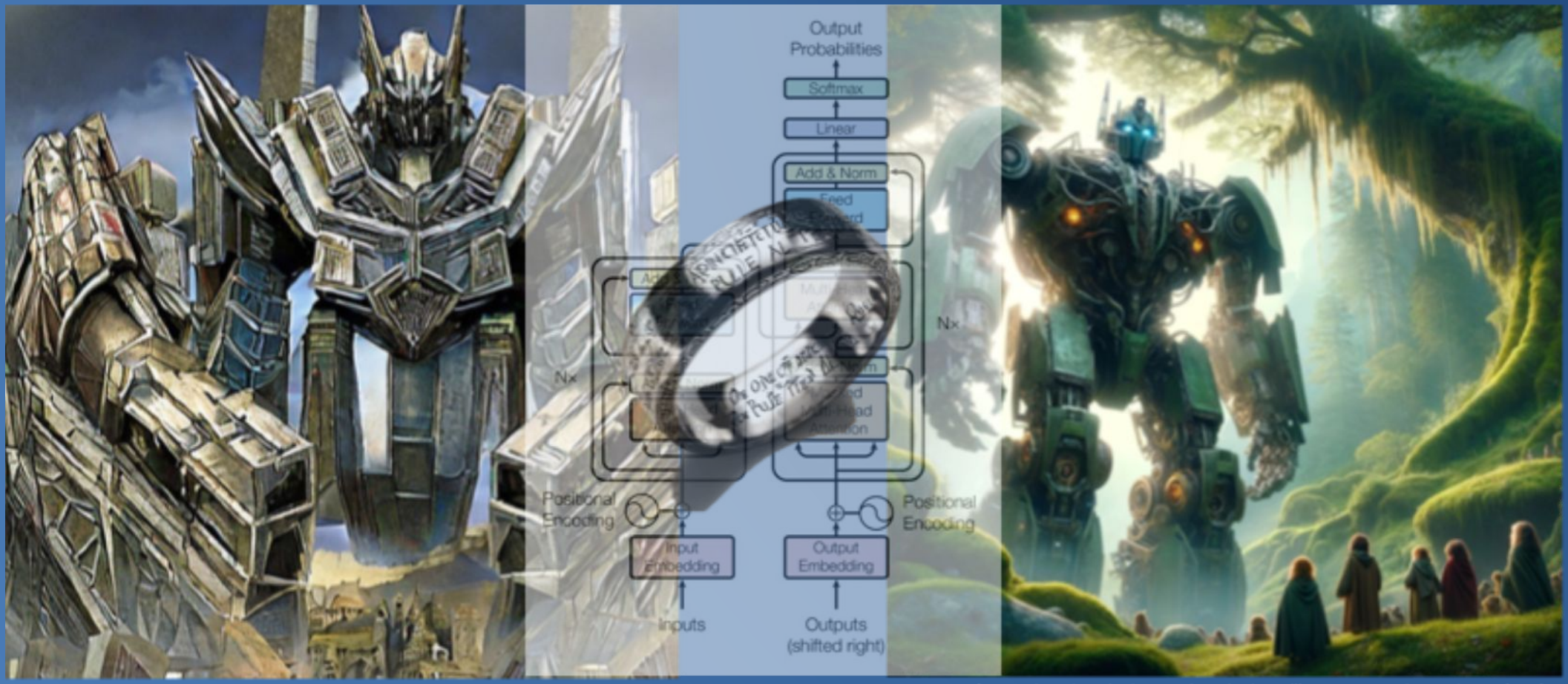
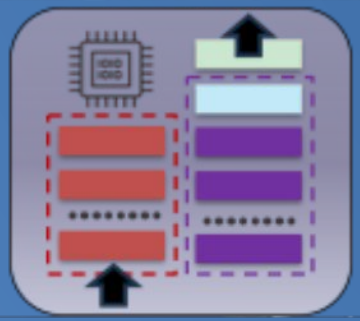



Image: Transformers: ' ...and one 'architecture' to rule them all' – Juxtaposition by Raeid Saqur (2024).



# More Transformers

CSC401/2511 – Natural Language Computing – Spring 2026

Ken Shi and Gerald Penn

Lecture 9

University of Toronto

# Recap

- We have now covered all the primitives you need for building a transformer!
- All of these equations look abstract, but not to worry …
- Assignment 2 was designed for you to implement all these concepts into a working MT model of your own.

| Tasks | Section | Class | criterion | Max mark | Sub-Total | File |
|---|---|---|---|---|---|---|
| 1 | Building Blocks | LayerNorm | :forward | 2 | | |
| 2 | | MultiHeadAttention | :attention | 4 | | |
| 3 | | | :forward | 5 | | |
| 4 | | FeedForwardLayer | :forward | 1 | 12 | |
| 5 | Architecture | TransformerEncoderLayer | :pre_layer_norm_forward | 2 | | |
| 6 | | | :post_layer_norm_forward | 1 | | |
| 7 | | TransformerDecoderLayer | :__init__ | 4 | | |
| 8 | | | :pre_layer_norm_forward | 2 | | |
| 9 | | | :post_layer_norm_forward | 2 | | |
| 10 | | TransformerDecoder | :forward | 3 | | *a2_transformer_model.py* |
| 11 | | TransformerEncoderDecoder | :create_pad_mask | 1 | | |
| 12 | | | :create_causal_mask | 2 | | |
| 13 | | | :forward | 3 | 20 | |
| 15 | | | :greedy_decode | 5 | | |

# Architectural Variants

- The Transformers architecture underpins most SoTA LLMs of today.

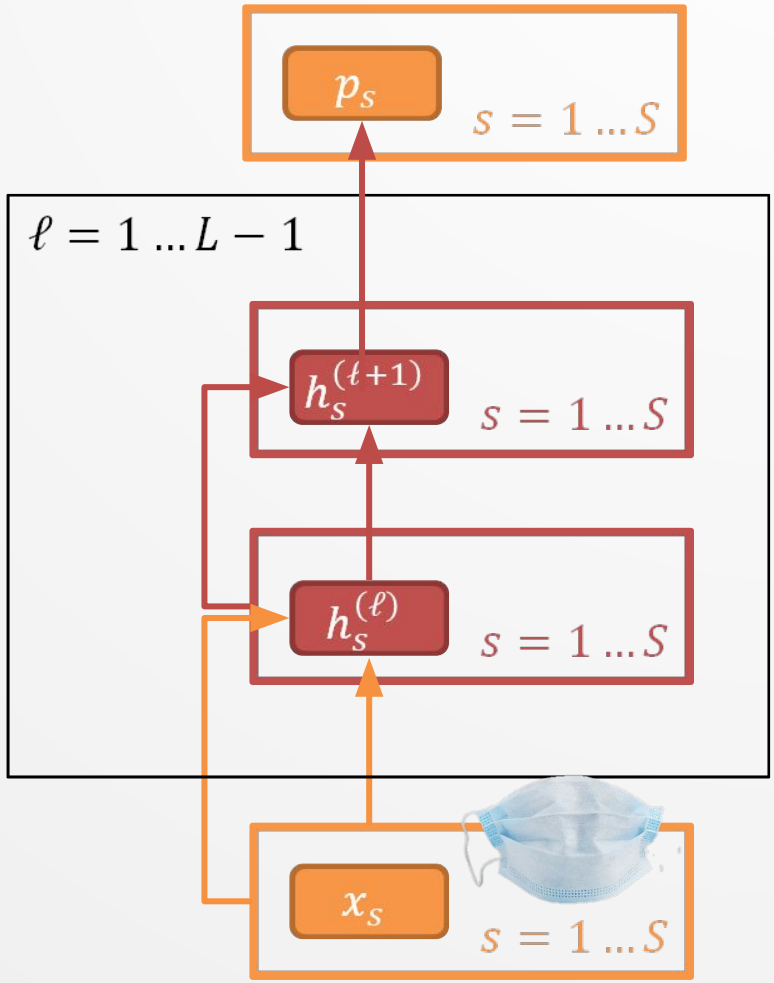
- There are 3 main variations of the encoder-decoder blocks we studied in L6

  - Encoders – e.g. models: **BERT** and BERT-variants
  - Decoders – e.g. the **GPT** series
  - Encoder-Decoder – e.g. vanilla transformer, **T5,** Flan-T5[1] (instruction tuned T5) etc.

[1] Flan-T5 - Chung, Hyung Won, et al. "Scaling instruction-finetuned language models.". link.

# BERT: Bidirectional Encoder Representations from Transformers

- First, **pre-trained** on (large) unlabeled data on two unsupervised tasks/objectives:
  - Masked LM (**MLM**), and
  - Next Sentence Prediction (**NSP**)
- Then, **fine-tuned** using labeled data from downstream tasks
- Training entails feeding the final hidden vectors to an output linear layer with softmax over the possibilities (e.g. the vocabulary as in a standard LM)




Devlin *et al.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2019). [arxiv]

Code and models: https://github.com/google-research/bert [Colab] Google AI

# BERT: Bidirectional Encoder Representations from Transformers

## Pre-training objectives

- Masked LM (**MLM**): predict randomly **masked** words:

**Input:** The man went to the $[MASK]_1$ . He bought a $[MASK]_2$ of milk .
**Labels:** $[MASK]_1$ = store; $[MASK]_2$ = gallon

  - 80% of the target words are masked with: [MASK]. 10% are replaced with another word, and 10% are kept as-is, to bias '*towards the observation*'.
  - *Variants*: masking granularity can be varied (word-piece, word, span) with respective quirks. E.g., masking named entities improves structured knowledge representation.

- Next sentence prediction (**NSP**): does sentence B follow A?

**Sentence A** = The man went to the store.
**Sentence B** = He bought a gallon of milk.
**Label** = IsNextSentence

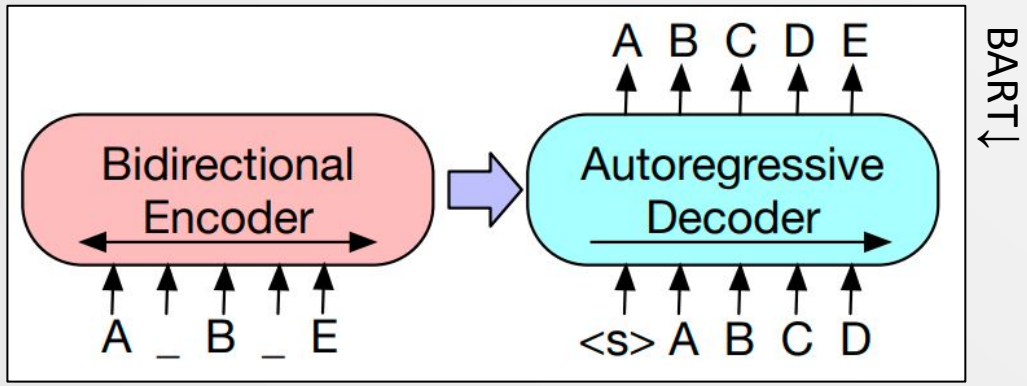
**Sentence A** = The man went to the store.
**Sentence B** = Penguins are flightless.
**Label** = NotNextSentence

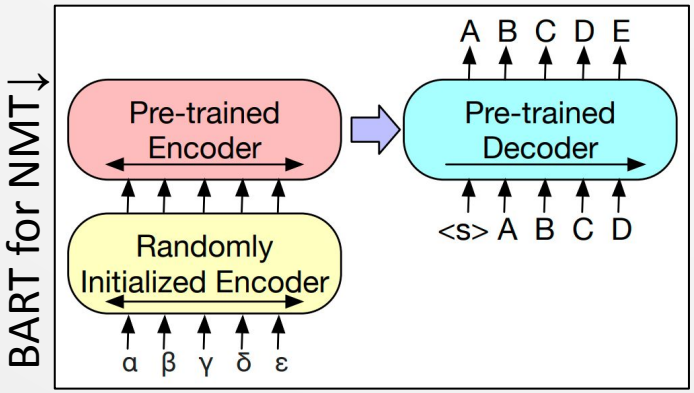
  - 50% of the time true, 50% of the time it's a random sentence.
  - Later research finds removing the NSP task does not hurt, or slightly improves performance. [1]

[1] Rogers, Anna et al. "A primer in BERTology: What we know about how BERT works." TACL(2020). link

# Aside: BERT → BART → NMT

- Explosion of variants to BERT
- Pretrained BERT language model used to re-score/fine-tune downstream NLP tasks
- BART (Lewis *et al*, 2020) adds the decoder back to BERT, keeping the BERT objective
- Add some source language layers on top to train for NMT




Lewis, Mike, et al. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension." (2019). link.

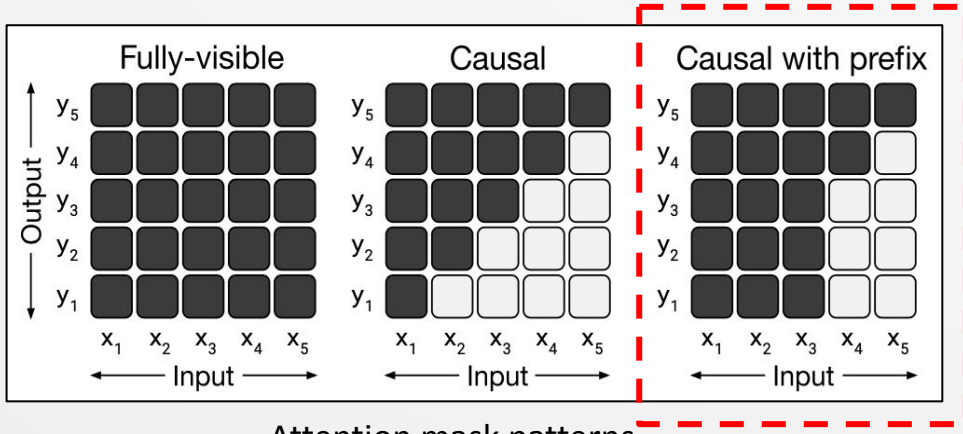
# T5: Text-to-Text Transfer Transformer

## A transformer refined with extensive scientific testing

- T5 is an unified framework that casts all NLP problems into a '*text-to-text*' format.
- Architecturally (almost) identical to the original Transformer (Vaswani et al., 2017).
- Draws from a systematic study comparing pre-training objectives, architectures, unlabeled data sets, transfer approaches, and other factors on dozens of language understanding tasks.
- Introduces and uses a new curated **dataset**: "*Colossal Clean Crawled Corpus*" (**C4**) for training.

**Distinguishing features**:

- Consistent, task-invariant MLE training objective.
- Self-attention "mask" with prefix.
- Unsupervised "denoising" training objectives: ***span corruption*** (conceptually same to MLM, mask 'spans' instead of words).



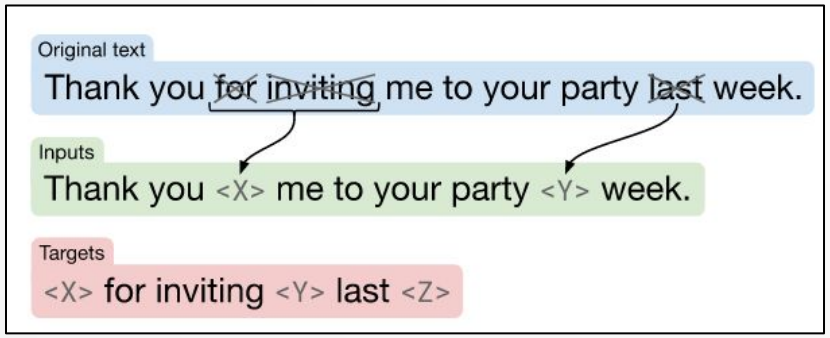

Attention mask patterns

1. Raffel et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." (2020).

# T5: Text-to-Text Transfer Transformer

**Example Task**: English to German (En-De) translation:

Input sentence: "*That is good.*"
Target: "*Das ist gut.*"

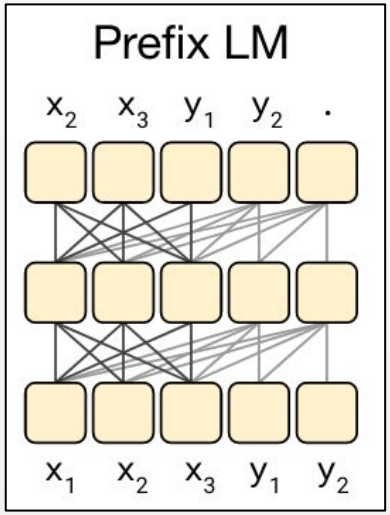



Input/Output format for training denoising objective

- **Training:** task specification is imbued by prepending task prefix to the input sequence. Model trained on next sequence prediction over the concatenated input sequence:

  "*translate En-De: That is good. Target: Das ist gut.*"

- For prediction, the model is fed **prefix**:
  - "*translate En-De: That is good. Target:*"
- For **classification** tasks, the model predicts a single word corresponding to the target label.
- E.g. MNLI task of entailment prediction:
  - "*mnli premise: I hate pigeons. hypothesis: I am hostile to pigeons. entailment.* "
- Model predicts label: {"entailment", "neutral", "contradiction"}.

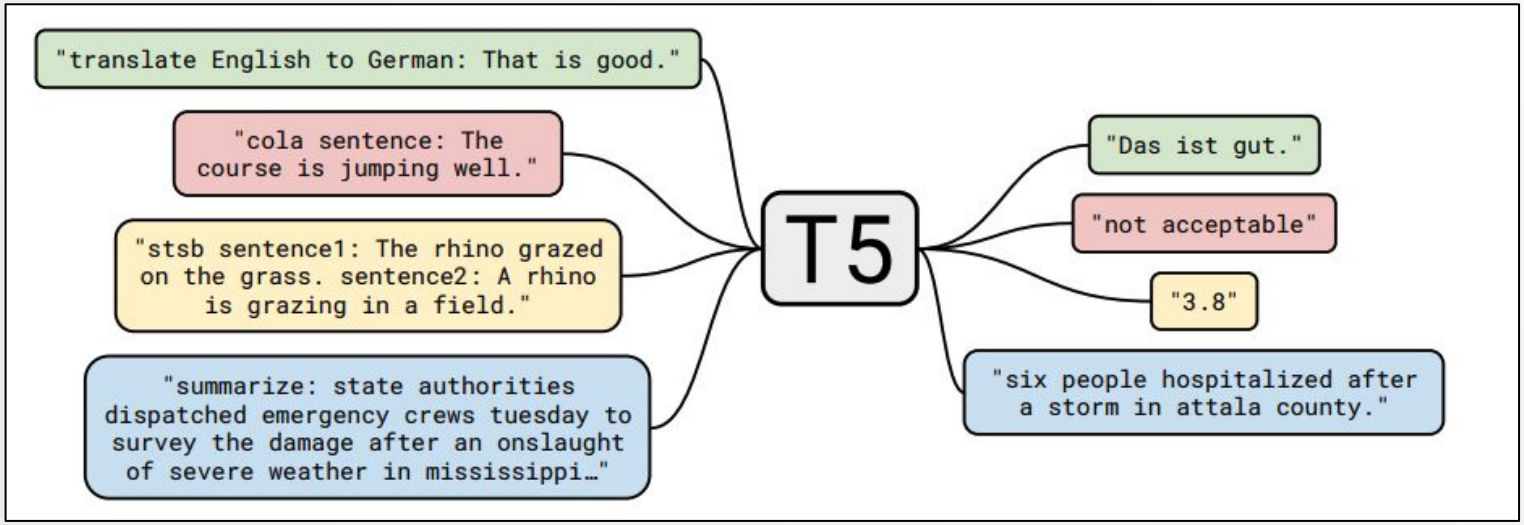


# T5: Text-to-Text Transfer Transformer

## Why T5 matters?

- Unifying diverse NLP problems as one (‘*text-to-text*’) format is a **really cool idea**.

- This allows us to use the **same** model, loss function, hyperparameters etc. across a diverse set of tasks




- Remarkable transfer learning capabilities: T5 can be finetuned to answer a wide range of (open-domain) questions, retrieving factoids from its parameters

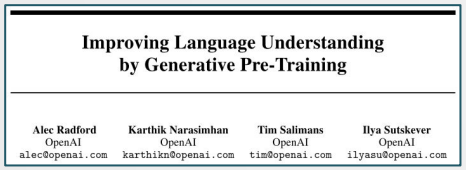
# GPT SERIES

(slides borrowed from Brown et al. (2020) Language Models are Few-Shot Learners, *Arxiv* 2005.14165)

# GPT: Generative Pretrained Transformers

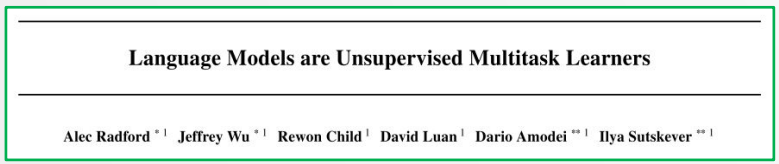
Open AI GPT-series of models – uses multi-layer decoder only blocks

- Open AI GPT papers: GPT (2018)[1], GPT-2 (2019)[2], GPT-3 (2020)[3]



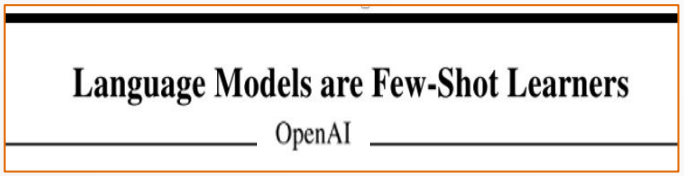
Improving Language Understanding by Generative Pre-Training

Alec Radford, OpenAI, alec@openai.com
Karthik Narasimhan, OpenAI, karthikn@openai.com
Tim Salimans, OpenAI, tim@openai.com
Ilya Sutskever, OpenAI, ilyasu@openai.com



Language Models are Unsupervised Multitask Learners

Alec Radford [*1] Jeffrey Wu [*1] Rewon Child [1] David Luan [1] Dario Amodei [**1] Ilya Sutskever [**1]



Language Models are Few-Shot Learners

OpenAI

- Architecturally (almost) identical – each scales (params, data) on predecessor
- Pretraining objective is classic '**language modeling**', to maximize the likelihood:

$$p(x) = \prod_{i=1}^{n} p(s_n | s_1, \ldots, s_{n-1})$$

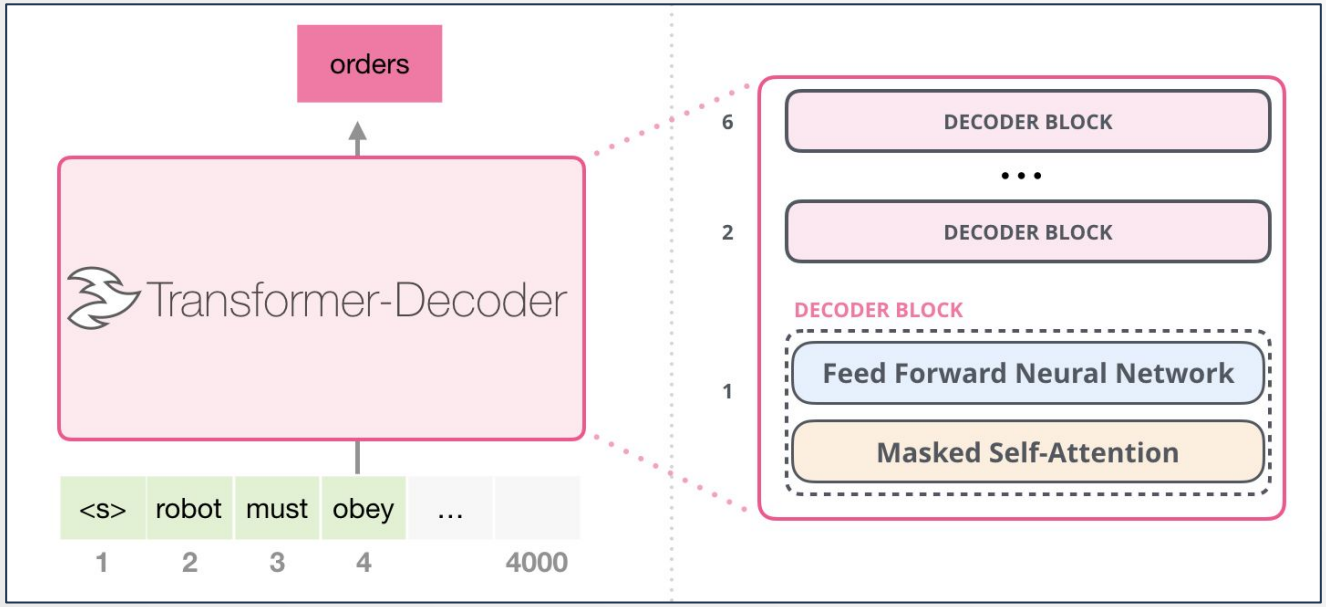
- Specifically, given an unsupervised corpus of tokens $\mu = \{\mu_1, \ldots, \mu_n\}$, where k is context window, P is modelled using a neural network with parameters θ.

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$


1. Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018)
2. Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI Blog 1.8 (2019)
3. Brown, Tom B., et al. "Language models are few-shot learners." arXiv preprint arXiv:2005.14165 (2020)

# GPT: Generative Pretrained Transformers

- Distinguishing features:
  - Uses multi-layer transformer **decoder only blocks**
  - Auto-regressive generative model, does not see the future (no bi-directional awareness)
  - Like traditional LMs, outputs one token at a time

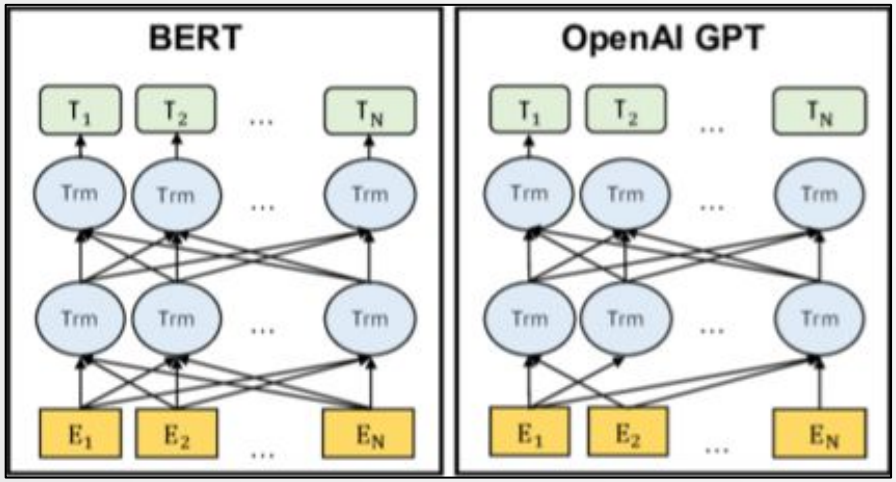




1. Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018)
2. Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI Blog 1.8 (2019)
3. Brown, Tom B., et al. "Language models are few-shot learners." arXiv preprint arXiv:2005.14165 (2020)

# Key architectural differences

- GPT vs. BERT-variants:
  - GPT uses ‘transformer’ blocks as *decoders*, and BERT as *encoders*.
  - Underlying (block level) ideology is same
  - GPT (later Transformer XL, XLNet) is an **autoregressive** model, BERT is not
    - At the cost of auto-regression, BERT has bi-directional context awareness.
  - GPT, like traditional LMs, outputs (predicts) one token at a time.




- Also compare with T5, BART that use encoder-decoder

[1] Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).

# GPT-3 LLMs take off …

- Increasingly convincing results permeating into the public sphere and *Zeitgeist*
- In-context learning:
  - Very large models (GPT-3 175B parameters; T5 11B parameters; GPT-4 ~1.8T parameters; GPT-5 ~5T parameters) exhibiting the **'in-context learning' (ICL) phenomenon**
  - Exhibits learning without any gradient updates (traditional learning), but merely from given examples!

1. Brown, Tom B., et al. "Language models are few-shot learners." arXiv preprint arXiv:2005.14165 (2020)

# GPT-3 LLMs take off …

- But you really won't expect these models (decoder-only) to do THAT well, unless they are of a very large amount of parameters, and:
- In-context learning:

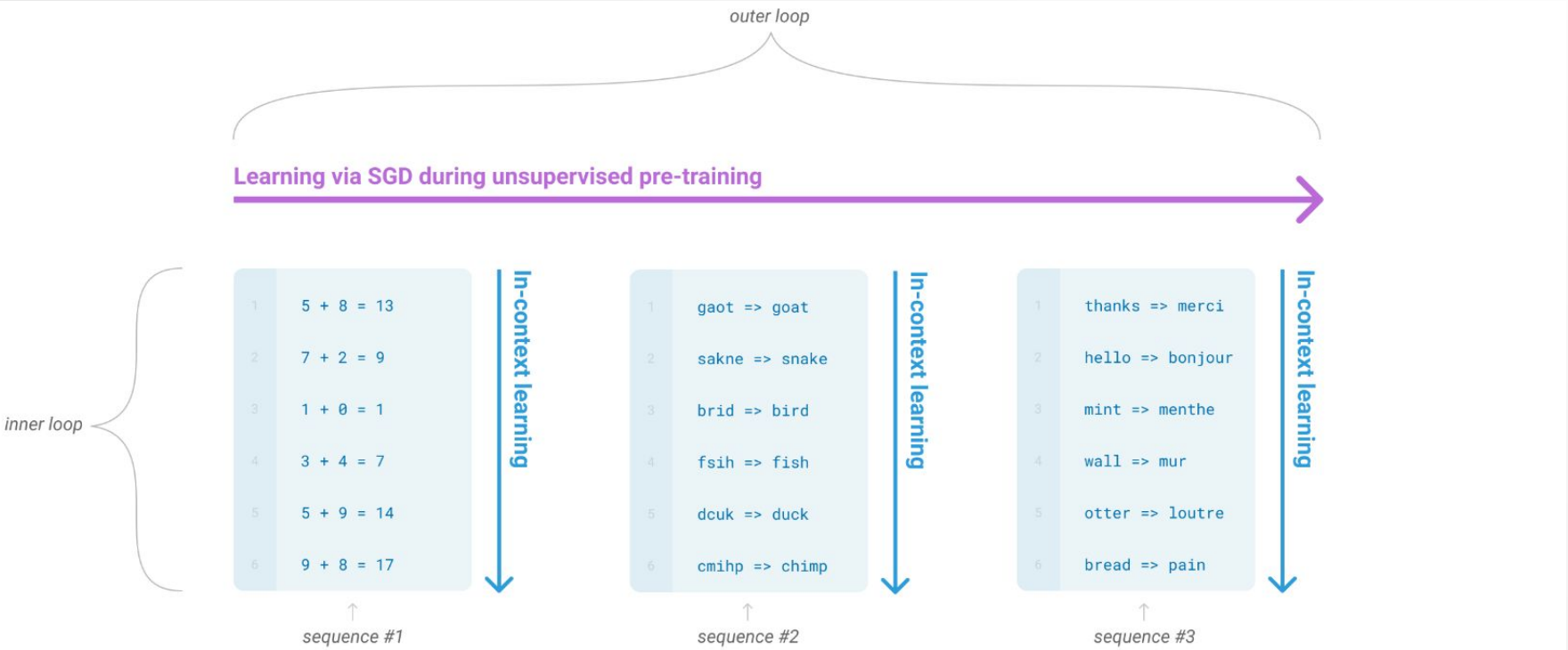



**Figure 1.1: Language model meta-learning.** During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. We use the term "in-context learning" to describe the inner loop of this process, which occurs within

1. Brown, Tom B., et al. "Language models are few-shot learners." arXiv preprint arXiv:2005.14165 (2020)

# GPT-3: In context learning + … prompting

- **Prompting**: Learning in which the prefix of the string contains the specification of the task.
- **ICL**: At least one-shot; Prompting instead of Supervised Fine-Tuning


**Figure 2.1: Zero-shot, one-shot and few-shot, contrasted with traditional fine-tuning.** The panels above show

1. Brown, Tom B., et al. "Language models are few-shot learners." arXiv preprint arXiv:2005.14165 (2020)

# GPT-3: In context learning + … prompting

- '**Prompting**' had been around as a convenience, but now a necessity …

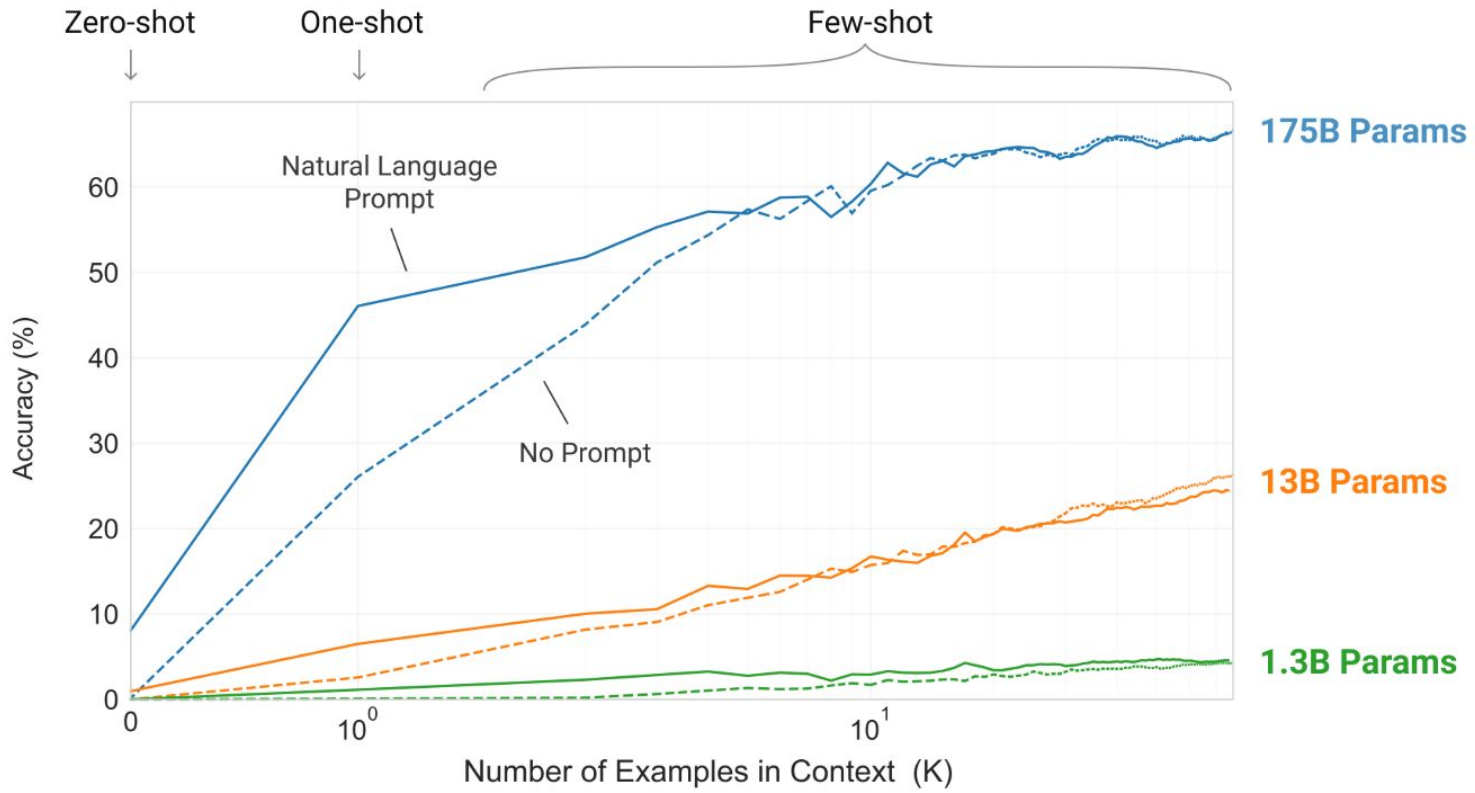



**Figure 1.2: Larger models make increasingly efficient use of in-context information.** We show in-context learning

1. Brown, Tom B., et al. "Language models are few-shot learners." arXiv preprint (2020)

# Prompting: Chain of Thought

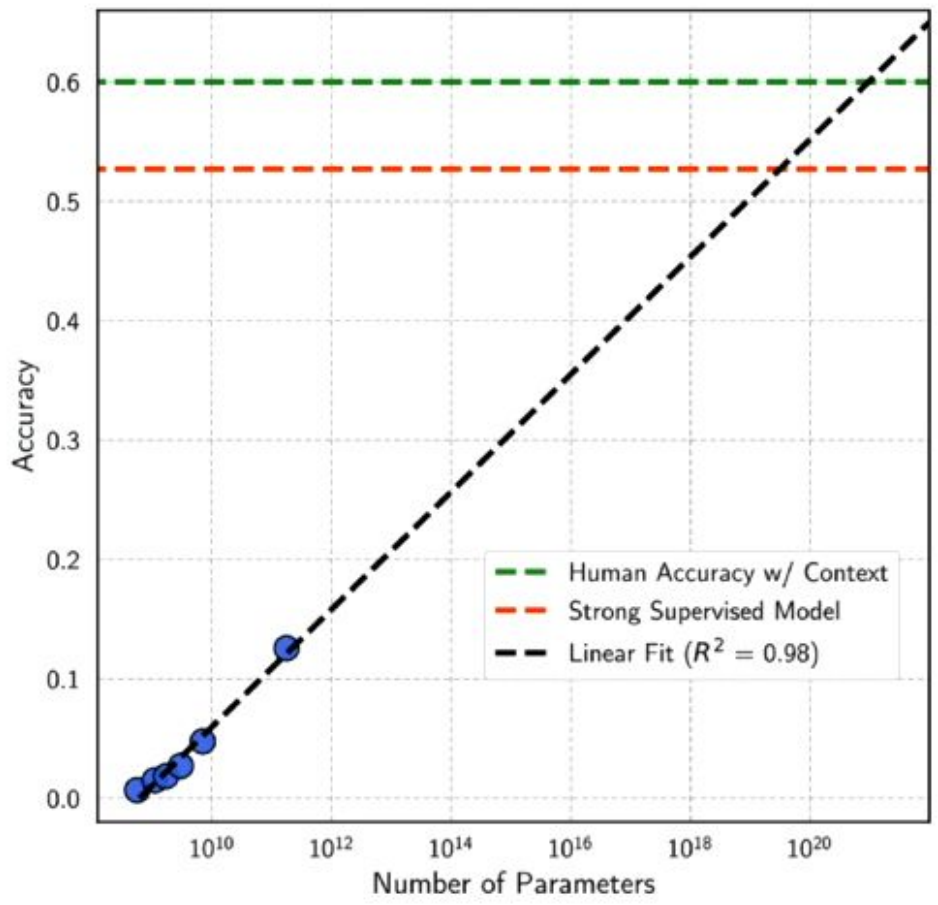
- Encourage the model to establish a logical chain of inference

| CoT Prompt (0-shot with LLAMA2-70B-CHAT) | Accuracy |
|---|---|
| Let's reflect on each answer option like an operations research expert | 0.487 |
| Let's think step by step | 0.518 |
| Let's think step by step like an operations research expert | 0.509 |
| Let's use step by step inductive reasoning, given the mathematical nature of the question | 0.510 |
| Let's work by elimination | 0.475 |
| **Prompt ensembling (majority vote)** | 0.537 |

- See also "least-to-most" prompting, in which the inference proceeds relative to a taxonomy or decomposition into subproblems.

# ICL/Prompting vs. Fine-tuning?

- Fine-tuning is probably here to stay, however…




*From "Large Language Models Struggle to Learn Long-Tail Knowledge" by Kandpal et al.*

# GPT-3: Results

- TL;DR: very impressive results across task domains
  - Performance (e.g. accuracy) **increases with size**
- Datasets grouped into 9 categories of downstream tasks
  - Examples: language modeling, QA, translation, Winograd, common-sense reasoning, reading comprehension, NLI etc.
  - Read the paper for details

| Setting | PTB |
|---|---|
| SOTA (Zero-Shot) | 35.8[a] |
| GPT-3 Zero-Shot | **20.5** |

**Table 3.1: Zero-shot results on PTB language modeling dataset.** Many other common language modeling datasets are omitted because they are derived from Wikipedia or other sources which are included in GPT-3's training data. [a][RWC+19]

Turing-NLG?

| Setting | LAMBADA (acc) | LAMBADA (ppl) | StoryCloze (acc) | HellaSwag (acc) |
|---|---|---|---|---|
| SOTA | 68.0[a] | 8.63[b] | **91.8**[c] | **85.6**[d] |
| GPT-3 Zero-Shot | **76.2** | **3.00** | 83.2 | 78.9 |
| GPT-3 One-Shot | **72.5** | **3.35** | 84.7 | 78.1 |
| GPT-3 Few-Shot | **86.4** | **1.92** | 87.7 | 79.3 |

**Table 3.2: Performance on cloze and completion tasks.** GPT-3 significantly improves SOTA on LAMBADA while achieving respectable performance on two difficult completion prediction datasets. [a][Tur20] [b][RWC+19] [c][LDL19] [d][LCH+20]

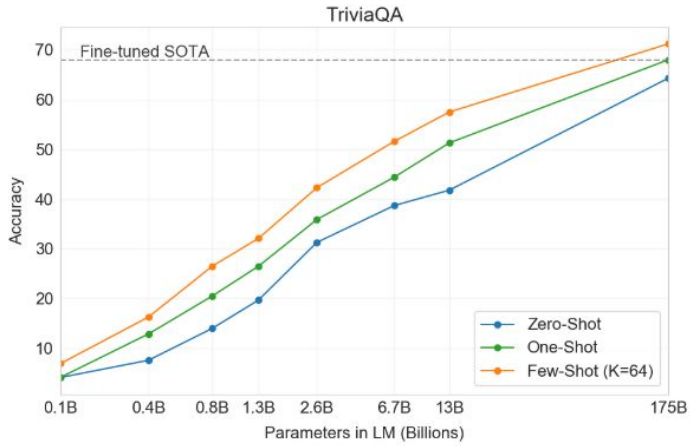



**Figure 3.3:** On TriviaQA GPT3's performance grows smoothly with model size, suggesting that language models continue to absorb knowledge as their capacity increases. One-shot and few-shot performance make significant gains over zero-shot behavior, matching and exceeding the performance of the SOTA fine-tuned open-domain model, RAG [LPP+20]

# DIFFERENT DIRECTIONS

# Token free models

- Unlike the ubiquitous pre-trained LMs that operate on sequences of tokens corresponding to word or sub-word units, *token free models*:

  - ⊕ Operate on raw text (bytes or characters) **directly.**
  - ⊕ Remove necessity for (error-prone, complex) text preprocessing pipelines.

  - ⊖ Con: raw sequences significantly **longer than token sequences**, increases computational complexity. (Reminder: *'attention' costs are quadratic in the length of input sequence*)

1. Clark et al. "**CANINE**: Pre-training an efficient tokenization-free encoder for language representation." (2021). link
2. Xue et al. "**ByT5**: Towards a token-free future with pre-trained byte-to-byte models." (2022). link

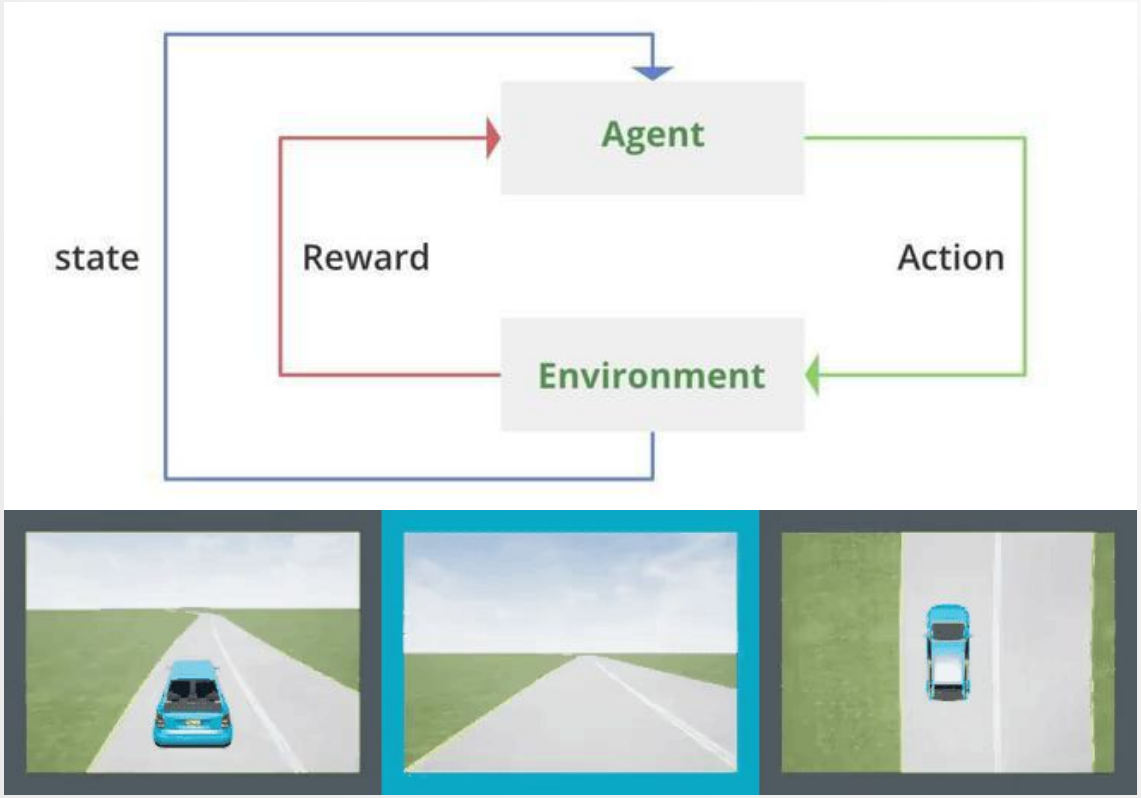
# Token free models

- Pitfalls of explicit (word, sub-word) tokenization:
  - Need for large language dependent (fixed) vocabulary mapping matrices.
  - Applies hand-engineered, costly, language-specific string tokenization/segmentation algorithms (e.g. BPE, word-piece, sentence-piece) requiring linguistic expertise.
  - Heuristic string-splitting, however nuanced, cannot capture full breadth of linguistic phenomena, (e.g. morphologically distant agglutinative, non-concatenative languages). Other examples include languages without white-space (Thai, Chinese), or that uses punctuation as letters (Hawaiian, Twi). *Fine-tuning* tokenization needs to match *pretraining* tokenization methods.
  - Brittle to noise, corruption of input (typos, adversarial manipulations). Corrupted tokens lose vocabulary coverage.

1. Clark et al. "**CANINE**: Pre-training an efficient tokenization-free encoder for language representation." (2021). link
2. Xue et al. "**ByT5**: Towards a token-free future with pre-trained byte-to-byte models." (2022). link

# Aside: Reinforcement Learning

- **Problem**: learn to make sequential decisions by interacting with an environment to maximize cumulative reward
- **Agent-Environment Loop**: observe state, take action, receive reward, transition to next state.



# RLHF: Reinforcement Learning from Human Feedback

## Value-based RL vs. Policy-based RL:

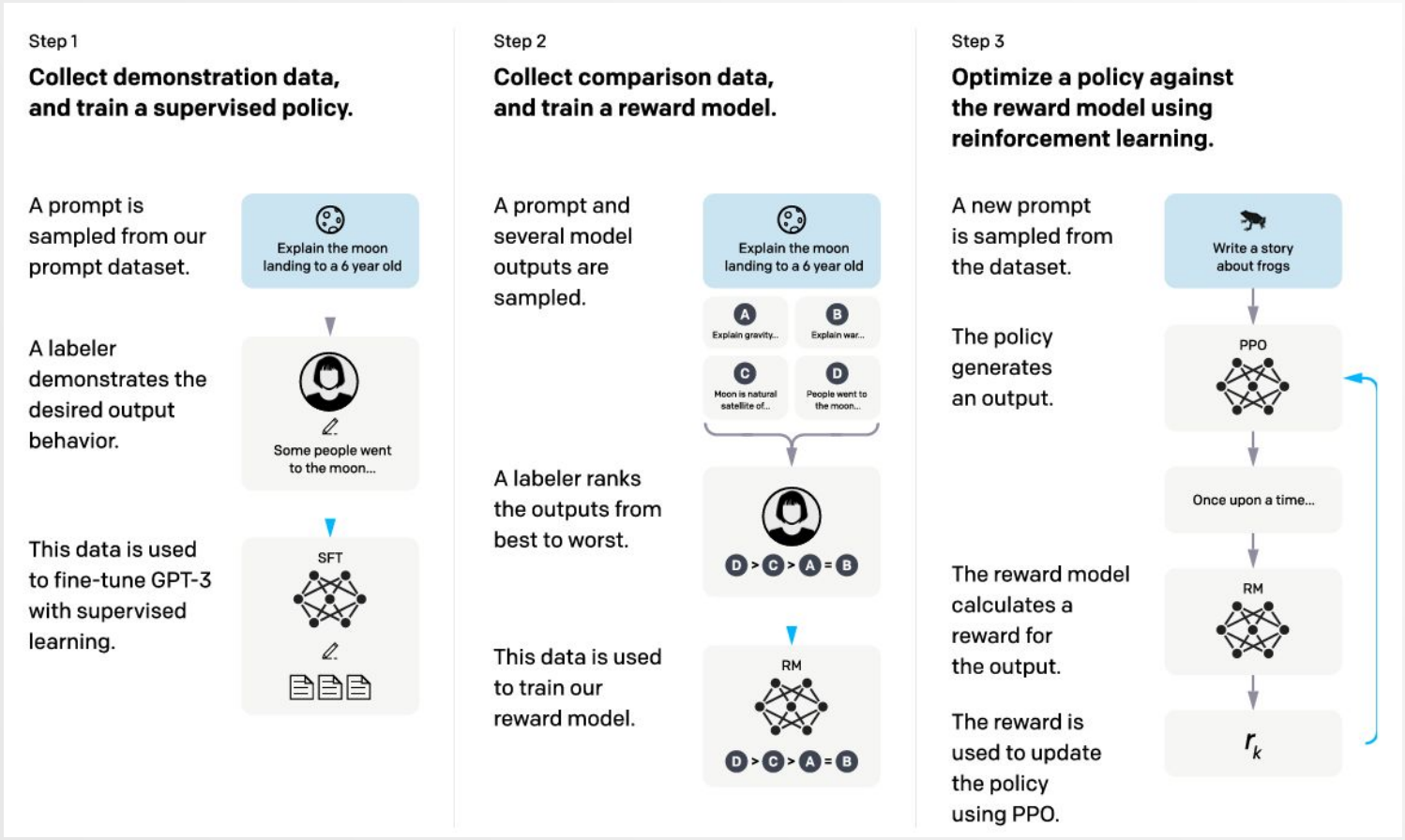
- Agent needs to determine what action to take for a given state, such rule(s) are called **Policy**.
- **Value-based RL**: learn a value function for actions, policy is usually implicit (e.g. Q-Learning, DQN, etc);
- **Policy-based RL**: learn the policy directly by optimizing the expected return (e.g. PPO, TRPO, etc).

## RLHF vs. RLVR (RL with Verifiable Rewards):

- A task might be **difficult to specify** yet **easy to judge**.
- Solution: Make human generate labels instead of data. (RLHF)
- Alternative: Establish rewards via deterministic verifier. (RLVR)

# InstructGPT

- Step 1: Supervised Fine-Tuning (SFT)
- Step 2: Reward Model (RM) Training
- Step 3: Proximal Policy Optimization (PPO)



# PPO vs. GRPO vs. DAPO

- A (very high-level) graph comparison: