Presentation: GPT Series

Weizhi Wang

GPT - ***G***enerative ***P***re-Training ***T***ransformer: Big Picture

Model	Title	Focus	Paradigm	Params
GPT-1	Improving ***Language Understanding*** by ***Generative Pre-Training***	NLU tasks, pre-trained model	Pre-training->Efficient Fine-tuning	117M
GPT-2	Language Models are ***Unsupervised Multitask*** Learners	Zero-shot Evaluation, NLG Tasks	Pre-training->Zero-shot Multitask Transfer	1.5B
GPT-3	Language Models are ***Few-Shot*** Learners	Few-shot Learning or In-context Learning	In-context Learning with a few demonstration examples	175B
GPT-3.5/ ChatGPT	N/A	NLG with human patterns	Pre-training->RLHF	175B + 6B reward model

GPT1: ***G***enerative ***P***re-***T***raining for NLU

- GPT is out before BERT.

Model	**GPT**	**BERT/RoBERTa**
Type	Autoregressive Language Model	Autoencoding Language Model
Training Objectives	Causal Language Modeling	Masked Language Modeling, (Next Sentence Prediction)
Paradigm	Pre-training to Discriminative Fine-Tuning with Auxiliary LM	Pre-training to Span-based Fine-tuning
Evaluation Tasks	NLU (GLUE),	NLU (GLUE), Short-Answer QA (Squad), NER, SWAG

AE Encoder/AR Decoder/Prefix-LM

Pre-training models has been a hot topic in the research of NLP. Since 2018, with the emergence of BERT, it has gained great attention from both the academy and the industry. The recent published PTMs can be classified into three types: BERT variants (XLNet, RoBERT, BART, TinyBERT etc.), task-oriented PTM (PLATO), and Cross-lingual PTM (NEZHA, FILTER).

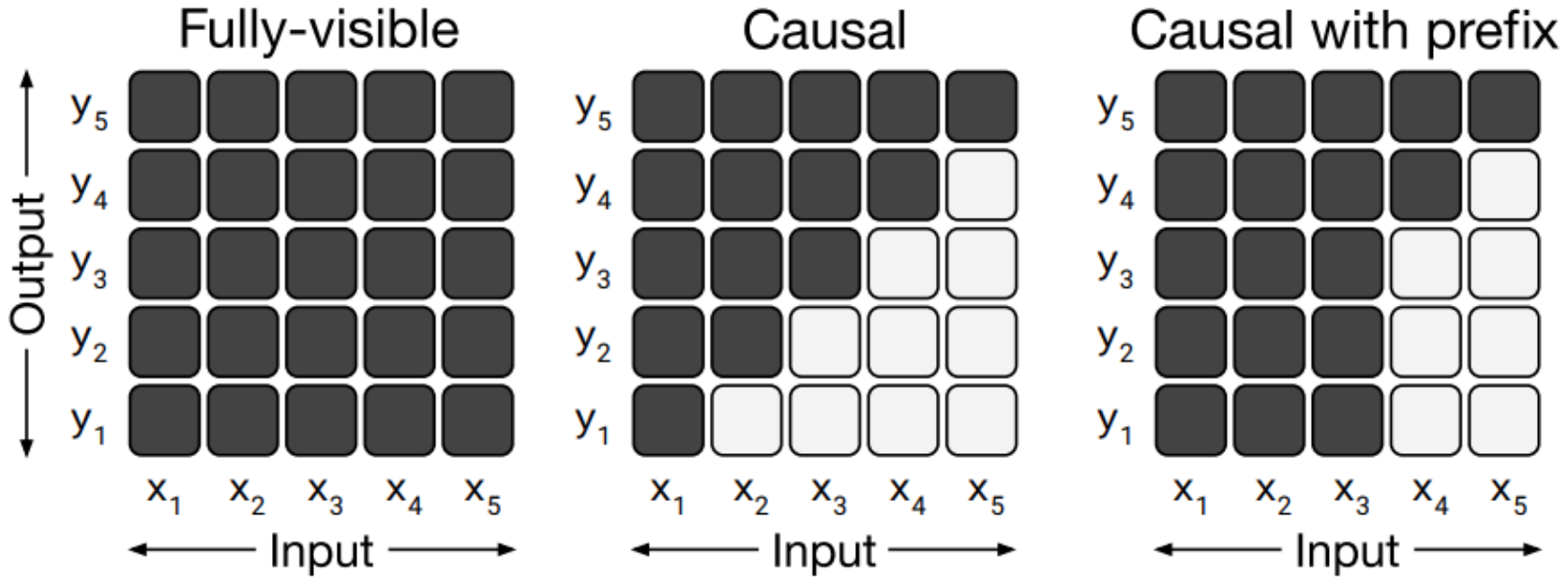


Figure 3: Matrices representing different attention mask patterns. The input and output

Unsupervised Pre-training

Training Objective: Causal Language Modeling
Maximize the likelihood on the text corpus:

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

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

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

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

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

Discriminative Fine-tuning

For labeled downstream task, maximize the log probability on each pair of instance (x, y)

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

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

This gives us the following objective to maximize:

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

Add auxiliary fine-tuning objective of language modeling will imporove the performance $L_3(C) = L_2(C) + \lambda * L_1(C)$

Discriminative Fine-tuning

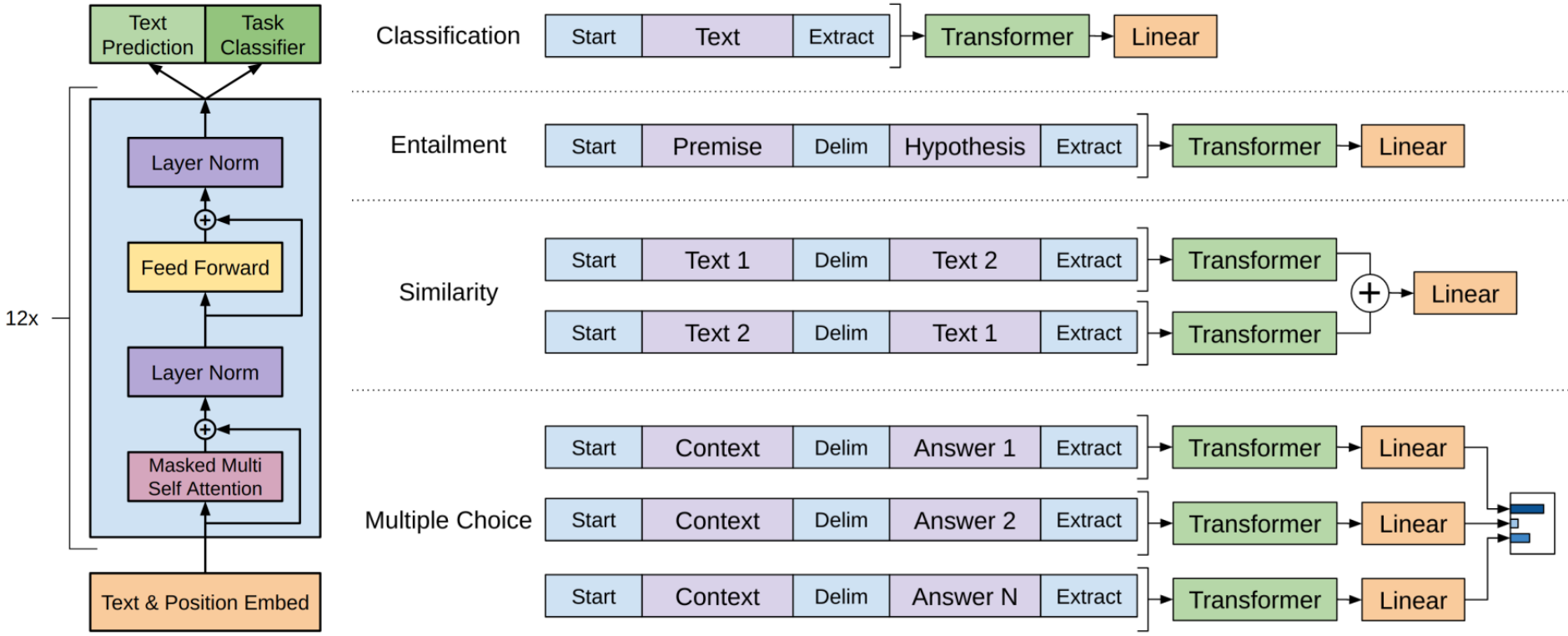


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

Results on Natural Language Understanding

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**

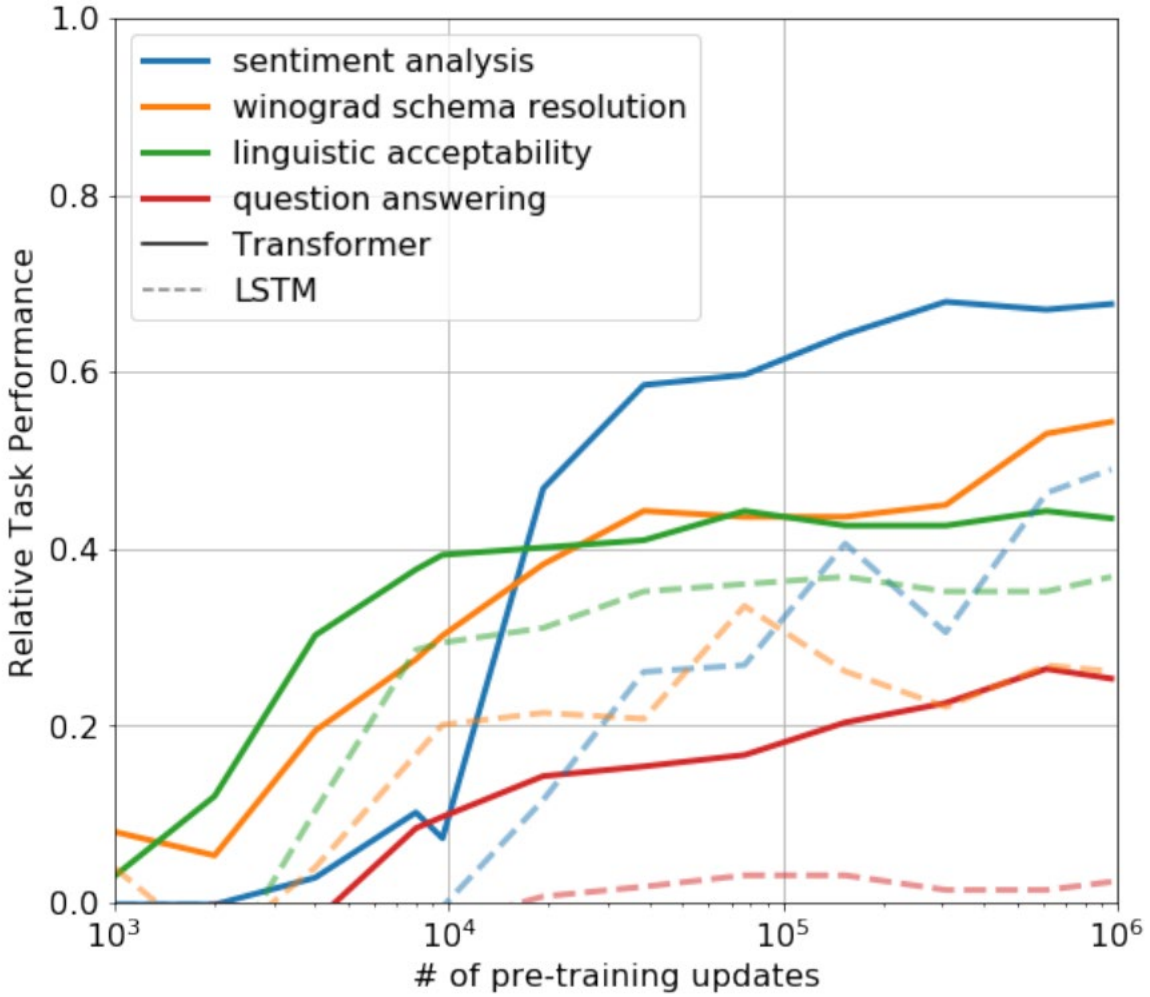
Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI 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.

Results on Natural Language Understanding

Table 4: Semantic similarity and classification results, comparing our model with current state-of-the-art methods. All task evaluations in this table were done using the GLUE benchmark. (*mc*= Mathews correlation, *acc*=Accuracy, *pc*=Pearson correlation)

Method	**Classification**		**Semantic Similarity**			**GLUE**
	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	
Sparse byte mLSTM [16]	-	**93.2**	-	-	-	-
TF-KLD [23]	-	-	**86.0**	-	-	-
ECNU (mixed ensemble) [60]	-	-	-	81.0	-	-
Single-task BiLSTM + ELMo + Attn [64]	35.0	90.2	80.2	55.5	66.1	64.8
Multi-task BiLSTM + ELMo + Attn [64]	18.9	91.6	83.5	72.8	63.3	68.9
Finetuned Transformer LM (ours)	**45.4**	91.3	82.3	**82.0**	**70.3**	**72.8**

Zero-shot Behaviors



The performance is normalized between the random guess baseline and SOTA model.

In the first time, GPT-2 proved the positive correlation between pre-training steps and zero-shot performance.

Therefore, **do not stop pre-training! And GPT-2 is on its way.**

GPT-2:

Transferring from NLU to NLG, which is more complicated.

Fully zero-shot evaluation, without any task-specific fine-tuning.

Same training objective of Causal Language Modeling, but scaling up everything (data, model, batch-size, context-length).

Achieved SOTA on most of NLG dataset compared with tuned model.

GPT-2: Language Modeling Benchamarks

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang et al., 2018) and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

GPT-2: RC, Translation, SUM, QA

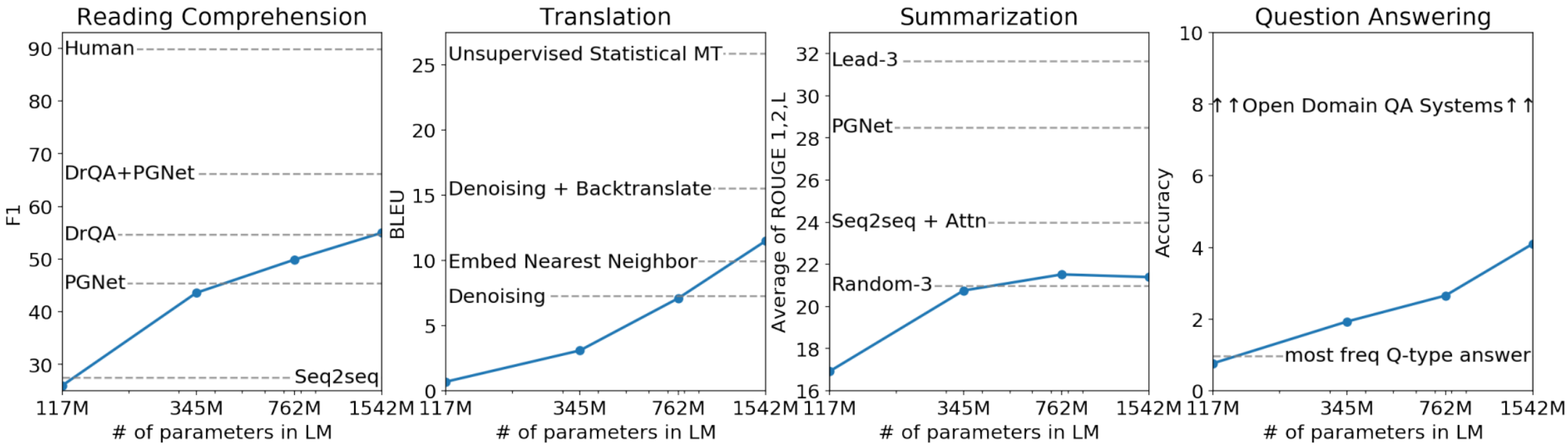


Figure 1. Zero-shot task performance of WebText LMs as a function of model size on many NLP tasks. Reading Comprehension results are on CoQA (Reddy et al., 2018), translation on WMT-14 Fr-En (Artetxe et al., 2017), summarization on CNN and Daily Mail (See et al., 2017), and Question Answering on Natural Questions (Kwiatkowski et al., 2019). Section 3 contains detailed descriptions of each result.

Large-Scale Data and Under-fitting

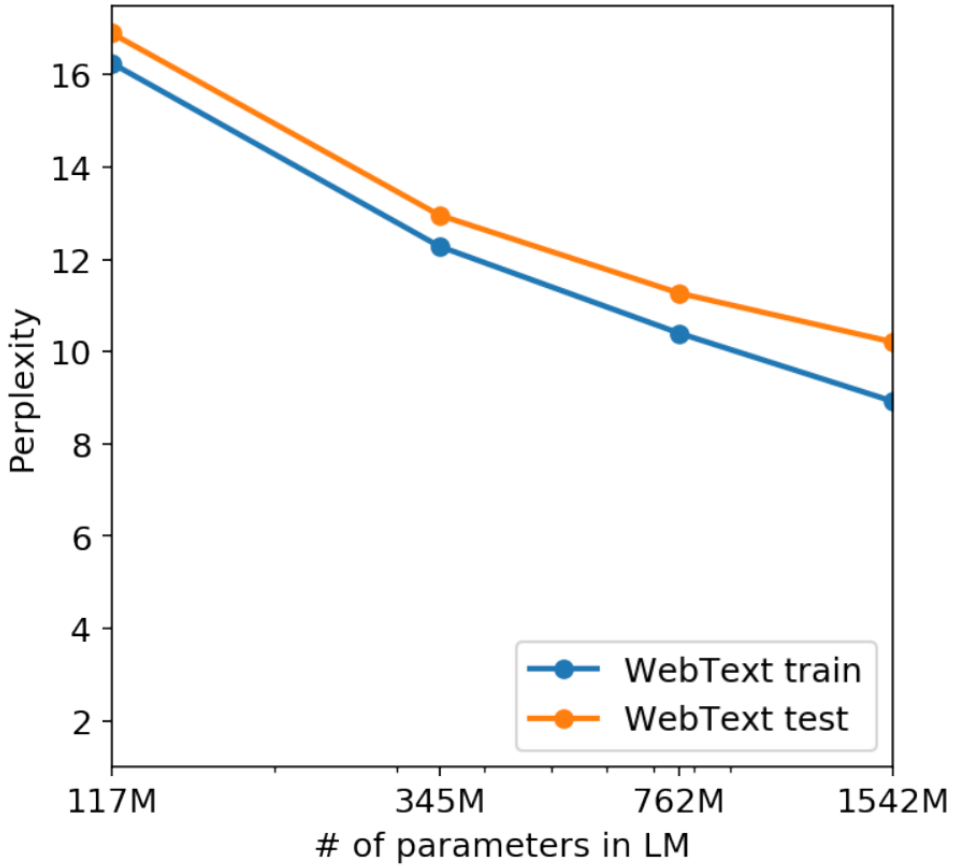


Figure 4. The performance of LMs trained on WebText as a function of model size.

Even with the increase of model parameters to 1.5B, the training dataset of WebText 1542M is still under fitting.

Therefore, the model ***can still be scaled up*** to better fit on the training dataset.

GPT-3 is on the way! A new era started!

GPT-3: What is in-context learning?

Three ways of in-context learning:

Zero-shot

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

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

One-shot

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

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

In a single sequence input, the prompted example can learn from previous demonstrations.

Few-shot

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

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

GPT-3: What is LM capable of in-context learning?

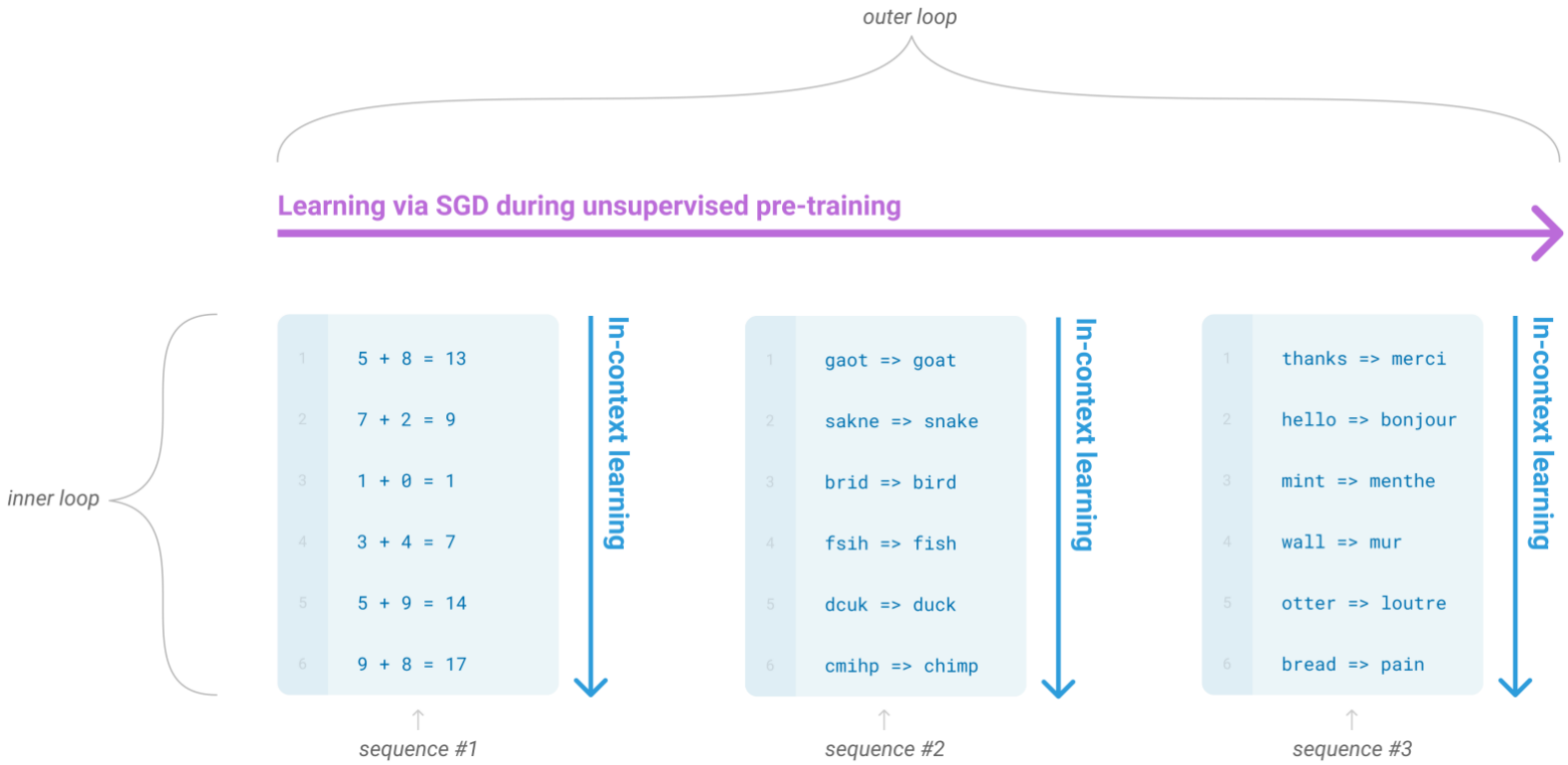


Figure 1.1: Language model meta-learning. During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. We use the term "in-context learning" to describe the inner loop of this process, which occurs within the forward-pass upon each sequence. The sequences in this diagram are not intended to be representative of the data a model would see during pre-training, but are intended to show that there are sometimes repeated sub-tasks embedded within a single sequence.

GPT-3 Results: NLU of SuperGLUE

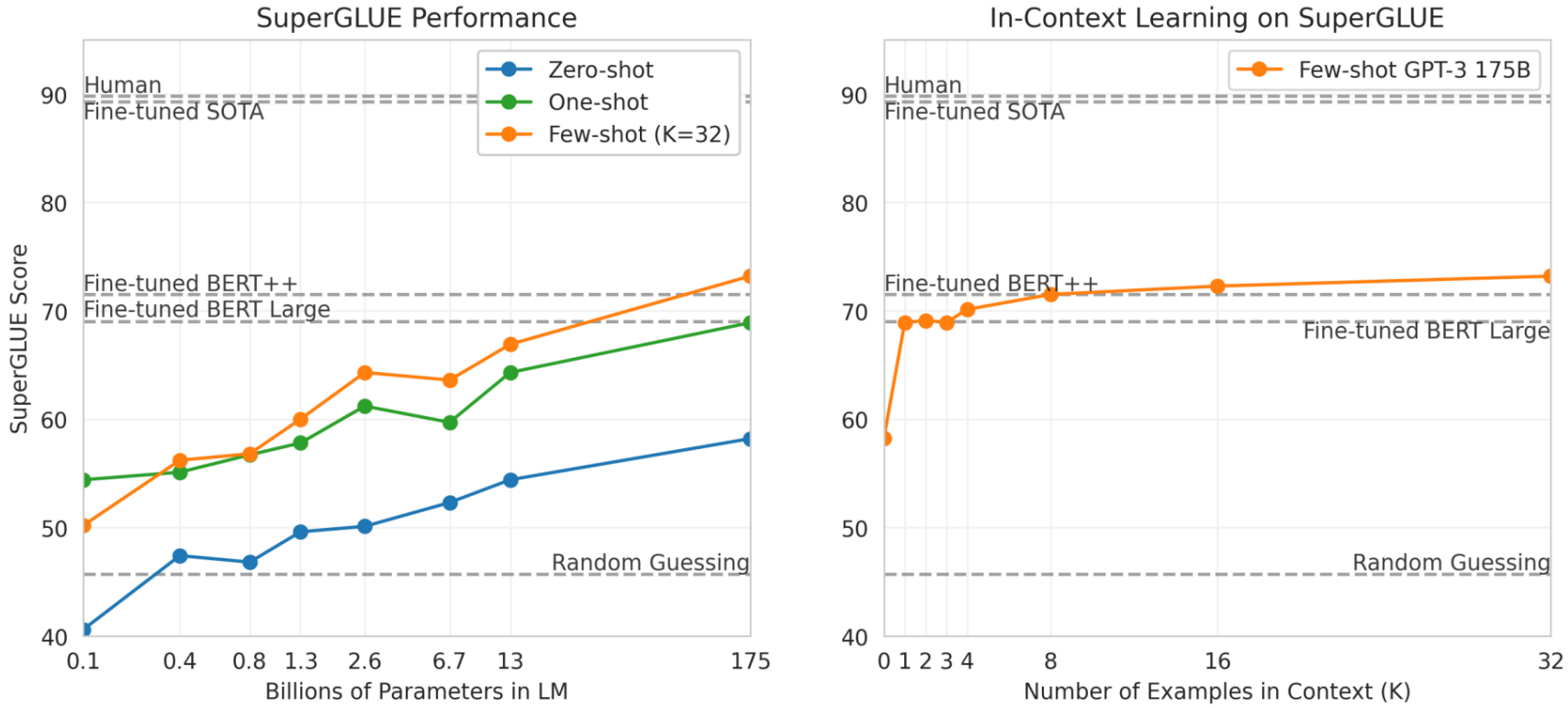


Figure 3.8: Performance on SuperGLUE increases with model size and number of examples in context. A value of $K = 32$ means that our model was shown 32 examples per task, for 256 examples total divided across the 8 tasks in SuperGLUE. We report GPT-3 values on the dev set, so our numbers are not directly comparable to the dotted reference lines (our test set results are in Table 3.8). The BERT-Large reference model was fine-tuned on the SuperGLUE training set (125K examples), whereas BERT++ was first fine-tuned on MultiNLI (392K examples) and SWAG (113K examples) before further fine-tuning on the SuperGLUE training set (for a total of 630K fine-tuning examples). We find the difference in performance between the BERT-Large and BERT++ to be roughly equivalent to the difference between GPT-3 with one example per context versus eight examples per context.

GPT-3 Results: Language Modeling

Setting	LAMBADA (acc)	LAMBADA (ppl)	StoryCloze (acc)	HellaSwag (acc)
SOTA	68.0[a]	8.63[b]	**91.8**[c]	**85.6**[d]
GPT-3 Zero-Shot	**76.2**	**3.00**	83.2	78.9
GPT-3 One-Shot	**72.5**	**3.35**	84.7	78.1
GPT-3 Few-Shot	**86.4**	**1.92**	87.7	79.3

Table 3.2: Performance on cloze and completion tasks. GPT-3 significantly improves SOTA on LAMBADA while achieving respectable performance on two difficult completion prediction datasets. [a][Tur20] [b][RWC+19] [c][LDL19] [d][LCH+20]

GPT-3 Results: Open-Domain QA

Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP+20]	**44.5**	**45.5**	**68.0**
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
GPT-3 Zero-Shot	14.6	14.4	64.3
GPT-3 One-Shot	23.0	25.3	**68.0**
GPT-3 Few-Shot	29.9	41.5	**71.2**

Table 3.3: Results on three Open-Domain QA tasks. GPT-3 is shown in the few-, one-, and zero-shot settings, as compared to prior SOTA results for closed book and open domain settings. TriviaQA few-shot result is evaluated on the wiki split test server.

GPT-3 Results: Machine Translation

Setting	En→Fr	Fr→En	En→De	De→En	En→Ro	Ro→En
SOTA (Supervised)	**45.6**[a]	35.0 [b]	**41.2**[c]	40.2[d]	**38.5**[e]	**39.9**[e]
XLM [LC19]	33.4	33.3	26.4	34.3	33.3	31.8
MASS [STQ+19]	37.5	34.9	28.3	35.2	35.2	33.1
mBART [LGG+20]	-	-	29.8	34.0	35.0	30.5
GPT-3 Zero-Shot	25.2	21.2	24.6	27.2	14.1	19.9
GPT-3 One-Shot	28.3	33.7	26.2	30.4	20.6	38.6
GPT-3 Few-Shot	32.6	39.2	29.7	40.6	21.0	39.5

Table 3.4: Few-shot GPT-3 outperforms previous unsupervised NMT work by 5 BLEU when translating into English reflecting its strength as an English LM. We report BLEU scores on the WMT'14 Fr↔En, WMT'16 De↔En, and WMT'16 Ro↔En datasets as measured by `multi-bleu.perl` with XLM's tokenization in order to compare most closely with prior unsupervised NMT work. SacreBLEU[f] [Pos18] results reported in Appendix H. Underline indicates an unsupervised or few-shot SOTA, bold indicates supervised SOTA with relative confidence. [a][EOAG18] [b][DHKH14] [c][WXH+18] [d][oR16] [e][LGG+20] [f][SacreBLEU signature: BLEU+case.mixed+numrefs.1+smooth.exp+tok.intl+version.1.2.20]

GPT-3 Results: Reading Comprehension

Setting	CoQA	DROP	QuAC	SQuADv2	RACE-h	RACE-m
Fine-tuned SOTA	**90.7**[a]	**89.1**[b]	**74.4**[c]	**93.0**[d]	**90.0**[e]	**93.1**[e]
GPT-3 Zero-Shot	81.5	23.6	41.5	59.5	45.5	58.4
GPT-3 One-Shot	84.0	34.3	43.3	65.4	45.9	57.4
GPT-3 Few-Shot	85.0	36.5	44.3	69.8	46.8	58.1

Table 3.7: Results on reading comprehension tasks. All scores are F1 except results for RACE which report accuracy.
[a][JZC+19] [b][JN20] [c][AI19] [d][QIA20] [e][SPP+19]

GPT-3 Results: Arithmetic

Setting	2D+	2D-	3D+	3D-	4D+	4D-	5D+	5D-	2Dx	1DC
GPT-3 Zero-shot	76.9	58.0	34.2	48.3	4.0	7.5	0.7	0.8	19.8	9.8
GPT-3 One-shot	99.6	86.4	65.5	78.7	14.0	14.0	3.5	3.8	27.4	14.3
GPT-3 Few-shot	100.0	98.9	80.4	94.2	25.5	26.8	9.3	9.9	29.2	21.3

Table 3.9: Results on basic arithmetic tasks for GPT-3 175B. {2,3,4,5}D{+,-} is 2, 3, 4, and 5 digit addition or subtraction, 2Dx is 2 digit multiplication. 1DC is 1 digit composite operations. Results become progressively stronger moving from the zero-shot to one-shot to few-shot setting, but even the zero-shot shows significant arithmetic abilities.

GPT-3 Results: Turing Test

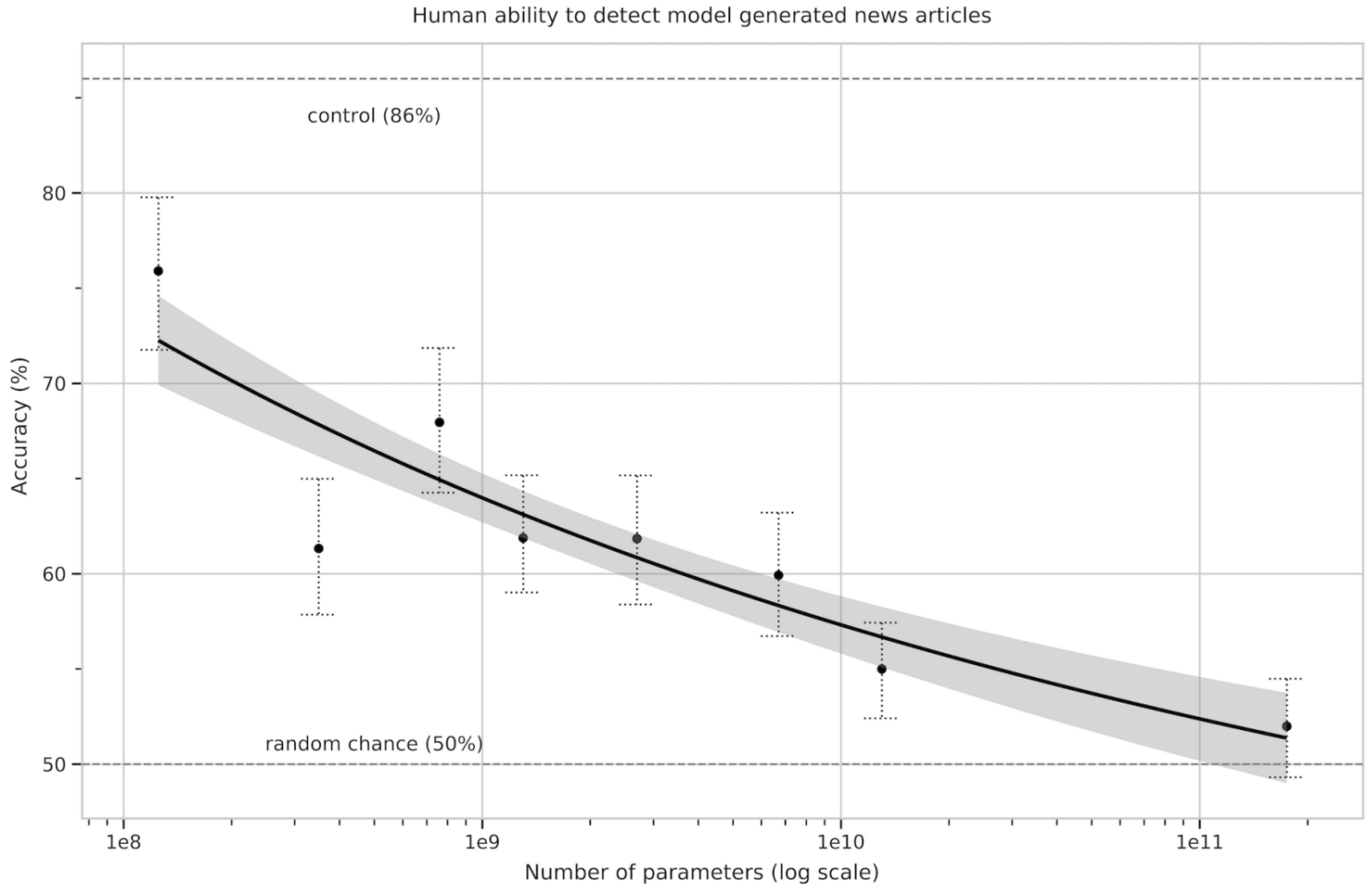


Figure 3.13: People's ability to identify whether news articles are model-generated (measured by the ratio of correct assignments to non-neutral assignments) decreases as model size increases. Accuracy on the outputs on the deliberately-bad control model (an unconditioned GPT-3 Small model with higher output randomness) is indicated with the dashed line at the top, and the random chance (50%) is indicated with the dashed line at the bottom. Line of best fit is a power law with 95% confidence intervals.

Key to Success: Data Resources

Model	Pre-training Data	Size
GPT-1	BooksCorpus (7000 books)	5GB
BERT	BooksCorpus, En-Wikipedia	16GB
GPT-2	WebText	40GB
RoBERTa	BooksCorpus, CC-News, OpenWebText(WebText), Stories	160GB
GPT-3	CC(Common Crawl), WebText2, Books1, Books2, Wikipedia	~700GB
GPT-J	Pile Corpus	800GB

Key to Success: Data Resources

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

Table 2.2: Datasets used to train GPT-3. "Weight in training mix" refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.

Key to Success: Scaling Up

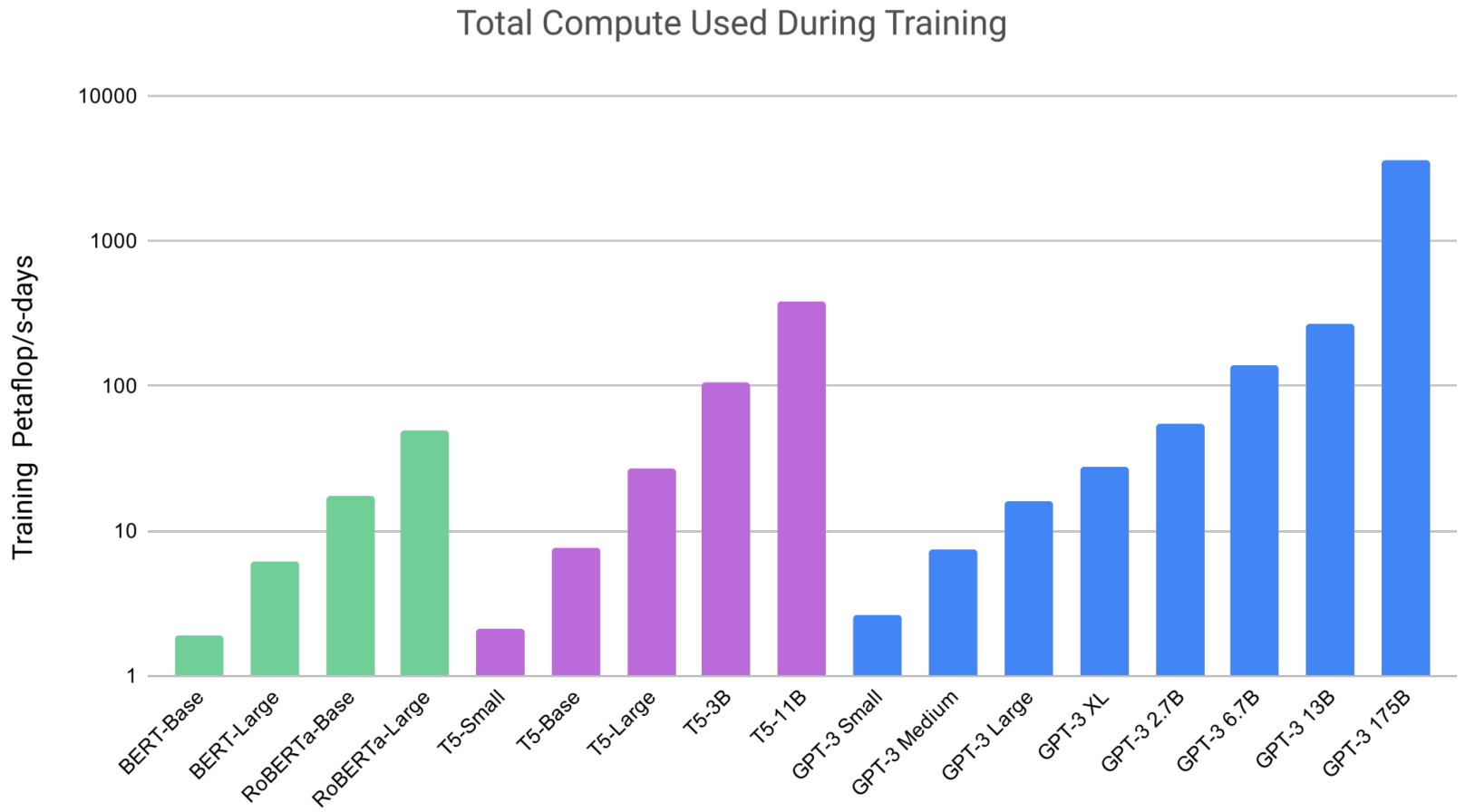


Figure 2.2: Total compute used during training. Based on the analysis in Scaling Laws For Neural Language Models [KMH+20] we train much larger models on many fewer tokens than is typical. As a consequence, although GPT-3 3B is almost 10x larger than RoBERTa-Large (355M params), both models took roughly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

Key to Success: Scaling Up

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.

Key to Success

- Conclude, Summarize, and Find emerging phenomena from systematical experiments:
  - in GPT-1, the experiment of the relation between #updates and zero-shot performance;
  - in GPT-2, the experiment of the relation between #params and training set ppl

- Insist on Simple yet Effective Architecture

- Keep on collecting high-quality web-crawled data

InstructGPT: Training language models to follow instructions with human feedback

Step 1: Collect demonstration data, and train a supervised policy. Our labelers provide demonstrations of the desired behavior on the input prompt distribution (see Section 3.2 for details on this distribution). We then fine-tune a pretrained GPT-3 model on this data using supervised learning.

Step 2: Collect comparison data, and train a reward model. We collect a dataset of comparisons between model outputs, where labelers indicate which output they prefer for a given input. We then train a reward model to predict the human-preferred output.

Step 3: Optimize a policy against the reward model using PPO. We use the output of the RM as a scalar reward. We fine-tune the supervised policy to optimize this reward using the PPO algorithm (Schulman et al., 2017).

InstructGPT

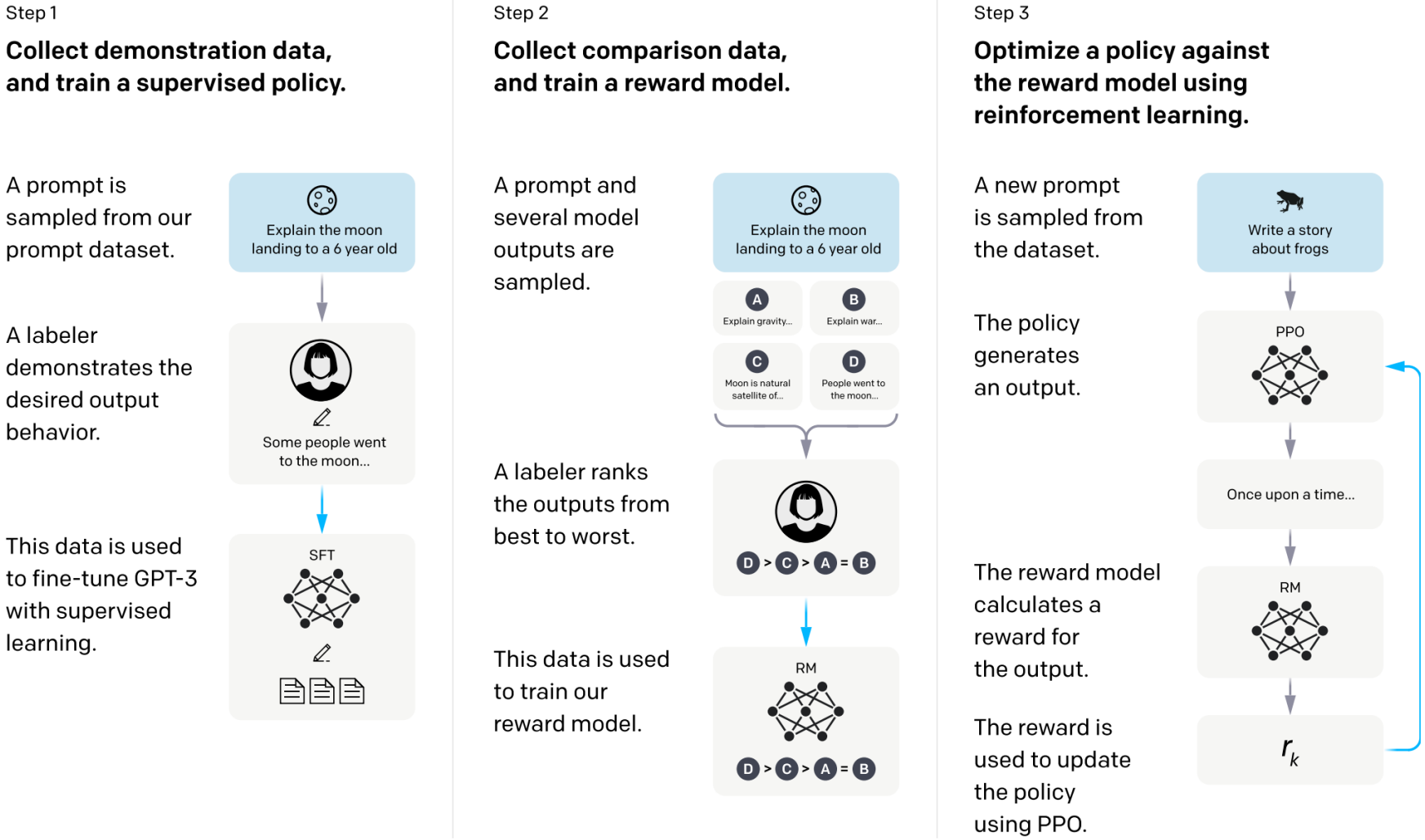


Figure 2: A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model. Blue arrows indicate that this data is used to train one of our models. In Step 2, boxes A-D are samples from our models that get ranked by labelers. See Section 3 for more details on our method.

InstructGPT: Reward Model

Specifically, the loss function for the reward model is:

$$\text{loss}\left(\theta\right) = -\frac{1}{\binom{K}{2}} E_{\left(x, y_w, y_l\right) \sim D}\left[\log\left(\sigma\left(r_\theta\left(x, y_w\right) - r_\theta\left(x, y_l\right)\right)\right)\right] \tag{1}$$

where $r_\theta(x, y)$ is the scalar output of the reward model for prompt x and completion y with parameters θ, y_w is the preferred completion out of the pair of y_w and y_l, and D is the dataset of human comparisons.

InstructGPT: PPO

We also experiment with mixing the pretraining gradients into the PPO gradients, in order to fix the performance regressions on public NLP datasets. We call these models "PPO-ptx." We maximize the following combined objective function in RL training:

$$\begin{aligned}\text{objective}\,(\phi) =& E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\mathrm{RL}}(y \mid x)/\pi^{\mathrm{SFT}}(y \mid x)\right)\right] + \\ & \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\mathrm{RL}}(x))\right]\end{aligned} \tag{2}$$

where π_ϕ^{RL} is the learned RL policy, π^{SFT} is the supervised trained model, and D_{pretrain} is the pretraining distribution. The KL reward coefficient, β, and the pretraining loss coefficient, γ, control the strength of the KL penalty and pretraining gradients respectively. For "PPO" models, γ is set to 0. Unless otherwise specified, in this paper InstructGPT refers to the PPO-ptx models.

Results

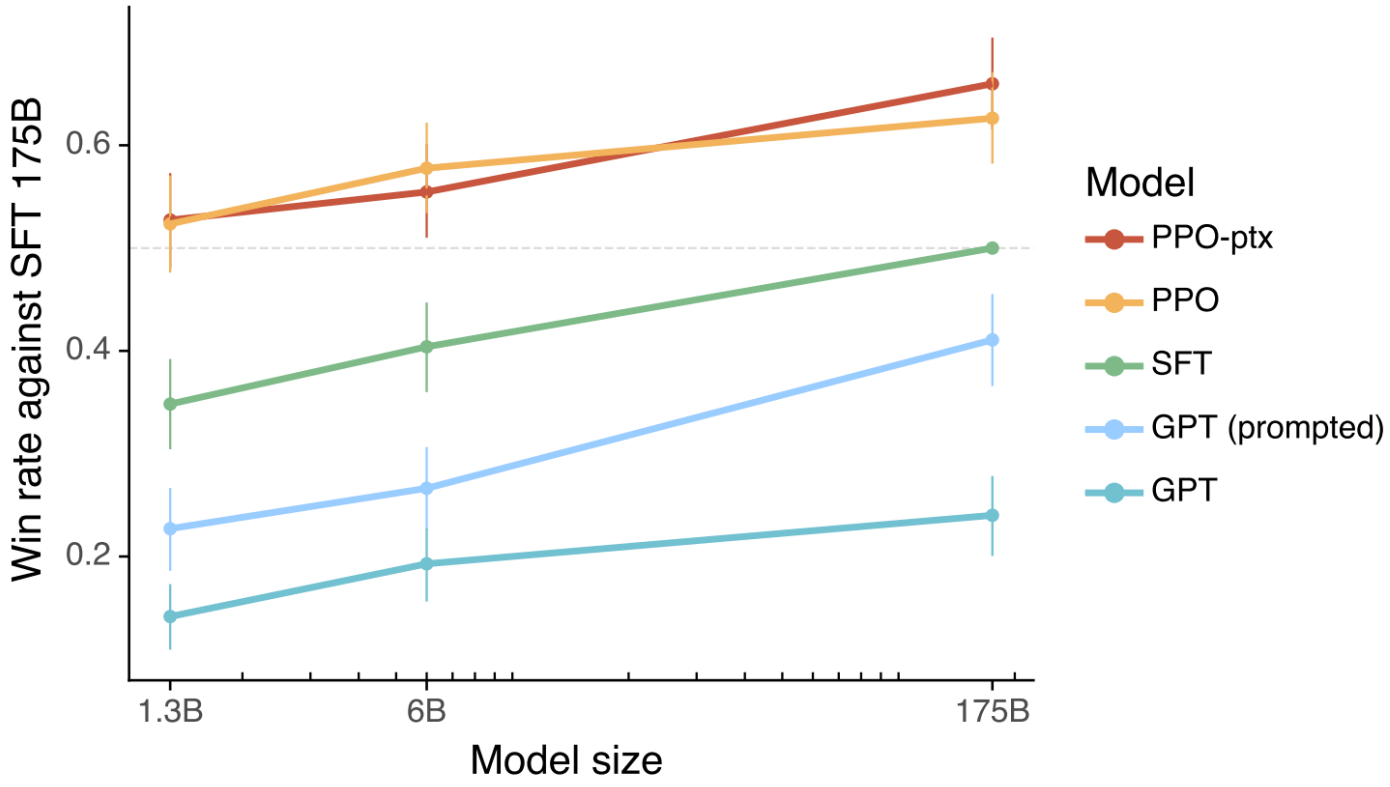


Figure 1: Human evaluations of various models on our API prompt distribution, evaluated by how often outputs from each model were preferred to those from the 175B SFT model. Our InstructGPT models (PPO-ptx) as well as its variant trained without pretraining mix (PPO) significantly outperform the GPT-3 baselines (GPT, GPT prompted); outputs from our 1.3B PPO-ptx model are preferred to those from the 175B GPT-3. Error bars throughout the paper are 95% confidence intervals.

Results

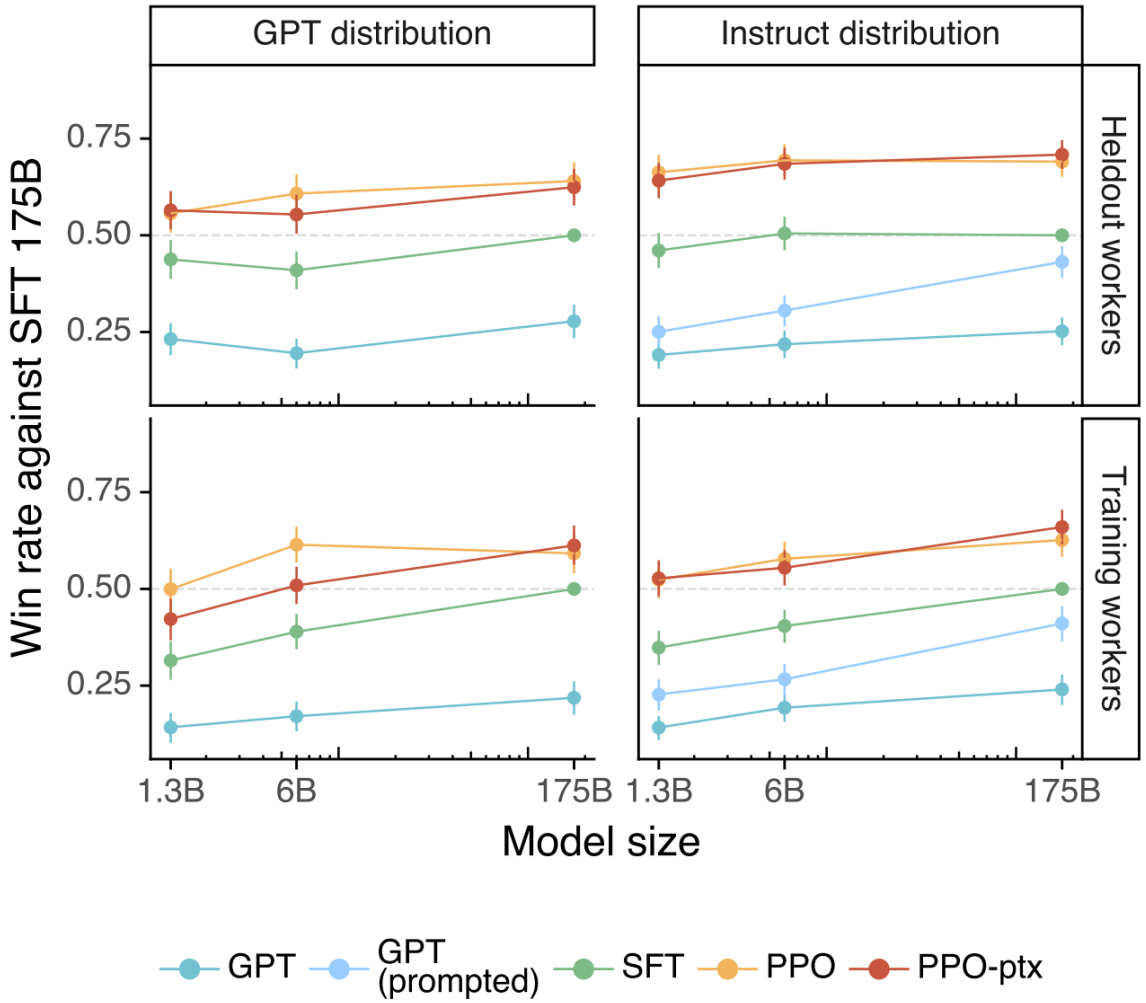
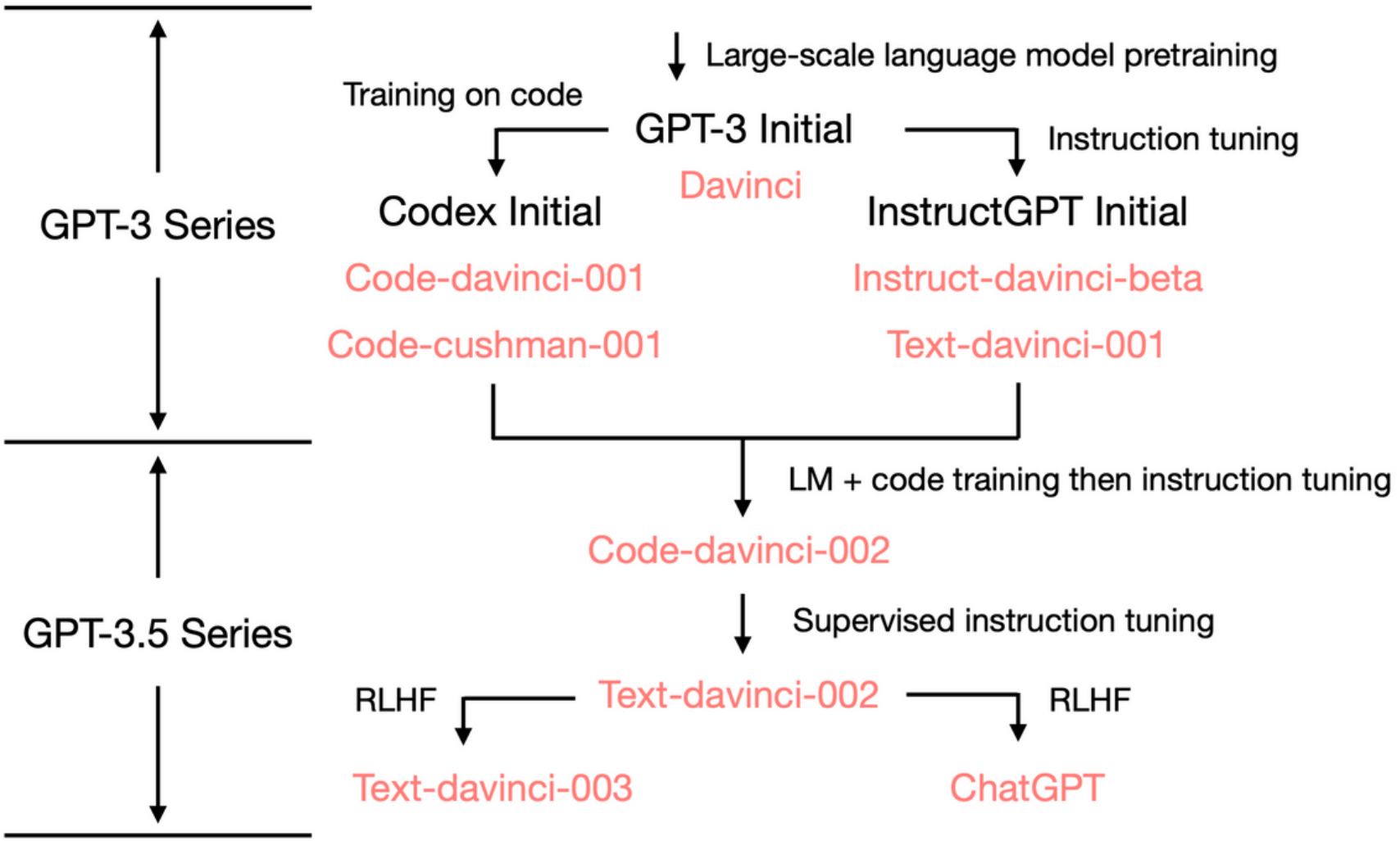


Figure 3: Preference results of our models, measured by winrate against the 175B SFT model. Left: results on prompts submitted to GPT models on the API; Right: results on prompts submitted to InstructGPT models on the API; Top: results from held-out labelers; Bottom: results from training labelers. We omit GPT (prompted) from the evals on prompts submitted to GPT-3 models (left) as these prompts are already designed to perform well for GPT-3, as opposed to prompts submitted to InstructGPT models (right).

Usage


GPT-3 Series
GPT-3.5 Series
Large-scale language model pretraining
Training on code
GPT-3 Initial
Davinci
Instruction tuning
Codex Initial
Code-davinci-001
Code-cushman-001
InstructGPT Initial
Instruct-davinci-beta
Text-davinci-001
LM + code training then instruction tuning
Code-davinci-002
Supervised instruction tuning
RLHF
Text-davinci-002
RLHF
Text-davinci-003
ChatGPT