# Pre-Training Variants

COMP 4420 / 5420 Natural Language Processing
Instructor: Anna Rumshisky

March 30, 2023

# What you should understand after this lecture

- Difference between encoder, decoder, and seq-to-seq architectures
- Different denoising objectives: MLM, CLM, shuffling, span deletion, etc.
- In-context learning vs. few-shot fine-tuning
- How to formulate any NLP task as a text-to-text problem

# Encoders, Decoders, and Seq-to-Seq models

# Three types of model architecture

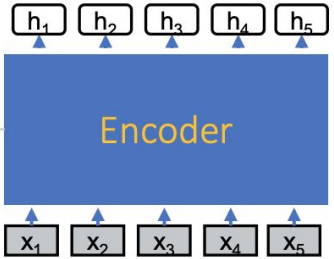
- Encoder-decoder models (aka sequence-to-sequence)
  - Currently, we do this with **transformers** (2018 - 2022)
  - Before that, we did it with **recurrent neural models** (2013 - 2018)
- Encoder only (BERT)
- Decoder only (GPT)
  - No cross-attention

# NLP Tasks

## NLU

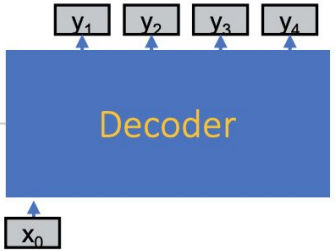
{ NL => X (≠ NL) }

Text (Language E) => { Class, Sequence of class, Score, ...}




## NLG

{ X (≠ audio/image/video) => NL }




### NLG (LM)

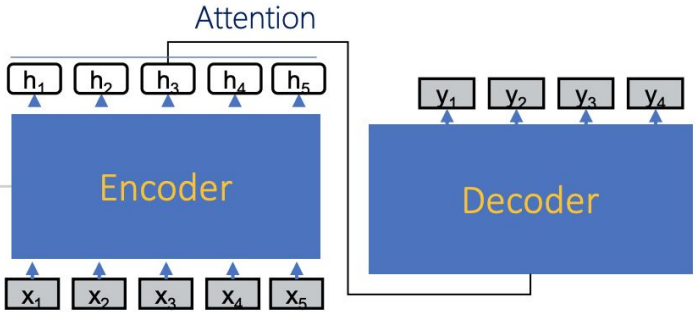
{ } => Text (Language E)

### NLG (S2S)

Text (Language E) => Text (Language E)

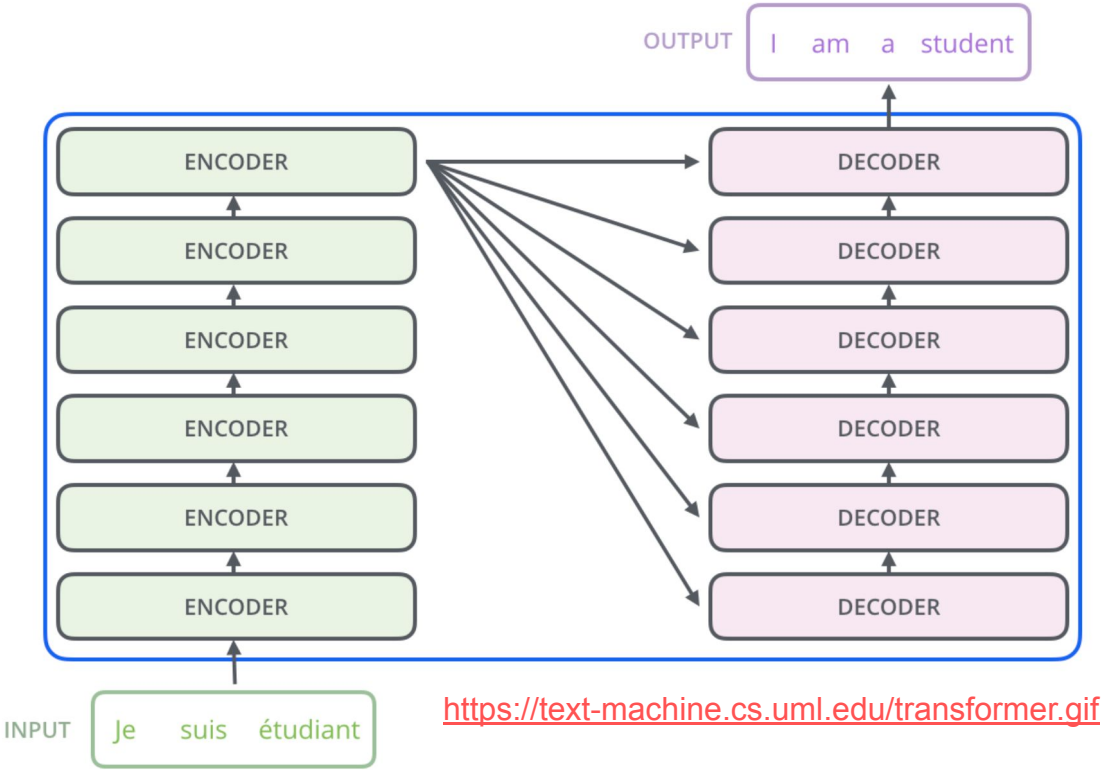
### NLG (XL)/MT

Text (Language E) => Text (Language F)




Slide credit: Furu Wei (UniLM)

# Transformer: Encoder-Decoder




https://text-machine.cs.uml.edu/transformer.gif

Image credit:
https://jalammar.github.io/illustrated-transformer/

# Transformer: Encoder-Decoder

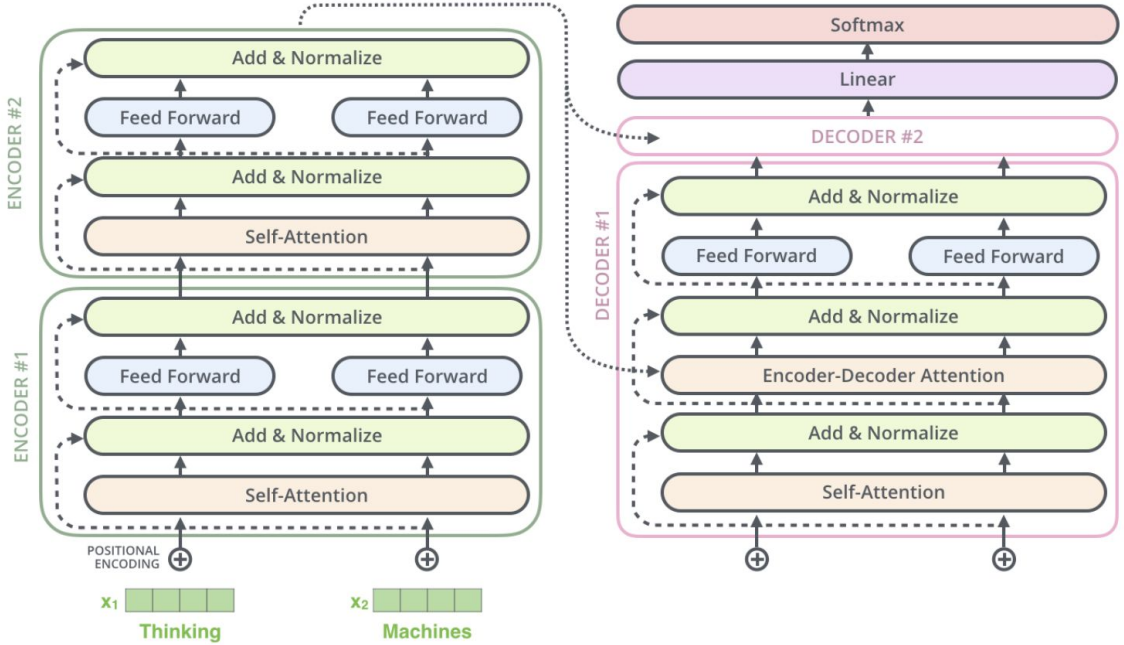
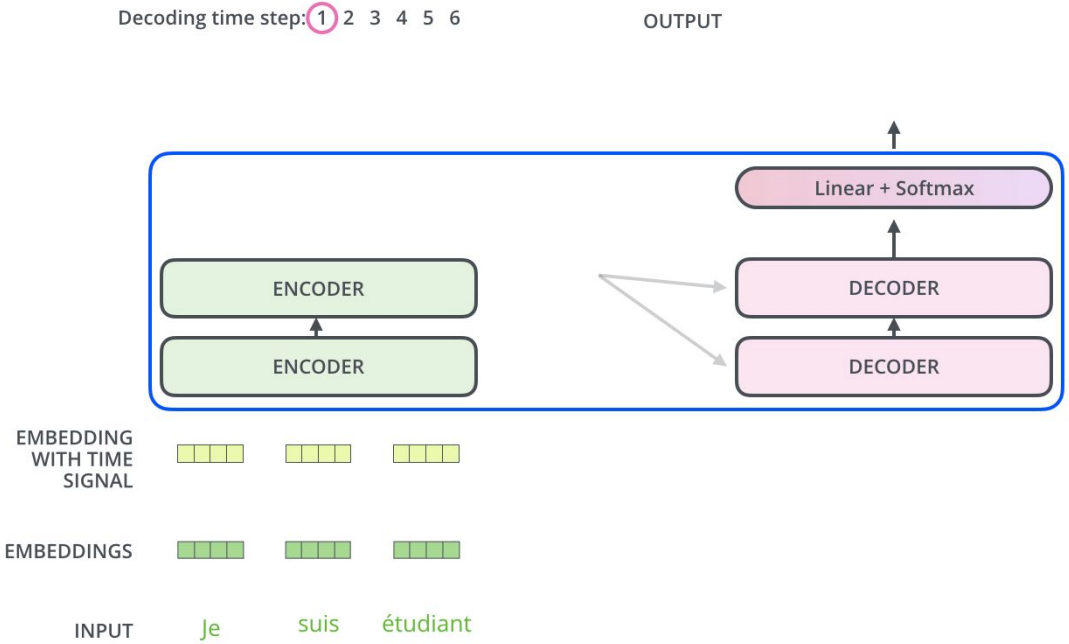



Image credit:
https://jalammar.github.io/illustrated-transformer/

# Transformer: Encoder-Decoder

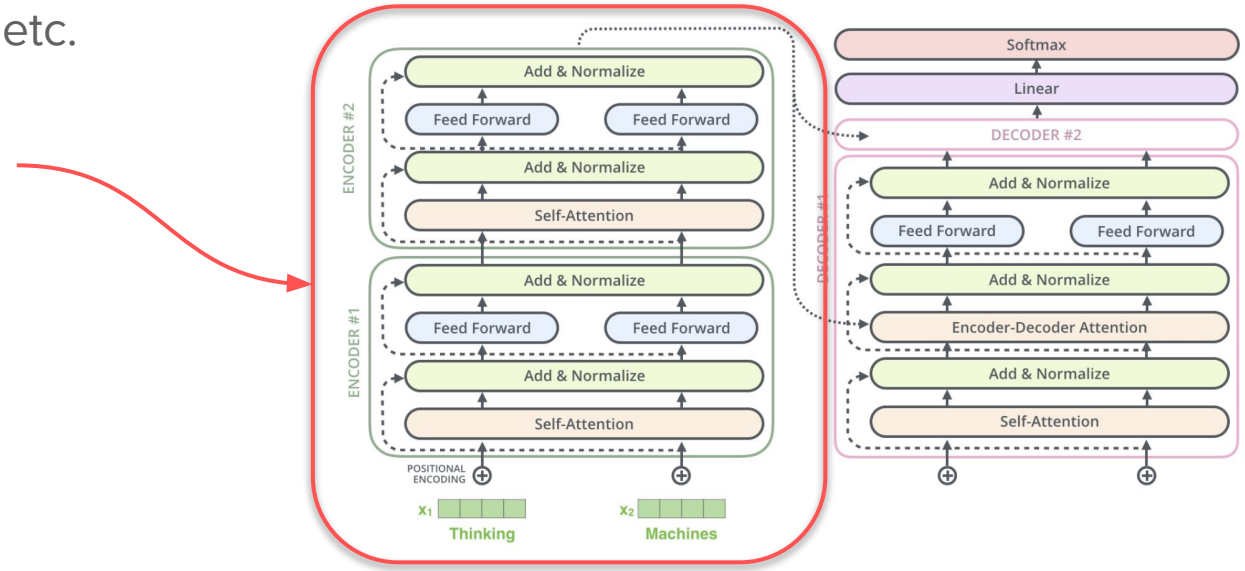



https://jalammar.github.io/images/t/transformer_decoding_1.gif

Image credit:
https://jalammar.github.io/illustrated-transformer/

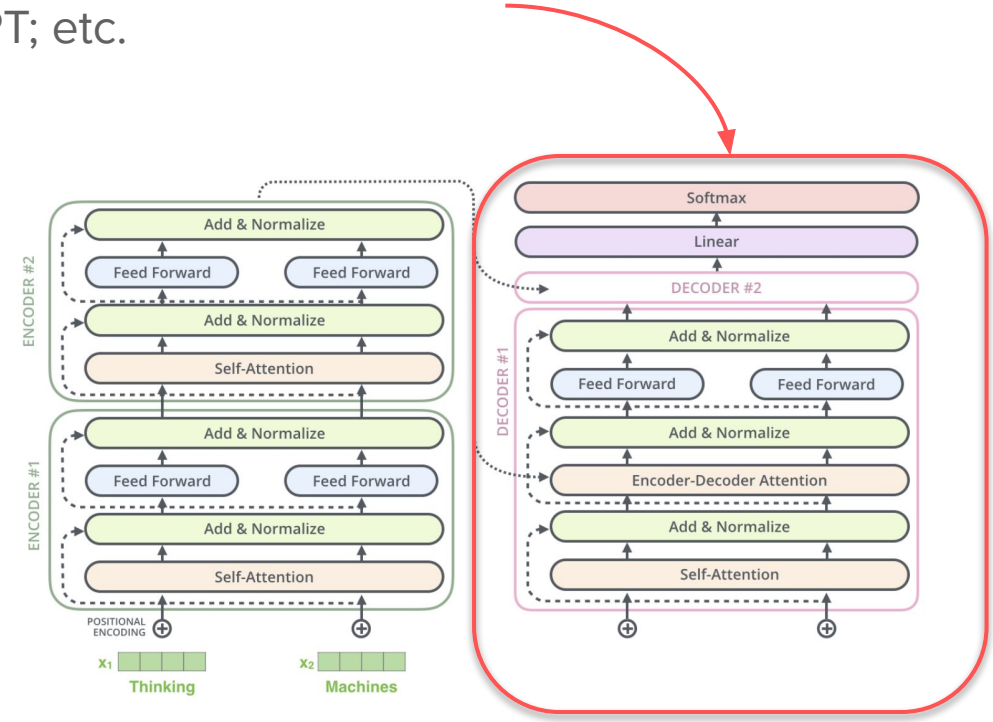
# Encoder models

- Good for natural language understanding tasks
  - Classification – this text is about topic A
  - Semantic role labeling – [Seller John] sold [Sold his Toyota] to [Buyer Jane] for [Price $10,000]
  - POS-tagging, NLI, paraphrase detection (pairs of objects encoded), etc.
- BERT, RoBERTa, etc.



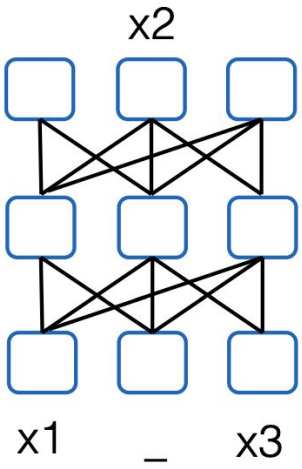
# Decoder models

- Good for sequence generation
- GPT, GPT-2, GPT-3; BLOOM; OPT; etc.



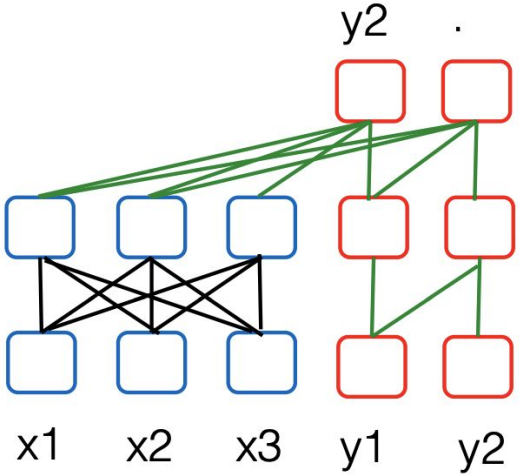
# Different Setups

**Encoder**



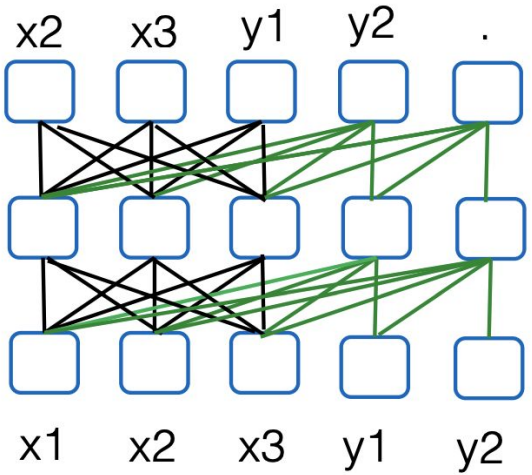

**BERT**

**SEQ2SEQ**




**MASS/T5/BART**

**Conditional / Prefixed LM**




**UniLM/T5**

Slide credit: Graham Neubig

# BERT vs. GPT

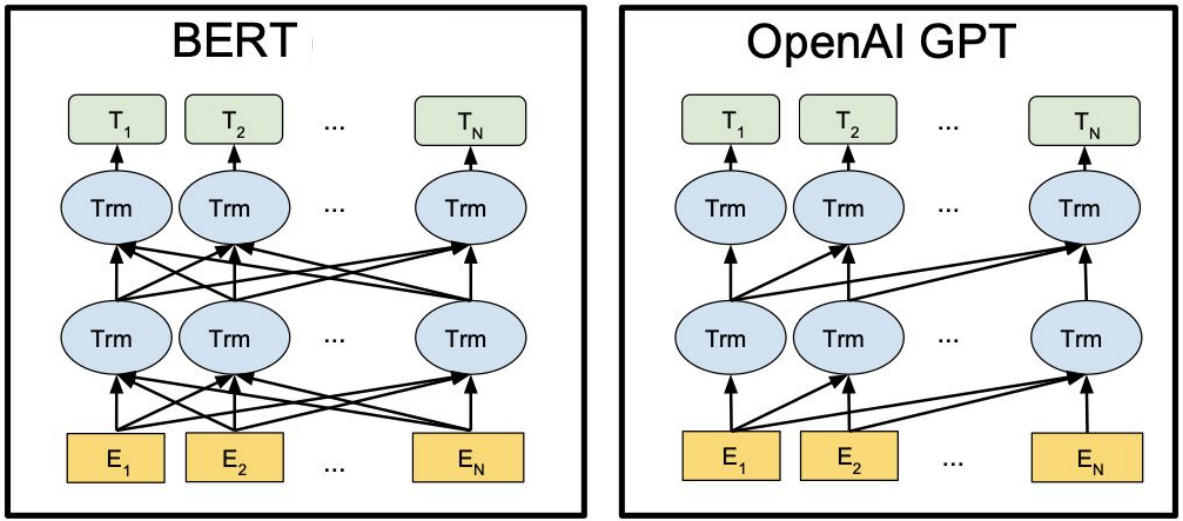



Image credit: Devlin et al 2018

# BERT: Input Representation

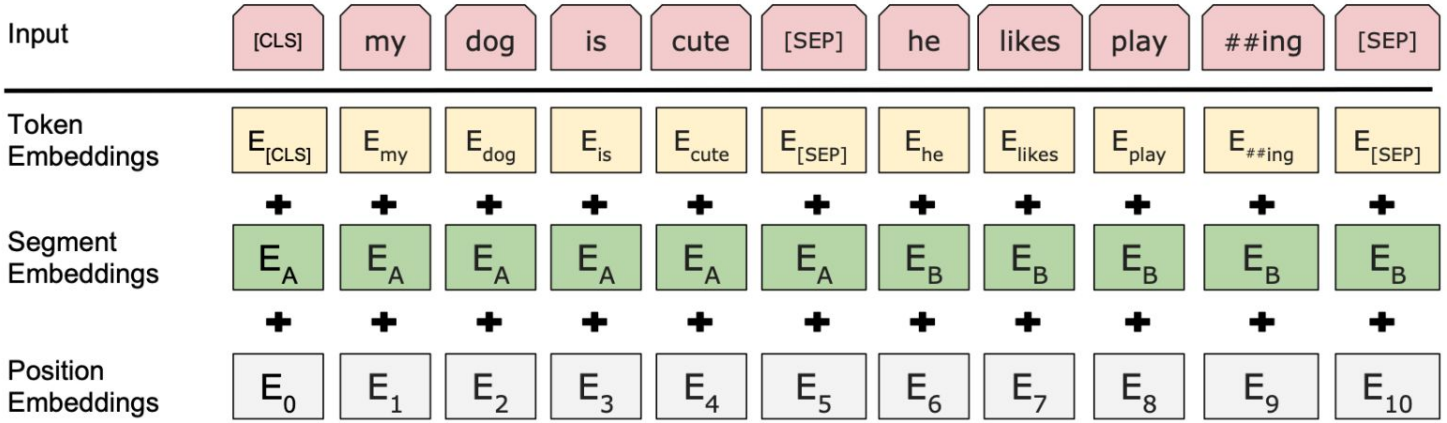



Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

# BERT: Input Formatting

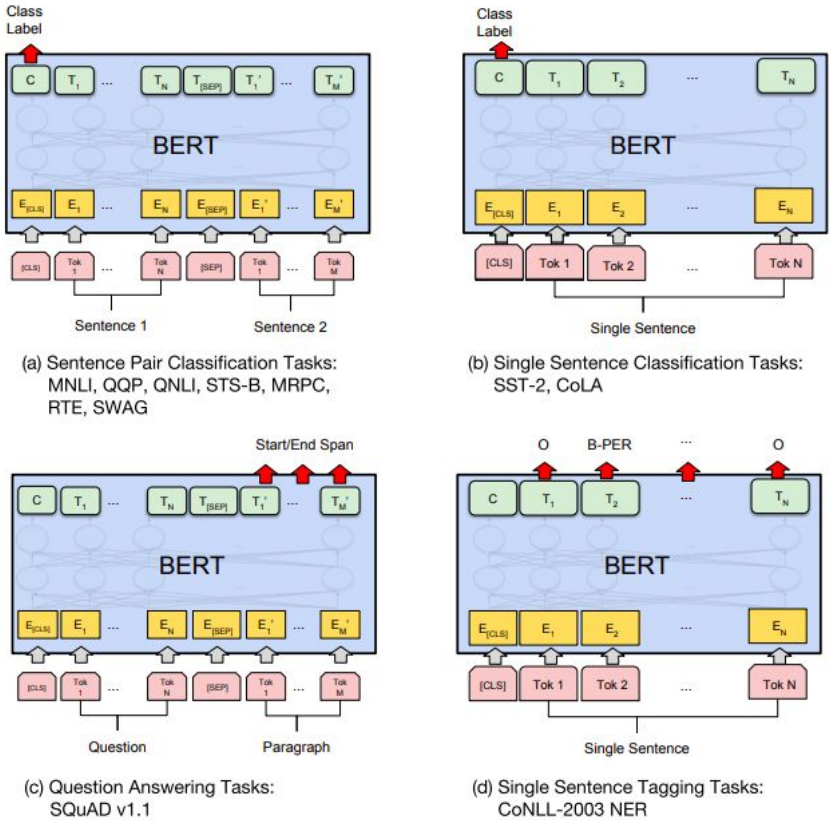



Image credit: Devlin et al 2018

# GPT Pre-Training

Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$, we use a standard language modeling objective to maximize the following likelihood:

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

where $k$ is the size of the context window, and the conditional probability $P$ is modeled using a neural network with parameters $\Theta$. These parameters are trained using stochastic gradient descent [51].

In our experiments, we use a multi-layer *Transformer decoder* [34] for the language model, which is a variant of the transformer [62]. This model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens:

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_l &= \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n] \\ P(u) &= \texttt{softmax}(h_n W_e^T) \end{aligned} \quad (2)$$

where $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, $n$ is the number of layers, $W_e$ is the token embedding matrix, and $W_p$ is the position embedding matrix.

Radford et al 2018

# GPT Supervised Fine-Tuning

After training the model with the objective in Eq. 1, we adapt the parameters to the supervised target task. We assume a labeled dataset $\mathcal{C}$, where each instance consists of a sequence of input tokens, $x^1, \ldots, x^m$, along with a label $y$. The inputs are passed through our pre-trained model to obtain the final transformer block's activation $h_l^m$, which is then fed into an added linear output layer with parameters $W_y$ to predict $y$:

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y). \tag{3}$$

This gives us the following objective to maximize:

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m). \tag{4}$$

We additionally found that including language modeling as an auxiliary objective to the fine-tuning helped learning by (a) improving generalization of the supervised model, and (b) accelerating convergence. This is in line with prior work [50, 43], who also observed improved performance with such an auxiliary objective. Specifically, we optimize the following objective (with weight $\lambda$):

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C}) \tag{5}$$

Radford et al 2018

# GPT: General Purpose Transformer

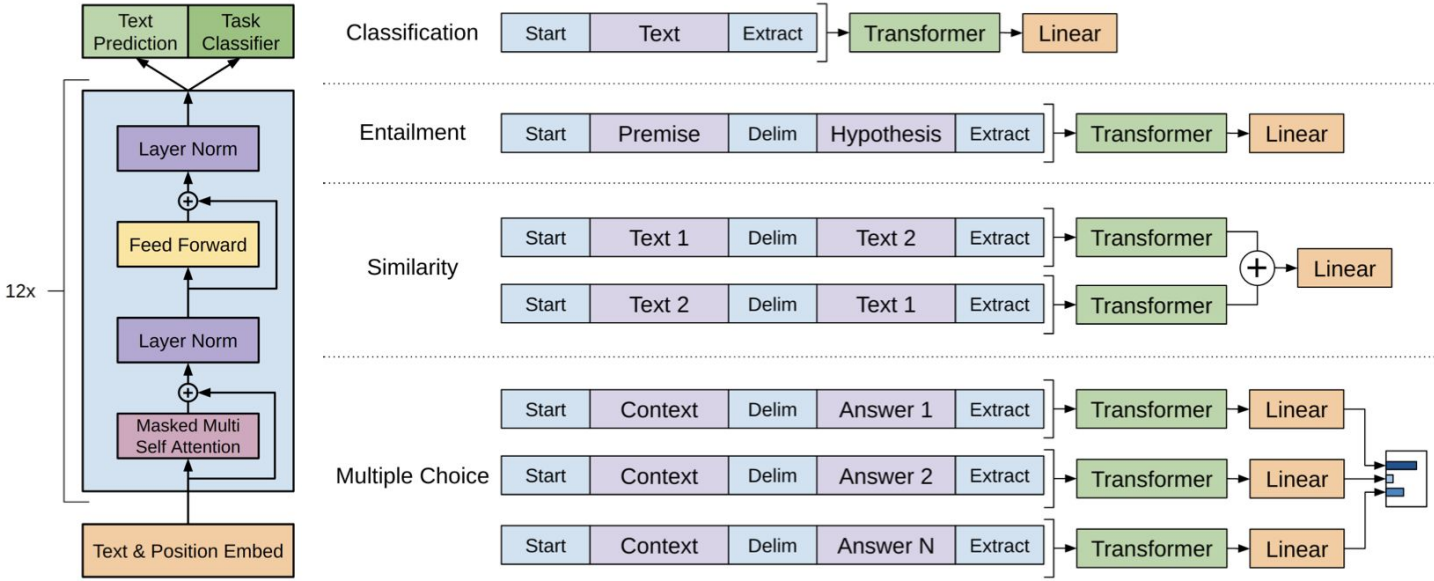



Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

# Large models can do *in-context learning*

- How do you get a pre-trained model to divulge its knowledge?

- Option 1: Pre-train it on your target task, tell the model what kind of task it has to solve; then ask it to solve it (at inference)
  - Requires fine-tuning, on a relatively small number of examples
  - Can be thought of as **few-shot learning**

- Option 2: Do not pre-train, just ask the model to continue some input
  - Formulate your task as a pair “prompt” - “desired output”
  - Prompt the model and ask it to continue generation
  - Prompt design becomes crucial!

- GPT-3 was the first model to successfully demonstrate **in-context learning** (learning without fine-tuning)

# GPT-3 Demo

# GPT-3: General Purpose Transformer

https://beta.openai.com/playground

**PROMPT**

Label argument structure:
Example: John bought a Toyota
Annotation: [Buyer John] bought [Purchase a Toyota]
Example: John sold his Toyota to Mary
Annotation:

# In-context learning with GPT-3

Label argument structure:
Example: John bought a Toyota
Annotation: [Buyer John] bought [Purchase a Toyota]
Example: John sold his Toyota to Mary
Annotation: [Seller John] sold [Sale his Toyota] to [Buyer Mary]

The basic structure of a ditransitive verb is [Agent X] [Verb] [Recipient Y] [Theme Z]. In your first example, "John" is the agent, "bought" is the verb, "a Toyota" is the theme, and there is no recipient. In your second example, "John" is the agent, "sold" is the verb, "his Toyota" is the theme, and "Mary" is the recipient.

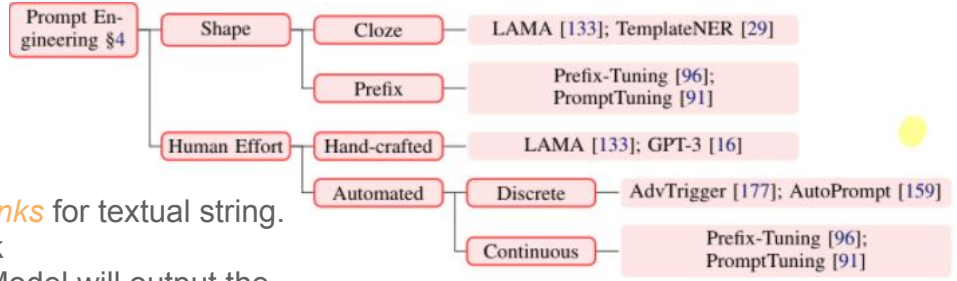
# 3. Design Considerations for Prompting

## Prompt Engineering

1. Shape *(shape of a prompt represents its type)*
   a. *Cloze prompt* - prompts with *fill in the blanks* for textual string. Model will predict the answer for the blank
   b. *Prefix prompt* - prompts with *text prefix*. Model will output the continuation of this text prefix.

2. Human effort
   a. Hand-crafted prompt
   b. Automated prompt *(prompts are generated by the model)*
      i. *Discrete /Hard prompt* - prompts with *actual string texts*
      ii. *Continuous/ Soft prompt* - prompts is not an text but is *described directly in the embedding space* of the underlying LM.LM is frozen, extra parameters for the prompt are trained on downstream task

3. Another categorization
   a. *Static prompt* - use *same* prompt template for each input
   b. *Dynamic prompt*- use/generate *custom* prompt template for each input

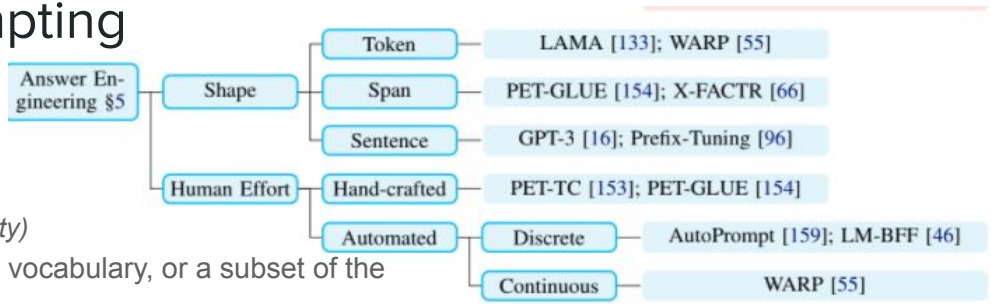



Discrete/Hard prompt -
- Prompt Mining - scrapes a large text corpus
- Prompt paraphrasing
- Gradient based search - apply gradient based search over actual tokens to find the shortest sequence which trigger the pretrained lM to generate desired target prediction.
- Prompt generation
- Prompt scoring - score some prompt candidates using LM and use the prompt with highest score

Continuous/Soft prompt -
- Prompt tuning -
- Prefix tuning -. Soft prompts as a prefix to the input
- Tuning Initialized with Discrete Prompts
- Hard-Soft Prompt Hybrid Tuning

Experimented against
- Length of the prompt
- Initialization of the prompts

# 3. Design Considerations for Prompting




## Answer Engineering

1. Shape *(a shape of an answer characterizes its granularity)*
   a. *Token* -One of the tokens in the pre-trained LM's vocabulary, or a subset of the vocabulary.
   b. *Span* - A short multi-token span. These are usually used together with cloze prompts.
   c. *Sentence* - A sentence or document. These are commonly used with prefix prompts.

2. Human effort
   a. Hand-crafted answer
      i. *Unconstrained space - no restriction the answer space*
      ii. *Constrained space - restriction on answer space*

   b. Automated prompt *(prompts are generated by the model)*
      i. *Discrete answer search*- answers with *actual string texts*
      ii. *Continuous answer search*- answers is not an text but is *described directly in the embedding space* of the underlying LM.( limited research in this one)

Discrete answer-
- Answer paraphrasing
- Prune then search - first search some tokens based on some condition. Use these tokens to form the answer.
- Label decomposition- For example, the relation ***per:city_of_death*** *d*ecomposed label words would be {person, city, death}. The probability of the answer span will be calculated as the sum of each token's probability.

# Pre-training objectives

# Causal (or autoregressive) language modeling objectives

- Predict the next word, given the previous words
- A causal mask prevents the *i*th output element from depending on any input elements from “the future”.

- Train with teacher forcing
  - at training time, feed into the model the actual word it was supposed to predict in previous position
  - not the one it actually predicted, which is what the model would have access to at inference

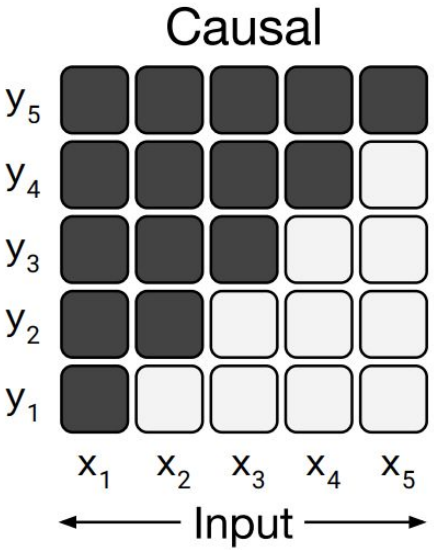



Image credit: Raffel et al 2020

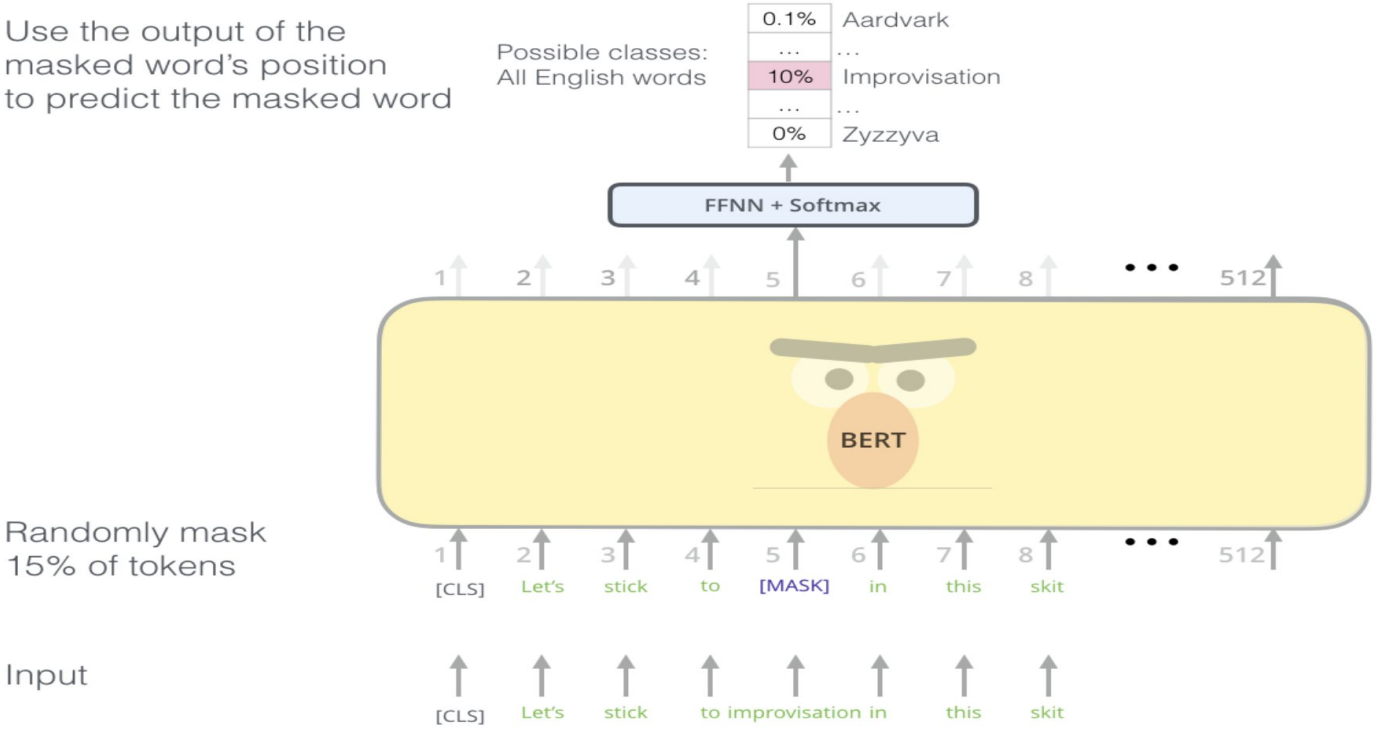
# De-noising objectives

- Model is trained to predict missing or otherwise corrupted tokens in the input

- BERT de-noising objective: **masked language modeling (MLM)**
  - randomly sample 15% of tokens and replace them with either [MASK] (90%) or a random token (10%); ask the model to reconstruct input

- T5 de-noising objective: **span dropout**
  - randomly sample and then drop out 15% of tokens in the input sequence. All consecutive spans of dropped-out tokens are replaced by a single sentinel token.

# BERT's De-noising MLM Objective




BERT's clever language modeling task masks 15% of words in the input and asks the model to predict the missing word.

Image credit: https://jalammar.github.io/illustrated-bert/

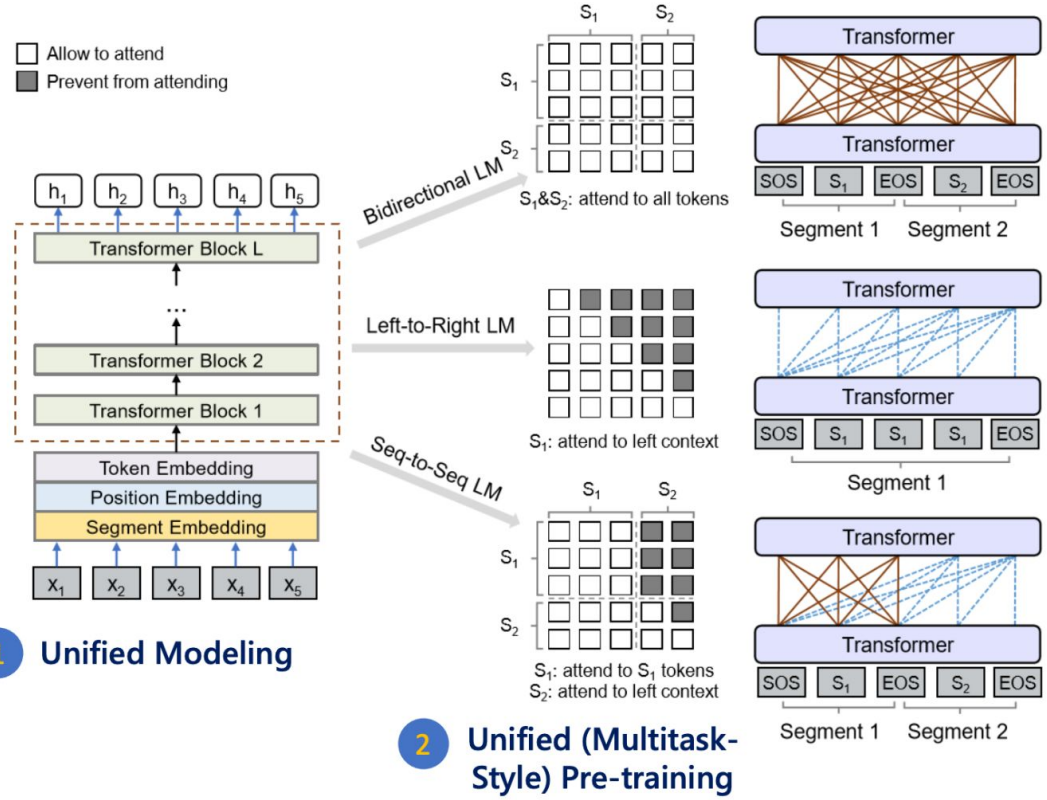
# Other encoders – BERT variants

- RoBERTa
  - optimized training regime for BERT: larger batches & learning rate
  - mask tokens dynamically: changes at each epoch (= ensembling)
- ALBERT
  - cross-layer parameter sharing
- XLNet
  - permute word order
- SpanBERT
  - mask spans of tokens
  - span boundary objective: predict masked span tokens from boundary tokens
  - inter-sentence coherence loss: reconstruct sentence order)

# Pre-training Sequence-to-Sequence Transformers

- UniLM
  - combine encoder and sequence-to-sequence objective
- MASS
  - reconstruct a sequence of masked tokens (sequence + no replacement, as in BERT)
- BART
  - combine encoder and decoder; encoder uses BERT-style objective, decoder (autoencoder) reproduce the original sequence token by token using previous tokens and encoder output
- T5
  - sequence-to-sequence
  - spans replaced with sentinel tokens; spans only need to be reconstructed
  - alternatively, spans are dropped; spans only need to be reconstructed
  - all NLP tasks cast in text-to-text format: prompt-answer; the model must generate the answer

# UniLM




BIDIRECTIONAL ENCODER

NLU: text classification, entity recognition, question answering, ...

UNIDIRECTIONAL DECODER

NLG: text generation, ...

BIDIRECTIONAL ENCODER AND UNIDIRECTIONAL DECODER

NLG (sequence-to-sequence): text summarization, question generation, ...

3 Unified Fine-tuning

Overview of the Unified Pre-training Framework in UniLM. We use different self-attention masks to control the access to context for each word token in different variants of LMs in pre-training and different NLU/NLG tasks in fine-tuning.

Slide credit: Furu Wei (UniLM)

# BART De-noising Objectives

- BERT-style encoder
- Decoder (autoencoder) reproduces the original sequence token by token

- Different noising strategies
  - **Token masking** – BERT-style MLM, but only replace with [MASK] tokens
  - **Token deletion** – Random tokens are dropped/deleted, with no replacement
  - **Text infilling** – token spans (length can very) are replaced with a single [MASK] token
  - **Sentence permutation** – the input is split on periods, sentences shuffled
  - **Document rotation** – a token is chosen at random, the sequence rotate to start with that token

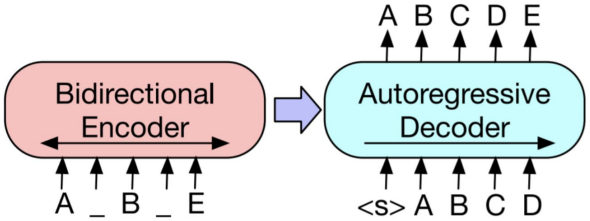



Image credit: Raffel et al 2020

# BART De-noising Objectives

- BERT-style encoder
- Decoder (autoencoder) reproduces the original sequence token by token

- Different noising strategies
  - **Token masking** – BERT-style MLM, but only replace with [MASK] tokens
  - **Token deletion** – Random tokens are dropped/deleted, with no replacement
  - **Text infilling** – token spans (length can very) are replaced with a single [MASK] token
  - **Sentence permutation** – the input is split on periods, sentences shuffled
  - **Document rotation** – a token is chosen at random, the sequence rotate to start with that token

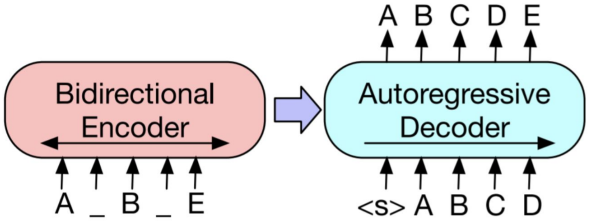



Image credit: Raffel et al 2020

# BART: Comparison between pre-training objectives

| Model | SQuAD 1.1 F1 | MNLI Acc | ELI5 PPL | XSum PPL | ConvAI2 PPL | CNN/DM PPL |
|---|---|---|---|---|---|---|
| BERT Base (Devlin et al., 2019) | 88.5 | **84.3** | - | - | - | - |
| Masked Language Model | 90.0 | 83.5 | 24.77 | 7.87 | 12.59 | 7.06 |
| Masked Seq2seq | 87.0 | 82.1 | 23.40 | 6.80 | 11.43 | 6.19 |
| Language Model | 76.7 | 80.1 | **21.40** | 7.00 | 11.51 | 6.56 |
| Permuted Language Model | 89.1 | 83.7 | 24.03 | 7.69 | 12.23 | 6.96 |
| Multitask Masked Language Model | 89.2 | 82.4 | 23.73 | 7.50 | 12.39 | 6.74 |
| BART Base | | | | | | |
| w/ Token Masking | 90.4 | 84.1 | 25.05 | 7.08 | 11.73 | 6.10 |
| w/ Token Deletion | 90.4 | 84.1 | 24.61 | 6.90 | 11.46 | 5.87 |
| w/ Text Infilling | **90.8** | 84.0 | 24.26 | **6.61** | **11.05** | 5.83 |
| w/ Document Rotation | 77.2 | 75.3 | 53.69 | 17.14 | 19.87 | 10.59 |
| w/ Sentence Shuffling | 85.4 | 81.5 | 41.87 | 10.93 | 16.67 | 7.89 |
| w/ Text Infilling + Sentence Shuffling | **90.8** | 83.8 | 24.17 | 6.62 | 11.12 | **5.41** |

# T5 De-noising Objective: Reconstructing Dropped Spans

- Randomly sample and then drop out 15% of tokens in the input sequence.
- All consecutive spans of dropped-out tokens are replaced by a single sentinel token.
- Ask the model to reconstruct the dropped span delimited by sentinels

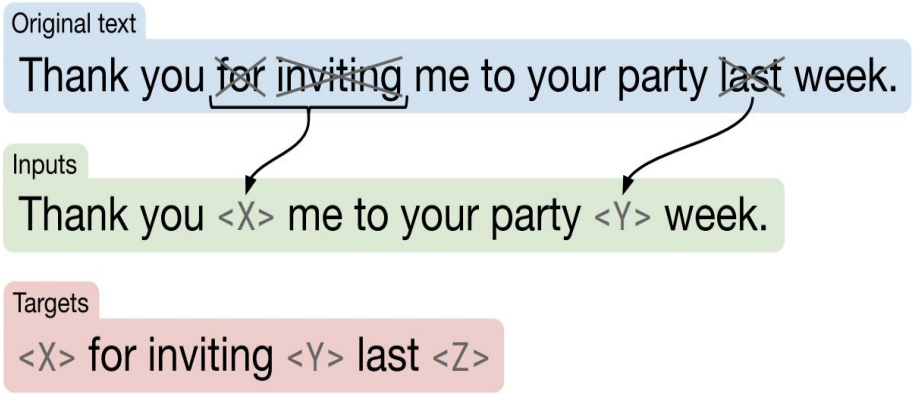
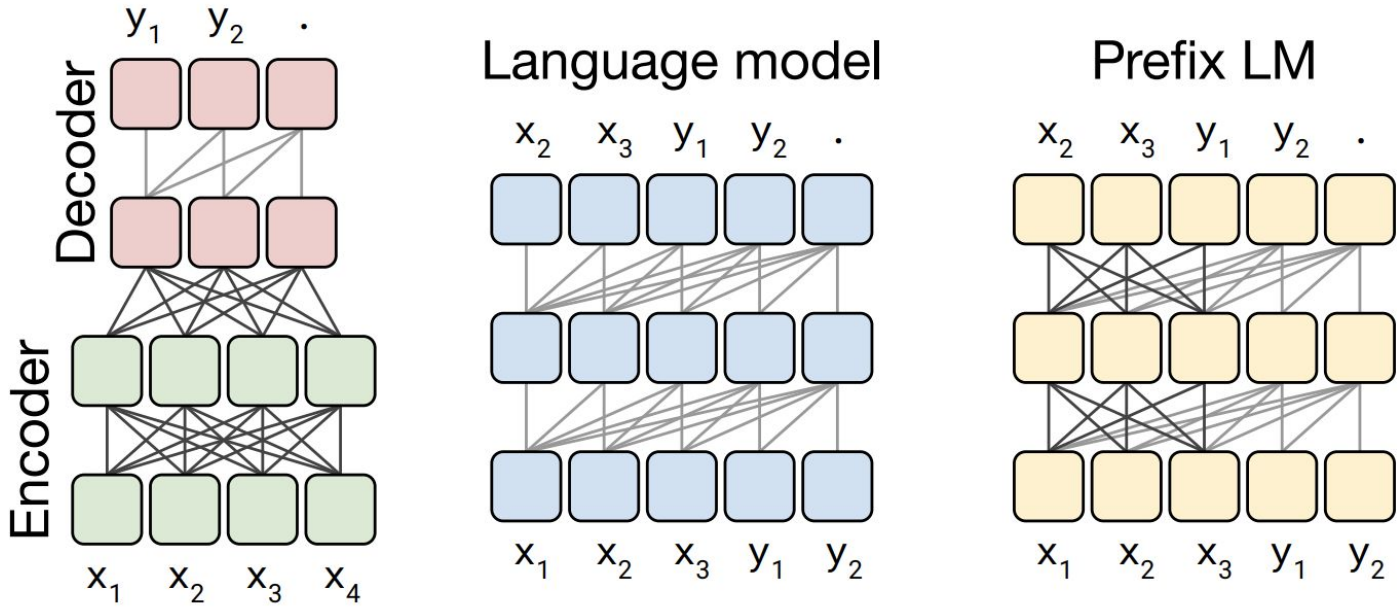
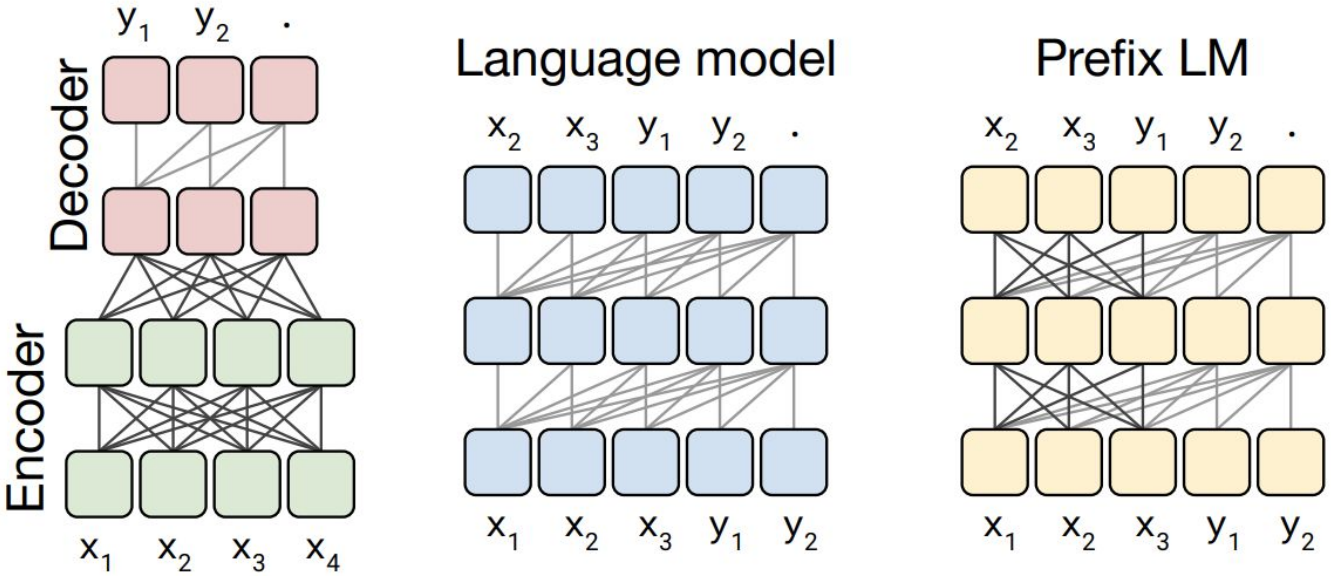



Image credit: Raffel et al 2020

# T5: Comparing Different Architectures



# T5: Comparing Different Architectures



# T5: Comparing Different Architectures

| Architecture | Objective | Params | Cost | GLUE | CNNDM | SQuAD | SGLUE | EnDe | EnFr | EnRo |
|---|---|---|---|---|---|---|---|---|---|---|
| ★ Encoder-decoder | Denoising | $2P$ | $M$ | **83.28** | **19.24** | **80.88** | **71.36** | **26.98** | **39.82** | **27.65** |
| Enc-dec, shared | Denoising | $P$ | $M$ | 82.81 | 18.78 | **80.63** | **70.73** | 26.72 | 39.03 | **27.46** |
| Enc-dec, 6 layers | Denoising | $P$ | $M/2$ | 80.88 | 18.97 | 77.59 | 68.42 | 26.38 | 38.40 | 26.95 |
| Language model | Denoising | $P$ | $M$ | 74.70 | 17.93 | 61.14 | 55.02 | 25.09 | 35.28 | 25.86 |
| Prefix LM | Denoising | $P$ | $M$ | 81.82 | 18.61 | 78.94 | 68.11 | 26.43 | 37.98 | 27.39 |
| Encoder-decoder | LM | $2P$ | $M$ | 79.56 | 18.59 | 76.02 | 64.29 | 26.27 | 39.17 | 26.86 |
| Enc-dec, shared | LM | $P$ | $M$ | 79.60 | 18.13 | 76.35 | 63.50 | 26.62 | 39.17 | 27.05 |
| Enc-dec, 6 layers | LM | $P$ | $M/2$ | 78.67 | 18.26 | 75.32 | 64.06 | 26.13 | 38.42 | 26.89 |
| Language model | LM | $P$ | $M$ | 73.78 | 17.54 | 53.81 | 56.51 | 25.23 | 34.31 | 25.38 |
| Prefix LM | LM | $P$ | $M$ | 79.68 | 17.84 | 76.87 | 64.86 | 26.28 | 37.51 | 26.76 |

Table 2: Performance of the different architectural variants described in Section 3.2.2. We use $P$ to refer to the number of parameters in a 12-layer base Transformer layer stack and $M$ to refer to the FLOPs required to process a sequence using the encoder-decoder model. We evaluate each architectural variant using a denoising objective (described in Section 3.1.4) and an autoregressive objective (as is commonly used to train language models).

# Testing Different Denoising Objectives

| Objective | Inputs | Targets |
|---|---|---|
| Prefix language modeling | Thank you for inviting | me to your party last week . |
| BERT-style Devlin et al. (2018) | Thank you <M> <M> me to your party apple week . | *(original text)* |
| Deshuffling | party me for your to . last fun you inviting week Thank | *(original text)* |
| MASS-style Song et al. (2019) | Thank you <M> <M> me to your party <M> week . | *(original text)* |
| I.i.d. noise, replace spans | Thank you <X> me to your party <Y> week . | <X> for inviting <Y> last <Z> |
| I.i.d. noise, drop tokens | Thank you me to your party week . | for inviting last |
| Random spans | Thank you <X> to <Y> week . | <X> for inviting me <Y> your party last <Z> |

# Results vary depending on the task

| Objective | GLUE | CNNDM | SQuAD | SGLUE | EnDe | EnFr | EnRo |
|---|---|---|---|---|---|---|---|
| BERT-style (Devlin et al., 2018) | 82.96 | 19.17 | **80.65** | 69.85 | 26.78 | **40.03** | 27.41 |
| MASS-style (Song et al., 2019) | 82.32 | 19.16 | 80.10 | 69.28 | 26.79 | **39.89** | 27.55 |
| ★ Replace corrupted spans | 83.28 | **19.24** | **80.88** | **71.36** | **26.98** | 39.82 | **27.65** |
| Drop corrupted tokens | **84.44** | **19.31** | **80.52** | 68.67 | **27.07** | 39.76 | **27.82** |

Table 5: Comparison of variants of the BERT-style pre-training objective. In the first two variants, the model is trained to reconstruct the original uncorrupted text segment. In the latter two, the model only predicts the sequence of corrupted tokens.

* MASS-style objective: no random token replacement, only masking omitted tokens
* Replace corrupted spans: the target is <X> <span 1> <Y> <span 2> ...
* Drop corrupted spans: the target is “<span 1><span 2> ...”

# Recasting all NLP tasks into Text-to-Text format

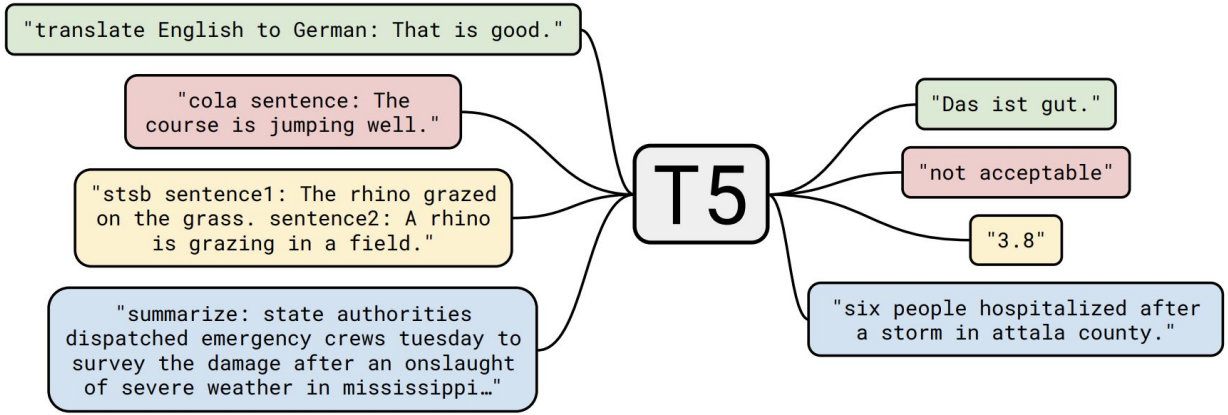



Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer".

Image credit: Raffel et al 2020

# Converting NLP tasks to T2T format

### D.4 MRPC

**Original input:**

**Sentence 1:** We acted because we saw the existing evidence in a new light , through the prism of our experience on 11 September , " Rumsfeld said .

**Sentence 2:** Rather , the US acted because the administration saw " existing evidence in a new light , through the prism of our experience on September 11 " .

**Processed input:** mrpc sentence1: We acted because we saw the existing evidence in a new light , through the prism of our experience on 11 September , " Rumsfeld said . sentence2: Rather , the US acted because the administration saw " existing evidence in a new light , through the prism of our experience on September 11 " .

**Original target:** 1

**Processed target:** equivalent

# Converting NLP tasks to T2T format

### D.5 QNLI

**Original input:**

**Question:** Where did Jebe die?

**Sentence:** Genghis Khan recalled Subutai back to Mongolia soon afterwards, and Jebe died on the road back to Samarkand.

**Processed input:** qnli question: Where did Jebe die? sentence: Genghis Khan recalled Subutai back to Mongolia soon afterwards, and Jebe died on the road back to Samarkand.

**Original target:** 0

**Processed target:** entailment

# T5 fine-tuned for open-domain QA, for generation

https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html

# Multi-task learning

- Methods for training on multiple tasks at once
- Produce a separate parameter setting that performs well on each task

- Since all tasks are text-to-text, including pre-training, it’s easy to combine the tasks into a single training regime

- They found that multitask training could be close to competitive with a pre-train-then-fine-tune approach but requires carefully choosing how often the model is trained on each task

# T0: Zero-shot generalization via multitask learning at scale

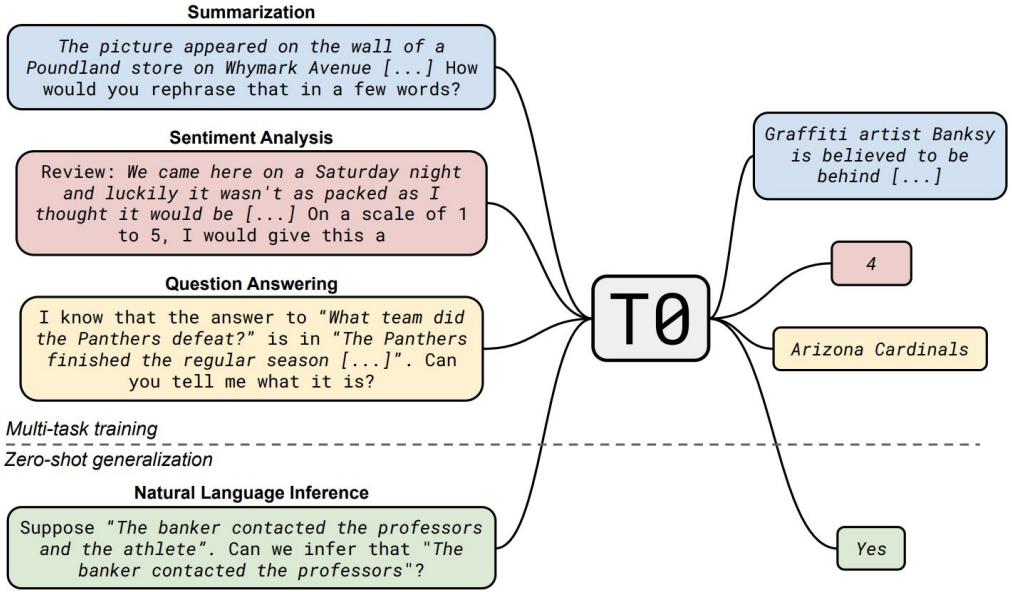



Figure 1: Our model and prompt format. T0 is an encoder-decoder model that consumes textual inputs and produces target responses. It is trained on a multitask mixture of NLP datasets partitioned into different tasks. Each dataset is associated with multiple prompt templates that are used to format example instances to input and target pairs. Italics indicate the inserted fields from the raw example data. After training on a diverse mixture of tasks (top), our model is evaluated on zero-shot generalization to tasks that are not seen during training (bottom).

Image credit: Sanh et al 2022

# T0: Zero-shot generalization via multitask learning at scale

- To convert a large set of natural language tasks into prompted form, they use a simple templating language for structured datasets.
- Develop an interface for prompt collection from public contributors that facilitated the collection of a large multitask mixture with multiple prompts per dataset (Bach et al., 2022).
- Train a variant of the T5 encoder-decoder model (Raffel et al., 2020; Lester et al., 2021) on a subset of the tasks (each with multiple datasets) and then evaluate tasks and prompts that the model was not trained on.
- T0 matches or exceeds the performance of GPT-3 on 9 out of 11 held-out datasets, despite being about 16× smaller.

# T0 Datasets

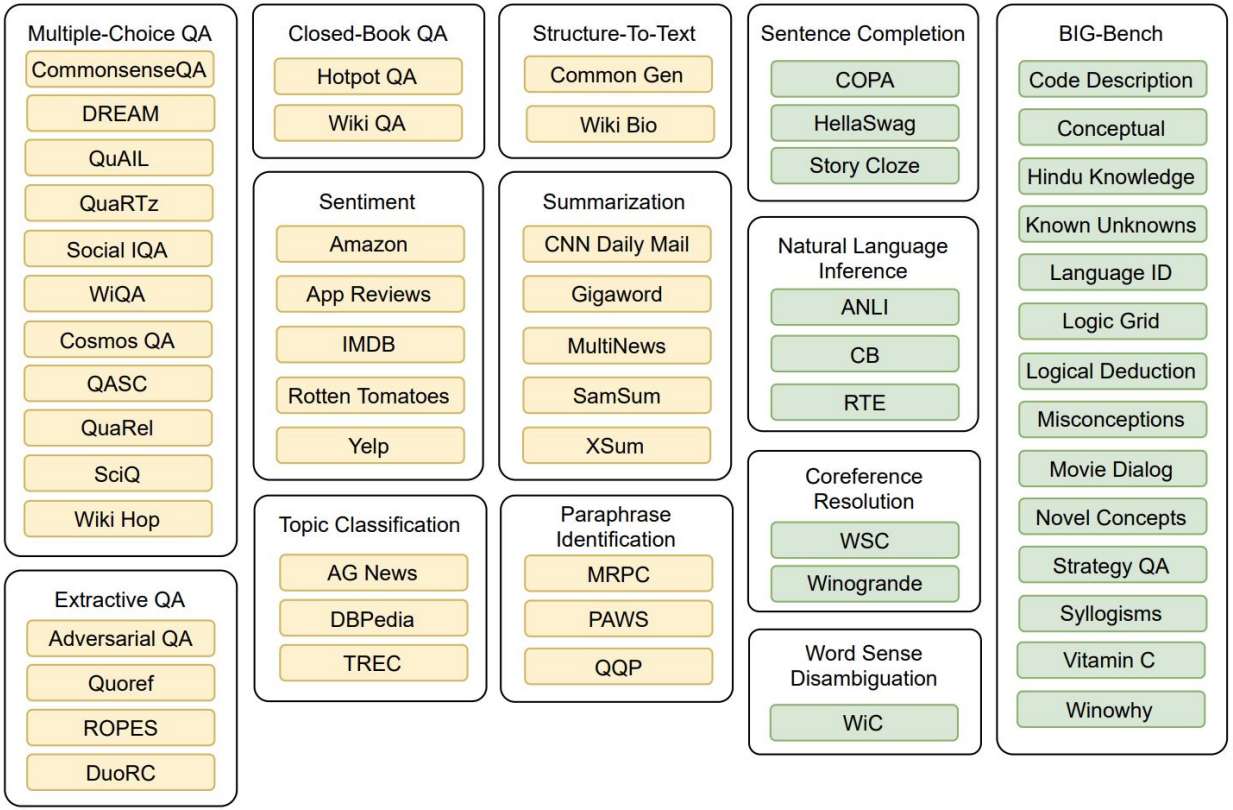



Figure 2: T0 datasets and task taxonomy. (T0+ and T0++ are trained on additional datasets. See Table 5 for the full list.) Color represents the level of supervision. Yellow datasets are in the training mixture. Green datasets are held out and represent tasks that were not seen during training. Hotpot QA is recast as closed-book QA due to long input length.

Image credit: Sanh et al 2022

# Public Pool of Prompts (P3)

2073 prompts for 177 datasets (11.7 prompts per dataset on average):

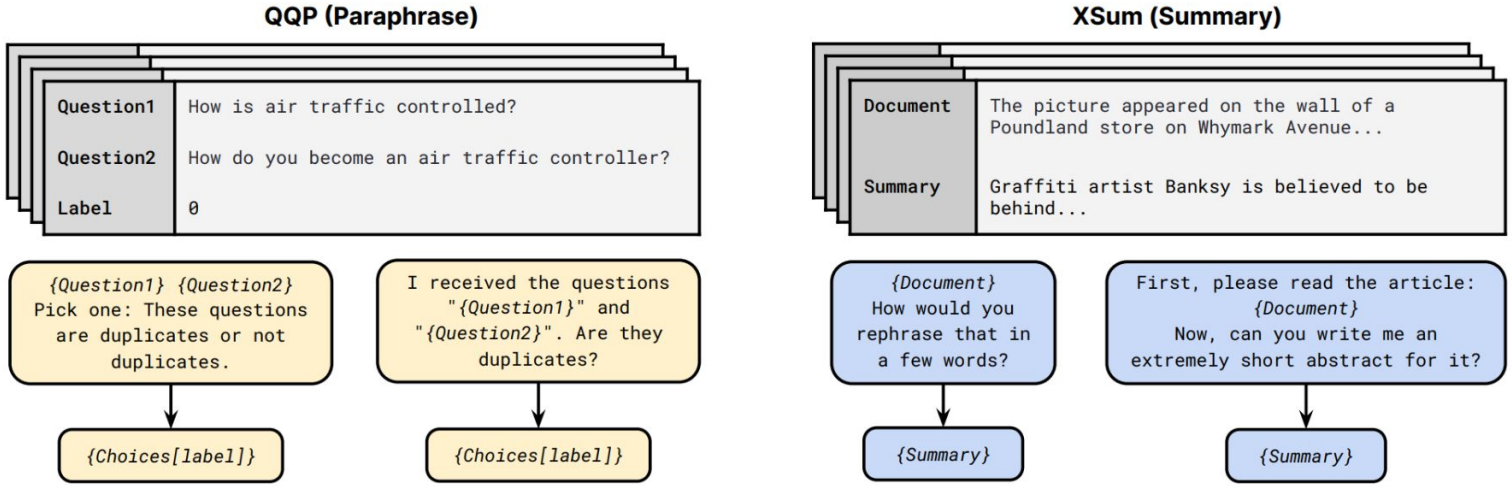



Figure 3: Prompt templates from the P3 prompt collection. Each dataset has multiple prompt templates consisting of an input and a target template. These use the fields of the raw data examples as well as template metadata, e.g., the left paraphrasing identification prompts use *Choices*, a template-level list variable `['Not duplicates', 'Duplicates']`. These templates are materialized

# Other Prompt Libraries

Natural Instructions:

**Cross-task generalization via natural language crowdsourcing instructions.** Mishra et al 2022. https://arxiv.org/abs/2104.08773

SuperNatural Instructions:

**Generalization via Declarative Instructions on 1600+ NLP Tasks**. Wang et al 2022. https://arxiv.org/pdf/2204.07705.pdf

- Single prompt per dataset
- 1600 datasets to choose from

P3 (Public Pool of Prompts) / PromptSource:

**PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts**. Bach et al 2022. https://arxiv.org/pdf/2202.01279.pdf

- Multiple prompts per dataset
- About 7 prompts per task on average

# Text-to-text task reformulation vs. prompting

Prompt consists of

1. Instruction
2. Demonstration (examples with answers) – skipped
3. Query (an example with no answer)
4. Target / verbalizer

# FLAN: Zero-shot generalization via multitask learning at scale

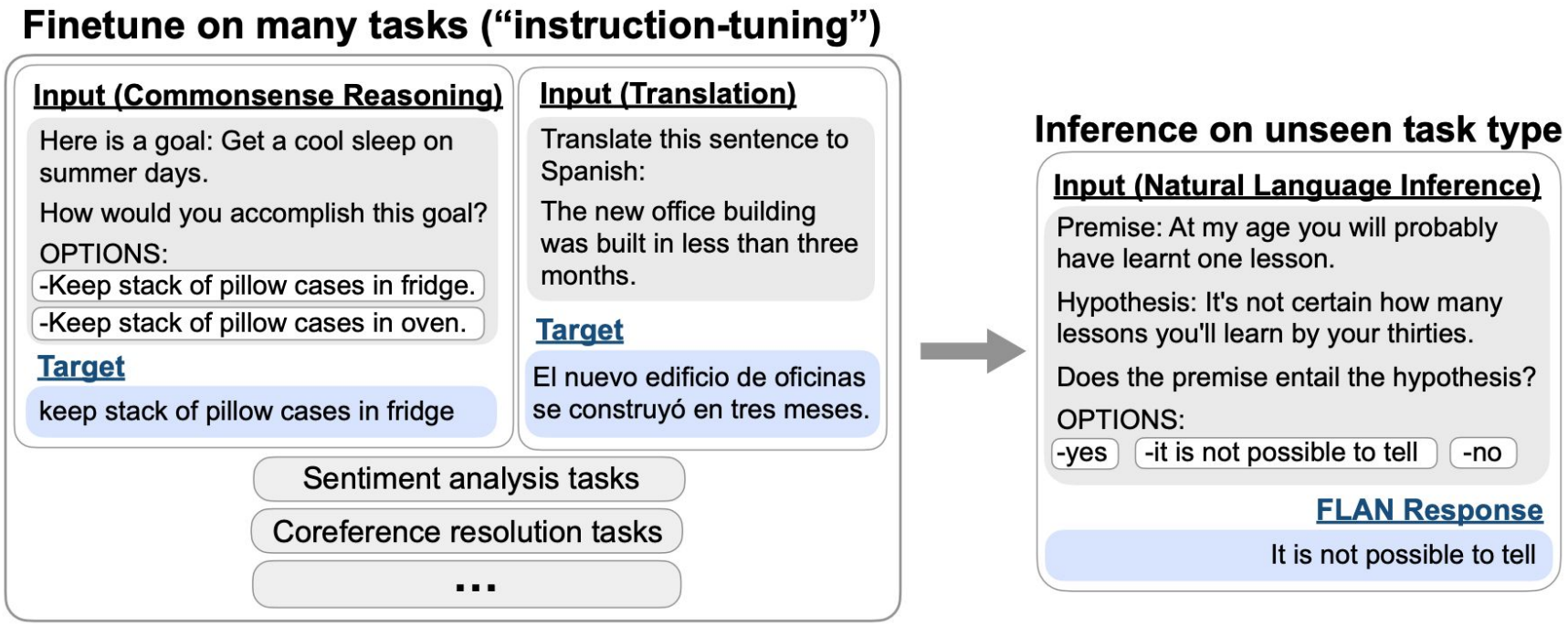



Image credit: Wei et al 2022

# FLAN: Zero-shot generalization via multitask learning at scale

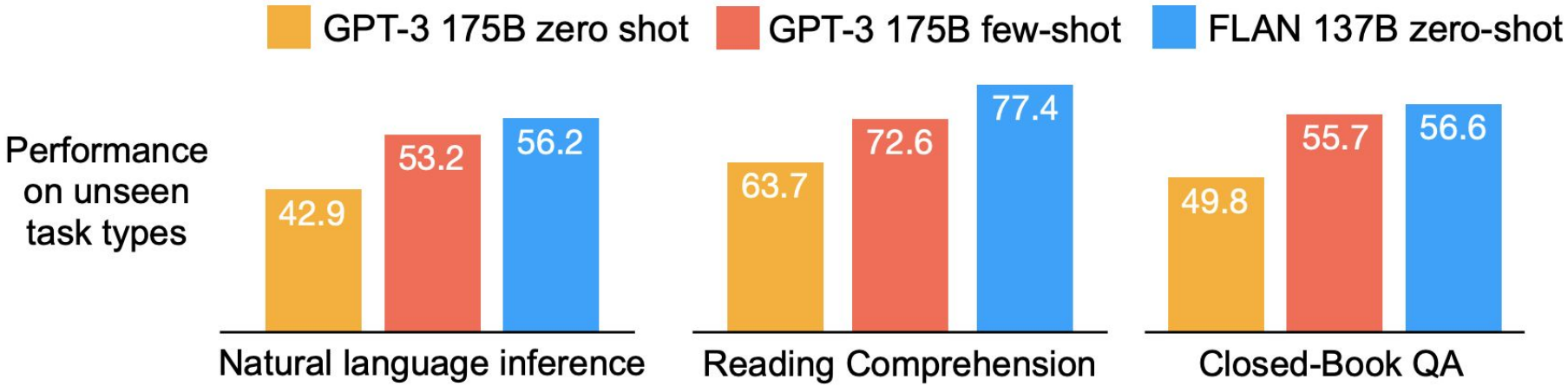



Image credit: Wei et al 2022

# Chain-of-Thought Prompting Elicits Reasoning in LLMs

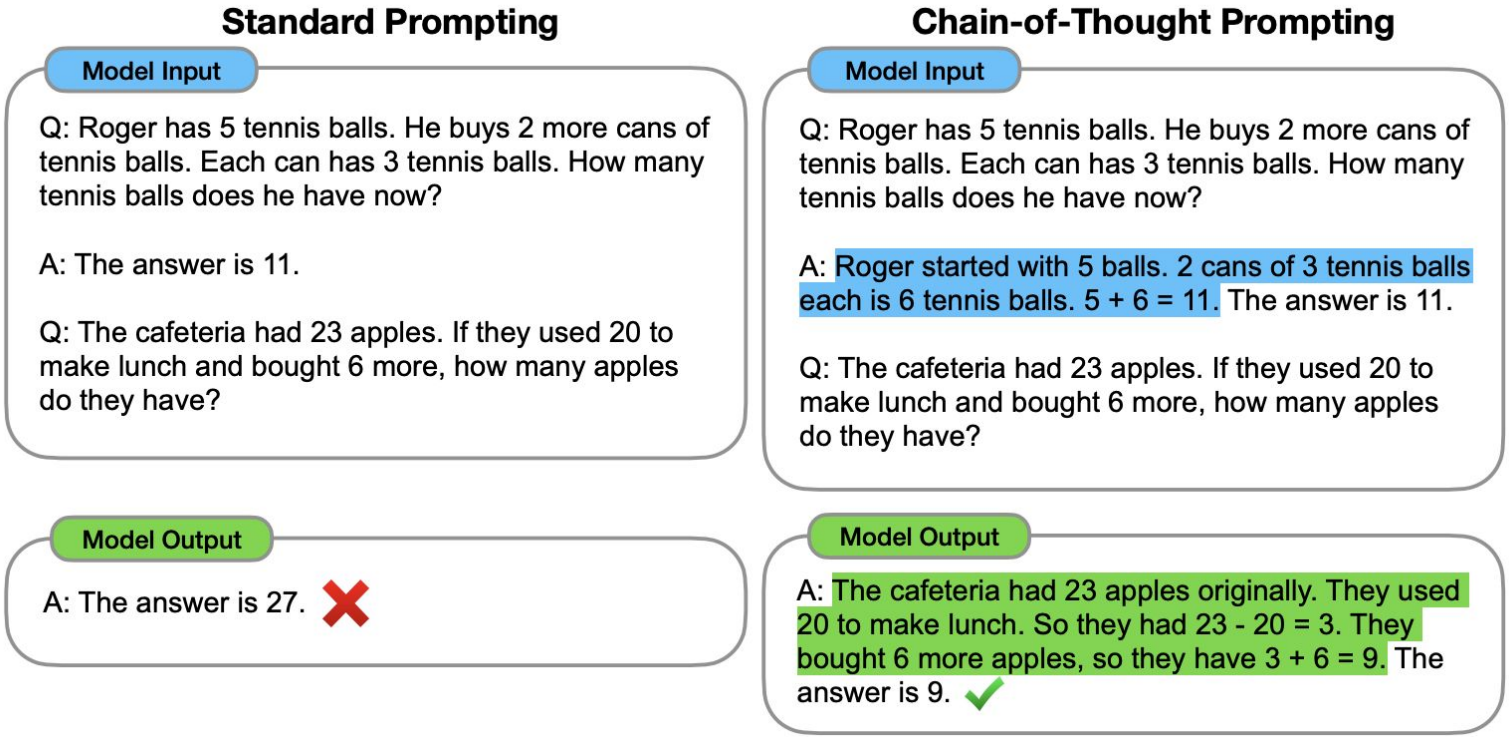



Image credit: Wei et al 2022

# Benchmarks

| Dataset | $N$ | Example problem |
|---|---|---|
| GSM8K | 1,319 | Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make? |
| SVAMP | 1,000 | Each pack of dvds costs 76 dollars. If there is a discount of 25 dollars on each pack. How much do you have to pay to buy each pack? |
| ASDiv | 2,096 | Ellen has six more balls than Marin. Marin has nine balls. How many balls does Ellen have? |
| AQuA | 254 | A car is being driven, in a straight line and at a uniform speed, towards the base of a vertical tower. The top of the tower is observed from the car and, in the process, it takes 10 minutes for the angle of elevation to change from 45° to 60°. After how much more time will this car reach the base of the tower? Answer Choices: (a) $5\sqrt{3}$ + 1 (b) $6\sqrt{3}$ + $\sqrt{2}$ (c) $7\sqrt{3}$ - 1 (d) $8\sqrt{3}$ - 2 (e) None of these |
| MAWPS: SingleOp | 562 | If there are 7 bottle caps in a box and Linda puts 7 more bottle caps inside, how many bottle caps are in the box? |
| MAWPS: SingleEq | 508 | Benny bought a soft drink for 2 dollars and 5 candy bars. He spent a total of 27 dollars. How much did each candy bar cost? |
| MAWPS: AddSub | 395 | There were 6 roses in the vase. Mary cut some roses from her flower garden. There are now 16 roses in the vase. How many roses did she cut? |
| MAWPS: MultiArith | 600 | The school cafeteria ordered 42 red apples and 7 green apples for students lunches. But, if only 9 students wanted fruit, how many extra did the cafeteria end up with? |

# Chain of thought vs. standard prompting vs prior best

Note that CoT does not really work with mid-size models, e.g. 20B, 50B.

In fact, in smaller models, it may hurt performance.

This seems to be one of the emergent behaviors that only larger models exhibit.

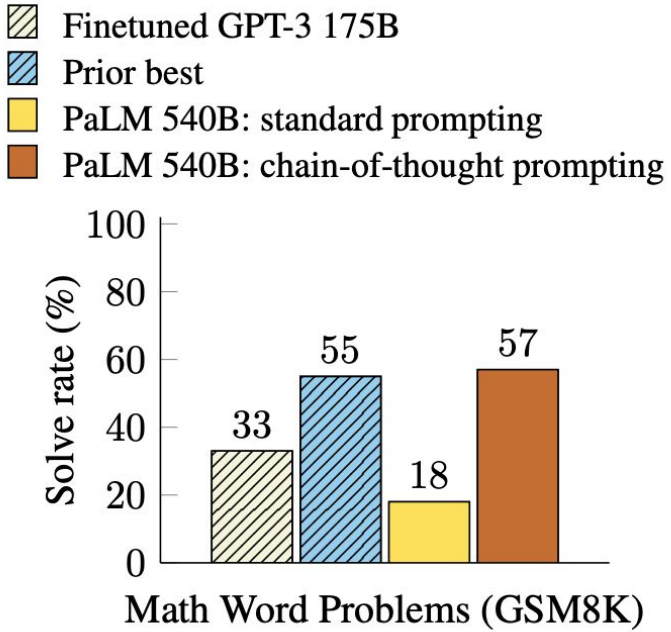



Image credit: Wei et al 2022

# “Automatic” Chain-of-Thought Prompting in LLMs

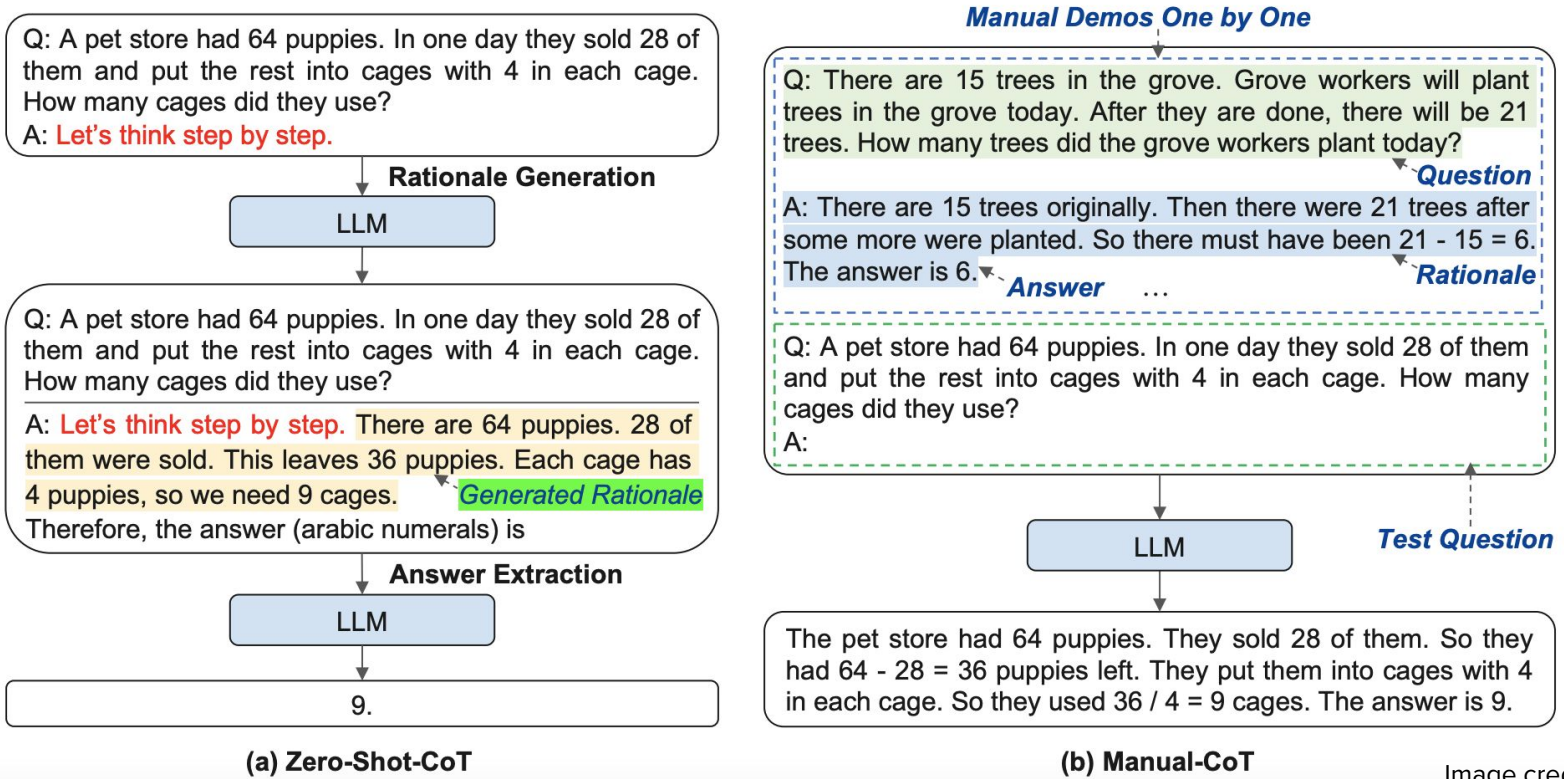



Image credit: Zhang et al 2022

*Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them.*

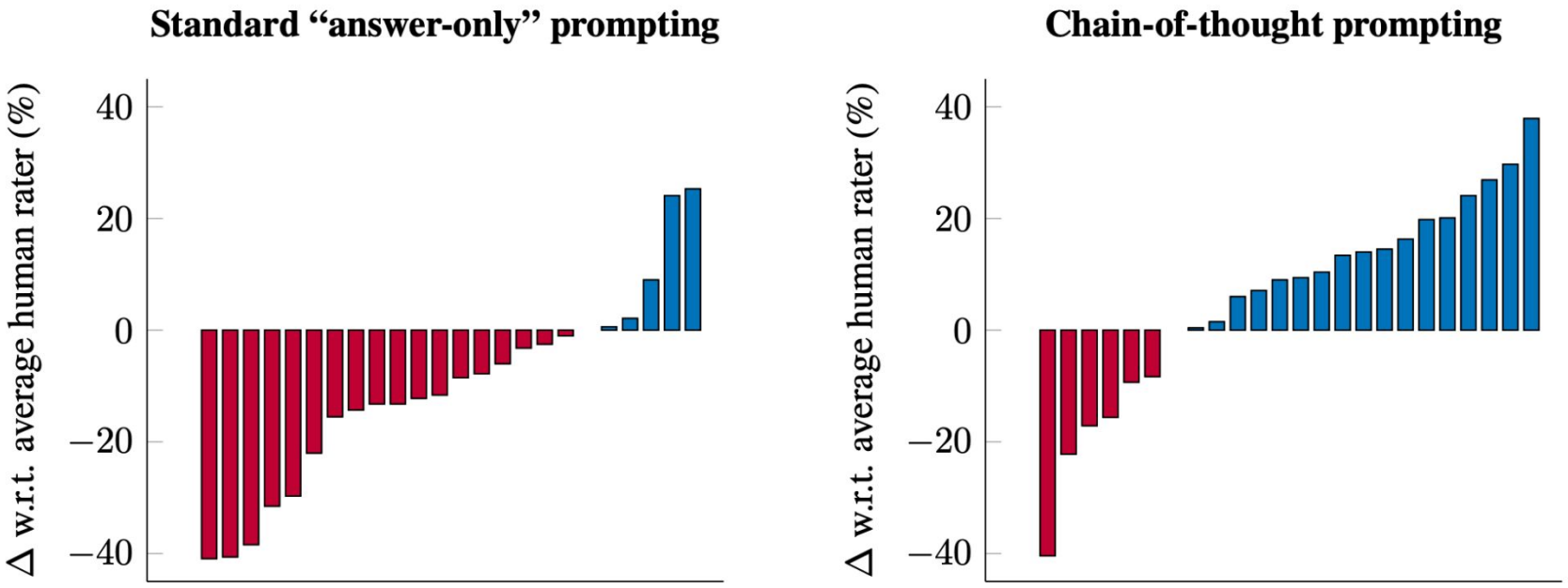



Figure 1: Per-task delta between Codex (code-davinci-002) and the average human-rater performance on 23 challenging tasks in BIG-Bench Hard, for standard *"answer-only"* (left) and *chain-of-thought* (right) prompting.

Image credit: Suzgun et al 2022

# Big Bench Tasks – Examples

**Model Input** (Navigate)

**<Few shot chain-of-thought exemplars omitted for brevity>**

**Q:** If you follow these instructions, do you return to the starting point? Turn left. Turn right. Take 5 steps. Take 4 steps. Turn around. Take 9 steps.
Options:
- Yes
- No
**A:** Let's think step by step.

**Model Output**

We start at the origin (0, 0), facing the positive y-axis.
(1) Turn left: (0, 0), facing the negative x-axis.
(2) Turn right: (0, 0), facing the positive y-axis.
(3) Take 5 steps: (0, 5), facing the positive y-axis.
(4) Take 4 steps: (0, 9), facing the positive y-axis.
(5) Turn around: (0, 9), facing the negative y-axis.
(6) Take 9 steps: (0, 0), facing the negative y-axis.
Since (0, 0) is (0, 0), we are indeed where we started. So the answer is **Yes**.

**Model Input** (Word Sorting)

**<Few shot chain-of-thought exemplars omitted for brevity>**

**Q:** Sort the following words alphabetically: List: burley bela arapaho bacteria bock
**A:** Let's think step by step.

**Model Output**

The first letter: "burley": "b" (2). "bela": "b" (2). "arapaho": "a" (1). "bacteria": "b" (2). "bock": "b" (2).
We now have: (1) "arapaho" < (2) ["burley" ? "bela" ? "bacteria" ? "bock"]. Now let's sort this subpart ["burley" ? "bela" ? "bacteria" ? "bock"] by looking at their second letters.
The second letter: "burley": "u" (21). "bela": "e" (5). "bacteria": "a" (1). "bock": "o" (15). We now have: (1) "bacteria" < (5) "bela" < (15) "bock" < (21) "burley". Hence, we have "arapaho" < ["bacteria" < "bela" < "bock" < "burley"]. So the answer is **arapaho bacteria bela bock burley**.

Figure 2: Examples of two tasks—*Navigate* (left) and *Word Sorting* (right)—from BBH with real model outputs.

# Big-Bench

BIG-bench currently consists of 204 tasks, contributed by 444 authors across 132 institutions. Task topics are diverse, drawing problems from linguistics, childhood development, math, common-sense reasoning, biology, physics, social bias, software development, and beyond.

**Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models**
https://arxiv.org/pdf/2206.04615.pdf

https://github.com/google/BIG-bench

# Using LLMs to self-engineer prompts

Large Language Models Are Human-Level Prompt Engineers.
https://arxiv.org/abs/2211.01910v1

- First, we use an LLM as an inference model (Ellis et al., 2021; Honovich et al., 2022) to generate instruction candidates based on a small set of demonstrations in the form of input-output pairs. Next, we guide the search process by computing a score for each instruction under the LLM we seek to control.

**Model Input**

I instructed my friend to **<M>**. The friend read the instruction and wrote an output for every one of the inputs. Here are the input-output pairs:

**Input:** Sentence 1: The dinosaurs became extinct. Sentence 2: A large object hit the Earth.
**Output:** A large object hit the Earth.
...
**Input:** Sentence 1: The company's posted strong earnings. Sentence 2: The company's stock went up.
**Output:** The company's posted strong earnings.

**Model Output**

**<M>: read both sentences and determine which one is the cause and which one is the effect. Choose the sentence that is the cause and write it as the output.**

# More Prompting Work

1. Minerva: Solving Quantitative Reasoning. *Solving Quantitative Reasoning Problems with Language Models* https://arxiv.org/abs/2206.14858
    - Trained PaLM model on arxiv to get the model to solve math problems better

2. *Self-Consistency Improves Chain of Thought Reasoning in Language Models.* https://arxiv.org/abs/2203.11171 [Denny Zhou's talk]
    - Ensembling over different (sampled) CoT paths that give the same answer in the end.

# Conclusions

- There is a variety of pre-training objectives
- Pre-training objectives matter: pre-training objectives should match what you’re planning to do downstream
- For large models, which are hard to fine-tune, in-context learning is a solid alternative to few-shot fine-tuning
- Reformulating NLP tasks in a single framework (e.g., text-to-text) allows you to blend pre-training and fine-tuning, so you get benefits of multi-task learning.
- Multi-task pre-training improves zero-shot generalization
- Large models acquire emergent abilities such as CoT prompting