# VIP Cheatsheet: Transformers & Large Language Models

Afshine Amidi and Shervine Amidi

March 23, 2025

*This VIP cheatsheet gives an overview of what is in the "Super Study Guide: Transformers & Large Language Models" book, which contains ∼600 illustrations over 250 pages and goes into the following concepts in depth. You can find more details at* `https://superstudy.guide`.

## 1 Foundations

### 1.1 Tokens

❒ **Definition** – A *token* is an indivisible unit of text, such as a word, subword or character, and is part of a predefined vocabulary.

*Remark: The unknown token `[UNK]` represents unknown pieces of text while the padding token `[PAD]` is used to fill empty positions to ensure consistent input sequence lengths.*

❒ **Tokenizer** – A *tokenizer* $T$ divides text into tokens of an arbitrary level of granularity.

this teddy bear is reaaaally cute → $T$ → `this` `teddy bear` `is` `[UNK]` `cute` `[PAD]` ... `[PAD]`

Here are the main types of tokenizers:

| Type | Pros | Cons | Illustration |
|---|---|---|---|
| Word | • Easy to interpret<br>• Short sequence | • Large vocabulary size<br>• Word variations not handled | `teddy` `bear` |
| Subword | • Word roots leveraged<br>• Intuitive embeddings | • Increased sequence length<br>• Tokenization more complex | `ted` `##dy` `bear` |
| Character<br>Byte | • No out-of-vocabulary concerns<br>• Small vocabulary size | • Much longer sequence length<br>• Patterns hard to interpret because too low-level | `t` `e` `d` `d` `y` ␣ `b` `e` `a` `r` |

*Remark: Byte-Pair Encoding (BPE) and Unigram are commonly-used subword-level tokenizers.*

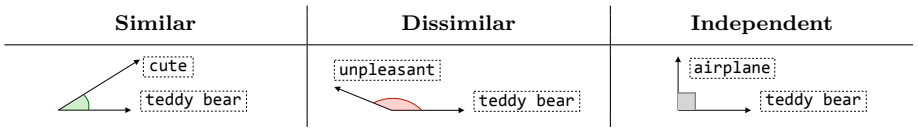
### 1.2 Embeddings

❒ **Definition** – An *embedding* is a numerical representation of an element (e.g. token, sentence) and is characterized by a vector $x \in \mathbb{R}^n$.

❒ **Similarity** – The *cosine similarity* between two tokens $t_1, t_2$ is quantified by:

$$\text{similarity}(t_1, t_2) = \frac{t_1 \cdot t_2}{||t_1||\ ||t_2||} = \cos(\theta) \in [-1, 1]$$

The angle $\theta$ characterizes the similarity between the two tokens:

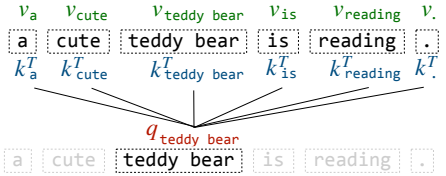



*Remark: Approximate Nearest Neighbors (ANN) and Locality Sensitive Hashing (LSH) are methods that approximate the similarity operation efficiently over large databases.*

## 2 Transformers

### 2.1 Attention

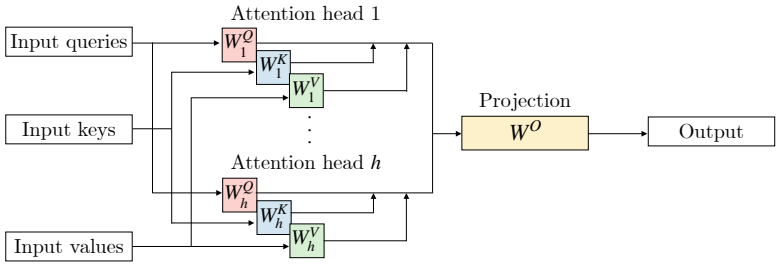
❒ **Formula** – Given a *query* $q$, we want to know which *key* $k$ the query should pay "attention" to with respect to the associated *value* $v$.




Attention can be efficiently computed using matrices $Q, K, V$ that contain queries $q$, keys $k$ and values $v$ respectively, along with the dimension $d_k$ of keys:

$$\text{attention} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

❒ **MHA** – A ***M**ulti-**H**ead **A**ttention* (MHA) layer performs attention computations across multiple heads, then projects the result in the output space.

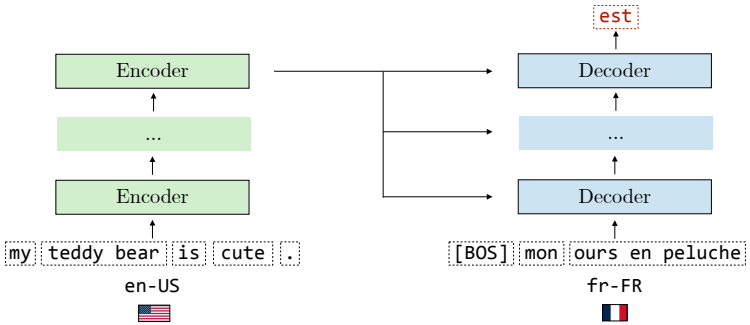



It is composed of $h$ attention heads as well as matrices $W^Q, W^K, W^V$ that project the input to obtain queries $Q$, keys $K$ and values $V$. The projection is done using matrix $W^O$.

*Remark: Grouped-Query Attention (GQA) and Multi-Query Attention (MQA) are variations of MHA that reduce computational overhead by sharing keys and values across attention heads.*

### 2.2 Architecture

❒ **Overview** – *Transformer* is a landmark model relying on the self-attention mechanism and is composed of encoders and decoders. Encoders compute meaningful embeddings of the input that are then used by decoders to predict the next token in the sequence.

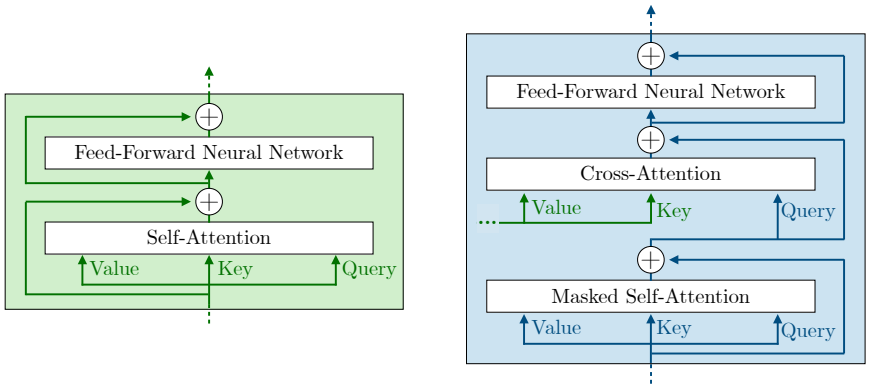



*Remark: Although the Transformer was initially proposed as a model for translation tasks, it is now widely used across many other applications.*

❒ **Components** – The *encoder* and *decoder* are two fundamental components of the Transformer and have different roles:

**Encoder**
Encoded embeddings encapsulate meaning of input

**Decoder**
Decoded embeddings encapsulate meaning of both input and output predicted so far

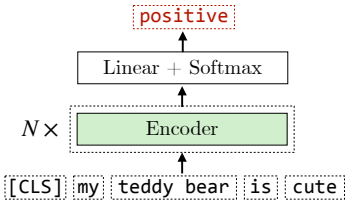



❒ **Position embeddings** – *Position embeddings* inform where the token is in the sentence and are of the same dimension as the token embeddings. They can either be arbitrarily defined or learned from the data.

*Remark: Rotary Position Embeddings (RoPE) are a popular and efficient variation that rotate query and key vectors to incorporate relative position information.*

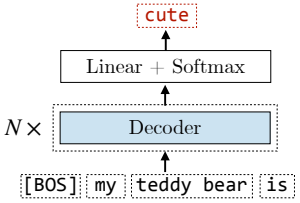
### 2.3 Variants

❒ **Encoder-only** – ***B**idirectional **E**ncoder **R**epresentations from **T**ransformers* (BERT) is a Transformer-based model composed of a stack of encoders that takes some text as input, and outputs meaningful embeddings, which can be later used in downstream classification tasks.




A [CLS] token is added at the beginning of the sequence to capture the meaning of the sentence. Its encoded embedding is often used in downstream tasks, such as sentiment extraction.

❒ **Decoder-only** – ***G**enerative **P**re-trained **T**ransformer* (GPT) is an autoregressive Transformer-based model that is composed of a stack of decoders. Contrary to BERT and its derivatives, GPT treats all problems as text-to-text problems.




Most of the current state-of-the-art LLMs rely on a decoder-only architecture, such as the GPT series, LLaMA, Mistral, Gemma, DeepSeek, etc.

*Remark: Encoder-decoder models, like T5, are also autoregressive and share many characteristics with decoder-only models.*

### 2.4 Optimizations

❒ **Attention approximation** – Attention computations are in $\mathcal{O}(n^2)$, which can be costly as the sequence length $n$ increases. There are two main methods to approximate computations:

- *Sparsity*: Self-attention does not happen through the whole sequence but only between more relevant tokens.


- *Low-rank*: The attention formula is simplified as the product of low-rank matrices, which brings down the computation burden.

❒ **Flash attention** – *Flash attention* is an exact method that optimizes attention computations by cleverly leveraging GPU hardware, using the fast ***S**tatic **R**andom-**A**ccess **M**emory* (SRAM) for matrix operations before writing results to the slower ***H**igh **B**andwidth **M**emory* (HBM).

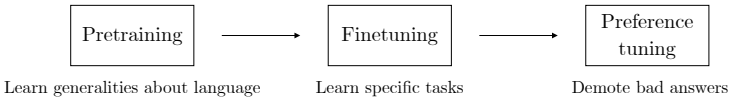
*Remark: In practice, this reduces memory usage and speeds up computations.*

# 3 Large language models

## 3.1 Overview

❒ **Definition** – A ***L**arge **L**anguage **M**odel* (LLM) is a Transformer-based model with strong NLP capabilities. It is "large" in the sense that it typically contains billions of parameters.

❒ **Lifecycle** – An LLM is trained in 3 steps: pretraining, finetuning and preference tuning.

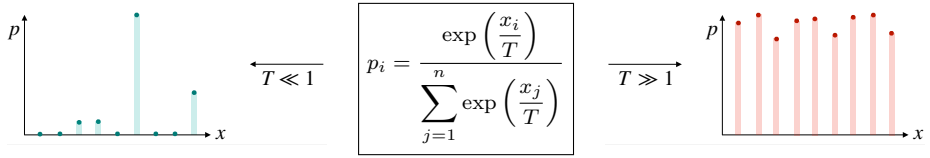



Finetuning and preference tuning are post-training approaches that aim at *aligning the model* to perform certain tasks.

## 3.2 Prompting

❒ **Context length** – The *context length* of a model is the maximum number of tokens that can fit into the input. It typically ranges from tens of thousands to millions of tokens.

❒ **Decoding sampling** – Token predictions are sampled from the predicted probability distribution $p_i$, which is controlled by the hyperparameter temperature $T$.




*Remark: High temperatures lead to more creative outputs whereas low temperatures lead to more deterministic ones.*

❒ **Chain-of-thought** – ***C**hain-**o**f-**T**hought* (CoT) is a reasoning process in which the model breaks down a complex problem into a series of intermediate steps. This helps the model to generate the correct final response. ***T**ree **o**f **T**houghts* (ToT) is a more advanced version of CoT.

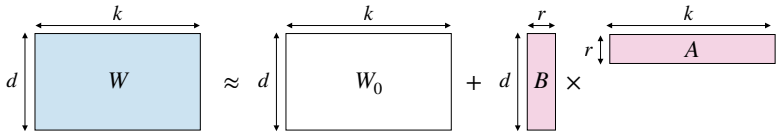
*Remark: Self-consistency is a method that aggregates answers across CoT reasoning paths.*

## 3.3 Finetuning

❒ **SFT** – ***S**upervised **F**ine**T**uning* (SFT) is a post-training approach that aligns the behavior of the model with an end task. It relies on high-quality input-output pairs aligned with the task.

*Remark: If the SFT data is about instructions, then this step is called "instruction tuning".*

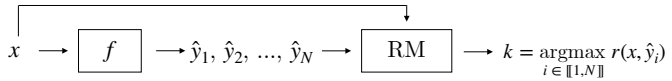
❒ **PEFT** – ***P**arameter-**E**fficient **F**ine**T**uning* (PEFT) is a category of methods used to run SFT efficiently. In particular, ***L**ow-**R**ank **A**daptation* (LoRA) approximates the learnable weights $W$ by fixing $W_0$ and learning low-rank matrices $A, B$ instead:



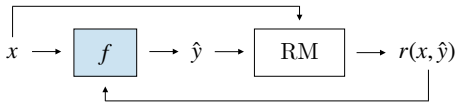

*Remark: Other PEFT techniques include prefix tuning and adapter layer insertion.*

## 3.4 Preference tuning

❒ **Reward model** – A ***R**eward **M**odel* (RM) is a model that predicts how well an output $\hat{y}$ aligns with desired behavior given the input $x$. ***B**est-**o**f-**N*** (BoN) sampling, also called *rejection sampling*, is a method that uses a reward model to select the best response among $N$ generations.




❒ **Reinforcement learning** – ***R**einforcement **L**earning* (RL) is an approach that leverages RM and updates the model $f$ based on rewards for its generated outputs. If RM is based on human preferences, this process is called ***R**einforcement **L**earning from **H**uman **F**eedback* (RLHF).

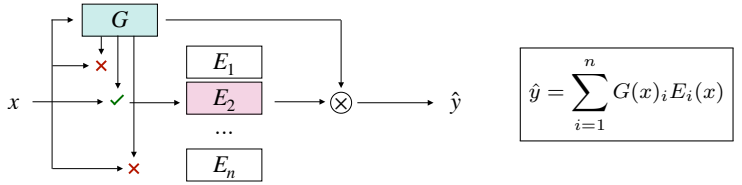



***P**roximal **P**olicy **O**ptimization* (PPO) is a popular RL algorithm that incentivizes higher rewards while keeping the model close to the base model to prevent reward hacking.

*Remark: There are also supervised approaches, like Direct Preference Optimization (DPO), that combine RM and RL into one supervised step.*

## 3.5 Optimizations

❒ **Mixture of experts** – A ***M**ixture **o**f **E**xperts* (MoE) is a model that activates only a portion of its neurons at inference time. It is based on a gate $G$ and experts $E_1, ..., E_n$.




MoE-based LLMs use this gating mechanism in their FFNNs.

*Remark: Training an MoE-based LLM is notoriously challenging, as mentioned in the LLaMA paper whose authors chose to not use this architecture despite its inference-time efficiency.*

❒ **Distillation** – *Distillation* is a process where a (small) student model $S$ is trained on the prediction outputs of a (big) teacher model $T$. It is trained using the KL divergence loss:

$$\text{KL}(\hat{y}_T || \hat{y}_S) = \sum_i \hat{y}_T^{(i)} \log\left(\frac{\hat{y}_T^{(i)}}{\hat{y}_S^{(i)}}\right)$$

*Remark: Training labels are considered as "soft" labels since they represent class probabilities.*

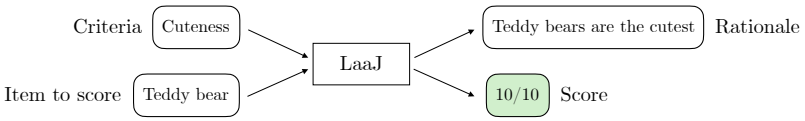
❒ **Quantization** – *Model quantization* is a category of techniques that reduces the precision of model weights while limiting its impact on the resulting model performance. As a result, this reduces the model's memory footprint and speeds up its inference.

*Remark: QLoRA is a commonly-used quantized variant of LoRA.*

# 4 Applications

## 4.1 LLM-as-a-Judge

❒ **Definition** – ***LLM-as-a-Judge*** (LaaJ) is a method that uses an LLM to score given outputs according to some provided criteria. Notably, it is also able to generate a rationale for its score, which helps with interpretability.




Contrary to pre-LLM era metrics such as ***R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation* (ROUGE), LaaJ does not need any reference text, which makes it convenient to evaluate on any kind of task. In particular, LaaJ shows strong correlation with human ratings when it relies on a big powerful model (e.g. GPT-4), as it requires reasoning capabilities to perform well.

*Remark: LaaJ is useful to perform quick rounds of evaluations but it is important to monitor the alignment between LaaJ outputs and human evaluations to make sure there is no divergence.*

❒ **Common biases** – LaaJ models can exhibit the following biases:

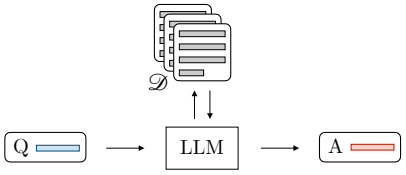
| | **Position bias** | **Verbosity bias** | **Self-enhancement bias** |
|---|---|---|---|
| **Problem** | Favors first position in pairwise comparisons | Favors more verbose content | Favors outputs generated by themselves |
| **Solution** | Average metric on randomized positions | Add a penalty on the output length | Use a judge built from a different base model |

A remedy to these issues can be to finetune a custom LaaJ, but this requires a lot of effort.

*Remark: The list of biases above is not exhaustive.*

## 4.2 RAG

❒ **Definition** – ***R**etrieval-**A**ugmented **G**eneration* (RAG) is a method that allows the LLM to access relevant external knowledge to answer a given question. This is particularly useful if we want to incorporate information past the LLM pretrained knowledge cut-off date.




Given a knowledge base $\mathscr{D}$ and a question, a **R**etriever fetches the most relevant documents, then **A**ugments the prompt with the relevant information before **G**enerating the output.

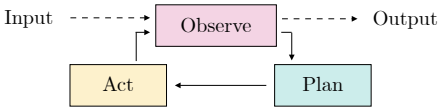
*Remark: The retrieval stage typically relies on embeddings from encoder-only models.*

❒ **Hyperparameters** – The knowledge base $\mathscr{D}$ is initialized by chunking the documents into *chunks* of size $n_c$ and embedding them into vectors of size $\mathbb{R}^d$.


## 4.3 Agents

❒ **Definition** – An *agent* is a system that autonomously pursues goals and completes tasks on a user's behalf. It may use different chains of LLM calls to do so.

❒ **ReAct** – ***R**eason* + ***Act*** (ReAct) is a framework that allows for multiple chains of LLM calls to complete complex tasks:




This framework is composed of the steps below:

- *Observe*: Synthesize previous actions and explicitly state what is currently known.
- *Plan*: Detail what tasks need to be accomplished and what tools to call.
- *Act*: Perform an action via an API or look for relevant information in a knowledge base.

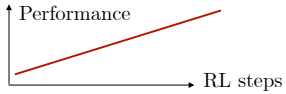
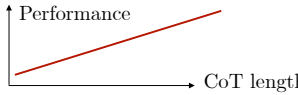
*Remark: Evaluating an agentic system is challenging. However, this can still be done both at the component level via local inputs-outputs and at the system level via chains of calls.*

## 4.4 Reasoning models

❒ **Definition** – A *reasoning model* is a model that relies on CoT-based reasoning traces to solve more complex tasks in math, coding and logic. Examples of reasoning models include OpenAI's o series, DeepSeek-R1 and Google's Gemini Flash Thinking.

*Remark: DeepSeek-R1 explicitly outputs its reasoning trace between `<think>` tags.*

❒ **Scaling** – Two types of scaling methods are used to enhance reasoning capabilities:

| | **Description** | **Illustration** |
|---|---|---|
| **Train-time scaling** | Run RL for longer to let the model learn how to produce CoT-style reasoning traces before giving an answer |   |
| **Test-time scaling** | Let the model think longer before providing an answer with budget forcing keywords such as `"Wait"` |   |