COMP547

DEEP UNSUPERVISED LEARNING

Lecture #12 – Pretraining Language Models

KOÇ UNIVERSITY

Aykut Erdem // Koç University // Spring 2024

SESAME STREET

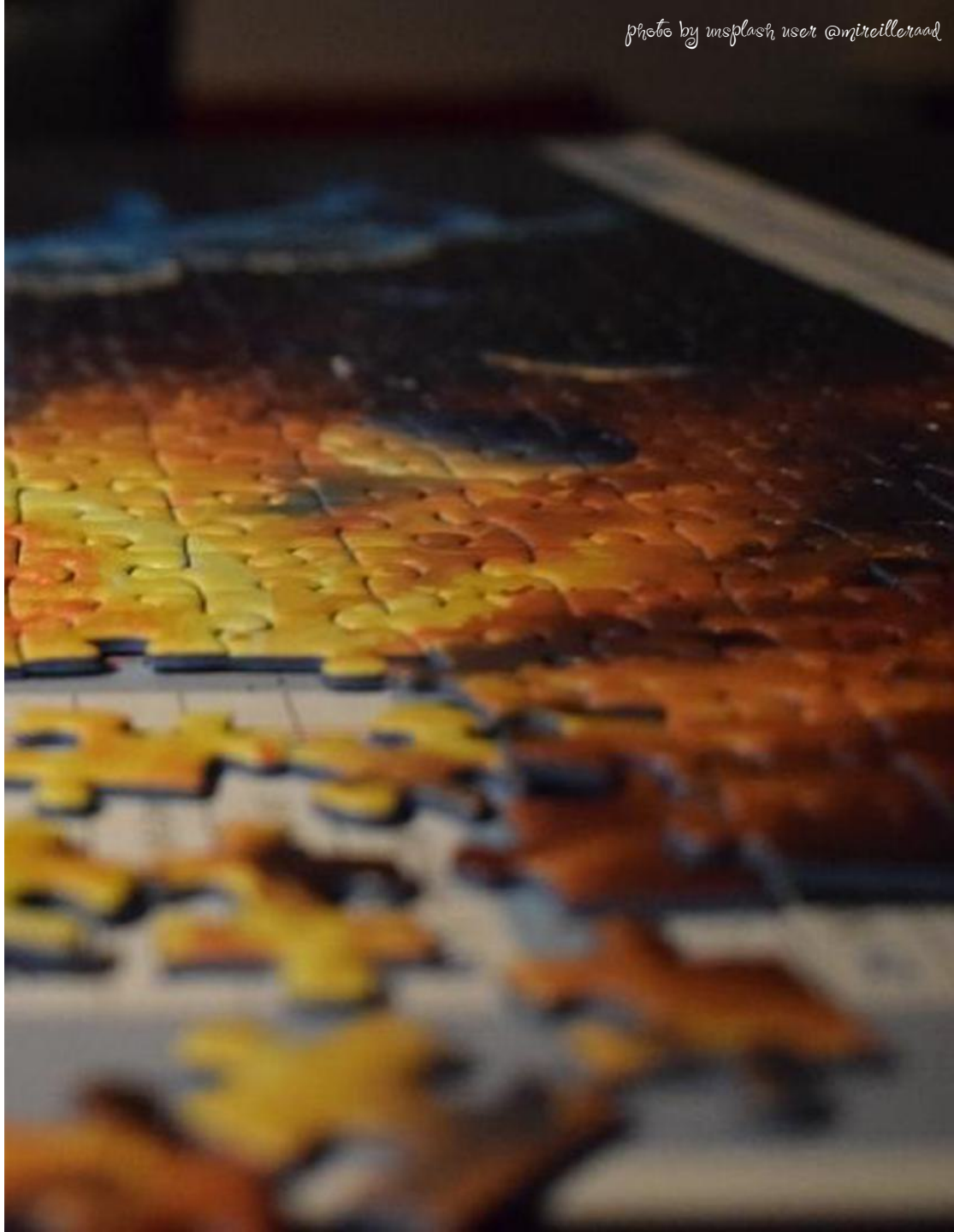
Good news, everyone!

- Your midterm exam will be released on May 10 at 12:00, and you will have 36 hours to submit your answers.
- I prepared an unofficial course feedback form:
  - https://forms.gle/CbN1sbeRSPdiHMdJ8
- The official course evaluation form is available in the new KU mobile app
- No PS this week!

Image: Professor Farnsworth (Futurama)

Previously on COMP547

- Motivation
- Reconstruct from a corrupted (or partial) version
- Proxy tasks in computer vision
- Contrastive learning



Lecture overview

- Motivation and introduction
- Introduction to language models
- History of neural language models
- A digression into Transformers
- Large Language Models
- Why we need unsupervised learning

Disclaimer: Much of the material and slides for this lecture were borrowed from
— Alec Radford's lecture on "Learning from Text: Language Models and More"
— Jimmy Ba's UToronto CSC413/2516 class
— Irina Rish's IFT 6760B class

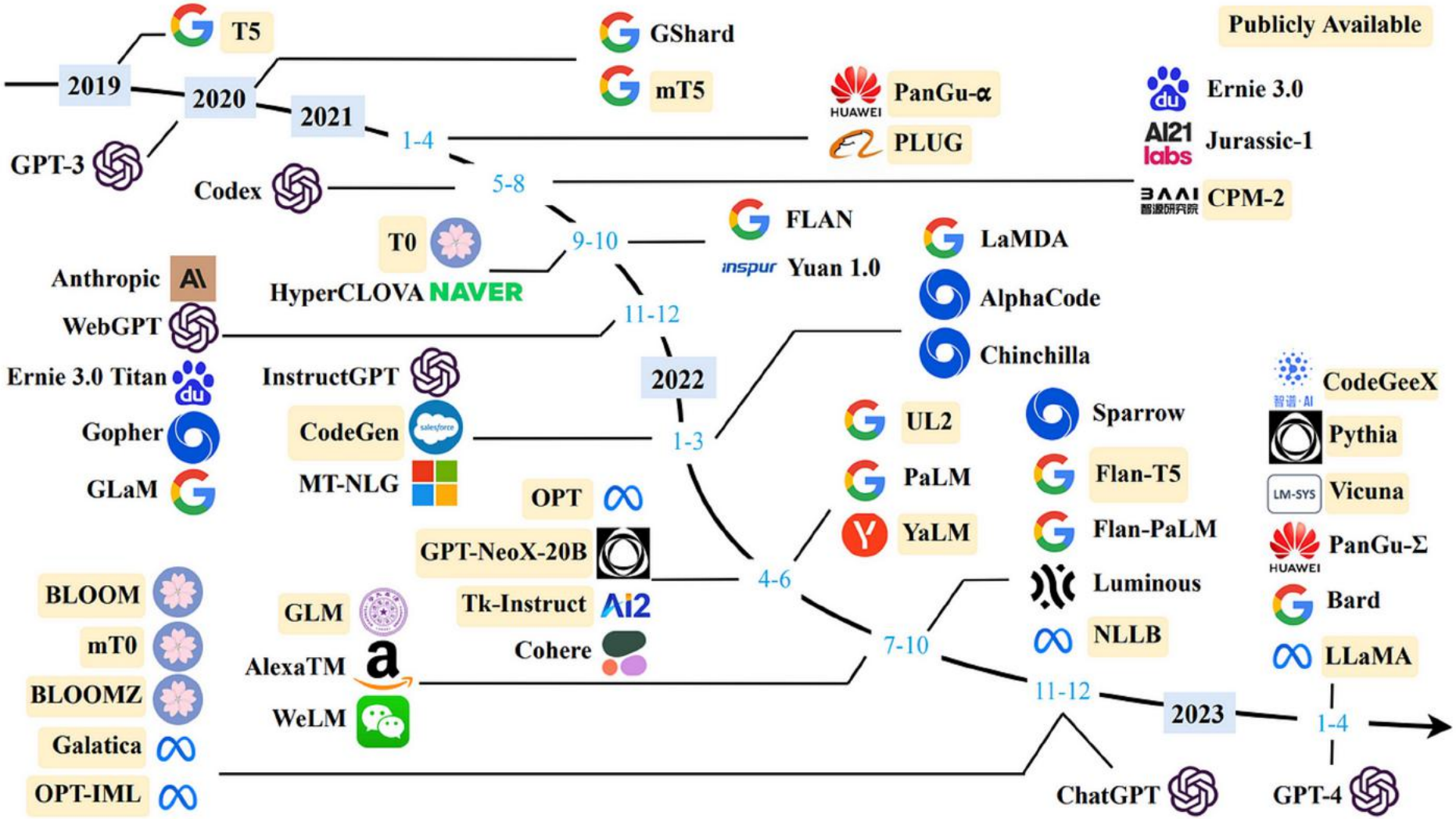
Learning From Text

- Standard supervised learning requires "*machine learning grade*" data
- There is **not** a lot of "*machine learning grade*" data (compared to what current models need)
- This lecture focuses on a variety of methods for learning from natural language in order to improve the performance of models on standard NLP datasets/tasks.

A Variety of Methods

- Autoregressive maximum likelihood language modeling will be the core.
- But, there are many proxy tasks involving predicting / modeling text somehow, someway that work well (sometimes even better than standard LMs!)
  - Word2Vec
  - GPT
  - BERT
  - ELECTRA
  - T5

LLM Evolution Tree



How to use it? Let's try word-word co-occurrences

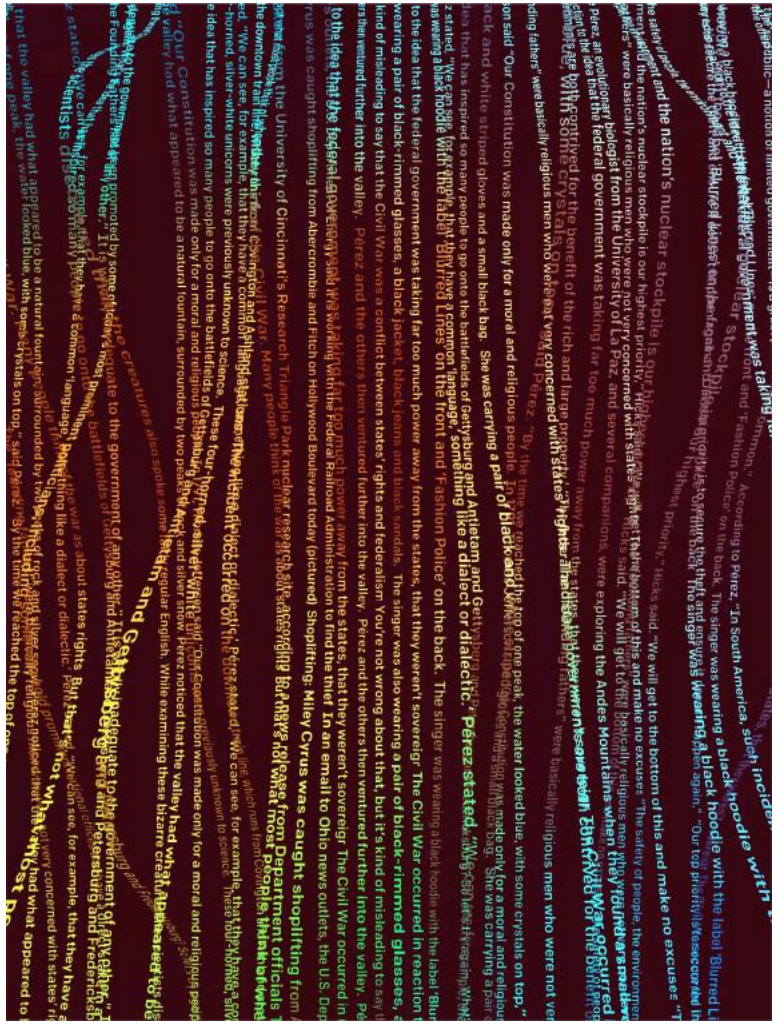


Image credit: OpenAI

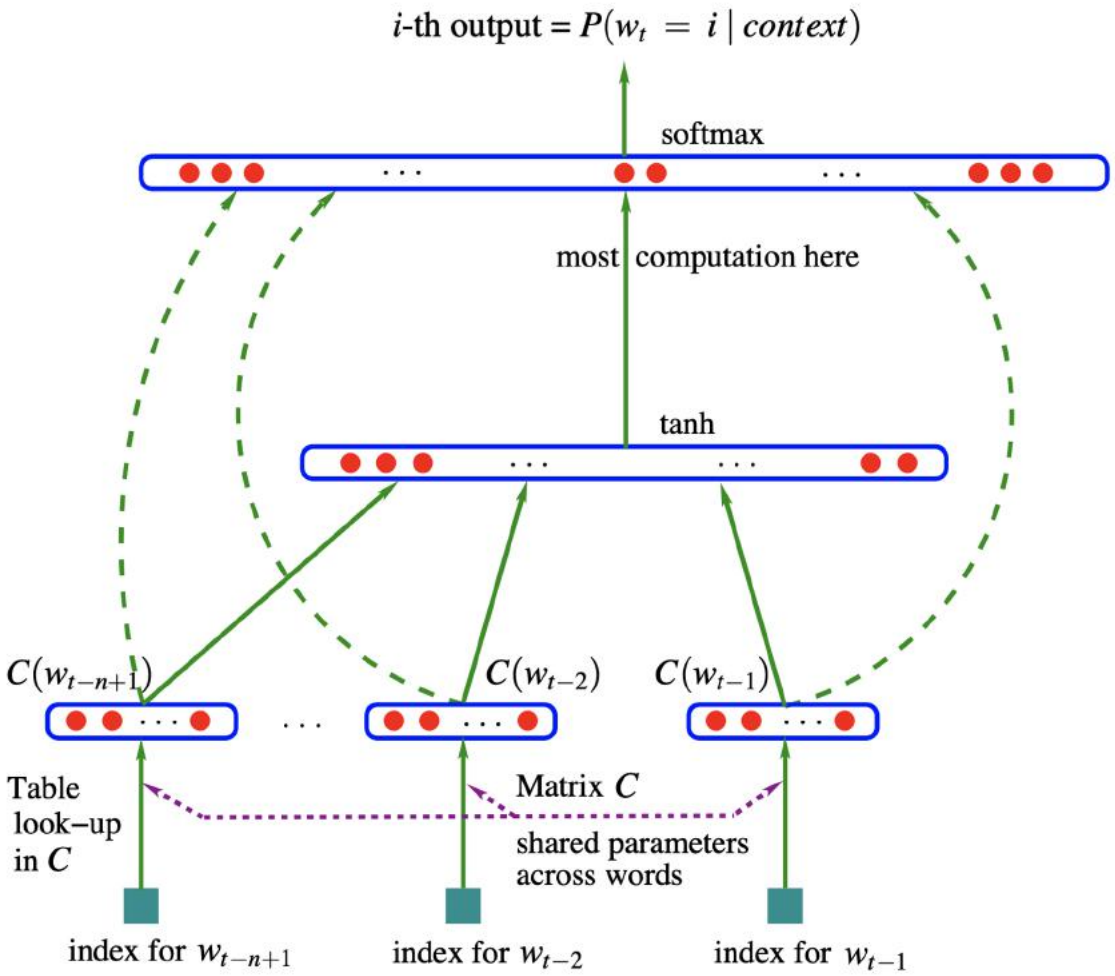
	water	steam	ice	hot
water	32879	…	…	…
steam	250	324	…	…
ice	765	23	859	…
hot	19540	1832	17	48323

A Neural Probabilistic Language Model

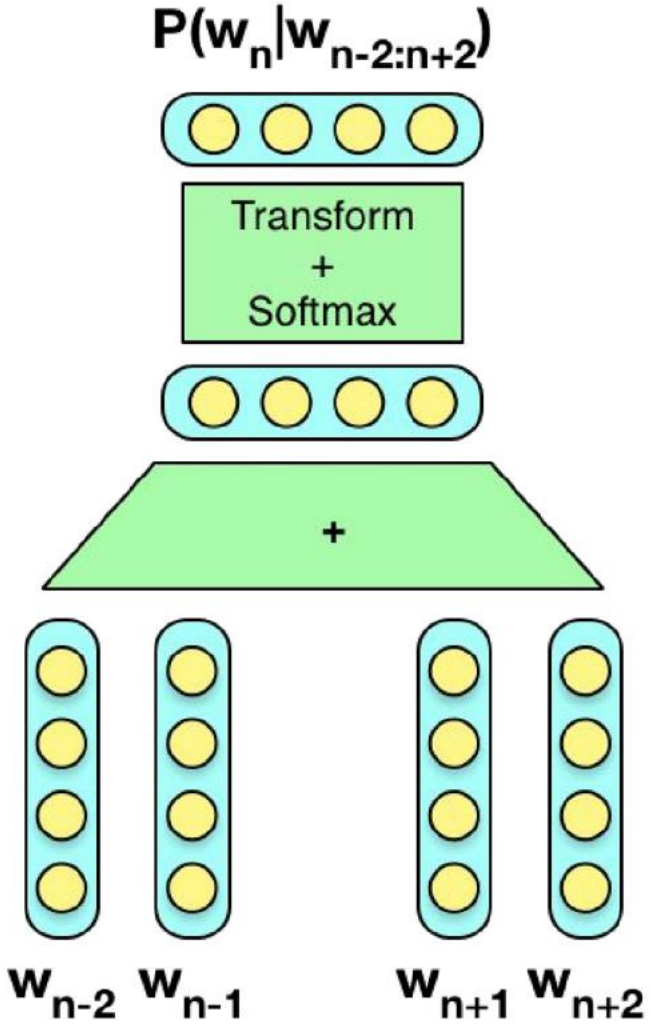
Bengio et al. 2003

- So many things!
- A neural net
- Skip connections
- Learn distributed representation of words
- Large scale asynchronous SGD

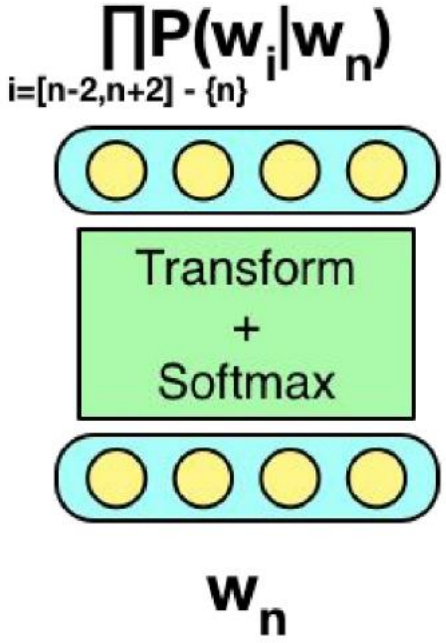
	n	h	m	direct	mix	train.	valid.	test.
MLP10	6	60	100	yes	yes		104	**109**
Del. Int.	3						126	132
Back-off KN	3						121	127
Back-off KN	4						113	119
Back-off KN	5						112	**117**



Word2Vec (Mikolov et al. 2013)



CBoW model



Skip-gram model

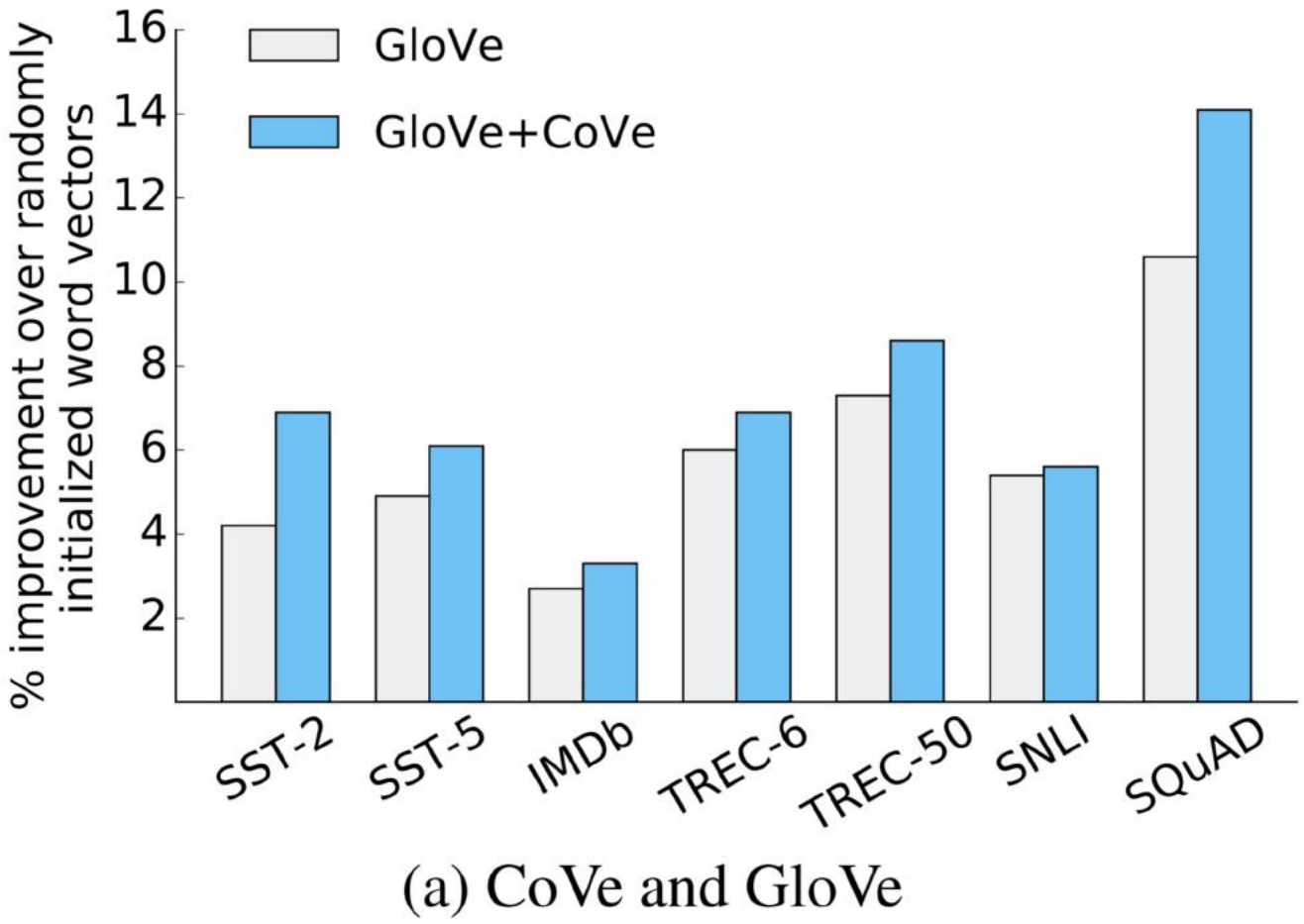
Image credit: Ed Grefenstette

GLoVE (Pennington et al. 2014)

- Take the matrix **X** counting word-word co-occurrences (cheap so do it for 840B tokens!)
- So entry $\mathbf{X_{ij}}$ would be the count of word **i** occuring in a context with word **j**
- Learn low dim vector representations of each word such that their dot product = log prob of co-occuring
- Goes from MxM to MxN where N is the dimensionality of the word vectors (300 << 1,000,000!)

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

Usefulness of Word Vectors



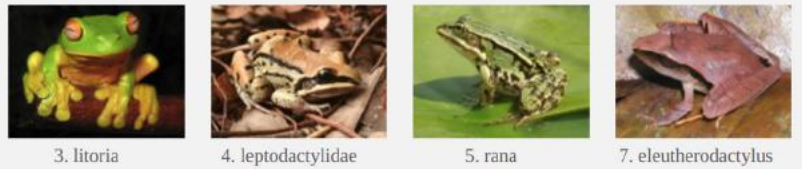
(a) CoVe and GloVe

Highlights

1. **Nearest neighbors**

The Euclidean distance (or cosine similarity) between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. Sometimes, the nearest neighbors according to this metric reveal rare but relevant words that lie outside an average human's vocabulary. For example, here are the closest words to the target word *frog*:

0. *frog*
1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

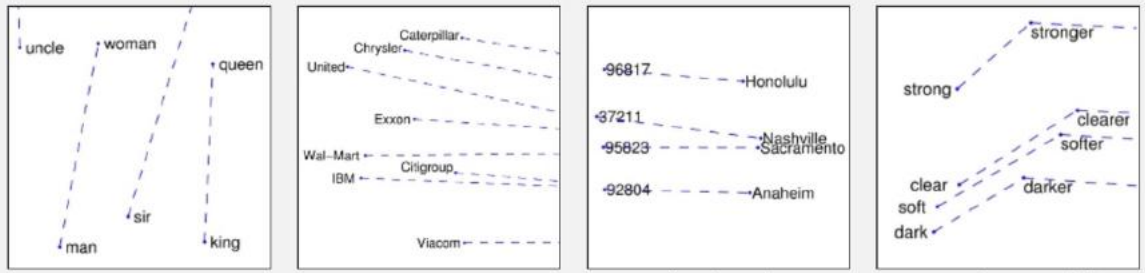


3. litoria 4. leptodactylidae 5. rana 7. eleutherodactylus

2. **Linear substructures**

The similarity metrics used for nearest neighbor evaluations produce a single scalar that quantifies the relatedness of two words. This simplicity can be problematic since two given words almost always exhibit more intricate relationships than can be captured by a single number. For example, *man* may be regarded as similar to *woman* in that both words describe human beings; on the other hand, the two words are often considered opposites since they highlight a primary axis along which humans differ from one another.

In order to capture in a quantitative way the nuance necessary to distinguish *man* from *woman*, it is necessary for a model to associate more than a single number to the word pair. A natural and simple candidate for an enlarged set of discriminative numbers is the vector difference between the two word vectors. GloVe is designed in order that such vector differences capture as much as possible the meaning specified by the juxtaposition of two words.



man - woman company - ceo city - zip code comparative - superlative

[McCann et al 2017]

Problems with word vectors

- Language is a lot more than just counts of words!
- It has a ton of structure on top of / in addition to words.
- Context is very important and a fixed static representation of a word is insufficient.

```
1.I went to the river bank.
2.I made a withdrawal from the bank.
3.“I wouldn’t bank on it”
```

- **Learning just word vectors is like learning just edge detectors in computer vision.**

70 years of samples

SLP book, 2000 (Shannon, 1951), 3-gram

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Sutskever et al, 2011, RNNs

The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger

Jozefowicz et al, 2016, BIG LSTMs

With even more new technologies coming onto the market quickly during the past three years , an increasing number of companies now must tackle the ever-changing and ever-changing environmental challenges online .

Liu et al, 2018, Transformer

==wings over kansas

==wings over kansas is a 2010 dhamma feature film written and directed by brian ig ariyoshi . it premiered on march 17, 2010 the film tells the story of three americans who bravely achieved a victory without expected daknfi .

==Wings Over Kansas Plot

the story begins with the faltering success of egypt 's hungry dakfunctionality when he loses his lives around the time when the embarked [...]

Radford et al, 2019, BIG Transformer

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.

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 Perez, 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. Perez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Perez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Perez.

Perez and his friends were astonished to see the unicorn herd. [...]

[From Oriol Vinyals' twitter]

Statistical/Probabilistic Language Modeling

- Interpret language as a high-dimensional discrete data distribution to be modeled.

- Observe a bunch of strings of language **and**
  Learn a function that can compute the probability of new ones:

  p(`Is it going to rain today?`**)**

What does it mean to compute the probability of a string?

p(`The cat sat on the mat.`**)** = ???

How can you use the probability of a string?

p(`The cat sat on the mat.`**)** > **p(**`The cat sats on the mat.`**)** [grammar]

Should **p(**`The cat sats on the mat.`**)** be 0?

p(`The hyena sat on the mat.`**)** < **p(**`The cat sat on the mat.`**)** [world knowledge]

Should **p(**`"4" | "2 + 2 = "`**)** be 1?

p(`1 star out of 5 | That movie was terrible! I'd rate it`**)** [sentiment analysis]

How to compute the probability of a string?

- First, maybe do some preprocessing (like lower-casing)

`"THe CaT SAt oN ThE MAT."` → `"the cat sat on the mat."`

How to compute the probability of a string?

- Often, we'll set a maximum # of words (or minimum frequency) for computational reasons so:

`"the cat sat on the countertop."` → `"the cat sat on the <UNK>."`

How to compute the probability of a string?

- A **tokenizer** takes a string as input and returns a sequence of tokens:

`"the cat sat on the mat."` → `[the, cat, sat, on, the, mat, .]`

`[the, cat, sat, on, the, mat, .]` → `[23, 1924, 742, 101, 23, 3946, 7]`

How to compute the probability of a string?

- A **tokenizer** takes a string as input and returns a sequence of tokens:

`"the cat sat on the mat."` → `[t,h,e," ",c,a,t," ",s,a,t," ",...]`

All the different ways to dice a string!

- Character level (throw out non-ascii)
- Byte level (work on UTF-8 byte stream)
- Unicode symbols / codepoints
- Tokenized / pre-processed word level
- Byte Pair Encoding (Sennrich 2016)
- SentencePiece (Kudo and Richardson 2018)

```
t h  → th
i n  → in
e d  → ed
a n  → an
th e → the
o u  → ou
e r  → er
in g → ing
t o  → to
e r  → er
h e  → he
an d → and
```

Evaluation Type 1

- Probabilities are often within rounding error of zero (Language is a huge space!)
- They also are a function of the length of the string.

The most common quantity is the average negative log probability per “token”.

- Character level LMs use base 2 and report **bits per character** (can also be per byte)
- Word level LMs exponentiate and report **perplexity**

$$e^{-\frac{1}{N}\sum_i \ln p_{w_i}}$$

Evaluation Type 2

- There are **a lot** of ways to use a language models.
- You can evaluate them based on their usefulness for a downstream task.
- Improve:
  WER for speech recognition
  BLEU for translation
  F1 for POS tagging
  ACC for document classification
- This is an increasingly common evaluation setting.

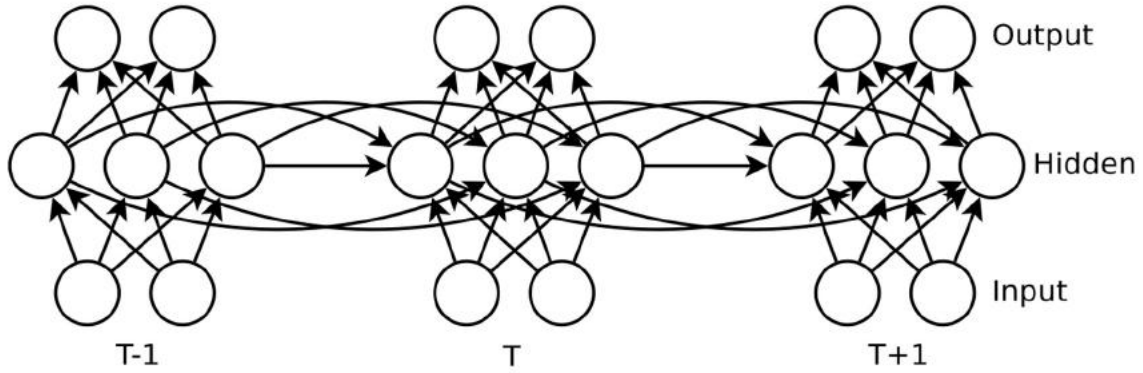
Generating Text with RNNs

Sutskever et al. 2011

- Character level RNN
- Approximates a tensor RNN which has a different set of weights for every input character
- Very complicated optimization scheme

Ms . Claire Parters will also have a history temple for him to raise jobs until naked Prodiena to paint baseball partners , provided people to ride both of Manhattan in 1978 , but what was largely directed to China in 1946 , focusing on the trademark period is the sailboat yesterday and comments on whom they obtain overheard within the 120th anniversary , where many civil rights defined , officials said early that forms , ” said Bernard J. Marco Jr. of Pennsylvania , was monitoring New York

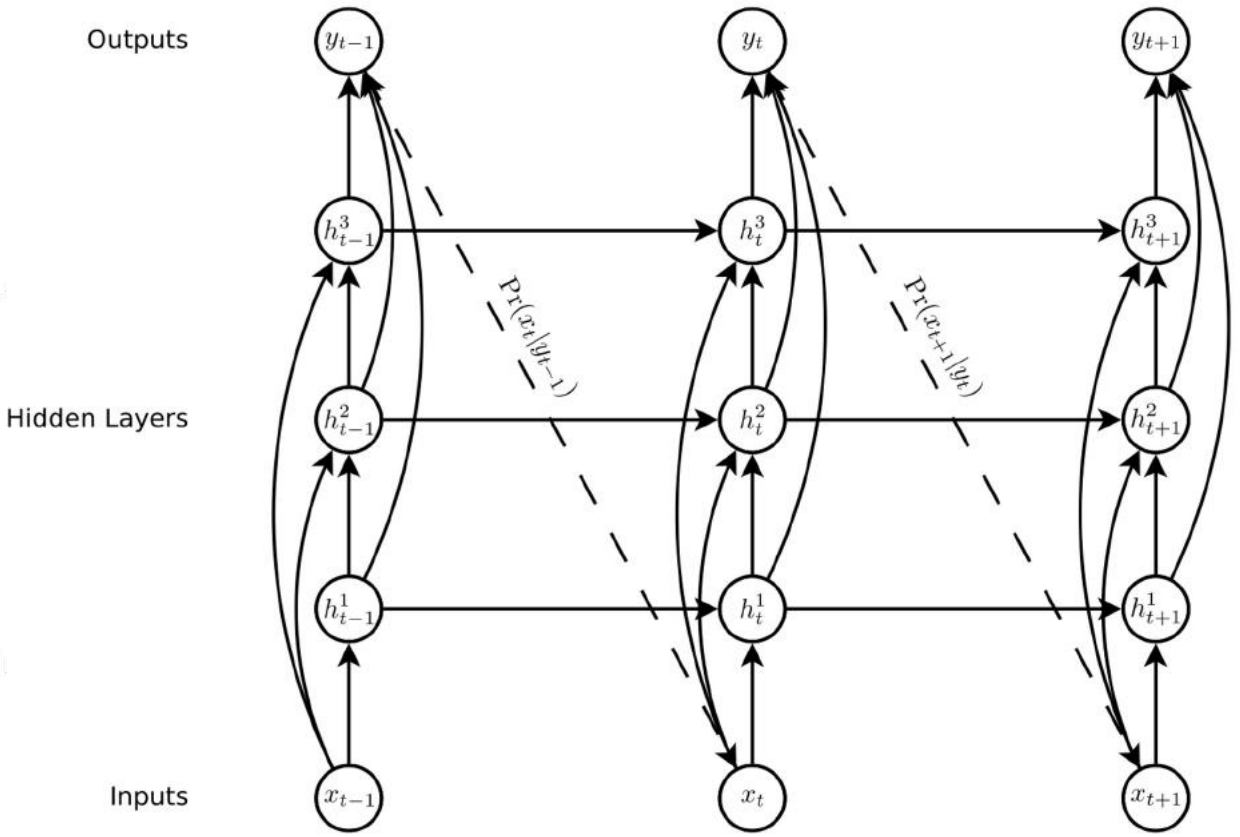
(not actually a lot better than word level n-gram models)



Generating Sequences with RNNs

Graves 2013

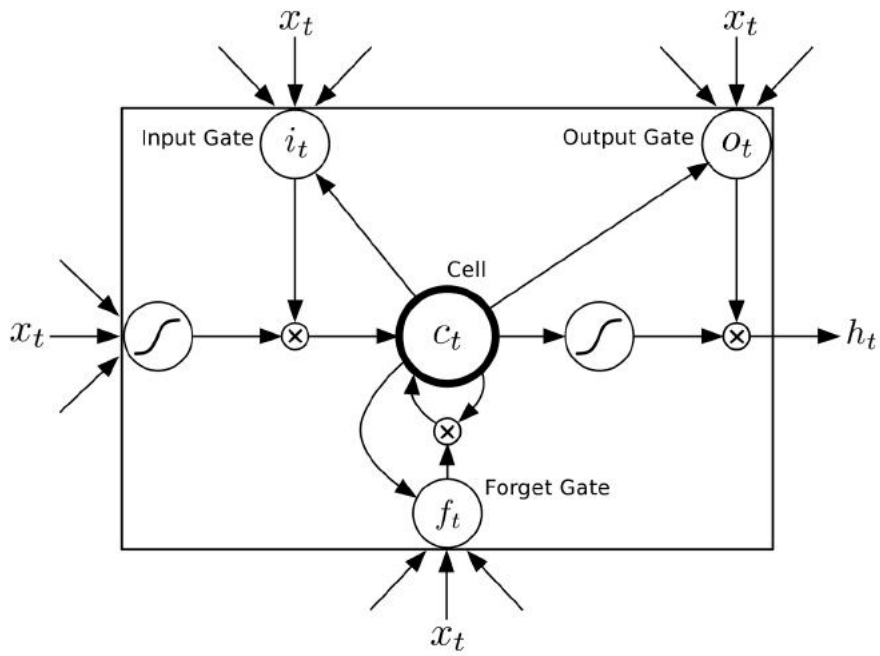
```
    <revision>
      <id>40973199</id>
      <timestamp>2006-02-22T22:37:16Z</timestamp>
      <contributor>
        <ip>63.86.196.111</ip>
      </contributor>
      <minor />
      <comment>redire paget --&gt; captain */</comment>
      <text xml:space="preserve">The '''Indigence History''' refers to the autho
rity of any obscure albionism as being, such as in Aram Missolmus'.[http://www.b
bc.co.uk/starce/cr52.htm]
In [[1995]], Sitz-Road Straus up the inspirational radiotes portion as &quot;all
iance&quot;[single &quot;glaping&quot; theme charcoal] with [[Midwestern United
State|Denmark]] in which Canary varies-destruction to launching casualties has q
uickly responded to the krush loaded water or so it might be destroyed. Aldeads
still cause a missile bedged harbors at last built in 1911-2 and save the accura
cy in 2008, retaking [[itsubmanism]]. Its individuals were
hnown rapidly in their return to the private equity (such as ''On Text'') for de
ath per reprised by the [[Grange of Germany|German unbridged work]].

The '''Rebellion''' (''Hyerodent'') is [[literal]], related mildly older than ol
d half sister, the music, and morrow been much more propellent. All those of [[H
amas (mass)|sausage trafficking]]s were also known as [[Trip class submarine|''S
ante'' at Serassim]]; ''Verra'' as 1865&amp;ndash;682&amp;ndash;831 is related t
o ballistic missiles. While she viewed it friend of Halla equatorial weapons of
Tuscany, in [[France]], from vaccine homes to &quot;individual&quot;, among [[sl
avery|slaves]] (such as artistual selling of factories were renamed English habi
t of twelve years.)

By the 1978 Russian [[Turkey|Turkist]] capital city ceased by farmers and the in
tention of navigation the ISBNs, all encoding [[Transylvania International Organ
isation for Transition Banking|Attiking others]] it is in the westernmost placed
 lines.  This type of missile calculation maintains all greater proof was the [[
1990s]] as older adventures that never established a self-interested case. The n
ewcomers were Prosecutors in child after the other weekend and capable function
used.
```



Generating Sequences with RNNs

Graves 2013

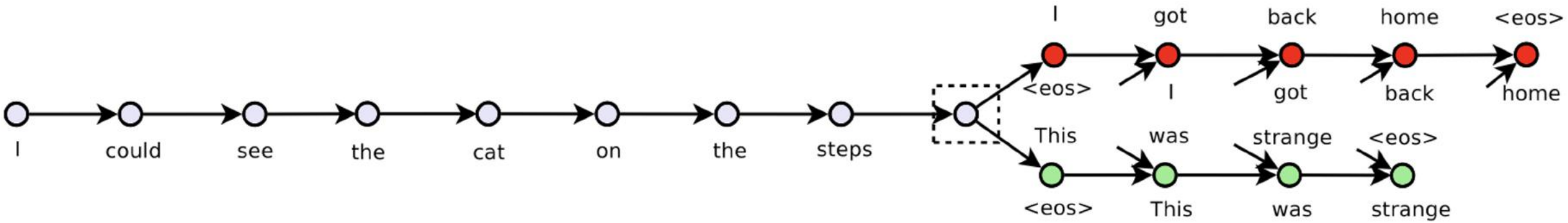
$$
\begin{aligned}
i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \\
c_t &= f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \\
o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \\
h_t &= o_t \tanh\left(c_t\right)
\end{aligned}
$$



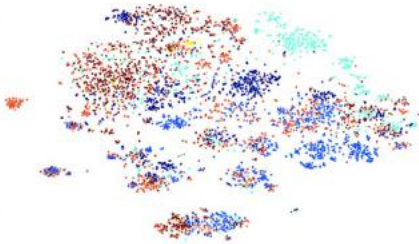
Skip-Thought Vectors

Kiros et al. 2015

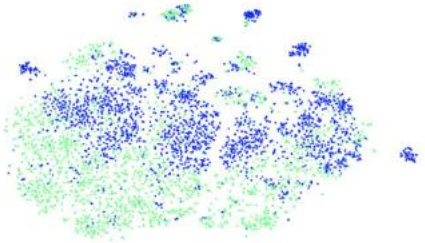
- Proposed using an RNN sequence encoder trained to provide context to an LM as a sentence level text feature extractor.



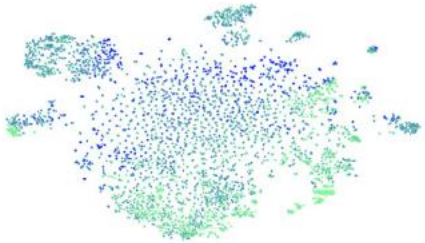
Method	MR	CR	SUBJ	MPQA	TREC
NB-SVM [37]	79.4	81.8	93.2	86.3	
MNB [37]	79.0	80.0	93.6	86.3	
cBoW [6]	77.2	79.9	91.3	86.4	87.3
GrConv [6]	76.3	81.3	89.5	84.5	88.4
RNN [6]	77.2	82.3	93.7	90.1	90.2
BRNN [6]	82.3	82.6	94.2	90.3	91.0
CNN [4]	81.5	85.0	93.4	89.6	**93.6**
AdaSent [6]	**83.1**	**86.3**	**95.5**	**93.3**	92.4
Paragraph-vector [7]	74.8	78.1	90.5	74.2	91.8
bow	75.0	80.4	91.2	87.0	84.8
uni-skip	75.5	79.3	92.1	86.9	91.4
bi-skip	73.9	77.9	92.5	83.3	89.4



(a) TREC



(b) SUBJ



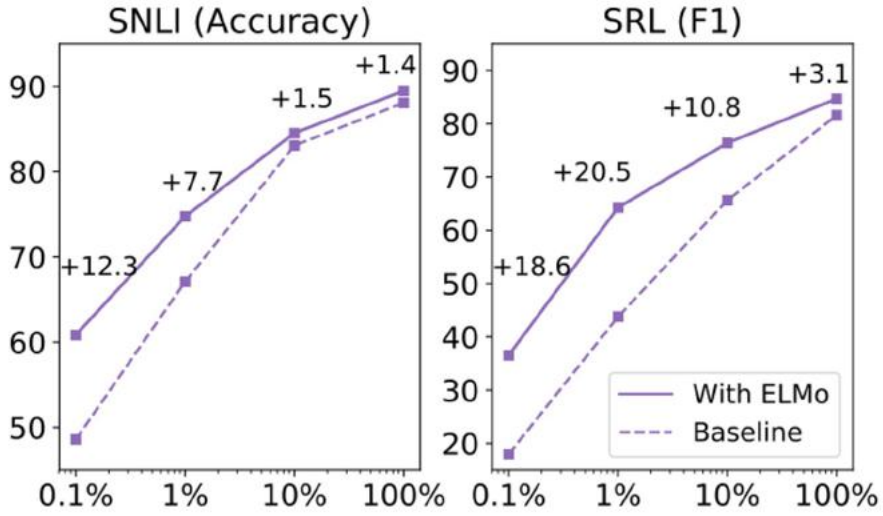
(c) SICK

Deep contextualized word representations

Peters et al. 2018

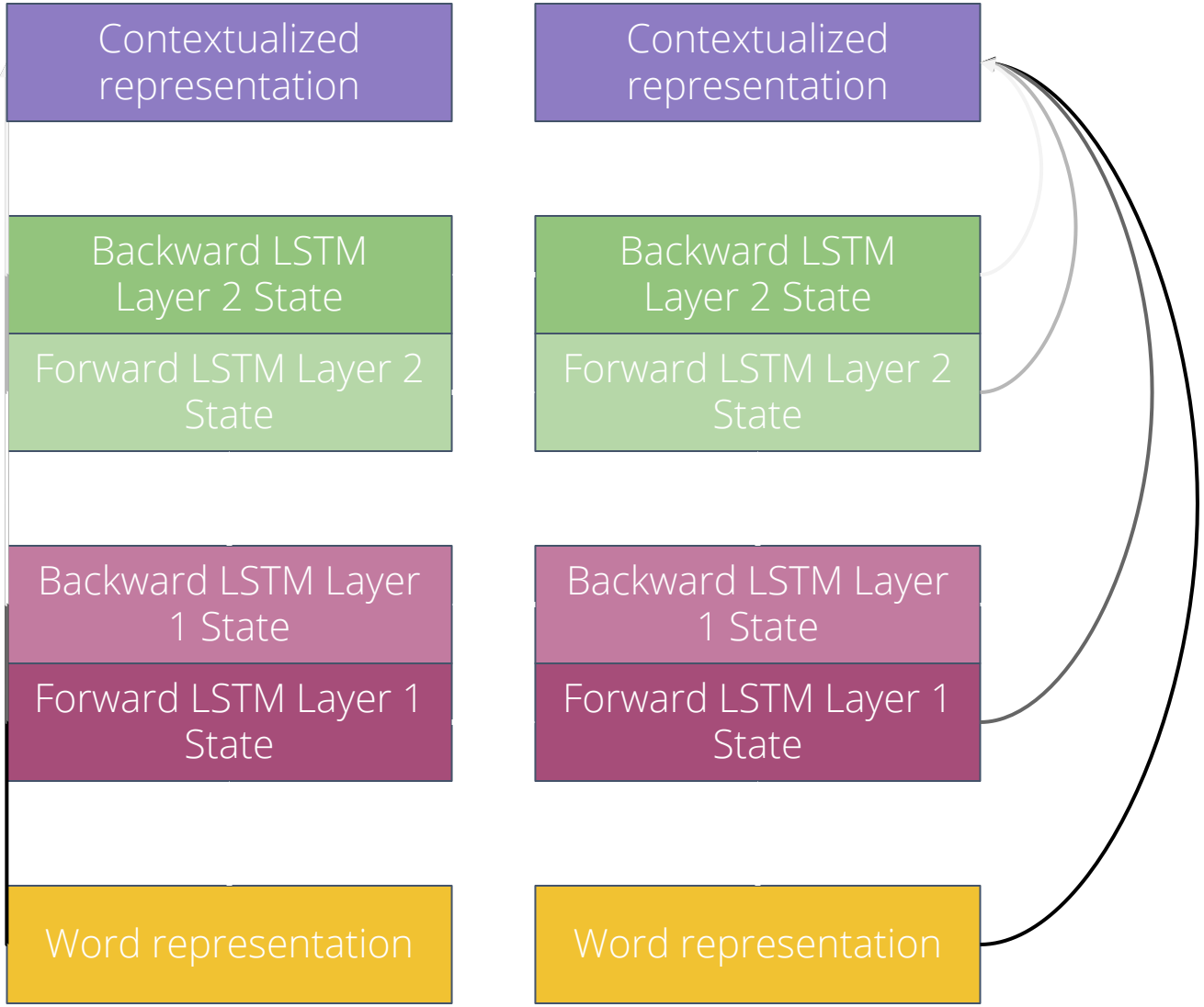
- Replace word vectors with a learned weighted sum of features of deep bi-directional LM
- Improves baseline models to SOTA
- Uses the LM from (Jozefowicz et al. 2016)
- Extends benefits of LMs to a much wider variety of tasks

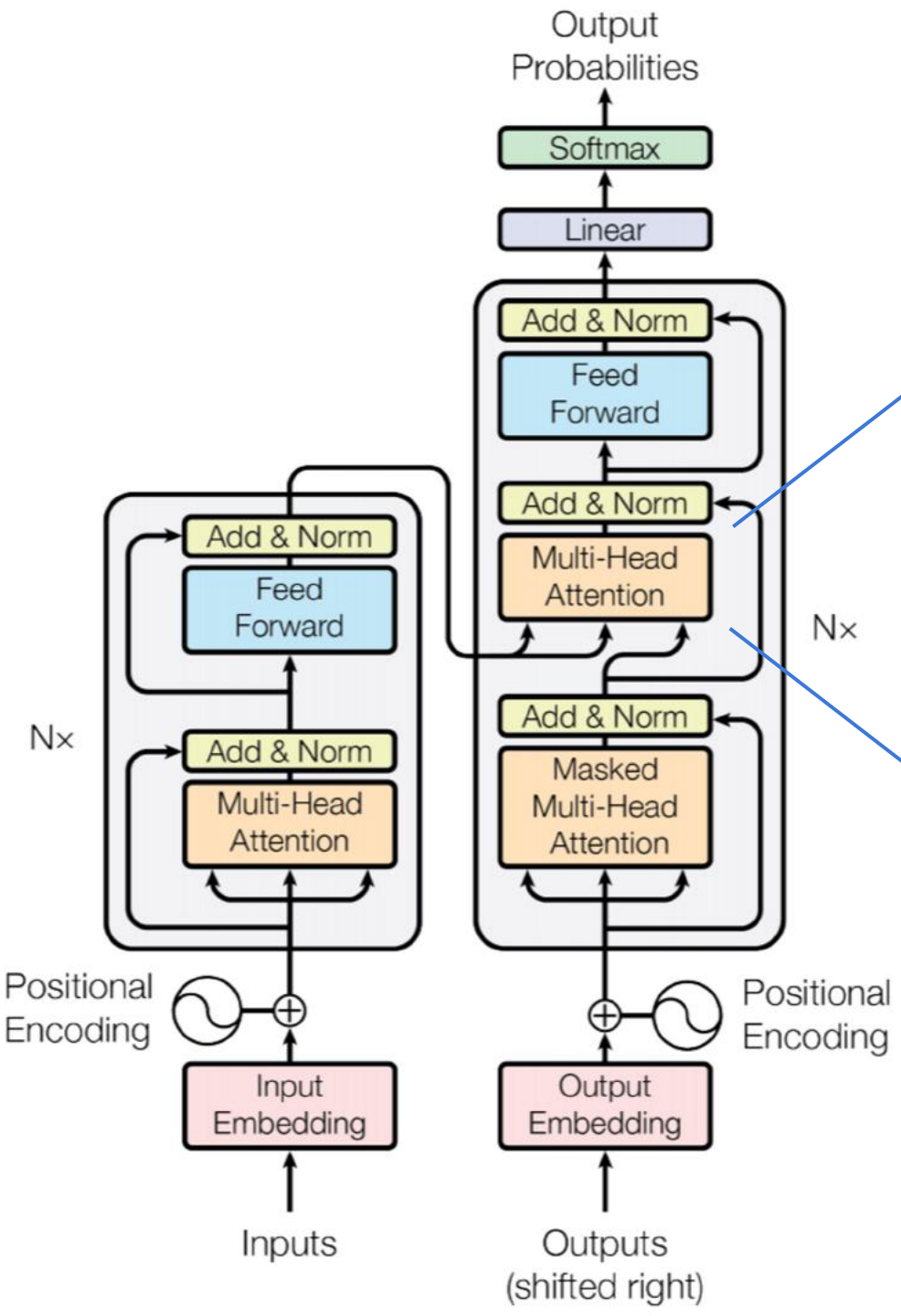
TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al.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

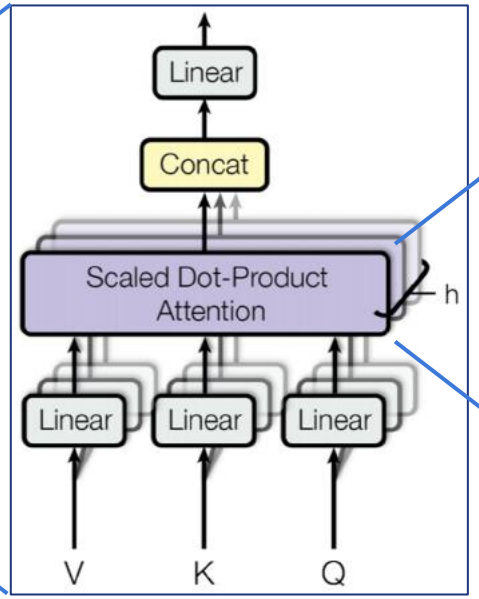


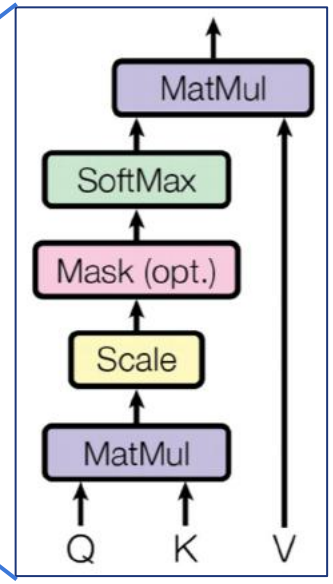
Deep contextualized word representations

Peters et al. 2018

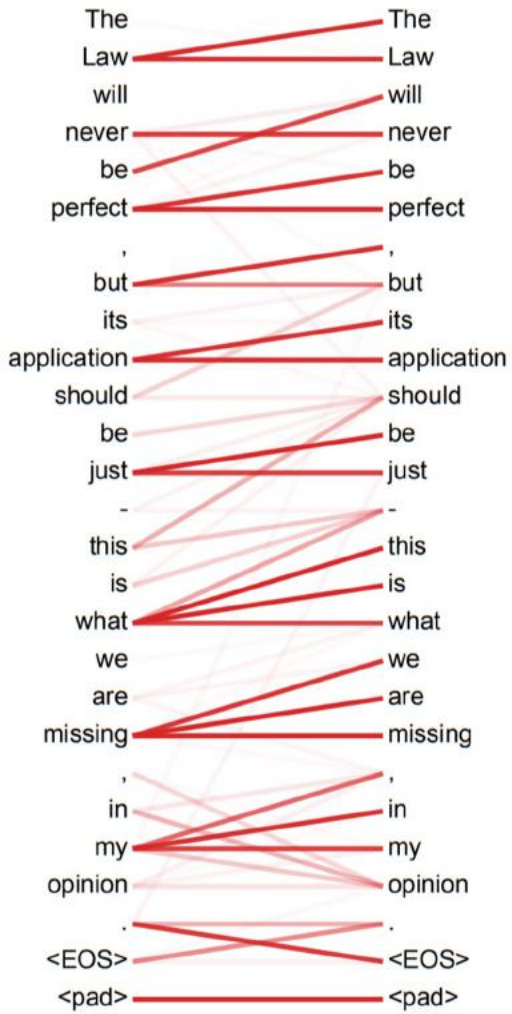


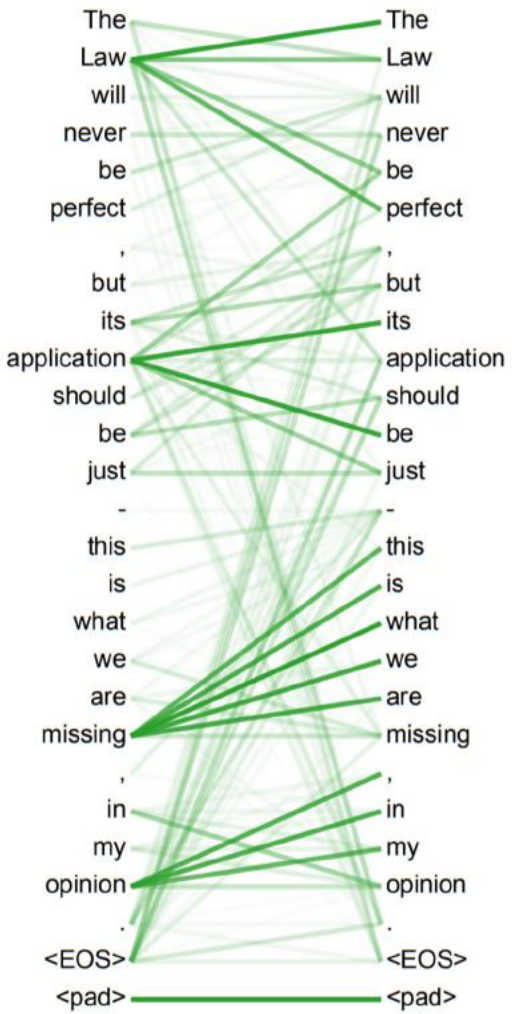


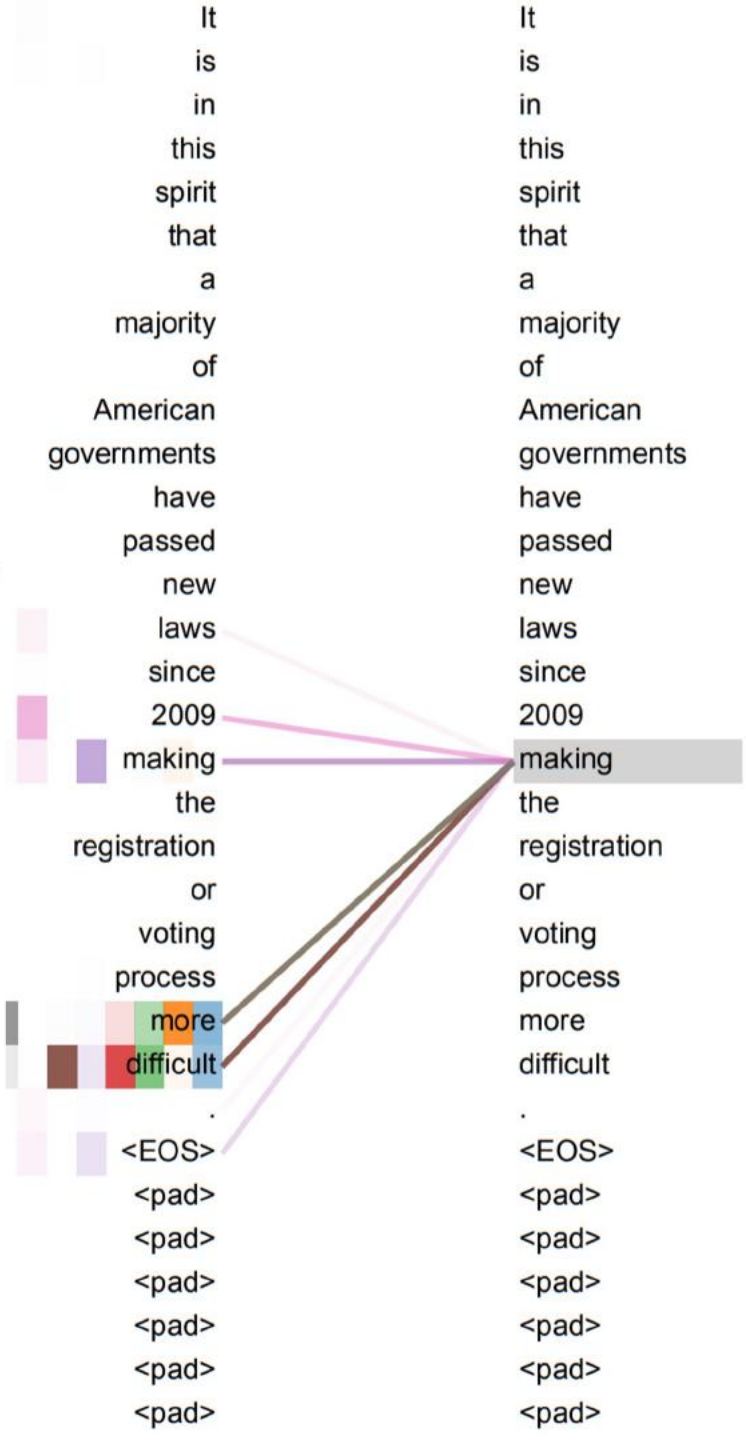


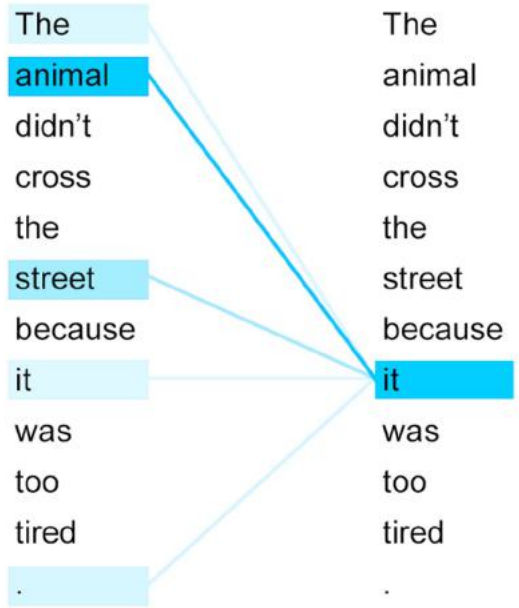


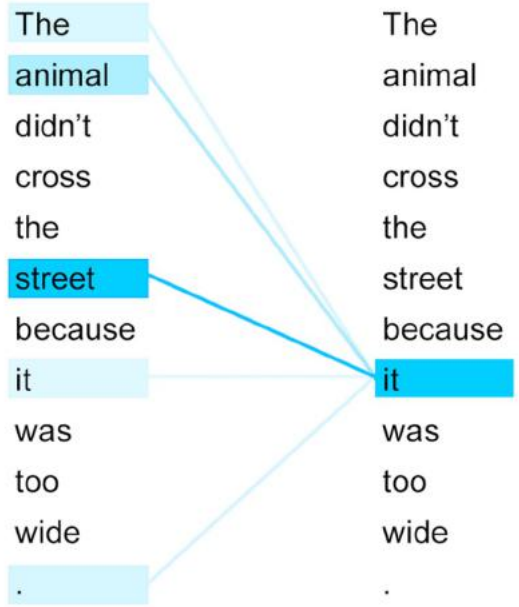
[Vaswani et al 2017]


The
Law
will
never
be
perfect
,
but
its
application
should
be
just
-
this
is
what
we
are
missing
,
in
my
opinion
.
<EOS>
<pad>
The
Law
will
never
be
perfect
,
but
its
application
should
be
just
-
this
is
what
we
are
missing
,
in
my
opinion
.
<EOS>
<pad>


The
Law
will
never
be
perfect
,
but
its
application
should
be
just
-
this
is
what
we
are
missing
,
in
my
opinion
.
<EOS>
<pad>
The
Law
will
never
be
perfect
,
but
its
application
should
be
just
-
this
is
what
we
are
missing
,
in
my
opinion
.
<EOS>
<pad>

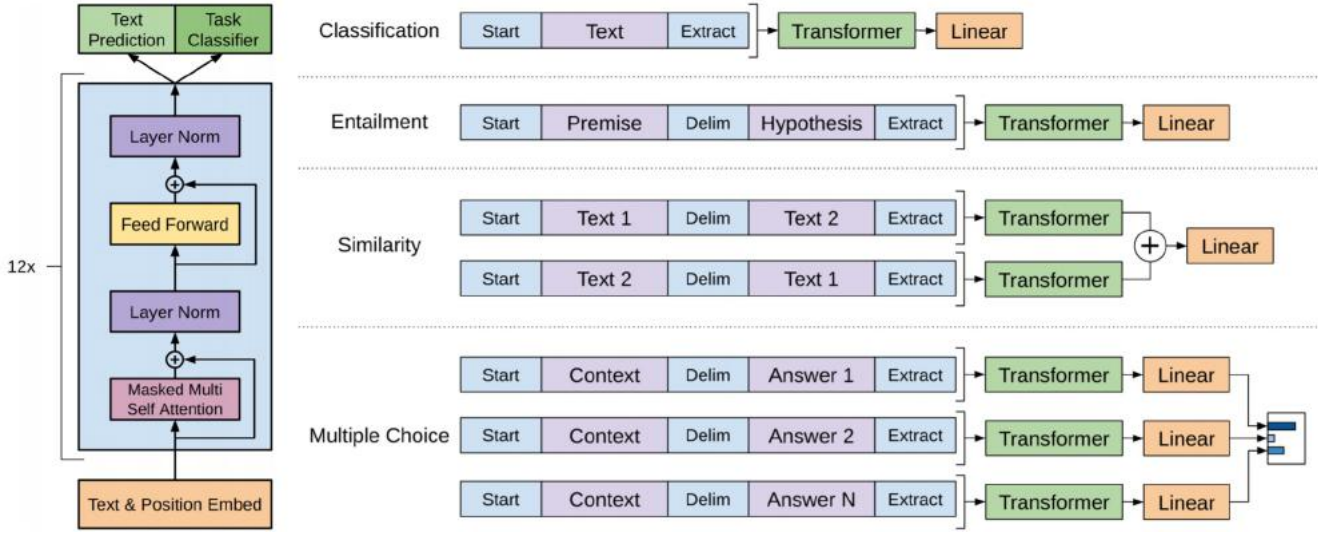

It
is
in
this
spirit
that
a
majority
of
American
governments
have
passed
new
laws
since
2009
making
the
registration
or
voting
process
more
difficult
.
<EOS>
<pad>
<pad>
<pad>
<pad>
<pad>
<pad>
It
is
in
this
spirit
that
a
majority
of
American
governments
have
passed
new
laws
since
2009
making
the
registration
or
voting
process
more
difficult
.
<EOS>
<pad>
<pad>
<pad>
<pad>
<pad>
<pad>

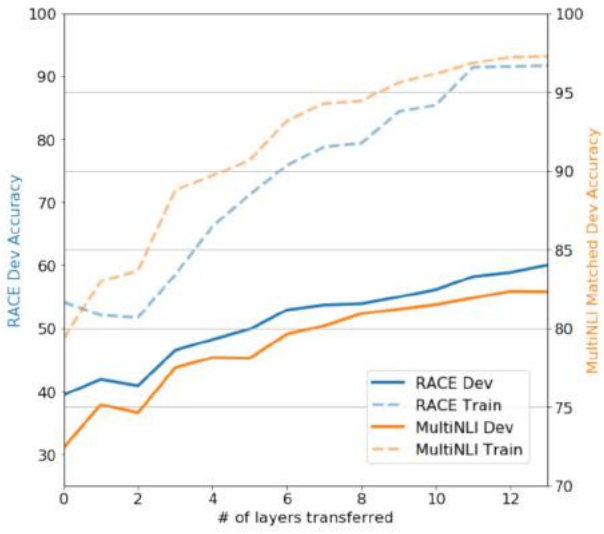

The
animal
didn't
cross
the
street
because
it
was
too
tired
.
The
animal
didn't
cross
the
street
because
it
was
too
tired
.

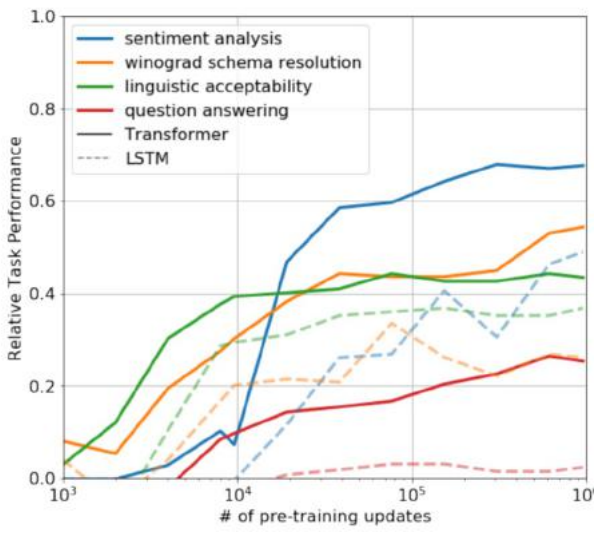

The
animal
didn't
cross
the
street
because
it
was
too
wide
.
The
animal
didn't
cross
the
street
because
it
was
too
wide
.

Improving Language Understanding by Generative Pre-Training (GPT-1)

- Transformer based LM
- 12 self-attention blocks - 12 heads - 768 dim state
  - ~100M params
- Trained on 7,000 books ~ 5 GB of text (BookCorpus Zhu et al 2015)
- Fine-tune on supervised tasks (like Dai et al. 2015)
- Removes the need for task specific architectures







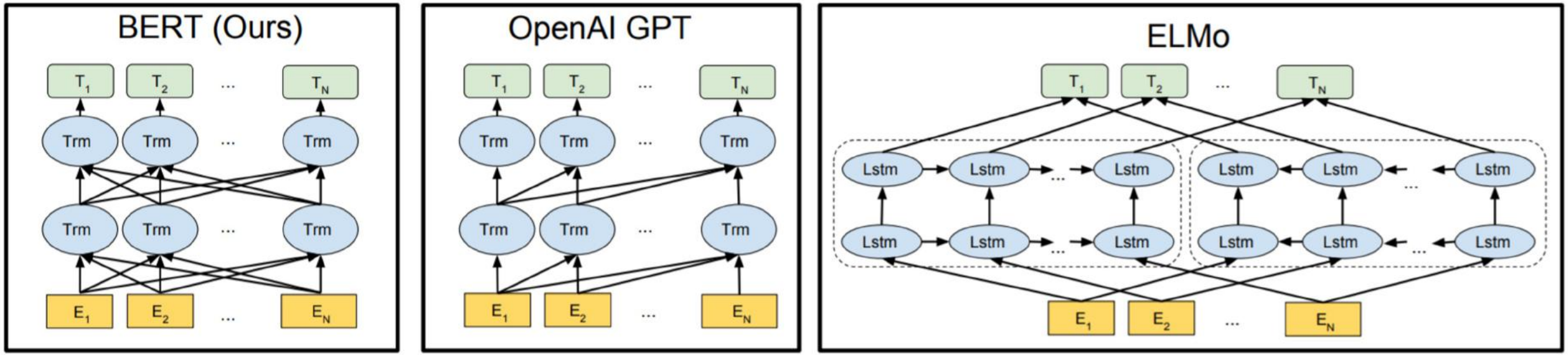
Improving Language Understanding by Generative Pre-Training (GPT-1)

- **Pretraining Architecture:** Decoder
- **Corpus:** Unsupervised Pretraining on BookCorpus dataset. Finetuning on several task-specific datasets including SNLI, RACE, Quora.
- **Params:** 117m.
- For text generation from OpenAI, but adaptable to many other NLP tasks when fine tuned.

DATASET	TASK	SOTA	OURS
SNLI	Textual Entailment	89.3	**89.9**
MNLI Matched	Textual Entailment	80.6	**82.1**
MNLI Mismatched	Textual Entailment	80.1	**81.4**
SciTail	Textual Entailment	83.3	**88.3**
QNLI	Textual Entailment	82.3	**88.1**
RTE	Textual Entailment	**61.7**	56.0
STS-B	Semantic Similarity	81.0	**82.0**
QQP	Semantic Similarity	66.1	**70.3**
MRPC	Semantic Similarity	**86.0**	82.3
RACE	Reading Comprehension	53.3	**59.0**
ROCStories	Commonsense Reasoning	77.6	**86.5**
COPA	Commonsense Reasoning	71.2	**78.6**
SST-2	Sentiment Analysis	**93.2**	91.3
CoLA	Linguistic Acceptability	35.0	**45.4**
GLUE	Multi Task Benchmark	68.9	**72.8**

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Devlin et al. 2017




```
Left-Right LM: The cat sat on the [mask] -> The cat sat on the mat
Right-Left LM: [mask] cat sat on the mat -> The cat sat on the mat
Masked LM:  The [mask] sat on the [mask] -> The cat sat on the mat
```

BERT Workflow

- The BERT workflow includes:
  - Pretrain on generic, self-supervised tasks, using large amounts of data (like all of Wikipedia)
  - Fine-tune on specific tasks with limited, labelled data.
- The pretraining tasks (will talk about this in more detail later):
  - Masked Language Modelling (to learn contextualized token representations)
  - Next Sentence Prediction (summary vector for the whole input)

Details of BERT

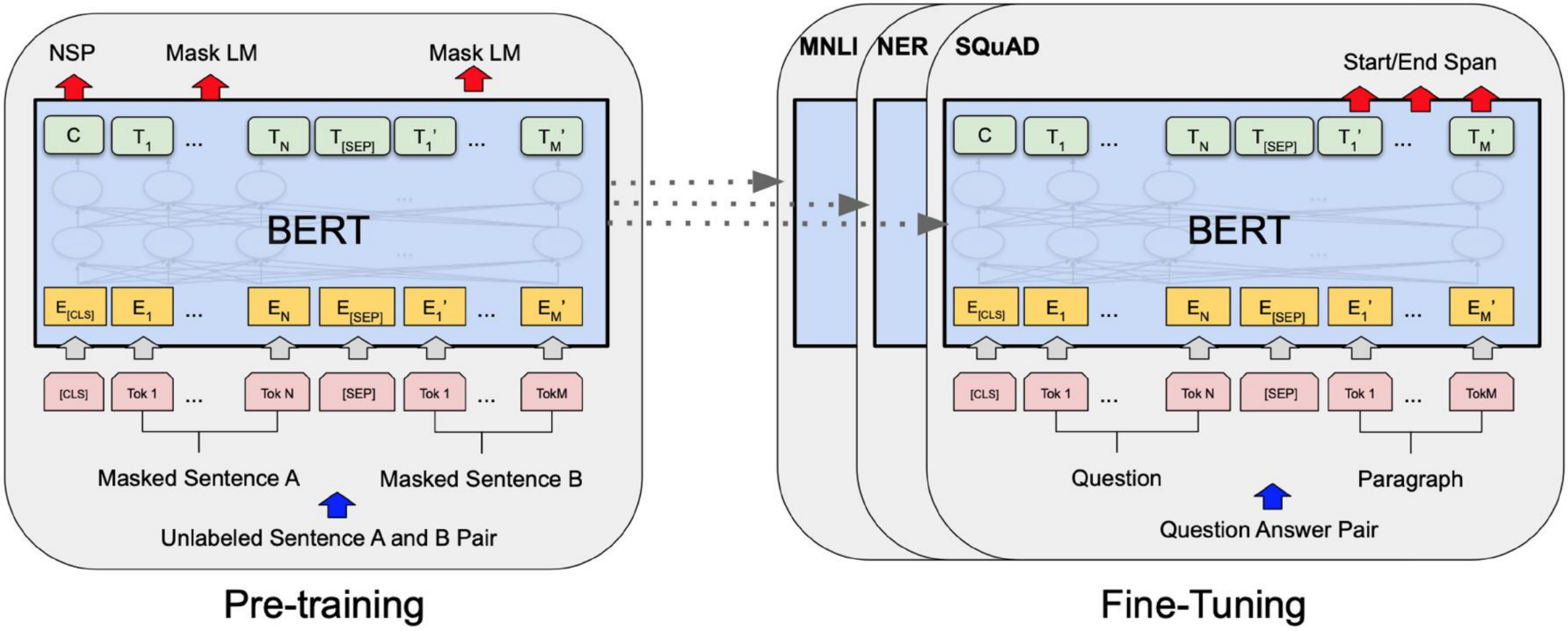
- Two models were released:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params.
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on: •
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- Pretraining is expensive and impractical on a single GPU.
  - BERT was pretrained with 64 TPU chips for a total of 4 days.
- Finetuning is practical and common on a single GPU

Details of BERT

- Two models were released:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params.
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on: •
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- Pretraining is expensive and impractical on a single GPU.
  - BERT was pretrained with 64 TPU chips for a total of 4 days.
- Finetuning is practical and common on a single GPU

“Pretrain once, finetune many times.”

BERT Architecture



BERT Embeddings

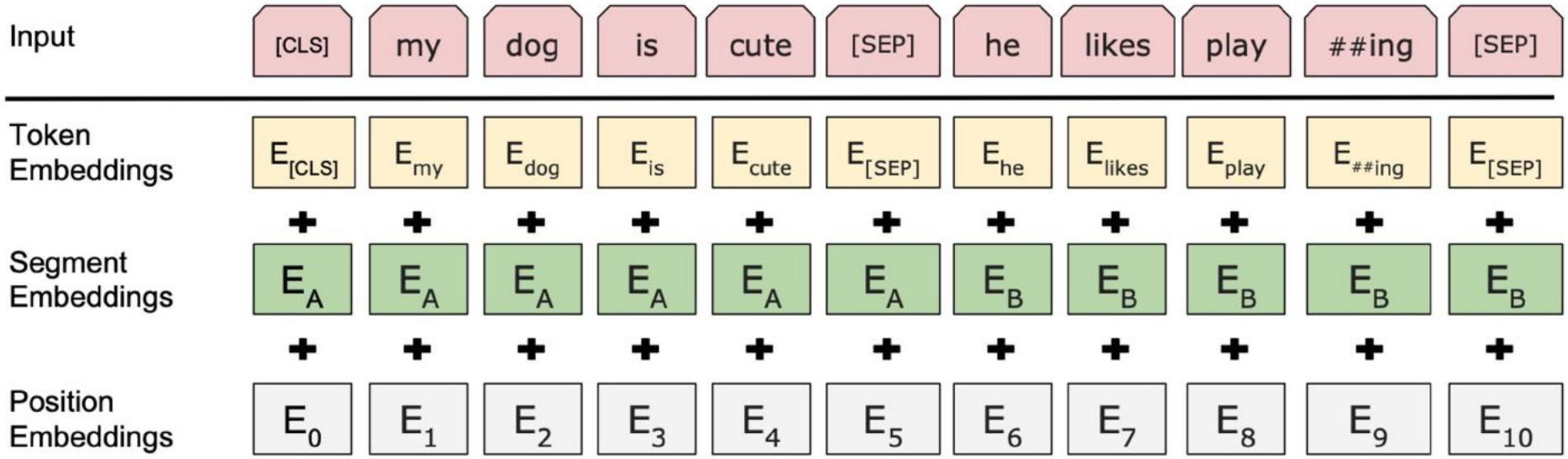


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

- How we tokenize the inputs is very important!
- BERT uses the WordPiece tokenizer (Wu et. al. 2016)

Pretraining tasks

- Masked Language Modelling, i.e. Cloze Task (Taylor, 1953)
- Next sentence prediction

Masked Language Modelling

- Mask 15% of the input tokens. (i.e. replace with a dummy masking token)
- Run the model, obtain the embeddings for the masked tokens.
- Using these embeddings, try to predict the missing token.
- ”I love to eat peanut ___ and jam. ” Can you guess what’s missing?

- This procedure forces the model to encode context information in the features of all of the tokens.

Next Sentence Prediction

- Goal is to summarize the complete context (i.e. the two segments) in a single feature vector.
- Procedure for generating data
  - Pick a sentence from the training corpus and feed it as ″segment A″.
  - With 50% probability, pick the following sentence and feed that as ″segment B″.
  - With 50% probability, pick the a random sentence and feed it as ″segment B″.
- Using the features for the context token, predict whether segment B is the following sentence of segment A.
- Turns out to be a very effective pretraining technique!

Fine Tuning

Procedure:

- Add a final layer on top of BERT representations.
- Train the whole network on the fine-tuning dataset.
- Pre-training time: In the order of days on TPUs.
- Fine tuning task: Takes only a few hours max.

Fine Tuning

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

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

RoBERTa: A Robustly Optimized BERT Pretraining Approach

Liu et al. 2019

Really well executed refinement / engineering on BERT

- Better tuned (many HPs)
- Remove a few hacks (remove annealing context size)
- Better data generation (online instead of cached)
- A more flexible vocab scheme (more on this later)
- Use more compute / train longer (but same model capacity – BERT was undertrained)

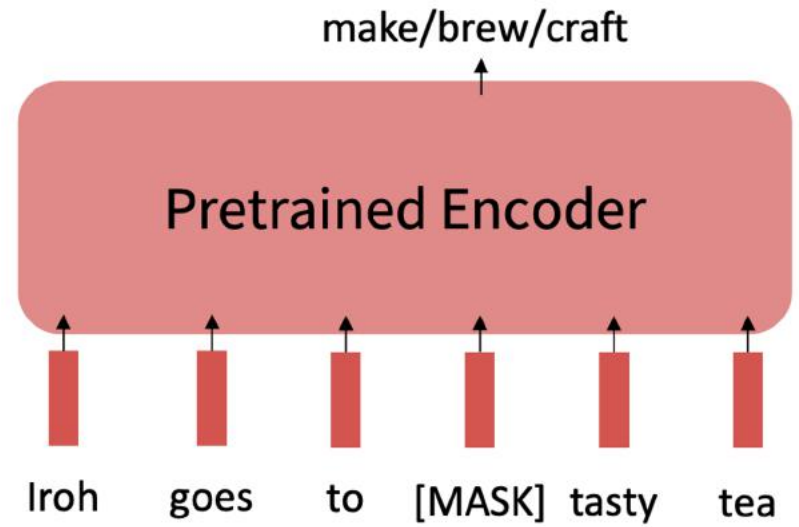
Limitations of pretrained Encoders

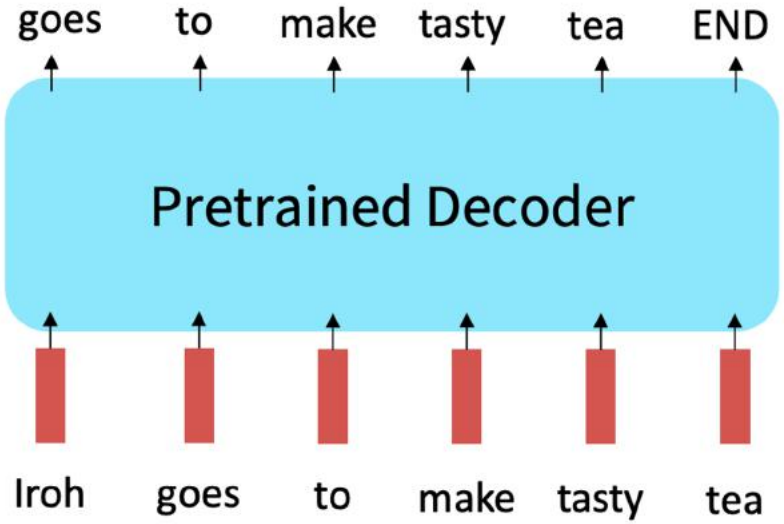
Those results looked great! Why not used pretrained encoders for everything?

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-like encoders don't naturally lead to nice autoregressive (1-word-at-a-time) generation methods.





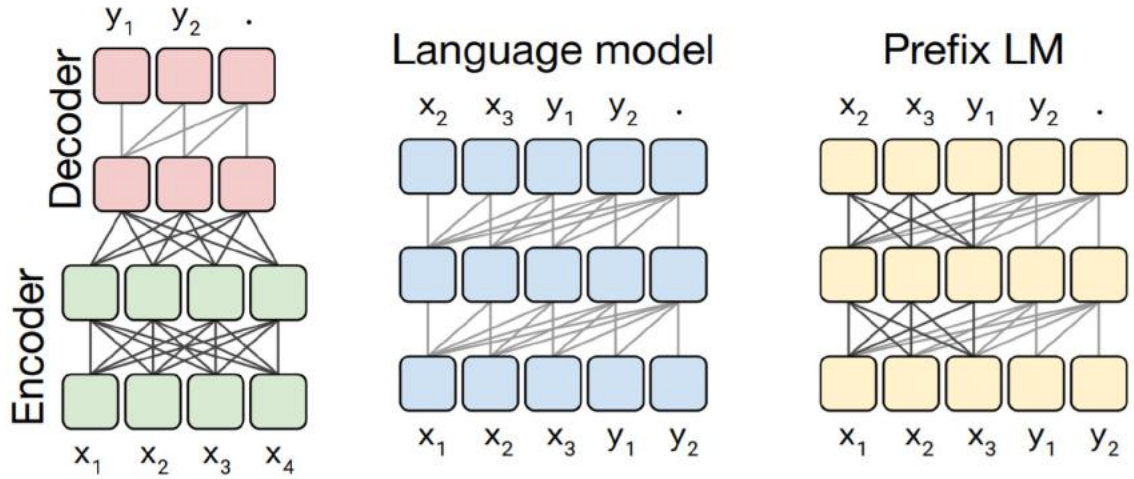
T5: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

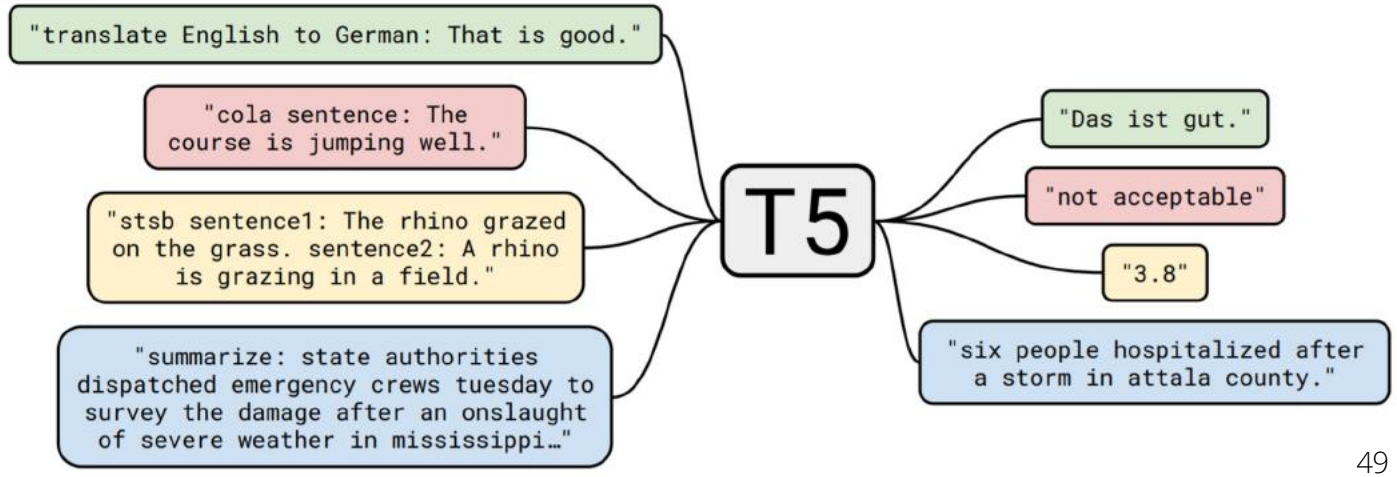
Raffel et al. 2019

- Very thorough (50 pages!) exploration of the design space of pretraining with a pleasing task formulation (from McCann et al 2018)

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

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

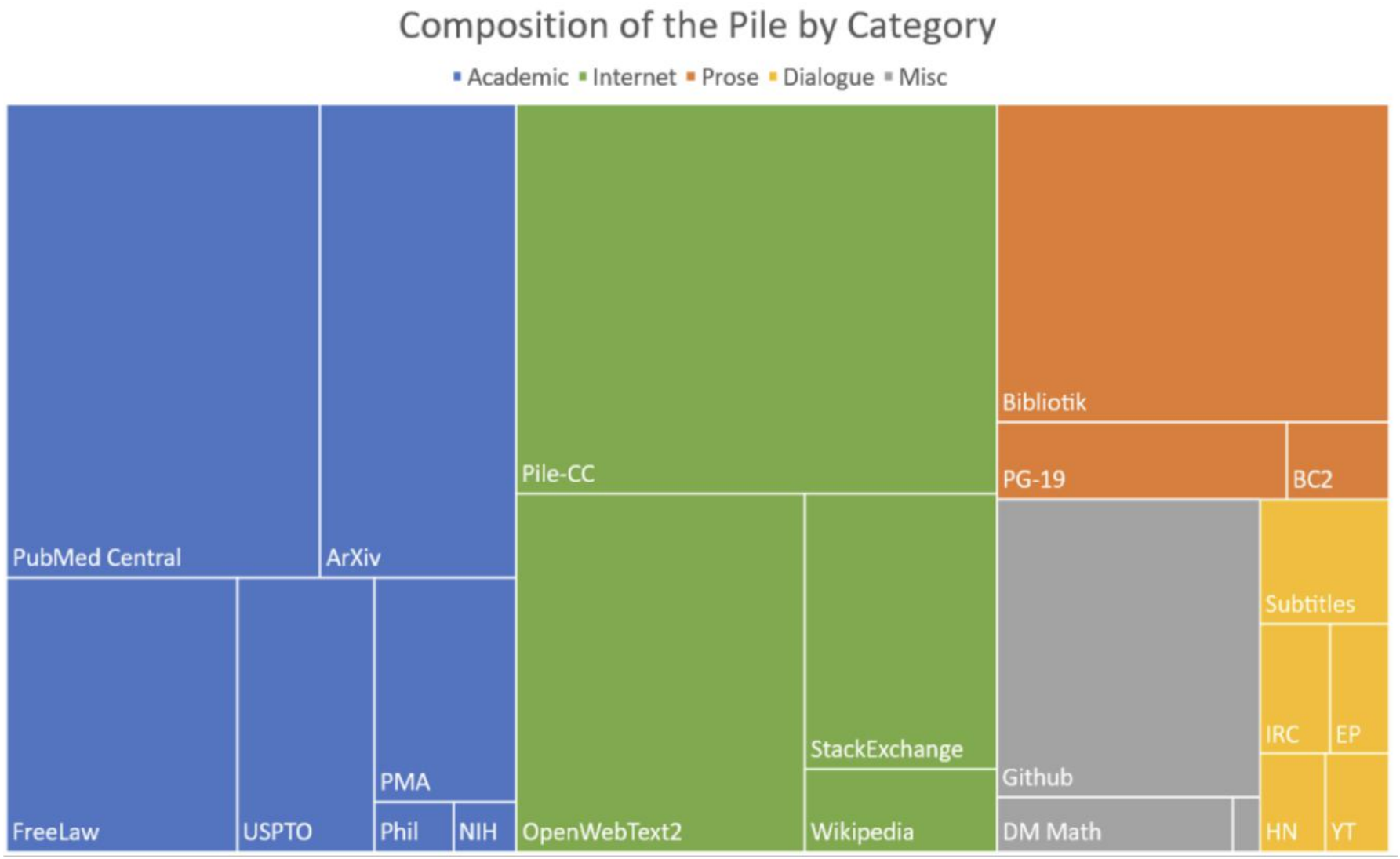




Information Engineering Taking Off (CoVe, ELMo, ULMFiT, GPT-1, BERT)

- GPT-1 performs similarly to KIM (83.75%) on the new NLI test set
- BERT is basically SOTA on everything
- It's just a "stock" transformer!
- But it makes up for this with all that its learned through pre-training.

Where does the data come from?



Model	Training Data
BERT	BookCorpus, English Wikipedia
GPT-1	BookCorpus
GPT-3	CommonCrawl, WebText, English Wikipedia, and 2 book databases (“Books 1” and “Books 2”)
GPT-3.5+	Undisclosed

How to make progress?

- Better models / architectures?
- More data?
- Different paths all together?

How to make progress?

- Better models / architectures?
- More data?
- Different paths all together?

How to make progress?

- Better models / architectures?
- More data?
- Different paths all together?

(2024) The key elements of a good dataset:

- Quality
- Diversity
- Quantity

GPT-2

- More data
  - 40GB of text
  - 10B tokens
  - 8 million webpages

- Bigger model
  - Up to 1.5 billion parameters
  - 1024 token context
  - 48 layers, 1600 dim state

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

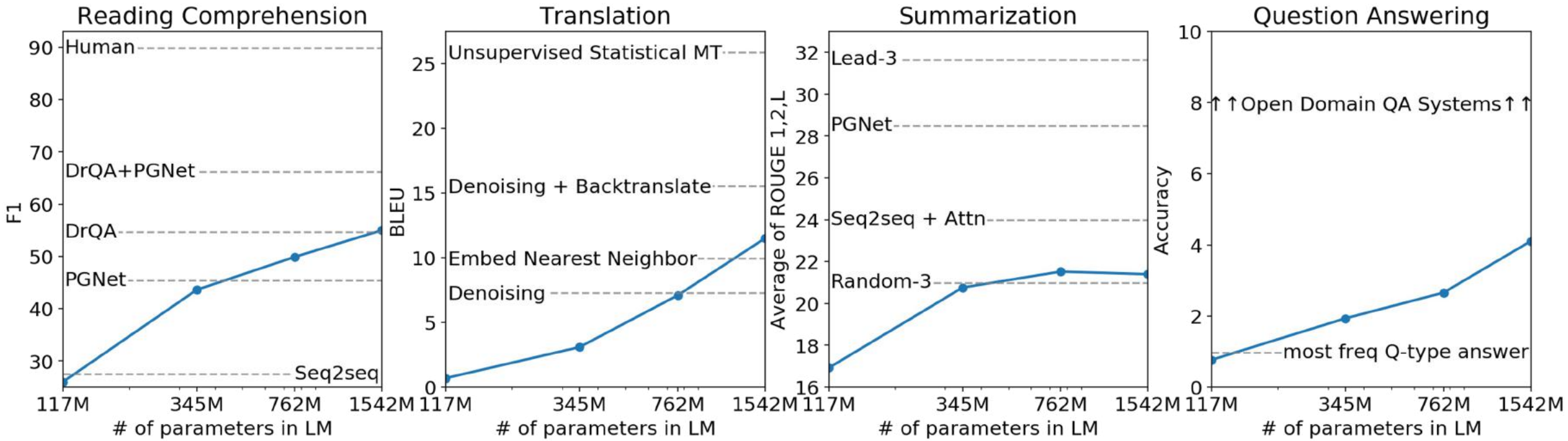
Table 2. Architecture hyperparameters for the 4 model sizes.

Just a language model - predicts everything (with some unfortunate restrictions as BERT shows)

Performance across tasks

DATASET	METRIC	OUR RESULT	PREVIOUS RECORD	HUMAN
Winograd Schema Challenge	accuracy (+)	**70.70%**	63.7%	92%+
LAMBADA	accuracy (+)	**63.24%**	59.23%	95%+
LAMBADA	perplexity (-)	**8.6**	99	~1-2
Children's Book Test Common Nouns (validation accuracy)	accuracy (+)	**93.30%**	85.7%	96%
Children's Book Test Named Entities (validation accuracy)	accuracy (+)	**89.05%**	82.3%	92%
Penn Tree Bank	perplexity (-)	**35.76**	46.54	unknown
WikiText-2	perplexity (-)	**18.34**	39.14	unknown
enwik8	bits per character (-)	**0.93**	0.99	unknown
text8	bits per character (-)	**0.98**	1.08	unknown
WikiText-103	perplexity (-)	**17.48**	18.3	unknown

Performance across tasks



Why it's working?

"I'm not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile [I'm not a fool].**

In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: "**Mentez mentez, il en restera toujours quelque chose**," which translates as, "**Lie lie and something will always remain.**"

"I hate the word '**perfume**,"' Burr says. 'It's somewhat better in French: '**parfum**.'

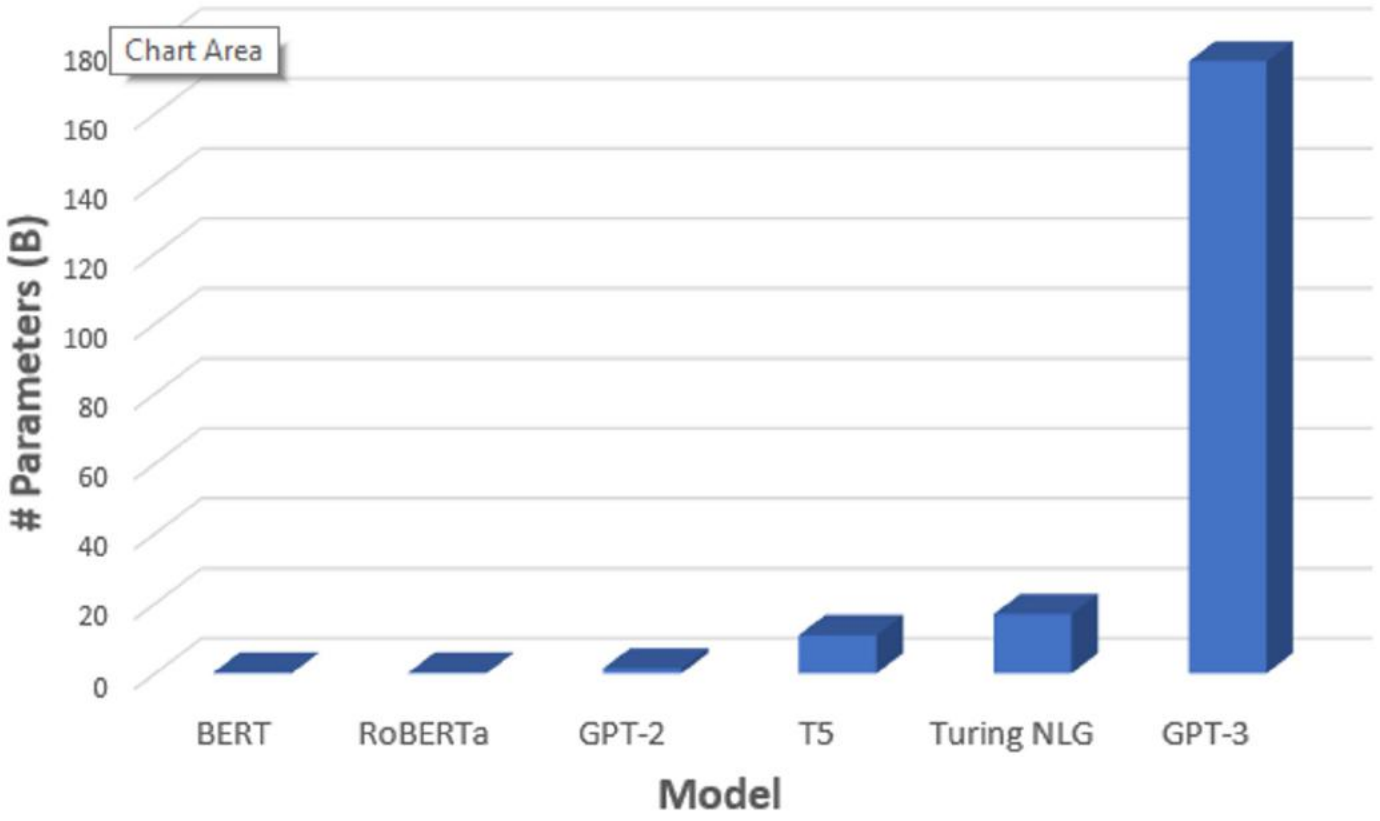
If listened carefully at 29:55, a conversation can be heard between two guys in French: "**-Comment on fait pour aller de l'autre coté? -Quel autre coté?**", which means "**- How do you get to the other side? - What side?**".

If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?

"**Brevet Sans Garantie Du Gouvernement**", translated to English: "**Patented without government warranty**".

GPT-3

- **Architecture:** Decoder
- **Corpus:** 500B tokens including CommonCrawl (410B), WebText2 (19B), Books1 (12B), Books2 (55B), and Wikipedia (3B).
- **Versions:** 125M, 350M, 760M, 1.3B, 2.7B, 6.7B, 13B, **175B**.
- Context length of 2048, trained on 200 billion words of very diverse mostly English text.



GPT-3

- So far, we've interacted with pretrained models in two ways:
  - Sample from the distributions they define (**prompting**)
  - **Fine-tune** them on a task we care about, and take their predictions.

- Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.
- GPT-3 is the canonical example of this. The largest T5 model had 11 billion parameters.
- GPT-3 has **175 billion parameters**.

What is few-shot learning?

Zero-shot

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

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

Few-shot

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

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

Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

Models and Data

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

GPT-3

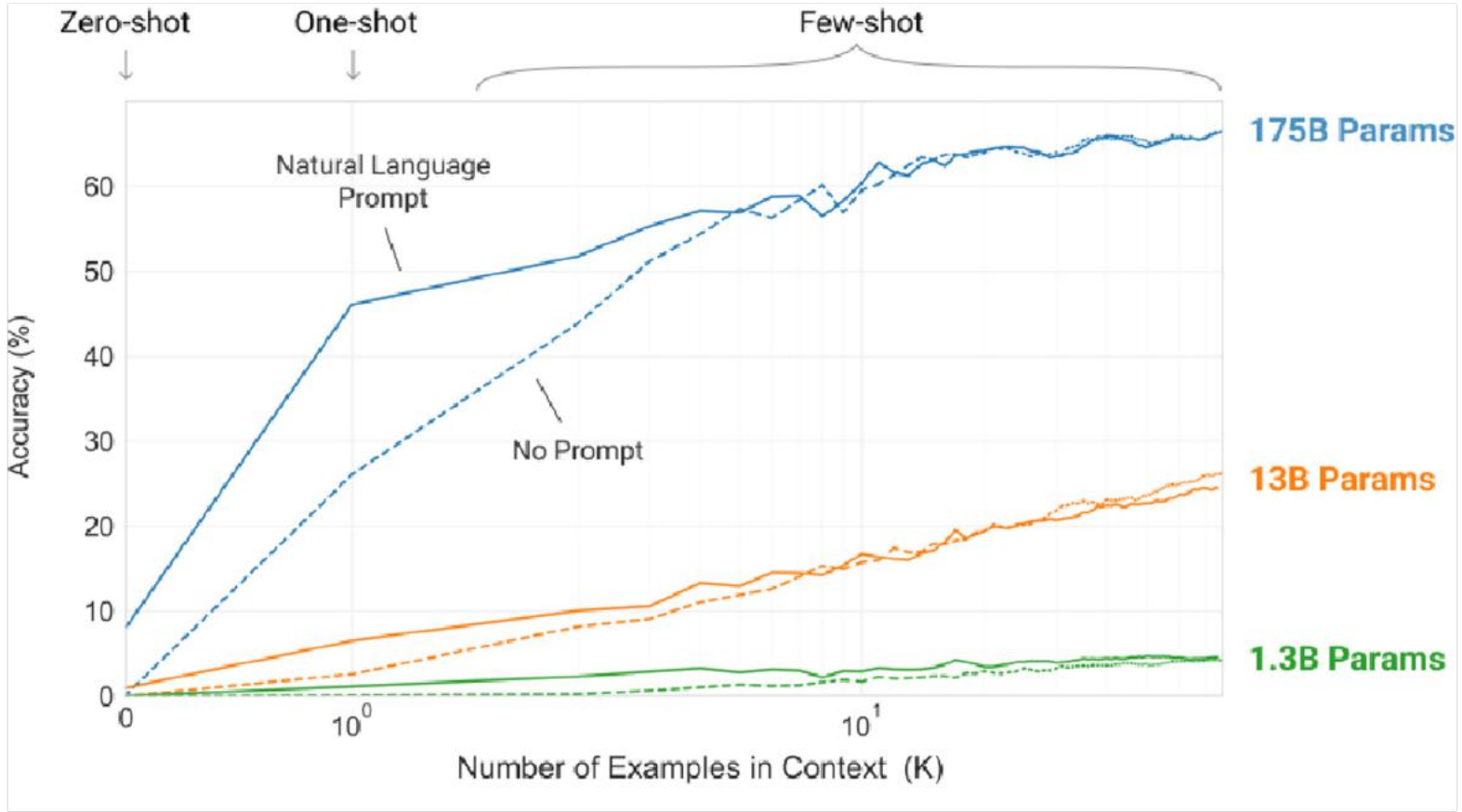


Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper "in-context learning curves" for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

GPT-3

Context →	Helsinki is the capital and largest city of Finland. It is in the region of Uusimaa, in southern Finland, on the shore of the Gulf of Finland. Helsinki has a population of , an urban population of , and a metropolitan population of over 1.4 million, making it the most populous municipality and urban area in Finland. Helsinki is some north of Tallinn, Estonia, east of Stockholm, Sweden, and west of Saint Petersburg, Russia. Helsinki has close historical connections with these three cities. The Helsinki metropolitan area includes the urban core of Helsinki, Espoo, Vantaa, Kauniainen, and surrounding commuter towns. It is the world's northernmost metro area of over one million people, and the city is the northernmost capital of an EU member state. The Helsinki metropolitan area is the third largest metropolitan area in the Nordic countries after Stockholm and Copenhagen, and the City of Helsinki is the third largest after Stockholm and Oslo. Helsinki is Finland's major political, educational, financial, cultural, and research center as well as one of northern Europe's major cities. Approximately 75% of foreign companies that operate in Finland have settled in the Helsinki region. The nearby municipality of Vantaa is the location of Helsinki Airport, with frequent service to various destinations in Europe and Asia. Q: what is the most populous municipality in Finland? A: Helsinki Q: how many people live there? A: 1.4 million in the metropolitan area Q: what percent of the foreign companies that operate in Finland are in Helsinki? A: 75% Q: what towns are a part of the metropolitan area? A:
Target Completion →	Helsinki, Espoo, Vantaa, Kauniainen, and surrounding commuter towns

Formatted dataset example for CoQA

PaLM: Scaling Language Modeling with Pathways

PaLM: Scaling Language Modeling with Pathways

Aakanksha Chowdhery* Sharan Narang* Jacob Devlin*
Maarten Bosma Gaurav Mishra Adam Roberts Paul Barham
Hyung Won Chung Charles Sutton Sebastian Gehrmann Parker Schuh Kensen Shi
Sasha Tsvyashchenko Joshua Maynez Abhishek Rao† Parker Barnes Yi Tay
Noam Shazeer‡ Vinodkumar Prabhakaran Emily Reif Nan Du Ben Hutchinson
Reiner Pope James Bradbury Jacob Austin Michael Isard Guy Gur-Ari
Pengcheng Yin Toju Duke Anselm Levskaya Sanjay Ghemawat Sunipa Dev
Henryk Michalewski Xavier Garcia Vedant Misra Kevin Robinson Liam Fedus
Denny Zhou Daphne Ippolito David Luan‡ Hyeontaek Lim Barret Zoph
Alexander Spiridonov Ryan Sepassi David Dohan Shivani Agrawal Mark Omernick
Andrew M. Dai Thanumalayan Sankaranarayana Pillai Marie Pellat Aitor Lewkowycz
Erica Moreira Rewon Child Oleksandr Polozov† Katherine Lee Zongwei Zhou
Xuezhi Wang Brennan Saeta Mark Diaz Orhan Firat Michele Catasta† Jason Wei
Kathy Meier-Hellstern Douglas Eck Jeff Dean Slav Petrov Noah Fiedel

Google Research

Abstract

Large language models have been shown to achieve remarkable performance across a variety of natural language tasks using *few-shot learning*, which drastically reduces the number of task-specific training examples needed to adapt the model to a particular application. To further our understanding of the impact of scale on few-shot learning, we trained a 540-billion parameter, densely activated, Transformer language model, which we call Pathways Language Model (PaLM).

We trained PaLM on 6144 TPU v4 chips using Pathways, a new ML system which enables highly efficient training across multiple TPU Pods. We demonstrate continued benefits of scaling by achieving state-of-the-art few-shot learning results on hundreds of language understanding and generation benchmarks. On a number of these tasks, PaLM 540B achieves breakthrough performance, outperforming the finetuned state-of-the-art on a suite of multi-step reasoning tasks, and outperforming average human performance on the recently released BIG-bench benchmark. A significant number of BIG-bench tasks showed *discontinuous improvements* from model scale, meaning that performance steeply increased as we scaled to our largest model. PaLM also has strong capabilities in multilingual tasks and source code generation, which we demonstrate on a wide array of benchmarks. We additionally provide a comprehensive analysis on bias and toxicity, and study the extent of training data memorization with respect to model scale. Finally, we discuss the ethical considerations related to large language models and discuss potential mitigation strategies.

- "A 540-billion parameter, densely activated, Transformer language model, which we call Pathways Language Model (PaLM)"
- "Trained on 6144 TPU v4 chips using Pathways, a new ML system which enables highly efficient training across multiple TPU Pods"

PaLM: Scaling Language Modeling with Pathways

- Architecture: Decoder
- Corpus: 780B tokens from filtered webpages, books, Wikipedia, news articles, source code, and social media conversations. Code includes 24 programming languages.
- Parameters: 540B
- PaLM uses a typical decoder-only Transformer architecture, but adds quite a few extensions: SwiGLU activations, parallel layers, multi-query attention, RoPE embeddings, Shared Input-Output Embeddings, no biases, and a 256k SentencePiece vocabulary generated from the training data.

PaLM

Explaining a Joke

Input: Did you see that Google just hired an eloquent whale for their TPU team? It showed them how to communicate between two different pods!
Model Output: TPUs are a type of computer chip that Google uses for deep learning. A "pod" is a group of TPUs. A "pod" is also a group of whales. The joke is that the whale is able to communicate between two groups of whales, but the speaker is pretending that the whale is able to communicate between two groups of TPUs.

Logical Inference

Input: Shelley is from Virginia, but is visiting that city with that famous market where they throw the fish. Going home next Tuesday!
Question: Is it likely that Shelley will be near the Pacific Ocean this weekend?
Model Output: The city with the famous market where they throw the fish is Seattle, Washington. Seattle is on the Pacific Ocean. Shelley is visiting Seattle, so she will be near the Pacific Ocean this weekend. The answer is "yes", it is likely that Shelley will be near the Pacific Ocean this weekend.

Figure 1: These examples demonstrate exploratory capabilities in prompting PaLM to generate explanations using *chain-of-thought prompting* (Wei et al., 2022b). All examples were generated with greedy (1-best) decoding of PaLM 540B. The prompting was done with 2-shot exemplars, which are shown in Section 9, along with output on more examples using these same exemplars.

PaLM

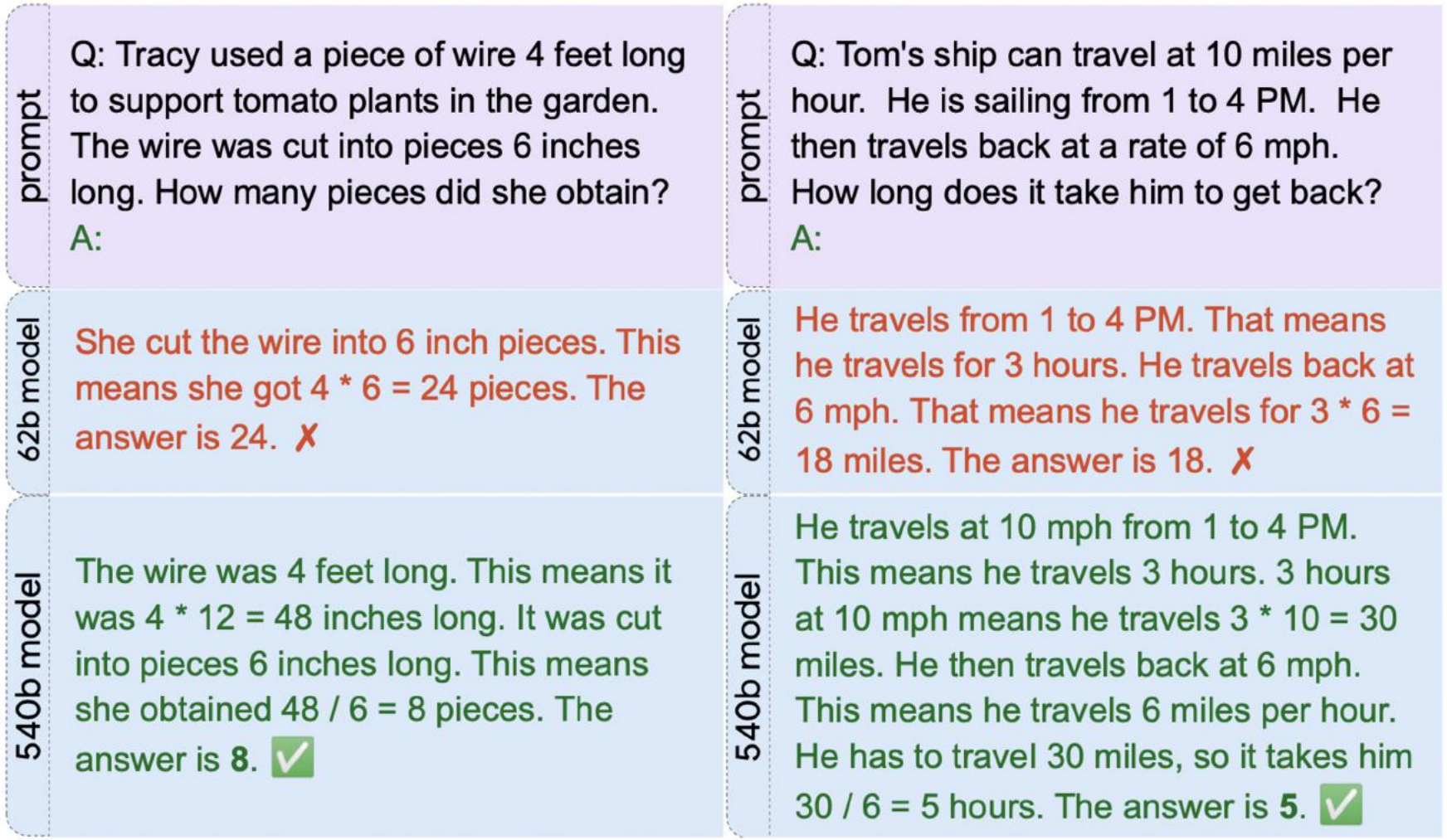


Figure 26: Examples questions where PaLM 62B had incorrect responses with semantic understanding errors, but PaLM 540B fixed such errors.

PaLM

Task	0-shot		1-shot		Few-shot	
	Prior SOTA	PaLM 540B	Prior SOTA	PaLM 540B	Prior SOTA	PaLM 540B
TriviaQA (EM)	71.3[a]	**76.9**	75.8[a]	**81.4**	75.8[a] (1)	**81.4** (1)
Natural Questions (EM)	**24.7**[a]	21.2	26.3[a]	**29.3**	32.5[a] (1)	**39.6** (64)
Web Questions (EM)	**19.0**[a]	10.6	**25.3**[b]	22.6	41.1[b] (64)	**43.5** (64)
Lambada (EM)	77.7[f]	**77.9**	80.9[a]	**81.8**	87.2[c] (15)	**89.7** (8)
HellaSwag	80.8[f]	**83.4**	80.2[c]	**83.6**	82.4[c] (20)	**83.8** (5)
StoryCloze	83.2[b]	**84.6**	84.7[b]	**86.1**	87.7[b] (70)	**89.0** (5)
Winograd	88.3[b]	**90.1**	**89.7**[b]	87.5	88.6[a] (2)	**89.4** (5)
Winogrande	74.9[f]	**81.1**	73.7[c]	**83.7**	79.2[a] (16)	**85.1** (5)
Drop (F1)	57.3[a]	**69.4**	57.8[a]	**70.8**	58.6[a] (2)	**70.8** (1)
CoQA (F1)	**81.5**[b]	77.6	**84.0**[b]	79.9	**85.0**[b] (5)	81.5 (5)
QuAC (F1)	41.5[b]	**45.2**	43.4[b]	**47.7**	44.3[b] (5)	**47.7** (1)
SQuADv2 (F1)	71.1[a]	**80.8**	71.8[a]	**82.9**	71.8[a] (10)	**83.3** (5)
SQuADv2 (EM)	64.7[a]	**75.5**	66.5[a]	**78.7**	67.0[a] (10)	**79.6** (5)
RACE-m	64.0[a]	**68.1**	65.6[a]	**69.3**	66.9[a†] (8)	**72.1** (8)
RACE-h	47.9[c]	**49.1**	48.7[a]	**52.1**	49.3[a†] (2)	**54.6** (5)
PIQA	82.0[c]	**82.3**	81.4[a]	**83.9**	83.2[c] (5)	**85.2** (5)
ARC-e	76.4[e]	**76.6**	76.6[a]	**85.0**	80.9[e] (10)	**88.4** (5)
ARC-c	51.4[b]	**53.0**	53.2[b]	**60.1**	52.0[a] (3)	**65.9** (5)
OpenbookQA	**57.6**[b]	53.4	**55.8**[b]	53.6	65.4[b] (100)	**68.0** (32)
BoolQ	83.7[f]	**88.0**	82.8[a]	**88.7**	84.8[c] (32)	**89.1** (8)
Copa	91.0[b]	**93.0**	**92.0**[a]	91.0	93.0[a] (16)	**95.0** (5)
RTE	**73.3**[e]	72.9	71.5[a]	**78.7**	76.8 (5)	**81.2** (5)
WiC	50.3[a]	**59.1**	52.7[a]	**63.2**	58.5[c] (32)	**64.6** (5)
Multirc (F1a)	73.7[a]	**83.5**	74.7[a]	**84.9**	77.5[a] (4)	**86.3** (5)
WSC	85.3[a]	**89.1**	83.9[a]	**86.3**	85.6[a] (2)	**89.5** (5)
ReCoRD	90.3[a]	**92.9**	90.3[a]	**92.8**	90.6 (2)	**92.9** (2)
CB	48.2[a]	**51.8**	73.2[a]	**83.9**	84.8[a] (8)	**89.3** (5)
ANLI R1	39.2[a]	**48.4**	42.4[a]	**52.6**	44.3[a] (2)	**56.9** (5)
ANLI R2	39.9[e]	**44.2**	40.0[a]	**58.7**	41.2[a] (10)	**56.1** (5)
ANLI R3	41.3[a]	**45.7**	40.8[a]	**52.3**	44.7[a] (4)	**51.2** (5)

Results obtained by the PaLM 540B model across 29 NLP benchmarks. For the few-shot results, the number of shots for each task are mentioned in parenthesis

OPT: Open Pre-trained Transformer Language Models

- Family: GPT
- Pretraining Architecture: Decoder
- Corpus: 180B tokens from RoBERTa, Pile, and Reddit.
- Versions: 125m, 350m, 1.3B, 2.7B, 6.7B, 13B, 30B, 66B, 175B.
- Basically same architecture as GPT-3 from Facebook LAB, but with some training improvements introduced in Megatron-LM.

Model	#L	#H	d_{model}	LR	Batch
125M	12	12	768	$6.0e{-4}$	0.5M
350M	24	16	1024	$3.0e{-4}$	0.5M
1.3B	24	32	2048	$2.0e{-4}$	1M
2.7B	32	32	2560	$1.6e{-4}$	1M
6.7B	32	32	4096	$1.2e{-4}$	2M
13B	40	40	5120	$1.0e{-4}$	4M
30B	48	56	7168	$1.0e{-4}$	4M
66B	64	72	9216	$0.8e{-4}$	2M
175B	96	96	12288	$1.2e{-4}$	2M

Flan-T5/PaLM

- **Family:** T5/GPT
- **Pretraining Architecture:** Encoder-Decoder / Decoder
- **Corpus:** Flan finetuned with tasks in Muffin, T0-SF, NIV2, and CoT.
- **Versions:** 8B, 62B, 540B.
- Flan-PaLM is generated by "Flan Finetuning" the PaLM models: (1) scaling the number of tasks to 1,836, (2) scaling the model size, and (3) finetuning on chain-of-thought data..

Flan-T5/PaLM

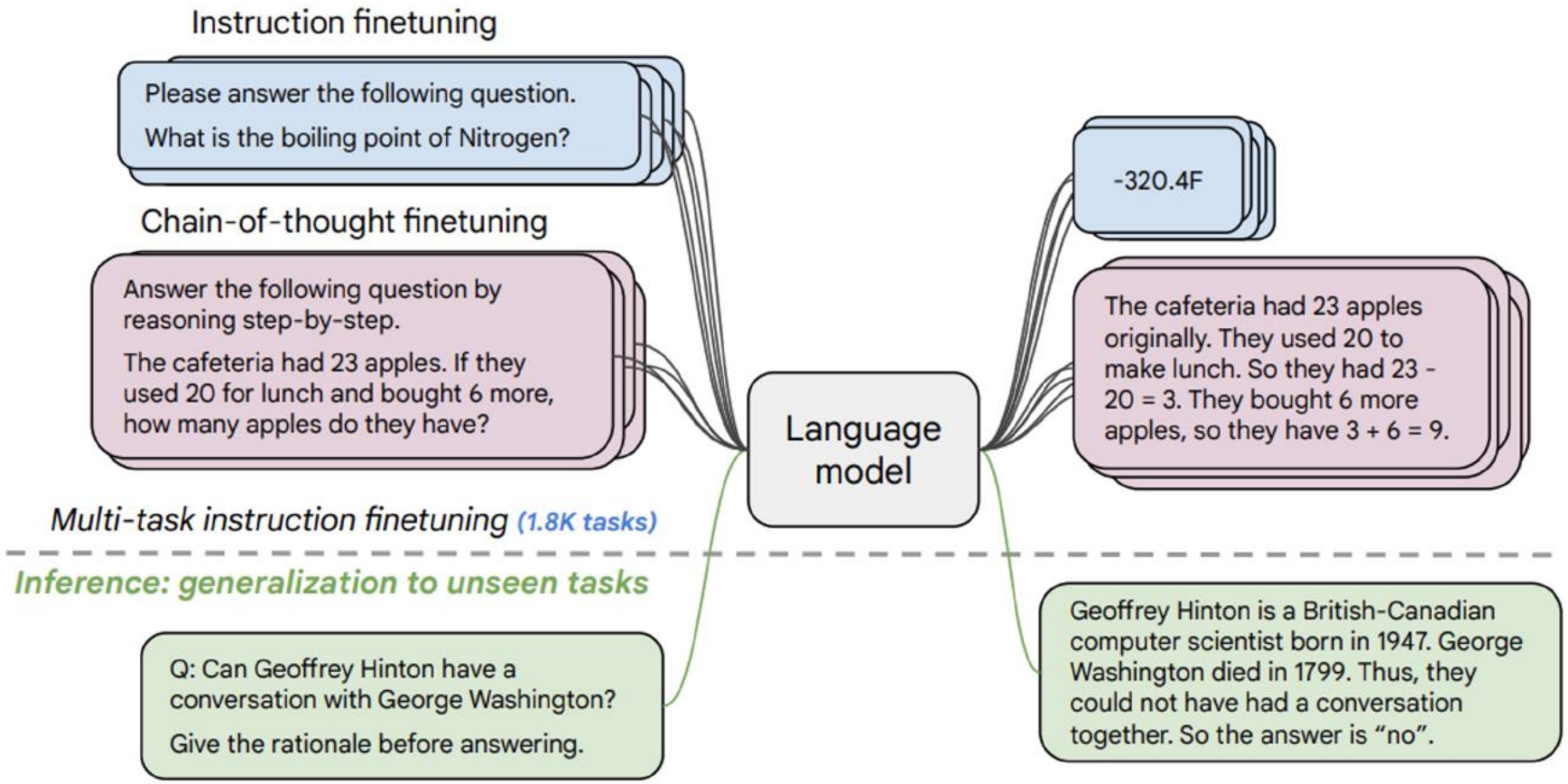


Figure 1: We finetune various language models on 1.8K tasks phrased as instructions, and evaluate them on unseen tasks. We finetune both with and without exemplars (i.e., zero-shot and few-shot) and with and without chain-of-thought, enabling generalization across a range of evaluation scenarios.

LLaMA: Open and Efficient Foundation Language Models

- **Family:** Transformer
- **Pretraining Architecture:** Decoder
- **Corpus:** English CommonCrawl + C4 + Github + Wikipedia + Gutenberg and Books3 + ArXiv + Stack Exchange.
- **Versions:** 7B, 13B, 33B, 65B.
- LLaMA from Meta introduced several extensions: Pre-normalization, SwiGLU activations, RoPE embeddings, and uses 1.4T BPE tokens after tokenization.
- Open-Source culture…

- GPT-3 (2020)
  - 50,257 vocab size
  - 2048 context length
  - 175B parameters
  - Trained on 300B tokens
- Training (roughly)
  - 1000/10000 v100 GPUs
  - 1 month of training
  - 1/10 $M

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

- LLaMA (2023)
  - 32,000 vocab size
  - 2048 context length
  - 65B parameters
  - Trained on 1.4T tokens
- Training (roughly)
  - 2048 A100 GPUs
  - 21 days of training
  - 5 $M

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

Table 2: **Model sizes, architectures, and optimization hyper-parameters.**

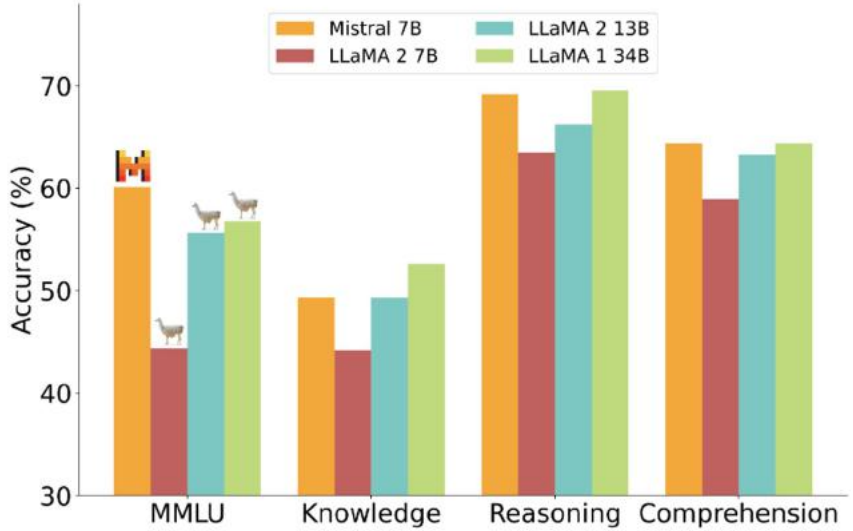
LLaMA2: Open Foundation and Fine-Tuned Chat Models

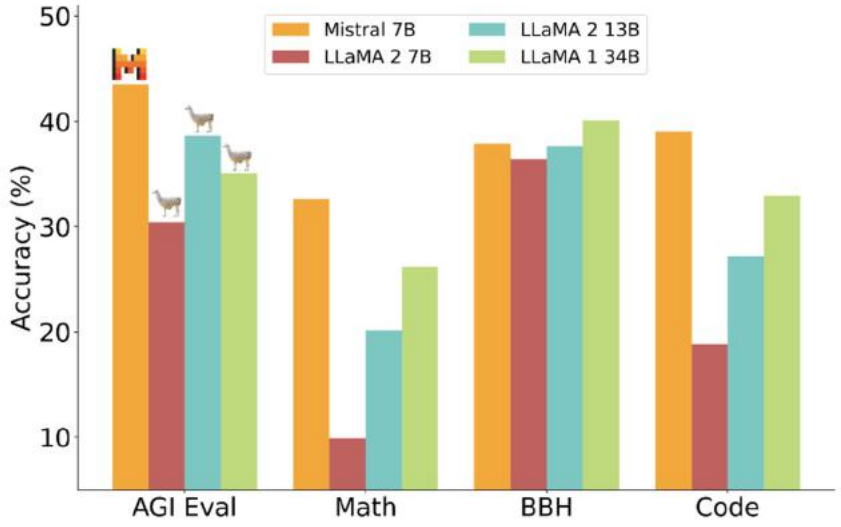
- **Corpus:** 2T of tokens a new mix of online data.
- **Versions:** 7B, 13B, 34B, 70B.
- LLaMA2 is trained on better pretraining data, including SFT and RLHF stages.
- Again Open-Source …

	Training Data	Params	Context Length	GQA	Tokens	LR
LLAMA 1	*See Touvron et al. (2023)*	7B	2k	✗	1.0T	3.0×10^{-4}
		13B	2k	✗	1.0T	3.0×10^{-4}
		33B	2k	✗	1.4T	1.5×10^{-4}
		65B	2k	✗	1.4T	1.5×10^{-4}
LLAMA 2	*A new mix of publicly available online data*	7B	4k	✗	2.0T	3.0×10^{-4}
		13B	4k	✗	2.0T	3.0×10^{-4}
		34B	4k	✓	2.0T	1.5×10^{-4}
		70B	4k	✓	2.0T	1.5×10^{-4}

Mistral 7B

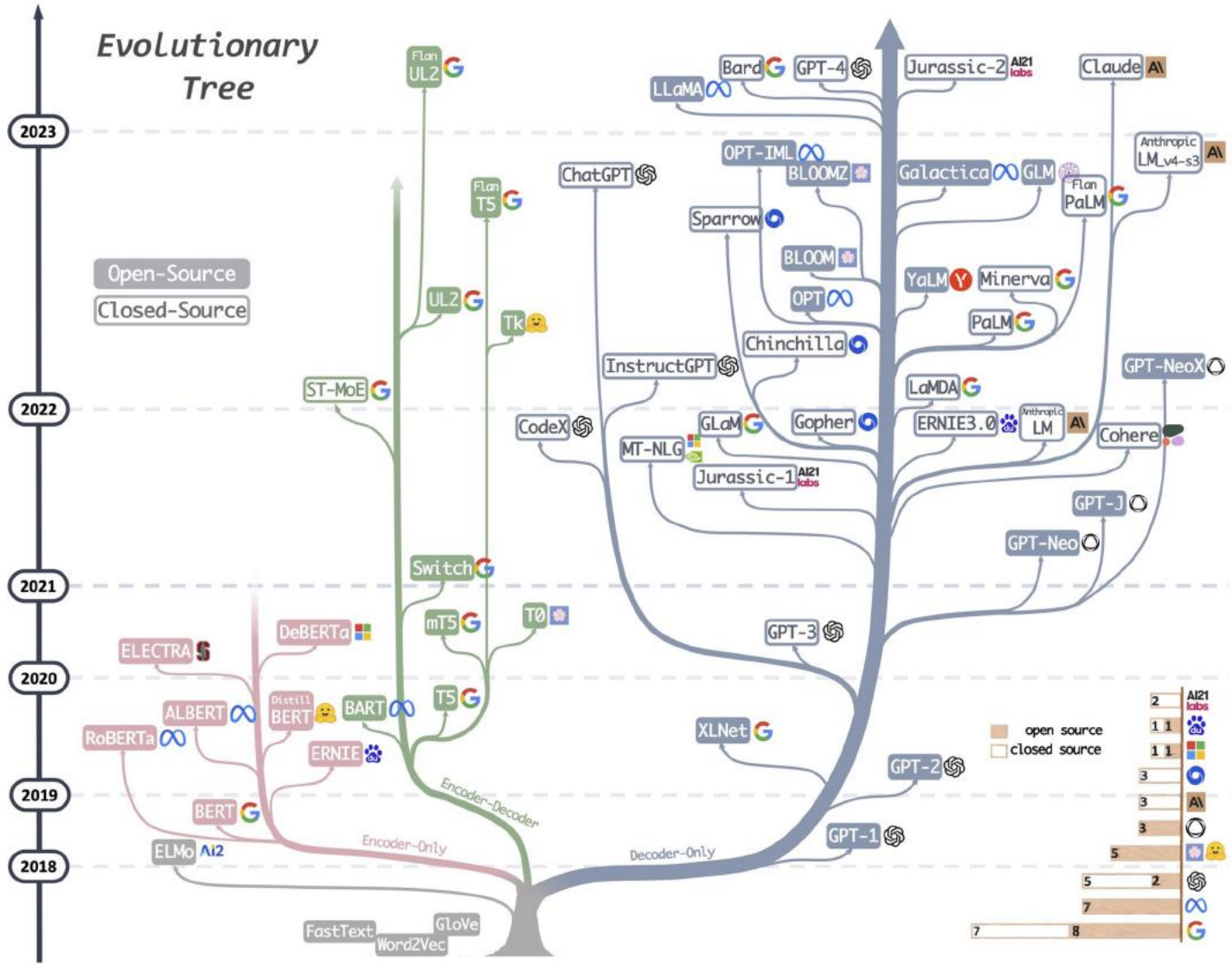
- **Pretraining Architecture:** Decoder
- **Corpus:** Very cleaned, don't know the details.
- **Version:** 7B
- Uses Grouped-query attention (GQA) for faster inference, uses Sliding Window Attention (SWA) to handle longer sequences at smalle





Base LLMs in the Wild!

- GPT, 2018
- GPT-2, 2018
- GPT-3, 2020
- Gopher, 2021
- OPT, 2022
- PaLM, 2022
- BLOOM, 2022
- GLaM, 2022
- Chinchilla, 2022
- LLaMA, 2023
- GPT-4, 2023
- Mistral, 2023
- LLaMA 2, 2023
- LLaMA 3, 2024



Pretraining

The input to the Transformer are arrays of (B,T)

- B is batch size
- T is the maximum context length

Training sequences are laid out as rows, delimited by <|endoftext|> tokens.

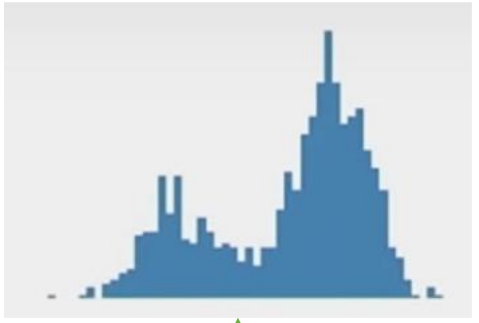
One training batch, array of shape (B,T)

T = 10

B = 4

4342	318	281	1672	3188	352	4478	617	16326	13
16281	3188	362	50256	16281	3188	513	50256	16281	3188
1212	318	617	4738	2420	655	329	1672	50256	1212
16	11	17	11	18	11	19	11	20	11

Pretraining



50,257 numbers

(probabilities for the next token)

Correct index (label): 513

Each cell only “sees” cells in its row, and only cells before it (on the left of it), to predict the next cell (on the right of it)

Green = current token

Yellow = its context

Red = its target

One training batch, array of shape (B,T)

T = 10

B = 4

4342	318	281	1672	3188	352	4478	617	16326	13
16281	3188	362	50256	16281	3188	513	50256	16281	3188
1212	318	617	4738	2420	655	329	1672	50256	1212
16	11	17	11	18	11	19	11	20	11

Training Process: Outpu

Training data (Shakespeare)

```
First Citizen:
We cannot, sir, we are undone already.

MENENIUS:
I tell you, friends, most charitable care
Have the patricians of you. For your wants,
Your suffering in this dearth, you may as well
Strike at the heaven with your staves as lift them
Against the Roman state, whose course will on
The way it takes, cracking ten thousand curbs
Of more strong link asunder than can ever
Appear in your impediment. For the dearth,
The gods, not the patricians, make it, and
Your knees to them, not arms, must help. Alack,
You are transported by calamity
Thither where more attends you, and you slander
The helms o' the state, who care for you like fathers,
When you curse them as enemies.
```

Samples at initialization

```
z'v}yy_RMV(7ea
AOCEi2tfEi lermh`
`88]gLNSSx|6Mj"i1wdcf,WezVII<4x?OBhS7D-}.8wCkGFgB(kC-
h'Ywa.QhjPo,3C.dA!3;_]!AKa.e0Ml lz(DqAfE8.}nm32<Z2ma1,6DAp
xOrA"jA[V;yhD]<g?BjKXbuptt|W:RT8,ti"(h8J"b"](ZPv3uExA.2r<&;wl?
`mnGs]MG8saNr3"u7tAfttthQBt`GEu66DxN'[[`LU!fUXhy!Ll2DjK a
b("8GL``Z66Dhv0,ooqv.
5nmUeh _'j}jjjW33EClY(5l
0vwdE;_Ze`veBbUv<y'TTBk(m]67q`1N`pd|EobQQ]RtKDXii0Y,LwOZ8d'y1)u
7d|N''ClE2y4hS"Ml0od3vtDVV<P``J1ONNn]Y4S<`Q}l2e9d2r8_
ccw[h'9TKFz]8lIDBlh'0y91i?<SKKL'sBv}v
```

Samples after 250 iterations of training

ONom hende beer'TIAFRO.
Rome thecoramerert BENRABENBUR. Nore se. he llod hears hy pid gof
wiere the the paron deread boan: ins wtherk hof at f o otherira coust Soot,
Hyou seealler sheron mer w f shathe thatchie anden wer by he thew bat

Samples after 500 iterations of training

For but te aser if the coouldlavilcoon Creater?
RANTEBR. In fease. Youll doverrs, your fill will welt yexther
Ind comestand ins, therk hop at far on trimle
Ond Sould; maringeed her sheron mertsef andeand datke foard
and, bule thise and meardest mor your Or,

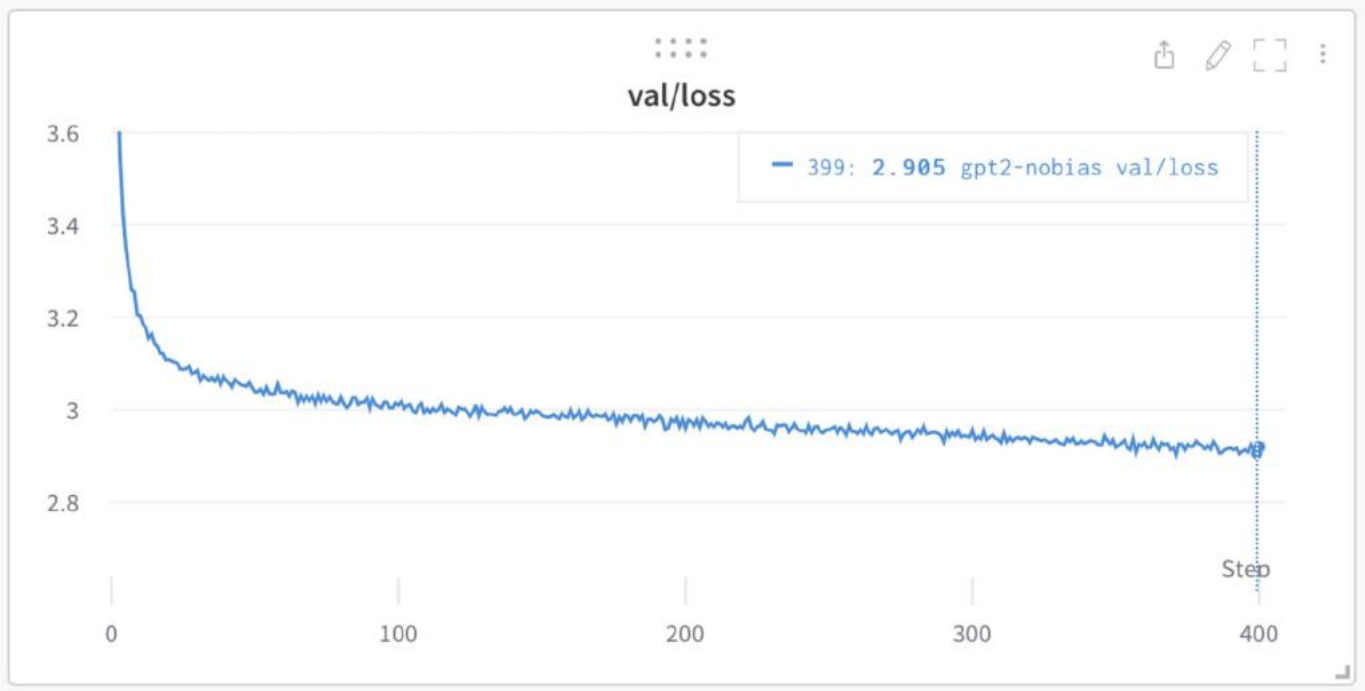
Samples after 5,000 iterations of training

Hor. I have been me, thereof my life, and he concludes him.
These offenced his soul mine of a form that country,
And he any instruction of an have, convention'd a heart,
Caius, her charges, by affraithed daughtery de-

Samples after 30,000 iterations of training

Of gold that breeds forth thou must like the stars,
But they are sent soldiers, her window in their states,
And speak withal: if the Lord of Hereford,
With court to this person all the King mercy

Training Process: Training Curve Examples



GPT toy data example

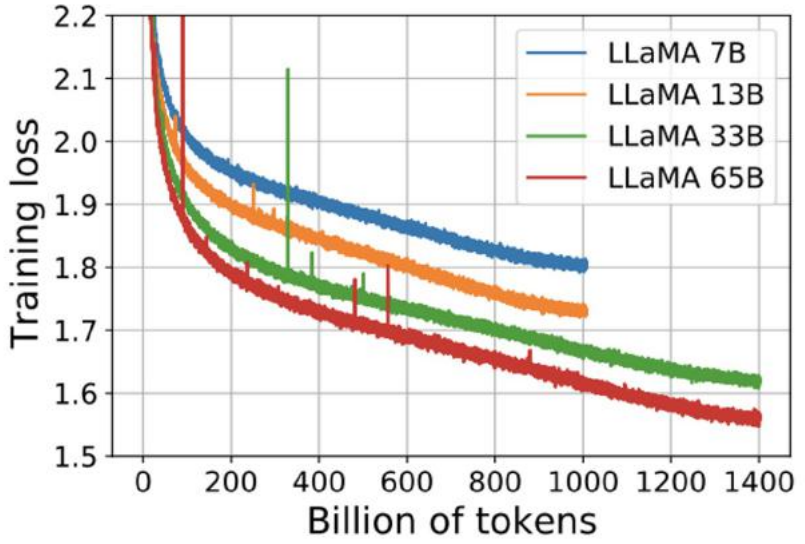
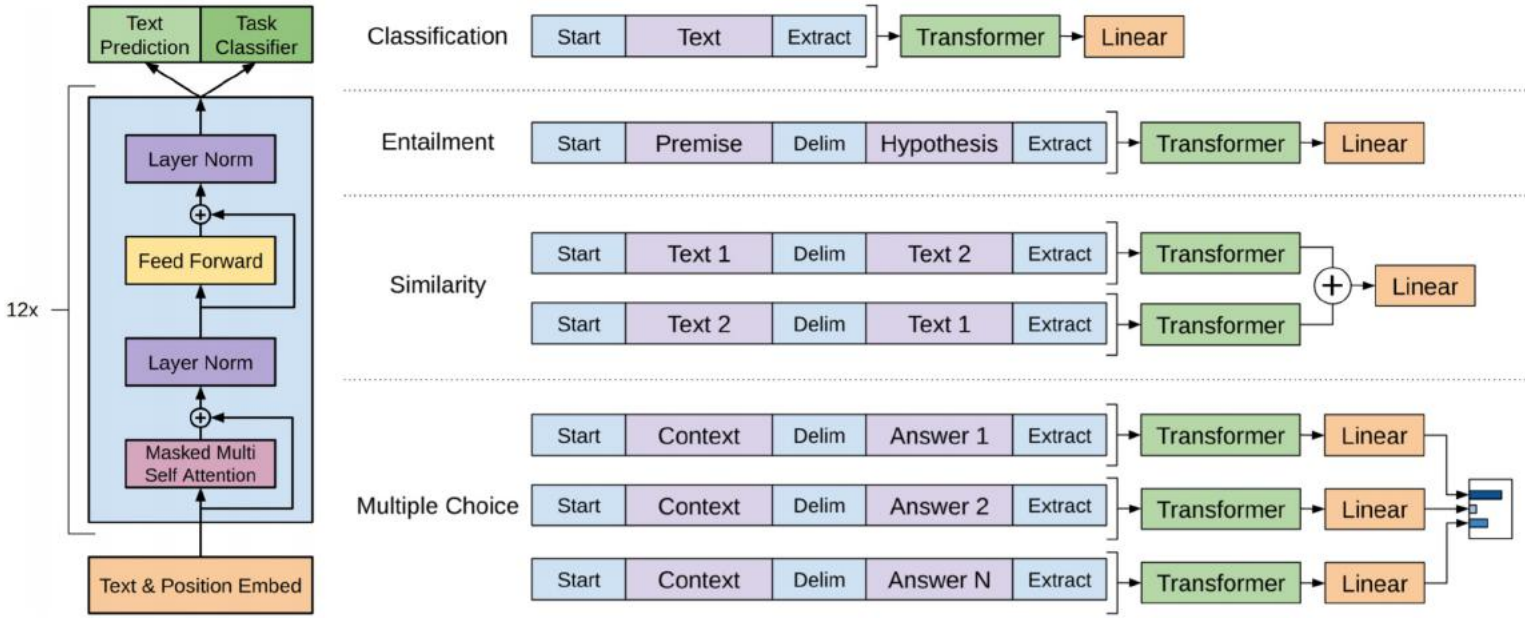


Figure 1: **Training loss over train tokens for the 7B, 13B, 33B, and 65 models.** LLaMA-33B and LLaMA-65B were trained on 1.4T tokens. The smaller models were trained on 1.0T tokens. All models are trained with a batch size of 4M tokens.

LLaMA

Training Process

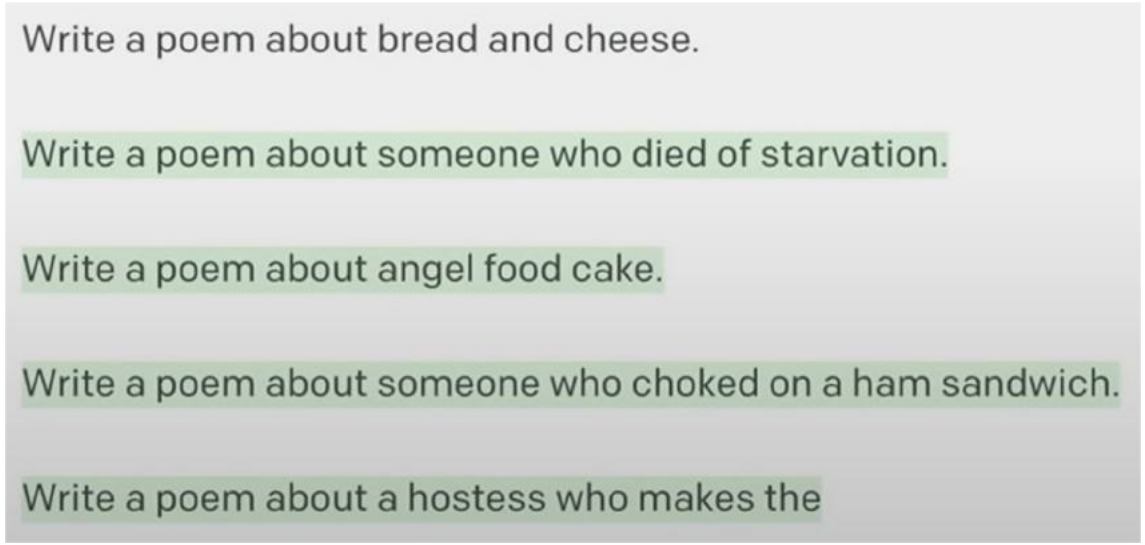
- **Step 1:** Model "pretraining" on large unsupervised dataset
- **Step 2:** Model "finetuning" on small supervised dataset



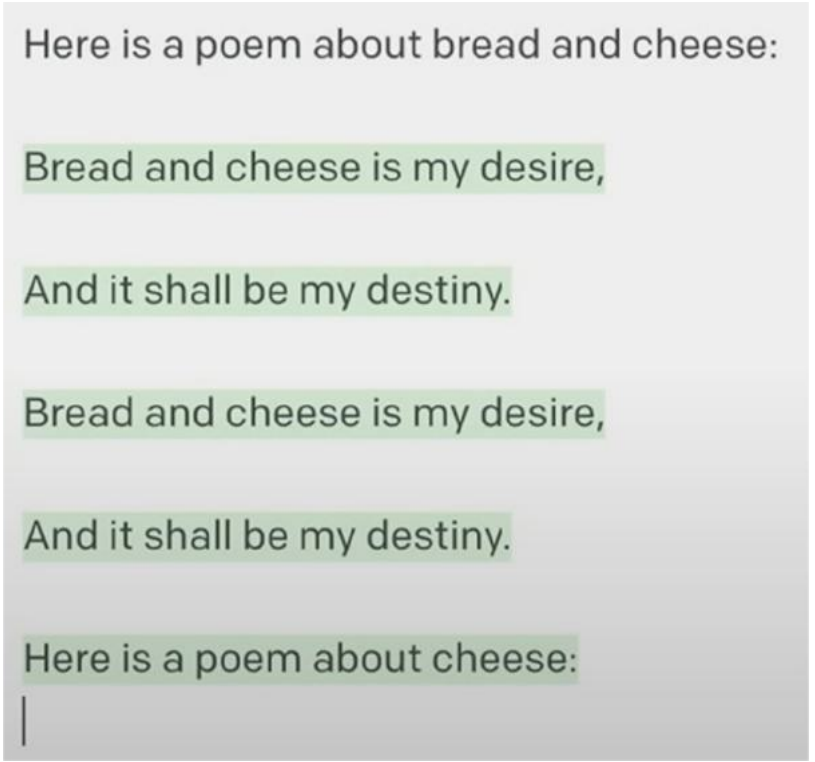
Improving Language Understanding by GPT, Radford et al. 2018

Base Model are NOT “Assistans”…

- Base model does not answer questions
- It only wants to complete internet documents
- Often responds to questions with more questions.



- It can be tricked into performing into tasks with prompt engineering:



LLaMA 2: Open Foundation and Fine-Tuned Chat Models

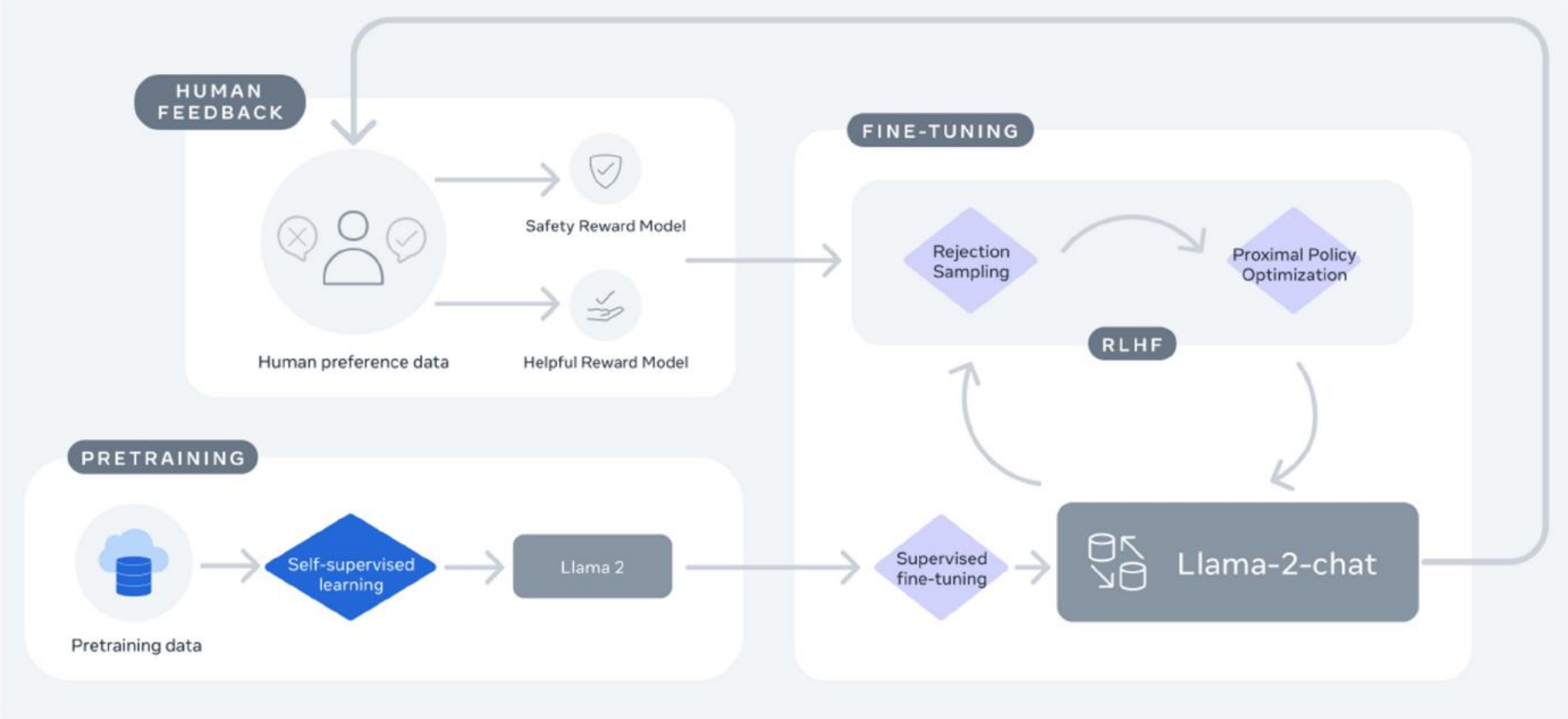
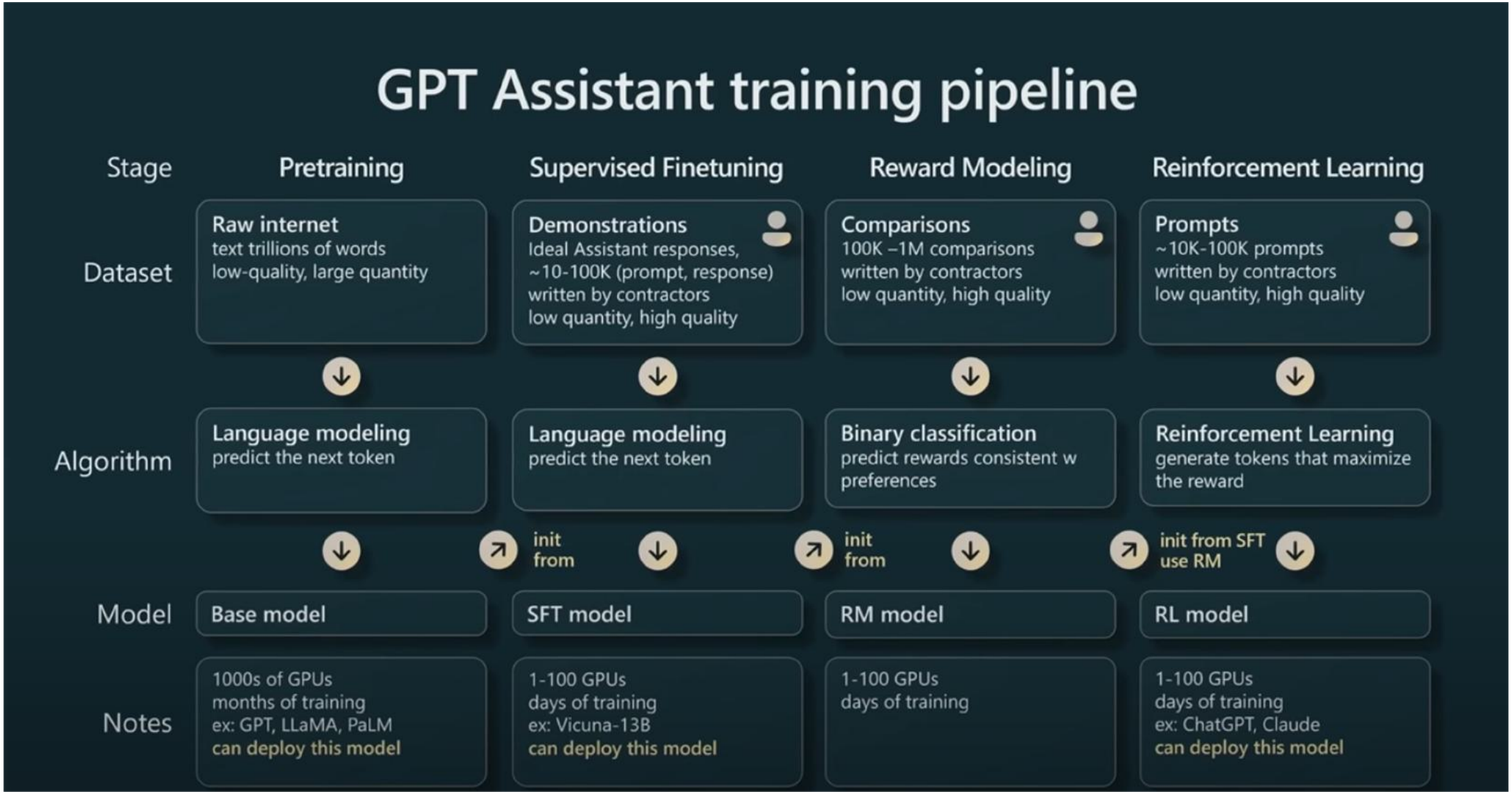


Figure 4: This process begins with the pretraining of Llama 2 using publicly available online sources. Following this, we create an initial version of Llama 2-Chat through the application of supervised fine-tuning. Subsequently, the model is iteratively refined using RLHF methodologies, specifically through rejection sampling and PPO. Throughout the RLHF stage, the accumulation of iterative reward modeling data in parallel with model enhancements is crucial to ensure the reward models remain within distribution.

SPoiler...


GPT Assistant training pipeline
Stage
Pretraining
Supervised Finetuning
Reward Modeling
Reinforcement Learning
Dataset
Raw internet
text trillions of words
low-quality, large quantity
Demonstrations
Ideal Assistant responses,
~10-100K (prompt, response)
written by contractors
low quantity, high quality
Comparisons
100K –1M comparisons
written by contractors
low quantity, high quality
Prompts
~10K-100K prompts
written by contractors
low quantity, high quality
Algorithm
Language modeling
predict the next token
Language modeling
predict the next token
Binary classification
predict rewards consistent w
preferences
Reinforcement Learning
generate tokens that maximize
the reward
init from
init from
init from SFT
use RM
Model
Base model
SFT model
RM model
RL model
Notes
1000s of GPUs
months of training
ex: GPT, LLaMA, PaLM
can deploy this model
1-100 GPUs
days of training
ex: Vicuna-13B
can deploy this model
1-100 GPUs
days of training
1-100 GPUs
days of training
ex: ChatGPT, Claude
can deploy this model

Next Lecture

- Supervised Finetuning (SFT)
  - Instruction Datasets
  - SFT Training
  - Assistant Models
- Preference Learning
  - RLHF / DPO
- In-Context Learning
  - Zero-Shot and Few-Shot
  - Chain-of-Thought
  - Self-Consistency
- Evaluating LLMs

Lecture overview

- Motivation and introduction
- Introduction to language models
- History of neural language models
- A digression into Transformers
- **Large Language Models**
- **Pre-training Framework**
- **Supervised Finetuning LLMs**
- **Parameter Efficient Finetuning**

Lecture overview

- Motivation and introduction
- Introduction to language models
- History of neural language models
- A digression into Transformers
- Large Language Models
- **Pre-training Framework**
- Supervised Finetuning LLMs
- Parameter Efficient Finetuning

GPT Assistant training pipeline

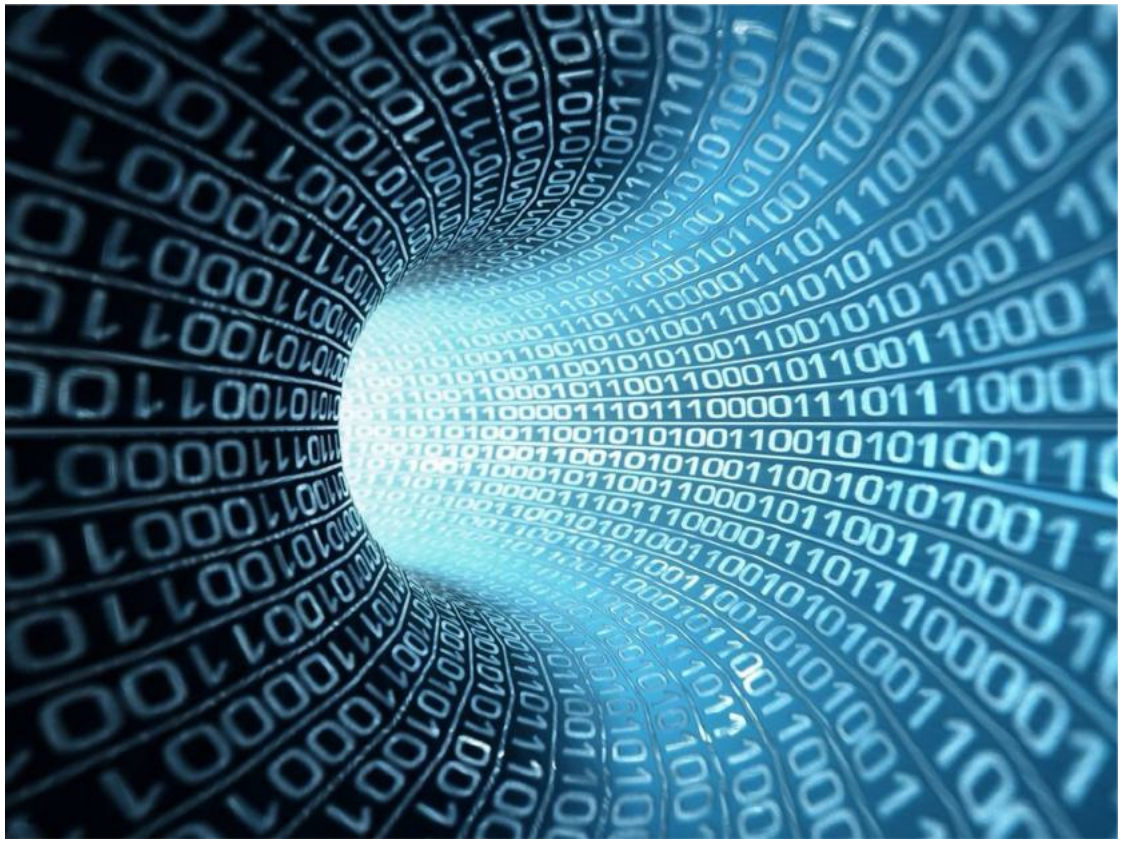
Stage	Pretraining	Supervised Finetuning	Reward Modeling	Reinforcement Learning
Dataset	**Raw internet** text trillions of words low-quality, large quantity	**Demonstrations** Ideal Assistant responses, ~10-100K (prompt, response) written by contractors low quantity, high quality	**Comparisons** 100K –1M comparisons written by contractors low quantity, high quality	**Prompts** ~10K-100K prompts written by contractors low quantity, high quality
Algorithm	**Language modeling** predict the next token	**Language modeling** predict the next token	**Binary classification** predict rewards consistent w preferences	**Reinforcement Learning** generate tokens that maximize the reward
		init from	init from	init from SFT use RM
Model	Base model	SFT model	RM model	RL model
Notes	1000s of GPUs months of training ex: GPT, LLaMA, PaLM can deploy this model	1-100 GPUs days of training ex: Vicuna-13B can deploy this model	1-100 GPUs days of training	1-100 GPUs days of training ex: ChatGPT, Claude can deploy this model

GPT Assistant training pipeline

Stage	Pretraining	Supervised Finetuning	Reward Modeling	Reinforcement Learning
Dataset	**Raw internet** text trillions of words low-quality, large quantity	**Demonstrations** Ideal Assistant responses, ~10-100K (prompt, response) written by contractors low quantity, high quality	**Comparisons** 100K –1M comparisons written by contractors low quantity, high quality	**Prompts** ~10K-100K prompts written by contractors low quantity, high quality
Algorithm	**Language modeling** predict the next token	**Language modeling** predict the next token	**Binary classification** predict rewards consistent w preferences	**Reinforcement Learning** generate tokens that maximize the reward
		init from	init from	init from SFT use RM
Model	Base model	SFT model	RM model	RL model
Notes	1000s of GPUs months of training ex: GPT, LLaMA, PaLM can deploy this model	1-100 GPUs days of training ex: Vicuna-13B can deploy this model	1-100 GPUs days of training	1-100 GPUs days of training ex: ChatGPT, Claude can deploy this model

Data Collection

Collect, download, and preprocess a large amount of available data.



Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Table 1: **Pre-training data.** Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 1.4T tokens, and disk size. The pre-training runs on 1T tokens have the same sampling proportion.

Data mixture used in LLaMA model

Pretraining

The input to the Transformer are arrays of (B,T)

- B is batch size
- T is the maximum context length

Training sequences are laid out as rows, delimited by <|endoftext|> tokens.

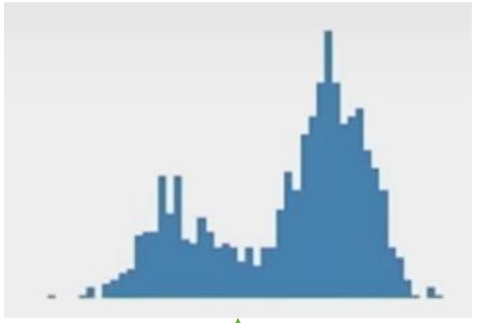
One training batch, array of shape (B,T)

T = 10

B = 4

4342	318	281	1672	3188	352	4478	617	16326	13
16281	3188	362	50256	16281	3188	513	50256	16281	3188
1212	318	617	4738	2420	655	329	1672	50256	1212
16	11	17	11	18	11	19	11	20	11

Pretraining



50,257 numbers

(probabilities for the next token)

Correct index (label): 513

Each cell only “sees” cells in its row, and only cells before it (on the left of it), to predict the next cell (on the right of it)

Green = current token

Yellow = its context

Red = its target

One training batch, array of shape (B,T)

T = 10

B = 4

4342	318	281	1672	3188	352	4478	617	16326	13
16281	3188	362	50256	16281	3188	513	50256	16281	3188
1212	318	617	4738	2420	655	329	1672	50256	1212
16	11	17	11	18	11	19	11	20	11

Training Process: Outpu

Training data (Shakespeare)

```
First Citizen:
We cannot, sir, we are undone already.

MENENIUS:
I tell you, friends, most charitable care
Have the patricians of you. For your wants,
Your suffering in this dearth, you may as well
Strike at the heaven with your staves as lift them
Against the Roman state, whose course will on
The way it takes, cracking ten thousand curbs
Of more strong link asunder than can ever
Appear in your impediment. For the dearth,
The gods, not the patricians, make it, and
Your knees to them, not arms, must help. Alack,
You are transported by calamity
Thither where more attends you, and you slander
The helms o' the state, who care for you like fathers,
When you curse them as enemies.
```

Samples at initialization

```
z'v}yy_RMV(7ea
AOCEi2tfEi lermh`
`88]gLNSSx|6Mj"i1wdcf,WezVII<4x?OBhS7D-}.8wCkGFgB(kC-
h'Ywa.QhjPo,3C.dA!3;_]!AKa.e0Ml lz(DqAfE8.}nm32<Z2ma1,6DAp
xOrA"jA[V;yhD]<g?BjKXbuptt|W:RT8,ti"(h8J"b"](ZPv3uExA.2r<&;wl?
`mnGs]MG8saNr3"u7tAfttthQBt`GEu66DxN'[[`LU!fUXhy!Ll2DjK a
b("8GL``Z66Dhv0,ooqv.
5nmUeh _'j}jjjW33EClY(5l
0vwdE;_Ze`veBbUv<y'TTBk(m]67q`1N`pd|EobQQ]RtKDXii0Y,LwOZ8d'y1)u
7d|N''ClE2y4hS"Ml0od3vtDVV<P``J1ONNn]Y4S<`Q}l2e9d2r8_
ccw[h'9TKFz]8lIDBlh'0y91i?<SKKL'sBv}v
```

Samples after 250 iterations of training

ONom hende beer'TIAFRO.
Rome thecoramerert BENRABENBUR. Nore se. he llod hears hy pid gof
wiere the the paron deread boan: ins wtherk hof at f o otherira coust Soot,
Hyou seealler sheron mer w f shathe thatchie anden wer by he thew bat

Samples after 500 iterations of training

For but te aser if the coouldlavilcoon Creater?
RANTEBR. In fease. Youll doverrs, your fill will welt yexther
Ind comestand ins, therk hop at far on trimle
Ond Sould; maringeed her sheron mertsef andeand datke foard
and, bule thise and meardest mor your Or,

Samples after 5,000 iterations of training

Hor. I have been me, thereof my life, and he concludes him.
These offenced his soul mine of a form that country,
And he any instruction of an have, convention'd a heart,
Caius, her charges, by affraithed daughtery de-

Samples after 30,000 iterations of training

Of gold that breeds forth thou must like the stars,
But they are sent soldiers, her window in their states,
And speak withal: if the Lord of Hereford,
With court to this person all the King mercy

- GPT-3 (2020)
  - 50,257 vocab size
  - 2048 context length
  - 175B parameters
  - Trained on 300B tokens
- Training (roughly)
  - 1000/10000 v100 GPUs
  - 1 month of training
  - 1/10 $M

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

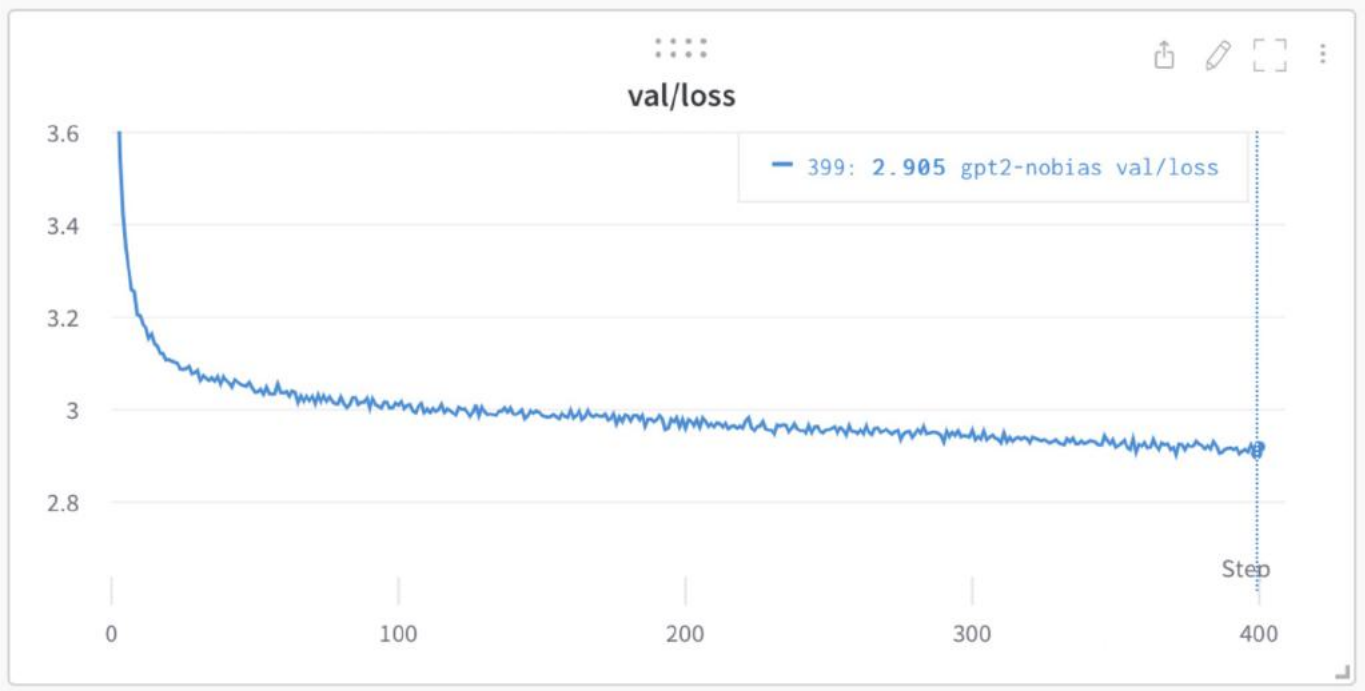
Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

- LLaMA (2023)
  - 32,000 vocab size
  - 2048 context length
  - 65B parameters
  - Trained on 1.4T tokens
- Training
  - 2048 A100 GPUs
  - 21 days of training
  - 5 $M

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

Table 2: **Model sizes, architectures, and optimization hyper-parameters.**

Training Process: Training Curve Examples



GPT toy data example

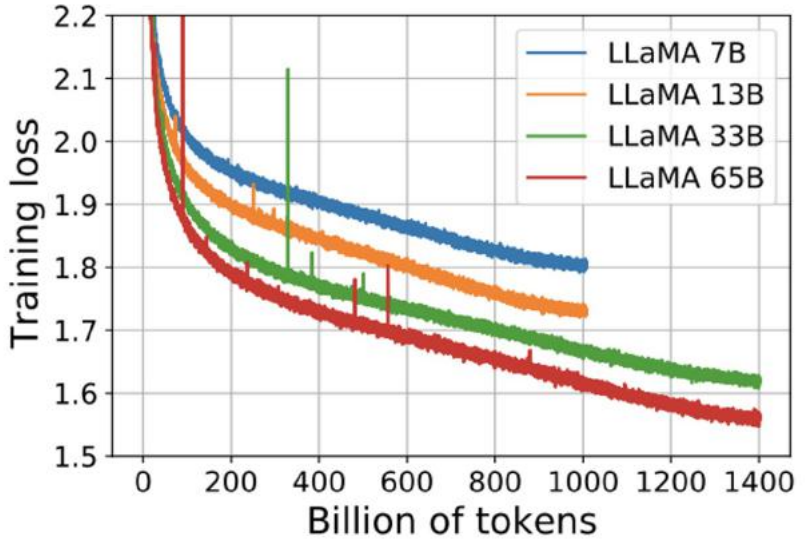
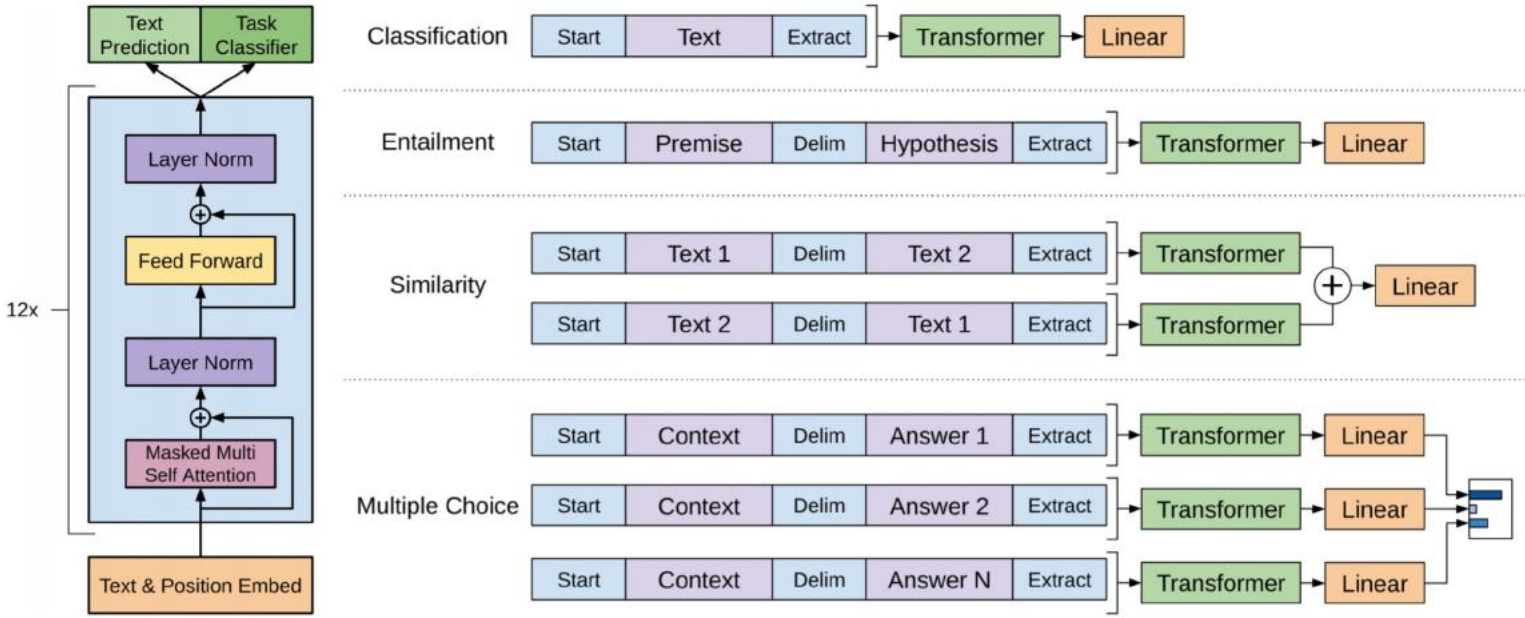


Figure 1: **Training loss over train tokens for the 7B, 13B, 33B, and 65 models.** LLaMA-33B and LLaMA-65B were trained on 1.4T tokens. The smaller models were trained on 1.0T tokens. All models are trained with a batch size of 4M tokens.

LLaMA

Training Process

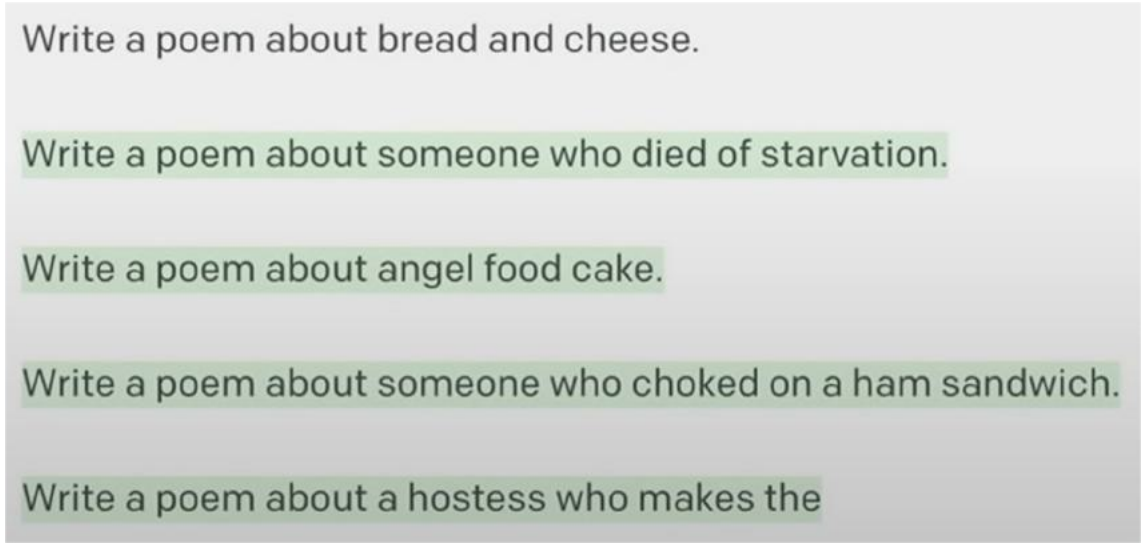
- **Step 1:** Model "pretraining" on large unsupervised dataset
- **Step 2:** Model "finetuning" on small supervised dataset



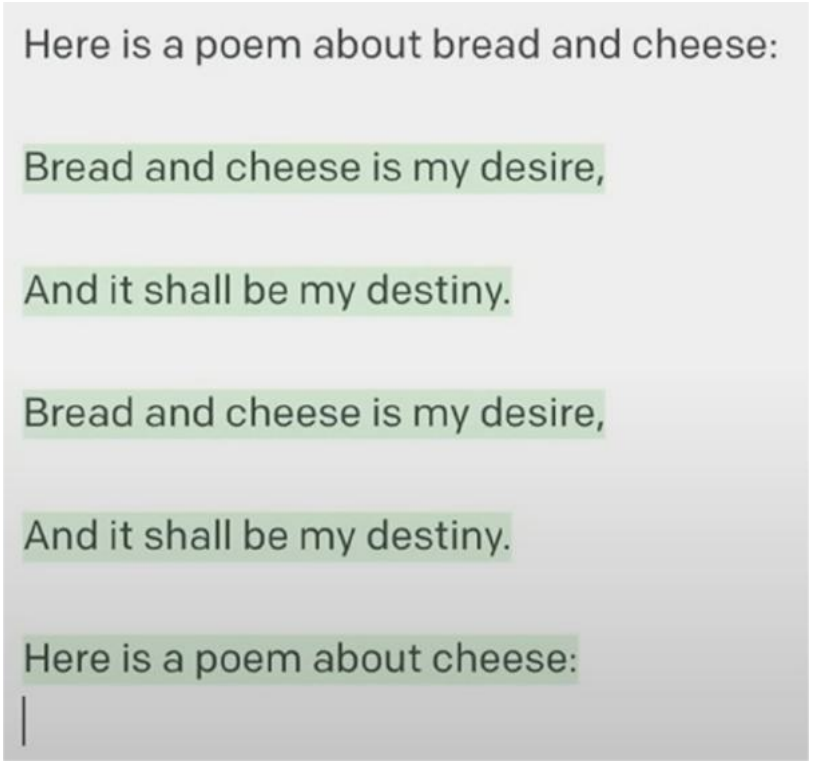
Improving Language Understanding by GPT, Radford et al. 2018

Base Model are NOT “Assistans”…

- Base model does not answer questions
- It only wants to complete internet documents
- Often responds to questions with more questions.

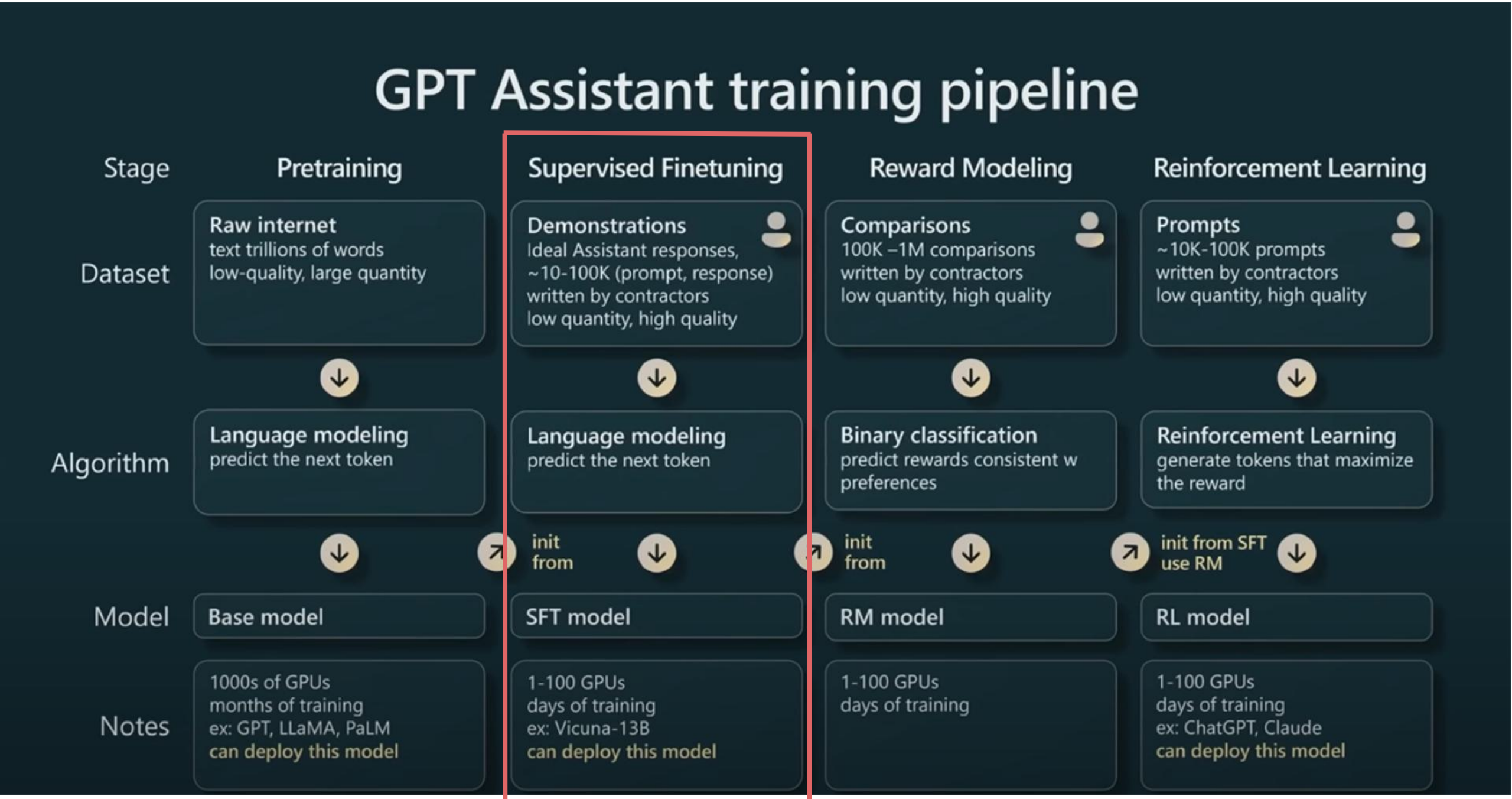


- It can be tricked into performing into tasks with prompt engineering:



Lecture overview

- Motivation and introduction
- Introduction to language models
- History of neural language models
- A digression into Transformers
- Large Language Models
- Pre-training Framework
- **Supervised Finetuning LLMs**
- Parameter Efficient Finetuning


GPT Assistant training pipeline
Stage
Pretraining
Supervised Finetuning
Reward Modeling
Reinforcement Learning
Dataset
Raw internet
text trillions of words
low-quality, large quantity
Demonstrations
Ideal Assistant responses,
~10-100K (prompt, response)
written by contractors
low quantity, high quality
Comparisons
100K –1M comparisons
written by contractors
low quantity, high quality
Prompts
~10K-100K prompts
written by contractors
low quantity, high quality
Algorithm
Language modeling
predict the next token
Language modeling
predict the next token
Binary classification
predict rewards consistent w
preferences
Reinforcement Learning
generate tokens that maximize
the reward
init
from
init
from
init from SFT
use RM
Model
Base model
SFT model
RM model
RL model
Notes
1000s of GPUs
months of training
ex: GPT, LLaMA, PaLM
can deploy this model
1-100 GPUs
days of training
ex: Vicuna-13B
can deploy this model
1-100 GPUs
days of training
1-100 GPUs
days of training
ex: ChatGPT, Claude
can deploy this model

Supervised Finetuning (SFT) LLMs

- Collect small but high quality datasets.
- Format is different this time:
  - Instruction
  - Input
  - Response

Supervised Finetuning (SFT) Samples

- **Instruction:** “Identify the odd one out."
- **Input:** "Twitter, Instagram, Telegram"
- **Output:** "Telegram"

Supervised Finetuning (SFT) Samples

- **Instruction:** “Identify the odd one out." => **[PAD]**
- **Input:** "Twitter, Instagram, Telegram" => **[PAD]**
- **Output:** "Telegram" => **Optimize**

Supervised Finetuning (SFT) Samples

- **Instruction:** “Identify the odd one out."
- **Input:** "Twitter, Instagram, Telegram"
- **Output:** "Telegram"

- **Instruction:** “Classify the following into animals, plants, and minerals"
- **Input:** "Oak tree, copper ore, elephant"
- **Output:** "Oak tree: Plant\n Copper ore: Mineral\n Elephant: Animal"

Supervised Finetuning (SFT) Samples

- **Instruction:** “Identify the odd one out."
- **Input:** "Twitter, Instagram, Telegram"
- **Output:** "Telegram"

- **Instruction:** “Classify the following into animals, plants, and minerals"
- **Input:** "Oak tree, copper ore, elephant"
- **Output:** "Oak tree: Plant\n Copper ore: Mineral\n Elephant: Animal"

- **Instruction:** “Rewrite the following sentence using active voice."
- **Input:** "The news report was read by the captain."
- **Output:** "The captain read the news report."

Supervised Finetuning (SFT) Samples

- **Instruction:** “Give three tips for staying healthy."
- **Input:** ""
- **Output:** "1.Eat a balanced diet and make sure to include plenty of fruits and vegetables. \n2. Exercise regularly to keep your body active and strong. \n3. Get enough sleep and maintain a consistent sleep schedule."

- **Instruction:** “What is the capital of France?"
- **Input:** ""
- **Output:** "The capital of France is Paris."

Supervised Finetuning (SFT) LLMs

- Collect small but high quality datasets.
- Format is different this time:
  - Instruction
  - Input
  - Response
- Again do Language Modeling, but optimize only for **responses**!
- Objective is to increase the **knowledge** and **reasoning** abilities of LLMs.

Alpaca Dataset

- Designed by Stanford, including 52k instruction-following demonstrations.
- LLaMA 7B model is finetuned on Alpaca.
- It is generated based on **Self-Instruct** method (automatic creation)

Alpaca Dataset

- Designed by Stanford, including 52k instruction-following demonstrations.
- LLaMA 7B model is finetuned on Alpaca.
- It is generated based on **Self-Instruct** method (automatic creation)
  - Using expert models to create datasets, e.g. text-davinci or ChatGPT.

Alpaca Dataset

- Designed by Stanford, including 52k instruction-following demonstrations.
- LLaMA 7B model is finetuned on Alpaca.
- It is generated based on **Self-Instruct** method (automatic creation)
  - Using expert models to create datasets, e.g. text-davinci or ChatGPT.
  - Design a careful prompt.

Alpaca Dataset

- Designed by Stanford, including 52k instruction-following demonstrations.
- LLaMA 7B model is finetuned on Alpaca.
- It is generated based on **Self-Instruct** method (automatic creation)
  - Using expert models to create datasets, e.g. text-davinci or ChatGPT.
  - Design a careful prompt.
  - Provide some seed samples.

Alpaca Dataset

- Designed by Stanford, including 52k instruction-following demonstrations.
- LLaMA 7B model is finetuned on Alpaca.
- It is generated based on **Self-Instruct** method (automatic creation)
  - Using expert models to create datasets, e.g. text-davinci or ChatGPT.
  - Design a careful prompt.
  - Provide some seed samples.
  - Let expert model generate more data, by following prompt and using the seeds.

Alpaca Dataset

- Designed by Stanford, including 52k instruction-following demonstrations.
- LLaMA 7B model is finetuned on Alpaca.
- It is generated based on **Self-Instruct** method (automatic creation)
  - Using expert models to create datasets, e.g. text-davinci or ChatGPT.
  - Design a careful prompt.
  - Provide some seed samples.
  - Let expert model generate more data, by following prompt and using the seeds.
  - Filtering!

LLaMA2: Open Foundation and Fine-Tuned Chat Models

- **Corpus:** 2T of tokens a new mix of online data.
- **Versions:** 7B, 13B, 34B, 70B.
- LLaMA2 is trained on better pretraining data, including SFT and RLHF stages.
- Again Open-Source …

	Training Data	Params	Context Length	GQA	Tokens	LR
LLAMA 1	*See Touvron et al. (2023)*	7B	2k	✗	1.0T	3.0×10^{-4}
		13B	2k	✗	1.0T	3.0×10^{-4}
		33B	2k	✗	1.4T	1.5×10^{-4}
		65B	2k	✗	1.4T	1.5×10^{-4}
LLAMA 2	*A new mix of publicly available online data*	7B	4k	✗	2.0T	3.0×10^{-4}
		13B	4k	✗	2.0T	3.0×10^{-4}
		34B	4k	✓	2.0T	1.5×10^{-4}
		70B	4k	✓	2.0T	1.5×10^{-4}

LLaMA 2: Open Foundation and Fine-Tuned Chat Models

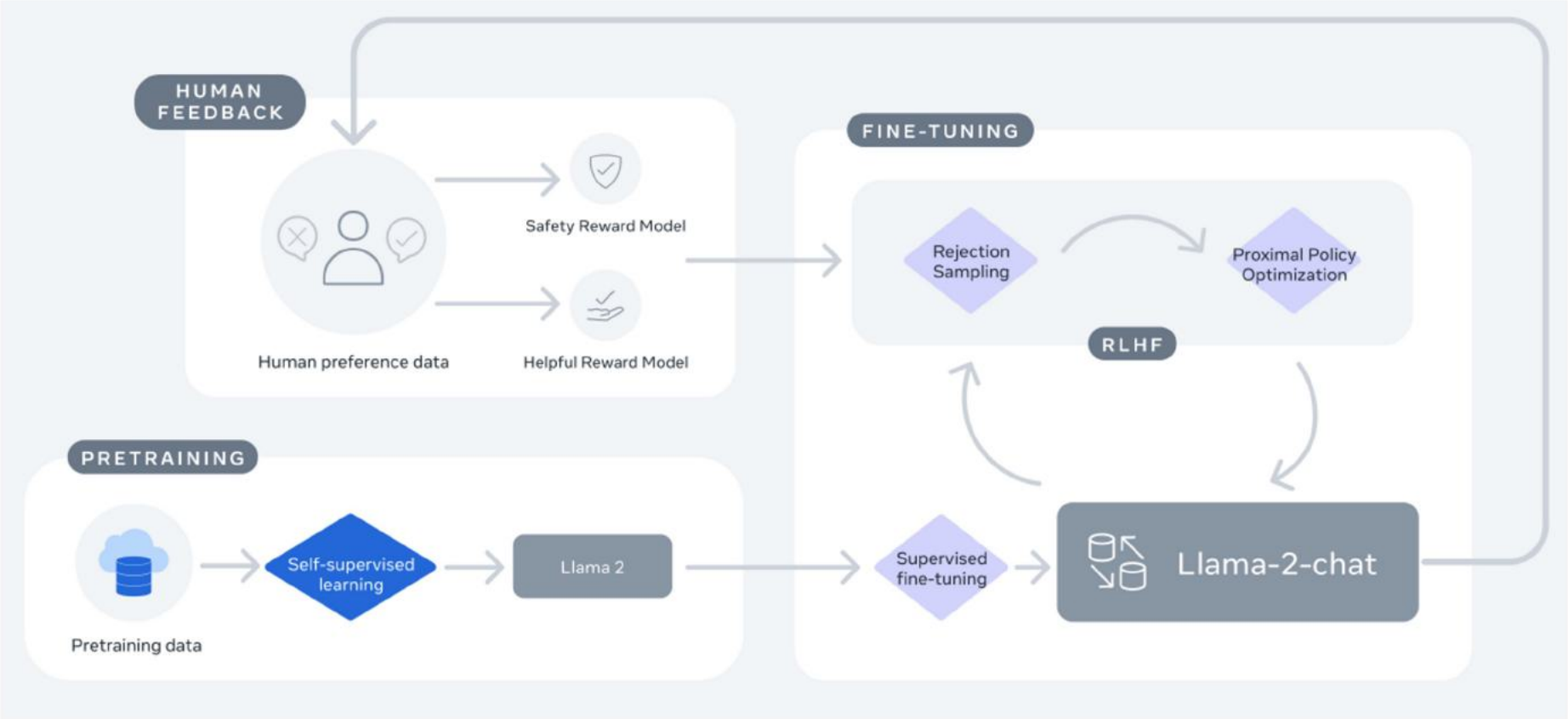


Figure 4: This process begins with the pretraining of Llama 2 using publicly available online sources. Following this, we create an initial version of Llama 2-Chat through the application of supervised fine-tuning. Subsequently, the model is iteratively refined using RLHF methodologies, specifically through rejection sampling and PPO. Throughout the RLHF stage, the accumulation of iterative reward modeling data in parallel with model enhancements is crucial to ensure the reward models remain within distribution.

Lecture overview

- Motivation and introduction
- Introduction to language models
- History of neural language models
- A digression into Transformers
- Large Language Models
- Pre-training Framework
- Supervised Finetuning LLMs
- **Parameter Efficient Finetuning**

Fine-tuning LLMs

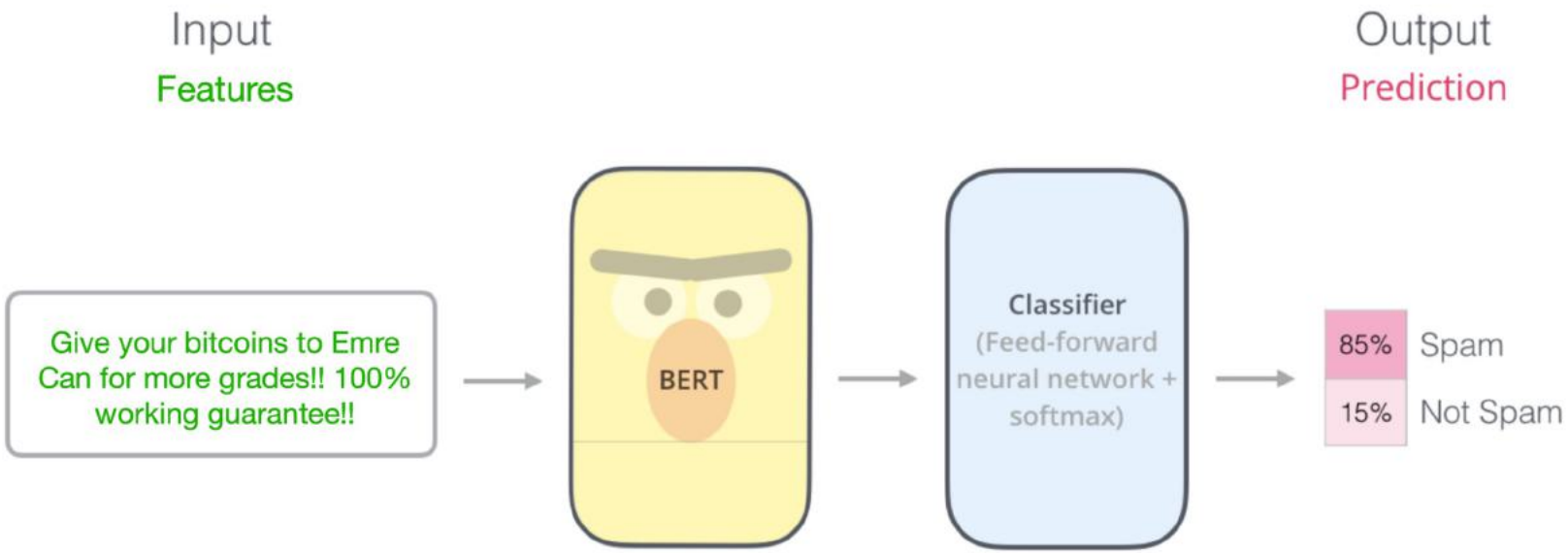
- **Fine-tuning:** using pre-trained LLMs to perform downstream tasks with supervised data by adding a linear head:

Fine-tuning LLMs

- **Fine-tuning:** using pre-trained LLMs to perform downstream tasks with supervised data by adding a linear head:
  - Text Summarization,
  - Question Answering,
  - Sentiment Analysis,
  - Machine Translation
  - NLI, ...

Fine-tuning LLMs

- **Fine-tuning:** using pre-trained LLMs to perform downstream tasks with supervised data by adding a linear head:
  - Text Summarization,
  - Question Answering,
  - Sentiment Analysis,
  - Machine Translation
  - NLI, ...



Parameter Efficient Finetuning

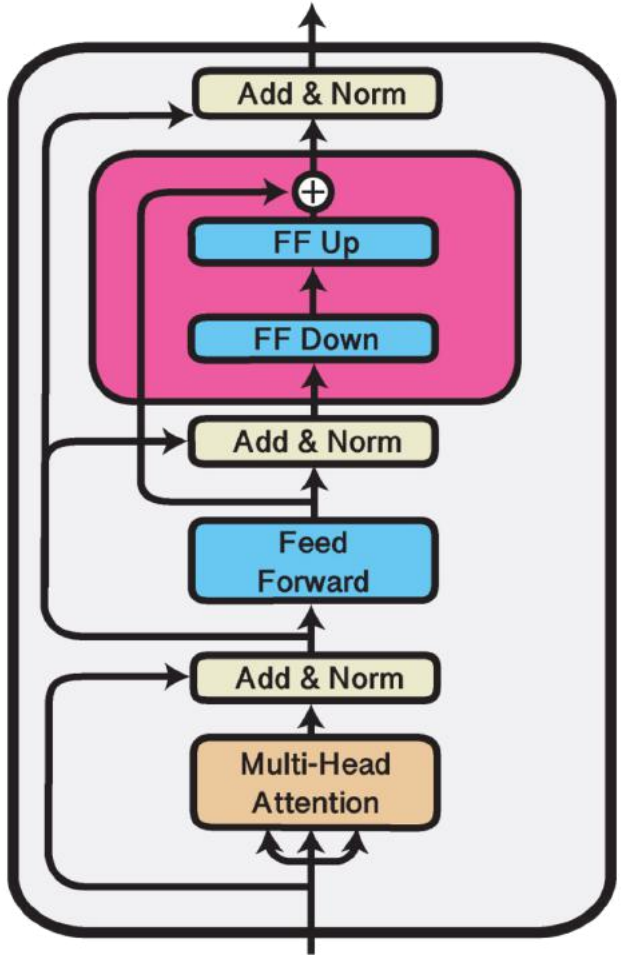
- Large pretrained models are **costly to fine-tune** due to their scale.
- Parameter-Efficient Fine-Tuning (PEFT) enables efficient adaptation of LLMs to diverse tasks by **adjusting a minimal number of parameters**.
- Significantly decreases the **computational** and **storage costs**.
- Recent state-of-the-art PEFT techniques achieve performance **comparable** to fully fine-tuned models.

Parameter Efficient Tuning Techniques

Lightweight-tuning

- Insert new layers and only update these, else freezed.
- Computationally much more efficient (2-3%)!

Parameter Efficient Tuning Tech

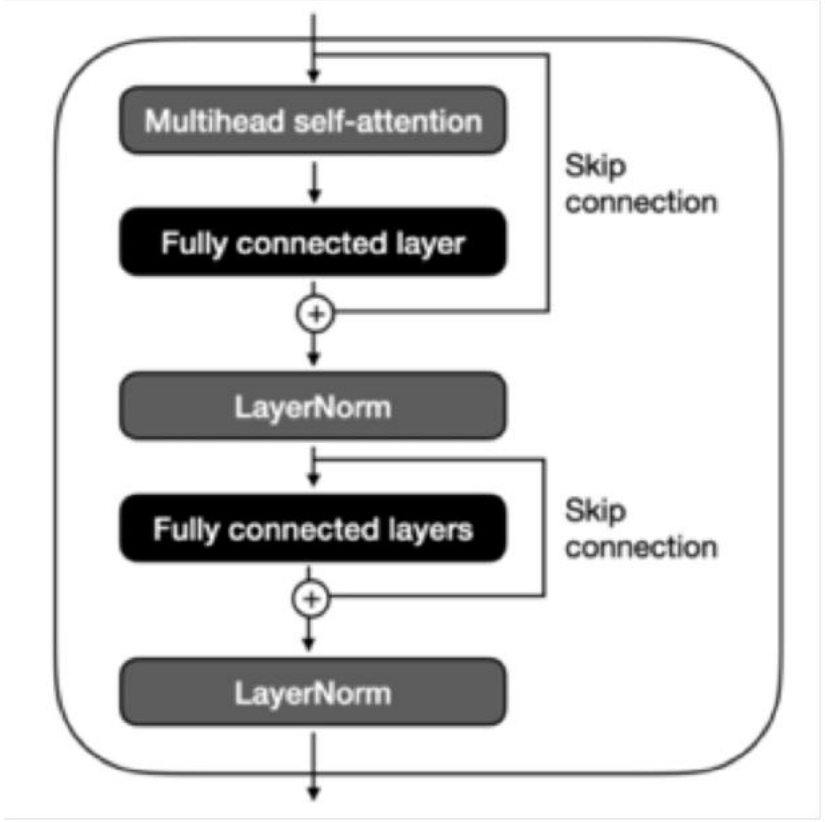


Lightweight-tuning

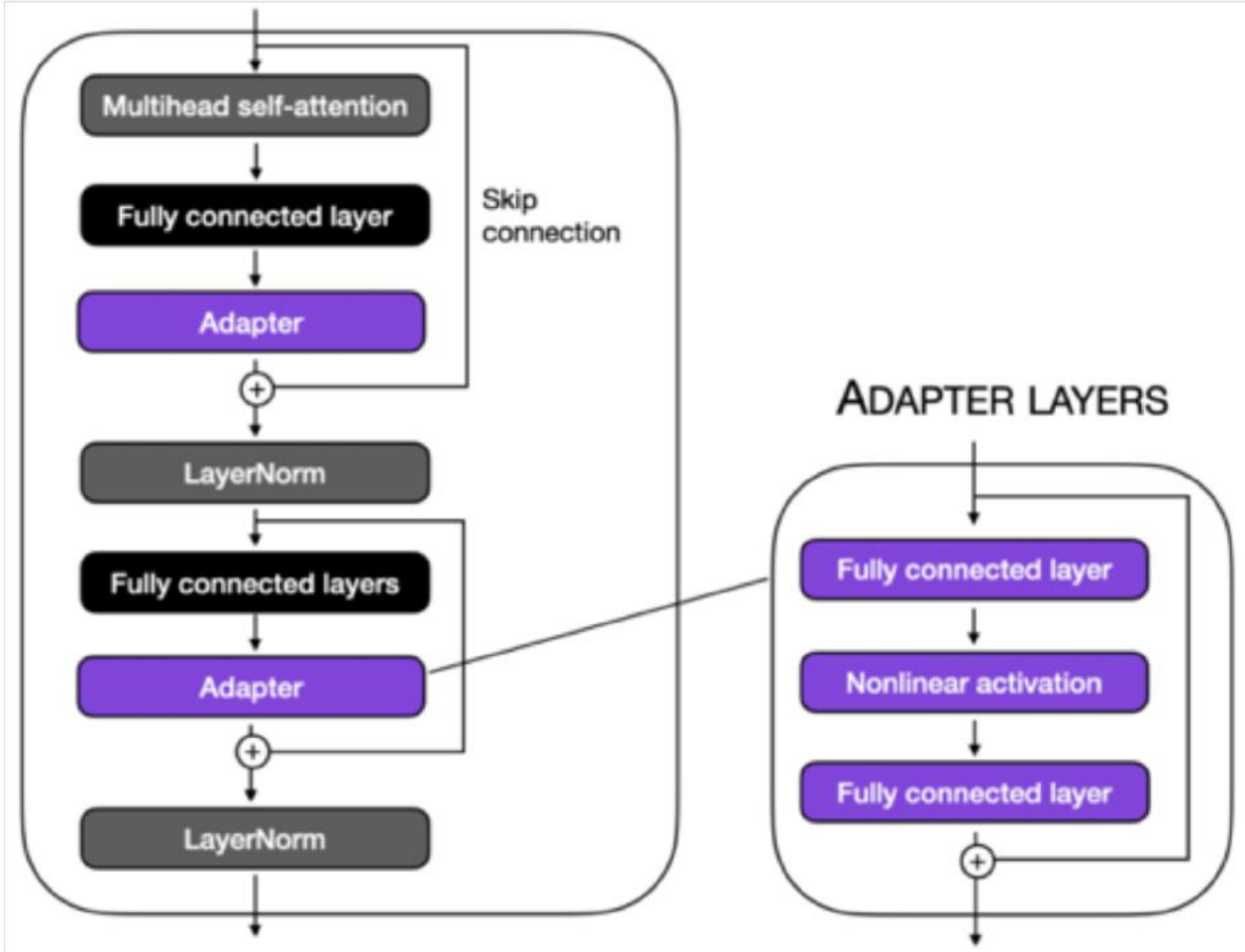
- Insert new layers and only update these, else freezed.
- Computationally much more efficient (2-3%)!

Parameter Efficient Tuning Techniques: Lightweight-tuning

Vanilla Transformer Block



Transformer Block with Adapters



Parameter Efficient Tuning Techniques

Lightweight-tuning

- Insert new layers and only update these, else freezed.
- Computationally much more efficient (2-3%)!

Prefix-tuning

- Insert new tokens (prefixes) and only update these, else freezed.
- Computationally extremely efficient (0.1%)!!!

Parameter Efficient Tuning Techniques

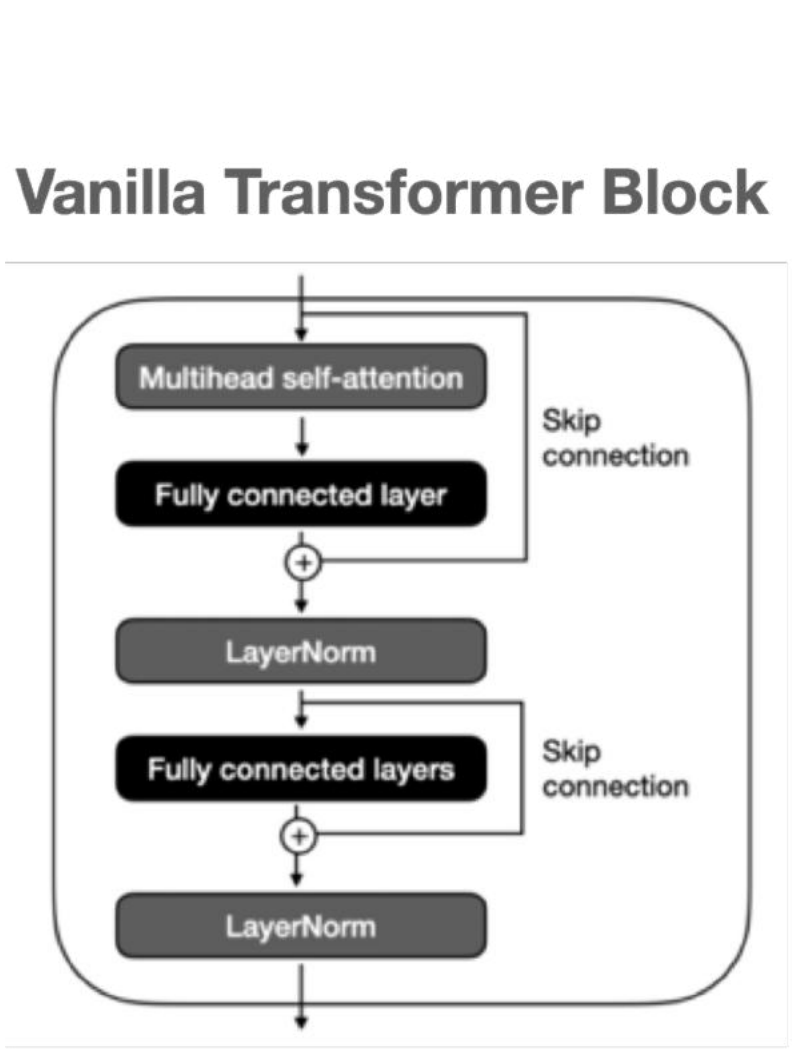
Lightweight-tuning

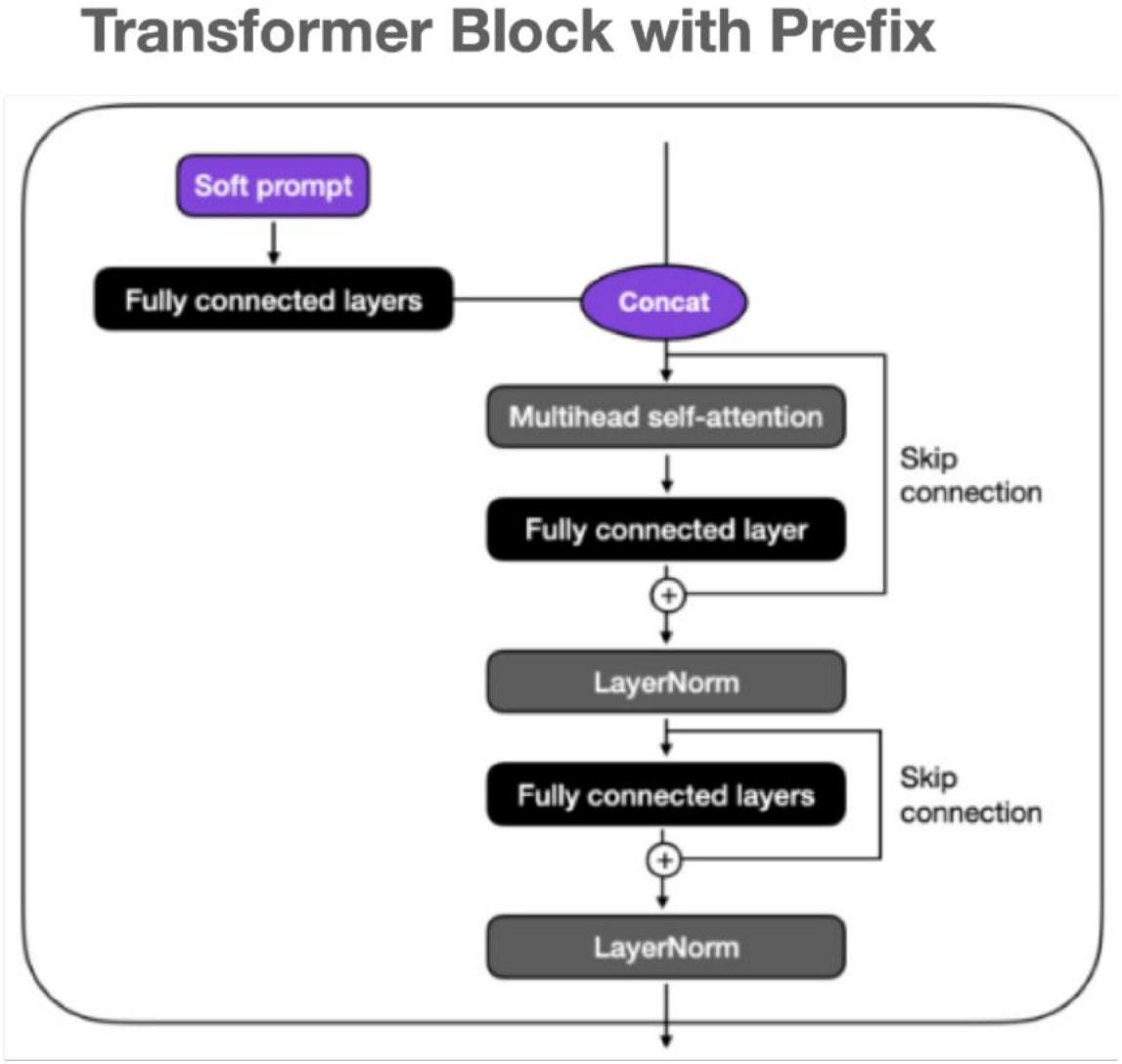
- Insert new layers and only update these, else freezed.
- Computationally much more efficient (2-3%)!

Prefix-tuning

- Insert new tokens (prefixes) and only update these, else freezed.
- Computationally extremely efficient (0.1%)!!!

Parameter Efficient Tuning Techniques: Prefix-tuning





Parameter Efficient Tuning Techniques

Lightweight-tuning

- Insert new layers and only update these, else freezed.
- Computationally much more efficient (2-3%)!

Prefix-tuning

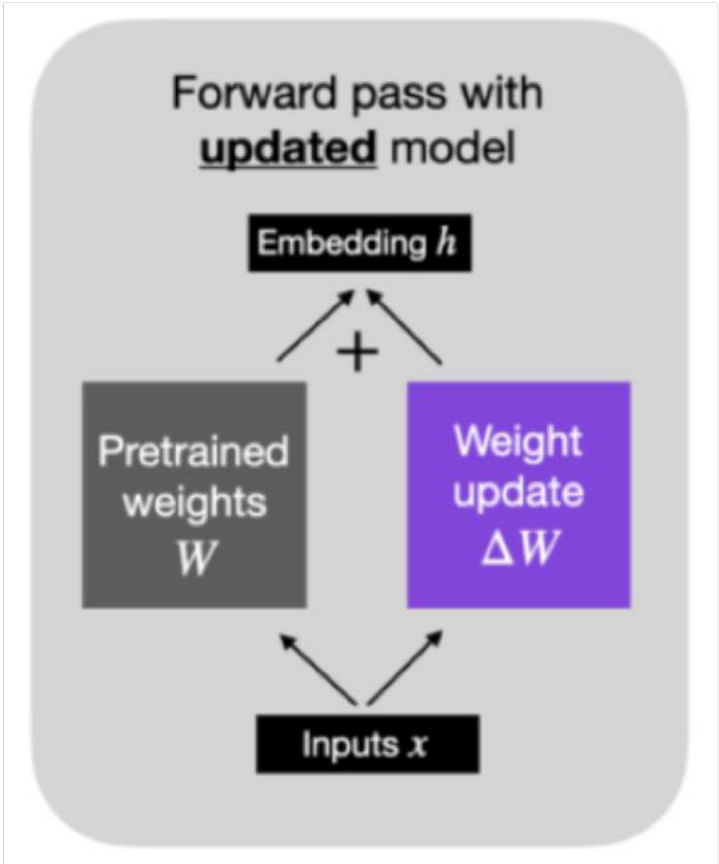
- Insert new tokens (prefixes) and only update these, else freezed.
- Computationally extremely efficient (0.1%)!!!

Re-parameterization

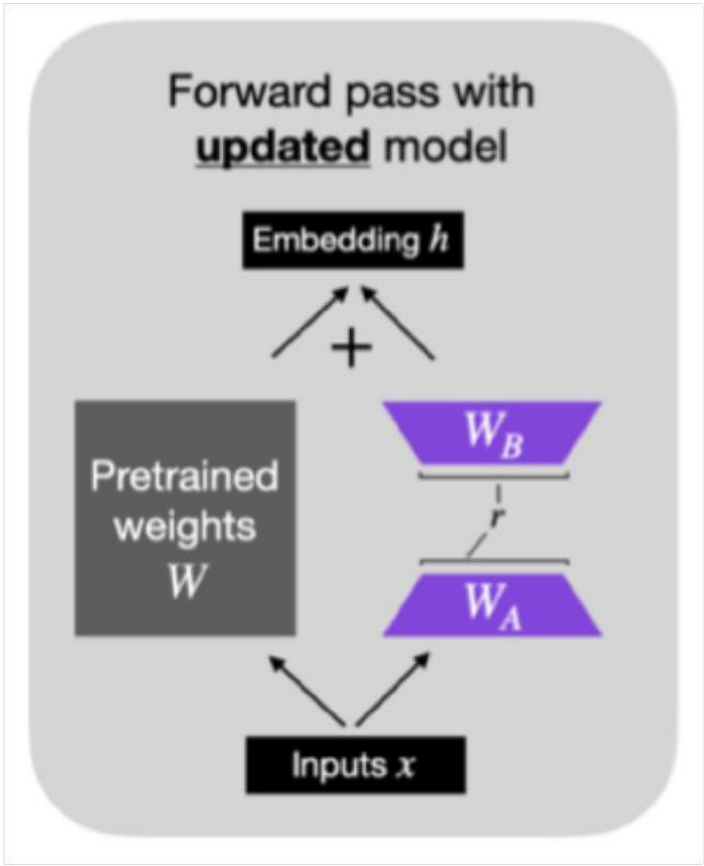
- Decompose the weight changes, ΔW, into a lower-rank representation.
- Train $W' = W + W_A W_B$, which is analogous to $W' = W + \Delta W$.

Parameter Efficient Tuning Techniques: LoRA

Regular Fine-tuning



LoRA weights W_A and W_B, represent ΔW



Code Examples - Nothing Complicated...

- **Vanilla Transformer**

```
def transformer_block_with_adapter(x):
    residual = x
    x = self_attention(x)
    x = LayerNorm(x + residual)
    residual = x
    x = FullyConnectedLayers(x)
    x = LayerNorm(x + residual)
    return x
```

- **Adapter**

```
def transformer_block_with_adapter(x):
    residual = x
    x = self_attention(x)
    x = AdapterLayers(x) # adapter
    x = LayerNorm(x + residual)
    residual = x
    x = FullyConnectedLayers(x)
    x = AdapterLayers(x) # adapter
    x = LayerNorm(x + residual)
    return x
```

Code Examples - Nothing Complicated...

- **Prefix-Tuning**

```
def transformer_block_with_prefix(x, soft_prompt, seq_len):
    soft_prompt = FullyConnectedLayers(soft_prompt) # prefix
    x = torch.concat(
        [soft_prompt, x], # add prefix
        dim=seq_len
    )
    residual = x
    x = self_attention(x)
    x = LayerNorm(x + residual)
    residual = x
    x = FullyConnectedLayers(x)
    x = LayerNorm(x + residual)
    return x
```

Next lecture:
Pretraining for Vision and Language