Large Language Models

Part 2: (Instruction) fine-tuning and prompting

David Samuel

Language Technology Group, University of Oslo



Table of contents

1. Pre-training / fine-tuning paradigm
2. Downstream fine-tuning
3. Prompt-based learning
4. Instruction fine-tuning
5. References

Pre-training / fine-tuning paradigm

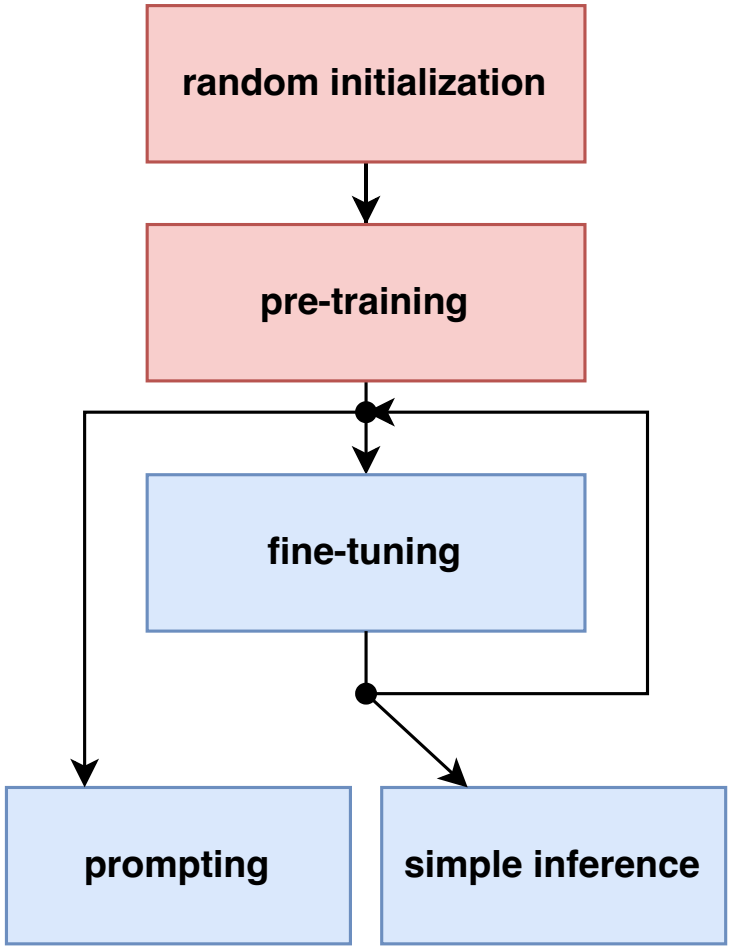


Table of contents

1. Pre-training / fine-tuning paradigm

2. Downstream fine-tuning

3. Prompt-based learning

4. Instruction fine-tuning

5. References

Downstream fine-tuning

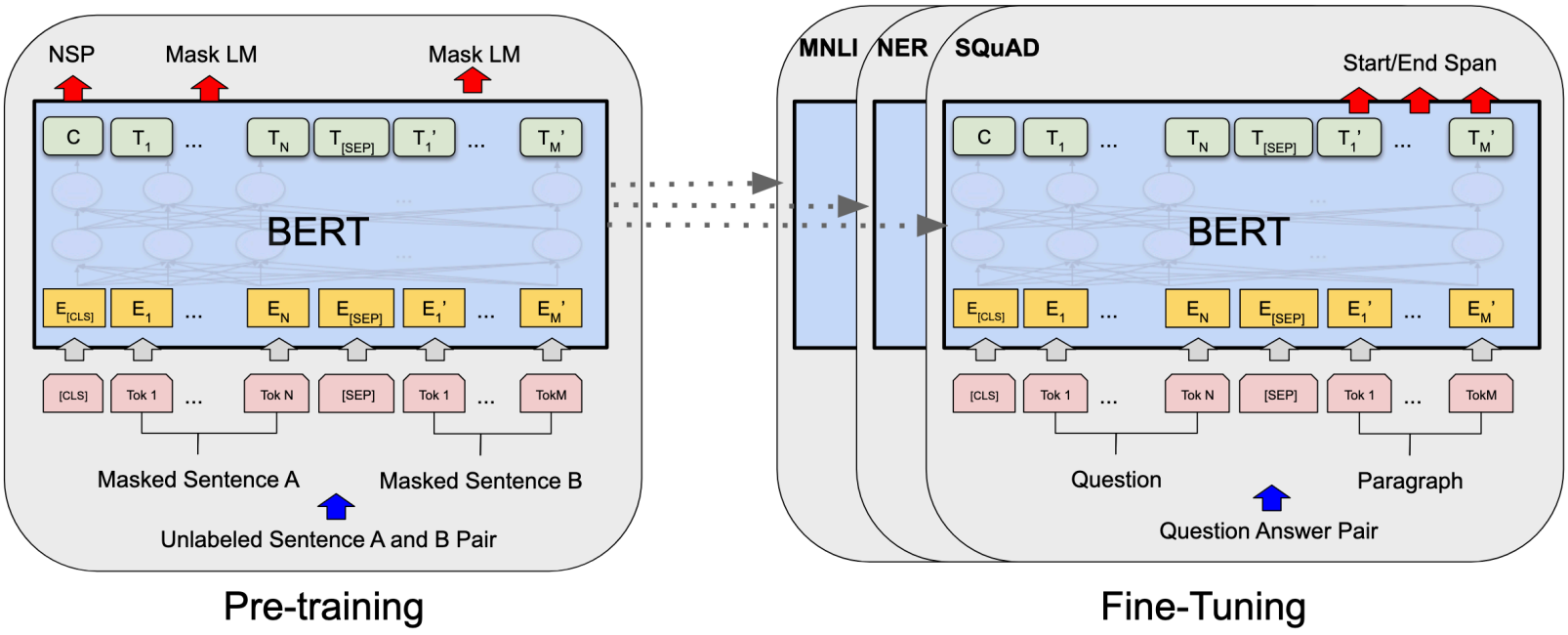


Fine-tuning

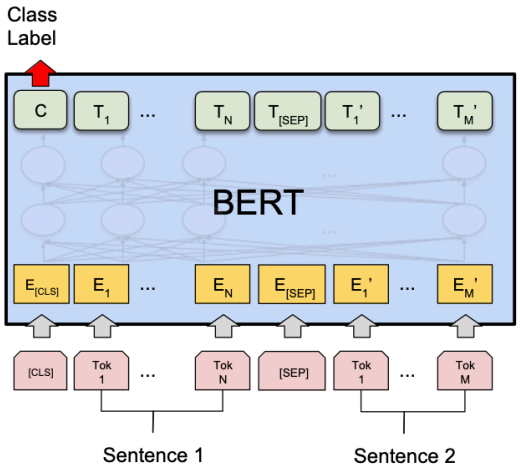
- Usually transfer learning from self-supervised task to supervised task
  - but in general "fine-tuning" refers to any kind of training that adapts/specializes a pre-trained model
- First, we train a general language model and then specialize it on a downstream task
- The language model can be either:
  - frozen: we don't change its weights, we only use its contextualized embeddings and train a classifier on top
  - further trained: we train it together with the classifier (if any)

Downstream fine-tuning

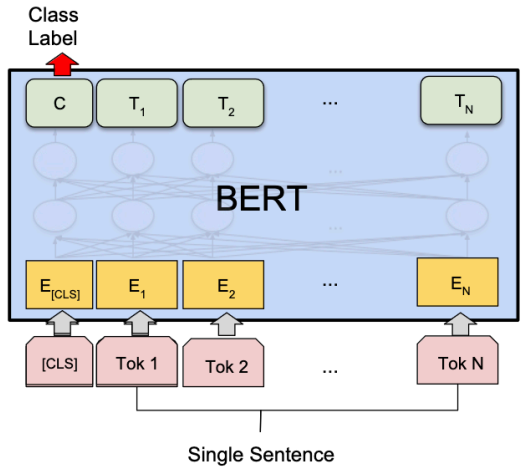
Fine-tuning of BERT



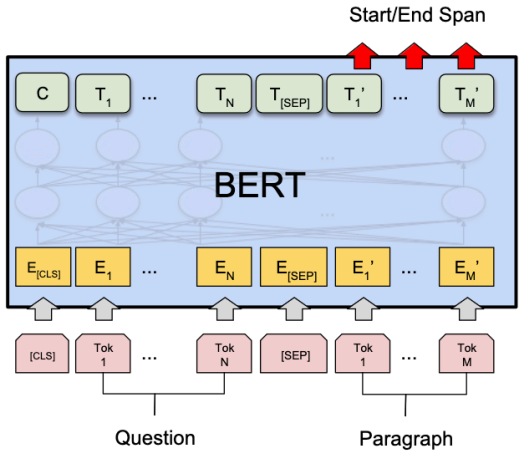
Fine-tuning of BERT



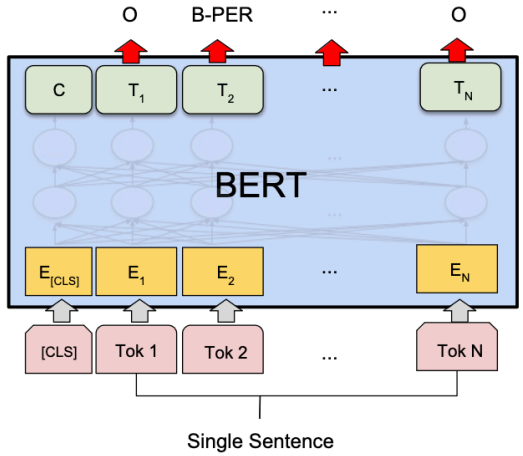
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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



(c) Question Answering Tasks: SQuAD v1.1



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



Fine-tuning of BERT, practicalities

- Almost always, the language models are overparametrized for the supervised data
- We have to be very careful with overfitting!
- We should also be *gentle*, BERT is used to do MLM, not our supervised task



Fine-tuning of BERT, practicalities

- Almost always, the language models are overparametrized for the supervised data
- We have to be very careful with overfitting!
- We should also be *gentle*, BERT is used to do MLM, not our supervised task

- Regularization is important
  - large weight decay (0.1 by default), early stopping (usually it is enough to train for just 2-8 epochs) and dropout
- Learning rate schedule is also crucial
  - The peak LR should be small (around 3e-5 for AdamW) and we should use warm-up

Fine-tuning of GPT

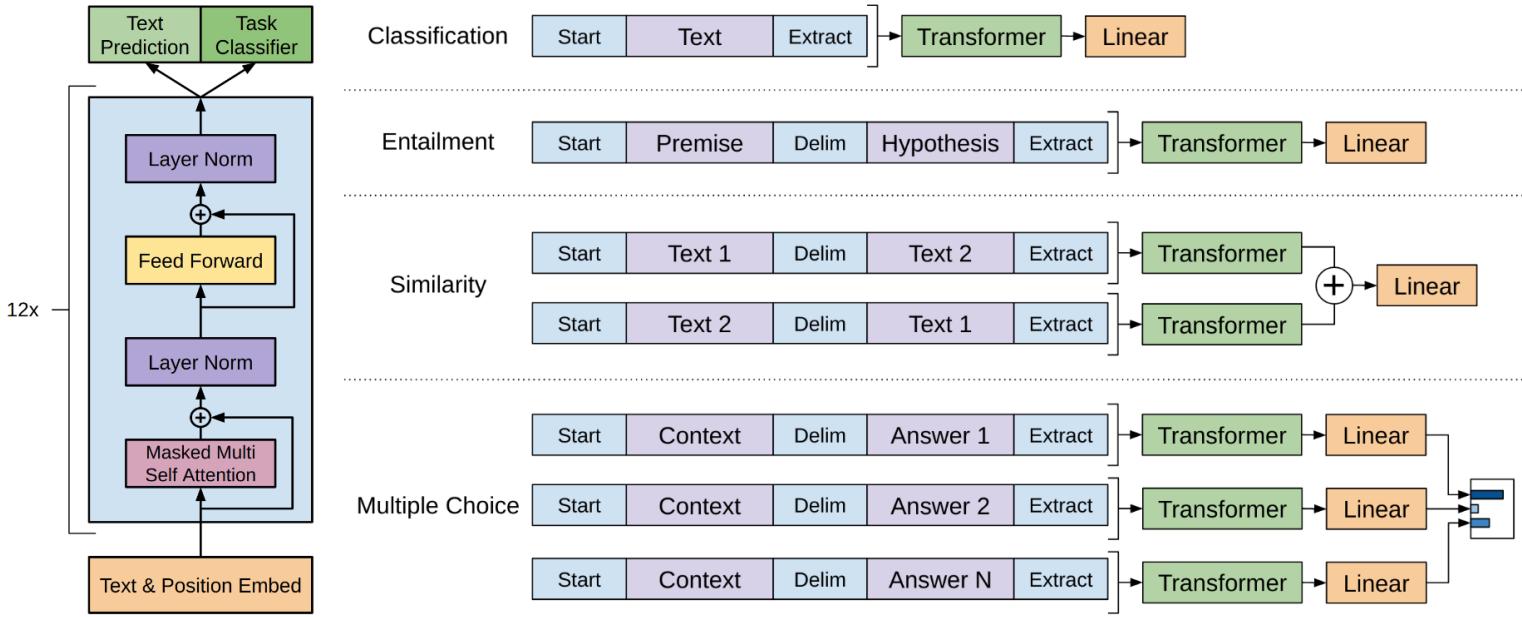
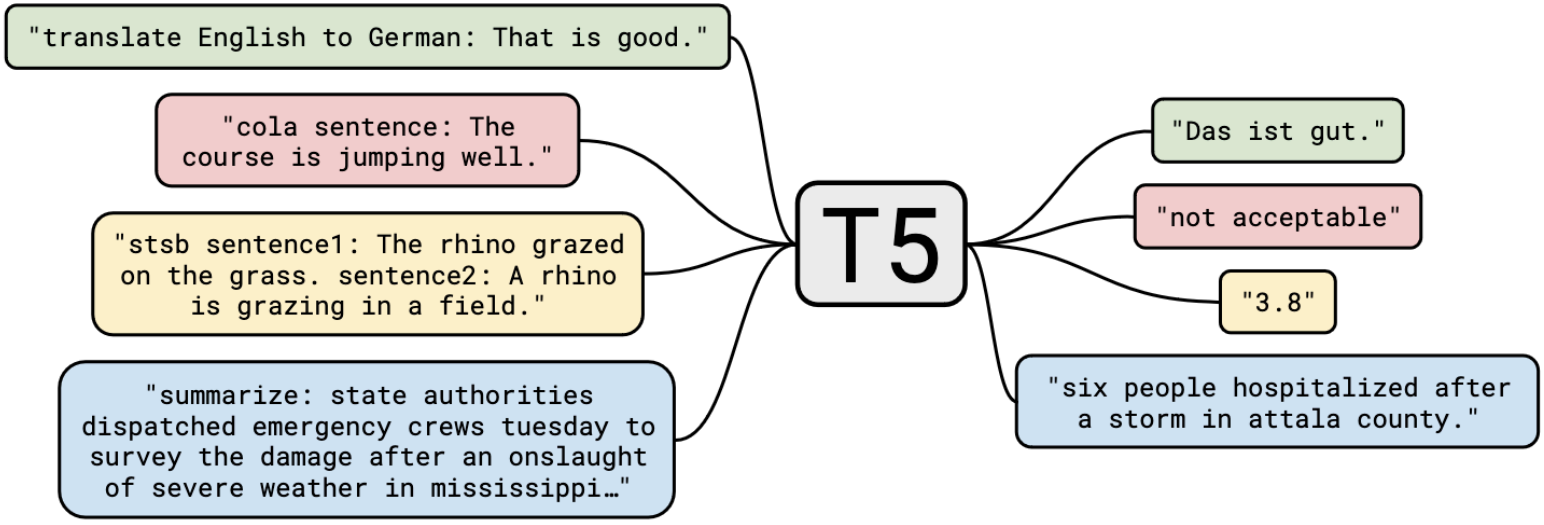


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

Fine-tuning of T5



- A different approach than BERT and GPT, it explain the name "text-to-text transfer transformer"
- We don't need any classifier on top of T5, we reformulate classification tasks as sequence-to-sequence tasks
- This approach can be used in any generative LM, for example GPT

Table of contents



1. Pre-training / fine-tuning paradigm

2. Downstream fine-tuning

3. Prompt-based learning

4. Instruction fine-tuning

5. References

Formal definition

- Supervised learning: we have a dataset of input-output pairs and we learn to assign the correct output y to any input x
  - We directly optimize $P(y|x;\theta_S)$
- But: We don't always have a large enough annotated dataset!
- Then: Let's use a language model that learns $P(x|\theta_{LM})$ of any text x and reformulate the task so that we can approximate $P(y|x;\theta_S)$ by language modeling
  - We will use a prompt template f that can transform any input-output pair x, y into natural text: $f(x, y)$

$$argmax_y\ P(y|x;\theta_S) \approx argmax_y\ P(f(x,y)|\theta_{LM})$$



Example (filled prompt)

- $x =$ *"I really enjoyed this movie."*
- $y =$ *"positive"*, $y' =$ *"negative"*



Example (filled prompt)

- $x =$ *"I really enjoyed this movie."*
- $y =$ *"positive"*, $y' =$ *"negative"*

- $f =$ "[X] *To sum up, it was a* ['good' if $Y ==$ 'positive' else 'bad'] *movie."*



Example (filled prompt)

- $x =$ *"I really enjoyed this movie."*
- $y =$ *"positive"*, $y' =$ *"negative"*

- $f =$ "[X] *To sum up, it was a* ['good' if $Y ==$ 'positive' else 'bad'] *movie."*

- $f(x, y) =$ *"I really enjoyed this movie. To sum up, it was a good movie."*
- $f(x, y') =$ *"I really enjoyed this movie. To sum up, it was a bad movie."*



Example (filled prompt)

- $x =$ *"I really enjoyed this movie."*
- $y =$ *"positive"*, $y' =$ *"negative"*
- $f =$ "[X] *To sum up, it was a* ['good' if $Y ==$ 'positive' else 'bad'] *movie."*
- $f(x, y) =$ *"I really enjoyed this movie. To sum up, it was a good movie."*
- $f(x, y') =$ *"I really enjoyed this movie. To sum up, it was a bad movie."*
- Predict *"positive"* iff:

$$P(f(x, y)|\theta_{LM}) > P(f(x, y')|\theta_{LM})$$



Cloze and prefix prompts

- This was an example of a filled prompt, which contains both the input x and answer y
- But some tasks have to be generated for efficiency (translation, summarization) and some APIs don't allow for probability estimation



Cloze and prefix prompts

- This was an example of a filled prompt, which contains both the input x and answer y
- But some tasks have to be generated for efficiency (translation, summarization) and some APIs don't allow for probability estimation

- Then we need cloze prompts (for MLMs) or prefix prompts (for CLMS)
- Here, the prompt template f only transforms x and the missing y is generated by a language model

Example (cloze prompt)

- $x =$ *"I felt asleep in the middle."*
- $y =$ *"positive"*, $y' =$ *"negative"*

Example (cloze prompt)

- $x =$ *"I felt asleep in the middle."*
- $y =$ *"positive"*, $y' =$ *"negative"*

- $f =$ "[X] *To sum up, it was a* [Y] *movie."*

Example (cloze prompt)

- $x =$ *"I felt asleep in the middle."*
- $y =$ *"positive"*, $y' =$ *"negative"*
- $f =$ "[X] *To sum up, it was a* [Y] *movie."*
- $f(x) =$ *"I felt asleep in the middle. To sum up, it was a* [Y] *movie."*

Example (cloze prompt)

- $x =$ *"I felt asleep in the middle."*
- $y =$ *"positive"*, $y' =$ *"negative"*
- $f =$ "[X] *To sum up, it was a* [Y] *movie."*
- $f(x) =$ *"I felt asleep in the middle. To sum up, it was a* [Y] *movie."*
- Now we have to generate text to replace [Y]

Example (cloze prompt)

- $x =$ *"I felt asleep in the middle."*
- $y =$ *"positive"*, $y' =$ *"negative"*
- $f =$ "[X] *To sum up, it was a* [Y] *movie."*
- $f(x) =$ *"I felt asleep in the middle. To sum up, it was a* [Y] *movie."*
- Now we have to generate text to replace [Y]
  - *"I felt asleep in the middle. To sum up, it was a boring movie."*
- Our language model decided to generate *"boring"*, we have to prepare a mapping function that aligns *"boring"* with *"negative"*



Example (prefix prompt)

- $x =$ *"lang og svingete vei"*
- $y =$ any possible English translation



Example (prefix prompt)

- $x =$ *"lang og svingete vei"*
- $y =$ any possible English translation

- $f =$ *"The Norwegian sentence '*[X]*' can be translated as:* [Y]

Example (prefix prompt)

- $x =$ *"lang og svingete vei"*
- $y =$ any possible English translation
- $f =$ *"The Norwegian sentence '*[X]*' can be translated as:* [Y]
- $f(x) =$ *"The Norwegian sentence 'lang og svingete vei' can be translated as:* [Y]

Example (prefix prompt)

- $x =$ *"lang og svingete vei"*
- $y =$ any possible English translation
- $f =$ *"The Norwegian sentence '*[X]*' can be translated as:* [Y]
- $f(x) =$ *"The Norwegian sentence 'lang og svingete vei' can be translated as:* [Y]
- Now we have to generate text to replace [Y]



Example (prefix prompt)

- $x =$ *"lang og svingete vei"*
- $y =$ any possible English translation

- $f =$ *"The Norwegian sentence '*[X]*' can be translated as:* [Y]

- $f(x) =$ *"The Norwegian sentence 'lang og svingete vei' can be translated as:* [Y]

- Now we have to generate text to replace [Y]
  - *"The Norwegian sentence 'lang og svingete vei' can be translated as: long and winding road"*
- In the case of machine translation, the generated text can be directly used as y

Zero-shot and few-shot learning

- zero-shot == no supervised data
- few-shot == ”few” supervised samples of data
  - For example, “8-shot” means that we show a model 8 gold input-output pairs (so-called “shots”)
- Usually, no training is involved, we don’t change the weights as in fine-tuning
- The gold input-output pairs are a part of a prompt, it shows what is the expected behavior



Zero-shot and few-shot learning

- zero-shot == no supervised data
- few-shot == ”few” supervised samples of data
  - For example, “8-shot” means that we show a model 8 gold input-output pairs (so-called “shots”)
- Usually, no training is involved, we don’t change the weights as in fine-tuning
- The gold input-output pairs are a part of a prompt, it shows what is the expected behavior

- Much more efficient than fine-tuning, simpler, no overfitting, no need for large annotated datasets

Zero-shot and few-shot learning

- zero-shot == no supervised data
- few-shot == ”few” supervised samples of data
  - For example, “8-shot” means that we show a model 8 gold input-output pairs (so-called “shots”)
- Usually, no training is involved, we don’t change the weights as in fine-tuning
- The gold input-output pairs are a part of a prompt, it shows what is the expected behavior

- Much more efficient than fine-tuning, simpler, no overfitting, no need for large annotated datasets
- But we need a huge LM with emergent abilities and the performance is mediocre compared to fine-tuning on a sufficiently large dataset

Example (3-shot prefix prompt)

- $x =$ *"ahh, what a waste of time :("*
- $y =$ *"positive"*, $y' =$ *"negative"*

- $f =$ *"I felt asleep in the middle. -> negative; I really enjoyed this movie. -> positive; What an amazing ending! -> positive;* [X] -> [Y]"

- $f(x) =$ *"I felt asleep in the middle. -> negative; I really enjoyed this movie. -> positive; What an amazing ending! -> positive;* ahh, what a waste of time :(-> [Y]"

More information

- We have only scratched the surface of prompt-based learning
- If you want to learn more the the survey paper by Liu et al. (2023) is a good start!

Table of contents



1. Pre-training / fine-tuning paradigm

2. Downstream fine-tuning

3. Prompt-based learning

4. Instruction fine-tuning

5. References

Alignment

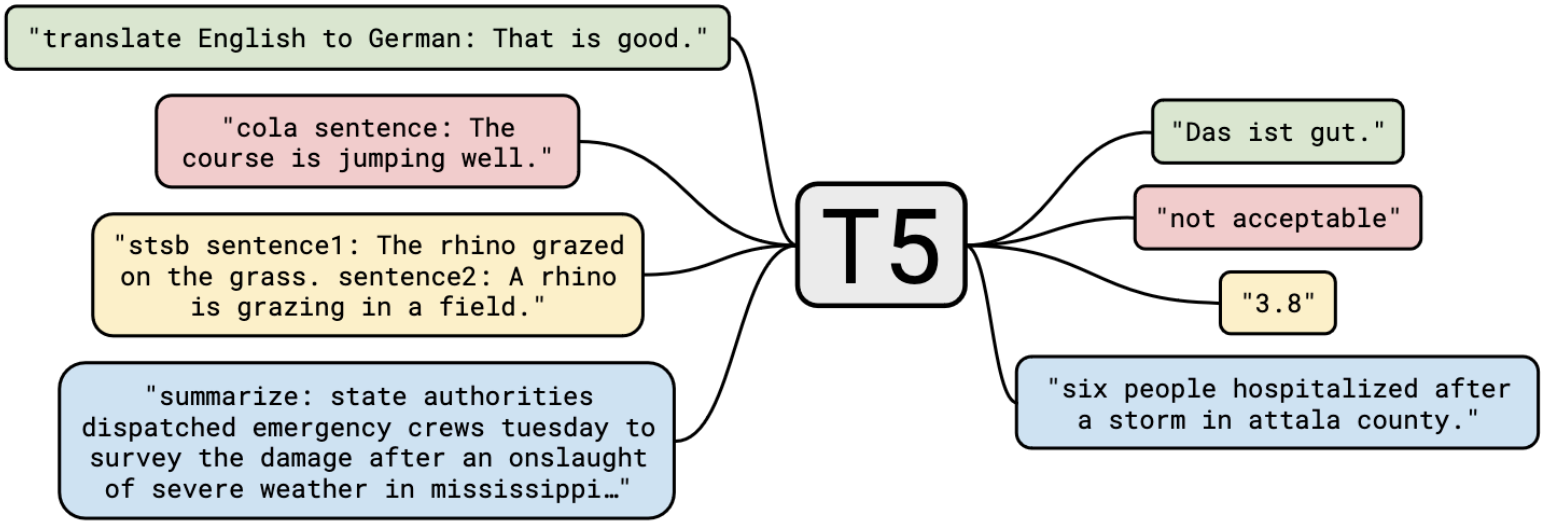
- LMs are pre-trained on enormous amounts of texts $\rightarrow$ they can imitate all kinds of characters and domains: article by a nuclear physicist, blogpost by a conspiracy theorist, or a mean comment by an internet troll
- A big part of prompting is persuading LMs to play the correct role
  -> we need to align their behavior with user intent

Alignment

- LMs are pre-trained on enormous amounts of texts → they can imitate all kinds of characters and domains: article by a nuclear physicist, blogpost by a conspiracy theorist, or a mean comment by an internet troll
- A big part of prompting is persuading LMs to play the correct role
  -> we need to align their behavior with user intent

- Or we can use instruction fine-tuning
  - We fine-tune a general LM to act as an obedient expert assistant that factually answers our questions

Remember T5?



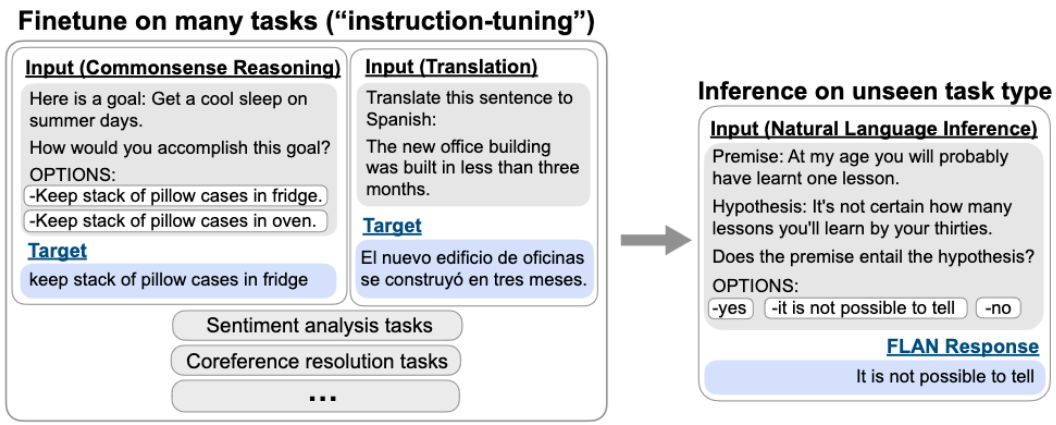
- Let's take a language model and fine-tune it on a bunch of tasks in the T5 instruction-prompt style
- Will it generalize to unseen tasks?

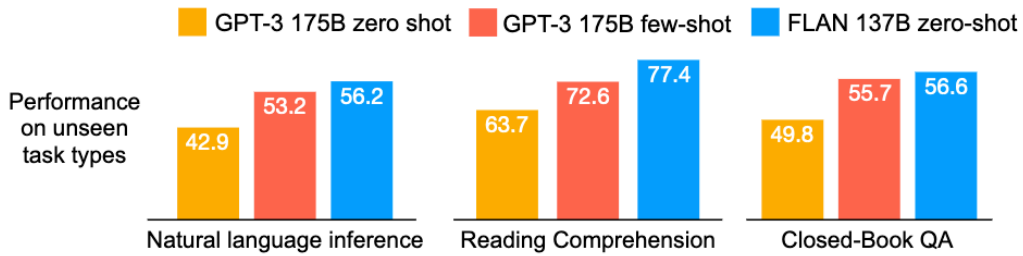
FLAN – Fine-tuned LANguage models (Wei et al., 2022)

Finetuned Language Models Are Zero-Shot Learners

Jason Wei*, Maarten Bosma*, Vincent Y. Zhao*, Kelvin Guu*, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le

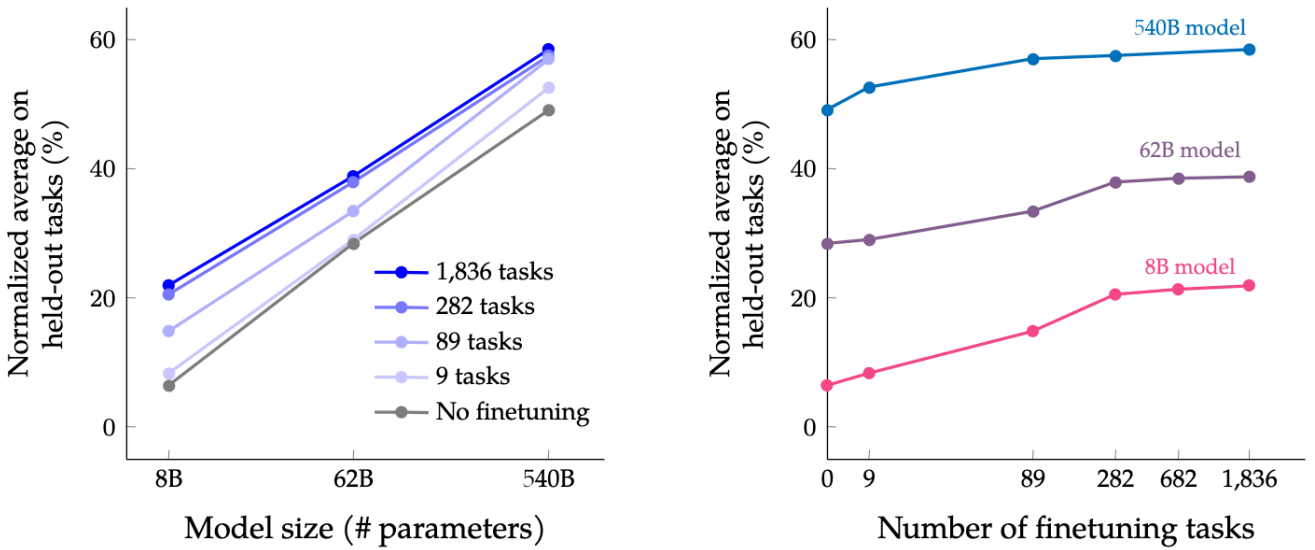
Google Research





FLAN

- Will it generalize? Yes! It substantially outperforms "pure" language models
- FLAN (Wei et al., 2022) was fine-tuned on a collection of 60 tasks
- Later, FLAN-T5 and FLAN-PaLM (Chung et al., 2022) was fine-tuned on 1836 tasks





is FLAN “the FLAN paper”?

- Note: T5 is not the first paper that casts all tasks as text-to-text, there are much older articles with a similar idea (Kumar et al., 2016)
- Note: FLAN is not the first paper that does instruction learning, this was introduced in older articles (Weller et al., 2020)
- But FLAN models are very popular nowadays, FLAN-T5s are openly released on HuggingFace

Instruction fine-tuning

What about ChatGPT?



What about ChatGPT?

- ChatGPT is simply an InstructGPT model fine-tuned on conversational data



What about ChatGPT?

- ChatGPT is simply an InstructGPT model fine-tuned on conversational data
  - well, as far as we can tell



What about ChatGPT?

- ChatGPT is simply an InstructGPT model fine-tuned on conversational data
  - well, as far as we can tell
- But what is InstructGPT?



ChatGPT → InstructGPT

- InstructGPT (Ouyang et al., 2022) is another, more complicated, approach to instruction fine-tuning
- Instead of "artificial" NLP datasets used by FLAN models, it is fine-tuned on more "human" problems:

Table 1: Distribution of use case categories from our API prompt dataset.

Use-case	(%)
Generation	45.6%
Open QA	12.4%
Brainstorming	11.2%
Chat	8.4%
Rewrite	6.6%
Summarization	4.2%
Classification	3.5%
Other	3.5%
Closed QA	2.6%
Extract	1.9%

Table 2: Illustrative prompts from our API prompt dataset. These are fictional examples inspired by real usage—see more examples in Appendix A.2.1.

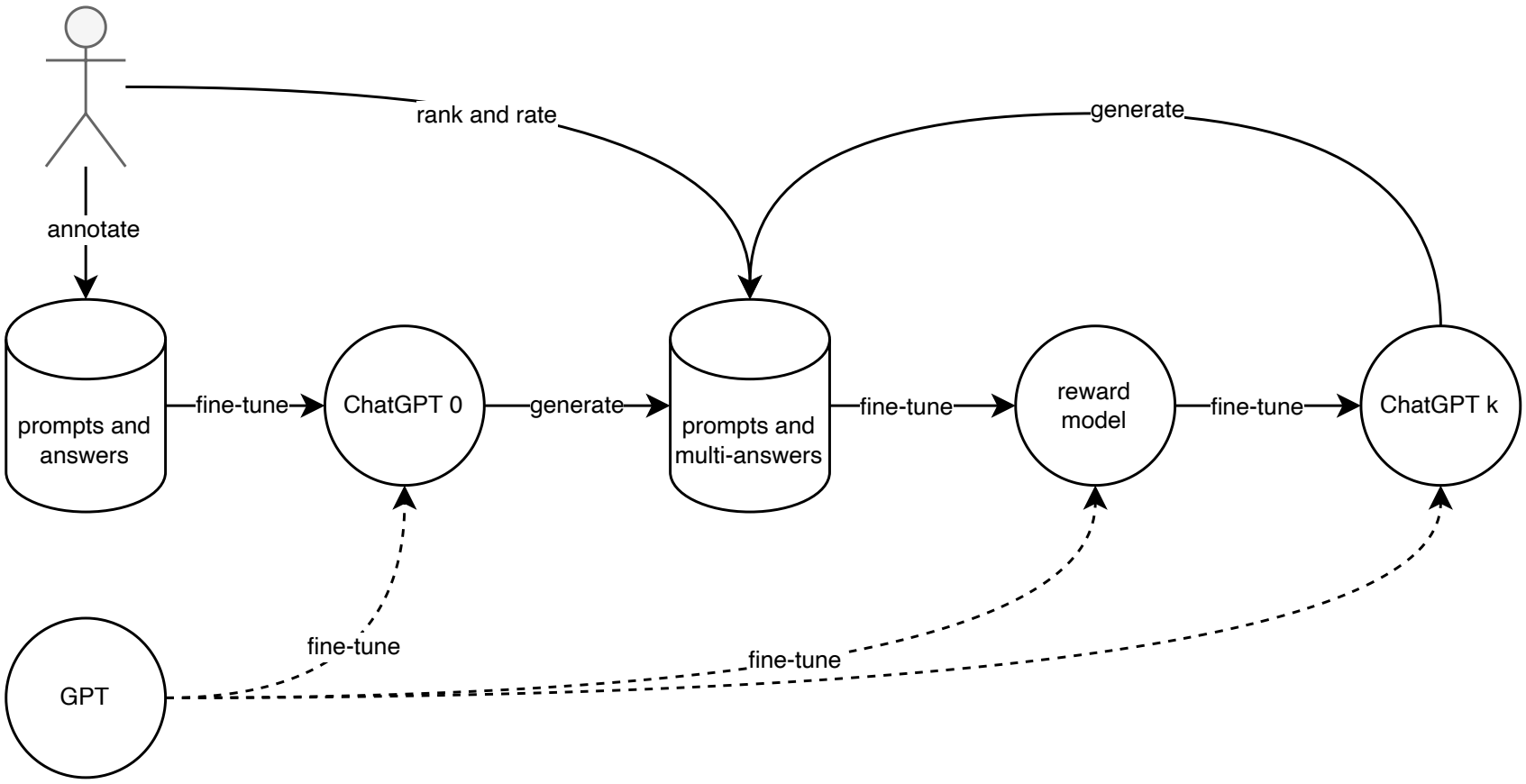
Use-case	Prompt
Brainstorming	List five ideas for how to regain enthusiasm for my career
Generation	Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home.
Rewrite	This is the summary of a Broadway play: """ {summary} """ This is the outline of the commercial for that play: """



ChatGPT → InstructGPT

- InstructGPT (Ouyang et al., 2022) is another, more complicated, approach to instruction fine-tuning
- It is based on RLHF – reinforcement learning from human feedback (Christiano et al., 2017)
  1. First, we **manually** create a dataset with prompts and gold answers
  2. Then we instruction fine-tune a LM on this data
  3. Then we take this model, generate a bunch of answers to every prompt from a new dataset and **manually** rank these answers
  4. Then we train a special 'reward' model that assigns a score for every pair of prompt and answer
  5. Then, finally, we use the reward model to fine-tune a language model with proximal policy optimization (PPO; Schulman et al., 2017)

ChatGPT → InstructGPT





ChatGPT → InstructGPT → PPO (optional)

- A deep reinforcement learning (RL) algorithm
  - Out of scope for this course, but super interesting!
  - An agent appears in an environment where it does a sequence of actions and gets a reward; we try to maximize the total reward
  - As opposed to supervised learning, which gives a reward for every action
  - Much more difficult to optimize but essential for intelligent agents
- PPO is one of many algorithms that optimize the behavior of an agent according to a reward function
- In our case, the environment is a human (who has a question and rates the answer), the agent is a language model and its action generates a new token



ChatGPT $\rightarrow$ InstructGPT $\rightarrow$ PPO (optional)

- The problem of RL: when an agent performs actions, their consequences in the environment are not differentiable
  - In our case, how would you differentiate the response of a human to a generated answer?
- We can't just directly maximize the reward! :(



ChatGPT → InstructGPT → PPO (optional)

- The problem of RL: when an agent performs actions, their consequences in the environment are not differentiable
  - In our case, how would you differentiate the response of a human to a generated answer?
- We can't just directly maximize the reward! :(

- One solution is to use an actor-critic algorithm, where the actor performs actions and the critic is a world model that simulates the environment and approximates the reward function
- The critic is a neural network → it is differentiable → we can teach the actor to maximize critic's reward!
- PPO is an actor-critic algorithm with some minor improvements

ChatGPT → InstructGPT → PPO (optional)

- This is the basic idea, the reality is more complicated and involves a lot of ad-hoc hyperparameters (Ouyang et al., 2022)

maximize the critic's reward

but stay close to the FLAN-tuned model

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

alright, but could you also not forget how you were pre-trained?!



ChatGPT

- We went from self-supervised language models that perform great with minimal supervision, to these super complicated contraptions that require large amounts of human annotation
- Is this the future?

Table of contents



1. Pre-training / fine-tuning paradigm
2. Downstream fine-tuning
3. Prompt-based learning
4. Instruction fine-tuning
5. References

References i



Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. (2017). Deep reinforcement learning from human preferences. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. (2022). Scaling instruction-finetuned language models.

Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R., and Socher, R. (2016). Ask me anything: Dynamic memory networks for natural language processing.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Gray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. (2022). Training language models to follow instructions with human feedback. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

References II



Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2022). Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Weller, O., Lourie, N., Gardner, M., and Peters, M. (2020). Learning from task descriptions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1361–1375, Online. Association for Computational Linguistics.