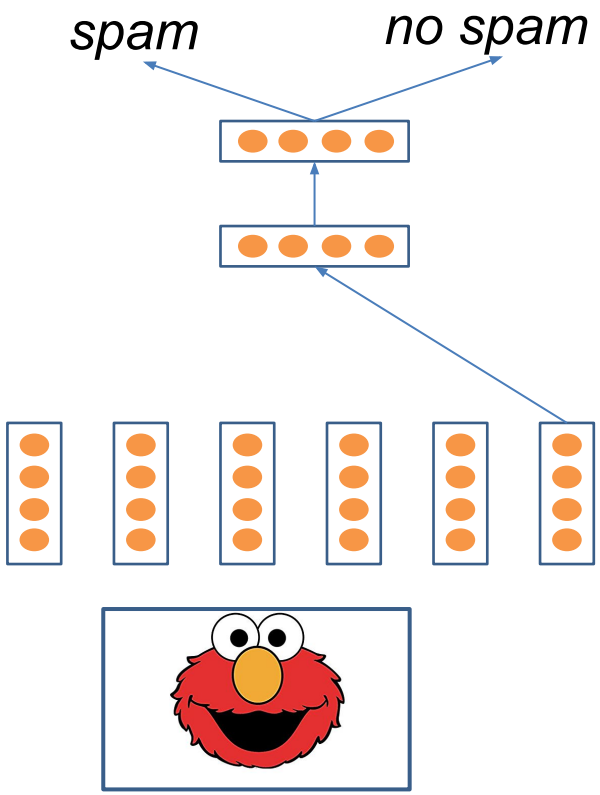
Back to LMs…

What to do with such representations?

classification

Hidden layer

Only use the last state

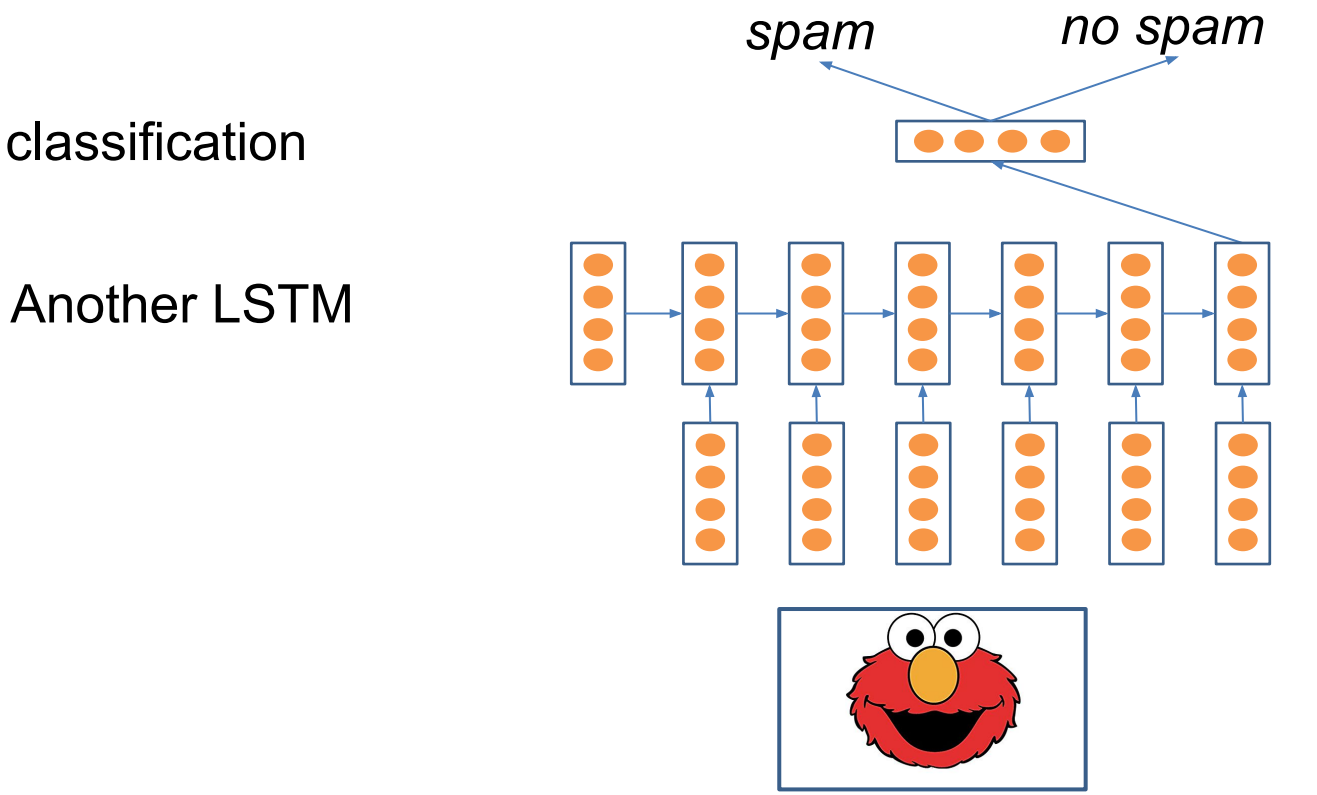


Nice flowers grow on the bank

Back to LMs…

What to do with such representations?

Use them as features/Embeddings!



Nice flowers grow on the bank

Interim Summary

- Language models: Probability distributions over words in a language
- Neural language models produce great continuous word representations that can be used as features/embeddings
- Word2Vec & distributional methods don’t look at context for word representations
- RNNs and ELMo-based language models get us better representations that depend on context

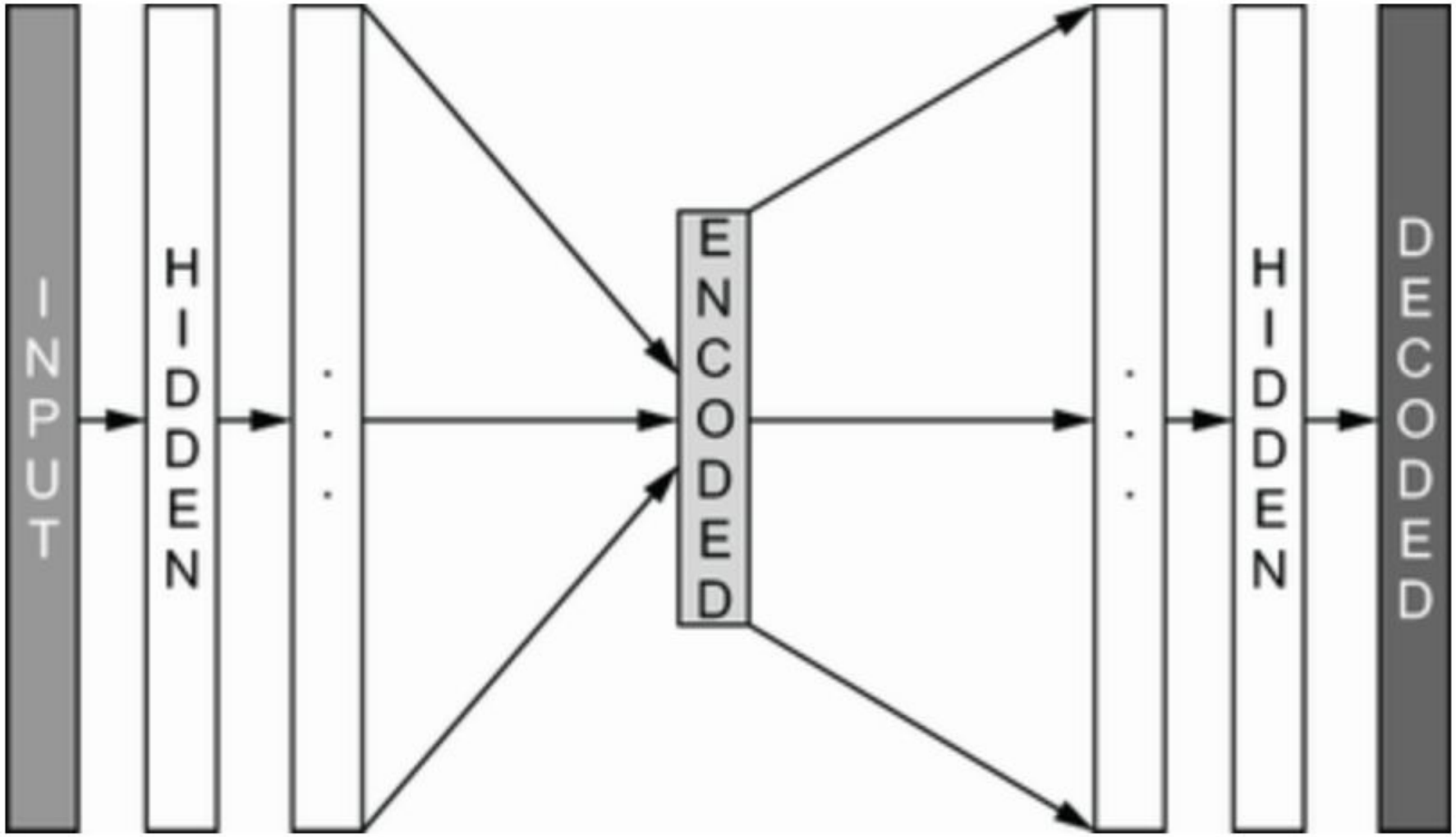
Sequence to Sequence Models

- For Translation: Input and Output are Sequences of Words
  - Sequence to Sequence Models
  - Need to use two RNNs in an encoder decoder model
    - **Encoder model purpose:** generate a representation for the input sequence in the last hidden state
    - **Decoder model purpose:** generate tokens conditioned on the encoder's representation

ANY TASK CAN BE FORMULATED as SEQ2SEQ

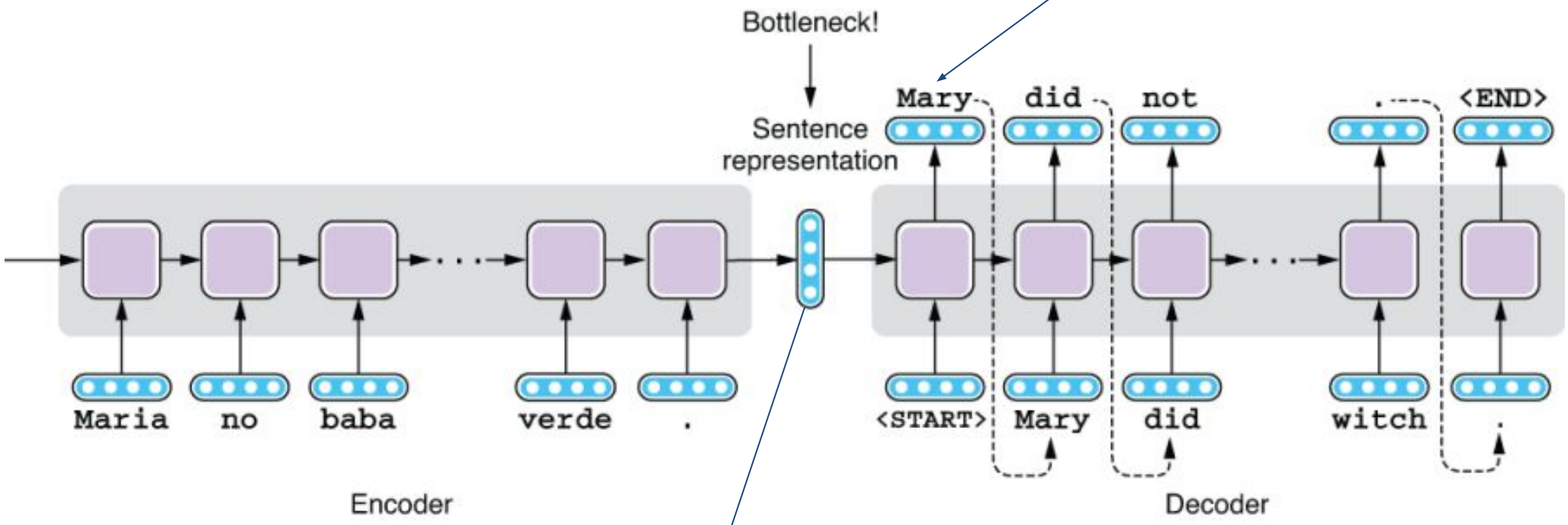
- Think Turing Machine

Architecture



Sequence to Sequence Models

Start Decoding using the encoder's final representation.



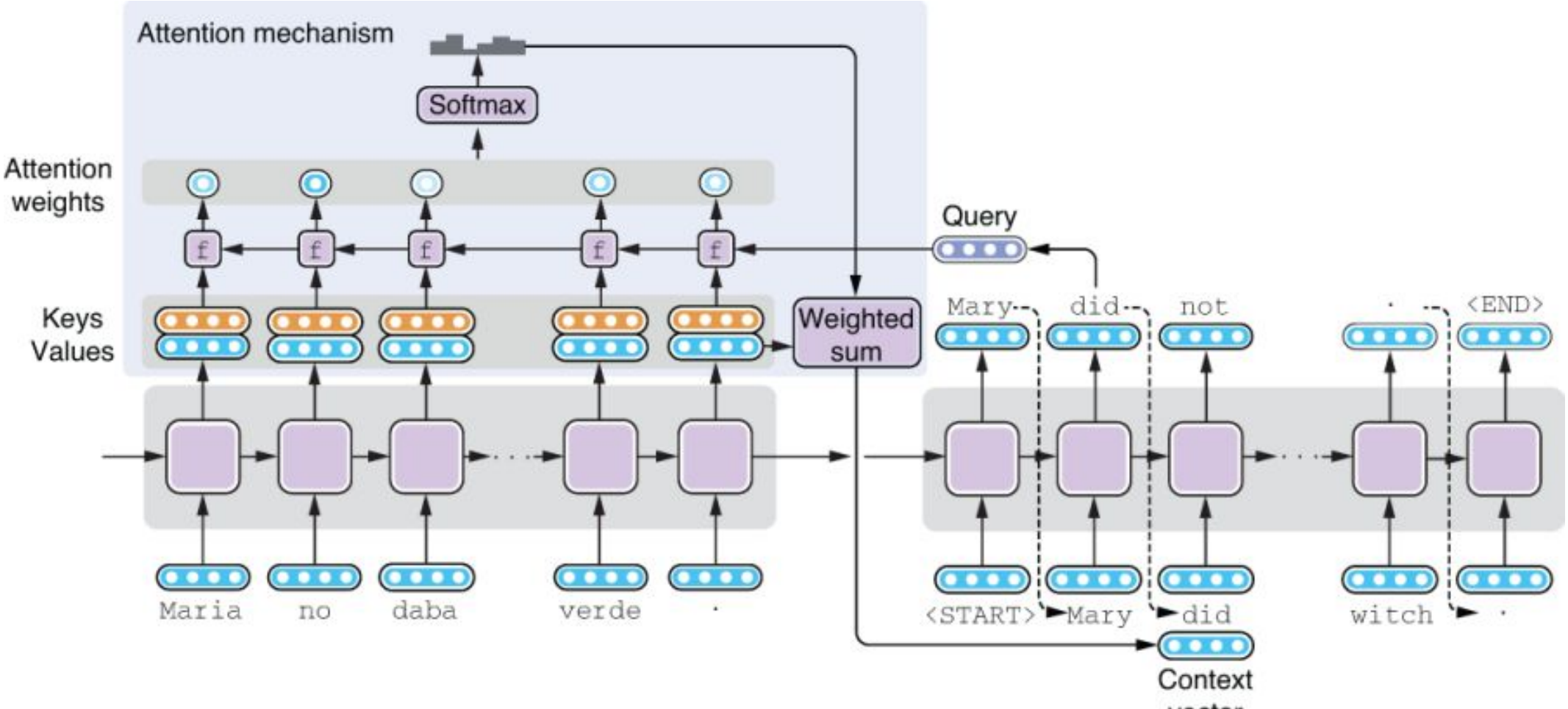
Final encoder hidden state (vector to represent the input sequence)

Decoder is fed its own previous output. It's a language model

Any problem with this architecture?

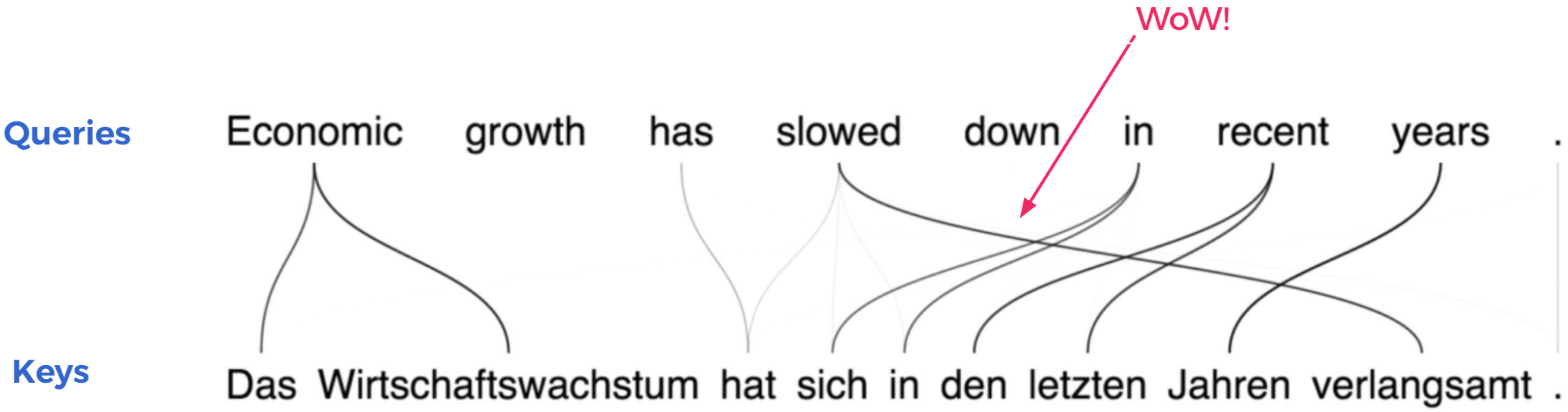
Attention

Solution: provide a connection between decoder and all encoder hidden states. This way it can pay attention to the encoder tokens it needs at the present moment



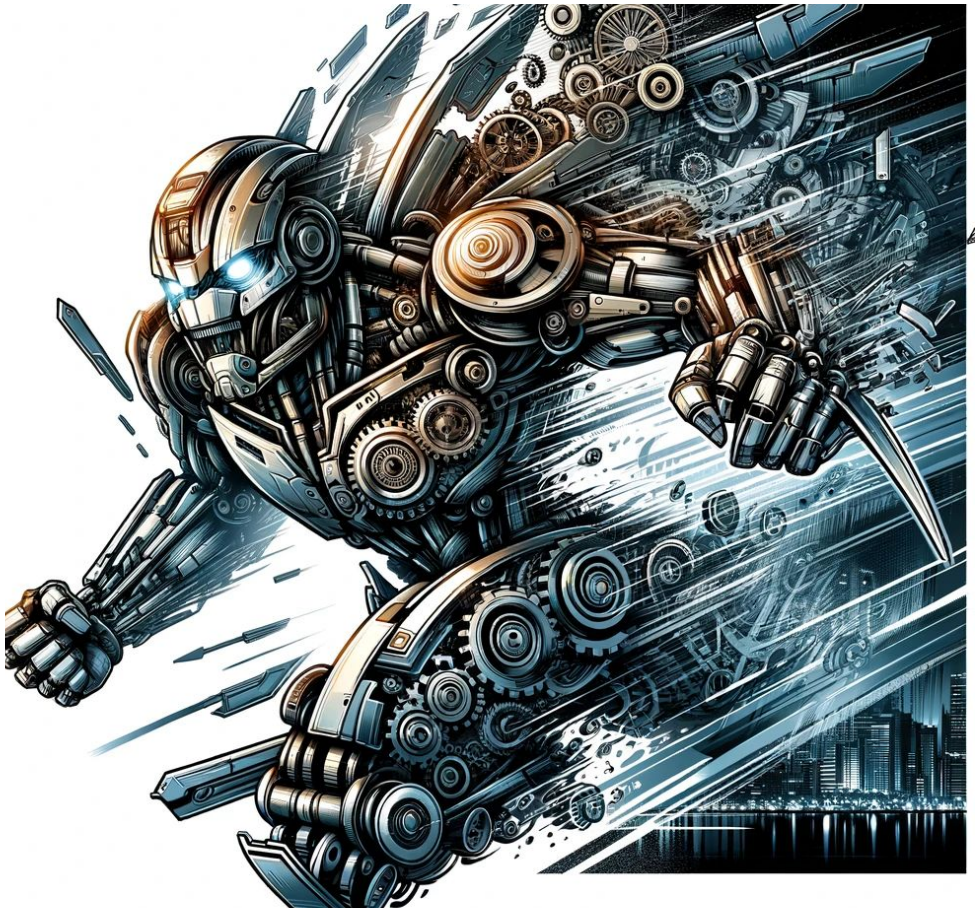
Attention Advantages

- Notice that attention is a human-like process
  - When you translate a sentence, you don't memorize it at once then translate. You can look at it while thinking!
- Big performance boost compared to the previous architecture
- It is sometimes interpretable

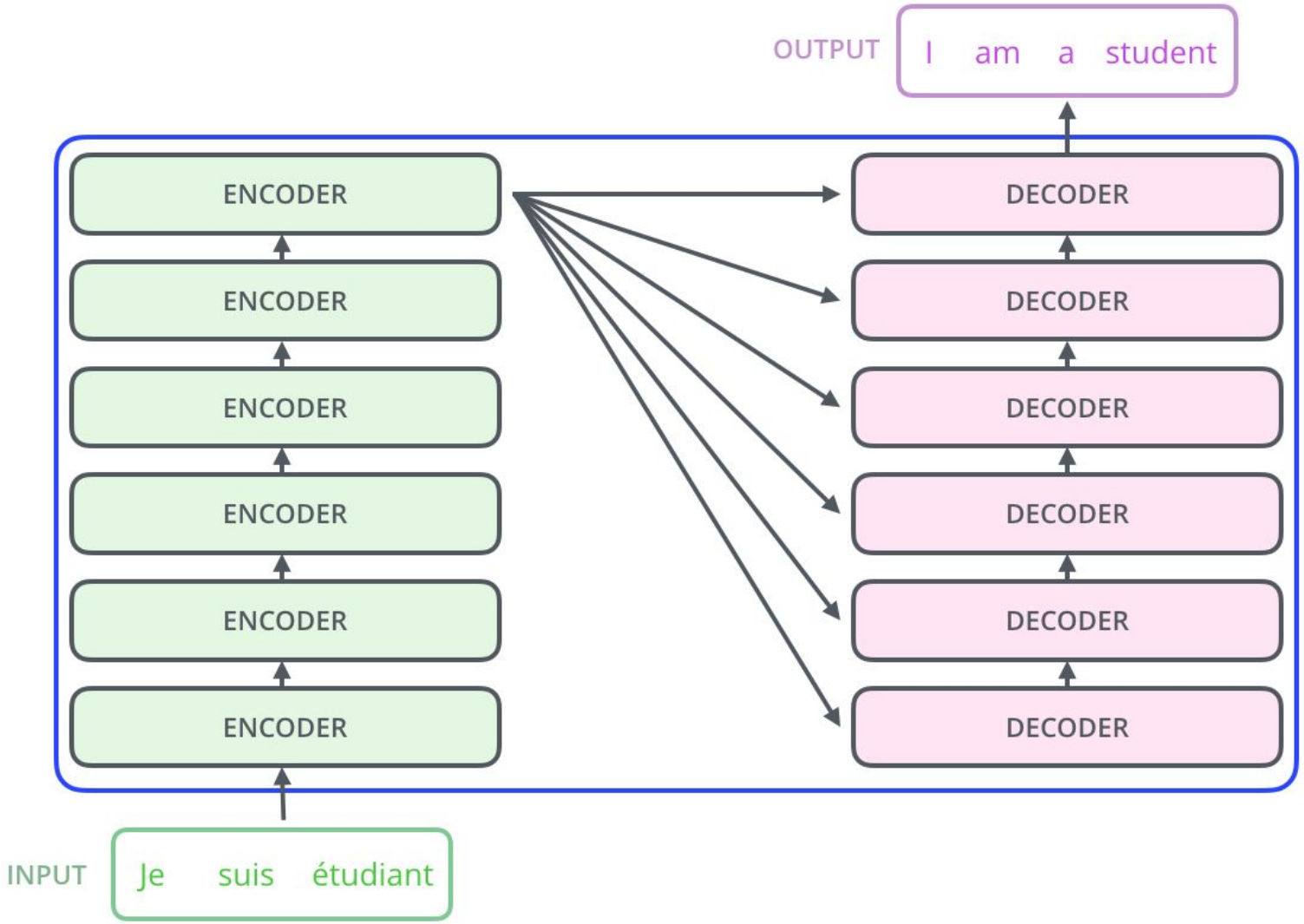


Darker color means higher A[i,j]

Transformer Architecture

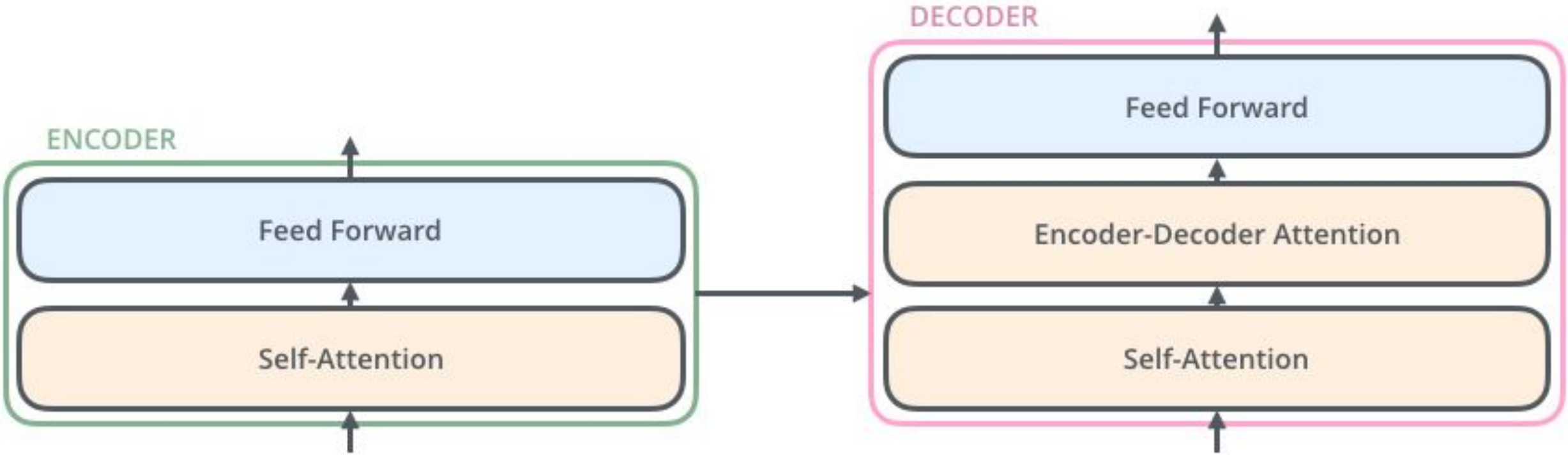


The Illustrated Transformer



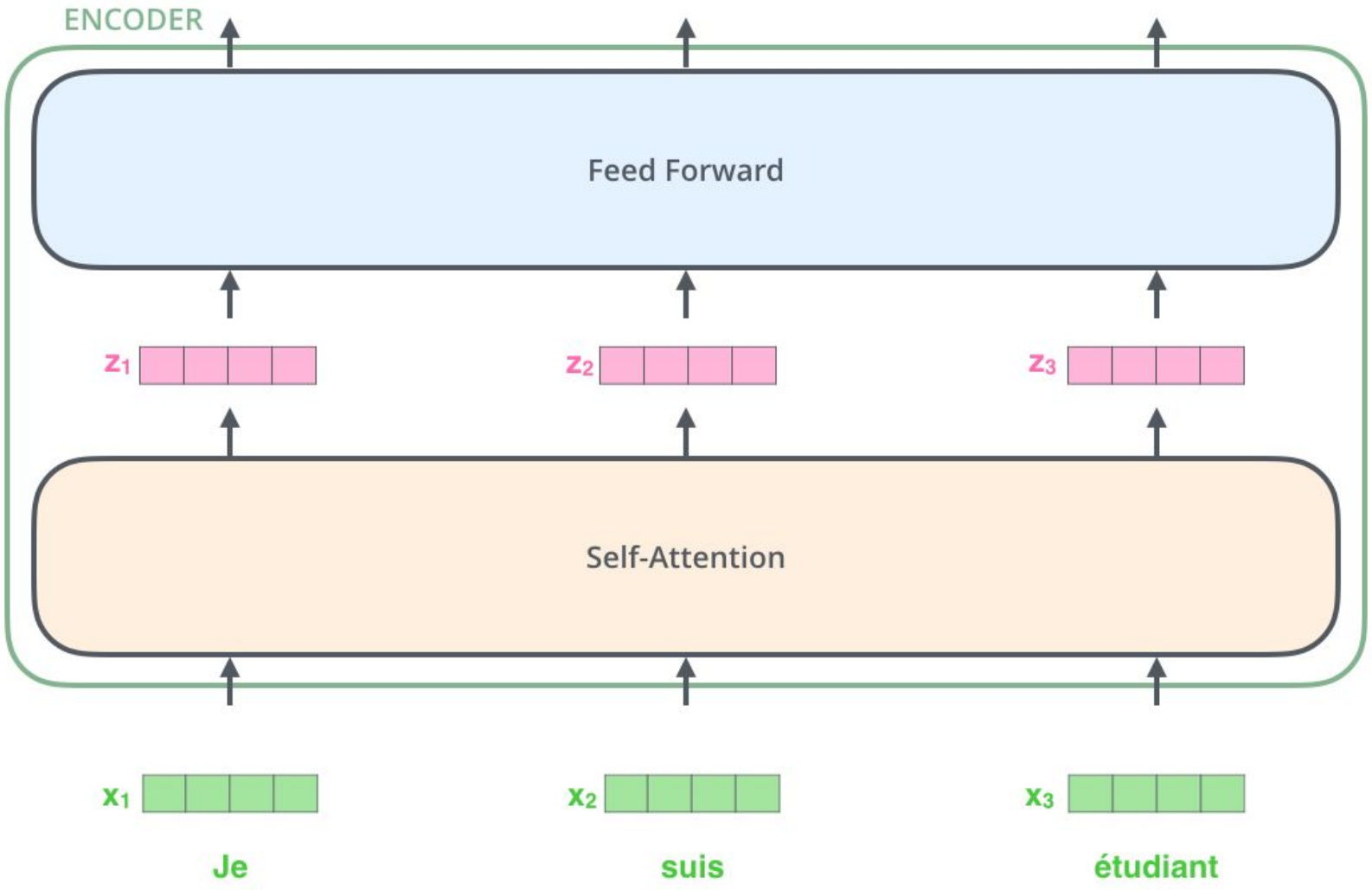
http://jalammar.github.io/illustrated-transformer/

The Illustrated Transformer



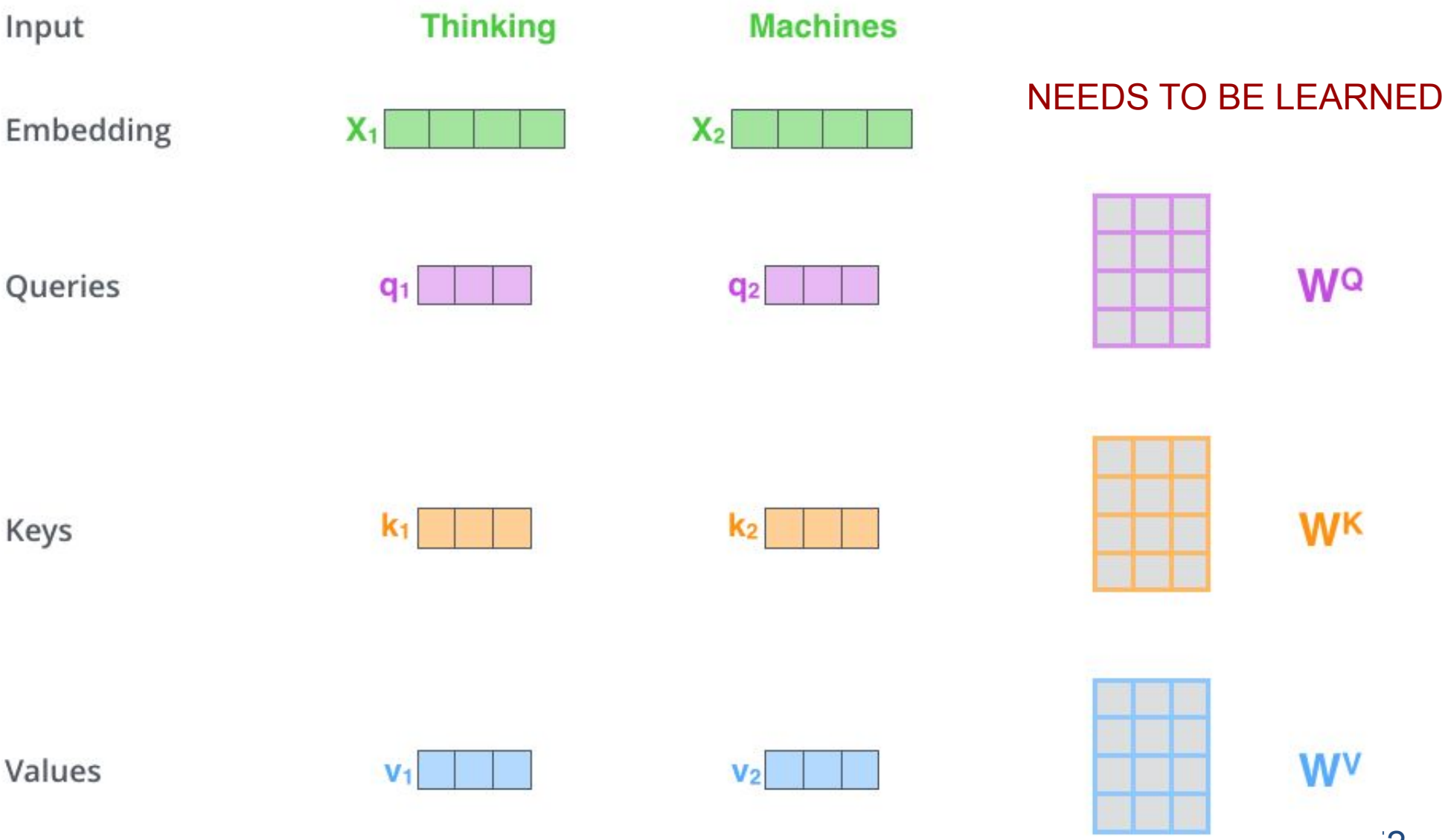
http://jalammar.github.io/illustrated-transformer/

Self - Attention



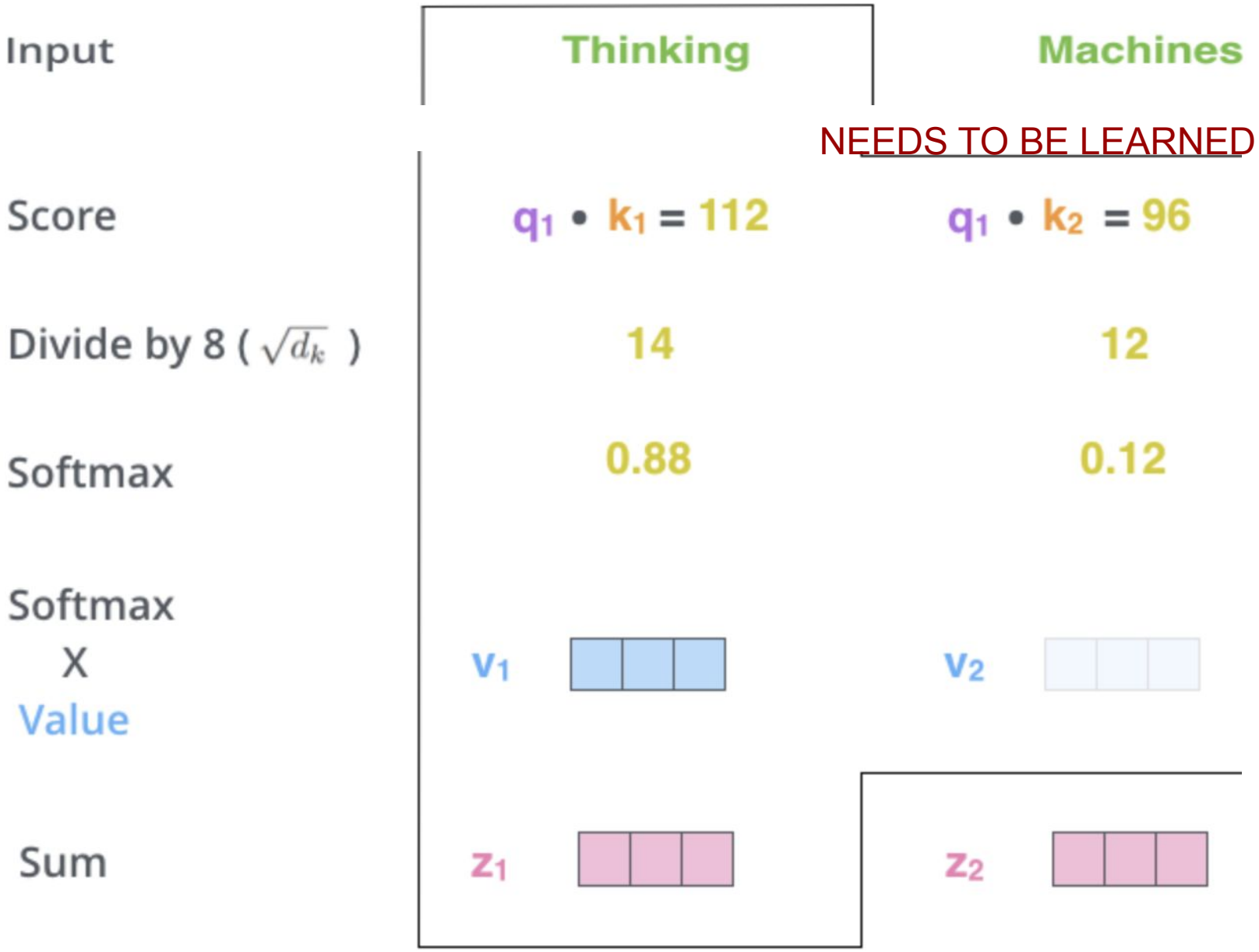
http://jalammar.github.io/illustrated-transformer/

Self - Attention



http://jalammar.github.ic
ransformer/

Self - Attention



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

Multi- Head Attention!

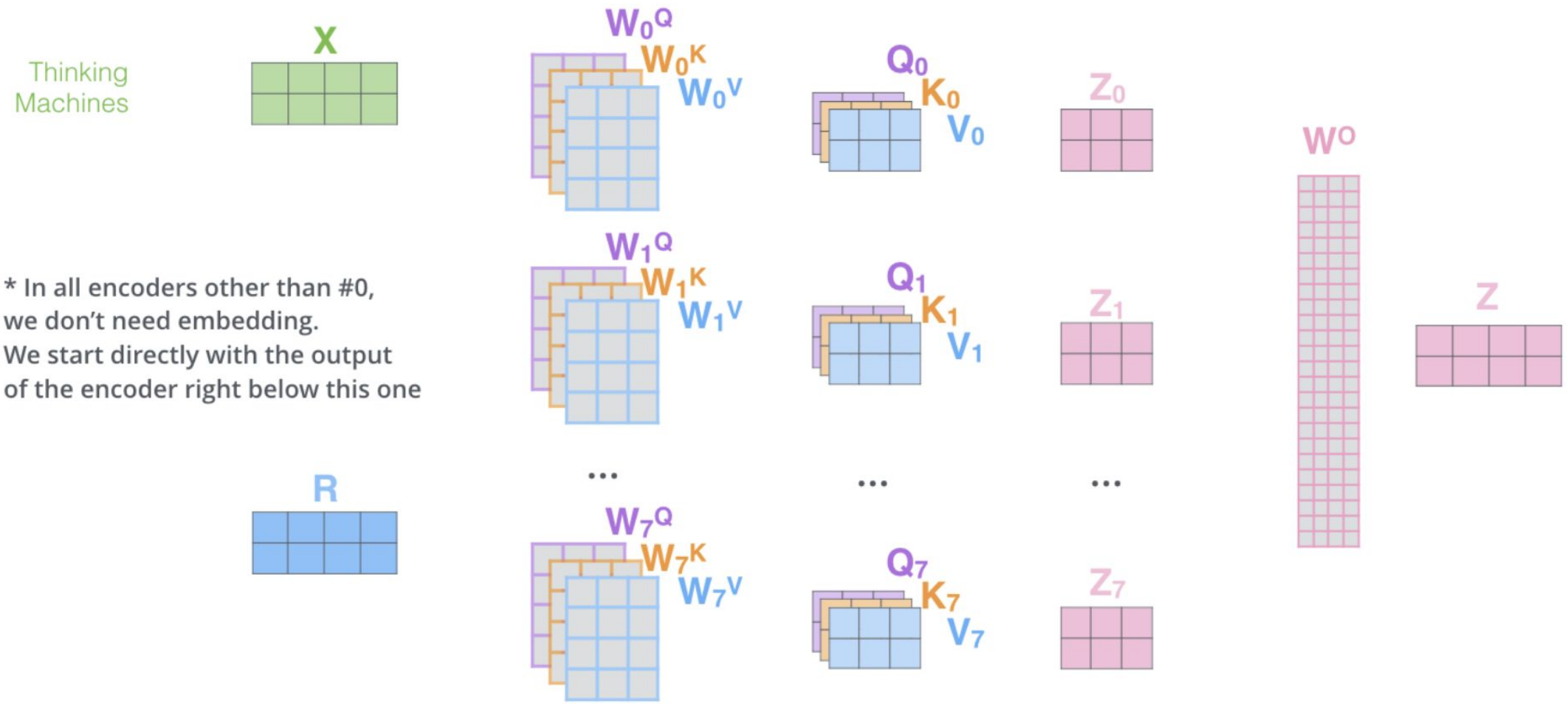
1) This is our input sentence*

2) We embed each word*

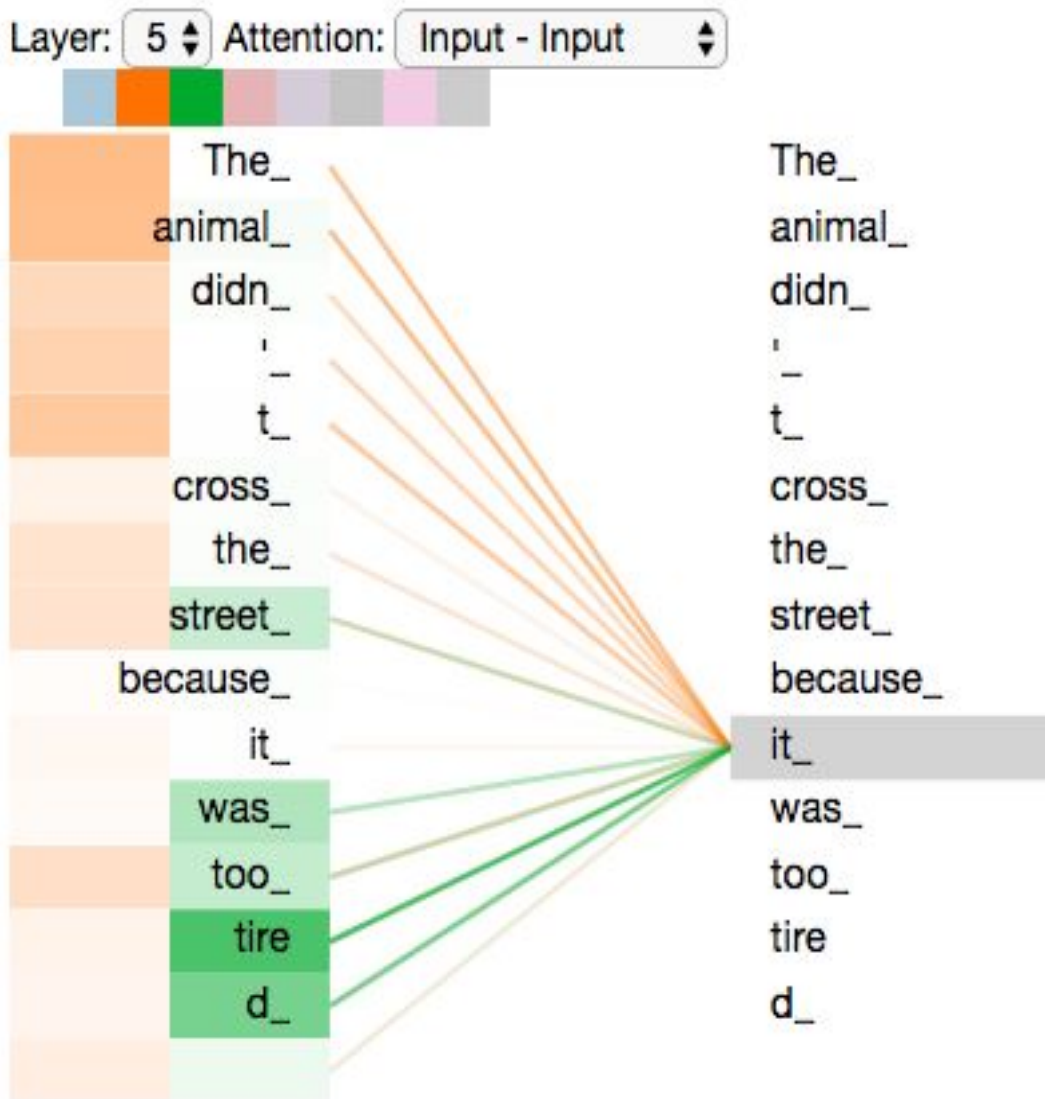
3) Split into 8 heads. We multiply X or R with weight matrices

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



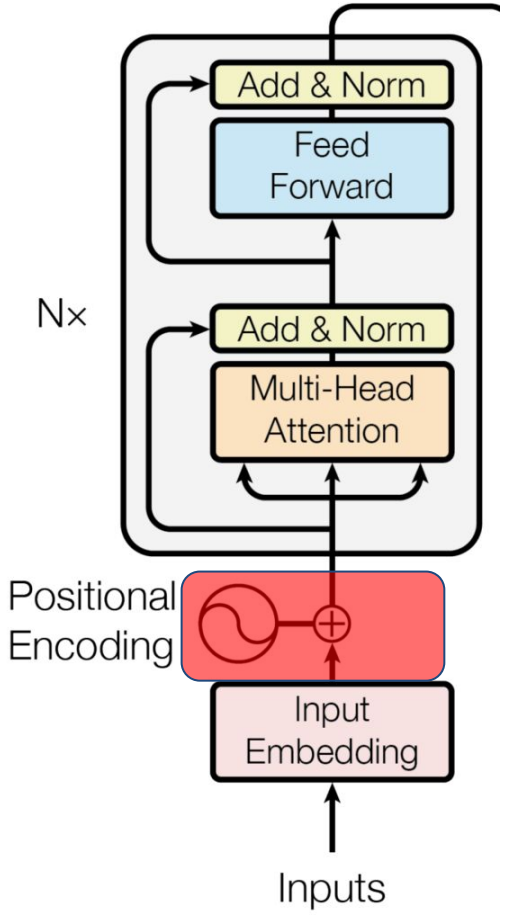
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one


Layer: 5
Attention: Input - Input
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_

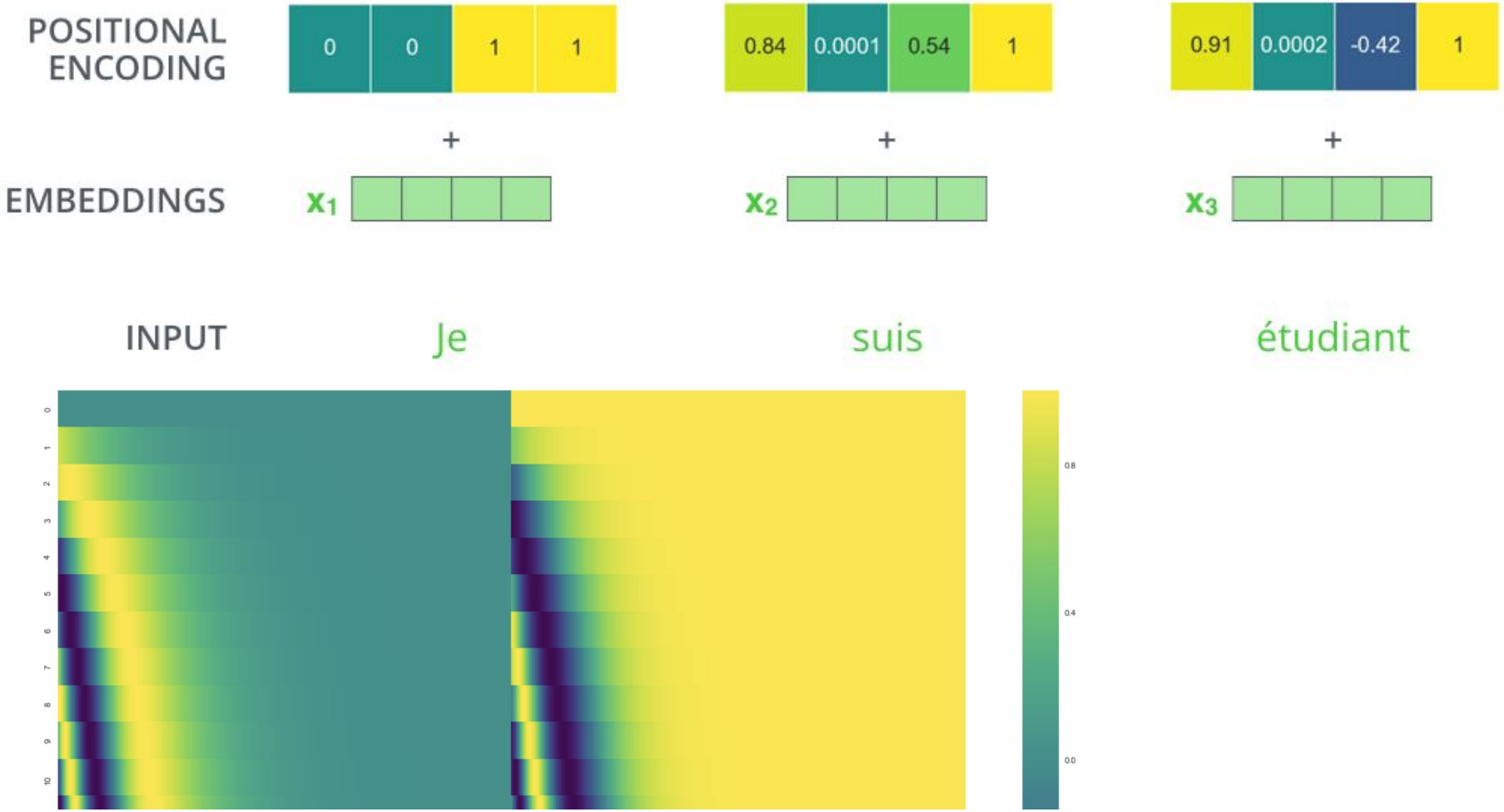
Positional Encodings

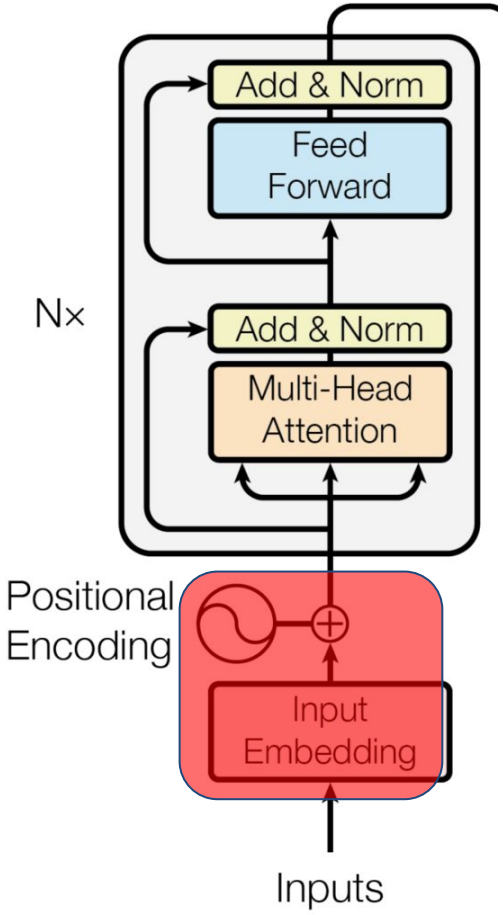
- **Problem I:**
  - Multi-head attention yields the same set of results regardless of the order of the tokens

- **Solution I:**
  - Allow attention to make use of positional information by injecting it beforehand via Positional Encoding
  - Positional Encoding adds a position dependent signal (vector) to each token in the sequence
    - Can allow model to learn it or set it explicitly



Embeddings Positional Encodings

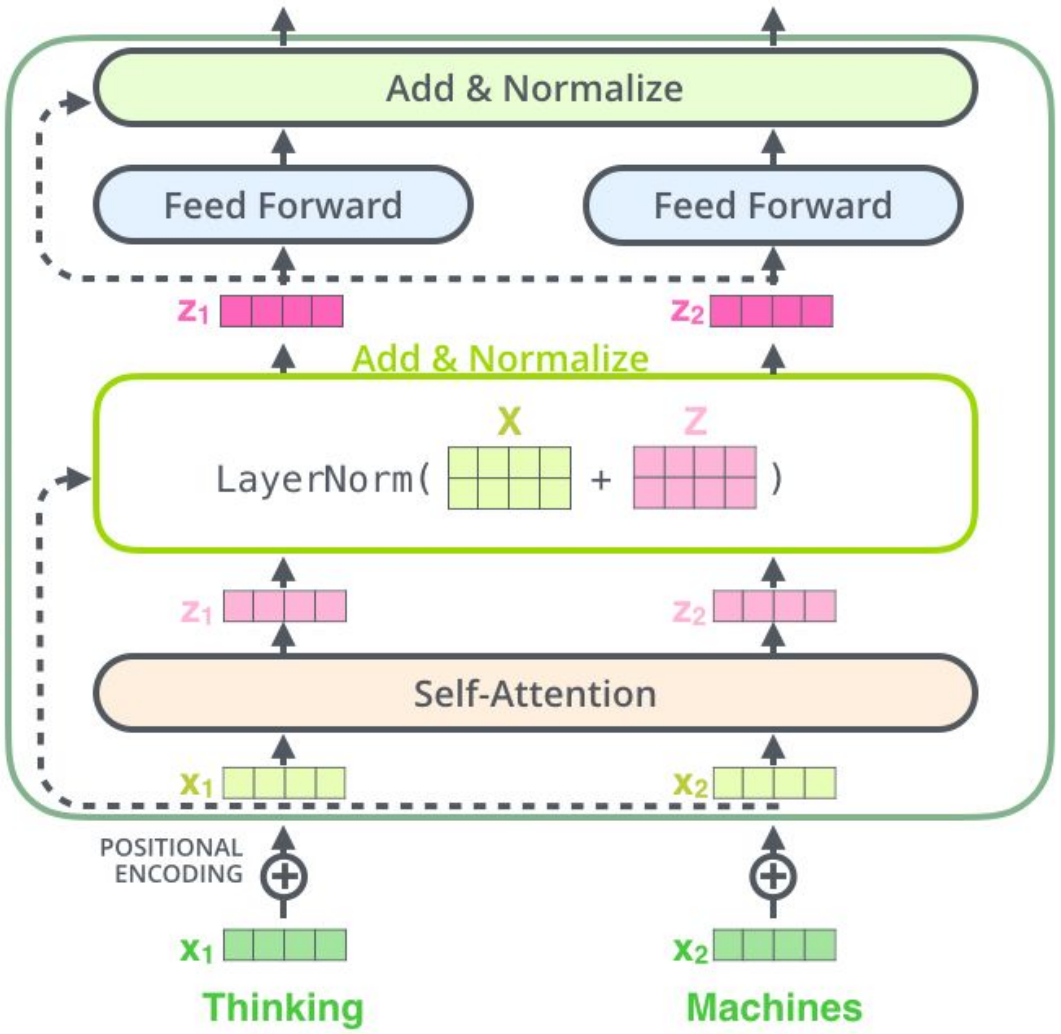




Source: Deep Learning for NLP http://jalammar.github.io/illustrated-transformer/

Residuals and Normalizations

- **Problem II:**
  - vanishing gradients and training can be slow. Transformer uses a sequence of N encoder layer.



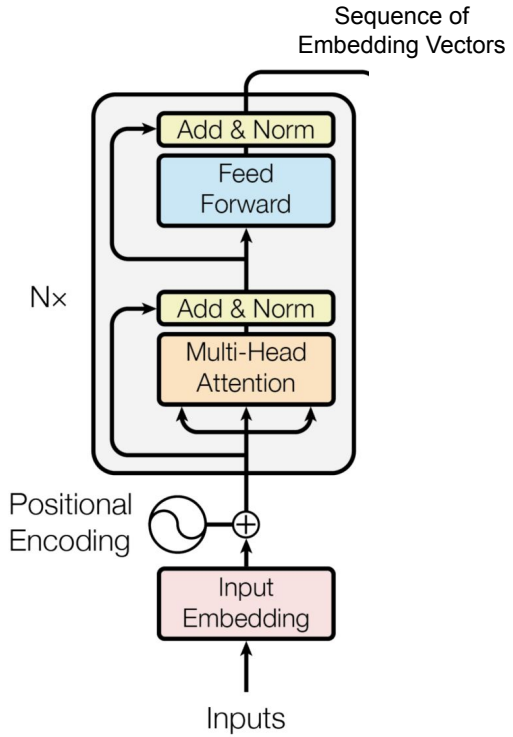
- 1. **Calculate the mean (μ)** of all the neuron activations within that single layer.
  2. **Calculate the variance (σ^2)** of those same activations.
  3. **Normalize the activations** by subtracting the mean and dividing by the standard deviation (plus a small constant ϵ for numerical stability).
  4. **Apply learnable scaling (γ) and shifting (β) parameters** to the normalized values, allowing the network to adjust the optimal mean and variance for each layer during training.

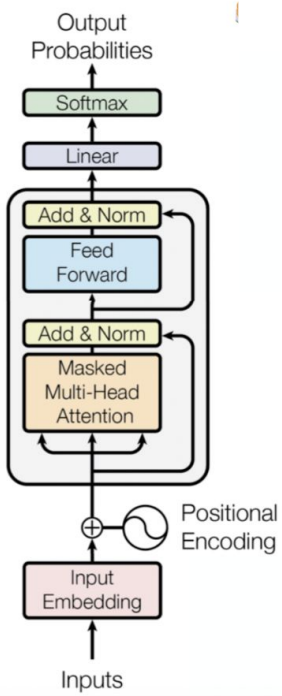
The formula for the final output (y_i) is:

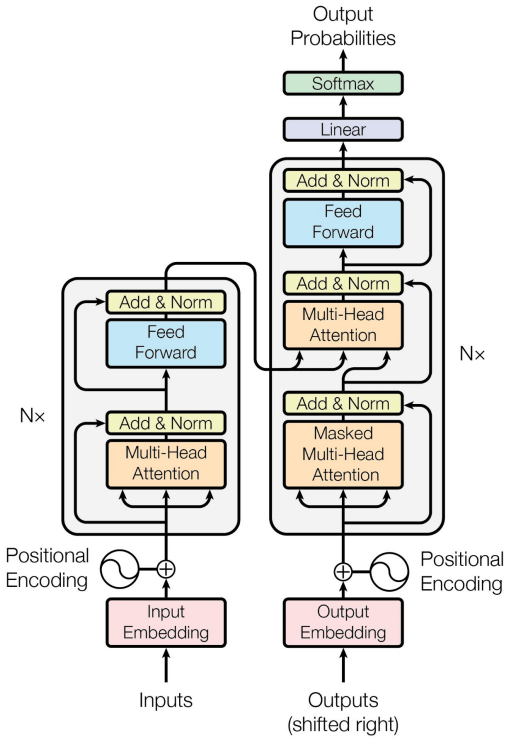
$$y_i = \gamma \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

Summary: Transformer Models

- Three types of transformer models:







Encoder-Only Models

Mostly classification tasks
(e.g., Named-entity Recognition)

Decoder-Only Models

Text Generation Tasks
(e.g., Language Modelling)

Encoder-Decoder Models

Sequence-to-sequence tasks
(e.g., translation)

It Works Superbly!

Andrej Karpathy
@karpathy

The Transformer is a magnificient neural network architecture because it is a general-purpose differentiable computer. It is simultaneously:
1) expressive (in the forward pass)
2) optimizable (via backpropagation+gradient descent)
3) efficient (high parallelism compute graph)

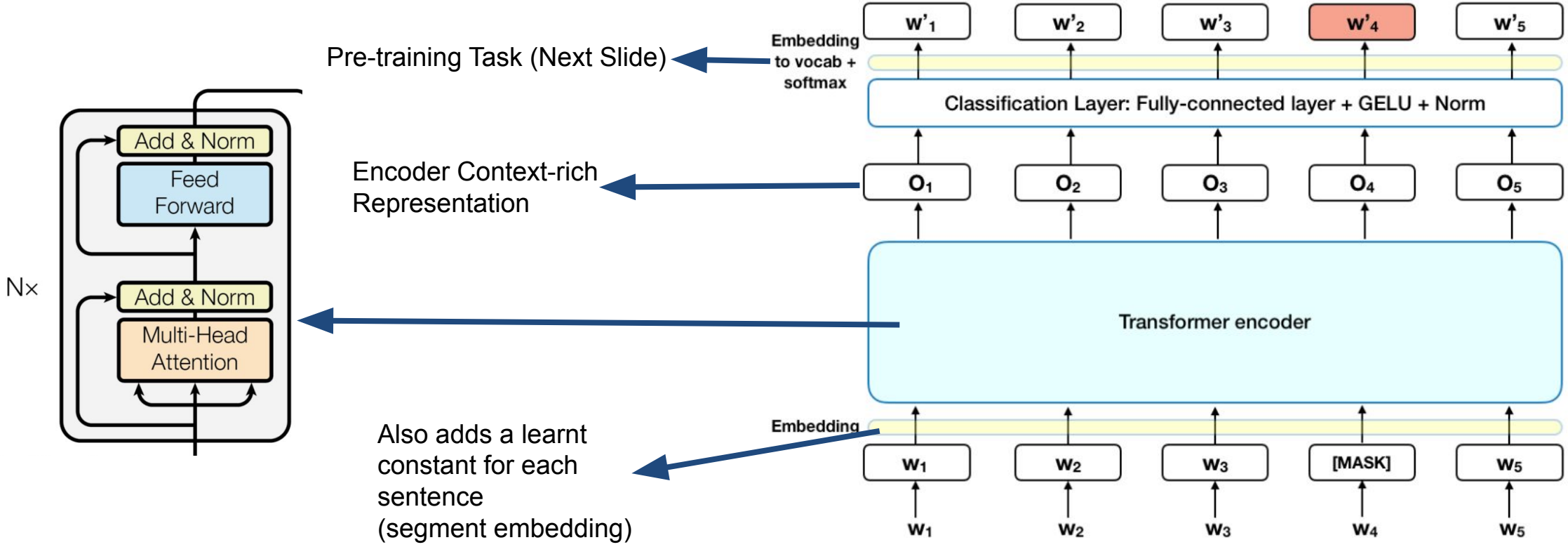
11:54 AM · Oct 19, 2022

490 Retweets **39** Quotes **3,670** Likes **1,023** Bookmarks

BERT

- **Stands for** “Bidirectional Encoder Representation for Transformers”
- A language representation model introduction by Google in 2018
- It’s pre-trained on Toronto Book Corpus and English Wikipedia (3.3B words)
- Consequently, BERT can be fine-tuned to accomplish many NLP tasks given a much smaller set of data

BERT: Architecture

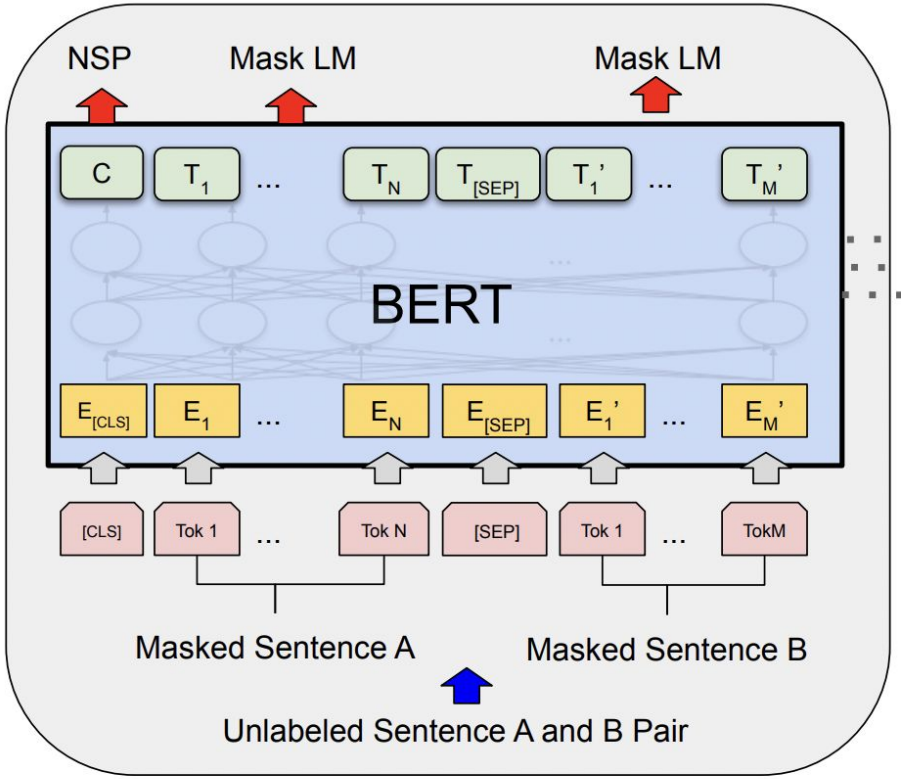


BERT Base (110M Param.)	BERT Large (340M Param.)
12 Encoders (N=12) each with 12 attention heads	24 Encoders (N=24) each with 16 attention heads
Dimensionality of encoder representation = 512	Dimensionality of encoder representation = 768

BERT: Pretraining

- BERT was simultaneously trained to accomplish **two tasks:**
  1. Next Sentence Prediction
  2. Masked Language Modelling

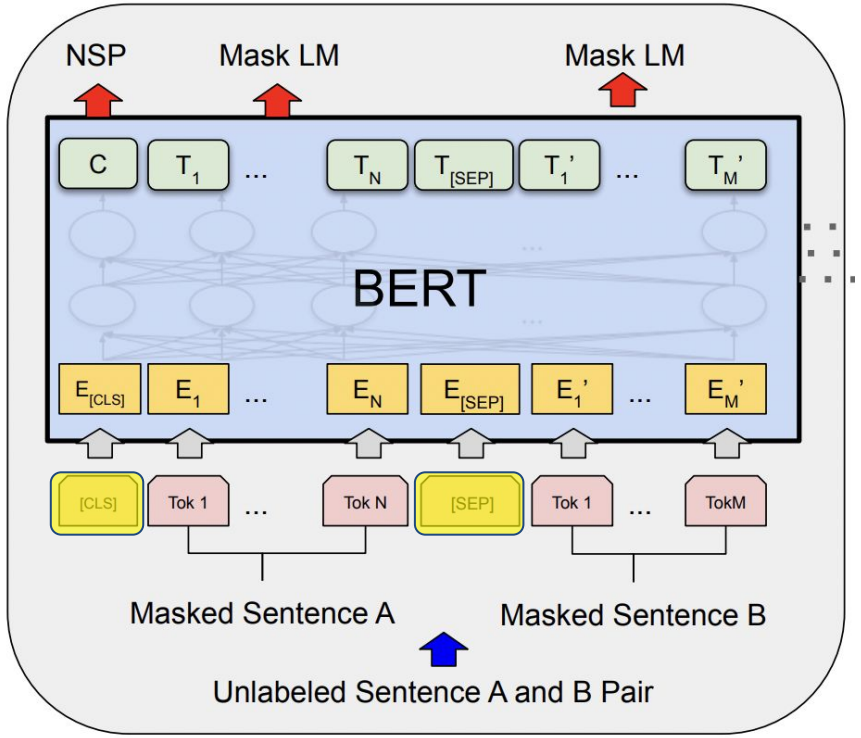
Masked Language Modelling	Next Sentence Prediction
Learn to assign rich contextual embeddings to any English token	Learn to relate two given sentences (or paragraphs)
Makes fine-tuning for any token classification task much easier	Makes fine-tuning for tasks such as question-answering easier or sentence classification



Pre-training

Pretraining: Next-Sentence Prediction

- **Pretraining Task I**: Predict if S2 follows S1 in the original text
- **Training Setup:**
  - Divide the sentences in the dataset into two sets (each with 50% of the sentences)
  - For the first, associate each sentence to the real next sentence
  - For the second, associate each sentence to a random next sentence
- Let the two sentences be S1 and S2, then a training example is now in the form
  [CLS] S1 [SEP] S2 [SEP]
  - [CLS] is a binary label to denote if S1 and S2 are pairs
  - Purpose of [SEP] is to separate the two sentences.

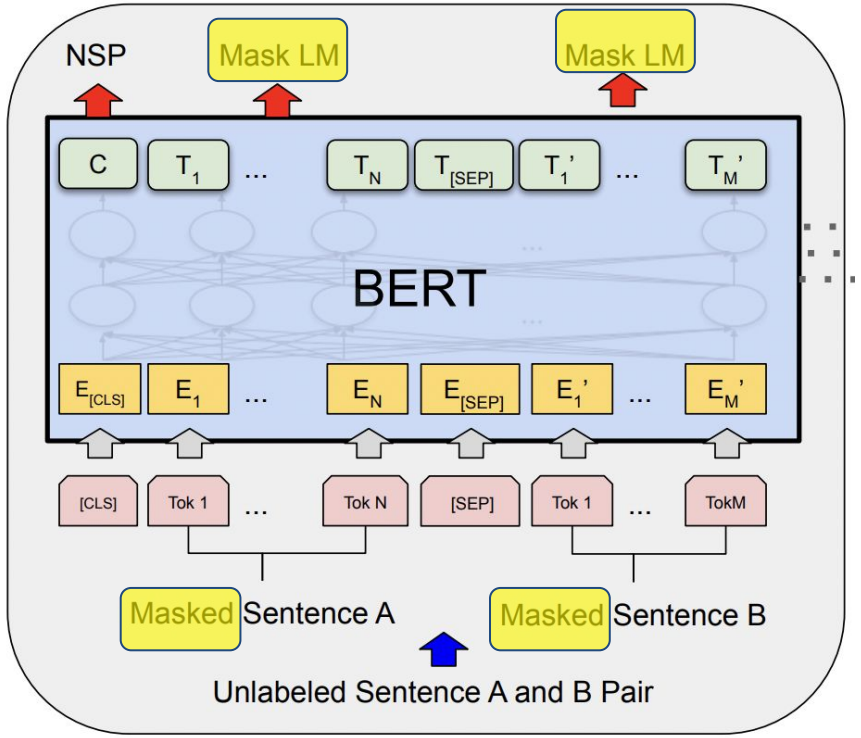


Pre-training

Pretraining: Masked Language Model

- **Training Setup:**
  - Randomly select 15% of all tokens in the dataset
    1. Replace 80% of them with [MASK] token
    2. Leave 10% as is
    3. Replace 10% with a random word

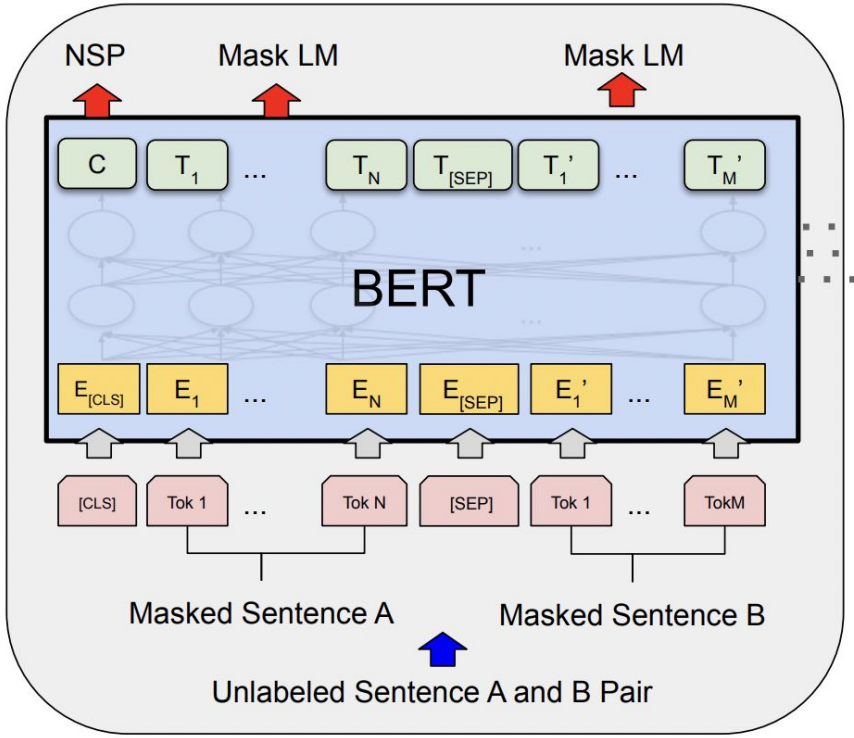
- **Pre-training Task II**:
  - Model is trained to predict the original word corresponding to any selected token (linear layer + Softmax)
  - Reason for (2), (3) is so that it does not overfit by only assigning good vectors to [MASK]



Pre-training

BERT Pretraining: Insights

- For BERT-Base, a batch size of around 256 is used, while for BERT-Large, it's around 512.
- BERT's training optimizes a combined loss function for both MLM and NSP tasks.
- The model processes the batch of sequences, makes predictions, calculates the loss, and then updates its parameters (weights) using backpropagation.
- The Adam optimizer is used to adjust the weights

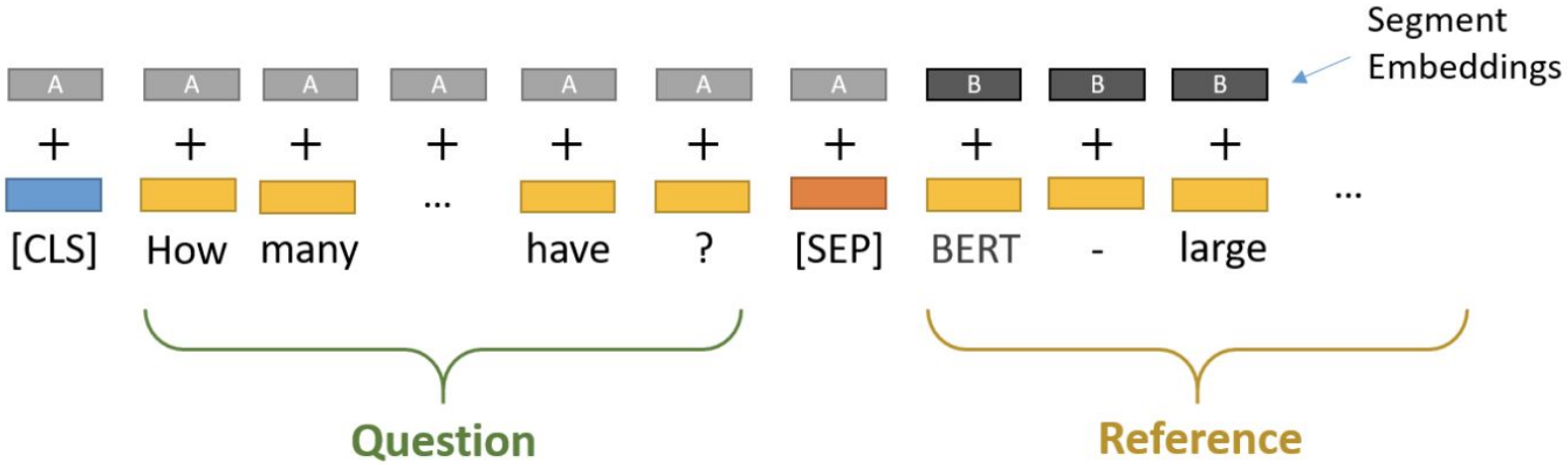


Pre-training

For more details, see the original BERT paper: https://arxiv.org/abs/1810.04805

BERT Fine-tuning for QA

- The purpose of pretraining the model is that now it can be trained on downstream tasks with much less data (fine-tuning)
- **Fine-tuning for Question-Answering**
  - Put each example in the dataset in the form: [CLS] Question [SEP] Context [SEP]
  - Dot-product the encoder representation with a “start weight vector” and “end weight vector”
    - Pass the results to Softmax to get a probability distribution of start and end
  - Answer to question is spanned by highest probability start and end positions in the context

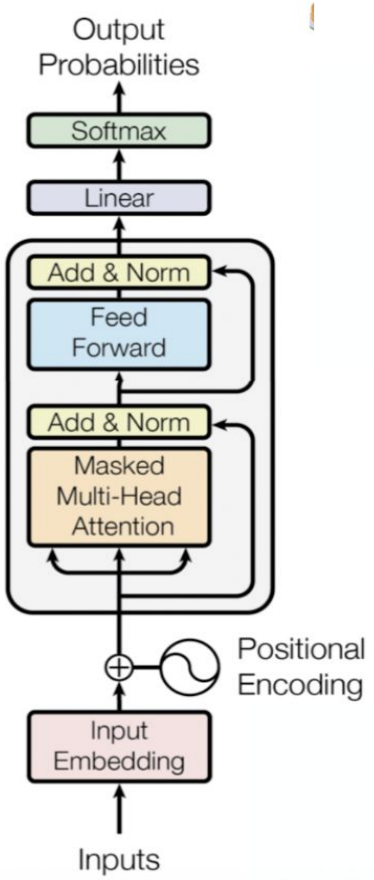


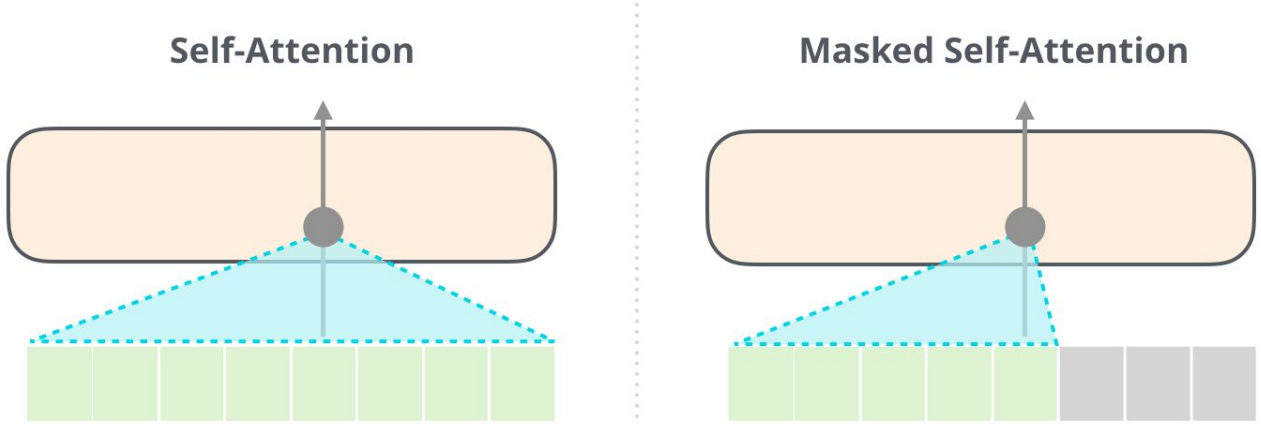
Improvements Over BERT

- **RoBERTa:**
  - Trained on 10 times as much data
  - Ditches NSP task
  - Uses dynamic masking (changes it each topic)
  - 2-20% improvement over BERT
- **DistilBERT**
  - Reduce size of BERT by 40% via knowledge distillation (approximating BERT)
  - Retains 97% of performance
- **ALBERT**
  - 70% of BERT parameters at better performance
  - Mathematical optimizations for the parameters
  - Replaces Next Sentence Prediction with Sequence Order Prediction

Transformer Decoder

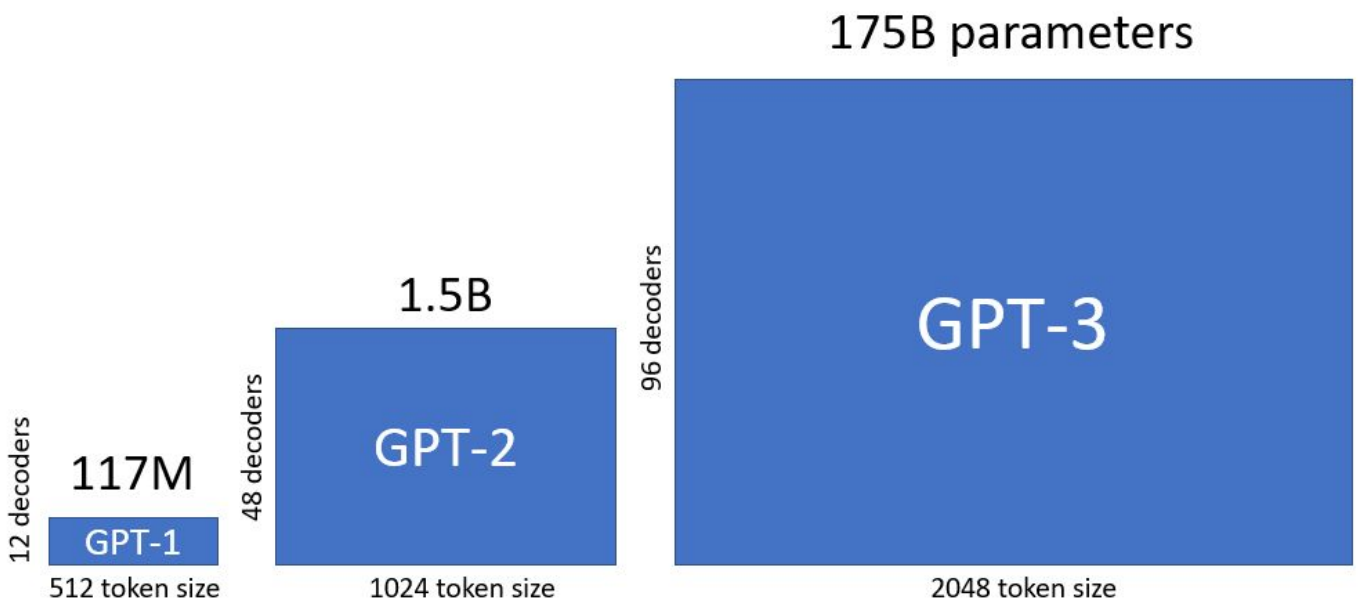
- **Almost same as the encoder**
  - Used to generate language (usually called auto-regressive)
    - Trained to predict the next word
    - Softmax at output assigns probability of being next toker to each token in vocabulary
  - One difference compared to encoder: masked self-attention
    - It's self-attention where no token can attend to another token that occurs after it in the sequence

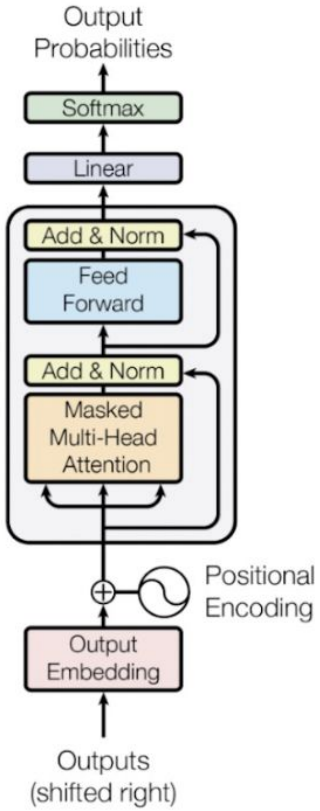




GPT

- **Generative Pre-trained Transformer (GPT)**
  - Decoder-only Model released by Open-AI
  - Released multiple model variants:





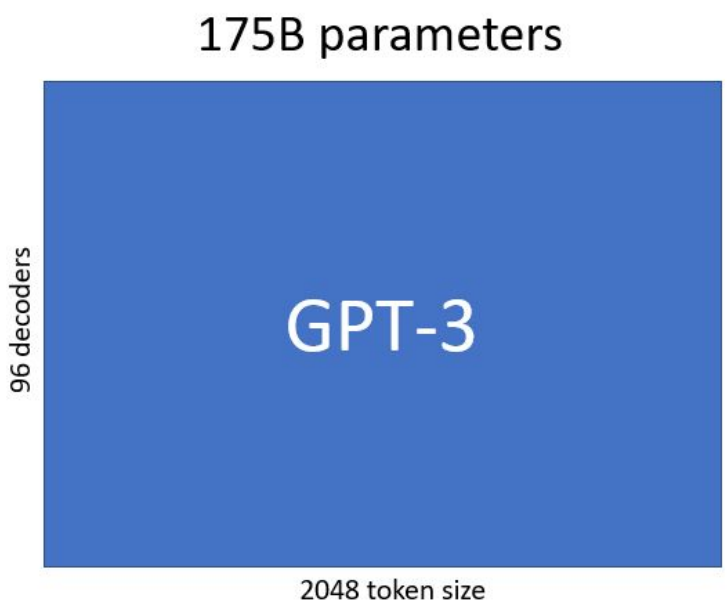
GPT: Training

- Training data is divided into sequences of a fixed length. For GPT-3, the maximum sequence length was 2,048 tokens.

- Batch Size: The batch size can vary. In large models like GPT-3, it's usually quite large, necessitating powerful computing resources.

- At each training step, the model generates a probability distribution for the next word, and the cross-entropy loss is computed between this distribution and the actual next word in the sequence.

- Adam is typically used as an optimizer



GPT Family

Generative Pre-trained Transformer (GPT)

GPT-3.5 (Jan 2022)

- Fine-tuned GPT-3 with RLHF (Reinforcement Learning).
- Aligned output with HHH (Helpful, Honest, Harmless).
- Enabled complex, zero-shot instruction-following.

ChatGPT (Nov 2022)

- Provided a user-friendly GUI on top of GPT-3.5.
- Safety and security layers added for public use.
- Brought generative AI to the mainstream.

GPT-4 (Mar 2023)

- A major leap in reasoning, accuracy, and quality.
- **Multimodal:** Could accept image inputs, not just text.
- Powered the ChatGPT Plus subscription.

GPT Family

GPT-4 Turbo (Nov 2023)

- "Upgraded" GPT-4 with a vastly larger **128k context window**.
- Faster, cheaper, and better at following instructions.

GPT-4o (May 2024)

- "**o**" for "Omni": Natively handles **text, audio, and image** I/O.
- GPT-4-level intelligence at much higher speeds.
- Rolled out to free users, democratizing access to top-tier AI.

GPT-4.1 (Apr 2025)

- Specialized model with a massive **1 million token context window**.
- Designed for in-depth analysis of long documents or codebases.

GPT-5 (Aug 2025)

- The new flagship, state-of-the-art model.
- Introduced flexible **"reasoning modes"** (Instant, Thinking, Pro) to control compute for complex tasks.

Another Popular Decoder: LLaMA

Llama 1 (February 2023)

- **What It Was:** A family of foundation models (7B-65B) released by Meta for **researchers**.
- **Key Tech:** Used now-standard features like RMSNorm, SwiGLU, and RoPE.
- **The Big Bang:** The model's weights were **leaked in March 2023**, igniting a global open-source AI movement as developers finally had a powerful, non-proprietary model to build upon.

Llama 2 (July 2023)

- **What It Was:** The **first official open-source release** for both research and **commercial use** (7B-70B models).
- **Key Upgrades:**
  - Trained on 40% more data than Llama 1.
  - Doubled the context length to 4,096 tokens.
  - **Llama 2-Chat:** The first version fine-tuned with Reinforcement Learning from Human Feedback (RLHF), making it a capable chatbot.
- **Impact:** Provided a powerful, free alternative to closed models, fueling a new ecosystem of AI startups and applications.

Another Popular Decoder: LLaMA

Llama 3 (April 2024)

- **What It Was:** A major leap in performance (8B & 70B models) that became the new **state-of-the-art** for open models.
- **Key Upgrades:**
  - **Massive Training Data:** Trained on a custom **15 Trillion token** dataset.
  - **Better Efficiency:** Used a new **128,000-token vocabulary** to process language more effectively.
  - Vastly improved reasoning, coding, and instruction-following.

The Rapid Evolution (2024-2025)

- **Llama 3.1:** Introduced a massive **405B parameter model** and a huge **128,000-token context window** (the size of a book).
- **Llama 3.2:** Added **multimodality**, giving the model the ability to understand images as well as text.
- **Llama 4:** The next generation, built to be **natively multimodal** and using a more efficient **Mixture-of-Experts (MoE)** architecture.

Overall Impact: The Llama family has fundamentally democratized access to high-performance AI, forcing the entire industry to adapt to the power of open-source development.

Transformer Encoder-Decoder

- Like RNN encoder-decoder models used for sequence-to-sequence tasks such as translation, summarization, dialogue, etc.
  - Decoder (language model) has an extra cross-attention block that allows it to attend over the encoder representation of input
  - Same reason why we used cross-attention in RNN encoder-decoder model

GEMMA

T5Gemma: A new collection of encoder-decoder Gemma models

JULY 9, 2025

Biao Zhang
Research Scientist

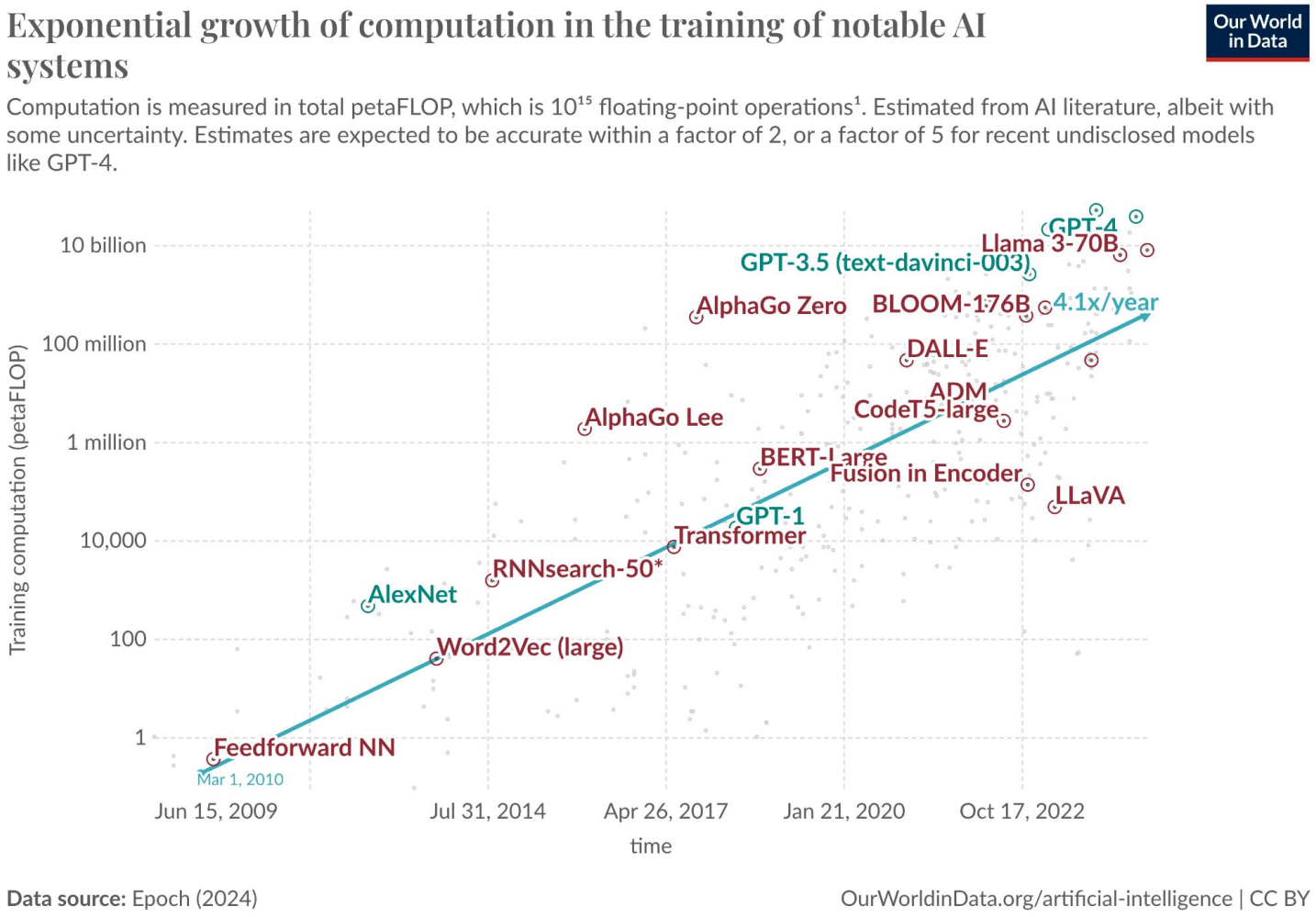
Paul Suganthan
Staff Software Engineer

Ben Hora
Product Manager

Share

Scaling Language Models

Computations Exponential growth



1. Floating-point operation: A floating-point operation (FLOP) is a type of computer operation. One FLOP represents a single arithmetic operation involving floating-point numbers, such as addition, subtraction, multiplication, or division.

https://ourworldindata.org/grapher/exponential-growth-of-computation-in-the-training-of-notable-ai-systems

What is “Scaling” mean?

- Increasing model size generally leads to better performance.
- Likewise, training on more data
  - expands the model's knowledge base
  - improving its ability to generalize effectively.

- Scaling involves several dimensions:
  - Increasing the number of parameters (Model Size)
    - But ensure that all parameters are fully utilized.

  - Adding more training data
    - But ensure that this data contributes new information rather than duplicating existing knowledge.

  - Increasing computational power
    - But ensure that this computation is efficiently used without waste.

- Ultimately, scaling requires a combination of all these factors: effectively compressing information through sufficient model capacity, optimized computation, and large, diverse datasets.

Let’s say you can use one – GPU for one day

Let’s say you can use one
GPU for one day

- Would you train a 5 million parameter LM on 100 books?
- What about a 500 million parameter LM on one book?
- Or a 100k parameter LM on 5k books?

Let's say you can use one - GPU for one day

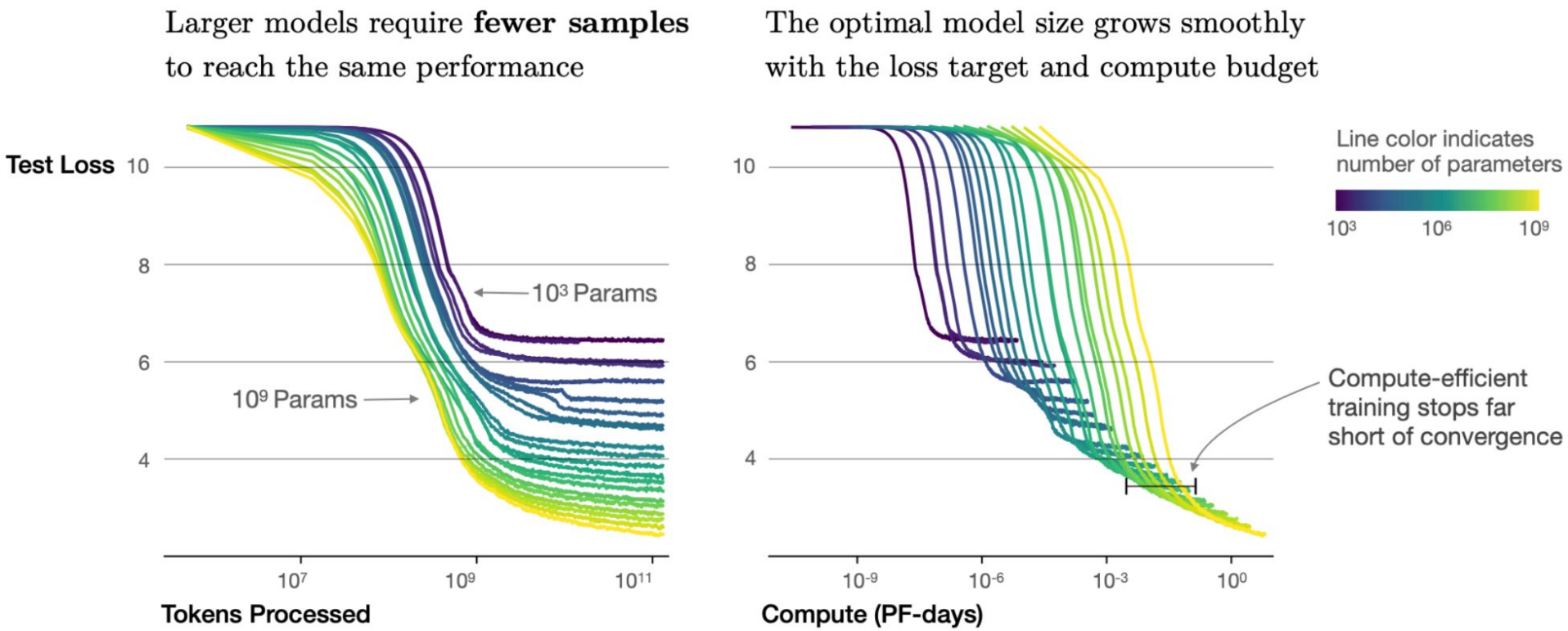


Figure 2 We show a series of language model training runs, with models ranging in size from 10^3 to 10^9 parameters (excluding embeddings).

Scaling power-law

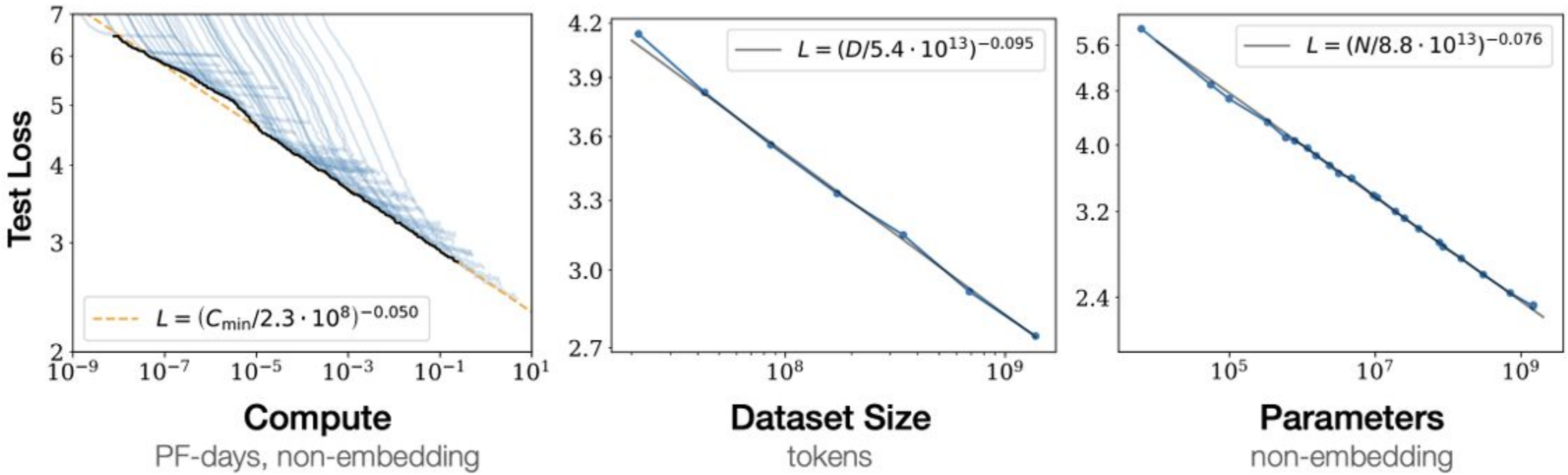


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Kaplan et al., 2020

Scaling power-law

- Suppose our compute budget is (C = 10) PetaFLOPs-days.
- The ideal model is one that fully utilizes the compute budget of (C).
- If we train a model larger than this optimal point, we will not achieve the best performance.
- If we train a model that is smaller, the performance will not be optimal.
- Optimal model size (N) and optimal data size (D), for a given compute budget (C):

Kaplan et. al. 2020	$N_{\text{opt}} \propto C^{0.73}$	$D_{\text{opt}} \propto C^{0.27}$

N_{opt} exponent >> D_{opt} exponent

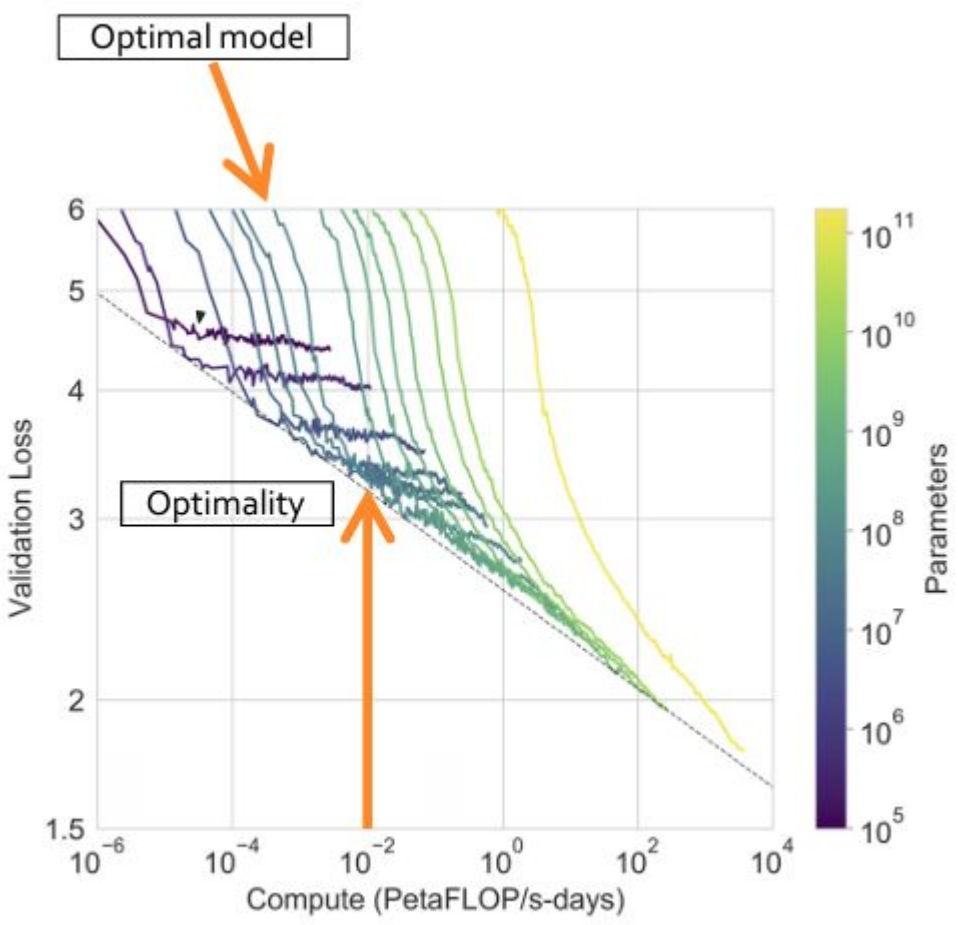


Photo credit: GPT3, Brown et. al., 2020

The Chinchilla law

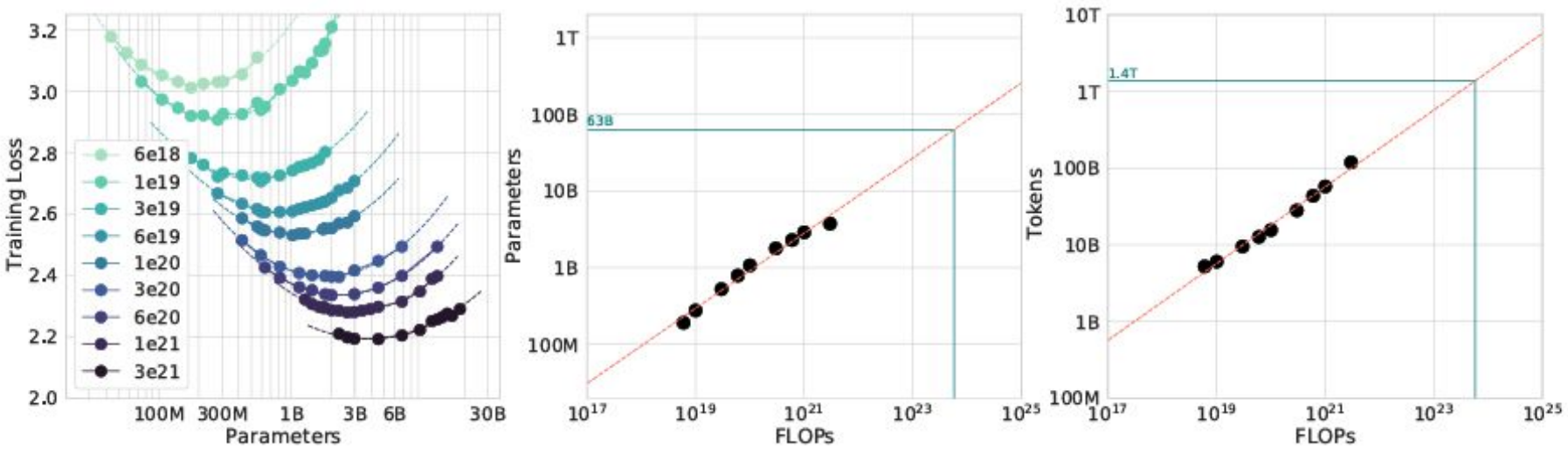


Figure 3 | **IsoFLOP curves.** For various model sizes, we choose the number of training tokens such that the final FLOPs is a constant. The cosine cycle length is set to match the target FLOP count. We find a clear valley in loss, meaning that for a given FLOP budget there is an optimal model to train (**left**). Using the location of these valleys, we project optimal model size and number of tokens for larger models (**center** and **right**). In green, we show the estimated number of parameters and tokens for an *optimal* model trained with the compute budget of *Gopher*.

The Chinchilla law

- Authors of the Chinchilla paper have shown that given a fixed number of parameters X, the optimal training size is ~20X tokens
  - Optimal in the sense that training on more or increase model size only data won't help much
  - It will just waste more resources for training or inference

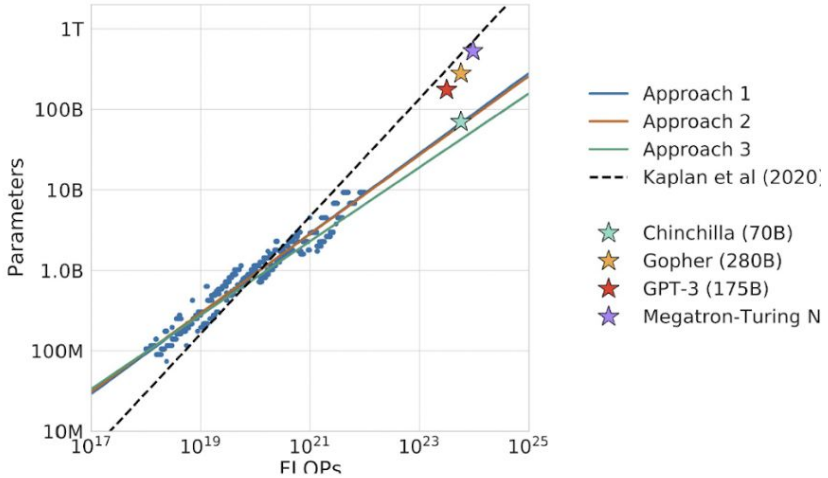
- Paper shows that most models (e.g., GPT-3) are undertrained.

DeepMind

Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*

*Equal contributions



Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

The Chinchilla law Vs power-law

- Optimal model size (N) and optimal data size (D), for a given compute budget (C):

Kaplan et. al. 2020	$N_{\text{opt}} \propto C^{0.73}$	$D_{\text{opt}} \propto C^{0.27}$
Hoffmann et. al., 2021	$N_{\text{opt}} \propto C^{0.5}$	$D_{\text{opt}} \propto C^{0.5}$

2022

$$N_{\text{opt}} \text{ exponent} \cong D_{\text{opt}} \text{ exponent}$$

- Kaplan et al. (2020) suggest that if you increase your compute budget, you should focus more on scaling up model size than data size:
  - For a 10x increase in compute, model size should increase by 5x, while data size should increase by 2x.

- In contrast, Hoffmann et al. (2022) propose scaling model size and data size at the same rate:
  - For a 10x increase in compute, both model and data size should increase by 3.1x.

Central Problem: Optimizing for pre-training loss (perplexity) is an "unreliable" and sometimes "harmful" proxy for desired downstream skills.

Critique 1: "Broken" Scaling Laws

- Performance isn't always a smooth power-law.
- Models show "delayed, sharp inflection points" (e.g., arithmetic), making small-scale extrapolation fail.

Critique 2: "Inverse" Scaling

- On some tasks, **larger models perform *worse***.
- **Cause:** The model learns flawed patterns, bad priors, or misleading heuristics from raw web data (e.g., the "Memo Trap").

Critique 3: "Unreliable" Scaling

- Meta-analysis found pre-training loss is a reliable predictor for downstream tasks only **39% of the time**.

Two parallel solutions to the "Broken/Inverse" scaling problem.

- **1. Data Quality (Q) Scaling**
  - **Paper:** "Textbooks Are All You Need" (Microsoft Phi).
  - **Strategy:** Ditch raw web data. Train on a tiny, curated dataset of "textbook quality" and synthetic data.
  - **Result: phi-3-mini (3.8B params)** rivals **GPT-3.5 (~175B params)**.
  - **New Law:** L(N,D,Q). Quality (Q) becomes an explicit variable.

2. Architectural (MoE) Scaling

  - **Models:** Mixture-of-Experts (e.g., Mixtral, DeepSeekMoE).
  - **Strategy:** Decouple knowledge capacity from computational cost.
    - **Knowledge:** Total Parameters (Ntotal)
    - **Compute Cost:** Active Parameters (Nactive)
  - **Result:** A 47B MoE model (like Mixtral 8x7B) has the inference speed of a ~12B dense model.
  - **New Law:** New MoE-specific laws show optimal models get *sparser* as they scale.

Frontier: The Multi-Vector Hypothesis

The New Consensus: Simple monolithic scaling is over. Performance is a complex function of multiple vectors optimized across the model's *entire lifecycle*.

Vector 1: Total Cost (The "Chinchilla Trap")

- Chinchilla-optimal is *training-optimal*, not *total-cost-optimal*.
- **Strategy (Llama 3):** Train *smaller* models on *extreme* data (e.g., 1,875:1 ratio) to lower lifetime inference cost.

Vector 2: Test-Time Compute (TTC)

- Scale compute *at inference*, not just training.
- **Strategy (OpenAI o1/o3):** Allow the model to "think longer" and use more compute to generate and evaluate reasoning steps before answering.

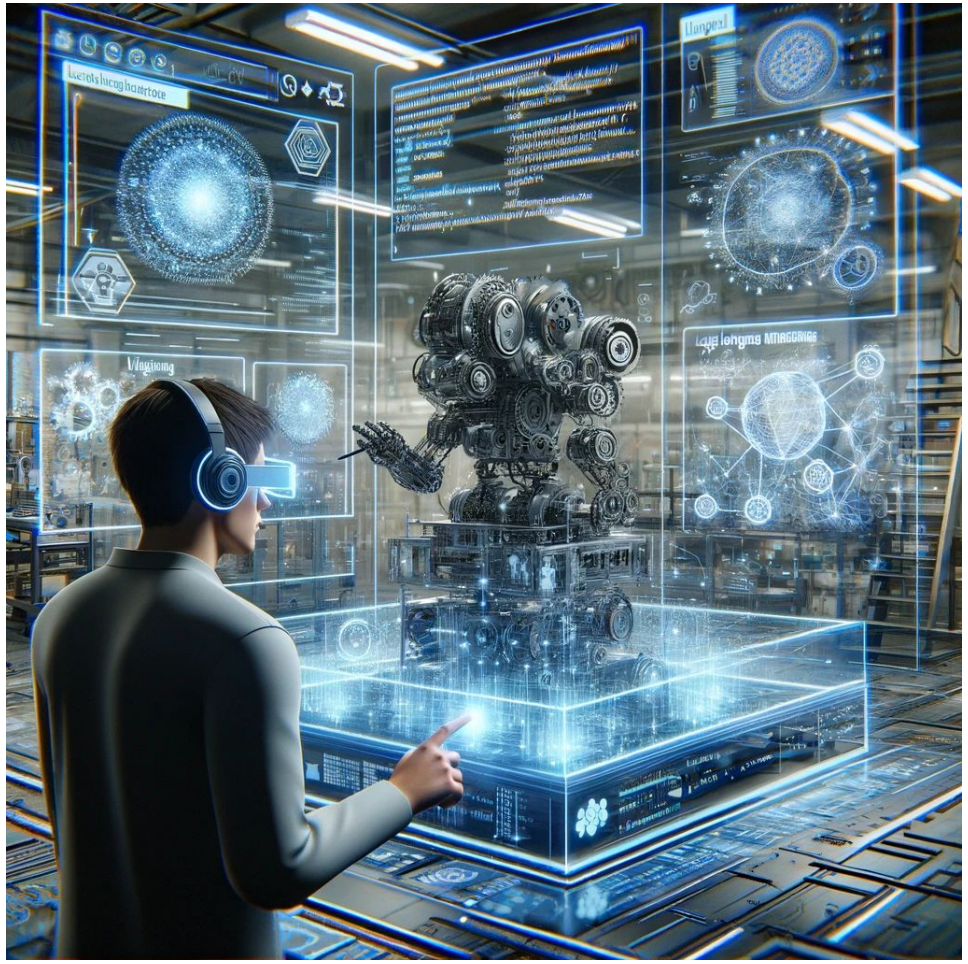
Vector 3: Latent Skills (The "Sloth" Framework)

- Stop modeling general perplexity.
- **Strategy:** Model the scaling of *low-dimensional latent skills* (e.g., "reasoning," "coding") directly.

Mistral

- A French AI startup
- Released a 7B model that surpasses Llama 2's 13B model across all evaluated benchmarks.
- Released **Mixtral 8x22B**, a sparse model with 8 expert pathways, each containing 22B parameters.
- Released **Mistral Large 2 (123B)** in July 2024, a 123-billion-parameter model designed for complex reasoning and high-context applications, supporting a 128,000-token context window.
- Model is also open-source.

Engineering LLMs



AWS Talks

1. **Explore Parameter Efficient Finetuning (LoRA) and Multitask Deployment**

“To tune Large Language Models in a practical and economical fashion, we need Parameter Efficient Finetuning (PEFT); training only ~1% of the model’s parameters and yet still roughly match the performance of a full finetuning. We also explore ideas to deploy adapters for multiple tasks in a single model. Running on a single inference endpoint, hot-swapping adapters as needed.”

1. **Look Ma, I shrunk BERT (Knowledge Distillation)**

“Large NLP models are dazzling us with their awesome performance. But their size can make them cumbersome to handle, making it harder to experiment and more expensive. Let’s use knowledge distillation to leverage the power of a high capacity teacher model to help a smaller student model to learn better, creating a model that is wise beyond its size”

1. **Scaling Training From a Local Machine towards AWS**

“Experimentation requires quick development cycles and can be done locally with scaled down models and datasets, ensuring the correctness of the code/dimensions and observing convergence locally before scaling a training/fine-tuning approach easily to multiple GPUs/nodes. This will allow us to quickly test ideas e.g. custom datasets or even derived model architectures while being able to scale beyond their local machines.

How much does it cost to train a BERT model?

(1) It depends
(2) A lot

Quick 2020 estimates

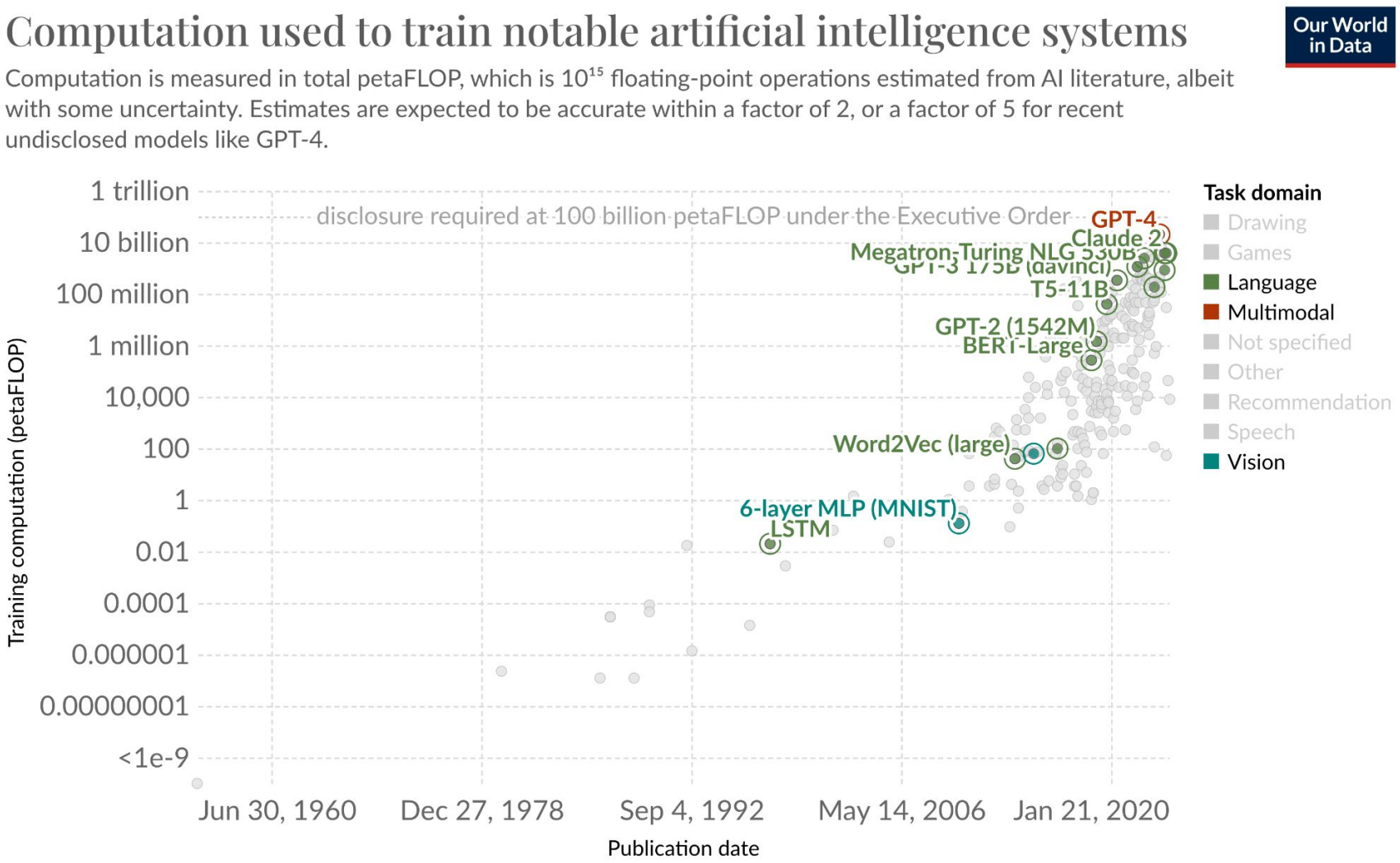
- $2.5k - $50k (110m parameter)
- $10k - $200k (340m parameter)
- $80k - $1.6m (1.5m parameter)

How much are you willing to pay for your project?

€2000?

€20,000?

Not only BERT..



Data source: Epoch (2023)

OurWorldInData.org/artificial-intelligence | CC BY

Note: The Executive Order on AI refers to a directive issued by President Biden on October 30, 2023, aimed at establishing guidelines and standards for the responsible development and use of artificial intelligence within the United States.

Open source ML models and tools

Hugging Face

http://huggingface.co/

Key Hugging face libraries

- `transformers`: Provides APIs and easy-to-use functions to utilize transformers or download pre-trained models.
- `tokenizers`: optimized tokenizers to use with transformer models
- `datasets`: easy access to datasets

huggingface.co/models

Hugging Face | Models | Datasets | Spaces | Docs | Solutions | Pricing | Log In | Sign Up

Tasks | Libraries | Datasets | Languages | Licenses | Other

Filter Tasks by name

Multimodal

- Feature Extraction
- Text-to-Image
- Image-to-Text
- Text-to-Video
- Visual Question Answering
- Document Question Answering
- Graph Machine Learning

Computer Vision

- Depth Estimation
- Image Classification
- Object Detection
- Image Segmentation
- Image-to-Image
- Unconditional Image Generation
- Video Classification

Models 393,321 | Filt | new Full-text search | Sort: Most Downloads

jonatasgrosman/wav2vec2-large-xlsr-53-english
Automatic Speech Recognition • Updated Mar 25 • 75M • 308

bert-base-uncased
Fill-Mask • Updated Jun 30 • 54.1M • 1.19k

timm/mobilenetv3_large_100.ra_in1k
Image Classification • Updated Apr 28 • 53.7M • 10

distilbert-base-uncased-finetuned-sst-2-english
Text Classification • Updated 18 days ago • 42.6M • 340

openai/clip-vit-large-patch14
Zero-Shot Image Classification • Updated Sep 15 • 29.6M • 688

gpt2
Text Generation • Updated Jun 30 • 25.4M • 1.49k

huggingface.co/models

Natural Language Processing

- Text Classification
- Token Classification
- Table Question Answering
- Question Answering
- Zero-Shot Classification
- Translation
- Summarization
- Conversational
- Text Generation
- Text2Text Generation
- Fill-Mask
- Sentence Similarity

huggingface.co/models -> Conversational

Tasks 1 | Libraries | Datasets | Languages | Licenses | Other

Filter Tasks by name | Reset Tasks

Multimodal

- Feature Extraction
- Text-to-Image
- Image-to-Text
- Text-to-Video
- Visual Question Answering
- Document Question Answering
- Graph Machine Learning

Computer Vision

- Depth Estimation
- Image Classification
- Object Detection
- Image Segmentation
- Image-to-Image
- Unconditional Image Generation
- Video Classification
- Zero-Shot Image Classification

Natural Language Processing

Models 2,569 | Filter by name | new Full-text search | ↑↓ Sort: Most Downloads

facebook/blenderbot-400M-distill
Conversational • Updated Mar 30 • 368k • 291

Number of downloads

microsoft/DialoGPT-medium
Conversational • Updated Jan 24 • 120k • 245

microsoft/DialoGPT-small
Conversational • Updated Jun 26 • 27.7k • 58

Who published it?

PygmalionAI/pygmalion-6b
Conversational • Updated Jan 13 • 23.5k • 699

microsoft/DialoGPT-large
Conversational • Updated Jan 24 • 21.8k • 213

Model name

af1tang/personaGPT
Conversational • Updated Aug 20 • 16.6k • 100

facebook/blenderbot_small-90M
Conversational • Updated Jan 24 • 11.5k • 40

Model cards - sentiment from Tweets

Models 245 | sentiment twitter | new Full-text search | ↑↓ Sort: Trending

cardiffnlp/twitter-roberta-base-sentiment
Text Classification • Updated Jan 20 • 10.1M • 220

cardiffnlp/twitter-xlm-roberta-base-sentiment
Text Classification • Updated Jul 19 • 1.12M • 148

cardiffnlp/twitter-roberta-base-sentiment-latest
Text Classification • Updated May 28 • 1.27M • 259

cardiffnlp/xlm-twitter-politics-sentiment
Text Classification • Updated Jan 10 • 259 • 9

Davlan/naija-twitter-sentiment-afriberta-large
Text Classification • Updated Jun 27, 2022 • 515 • 3

Frankie2666/twitter-roberta-base-sentiment
Updated Mar 5, 2021

JP040/bert-german-sentiment-twitter
Text Classification • Updated May 18, 2021 • 178

MichaelReusens/dutch_twitter_sentiment_model
Updated Dec 17, 2021

Twitter Roberta model card

cardiffnlp / **twitter-roberta-base-sentiment** like 220

Text Classification | Transformers | PyTorch | TensorFlow | JAX | tweet_eval | English | roberta | Inference Endpoints | arxiv:2010.12421

Model card | Files and versions | Community 11 | Train | Deploy | Use in Transformers

Edit model card

Twitter-roBERTa-base for Sentiment Analysis

This is a roBERTa-base model trained on ~58M tweets and finetuned for sentiment analysis with the TweetEval benchmark. This model is suitable for English (for a similar multilingual model, see XLM-T).

- Reference Paper: *TweetEval* (Findings of EMNLP 2020).
- Git Repo: Tweeteval official repository.

Labels: 0 -> Negative; 1 -> Neutral; 2 -> Positive

New! We just released a new sentiment analysis model trained on more recent and a larger quantity of tweets. See twitter-roberta-base-sentiment-latest and TweetNLP for more details.

Example of classification

```
from transformers import AutoModelForSequenceClassification
```

Downloads last month
10,147,807

Inference API

Text Classification | Examples

I like you. I love you

Compute

Computation time on Intel Xeon 3rd Gen Scalable cpu: cached

LABEL_2	0.978
LABEL_1	0.019
LABEL_0	0.003

JSON Output | Maximize

Dataset used to train cardiffnlp/twitter-roberta-base-senti…

Twitter Roberta model card

cardiffnlp/twitter-roberta-base-sentiment | like 220

Text Classification | Transformers | PyTorch | TensorFlow | JAX | tweet_eval | English | roberta | Inference Endpoints | arxiv:2010.12421

Model card | Files and versions | Community 11 | Train | Deploy | Use in Transformers

Edit model card

Twitter-roBERTa-base for Sentiment Analysis

This is a roBERTa-base model trained on ~58M tweets and finetuned for sentiment analysis with the TweetEval benchmark. This model is suitable for English (for a similar multilingual model, see XLM-T).

- Reference Paper: *TweetEval* (Findings of EMNLP 2020).
- Git Repo: Tweeteval official repository.

Labels: 0 -> Negative; 1 -> Neutral; 2 -> Positive

New! We just released a new sentiment analysis model trained on more recent and a larger quantity of tweets. See twitter-roberta-base-sentiment-latest and TweetNLP for more details.

Example of classification

```
from transformers import AutoModelForSequenceClassification
```

Downloads last month
10,147,807

Inference API

Text Classification | Examples

I like you. I love you

Compute

Computation time on Intel Xeon 3rd Gen Scalable cpu: cached

LABEL_2	0.978
LABEL_1	0.019
LABEL_0	0.003

JSON Output | Maximize

Dataset used to train cardiffnlp/twitter-roberta-base-senti…

Use case: politicians on “X” (Twitter)

Task: Classify tweets from politicians based on their sentiment

Evaluation problems: translation

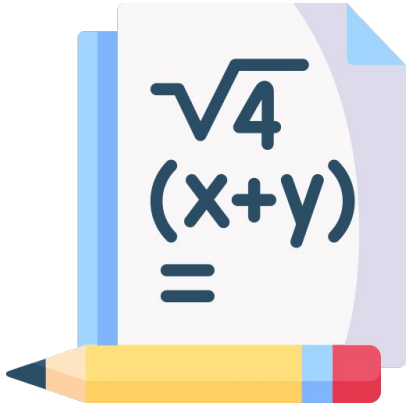
How do we evaluate machine translation?

Evaluation problems: translation

How do we evaluate machine translation (**automatically**)?

Find a metric that **correlates** with scores from **human judges**.

Gold standard



Proxy

Evaluation problems: translation

This is not an easy thing to achieve!

- Do you look at sentence or corpus level?
- It turns out domain of text matters
- Factors that matter:

correlation consistency sensitivity

reliability generality

Evaluation problems: translation

BLEU - BiLingual Evaluation Understudy

- One of the first metrics that reported high correlation
- Key idea: Compare a candidate translation against multiple reference translations.

Evaluation problems: translation

Human translations from Chinese to English

- Reference 1: It is a **guide to action** that ensures that the **military** will forever heed **Party commands**.
- Reference 2: It is the **guiding principle** which guarantees the **military forces** always being under the **command of the Party**.
- Reference 3: It is the **practical guide** for the **army** always to heed the **directions of the party**.

Evaluation problems: translation

Machine translations from Chinese to English

- Candidate 1: It is a guide to action which ensures that the military always obeys the commands of the party.
- Candidate 2: It is to insure the troops forever hearing the activity guidebook that party direct.

Evaluation problems: translation

- Candidate 1: **It is a guide to action** **which** **ensures that the military** **always** obeys the **commands** **of the party**.

- H1: **It is a guide to action** that **ensures that the military** will forever heed Party **commands**.
- H2: It is the guiding principle **which** guarantees the military forces **always** being under the command of the Party.
- H3: It is the practical guide for the army **always** to heed the directions **of the party**.

Evaluation problems: translation

- Candidate 2: It is to insure the troops forever hearing the activity guidebook that party direct.

- H1: It is a guide to action that ensures that the military will forever heed Party commands.
- H2: It is the guiding principle which guarantees the military forces always being under the command of the Party.
- H3: It is the practical guide for the army always to heed the directions of the party.

Evaluation problems: translation

Precision is a core part of BLEU

- precision = number of candidate translation words (unigrams) which occur in any reference translation divided by the total number of words in the candidate translation.

- BUT MT systems can overgenerate “reasonable” words, resulting in improbable but high-precision translations

Candidate: **the the** the the the the the.
H 1: **The** cat is on **the** mat.
H 2: There is a cat on the mat.

Evaluation problems: translation

Next requirement: Translation should be the right length

Candidate: **of the**

H1: It is a guide to action that ensures that the military will forever heed Party commands.

H 2: It is the guiding principle which guarantees the military forces always being under the command **of the** Party.

H 3: It is the practical guide for the army always to heed the directions **of the** party.

Evaluation problems: translation

Why doesn't recall work

Candidate 1: I always invariably perpetually do.
Candidate 2: I always do.

H1: I always do.
H1: I invariably do.
H1: I perpetually do.

Evaluation problems: translation

BLEU solution: a multiplicative brevity penalty factor, finally getting us to the final formula:

Score Calculation in BLEU

Unigram precision $\mathbf{P} = \frac{m}{w_t}$

Brevity penalty $p = \begin{cases} 1 & \text{if } c > r \\ e^{\left(1-\frac{r}{c}\right)} & \text{if } c \leq r \end{cases}$

$$\text{BLEU} = p.\, e^{\sum_{n=1}^{N} \left(\frac{1}{N} * \log Pn\right)}$$

Exercise More Metrics

Results of WMT22 Metrics Shared Task: Stop Using BLEU – Neural Metrics Are Better and More Robust

Markus Freitag[1], **Ricardo Rei**[2,3,4], **Nitika Mathur**[5], **Chi-kiu Lo**[6], **Craig Stewart**[2], **Eleftherios Avramidis**[8], **Tom Kocmi**[7], **George Foster**[1], **Alon Lavie**[2] **and André F. T. Martins**[2,3,9]

[1]Google Research [2]Unbabel [3]INESC-ID [4]Instituto Superior Técnico

[5]Oracle Digital Assistant [6]National Research Council Canada [7]Microsoft

[8]German Research Center for Artificial Intelligence (DFKI) [9]Instituto de Telecomunicações

wmt22-metric@googlegroups.com

Large Language Models

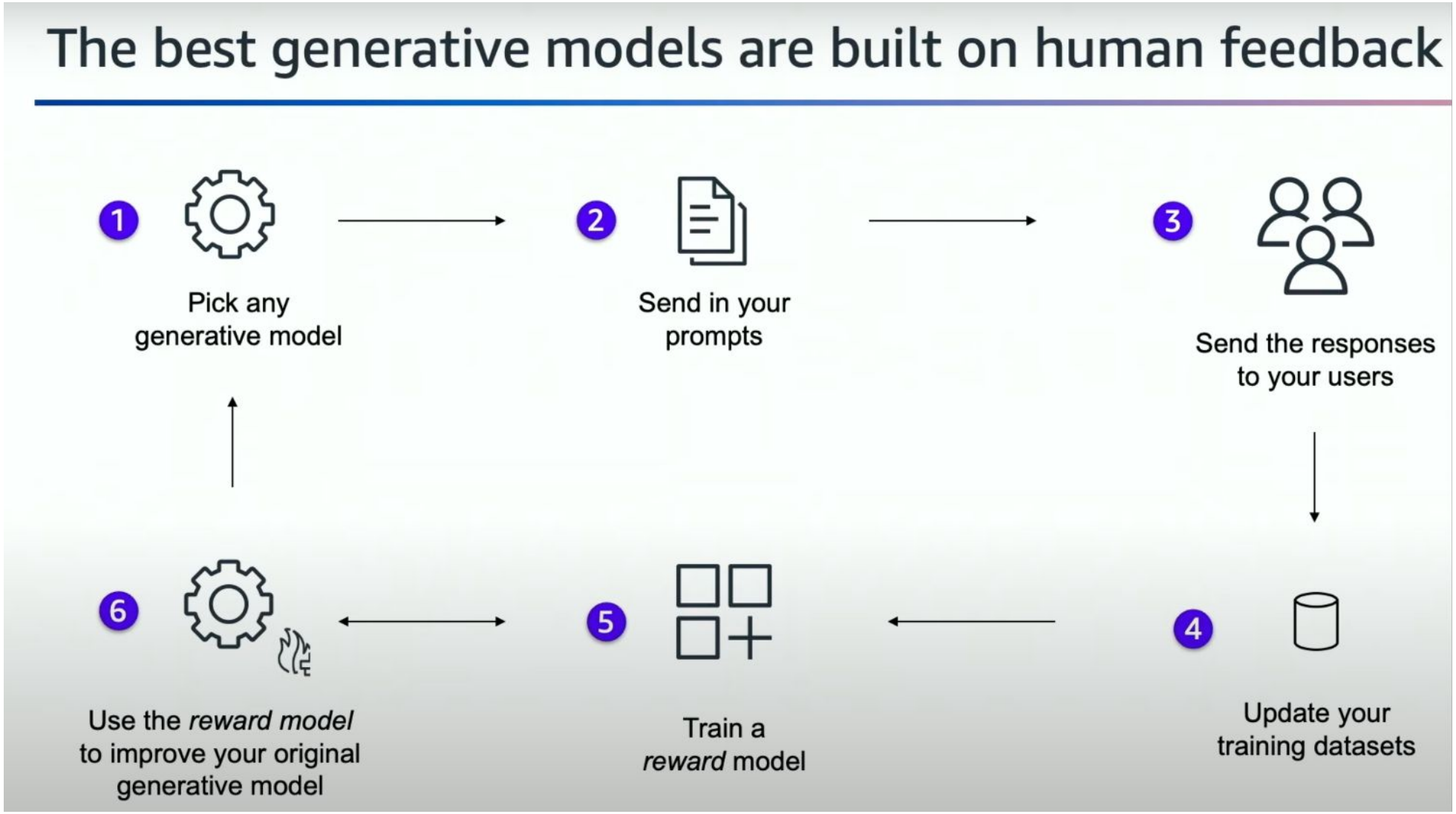
Large Language Models

- Language model with over 100M parameters
  - Largest one so far may be Megatron with over 0.5T parameters
  - GPT-4 was estimated to have 1.7T parameters but it is not official
- Typically use the transformer decoder or a variation/optimization over it
- Trained on vast amounts of data to predict the next word and then typically fine-tuned for chat

Encoding Human Knowledge!

- In some sense, the model's parameters (weights) encode what it has learnt
  - Must use the model to access the corresponding information
- What sets them apart from us is that they can do this really fast
  - Suppose a human can read an online page in 52 seconds
  - There are about 50 billion pages on the internet
  - Would take human over 255K years to go over 11.5% of the internet
  - A foundational model can do this in a few months, provided enough electricity for its GPUs.

Encoding Human Knowledge!



Encoding Human Knowledge!

- In all cases, there is a lot of bad stuff out there on the internet
- Model may learn to be biased or speak in bad ways
  - Reinforcement Learning with Human Feedback is one approach to tackle this issue
  - Let humans give feedback on model responses (e.g., discrete good or bad)
  - Use reinforcement learning to incorporate them in the loss function; reward it when responses are good and penalize it otherwise

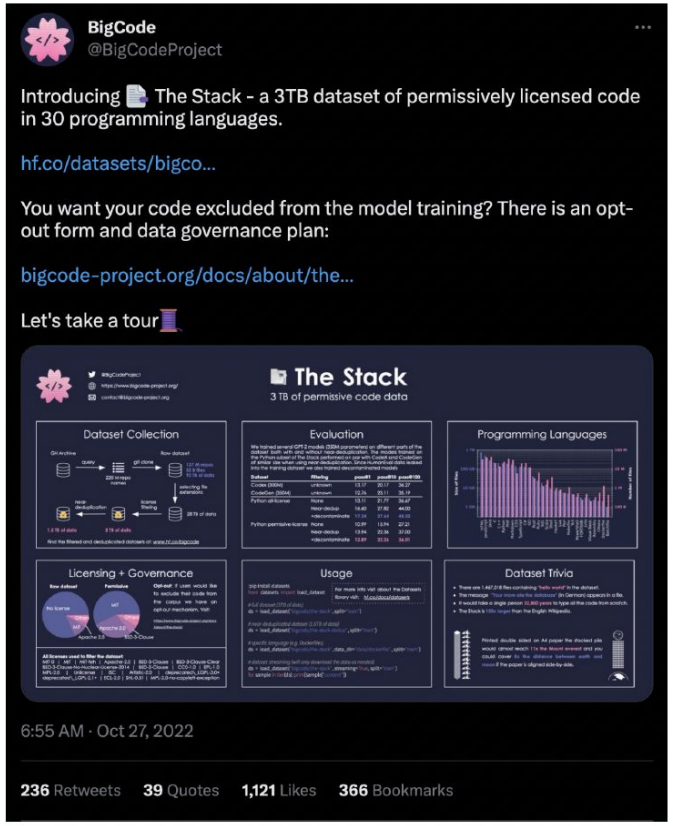
Encoding Human Knowledge!

- Training on code is helpful as well
- Older LLMs such as T5 and GPT-3 removed code
  - OpenAI later released Codex which is a GPT-based model trained on public Github code
  - Empirically, this improves performance even on non-code tasks!
  - Open-source datasets such as The Stack also exist
- Obviously, newer models include both and ordinary text

Including Code Helps!

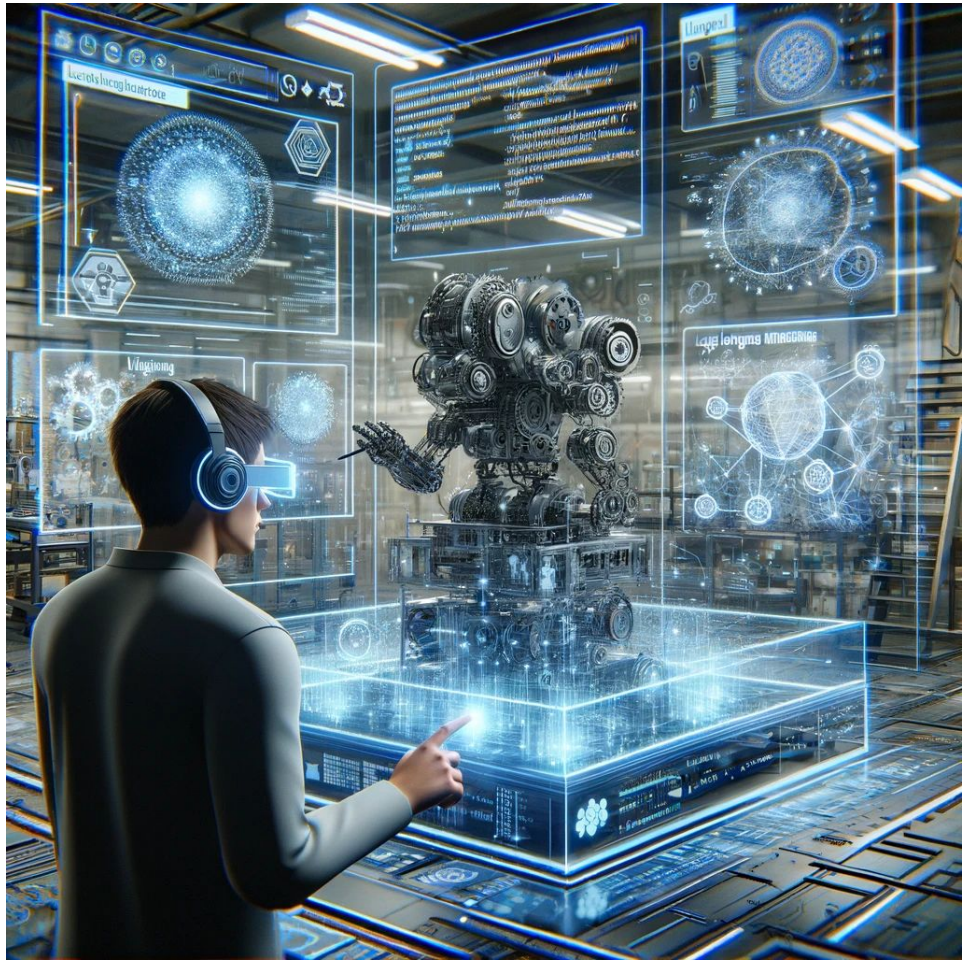
Including code in training data

- T5 and GPT-3 (2020) specifically removed code. But most recent models are trained on ~5% code. Why?
- Code-specific models such as OpenAI Codex (2021) was GPT-3 further trained on public GitHub code.
- Empirically, this improved performance on non-code tasks!
- Open-source dataset: The Stack (3TB of permissively licensed source code)



Yao Fu et al. How does GPT Obtain its Ability?

Engineering with LLMs



So What do we know till now?

Engineering with LLMs

1. **Prompting**:
   - ChatGPT-style UI
   - API, e.g., OpenAI API
   - Command-line with your own instance
2. **Encoding**:
   - Convert NL strings into vector (blog here)
   - E.g., for semantic search (BERT encodings → cosine similarity)
3. **Transfer Learning**:
   - Fine-tune pre-trained model to your specialized domain/task
   - E.g.:
     - Fine-tune BERT to classify financial documents
     - Fine-tune T5 to generate strings corresponding to integers

Engineering with LLMs

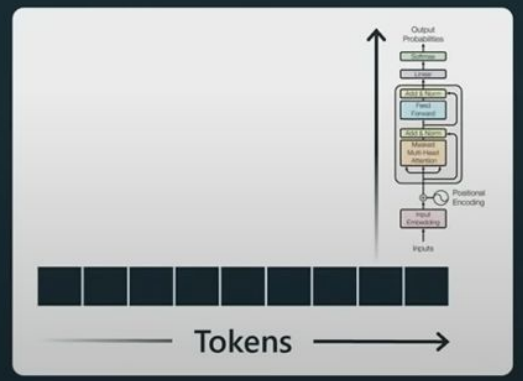
Chain of thought

"Models need tokens to think"

Break up tasks into multiple steps/stages

Prompt them to have internal monologue

Spread out reasoning over more tokens



(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16 / 2 = 8 golf balls. Half of the golf balls are blue. So there are 8 / 2 = 4 blue golf balls. The answer is 4. ✓

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

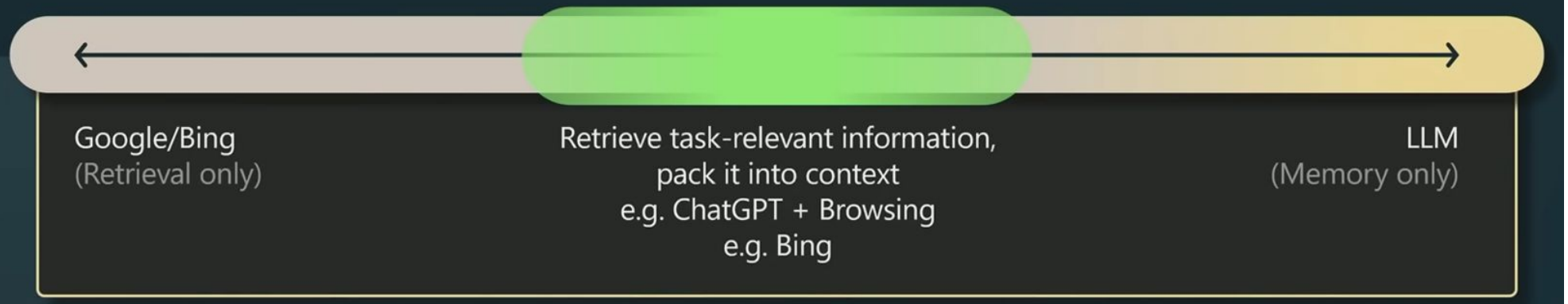
(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

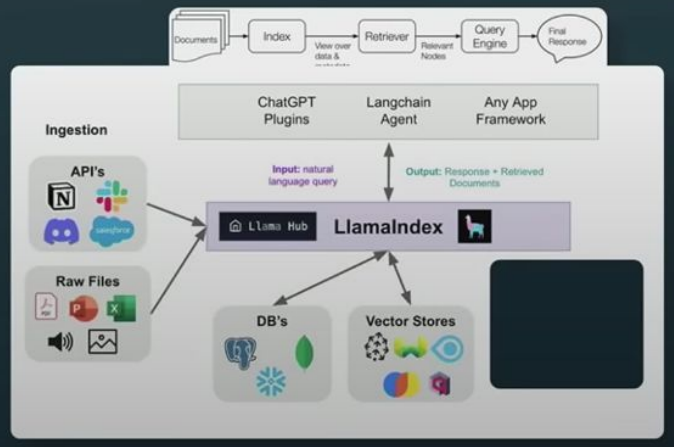
[Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, Wei et al. 2022]
[Large Language Models are Zero-Shot Reasoners, Kojima et al. 2022]

Engineering with LLMs

Retrieval-Augmented LLMs

Load related context/information into "working memory" context window





Emerging recipe:

- Break up relevant documents into chunks
- Use embedding APIs to index chunks into a vector store
- Given a test-time query, retrieve related information
- Organize the information into the prompt

Engineering with LLMs

Tool use / Plugins

Offload tasks that LLMs are not good at
Importantly: they don't "know" they are not good

Intersperse text with special tokens that call external APIs

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

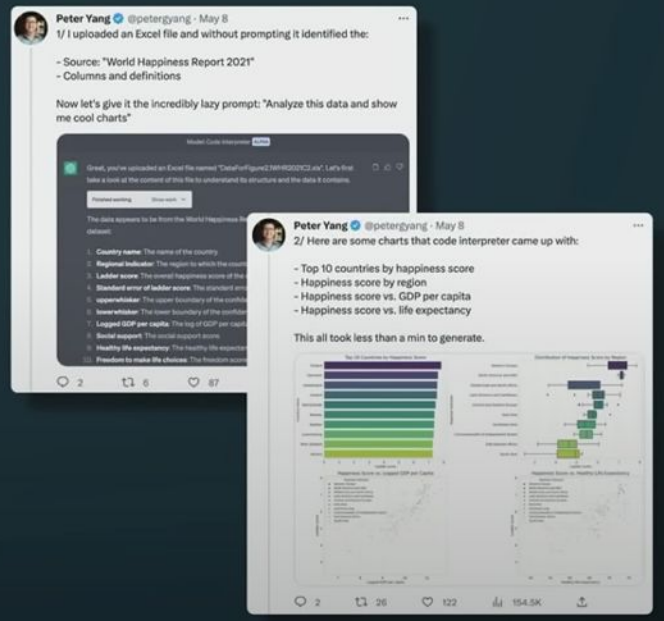
The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.

Figure 1: Exemplary predictions of Toolformer. The model autonomously decides to call different APIs (from top to bottom: a question answering system, a calculator, a machine translation system, and a Wikipedia search engine) to obtain information that is useful for completing a piece of text.

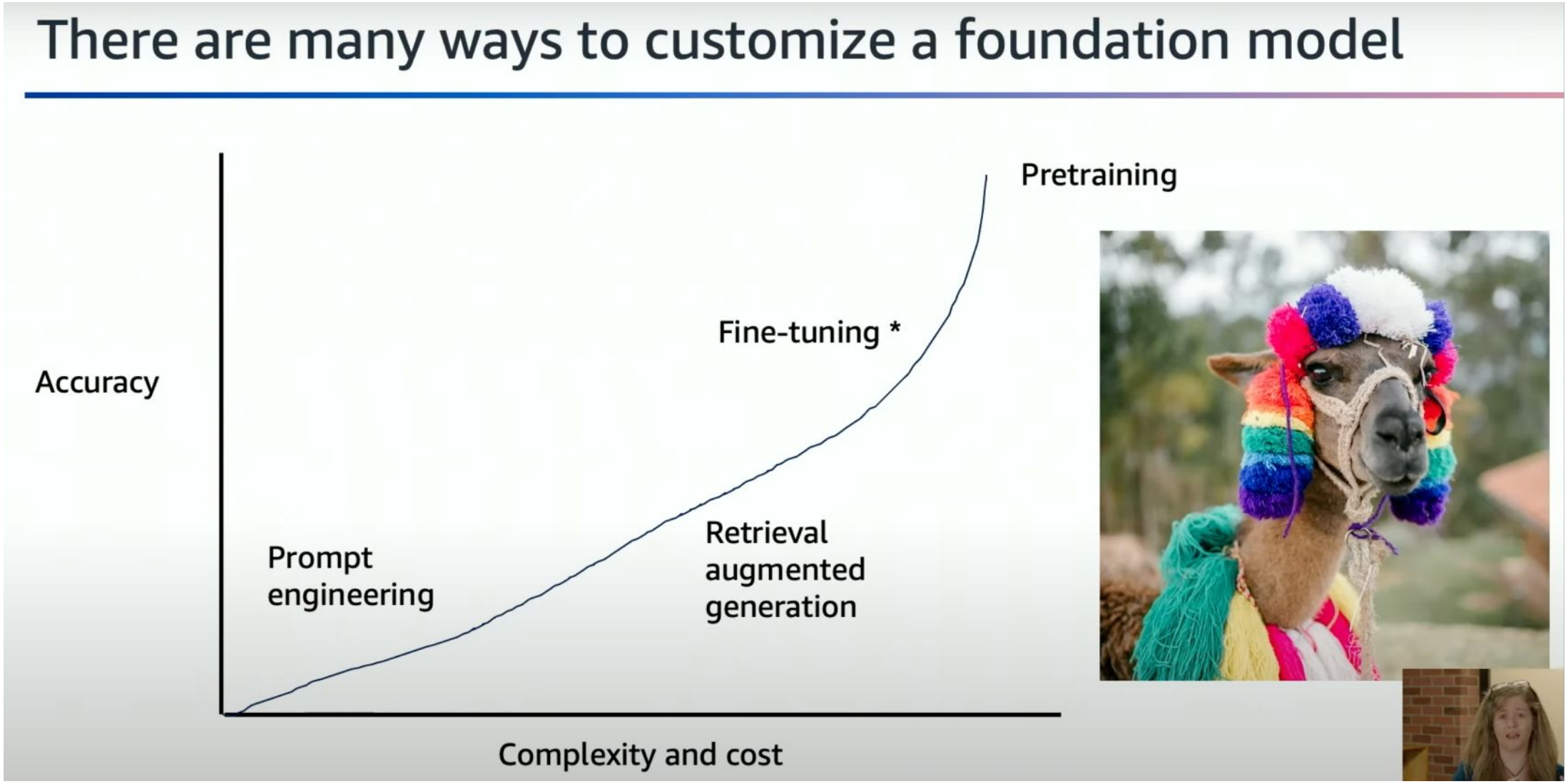
[Toolformer, Schick et al. 2023]

ChatGPT plugins

ChatGPT Code Interpreter



Engineering with LLMs



Conclusions

References and Resources

Stanford's NLP with Deep Learning: http://web.stanford.edu/class/cs224n/

Deep Learning for NLP, Stephan Raaijmakers

Jay AlAmmar, the Illustrated Transformer:
http://jalammar.github.io/illustrated-transformer/

Full Stack LLM Bootcamp: https://fullstackdeeplearning.com/llm-bootcamp/

Next Lecture: LLM Training, Fine-Tuning and Alignment

Next Exercise: Introduction to N8N