Pre-training & In-Context Learning

Shansan Gong & Lei Li

2023/09

{hisansas, nlp.lilei}@gmail.com

Pre-training

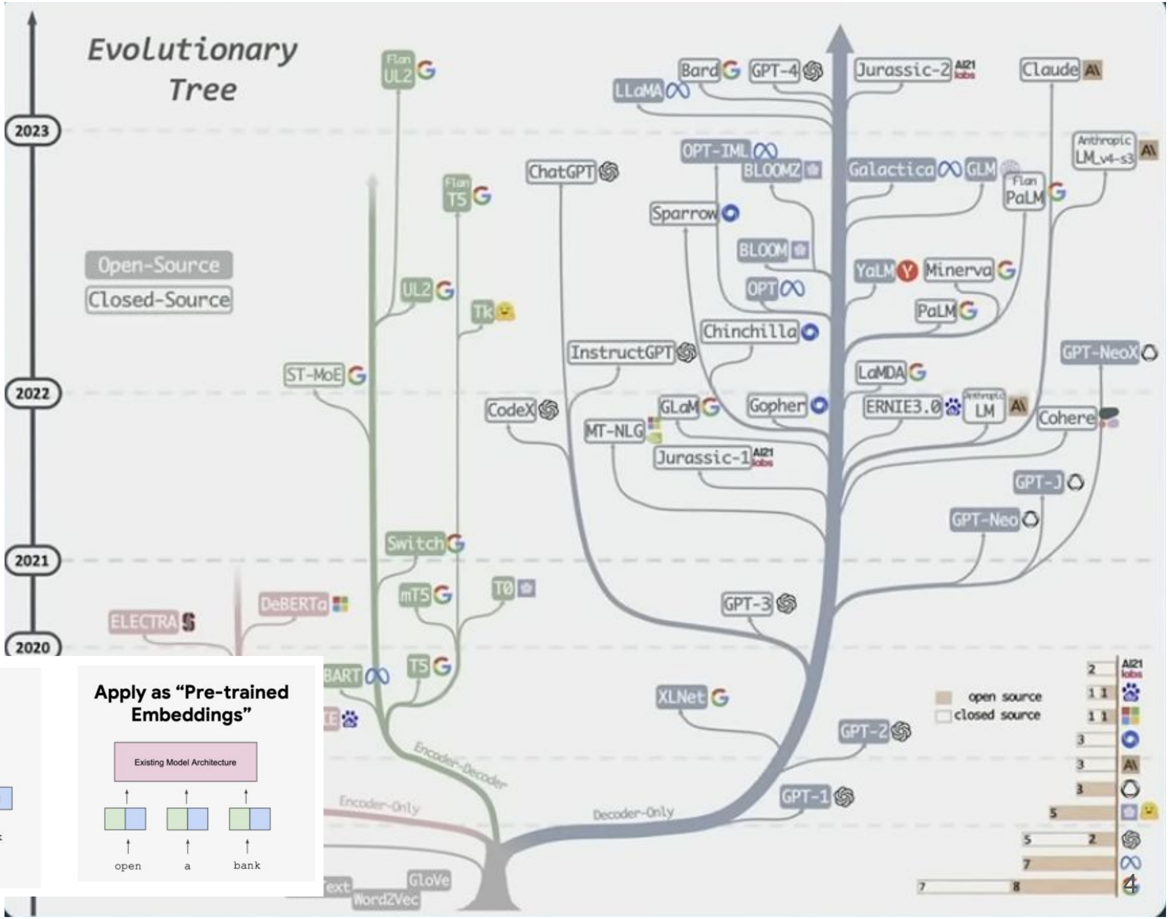
Outlines

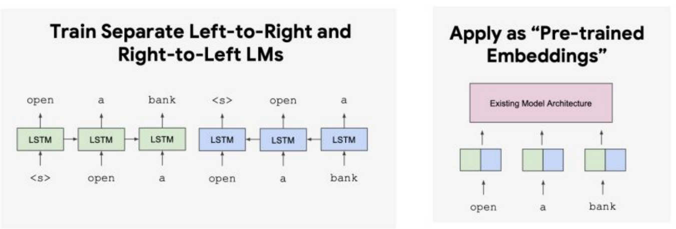
- Overview of PLM/LLM
- The pipeline of PLM
- Some training details
- Paper reading

Evolution Tree of PLM

ELMo:
Based on LSTM, pre-trained representations

Encoder-only:
 BERT (MLM, NSP)
 RoBERTa, …



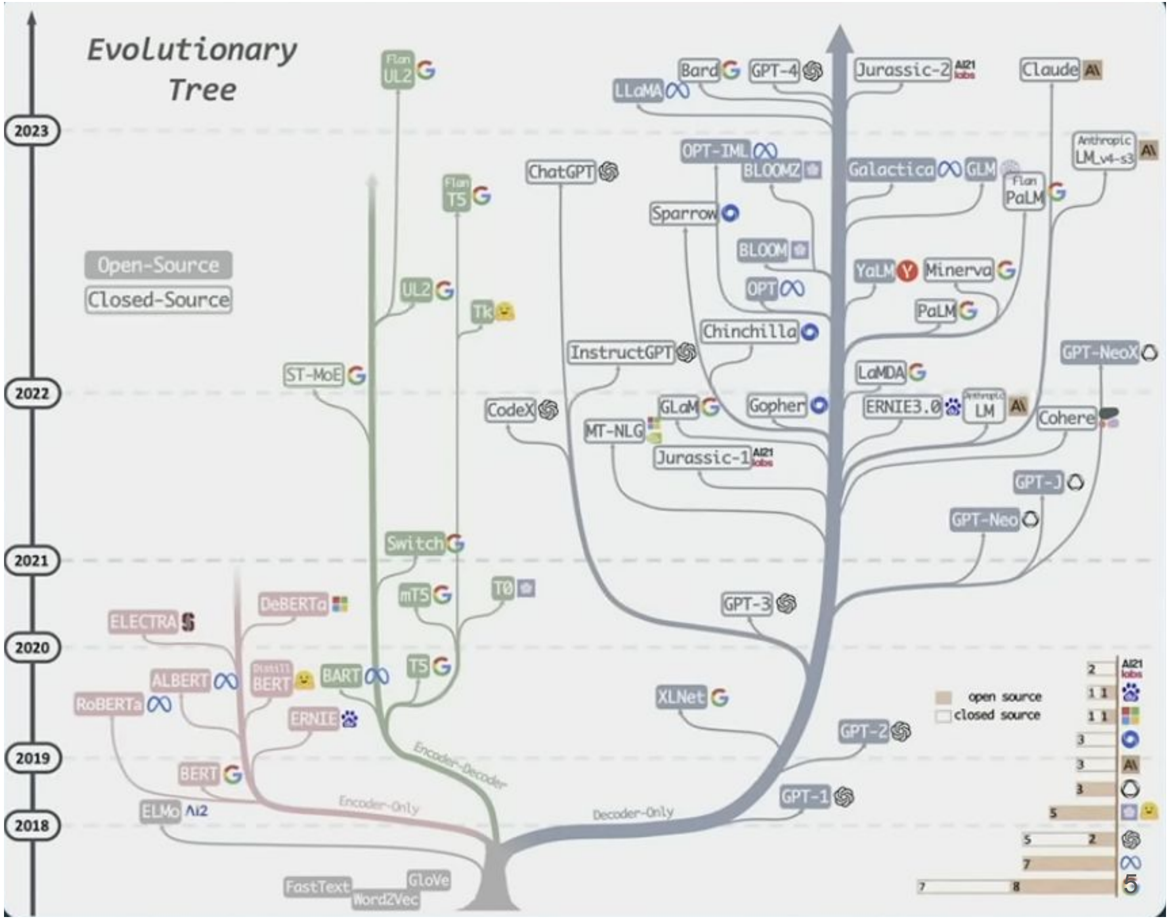


Evolution Tree of PLM

Encoder-only:
BERT, RoBERTa, …

Encoder-decoder:
BART, T5, …

Decoder-only:
GPT, OPT, Chinchilla, LLaMa, …



Data collection

Usually publicly 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 processing:

(1) Filtering
(2) Deduplication

Data recipe

(training data from LLaMA model…)

ref: LLaMA: Open and Efficient Foundation Language Model, 2023

Tokenization

Tokenization
Transform all text into one very long list of integers

Typical numbers:
~10-100K possible tokens
1 token ~= 0.75 of word

Typical algorithm:
Byte Pair Encoding

raw text

```
The GPT family of models process text using tokens, which are common
sequences of characters found in text. The models understand the
statistical relationships between these tokens, and excel at producing
the next token in a sequence of tokens.

You can use the tool below to understand how a piece of text would be
tokenized by the API, and the total count of tokens in that piece of
text.
```

tokens

```
The GPT family of models process text using tokens, which are common
sequences of characters found in text. The models understand the
statistical relationships between these tokens, and excel at producing
the next token in a sequence of tokens.

You can use the tool below to understand how a piece of text would be
tokenized by the API, and the total count of tokens in that piece of
text.
```

integers

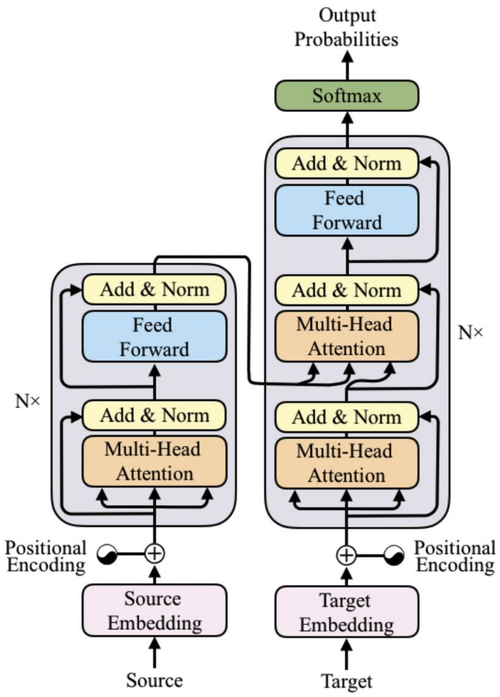
```
[464, 402, 11571, 1641, 286, 4981, 1429, 2420, 1262, 16326, 11, 543, 389,
2219, 16311, 286, 3435, 1043, 287, 2420, 13, 383, 4981, 1833, 262, 13905,
6958, 1022, 777, 16326, 11, 290, 27336, 379, 9194, 262, 1306, 11241, 287,
257, 8379, 286, 16326, 13, 198, 198, 1639, 460, 779, 262, 2891, 2174,
284, 1833, 703, 257, 3704, 286, 2420, 561, 307, 11241, 1143, 416, 262,
7824, 11, 290, 262, 2472, 954, 286, 16326, 287, 326, 3704, 286, 2420, 13]
```

Architecture

Encoder-Decoder

Position Embedding:

Learned PE (absolute)
Relative PE: Sinusoidal PE
ALiBi, RoPE, xPos



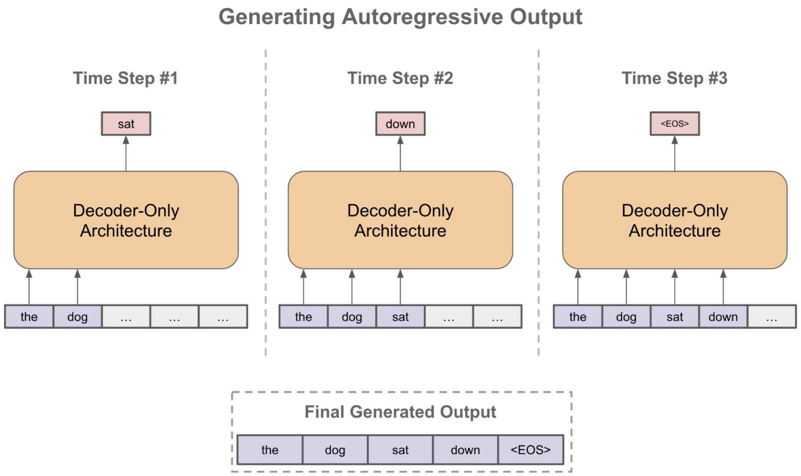
ref: Attention Is All You Need, 2017

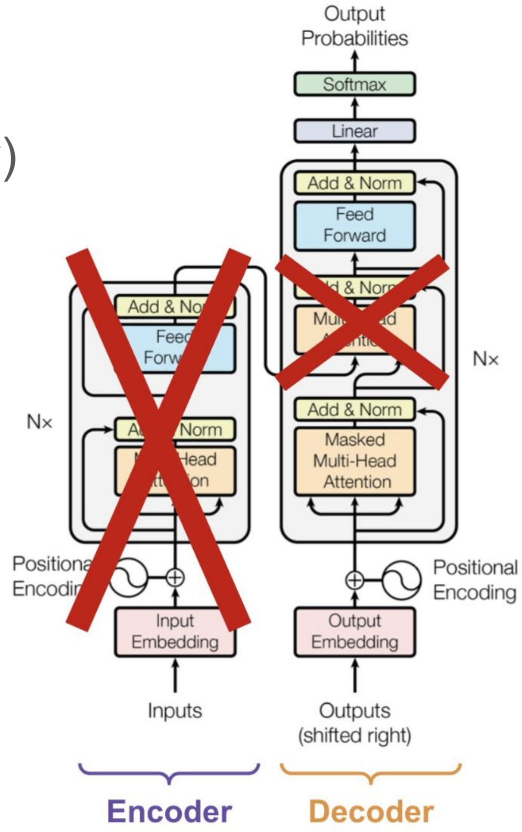
- [The Illustrated Transformer]
- [The Annotated Transformer]
- [HuggingFace's course on Transformers]

ref: The Impact of Positional Encoding on Length Generalization in Transformers, 2023

Architecture

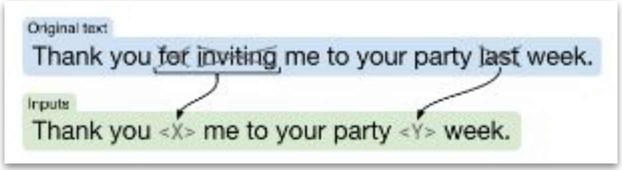
Decoder-only
(causal language model, auto-regressively)

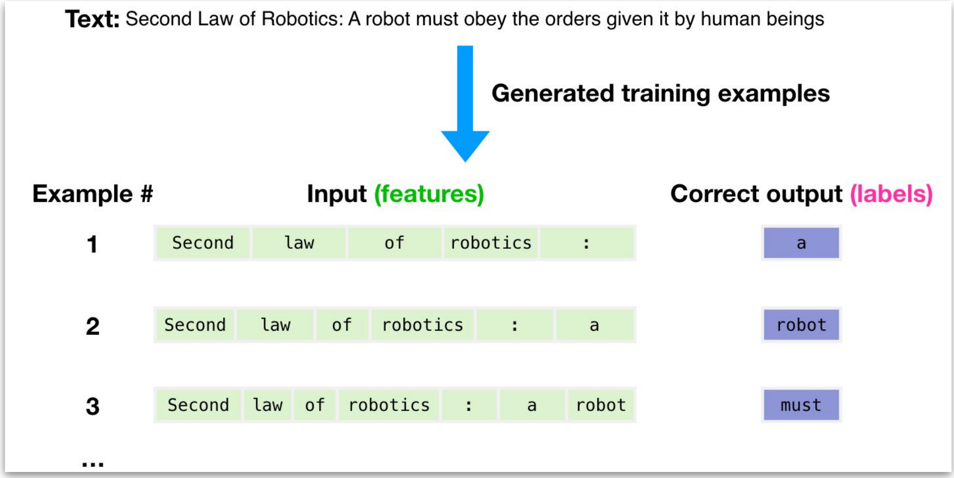




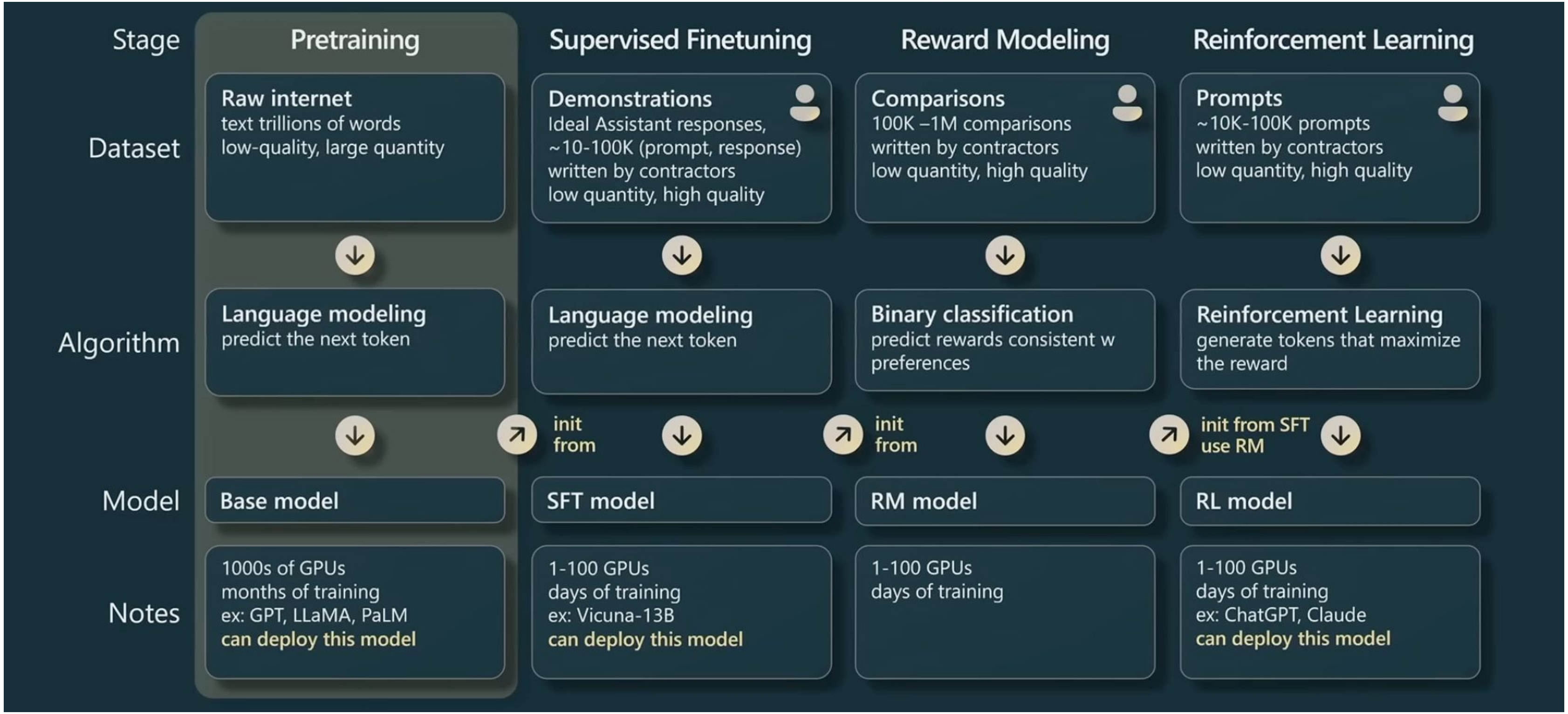
Pre-training objectives

- MLM & NSP
- Denoise
- Next token prediction (auto-regressively)





Today’s Pre-training in LLM training pipeline



ref: https://www.youtube.com/watch?v=bZQun8Y4L2A, 2023

Today’s Pre-training formatting

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

- B is the batch size (e.g. 4 here)
- T is the maximum context length (e.g. 10 here)

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

Examples:

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

The power of Today's PLM (LLM)

models can be prompted into completing tasks

Context (passage and previous question/answer pairs)

Tom goes everywhere with Catherine Green, a 54-year-old secretary. He moves around her office at work and goes shopping with her. "Most people don't seem to mind Tom," says Catherine, who thinks he is wonderful. "He's my fourth child," she says. She may think of him and treat him that way as her son. He moves around buying his food, paying his health bills and his taxes, but in fact Tom is a dog.

Catherine and Tom live in Sweden, a country where everyone is expected to lead an orderly life according to rules laid down by the government, which also provides a high level of care for its people. This level of care costs money.

People in Sweden pay taxes on everything, so aren't surprised to find that owning a dog means more taxes. Some people are paying as much as 500 Swedish kronor in taxes a year for the right to keep their dog, which is spent by the government on dog hospitals and sometimes medical treatment for a dog that falls ill. However, most such treatment is expensive, so owners often decide to offer health and even life _ for their dog.

In Sweden dog owners must pay for any damage their dog does. A Swedish Kennel Club official explains what this means: if your dog runs out on the road and gets hit by a passing car, you, as the owner, have to pay for any damage done to the car, even if your dog has been killed in the accident.

Q: How old is Catherine?
A: 54

Q: where does she live?
A:

GPT-2 is "tricked" into performing a task by completing the document

LLM framework

Training:

- DeepSpeed - DeepSpeed is a deep learning optimization library that makes distributed training and inference easy, efficient, and effective.
- Megatron-DeepSpeed - DeepSpeed version of NVIDIA's Megatron-LM that adds additional support for several features such as MoE model training, Curriculum Learning, 3D Parallelism, and others.
- FairScale - FairScale is a PyTorch extension library for high performance and large scale training.
- Megatron-LM - Ongoing research training transformer models at scale.
- Colossal-AI - Making large AI models cheaper, faster, and more accessible.

Deploying LLM:

- FastChat - A distributed multi-model LLM serving system with web UI and OpenAI-compatible RESTful APIs.
- LangChain - Building applications with LLMs through composability

ref: https://github.com/Hannibal046/Awesome-LLM#llm-training-frameworks

Training Cost

LLaMA: 380 tokens/sec/GPU. (p.s. pre-trained with sequence length = 2048)

pre-train [1]

params	≈days on 2048 A100 80G GPUs	GPU hours
6.7B	1.7	82432
13B	3	135168
32.5B	11	530432
65.2B	21	1022362

Examples

T5 (2019)	GPT-3 (2020)	LLaMA (2023)	LLaMA 2 (2023)
pre-training data: C4 trained tokens: 34B vocab: 32000 context size: 512 model size: small, base, large, 3B, 11B	pre-training data: mixed with book, wiki, web trained tokens: 300B vocab: 50257 context size: 2048 model size: s,m,l,xl,...,13B,175B	pre-training data: RedPajama trained tokens: 1-1.4TB vocab: 32000 context size: 2048 model size: 7B,13B, 33B, 65B	pre-training data: truthful~ trained tokens: 2TB vocab: 32000 context size: 4096 model size: 7B,13B, 34B, 70B

ref:

- *T5: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*
- *GPT3: Language Models are Few-Shot Learners*
- *LLaMA: Open and Efficient Foundation Language Models*
- *Llama 2: Open Foundation and Fine-Tuned Chat Models*

T5 paper reading

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

Colin Raffel* CRAFFEL@GMAIL.COM

Noam Shazeer* NOAM@GOOGLE.COM

Adam Roberts* ADAROB@GOOGLE.COM

Katherine Lee* KATHERINELEE@GOOGLE.COM

Sharan Narang SHARANNARANG@GOOGLE.COM

Michael Matena MMATENA@GOOGLE.COM

Yanqi Zhou YANQIZ@GOOGLE.COM

Wei Li MWEILI@GOOGLE.COM

Peter J. Liu PETERJLIU@GOOGLE.COM

Google, Mountain View, CA 94043, USA

T5 paper reading

the motivation of T5
Every task, one format!

Input and output format:
“[Task-specific prefix]: [Input text]” -> “[output text]”

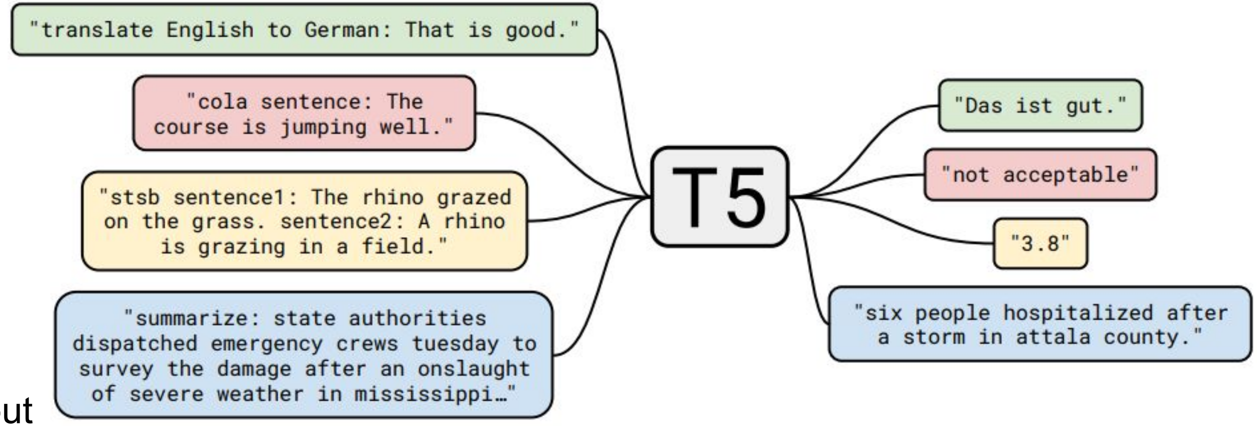
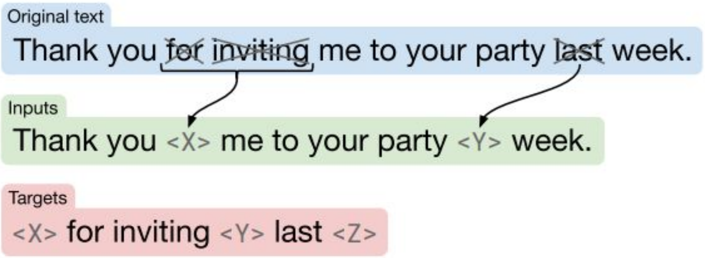


Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across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

T5 paper reading

denoising objective: the model is trained to predict missing or otherwise corrupted tokens in the input.



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

T5 paper reading

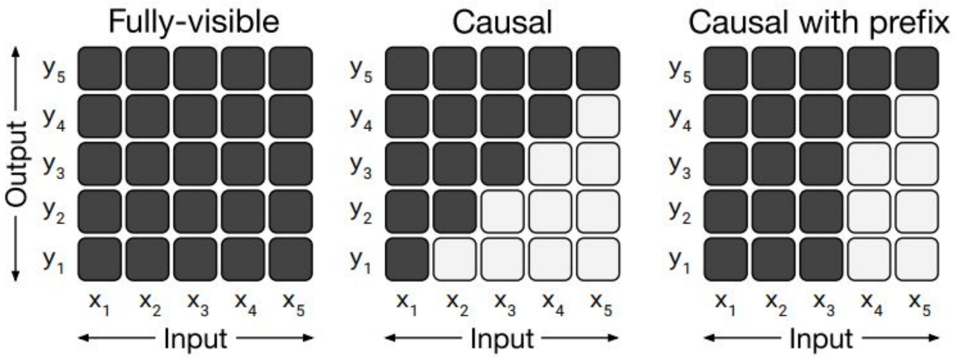


Figure 3: Matrices representing different attention mask patterns. The input and output of the self-attention mechanism are denoted x and y respectively. A dark cell at row i and column j indicates that the self-attention mechanism is allowed to attend to input element j at output timestep i. A light cell indicates that the self-attention mechanism is *not* allowed to attend to the corresponding i and j combination. Left: A fully-visible mask allows the self-attention mechanism to attend to the full input at every output timestep. Middle: A causal mask prevents the ith output element from depending on any input elements from "the future". Right: Causal masking with a prefix allows the self-attention mechanism to use fully-visible masking on a portion of the input sequence.

T5 paper reading

experiment results:

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
BERT-style (Devlin et al., 2018)	**82.96**	**19.17**	**80.65**	**69.85**	**26.78**	**40.03**	**27.41**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62

Table 4: Performance of the three disparate pre-training objectives described in Section 3.3.1.

Corruption rate	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
10%	**82.82**	19.00	**80.38**	69.55	**26.87**	39.28	**27.44**
★ 15%	**83.28**	19.24	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
25%	**83.00**	**19.54**	**80.96**	70.48	**27.04**	**39.83**	**27.47**
50%	81.27	19.32	79.80	70.33	**27.01**	**39.90**	**27.49**

Table 6: Performance of the i.i.d. corruption objective with different corruption rates.

T5 paper reading

repeating data

Number of tokens	Repeats	G
★ Full data set	0	8:
2^{29}	64	8:
2^{27}	256	8
2^{25}	1,024	7
2^{23}	4,096	7

Table 9: Measuring the effect of
we only use the first N
first column) but still p
repeated over the cou
experiment shown in th
Figure 6).

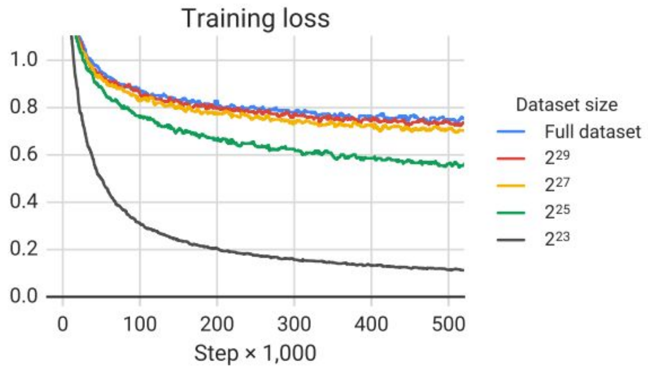


Figure 6: Pre-training loss for our original C4 data set as well as 4 artificially truncated versions. The sizes listed refer to the number of tokens in each data set. The four sizes considered correspond to repeating the data set between 64 and 4,096 times over the course of pre-training. Using a smaller data set size results in smaller training loss values, which may suggest some memorization of the unlabeled data set.

GPT3 paper reading

Language Models are Few-Shot Learners

Tom B. Brown[*] Benjamin Mann[*] Nick Ryder[*] Melanie Subbiah[*]

Jared Kaplan[†] Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

GPT3 paper reading

motivation of GPT3: task-agnostic performance
(Scaling Law, Emergent Ability)

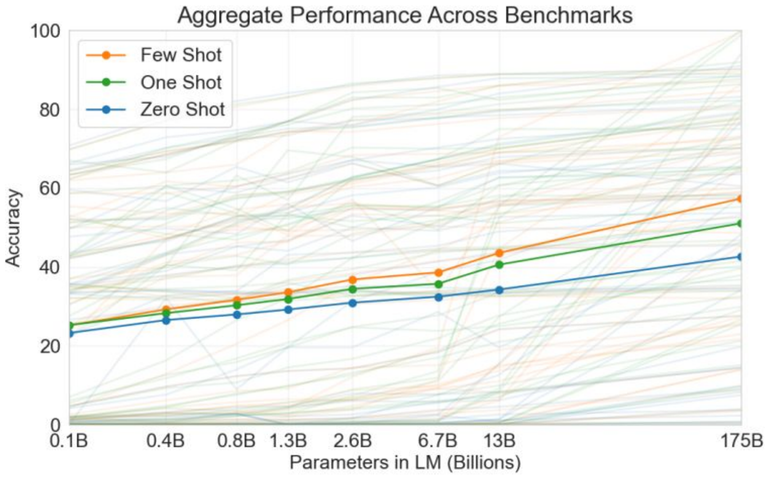
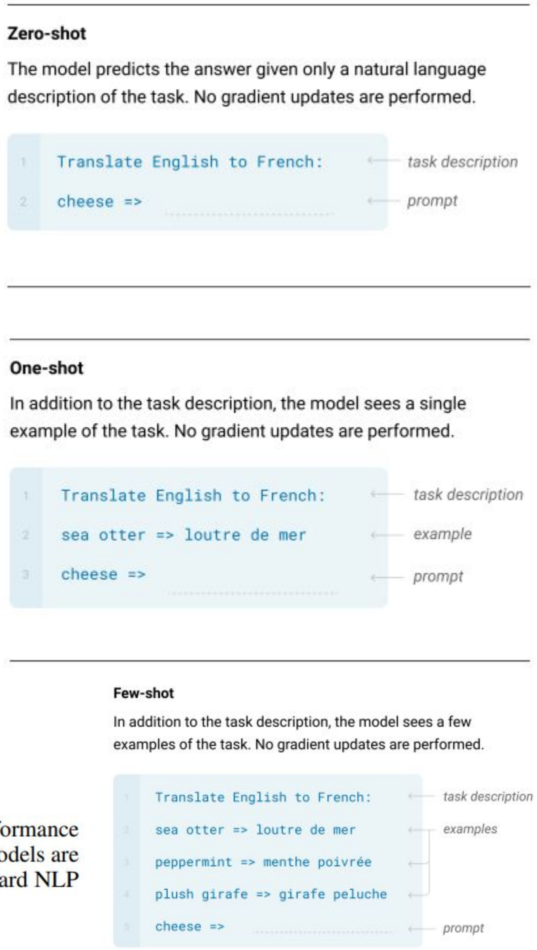


Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed analysis on SuperGLUE, a standard NLP benchmark suite.



GPT3 paper reading

pre-training details:

Dataset
Commo
WebTex
Books1
Books2
Wikiped

Table 2.2: Datasets used
that are drawn from a giv
result, when we train for 3
are seen less than once.

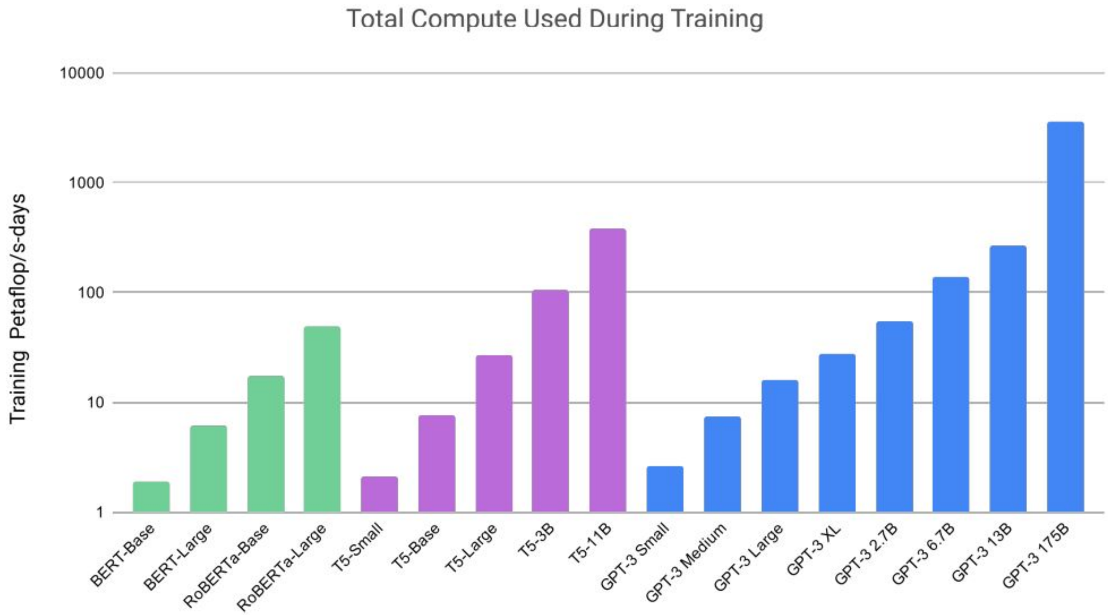
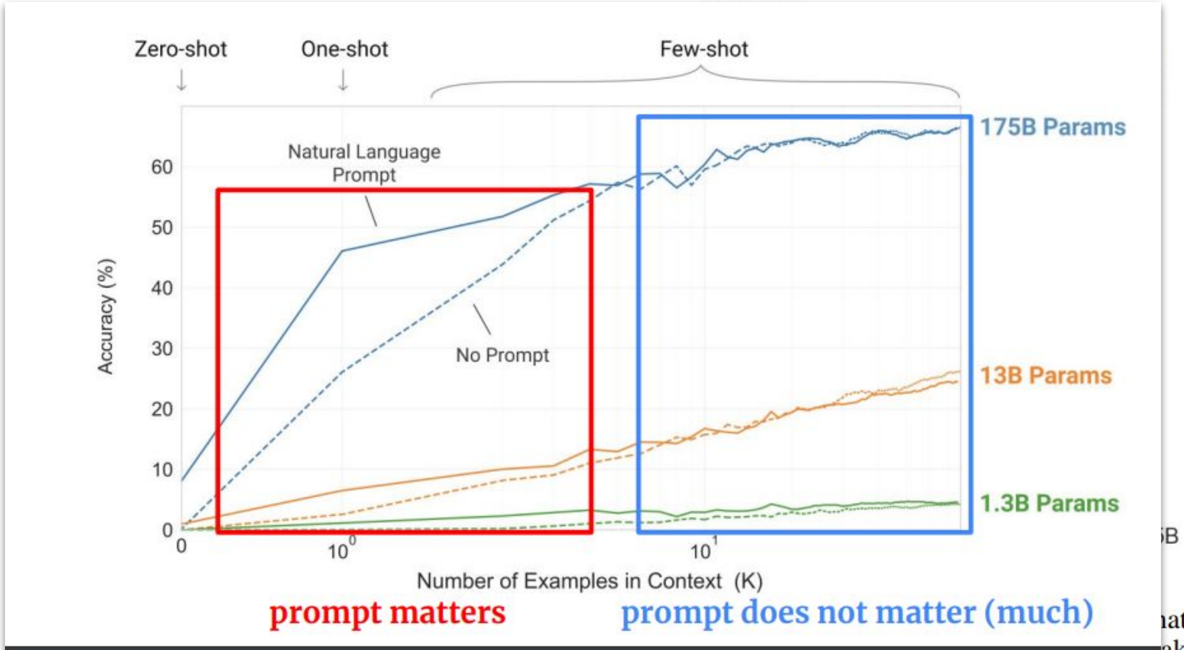


Figure 2.2: Total compute used during training. Based on the analysis in Scaling Laws For Neural Language Models [KMH+20] we train much larger models on many fewer tokens than is typical. As a consequence, although GPT-3 3B is almost 10x larger than RoBERTa-Large (355M params), both models took roughly 50 petaflop/s-days of compute during pre-training. Methodology for these calculations can be found in Appendix D.

GPT3 paper reading

experimental results



...hat language models continue to absorb knowledge as their capacity increases. One-shot and few-shot performance make significant gains over zero-shot behavior, matching and exceeding the performance of the SOTA fine-tuned open-domain model, RAG [LPP+20]

Many tasks: Language Modeling, long-range dependencies, StoryCloze, Closed Book QA, Translation, Common Sense Reasoning, Reading Comprehension, SuperGLUE, NLI, Arithmetic, Word Manipulation, Article generation

GPT3 paper reading

Limitations:

- GPT-3 samples still sometimes repeat themselves semantically at the document level, start to lose coherence over sufficiently long passages, contradict themselves

- potentially worse performance on tasks which empirically benefit from bidirectionality (in-filling)

- current objective weights every token equally and lacks a notion of what is most important to predict and what is less important

- GPT-3 is ambiguity about whether few-shot learning actually learns new tasks “from scratch” at inference time, or if it simply recognizes and identifies tasks that it has learned during training: not easily interpretable

- both expensive and inconvenient to perform inference on

LLaMA 2 paper reading

Motivations: Based on LLaMA, LLaMA2 and LLaMa-chat is focus on: helpfulness and safety

using safety-specific data annotation and tuning, as well as conducting red-teaming and employing iterative evaluations

Best open source llm so far

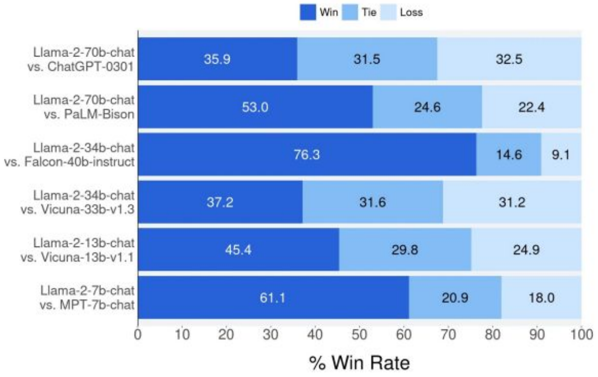


Figure 1: Helpfulness human evaluation results for Llama 2-Chat compared to other open-source and closed-source models. Human raters compared model generations on ~4k prompts consisting of both single and multi-turn prompts. The 95% confidence intervals for this evaluation are between 1% and 2%. More details in Section 3.4.2. While reviewing these results, it is important to note that human evaluations can be noisy due to limitations of the prompt set, subjectivity of the review guidelines, subjectivity of individual raters, and the inherent difficulty of comparing generations.

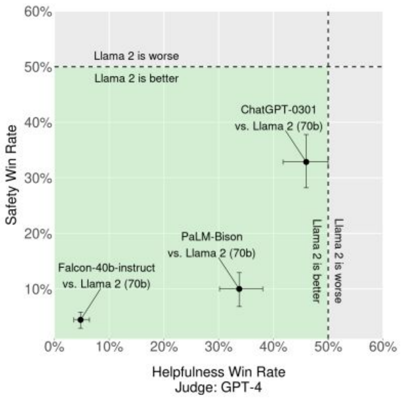
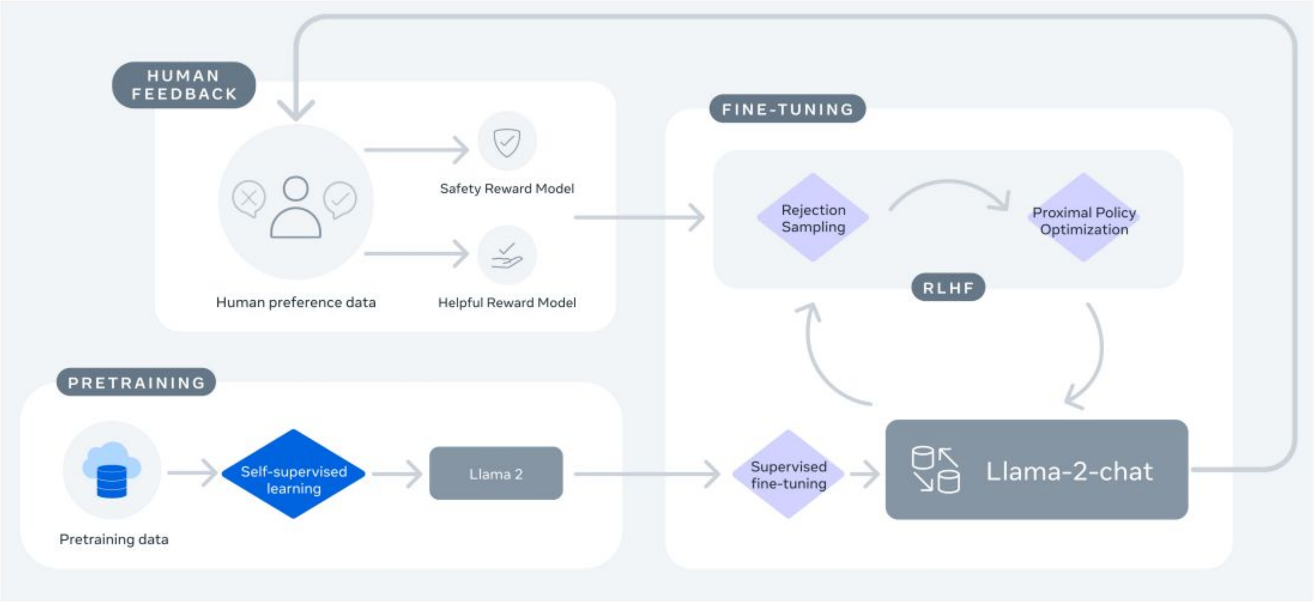


Figure 2: Win-rate % for helpfulness and safety between commercial-licensed baselines and Llama 2-Chat, according to GPT-4. To complement the human evaluation, we used a more capable model, not subject to our own guidance. Green area indicates our model is better according to GPT-4. To remove ties, we used $win/(win + loss)$. The orders in which the model responses are presented to GPT-4 are randomly swapped to alleviate bias.

LLaMA 2 paper reading

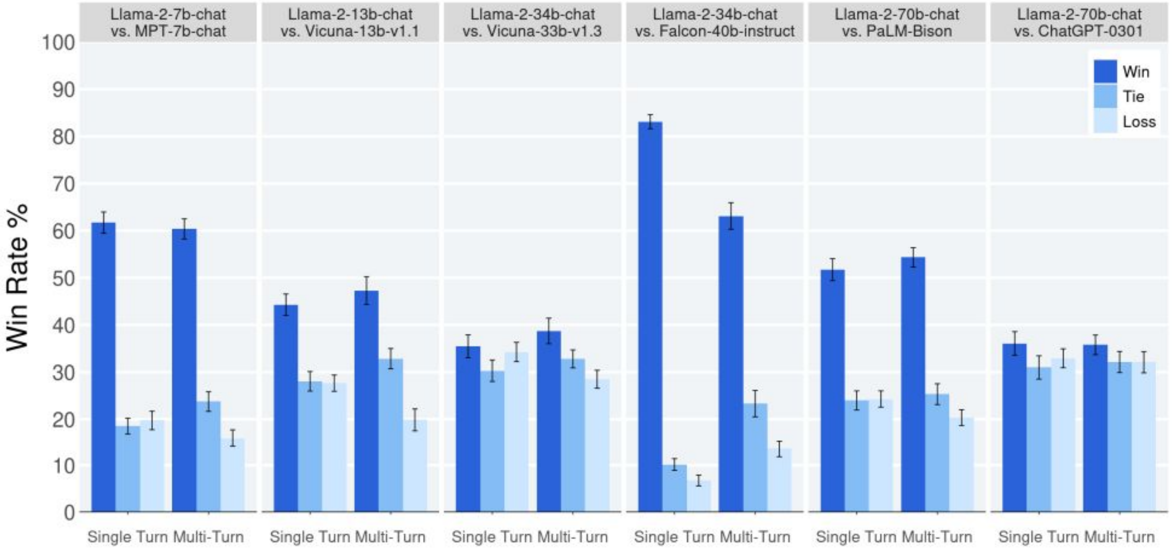
Training details



T
a
i

LLaMA 2 paper reading

Experimental Results



TruthfulQA ↑	ToxiGen ↓
29.13	22.32
35.25	22.61
25.95	**14.53**
40.39	23.44
27.42	23.00
41.74	23.08
44.19	22.57
48.71	21.77
33.29	21.25
41.86	26.10
43.45	21.19
50.18	24.60

Figure 12: Human evaluation results for Llama 2-Chat models compared to open- and closed-source models across ~4,000 helpfulness prompts with three raters per prompt.

questions

1. What is the relation between pretraining data and the ability of LLM?
2. Any other architecture of PLM?
3. How to extend the context length of PLM?