
UF
Herbert Wertheim
College of Engineering
UNIVERSITY of FLORIDA
CIS 6930 Special Topics in Large Language Models
Pre-Training of LLMs
LEADING THE CHARGE, CHARGING AHEAD

Outline

- Pre-Training and Fine-Tuning paradigm

- Pre-Training of three types of transformer-based LM
  - Encoder LM (e.g., BERT)
  - Encoder-Decoder LM (e.g., T5)
  - Decoder-only LM (e.g., GPT)

Pre-Training and Fine-Tuning

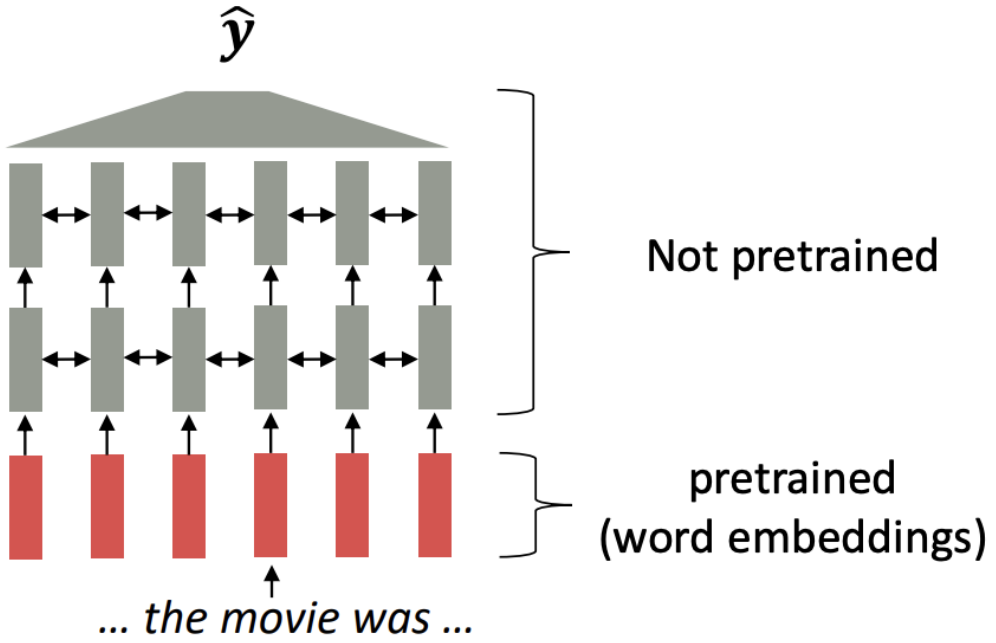
Motivation comes from pretrained word embeddings

Circa 2017:

- Start with pretrained word embeddings (no context!)
- Learn how to incorporate context in an LSTM or Transformer while training on the task.

Some issues to think about:

- The training data we have for our **downstream task** (like question answering) must be sufficient to teach all contextual aspects of language.
- Most of the parameters in our network are randomly initialized!



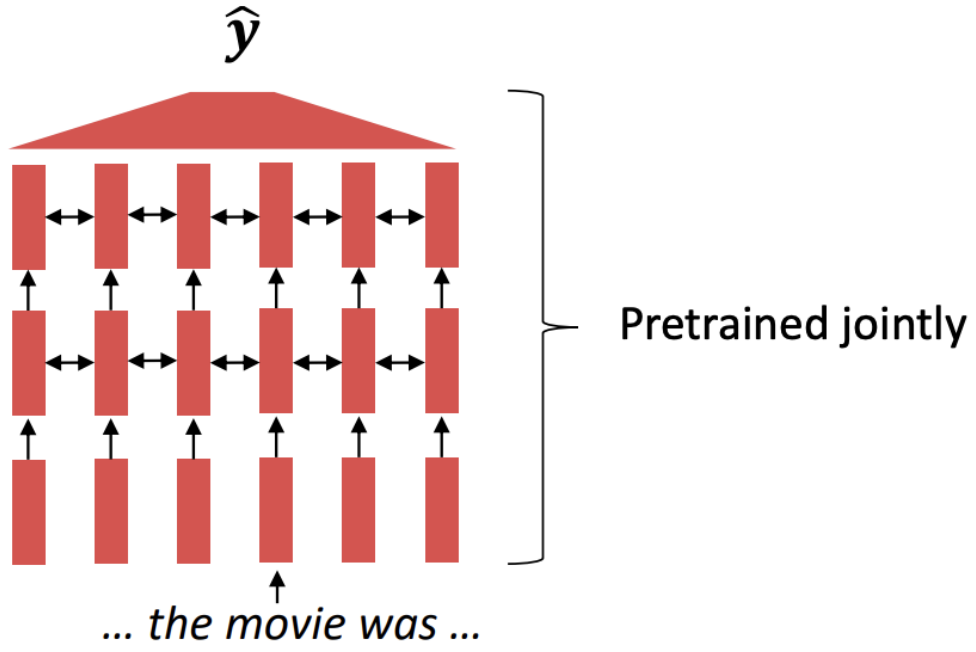
[Recall, *movie* gets the same word embedding, no matter what sentence it shows up in]

Pretraining Whole Models

In modern NLP:

- All (or almost all) parameters in NLP networks are initialized via **pretraining**.
- Pretraining methods hide parts of the input from the model, and train the model to reconstruct those parts.

- This has been exceptionally effective at building strong:
  - **representations of language**
  - **parameter initializations** for strong NLP models.
  - **Probability distributions** over language that we can sample from

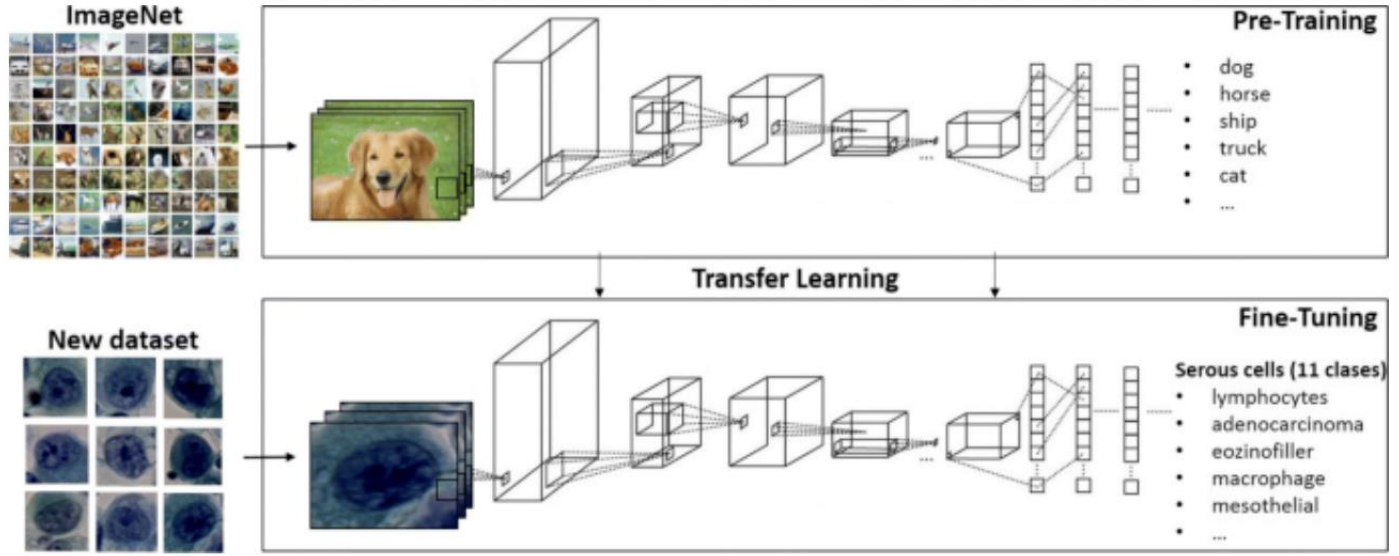


[This model has learned how to represent entire sentences through pretraining]

Pre-Training and Fine-Tuning

- "Pre-train" a model on a large dataset for task X, then "fine-tune" it on a dataset for task Y
- Key idea: X is somewhat related to Y, so a model that can do X will have some good neural representations for Y as well
- ImageNet pre-training is huge in computer vision: learning generic visual features for recognizing objects

Can we find some task X that can be useful for a wide range of downstream tasks Y?



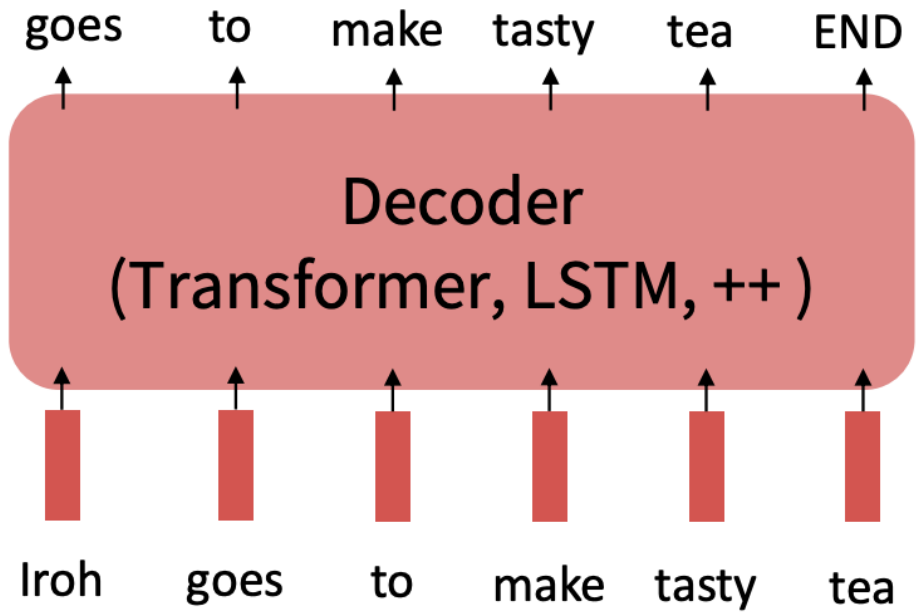
Pre-Training through Language Modeling

Recall the **language modeling** task:

- Model $p_\theta(w_t|w_{1:t-1})$, the probability distribution over words given their past contexts.
- There's lots of data for this! (In English.)

Pretraining through language modeling:

- Train a neural network to perform language modeling on a large amount of text.
- Save the network parameters.



Most Pre-Training is Reconstructing Input

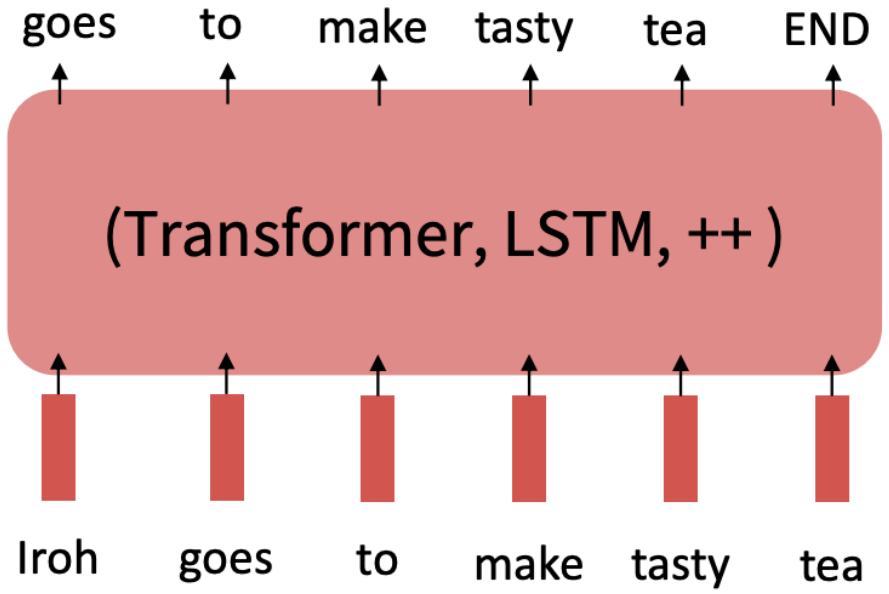
- *I put ___ fork down on the table.* [syntax]
- *The woman walked across the street, checking for traffic over ___ shoulder.* [coreference]
- *I went to the ocean to see the fish, turtles, seals, and _____.* [lexical semantics/topic]
- *Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was ___.* [sentiment]
- Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the _____. [some reasoning – this is harder]
- I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ____ [some basic arithmetic; they don't learn the Fibonnaci sequence]
- Models also learn – and can exacerbate racism, sexism, all manner of bad biases.

Pre-Training and Fine-Tuning

Pretraining can improve NLP applications by serving as parameter initialization.

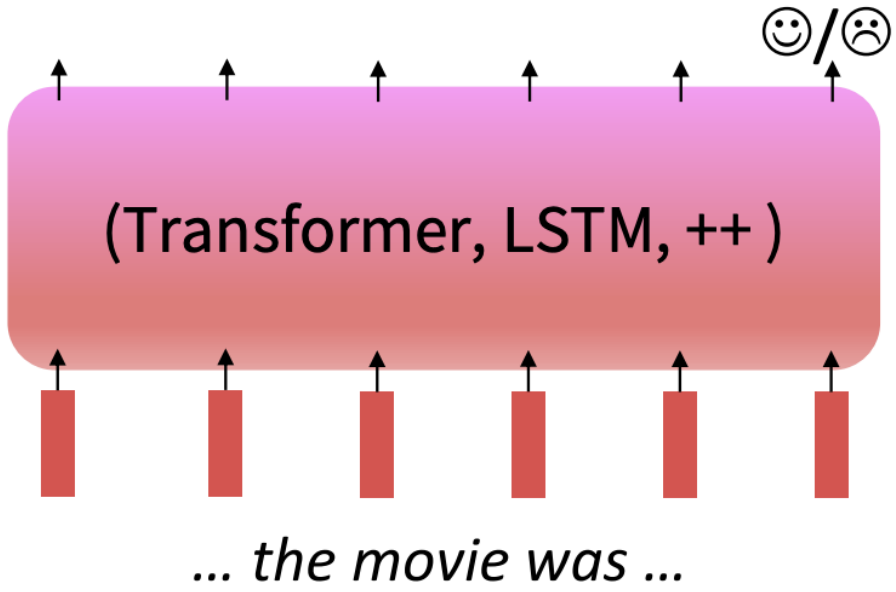
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



Step 2: Finetune (on your task)

Not many labels; adapt to the task!



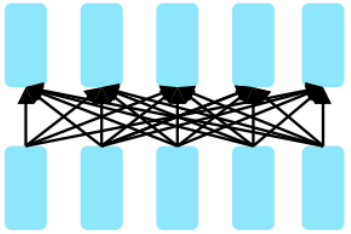
Pre-Training and Fine-Tuning

Why should pretraining and finetuning help, from a "training neural nets" perspective?

- Consider, provides parameters $\hat{\theta}$ by approximating $\min_{\theta} \mathcal{L}_{\text{pretrain}}(\theta)$.
  - (The pretraining loss.)
- Then, finetuning approximates $\min_{\theta} \mathcal{L}_{\text{finetune}}(\theta)$, starting at $\hat{\theta}$.
  - (The finetuning loss)
- The pretraining may matter because stochastic gradient descent sticks (relatively) close to $\hat{\theta}$ during finetuning.
  - So, maybe the finetuning local minima near $\hat{\theta}$ tend to generalize well!
  - And/or, maybe the gradients of finetuning loss near $\hat{\theta}$ propagate nicely!

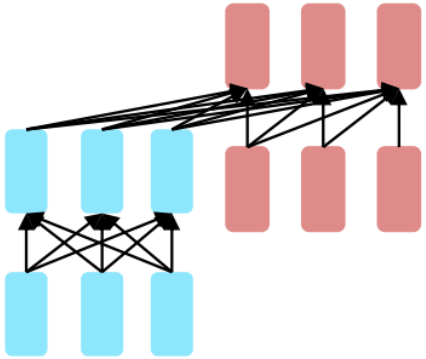
Pre-Training for three types of Transformer LM

The neural architecture influences the type of pretraining, and natural use cases.



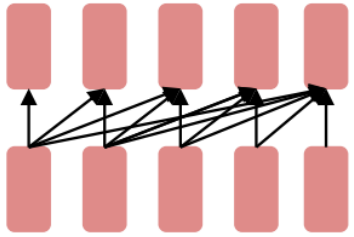
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



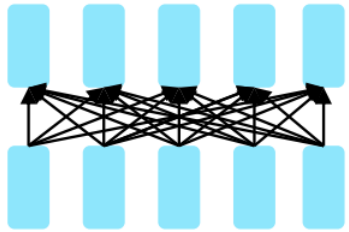
Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

Pre-Training for Encoder LM

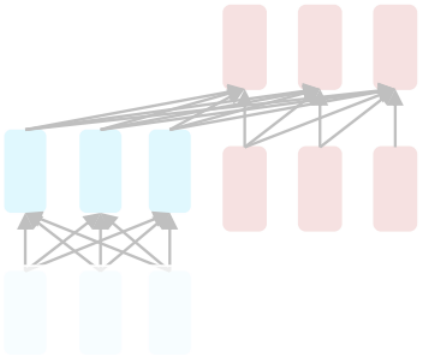
Pre-Training Encoder

The neural architecture influences the type of pretraining, and natural use cases.



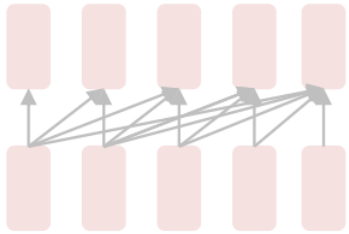
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



Decoders

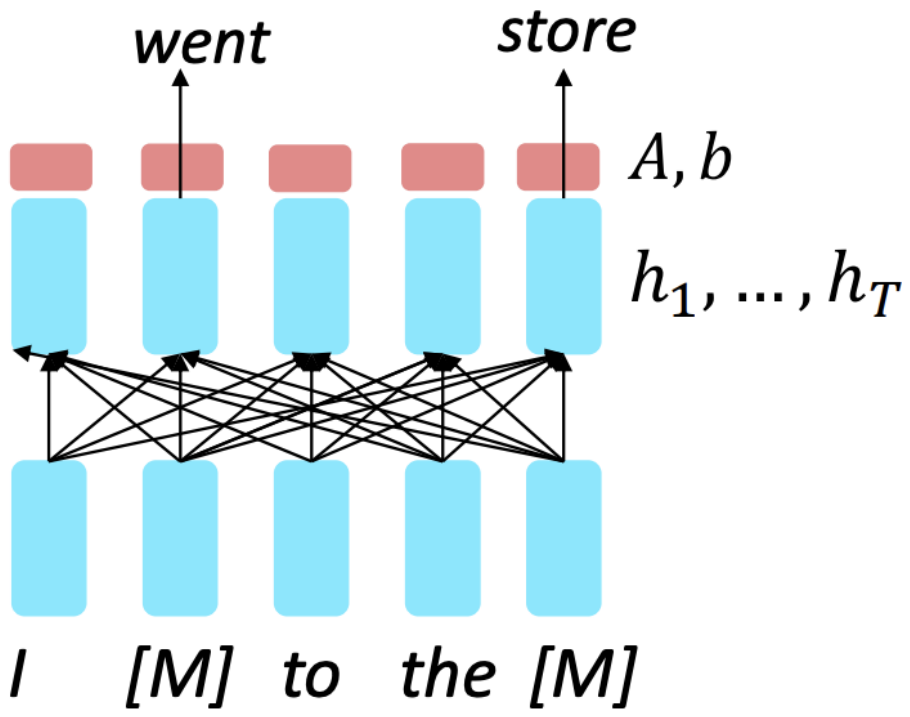
- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

Pre-Training Encoder

So far, we've looked at language model pretraining. But **encoders get bidirectional context,** so we can't do language modeling!

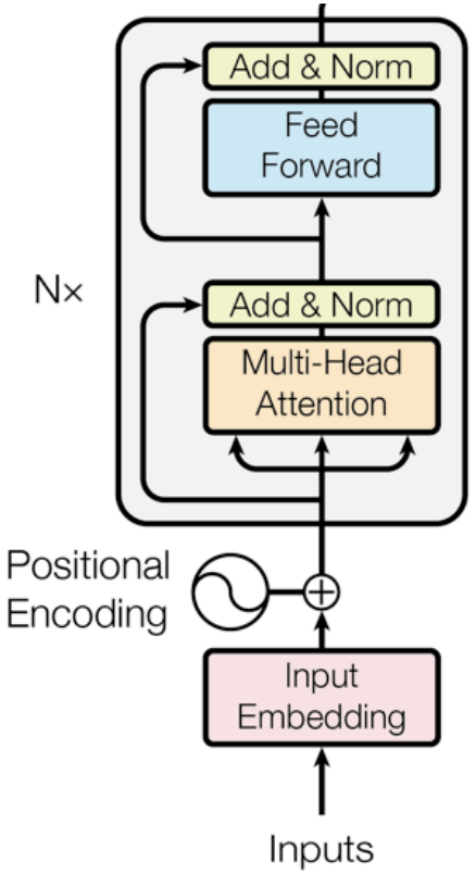
Idea: replace some fraction of words in the input with a special [MASK] token; predict these words.

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$y_i \sim Aw_i + b$$



Only add loss terms from words that are "masked out." If $\tilde{x}$ is the masked version of x, we're learning $p_\theta(x|\tilde{x})$. Called **Masked LM**.

BERT Bidirectional Encoder Representations from Transformers



- It is a fine-tuning approach based on a deep **bidirectional Transformer encoder** instead of a Transformer decoder
- The key: learn representations based on **bidirectional contexts**

  Example #1: we went to the river bank.

  Example #2: I need to go to bank to make a deposit.

- Two new pre-training objectives:
  - **Masked language modeling (MLM)**
  - Next sentence prediction (NSP) - Later work shows that NSP hurts performance though..

Masked Language Modeling (MLM)

- Q: Why we can't do language modeling with bidirectional models?

- Solution: Mask out k% of the input words, and then predict the masked words

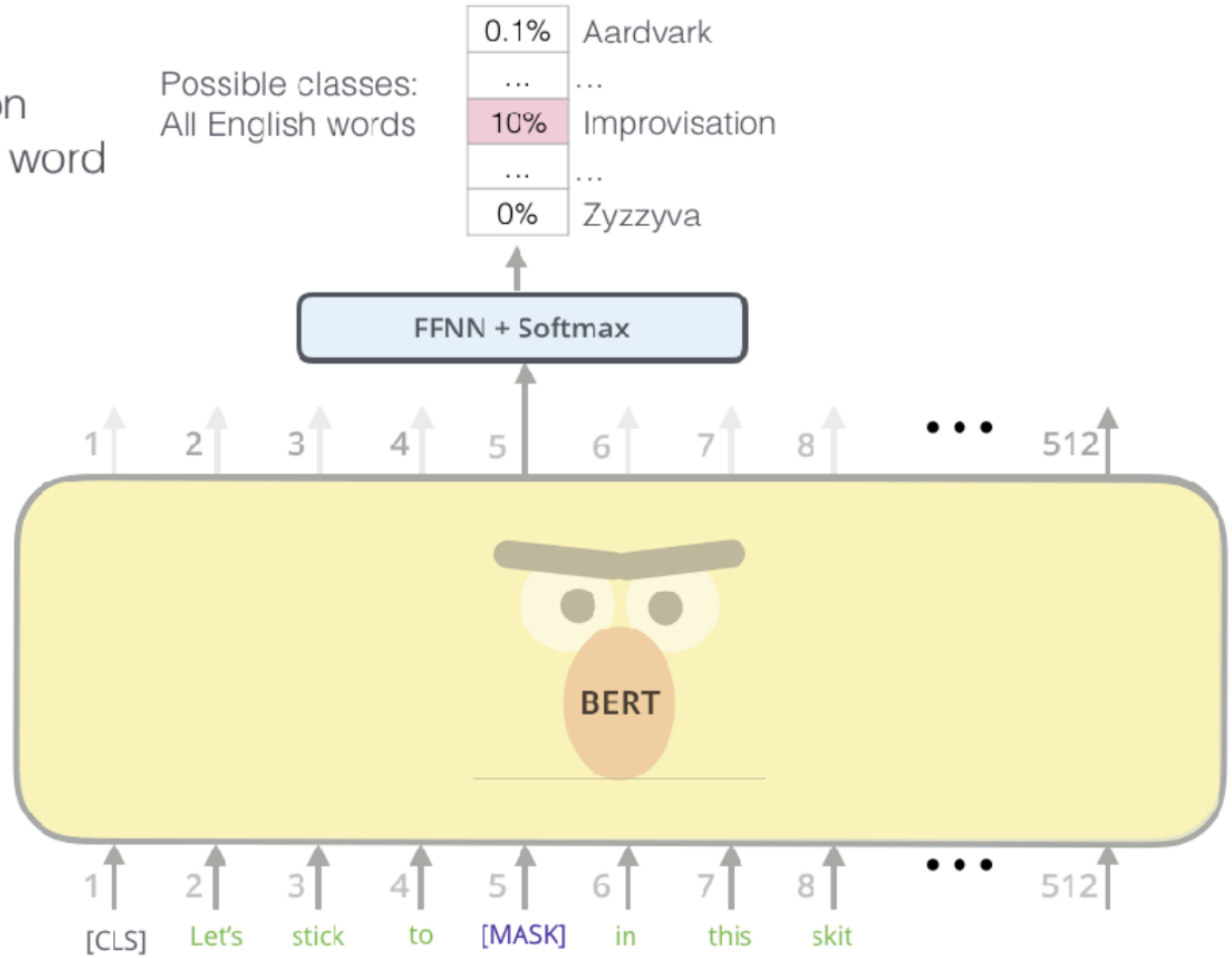
store gallon

the man went to [MASK] to buy a [MASK] of milk

k = 15% in practice

Masked Language Modeling (MLM)

Use the output of the masked word's position to predict the masked word



Randomly mask 15% of tokens

Input

Masked Language Modeling (MLM) 8/1/1 corruption

For the 15% predicted words,

- 80% of the time, they replace it with [MASK] token

  went to the store ⟶ went to the [MASK]

- 10% of the time, they replace it with a random word in the vocabulary

  went to the store ⟶ went to the running

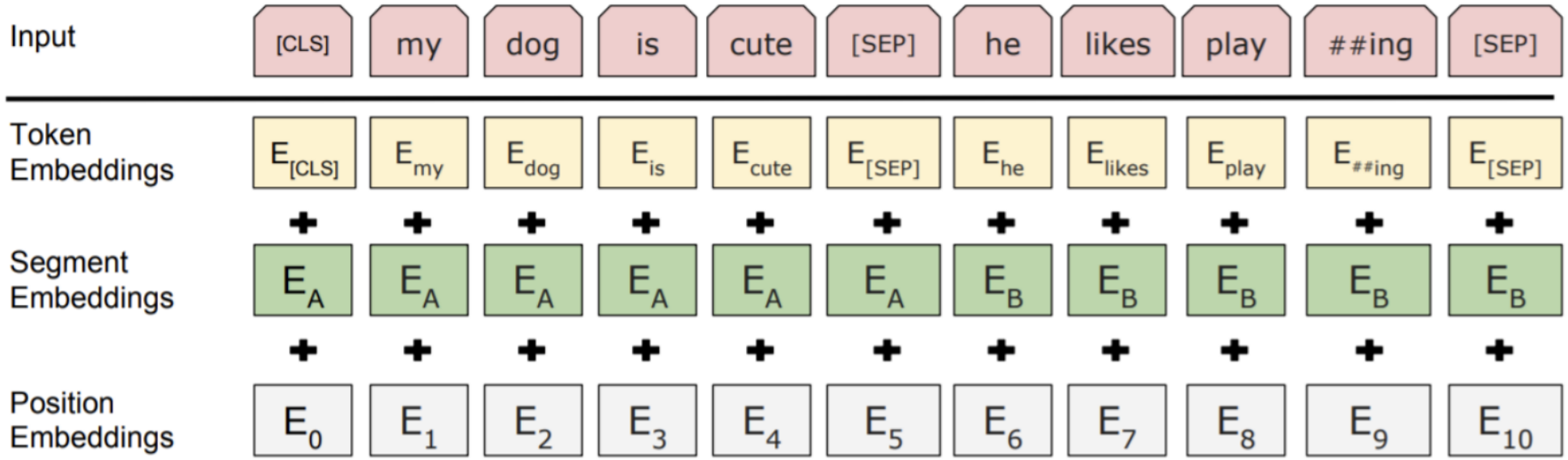
- 10% of the time, they keep it unchanged

  went to the store ⟶ went to the store

Why? Because [MASK] tokens are never seen during fine-tuning

Next Sentence Prediction

- The pretraining input to BERT was two separate contiguous chunks of text:



- BERT was trained to predict whether one chunk follows the other or is randomly sampled.
  - Later work has argued this “next sentence prediction” is not necessary.

Next Sentence Prediction

- Motivation: many NLP downstream tasks require understanding the relationship between two sentences (natural language inference, paraphrase detection, QA)
- NSP is designed to reduce the gap between pre-training and fine-tuning

[CLS]: a special token always at the beginning

[SEP]: a special token used to separate two segments

Input = `[CLS] the man went to [MASK] store [SEP]`

`he bought a gallon [MASK] milk [SEP]`

Label = `IsNext`

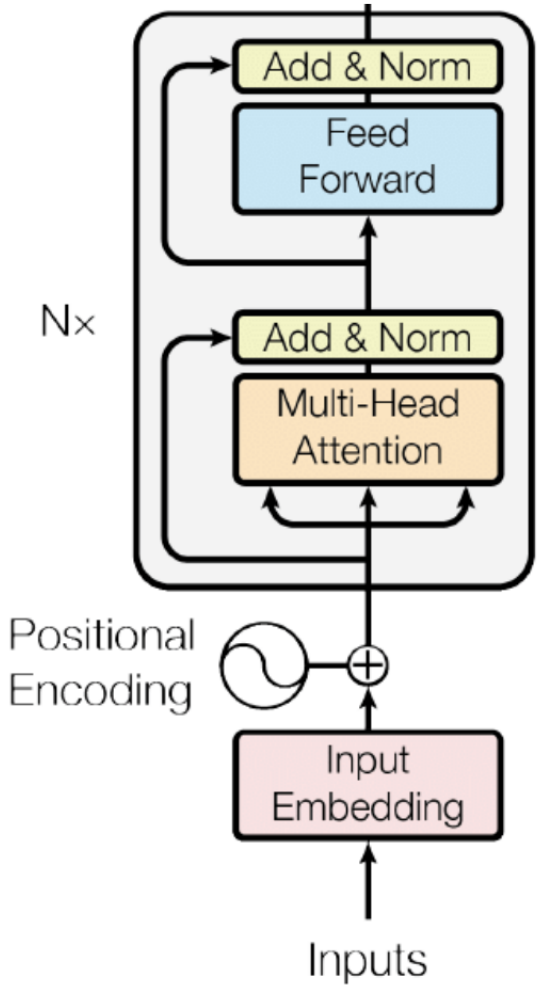
Input = `[CLS] the man [MASK] to the store [SEP]`

`penguin [MASK] are flight ##less birds [SEP]`

Label = `NotNext`

They sample two contiguous segments for 50% of the time and another random segment from the corpus for 50% of the time

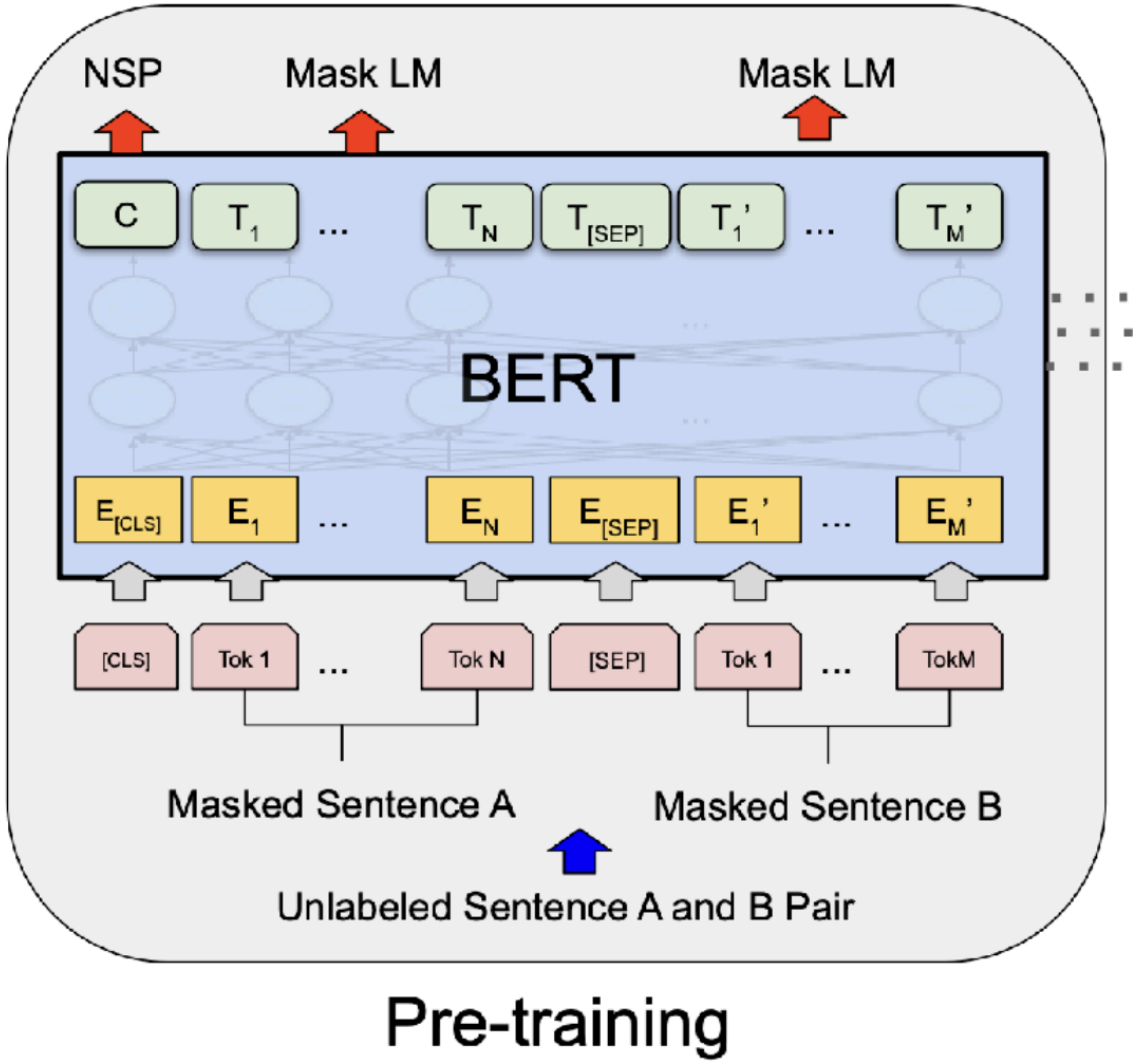
BERT Pre-Training



- BERT-base: 12 layers, 768 hidden size, 12 attention heads, 110M parameters
- BERT-large: 24 layers, 1024 hidden size, 16 attention heads, 340M parameters

- Training corpus: Wikipedia (2.5B) + BooksCorpus (0.8B)
- Max sequence size: 512 wordpiece tokens (roughly 256 and 256 for two non-contiguous sequences)
- Trained for 1M steps, batch size 128k

BERT Pre-Training



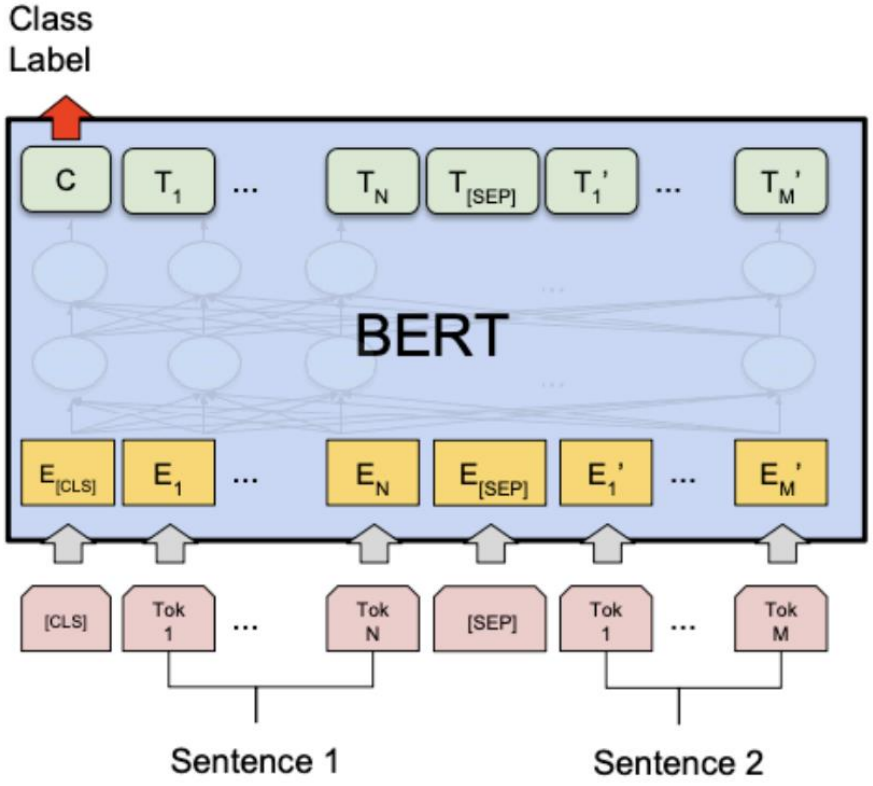
Pre-training

- MLM and NSP are trained together
- [CLS] is pre-trained for NSP
- Other token representations are trained for MLM

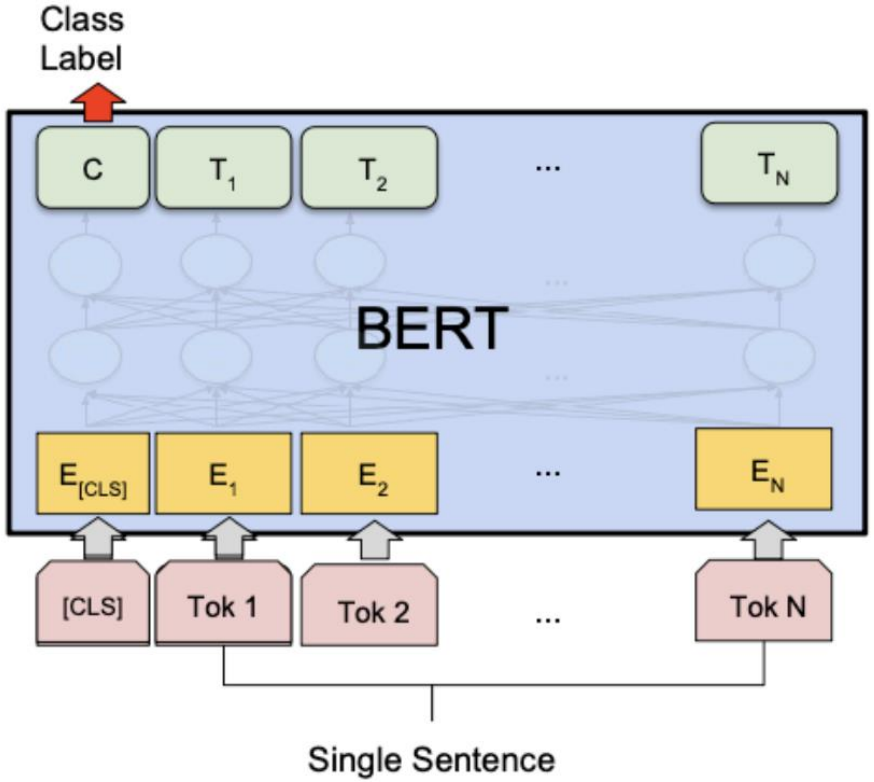
BERT Fine-Tuning – Sentence-level Tasks

"Pre-train once, finetune many times."

sentence-level tasks



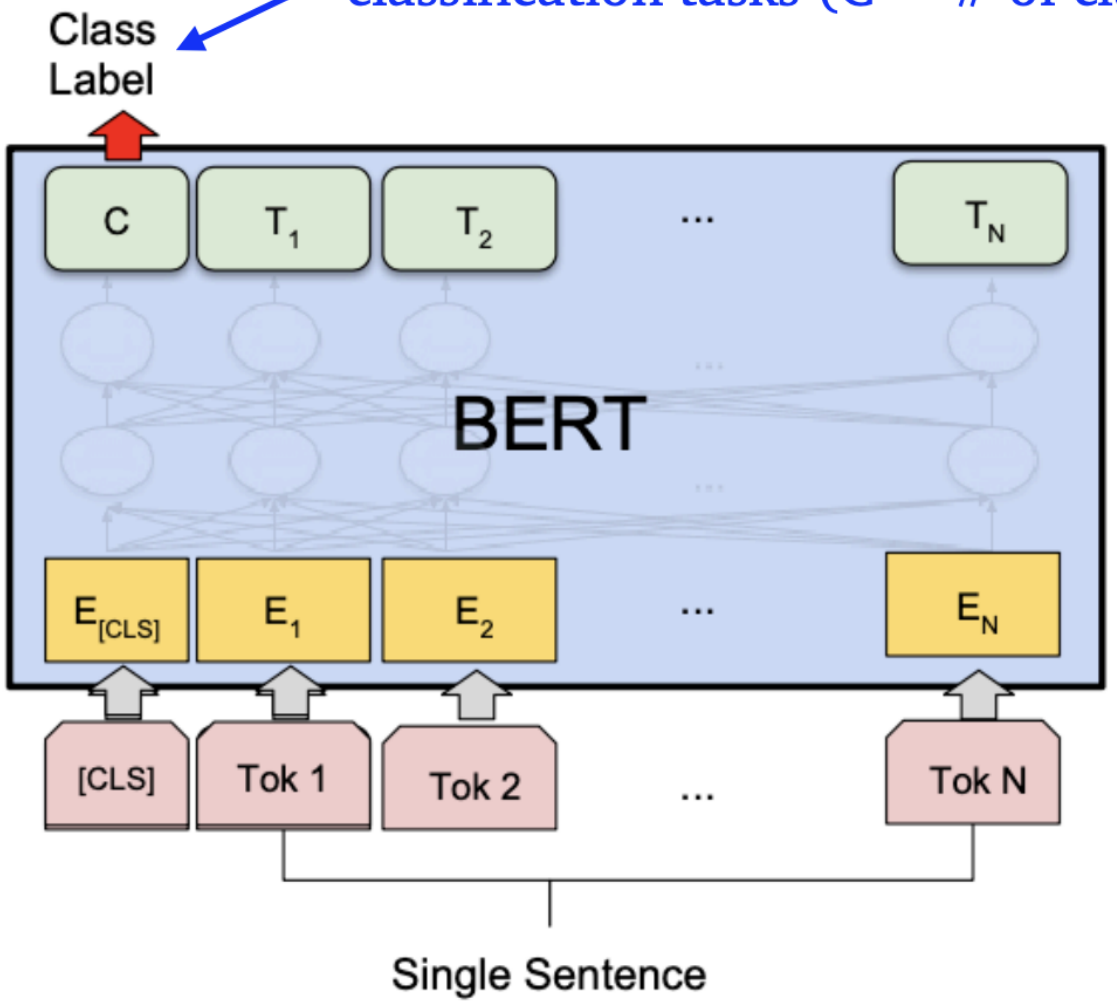
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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

Example: Sentiment Classification

We just need to introduce $C \times h$ parameters for classification tasks (C = # of classes, h = hidden size)!



$$P(y = k) = softmax_k(\mathbf{W}_o \mathbf{h}_{[CLS]})$$

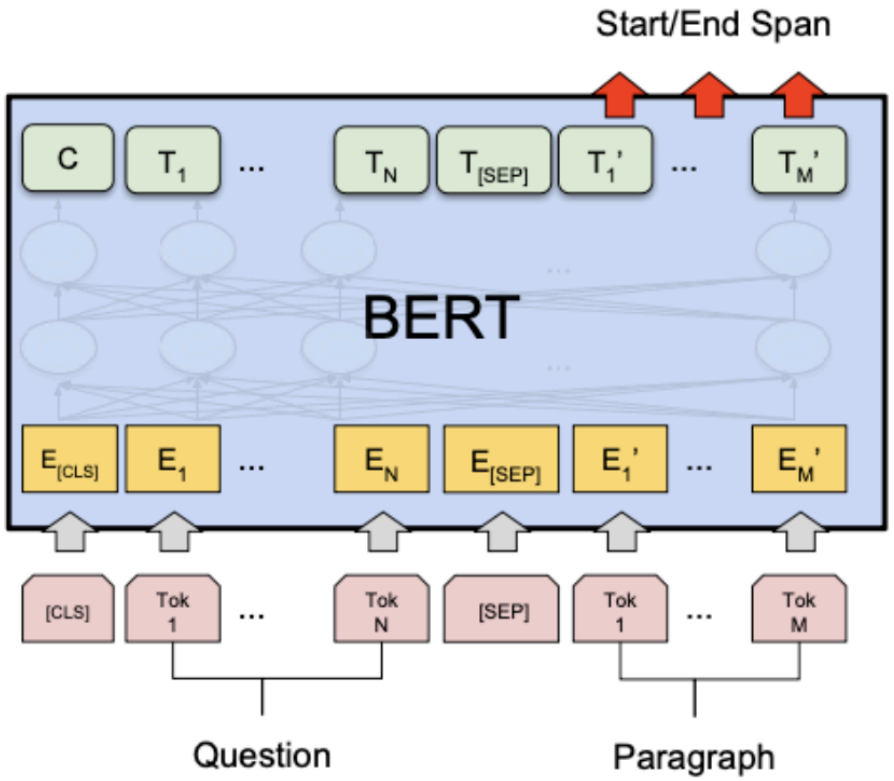
$$\mathbf{W}_o \in \mathbb{R}^{C \times h}$$

All the parameters will be learned together (original BERT parameters + new classifier parameters)

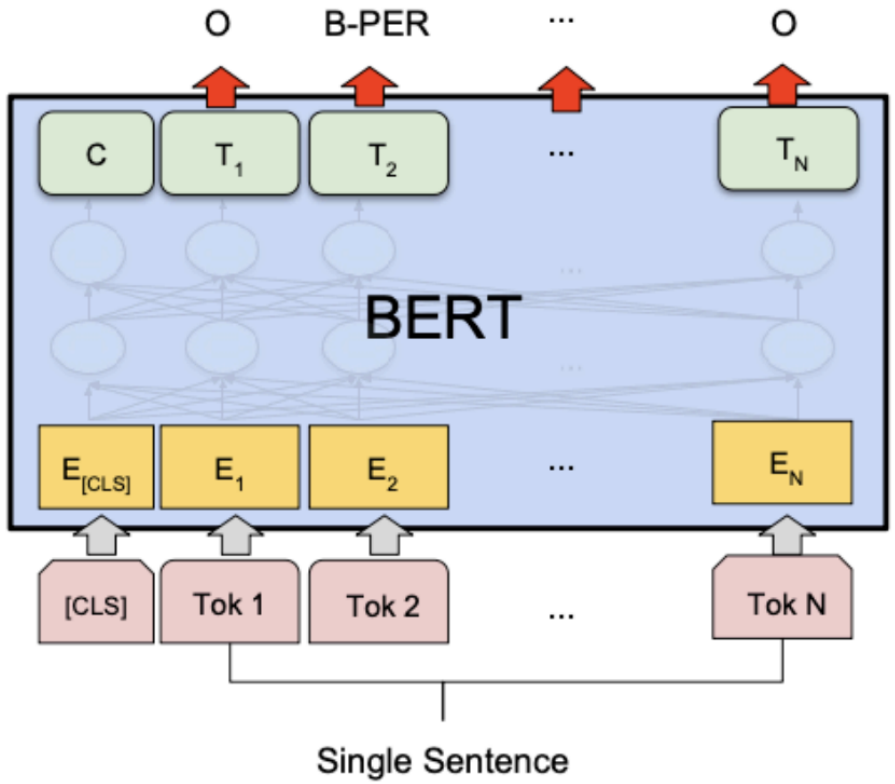
BERT Fine-Tuning – Token-level Tasks

"Pretrain once, finetune many times."

token-level tasks



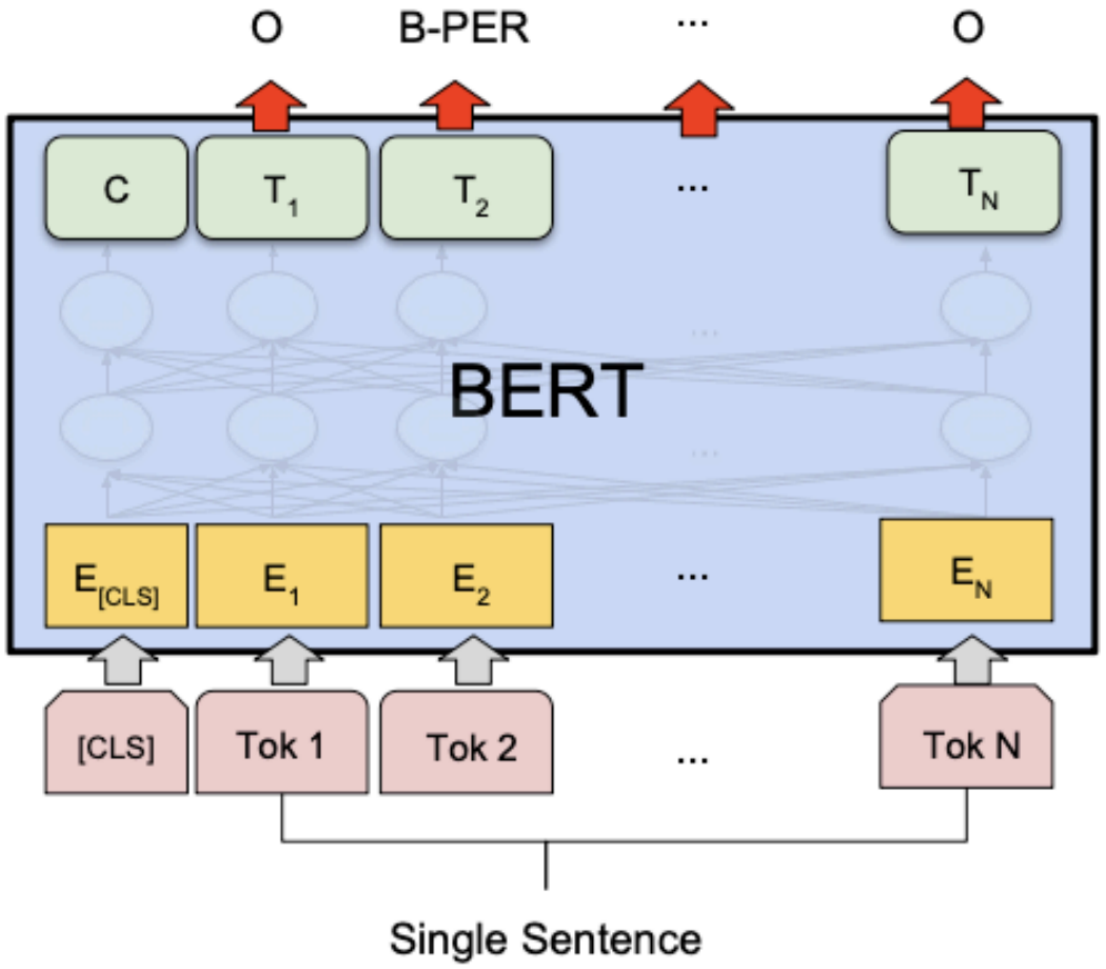
(c) Question Answering Tasks: SQuAD v1.1



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

Example: Name Entity Recognition (NER)

We just need to introduce $C \times h$ parameters for classification tasks (C = # of classes, h = hidden size)!



$$P(y_i = k) = softmax_k(\mathbf{W}_o \mathbf{h}_i)$$

$$\mathbf{W}_o \in \mathbb{R}^{C \times h}$$

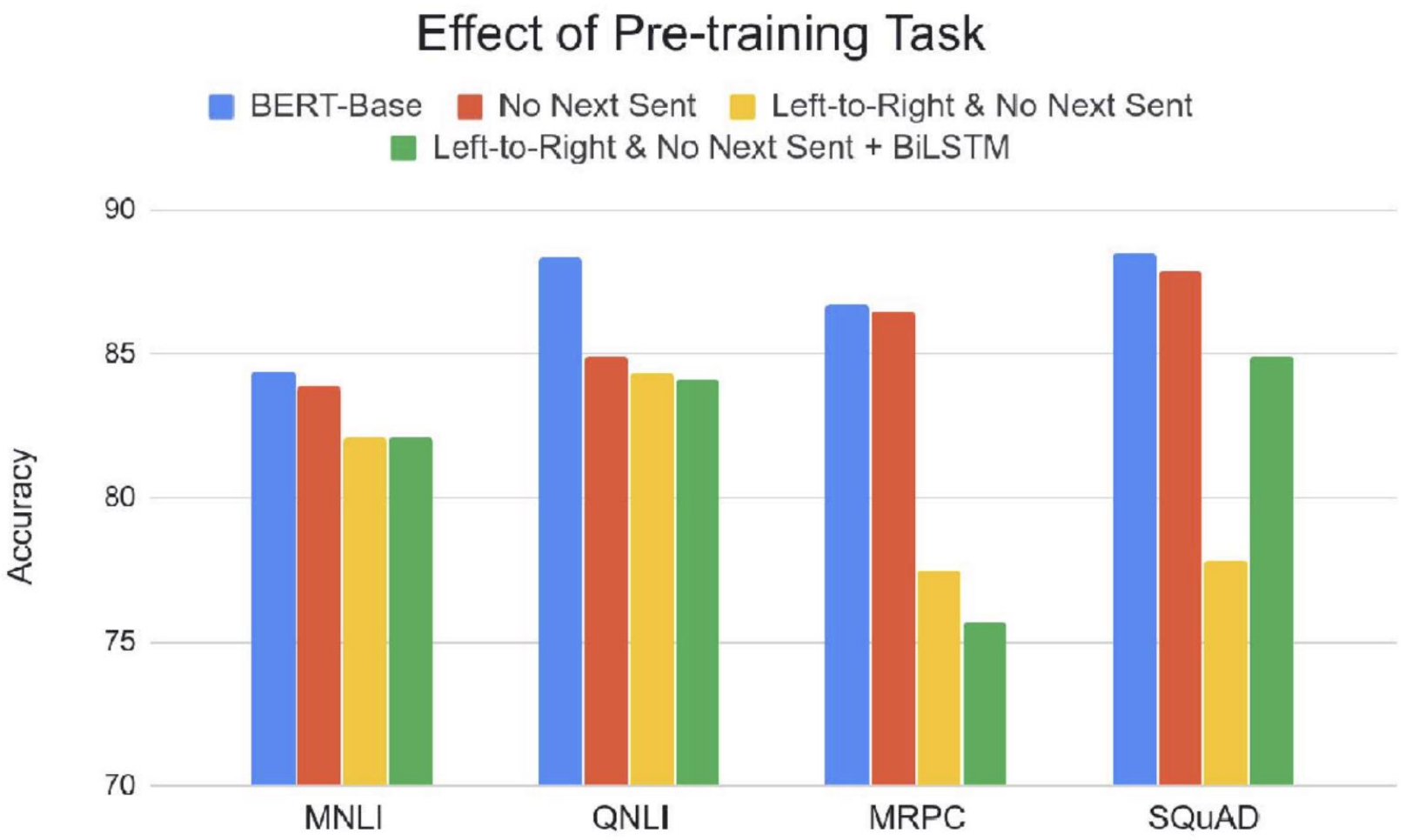
Performance

BERT was massively popular and hugely versatile; finetuning BERT led to new state-of-the-art results on a broad range of tasks.

- **QQP:** Quora Question Pairs (detect paraphrase questions)
- **QNLI**: natural language inference over question answering data
- **SST-2**: sentiment analysis
- **CoLA**: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B**: semantic textual similarity
- **MRPC**: microsoft paraphrase corpus
- **RTE**: a small natural language inference corpus

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**

Ablation Study: Pre-Training Tasks



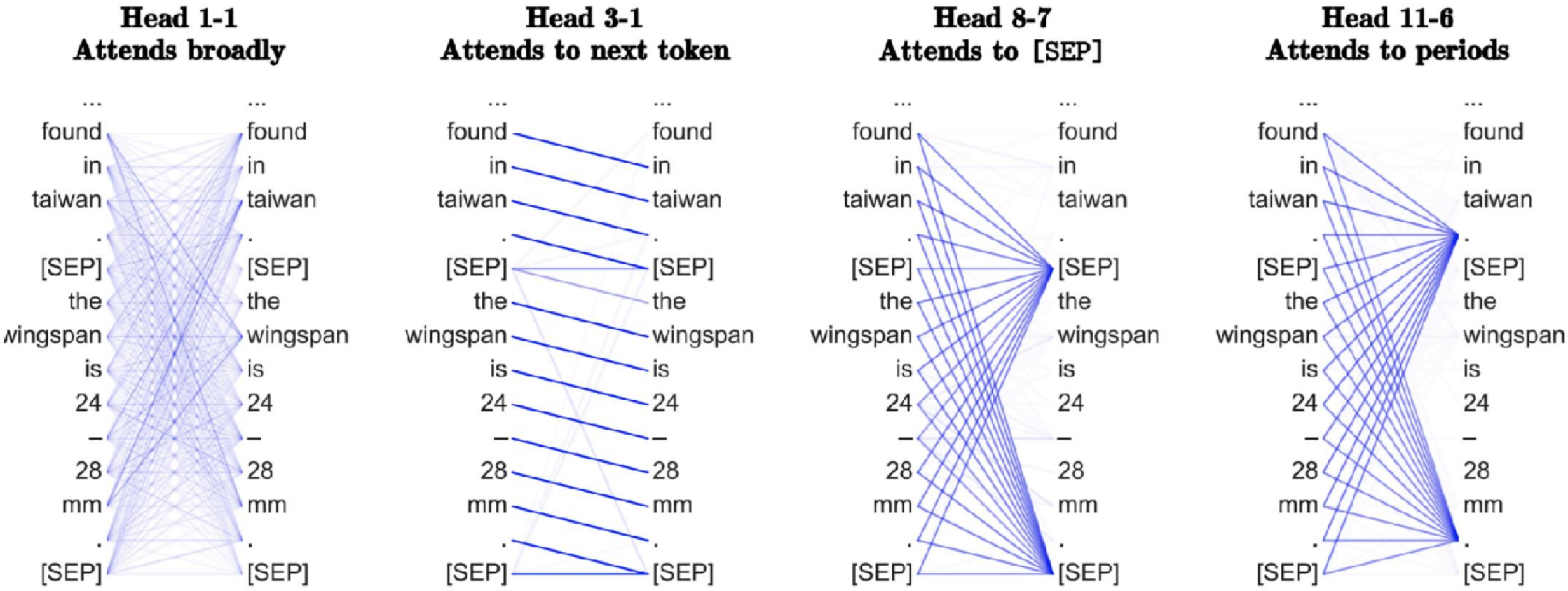
- MLM >> left-to-right LMs
- NSP improves on some tasks
- Note: later work (Joshi et al., 2020; Liu et al., 2019) argued that NSP is not useful

Ablation Study: Model Sizes

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

The bigger, the better!

What does BERT learn?



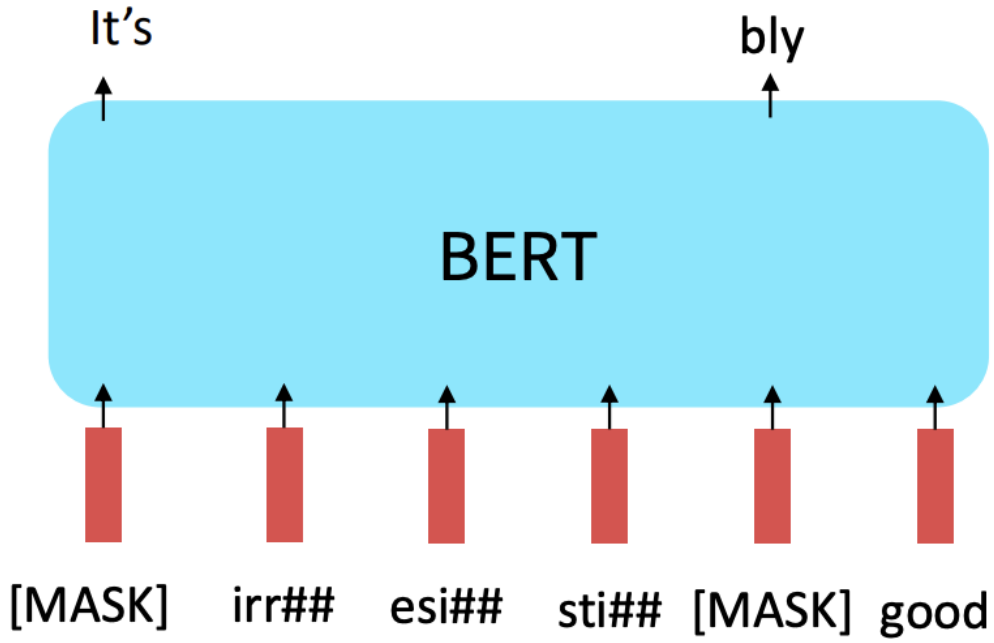
Extensions of BERT – RoBERTa

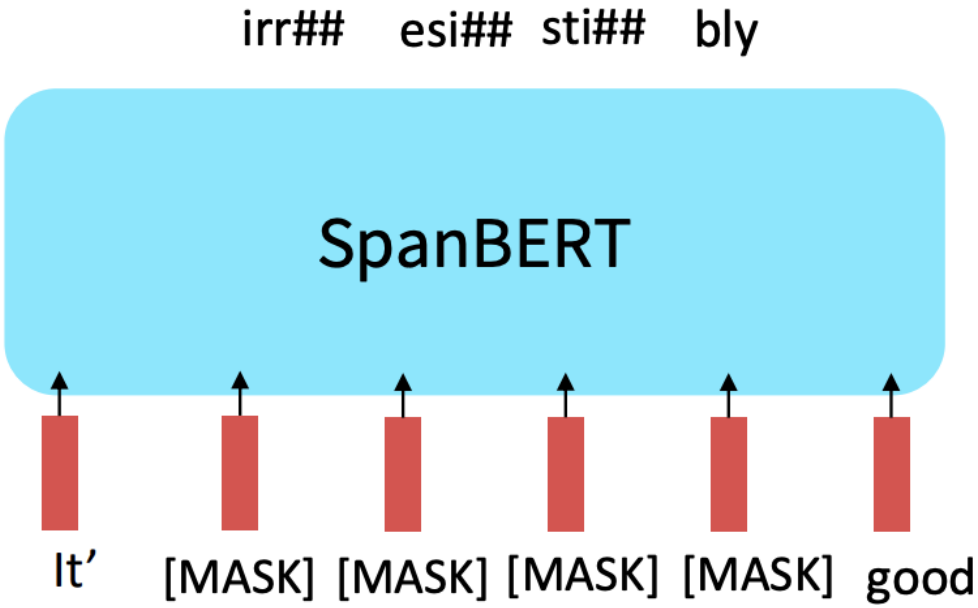
- BERT is still under-trained
- Removed the next sentence prediction pre-training — it adds more noise than benefits!
- Trained longer with 10x data & bigger batch sizes
- Pre-trained on 1,024 V100 GPUs for one day in 2019

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

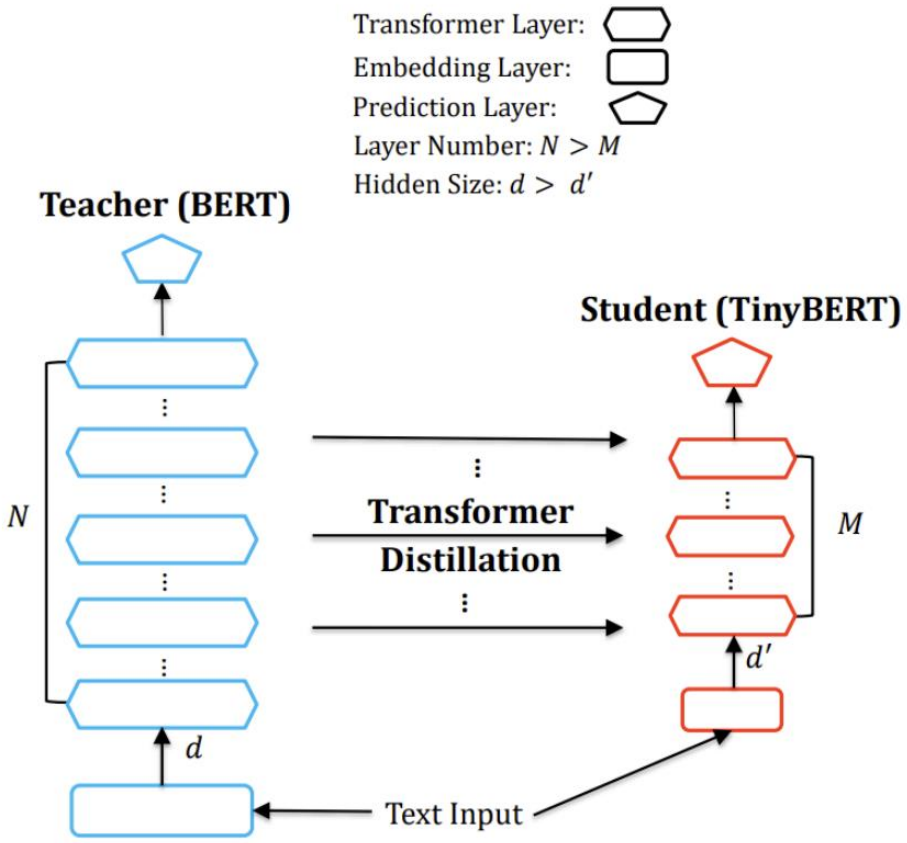
Extensions of BERT – SpanBERT

- SpanBERT: Masking contiguous spans of words makes a harder, more useful pretraining task





Extensions of BERT – DistillBERT



https://github.com/abhilash1910/DistilBERT--SQuAD-v1-Notebook

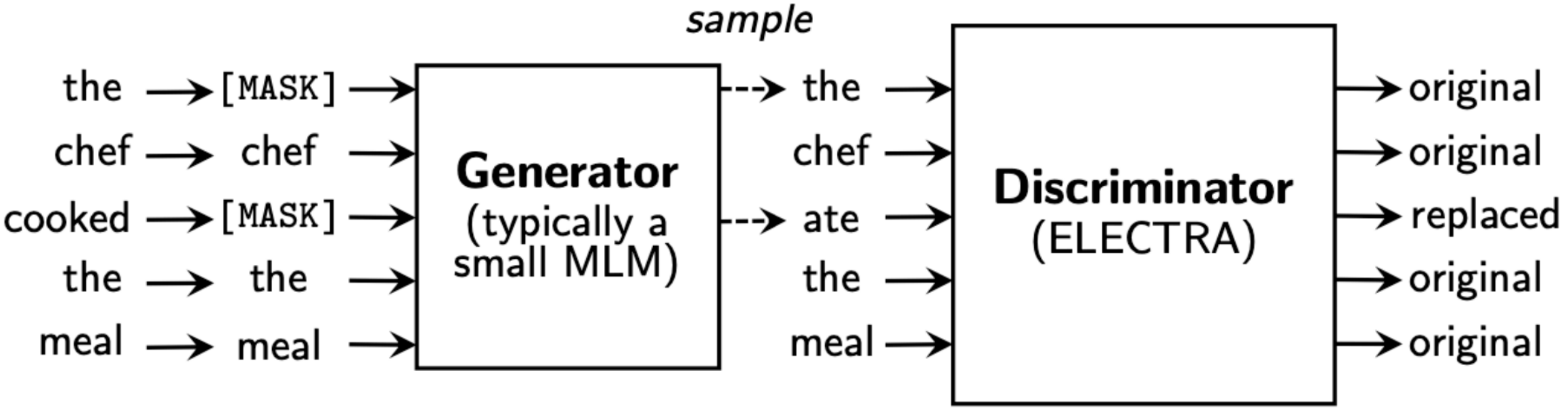
Key idea: produce a smaller model (student) that distill information from the BERT models (teacher)

Table 1: **DistilBERT retains 97% of BERT performance.** Comparison on the dev sets of the GLUE benchmark. ELMo results as reported by the authors. BERT and DistilBERT results are the medians of 5 runs with different seeds.

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

Extensions of BERT – ELECTRA

ELECTRA provides a more **efficient** training method,
because it predicts 100% of tokens (instead of 15%) every time

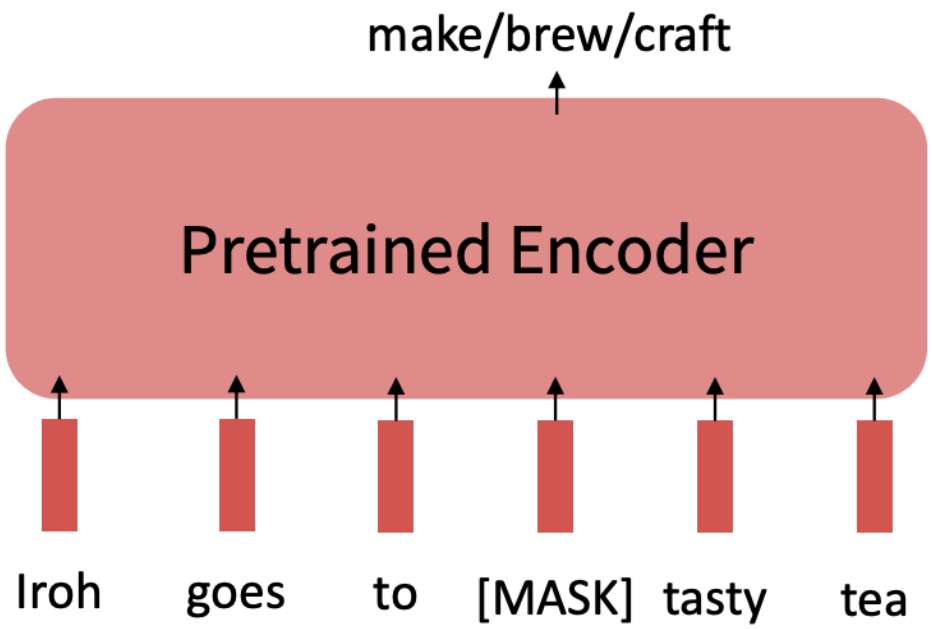


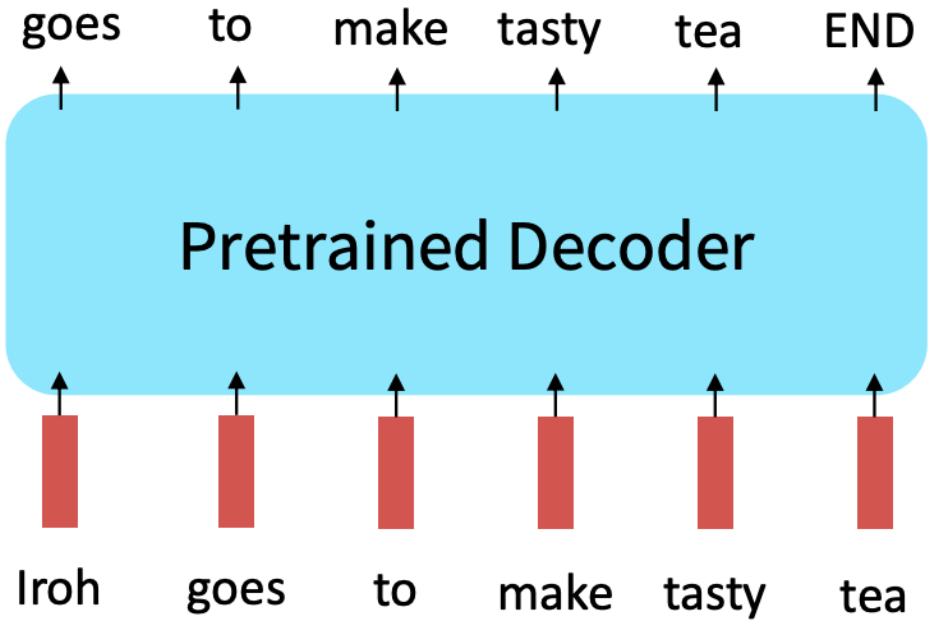
Only the discriminator will be used for downstream fine-tuning

Limitations of Pretrained Encoders

Those results looked great! Why not used pretrained encoders for everything?

If your task involves generating sequences, consider using a pretrained decoder; BERT and other pretrained encoders don't naturally lead to nice autoregressive (1-word-at-a-time) generation methods.

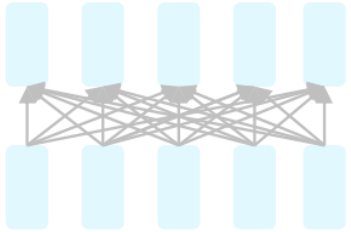




Pre-Training for Encoder-Decoder LM

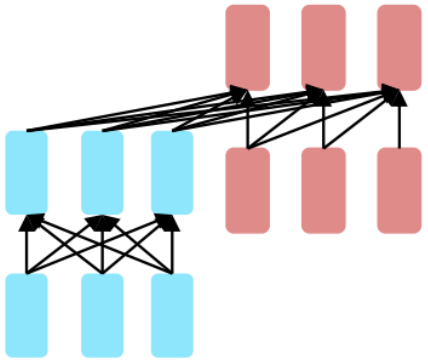
Pre-Training Encoder-Decoder

The neural architecture influences the type of pretraining, and natural use cases.



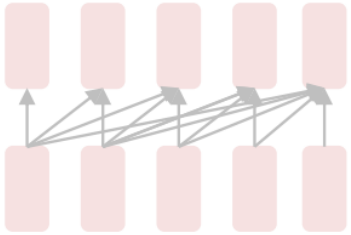
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



Decoders

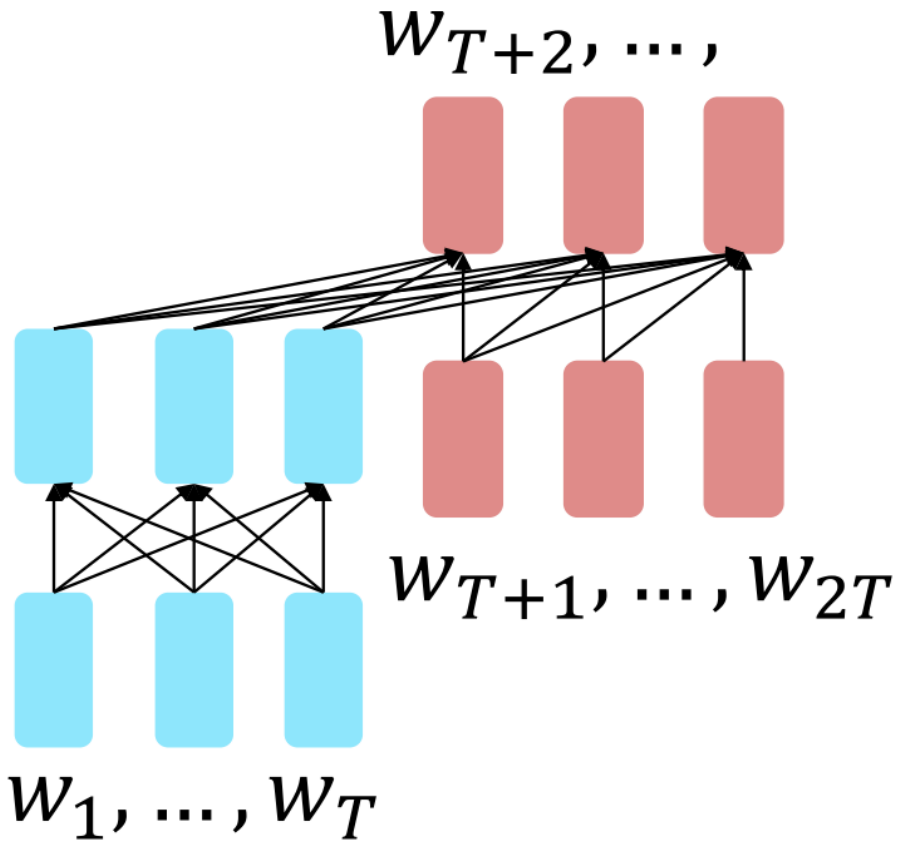
- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

Pre-Training Encoder-Decoder

For **encoder-decoders**, we could do something like **language modeling**, but where a prefix of every input is provided to the encoder and is not predicted.

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$h_{T+1}, \ldots, h_2 = Decoder(w_1, \ldots, w_T, h_1, \ldots, h_T)$$
$$y_i \sim Ah_i + b, i > T$$

The **encoder** portion benefits from bidirectional context; the **decoder** portion is used to train the whole model through language modeling.



Pre-Training T5 – Span Corruption

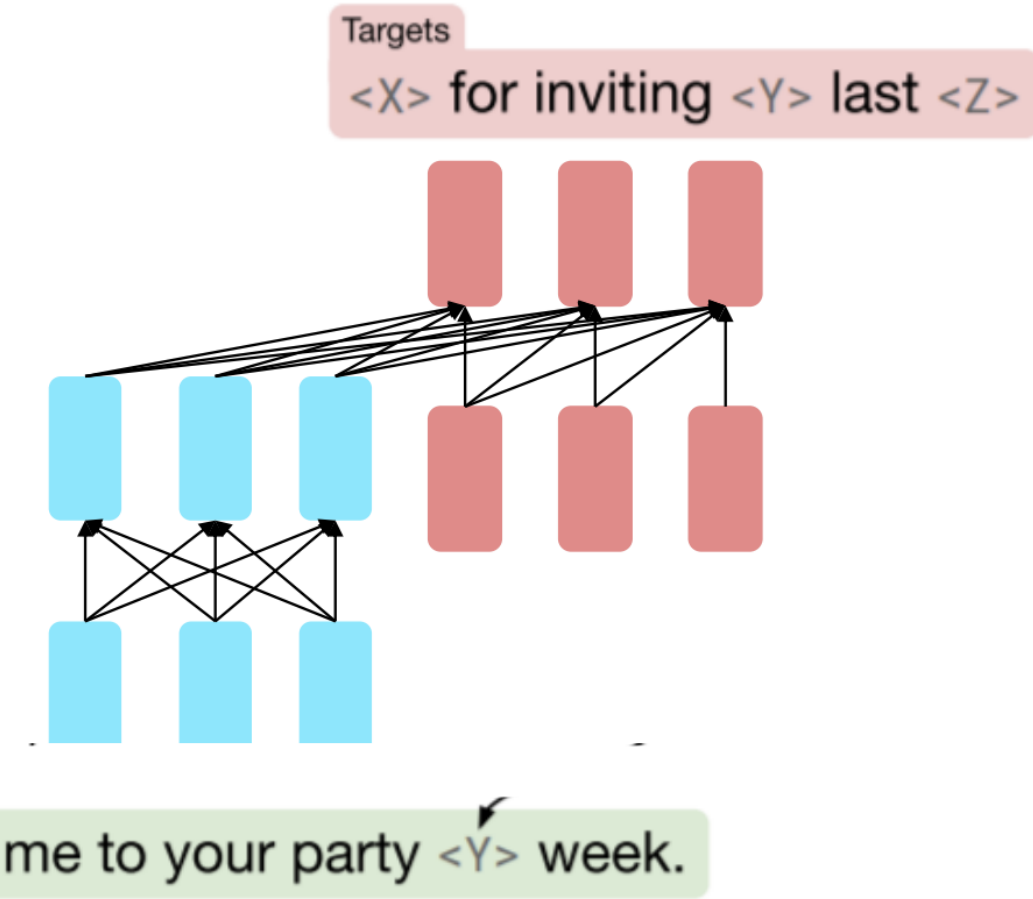
What Raffel et al., 2018 found to work best was **span corruption.** Their model: **T5.**

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!

Original text

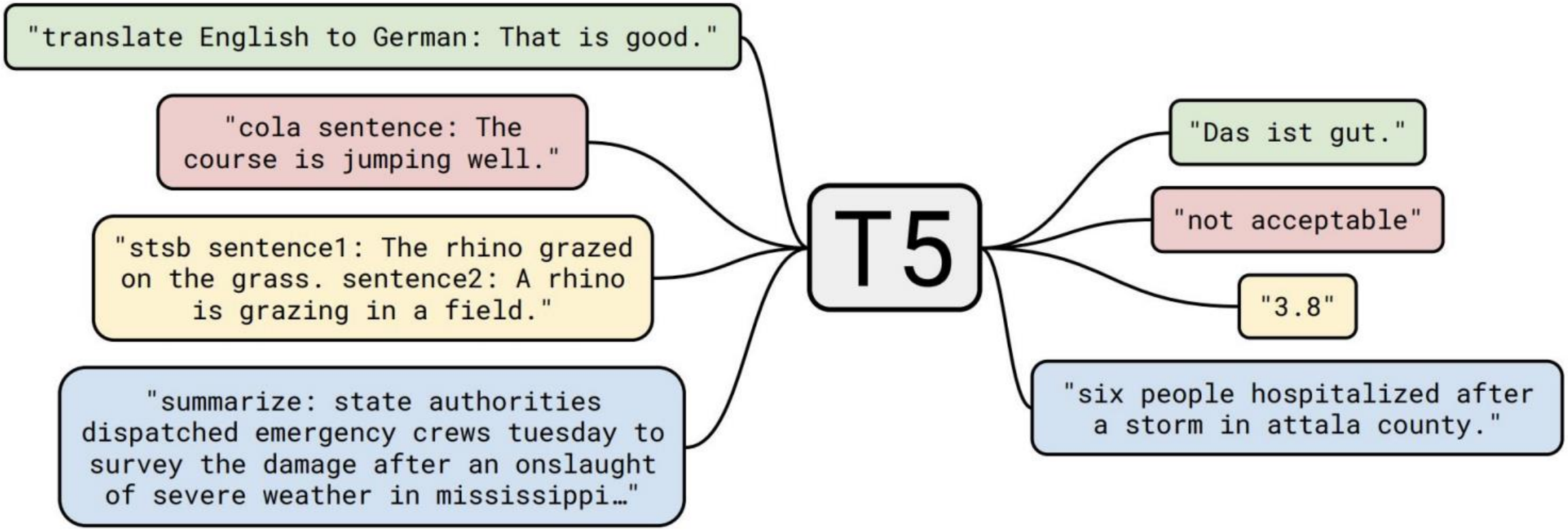
Thank you ~~for inviting~~ me to your party ~~last~~ week.

This is implemented in text preprocessing: it's still an objective that looks like **language modeling** at the decoder side.



Fine-Tuning T5 – Task-specific Prefix

- "[Task-specific prefix]: [Input text]" -> "[output text]"



Fine-Tuning T5 – Task-specific Prefix

[Task-specific prefix]: [Input text]

- EnDe (Translation):
  "translate English to German: That is good" -> "Das ist gut"

- CNNDM (Summarization):
  "summarize: state authorities dispatched…" -> "six people hospitalized after storm"

Fine-Tuning T5 – Task-specific Prefix

[Task-specific prefix]: [Input text]

- CoLA (GLUE; Classification):
  "cola sentence: The course is jumping well." -> "not acceptable"

- STS-B (GLUE; Regression):
  "stsb sentence1: The rhino grazed. sentence2: A rhino is grazing." -> "3.8"

T5 can retrieve knowledge from its parameters

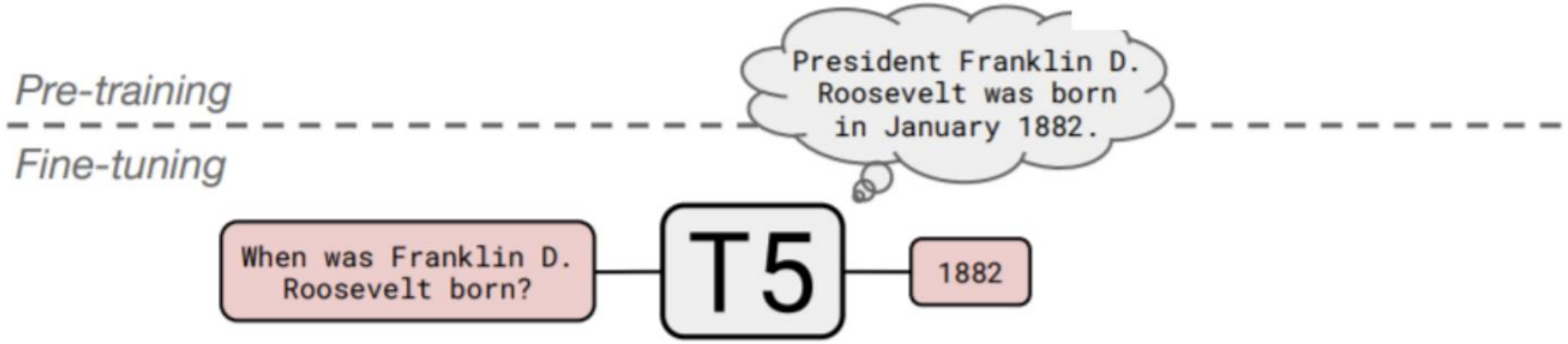
A fascinating property of T5: it can be finetuned to answer a wide range of questions, retrieving knowledge from its parameters.

NQ: Natural Questions

WQ: WebQuestions

TQA: Trivia QA

All “open-domain” versions

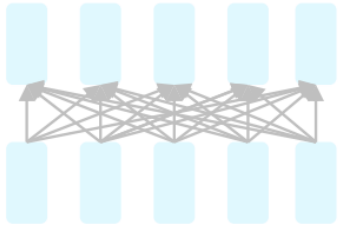


	NQ	WQ	TQA dev	TQA test	
Karpukhin et al. (2020)	**41.5**	42.4	**57.9**	–	
T5.1.1-Base	25.7	28.2	24.2	30.6	**220 million params**
T5.1.1-Large	27.3	29.5	28.5	37.2	**770 million params**
T5.1.1-XL	29.5	32.4	36.0	45.1	**3 billion params**
T5.1.1-XXL	32.8	35.6	42.9	52.5	**11 billion params**
T5.1.1-XXL + SSM	35.2	**42.8**	51.9	**61.6**	

Pre-Training for Decoder-only LM

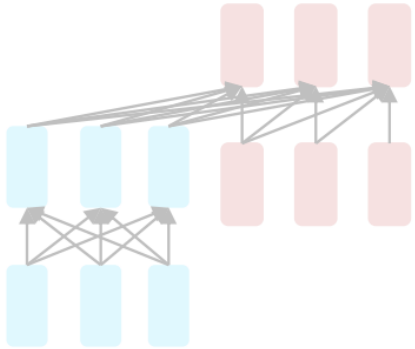
Pre-Training Decoder

The neural architecture influences the type of pretraining, and natural use cases.



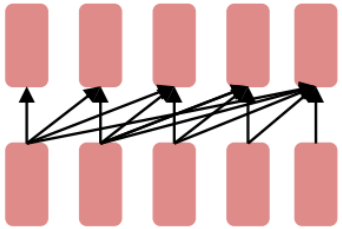
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?

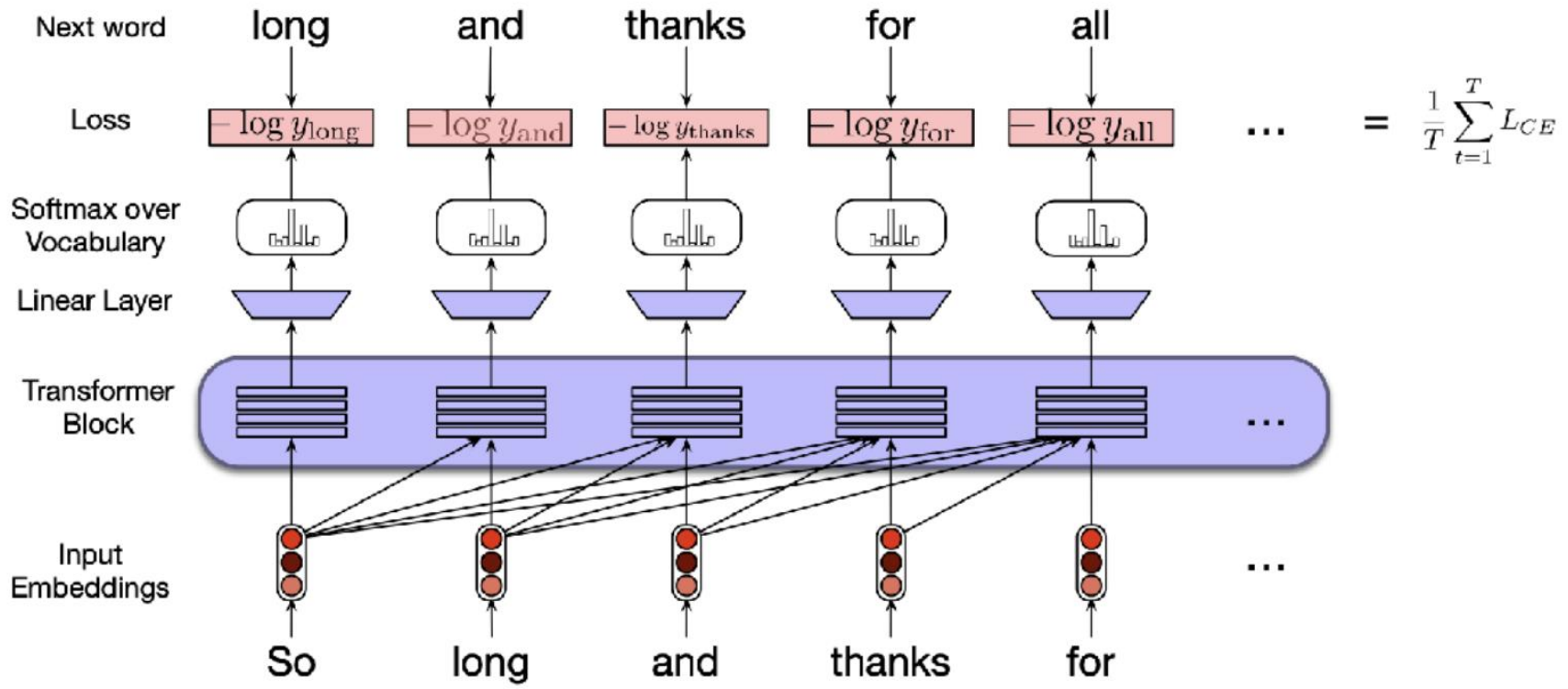


Decoders

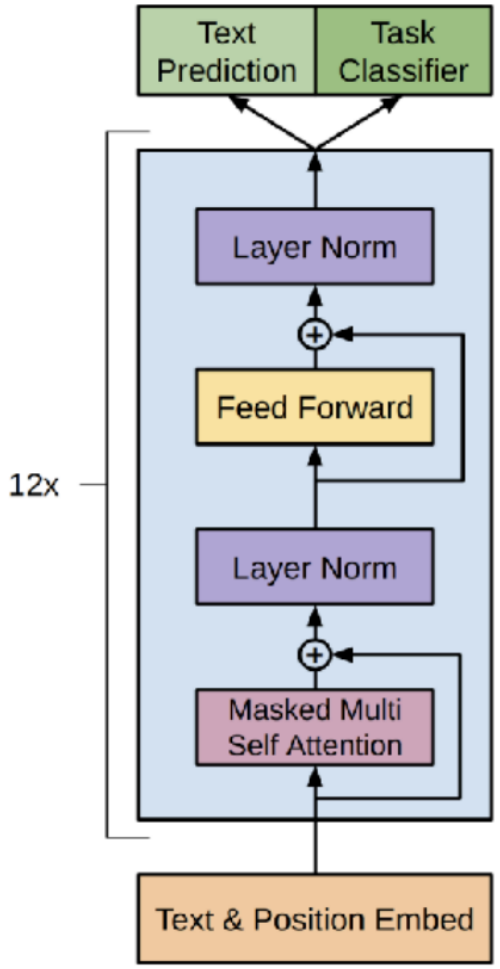
- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words
- All the biggest pretrained models are Decoders.

Pre-Training GPT

- Use a **Transformer decoder** (unidirectional; left-to-right) instead of LSTMs
- Use **language modeling** as a pre-training objective
- Trained on longer segments of text (**512 BPE tokens**), not just single sentences



Pre-Training GPT



- 12 layers, 768 hidden size, 12 attention heads, 110M parameters

Same as BERT-base

Recall: BERT was trained on this + Wikipedia!

- Training corpus: BooksCorpus (0.8B)
- Max sequence size: 512 wordpiece tokens
- Trained for 100 epochs, batch size 64

Fine-Tuning GPT

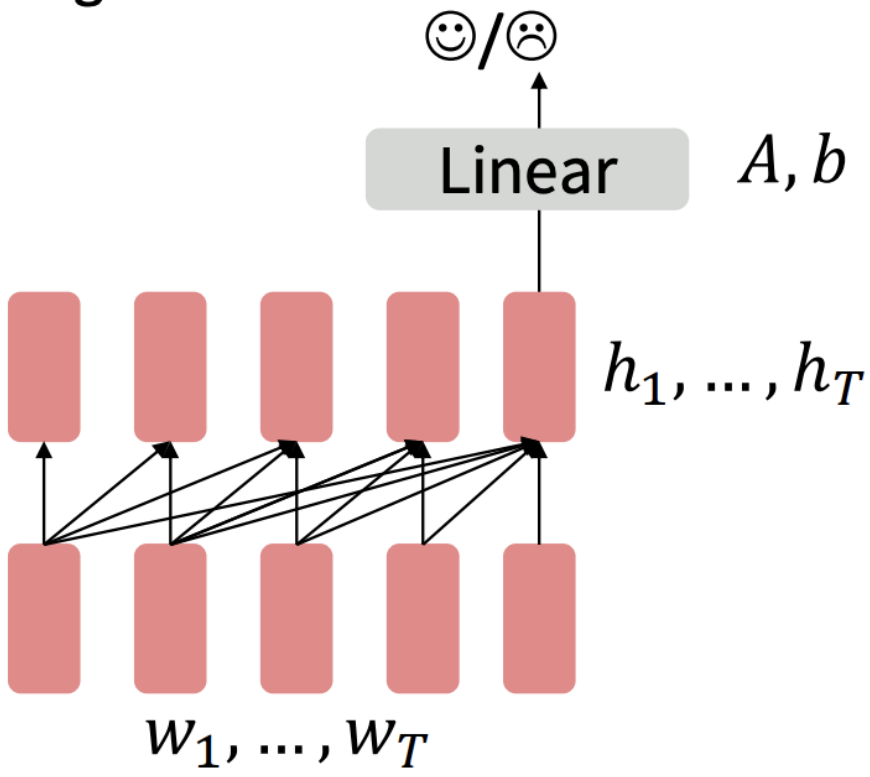
When using language model pretrained decoders, we can ignore that they were trained to model $p(w_t|w_{1:t-1})$.

We can finetune them by training a classifier on the last word's hidden state.

$$h_1, \ldots, h_T = \text{Decoder}(w_1, \ldots, w_T)$$
$$y \sim Ah_T + b$$

Where A and b are randomly initialized and specified by the downstream task.

Gradients backpropagate through the whole network.



[Note how the linear layer hasn't been pretrained and must be learned from scratch.]

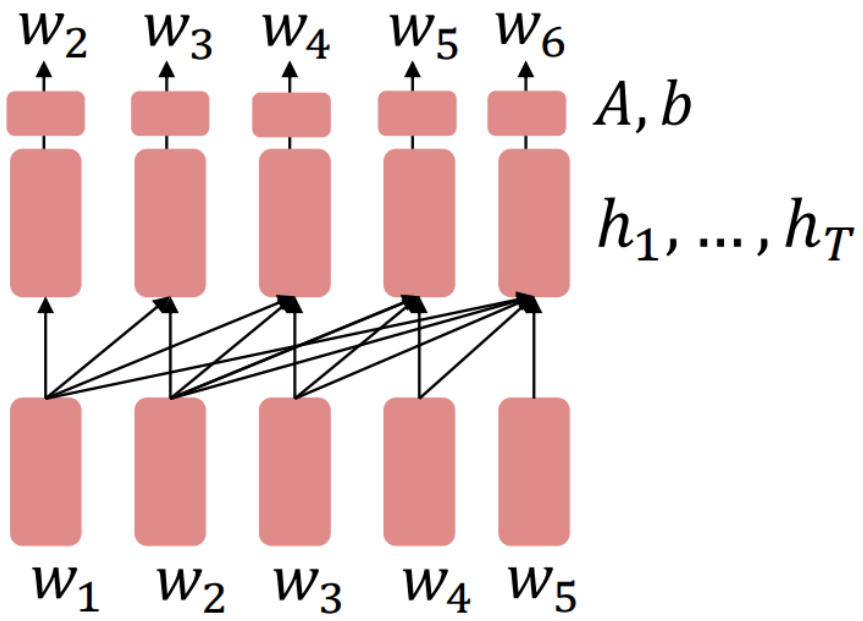
Fine-Tuning GPT

It's natural to pretrain decoders as language models and then use them as generators, finetuning their $p_\theta(w_t|w_{1:t-1})$!

This is helpful in tasks **where the output is a sequence** with a vocabulary like that at pretraining time!

- Dialogue (context=dialogue history)
- Summarization (context=document)

$$h_1, \ldots, h_T = \text{Decoder}(w_1, \ldots, w_T)$$
$$w_t \sim Ah_{t-1} + b$$

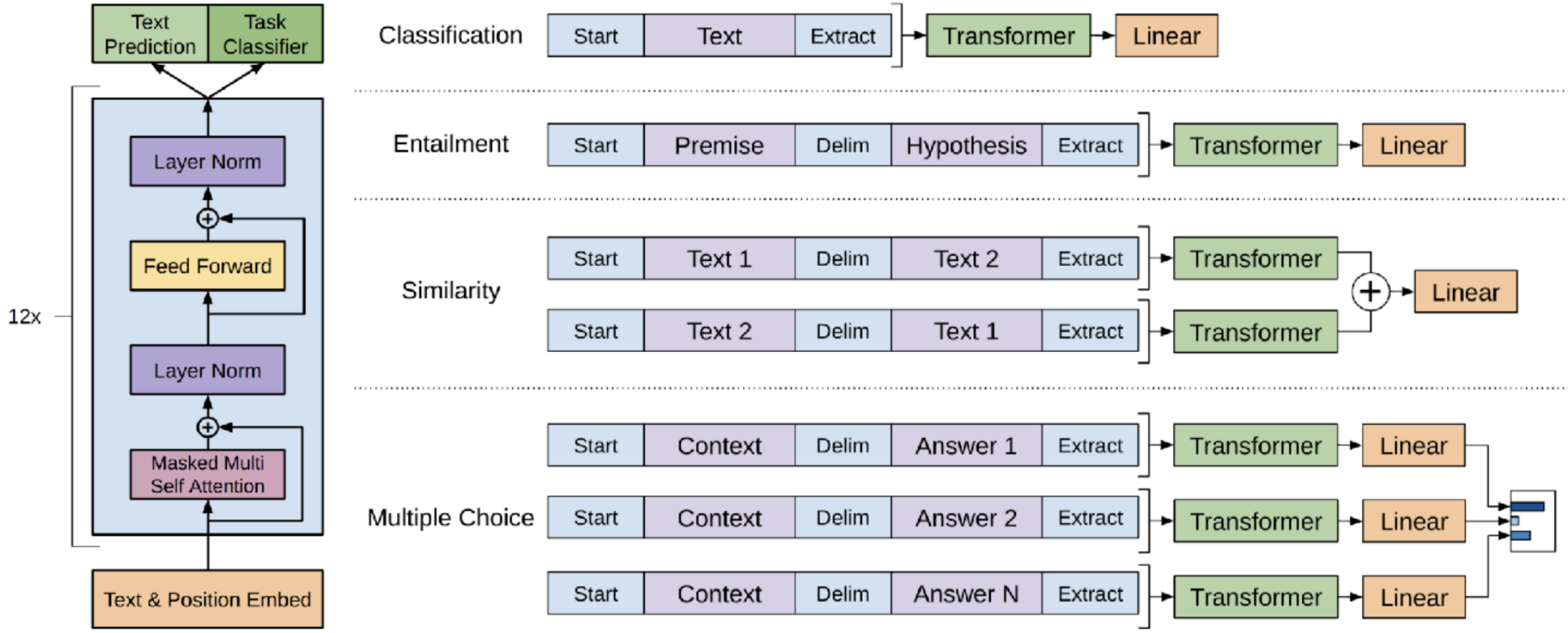


[Note how the linear layer has been pretrained.]

Where A, b were pretrained in the language model!

Fine-Tuning GPT

- "Fine-tune" the entire set of model parameters on various downstream tasks

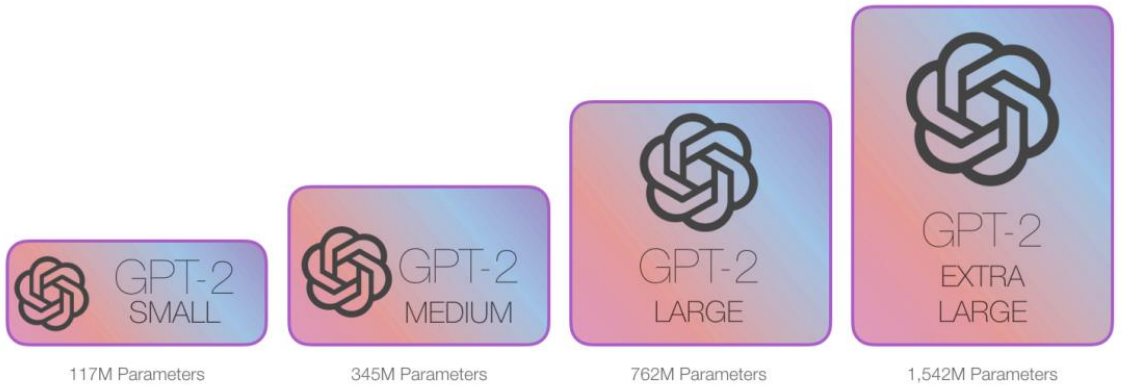


From GPT to GPT-2 to GPT-3 …

- All **decoder-only Transformer-based language models**
- Model size ↑, training corpora ↑

GPT-2

Context size = 1024



.. trained on 40Gb of Internet text ..

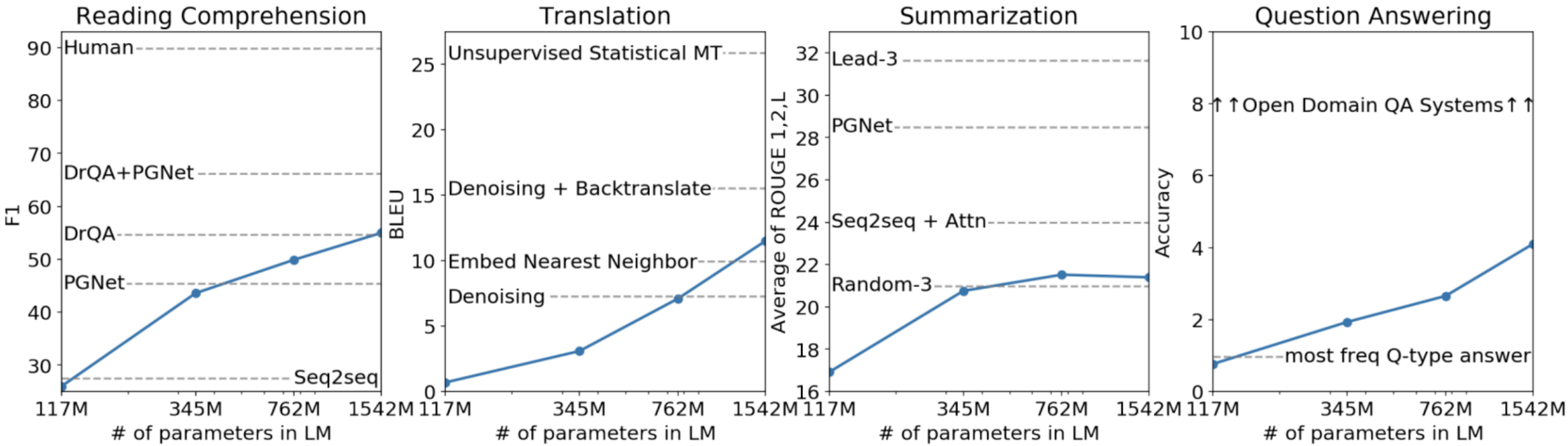
GPT-2: Increasingly convincing generations

We mentioned how pretrained decoders can be used **in their capacities as language models.**

GPT-2, a larger version (1.5B) of GPT trained on more data, was shown to produce relatively convincing samples of natural language.

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.
GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

GPT-2: Start to achieve strong zero-shot performance



GPT-3: In-Context Learning (Few-shot Learning)

- Before GPT-3, **fine-tuning** is the default way of doing learning in models like BERT/T5/GPT-2
  - SST-2 has 67k examples, SQuAD has 88k (passage, answer, question) triples

- Fine-tuning requires computing the gradient and applying a parameter update on every example (or every K examples in a mini-batch)

- However, this is very expensive for the 175B GPT-3 model

GPT-3: In-Context Learning (Few-shot Learning)

- GPT-3 proposes an alternative: **in-context learning**

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

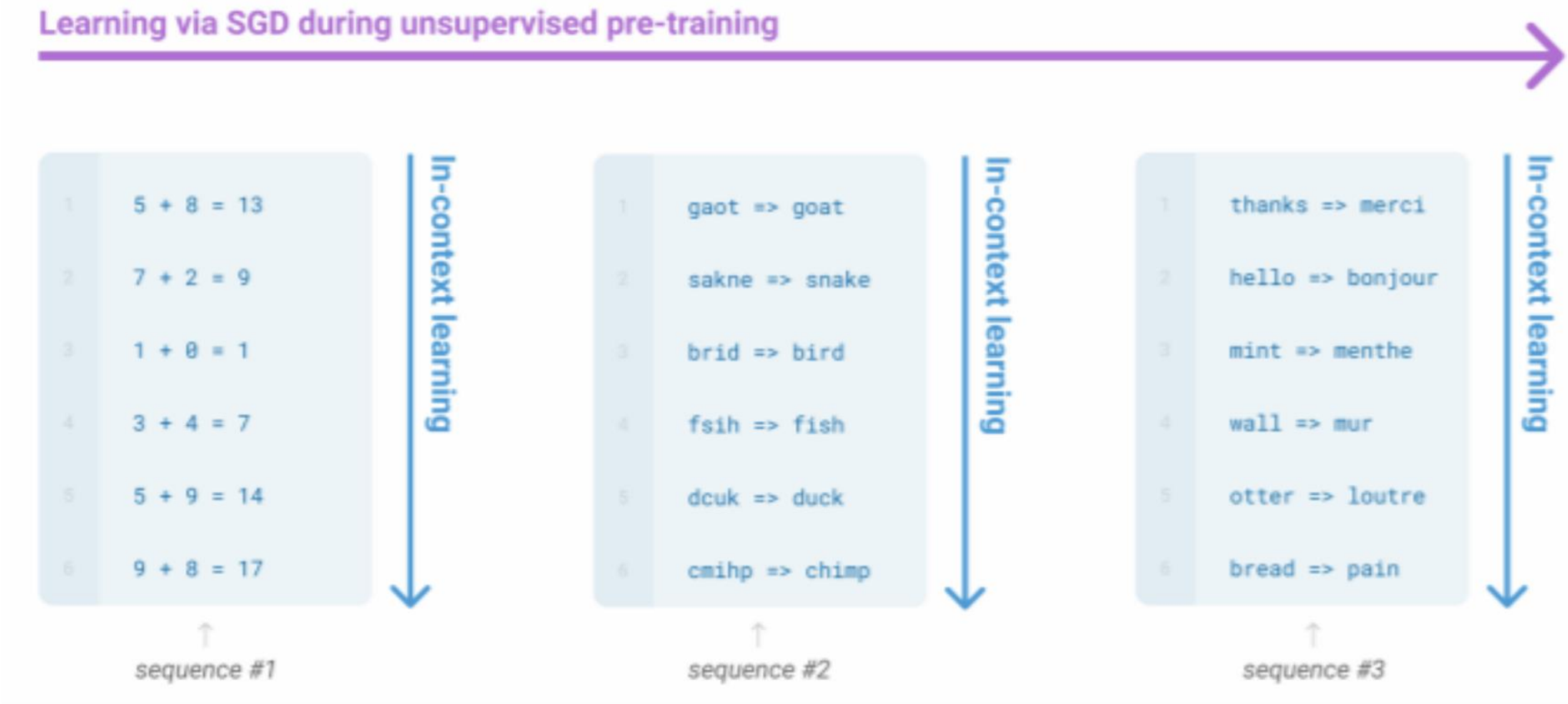
```
1  Translate English to French:                    ← task description
2  sea otter => loutre de mer                      ← examples
3  peppermint => menthe poivrée
4  plush girafe => girafe peluche
5  cheese =>                                       ← prompt
```

- This is just a forward pass, **no gradient update at all**!
- You only need to feed a small number of examples (e.g., 32)

(On the other hand, you can't feed many examples at once too as it is bounded by context size)

GPT-3: In-Context Learning (Few-shot Learning)

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.



GPT-3: In-Context Learning (Task Specifications)

Context →	Passage: Saint Jean de Brébeuf was a French Jesuit missionary who travelled to New France in 1625. There he worked primarily with the Huron for the rest of his life, except for a few years in France from 1629 to 1633. He learned their language and culture, writing extensively about each to aid other missionaries. In 1649, Brébeuf and another missionary were captured when an Iroquois raid took over a Huron village . Together with Huron captives, the missionaries were ritually tortured and killed on March 16, 1649. Brébeuf was beatified in 1925 and among eight Jesuit missionaries canonized as saints in the Roman Catholic Church in 1930. Question: How many years did Saint Jean de Brébeuf stay in New France before he went back to France for a few years? Answer:
Target Completion →	4

DROP
(a reading comprehension task)

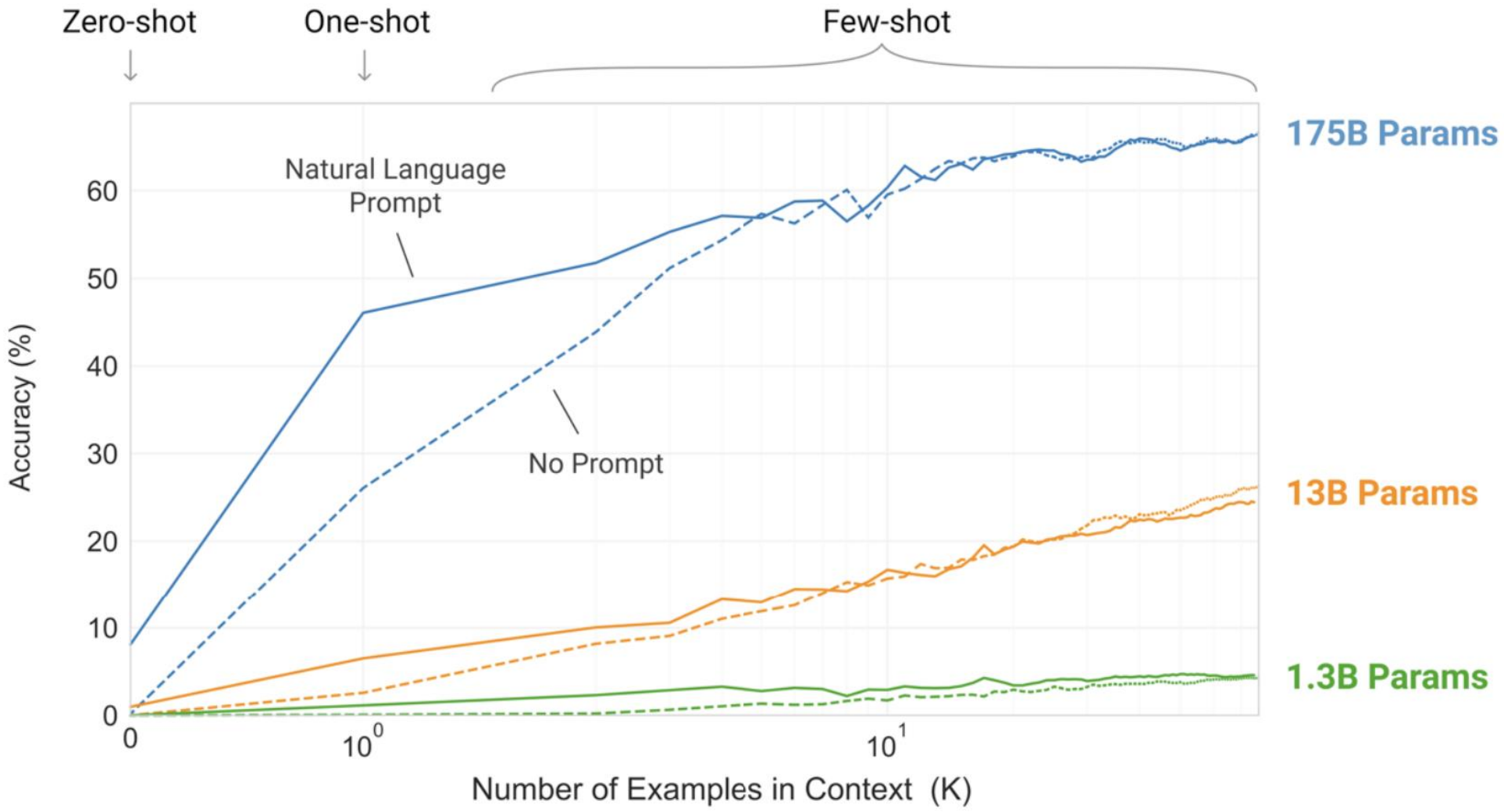
Context →	Please unscramble the letters into a word, and write that word: skicts =
Target Completion →	sticks

Unscrambling words

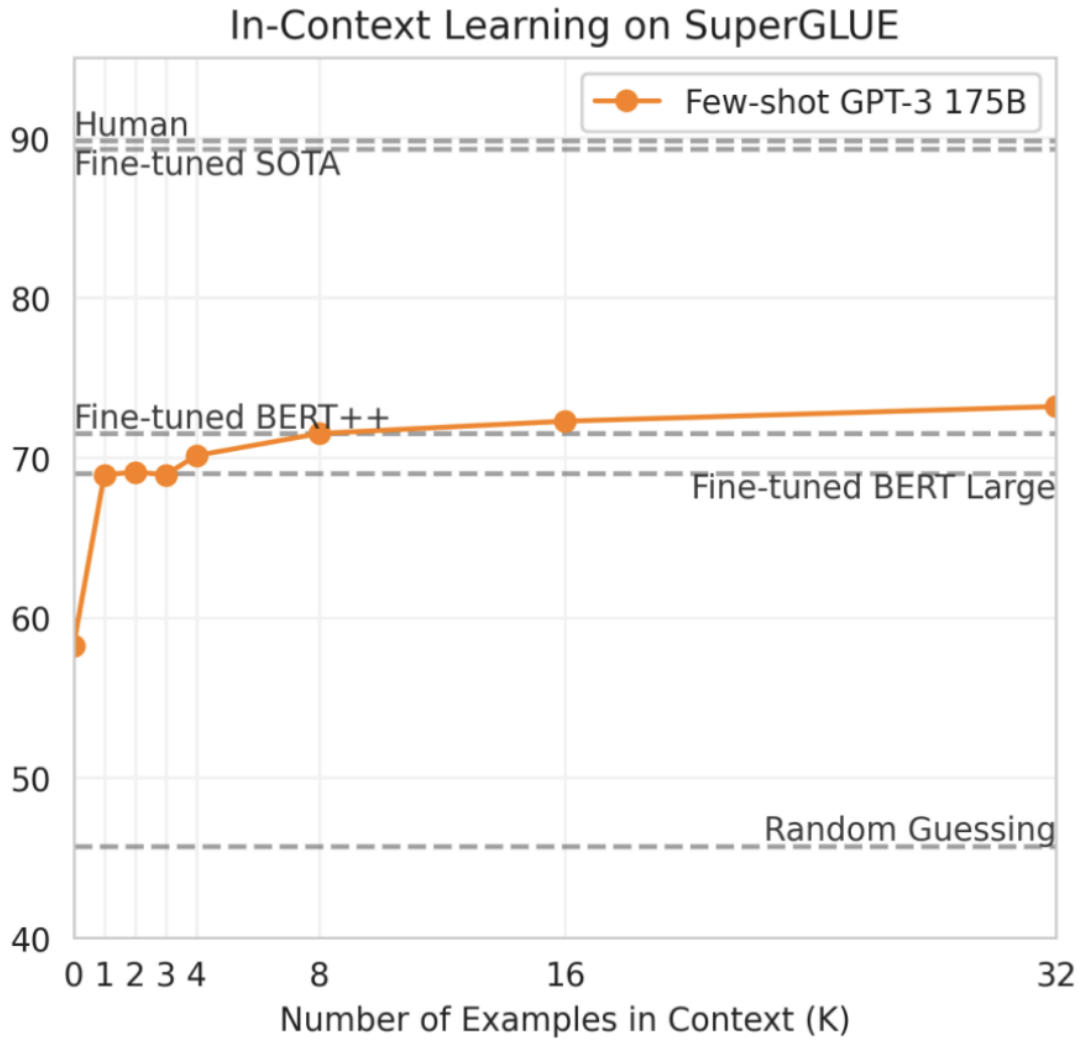
Context →	An outfitter provided everything needed for the safari. Before his first walking holiday, he went to a specialist outfitter to buy some boots. question: Is the word 'outfitter' used in the same way in the two sentences above? answer:
Target Completion →	no

Word in context (WiC)

GPT-3: In-Context Learning



GPT-3: In-Context Learning



GPT-3: In-Context Learning

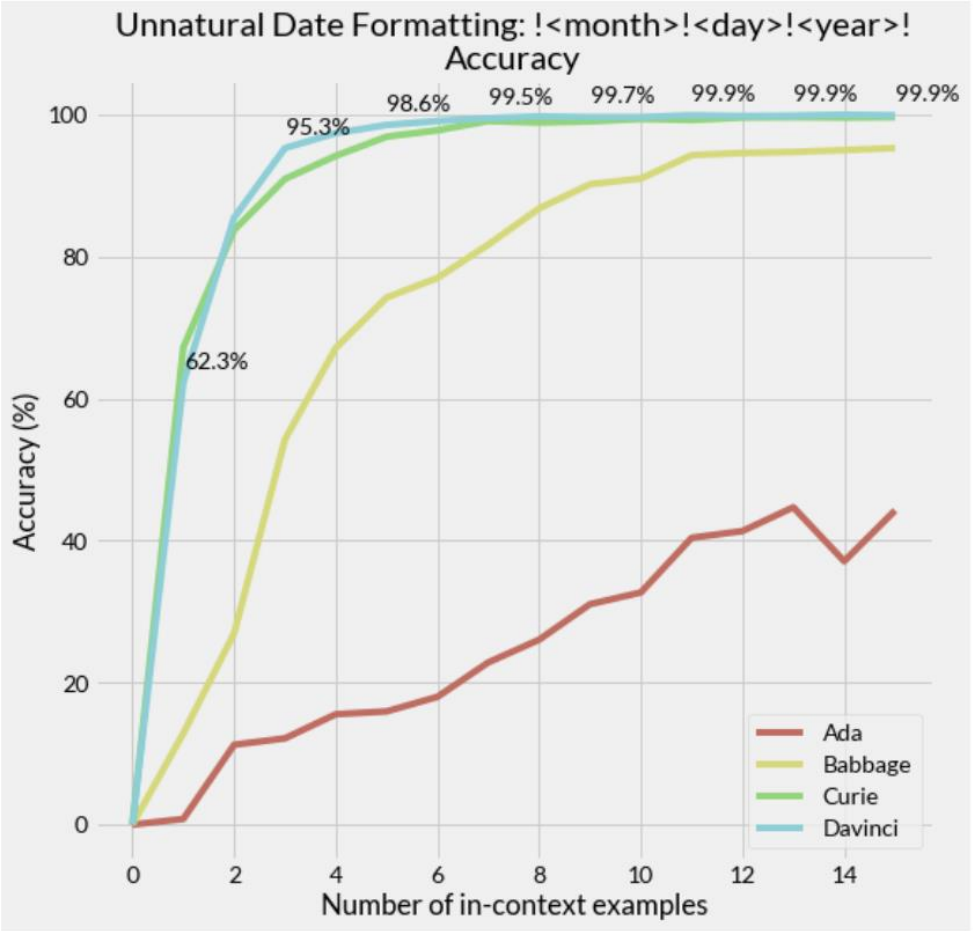
```
Input: 2014-06-01
Output: !06!01!2014!
Input: 2007-12-13
Output: !12!13!2007!
Input: 2010-09-23
Output: !09!23!2010!
Input: 2005-07-23
Output: !07!23!2005!
```

in-context examples

test example

model completion

http://ai.stanford.edu/blog/in-context-learning/



GPT-3: Chain-of-Thought Prompting

Standard Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain of Thought Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

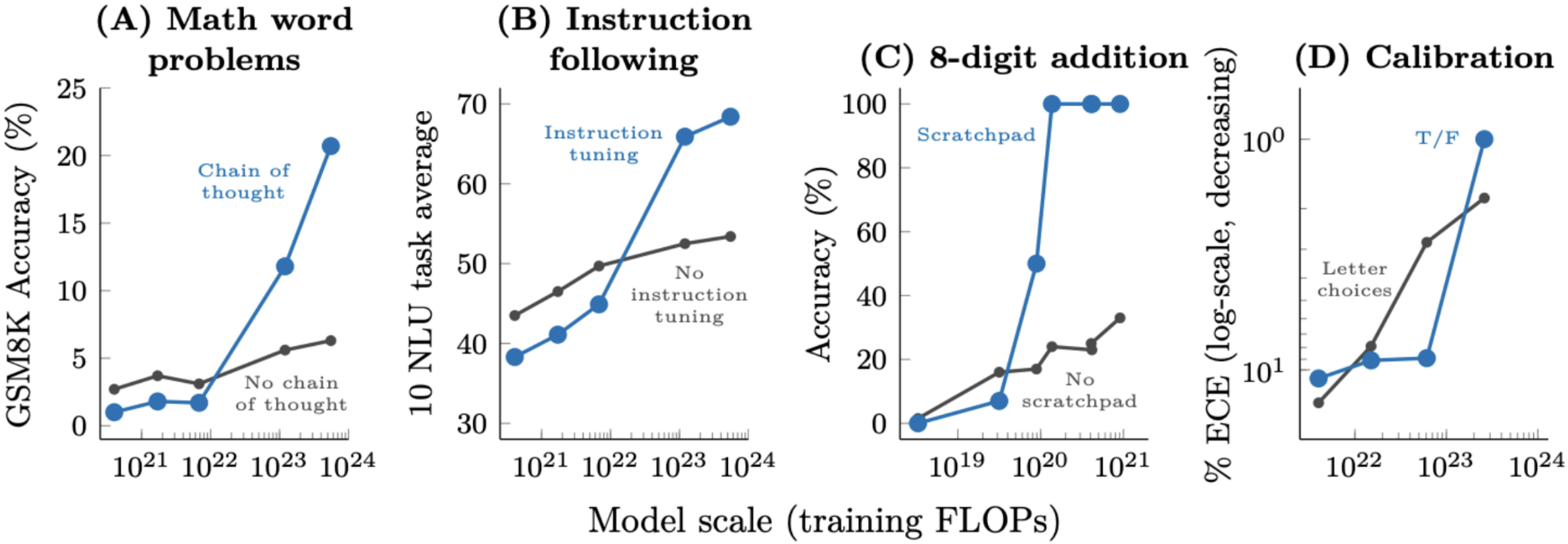
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

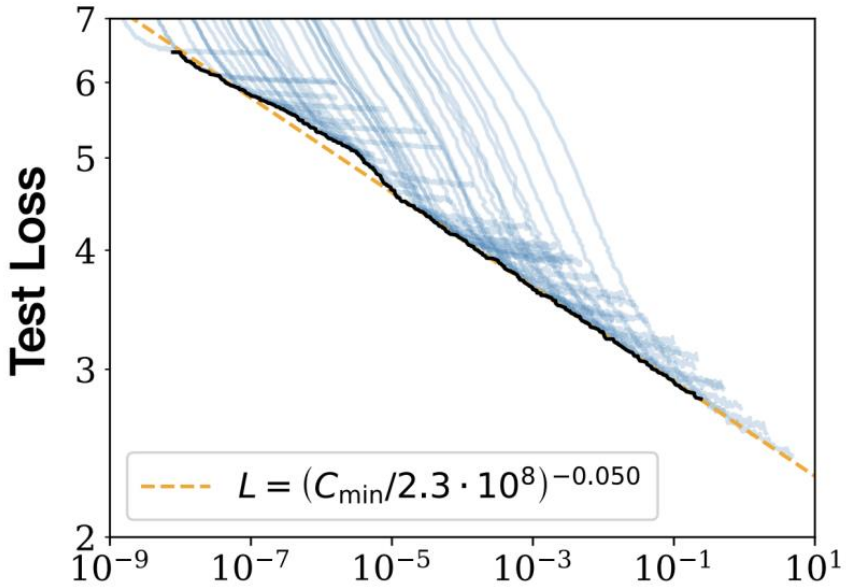
Model Output

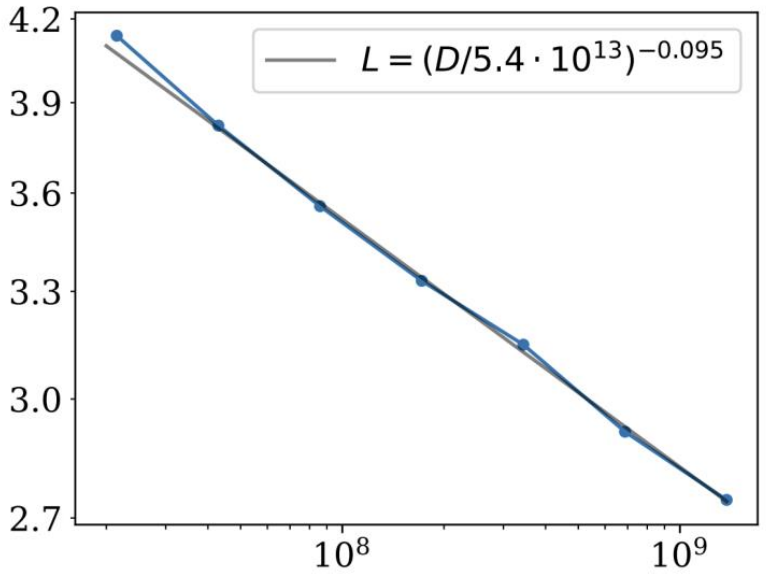
A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

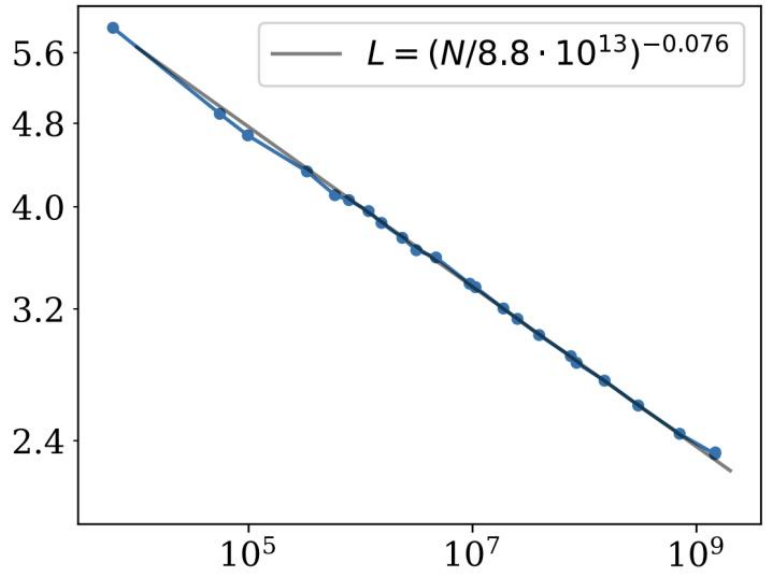
Emergent Properties of LLMs



Scaling Law of LLMs







$\text{Loss} \propto (\text{Compute})^{-\alpha}$ $\text{Loss} \propto (\text{Data})^{-\beta}$ $\text{Loss} \propto (\text{Model params})^{-\gamma}$

Loss goes down predictably wrt compute, data, model size!

Chinchilla Scaling Laws: Model Size vs Dataset Size

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}.$$

L: loss
N: number of params
D: dataset size
E, A, B, α, β: fit based on data

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

Rule of thumb: Increase dataset size proportional to model size
(e.g. 20 token per param)

Open-Weight LLM: LLaMA

RESEARCH

Introducing LLaMA: A foundational, 65-billion-parameter large language model

February 24, 2023

- **Smaller models** trained on **1.4T**, high-quality & publicly available data
- "LLaMA-13B outperforms GPT-3 (175B) on most benchmarks, and LLaMA-65B is competitive with the best models, Chinchilla-70B and PaLM-540B"

Recent LLMs are trained much longer

- Llama-3: 8B, 70B, 405B trained on **15T** tokens
- Qwen-2.5: 0.5B, 1.5B, 3B, 7B, 14B, 32B, 72B trained on **18T** tokens
- DeepSeek V3: 671B (37B active) trained on **14.8T** tokens

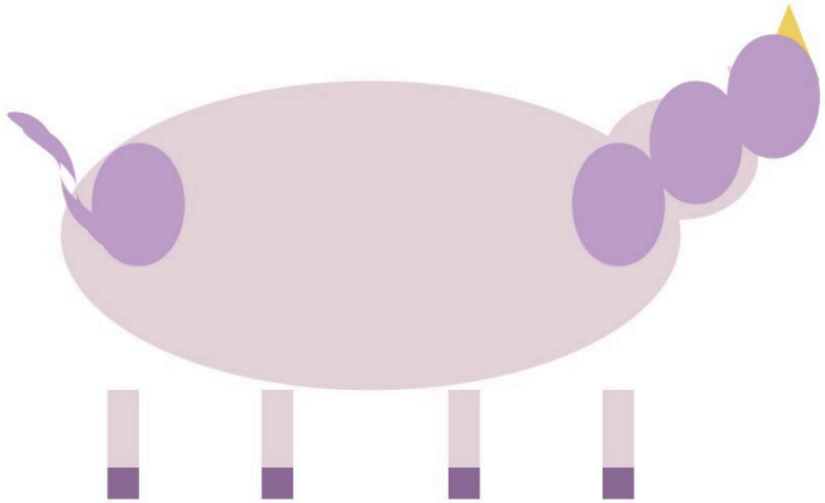
Optimize for Inference: “over-trained” smaller models are faster during inference

GPT-4: Multi-Modal Capabilities

- GPT-4: a **multi-modal** model capable of processing image and text inputs and producing text outputs.
- Model size and training details unknown
- Can process up to 32k context size

Prompt: Draw a unicorn in TiKZ.

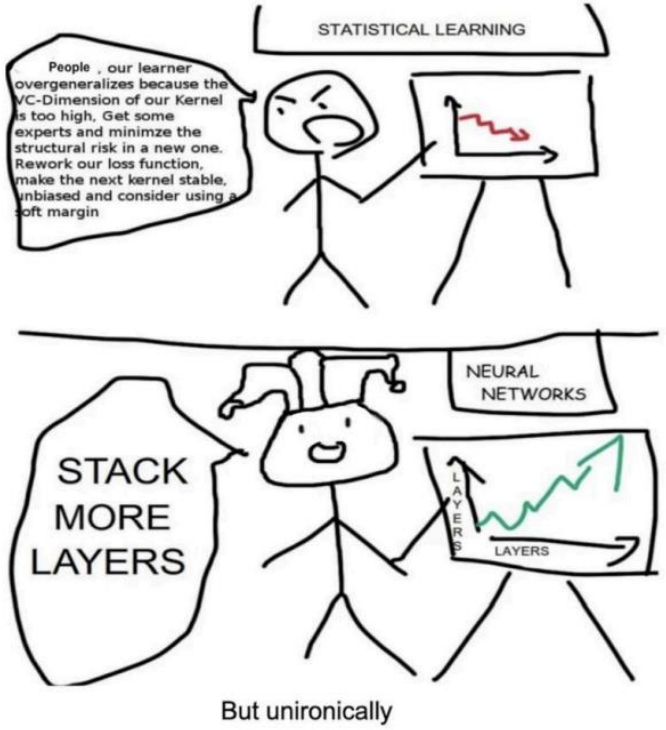
GPT-4: [Produces LaTeX compiling to following picture.]



GPT-4: Multi-Modal Capabilities

GPT-4 visual input example, Moar Layers:

User: Can you explain why this is funny. Think about it step-by-step.



GPT-4: The comic is satirizing the difference in approaches to improving model performance between statistical learning and neural networks.

In statistical learning, the character is shown to be concerned with overfitting and suggests a series of complex and technical solutions, such as minimizing structural risk, reworking the loss function, and using a soft margin.

In contrast, the neural networks character simply suggests adding more layers to the model. This is often seen as a common solution to improving

GPT-4 visual input example, Extreme Ironing:

User: What is unusual about this image?

Source: https://www.barnorama.com/wp-content/uploads/2016/12/03-Confusing-Pictures.jpg

GPT-4: The unusual thing about this image is that a man is ironing clothes on an ironing board attached to the roof of a moving taxi.


Thank you!
UF
Herbert Wertheim
College of Engineering
UNIVERSITY of FLORIDA
LEADING THE CHARGE, CHARGING AHEAD