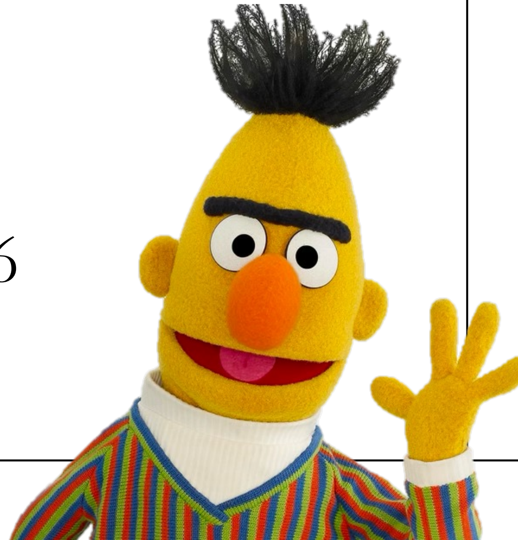
Advanced NLP -

Session 4: Transformer Based Models

Prof. Dr. Richard Sieg

TH Köln IWS - WS 25/26



Agenda

01. Tokenizer

02. Encoder Based & BERT

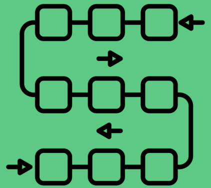
03. Decoder Based & GPT

Encoder-Decoder BART & T5

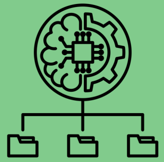
Pre-Training

Tutorial

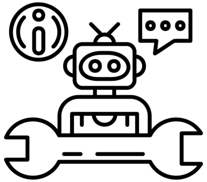
The 3 Ingredients of LLMs



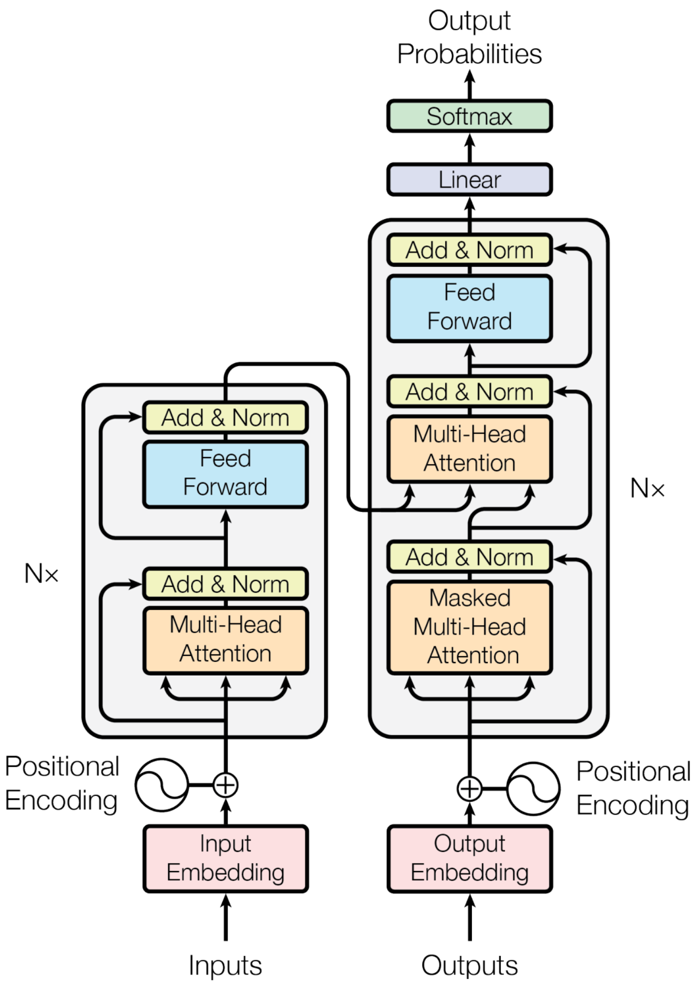
Process long sequences and context



Efficient training on huge datasets

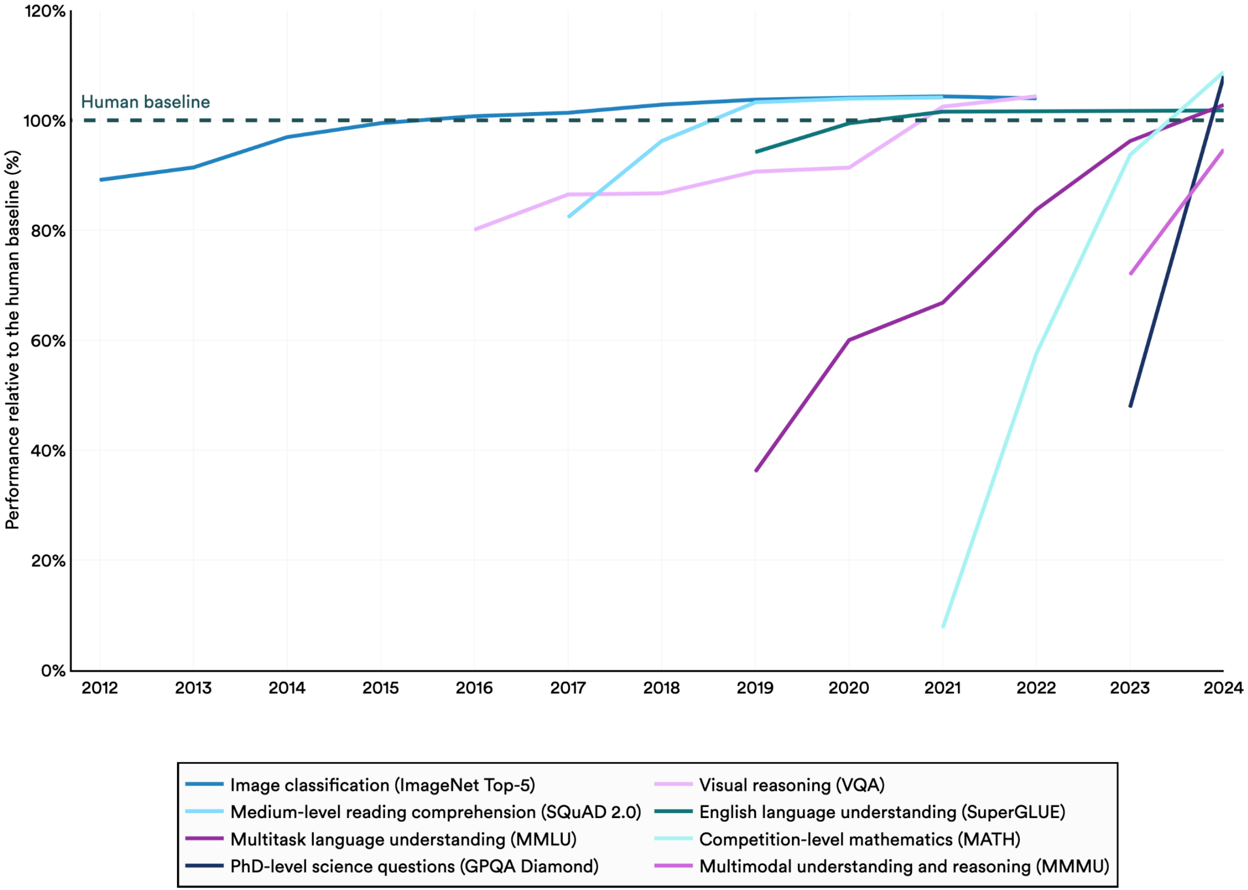


Follow (human) instructions


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs

Select AI Index technical performance benchmarks vs. human performance

Source: AI Index, 2025 | Chart: 2025 AI Index report

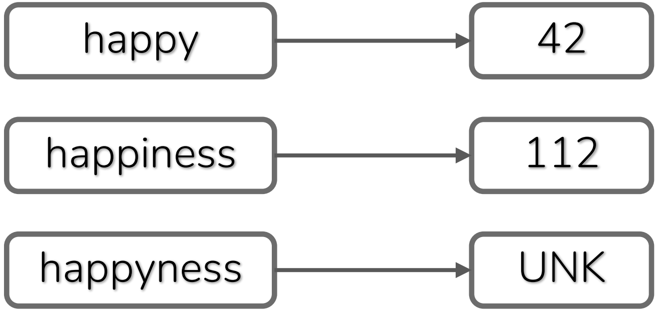


01.

Tokenizers

Fixed Vocabulary

- Up until now, we worked with a fixed vocabulary that we assign indices to
- Novel words are mapped to an <UNK> token
- **Issue 1:** We probably end up with a huge vocabulary
- **Issue 2:** We are unable to map novel words or leverage subword relations (e.g. "*happiness*" relates to "*happy*")



Tokenizers

- ***Token:*** Any sequence of characters – mostly part of words (subwords)
- ***Tokenization:*** Splitting raw text into tokens

We <3 Python!

- Words: [“We”,”<3”,”Python”,”!”]
- Substrings of length 3: [“We#”,”<3#”,”Pyt”,”hon”,”!##”]
- BPE Tokenizer (GPT): [“We”, “#<“,”3”,”#Python”,”!”]

GPT Tokenizer Playground

Byte Pair Encoding (BPE)

- **Goal:** Build a subword vocabulary up until a given size (e.g. 40k tokens)
- 1. Start with all ASCII characters and a symbol for "end of word"
- 2. We then check all pairs of characters and calculate their frequency in a corpus – the most common pairs are merged into new subwords
- 3. Replace the instances of these pairs with the subwords and continue with the algorithm until the subword vocabulary has the desired size
- This tokenizer algorithm is used for most LLMs as of today

See also: HuggingFace LLM Course

WordPiece Tokenizer

- Tokenizer developed by Google for their BERT model
- Similar to BPE but merges not by frequency but a score that takes the frequency of the two parts in a merge into account

  $$score(part_1, part_2) = \frac{freq(pair\ part_1 \& \ part_2)}{freq(part_1) \cdot freq(part_2)}$$

- This also takes into account what we loose by merging two subwords
- More difficult to implement and we might get out of vocabulary characters since we worked with predefined character set

02.

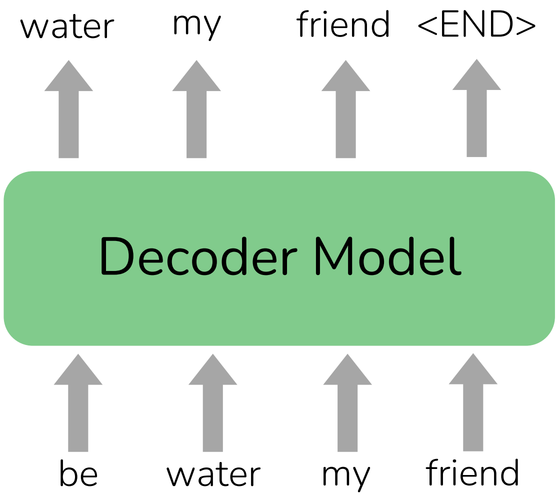
Pre-Training

From Pre-trained Word Emebddings to Pre-trained Models

- Up until 2017 we used pre-trained word embeddings as initial inputs for training an NLP task in an LSTM or Transformer
- These models are initialized with random parameters
- The models itselves have no initial idea of how language works and all contextual aspects of language
- Nowadays: The neural networks we use for a specific task are pre-trained on huge corpora
- Pre-trained models already have a strong representation of language and are a powerful initialization for neural networks in NLP
- The ImageNet moment of NLP

Pre-Training with Unsupervised Learning

- Pre-Training a model works by masking some parts of the input and train the model to reconstruct these parts
- This allows us to process massive amounts of data without any labels (unsupervised)
- One option: Pretraining through language modeling, i.e. masking the next word and let the model predict the next word
  → Decoder Models (GPT etc)
- Or mask some random parts of the input like a cloze
  → Encoder Models (BERT etc)

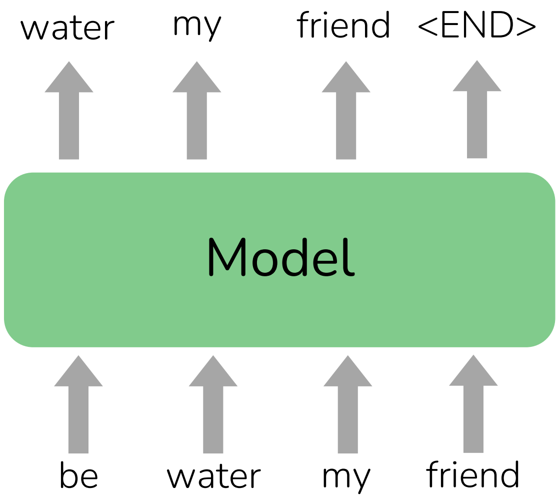


Pretraining through Language Modeling

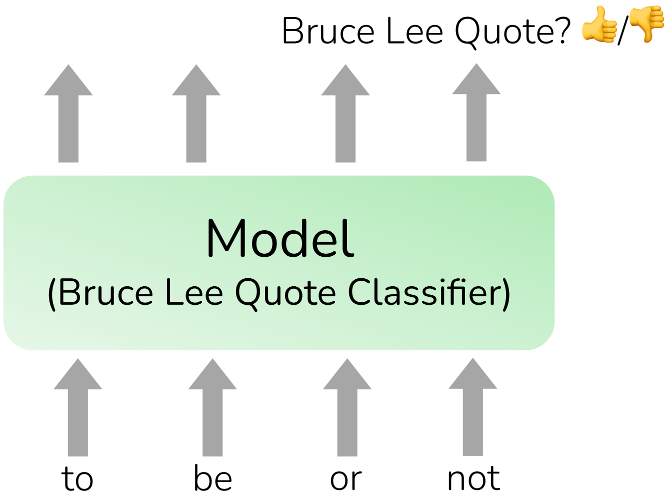
Pre-Training Paradigm

- Nowadays, we take a pre-trained model for our language (or a multilingual model) and use this as initial parameters to train our downstream task

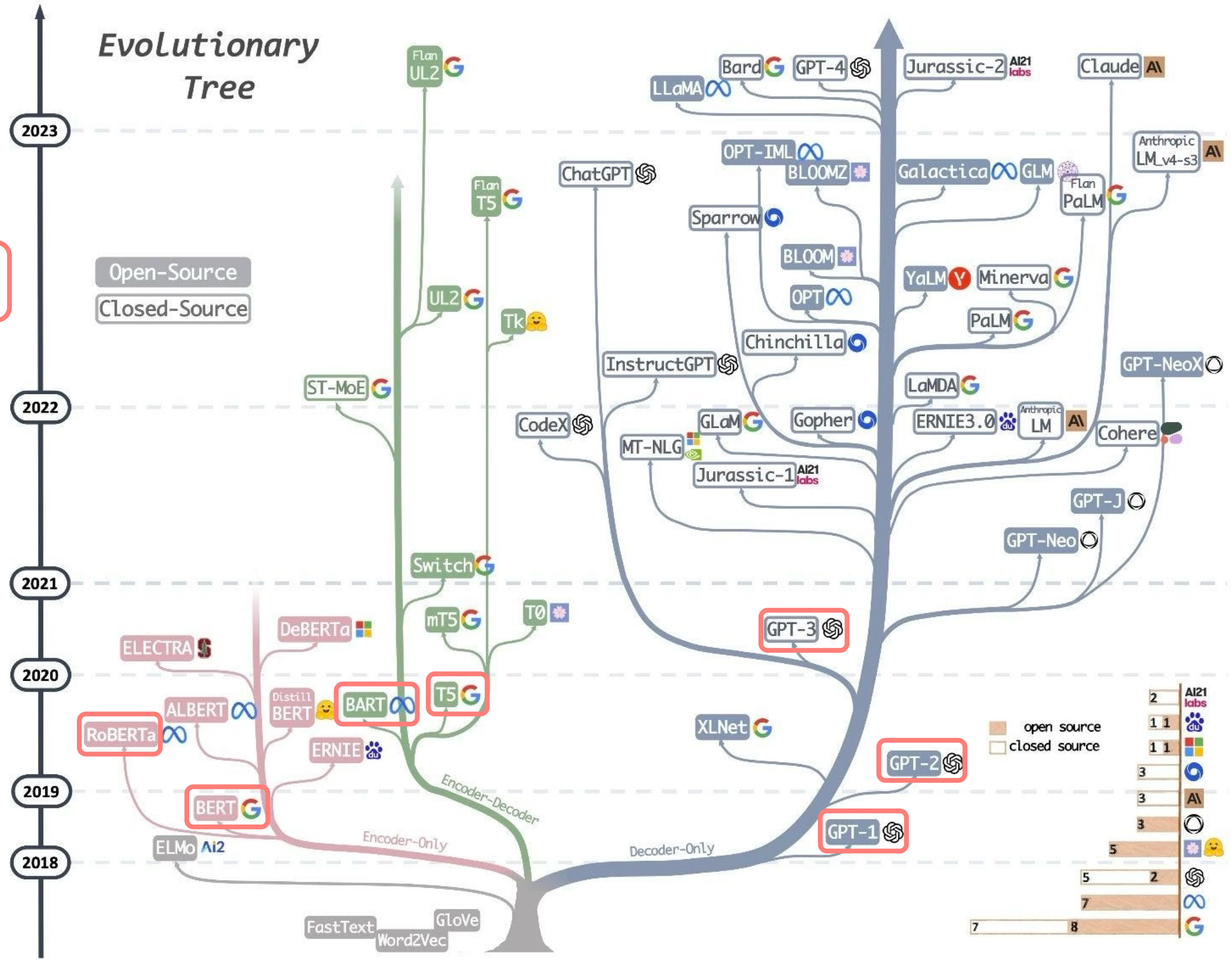
Step1: Pre-train model on lots of text



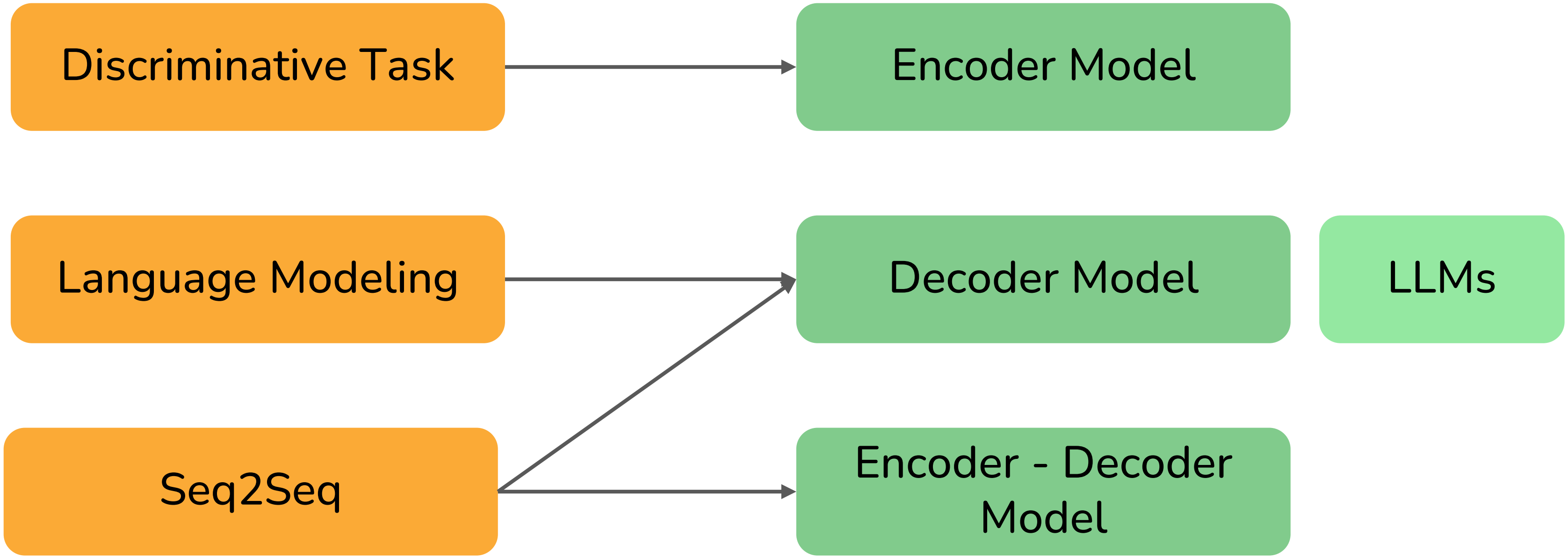
Step2: Fine-tune for your task



This lecture



Which Model for Which Task?

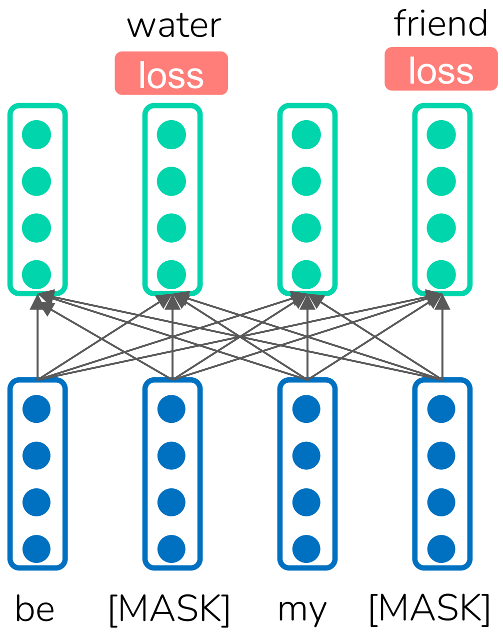


03.

Encoder Models & BERT

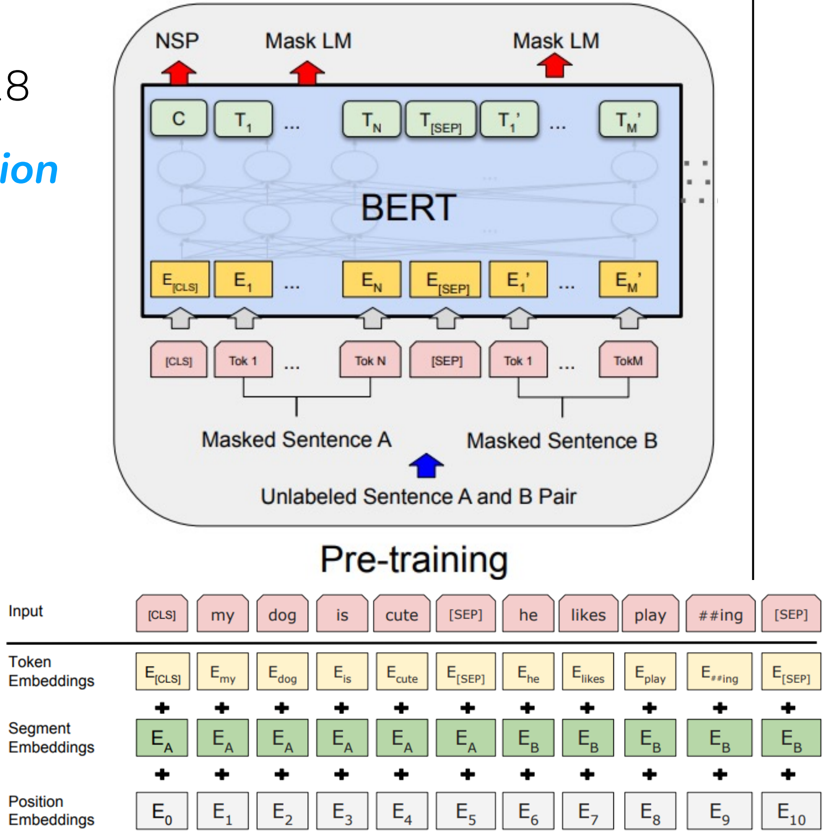
Pre-Training Encoders

- Encoders get bidirectional context and cannot be trained for language modeling
- ***Masked Language Modeling:*** Replace random words with [MASK] token and train the model to predict these words (like a cloze)
- We only compute the loss for the wßords that are masked out



BERT: Bidirectional Encoder Representations from Transformers

- Encoder based model developed by Google in 2018
- Trained on Masked LM and ***next sentence prediction***
- Masked LM details:
    - 15% of tokens are masked and out of these
    - 80% are replaced with [MASK]
    - 10% replaced with random other token
    - 10% are unchanged but still predicted
- Next sentence prediction: predict whether one sentence follows the other or randomly sampled
    - Other BERT variants neglect this



BERT [Devlin et al 2018]

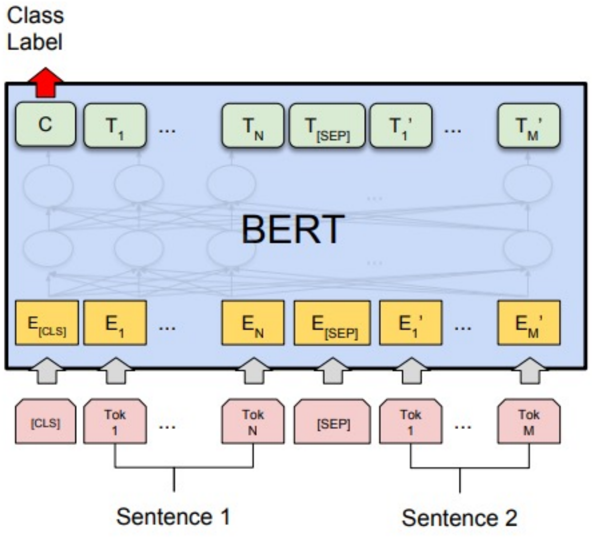
BERT Training Details

- BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params.
- BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on:
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- BERT was pretrained with 64 TPU chips for a total of 4 days.

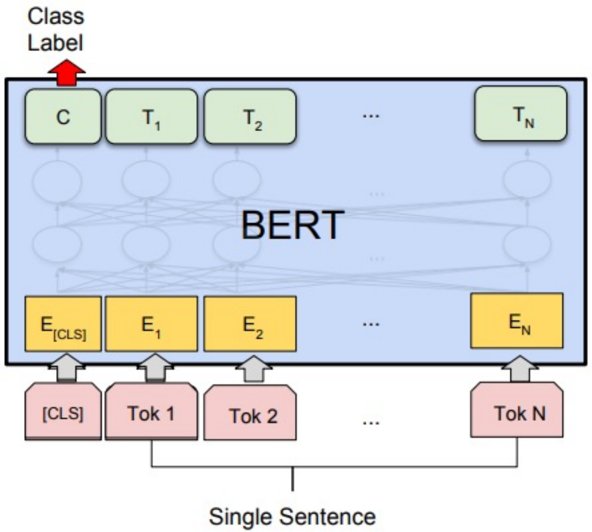
BERT [Devlin et al 2018]

BERT Fine-Tuning Task – Text Classification

- For Text Classification, we start every sentence with a special [CLS] token
- The context vector of this token can then be used in a linear layer and softmaxed

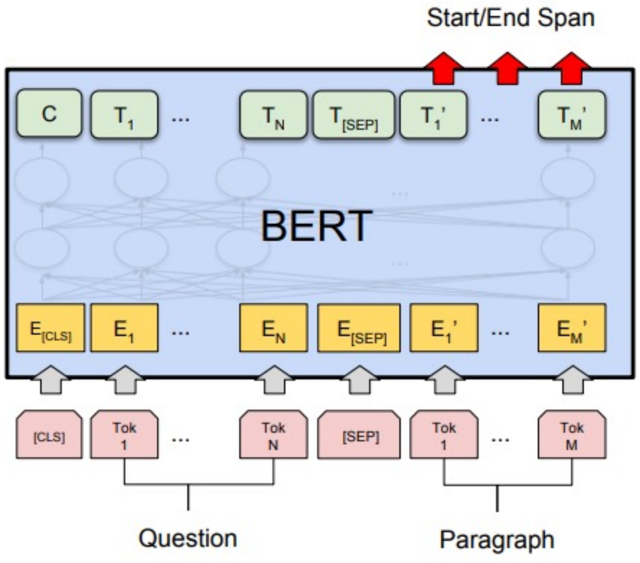


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

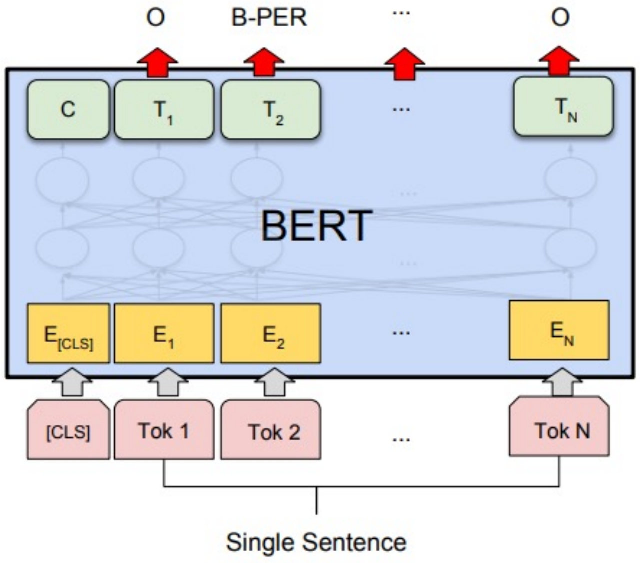


(b) Single Sentence Classification Tasks: SST-2, CoLA

BERT Fine-Tuning Task – QA & Tagging



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

BERT Success

- By fine-tuning, BERT yielded state-of-the-art results on many tasks
- Is considered the powerhouse of modern NLP - in particular its variants
- Use it for any discriminative task in particular text classification

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard).

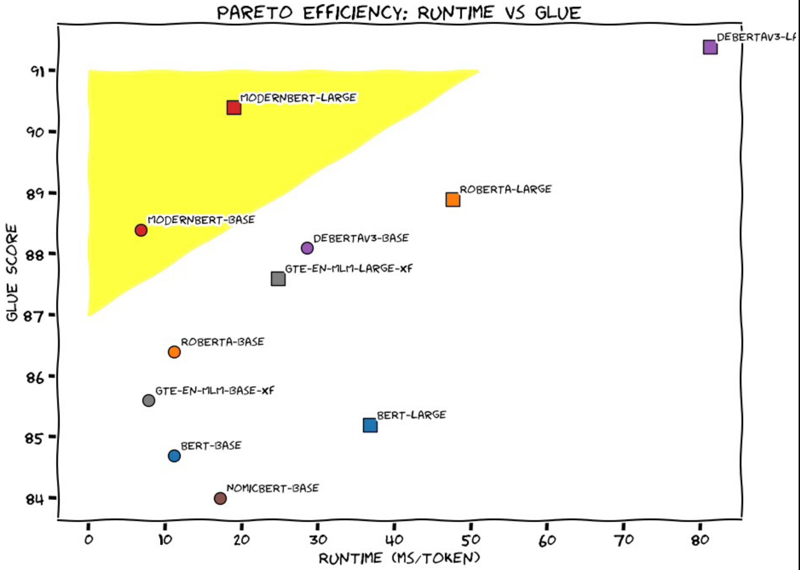
BERT [Devlin et al 2018]

BERT Variants

- RoBERTa [2019]
  - No next sentence prediction
  - Dynamic masking (other masks every epoch)
  - BPE tokenizer instead of WordPiece
  - Small training adjustments
- DistilBERT
  - Much smaller version (40% less params) trained via student-teacher
- Lots of variants for different languages or use cases – just search HuggingFace 🤗

modernBERT (12/2024)

- 8192 Tokens in the Context Window (vs 512 Vanilla BERT)
- Improved Transformer architecture using global and local Attention
- Lots of smaller tricks from years of LLM training experience
- EuroBERT (03/2025)
    - Basically modernBERT trained on European languages

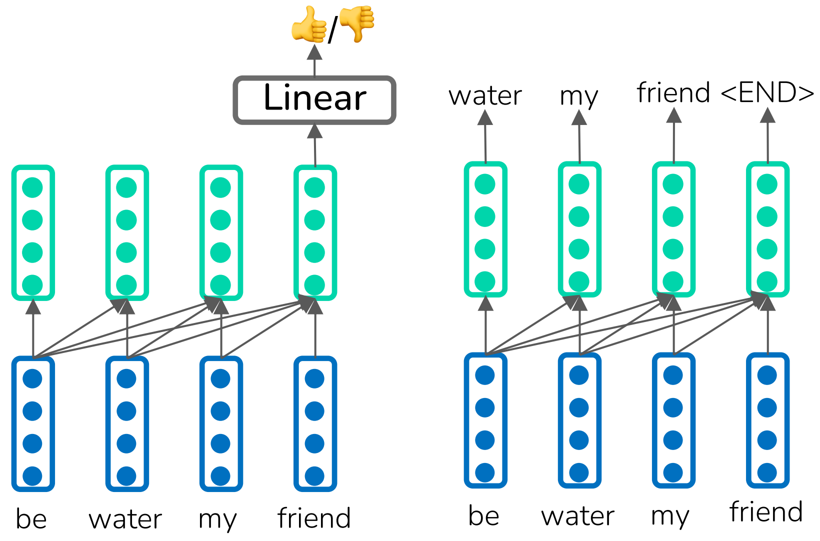


04.

Decoder Models & GPT

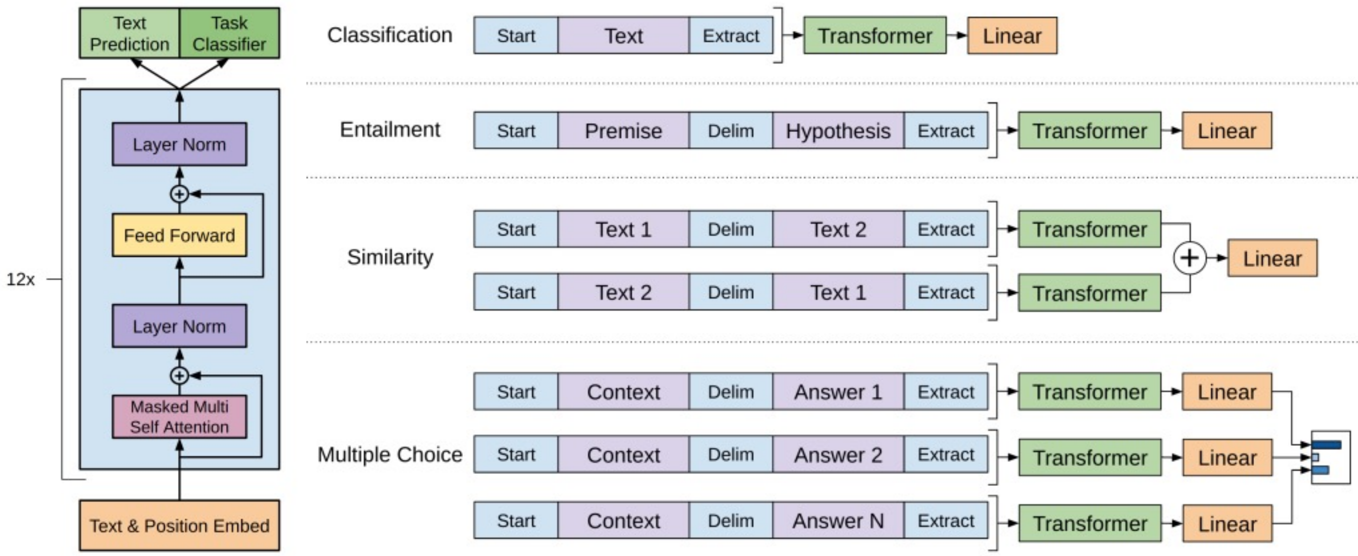
Pre-Training Decoders

- Decoders can also be fine-tuned for classification by adding a linear layer to the last hidden state (inferior to Encoder)
- More often: fine-tune them for generative seq2seq task, i.e. fine-tuning the language modeling task for a specific task



Generative Pretrained Transformer (GPT)

- Transformer Decoder with 12 layers
- BPE tokenizer of size 40k
- Trained on BookCorpus



Improving Language Understanding by Generative Pre-Training [Radford et al, 2018]

GPT2 [2019]

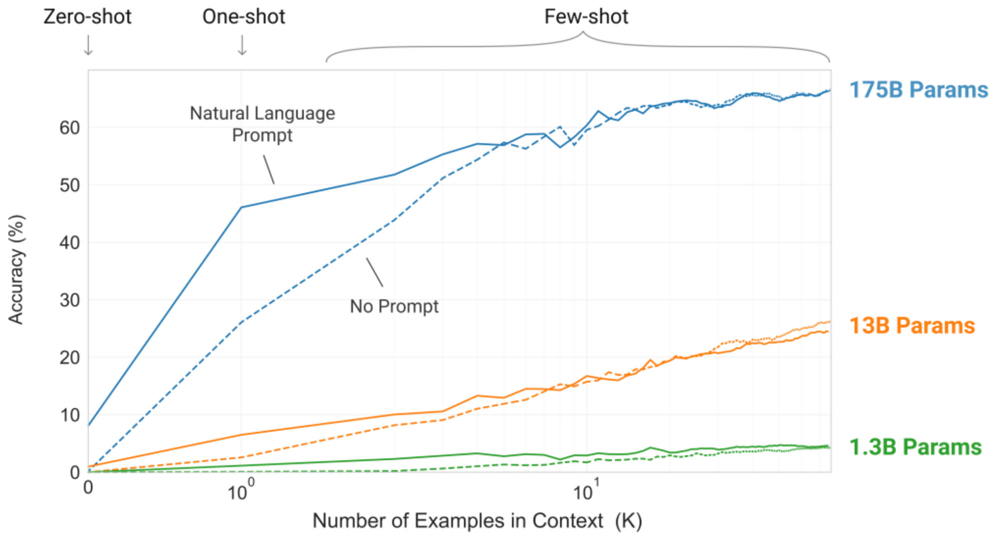
- Jump from 117m to 1.5b parameters and trained on much larger dataset (40Gb)
- First model that showed good results in ***zero-shot or few-shot prompting***
- Solve a task without fine-tuning, only prompt it with no or some examples
- And scaling seems increases the performance

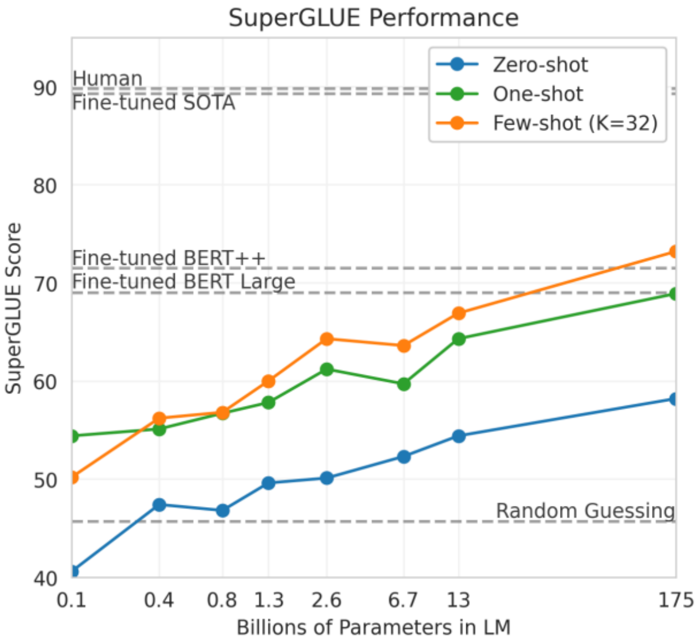
GPT3 [2020]

- Jump from 1.5b to 175b parameters and trained on 570Gb of data
- First model that performs exceptionally well with few-shot prompting
- ***The end of the fine-tuning paradigm!?***

Language Models are Unsupervised Multitask Learners [Radford et al, 2019] Language Models are Few-Shot Learners [Radford et al, 2020]

Few-Shot Capabilities of GPT 3





Language Models are Few-Shot Learners [Radford et al, 2020]

GPT Development

1 GPT-1

- Released: 2018
- 117 million parameters
- Trained using unsupervised learning
- Gave state of the art performance for LAMBADA.
- Gave Competitive performance for GLUE and SQuAD

2 GPT-2

- Released: 2019
- Improved version of GPT-1.(1.5 billion Params)
- Trained on larger dataset.
- Additional improvements like including a modified training objective, and a more efficient sampling algorithm for generating text.

3 GPT-3

- Released: 2020
- Improved version of GPT-2 (175 billion Params)
- Trained on larger dataset.
- Additional improvements like better training methodology (Gshard) and Few shot learning Capability.

4 GPT-3.5

- Released:2022
- Improved version of GPT-3
- Three variants , 1.3 B, 6B and 175 B params
- The main element is to eliminate toxic outputs to certain extent using RLHF (reinforcement learning with human feedback).

5 GPT-4

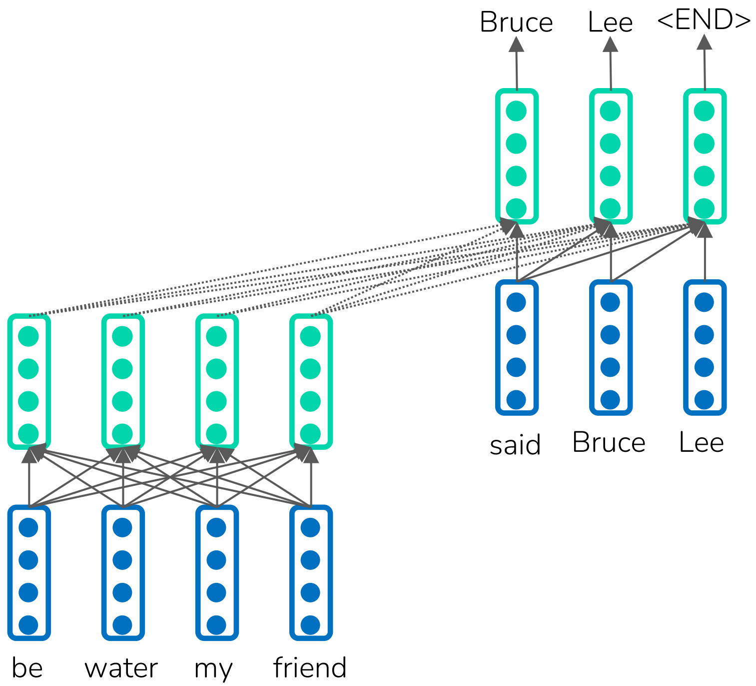
- Released: 2023
- Not sure if its a fine tuned version of previous model.
- 1 trillion Parameters
- It can take text and image as inputs and produces text outputs.
- RLHF is used in the training.

05.

Encoder Decoder Models

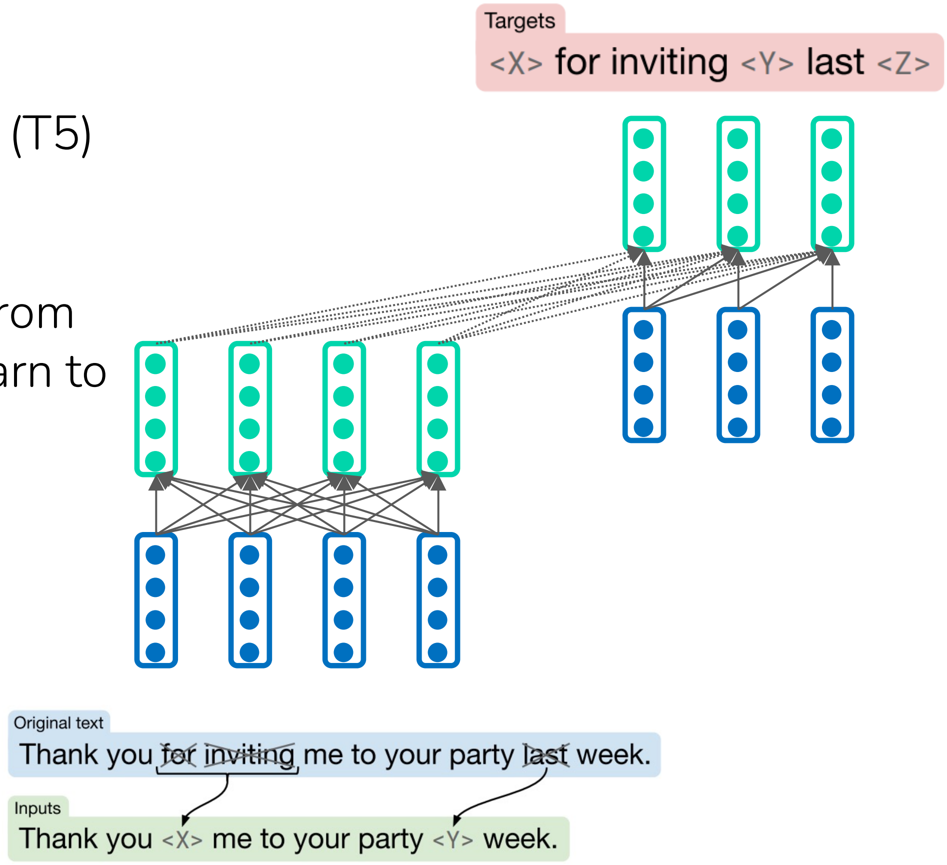
Pre-Training Encoder-Decoder

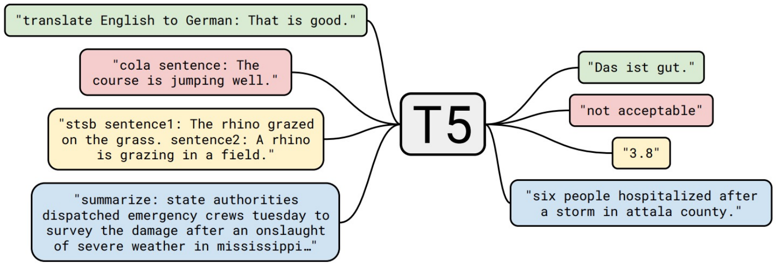
- Train like language modeling but give the first half of a sequence to the encoder and do not predict it and the decoder learns to predict the second half of the sequence



T5 [Google, 2019]

- Text-to-Text Transfer Transformer (T5)
- Trained on ***span corruption*** task
- Replace spans of different length from the input with placeholders and learn to fill the spans

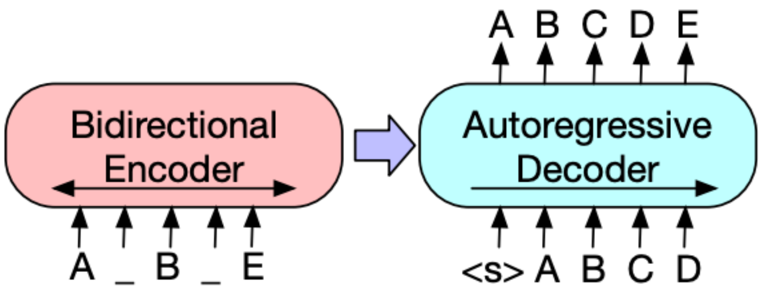


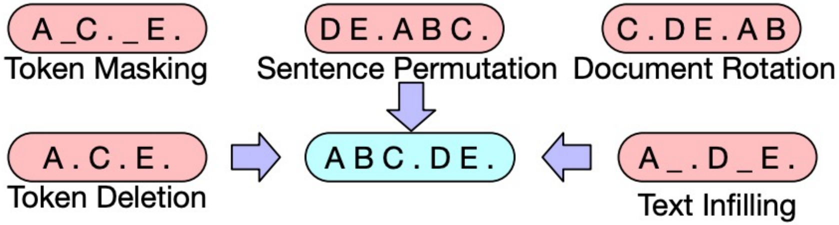


Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer [Raffel et al, 2019]

BART [Meta / FAI, 2019]

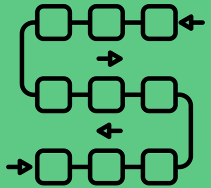
- Mixes BERT and GPT training
- Adds noise to documents in multiple ways and then learns to reconstruct the original document (***denoising***)



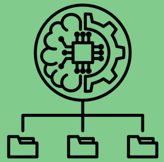


BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension [Lewis et al, 2019]

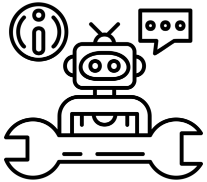
The 3 Ingredients of LLMs



Process long sequences and context



Efficient training on huge datasets



Follow (human) instructions

Tutorial