From Prompt to Product: A Workshop on the Working Principles of Large Language Models

Mushahid Intesum
Fabiha Haider

Topics

LLM Introduction

Text Representation

Transformer Model & Self-Attention Mechanism

Pre-training & Fine-tuning

Benchmark

Applications

What is LLM?

Large

Trained on very large data

Model size has billions of parameters

Language Model

Model pre-trained on language modeling task to create a foundation model

A brief history

1950s - 2000s

Rule-based and Statistical Methods, pre deep learning

2006 - 2011

Early deep learning
Deep learning shows promise
ReLU and other activation functions introduced

2012 - 2016

Deep learing breakthrough
AlexNet beats ImageNet.
GAN, ResNet, Word2Vec, Attention mechanism

2017 - 2020

Transformers introduced
BERT, GPT2

2020 - 2022

LLMs introduced
GPT3, Dall-E

2023 - present

Instruction tuned and agentic LLMs
Claude, Gemini, Qwen, DeepSeek

Pre-Requisites

Some Pre-Requisites

Parameters

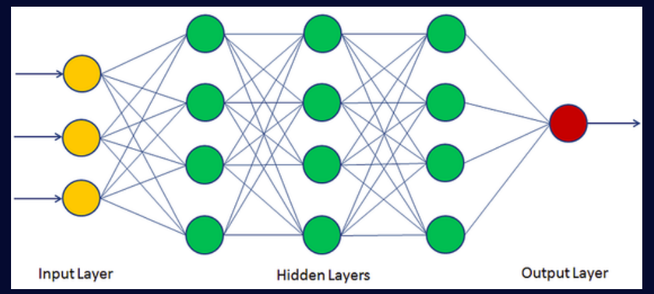
Actual trainable values of neural networks

Represented in matrix form using Tensors

W . x + **b** = y

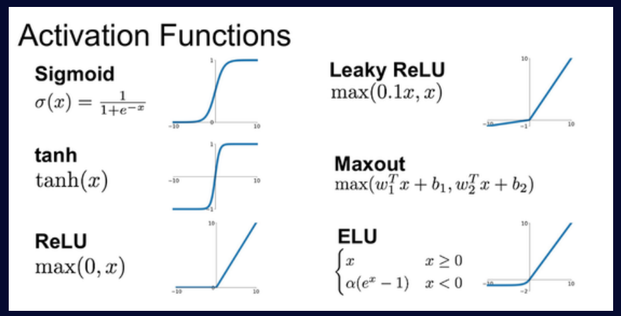
Multi-layer perceptron

A basic feed-forward neural network with multiple hidden layers



Activation Functions

Non-linear functions that enable neural nets to learn non-linear relationships



Language Modeling

Predict next word given previous words

This is a language modeling task

Input	Target
This	is

Language Modeling

Predict next word given previous words

This is a language modeling task

Input	Target
This	is
This is	a

Language Modeling

Predict next word given previous words

This is a language modeling task

Input	Target
This	is
This is	a
This is a	language

Language Modeling

Predict next word given previous words

This is a language modeling task

Input	Target
This	is
This is	a
This is a	language
This is a language	modeling

Language Modeling

Predict next word given previous words

This is a language modeling task

Input	Target
This	is
This is	a
This is a	language
This is a language	modeling
This is a language modeling	task

Text Processing

Text Representation

Transform Text to Machine readable format

- Machines cannot understand human language
- Transform input text into intermediate representations to feed into the network
- Tokenization, Positional Encoding, Embeddings

Text Tokenization

- Split words of each sentence & make a “vocabulary”
- Assign each word in vocabulary a unique value which serves as token for that word
- Sentence Piece, Byte-pair encoding, Index tokenizer; pick whichever is suitable

Special tokens are used to signify special characters:

<BOS> : Beginning of sentence

<EOS>: End of sentence

<UNK>: Unknown word

<SPACE>: Space between words

Text Tokenization : Tokenize by Index

I love learning
Learning is my hobby

I (0)
love (1)
learning (3)
...
<BOS> (6)
<EOS> (7)

I love learning

<BOS> I love learning <EOS>

[“6” ,“0” ,“1”, “3”, “7”]

Text Tokenization: Positional Encoding

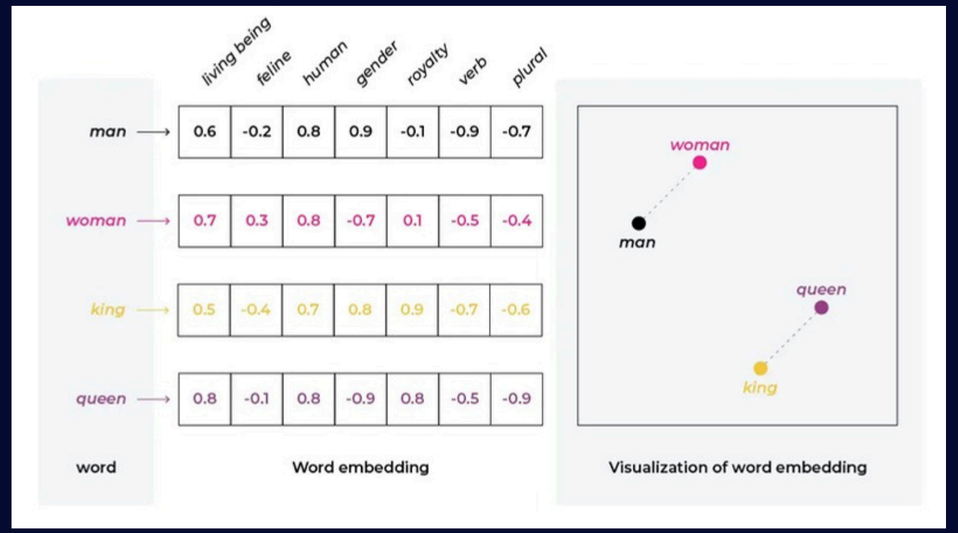
- Assigns positional information of tokens in a text
- Needed as attention mechanism of transformers does not store positional information
- Will be discussed more for when discussing transformers

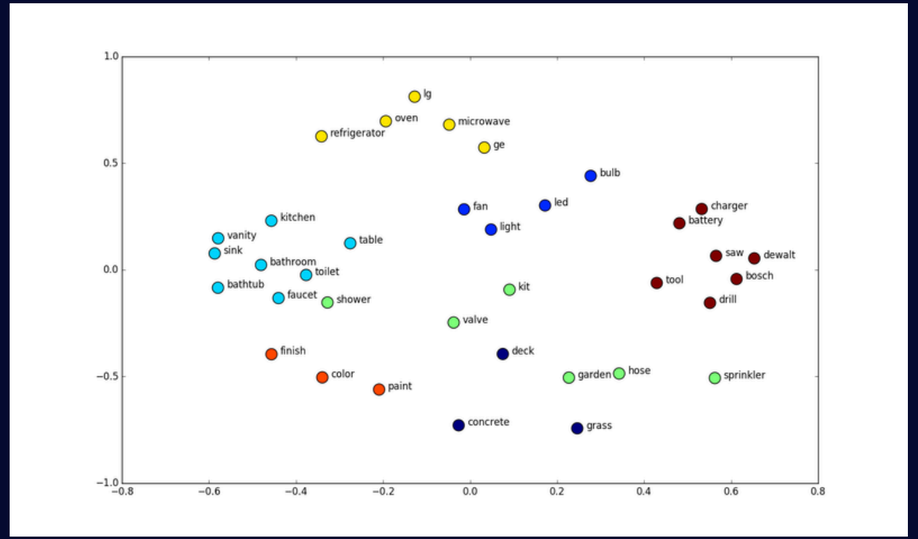
$$\mathrm{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

$$\mathrm{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right)$$

What are Word Embedding?

- Representation of text in high-dimensional vector form
- Able to capture relationship between words **(eg. *man + woman - king = queen)* and clusters similar things**
- Made using neural networks utilizing methods such as **CBOW, Skip-Gram and Co-Occurrence Matrix**
- Methodologies**: word2vec, GLoVe**
- Currently, Pytorch has trainable Embedding Layer





Batching Text Data

- Text is broken into blocks, multiple blocks are aggregated to a batch

- Broken into blocks:
  - Cannot process entire data at once due to memory constraints

  - Model will not learn anything if one token is fed at a time, no context to learn from

- Blocks are batched:
  - Utilize parallelizing capability of GPU, making training efficient

Batching Text Data

This is a long sentence which will be batched for processing large language models. The sentence is used for illustration purpose.

Block size : 4, Batch size : 2, No. of Batches: 3

Batch: 1 { **This is a long sentence**
which will be batched

Batch: 2 { **for processing large language**
models. The sentence is

Batch: 3 { **used for illustration purpose.**
<pad> <pad> <pad> <pad>

Putting the ‘Large’ in Large Language Model

Data

CommonCrawl:
410 billion tokens
3.3 TB disk space

Wikipedia:
3 billion tokens
83 GB disk space

Data used to train *Llama2*:
2T tokens

Data used to train GPT-3:
499B tokens

Data used to train *Llama3*:
15T tokens

Model Size

GPT-3:
175 billion parameters
350GB disk space

Llama3:
405 billion parameters
2TB disk space

Text Representation: Putting everything together

Collect data and prepare vocabulary → Create tokens using an encoding scheme based on requirements → Encode tokens with positional encoding → Embeddings

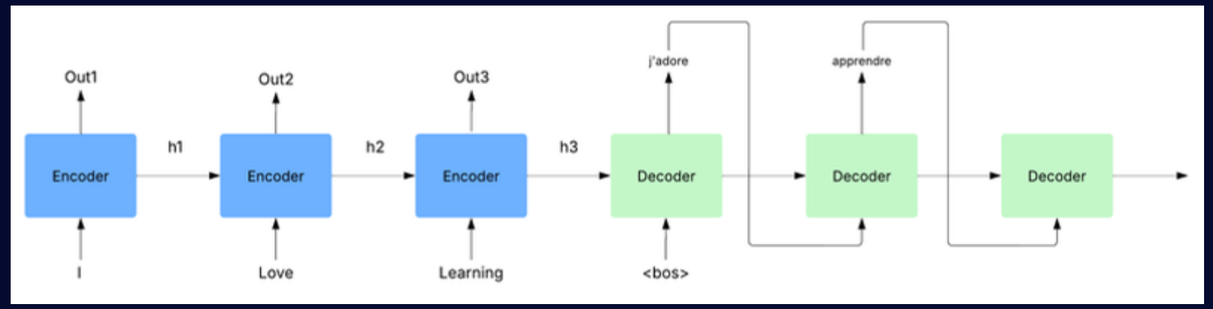
Transformers and Attention

Transformers and Attention

- Introduced in the seminal paper ‘Attention is All You Need’ in 2017
- Provided solutions to many issues of previous methods such as handling long contexts effectively
- Backbone of modern day LLMs

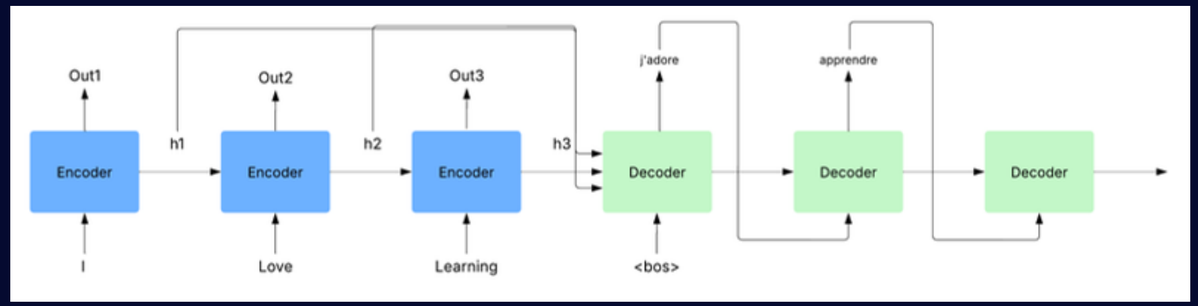
Sequence to Sequence Models

- Before Transformer, Seq2Seq models like RNN, GRU, LSTM were used for specific tasks
- Consisted of **Encoder & Decoder**
- Encoder passed a **hidden state** to decoder which contained all context information
- Struggled with long context



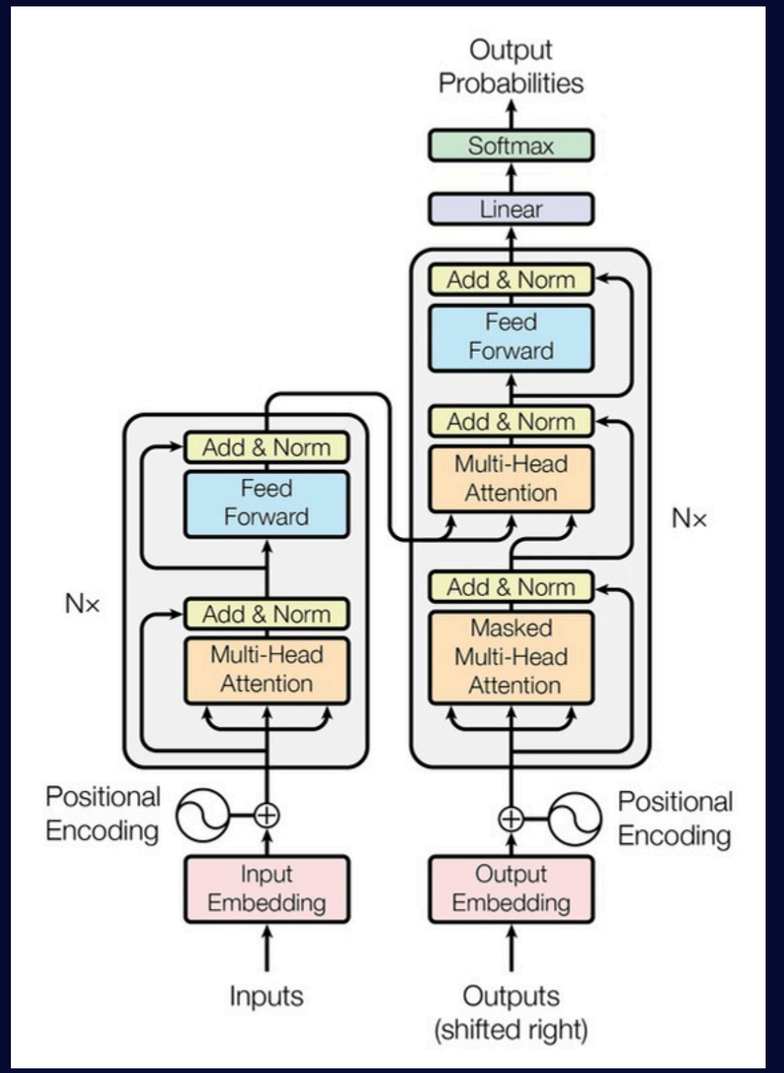
‘Attention’ in Seq2Seq Models

- Instead of passing hidden state from only previous layer, send hidden state of all previous and future layers
- This is called **Self-Attention**
- Though promising, the sequential nature of seq2seq models still proved to be a bottleneck



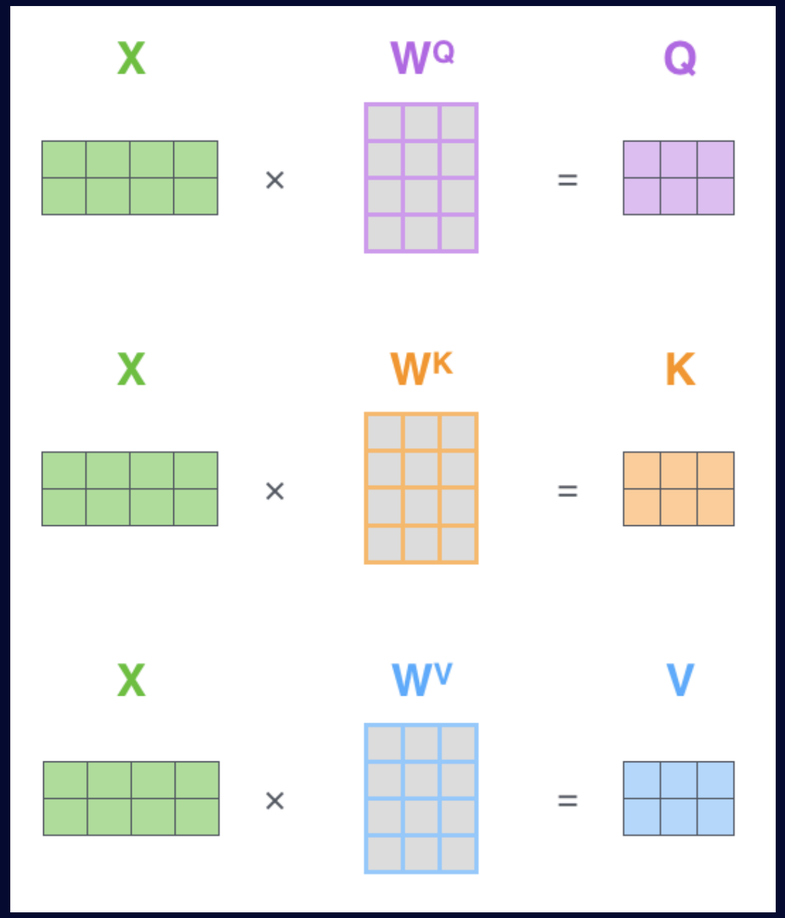
Transformers to the Rescue!

- Eliminate sequential dependence by capturing global context
- Multi-head Attention to fully capture token dependency
- Easily scalable to large sizes

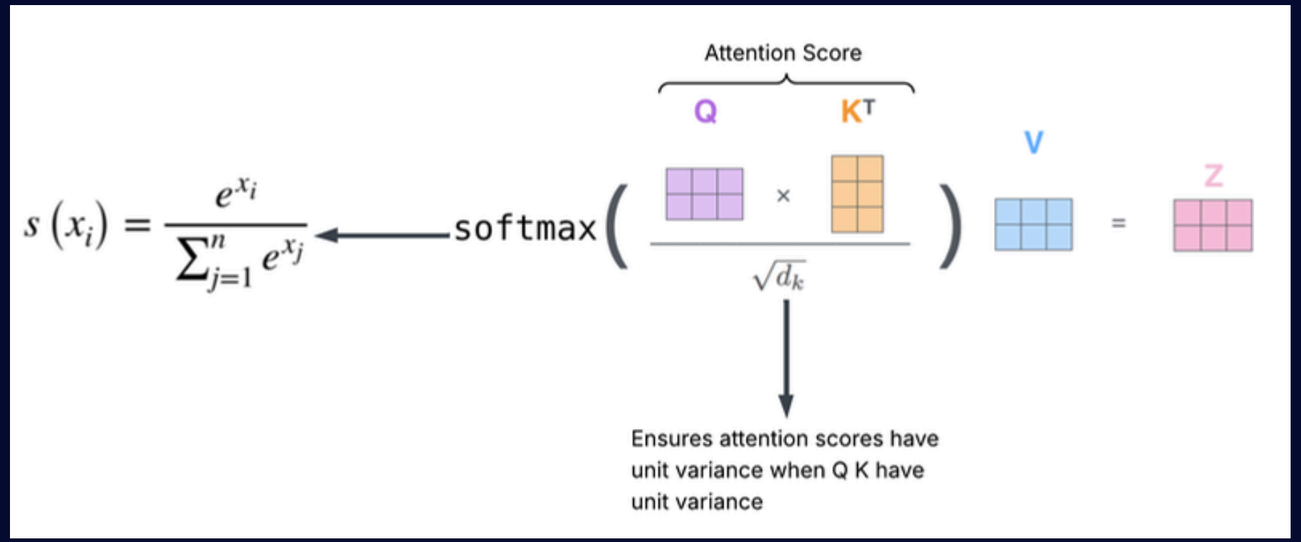


Attention Mechanism

Extract Query(Q), Key(K) and Value(V) vectors from the input text

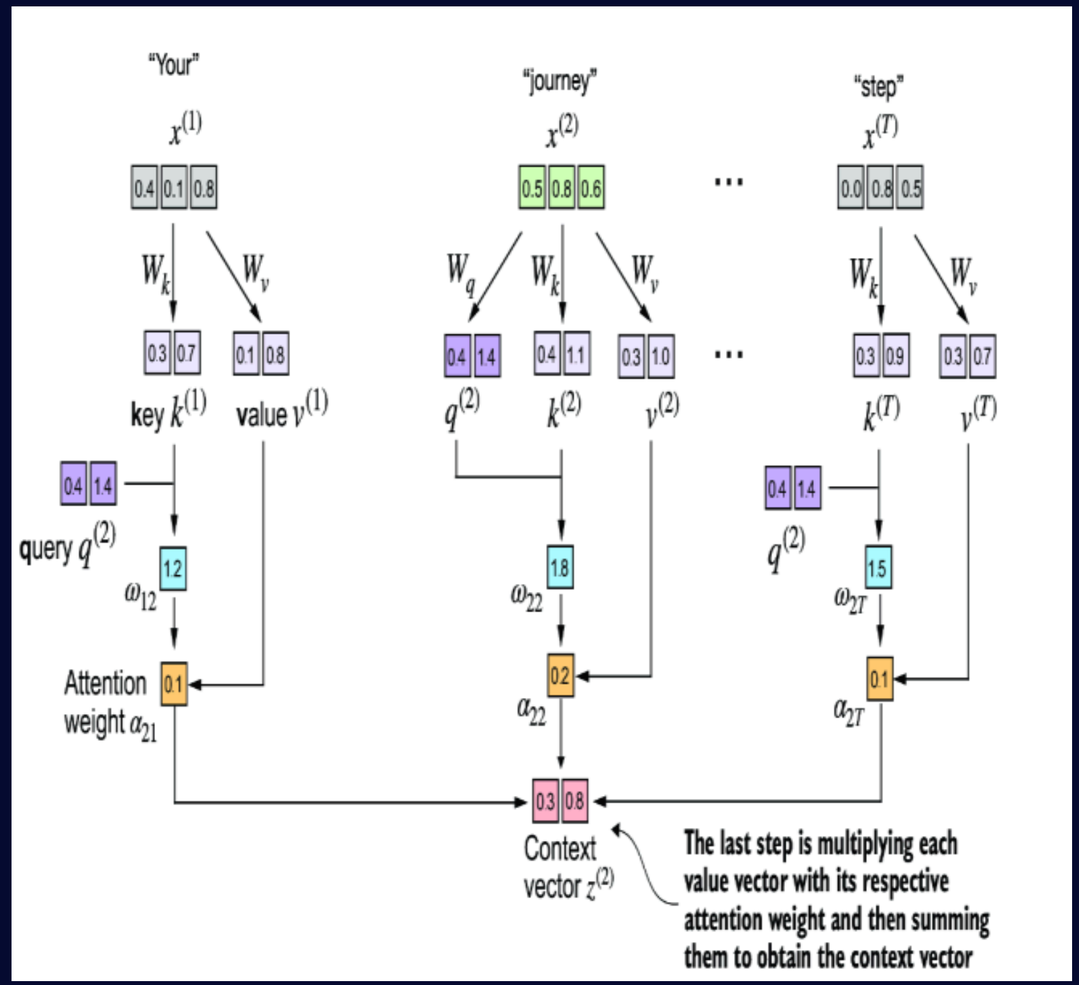


Calculate attention scores based on the formula given in the next slide



What do Q, K, V vectors mean?

- Query is the question where a token asks the other tokens of their relevance or importance
- Key is the information stored in the other tokens
- By multiplying the key and the query we get the attention score (the relevance) between the query token and one of the other tokens
- The input text can be thought of as hidden information, and QKV are the classified information it is willing to divulge
- These abstractions enable better understanding of the text by the model



Masked-Attention

- Attention does not care about order, information from any state can be passed to anyone else, useful for tasks such as translation and sentiment analysis
    - Full context needed for proper understanding
- Language modeling only needs information from past values
    - Mask out attention scores from future tokens to limit propagation

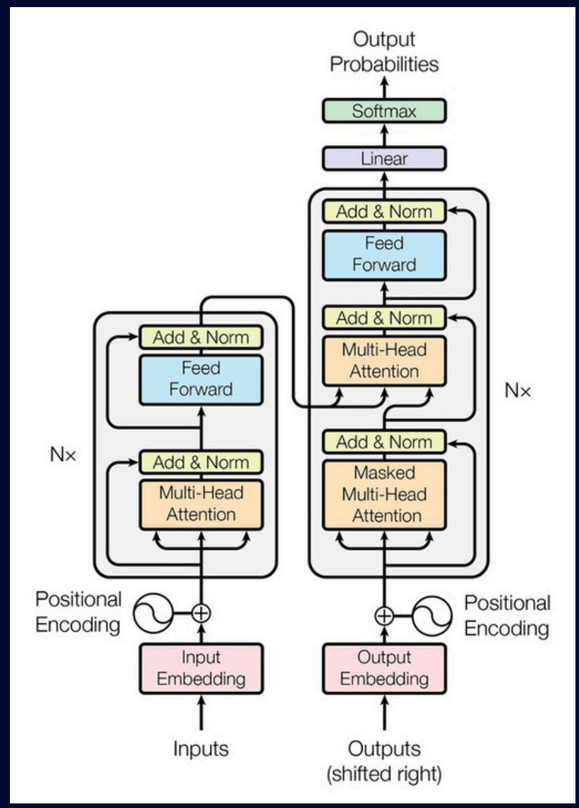
0.4	0.1	0.2
0.6	0.8	0.1
0.7	0.9	0.3

+

0	-inf	-inf
0	0	-inf
0	0	0

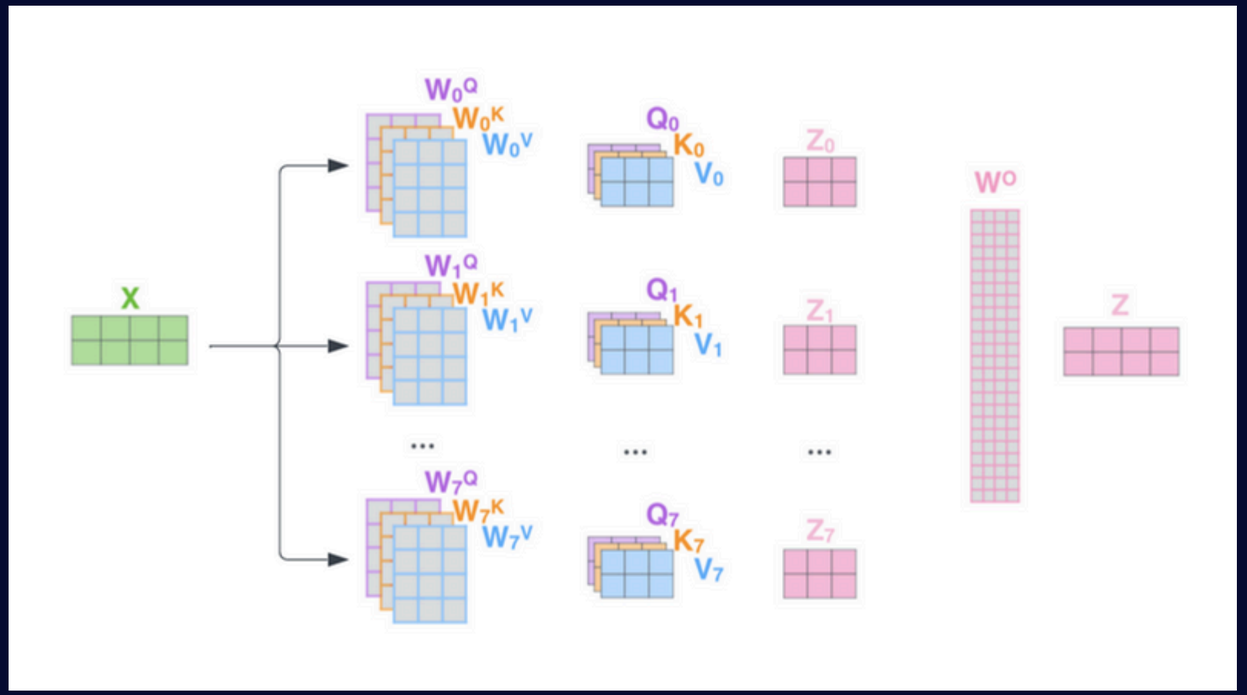
→

0.4	-inf	-inf
0.6	0.8	-inf
0.7	0.9	0.3



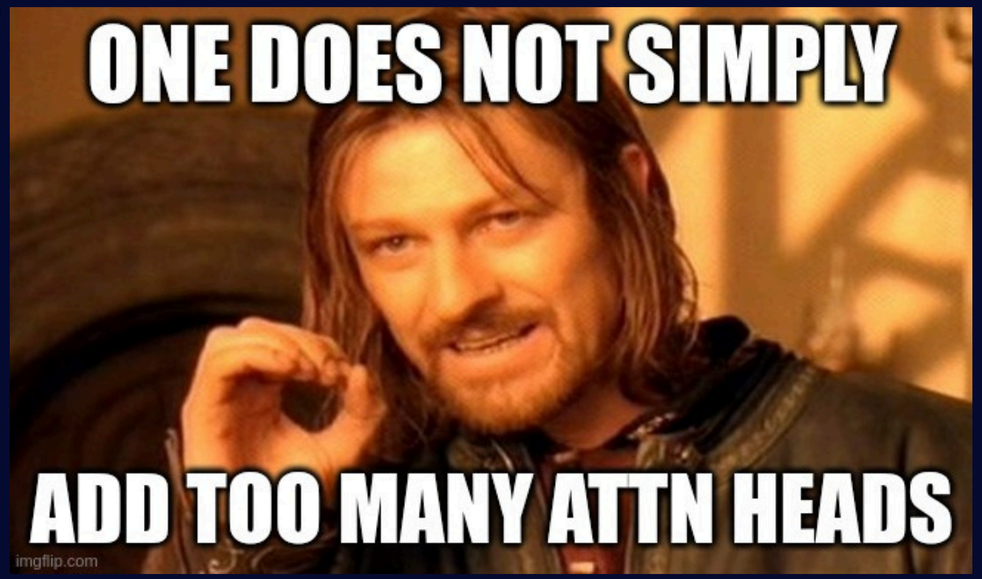
Multi-Head Attention

- Instead of calculating 1 attention score for each input block, calculate multiple attention scores for each input (**attention head)**
- Each input will have *n* Q-K-V matrices and *n* corresponding attention scores
  - Some designs have shared weight matrices, others have separate weight matrices for each attention head
- Scores are concatenated, multiplied by a shared weight matrix to get the final score



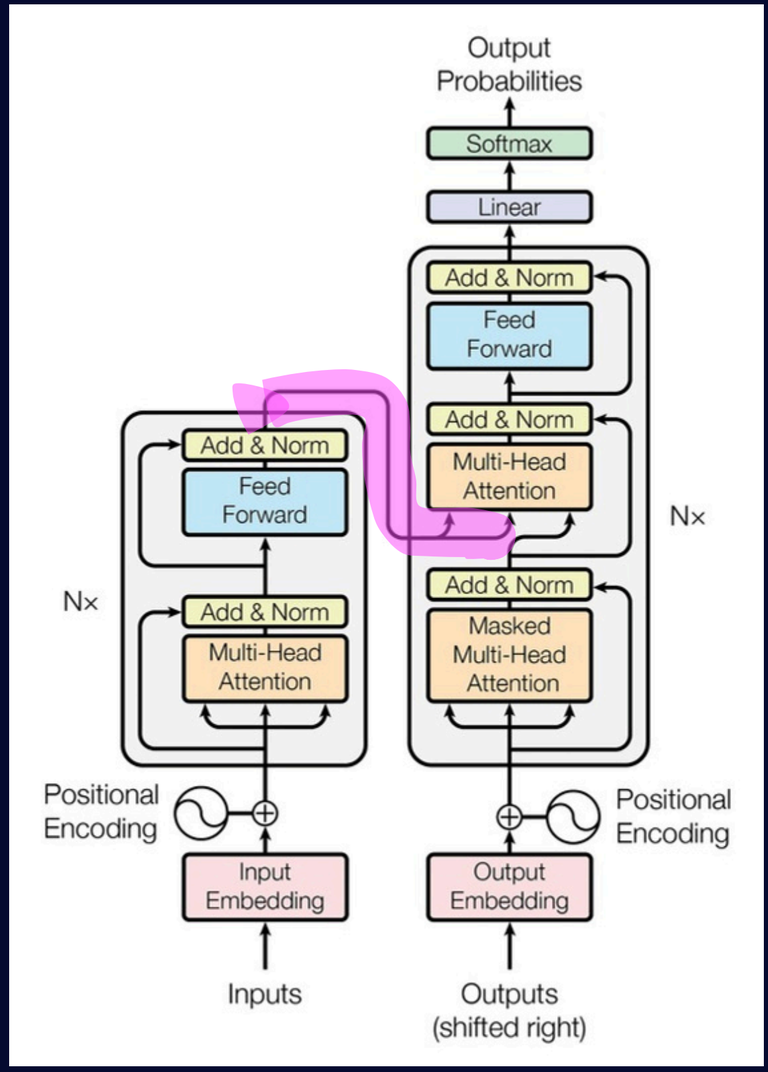
Multi-Head Attention (Contd.)

- A single attention head might cause a token pay ‘attention’ to only one token
- Multiple-attention heads adds an element of randomness
- This randomness allows the model to pay attention to other tokens as well, enabling better learning
- More than one heads yields better result
  - Adding too many heads will cause the model to reach an eventual plateau
  - Will result in unnecessary computation overhead



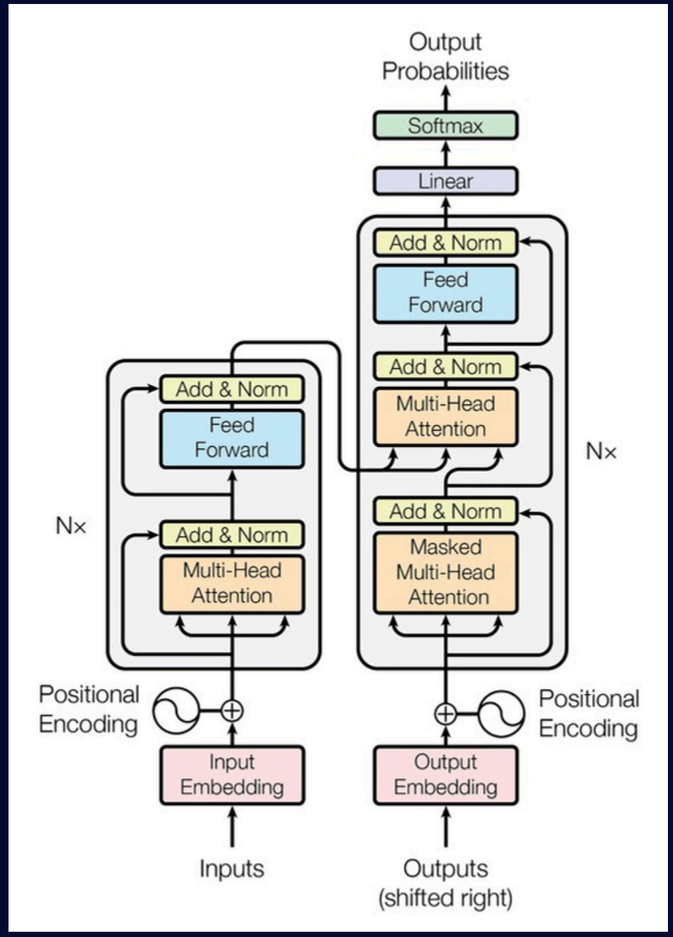
Cross Attention

- Encoder ‘Remembers’ what the input text means or is trying to say
- Decoder ‘Processes’ what it has received and tries to come up with an output
- Tasks like translation and QA needs to understand information it has received as input
- Attention score of encoder is passed to decoder so that this can be taken into account



Putting everything together is all you need

- Feed-forward layers are applied so that the model can have time to process the attention values
- Layer-norm ensures that the variance of the values in each layer have unit variance
  - Not doing so may cause high variance
  - Give more weight to some values than others
- Positional encoding used as attention scores are position invariate



LLM Architecture

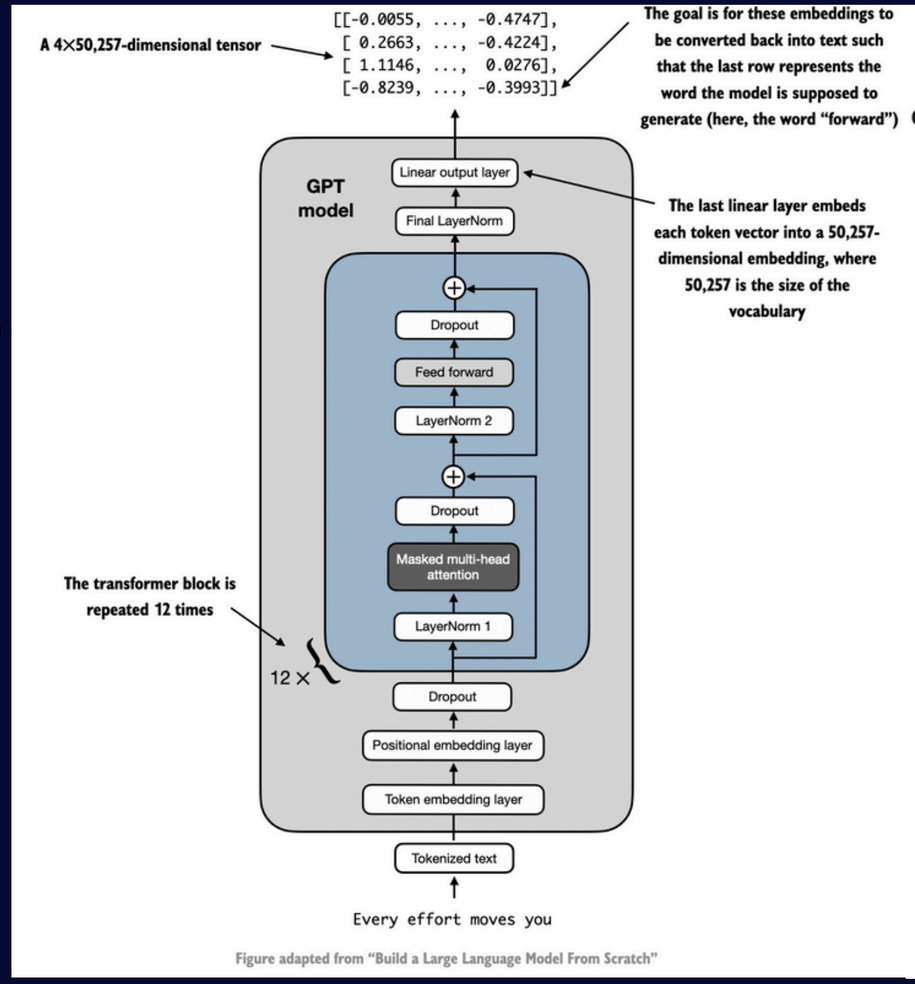


Figure adapted from "Build a Large Language Model From Scratch"

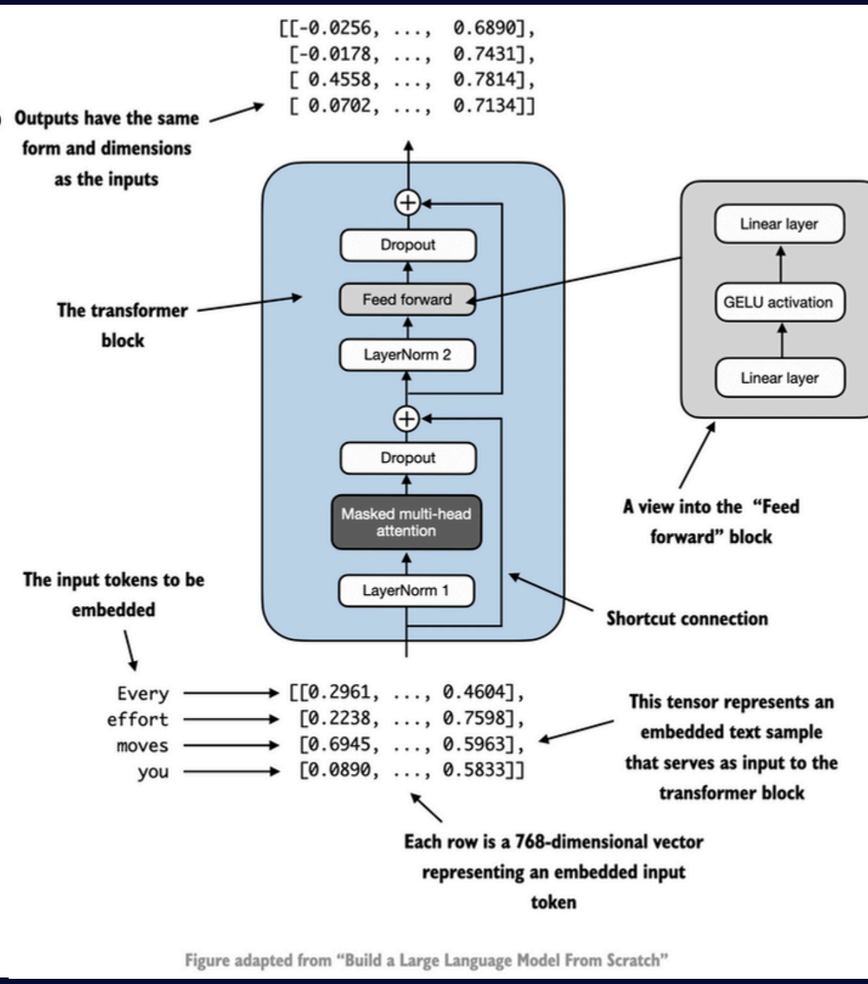


Figure adapted from "Build a Large Language Model From Scratch"

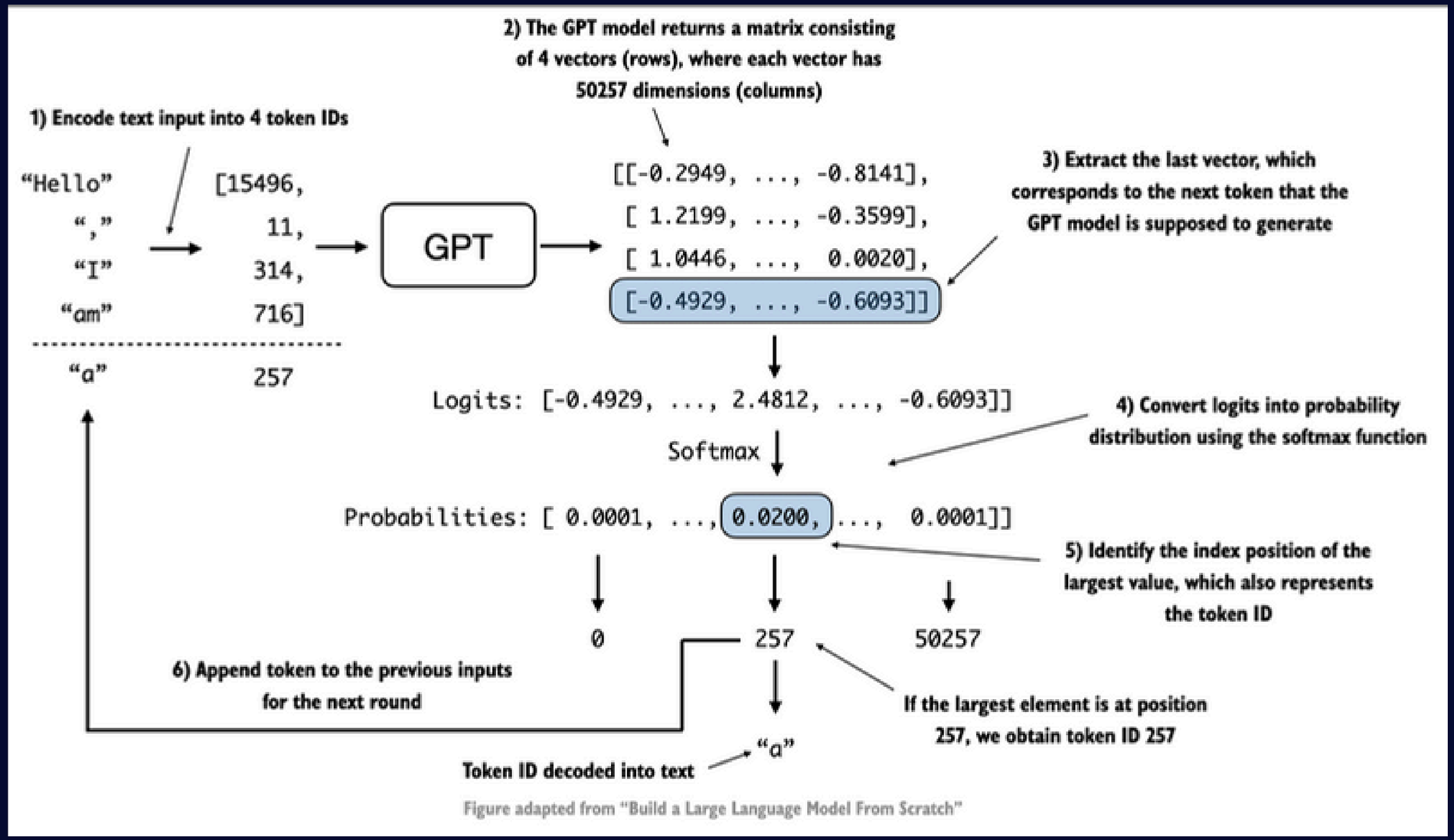
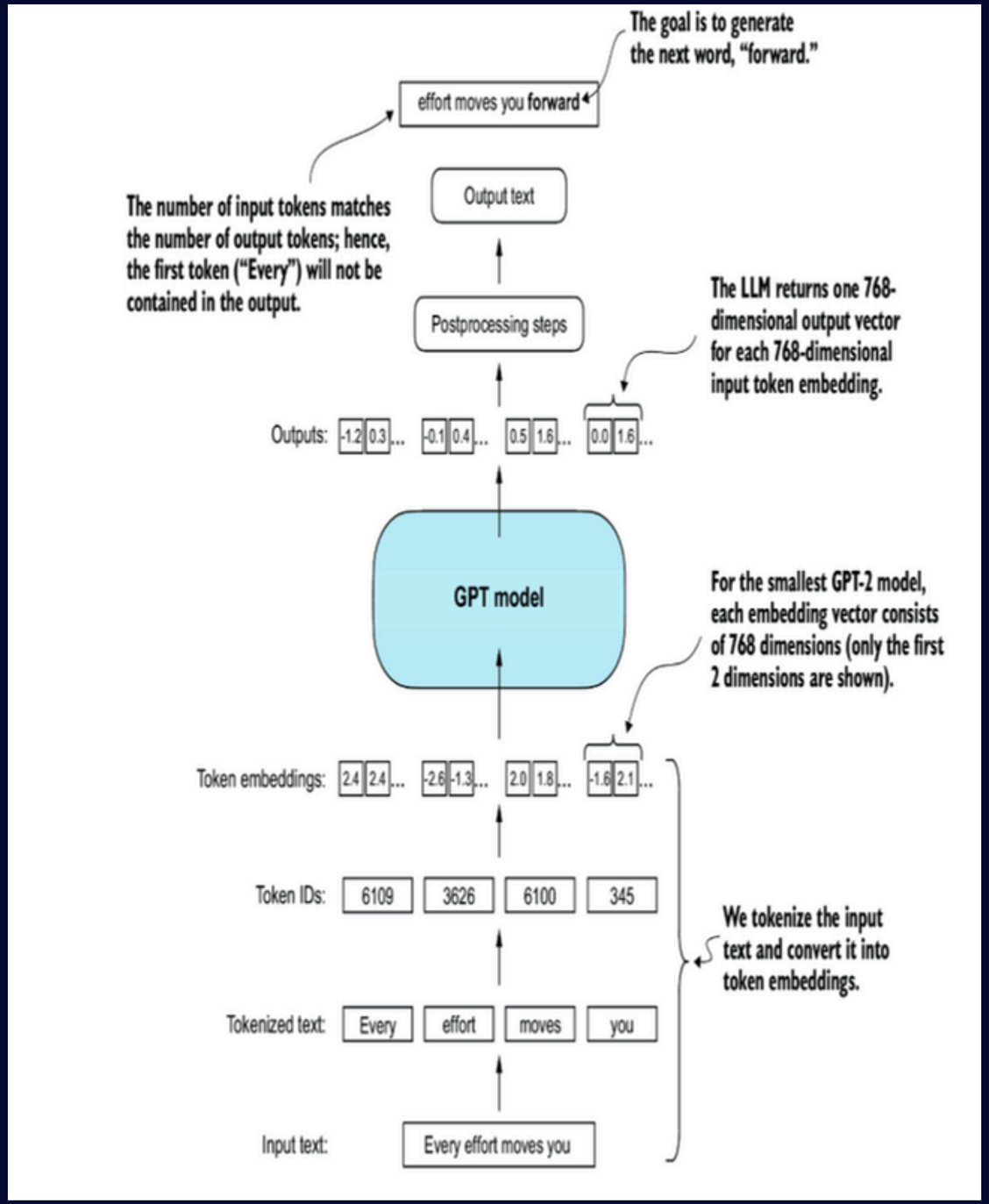


Figure adapted from "Build a Large Language Model From Scratch"


The goal is to generate the next word, "forward."
effort moves you forward
The number of input tokens matches the number of output tokens; hence, the first token ("Every") will not be contained in the output.
Output text
Postprocessing steps
The LLM returns one 768-dimensional output vector for each 768-dimensional input token embedding.
Outputs: -1.2 0.3 ... -0.1 0.4 ... 0.5 1.6 ... 0.0 1.6 ...
GPT model
For the smallest GPT-2 model, each embedding vector consists of 768 dimensions (only the first 2 dimensions are shown).
Token embeddings: 2.4 2.4 ... -2.6 -1.3 ... 2.0 1.8 ... -1.6 2.1 ...
Token IDs: 6109 3626 6100 345
We tokenize the input text and convert it into token embeddings.
Tokenized text: Every effort moves you
Input text: Every effort moves you

Pre-Tuning and Fine-Tuning

Pre-Training

The next -word prediction is task is a form of self-supervised learning, which is a form of self-labeling.

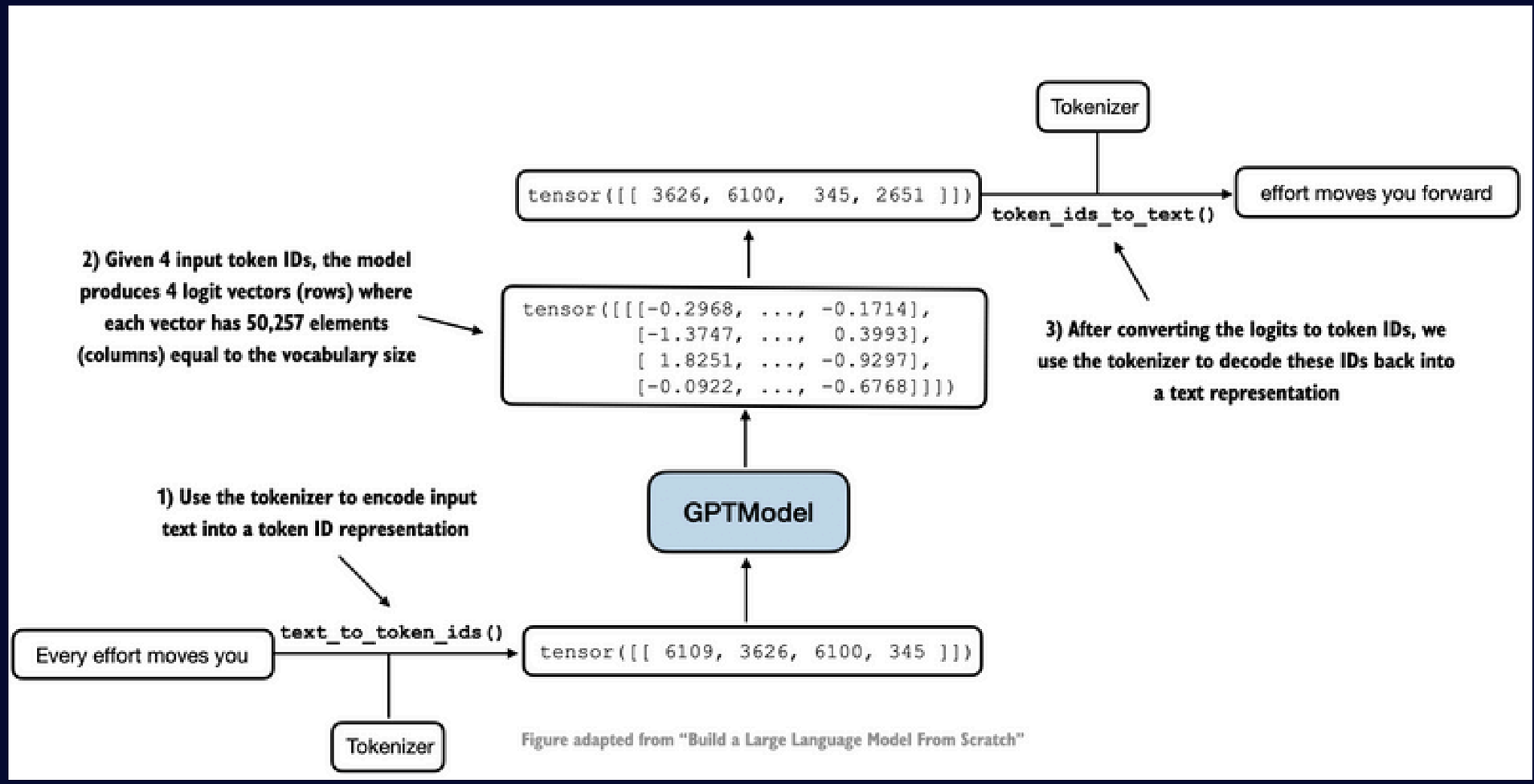


Figure adapted from "Build a Large Language Model From Scratch"

Corpus

Table 1.1 The pretraining dataset of the popular GPT-3 LLM

Dataset name	Dataset description	Number of tokens	Proportion in training data
CommonCrawl (filtered)	Web crawl data	410 billion	60%
WebText2	Web crawl data	19 billion	22%
Books1	Internet-based book corpus	12 billion	8%
Books2	Internet-based book corpus	55 billion	8%
Wikipedia	High-quality text	3 billion	3%

Pre-Training

Preparing the dataset

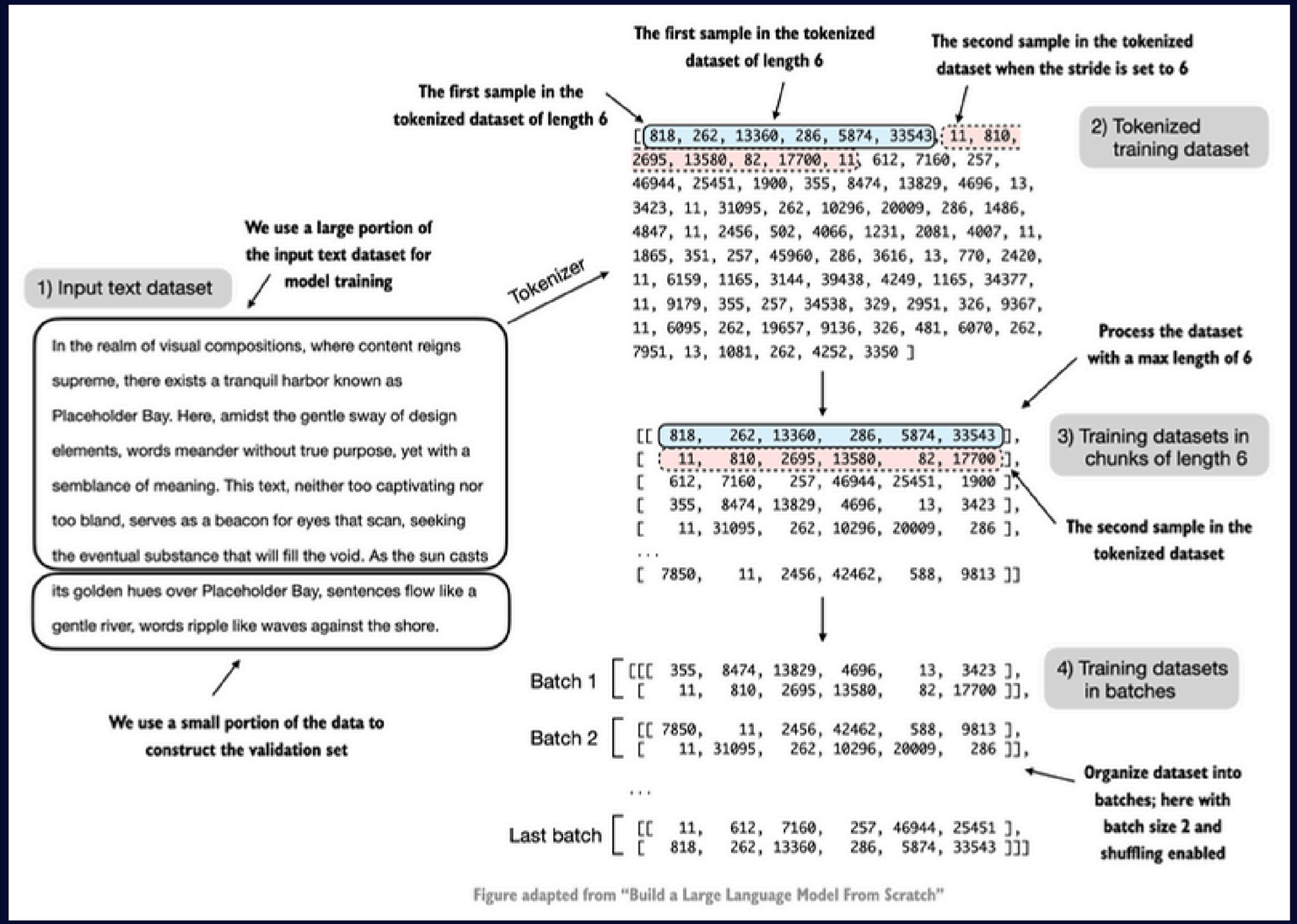


Figure adapted from "Build a Large Language Model From Scratch"

Pre-Training

Training LLM

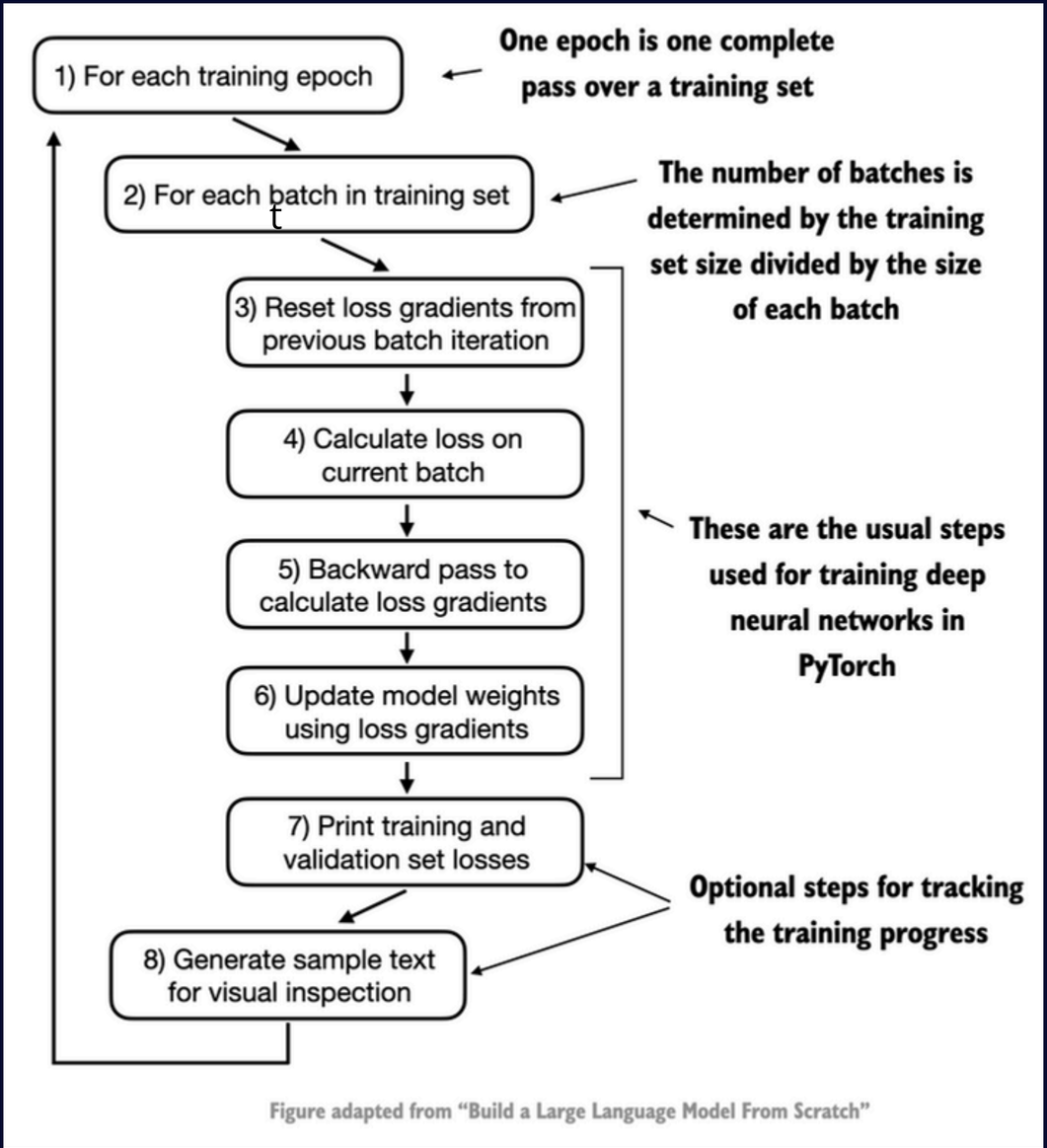


Figure adapted from "Build a Large Language Model From Scratch"

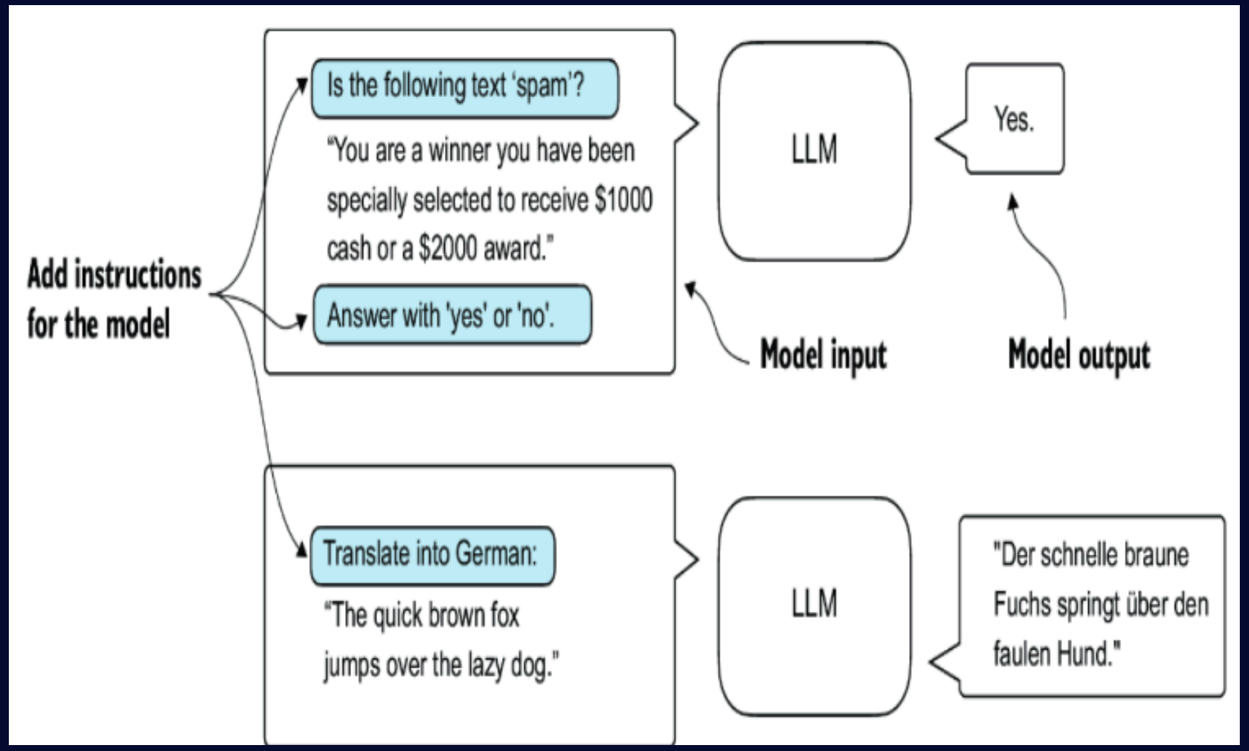
$$L = -\sum_{k=1}^{K} y_k \log(p_k)$$

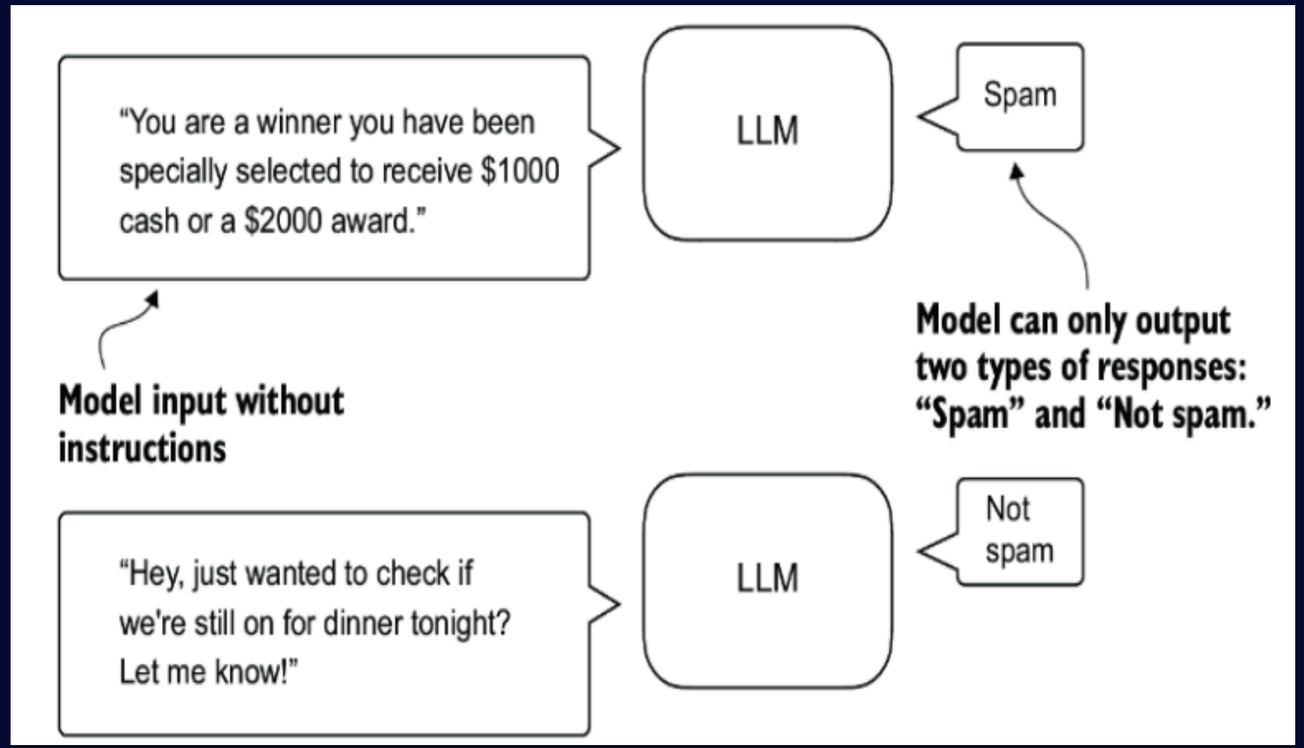
- y_k is the true probability of class k, typically represented as 1 for the correct class and 0 for all other classes.
- p_k is the predicted probability of class k

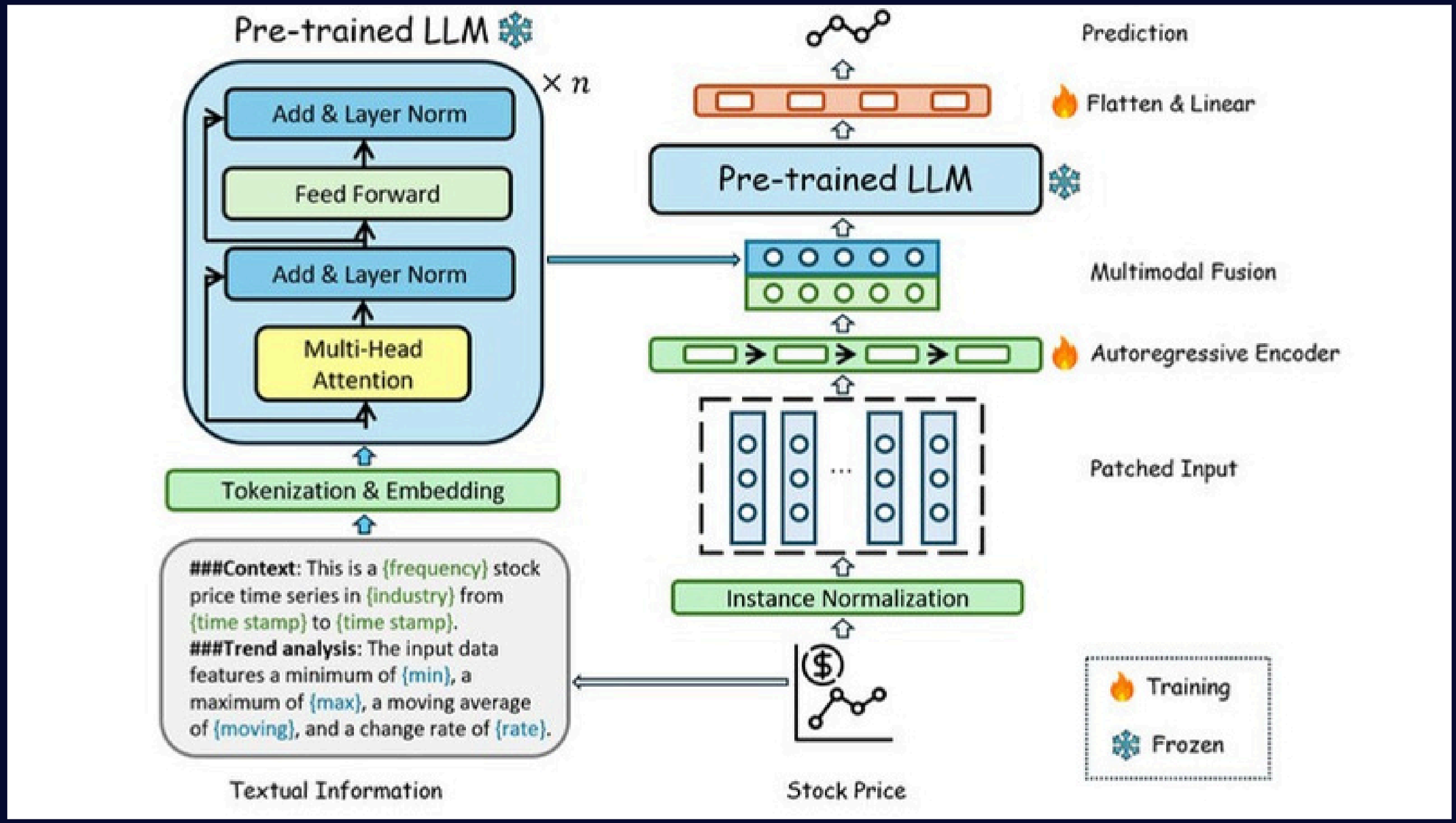
Fine-Tuning

Instruction Fine-Tuning: Instruction fine-tuning involves training a language model on a set of tasks using specific instructions to improve its ability to understand and execute tasks described in natural language prompts.

Classification Fine-Tuning: The model is trained to recognize a specific set of class labels, such as “spam” and “not spam.”






Pre-trained LLM
× n
Add & Layer Norm
Feed Forward
Add & Layer Norm
Multi-Head Attention
Tokenization & Embedding
###Context: This is a {frequency} stock price time series in {industry} from {time stamp} to {time stamp}.
###Trend analysis: The input data features a minimum of {min}, a maximum of {max}, a moving average of {moving}, and a change rate of {rate}.
Textual Information
Prediction
Flatten & Linear
Pre-trained LLM
Multimodal Fusion
Autoregressive Encoder
Patched Input
Instance Normalization
Stock Price
Training
Frozen

Instruction Fine-Tuning

- Trains the model on a set of tasks
- Uses instructions in Natural Language Prompts
- Undertakes a broader range of tasks
- Requires larger dataset and greater computational resources to develop models

Classification Fine-Tuning

- Predicts classes based on the training dataset
- Does not require instructions alongside the input-target pair
- Highly specialized model
- Requires less data and compute power

Fine-Tuning

LoRA (Low-Rank Adaptation)

Efficient Fine-Tuning for Large Language Models:

Background & Motivation

- Large pre-trained models (e.g., GPT, BERT) are powerful but expensive to fine-tune.
- Full fine-tuning requires updating billions of parameters, making it:
  - Costly in terms of compute and storage.
  - Hard to deploy and manage multiple task-specific versions.

Key Idead about LoRa

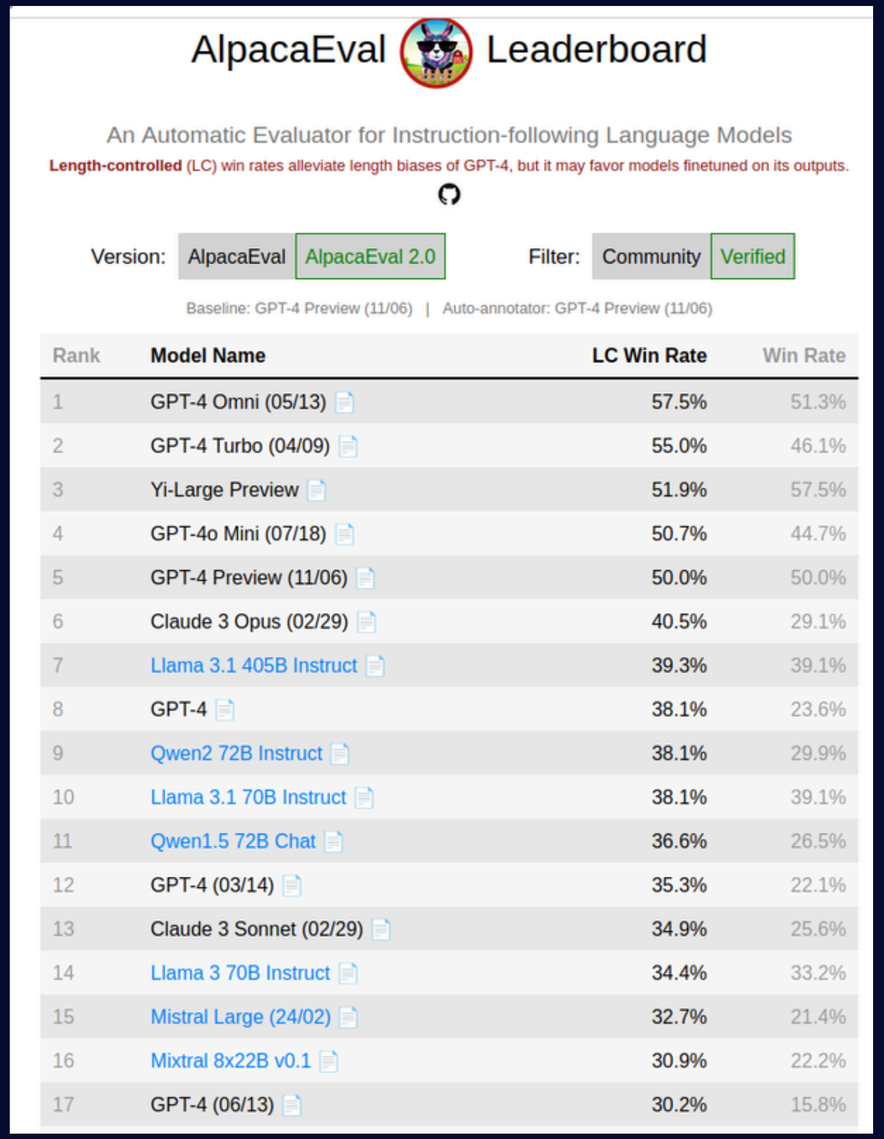
- Instead of updating all weights, LoRA:
- Freezes the pre-trained model weights.
- Adds trainable low-rank matrices (A & B) to specific layers (usually attention layers).
- Only A and B are updated during fine-tuning.

Applications

LLM Benchmark & Comparison

Alpaca Eval

- AlpacaEval
  - Evaluates a model's ability to generate a response to a single instruction or prompt



AlpacaEval Leaderboard

An Automatic Evaluator for Instruction-following Language Models

Length-controlled (LC) win rates alleviate length biases of GPT-4, but it may favor models finetuned on its outputs.

Version: AlpacaEval | AlpacaEval 2.0 Filter: Community | Verified

Baseline: GPT-4 Preview (11/06) | Auto-annotator: GPT-4 Preview (11/06)

Rank	Model Name	LC Win Rate	Win Rate
1	GPT-4 Omni (05/13)	57.5%	51.3%
2	GPT-4 Turbo (04/09)	55.0%	46.1%
3	Yi-Large Preview	51.9%	57.5%
4	GPT-4o Mini (07/18)	50.7%	44.7%
5	GPT-4 Preview (11/06)	50.0%	50.0%
6	Claude 3 Opus (02/29)	40.5%	29.1%
7	Llama 3.1 405B Instruct	39.3%	39.1%
8	GPT-4	38.1%	23.6%
9	Qwen2 72B Instruct	38.1%	29.9%
10	Llama 3.1 70B Instruct	38.1%	39.1%
11	Qwen1.5 72B Chat	36.6%	26.5%
12	GPT-4 (03/14)	35.3%	22.1%
13	Claude 3 Sonnet (02/29)	34.9%	25.6%
14	Llama 3 70B Instruct	34.4%	33.2%
15	Mistral Large (24/02)	32.7%	21.4%
16	Mixtral 8x22B v0.1	30.9%	22.2%
17	GPT-4 (06/13)	30.2%	15.8%

ARC-AGI

- ARC-AGI
  - Evaluates artificial intelligence through abstract reasoning capabilities rather than specific skills or memorization

AI System	Organization	System Type	ARC-AGI-1	ARC-AGI-2	Cost/Task	Code / Paper
Human Panel	Human	N/A	98.0%	100.0%	$17.00	–
Claude Opus 4 (Thinking 16K)	Anthropic	CoT	35.7%	8.6%	$1.93	📄
o3 (High)	OpenAI	CoT	60.8%	6.5%	$0.834	–
o4-mini (High)	OpenAI	CoT	58.7%	6.1%	$0.856	–
Claude Sonnet 4 (Thinking 16K)	Anthropic	CoT	40.0%	5.9%	$0.486	📄
o3-Pro (High)	OpenAI	CoT + Synthesis	59.3%	4.9%	$7.55	📄
Claude Opus 4 (Thinking 8K)	Anthropic	CoT	30.7%	4.5%	$1.16	📄
o3-preview (Low)•	OpenAI	CoT + Synthesis	75.7%	4.0%	$200.00	📄
Gemini 2.5 Pro (Preview)	Google	CoT	33.0%	3.8%	$0.813	📄
Gemini 2.5 Pro (Preview, Thinking 1K)	Google	CoT	31.3%	3.4%	$0.804	📄
o3-mini (High)	OpenAI	CoT	34.5%	3.0%	$0.547	–
o3 (Medium)	OpenAI	CoT	53.8%	3.0%	$0.479	–
Gemini 2.5 Flash (Preview) (Thinking 24K)	Google	CoT	32.3%	2.5%	$0.319	📄 💻
ARChitects	ARC Prize 2024	Custom	56.0%	2.5%	$0.200	📄 💻
o4-mini (Medium)	OpenAI	CoT	41.8%	2.4%	$0.231	–

LMArena

- LMArena
  - Places two unidentified large language models (LLMs) in a battle to see which can best tackle a prompt, with users of the benchmark voting for the output they like most

Model 210 / 210	Overall	Hard Prompts	Coding	Math	Creative Writing	Instruction Following	Longer Query	Multi-Turn
gemini-2.5-pro-prev…	1	1	1	1	1	1	1	1
o3-2025-04-16	2	2	2	1	3	3	6	4
gemini-2.5-pro-prev…	2	2	4	1	1	1	1	2
chatgpt-4o-latest-2…	3	2	2	10	2	3	2	1
gpt-4.5-preview-202…	4	4	2	5	2	3	2	1
claude-opus-4-20250…	6	2	2	5	2	3	2	3
gemini-2.5-flash-pr…	6	5	7	2	3	3	2	6
deepseek-r1-0528	6	4	2	5	4	10	9	6
gpt-4.1-2025-04-14	6	6	5	14	4	7	3	4
grok-3-preview-02-24	8	7	6	13	7	8	5	8
gemini-2.5-flash-pr…	8	7	14	5	5	3	7	8
qwen3-235b-a22b-no-…	10	7	5	4	11	12	7	4
deepseek-v3-0324	11	10	5	15	5	12	8	4
o4-mini-2025-04-16	11	10	7	3	18	15	17	9
o1-2024-12-17	12	10	11	6	14	8	9	16
claude-sonnet-4-202…	12	7	5	7	4	7	5	6

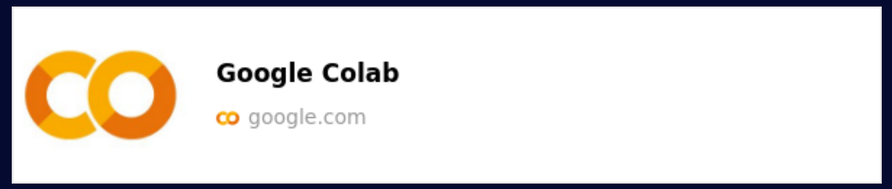
MMLU

- MMLU
  - Evaluate language understanding models across broader and more challenging tasks

Models	Model Size(B)	Data Source	Overall	Biology	Business	Chemistry	Computer Science	Economics	Engin
GPT-o1	unknown	Self-Reported	0.893	-	-	-	-	-	-
Hunyuan-T1	unknown	Self-Reported	0.872	-	-	-	-	-	-
GPT-4.5	unknown	Self-Reported	0.861	-	-	-	-	-	-
Gemini-2.5-Pro-Exp-03-25	unknown	TIGER-Lab	0.8452	0.9299	0.893	0.8741	0.8556	0.8829	0.735
DeepSeek-R1	671	Self-Reported	0.84	-	-	-	-	-	-
Claude-3.7-Sonnet-Thinking	unknown	Self-Reported	0.84	-	-	-	-	-	-
GPT-4-mini (high)	unknown	Self-Reported	0.83	-	-	-	-	-	-
Grok-3-mini	unknown	Self-Reported	0.83	-	-	-	-	-	-
Llama4-Behemoth	2000	Self-Reported	0.828	-	-	-	-	-	-
Deepseek-V3-0324	671	Self-Reported	0.813	-	-	-	-	-	-
Llama4-Maverick	400	Self-Reported	0.805	-	-	-	-	-	-
GPT-o1-mini	unknown	Self-Reported	0.803	-	-	-	-	-	-
Doubao-1.5-Pro	unknown	Self-Reported	0.801	-	-	-	-	-	-
Grok3-Beta	unknown	Self-Reported	0.799	-	-	-	-	-	-
GPT-o3-mini	unknown	Self-Reported	0.794	-	-	-	-	-	-
Gemini-2.0-Pro	unknown	Self-Reported	0.791	-	-	-	-	-	-
HunyuanTurboS	unknown	Self-Reported	0.79	-	-	-	-	-	-
Grok3-mini-Beta	unknown	Self-Reported	0.789	-	-	-	-	-	-
Claude-3.5-Sonnet (2024-10-22)	unknown	Self-Reported	0.78	-	-	-	-	-	-
GPT-4o (2024-11-20)	unknown	Self-Reported	0.779	-	-	-	-	-	-
Claude-3.5-Sonnet (2024-10-22)	unknown	TIGER-LAb	0.7764	0.8856	0.8137	0.7853	0.8244	0.859	0.613
Gemini-2.0-Flash	unknown	Self-Reported	0.776	-	-	-	-	-	-

Demonstration

- Colab Notebook link for transformer task



- Google Drive link for Fine-tuning task

  https://drive.google.com/drive/folders/17ffggVa5xQDb6MrkXdhWYmm-JXDsn4td?usp=drive_link

References

- https://jalammar.github.io/illustrated-transformer/
- https://huggingface.co/spaces/lmarena-ai/chatbot-arena-leaderboard
- https://lmarena.ai/leaderboard?utm_campaign=hf_banner
- https://tatsu-lab.github.io/alpaca_eval/
- https://www.youtube.com/watch?v=kCc8FmEb1nY
- https://huggingface.co/spaces/TIGER-Lab/MMLU-Pro
- https://arcprize.org/arc-agi
- https://analyticsindiamag.com/ai-features/the-environmental-impact-of-llms/
- https://apxml.com/posts/system-requirements-deepseek-models
- https://huggingface.co/blog/llama31

References

- https://www.researchgate.net/figure/The-StockTime-framework-operates-as-follows-1-Stock-correlations-statistical-trends_fig2_384057825
- https://arxiv.org/abs/1409.0473
- https://arxiv.org/abs/1706.03762
- https://arxiv.org/abs/2106.09685

Thank you!