Applied Text Analytics (F20/F21AA)

Transformers II

Heriot Watt University , Dubai Campus

Dr. Radu Mihailescu

Previous Lecture

Building blocks of a Tranformer

- Encoder-Decoder Architecture
- Word Encoding
  - Context Encoding
  - Positional Encoding
- Attention
  - Self attention
  - Multi headed attention
  - Encoder Decoder attention

Today

More on Self attention & context encoding
Positional Encoding

Transfer Learning
Overview on Foundation Models & ChatGPT

Transformer Applications (Hugging Face)

Practical session:
- Using pipeline (Hugging Face)
- Translating Text with Transformers (tensorflow tutorial)

Transformers

Are Deep Neural Network models that excel in capturing relationships and attention in text

Unlike RNNs Transformers do not take input as a sequence and use positional encoding to retain positional information

Operations in Transformers can be easily parellized

Transformer models are the base of many foundation models that are trained on huge corpra for months. Those models can be fine tuned to create custom AI model in few hours

Transformers (original architecture)

- No recurence (sequential processing)
- Rely on self-attention
- State of the art in most NLP tasks and now more success with multimodal data (vision & speech)

The Transformer Architecture

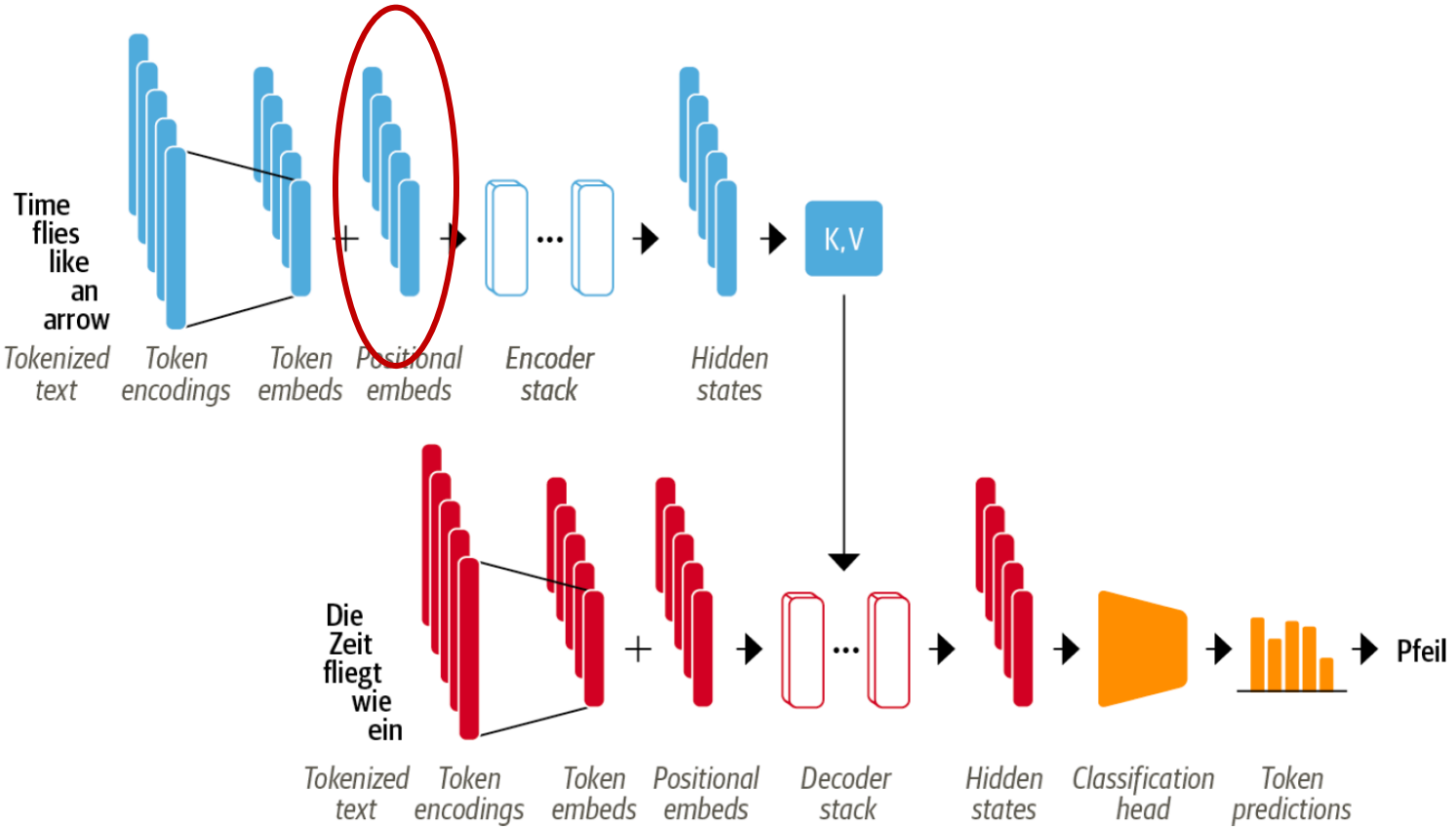
Encoder

- Converts an input sequence of tokens into a sequence of embedding vectors, often called the hidden state or context **(machine readable representation)**

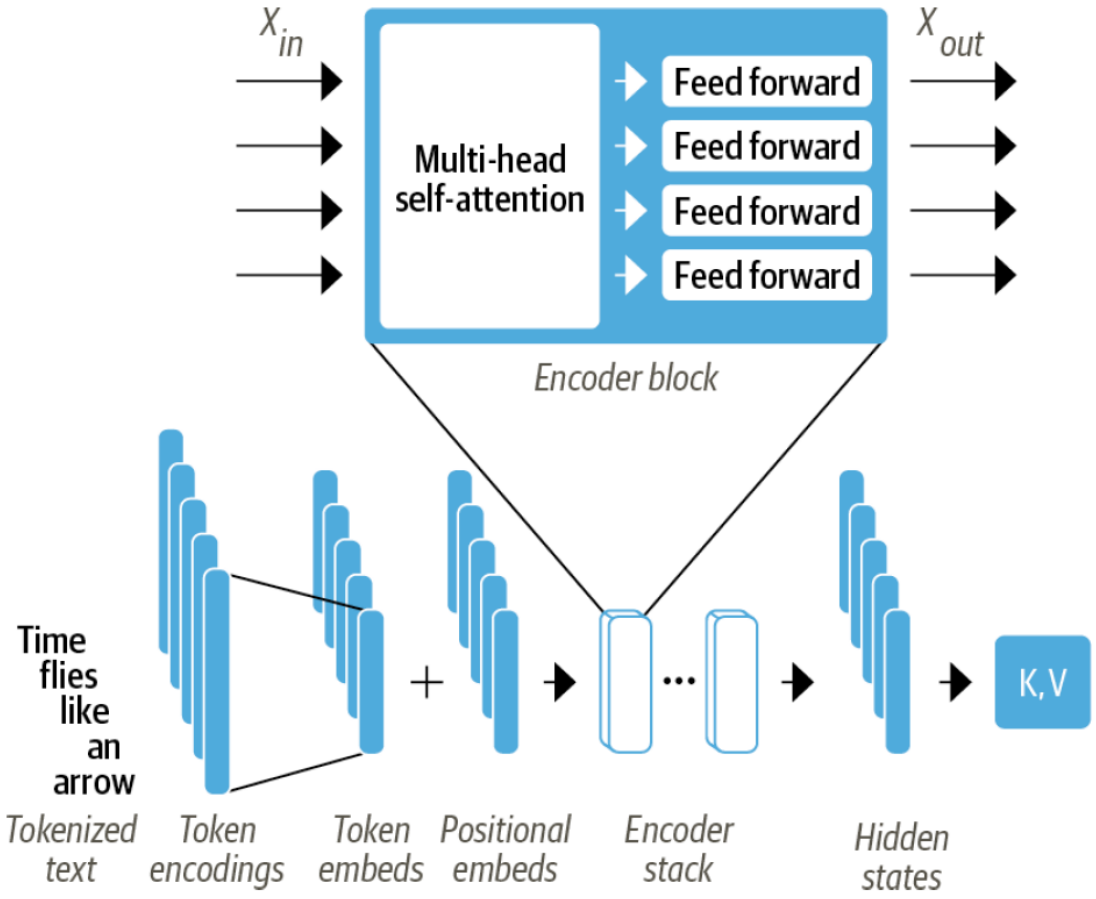
Decoder

- Uses the encoder's hidden state to *iteratively* generate an output sequence of tokens, one token at a time **(generate human readable sequence)**

The Transformer Architecture (original architecture)



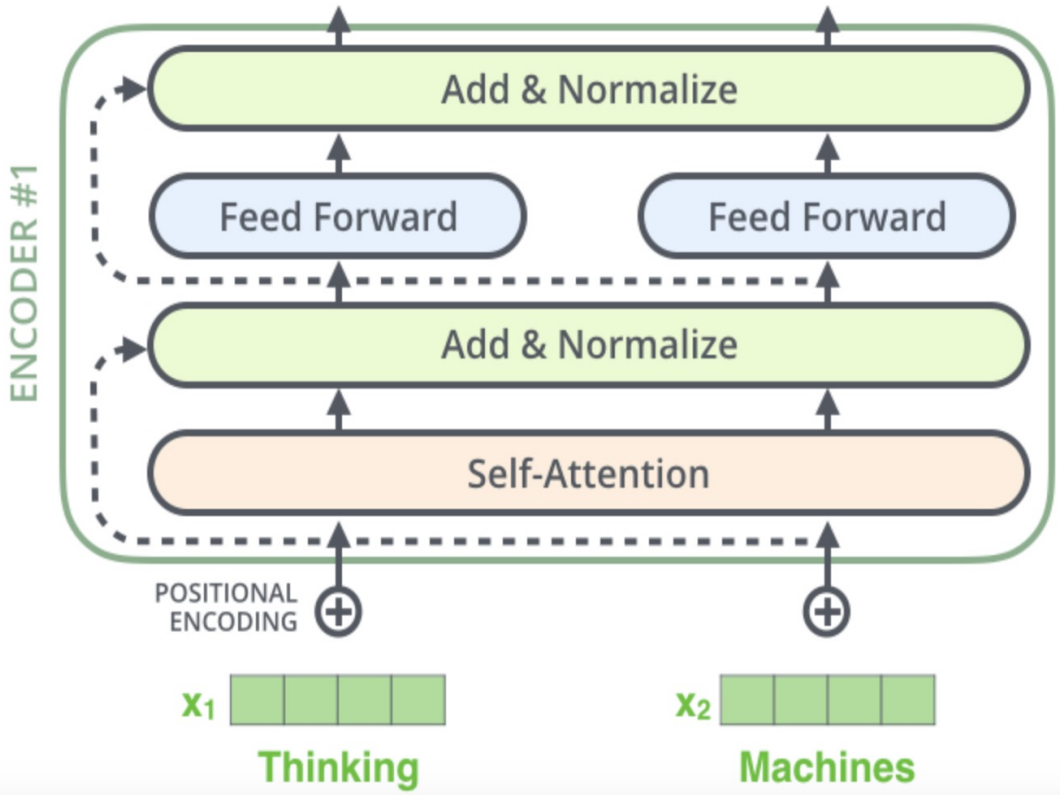
Transformer Encoder



Each encoder layer receives a sequence of embeddings and feeds them through sublayers:

- A multi-head self-attention layer
- A fully connected feed-forward layer that is applied to each input embedding

Transformer Encoder



Each encoder layer receives a sequence of embeddings and feeds them through sublayers:

- A multi-head self-attention layer
- A fully connected feed-forward layer that is applied to each input embedding

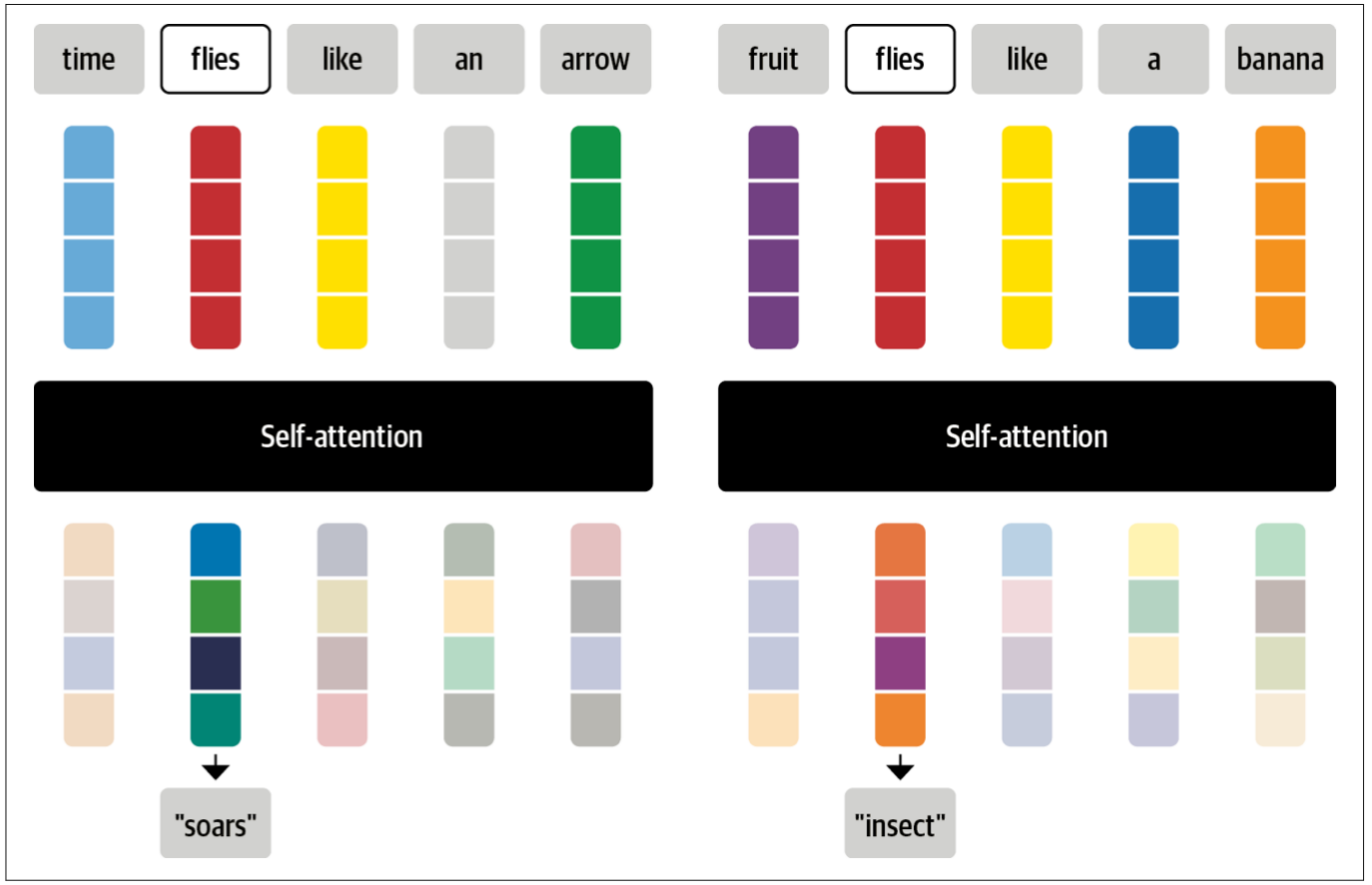
Sublayers also uses skip connections and layer normalization, which are standard tricks to train deep neural networks effectively.

Self-Attention Intuition

- ***Attention*** is a mechanism that allows neural networks to assign a different weights “attention” to each element in a sequence

- Instead of using a fixed embedding for each token, we can use the whole sequence to compute a ***weighted average*** of each embedding.

From raw to contextualized word embedding

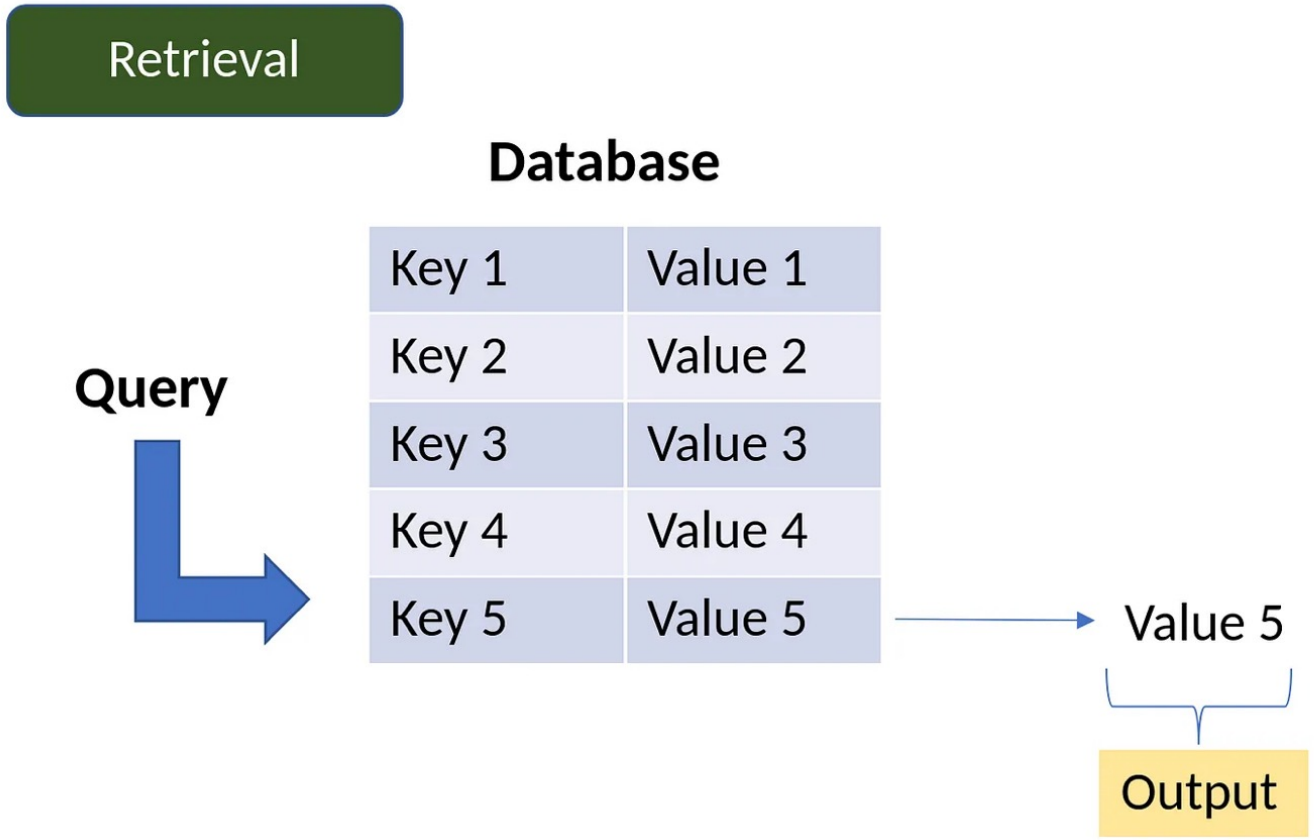


Embedding contains information about entire sequence

Self-Attention using query, key & value

- Step1: Project each token embedding into three vectors called query, key, and value
- Step2:Compute attention scores.
- Step3: Compute attention weights.
- Step4:Update the token embeddings.

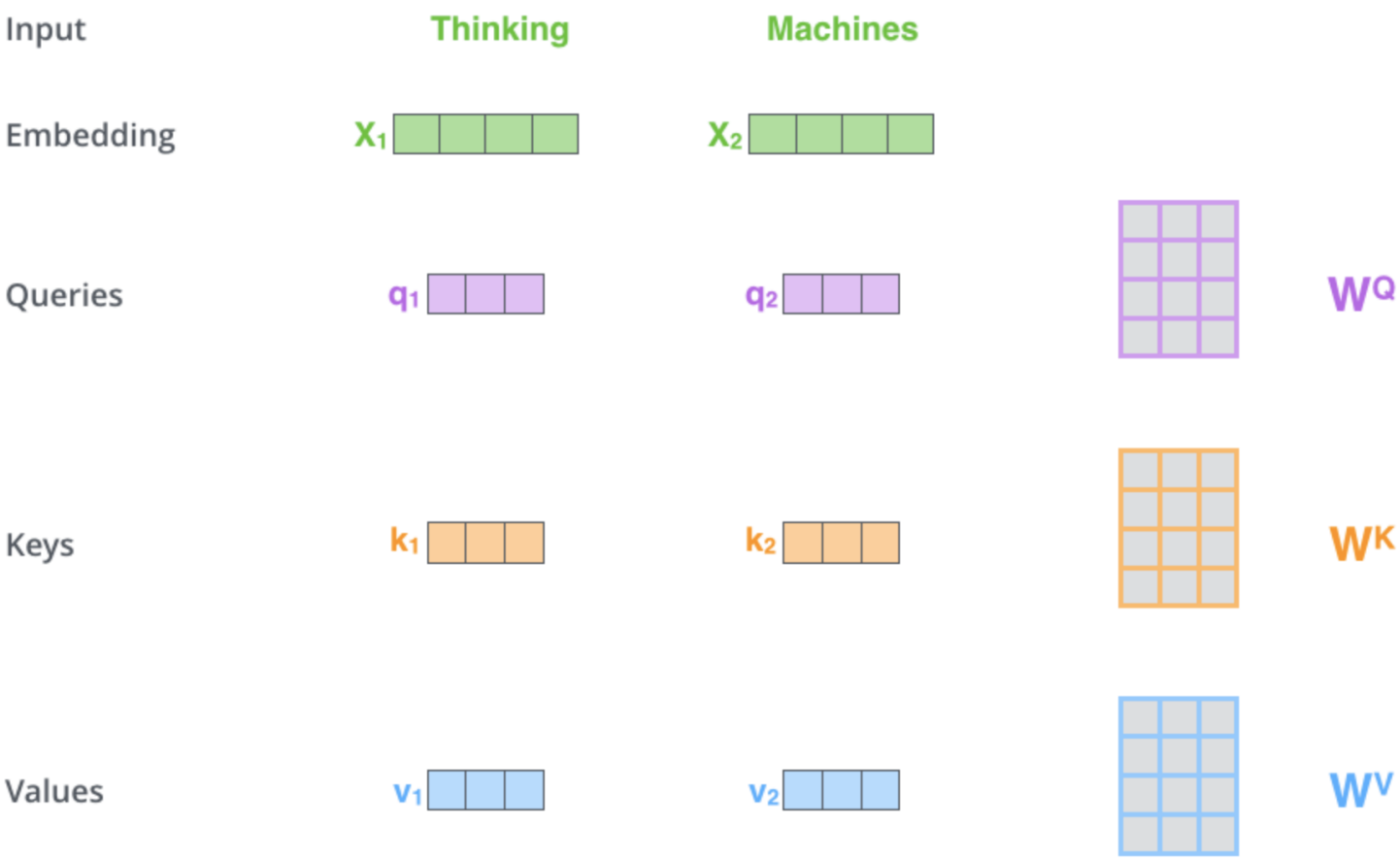
Query, Key and value .. from Information Retrieval



https://towardsdatascience.com/all-you-need-to-know-about-attention-and-transformers-in-depth-understanding-part-1-552f0b4

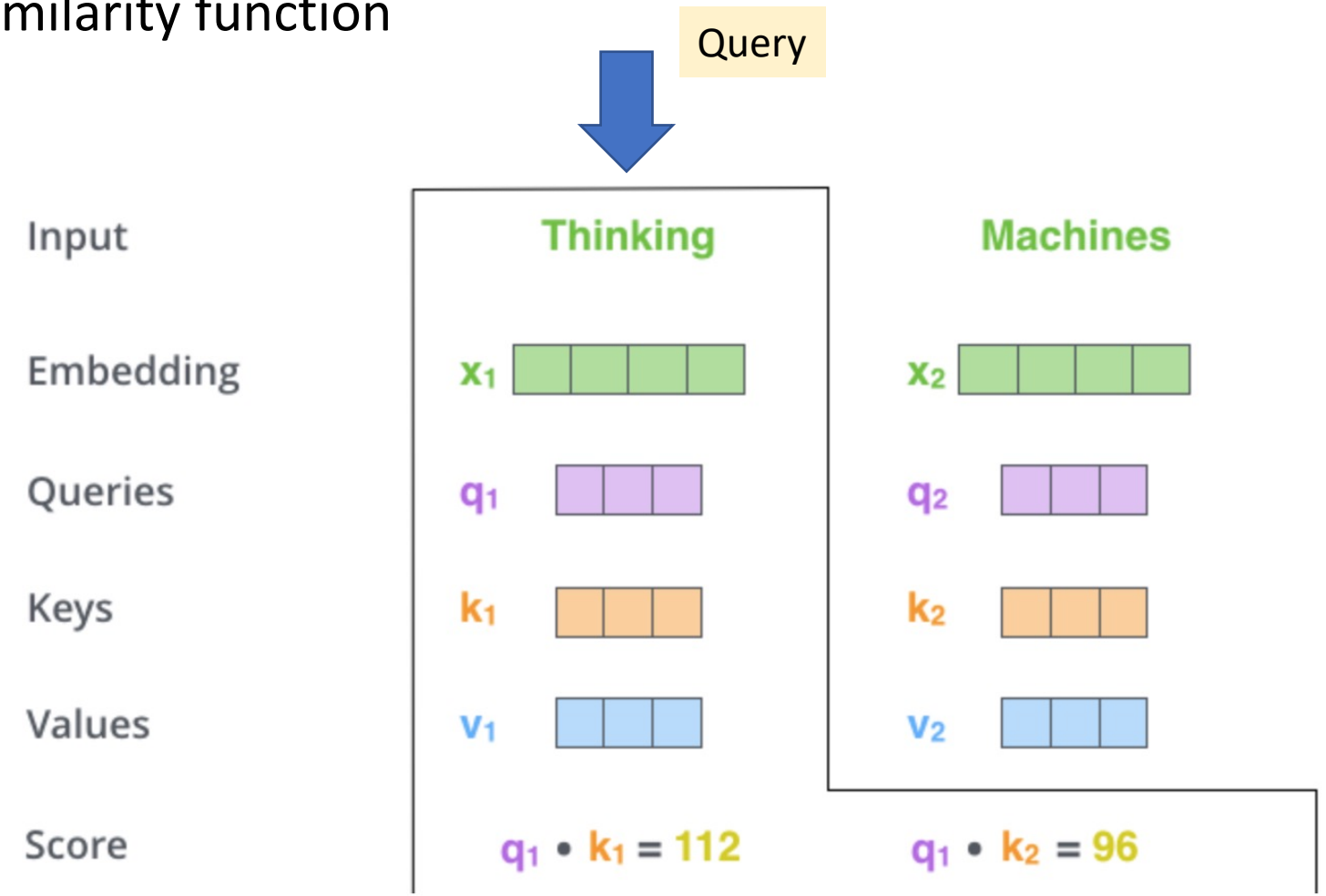
Self Attention using query, key & value

- Step1: Project each token embedding into three vectors called query, key, and value



Self Attention using query, key & value

- Step2:Compute attention scores= how much the query and key vectors relate to each other using a similarity function

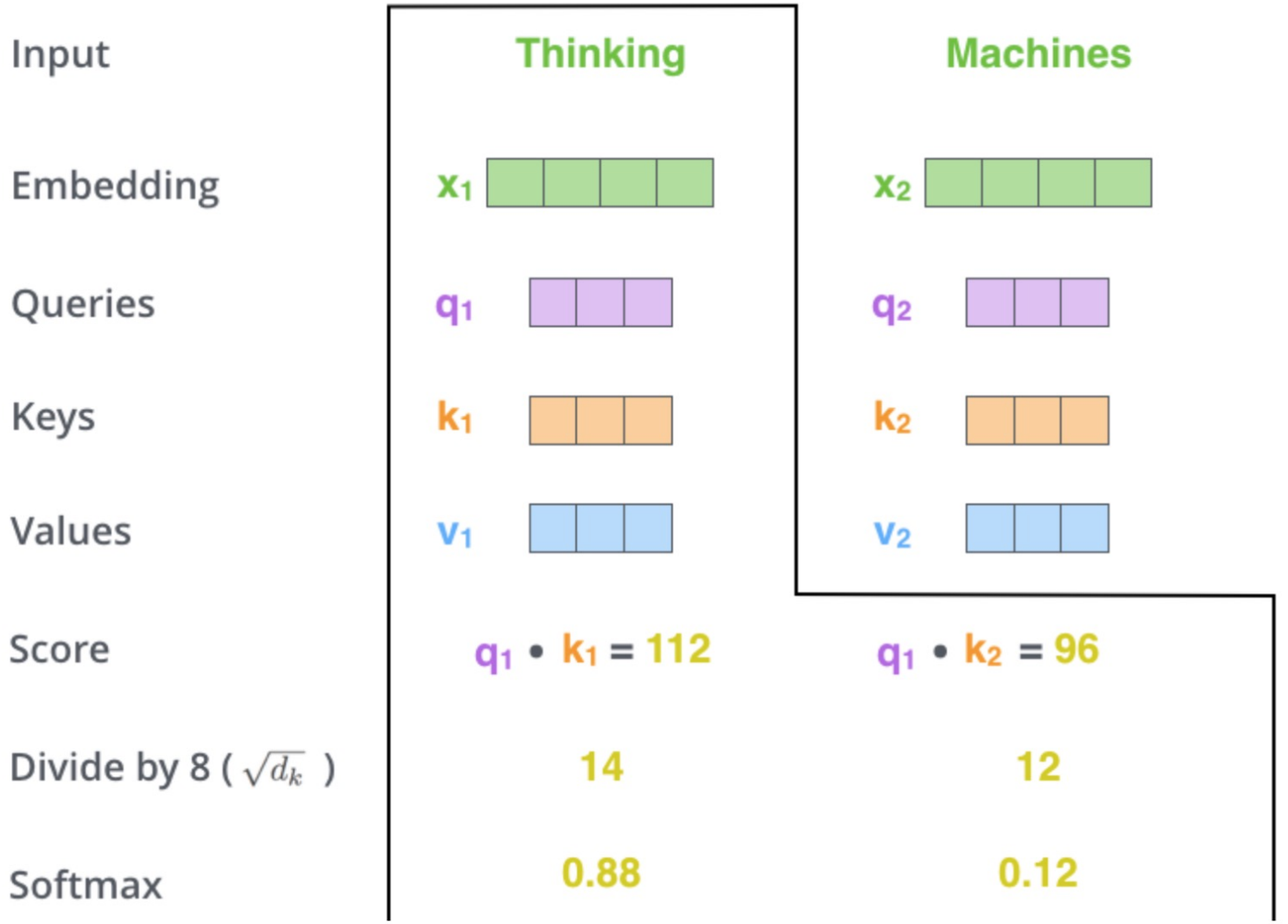


Self Attention using query, key & value

Step3:

Compute attention weights

= divide by scaling factor and normalize to probabilities using softmax

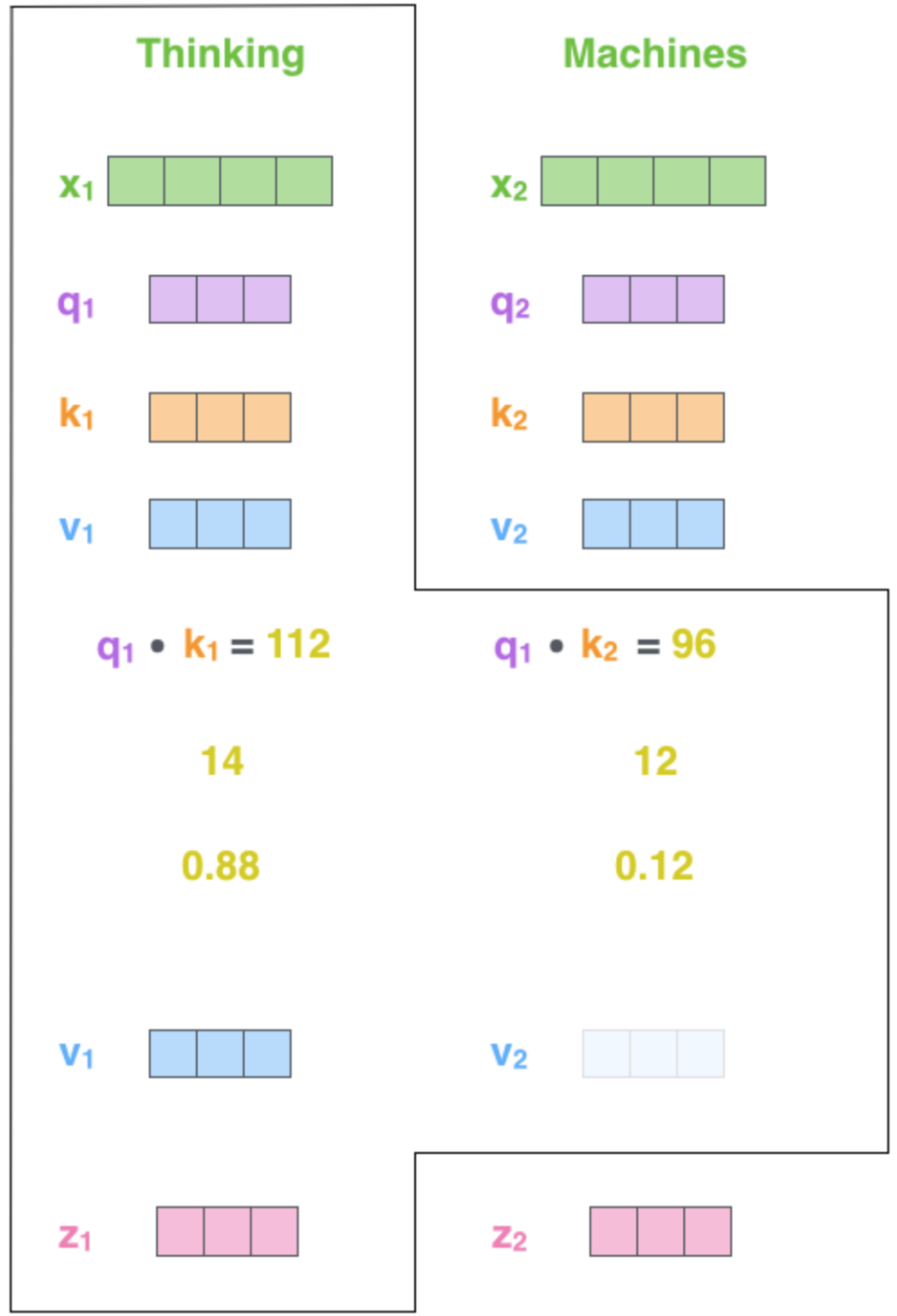


Self Attention using query, key & value

Step4:

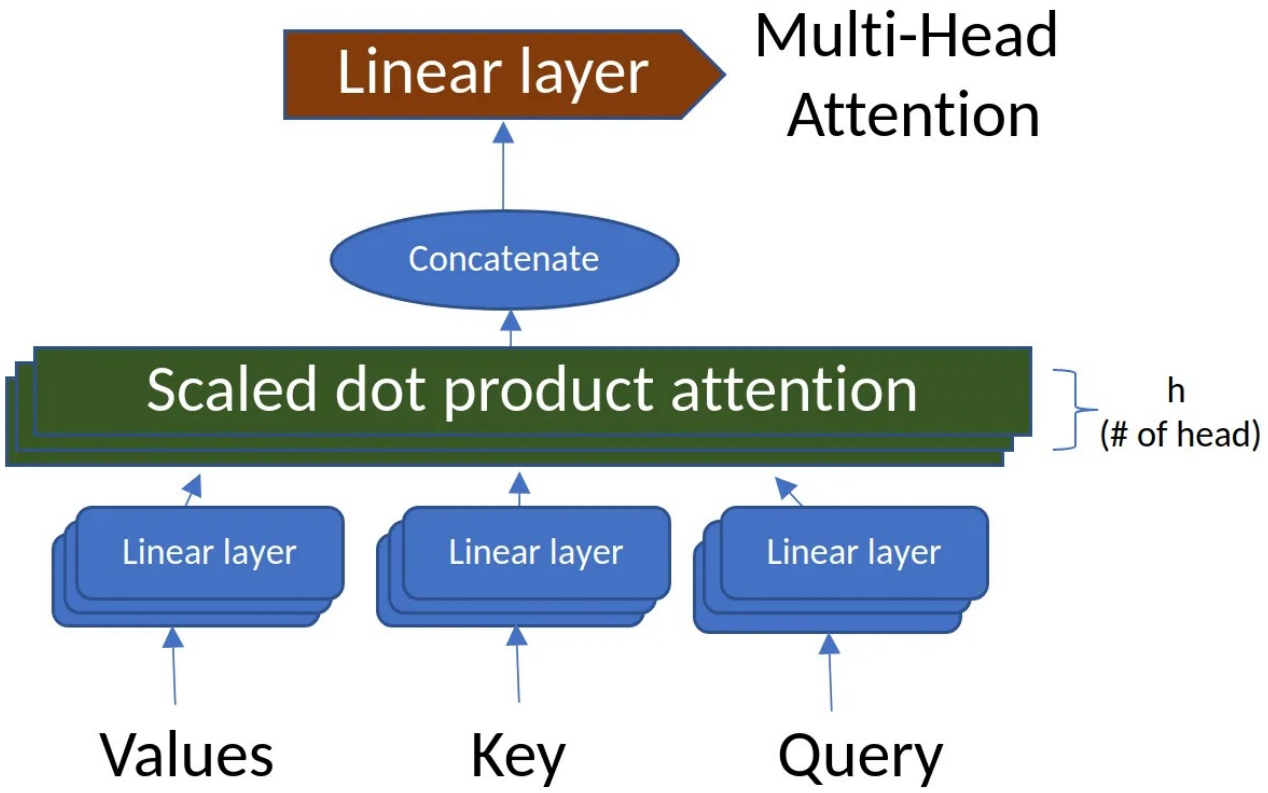
Update the token embeddings=

multiply attention weights with value vectors to generate an updated embedding representation

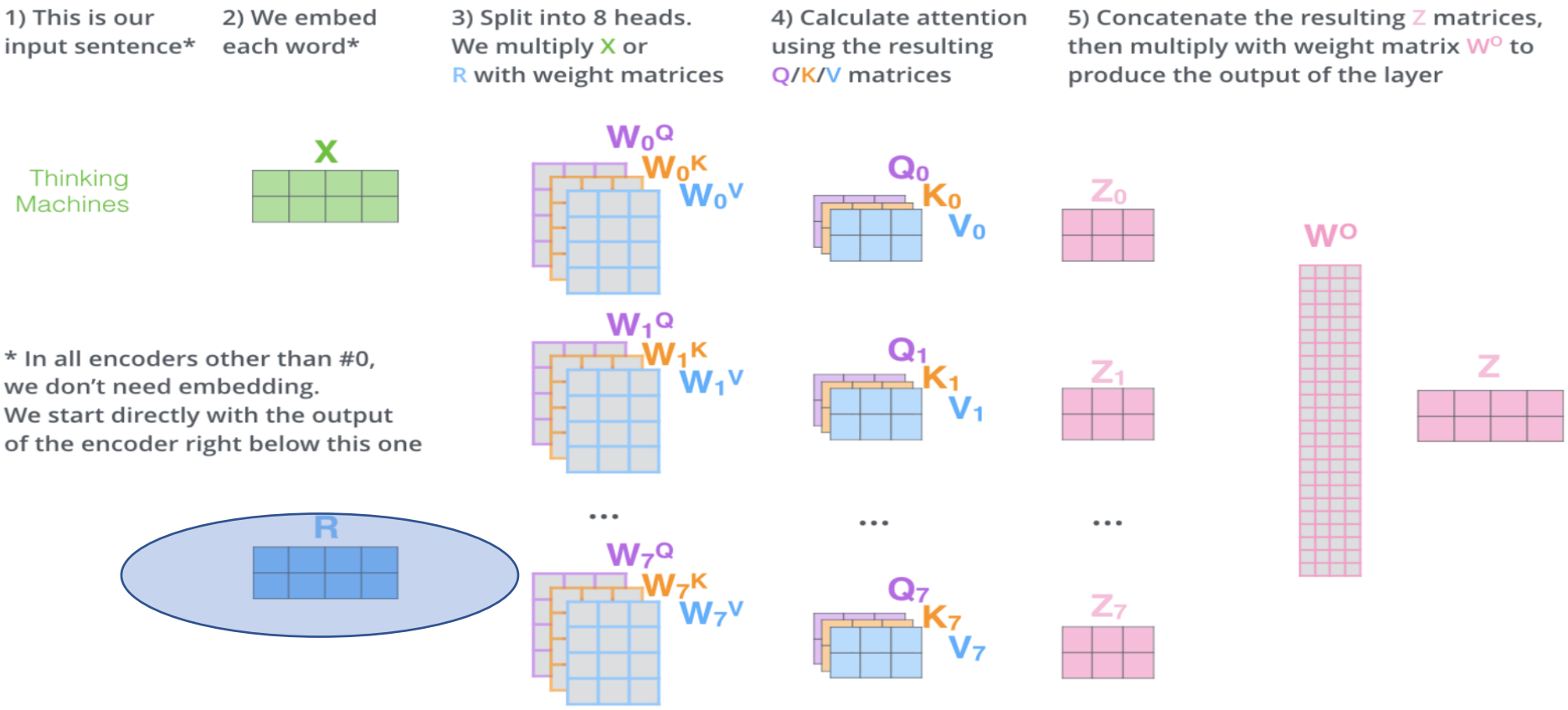


$$attention(q,k,v) = \sum_i similarity(q,k_i) * v_i$$

Adding on mulyi-head attention



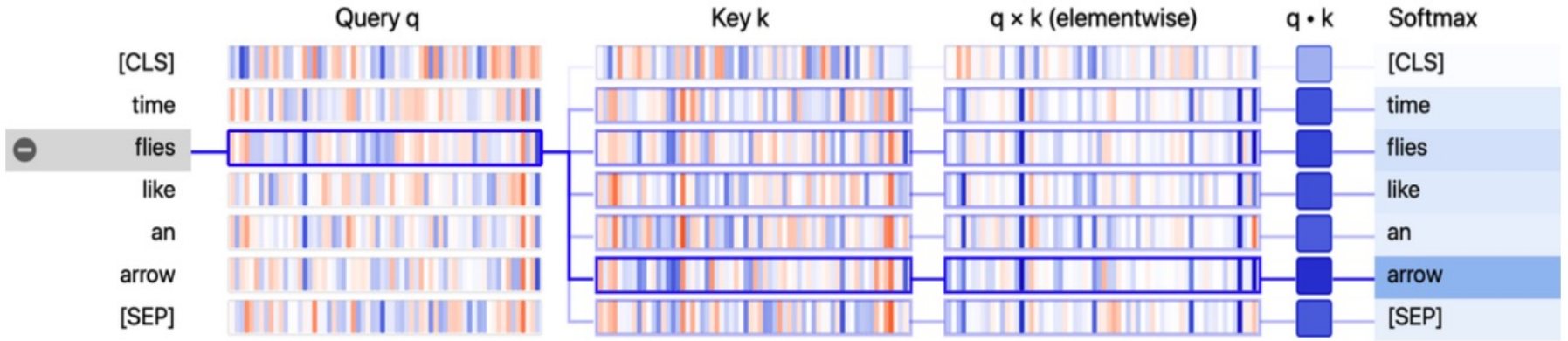
Encoder self-attention (putting it all together)



$$attention(q,k,v) = \sum_i similarity(q,k_i) * v_i$$

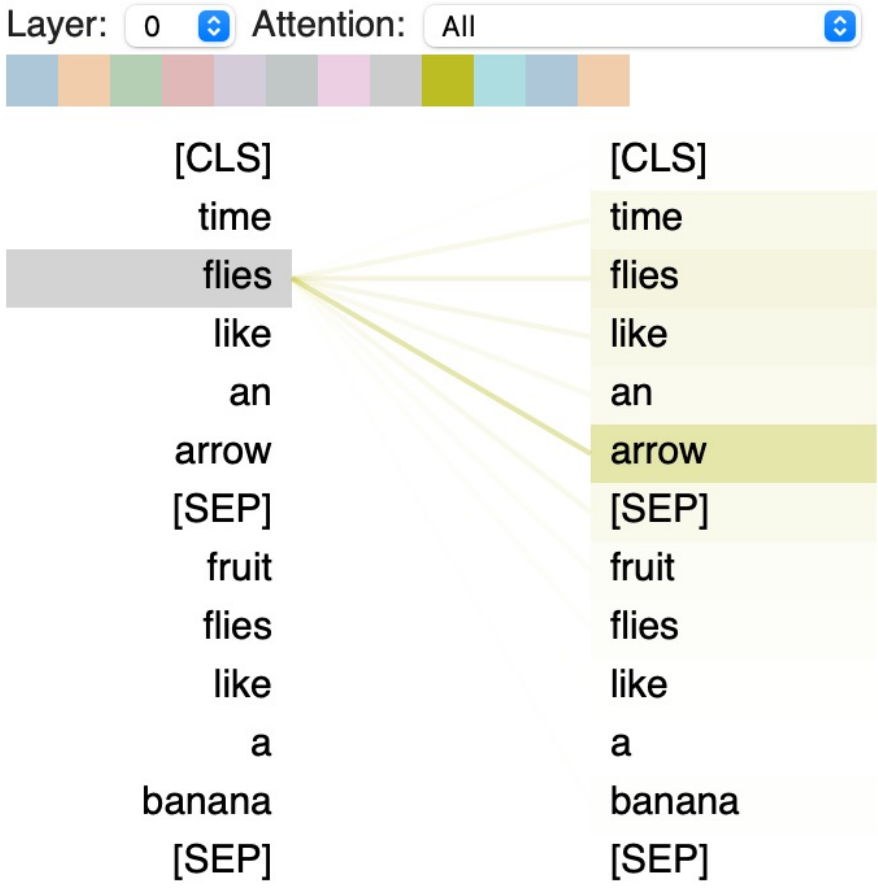
Visualizing attention weights

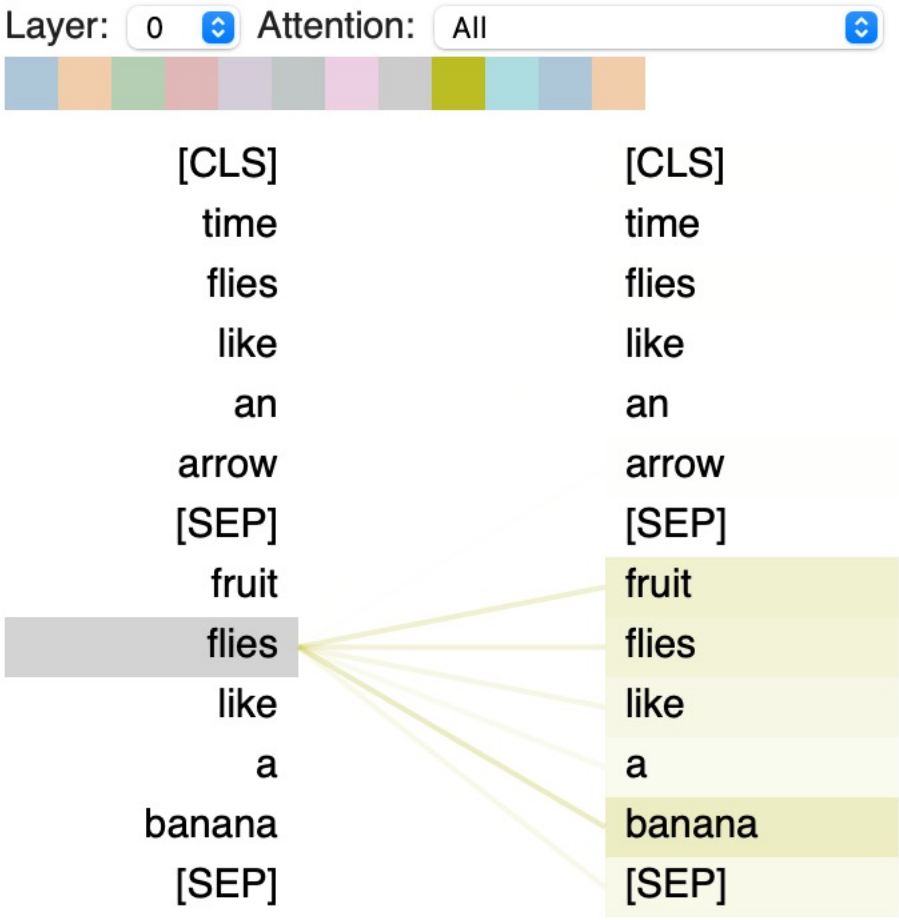
- BertViz is an interactive tool for visualizing attention in Transformer language models such as BERT, GPT2, or T5.



Run the code
https://colab.research.google.com/github/nlp-with-transformers/notebooks/blob/main/03_transformer-anatomy.ipynb

Visualizing attention weights





Positional Embedding

10,000 set by the authors of Attention Is All You Need.

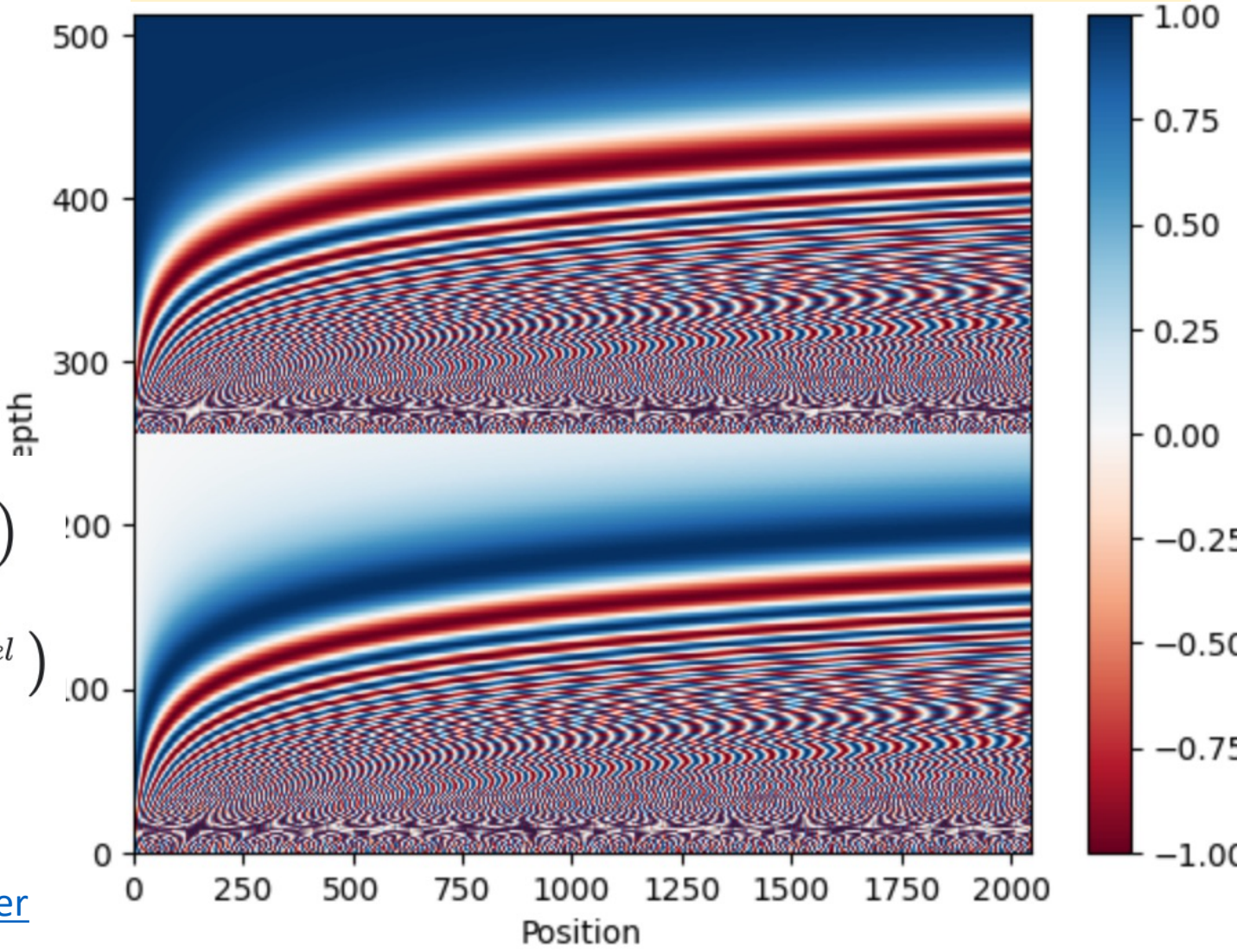
d= Dimension of the output embedding space

The transformer contains no recurrence, for the model to make use of the order of the sequence

we must inject some information about the relative or absolute position of the tokens in the sequence.

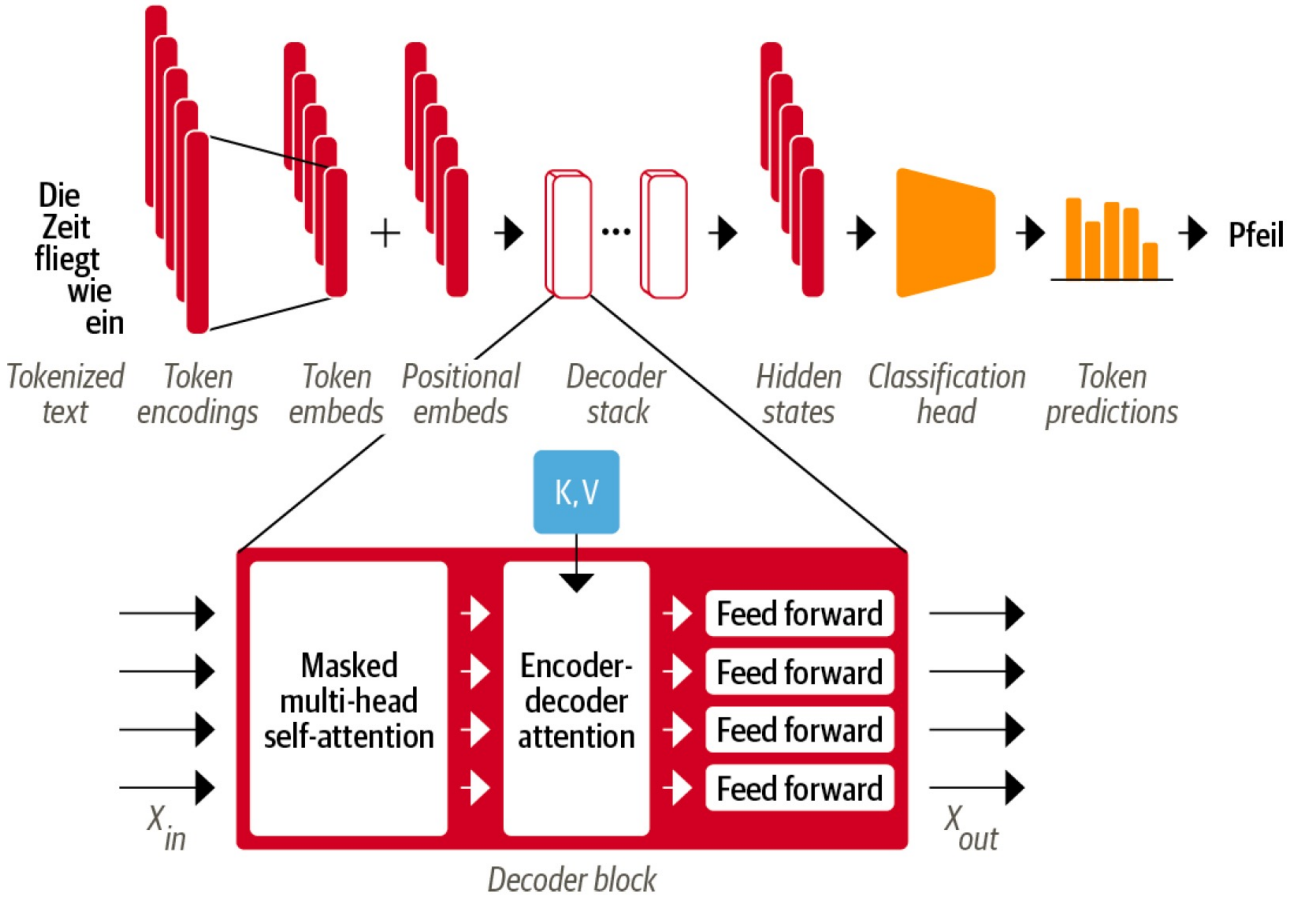
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$



https://www.tensorflow.org/text/tutorials/transformer

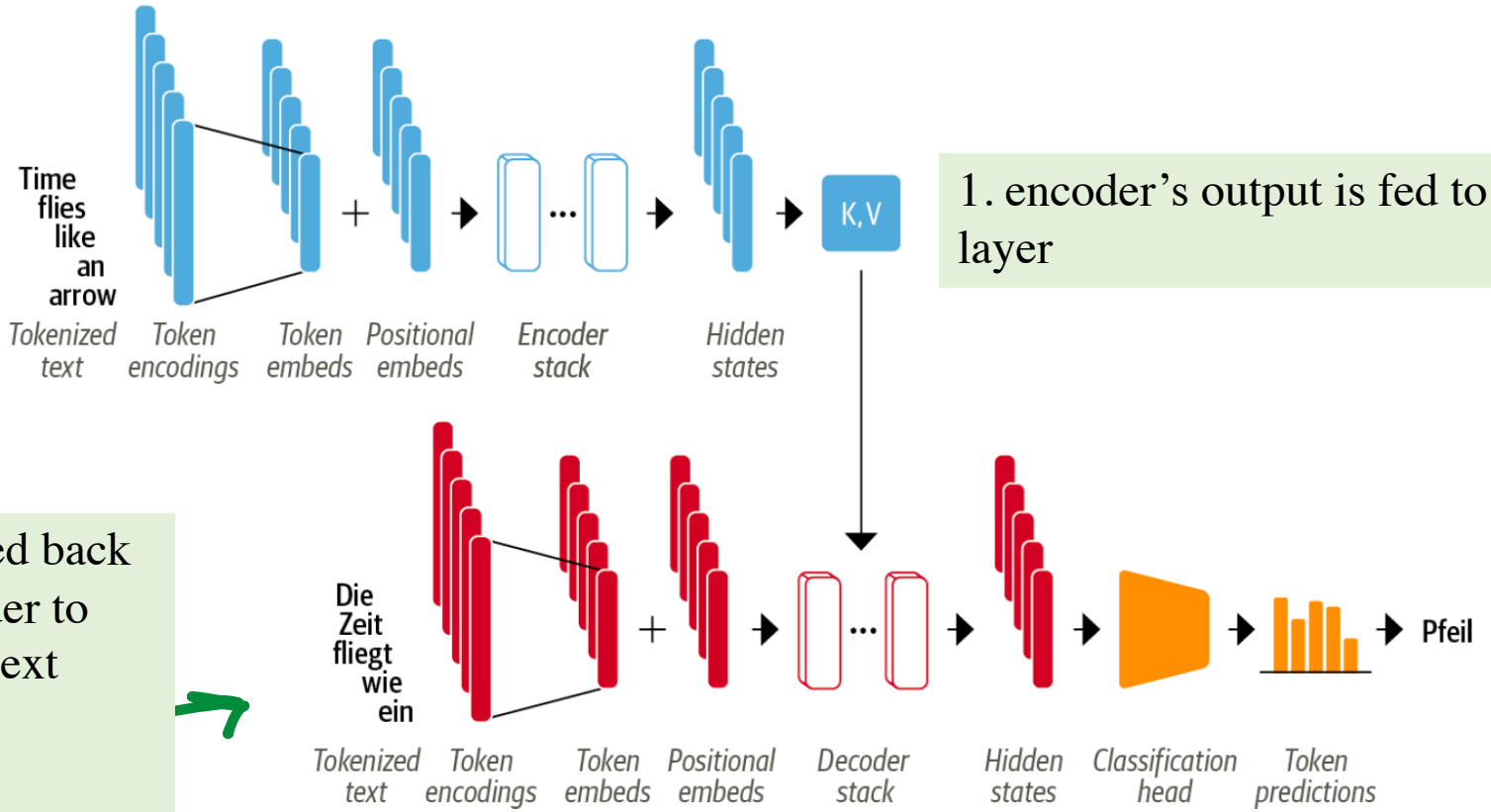
Transformer Decoder



Masked multi-head self-attention layer Ensures that the tokens we generate at each timestep are only based on the past outputs

The “Encoder-Decoder Attention” creates its Queries matrix from the layer below it and takes the Keys and Values matrix from the output of the encoder stack.

The Transformer Architecture

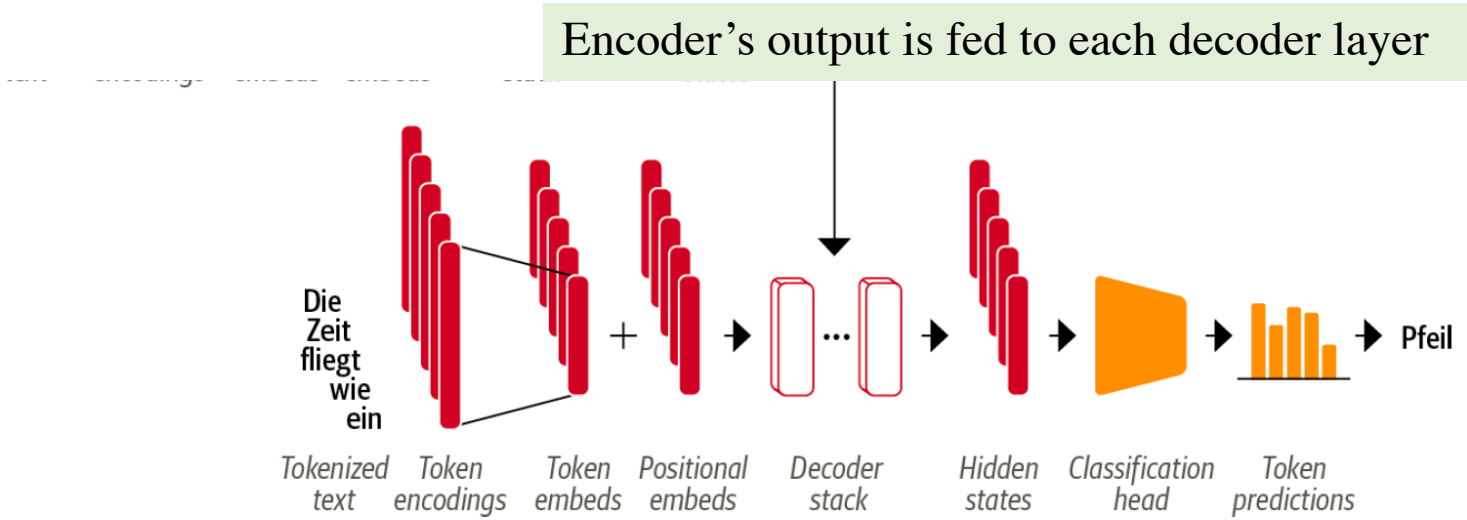


1. encoder's output is fed to each decoder layer

2. decoder then generates a prediction for the most probable next token in the sequence.

3. Output is fed back into the decoder to generate the next token.
Repeat until
a (EOS) token is reached.

The Transformer Architecture- Training Decoder side

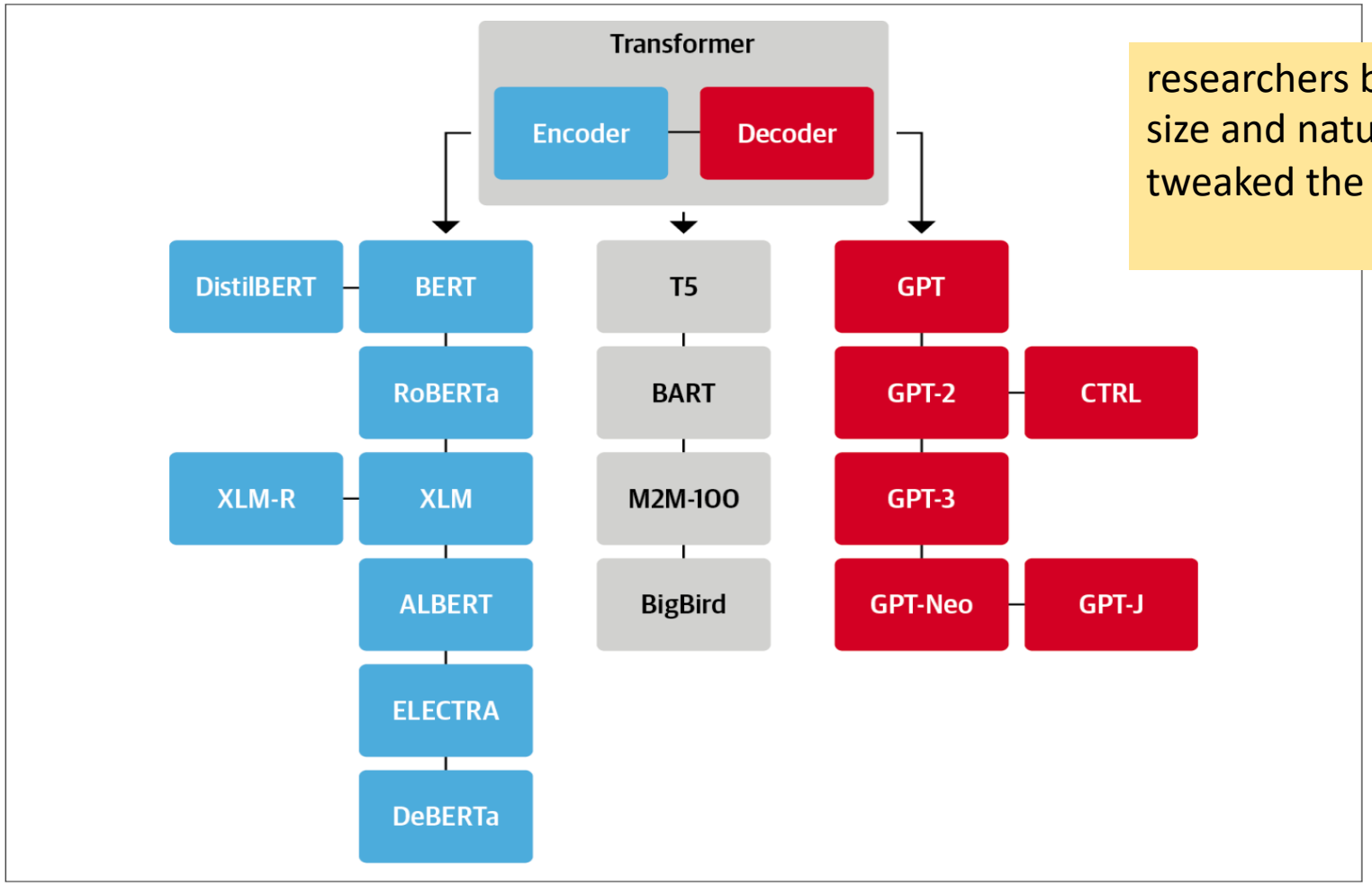


<SOS>	Die
Die	Zeit
Die Zeit	fliegt
Die Zeit fliegt	wie
Die Zeit fliegt wie	ein
Die Zeit fliegt wie ein	Pfeil
Die Zeit fliegt wie ein Pfeil	<EOS>

Key Transformer models

- BERT (Google)
- XLNet (Google/CMU)
- RoBERTa (Facebook)
- DistilBERT (HuggingFace)
- CTRL (Salesforce)
- GPT-2 (OpenAI)
- ALBERT (Google)
- Megatron (NVIDIA)

Key Transformer models (p.80-84)



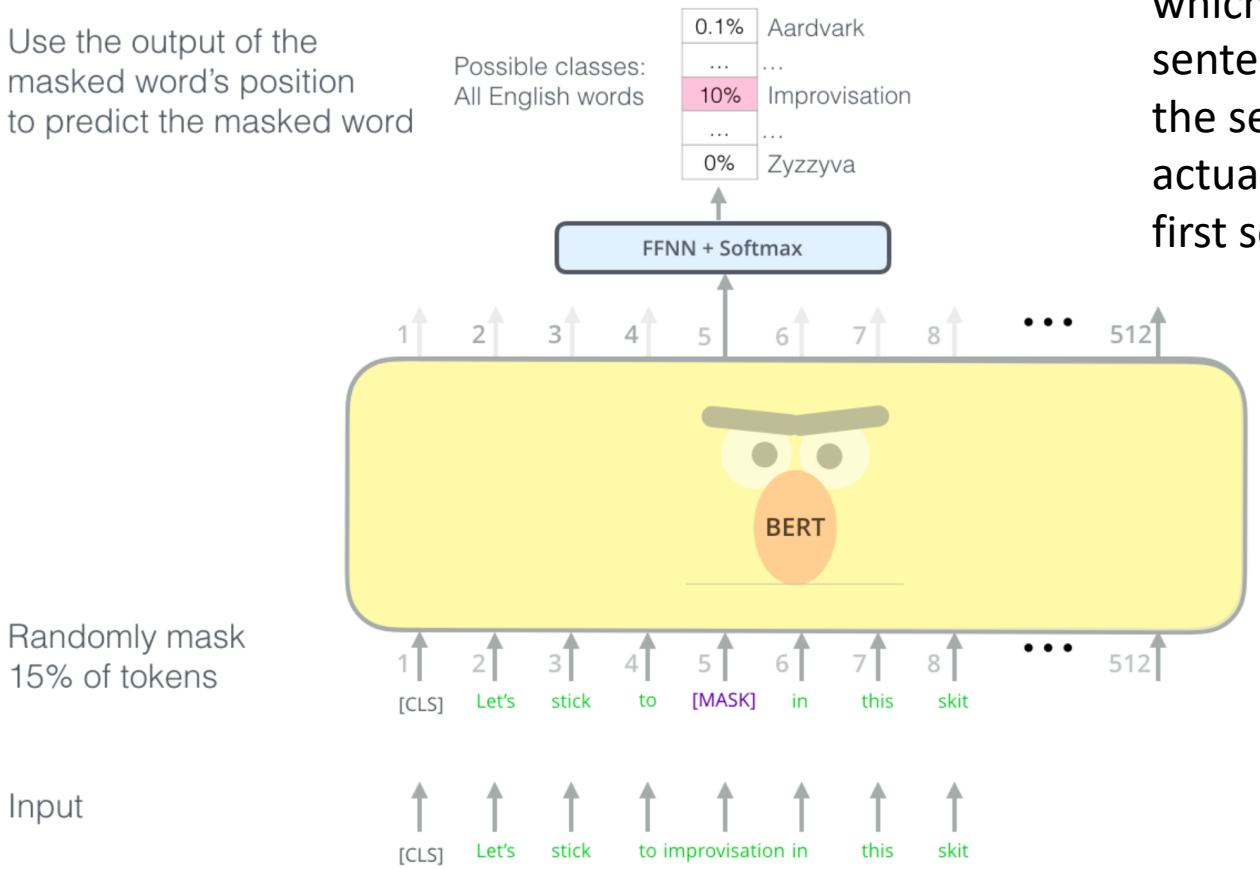
researchers built models on various datasets of different size and nature, used new pretraining objectives, and tweaked the architecture to further improve performance

Transformer evolution

Model	Layers	Width	Heads	Params	Data	Training
Transformer-Base	12	512	8	65M		8x P100 (12 hours)
Transformer-Large	12	1024	16	213M		8x P100 (3.5 days)
BERT-Base	12	768	12	110M	13 GB	
BERT-Large	24	1024	16	340M	13 GB	
XLNet-Large	24	1024	16	~340M	126 GB	512x TPU-v3 (2.5 days)
RoBERTa	24	1024	16	355M	160 GB	1024x V100 GPU (1 day)
GPT-2	48	1600	?	1.5B	40 GB	
Megatron-LM	72	3072	32	8.3B	174 GB	512x V100 GPU (9 days)
Turing-NLG	78	4256	28	17B	?	256x V100 GPU
GPT-3	96	12288	96	175B	694GB	?

BERT

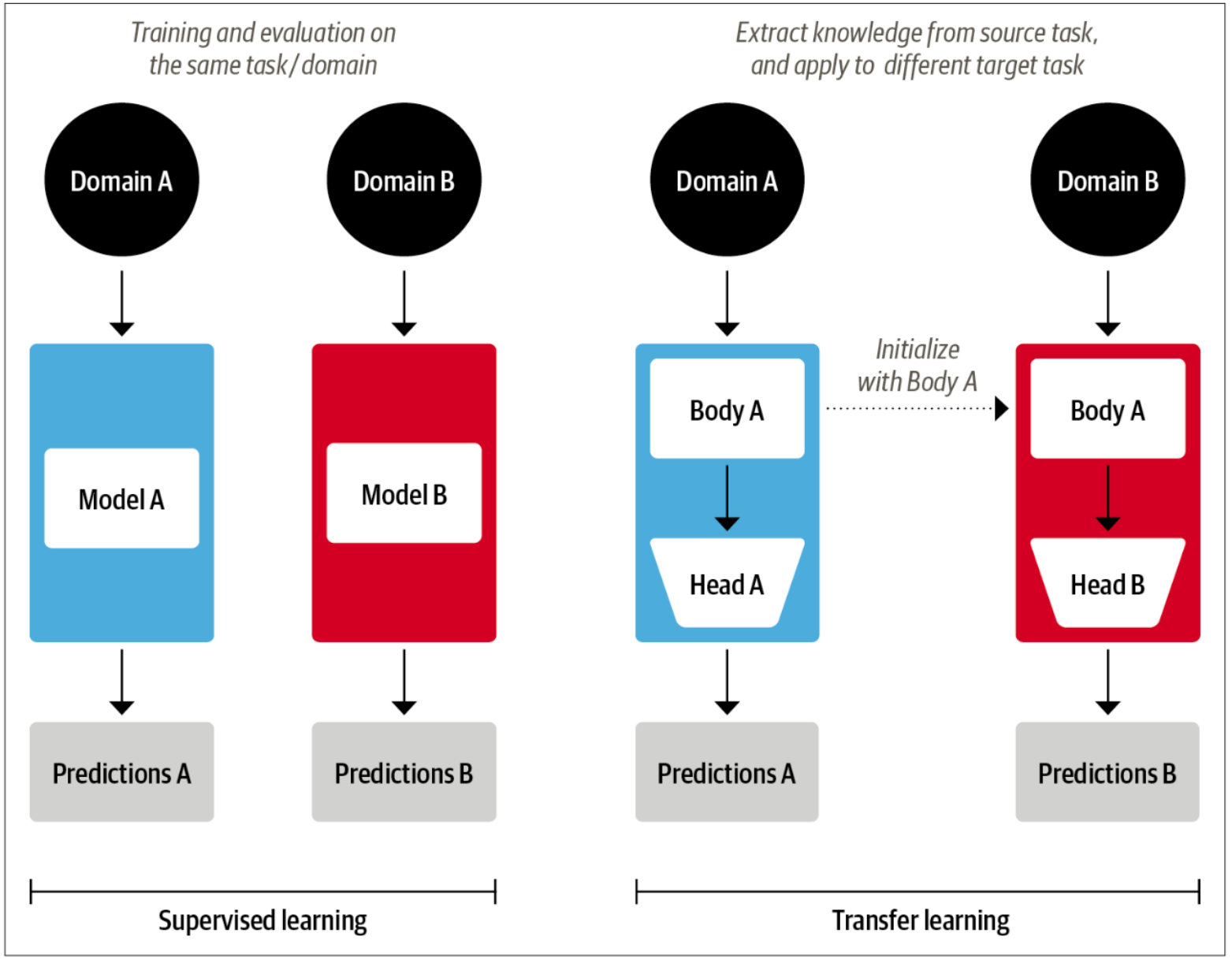
Training BERT: (1) **Masked Language Modelling.** Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence.



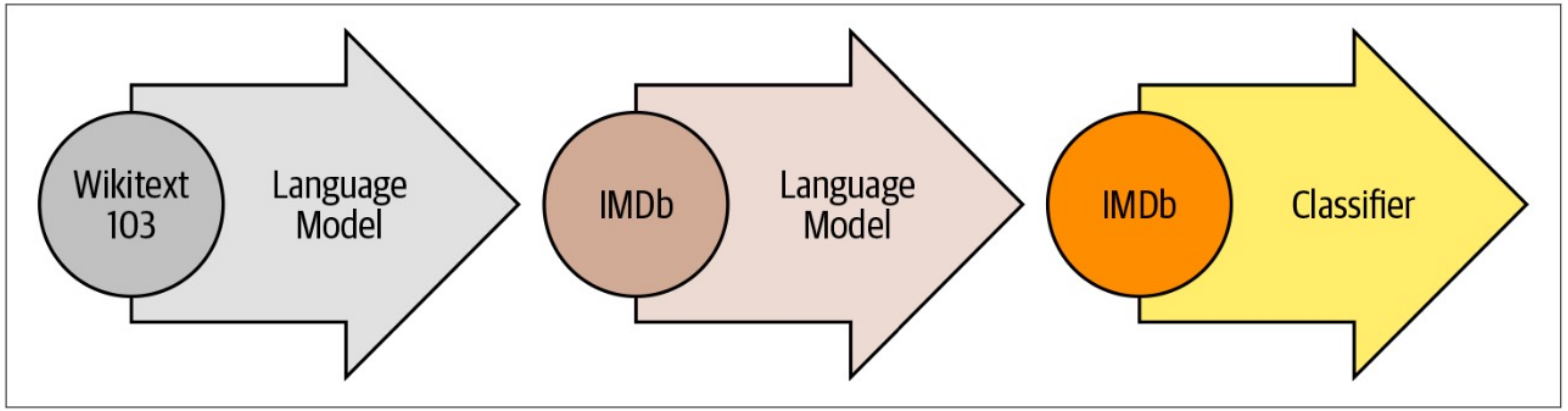
Training BERT: (2) **Next sentence prediction task** is a binary classification task in which, given a pair of sentences, it is predicted if the second sentence is the actual next sentence of the first sentence.

Transfer Learning in NLP

- NLP applications typically required large amounts of labeled data to achieve high performance.
- Transfer Learning produces high-quality models that can be trained much more efficiently on a variety of downstream tasks, and with much less labeled data.
- GPT and BERT set a new state of the art by combined self-attention with transfer learning for NLP



Transfer Learning in NLP



ULMFiT model

1. Predict the next word based on the previous words using large corpus; Wikipedia (language modeling)
2. Adapt it to the in-domain corpus (IMDb corpus of movie reviews-language modeling on target corpus)
3. Add classification layer for the target task (e.g., classifying the sentiment of movie reviews)

Transfer Learning in NLP

GPT:

- uses only decoder part & language modeling
- model was pretrained by predicting the next word based on the previous ones.
- pretrained on the BookCorpus,11 which consists of 7,000 unpublished books

BERT:

- Uses the encoder part of the Transformer architecture
- pretrained on the BookCorpus and English Wikipedia.
- Trained to predict randomly masked words in a text.

  "I looked at my [MASK] and saw that [MASK] was late."
- Also trained on *next sentence prediction* (NSP).

Hugging Face Transformers

- It provides a standardized interface to a wide range of transformer models as well as code and tools to adapt these models to new use cases.
- Supports three major deep learning frameworks (PyTorch, TensorFlow, and JAX)
- Provides task-specific heads so you can easily fine-tune transformers on downstream tasks such as text classification, named entity recognition, and question answering.
- Reduces the time it takes a practitioner to train and test a handful of models
- Released more than 50 models https://github.com/huggingface/transformers
- Tutorial how to load models, fine tune them and use data https://huggingface.co/course/chapter0/1?fw=pt

[1] Natural Language Processing with Transformers, Revised Edition, 2022, by Lewis Tunstall, Leandro von Werra, and Thomas Revised copy available on discovery (EbookLinks to an external site.)

Pipelines in Huggingface –today lab

- https://huggingface.co/docs/transformers/main_classes/pipelines

  easy way to use models for inference

- Pipelines are objects that abstract most of the complex code
- Offers a simple API for tasks, like Named Entity Recognition, Masked Language Modeling, Sentiment Analysis, Feature Extraction and Question Answering.

```
>>> pipe = pipeline("text-classification
>>> pipe(["This restaurant is awesome",
[{'label': 'POSITIVE', 'score': 0.999874
 {'label': 'NEGATIVE', 'score': 0.9996669292449951}]
```

```
>>> pipe = pipeline(model="roberta-large-mnli")
>>> pipe("This restaurant is awesome")
[{'label': 'NEUTRAL', 'score': 0.7313136458396912}]
```

Transformer NLP Applications (Hands-on)

Test Classification: Identify if a customer feedback is positive or negative (sentiment analysis)

Named Entity Recognition: identify named entities like products, places, and people (named entity recognition -NER)

Question Answering: provide the model with a passage of text called "context", along with a question whose answer we'd like to extract. The model then returns the span of text corresponding to the answer **(example here is *extractive question answering* because the answer is extracted directly from the text)**

https://colab.research.google.com/github/nlp-with-transformers/notebooks/blob/main/01_introduction.ipynb#scrollTo=5ygcsdXvVdcc

Transformer NLP Applications (Hands-on)-cont.

Summarization: take a long text as input and generate a short version with all the relevant facts. More complicated task that requires the model to generate coherent text.

Translation: generated text in a different language

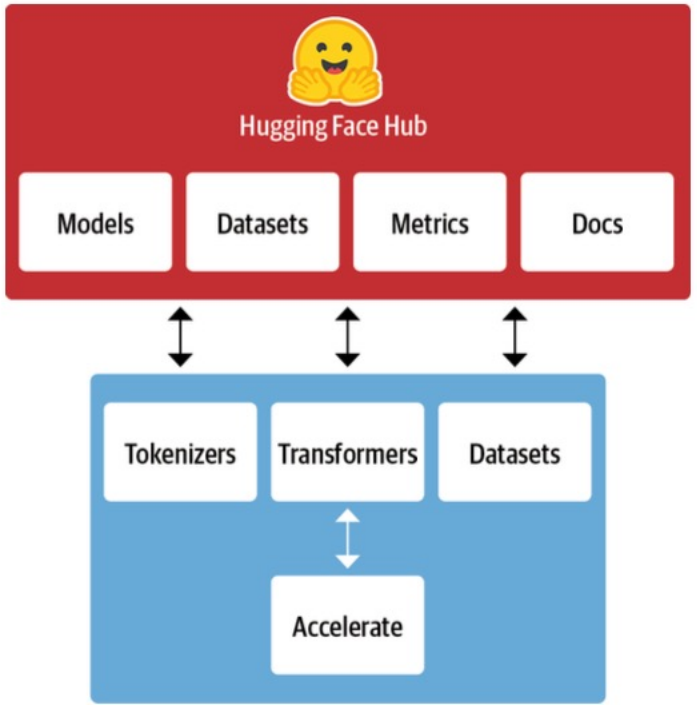
```
translator = pipeline("translation_en_to_de", model="Helsinki-NLP/opus-mt-en-de")
# override the default model in the pipeline to pick the best one for your application
outputs = translator(text, clean_up_tokenization_spaces=True, min_length=100)
print(outputs[0]['translation_text'])
```

Text Generation: faster replies to customer feedback by having access to an autocomplete function.

https://colab.research.google.com/github/nlp-with-transformers/notebooks/blob/main/01_introduction.ipynb#scrollTo=5ygcsdXvVdcc

The Hugging Face Ecosystem

- Provides many libraries and tools to accelerate your NLP and machine learning projects.



pretrained model weights, datasets, scripts for the evaluation metrics
hosts over 20,000 freely available models

Libraries of code

Learn how to adapt transformers to a wide range of use cases raining and integrate those models into your own applications

[1] Natural Language Processing with Transformers, Revised Edition, 2022, by Lewis Tunstall, Leandro von Werra, and Thomas Revised copy available on discovery (EbookLinks to an external site.)

GPT-3 vs. LLaMA

GPT-3 (2020)

50,257 vocabulary size
2048 context length
175B parameters
Trained on 300B tokens

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

Training: (rough order of magnitude to have in mind)

- O(1,000 - 10,000) V100 GPUs
- O(1) month of training
- O(1-10) $M

LLaMA (2023)

32,000 vocabulary size
2048 context length
65B parameters
Trained on 1-1.4T tokens

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

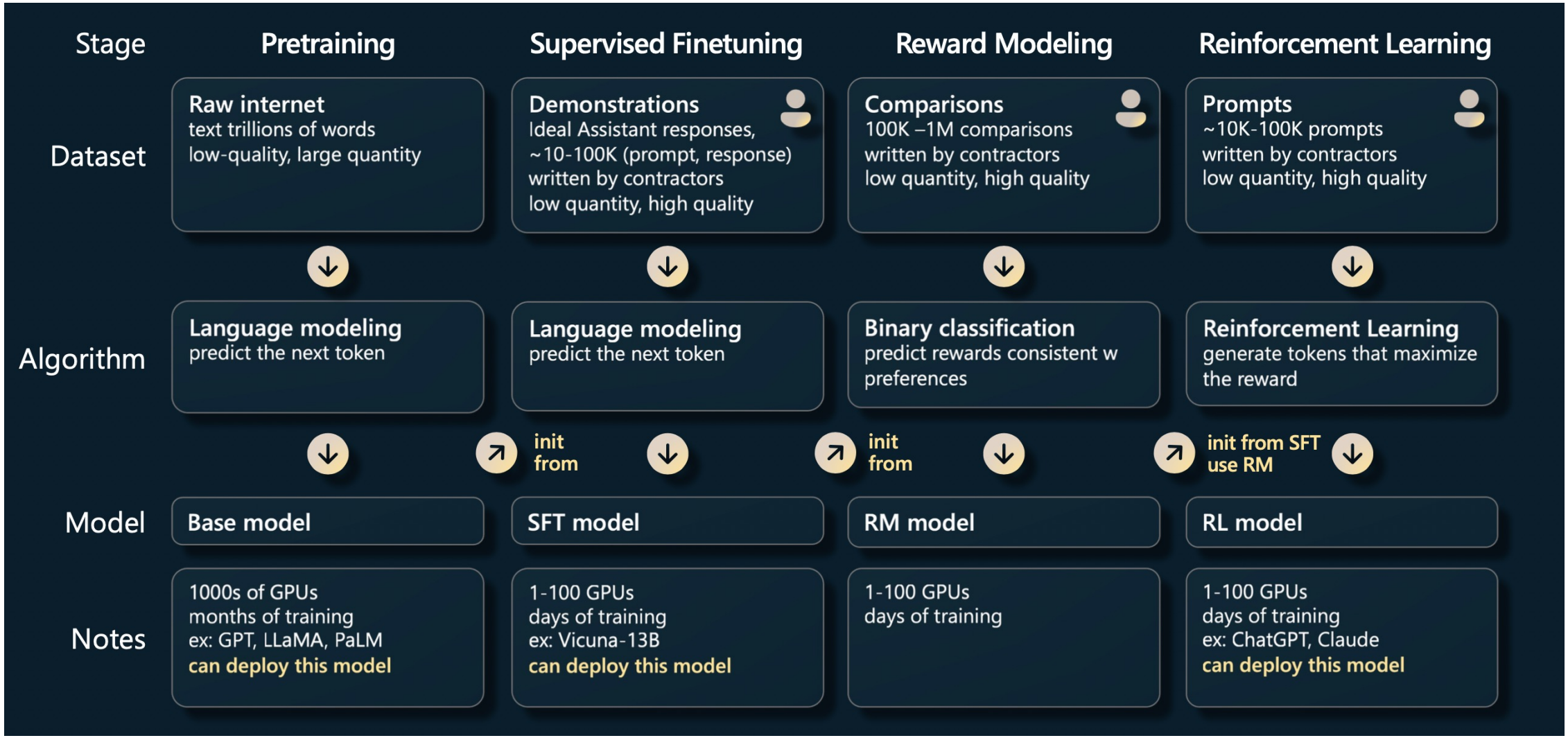
Table 2: **Model sizes, architectures, and optimization hyper-parameters.**

Training for 65B model:

- 2,048 A100 GPUs
- 21 days of training
- $5M

[Language Models are Few-Shot Learners, OpenAI 2020]
[LLaMA: Open and Efficient Foundation Language Models, Meta AI 2023]

Training Pipeline



LLM Leaderboard from Chatbot Arena

Model	⭐ Arena Elo rating	📈 MT-bench (score)	MMLU	License
GPT-4-Turbo	1210	9.32		Proprietary
GPT-4	1159	8.99	86.4	Proprietary
Claude-1	1146	7.9	77	Proprietary
Claude-2	1125	8.06	78.5	Proprietary
Claude-instant-1	1106	7.85	73.4	Proprietary
GPT-3.5-turbo	1103	7.94	70	Proprietary
WizardLM-70b-v1.0	1093	7.71	63.7	Llama 2 Community
Vicuna-33B	1090	7.12	59.2	Non-commercial
OpenChat-3.5	1070	7.81	64.3	Apache-2.0
Llama-2-70b-chat	1065	6.86	63	Llama 2 Community
WizardLM-13b-v1.2	1047	7.2	52.7	Llama 2 Community
zephyr-7b-beta	1042	7.34	61.4	MIT
MPT-30B-chat	1031	6.39	50.4	CC-BY-NC-SA-4.0

A Survey of Large Language Models

Wayne Xin Zhao, Kun Zhou*, Junyi Li*, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie and Ji-Rong Wen

Abstract—Ever since the Turing Test was proposed in the 1950s, humans have explored the mastering of language intelligence by machine. Language is essentially a complex, intricate system of human expressions governed by grammatical rules. It poses a significant challenge to develop capable artificial intelligence (AI) algorithms for comprehending and grasping a language. As a major approach, *language modeling* has been widely studied for language understanding and generation in the past two decades, evolving from statistical language models to neural language models. Recently, pre-trained language models (PLMs) have been proposed by pre-training Transformer models over large-scale corpora, showing strong capabilities in solving various natural language processing (NLP) tasks. Since the researchers have found that model scaling can lead to an improved model capacity, they further investigate the scaling effect by increasing the parameter scale to an even larger size. Interestingly, when the parameter scale exceeds a certain level, these enlarged language models not only achieve a significant performance improvement, but also exhibit some special abilities (*e.g.,* in-context learning) that are not present in small-scale language models (*e.g.,* BERT). To discriminate the language models in different parameter scales, the research community has coined the term *large language models (LLM)* for the PLMs of significant size (*e.g.,* containing tens or hundreds of billions of parameters). Recently, the research on LLMs has been largely advanced by both academia and industry, and a remarkable progress is the launch of ChatGPT (a powerful AI chatbot developed based on LLMs), which has attracted widespread attention from society. The technical evolution of LLMs has been making an important impact on the entire AI community, which would revolutionize the way how we develop and use AI algorithms. Considering this rapid technical progress, in this survey, we review the recent advances of LLMs by introducing the background, key findings, and mainstream techniques. In particular, we focus on four major aspects of LLMs, namely pre-training, adaptation tuning, utilization, and capacity evaluation. Furthermore, we also summarize the available resources for developing LLMs and discuss the remaining issues for future directions. This survey provides an up-to-date review of the literature on LLMs, which can be a useful resource for both researchers and engineers.

Index Terms—Large Language Models; Emergent Abilities; Adaptation Tuning; Utilization; Alignment; Capacity Evaluation

Yet, Transformers do NOT really understand language

Two plus two is

- a three, five is a six, seven
- the result of a simple equation, and the
- four, and two plus three is four,

Two minus two is

- seven, plus two is six.
- a little too low for me.
- a perfect match for the numbers 1 to 20 in

Todays Hype

Foundation Models & Generative AI

“Foundation models are large AI models trained on enormous quantities of unlabeled data—usually through self-supervised learning. This process results in generalized models capable of a wide variety of tasks in NLP and Vision/Speech”

Examples: BERT, GPT-3 , ChatGPT, Stable Diffusion, Google’s LaMDA, DALL-E, **GPT-4**…

Guide to Foundation modles and their adoption in enterprise

Transformers Summary

- Transformers enable modelling long dependencies between input sequence elements
- Support parallel processing of sequence as opposed to recurrent networks e.g. LSTMs
- Different from convolutional networks, Transformers require minimal inductive biases for their design
- Naturally suited as set-functions
- Transformers allows processing multiple modalities (*e.g.*, images, videos, text and speech) using similar processing blocks
- Demonstrates excellent scalability to very large capacity networks and huge datasets

Resources

Code:

- Chapter 3 https://colab.research.google.com/github/nlp-with transformers/notebooks/blob/main/03_transformer-anatomy.ipynb
- Chapter 1 https://colab.research.google.com/github/nlp-with-transformers/notebooks/blob/main/01_introduction.ipynb

Readings Chapter 1 &3:

[1] Natural Language Processing with Transformers, Revised Edition, 2022, by Lewis Tunstall, Leandro von Werra, and Thomas Revised copy available on discovery (EbookLinks to an external site.)

Invited talk by Amazon Alexa on NLP 2022 (session recordingLinks to an external site.)

Thank you!