Since 2004

UET

ĐẠI HỌC CÔNG NGHỆ, ĐHQGHN

VNU-University of Engineering and Technology

Since 1906

VNU

ĐẠI HỌC QUỐC GIA HÀ NỘI

Vietnam National University, Hanoi

INT3121 – Topics in Computer Science

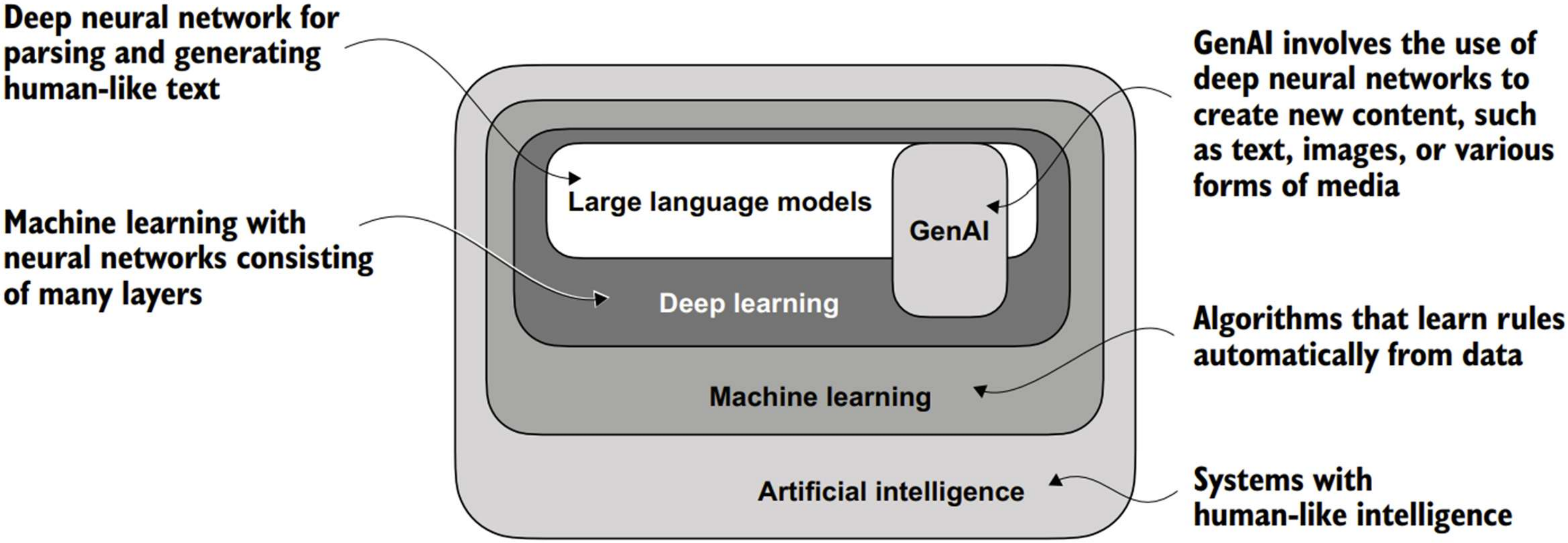
Lecture 1: Foundation of LLMs

Hanoi, 09/2025

Course Introduction

- Introduction to Large Language Models (LLMs)
- Word embedding
- Tokenization
- Self-attention
- Muli-headed self-attention
- Vision Transformer

From AI to Large Language Models



Deep neural network for parsing and generating human-like text

Machine learning with neural networks consisting of many layers

GenAI involves the use of deep neural networks to create new content, such as text, images, or various forms of media

Algorithms that learn rules automatically from data

Systems with human-like intelligence

Source: BLLM

Source: COS 597G

Application of LLMs

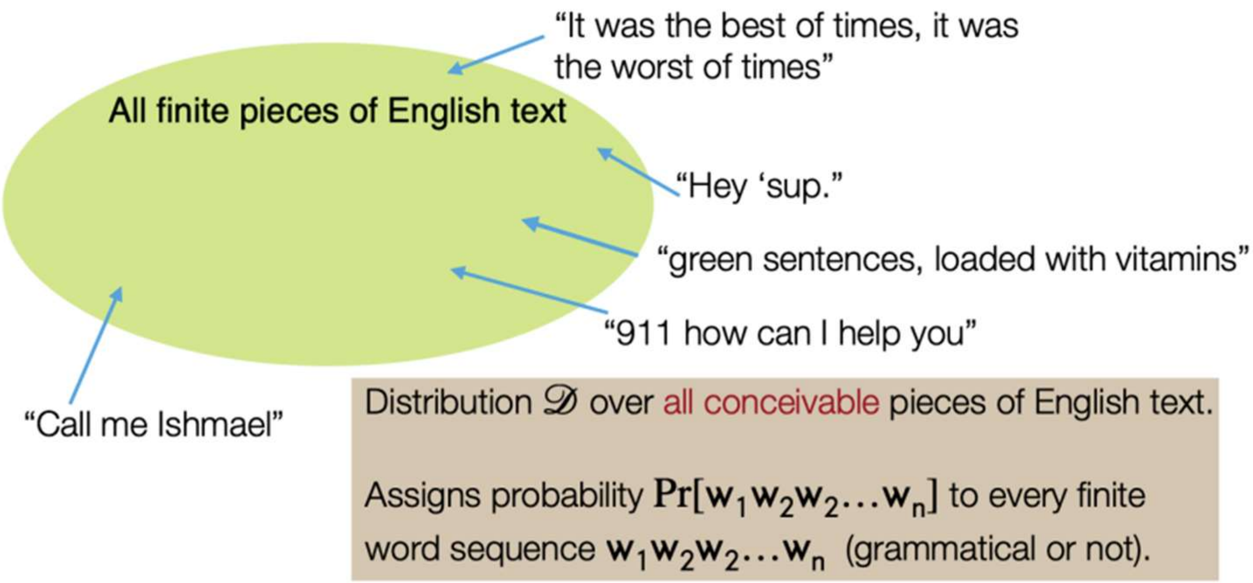
- Chatbot (ChatGPT, Gemini,…)
- Machine translation
- Generation of novel texts
- Sentiment analysis
- Text summarization
- Language-instructed image generation
- Domain-specific LLM-based applications
  - Code generation (Copilot)
  - Task/project magagement (Notion)
  - Multimedia generation/editing (Canva)

Source: BLLM

Source: COS 597G

Language Models: Narrow Sense

- A probabilistic model that assigns a probability $P[w_1, w_2, \ldots, w_n]$ to every finite sequence $w_1, \ldots, w_n$ (grammatical or not)

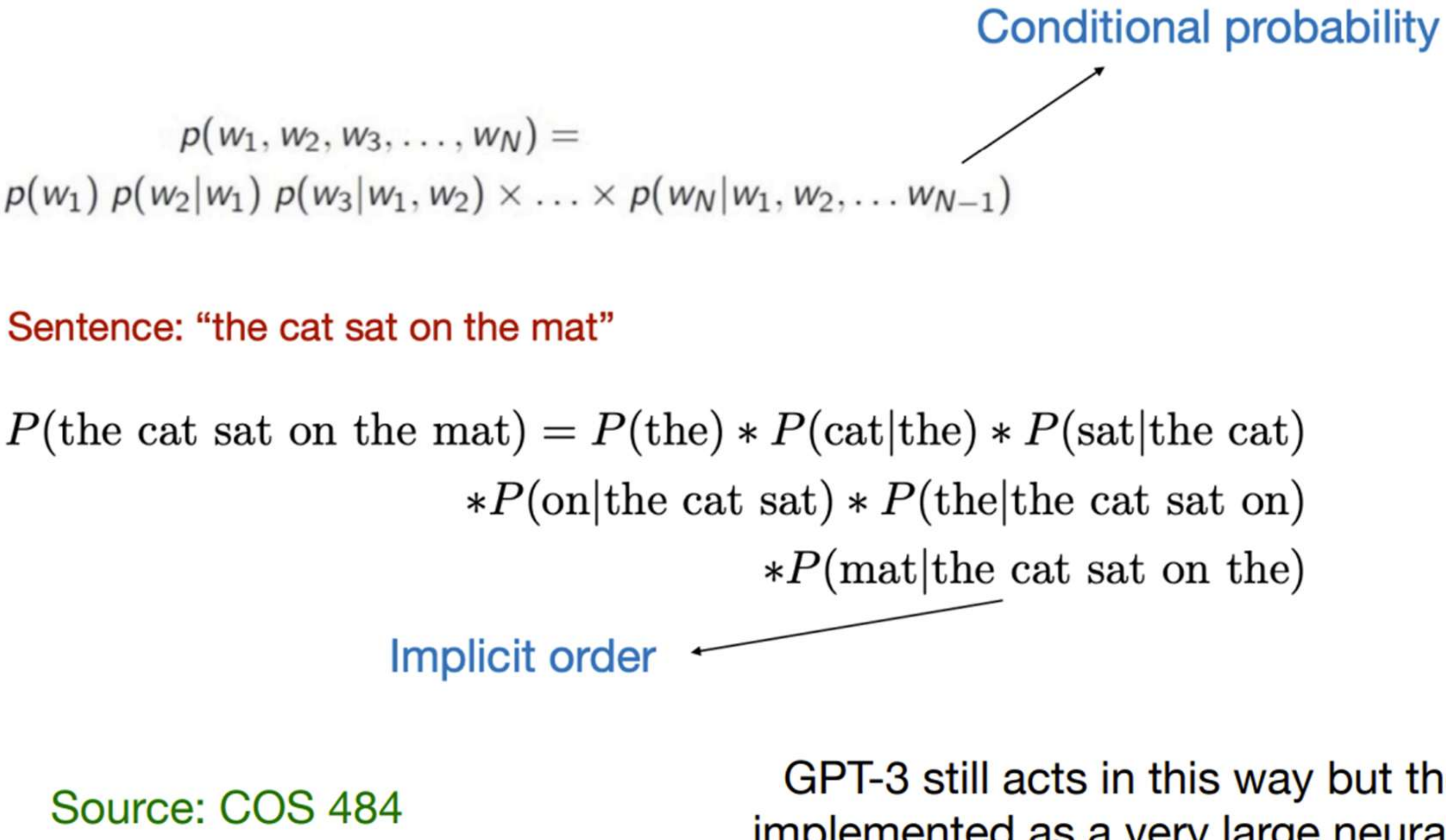

All finite pieces of English text
"It was the best of times, it was the worst of times"
"Hey 'sup."
"green sentences, loaded with vitamins"
"911 how can I help you"
"Call me Ishmael"

Distribution $\mathscr{D}$ over all conceivable pieces of English text.

Assigns probability $\Pr[w_1 w_2 w_2 \ldots w_n]$ to every finite word sequence $w_1 w_2 w_2 \ldots w_n$ (grammatical or not).

Source: COS 324

Language Models: Narrow Sense



Conditional probability

$$p(w_1, w_2, w_3, \ldots, w_N) =$$
$$p(w_1)\ p(w_2|w_1)\ p(w_3|w_1, w_2) \times \ldots \times p(w_N|w_1, w_2, \ldots w_{N-1})$$

Sentence: “the cat sat on the mat”

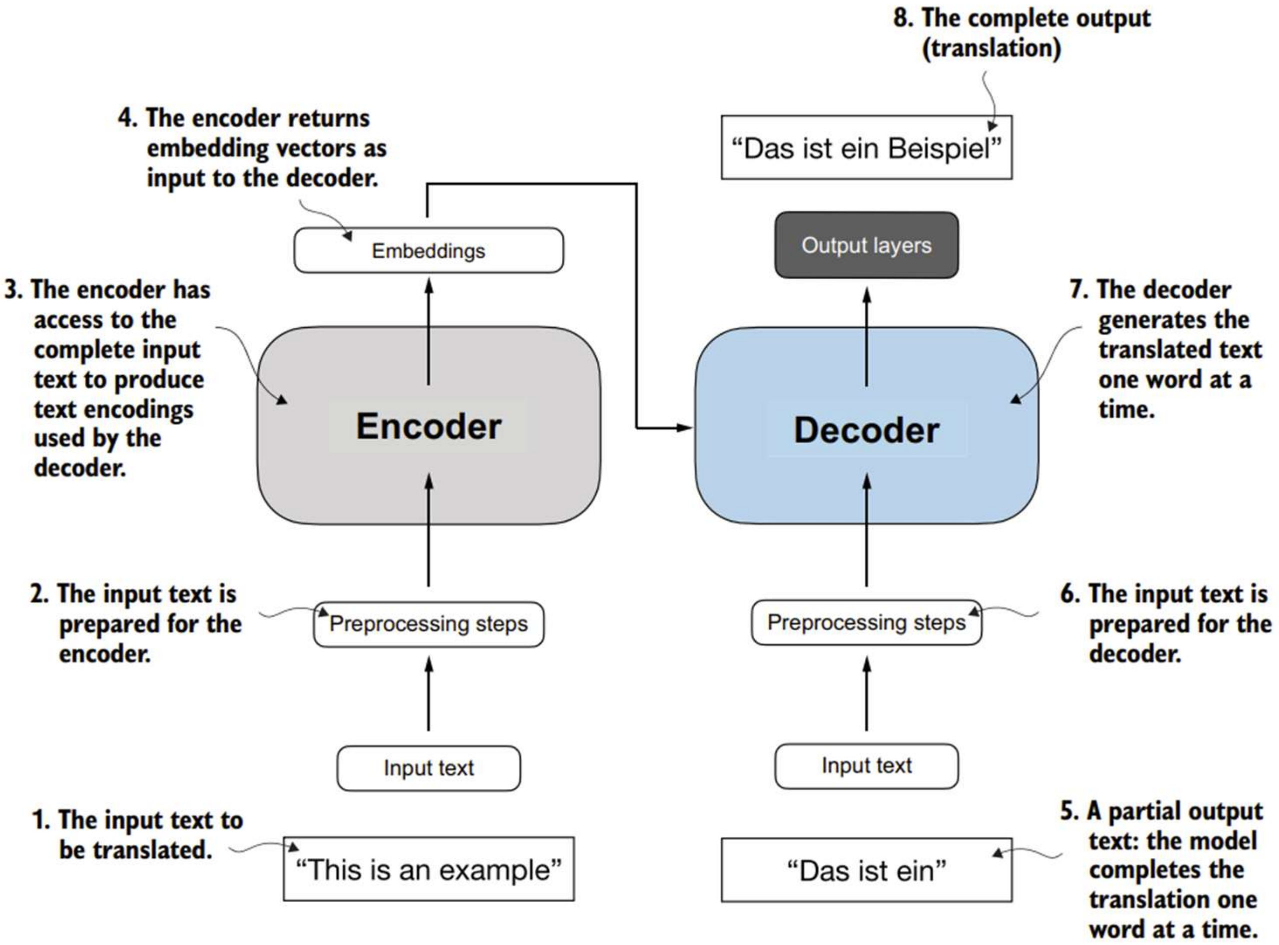
$$\begin{aligned} P(\text{the cat sat on the mat}) = P(\text{the}) * P(\text{cat}|\text{the}) * P(\text{sat}|\text{the cat}) \\ *P(\text{on}|\text{the cat sat}) * P(\text{the}|\text{the cat sat on}) \\ *P(\text{mat}|\text{the cat sat on the}) \end{aligned}$$

Implicit order

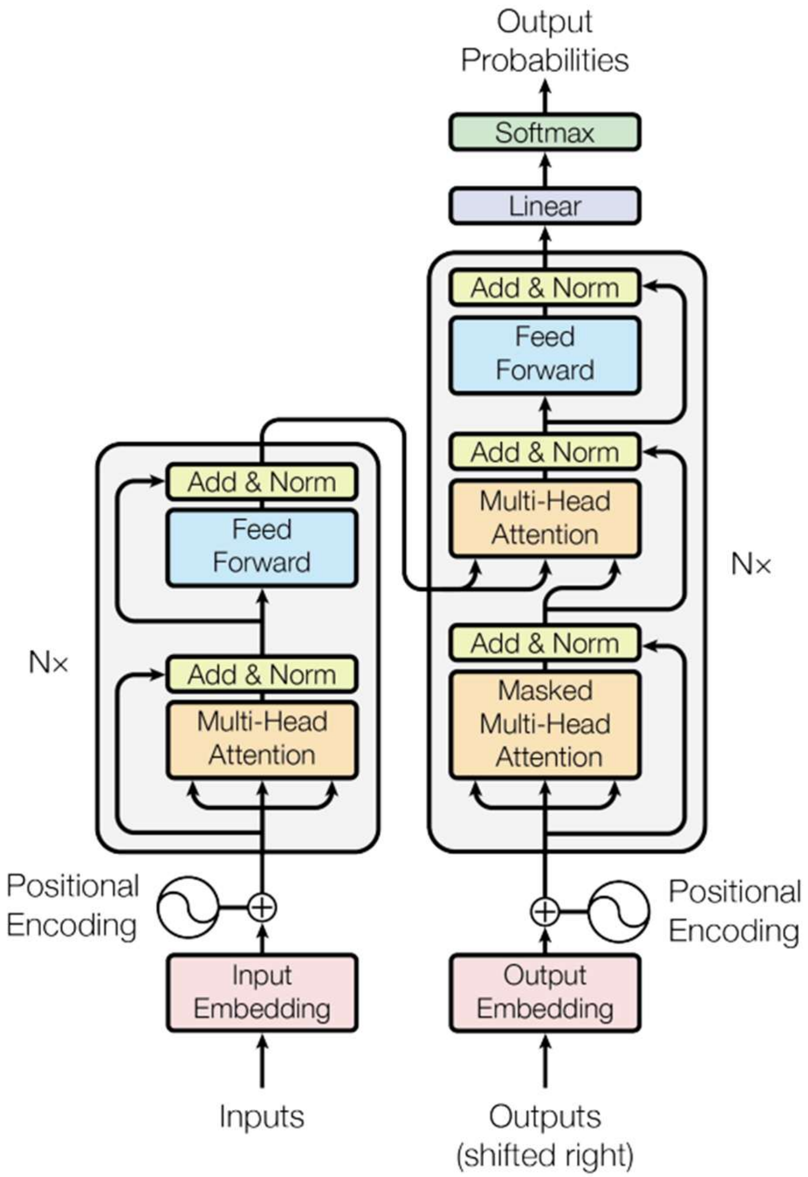
Source: COS 484

GPT-3 still acts in this way but the model is implemented as a very large neural network of 175-billion parameters!

The Transformer Architecture: Overview

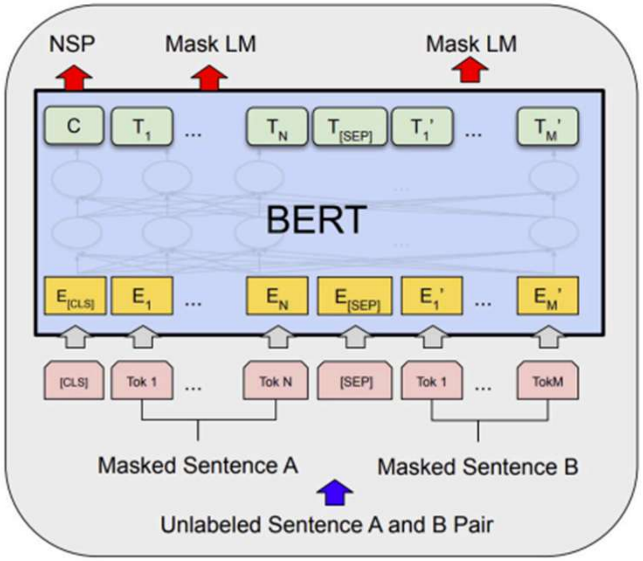

8. The complete output (translation)
"Das ist ein Beispiel"
4. The encoder returns embedding vectors as input to the decoder.
Embeddings
Output layers
3. The encoder has access to the complete input text to produce text encodings used by the decoder.
Encoder
Decoder
7. The decoder generates the translated text one word at a time.
2. The input text is prepared for the encoder.
Preprocessing steps
Preprocessing steps
6. The input text is prepared for the decoder.
Input text
Input text
1. The input text to be translated.
"This is an example"
"Das ist ein"
5. A partial output text: the model completes the translation one word at a time.

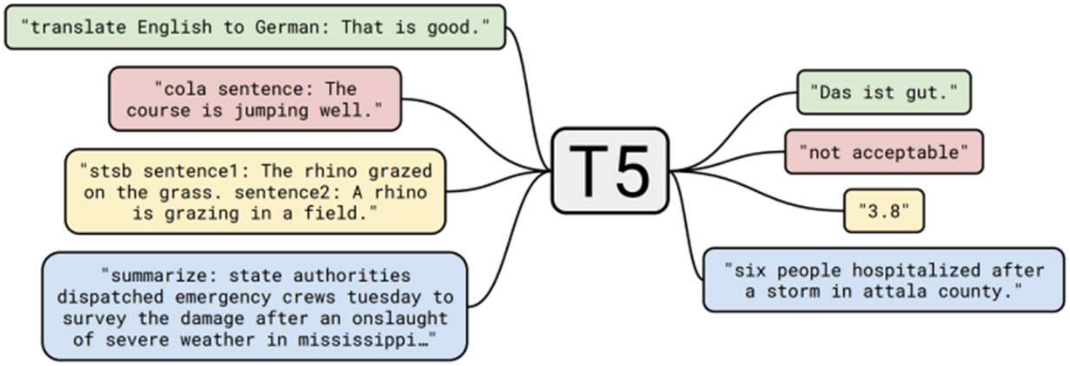
The Transformer Architecture: Detail



Language Models: Broad Sense

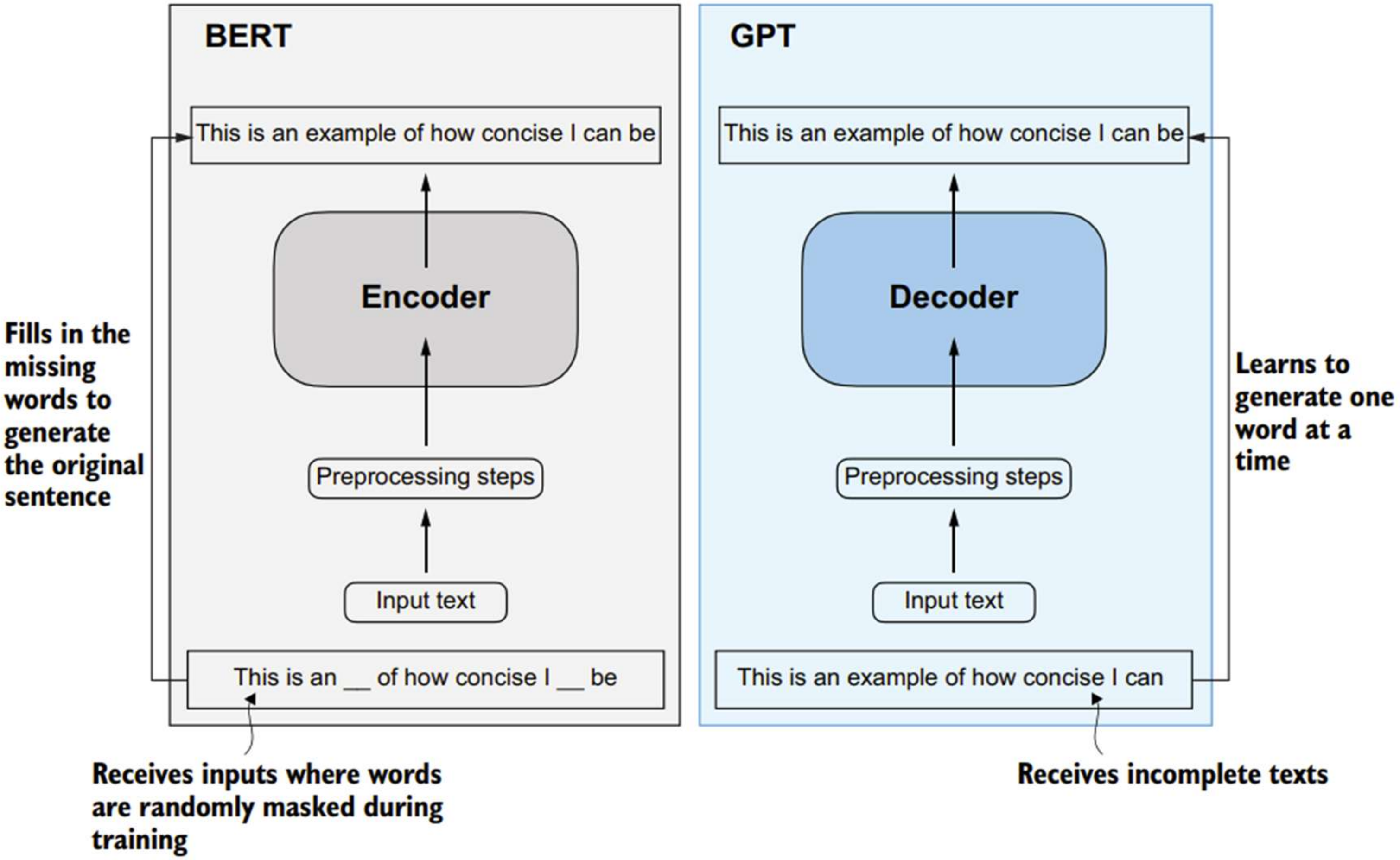
- Decoder-only models (GPT-x models)
- Encoder-only models (BERT, RoBERTa, ELECTRA)
- Encoder-decoder models (T5, BART)




"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

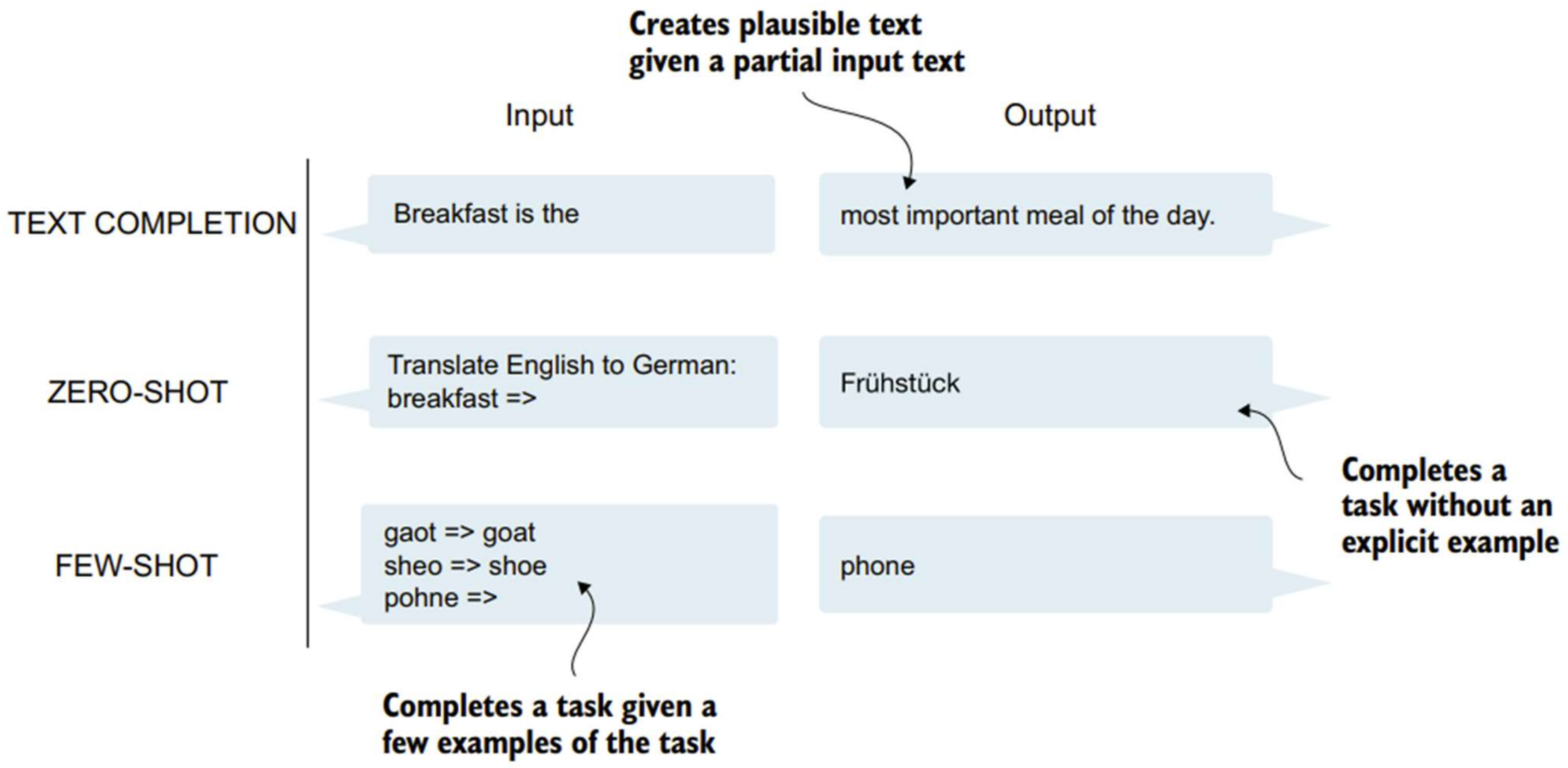
Source: COS 597G

GPT vs BERT



BERT
This is an example of how concise I can be
Encoder
Preprocessing steps
Input text
This is an __ of how concise I __ be
Fills in the missing words to generate the original sentence
Receives inputs where words are randomly masked during training
GPT
This is an example of how concise I can be
Decoder
Preprocessing steps
Input text
This is an example of how concise I can
Learns to generate one word at a time
Receives incomplete texts

GPT-based Model Capabilities



Creates plausible text given a partial input text

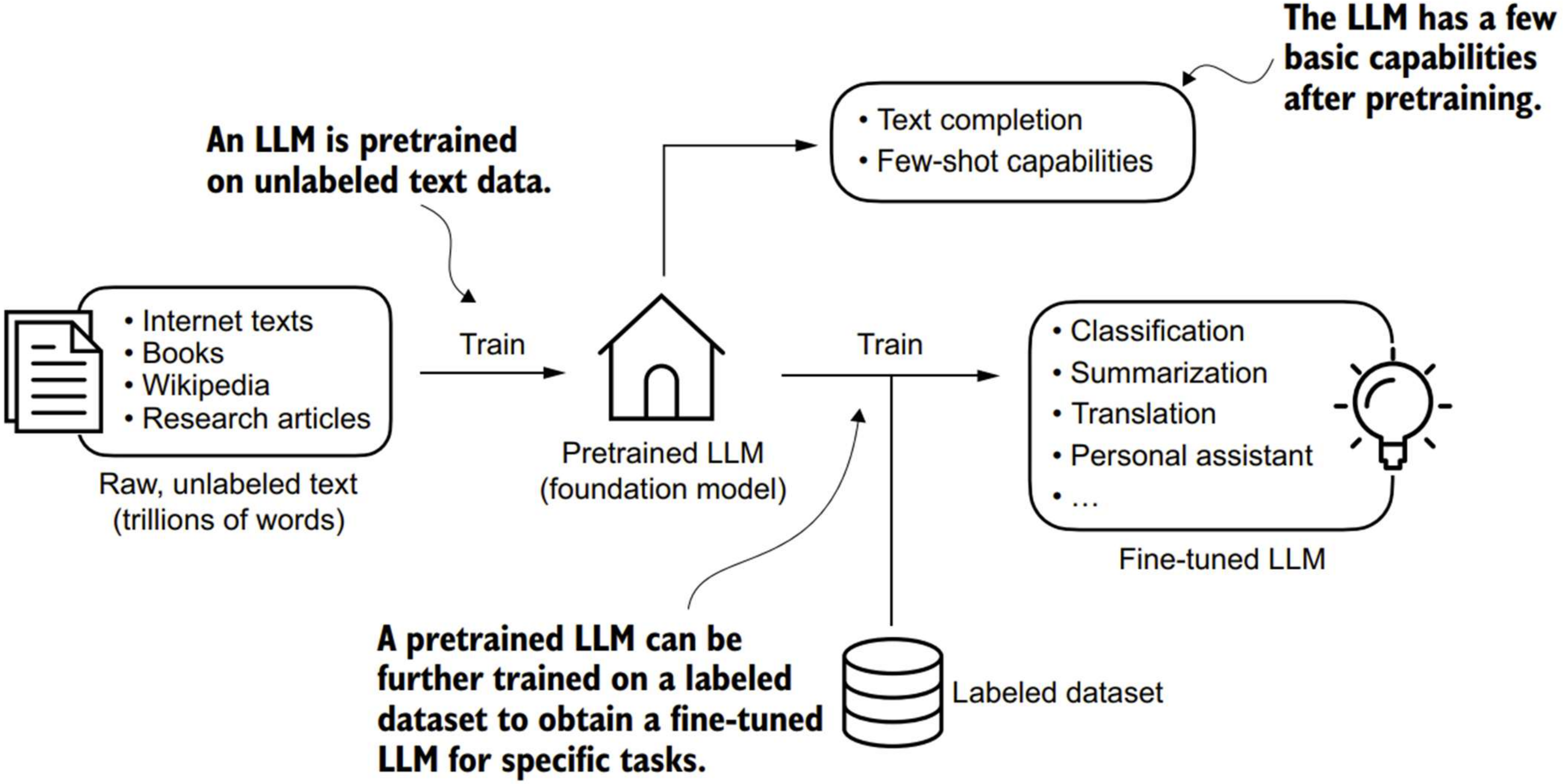
	Input	Output
TEXT COMPLETION	Breakfast is the	most important meal of the day.
ZERO-SHOT	Translate English to German: breakfast =>	Frühstück

FEW-SHOT	gaot => goat
	sheo => shoe
	pohne =>
		phone

Completes a task without an explicit example

Completes a task given a few examples of the task

Stages of Building and Using LLMs



The LLM has a few basic capabilities after pretraining.
An LLM is pretrained on unlabeled text data.
• Text completion
• Few-shot capabilities
• Internet texts
• Books
• Wikipedia
• Research articles
Raw, unlabeled text (trillions of words)
Train
Pretrained LLM (foundation model)
Train
• Classification
• Summarization
• Translation
• Personal assistant
• …
Fine-tuned LLM
A pretrained LLM can be further trained on a labeled dataset to obtain a fine-tuned LLM for specific tasks.
Labeled dataset

GPT 3 Training Dataset

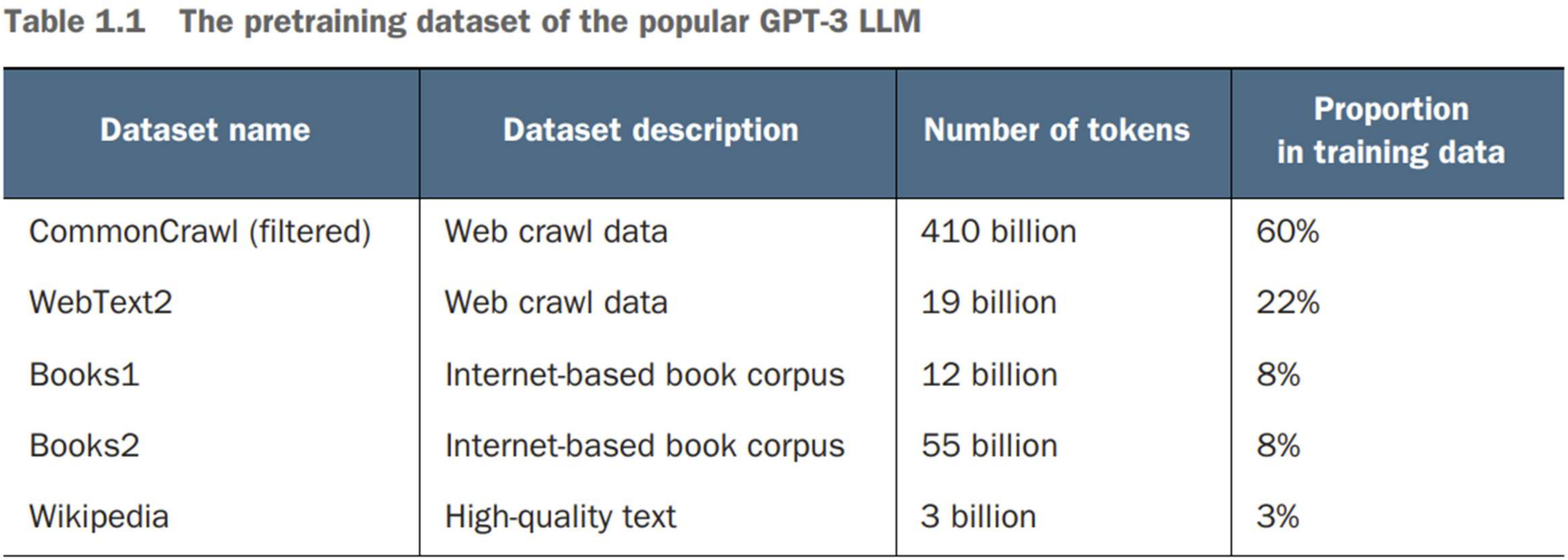
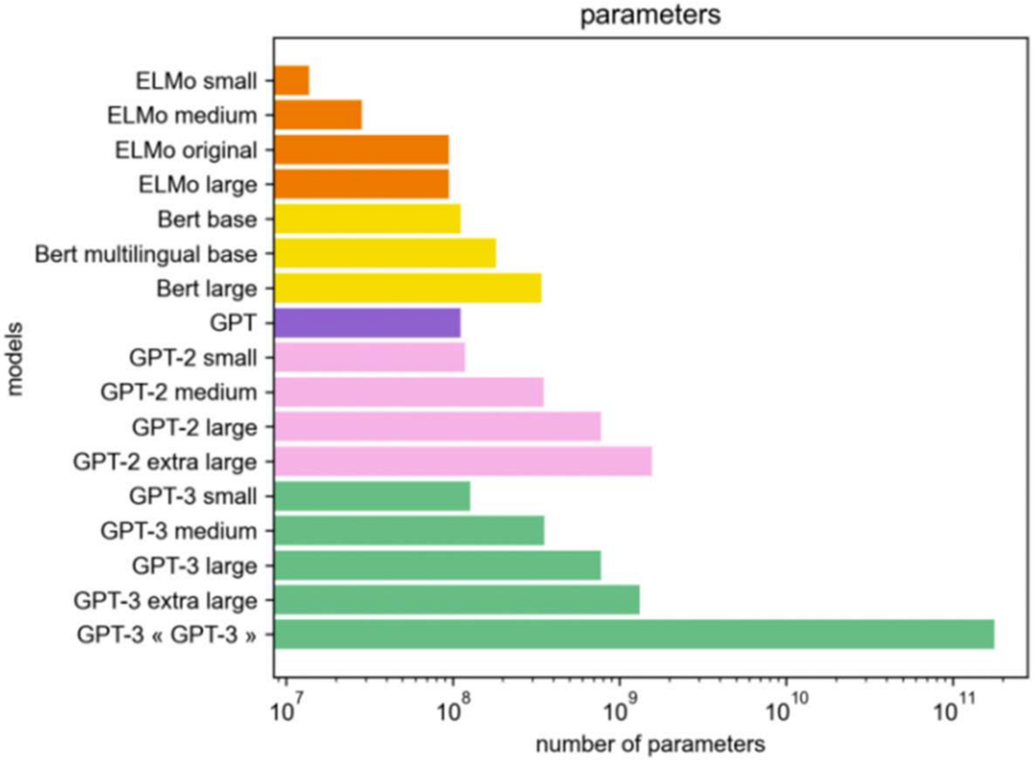


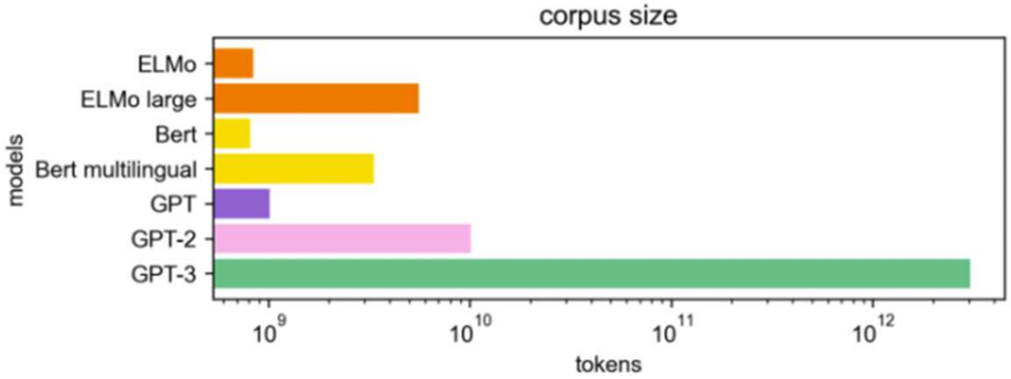
Table 1.1 The pretraining dataset of the popular GPT-3 LLM

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%

GPT-3 has 175 billion parameters

LLMs Comparison

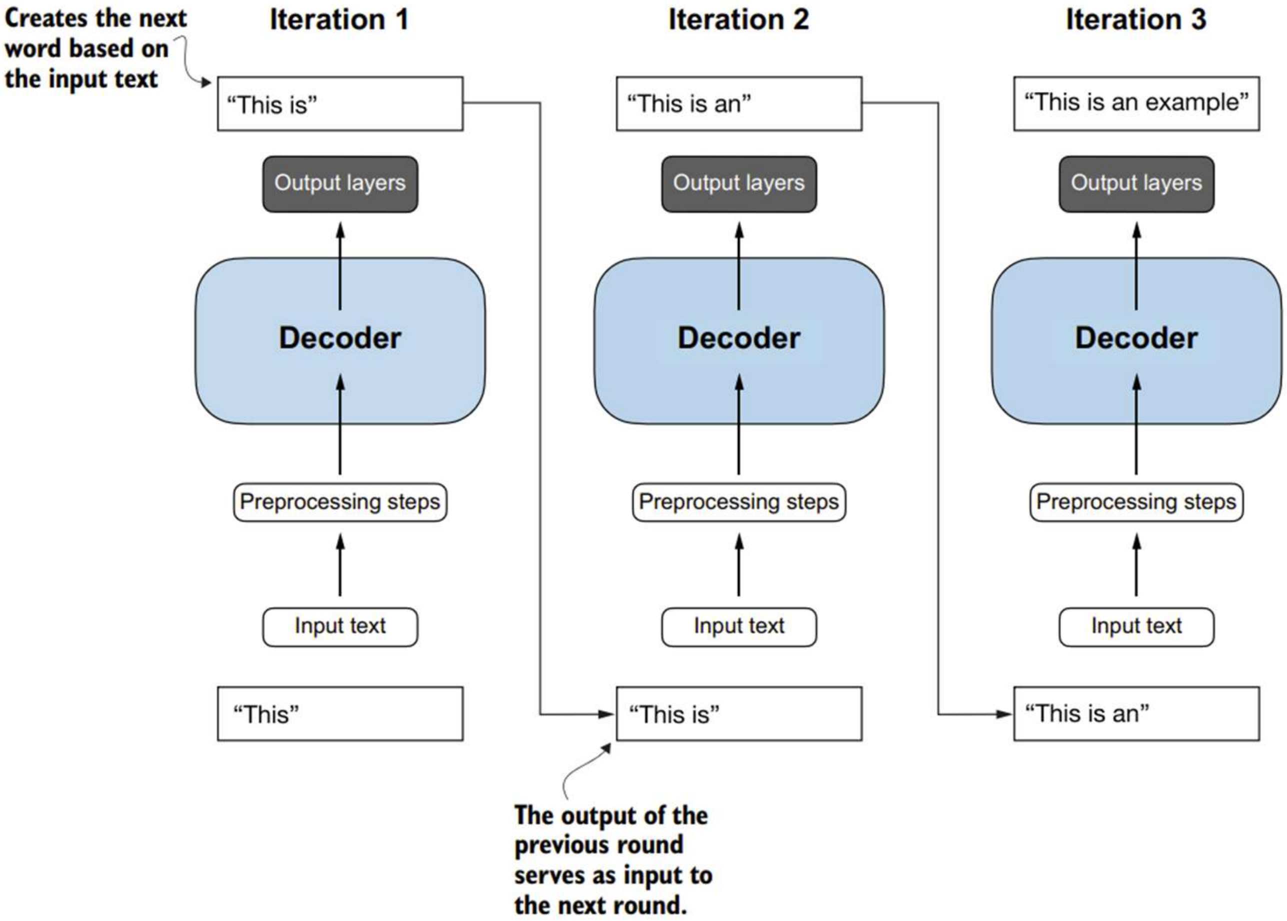




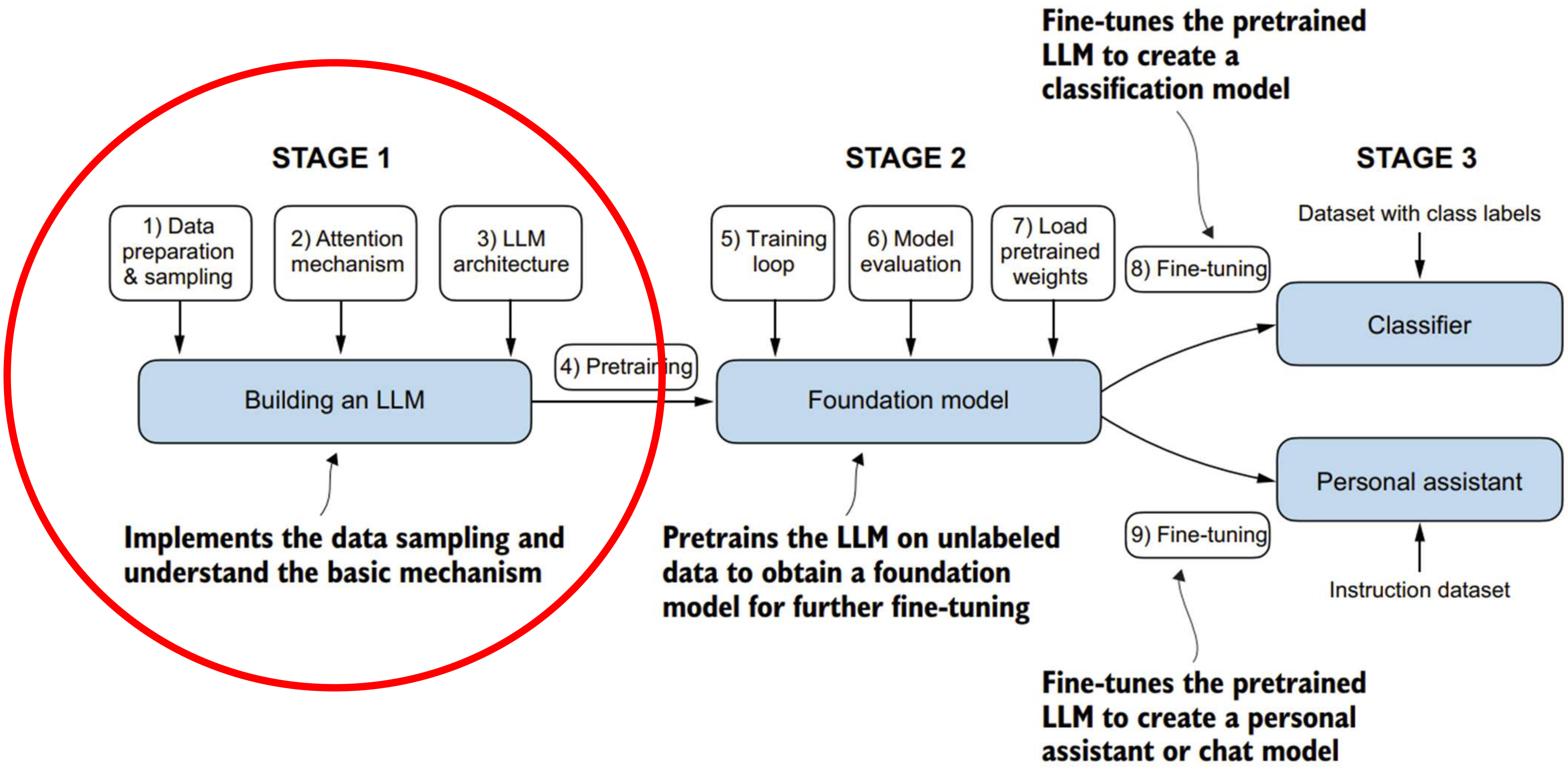
More recent models: PaLM (540B), OPT (175B), BLOOM (176B)…

Image source: https://hellofuture.orange.com/en/the-gpt-3-language-model-revolution-or-evolution/

A Closer Look at the GPT Architecture

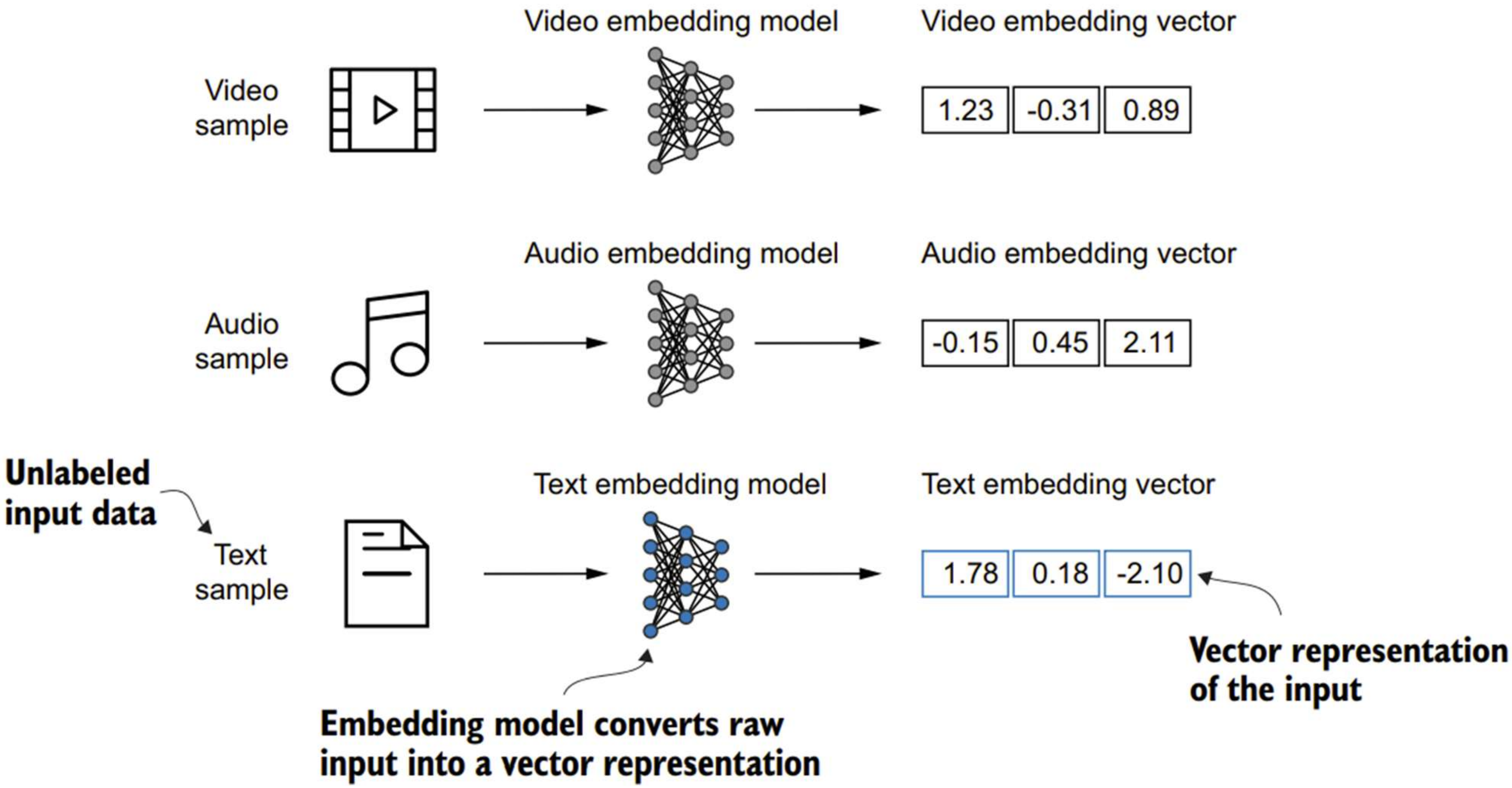


Implementing LLMs



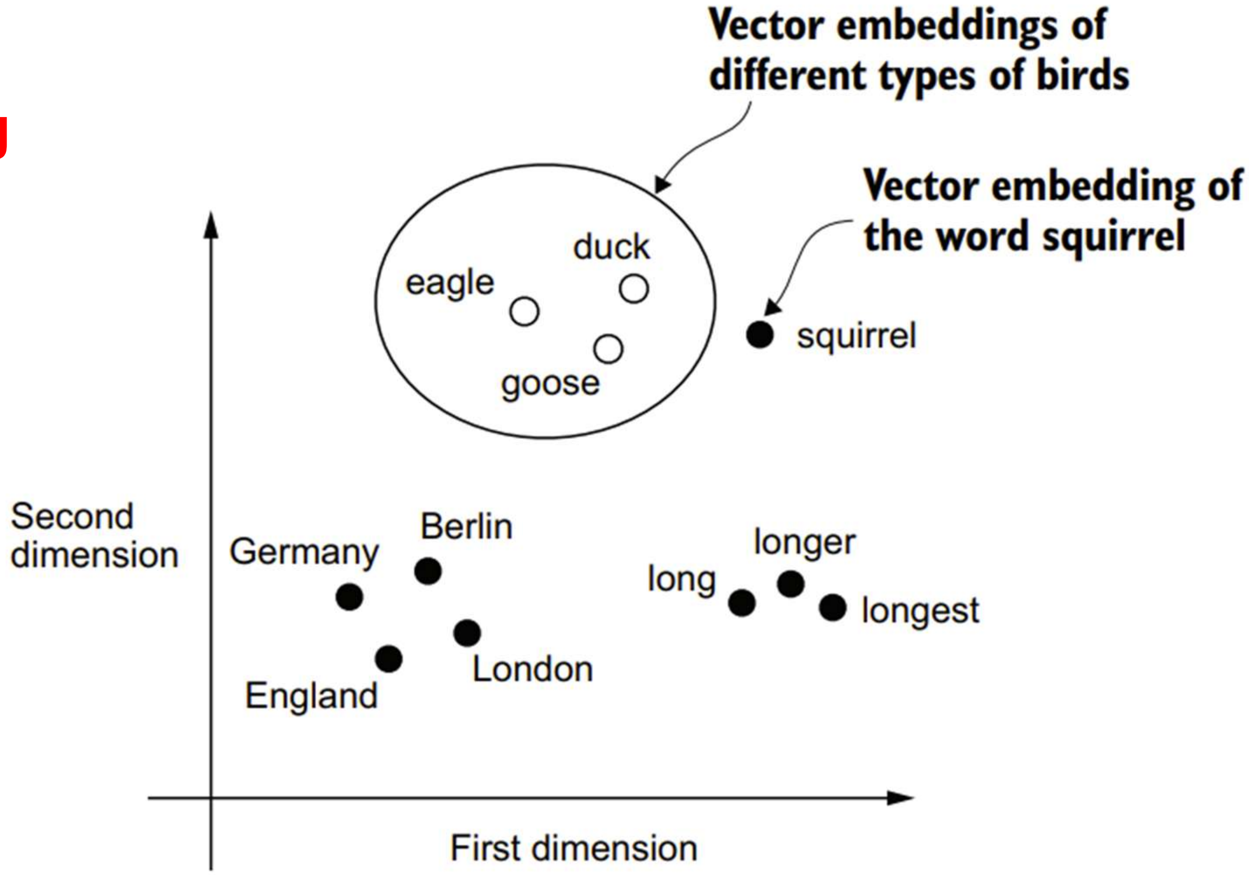
Fine-tunes the pretrained LLM to create a classification model
STAGE 1
STAGE 2
STAGE 3
1) Data preparation & sampling
2) Attention mechanism
3) LLM architecture
5) Training loop
6) Model evaluation
7) Load pretrained weights
8) Fine-tuning
Dataset with class labels
Classifier
Building an LLM
4) Pretraining
Foundation model
Personal assistant
9) Fine-tuning
Instruction dataset
Implements the data sampling and understand the basic mechanism
Pretrains the LLM on unlabeled data to obtain a foundation model for further fine-tuning
Fine-tunes the pretrained LLM to create a personal assistant or chat model

Word embedding – Modelling word in computer



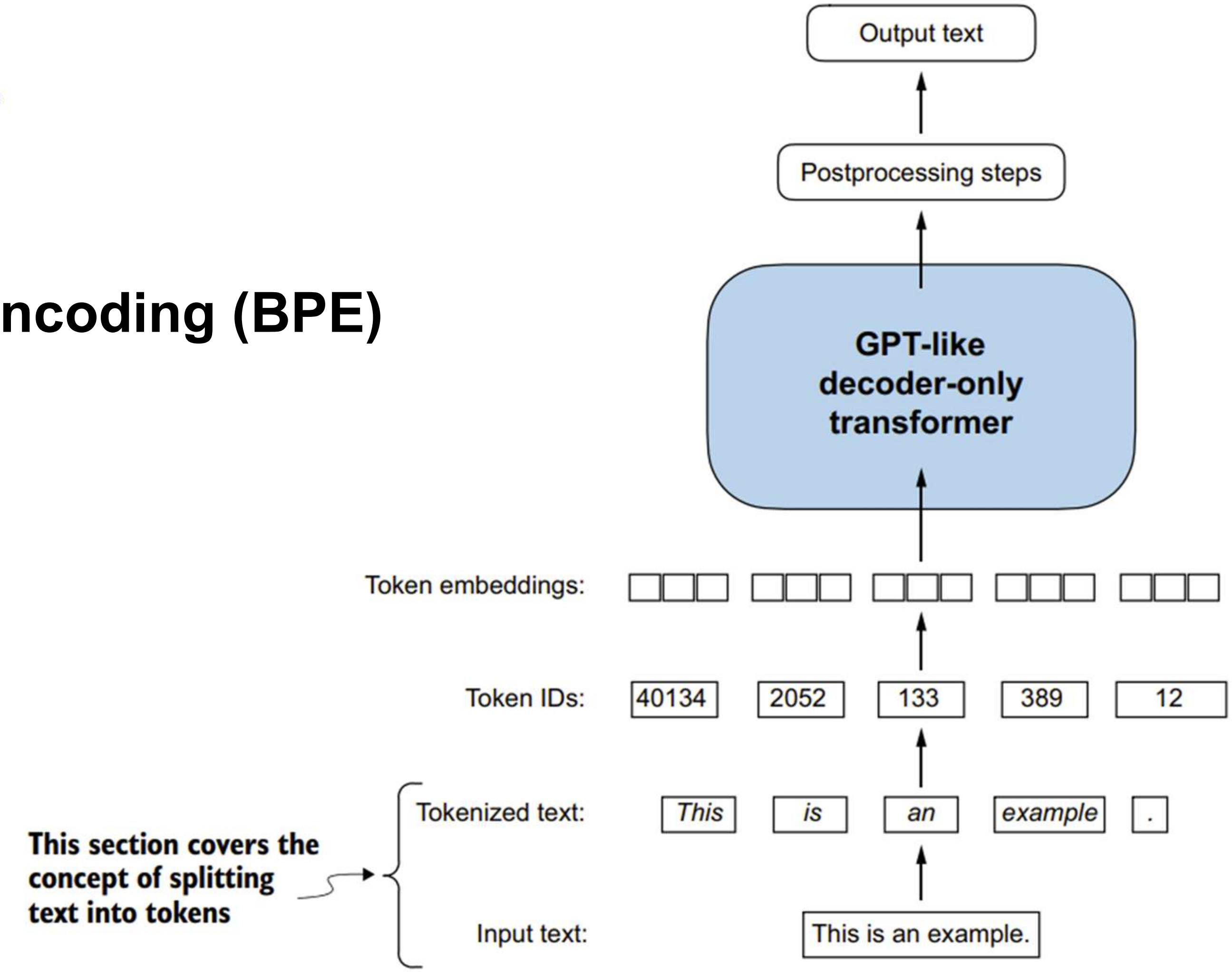
Word embedding – Modelling word in computer

Bringing words that have closer meaning closer in the embedding space



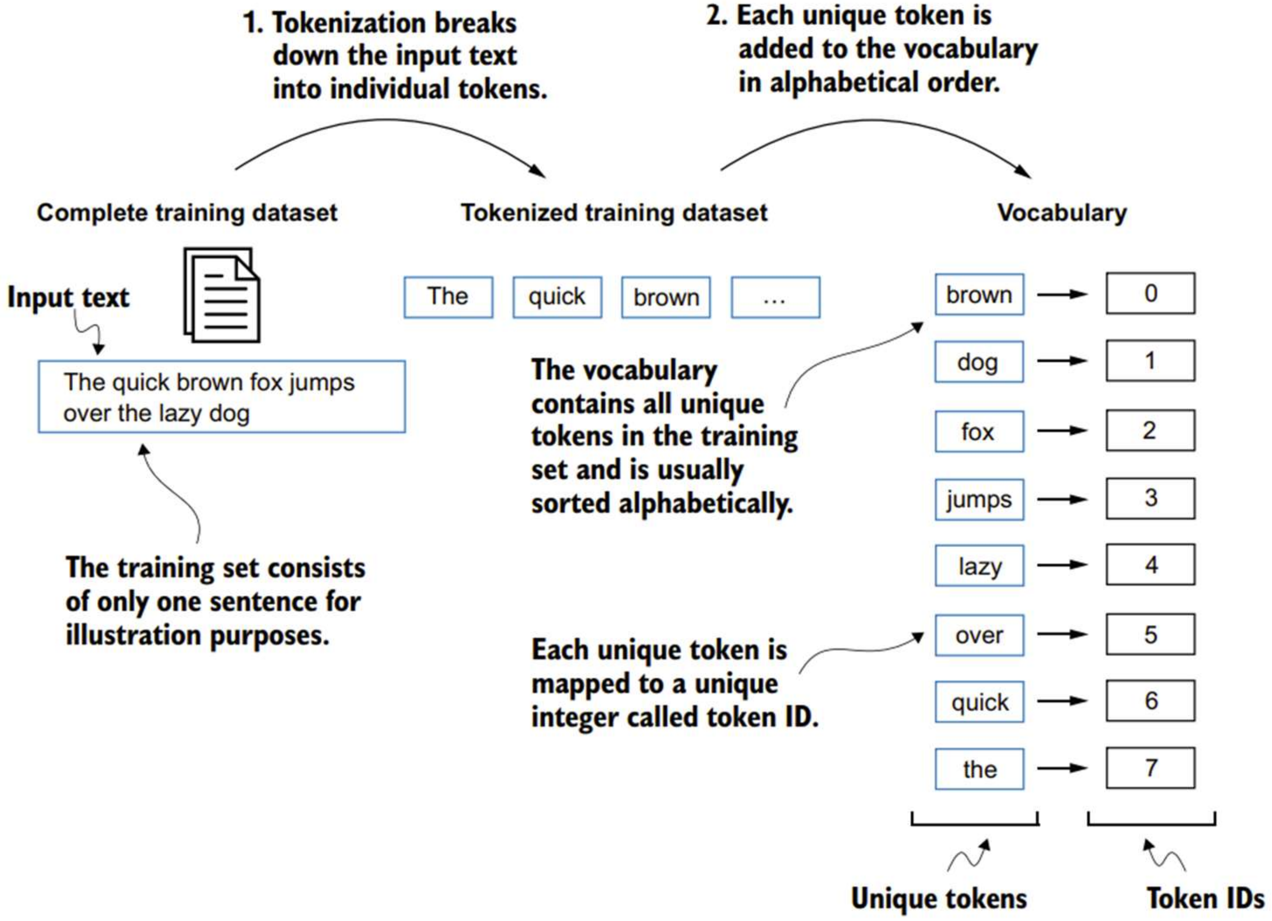
Tokenization

ChatGPT: Byte Pair Encoding (BPE)



This section covers the concept of splitting text into tokens

Tokenization in action


Complete training dataset
Tokenized training dataset
Vocabulary
Input text
The quick brown fox jumps over the lazy dog
The
quick
brown
...
Unique tokens
Token IDs

1. Tokenization breaks down the input text into individual tokens.

2. Each unique token is added to the vocabulary in alphabetical order.

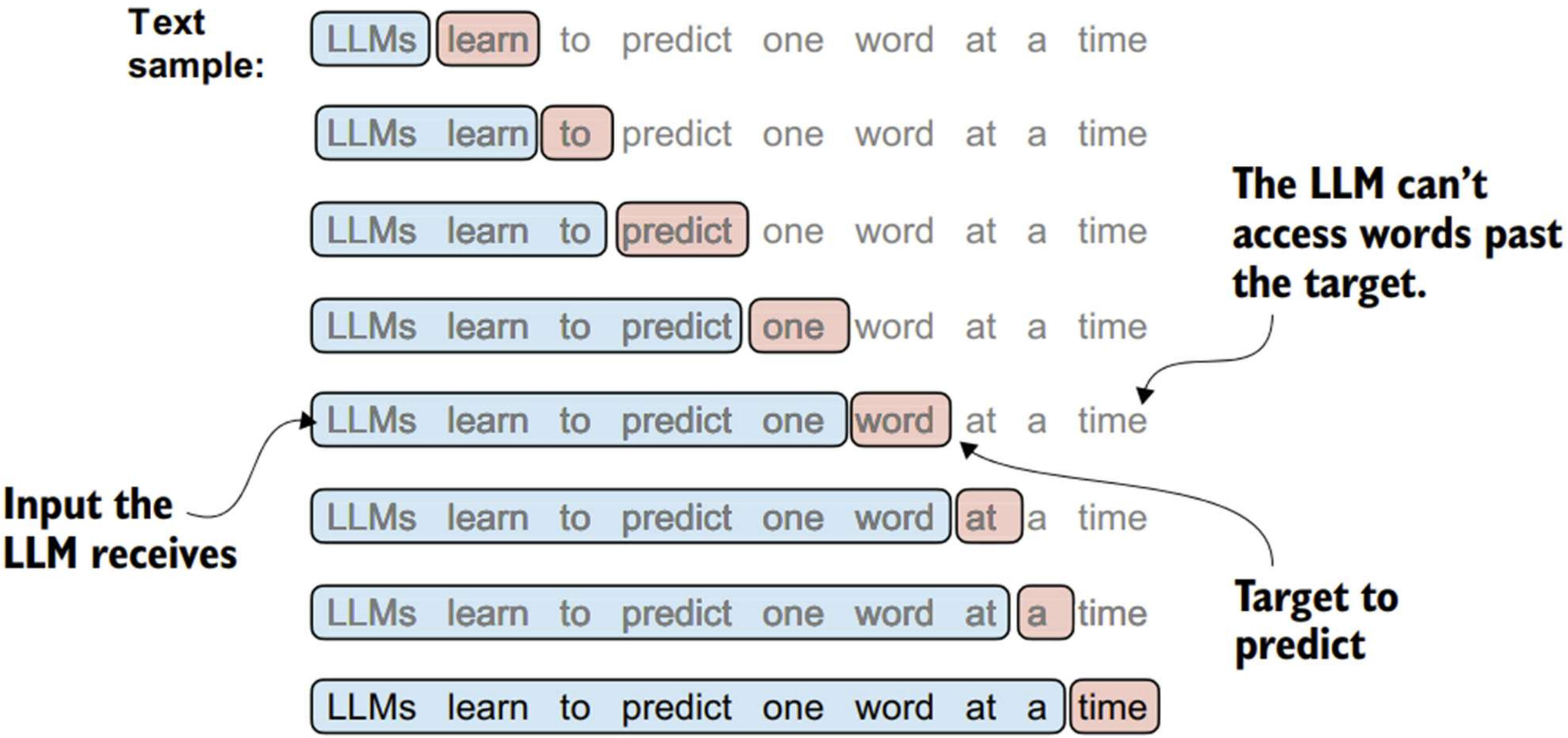
The training set consists of only one sentence for illustration purposes.

The vocabulary contains all unique tokens in the training set and is usually sorted alphabetically.

Each unique token is mapped to a unique integer called token ID.

Unique tokens	Token IDs
brown	0
dog	1
fox	2
jumps	3
lazy	4
over	5
quick	6
the	7

Input to LLM



Text sample:

LLMs learn to predict one word at a time

LLMs learn to predict one word at a time

LLMs learn to predict one word at a time

LLMs learn to predict one word at a time

LLMs learn to predict one word at a time

LLMs learn to predict one word at a time

LLMs learn to predict one word at a time

LLMs learn to predict one word at a time

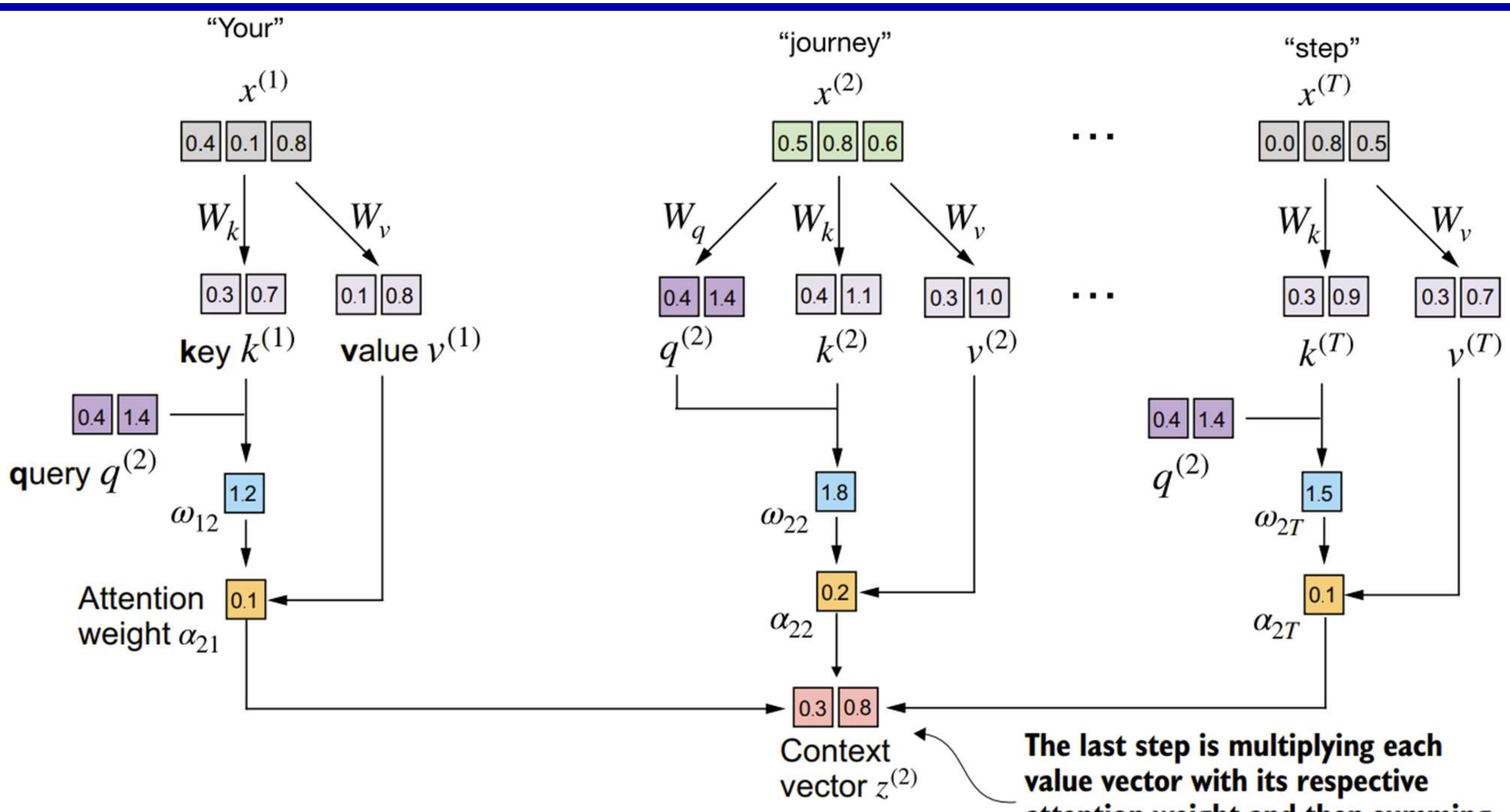
The LLM can't access words past the target.

Input the LLM receives

Target to predict

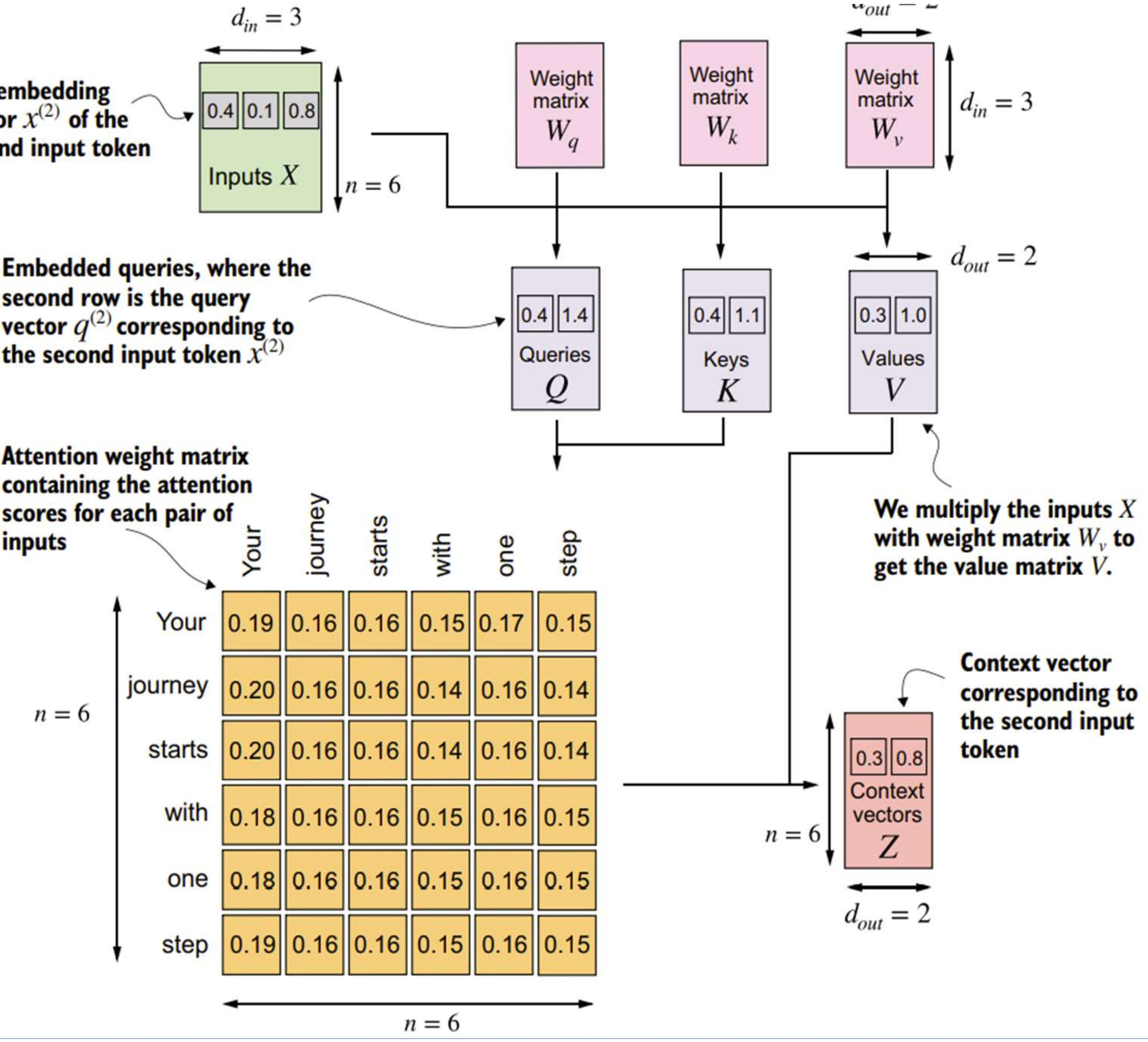
Self-Attention mechanism

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$



The last step is multiplying each value vector with its respective attention weight and then summing them to obtain the context vector

Self-Attention Overall Picture



The embedding vector $x^{(2)}$ of the second input token

Embedded queries, where the second row is the query vector $q^{(2)}$ corresponding to the second input token $x^{(2)}$

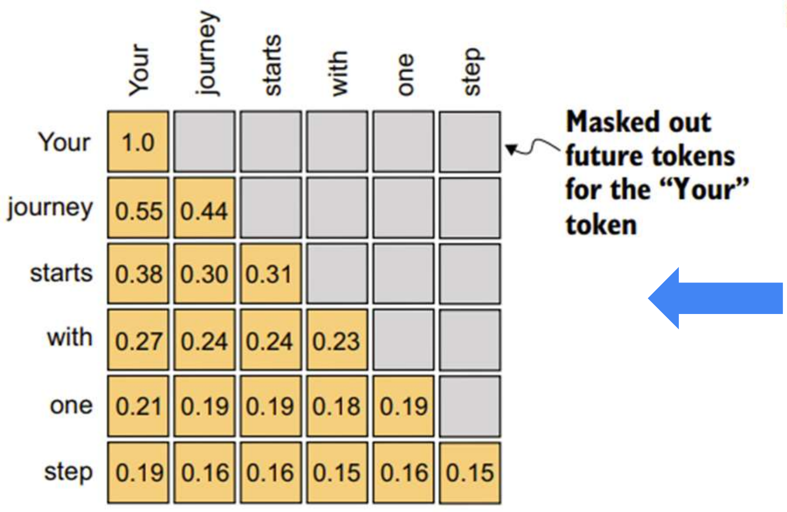
Attention weight matrix containing the attention scores for each pair of inputs

We multiply the inputs X with weight matrix W_v to get the value matrix V.

Context vector corresponding to the second input token

	Your	journey	starts	with	one	step
Your	0.19	0.16	0.16	0.15	0.17	0.15
journey	0.20	0.16	0.16	0.14	0.16	0.14
starts	0.20	0.16	0.16	0.14	0.16	0.14
with	0.18	0.16	0.16	0.15	0.16	0.15
one	0.18	0.16	0.16	0.15	0.16	0.15
step	0.19	0.16	0.16	0.15	0.16	0.15

Causal attention



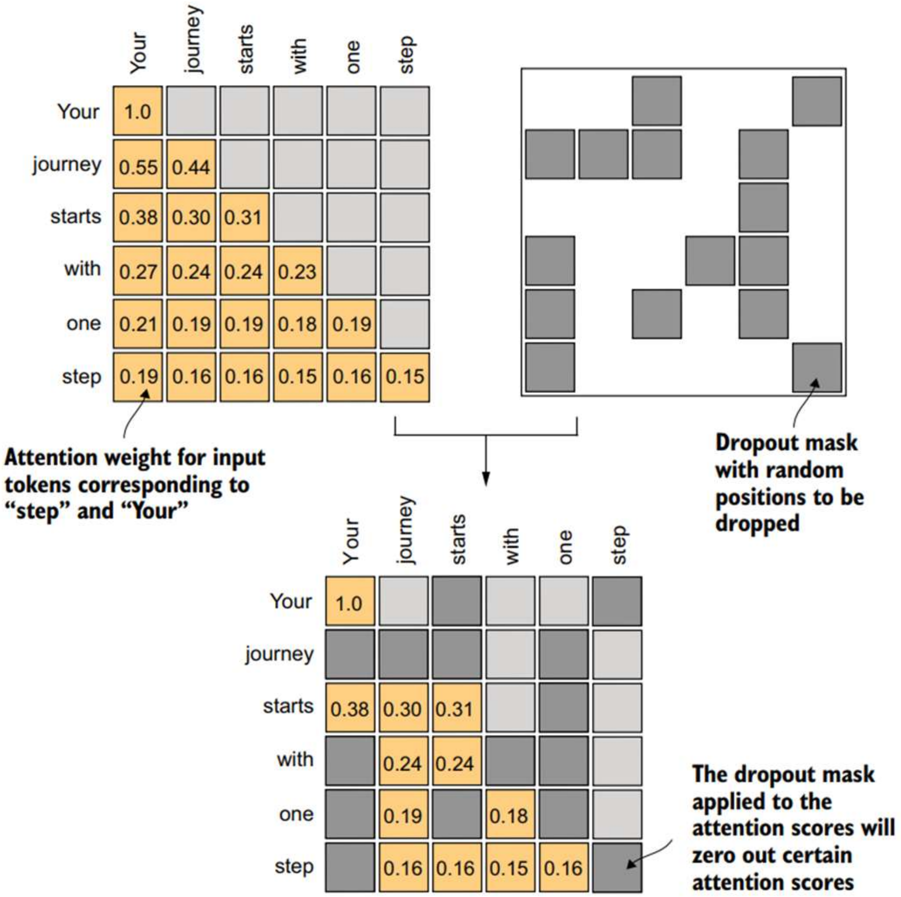
	Your	journey	starts	with	one	step
Your	1.0					
journey	0.55	0.44				
starts	0.38	0.30	0.31			
with	0.27	0.24	0.24	0.23		
one	0.21	0.19	0.19	0.18	0.19	
step	0.19	0.16	0.16	0.15	0.16	0.15

Masked out future tokens for the “Your” token

Self-Attention mechanism with dropout

Help prevent overfitting

Only used during training



	Your	journey	starts	with	one	step
Your	1.0					
journey	0.55	0.44				
starts	0.38	0.30	0.31			
with	0.27	0.24	0.24	0.23		
one	0.21	0.19	0.19	0.18	0.19	
step	0.19	0.16	0.16	0.15	0.16	0.15

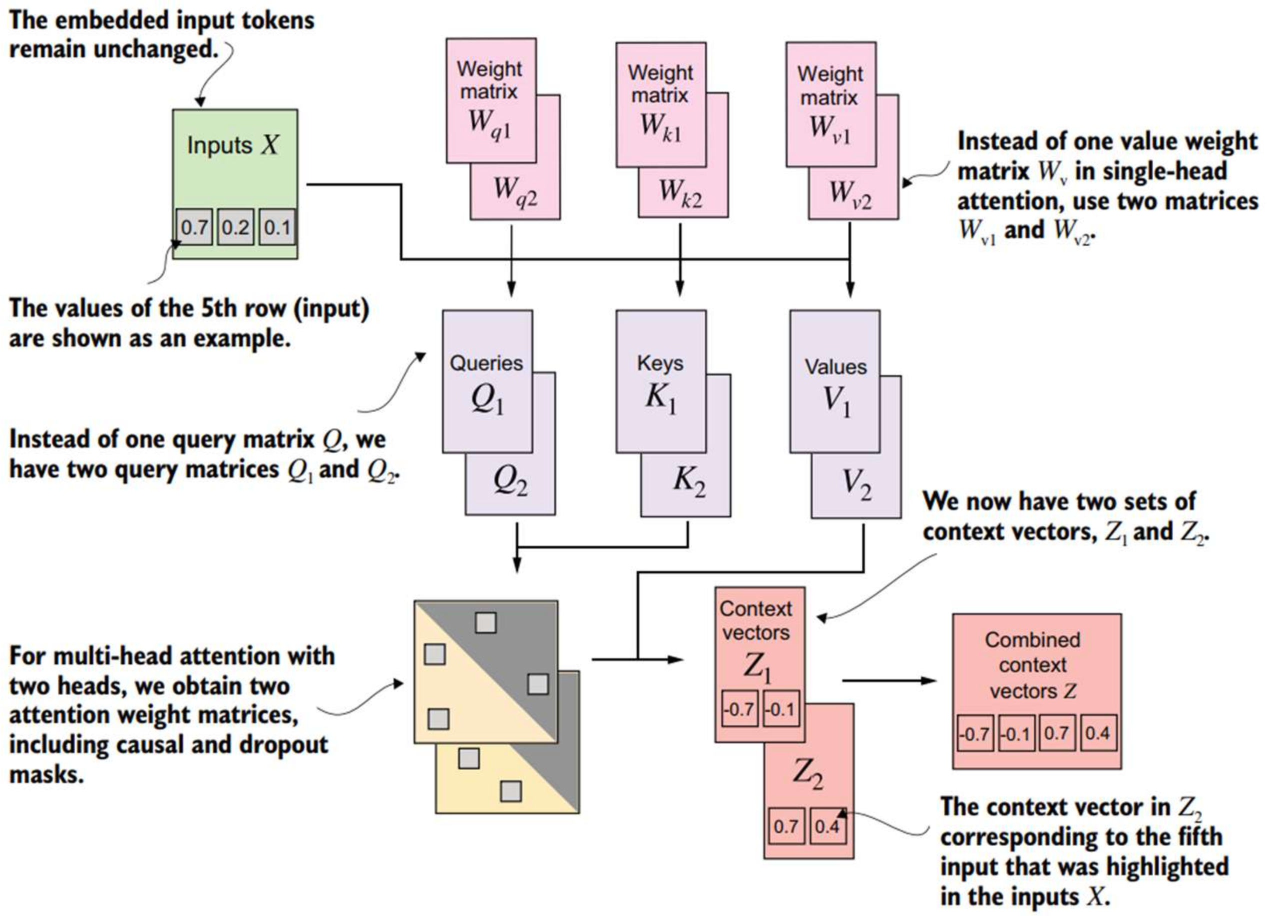
Attention weight for input tokens corresponding to "step" and "Your"

Dropout mask with random positions to be dropped

	Your	journey	starts	with	one	step
Your	1.0		▨			▨
journey	▨	▨	▨		▨	
starts	0.38	0.30	0.31		▨	
with	▨	0.24	0.24	▨	▨	
one	▨	0.19	▨	0.18	▨	
step	▨	0.16	0.16	0.15	0.16	▨

The dropout mask applied to the attention scores will zero out certain attention scores

Multi-headed self-attention mechanism



The embedded input tokens remain unchanged.

The values of the 5th row (input) are shown as an example.

Instead of one value weight matrix W_v in single-head attention, use two matrices W_{v1} and W_{v2}.

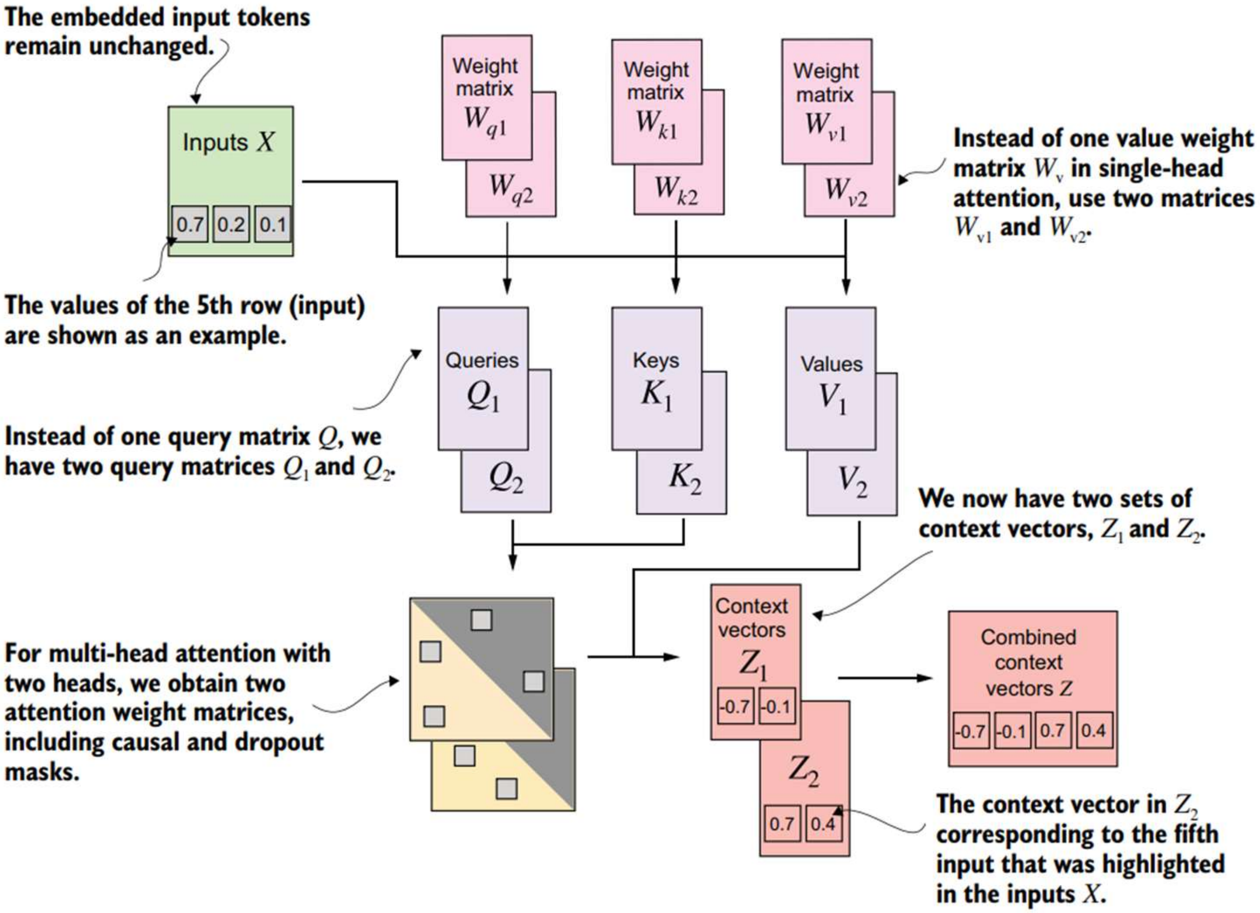
Instead of one query matrix Q, we have two query matrices Q_1 and Q_2.

We now have two sets of context vectors, Z_1 and Z_2.

For multi-head attention with two heads, we obtain two attention weight matrices, including causal and dropout masks.

The context vector in Z_2 corresponding to the fifth input that was highlighted in the inputs X.

Multi-headed self-attention mechanism



The embedded input tokens remain unchanged.

The values of the 5th row (input) are shown as an example.

Instead of one value weight matrix W_v in single-head attention, use two matrices W_{v1} and W_{v2}.

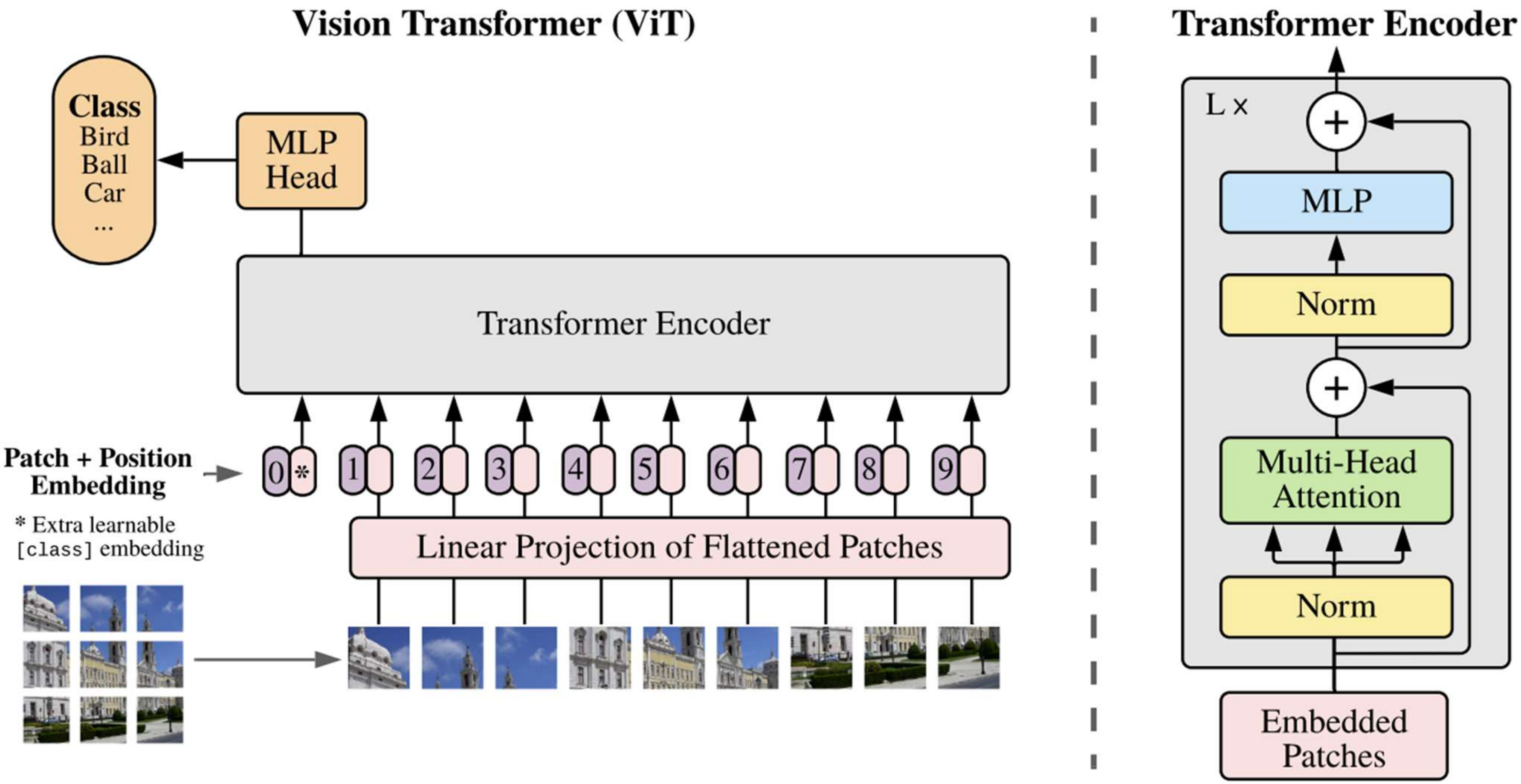
Instead of one query matrix Q, we have two query matrices Q_1 and Q_2.

We now have two sets of context vectors, Z_1 and Z_2.

For multi-head attention with two heads, we obtain two attention weight matrices, including causal and dropout masks.

The context vector in Z_2 corresponding to the fifth input that was highlighted in the inputs X.

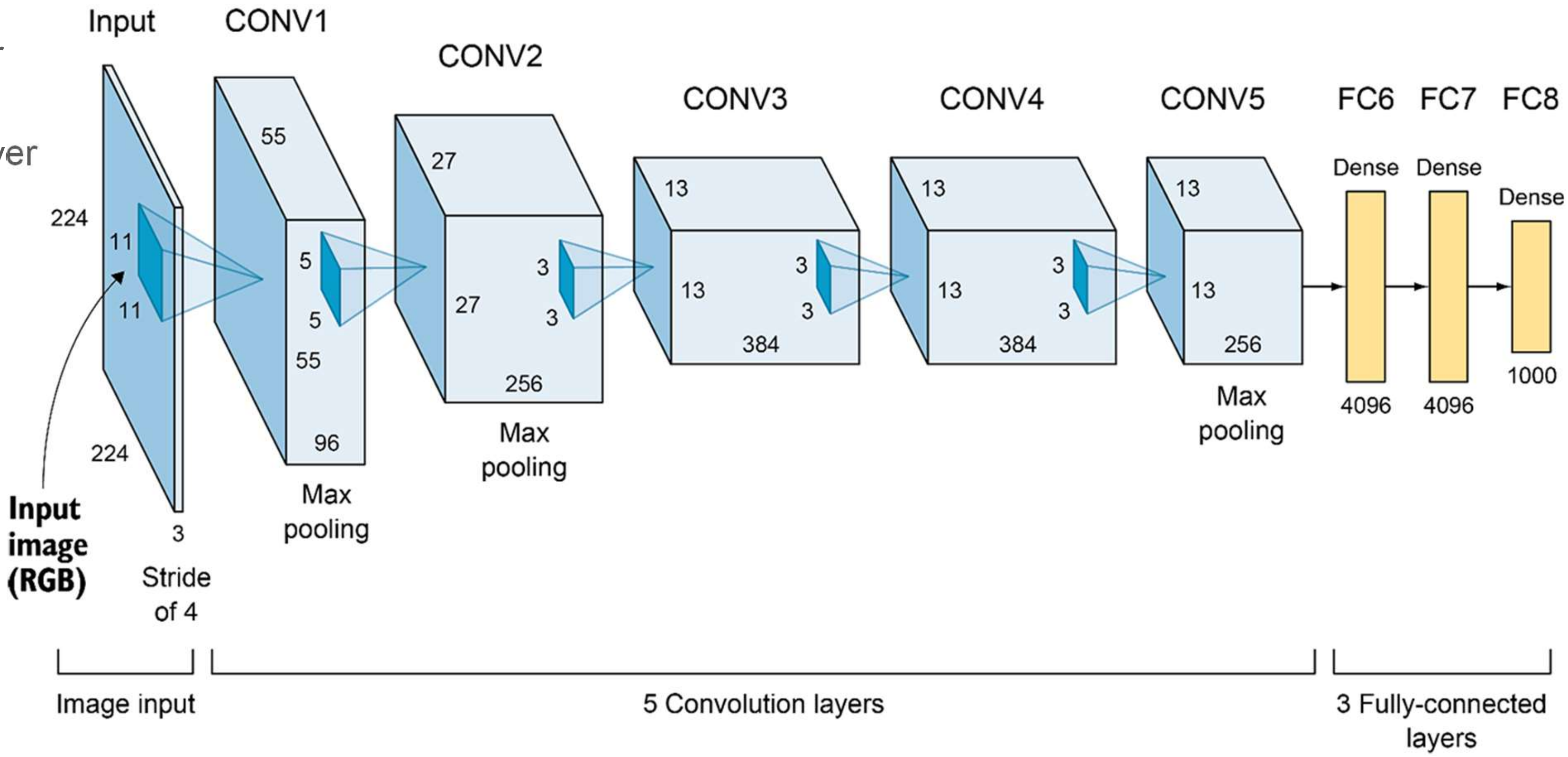
Vision Transformer (ViT)



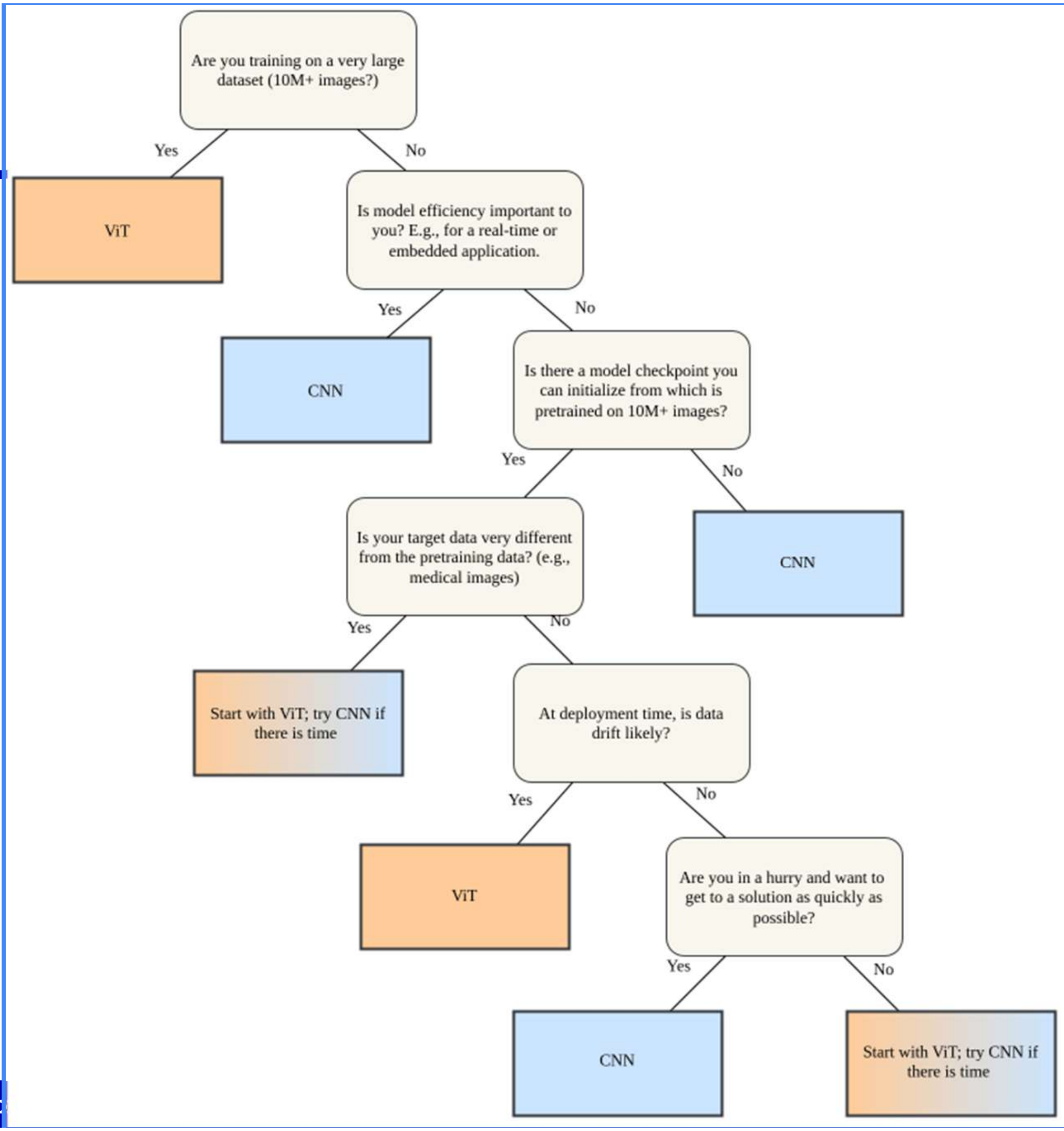
Convolutional neural networks (CNN)

Layer types:

- *Convolutional layer*
- Pooling layer
- Fully-connected layer



ViT vs CNN for Vision



Are you training on a very large dataset (10M+ images?)

Yes

ViT

No

Is model efficiency important to you? E.g., for a real-time or embedded application.

Yes

CNN

No

Is there a model checkpoint you can initialize from which is pretrained on 10M+ images?

Yes

Is your target data very different from the pretraining data? (e.g., medical images)

Yes

Start with ViT; try CNN if there is time

No

At deployment time, is data drift likely?

Yes

ViT

No

Are you in a hurry and want to get to a solution as quickly as possible?

Yes

CNN

No

Start with ViT; try CNN if there is time

No

CNN

https://tobiasvanderwerff.com/2024/05/15/cnn-vs-vit.html

Further reading

- Attention Is All You Need (https://arxiv.org/pdf/1706.03762)
- The Illustrated Transformer (https://jalammar.github.io/illustrated-transformer/)
- Build a Large Language Model (From Scratch) (https://github.com/rasbt/LLMs-from-scratch)