UNIVERSITY OF AUCKLAND
Waipapa Taumata Rau
NEW ZEALAND

SCIENCE

COMPSC 760
Advanced Neural Networks

Deep Learning - Lecture 3

Learning outcomes

Lecture 1: Deep Neural Networks Review

- Review what a deep neural network is and the differences classical ML approaches.
- Review the different steps involved in training a deep NN.
- Review network initialisation and the different activation functions.
- Review what the hyperparameters of a DNN are.
- Review the different strategies to improve the performance of a deep NN.
- Review the different strategies to tune a deep NN.

Lectures 2 & 3: Learning with sequences (RNNs, Transformers, LLMs)

- Understand how recurrent neural networks work.
- Recognise commonly used neural network architectures based on RNN (LSTM, GRU).
- Understand how transformers work.
- Understand the principles of Large Language models.

Transformers and LLMs

- Understand how transformers work.
- Understand the principles of Large Language Models (LLMs).
- Be aware of models/tools using LLMs.

If you want to go further:

- Deep Learning, Part II Deep Networks, chap. 10.
  https://www.deeplearningbook.org/,
- Stanford Deep Learning courses:
  https://stanford.edu/~shervine/teaching/cs-230/
  https://cs230.stanford.edu/
  http://cs231n.stanford.edu/

Transformer - Motivation

- Traditional RNNs suffer some drawbacks

  1. Sequential processing leading to low training times (especially for long sequences) + hard to parallelise.

  2. Difficulty to model long term dependencies as information from earlier steps becomes increasingly diluted.

- The Transformer architecture addresses these issues by using the self-attention mechanism.

Self-attention mechanism

Self-attention allows the model to weigh the importance of different parts of the input sequence when making predictions.

E.g., with language models, the model "focuses" on different words of the sequence depending on their relevance to the task at hand:

Predicting the next word in: "The rabbits are eating"

What are the important words to focus on for the prediction?

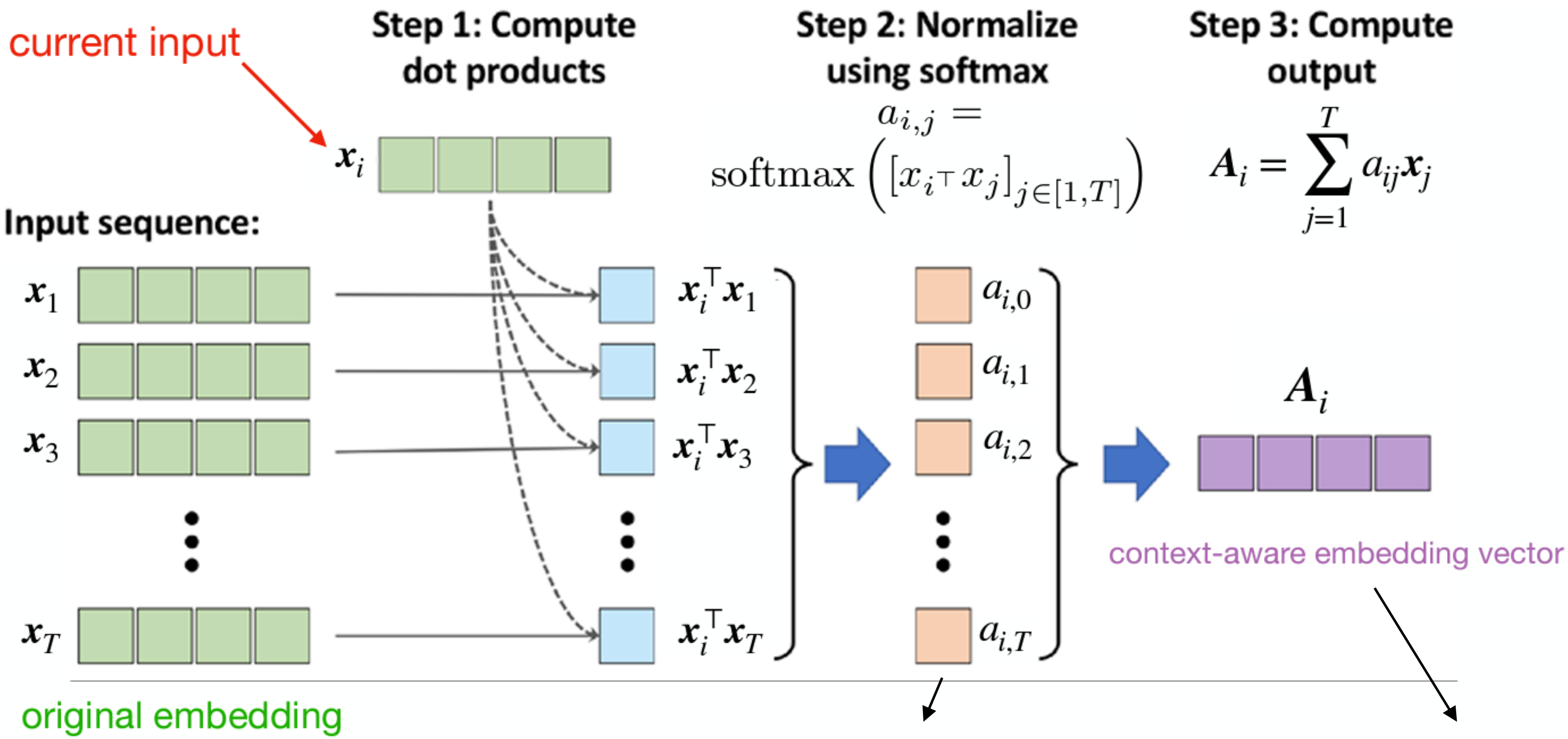
Word embeddings

Individual words are represented as vector of numerical values in a lower dimension space. Such representations are called word embeddings.

Word embeddings aim at capturing the meaning of words and their relationship to other words (semantic and syntax).

- Bag of words (BOW) are one of the simplest word embedding, but they can be intensive to compute, and they fail to capture the relationship between words (i.e., do not consider the order).
- Modern word embeddings are learned through ML and take in considering the local or global context of the words. A few popular embeddings: Word2Vec, GloVe, FastText and ELMO.

Self-attention – basic version

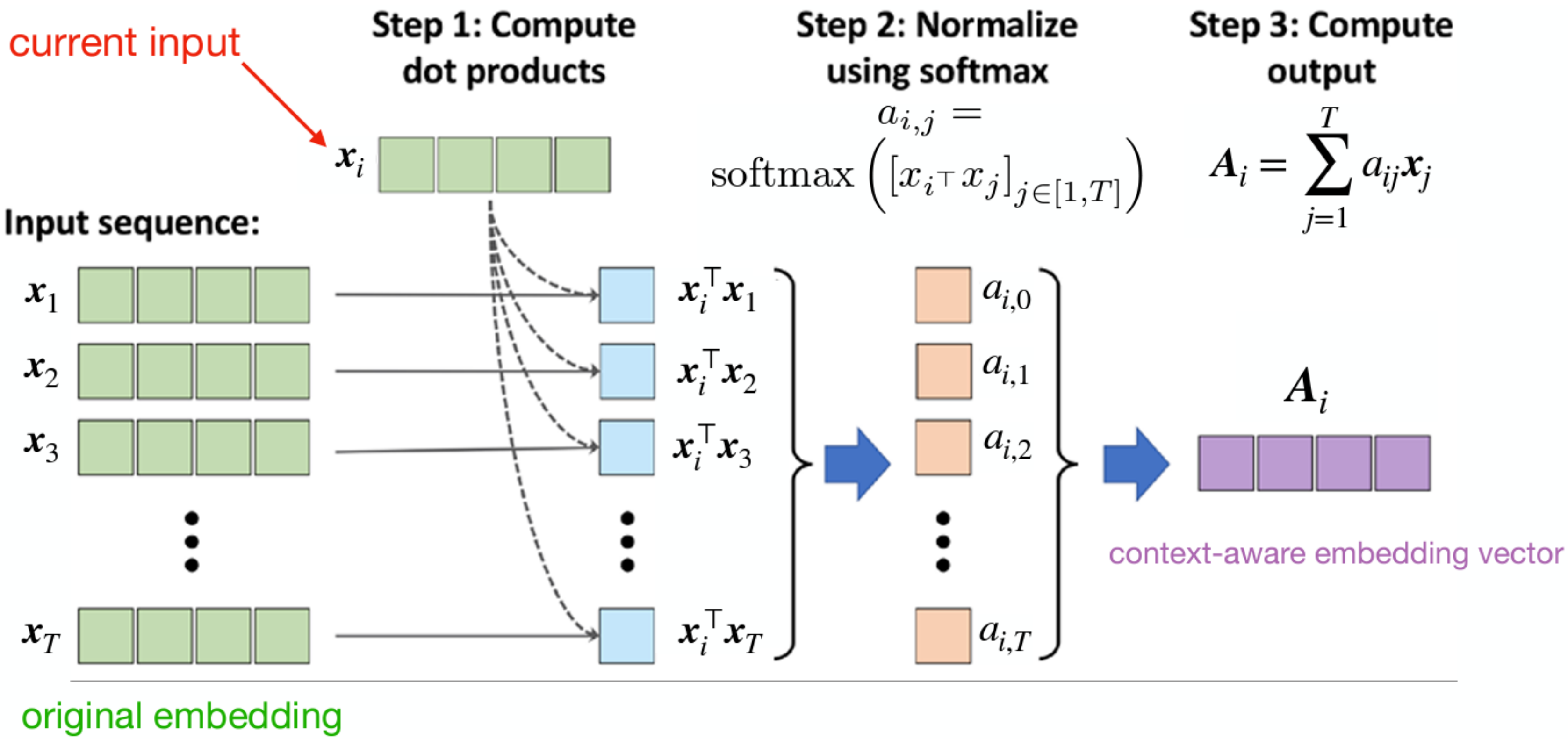


How similar is the i^{th} element to the other elements of the sequence?

Output for the i^{th} element

Image source: Raschka & Mirjalili 2019. Python Machine Learning, 3rd edition

Self-attention – basic version



Limitation of this basic version: no learnable parameters!

Image source: Raschka & Mirjalili 2019. Python Machine Learning, 3rd edition

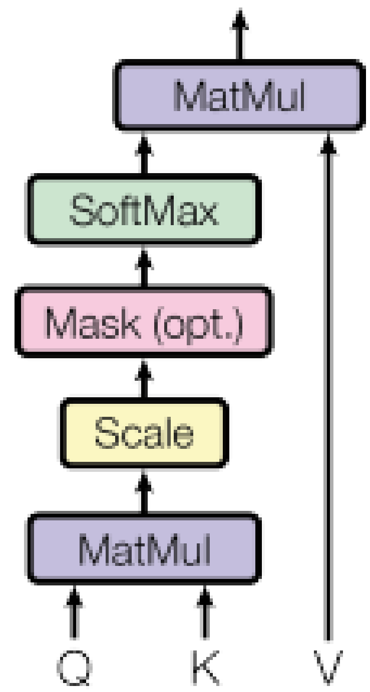
Scaled Dot-Product attention

The Transformer architecture uses an implementation of self-attention called « Scaled Dot-Product attention ».

1. Transform the input matrix $X = [x_1, x_2, \ldots, x_n]$ into 3 matrices:
   - Query: $Q = W^Q X$
   - Key: $K = W^K X$
   - Value: $V = W^V X$

W^Q, W^K, W^V are learnable weight matrices that transform the input matrix into query, key and value.

Scaled Dot-Product Attention



[Attention Is All You Need, Vaswani et al., 2017]

Scaled Dot-Product attention

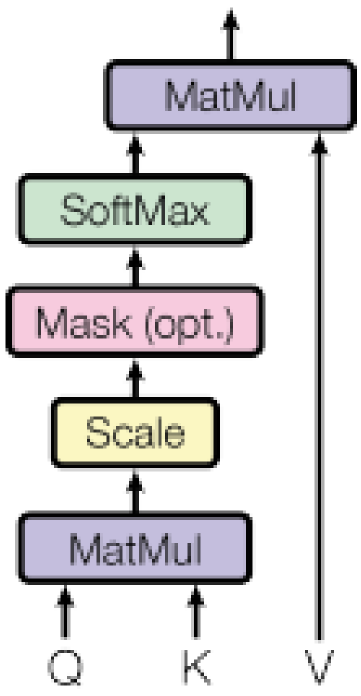
2. Calculate the attention « score »:

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

d_k is the size of the keys and queries
(d_k = d_q = d_v in the original paper).

Scaling prevents the dot products to grow large, thus avoiding vanishing gradients (application of softmax to large values would yield small gradients).

Scaled Dot-Product Attention



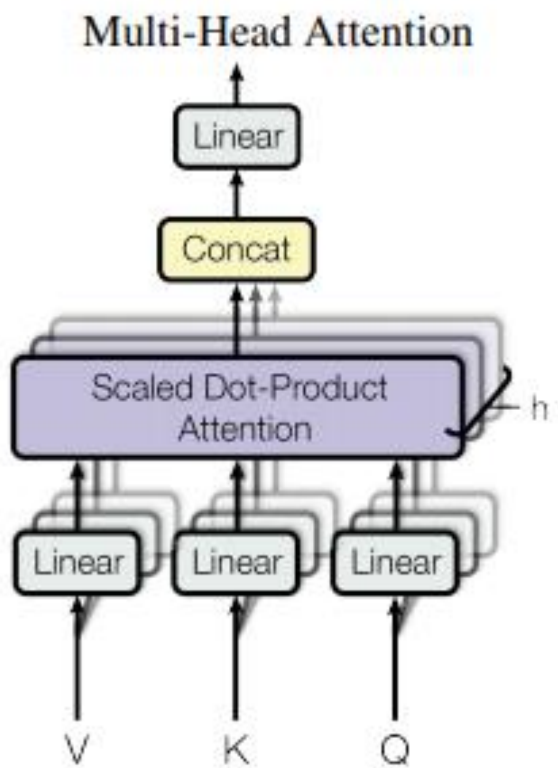
[Attention Is All You Need, Vaswani et al., 2017]

Multi-Head attention

UNIVERSITY OF AUCKLAND Waipapa Taumata Rau NEW ZEALAND | SCIENCE

The Transformer architecture actually stacks several scaled dot-product attention layers in parallel.

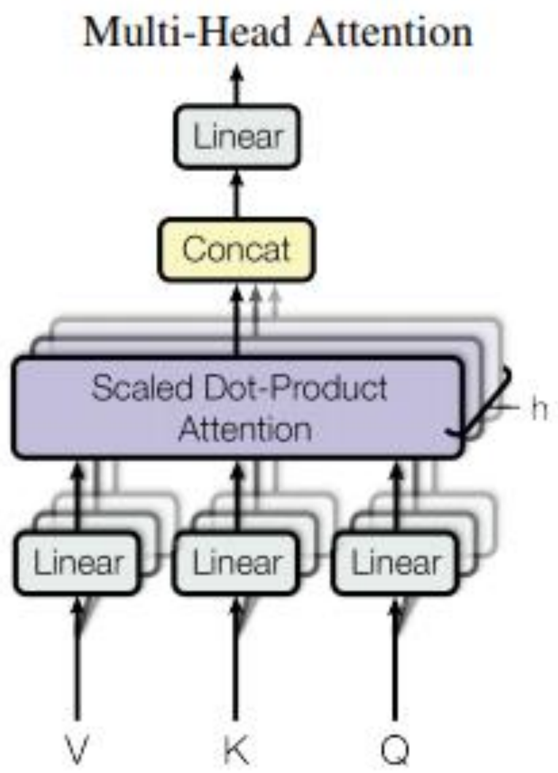
- *h* parallel layers, also called “heads”.
- Values, keys and queries are projected linearly *h* times with different learned linear projections.
- Each projection is the input of an head, which has its own different W^Q, W^K, W^V matrices.



[Attention Is All You Need, Vaswani et al., 2017]

Multi-Head attention

The Transformer architecture actually stacks several scaled dot-product attention layers in parallel.



- Results are concatenated and projected again to obtain the output:

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

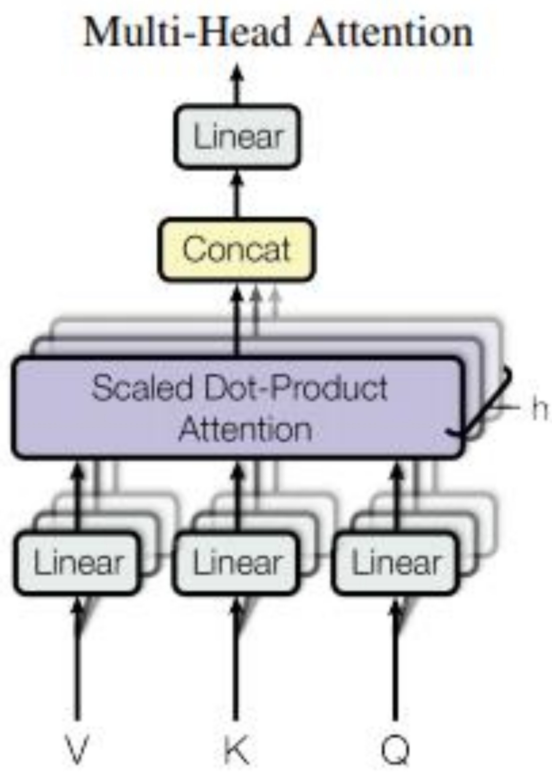
[Attention Is All You Need, Vaswani et al., 2017]

Multi-Head attention

Multi-head attention is beneficial because it allows the Transformer to focus on different aspects of the input sequence.

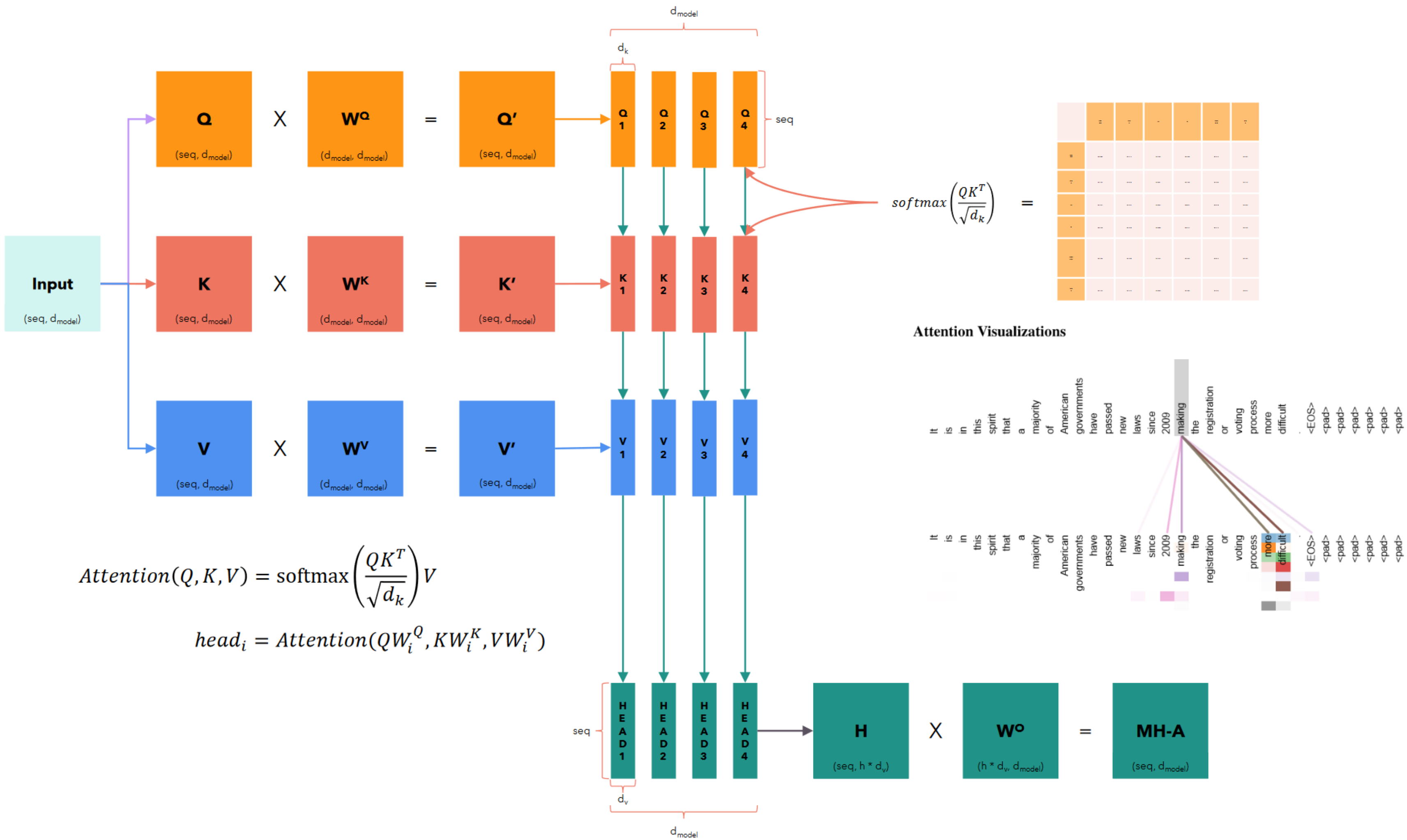
Using a single attention head would have a limiting effect.

Intuitively, the multi-head attention allows the Transformer to spread its attention on different parts of the sequence, instead of averaging it over the full sequence.



[Attention Is All You Need, Vaswani et al., 2017]

Multi-Head attention



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

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

https://github.com/hkproj/transformer-from-scratch-notes

$$MultiHead(Q,K,V) = Concat(head_1 \ \ldots head_h)W^O$$

Transformers – Attention Visualisation

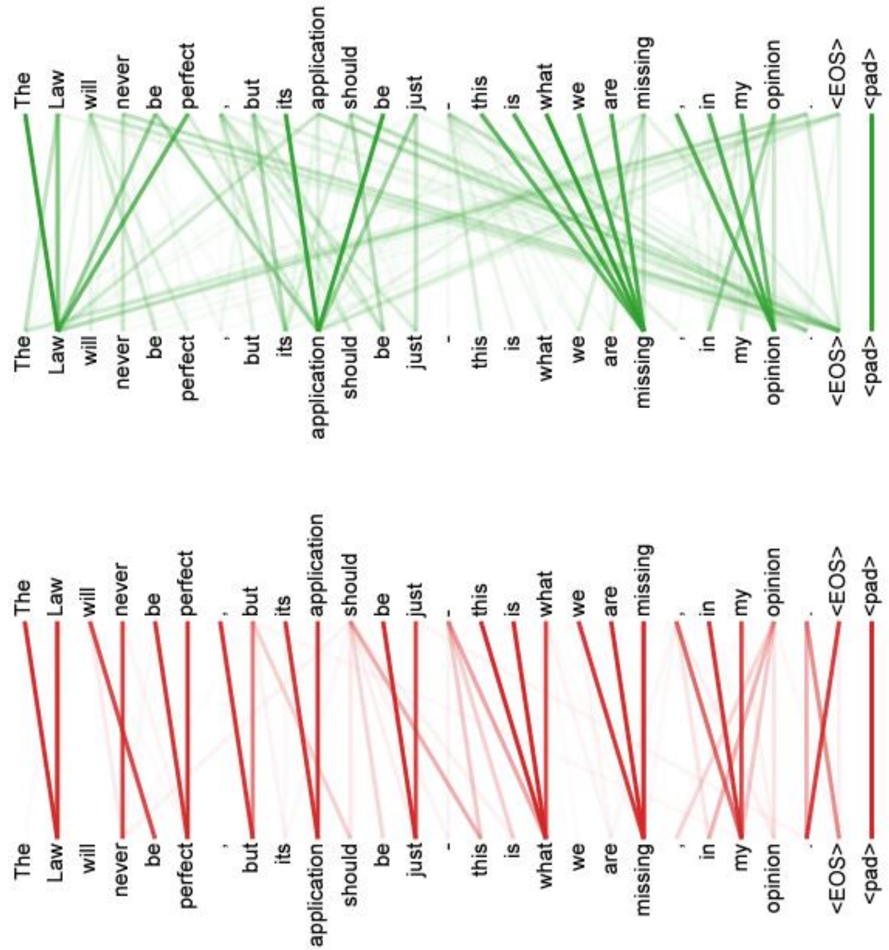
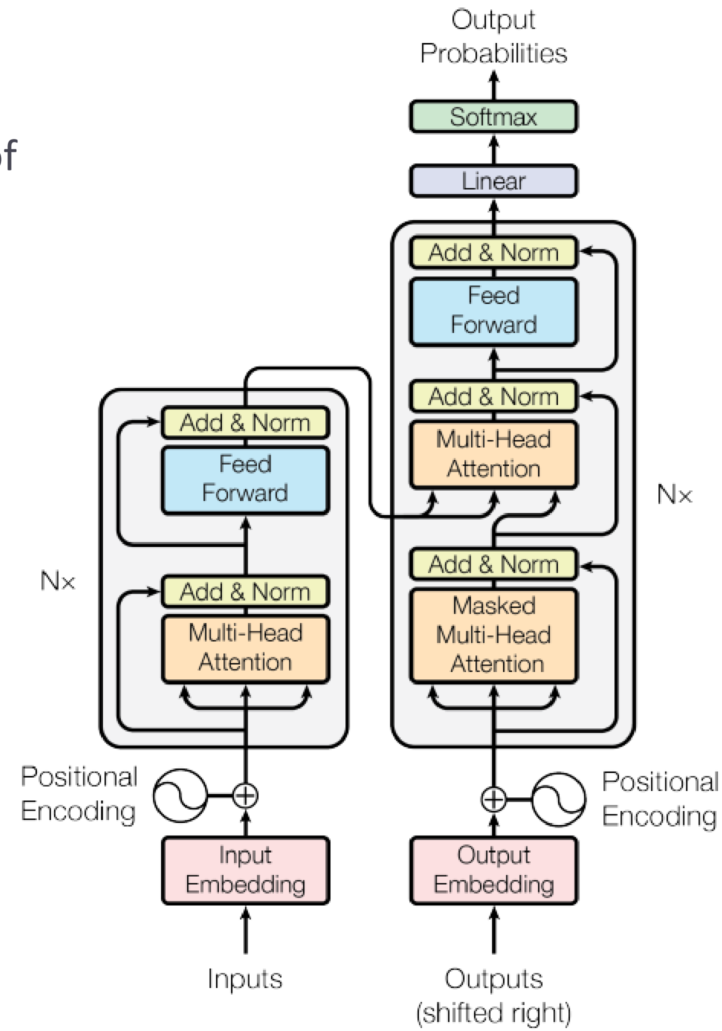


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

Transformer – Full architecture

- Seq2Seq architecture (encoder/decoder)
  - Encoder: takes an input sequence and produces a set of hidden representations, also known as context vector.
  - Decoder: takes the context vector and generates the output sequence.
- Not a RNN!
  - No sequential processing, it uses embedded representations to **encode positions in the sequence**.
- Uses the **attention mechanism** to retain information about which parts of the sequence are important.

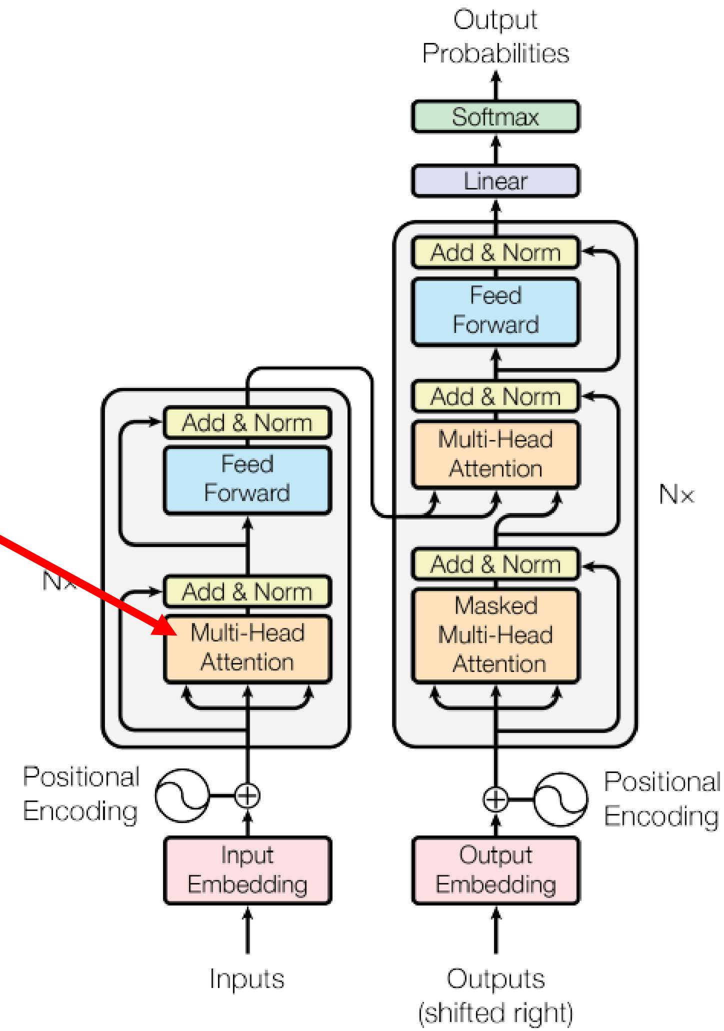


[Attention Is All You Need, Vaswani 2017]

Transformer – Full architecture

Encoder block components:

1. Multi-head attention



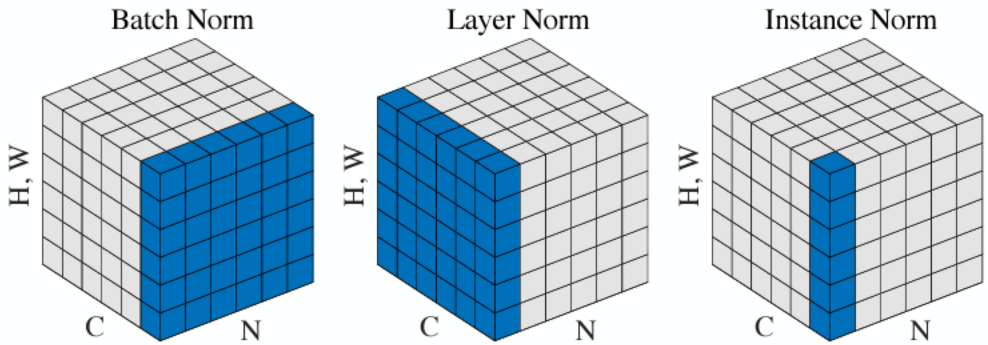
[Attention Is All You Need, Vaswani 2017]

Transformer – Full architecture

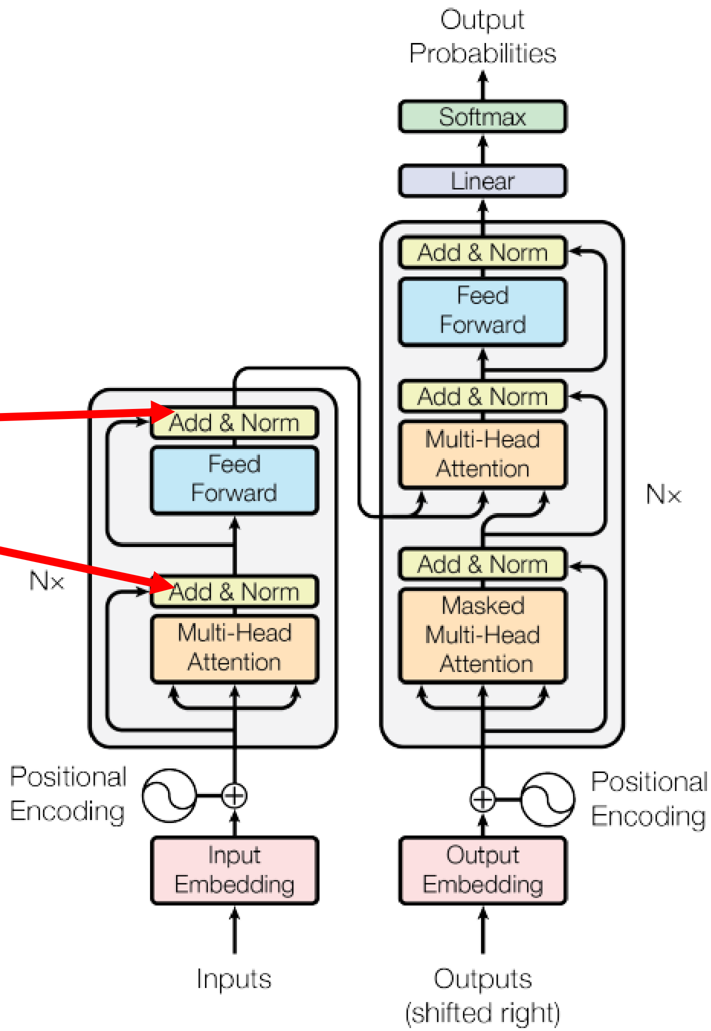
Encoder block components:

1. Multi-head attention

2. Layer normalisation



[Group normalization. ECCV, Wu, Y., & He, K., 2018]



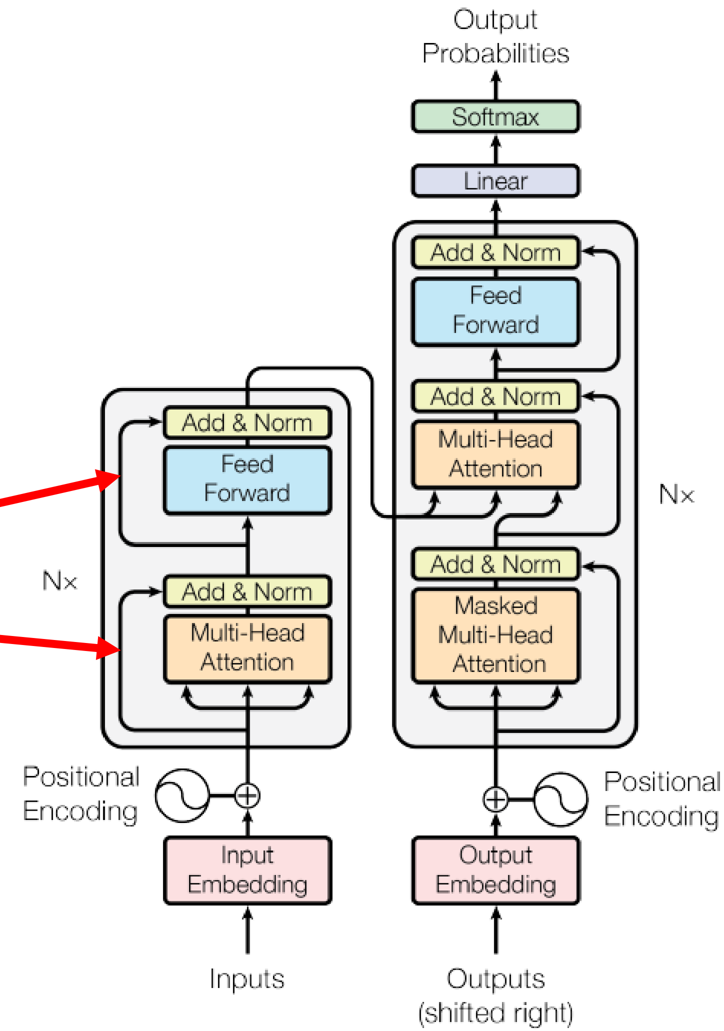
[Attention Is All You Need, Vaswani 2017]

Transformer – Full architecture

Encoder block components:

1. Multi-head attention
2. Layer normalisation
3. Residual/skip connection

$$\text{out}_{\text{norm}} = \text{LayerNorm}(x + \text{Sublayer}(x))$$

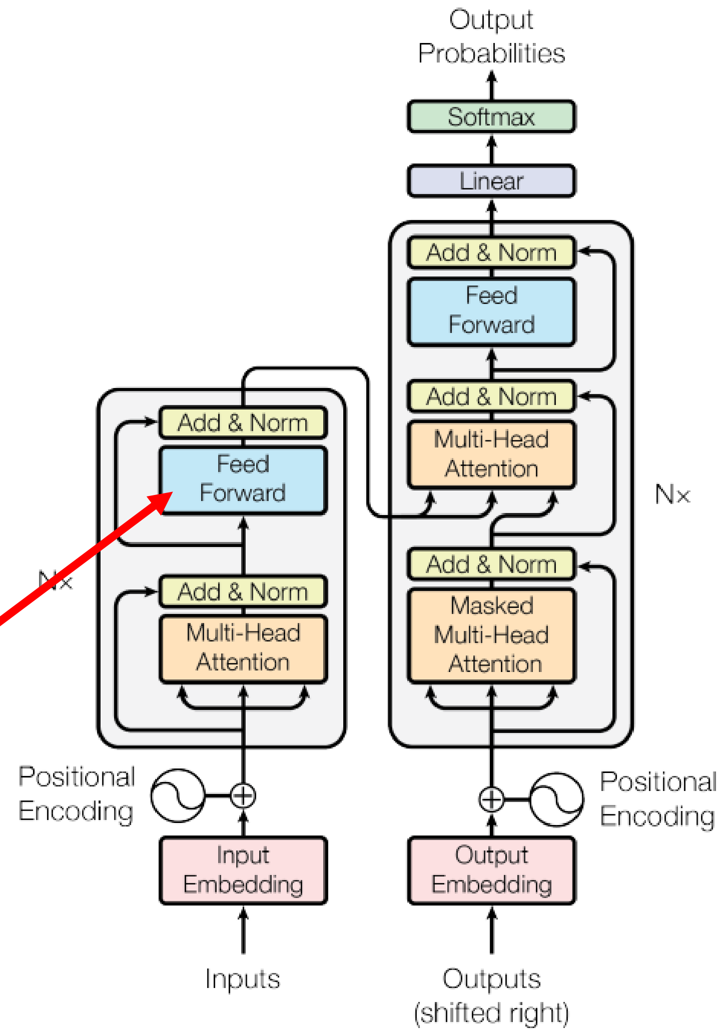


[Attention Is All You Need, Vaswani 2017]

Transformer – Full architecture

Encoder block components:

1. Multi-head attention
2. Layer normalisation
3. Residual/skip connection
4. Fully-connected feed forward NN (multilayer perceptron)
   - Transforms each attention vector into a form adapted for the next block → can be easily parallelised (treat all words at the same time).

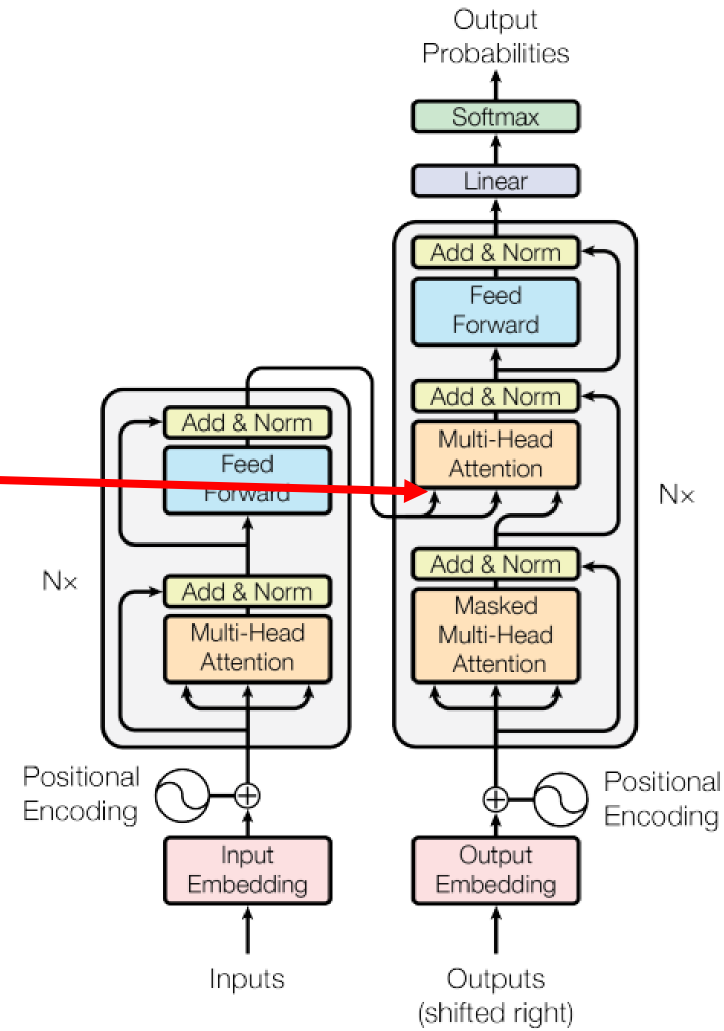


[Attention Is All You Need, Vaswani 2017]

Transformer – Full architecture

Decoder block components:

- Similar components as the encoder.
- Encoder output passed to multi-head attention.

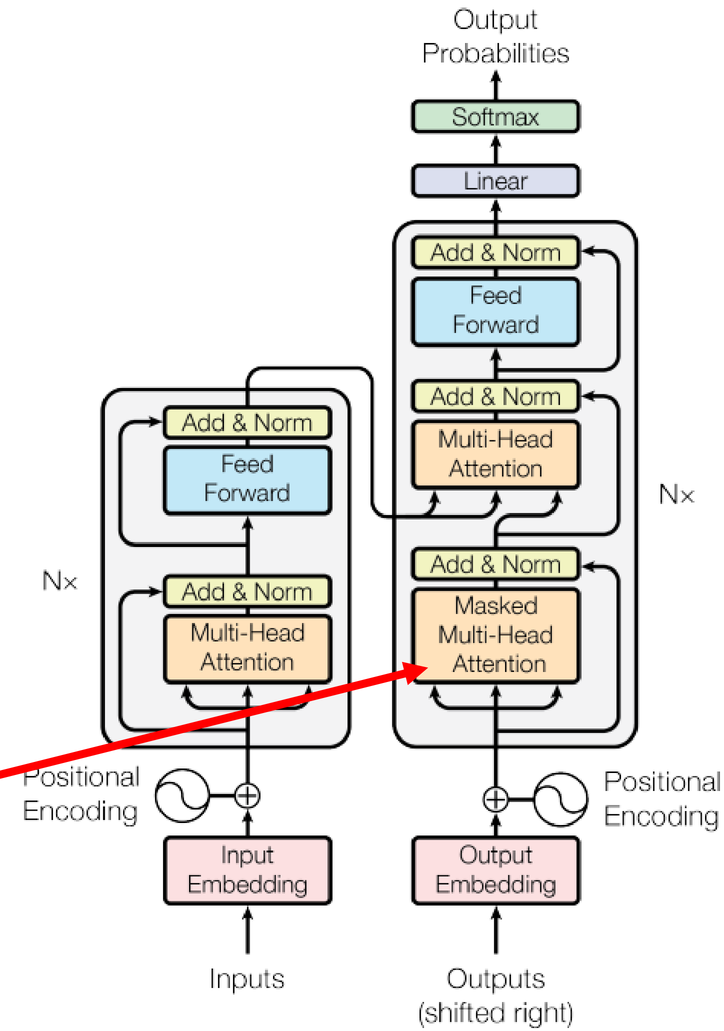


[Attention Is All You Need, Vaswani 2017]

Transformer – Full architecture

Decoder block components:

- Similar components as the encoder.
- Encoder output passed to multi-head attention.
- First multi-head attention sublayer is masked to prevent the model to "cheat" and look at what is coming next in the sentence.
  - Ensures the prediction at position i only depends on outputs at position less than i.

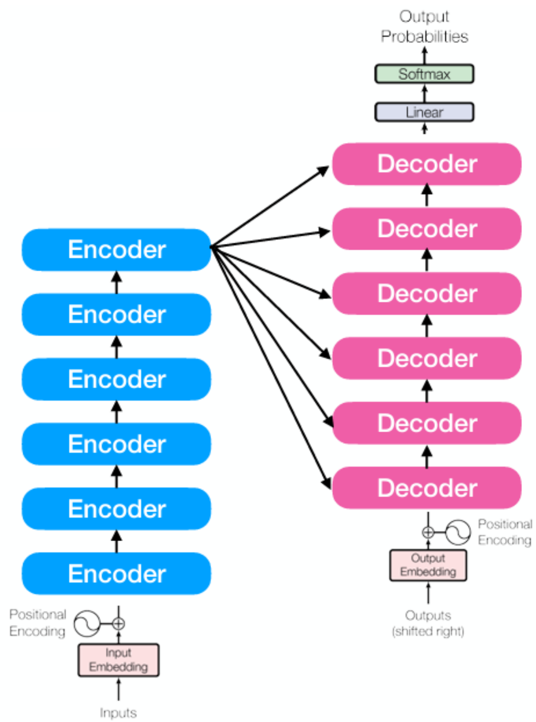


[Attention Is All You Need, Vaswani 2017]

Transformer – Full architecture

Several encoder and decoder blocks are stacked.

- In original Transformer paper: $N_x = 6$



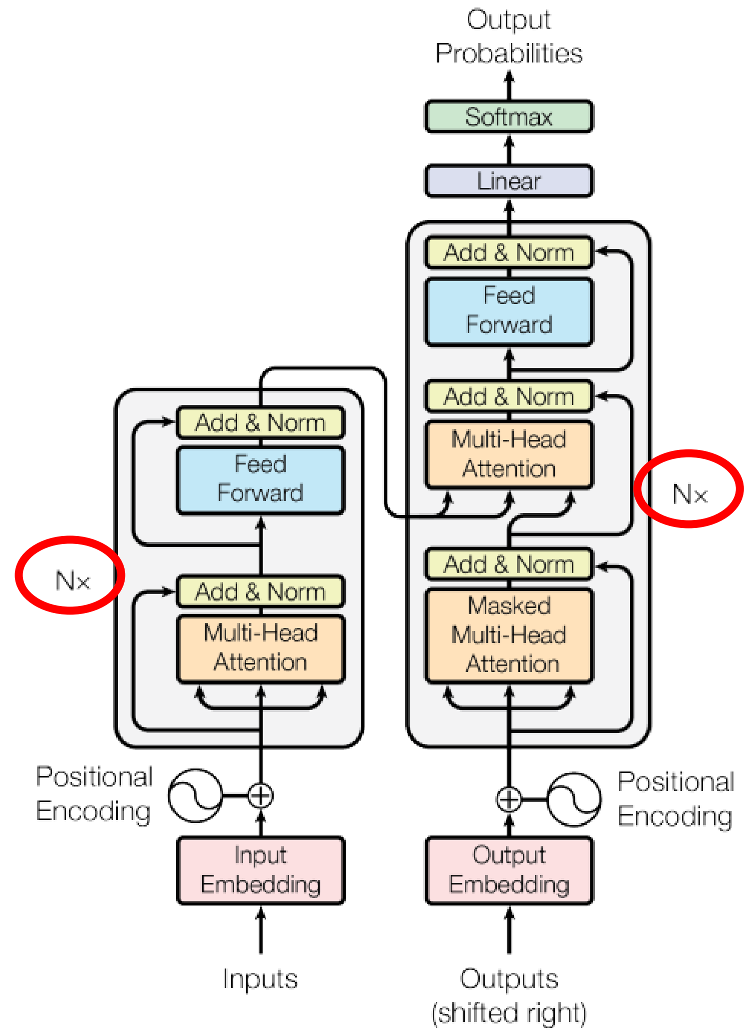


Image source: Sebastian Raschka, STAT 453: Intro to Deep Learning

[Attention Is All You Need, Vaswani 2017]

Transformer – Full architecture

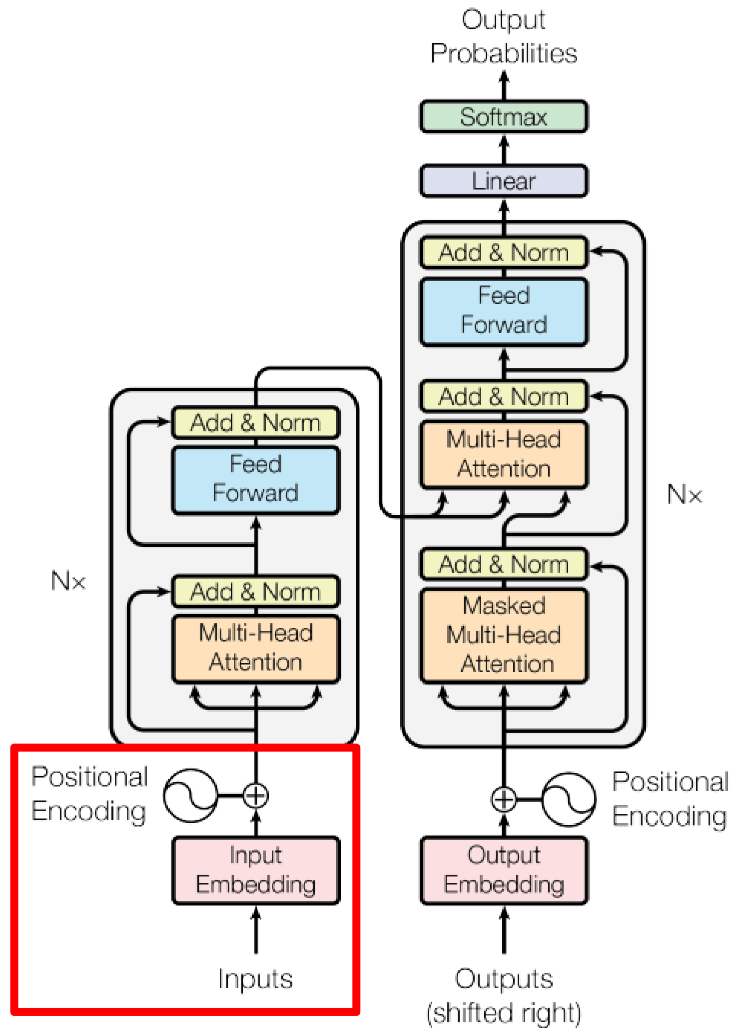
Input/output embedding:

- Original embeddings size: $d_{model} = 512$
- Embeddings are learned during the training process.

Positional encoding:

- RNNs were by design learning about the position of elements in the sequence.
- Transformers loose this information as they do not process the data sequentially!
- Positional encoding is used to retrieve the order information.

[Full explanation with example for positional embedding (Hedu AI - Youtube)]



[Attention Is All You Need, Vaswani 2017]

Transformer – Full architecture

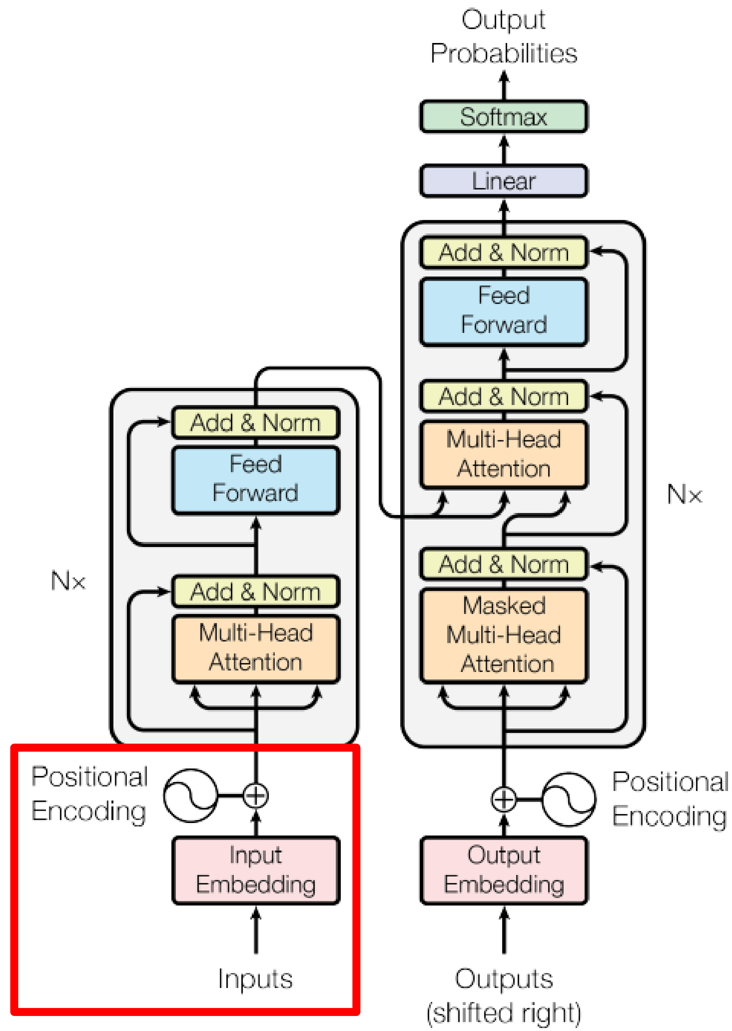
Input/output embedding:

- Original embeddings size: $d_{model} = 512$
- Embeddings are learned during the training process.

Positional encoding:

- RNNs were by design learning about the position of elements in the sequence.
- Transformers lose this information as they do not process the data sequentially!
- Positional encoding is used to retrieve the order information.

[Full explanation with example for positional embedding (Hedu AI - Youtube)]

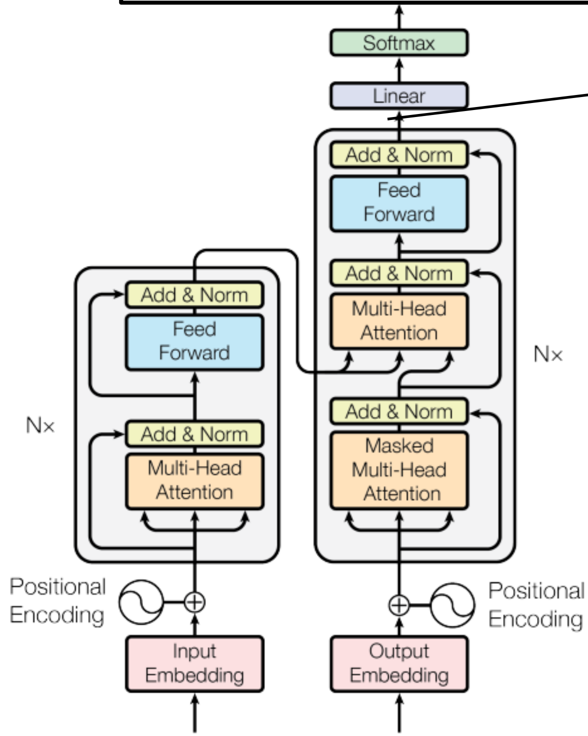


[Attention Is All You Need, Vaswani 2017]

Transformer – Translation (Inference)

Predicted word = highest softmax probability.

Trees



Output = probabilities of each word of the vocabulary to be predicted.

Pr(Trees) = 0.8

Pr(The) = 0.2

Les arbres sont grands

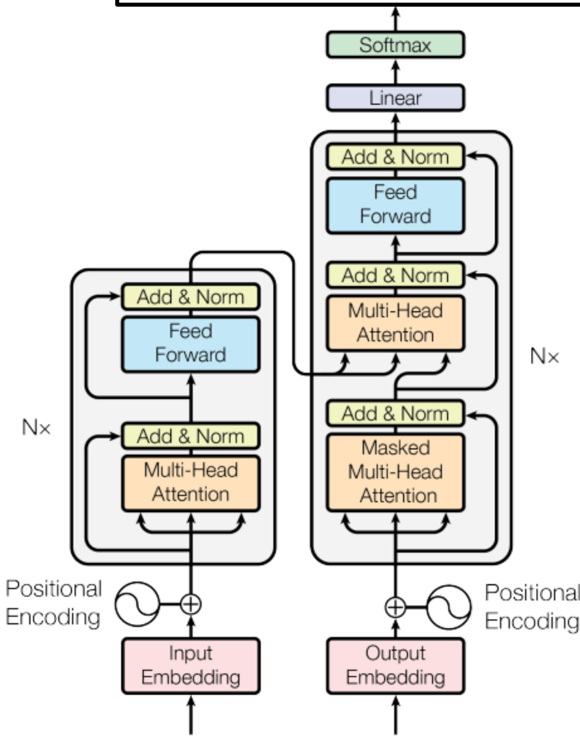
SOS

input

Transformer – Translation

Trees **are**

Pr(are) = 0.95
Pr(is) = 0.05



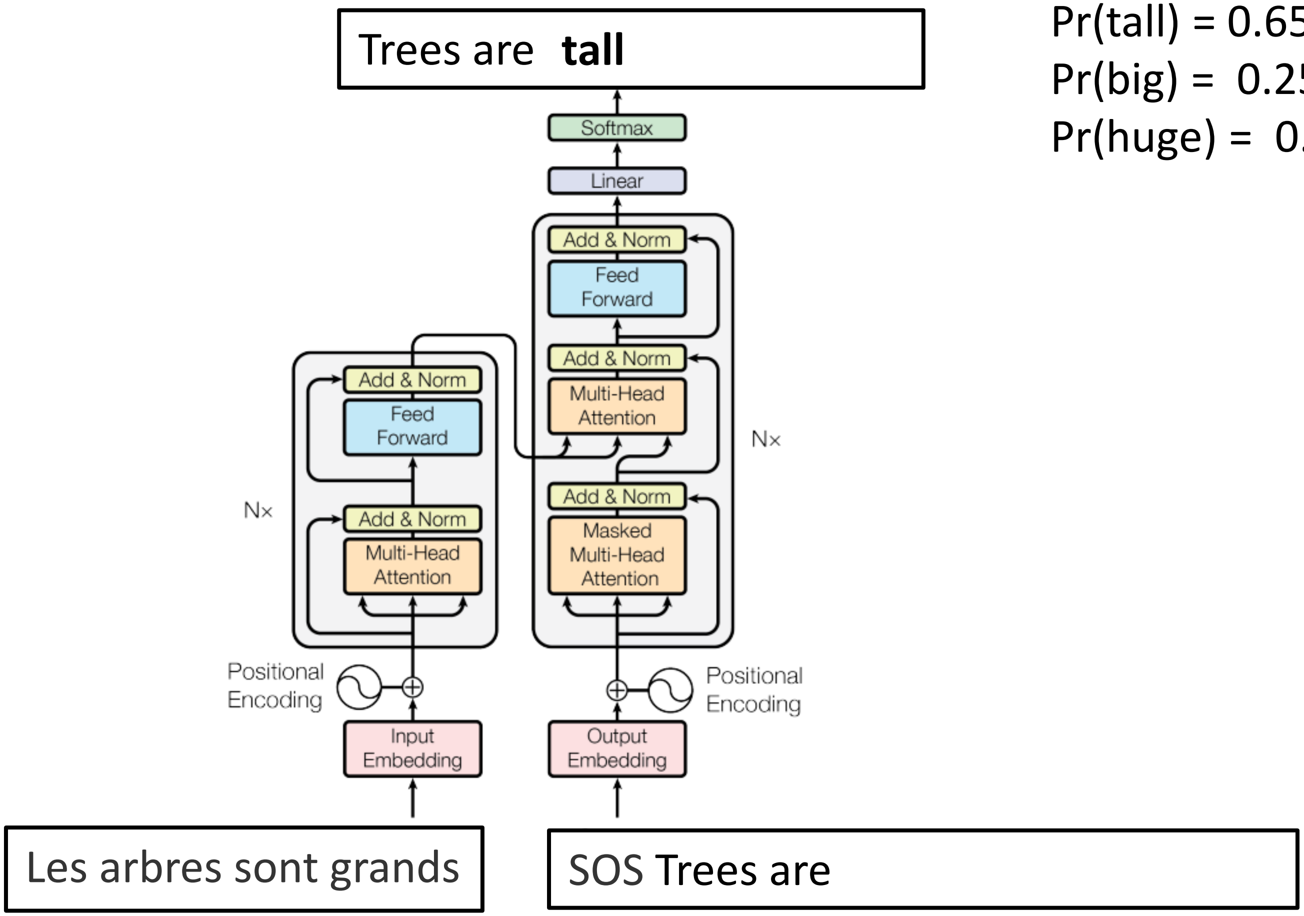
Les arbres sont grands

SOS Trees

input

Transformer – Translation

Pr(tall) = 0.65
Pr(big) = 0.25
Pr(huge) = 0.1



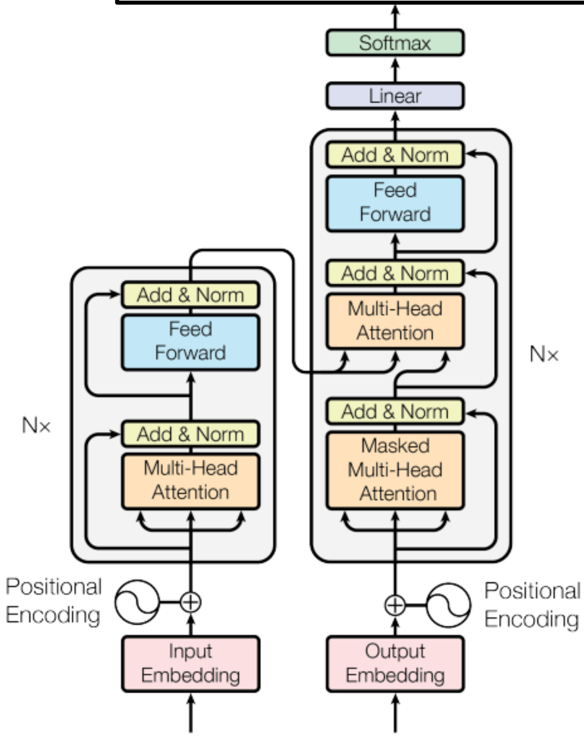
input

Transformer – Translation

Pr(EOS) = 0.99

...

Trees are tall **EOS**



Les arbres sont grands

input

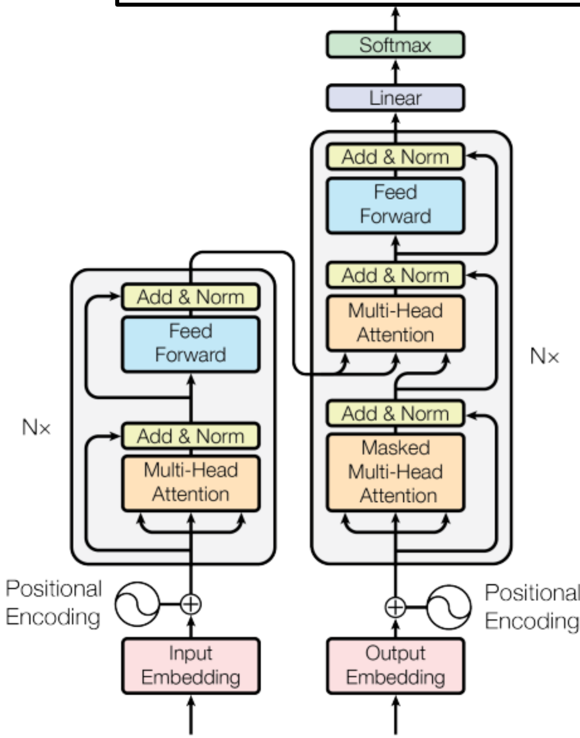
SOS Trees are tall

Transformer – Translation

Trees are tall **EOS**

Pr(EOS) = 0.99

…



Les arbres sont grands

SOS Trees are tall EOS

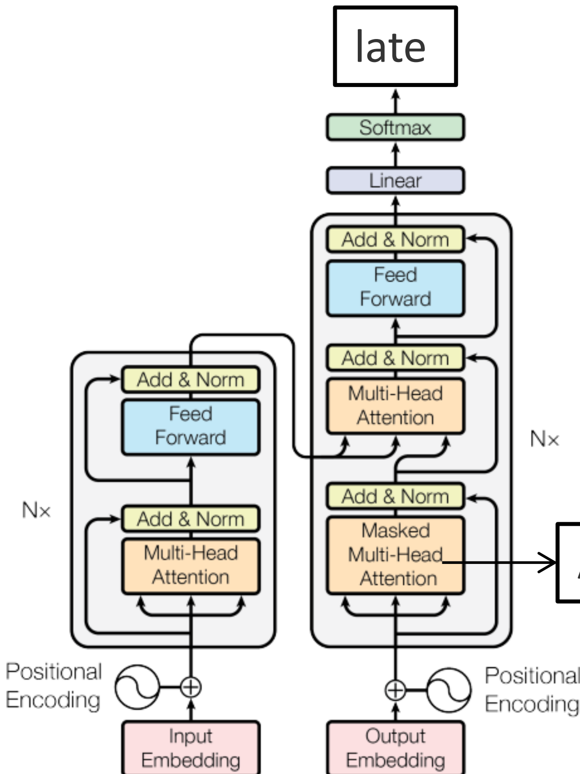
input

Transformer – Next word prediction (Training)

UNIVERSITY OF AUCKLAND Waipapa Taumata Rau NEW ZEALAND | SCIENCE

« late » is predicted considering the first 4 words of the input to the decoder.

Transformers computes all next words probabilities in parallel.
Masking is used to avoid « cheating ».



A wizard is never

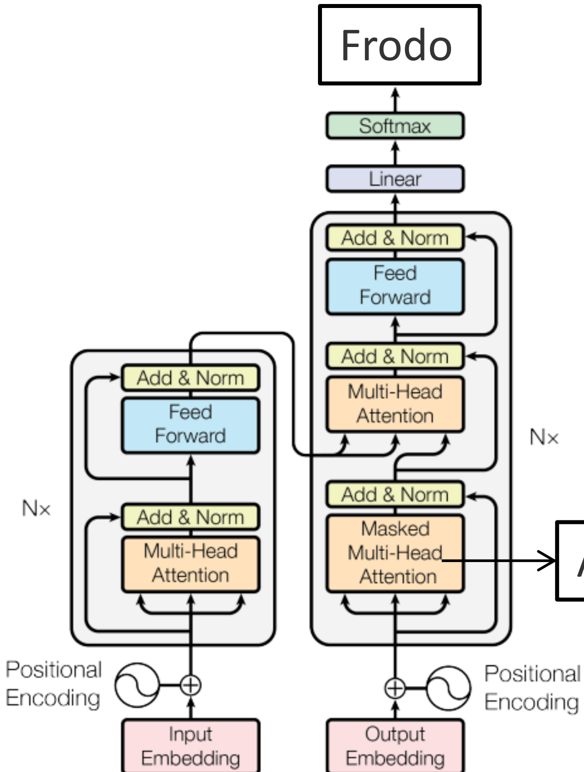
A wizard is never

input

A wizard is never late Frodo Baggins

whole output (**only when training**)

Transformer – Next word prediction



« Frodo » is this time predicted considering the first 5 words of the input to the decoder.

A wizard is never late [redacted] [redacted]

Transformers computes all next words probabilities in parallel.
Masking is used to avoid « cheating ».

A wizard is never

input

A wizard is never late Frodo Baggins

whole output (**only when training**)

Vision Transformer (ViT)

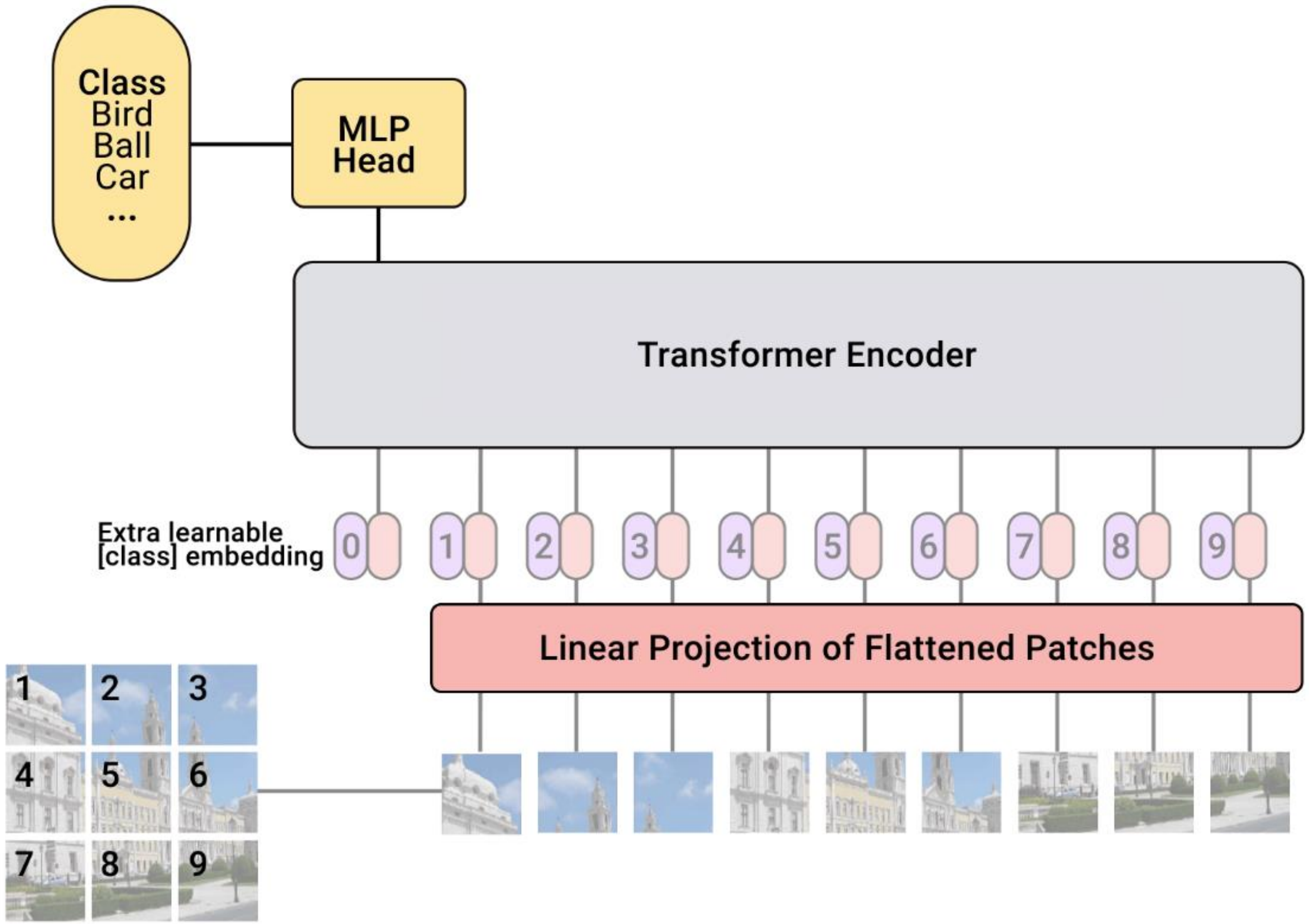


Image source: ai.googleblog.com/2020/12/transformers-for-image-recognition-at.html

Large language models

A **large language model** (LLM) is a general purpose language model consisting of a neural network with many parameters (typically billions of weights or more). LLMs trained on large quantities of unlabelled text perform well at a wide variety of tasks, a development which, since their emergence around 2018, has shifted the focus of natural language processing research away from the previous paradigm of training specialized supervised models for specific tasks.

Wikipedia

A **large language model** is an artificial neural network designed to analyze and generate natural language data. It is trained on vast amounts of text data and can perform various language tasks such as translation, summarization, and sentiment analysis. Large language models have revolutionized natural language processing, allowing machines to understand and generate human-like language with high accuracy.

ChatGPT

LLMs - A parameter story

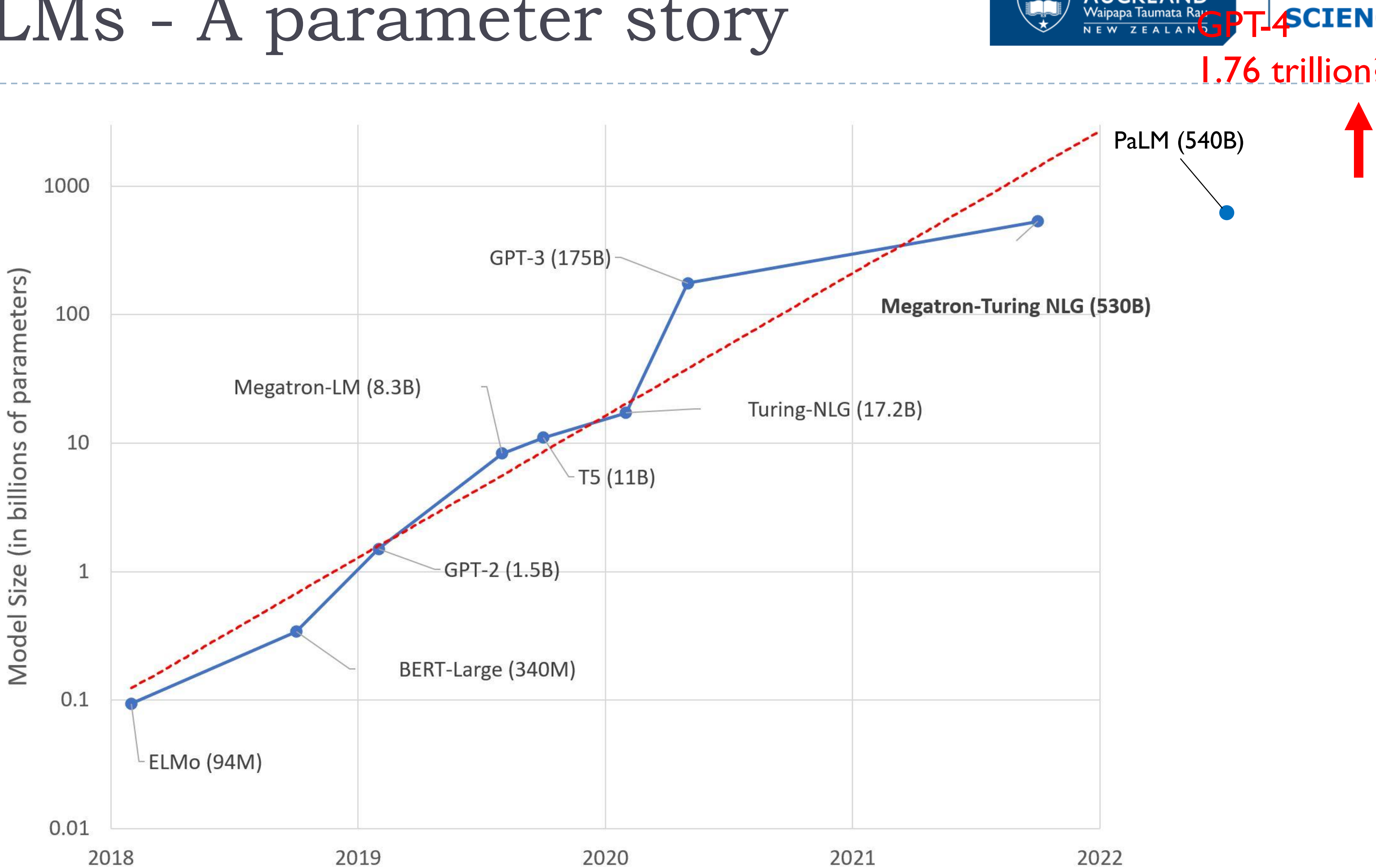


Image source: huggingface.co/blog/large-language-models

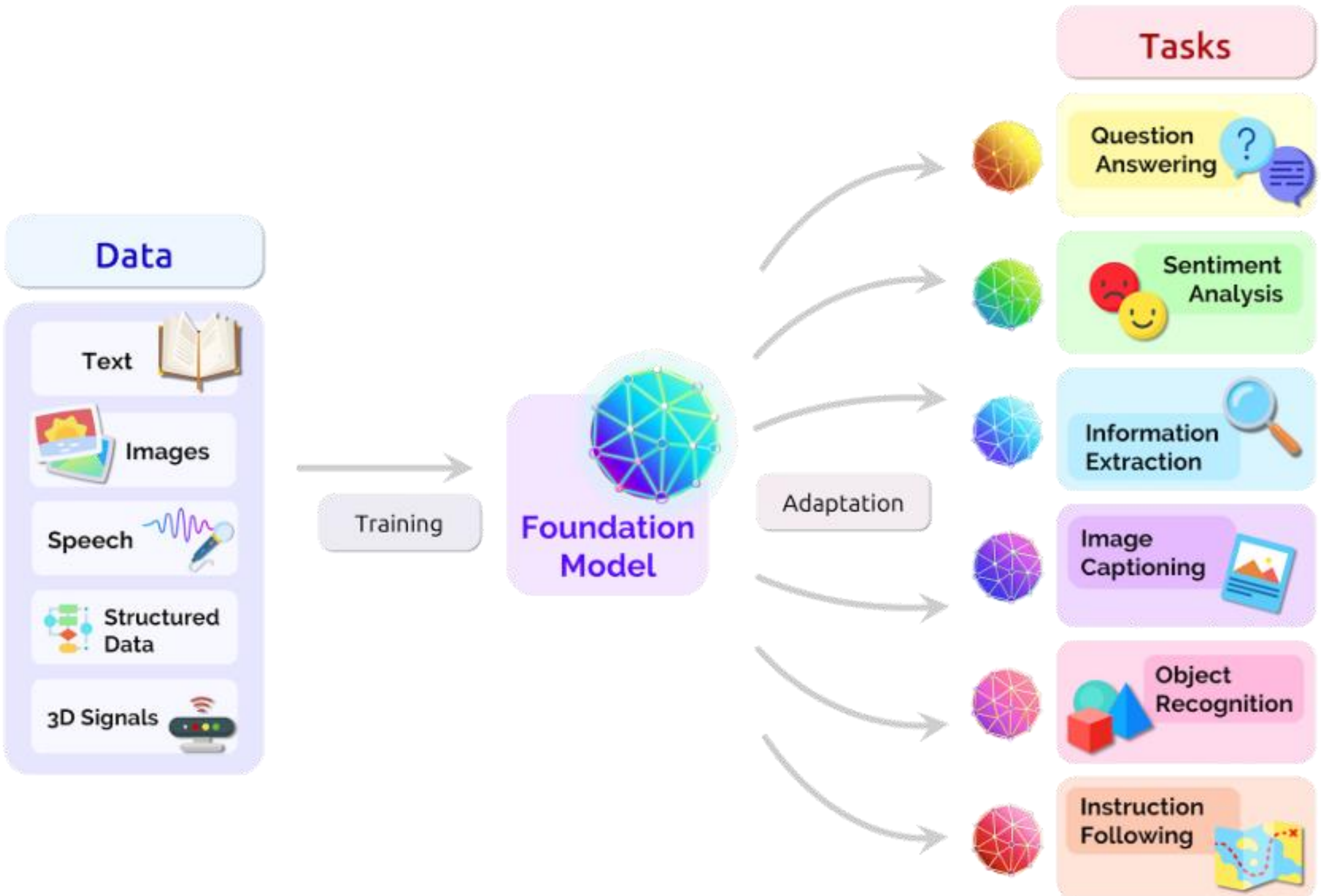
LLMs - Training

LLMs are trained following 2 phases:

1. Pre-training
   - Large amount of unlabelled data
   - Self-supervised learning (learn one part of the input from another part of the input)
   - General training (not task specific)
   - Computationally expensive

2. Fine-tuning
   - Labelled data
   - Specific to down-stream task (e.g., translation, summarisation, Q&A, …)
   - Computationally cheaper

LLMs – Foundation models



[On the Opportunities and Risks of Foundation Models, Bommasani et al., 2022]

LLMs – Popular models

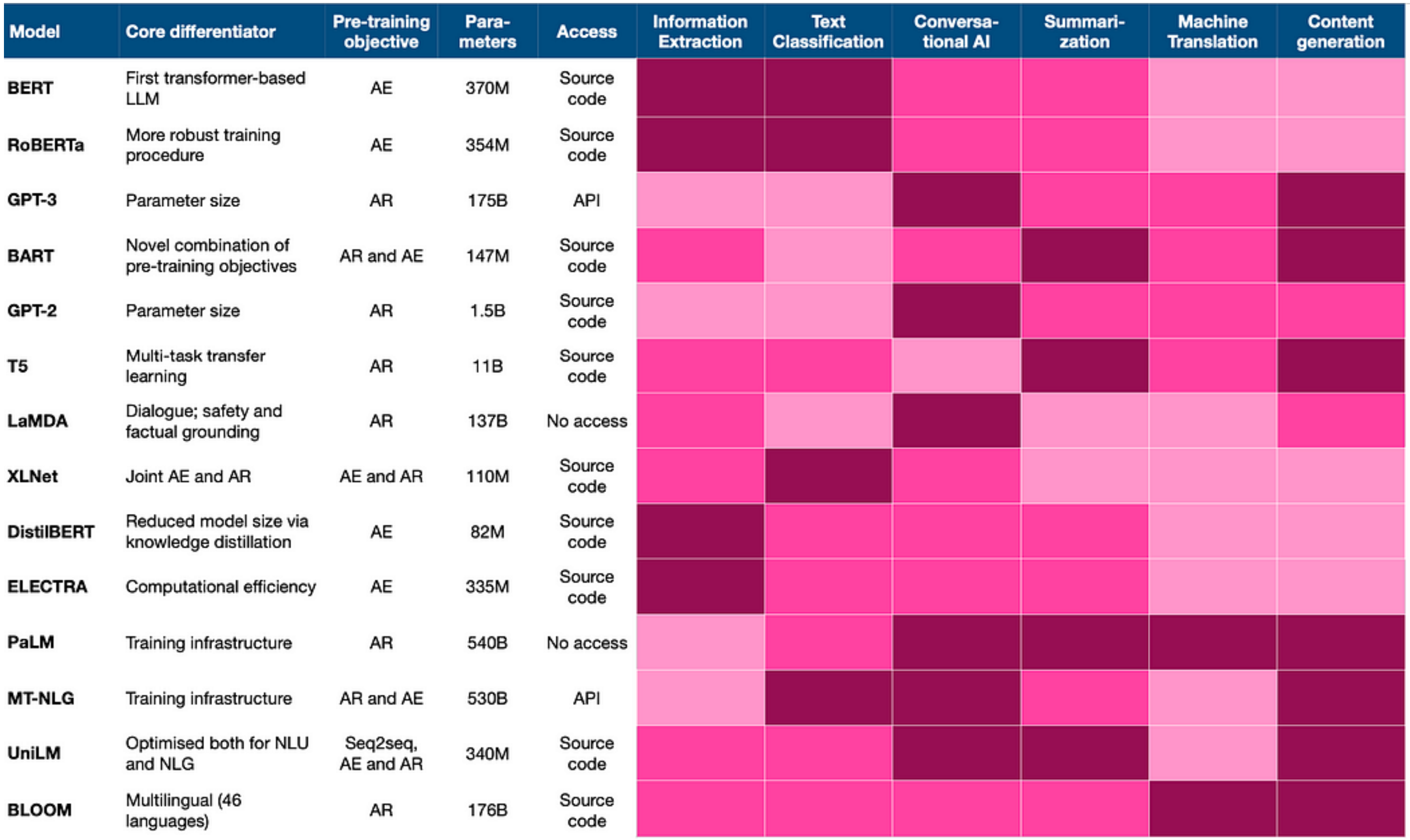
UNIVERSITY OF AUCKLAND Waipapa Taumata Rau NEW ZEALAND | SCIENCE

LLM	DEVELOPER	POPULAR APPS THAT USE IT	# OF PARAMETERS	ACCESS
GPT	OpenAI	Microsoft, Duolingo, Stripe, Zapier, Dropbox, ChatGPT	175 billion+	API
Gemini	Google	Some queries on Bard	Nano: 1.8 & 3.25 billion; others unknown	API
PaLM 2	Google	Google Bard, Docs, Gmail, and other Google apps	340 billion	API
Llama 2	Meta	Undisclosed	7, 13, and 70 billion	Open source
Vicuna	LMSYS Org	Chatbot Arena	7, 13, and 33 billion	Open source
Claude 2	Anthropic	Slack, Notion, Zoom	Unknown	API
Stable Beluga	Stability AI	Undisclosed	7, 13, and 70 billion	Open source
StableLM	Stability AI	Undisclosed	7, 13, and 70 billion	Open source
Coral	Cohere	HyperWrite, Jasper, Notion, LongShot	Unknown	API
Falcon	Technology Innovation Institute	Undisclosed	1.3, 7.5, 40, and 180 billion	Open source
MPT	Mosaic	Undisclosed	7 and 30 billion	Open source
Mixtral 8x7B	Mistral AI	Undisclosed	46.7 billion	Open source
XGen-7B	Salesforce	Undisclosed	7 billion	Open source
Grok	xAI	Grok Chatbot	Unknown	Chatbot

https://zapier.com/blog/best-llm/

LLMs – Features and tasks

UNIVERSITY OF AUCKLAND
Waipapa Taumata Rau
NEW ZEALAND
SCIENCE



Model	Core differentiator	Pre-training objective	Para-meters	Access	Information Extraction	Text Classification	Conversa-tional AI	Summari-zation	Machine Translation	Content generation
BERT	First transformer-based LLM	AE	370M	Source code						
RoBERTa	More robust training procedure	AE	354M	Source code						
GPT-3	Parameter size	AR	175B	API						
BART	Novel combination of pre-training objectives	AR and AE	147M	Source code						
GPT-2	Parameter size	AR	1.5B	Source code						
T5	Multi-task transfer learning	AR	11B	Source code						
LaMDA	Dialogue; safety and factual grounding	AR	137B	No access						
XLNet	Joint AE and AR	AE and AR	110M	Source code						
DistilBERT	Reduced model size via knowledge distillation	AE	82M	Source code						
ELECTRA	Computational efficiency	AE	335M	Source code						
PaLM	Training infrastructure	AR	540B	No access						
MT-NLG	Training infrastructure	AR and AE	530B	API						
UniLM	Optimised both for NLU and NLG	Seq2seq, AE and AR	340M	Source code						
BLOOM	Multilingual (46 languages)	AR	176B	Source code						

AR = Autoregression
AE = Autoencoding
Seq2seq = Sequence-to-sequence

Highly appropriate
Appropriate
Somewhat appropriate

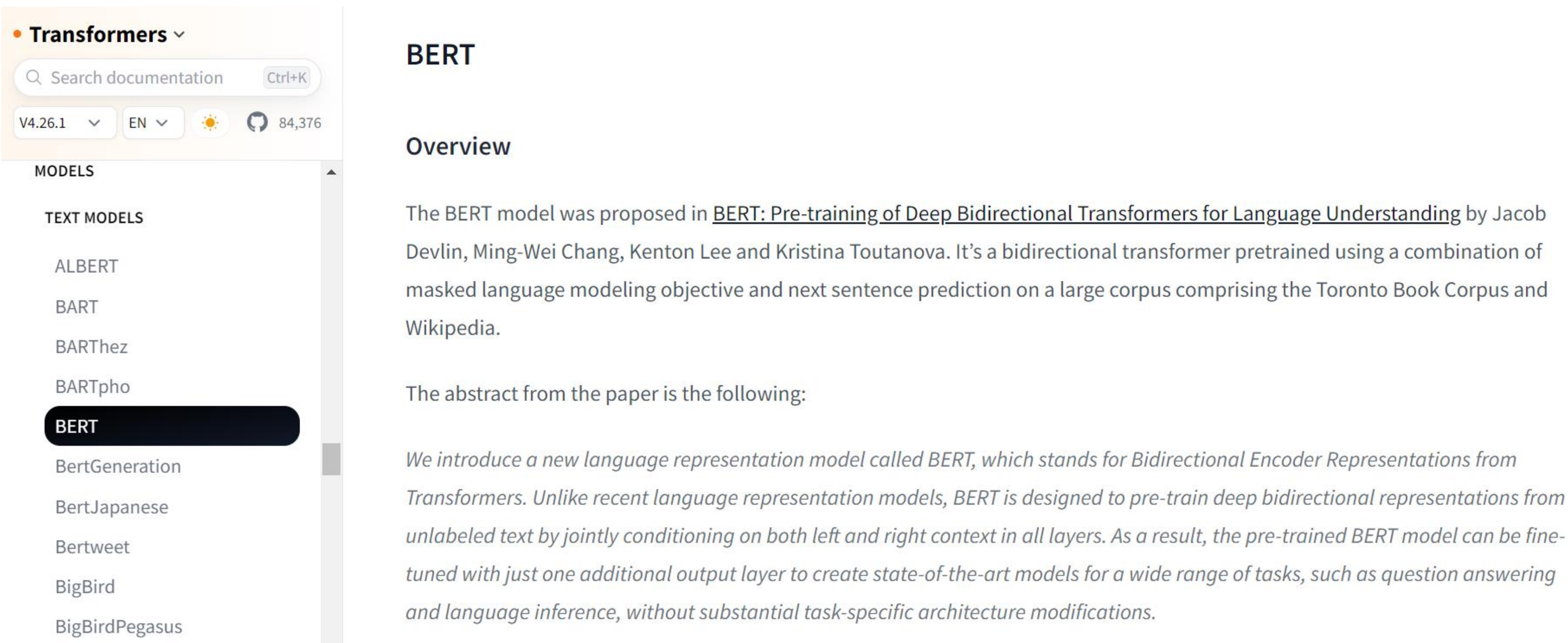
Image source: Janna Lipenkova - Choosing the right language model for your NLP use case

LLMs – Documentation and ressources

UNIVERSITY OF AUCKLAND Waipapa Taumata Rau NEW ZEALAND | SCIENCE

HuggingFace Transformers library

- Large collection of documentations and ressources about models and datasets.



Transformers

Search documentation Ctrl+K

V4.26.1 EN 84,376

MODELS

TEXT MODELS

ALBERT
BART
BARThez
BARTpho
BERT
BertGeneration
BertJapanese
Bertweet
BigBird
BigBirdPegasus

BERT

Overview

The BERT model was proposed in BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. It's a bidirectional transformer pretrained using a combination of masked language modeling objective and next sentence prediction on a large corpus comprising the Toronto Book Corpus and Wikipedia.

The abstract from the paper is the following:

We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

https://huggingface.co/docs/transformers/v4.26.1/en/model_doc/index

ChatGPT

- Chat-GPT is a chatbot based on the GPT-3.5 LLMs series.

- Fine-tuned on a variety of NLP tasks including translation, summarisation, Q&A and dialogue generation.

The model used for Chat-GPT is specially fine-tuned for chatbot applications, where the goal is to generate human-like responses to user inputs in a conversational manner.

ChatGPT

- Also fine-tuned using reinforcement learning.

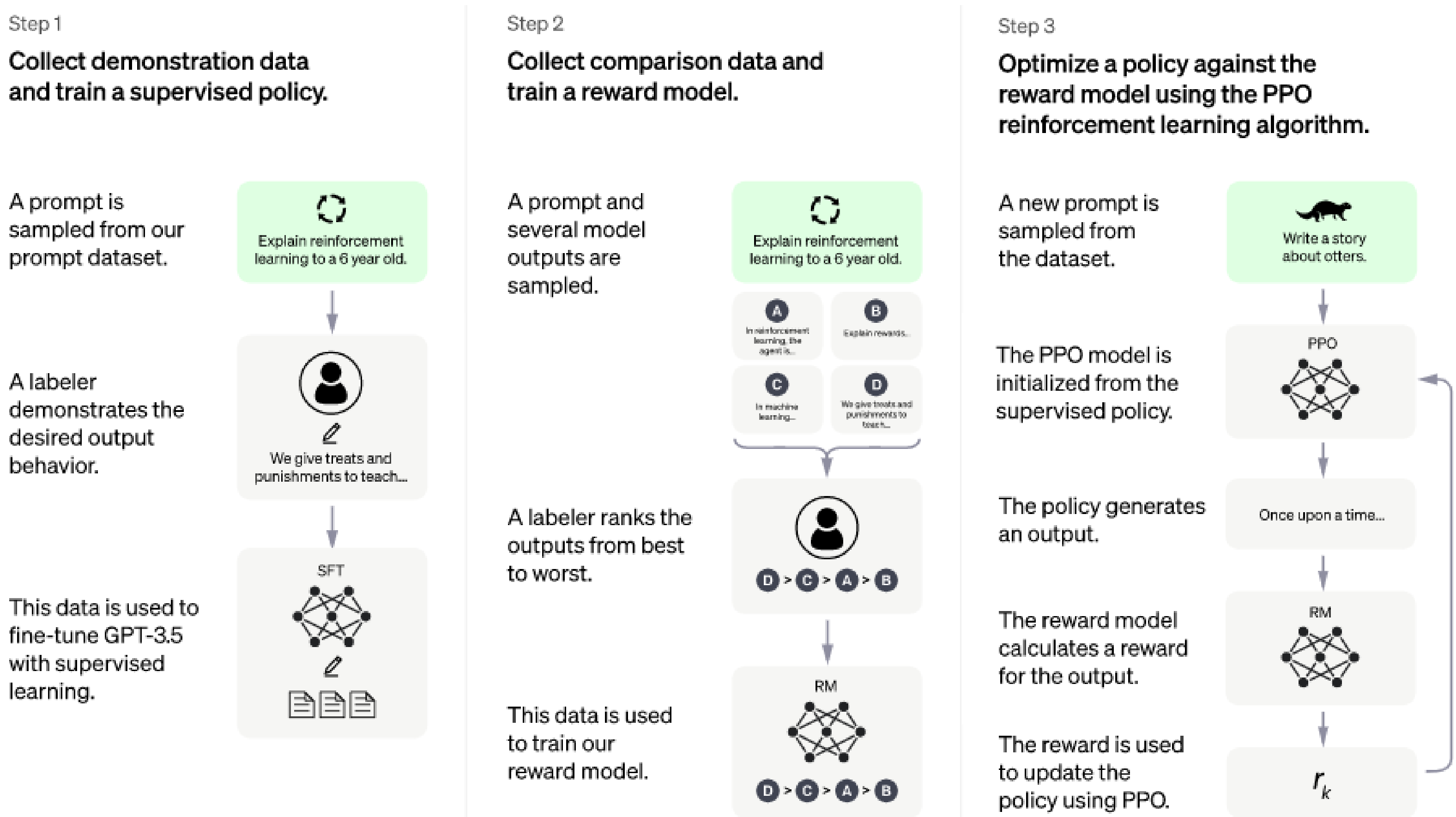


Image source: https://openai.com/blog/chatgpt/

Cost of training large language models

- Cost of training vs model size:
  - $2.5k- $50k (110 million parameter model)
  - $10k- $200k (340 million parameter model)
  - $80k- $1.6m (1.5 billion parameter model)

[The cost of training nlp models: A concise overview, Sharir et al., 2020]

ChatGPT:

  - Newest version (gpt-3.5-turbo): $0.002 per 1000 tokens (10x less than a few months ago)

- Training GPT-3 consumed an estimated 1,287 MWh (~65k inhabitant city consumption per day in NZ) and produced 552 CO2e (~80 Auckland-London return flights in economy class).

[The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink, Patterson et al., 2022]

Litterature

UNIVERSITY OF AUCKLAND Waipapa Taumata Rau NEW ZEALAND | SCIENCE

- Transformers/Attention: [Attention Is All You Need, Vaswani et al., 2017]
  https://papers.nips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

- Popular models based on Transformers:
  - Google AI BERT [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", Devlin et al., 2018]
    https://arxiv.org/pdf/1810.04805.pdf
  - OpenAI GPT-2 [Language Models are Unsupervised Multitask Learners, Radford et al., 2018]
    https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
  - OpenAI GPT-3 [Language Models are Few-Shot Learners, Brown et al., 2020]
    https://arxiv.org/pdf/2005.14165.pdf
  - Facebook AI BART [BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, Lewis et al., 2019]
    https://arxiv.org/pdf/1910.13461
  - Google AI T5 [Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, Raffel et al., 2020]
    https://arxiv.org/pdf/1910.10683.pdf

Literature

UNIVERSITY OF AUCKLAND Waipapa Taumata Rau NEW ZEALAND | SCIENCE

- NVIDIA Megatron-LM [Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism, Shoeybi et al., 2020]

  https://arxiv.org/abs/1909.08053
- Microsoft Turing & NVIDIA [Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model, Smith et al., 2022]

  https://arxiv.org/pdf/2201.11990

Vision Transformers:

- [Image transformer, Parmar et al., 2018]

  http://proceedings.mlr.press/v80/parmar18a/parmar18a.pdf
- [An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, Dosovitskiy et al. , 2020]

  https://arxiv.org/pdf/2010.11929.pdf

  https://ai.googleblog.com/2020/12/transformers-for-image-recognition-at.html