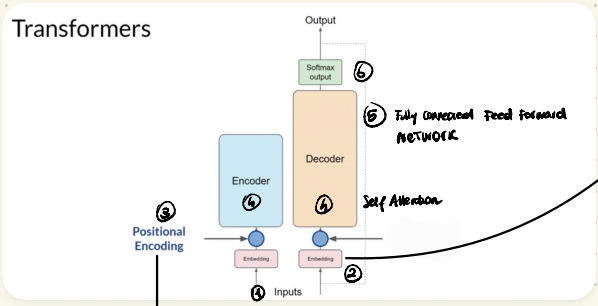
Week 1

① TOKENIZER: Converts words into numbers, each # represents a position within a dictionary

Input (as a number) can be passed to the



② EMBEDDING LAYER

Trainable - vector - embedding - space

each token is represented as a vector and occupies a unique location within that space

Vectors encode the meaning & context of the tokens in the input sequence

③ ADD positional encoding

preserve info about word order that would get lost since the model processes all input tokens in parallel!

④ Pass the vectors to the self attention layer

the model analyzes the relationship between the tokens in the input

self attention weights are learned during training and stored in these layers

↓

show dependencies between the words

multi-headed self attention: multiple sets of attention weights (HEADS) are learned in parallel, independently of each other

~ 12-100 Attention Heads are learned

each learns a different aspect of language (ACTIVITY, PEOPLE)

⑤ Now that attention weights are added to the input data

the output is processed through a fully connected FEED-FORWARD Network

The output is a vector of logits proportional to the probability score for each token in the dictionary

⑥ Logits are passed to a Softmax Layer where they are normalized in a probability score for each word.

1 TOKEN has a score higher than the rest : most likely predicted token
Various selection methods from this vector of probabilities

END-TO-END PROCESS

① Input sequence is fed in the ENCODER → embedding layer → fed into multi headed attention layer → FFN → we have a deep representation of structure + meaning of input sequence

② This representation is inserted in the middle of the decoder to influence its self attention mechanism.

③ A START-OF-SEQUENCE token is added to the input of the decoder. Decoder is triggered to predict the next token based on [contextual understanding from ENCODER]

④ output of Decoder's self attention layer → FFN → Softmax layer → 1ST TOKEN!

⑤ Feed the output token back to the input UNTIL the model produces an END-OF-SEQUENCE token

⑥ Final series of tokens is detokenized into words

IN CONTEXT LEARNING = providing examples inside the context window

zero shot inference ⟶ ok for large models

1/few shot inference → 1 or more examples

Configuration Parameters

Greedy decoding ⟶ model chooses the token with highest probability

↳ only for short generation bc of repeated sequences

Random sampling ⟶ choose at random using the probability distrib. to weight the selection

↳ could wander too much, too random

top-K and top-p sampling → increase chance that output is sensible

- top-K: choose among top K token with ↑ probab.
- top-p: choose among top ranked tokens with cumulative_probability < p

Temperature → influence shape of probability distribution of the next token

↳ the higher it is, (>1) the higher the randomness.

↳ if =1 softmax function left as default

Gen-AI Projects

Use case → model → Adapt → integration

- use case: many tasks or 1 specific?
- model: existing vs trained
- Adapt:
  - prompt eng.
  - fine tuning
  - RL with human feedback
  - ↳ evaluate
- integration: ↳ optimize resources

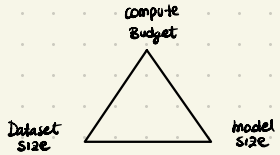
Model Architectures

ENCODER ONLY
- aka autoencoding
- masked language modeling (predict 1 token to reconstruct original input)
- bi-directional context
- sentiment analysis, NER, word classification
- BERT ROBERTA

DECODER ONLY
- aka Autoregressive
- causal language modeling
- objective: → predict next token
- context is unidirectional (only previous ones)
- Text Generation + works well for ∅ shot
- GPT BLOOM

ENCODER DECODER
- Sequence-to-sequence. Body of text as input & output
- many objective
- → popular is SPAN CORRUPTION
  - 1 encoder masks random tokens → Sentinel token
  - 2 decoder predict masked tokens
- T5 BART
- Translation, Summarization, Q&A

Training compute optimal LLM



→ chinchilla scaling law (compute optimal): the optimal dataset size is ~ 20x the number of parameters

behind Bloomberg GPT (50B param)

If domain uses vocabulary and language structure → pre-training is needed