ASC

Austrian Scientific Computing

Foundations of LLM Mastery:
Prompt Engineering Essentials

25 August 2025

ONLINE

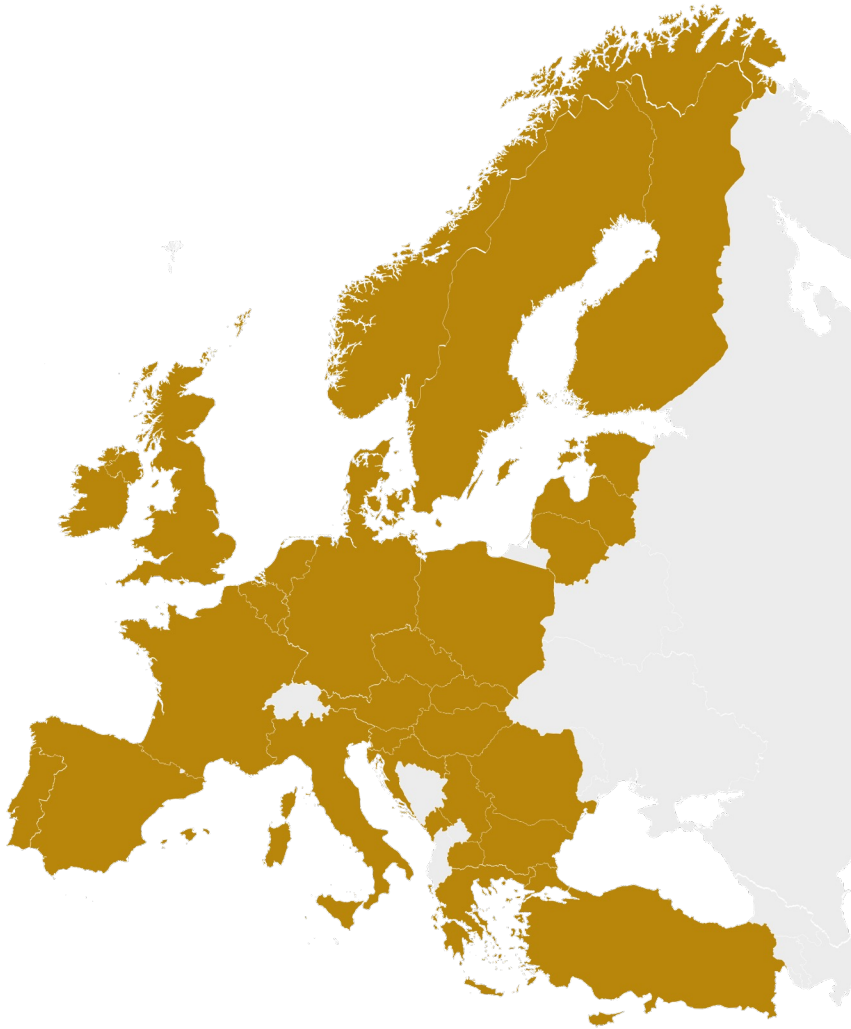
EURO AUSTRIA

AI:AT

EuroCC

Fully funded EU project

- EuroCC is EU-funded international initiative aimed to support the uptake of AI and High-Performance Computing (HPC) in Europe
- Set up of 32 National Competence Centres (NCCs) across Europe
- EuroCC Austria is one of them
- Service Provider for AI, HPC and HPDA



European HPC Landscape

EuroHPC JU systems

Different access modes:
Calls for Proposals

AI Playground Access: Get 5000 GPU hours on a supercomputer – for free

AI Fast Lane Access: Get up to 50.000 GPU hours on a supercomputer – for free

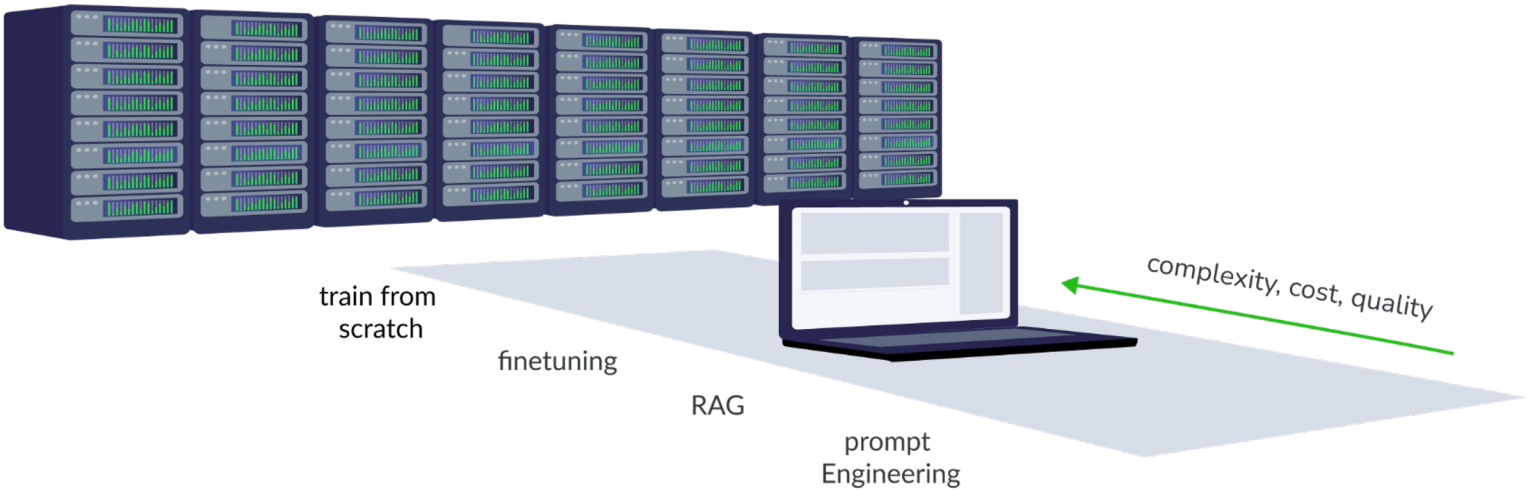
AI Factory

In production from October 2025

Large Language Models on Supercomputers

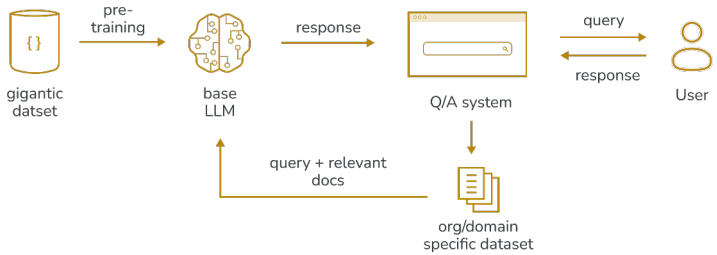
A brief intro

How can you influence LLMs?



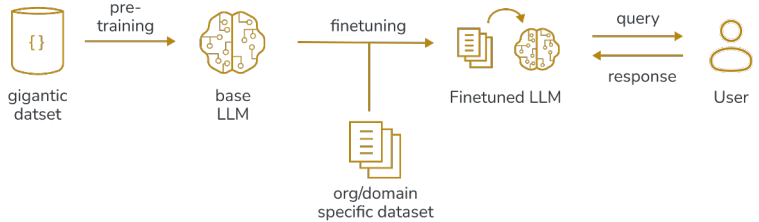
How can you use LLMs with your data?

RAG:
Retrieval Augmented Generation



- Ideal for tapping into company‘s knowledge DBs
- Minimises hallucinations by grounding response on retrieved evidence
- Can quickly adapt to changing data
- Makes it easier to interpret result

Finetuning



- Ideal if plenty of labelled data is available
- Teaches model domain specific vocabulary
- Company‘s writing/answer style is „baked“ into model through fine-tuned parameters

Transformer Anatomy

Attention is really all you need?

Transformer Anatomy

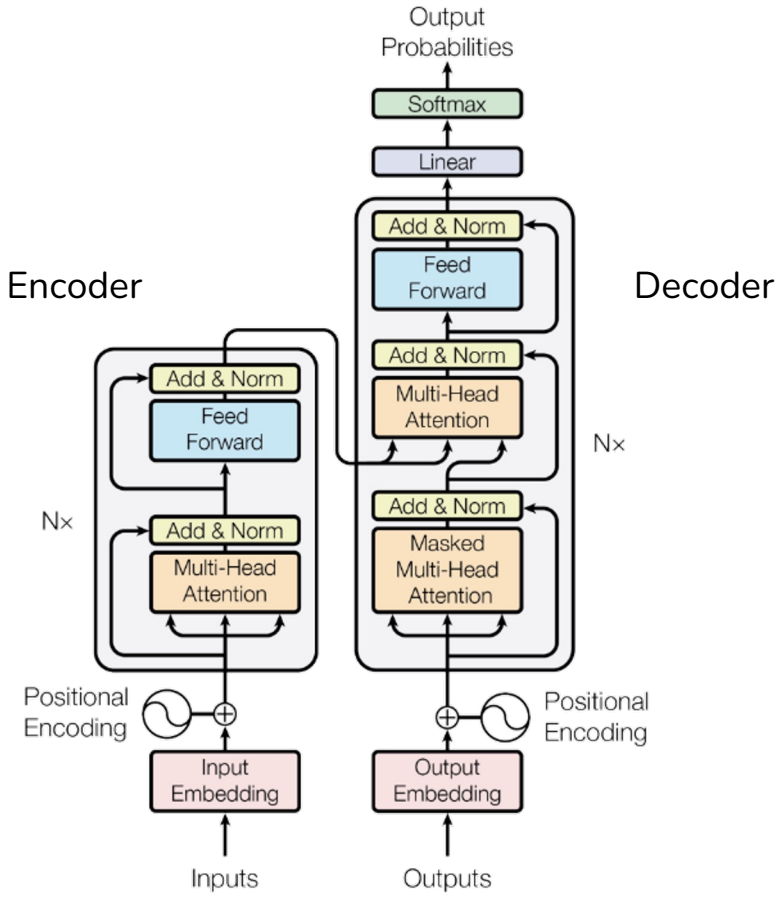
The original architecture

A transforer consists of a encoder and/or decoder block.

Words (tokens) are input as numerical representations (embeddings).

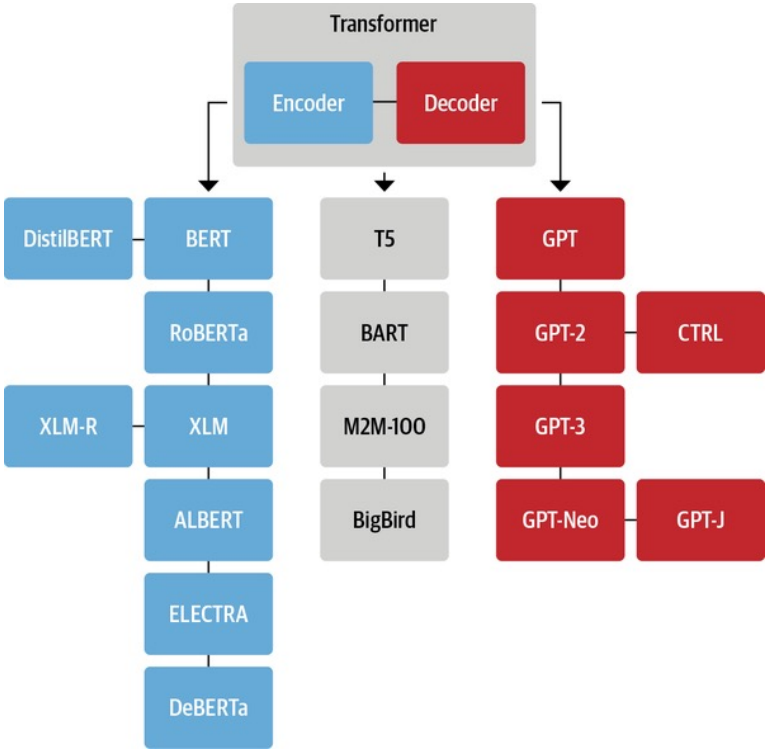
About 1/3 of all parameters are in the multi-head attention blocks

About 2/3 of all parameters are in the feed forward networks (also known as multi layer perceptron)



Source: “Attention Is All You Need”, Vaswani et al.

Transformer Family



Source: Natural Language Processing with Transformers, O'Reilly

Encoder only:
These models ecxel at text classification, named entity recognition, and question answering

Decoder only:
Very good at predicting the next word in a sequence, therefore mostly used for text generation

Encoder-Decoder:
These models are often used for machine translation or summerization tasks.

Tokenization & Embeddings

Turning words into numbers

From Text to Tokens

Word Tokenization

- Model does not have to learn words from characters
- Each word has specific ID
- Size of vocabulary explodes
- Model needs to learn different tokens for e.g. singular and plural

Large lanugage models on supercomputers

From Text to Tokens

Subword Tokenization

- Best of both worlds
- Deal with complex words easily
- Frequent words remain as one token

Large lanugage models
on super com put ers

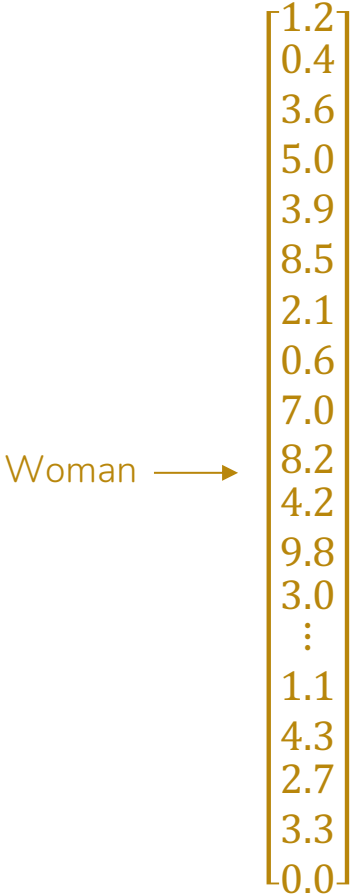
From Tokens to Vectors

Embeddings

Tokens are mapped to unique integers accoding to the vocabulary size of the tokenizer.

Now, the tokens need to be embedded, which means turned into a vector representation.

This is done by an embedding layer of a model. The model takes each token ID and looks it up in an embedding matrix. The embedding matrix is a learned set of weights that maps each token ID to a corresponding high-dimensional vector (embedding).

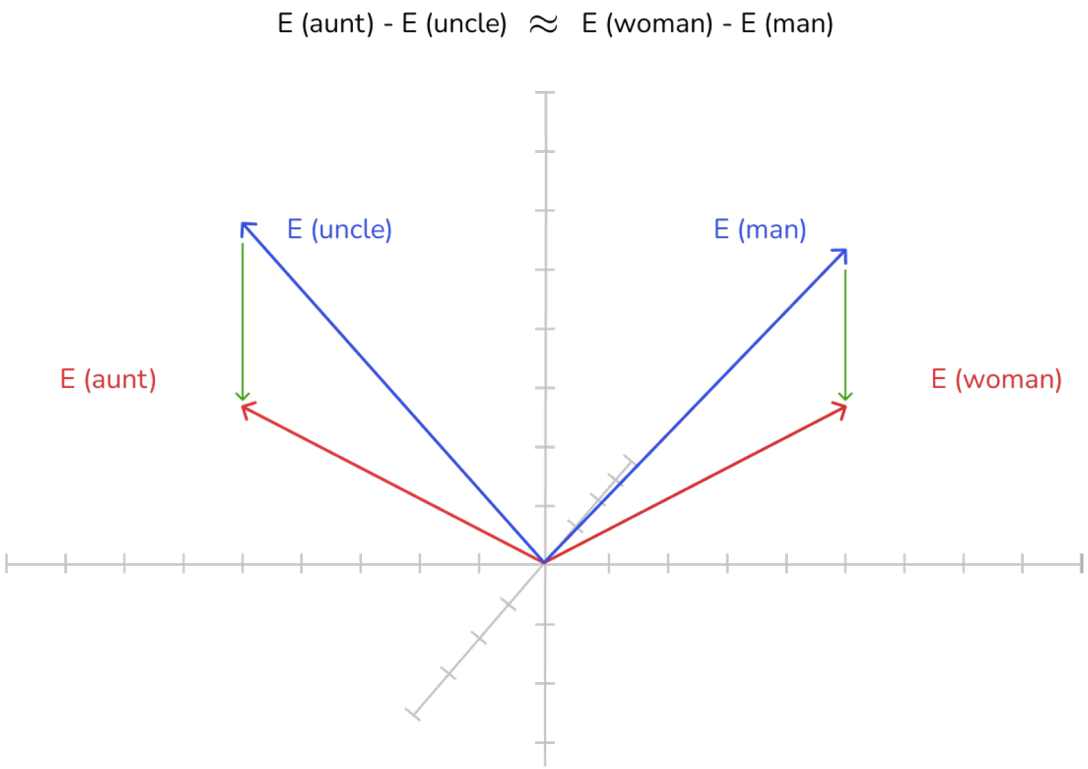


From Tokens to Vectors

Embeddings

Each word/token has a unique direction in the embedding space

Similar words point in a similar direction.



Context Is All You Need

Embeddings

Here, we will refer to “word” instead of “token”, as it makes the content easier to explain.

A word embedding comes as a multi dimensional vector (e.g. 12.000 dim).

The initial word embedding in all of the examples of the word „mole“ is the same.

The European mole is a mammal

Take a biopsy of the mole

6.02×10^{23}

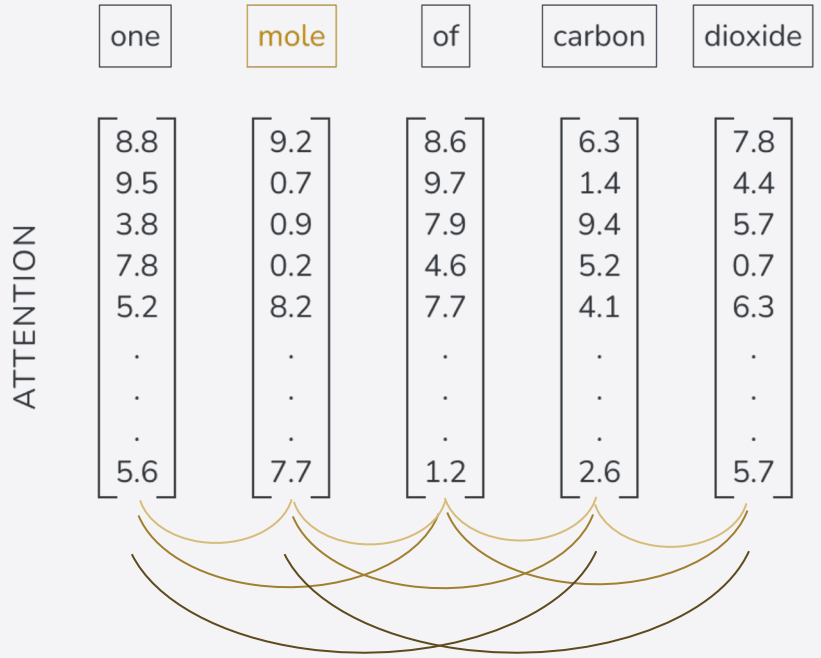
One mole of carbon dioxide

Context Is All You Need

Attention

The word „mole“ should be represented by a **unique vector** in the embedding space, depending on ist **context.**

An **attention** block should **compute the vectors that you need to add** to the original, generic vector to get it to the correct, meaningful, rich representation, depending on the context in which the word is used.

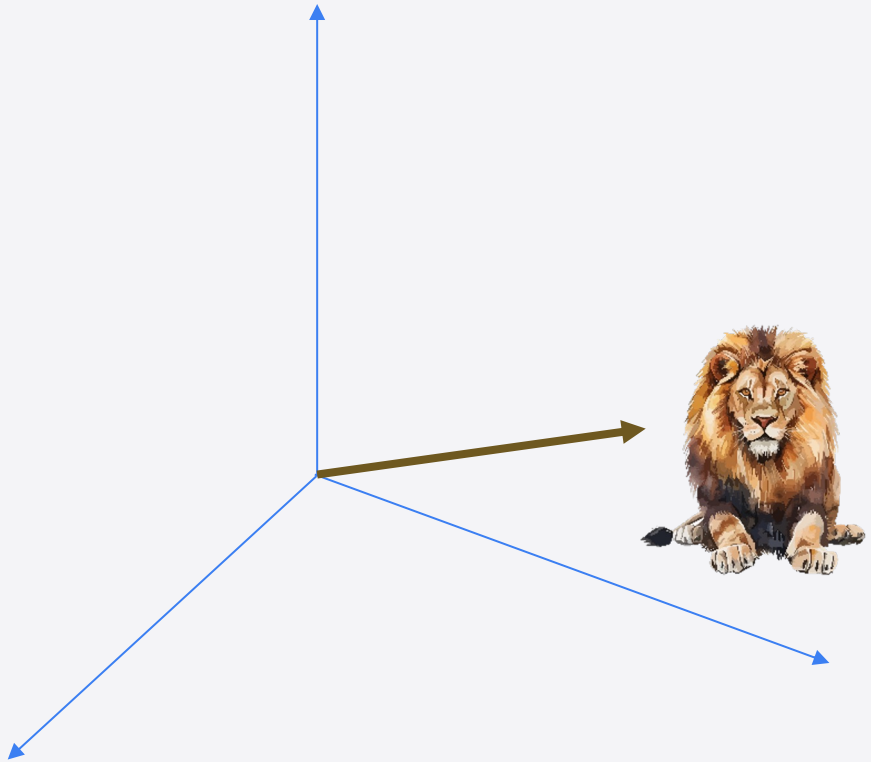


Context Is All You Need

Lion

We associate the word „lion“ with a big cat, living wild on the African continent.
We probably imagine a majestic predator with a big mane.

The embedding of the word „lion“ is a vector with a certain length and direction within the embedding space.

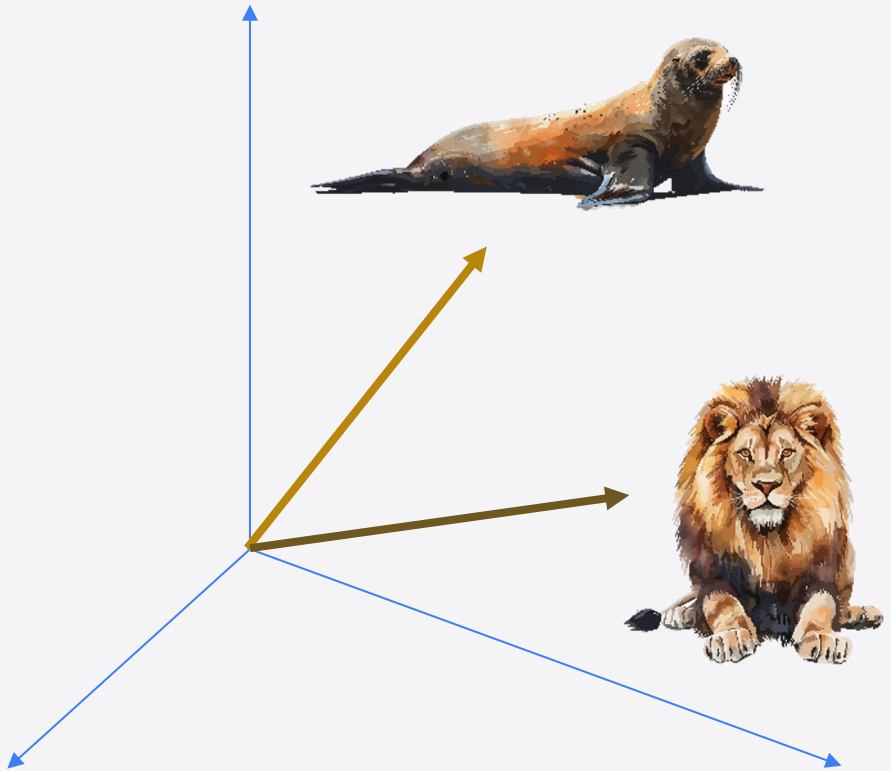


Context Is All You Need

Sea Lion

However, as soon we add the word „sea“ infront of „lion“ we imagine a totally different animal.

The same goes for the embedding. The attiontion mechanism needs to update the direction and length of the vector so that it represents the animal in question correctly.



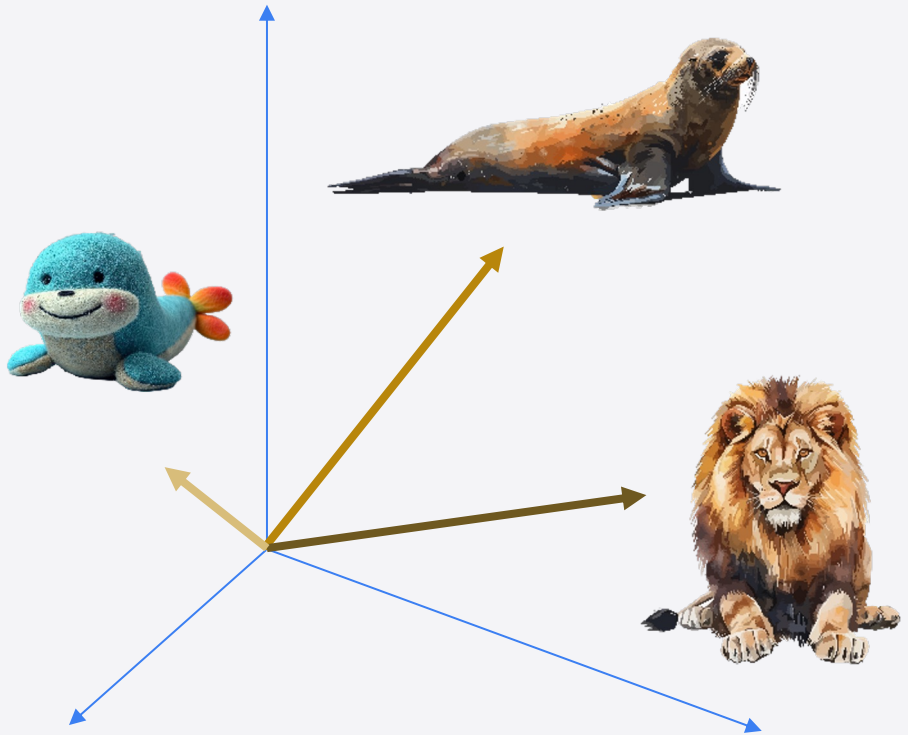
Context Is All You Need

Sea Lion Cuddly Toy

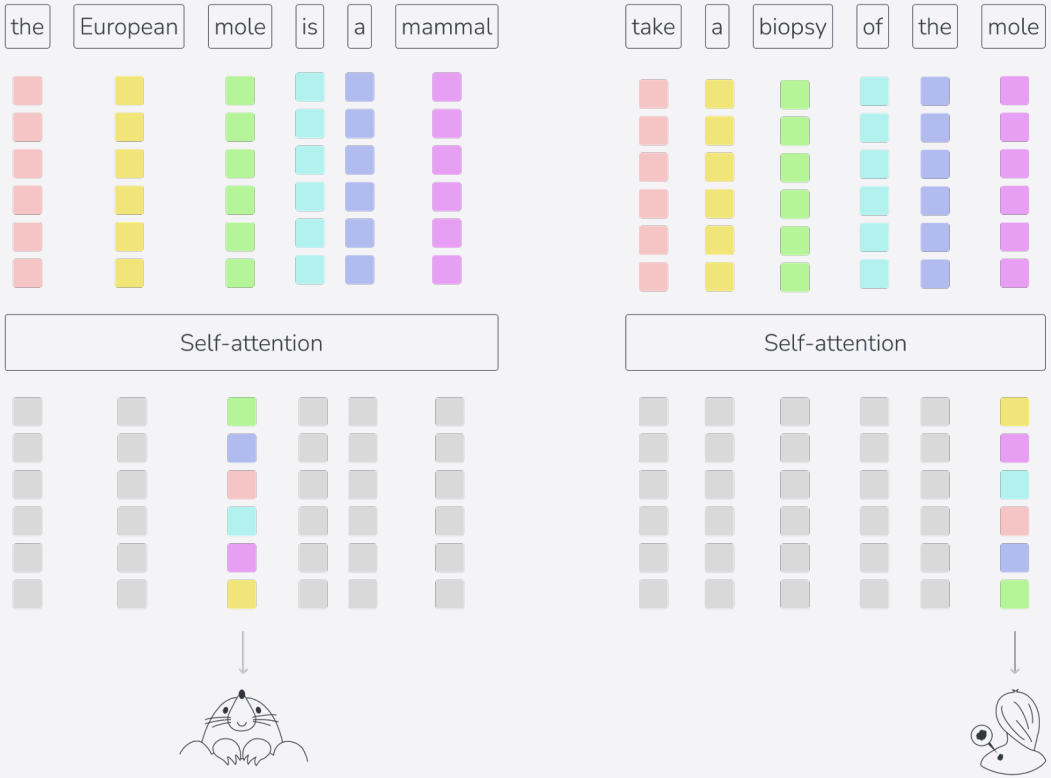
The context depends on more than just the immediate words to the left and right.

The embedding of „sea lion cuddly toy“ will certainly be very different of just „lion“.

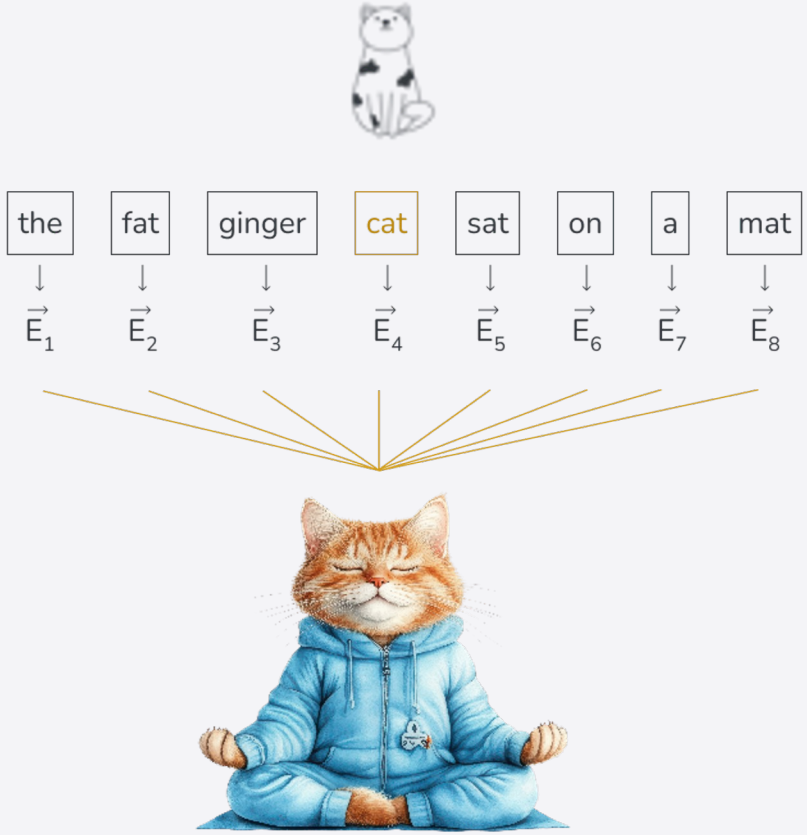
In order to achieve that the vector for „lion“ needs to attend to all the other words in the input (context size).



Self Attention



Self Attention



Attention! Queries, Keys and Values

The Attention Score

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

Q.....query matrix

K.....key matrix

V.....value matrix

d.....dimension of (smaller) query-key space

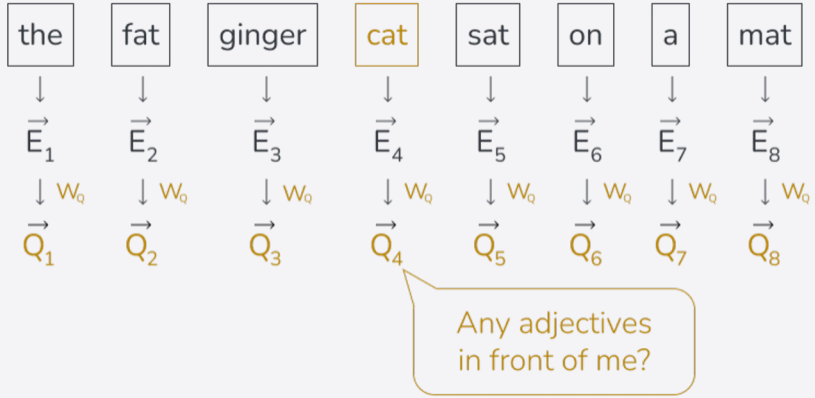
Attention! Queries, Keys and Values

Query

The query vector is obtained by multiplying a matrix full of trainable parameters with the embedding vector of that word (token).
Each word has its own query vector:

$$\overrightarrow{Q_i} = W_Q \overrightarrow{E_i}$$

It mappes the embedding vector to a much smaller dimensional space (e.g. 128 dimensions)



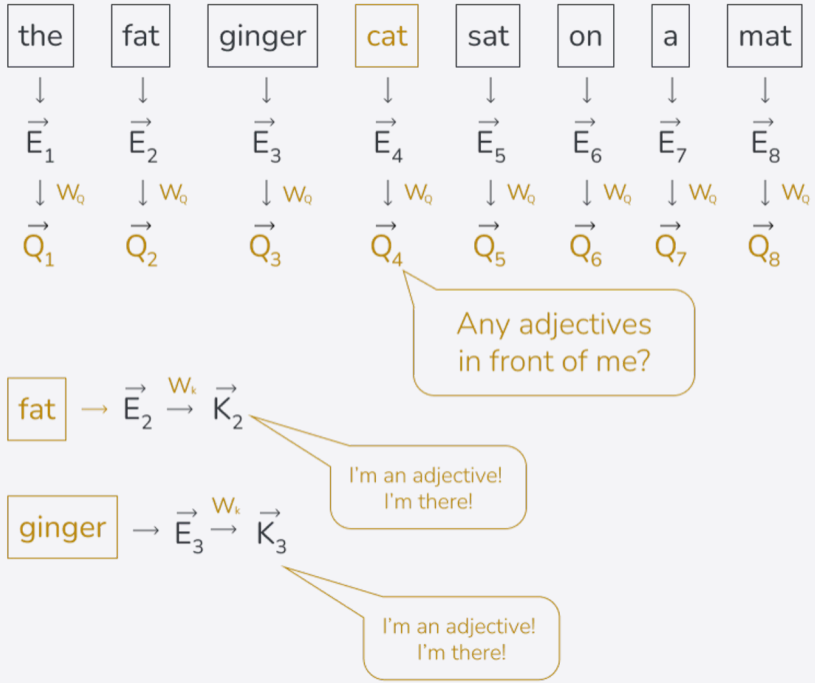
Imagine it like this:
One particular attention head is focussing on nouns. The query vector is like looking for labels with „adjective“ on them to better understand the noun.
In reality, this is much more abstract.

Attention! Queries, Keys and Values

Key

The key vector is also obtained by multiplying a matrix full of trainable parameters with the embedding vector of that word (token). Each word also has its own key vector: $\overrightarrow{K_i} = W_K \overrightarrow{E_i}$

Imagine it like this:
We are still dealing with the particular attention head that is focussing on nouns. The key is answering the question the query raised.

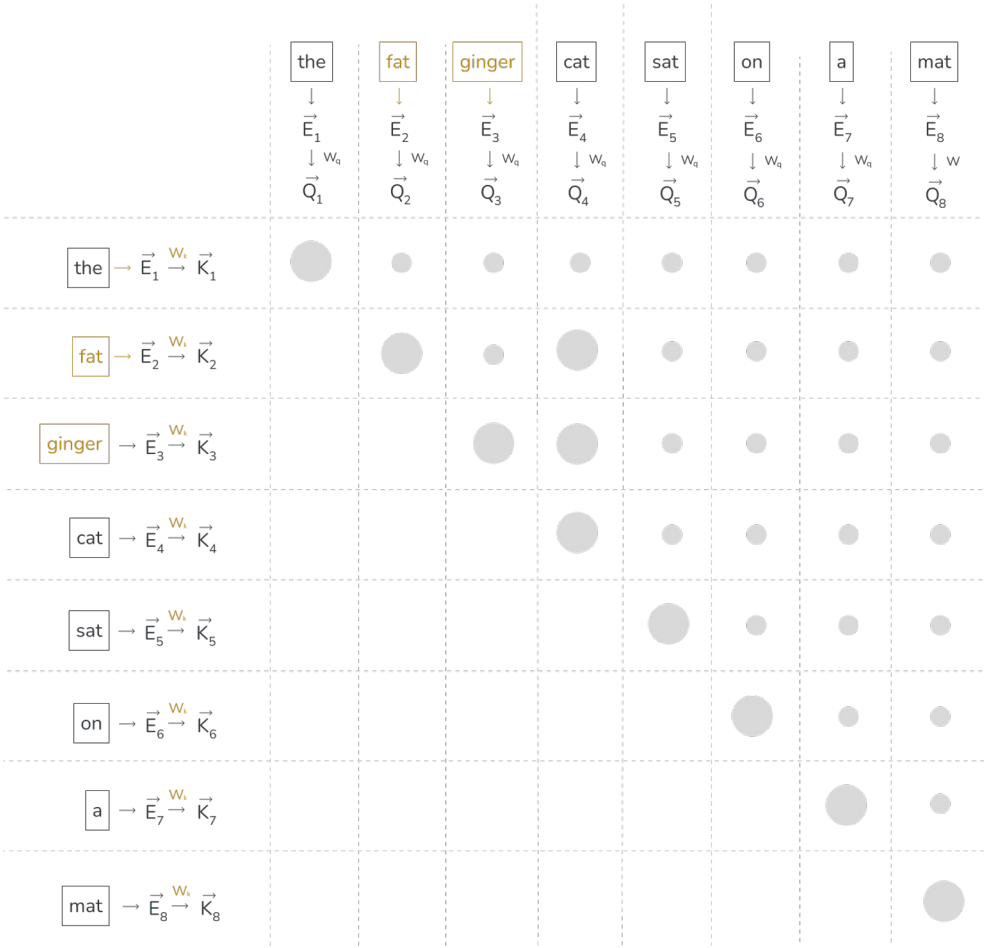


Attention! Queries, Keys and Values

Query – Key

Compute the dot product with each query-key pair, to determine how well the key matches the query. Where the queries and keys align, the dot product is larger.

Imagine it like this:
With our previous example, the dot product of the key vectors of „fat“ and „ginger“ with the vector of „cat“ yields the largest result. The embeddings of „fat“ and „ginger“ attend to „cat“.



Attention! Queries, Keys and Values

Attention

So far, we have determined which word is relevant to which other word (dot product of query and key).

We would like to use this as a score for how relevant every word is to update the meaning of other words.

For numerical stability the dot product is devided by the square root of the dimension of the query-key space.

To normalize the numbers to be between 0 and 1 we apply softmax.

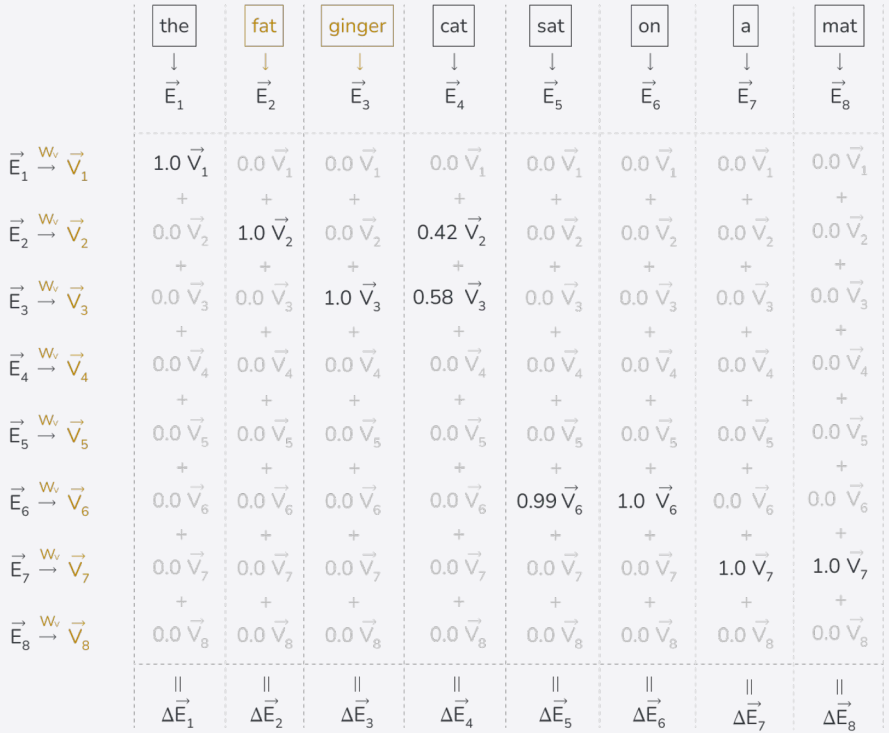
$$Attn(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Attention! Queries, Keys and Values

Value

The value matrix multiplied by the embedding of the preceding word results in the value vector.

$$\overrightarrow{V_i} = W_K \overrightarrow{E_i}$$



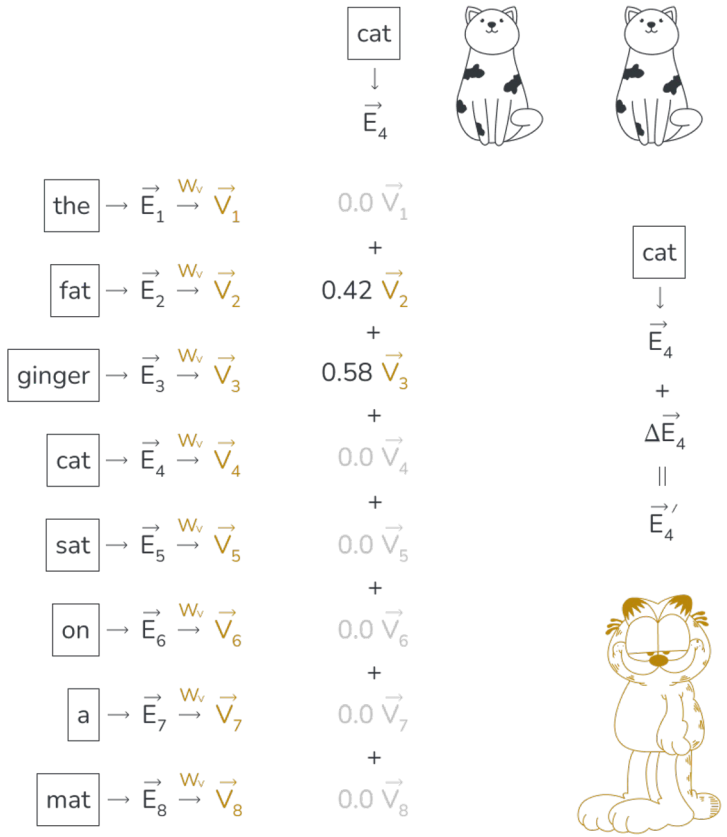
Attention! Queries, Keys and Values

Value

Eeach value vector is then multiplied by the corresponding weight for this word and the results summed up.

The result $\Delta\overrightarrow{E_i}$ is the change, that needs to be added to the origial embedding to get an updated, richer meaning.

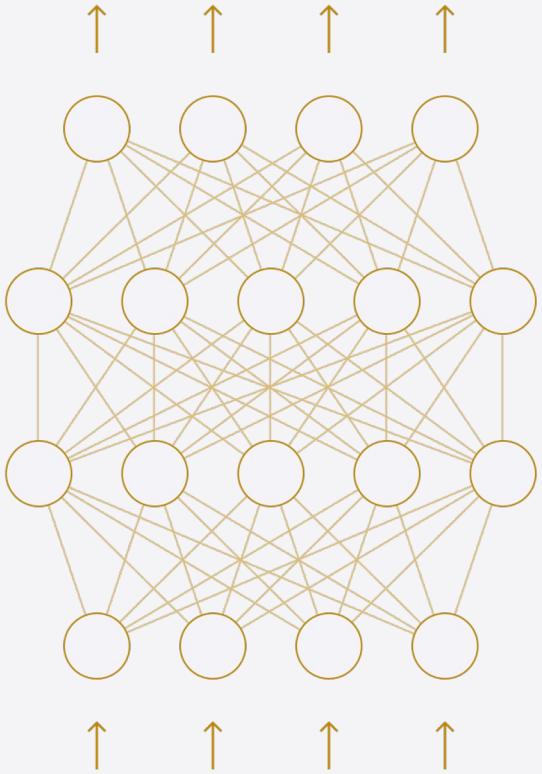
Since this happens to every word in the sequence, we end up with a set of more refined embeddings.



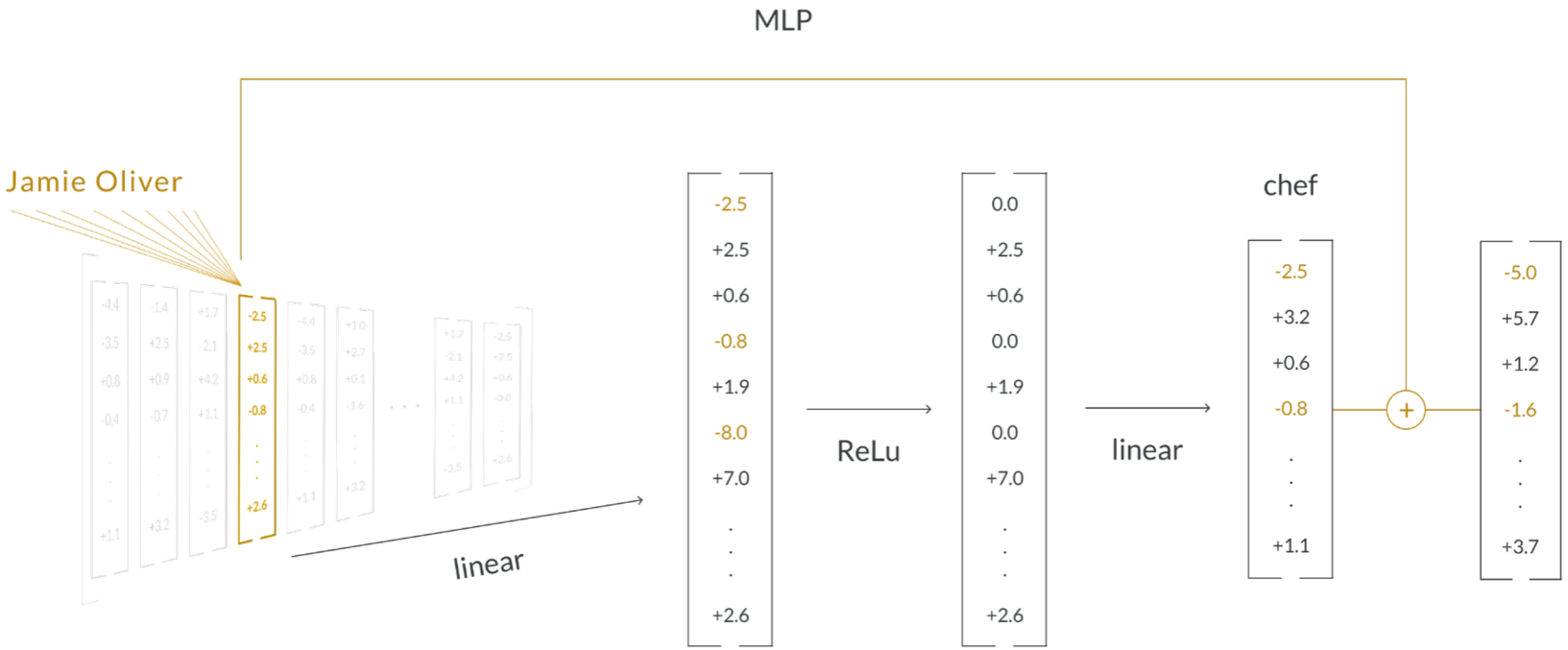
Feed Forward Network

Multi Layer Perceptron (MLP)

- Home to approx. 2/3s of all parameters
- This is where „knowledge“ is baked in
- Source of halluzinations
- Facts that are associated with input embeddings are added to the input embeddings
- Each embedding vector can be processed independently -> parallelization!



Feed Forward Network



Prompt Engineering

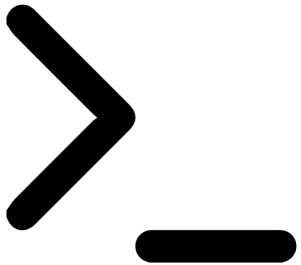
Definition:

The practice of designing inputs ("prompts") to guide the behavior and output of large language models (LLMs).

Goal:

Improve

- relevance
- accuaracy
- reliability (or reproducibility)
- controllability



Five Principles

Give Direction

Describe the desired style in detail

Specify Format

Define the structure and rules which should be followed

Provide Examples

Insert a diverse set of analogue cases

Divide Labour

Split tasks into multiple steps and chain them together for complex goals

Evaluate Quality

Conduct robust tests to determine which prompts improve the outcome

Role Prompting

Definition

Instructing the model to take on a specific role or persona.

Examples

- "You are a helpful legal assistant.“
- "Act as a professional proofreader.“

Effect

- Sets the tone and style
- Influences vocabulary and formality

Prompt Chaining

Definition

Connecting multiple prompts so that the output of one becomes the input of the next.

Example

1. P1: Summarize this legal clause
2. P2: Based on this summary, list 3 follow-up questions

Effect

- Enables complex workflows
- Decomposes task

LangChain

LangChain is an open-source framework for building applications with large language models. It provides abstractions for prompt templates, chains, memory, agents and tool integration.

Documentation

Meta Prompting

Definition

Prompts that instruct the model on how to behave or how to generate prompts themselves.

Examples

- “Generate a prompt that would help someone learn about contract law.“
- “List questions that would clarify this legal clause.“

Effect

Self-reflection and model steering

Evaluation

Or: how to size up a Llama

Measuring up

Huggingface evaluate library: https://huggingface.co/docs/evaluate/

Different types of evaluation, depending on
- Goals
- Datasets
- Models

Huggingface currently offers/supports
- Metrics
- Comparisons
- Measurements

Metrics

Generic

Metrics that can be applied to many different tasks and datasets.

- accuracy, precision, recall
- F1 score
- Perplexity

Task specific

Tasks such as Machine Translation or Summarisation have specific metrics.

- BLEU
- ROGUE

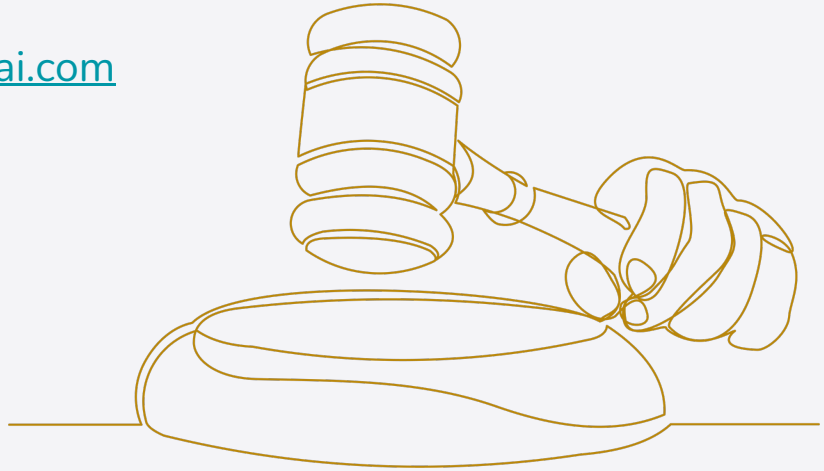
Dataset specific

Datasets, which are used as benchmarks use specific metrics.

- GLUE
- SQuAD

LLM as a Judge

Evidently evaluation library: evidentlyai.com



Popular method

- Less costly than human evaluation
- Scalable
- More meaningful than metrics

How it works

- Uses LLM with evaluation prompts
- Can test for bias, tone, toxicity, hallucinations etc.
- Can return scores, labels or descriptive judgements

STAY IN TOUCH

EuroCC Austria

@eurocc-austria

eurocc-austria.at

THANK YOU

EURO
AUSTRIA

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia