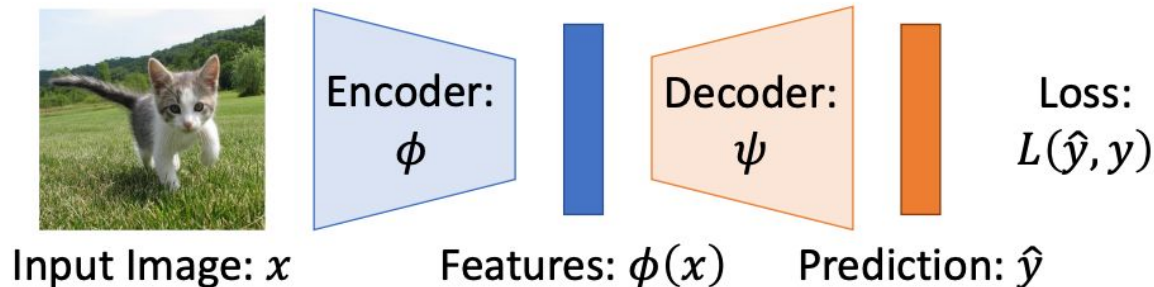


Lecture 14:

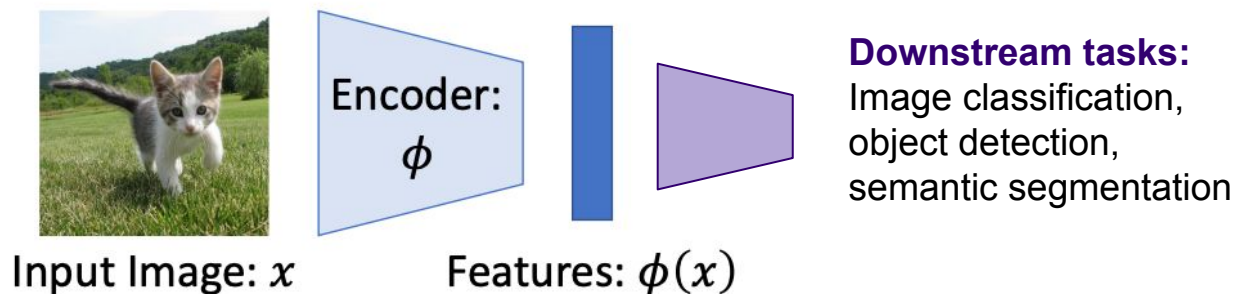
Large Language Models

Last time: Self-Supervised Learning Pretext then Transfer

Step 1: Pretrain a network on a pretext task that doesn't require supervision



Step 2: Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning



Recap: Self-Supervised

Get labels for free!

- What we want are good representations, which we can use for downstream tasks.
- The more data, the better → but labeled data is expensive and limited
- So people started using unlabeled data, which let them scale, which improved representations.

Recap: Self-Supervised

Skills needed to classify dogs

- Identify different textures
- Identify different colors
- Identify which pixels are parts of an object
- Understand parts of objects which make up the whole
- Understand the context of the object/animal in the image
- Understand lighting conditions
- Understand objects close up/far away
- Have common-sense reasoning skills
- Learn which features are associated with which dog

Recap: Self-Supervised

General visual
modelling skills
(Can be learned with
Self Supervised)

Specific (Supervised)

- Identify different textures
- Identify different colors
- Identify which pixels are parts of an object
- Understand parts of objects which make up the whole
- Understand the context of the object/animal in the image
- Understand lighting conditions
- Understand objects close up/far away
- Have common-sense reasoning skills
- Learn which features are associated with which dog

Self-Supervised for Language

Skills needed to classify book genres

Knowledge of words/letters

Knowledge of grammar

Meanings of words

Understanding context of words

Keeping track of entities over time

Understanding expressions/idioms

Understanding tone

Learning which features are associated with each genre

Self-Supervised for Language

General language
modelling skills
(Can be learned with
Self Supervised)

Specific (Supervised)

Knowledge of words/letters

Knowledge of grammar

Meanings of words

Understanding context of words

Keeping track of entities over time

Understanding expressions/idioms

Understanding tone

Learning which features are associated with each genre

Self-Supervised for Language

General language
modelling skills
(Can be learned with
Self Supervised)

Knowledge of words/letters
Knowledge of grammar
Meanings of words
Understanding context of words
Keeping track of entities over time
Understanding expressions/idioms
Understanding tone

Specific (Supervised) | Learning which features are associated with each genre

Want: model with general understanding of language (Language model!)

LLMs

Building LLMs: Pre-training objectives + architectures

- Encoder only
- Decoder only
- Encoder Decoder

Improving Models without Finetuning

Improving Models with Finetuning

LLMs

Building LLMs: Pre-training objectives + architectures

- **Encoder only**
- **Decoder only**
- **Encoder Decoder**

Improving Models without Finetuning

Improving Models with Finetuning

Last time

Pre-training tasks

Pretext...

then Transfer!



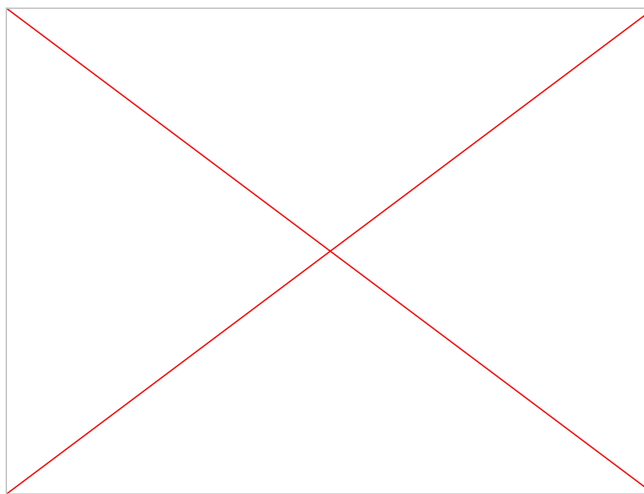
Last time

Pre-training tasks

Pretext...



then Transfer!



Last time

Pre-training tasks

rotation



[Gidaris et al. 2018](#)

in-painting



[Pathak et al., 2016](#)

colorization



Source: [Google AI blog post](#)

1. Solving the pretext tasks allow the model to learn good features.
2. We can automatically generate labels for the pretext tasks.

Not too easy,
not too hard!

Last time

Pre-training tasks

Common recipe in vision:

1. Take original data
2. Remove/obscure information
3. Ask model to get back to the original instance

colorization



Source: [Google AI blog post](#)

rotation



[Gidaris et al. 2018](#)

in-painting



[Pathak et al., 2016](#)

Last time

Pre-training tasks

rotation



90° rotation

in-painting



colorization



What to use as pre-training task for language?

Pre-training task for language

It's cold today! Don't forget to wear a jacket.

The Space Needle is a popular tourist attraction in Seattle.

I missed the bus.

I had 3 pencils and lost one so now I have 2 pencils.

Pre-training task for language

It's cold today! Don't forget to wear a _____.

The _____ is a popular tourist attraction in Seattle.

I missed _____ bus.

I had 3 pencils and lost one so now I have _____ pencils.

Pre-training task for language

It's cold today! Don't forget to wear a jacket.

The Space Needle is a popular tourist attraction in Seattle.

I missed the bus.

I had 3 pencils and lost one so now I have 2 pencils.

Pre-training task for language

It's cold today! Don't forget to wear a jacket.

Common Sense

The Space Needle is a popular tourist attraction in Seattle.

Factual knowledge

I missed the bus.

Grammar

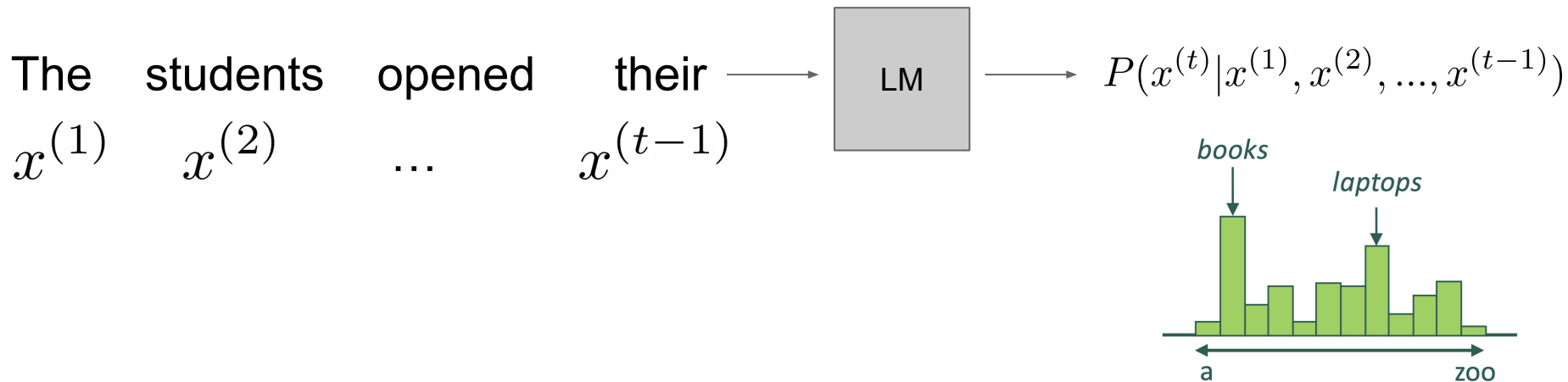
I had 3 pencils and lost one so now I have 2 pencils.

Math/ Reasoning

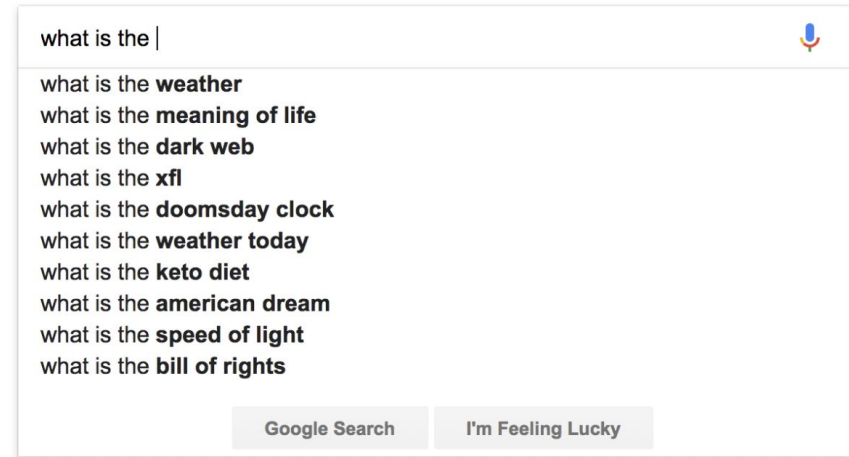
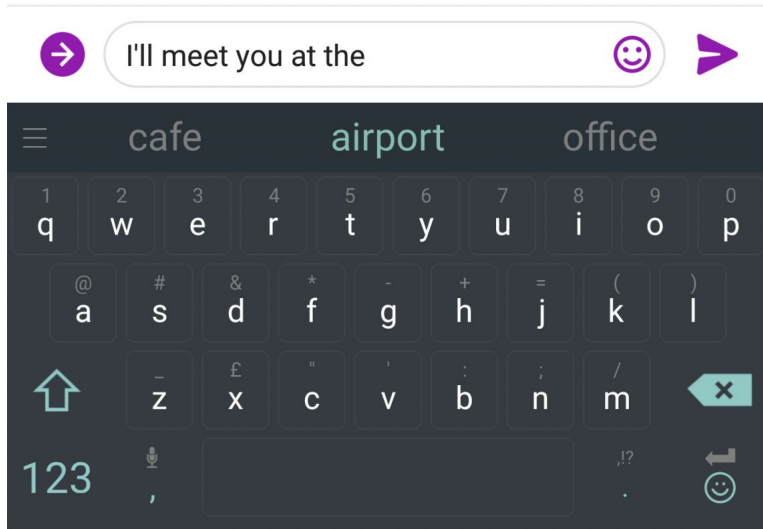
What is a Language Model?

A model that can calculate the probability of a sequence of text.

It can calculate the probability distribution of the next word, given the words that came before it.



What is a Language Model?



What is a Language Model?

ChatGPT

Examples	Capabilities	Limitations
"Explain quantum computing in simple terms"	Remembers what user said earlier in the conversation	May occasionally generate incorrect information
"Got any creative ideas for a 10 year old's birthday?"	Allows user to provide follow-up corrections	May occasionally produce harmful instructions or biased content
"How do I make an HTTP request in Javascript?"	Trained to decline inappropriate requests	Limited knowledge of world and events after 2021

ChatGPT is optimized for dialogue. Our goal is to make AI systems more natural to interact with, and your feedback will help us improve our system.

```
pub(crate) struct TransportStack {
    72: listenerEndpoint: ListenerEndpoint,
    73: tls: Option<Arc<Acceptor>>,
    74: // listeners sent from the old process for graceful upgrade
    75: #cfg(unix)
    76: upgrade_listeners: Option<ListenerFds>,
    77: upgrade_listeners: ListenerFds,
    78: }
    79:
    80: impl TransportStack {
    81:     pub fn as_str(&self) -> &str {
    82:         self.id.as_str()
    83:     }
    84:
    85:     pub async fn listen(&mut self) -> Result<()> {
    86:         self.id
    87:         .listen(
    88:             #cfg(unix)
    89:             self.upgrade_listeners.take(),
    90:             Some(self.upgrade_listeners.take()),
    91:             .await
    92:         )
    93:     }
    94:
    95:     pub async fn accept(&mut self) -> Result<UninitializedStream> {
    96:         let stream = self.id.accept().await?;
    97:         Ok(UninitializedStream {
    98:             id: stream,
    99:             tls: self.tls.clone(),
    100:         })
    101:     }
    102: }
```

CHAT COMPOSER

mod.rs Current File

Could you make it easier to switch certificates in the transport listeners?

I'll help modify the code to make certificate switching more flexible. The main changes will be to enhance the `TlsAccept` trait and modify how certificates are handled in the `TlsSettings`. Here are the key changes:

```
... existing code ...
// The APIs to customize things like certificate during TLS ser
#async_trait
pub trait TlsAccept {
    // Called during TLS handshake to select and apply certifi
    // Returns true if certificates were successfully applied
    async fn certificate_callback(&self, sst: &mut TlsRef) -> bo
    // Default implementation does nothing and indicates no fi
}
// Called when certificate selection fails
async fn on_cert_error(&self, _error: &str) {
    // Default implementation does nothing
}
}
// Add a default no-op implementation that can be used when no c
#[derive(Default)]
```

mod.rs Current File

Ask followup (NY), ↑ to select

cloud-3-5-sonnet-2024-1022 @ Mention ⌘ no context ↩ chat ⌘ codebase

What is an LLM?

A Large Language Model!

The term is relatively new (~2020), but the models themselves have been around for a while before that, so we're going to refer to them as LLMs.

LLMs

Encoder Only:

I love cake

understand

Decoder Only:

I love

generate

Encoder-Decoder:

I love cake

me gusta

understand + generate

LLMs

Encoder Only:

I love cake

Decoder Only:

I love

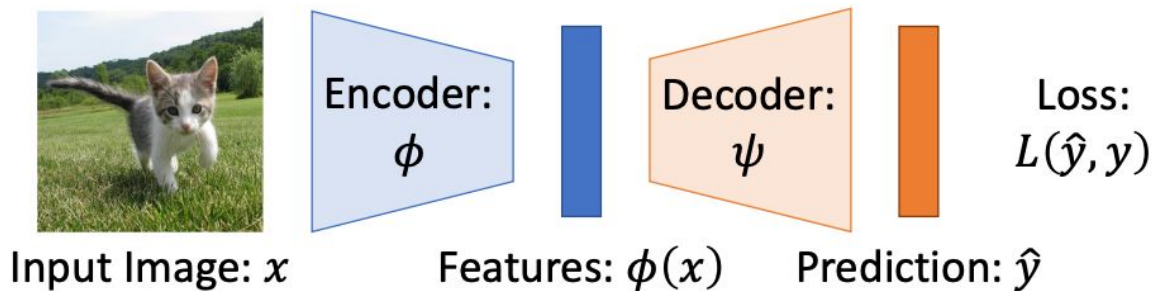
Encoder-Decoder:

I love cake

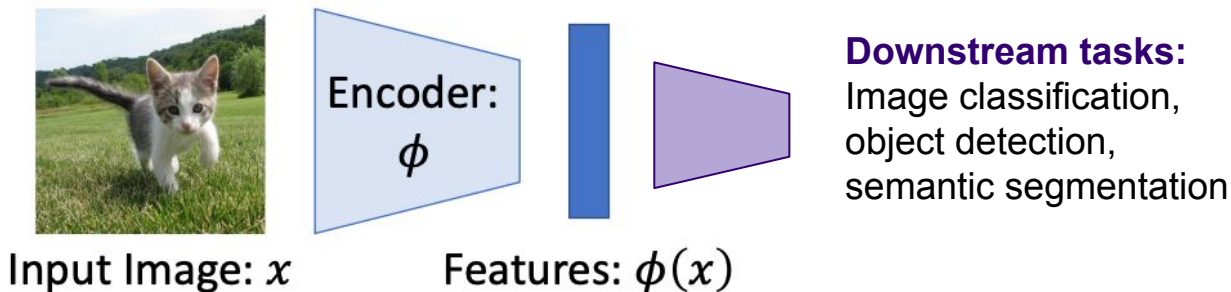
me gusta

Remember: Pre-train and then fine-tune

Step 1: Pretrain a network on a pretext task that doesn't require supervision



Step 2: Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning

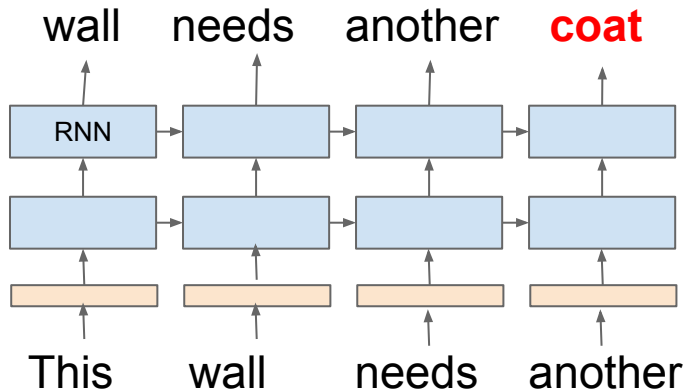


ELMo (Embeddings from Language Models)

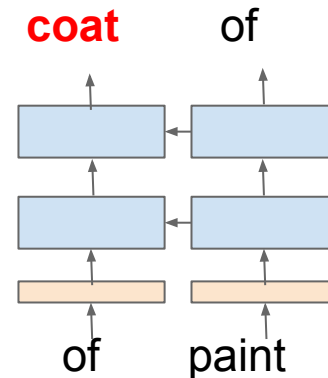
Pre-training task

This wall needs another coat of paint

Predict word based on previous words



Predict word based on following words

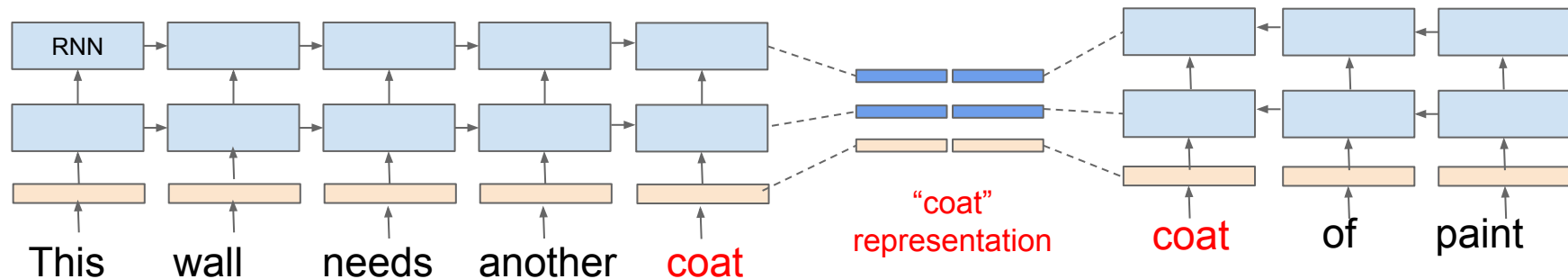


Peters et al. Deep contextualized word representations. 2018.

ELMo (Embeddings from Language Models)

Pre-training task

This wall needs another coat of paint

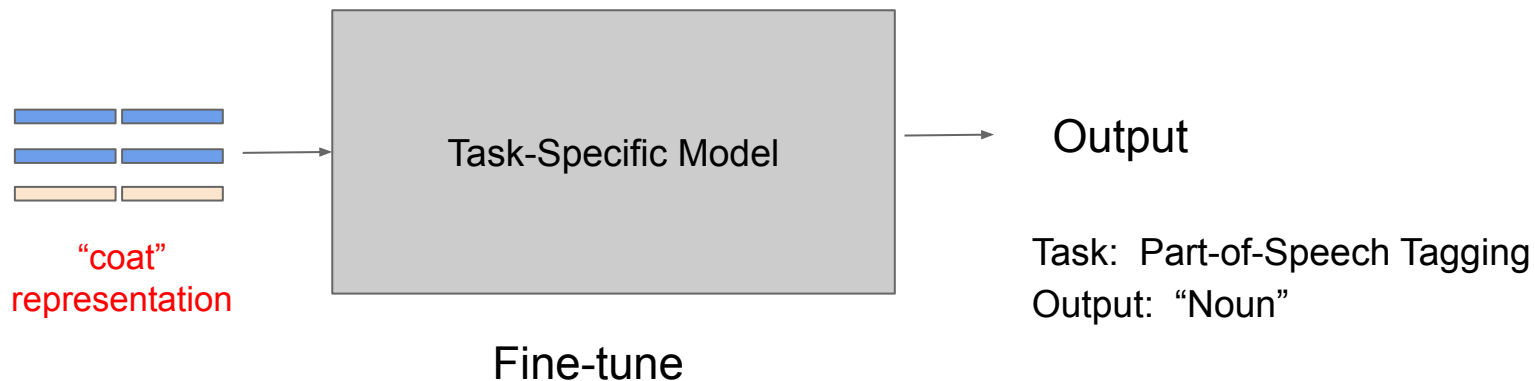


Peters et al. Deep contextualized word representations. 2018.

ELMo (Embeddings from Language Models)

Application to downstream tasks

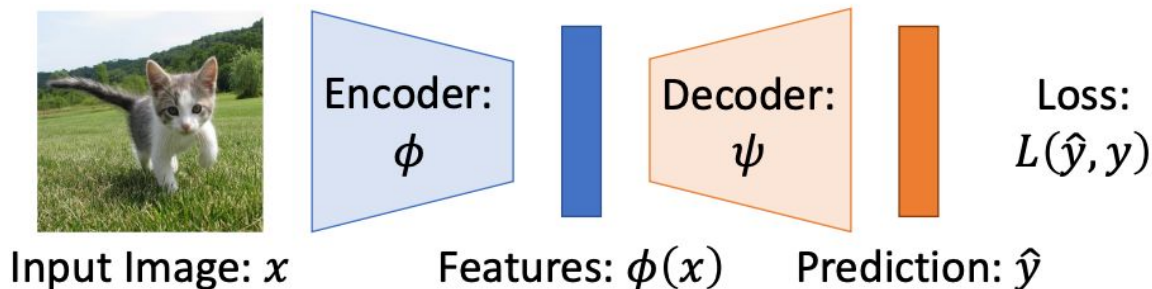
This wall needs another coat of paint



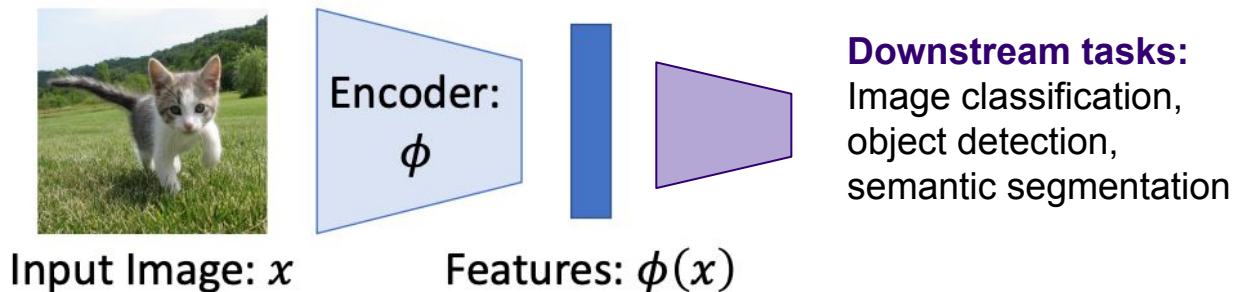
Peters et al. Deep contextualized word representations. 2018.

Vision: Pre-train and then fine-tune

Step 1: Pretrain a network on a pretext task that doesn't require supervision

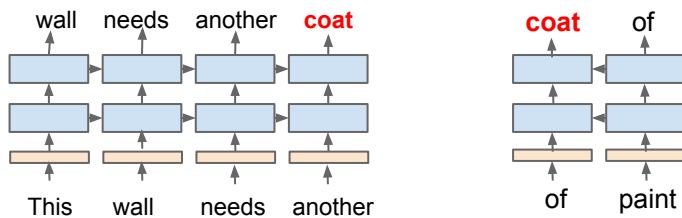


Step 2: Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning

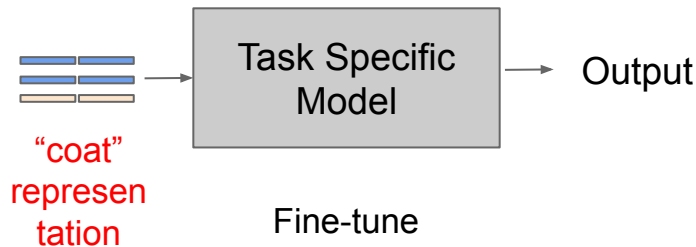


Language: Pre-train and then fine-tune

Step 1: Pretrain a network on a pretext task that doesn't require supervision



Step 2: Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning



Downstream tasks:
Sentiment classification, NLI,
...

ELMo vs GloVe

GloVe embeddings use the same vector for every instance of a word, no matter the context.

It's cold today! Don't forget to wear a coat.

This wall needs another coat of paint

$$\text{coat} = \begin{bmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{bmatrix}$$

ELMo embeddings are contextualized.

ELMo Results

TASK	PREVIOUS SOTA		OUR BASELINE	ELMo + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 \pm 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 \pm 0.19	90.15	92.22 \pm 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 \pm 0.5	3.3 / 6.8%

Source: [Peters et al. Deep contextualized word representations. 2018.](#)

ELMo

Used RNNs... what about transformers?

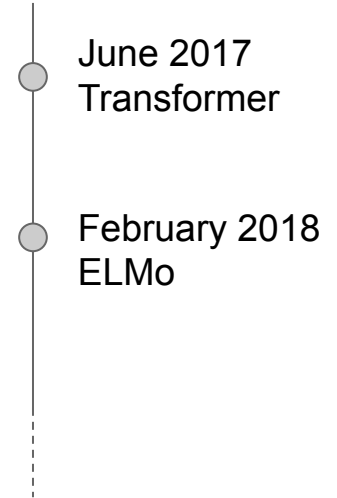


ELMo

Used RNNs... what about transformers?

Let's combine:


- (1) ELMo approach of gathering general knowledge about world
- (2) Powerful multi-purpose architecture that is the transformer



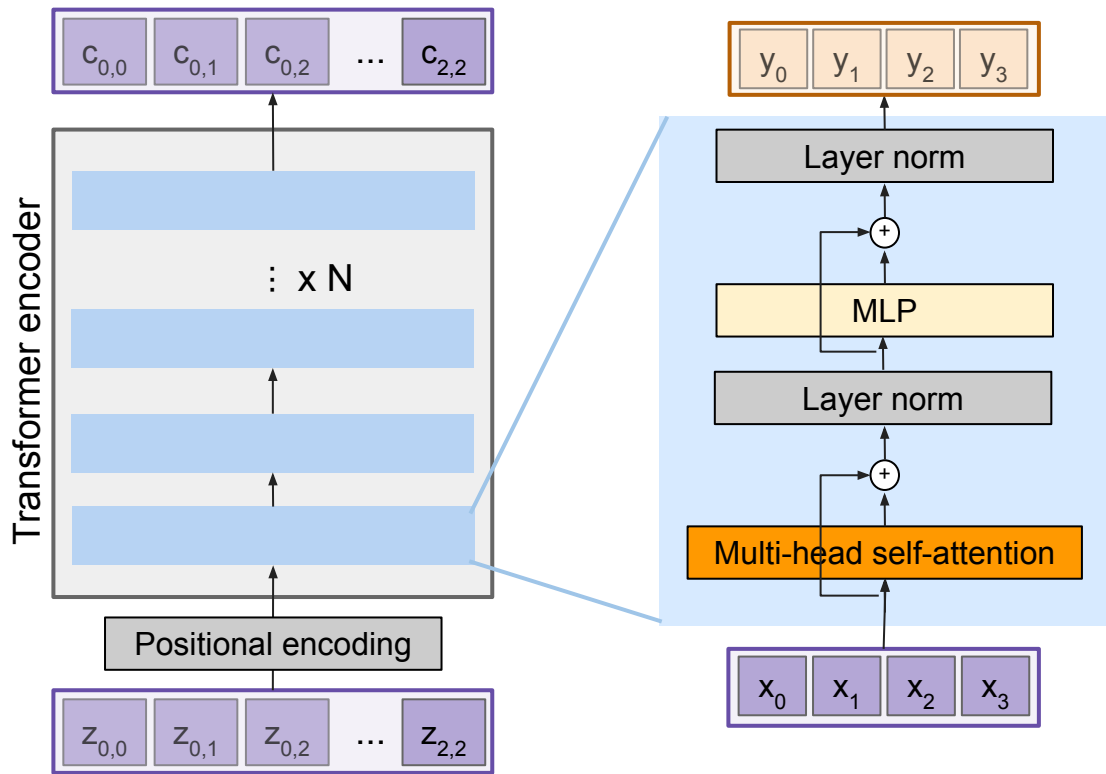
...  July 2020
BIG BIRD

 May 2019
GROVER

 May 2019
ERNIE

 October 2018
BERT

Recall: Transformer encoder block



Transformer Encoder Block:

Inputs: Set of vectors x

Outputs: Set of vectors y

Self-attention is the only interaction between vectors.

Layer norm and MLP operate independently per vector.

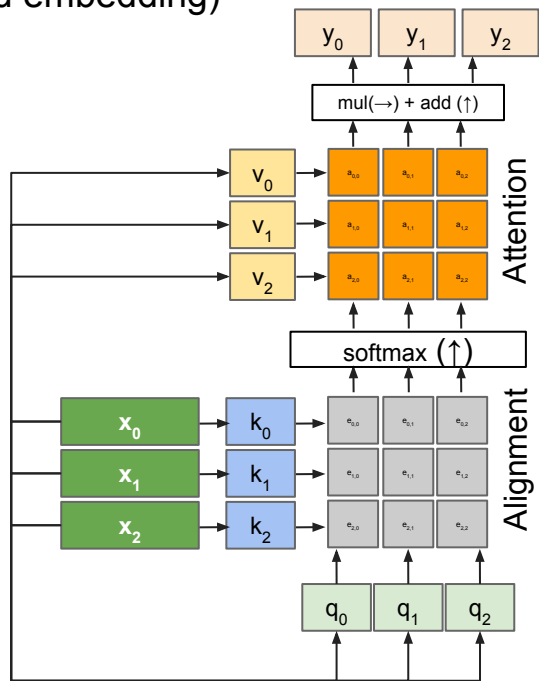
Highly scalable, highly parallelizable, but high memory usage.

Vaswani et al, "Attention is all you need", NeurIPS 2017

Encoder Only: BERT (Bidirectional Encoder Representations from Transformers)

Input: Text sequence

Output: Feature Vector (contextualized embedding)



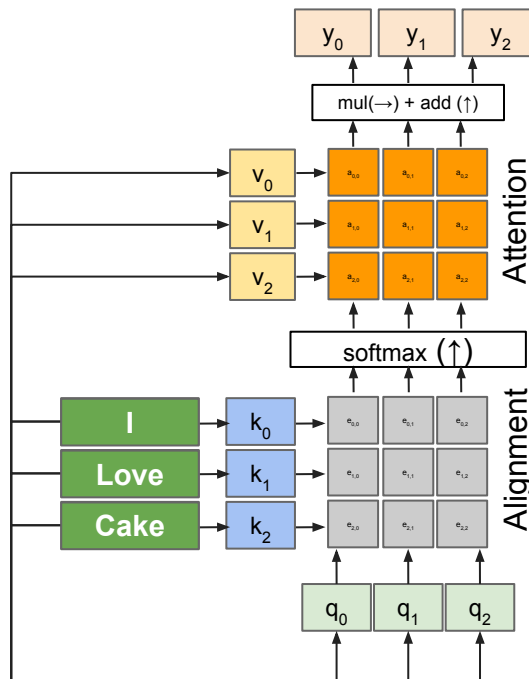
Outputs:
context vectors: \mathbf{y} (shape: D_v)

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

Encoder Only: BERT

Input: Text sequence

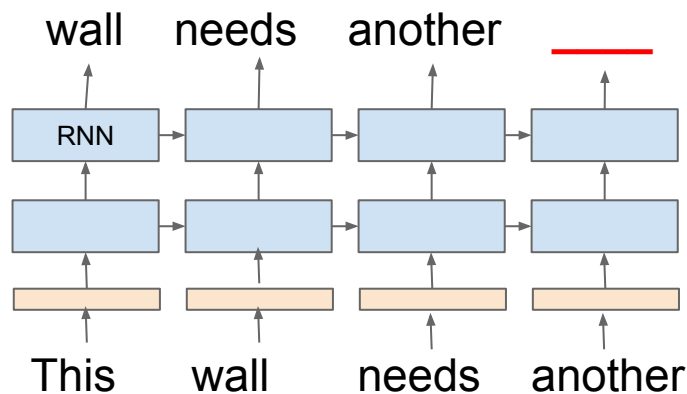
Output: Feature Vector



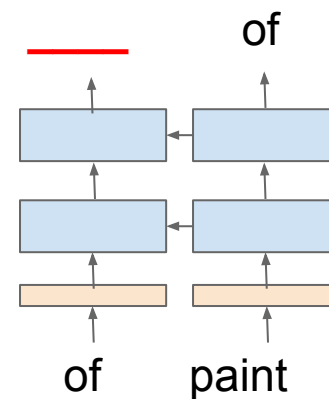
Outputs:
context vectors: \mathbf{y} (shape: D_v)

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

ELMo (Embeddings from Language Models)



“coat”



Encoder Only: BERT

Input: Text sequence

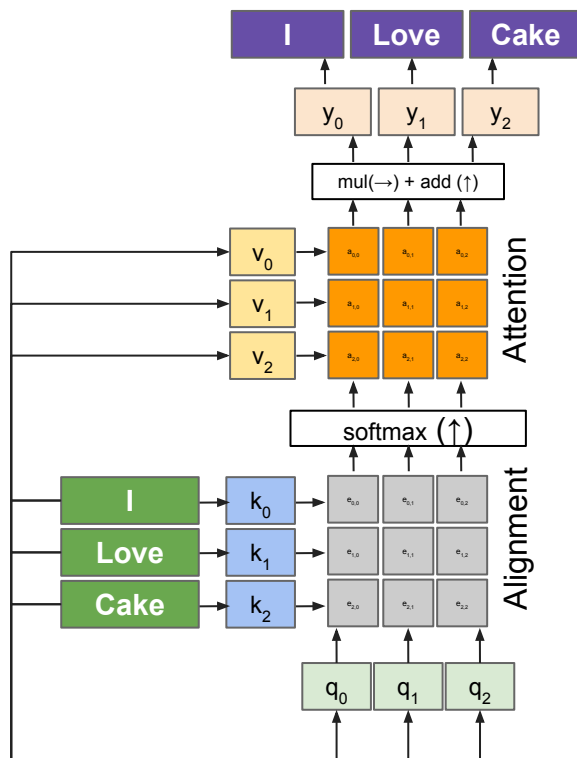
Output: Feature Vector

What information do the y vectors contain?

Just copying the input!

Because they can see the whole input, unlike sequential RNNs.

Pre-training task too easy!!



Outputs:
context vectors: y (shape: D_v)

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

Encoder Only: BERT

Input: Text sequence

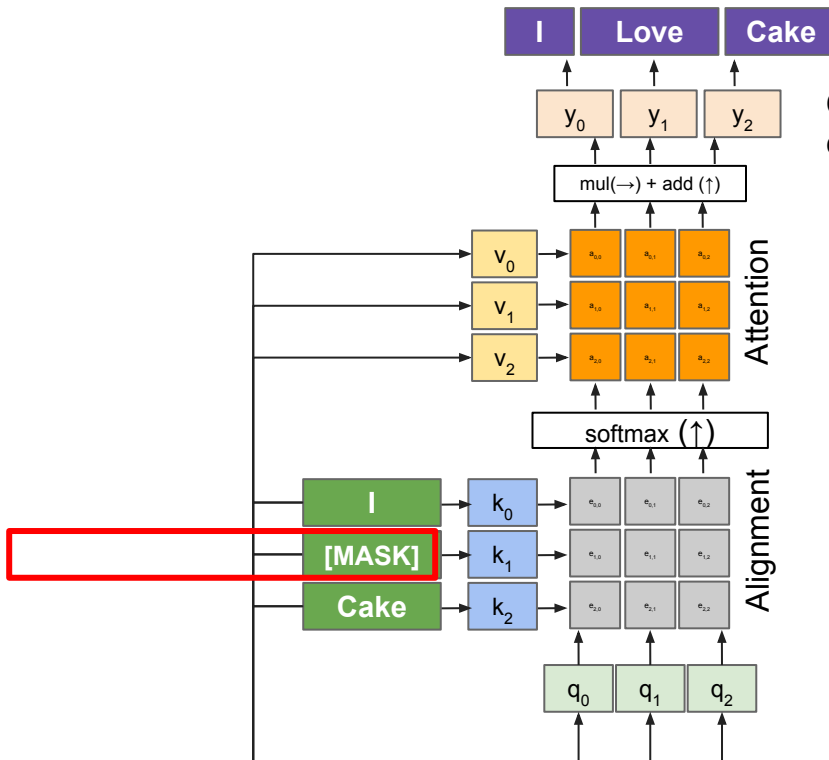
Output: Feature Vector

Randomly select 15% of tokens.

80% – [MASK]

10% – random token

10% – keep same



Outputs:
context vectors: \mathbf{y} (shape: D_v)

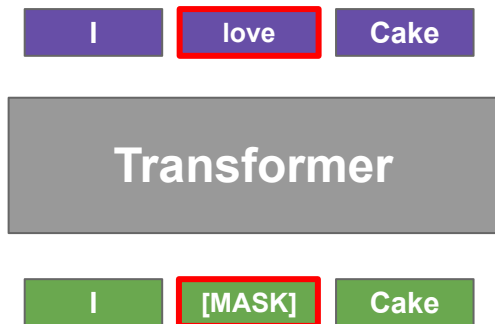
Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

Encoder Only: BERT

Input: Text sequence

Output: Feature Vector

1. Masked Language Modeling



Randomly select 15% of tokens.

80% – [MASK]

10% – random token

10% – keep same

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

Encoder Only: BERT

Input: Text sequence

Output: Feature Vector

1. Masked Language Modeling
2. Next Sentence Prediction

Does sentence 2 follow sentence 1?

[CLS] I love Cake [SEP] It's so good [SEP]

Transformer

[CLS] I [MASK] Cake [SEP] It's so good [SEP]

Randomly select 15% of tokens.

80% – [MASK]

10% – random token

10% – keep same

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

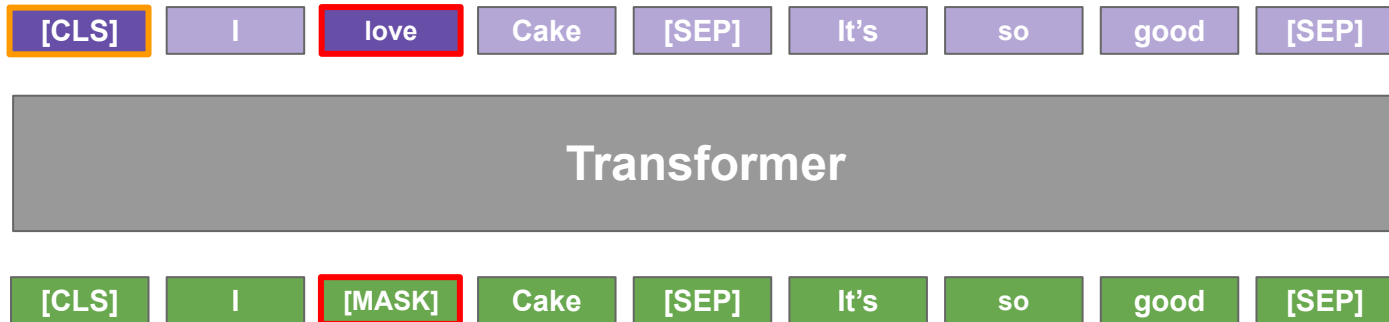
Encoder Only: BERT

Input: Text sequence

Output: Feature Vector

1. Masked Language Modeling → reconstruction
2. Next Sentence Prediction → classification

Does sentence 2 follow sentence 1?



Randomly select 15% of tokens.

80% – [MASK]

10% – random token

10% – keep same

Note: “reconstruction” is classification over the vocabulary, the same way NSP “classification” is classification over 2 classes.

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.

Encoder Only: BERT

Application to downstream tasks

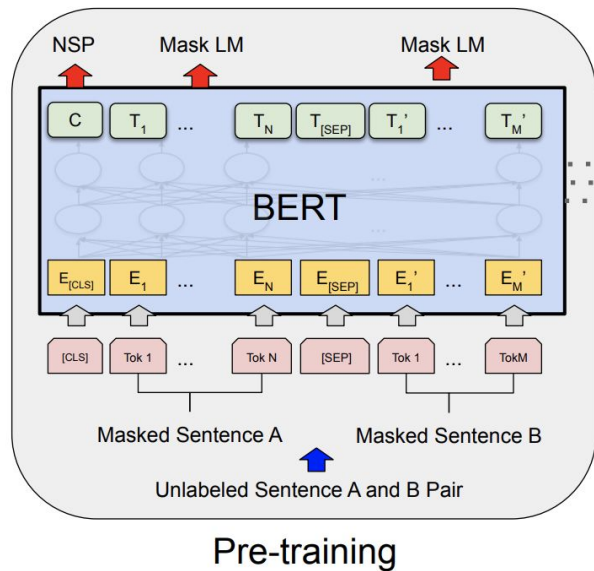


Image Source: [Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.](#)

Encoder Only: BERT

Application to downstream tasks

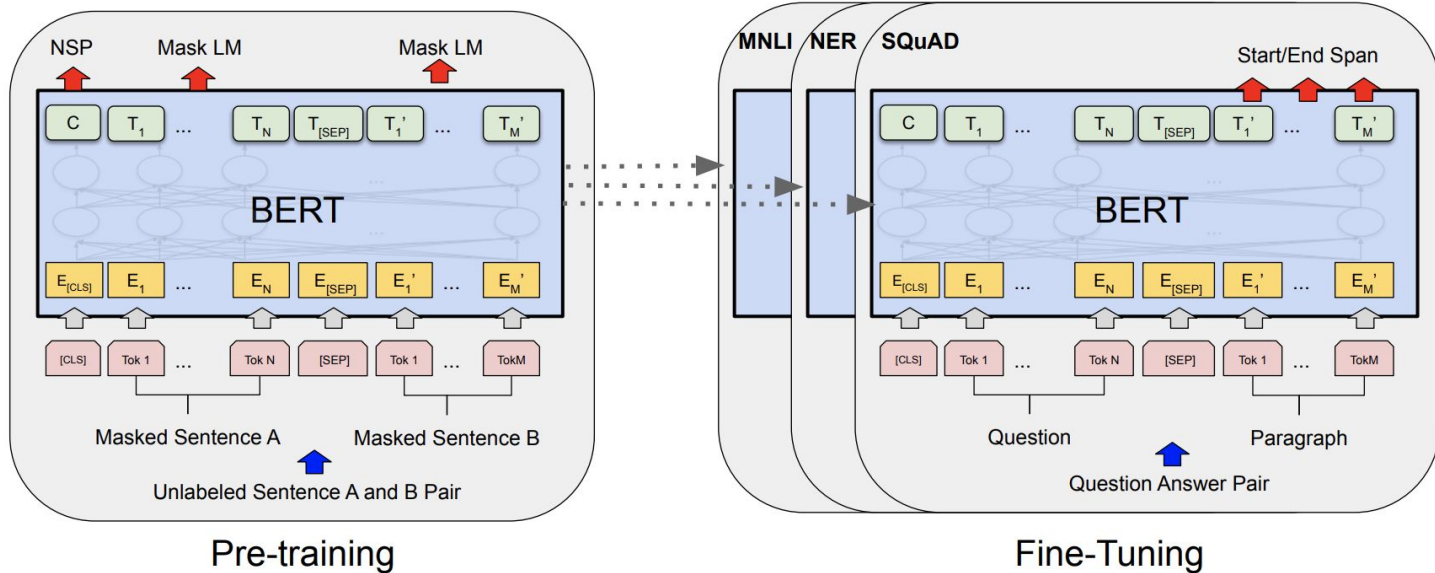
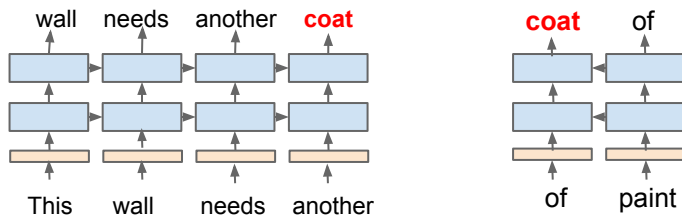


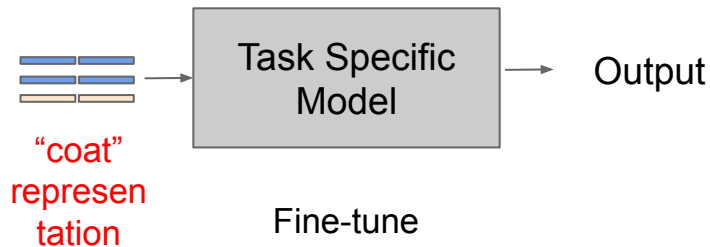
Image Source: [Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.](#)

ELMO: Two step process

Step 1: Pretrain a network on a pretext task that doesn't require supervision



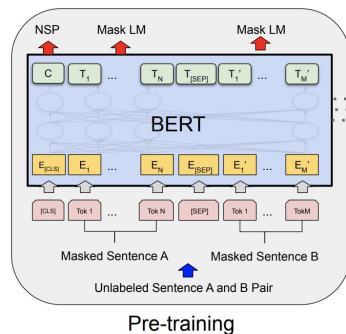
Step 2: Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning



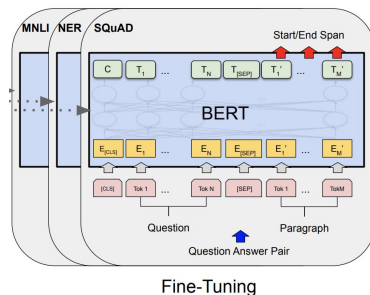
Downstream tasks:
Sentiment classification, NLI,
...

BERT: Two step process

Step 1: Pretrain a network on a pretext task that doesn't require supervision



Step 2: Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning



Downstream tasks:
Sentiment
classification, NLI,
...

Image Source: [Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.](#)

BERT Results

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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT _{BASE}	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	91.1	94.9	60.5	86.5	89.3	70.1	81.9

Image Source: [Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.](#)

BERT Results

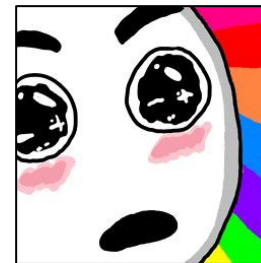
System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
BERT _{BASE}	81.6	-
BERT _{LARGE}	86.6	86.3
Human (expert) [†]	-	85.0
Human (5 annotations) [†]	-	88.0

Table 4: SWAG Dev and Test accuracies. [†]Human performance is measured with 100 samples, as reported in the SWAG paper.

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
BERT _{BASE} (Single)	80.8	88.5	-	-
BERT _{LARGE} (Single)	84.1	90.9	-	-
BERT _{LARGE} (Ensemble)	85.8	91.8	-	-
BERT _{LARGE} (Sgl.+TriviaQA)	84.2	91.1	85.1	91.8
BERT _{LARGE} (Ens.+TriviaQA)	86.2	92.2	87.4	93.2

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

Model researchers:



Dataset researchers:



Image Source: [Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018.](#)

LLMs

Encoder Only:

I love cake

ELMO: Bi-directional next word prediction

BERT: Masked language modeling, Next Sentence Prediction

Decoder Only:

I love

Encoder-Decoder:

I love cake

me gusta

LLMs

Encoder Only:

I love cake

ELMO: Bi-directional next word prediction

BERT: Masked language modeling, Next Sentence Prediction

Decoder Only:

I love

Encoder-Decoder:

I love cake

me gusta

Decoder Only: GPT

We learned about Masked Language Models, but we don't use them today → why?

They can be used for NLP tasks (primarily classification), but they're not very good at generating sentences: they're best at filling in blanks when they have *bidirectional context*.

We argue that current techniques severely restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers of the Transformer (Vaswani et al., 2017).

Such restrictions are sub-optimal for sentence-level tasks, and could be devastating when applying fine-tuning based approaches to token-level tasks such as SQuAD question answering (Rajpurkar et al., 2016), where it is crucial to incorporate context from both directions.

And BERT did beat GPT. A lot of researchers bought into bidirectional models.

RoBERTa (2019)
ALBERT (2019)
ELECTRA (2020)

...

But, it turns out that with enough scaling...

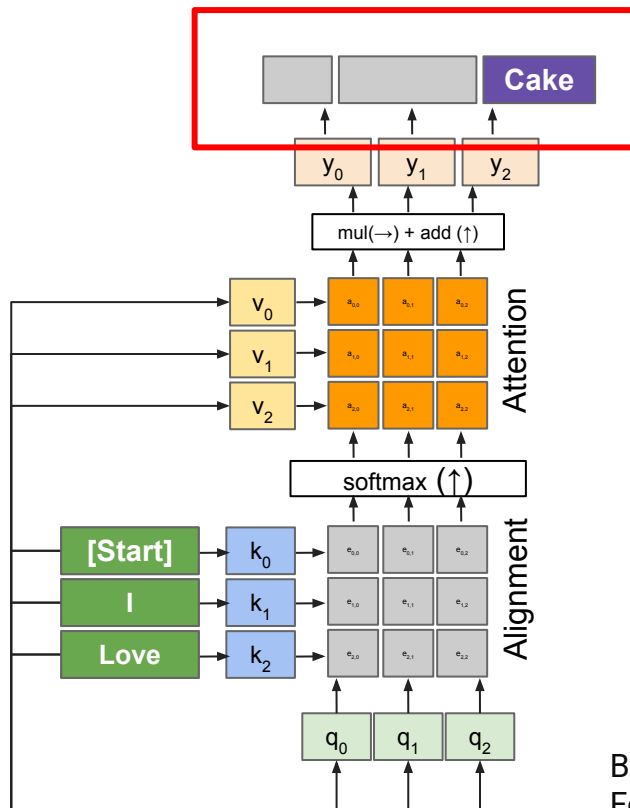
GPT-3 (2020)

Uni-directional models win.

Decoder Only: GPT

Input: Text sequence

Output: Completed text sequence



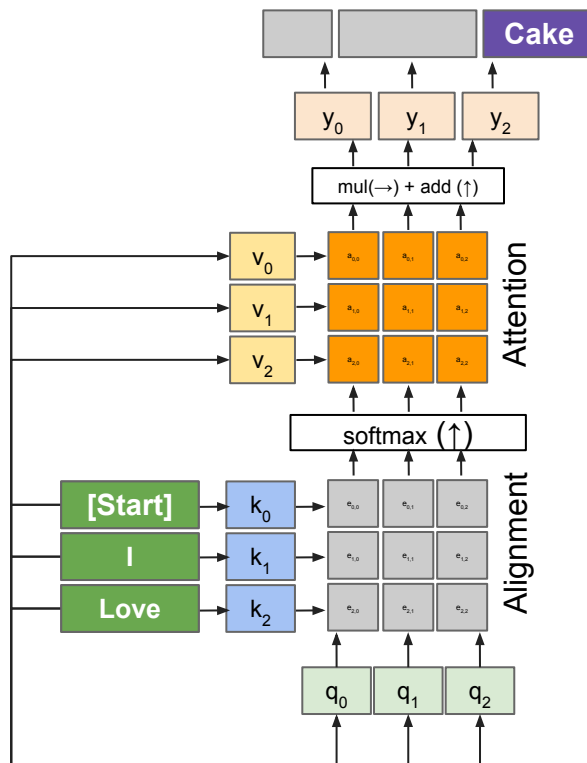
Brown et al. Language Models are Few-Shot Learners. 2020.

Decoder Only: GPT

Input: Text sequence

Output: Completed text sequence

Cons: Need to process entire sentence in order to get loss from one word - not very much signal for the amount of processing



Brown et al. Language Models are Few-Shot Learners. 2020.

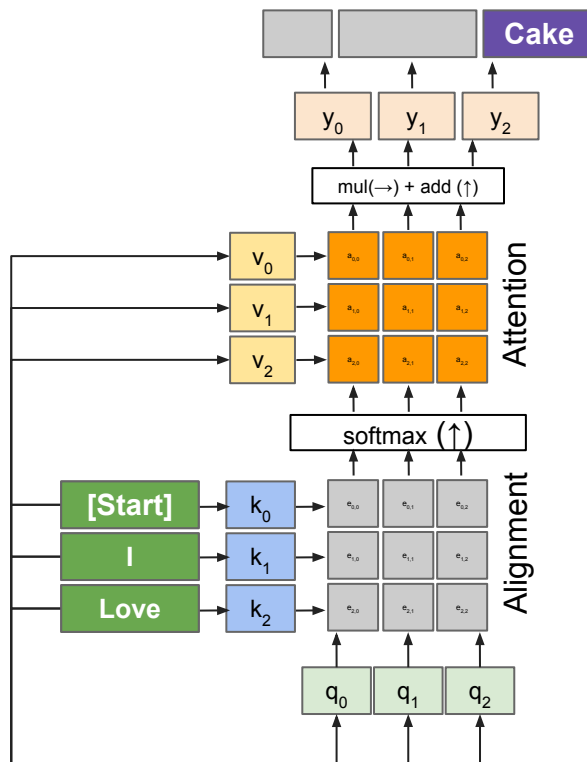
Decoder Only: GPT

Input: Text sequence

Output: Completed text sequence

Cons: Need to process entire sentence in order to get loss from one word - not very much signal for the amount of processing

Solution: predict each word given previous words so far



Brown et al. Language Models are Few-Shot Learners. 2020.

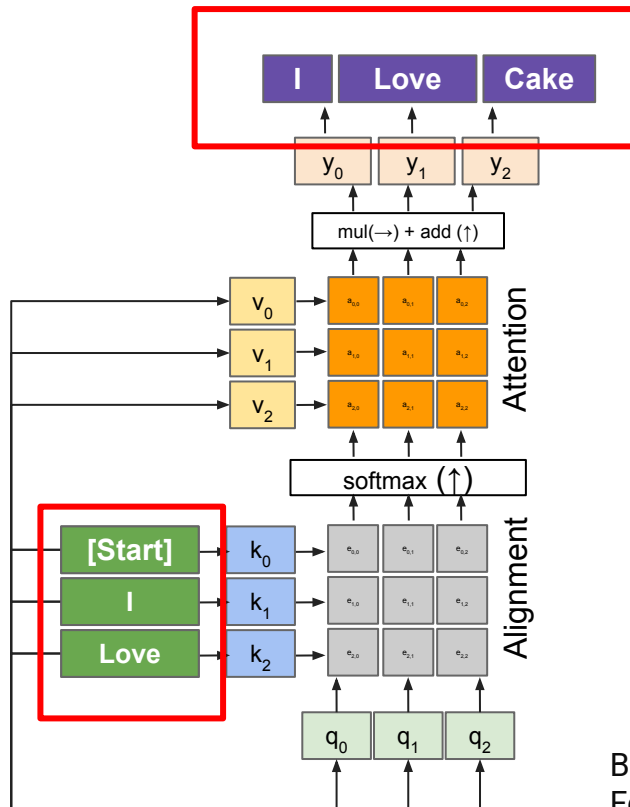
Decoder Only: GPT

Input: Text sequence

Output: Completed text sequence

Cons: Need to process entire sentence in order to get loss from one word - not very much signal for the amount of processing

Solution: predict each word given previous words so far



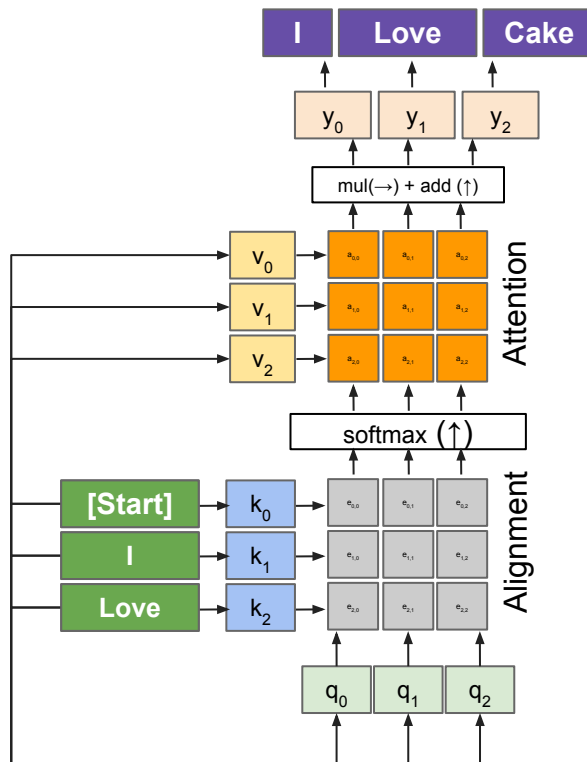
Brown et al. Language Models are Few-Shot Learners. 2020.

Decoder Only: GPT

Input: Text sequence

Output: Completed text sequence

What's wrong with this?



Brown et al. Language Models are Few-Shot Learners. 2020.

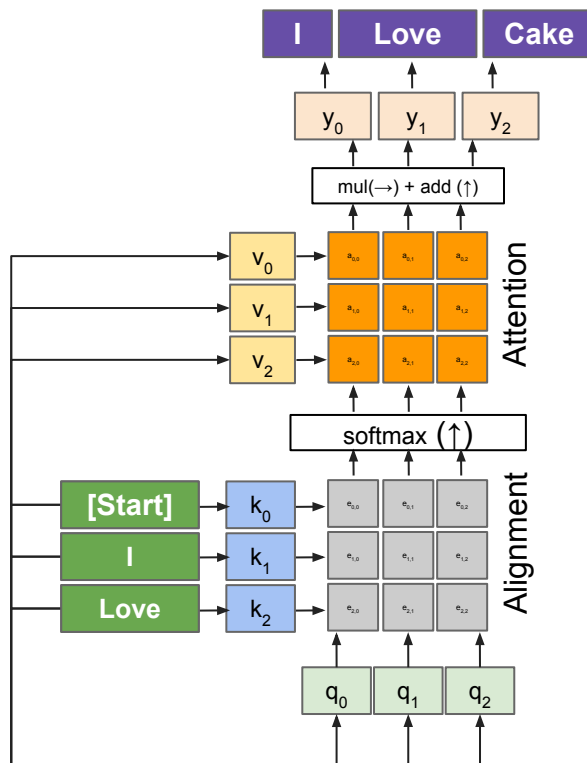
Decoder Only: GPT

Input: Text sequence

Output: Completed text sequence

What's wrong with this?

It can see the answer!



Brown et al. Language Models are Few-Shot Learners. 2020.

Decoder Only: GPT

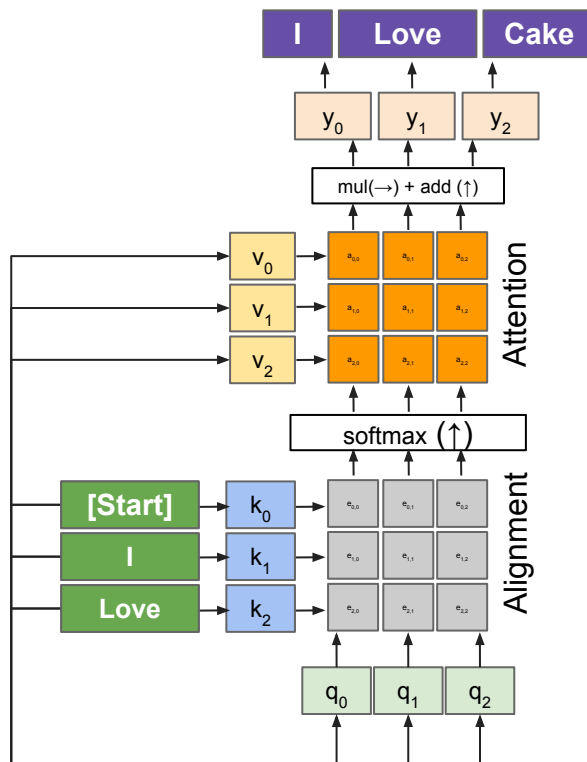
Input: Text sequence

Output: Completed text sequence

What's wrong with this?

It can see the answer!

Solution: zero out values from future words



Brown et al. Language Models are Few-Shot Learners. 2020.

Decoder Only: GPT

Input: Text sequence

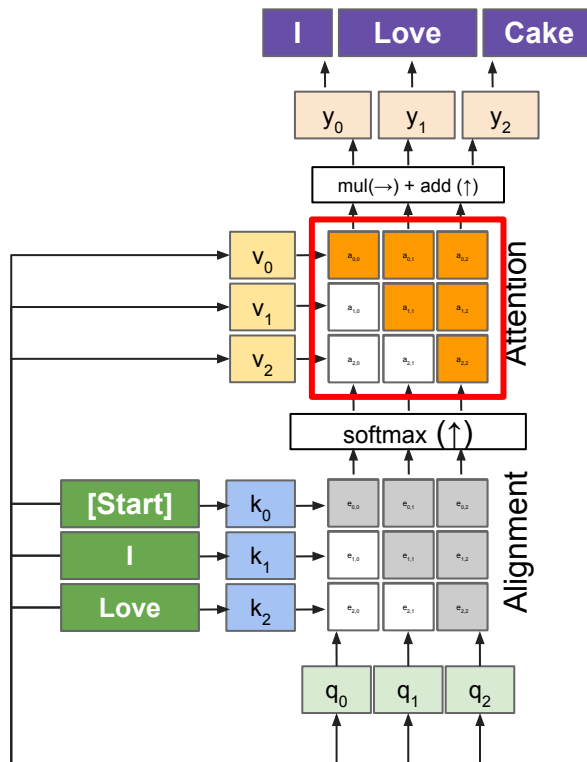
Output: Completed text sequence

What's wrong with this?

It can see the answer!

Solution: zero out values from future words

→ “causal masking”



Brown et al. Language Models are Few-Shot Learners. 2020.

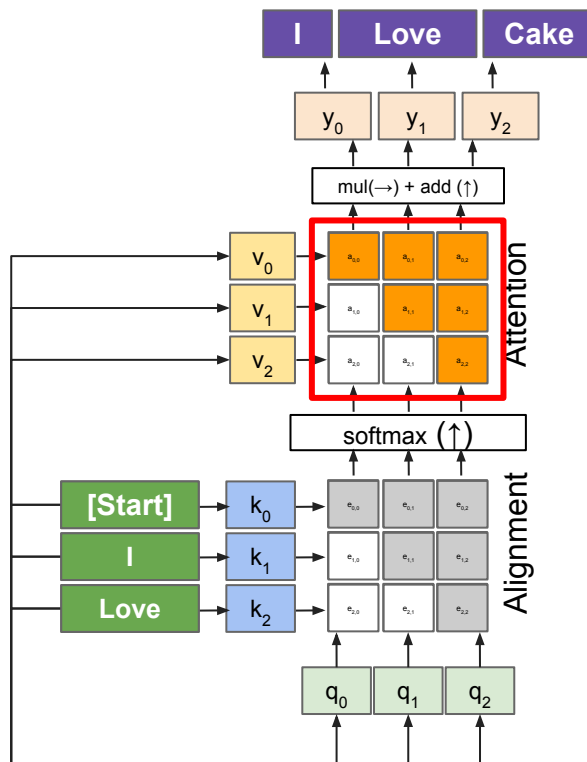
Decoder Only: GPT

Input: Text sequence

Output: Completed text sequence

To pre-train: predict next words from previous words for large text corpus

“auto-regressive”



Brown et al. Language Models are Few-Shot Learners. 2020.

LLMs

Encoder Only:

I love cake

ELMO: Bi-directional next word prediction

BERT: Masked language modeling, Next Sentence Prediction

Decoder Only:

I love

GPT: next token prediction (auto-regressive)

Encoder-Decoder:

I love cake

me gusta

LLMs

Encoder Only:

I love cake

ELMO: Bi-directional next word prediction

BERT: Masked language modeling, Next Sentence Prediction

Decoder Only:

I love

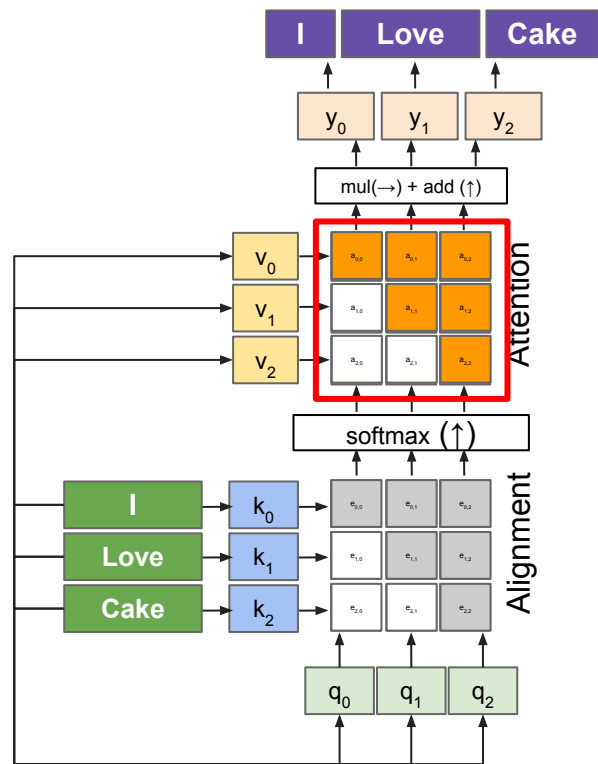
GPT: next token prediction (auto-regressive)

Encoder-Decoder:

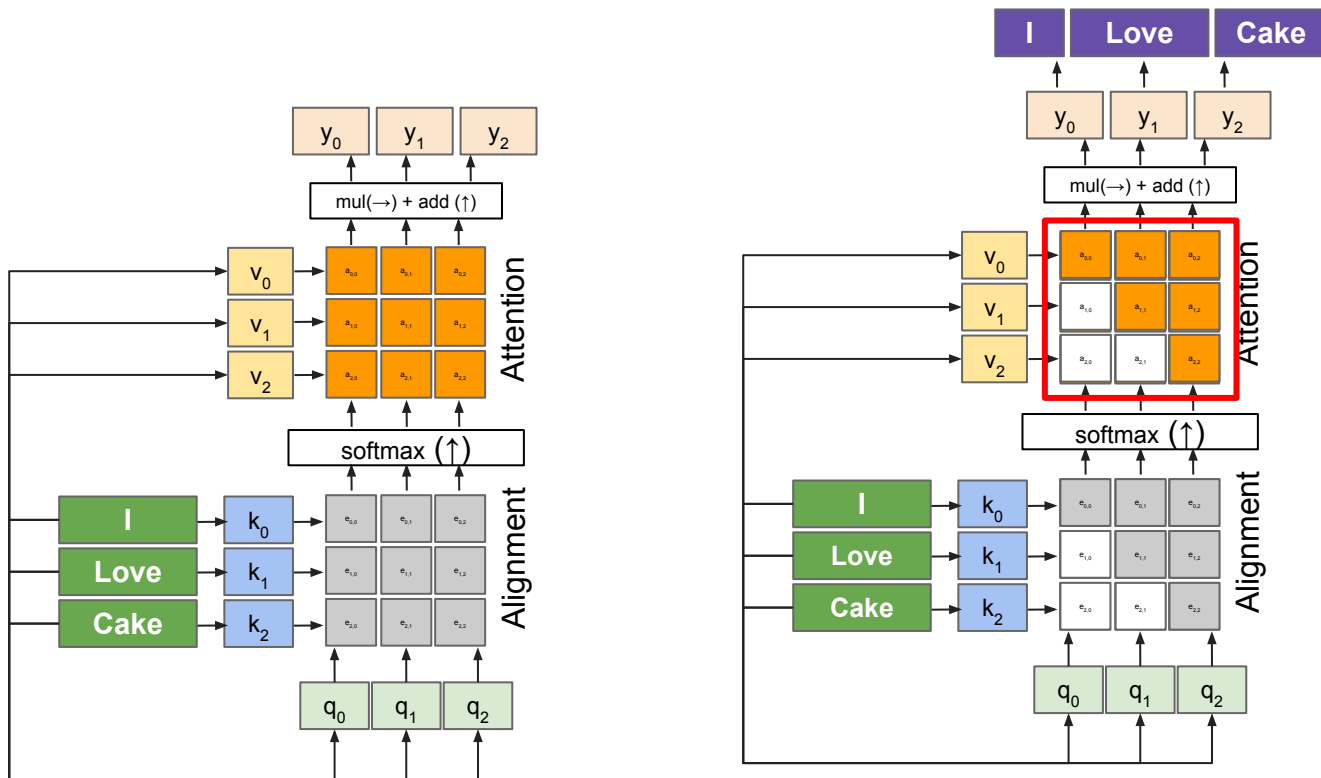
I love cake

me gusta

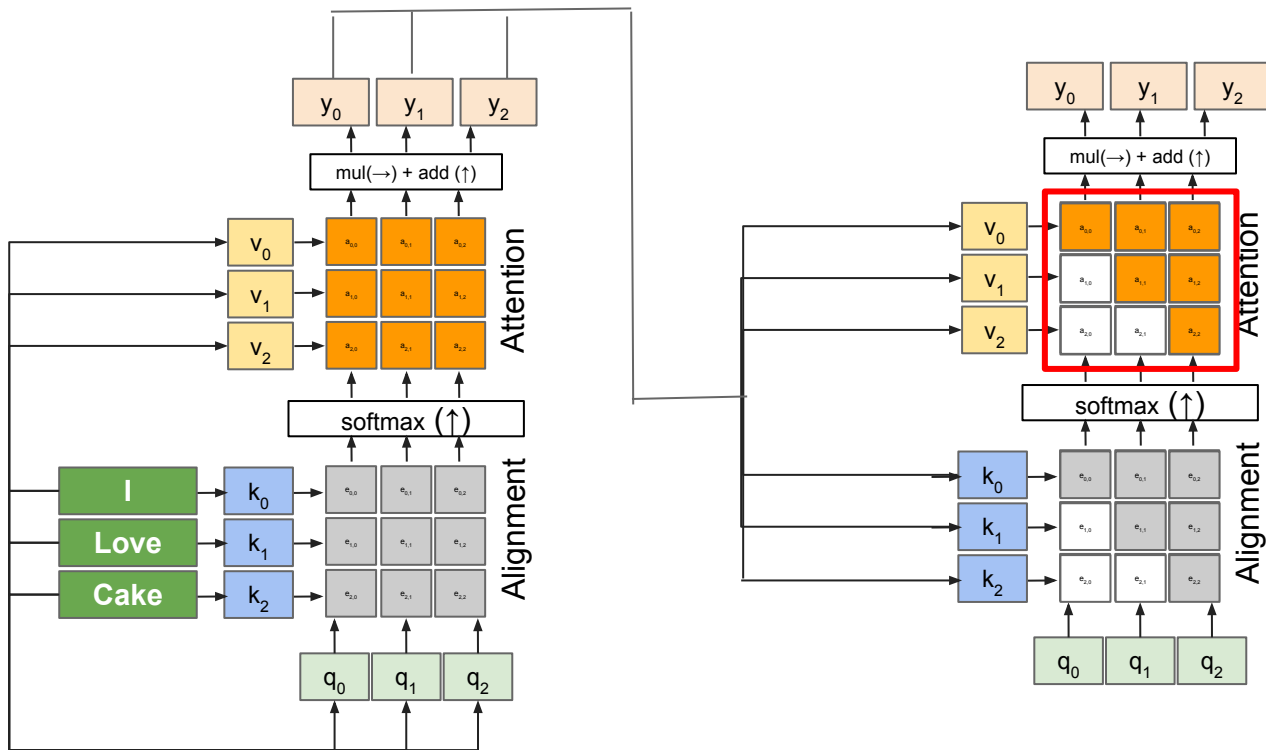
Encoder-Decoder: Generate text based on previously generated text...



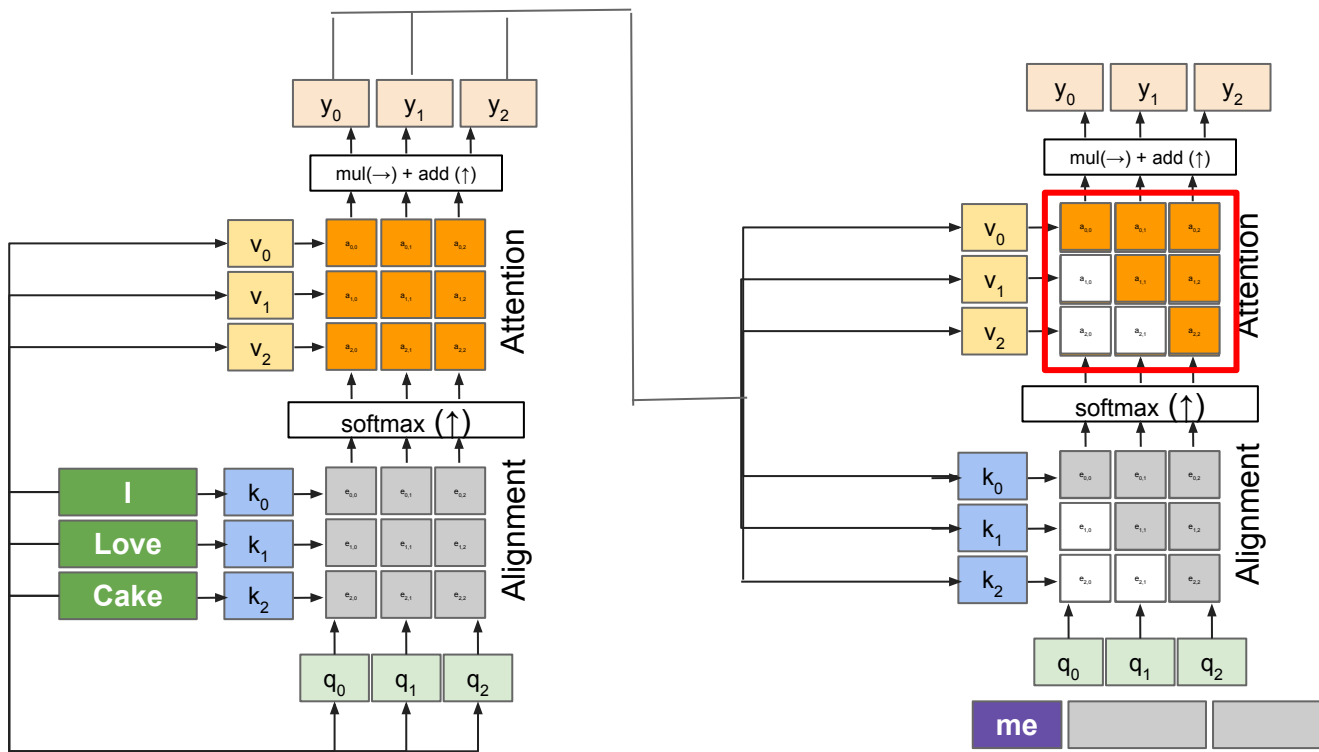
Encoder-Decoder: Generate text based on previously generated text... and the meaning of a separate sequence



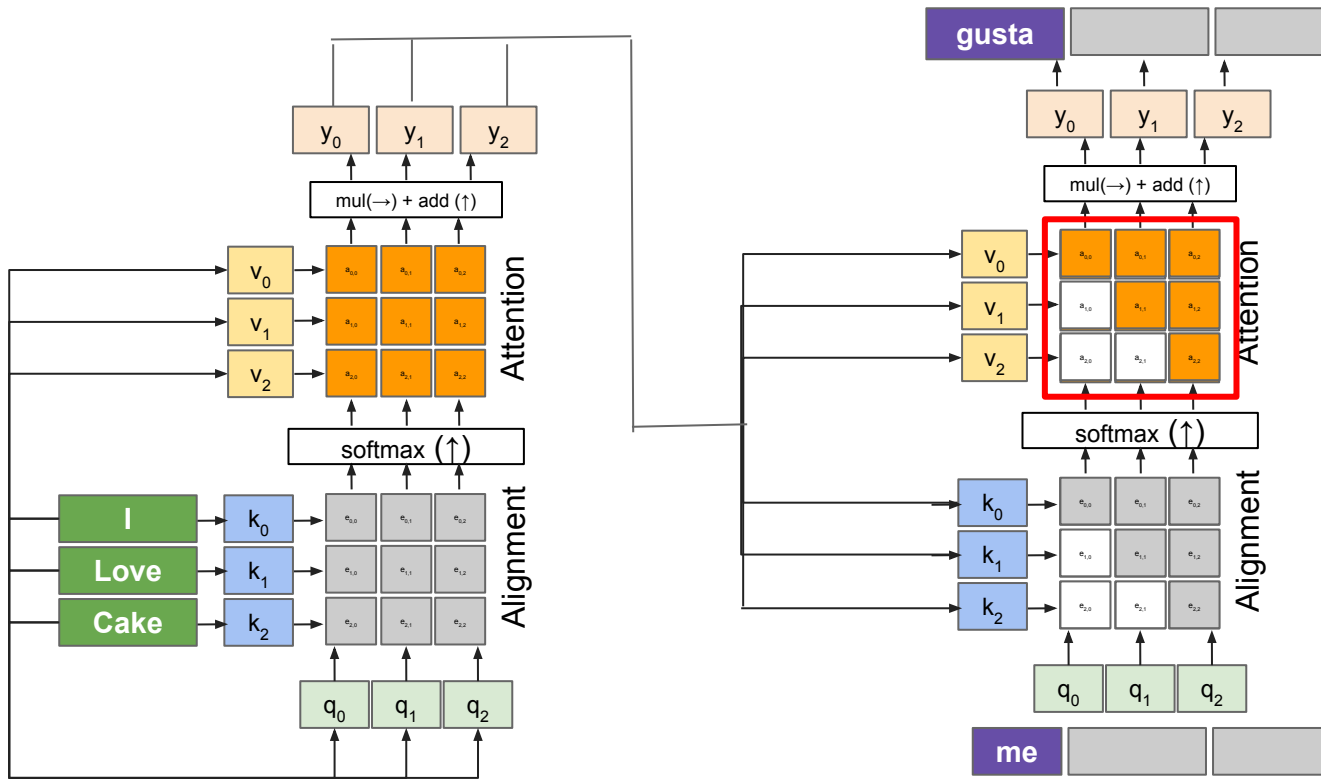
Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence



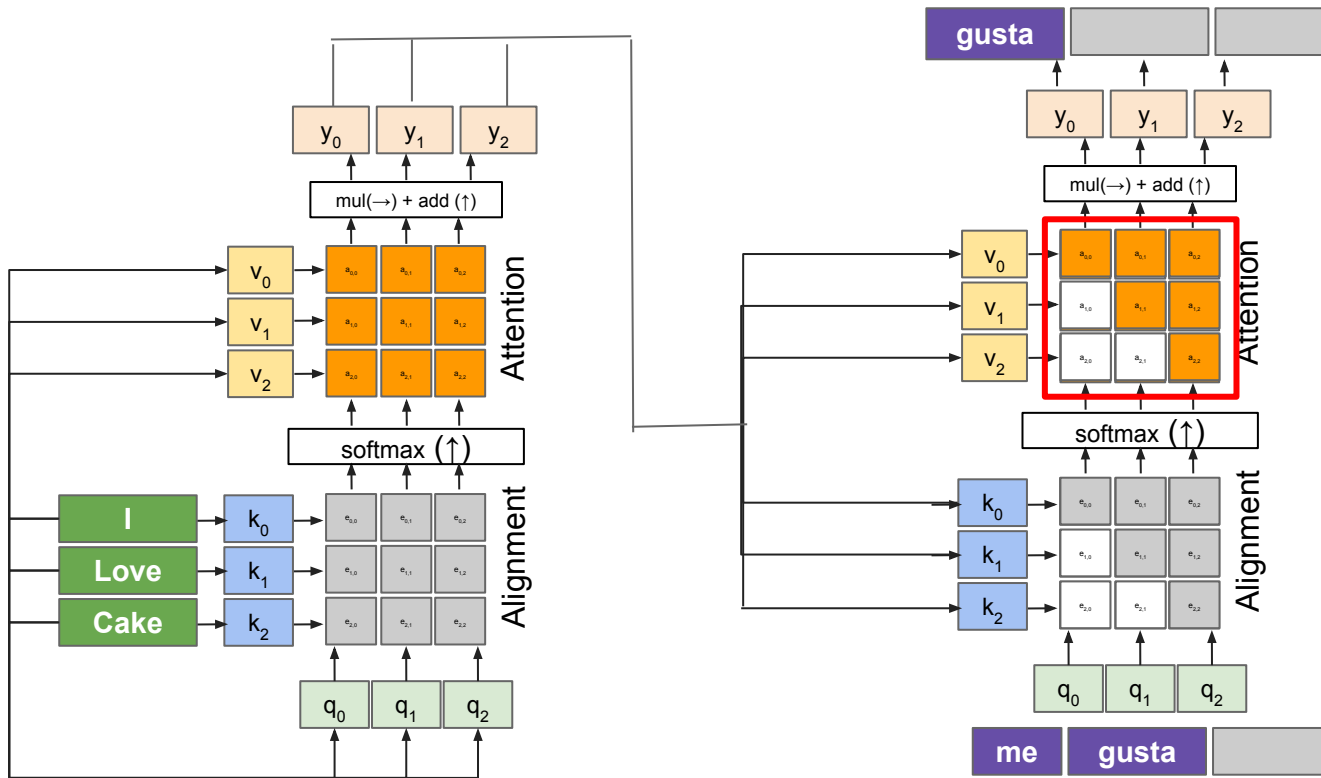
Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence



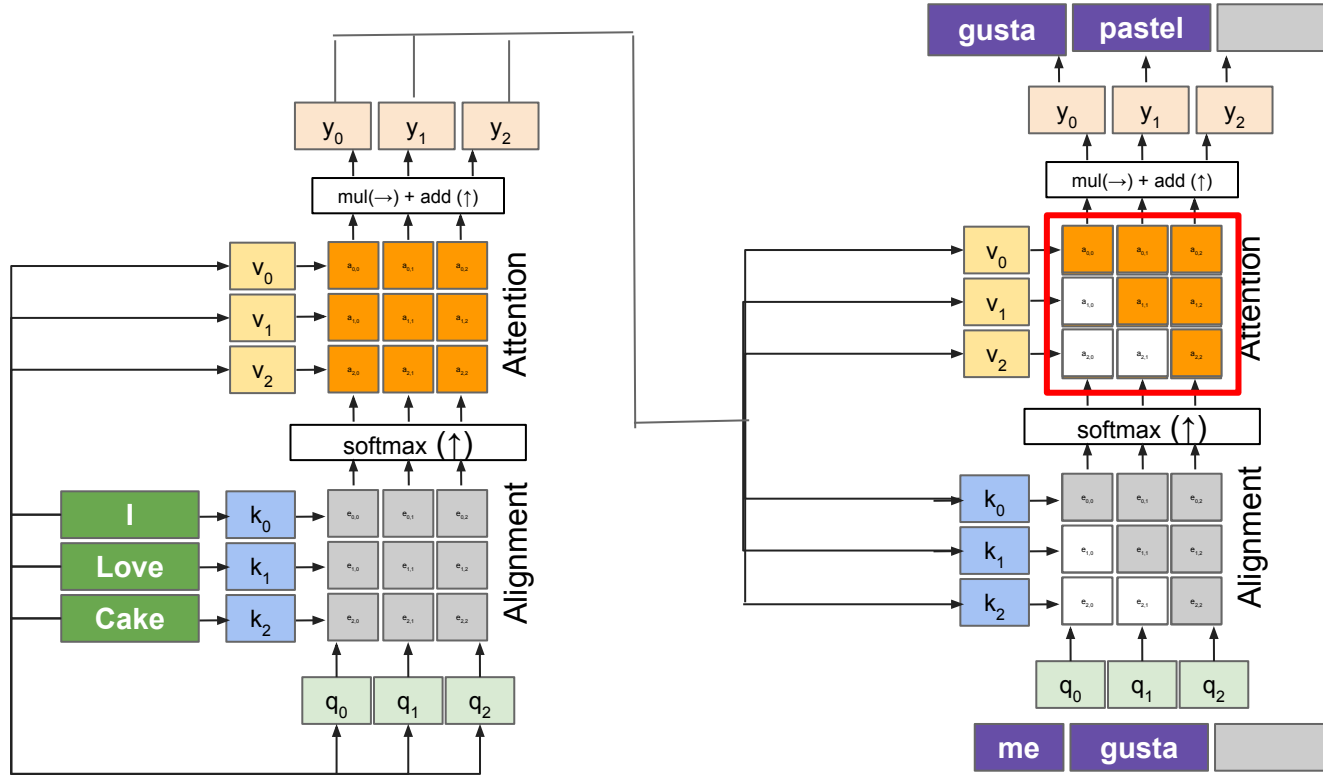
Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence



Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence



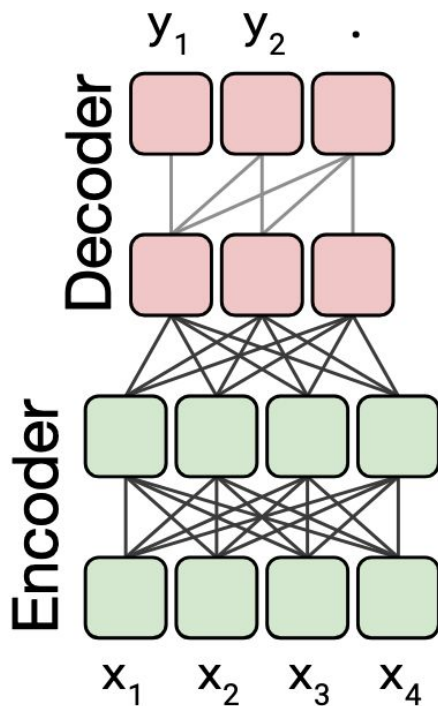
Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence



T5

[Raffel et al., 2019](#)

Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence



[Raffel et al., 2019](#)

Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence

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 .	<i>(original text)</i>
Deshuffling	party me for your to . last fun you inviting week Thank	<i>(original text)</i>
MASS-style Song et al. (2019)	Thank you <M> <M> me to your party <M> week .	<i>(original text)</i>
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>

“We include the language modeling objective due to its historic use as a pre-training objective...”

Image Source: Raffel et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. 2019

LLMs

Encoder Only:

I love cake

ELMO: Bi-directional next word prediction

BERT: Masked language modeling, Next Sentence Prediction

Decoder Only:

I love []

GPT: next token prediction

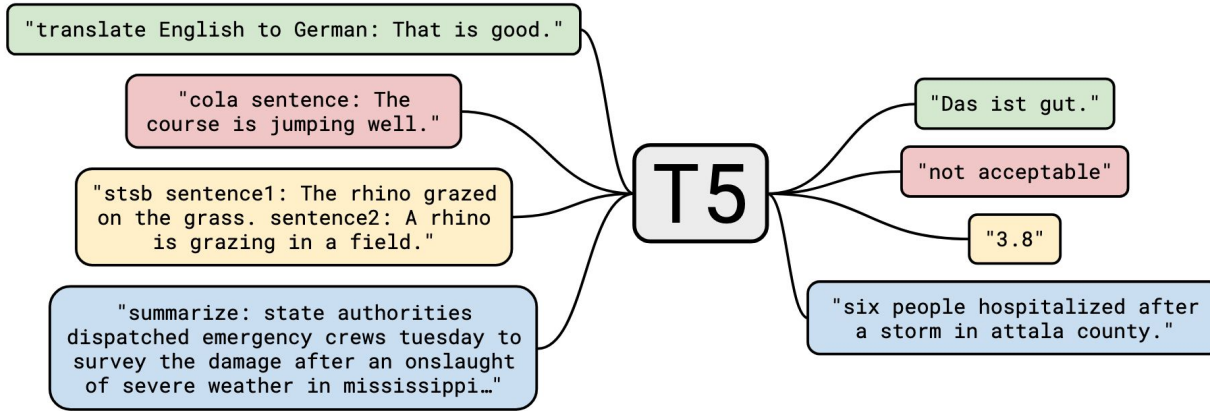
Encoder-Decoder:

I love cake

me gusta []

T5: Masked language modeling
(span corruption)

T5: The Real Takeaway



“our text-to-text framework allows us to directly apply the same model, objective, training procedure, and decoding process to every task we consider.”

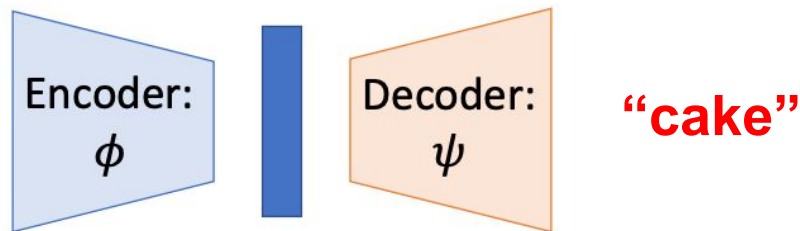
Made its way into how we use models today:
treat everything like a text-to-text task!

Image Source: Raffel et al. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. 2019

The old dynamic

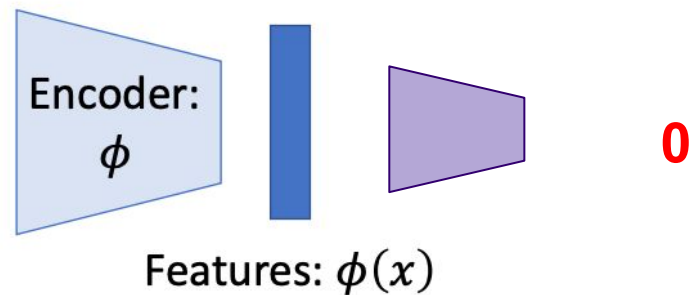
Step 1: Pretrain a network on a pretext task that doesn't require supervision

“I love ____”



Step 2: Transfer encoder to downstream tasks via linear classifiers, KNN, finetuning

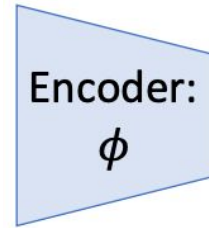
“I hated the movie”



The new dynamic

Step 1: Pretrain a network on a pretext task that doesn't require supervision

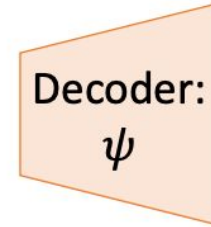
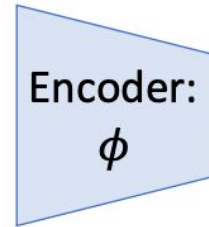
“I love _____”



“cake”

Step 2: Use the model out of the box in a creative way!

“The movie review ‘I hated the movie’ is _____”



“negative”

Enabled *scaling*

No task-specific architectures, or even task-specific losses.

Just train on everything as a next-token prediction task.

And if you train big enough models on big enough data, they get really good.



LLMs

Building LLMs: Pre-training objectives + architectures

- Encoder only
- Decoder only
- Encoder Decoder

Improving Models without Finetuning

Improving Models with Finetuning

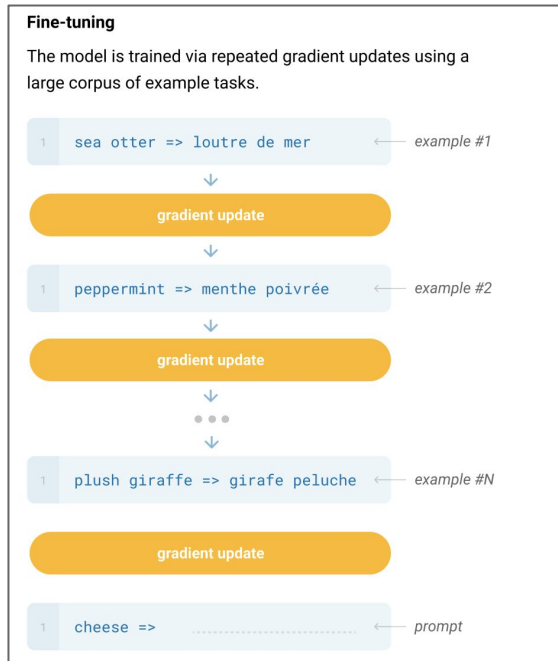
Improving models...

Oct 2019: T5: Treat everything like a text-to-text task!

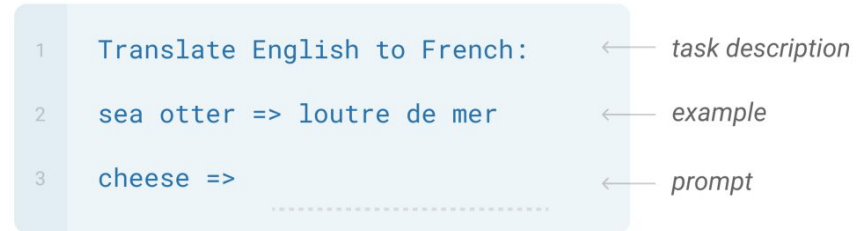
May 2020: GPT-3: We did that with a decoder model, trained on a TON of data, and it worked REALLY well.

Image Source: [Language Models are Few-Shot Learners, Brown et al](#)

Improving models... without finetuning.



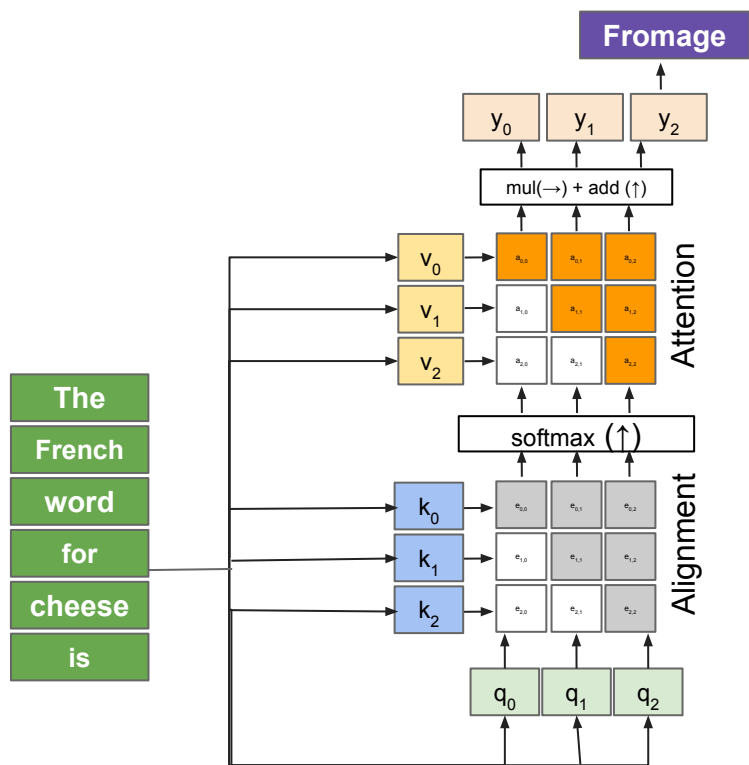
If you scale enough...



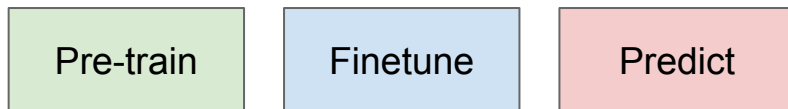
Language Models are Few-Shot Learners

Image Source: [Language Models are Few-Shot Learners, Brown et al](#)

Zero-shot, Few-shot



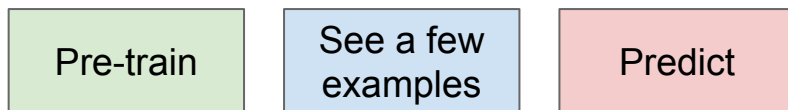
Usual method:



Zero-shot:



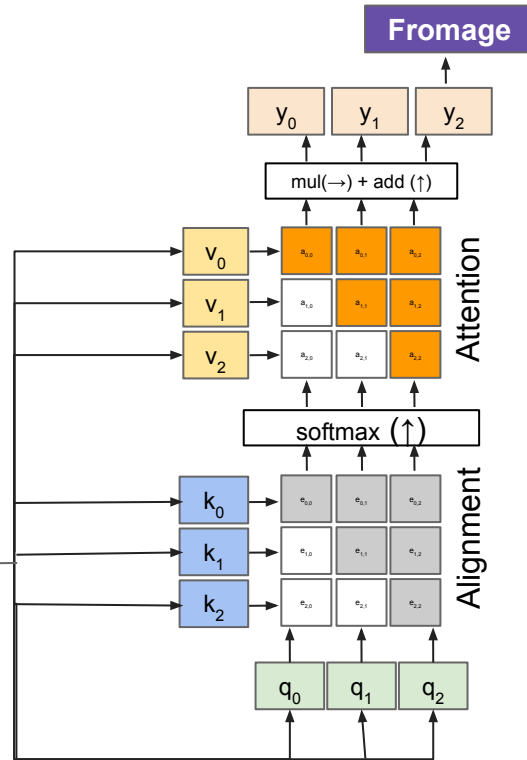
Few-shot:



1-shot

context

Translate English to French:
sea otter => loutre de mer
cheese =>



“In-Context Learning”

Image Source: [Language Models are Few-Shot Learners, Brown et al](#)

In-Context Learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

One-shot

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

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← example
3 cheese => ..... ← prompt
```

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
```

No training. No gradients.

Image Source: [Language Models are Few-Shot Learners, Brown et al](#)

In-Context Learning

With enough examples, you don't even need the prompt.

s.u!c/c!e.s s i/o/n = succession

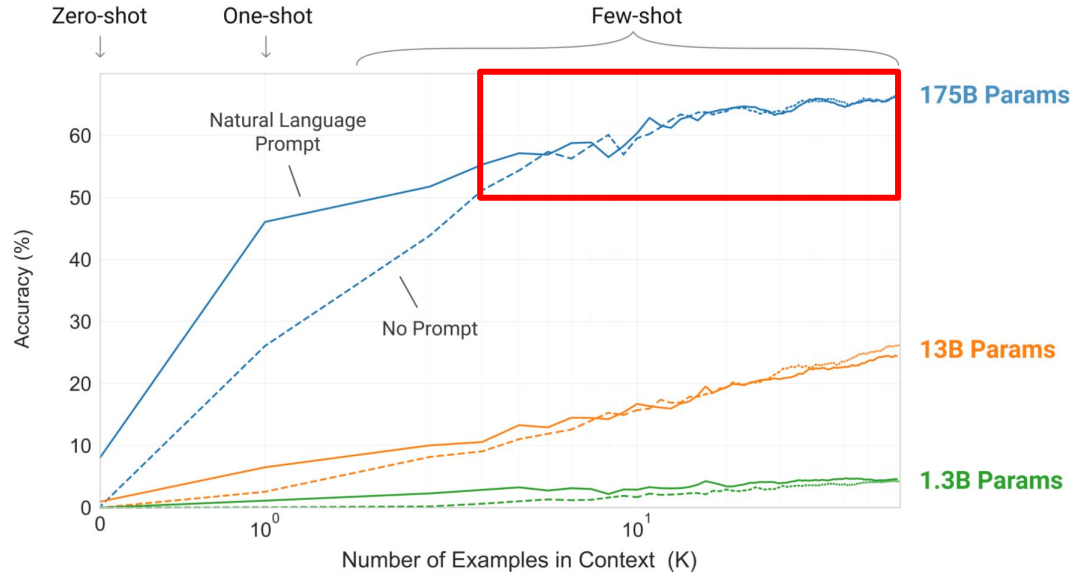


Image Source: [Language Models are Few-Shot Learners, Brown et al](#)

Benefits of scaling

1. Big models learn from examples better.

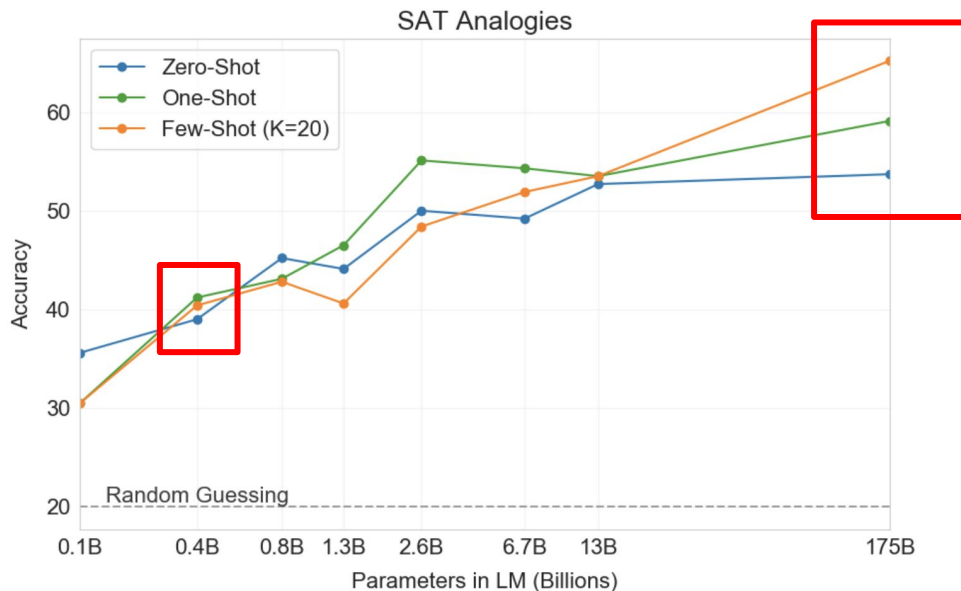
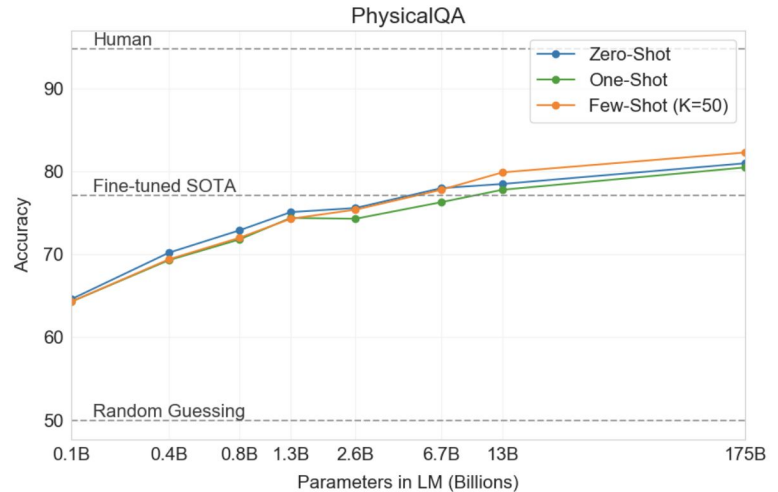
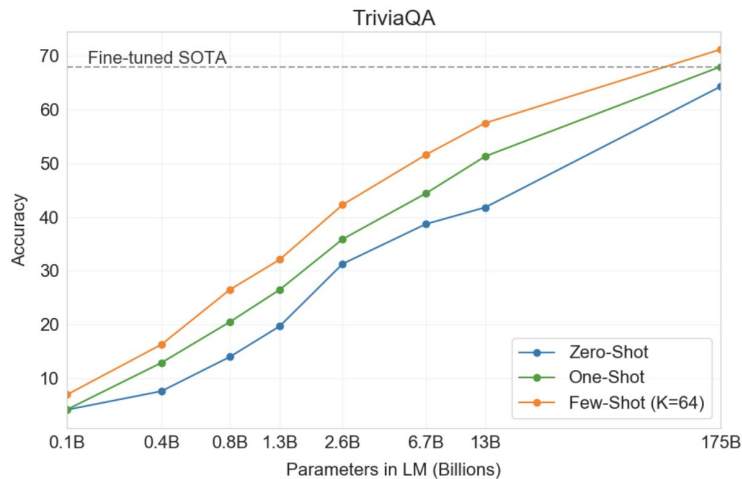


Image Source: [Language Models are Few-Shot Learners, Brown et al](#)

Benefits of scaling

2. Big models that see a few examples can even beat (smaller) finetuned models!



“Data Contamination”

Image Source: [Language Models are Few-Shot Learners, Brown et al](#)

GPT Results

	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	89.0	91.0	96.9	93.9	94.8	92.5
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0

	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	76.1	93.8	62.3	88.2	92.5	93.3
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1

Table 3.8: Performance of GPT-3 on SuperGLUE compared to fine-tuned baselines and SOTA. All results are reported on the test set. GPT-3 few-shot is given a total of 32 examples within the context of each task and performs no gradient updates.

Image Source: [Language Models are Few-Shot Learners, Brown et al](#)

GPT Results

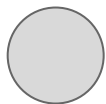
	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	89.0	91.0	96.9	93.9	94.8	92.5
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0

	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	76.1	93.8	62.3	88.2	92.5	93.3
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1

Table 3.8: Performance of GPT-3 on SuperGLUE compared to fine-tuned baselines and SOTA. All results are reported on the test set. GPT-3 few-shot is given a total of 32 examples within the context of each task and performs no gradient updates.

All GPT was trained to do was predict the next token, how is it so good???

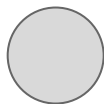
Scale!
(in model size)



BERT (Large)
340 million parameters

GPT - 3
175 billion parameters

Scale! (in data size)



BERT

3.3 Billion tokens¹

- All of english wikipedia
- 11,000 Books

GPT - 3

~300 billion tokens

- Common Crawl (Much of the internet)

¹<https://aclanthology.org/W19-4828.pdf>

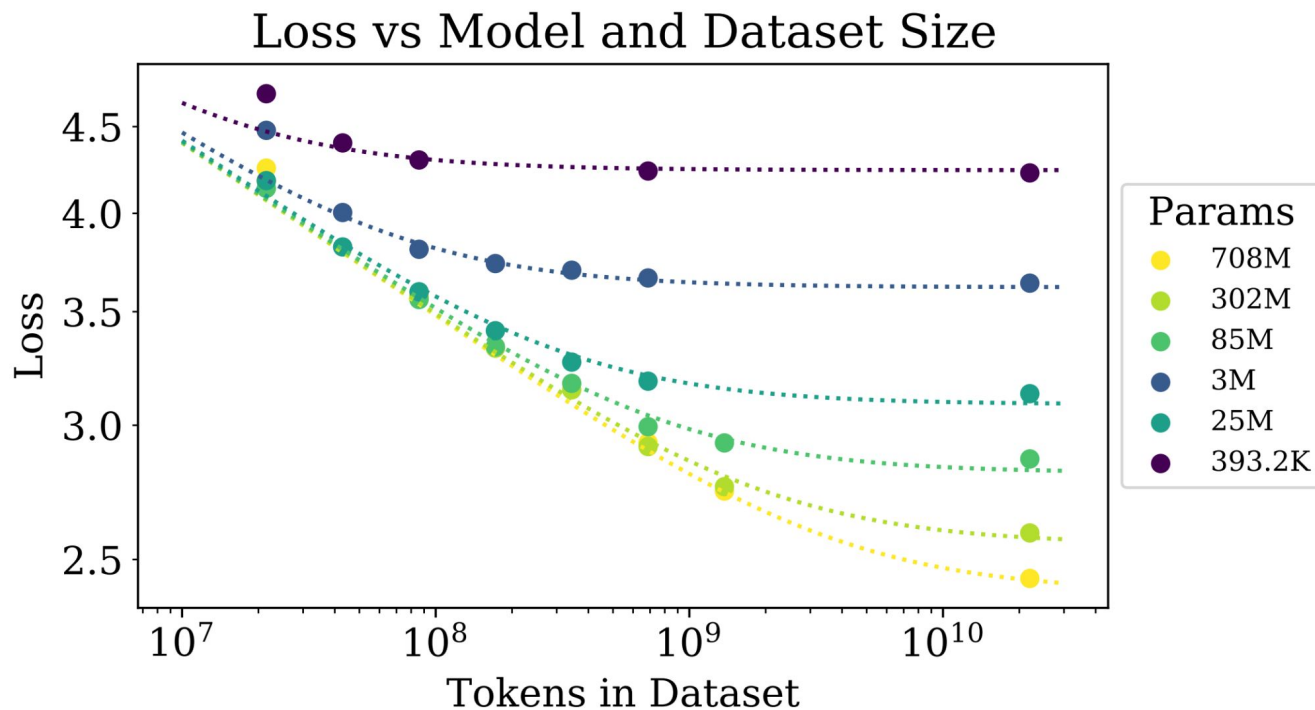


Image source: [Kaplan et al. Scaling Laws for Neural Language Models. 2020.](#)

The Bitter Lesson

Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

<http://www.incompleteideas.net/Incldeas/BitterLesson.html>

Inverse Scaling Prize

TL;DR: Win up to \$100,000 for finding an important task where larger language models do worse.

2022 11 tasks

Input

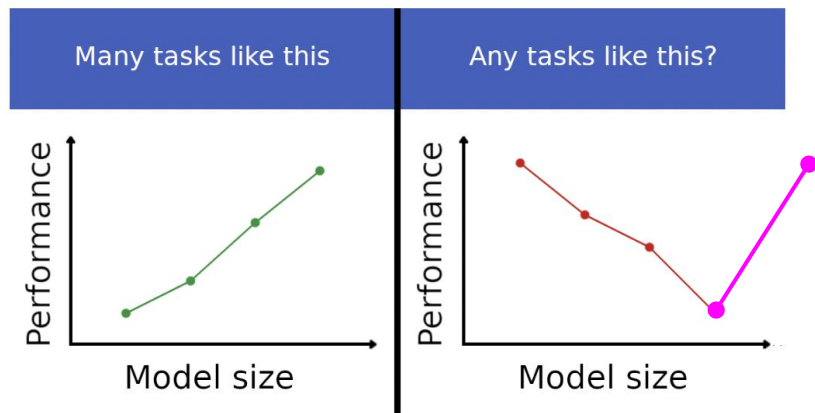
[...] Write a quote that ends in the word "heavy":
Absence makes the heart grow

Targets

{fonder, **heavy.**}

2023 Google: "Maybe you just didn't scale enough"

6 of 11 tasks: U-shaped scaling



"Inverse scaling can become U-shaped", Wei et al., 2023

Improving Models without Finetuning

1. Scaling data/model size → in-context learning “for free”
“emergent behavior”
2. Chain-of-Thought

Chain-of-Thought prompting

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: John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. How many hours a week does he spend taking care of dogs?

A:

GPT (or similar)

The answer is 50. ❌

	GSM8K
Finetuned GPT-3 175B	33%
Finetuned GPT-3 175B + verifier (prior SOTA)	55%
9–12 year olds (Cobbe et al., 2021)	60%
PaLM 540B: standard prompting	17.9%

Image Source: [Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.](#)

Chain-of-Thought prompting

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: John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. How many hours a week does he spend taking care of dogs?

A:

GPT (or similar)

	GSM8K
Finetuned GPT-3 175B	33%
Finetuned GPT-3 175B + verifier (prior SOTA)	55%
9–12 year olds (Cobbe et al., 2021)	60%
PaLM 540B: standard prompting	17.9%

Image Source: [Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.](#)

Chain-of-Thought prompting

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: John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. How many hours a week does he spend taking care of dogs?

A:

GPT (or similar)

John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. So that is $10 \times .5 = 5$ hours a day. $5 \text{ hours a day} \times 7 \text{ days a week} = 35 \text{ hours a week}$. The answer is 35 hours a week. ✓

	GSM8K
Finetuned GPT-3 175B	33%
Finetuned GPT-3 175B + verifier (prior SOTA) 9–12 year olds (Cobbe et al., 2021)	55% 60%
PaLM 540B: standard prompting	17.9%
PaLM 540B: chain of thought prompting	58.1%

Image Source: [Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.](#)

Chain-of-Thought prompting: Works for...

Math problems (GSM8K)

QUESTION: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?

MODEL ANSWER (CORRECT): James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. So he runs 60 meters x 3 sprints x 3 times a week. That is 60 meters x 9. The answer is 540. ✓

Symbolic Reasoning

QUESTION: Take the last letters of the words in “Waldo Schmidt” and concatenate them.

MODEL ANSWER (CORRECT): The last letter of “Waldo” is “o”. The last letter of “Schmidt” is “t”. Concatenating them is “ot”. So the answer is ot. ✓

Physical Reasoning

QUESTION: A coin is heads up. Andree flips the coin. Audrie does not flip the coin. Is the coin still heads up?

MODEL ANSWER (CORRECT): The coin was flipped by Andree. So the coin was flipped 1 time, which is an odd number. The coin started heads up, so after an odd number of flips, it will be tails up. So the answer is no. ✓

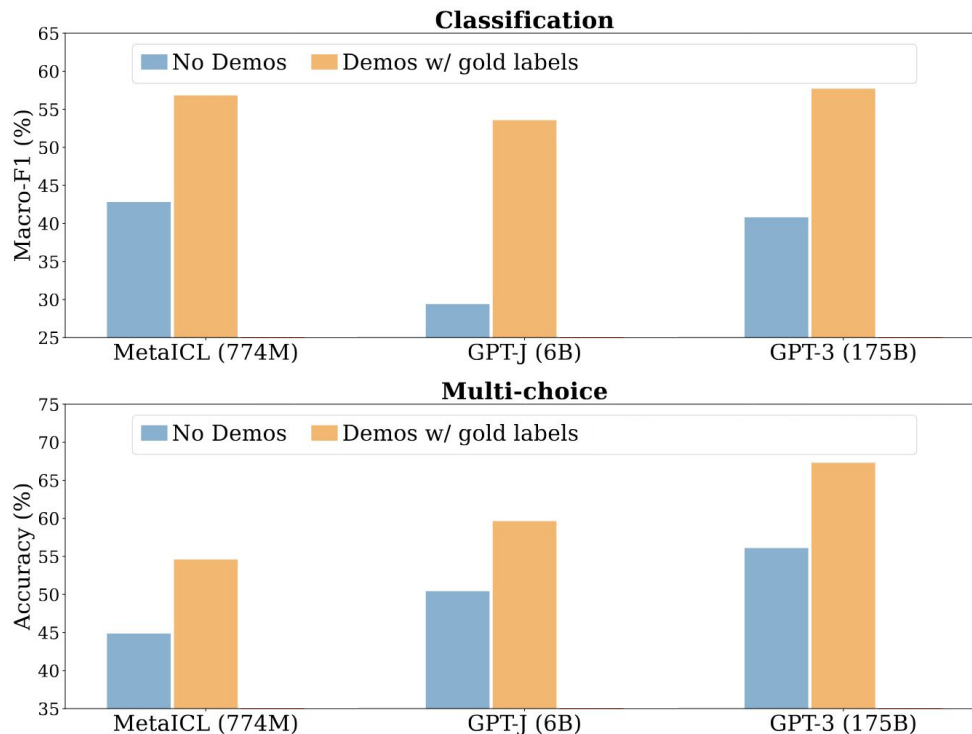
Chain-of-Thought results

	GSM8K	SVAMP	ASDiv	MAWPS
Standard prompting	6.5 \pm 0.4	29.5 \pm 0.6	40.1 \pm 0.6	43.2 \pm 0.9
Chain of thought prompting	14.3 \pm 0.4	36.7 \pm 0.4	46.6 \pm 0.7	57.9 \pm 1.5

	Commonsense			Symbolic	
	Date	Sports	SayCan	Concat	Coin
Standard prompting	21.5 \pm 0.6	59.5 \pm 3.0	80.8 \pm 1.8	5.8 \pm 0.6	49.0 \pm 2.1
Chain of thought prompting	26.8 \pm 2.1	85.8 \pm 1.8	91.7 \pm 1.4	77.5 \pm 3.8	99.6 \pm 0.3

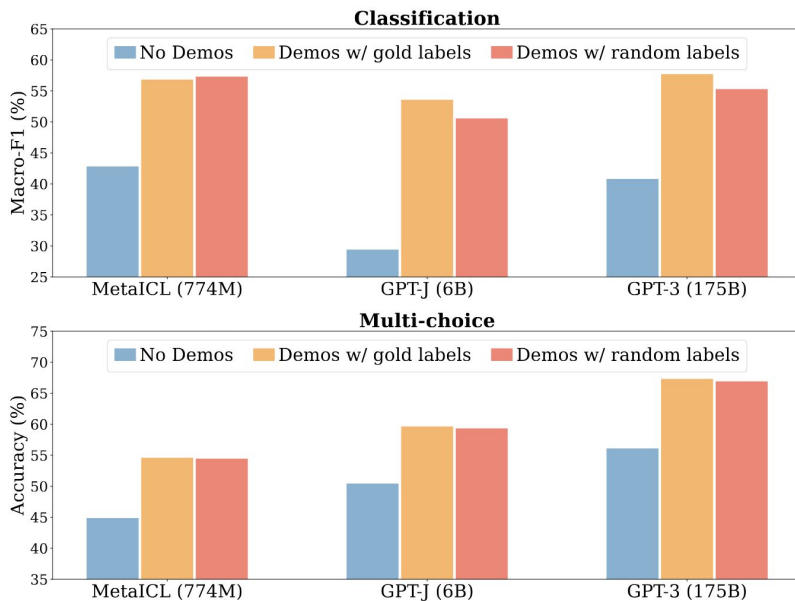
Image Source: [Wei et al. Chain of Thought Prompting Elicits Reasoning in Large Language Models. 2022.](#)

Quality of In-Context Learning examples



[Min et al., 2022](#)

Quality of In-Context Learning examples doesn't matter much!



Same for
Chain-of-Thought
(Madaan et al., 2022)

Never stop asking questions!!

[Min et al., 2022](#)

If quality doesn't matter much, can we drop examples?

(a) Few-shot

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. X

(b) Few-shot-CoT

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: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 X

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Image Source: [Kojima et al. Large Language Models are Zero-Shot Reasoners. 2022.](#)

“Let’s think step by step.”

	MultiArith	GSM8K
Zero-Shot	17.7	10.4
Few-Shot (2 samples)	33.7	15.6
Few-Shot (8 samples)	33.8	15.6
Zero-Shot-CoT	78.7	40.7
Few-Shot-CoT (2 samples)	84.8	41.3

But REALLY wins over zero-shot non-CoT

Loses by a small amount to few-shot CoT

[Kojima et al. Large Language Models are Zero-Shot Reasoners. 2022.](#)

Note: Explicit Chain-of-Thought is usually not required today

Models today are already **fine-tuned** on CoT traces, so they often do CoT on their own.

However, few-shot examples are still useful

→ CoT elicits a *behavior*, while few-shot provides useful information.

Improving Models without Finetuning

1. Scaling data/model size → in-context learning “for free”
“emergent behavior”
2. Chain-of-Thought
3. Self-consistency

Self Consistency

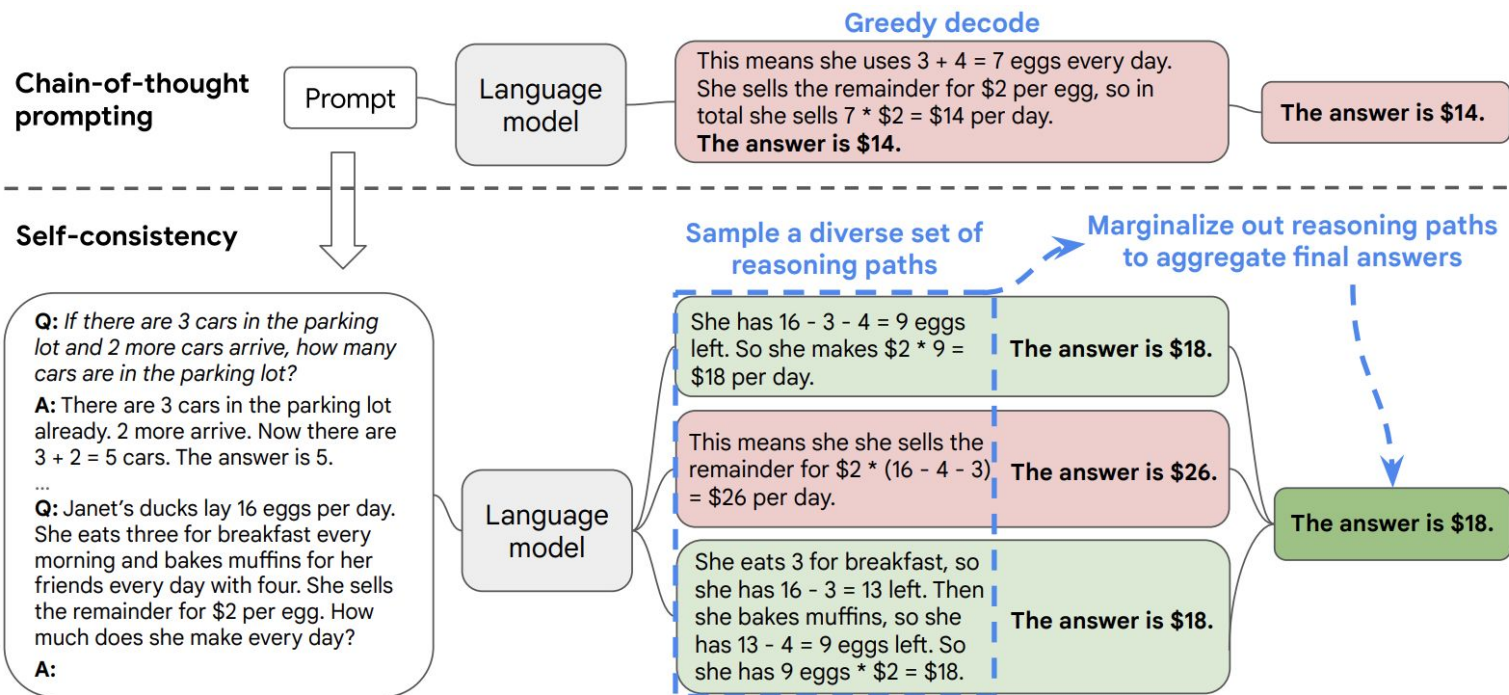


Image Source: [Xie et al. Self-Consistency Improves Chain of Thought Reasoning in Language Models. 2022.](#)

Self Consistency

	GSM8K	MultiArith	AQuA	SVAMP	CSQA	ARC-c
Greedy decode	56.5	94.7	35.8	79.0	79.0	85.2
Weighted avg (unnormalized)	56.3 \pm 0.0	90.5 \pm 0.0	35.8 \pm 0.0	73.0 \pm 0.0	74.8 \pm 0.0	82.3 \pm 0.0
Weighted avg (normalized)	22.1 \pm 0.0	59.7 \pm 0.0	15.7 \pm 0.0	40.5 \pm 0.0	52.1 \pm 0.0	51.7 \pm 0.0
Weighted sum (unnormalized)	59.9 \pm 0.0	92.2 \pm 0.0	38.2 \pm 0.0	76.2 \pm 0.0	76.2 \pm 0.0	83.5 \pm 0.0
Weighted sum (normalized)	74.1 \pm 0.0	99.3 \pm 0.0	48.0 \pm 0.0	86.8 \pm 0.0	80.7 \pm 0.0	88.7 \pm 0.0
Unweighted sum (majority vote)	74.4 \pm 0.1	99.3 \pm 0.0	48.3 \pm 0.5	86.6 \pm 0.1	80.7 \pm 0.1	88.7 \pm 0.1

Table 1: Accuracy comparison of different answer aggregation strategies on PaLM-540B.



[Xie et al. Self-Consistency Improves Chain of Thought Reasoning in Language Models. 2022.](#)

LLMs

Building LLMs: Pre-training objectives + architectures

- Encoder only
- Decoder only
- Encoder Decoder

Improving Models without Finetuning

Improving Models with Finetuning

Where are we?

May 2020: GPT-3

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← example
3 cheese => ..... ← prompt
```



“Help me write an email...”



How do we make language models useful to people?

Instruction Tuning: Turning GPT into ChatGPT

May 2020: GPT-3

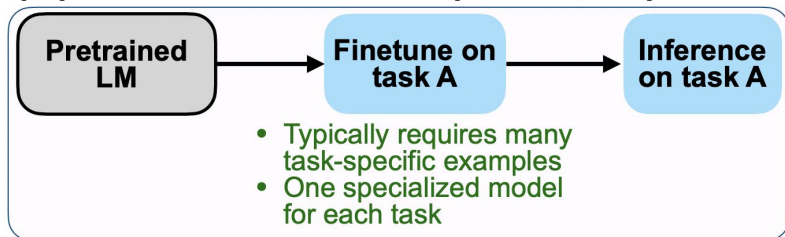
April, September 2021: Instruction Tuning

March 2022: InstructGPT

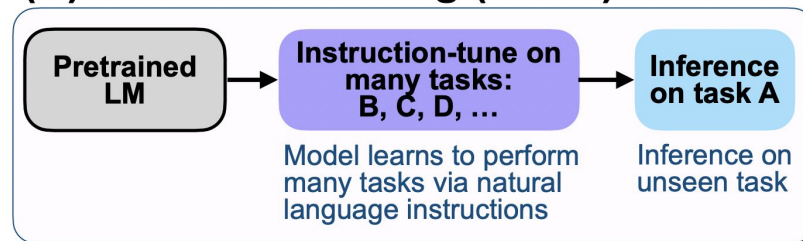
November 2022: ChatGPT

Instruction Tuning

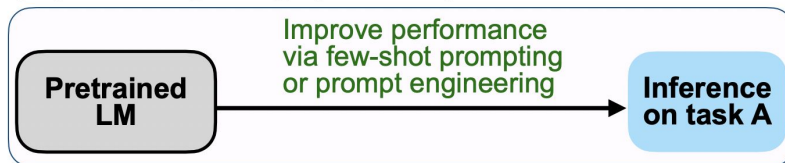
(A) Pretrain–finetune (BERT, T5)



(C) Instruction tuning (FLAN)

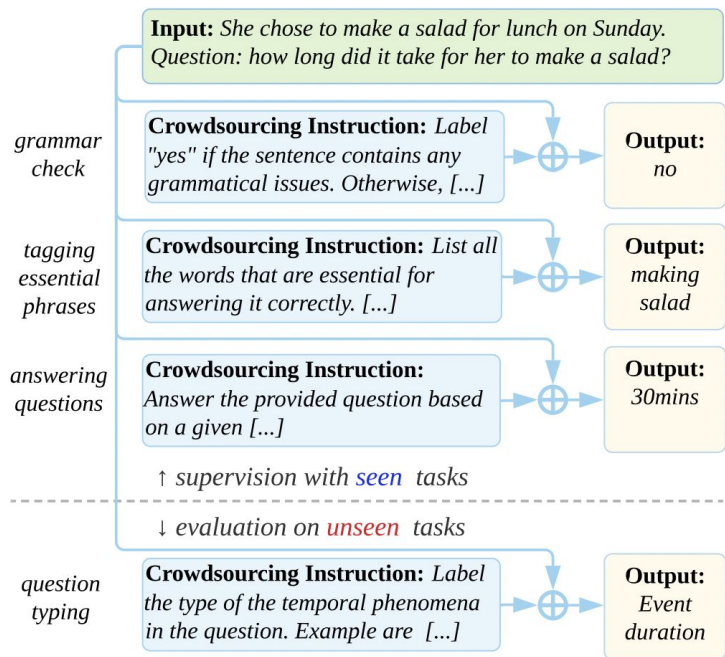


(B) Prompting (GPT-3)



“Finetuned Language Models are Zero-Shot Learners”, Wei et al., 2021

Instruction Tuning



“Cross-Task Generalization via Natural Language Crowdsourcing Instructions”, Mishra et al., 2021

Instruction Tuning

Great results...

Main takeaway: Teach your models to follow instructions!

One last finetuning step to actually get ChatGPT:

Reinforcement Learning from Human Feedback

Lectures 18 and 19!

Summary Slide

Encoder Only: Capture the meaning of an entire sequence

I love cake

ELMO: Bi-directional next word prediction

BERT: Masked language modeling, Next Sentence Prediction

Decoder Only: Generate text based on previously generated text

I love

GPT: next token prediction (autoregressive)

Encoder-Decoder: Generate text based on previously generated text and the meaning of a separate sequence

I love cake

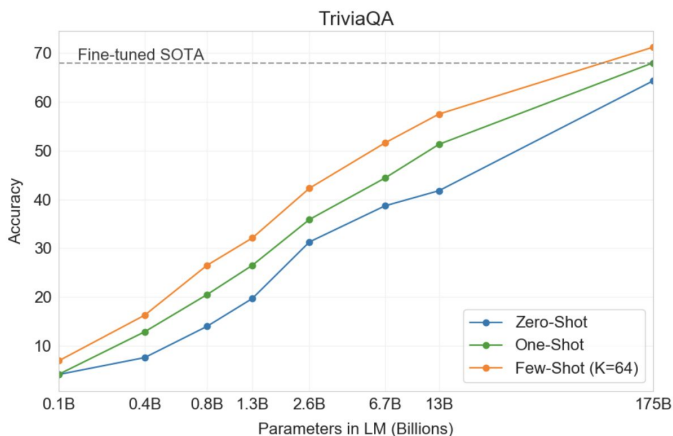
me gusta

T5: Masked language modeling (span corruption)

Summary

Improving models without finetuning

Scaling → in-context learning



Chain-of-Thought

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?

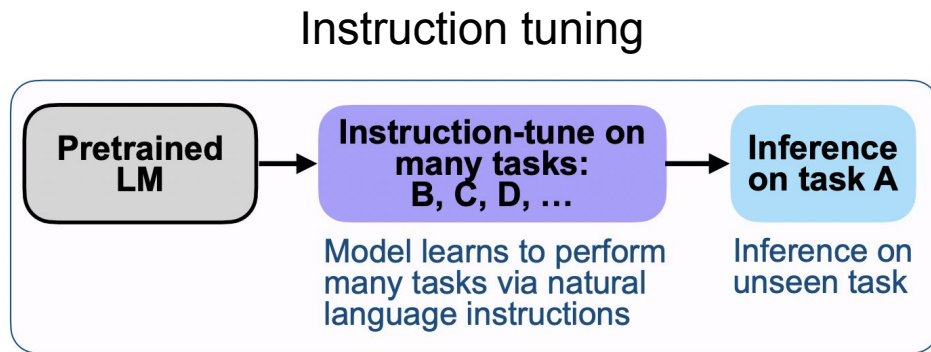
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Self-Consistency



Summary

Improving models with finetuning





<https://twitter.com/alirahimi0>