CS 577: NATURAL LANGUAGE PROCESSING

Abulhair Saparov

Lecture 17: Distillation

LAST TIME: PRUNING VIA L0 REGULARIZATION

- Reminder: Last time, we discussed model pruning.
  - I.e., removing parts of the model to make it smaller and faster.
- Among other methods, we discussed L0-regularization:

$$\arg\min_\theta L(\theta) \text{ subject to } \sum_i ||\theta_i||_0 = k$$

- This is not differentiable, so we introduced continuous mask variables z_i:

$$\arg\min_\theta \max_\lambda \mathbb{E}_z[L(\theta \odot z) + \lambda(k - \sum_i z_i)]$$

$$= \arg\min_\theta \max_\lambda \mathbb{E}_u[L(\theta \odot z)] + \lambda(k - \sum_i \mathbb{E}_u[z_i])$$

$$= \arg\min_{\theta,\alpha} \max_\lambda \mathbb{E}_u[L(\theta \odot z)] + \lambda \sum_i \sigma(\alpha_i - \beta \log(0.1/1.1))$$

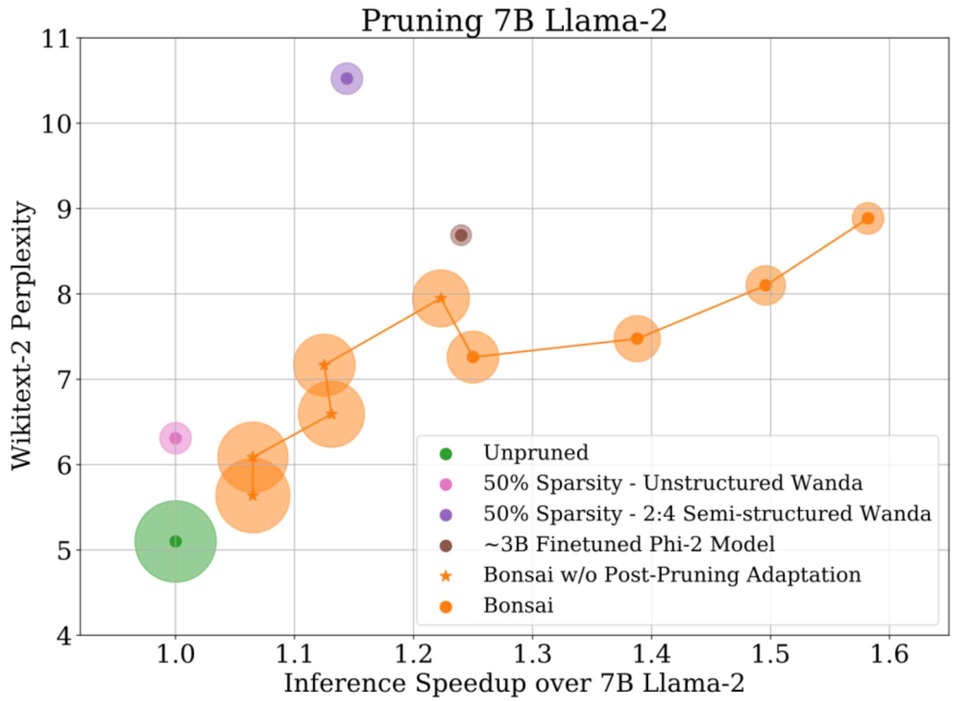
- We train the model using this objective, and the learned z_i parameters tells us which parts of the model to prune.

PRUNING VIA L0 REGULARIZATION

- Pruning via L0 regularization and training can be expensive.
    - Especially in terms of memory (we need to store gradients for the mask variables).
- Can we avoid training?
- What if the model is so large that we can only do forward passes?
- One idea is to use forward passes to estimate the “relevance” of various model components.
    - As before, the model components can be attention heads, FF dimensions, embeddings dimensions, or even entire layers.
- Once we have an estimated relevance value for each component, prune the components with the lowest relevance.

PRUNING VIA FORWARD PASSES

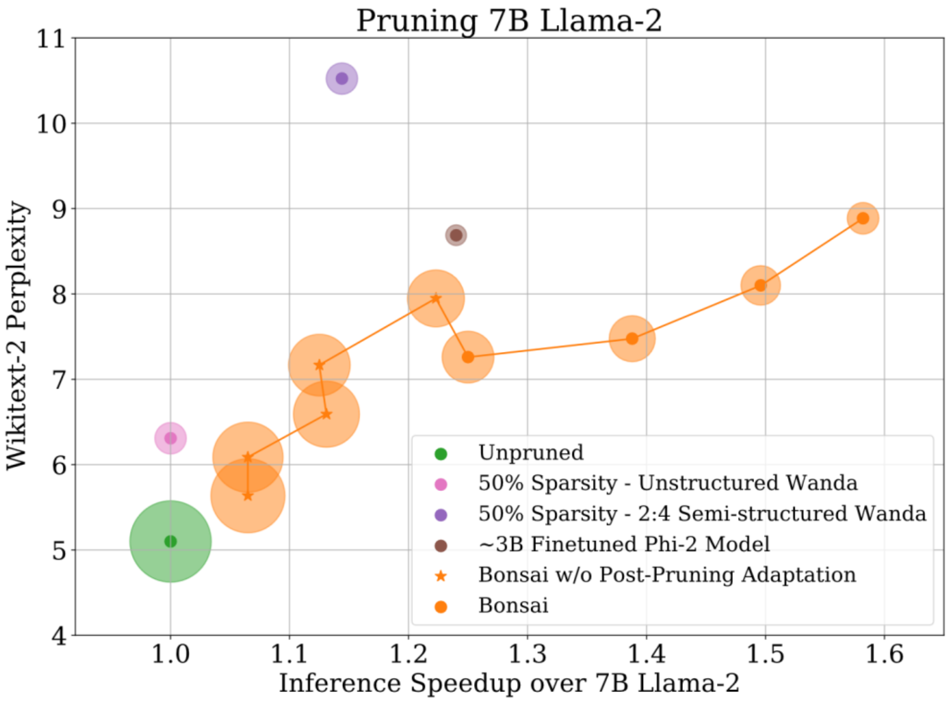
- This approach was proposed by Dery et al. (2024) and is called Bonsai.
- The size of the circle indicates the model size.



[Dery et al., 2024]

PRUNING VIA FORWARD PASSES

- The resulting pruned model can be made significantly faster than 2:4 structured Wanda,
  - But is also more accurate (lower perplexity).



[Dery et al., 2024]

EFFECTS OF PRUNING

- We find that pruning generally causes decrease in model performance on downstream tasks.
- But what about other aspects of the model?
  - Out-of-distribution performance?
  - Hallucination frequency?
  - etc...
- Much remains unexplored.
- Chrysostomou and Zhao and Williams et al. (2024) examined some of these other properties of pruned models (using SparseGPT and Wanda).

EFFECTS OF PRUNING

- Chrysostomou and Zhao and Williams et al. (2024) used automated metrics to quantify hallucination risk.
- They focus on the summarization task:
  - Given a document, output a short summary that contains all the important high-level information in the given document.
- Then they compute the hallucination risk ratio,
  - Where a ratio of less than 1 indicates the pruned model has lower hallucination risk than the original model.
  - A ratio of greater than 1 indicates the pruned model has greater hallucination risk than the original model.

EFFECTS OF PRUNING

Dataset	Metric	Llama-2 7B				Llama-2 13B				Llama-2 70B				Mistral 7B				OPT-IML 30B			
		SparseGPT		Wanda		SparseGPT		Wanda		SparseGPT		Wanda		SparseGPT		Wanda		SparseGPT		Wanda	
		2:4	50%	2:4	50%	2:4	50%	2:4	50%	2:4	50%	2:4	50%	2:4	50%	2:4	50%	2:4	50%	2:4	50%
FactCC	HaRiM$^+$	**0.98**	**0.95**	**0.94**	**0.95**	**0.77**	**0.95**	**0.69**	**0.91**	**0.93**	**0.96**	**0.93**	**0.96**	**0.93**	**0.94**	**0.91**	**0.94**	**0.83**	**0.87**	**0.87**	**0.85**
	SummaC$_{conv}$	**0.64**	**0.82**	**0.56**	**0.81**	**0.76**	**0.83**	**0.64**	**0.84**	**0.76**	**0.92**	**0.77**	**0.90**	**0.79**	**0.88**	**0.74**	**0.86**	**0.80**	**0.86**	**0.84**	**0.83**
	SummaC$_{zs}$	**0.47**	**0.65**	**0.39**	**0.65**	**0.50**	**0.61**	**0.41**	**0.61**	**0.63**	**0.86**	**0.63**	**0.83**	**0.76**	**0.85**	**0.68**	**0.82**	**0.80**	**0.87**	**0.85**	**0.83**
Polytope	HaRiM$^+$	**0.97**	**0.97**	**0.97**	**0.97**	**0.78**	**0.93**	**0.71**	**0.85**	**0.94**	0.96	**0.95**	1.00	**0.95**	**0.95**	**0.94**	**0.96**	**0.87**	**0.93**	**0.92**	**0.88**
	SummaC$_{conv}$	**0.67**	**0.83**	**0.69**	**0.83**	**0.70**	**0.78**	**0.65**	**0.79**	**0.77**	**0.93**	**0.78**	**0.92**	**0.78**	**0.82**	**0.76**	**0.84**	**0.86**	**0.95**	**0.91**	**0.92**
	SummaC$_{zs}$	**0.64**	**0.85**	**0.64**	**0.75**	**0.58**	**0.69**	**0.56**	**0.69**	**0.75**	**0.88**	**0.74**	**0.83**	**0.76**	**0.81**	**0.75**	**0.84**	**0.88**	**0.95**	**0.92**	**0.93**
SummEval	HaRiM$^+$	**0.88**	**0.93**	**0.81**	**0.93**	**0.80**	**0.97**	**0.69**	**0.96**	**0.95**	0.98	**0.95**	0.98	**0.93**	**0.94**	**0.92**	**0.95**	**0.91**	**0.92**	**0.90**	**0.89**
	SummaC$_{conv}$	**0.55**	**0.81**	**0.46**	**0.76**	**0.67**	**0.81**	**0.59**	**0.81**	**0.78**	0.96	**0.79**	0.93	**0.79**	**0.85**	**0.77**	**0.87**	**0.86**	**0.88**	**0.83**	**0.85**
	SummaC$_{zs}$	**0.49**	**0.75**	**0.4**	**0.68**	**0.56**	**0.71**	**0.49**	**0.66**	**0.70**	**0.92**	**0.70**	**0.88**	**0.79**	**0.84**	**0.76**	**0.88**	**0.86**	**0.89**	**0.85**	**0.86**
Legal Contracts	HaRiM$^+$	**0.99**	**0.85**	**0.90**	**0.85**	**0.83**	**0.88**	**0.76**	**0.88**	**0.87**	**0.92**	**0.89**	**0.95**	**0.85**	**0.94**	**0.89**	**0.93**	**0.85**	**0.89**	**0.81**	**0.83**
	SummaC$_{conv}$	0.98	0.85	0.93	0.94	**0.82**	**0.81**	**0.76**	**0.81**	**0.79**	**0.88**	**0.83**	**0.91**	**0.83**	**0.92**	**0.92**	**0.89**	**0.85**	**0.88**	**0.81**	**0.86**
	SummaC$_{zs}$	1.01	**0.86**	0.96	**0.90**	**0.93**	**0.86**	**0.88**	**0.88**	**0.85**	0.93	**0.88**	0.95	**0.88**	**0.92**	**0.93**	**0.92**	**0.93**	0.96	**0.94**	1.00
RCT	HaRiM$^+$	**0.92**	**0.96**	**0.87**	**0.92**	**0.86**	**0.99**	**0.80**	**0.97**	**0.93**	**0.96**	**0.93**	**0.97**	**0.93**	**0.96**	**0.93**	**0.95**	**0.85**	**0.88**	**0.83**	**0.87**
	SummaC$_{conv}$	**0.69**	**0.86**	**0.70**	**0.88**	**0.78**	**0.89**	**0.79**	**0.88**	**0.82**	**0.92**	**0.82**	**0.93**	**0.82**	**0.88**	**0.81**	**0.87**	**0.83**	**0.88**	**0.79**	**0.88**
	SummaC$_{zs}$	**0.71**	**0.83**	**0.71**	**0.82**	**0.69**	**0.81**	**0.70**	**0.82**	**0.79**	**0.90**	**0.79**	**0.90**	**0.84**	**0.89**	**0.82**	**0.89**	**0.77**	**0.80**	**0.77**	**0.83**
Average	HaRiM$^+$	0.95	0.93	0.90	0.92	0.81	0.95	0.73	0.91	0.92	0.96	0.93	0.97	0.92	0.95	0.92	0.95	0.87	0.90	0.87	0.87
	SummaC$_{conv}$	0.70	0.83	0.67	0.85	0.74	0.82	0.68	0.83	0.78	0.92	0.80	0.92	0.80	0.87	0.80	0.87	0.84	0.89	0.84	0.87
	SummaC$_{zs}$	0.67	0.79	0.62	0.76	0.65	0.74	0.61	0.73	0.74	0.90	0.75	0.88	0.81	0.86	0.79	0.87	0.85	0.90	0.86	0.89

[Chrysostomou and Zhao and Williams et al., 2024]

EFFECTS OF PRUNING

- Interestingly, pruned models are less likely to hallucinate.
- 2:4 structured pruned models are less likely to hallucinate than 50% unstructured pruned models.
- The reduction in hallucination risk was larger in the smaller Llama models, as compared to Llama2-70B, Mistral-7B, and OPT-IML-30B.

EFFECTS OF PRUNING

- They also performed human evaluation on these models to validate the results from their automated hallucination risk metrics.
  - They gave each human evaluator 100 news articles as well as summaries from both the pruned model and original model.
  - Each evaluator was tasked to annotate:
    - Which summary has more hallucinations?
    - Which summary has more omissions?
    - Which summary has more repetitive information?
    - Which summary is more semantically aligned with the original article?

EFFECTS OF PRUNING

- Inter-annotator agreement (IAA) is measured using Cohen's kappa score.
  - A score closer to 1 indicates better agreement.
- Humans also find that the pruned model produces fewer hallucinations.

Model	Halluc. Q1 (↓)	Omiss. Q2 (↓)	Repet. Q3 (↓)	Align. Q4 (↑)
Llama-2 7B	31	**5**	**0**	**28**
w/ SparseGPT	**14**	18	9	21
IAA (κ)	0.82	0.63	0.62	0.53
Mistral 7B	12	**9**	**0**	**31**
w/ SparseGPT	**10**	13	5	23
IAA (κ)	0.87	0.61	0.67	0.59

[Chrysostomou and Zhao and Williams et al., 2024]

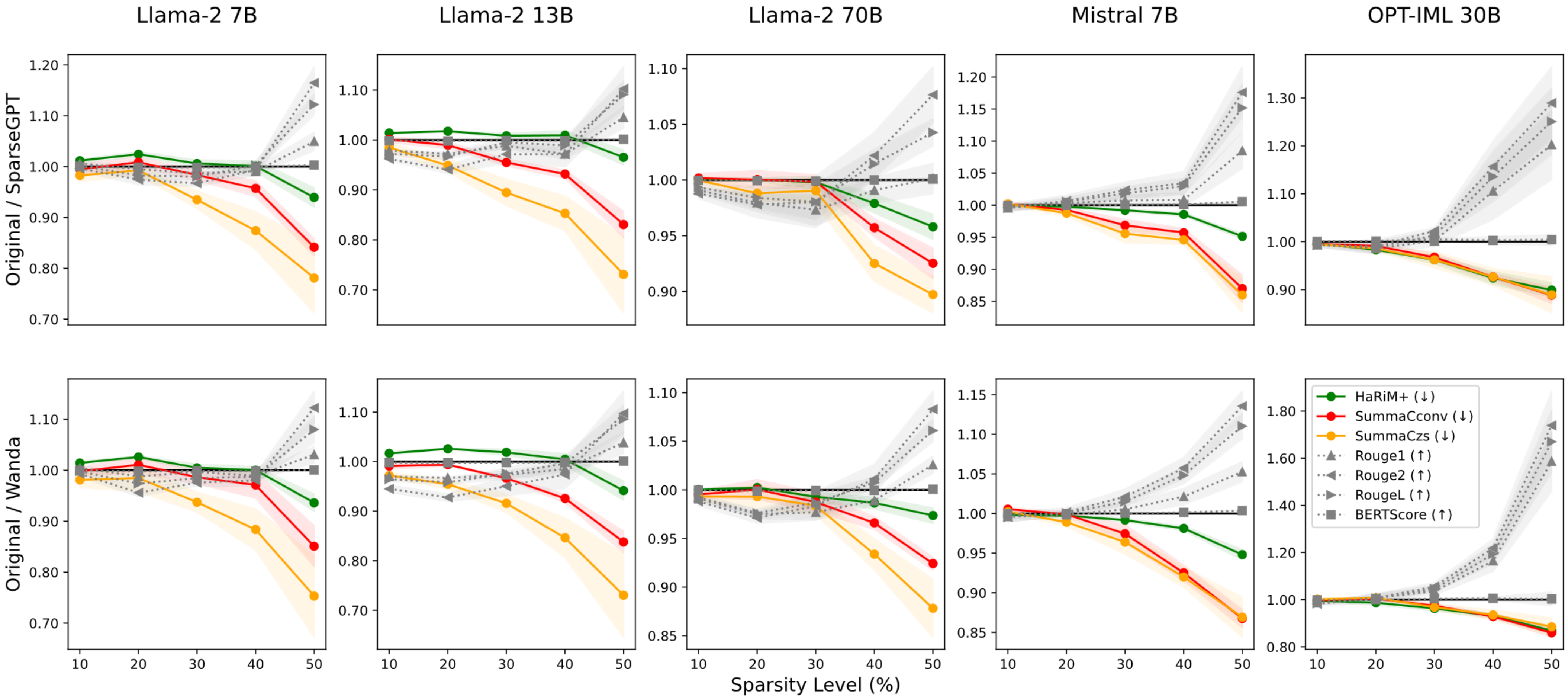
EFFECTS OF PRUNING

- But humans find that the pruned models omitted important information more often.
  - And the pruned models had more repetition.

Model	Halluc. Q1 (↓)	Omiss. Q2 (↓)	Repet. Q3 (↓)	Align. Q4 (↑)
Llama-2 7B	31	**5**	**0**	**28**
w/ SparseGPT	**14**	18	9	21
IAA (κ)	0.82	0.63	0.62	0.53
Mistral 7B	12	**9**	**0**	**31**
w/ SparseGPT	**10**	13	5	23
IAA (κ)	0.87	0.61	0.67	0.59

[Chrysostomou and Zhao and Williams et al., 2024]

HALLUCINATION RISK VS SPARSITY



[Chrysostomou and Zhao and Williams et al., 2024]

PRUNING SUMMARY

- In this lecture, we discussed pruning as a strategy to reduce the memory footprint of large models.
  - Pruning can also be used to make models faster.
  - But this is less likely for unstructured pruning methods.
- Structured pruning can produce models that are both smaller and faster.
- But there is an approximation cost.

MODEL COMPRESSION SUMMARY

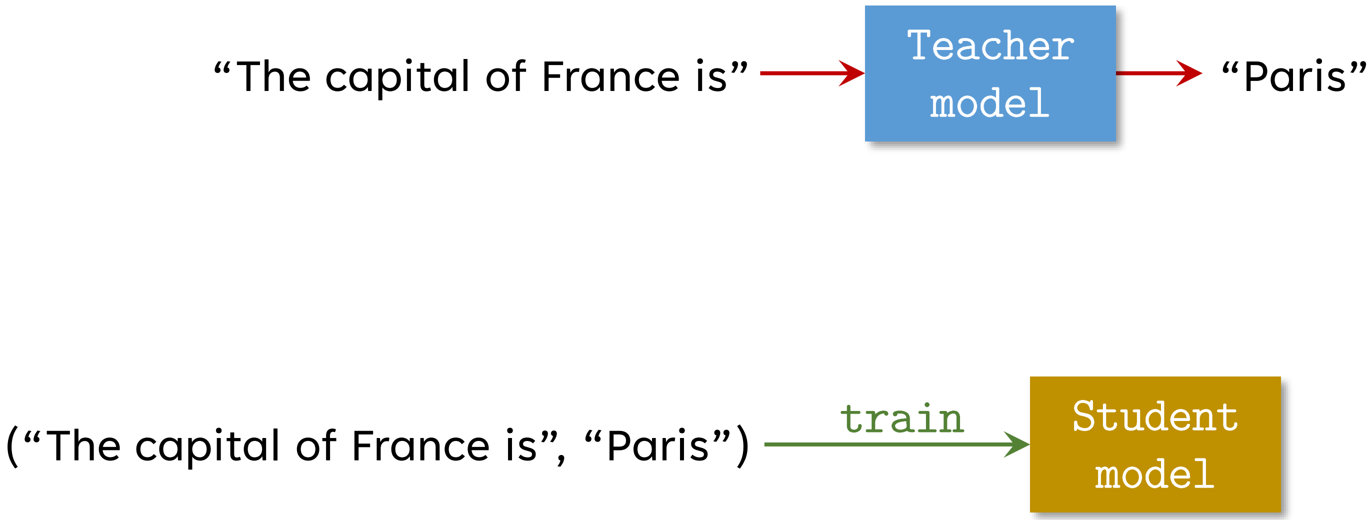
- So far, we have covered two high-level approaches for model compression:
  - Quantization: Reducing the precision of the parameters.
  - Pruning: Removing parts of the model.
- Next, we will discuss model distillation, where a larger model is used to teach smaller models,
  - With the goal of improving the performance of the smaller model to match that of the larger model.

WEAK SUPERVISION

- The idea of using another model to provide training labels for otherwise unlabeled training data is not new.
- In weak supervision, we start with unlabeled data.
  - We use another model to label the examples in the data,
  - These labels are called *pseudo-labels*.
  - Then we train the target model on the data with pseudo-labels.
- This is an old idea, with many adaptations/applications:
  - Self-training (Yarowski, 1995)
  - Co-training (Blum and Mitchell, 1998)
  - Meta pseudo-labels (Pham et al., 2020)

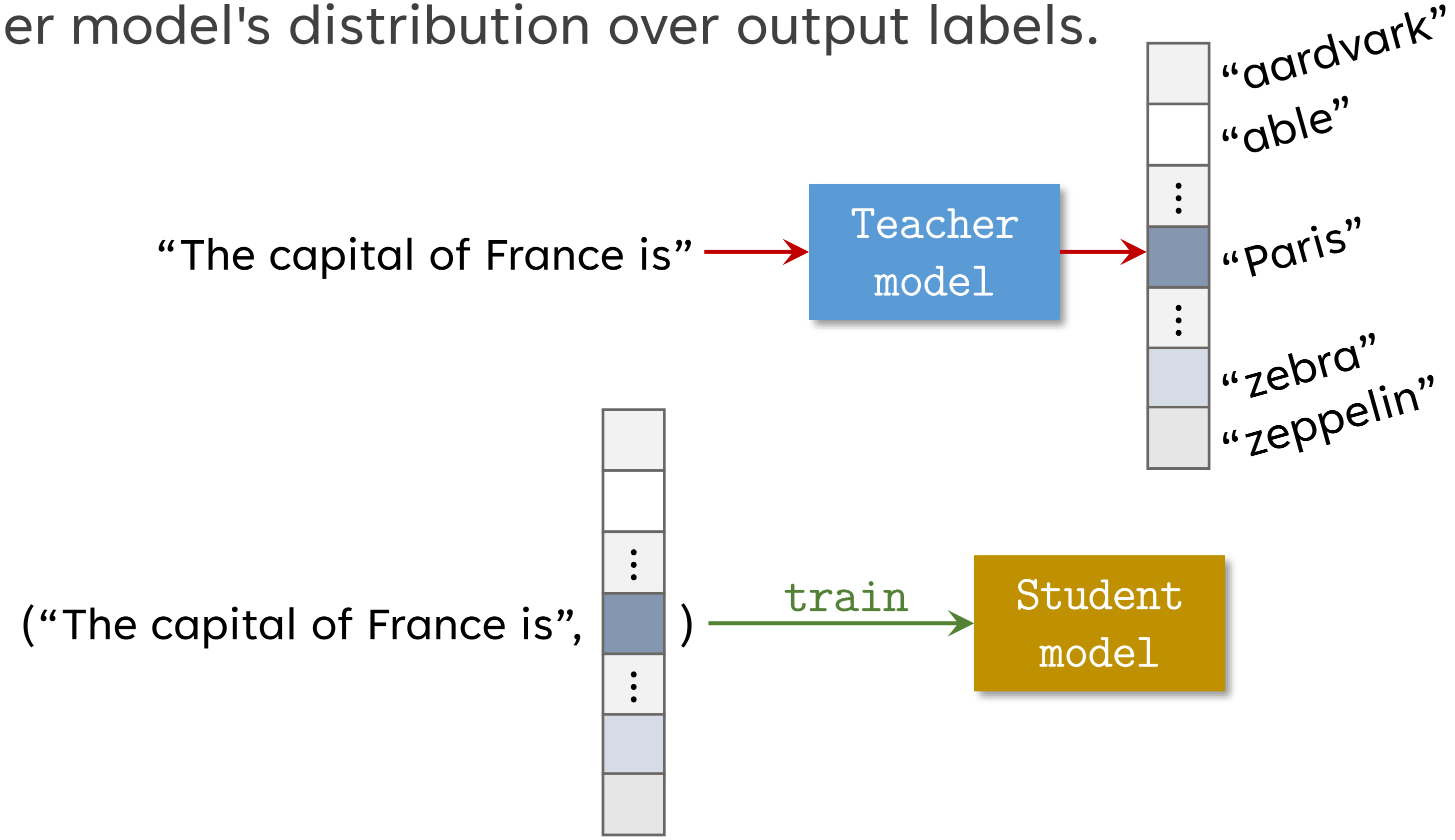
WEAK SUPERVISION ON HARD TARGETS

- In vanilla weak supervision, we use another model (i.e., “teacher”) to produce labels on a collection of unlabeled examples.



WEAK SUPERVISION ON SOFT TARGETS

- For probabilistic models, we have the option to train the “student” model on another model's distribution over output labels.



WEAK SUPERVISION ON SOFT TARGETS

- For probabilistic models, we have the option to train the model on another model's distribution over output labels.
- The cross-entropy loss can also be used to train probabilistic models on data with soft targets (where the pseudo-label is a distribution over labels).
- This approach was suggested by Hinton and Vinyals et al. (2015).
- Soft targets can contain a lot more information that hard targets alone.
  - For example, given the input “The capital of France is”,
  - The teacher model may assign high probability to capital cities,
  - And lower probability to cities that are not capitals,
  - And even lower probability to locations that are not cities, etc...
- The distribution has information about the expected type of the output.

WEAK SUPERVISION ON SOFT TARGETS

- This approach is called knowledge distillation.
- Hinton and Vinyals et al. (2015) tested this on the speech recognition task:
  - Given an input waveform of the recorded speech,
  - Predict a sequence of phonemes.
- An 85M parameter model was trained on:
  - Hard labels using 100% of the data,
  - Hard labels using 3% of the data,
  - Or soft labels using 3% of the data.

System & training set	Train Frame Accuracy	Test Frame Accuracy
Baseline (100% of training set)	63.4%	58.9%
Baseline (3% of training set)	67.3%	44.5%
Soft Targets (3% of training set)	65.4%	57.0%

INTERMEDIATE LAYER DISTILLATION

- For teacher and student models with multi-layer architectures, another distillation method is possible:
  - We can teach the student model to produce similar intermediate-layer activations as the teacher model.
  - The dimensionality of the intermediate activations in the teacher and student models must be the same.
- For transformer models, if the context lengths of the teacher and student models are the same,
  - We can perform intermediate layer distillation on the attention values at each layer.

INTERMEDIATE LAYER DISTILLATION

- We can use various loss functions for intermediate layer distillation:
  - Suppose x is the intermediate layer activation of the teacher model

    And y is that of the student model.
  - L2 loss: $||x - y||_2^2 = \sum_{i=1}^{n}(x_i - y_i)^2$.
  - Cosine "distance": $1 - \cos(\theta) = 1 - \frac{x^T y}{\|x\|_2 \|y\|_2}$.
  - Note that cosine distance does not take into account the magnitude of the input vectors.
- Since intermediate layer activations are not probabilities, there is no strong motivation to use cross-entropy loss/KL divergence.

DISTILBERT

- A well-known application of distillation is DistilBERT (Sanh et al., 2019).
- They removed half of the layers of BERT (Devlin et al., 2018), but did not change the model dimension.
    - The resulting model had 40% fewer parameters than BERT.
    - They initialized their smaller model by copying the parameters of every other layer of BERT.
- They trained DistilBERT on the same corpus as the original BERT.
    - Took 90 hours on 8 16GB V100 GPUs (720 GPU-hours).
    - In comparison, training RoBERTa took ~1 day on 1024 32GB V100s (~25,000 GPU-hours).

DISTILBERT

- Their loss function consisted of three terms:
  - The knowledge distillation loss from the teacher (cross-entropy).
  - The masked language modeling loss (cross-entropy).
    - This is the same loss you use when pretraining BERT,
    - And we can use it here since we are training DistilBERT on a dataset with ground truth labels, in addition to the teacher's pseudo-labels.
  - An intermediate layer distillation loss (cosine distance).

DISTILBERT

- They compared DistilBERT to BERT on the GLUE benchmark (a suite containing many language understanding tasks).

Table 1: **DistilBERT retains 97% of BERT performance.** Comparison on the dev sets of the GLUE benchmark. ELMo results as reported by the authors. BERT and DistilBERT results are the medians of 5 runs with different seeds.

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

[Sanh et al., 2019]

DISTILBERT

- They also compared the inference time of both DistilBERT and BERT and found that DistilBERT was about 39% faster.

Table 3: **DistilBERT is significantly smaller while being constantly faster.** Inference time of a full pass of GLUE task STS-B (sentiment analysis) on CPU with a batch size of 1.

Model	# param. (Millions)	Inf. time (seconds)
ELMo	180	895
BERT-base	110	668
DistilBERT	66	410

[Sanh et al., 2019]

DISTILBERT

- They also performed an ablation study to investigate the importance of the three terms in the loss function.
  - L_{ce} is the knowledge distillation loss (cross-entropy),
  - L_{cos} is the intermediate layer distillation loss (cosine distance),
  - And L_{mlm} is the masked language modeling loss.

Ablation	**Variation on GLUE macro-score**
$\emptyset$ - L_{cos} - L_{mlm}	-2.96
L_{ce} - $\emptyset$ - L_{mlm}	-1.46
L_{ce} - L_{cos} - $\emptyset$	-0.31
Triple loss + random weights initialization	-3.69

[Sanh et al., 2019]

DISTILBERT

- They also performed an ablation study to investigate the importance of the three terms in the loss function.
  - The first row is the result without the knowledge distillation loss.
  - The second row is the result without the intermediate layer distillation.
  - The third row is the result without the masked language modeling loss.

Ablation	**Variation on GLUE macro-score**
$\emptyset$ - L_{cos} - L_{mlm}	-2.96
L_{ce} - $\emptyset$ - L_{mlm}	-1.46
L_{ce} - L_{cos} - $\emptyset$	-0.31
Triple loss + random weights initialization	-3.69

[Sanh et al., 2019]

DISTILBERT

- Evidently, the knowledge distillation loss is important for achieving high accuracy,
  - As well as using the teacher model to initialize the weights of the student.
  - But the masked language model loss doesn't seem to help too much.

Ablation	**Variation on GLUE macro-score**
$\emptyset$ - L_{cos} - L_{mlm}	-2.96
L_{ce} - $\emptyset$ - L_{mlm}	-1.46
L_{ce} - L_{cos} - $\emptyset$	-0.31
Triple loss + random weights initialization	-3.69

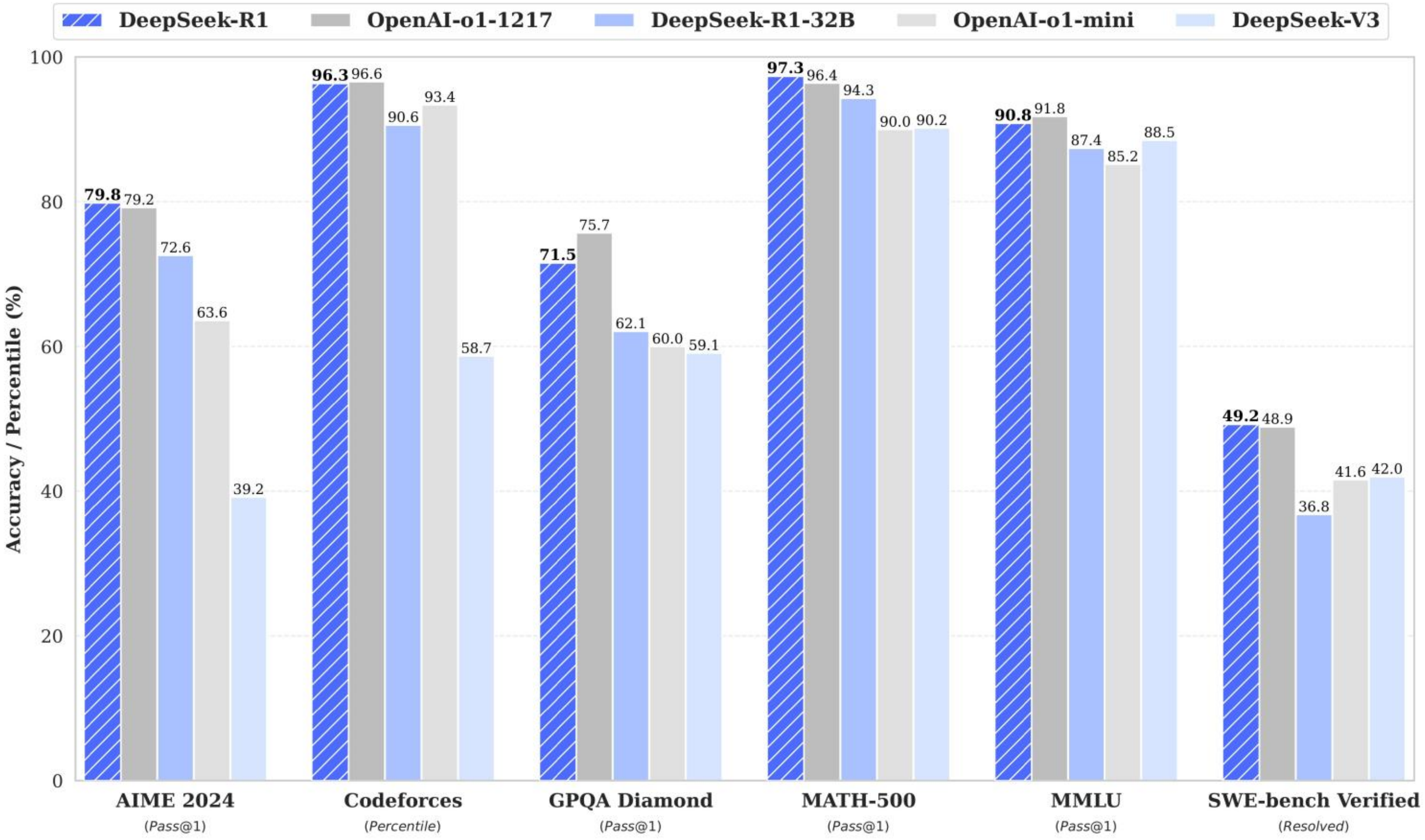
[Sanh et al., 2019]

DISTILLATION OF LLMS

- Distillation can be applied to large language models.
- For example, DeepSeek `r1` has 671B parameters, which makes it very difficult to run with limited hardware.
- `r1` was used to create a labeled dataset, with 800k examples, which can then be used to fine-tune smaller models.
  - DeekSeek (2025) fine-tuned `Qwen` and `Llama` models of various sizes.
  - They evaluated the distilled models on a handful of mathematical reasoning and programming benchmarks.

DISTILLATION OF LLMS

- The 32B distilled version of `r1` performed similarly to `r1`, but not identically.



[DeepSeek, 2025]

DISTILLATION OF LLMS

- Smaller models are more difficult to distill:
  - They are not as capable of mimicking the abilities of the teacher model.

Model	**AIME 2024**		**MATH-500**	**GPQA Diamond**	**LiveCode Bench**	**CodeForces**
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
GPT-4o-0513	9.3	13.4	74.6	49.9	32.9	759
Claude-3.5-Sonnet-1022	16.0	26.7	78.3	65.0	38.9	717
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	**1820**
QwQ-32B-Preview	50.0	60.0	90.6	54.5	41.9	1316
DeepSeek-R1-Distill-Qwen-1.5B	28.9	52.7	83.9	33.8	16.9	954
DeepSeek-R1-Distill-Qwen-7B	55.5	83.3	92.8	49.1	37.6	1189
DeepSeek-R1-Distill-Qwen-14B	69.7	80.0	93.9	59.1	53.1	1481
DeepSeek-R1-Distill-Qwen-32B	**72.6**	83.3	94.3	62.1	57.2	1691
DeepSeek-R1-Distill-Llama-8B	50.4	80.0	89.1	49.0	39.6	1205
DeepSeek-R1-Distill-Llama-70B	70.0	**86.7**	**94.5**	**65.2**	**57.5**	1633

[DeepSeek, 2025]

LIMITATIONS OF DISTILLATION

- Obtaining the distilled model is expensive:
  - We need to generate the data from the teacher model,
  - And perform fine-tuning on the student model.
- Once we have the distilled model, we are able to enjoy the savings in terms of both memory and compute.
- With standard distillation, the student model is not able to learn to perform any better than the teacher model.
- Question: Is it possible to train the student model to perform better than the teacher?
  - Yes, *if* we *augment* the teacher.
  - Let's look at some examples.

SELF-INSTRUCT

- Ideas from distillation/weak supervision can be applied to teach a model to improve its own capabilities.
- Consider the problem of instruction tuning:
  - We want to fine-tune a model to be better at following instructions.
  - Typically, this requires a large instruction tuning dataset.
    - This dataset contains many different tasks, each with a set of instructions, as well as a number of examples.
  - But creating such a dataset can be expensive.
- Can we use a model to generate its own instructing tuning dataset?

SELF-INSTRUCT

- A method called Self-Instruct (Wang et al., 2023) claims to do so.
- They start with a collection of 175 human-annotated tasks.
  - Each task has 1 instruction and 1 example.
- Step 1: Repeatedly prompt the model to generate task instructions.
  - They use 8-shot prompting.

```
Come up with a series of tasks:

Task 1:  {instruction for existing task 1}
Task 2:  {instruction for existing task 2}
Task 3:  {instruction for existing task 3}
Task 4:  {instruction for existing task 4}
Task 5:  {instruction for existing task 5}
Task 6:  {instruction for existing task 6}
Task 7:  {instruction for existing task 7}
Task 8:  {instruction for existing task 8}
Task 9:
```

[Wang et al., 2023]

SELF-INSTRUCT

- In each prompt, they randomly sample 6 human-written instructions and 2 model-generated instructions.
  - Their rationale was to increase the diversity of output instructions.
- They let the model generate more than 1 task instruction in each prompt, up to 8.

```
Come up with a series of tasks:

Task 1:  {instruction for existing task 1}
Task 2:  {instruction for existing task 2}
Task 3:  {instruction for existing task 3}
Task 4:  {instruction for existing task 4}
Task 5:  {instruction for existing task 5}
Task 6:  {instruction for existing task 6}
Task 7:  {instruction for existing task 7}
Task 8:  {instruction for existing task 8}
Task 9:
```

[Wang et al., 2023]

SELF-INSTRUCT

- Step 2: For each task instruction, they determine whether the task is a classification task or not.
  - Again using few-shot prompting.
- Step 3: Generate examples for each task.
  - For non-classification tasks, for each example, first prompt the model to generate an input.
  - Then given the task instruction and the example input, prompt the model to generate the output.
  - This was also done using few-shot prompting.

SELF-INSTRUCT

```
Come up with examples for the following tasks. Try to generate multiple examples when possible.
If the task doesn't require additional input, you can generate the output directly.

Task: Suggest a better and more professional rephrasing of the following sentence.
Example 1
Sentence: This house is surprisingly not constructed very well, and you probably need more
money to fix it after you buy it. If you ask me, I would suggest you to consider other
candidates.
Output: This house does not seem to be constructed well, so you may need to spend more money
to fix it after you purchase it. I would suggest that you look at other properties.
Example 2
Sentence: Just so you know, we did an experiment last week and found really surprising results
- language model can improve itself!
Output: Our experiments last week demonstrated surprising results, proving that the language
model can improve itself.

...

Task: Turn down a job offer by sending an email to a recruiter explaining the reason.
Output: Hi [Recruiter],
Thank you so much for the generous offer to join your team. As we discussed, I've admired the
company for a number of years, and am a proud endorser of its products. However, after further
consideration of where I currently am in my career, I've decided to accept an offer at another
company.
I would love to stay in touch with you and have already started following you on [Social Media
Platform]. Again, thank you so much for your time and consideration.
Thanks again,
[Your Name]

Task: {Instruction for the target task}
```

[Wang et al., 2023]

SELF-INSTRUCT

- Step 3 (continued): Generate examples for each task.
  - For classification tasks, for each example, first prompt the model to generate the possible output class labels.
  - Then randomly select a class label, and generate an input based on the selected output label.
  - This was also done using few-shot prompting.

SELF-INSTRUCT

```
Given the classification task definition and the class labels, generate an input that
corresponds to each of the class labels.  If the task doesn't require input, just generate the
correct class label.

Task:  Classify the sentiment of the sentence into positive, negative, or mixed.
Class label:  mixed
Sentence:  I enjoy the flavor of the restaurant but their service is too slow.
Class label:  Positive
Sentence:  I had a great day today.  The weather was beautiful and I spent time with friends.
Class label:  Negative
Sentence:  I was really disappointed by the latest superhero movie.  I would not recommend it.

...

Task:  Which of the following is not an input type?  (a) number (b) date (c) phone number (d)
email address (e) all of these are valid inputs.
Class label:  (e)

Task:  {instruction for the target task}
```

[Wang et al., 2023]

SELF-INSTRUCT

- Step 4: Fine-tune the model on the generated instruction tuning data.
  - The generated data is perturbed using templates to improve its diversity.
  - E.g., so that instructions are not always prefixed with “Task:” etc.
- Wang et al. (2023) performed this procedure using the base GPT3 model (`davinci`).
  - Their resulting dataset contains 52,445 instructions, and 82,439 examples.
  - They fine-tuned the same model using OpenAI’s API for 2 epochs.
    - (note: it’s unclear exactly how OpenAI’s fine-tuning API works)

SELF-INSTRUCT

- They compared the resulting model with:
  - Models that were not instruction-tuned.
  - Models instruction-tuned on `SuperNaturalInstructions` (Wang et al., 2022).
  - They evaluated on a held-out set of tasks in `SuperNI`.

Model	# Params	ROUGE-L
Vanilla LMs		
T5-LM	11B	25.7
GPT3	175B	6.8
Instruction-tuned w/o SUPERNI		
T0	11B	33.1
GPT3 + T0 Training	175B	37.9
$GPT3_{SELF-INST}$ (Ours)	175B	39.9
$InstructGPT_{001}$	175B	**40.8**
Instruction-tuned w/ SUPERNI		
Tk-INSTRUCT	11B	46.0
GPT3 + SUPERNI Training	175B	49.5
$GPT3_{SELF-INST}$ + SUPERNI Training (Ours)	175B	**51.6**

[Wang et al., 2023]

SELF-INSTRUCT

- Their approach is comparable to OpenAI's first instruction-tuned GPT3 model.
- Though not quite as good as models instruction-tuned using `SuperNI`.
- However, they found that combining their generated data with `SuperNI` yields even better performance.

Model	# Params	ROUGE-L
Vanilla LMs		
T5-LM	11B	25.7
GPT3	175B	6.8
Instruction-tuned w/o SUPERNI		
T0	11B	33.1
GPT3 + T0 Training	175B	37.9
$\text{GPT3}_{\text{SELF-INST}}$ (Ours)	175B	39.9
InstructGPT_{001}	175B	**40.8**
Instruction-tuned w/ SUPERNI		
T*k*-INSTRUCT	11B	46.0
GPT3 + SUPERNI Training	175B	49.5
$\text{GPT3}_{\text{SELF-INST}}$ + SUPERNI Training (Ours)	175B	**51.6**

[Wang et al., 2023]

SELF-INSTRUCT

- What is the quality of the self-generated instruction tuning data?
- One of the authors randomly selected 200 task instructions and 1 corresponding example for each task.
- They manually inspected and evaluated each instruction and example.

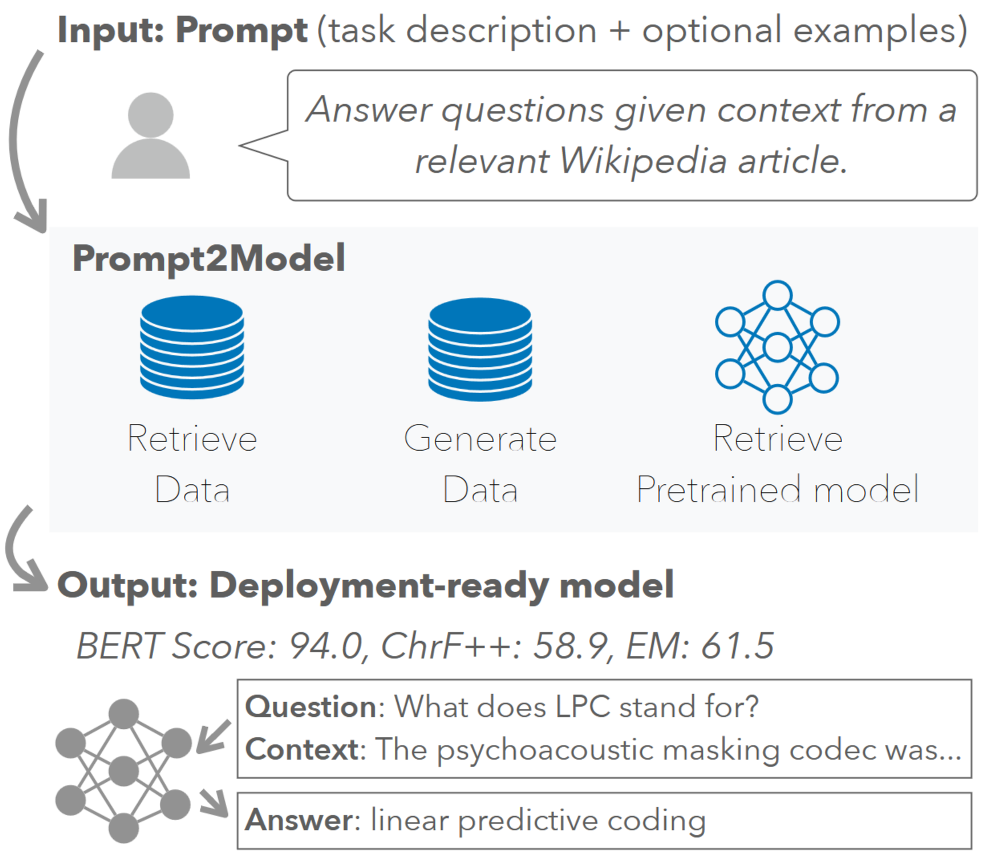
Quality Review Question	Yes %
Does the instruction describe a valid task?	92%
Is the input appropriate for the instruction?	79%
Is the output a correct and acceptable response to the instruction and input?	58%
All fields are valid	54%

But there is no comparison to other instruction tuning datasets.

[Wang et al., 2023]

PROMPT2MODEL

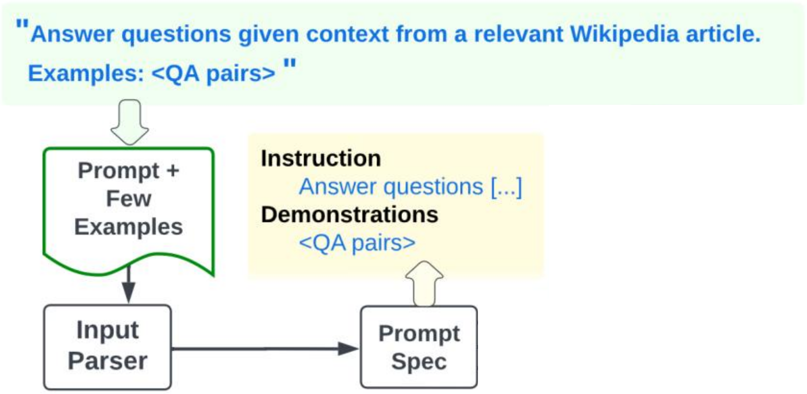
- Distillation can be used to teach task-specific student models.
- Prompt2Model (Viswanathan and Zhao et al., 2023) developed an automated pipeline that produces task-specific smaller models.
  - The input is a description of the task,
  - And an optional example.



[Viswanathan and Zhao et al., 2023]

PROMPT2MODEL

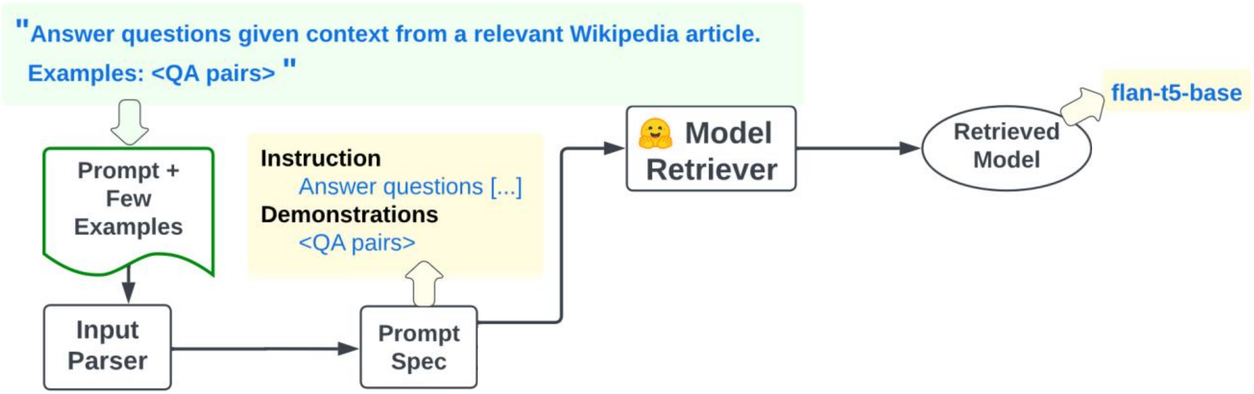
- Step 1: Parse the input, identifying the instruction and each optional example. (using `gpt-3.5-turbo-0613` and few-shot prompting)



[Viswanathan and Zhao et al., 2023]

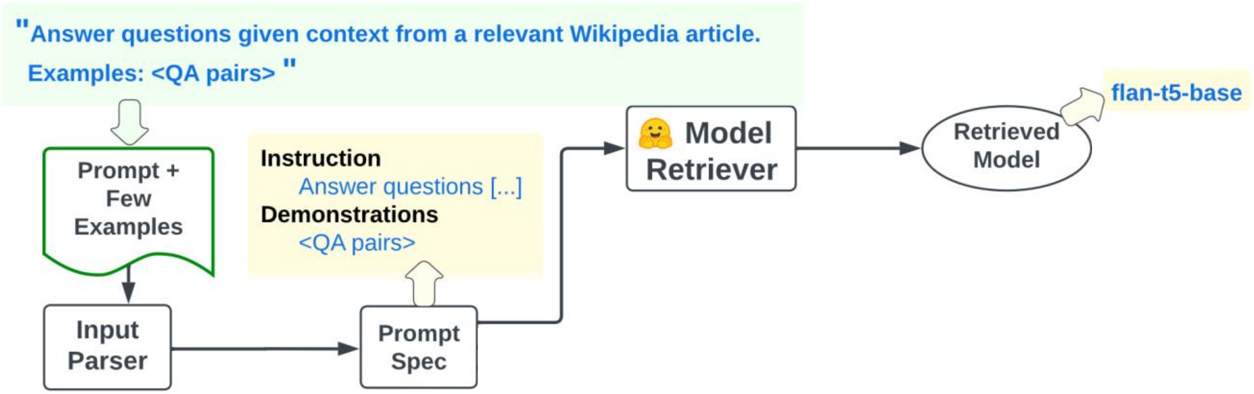
PROMPT2MODEL

- Step 2: Select the model.
  - `gpt-3.5-turbo` generates a “hypothetical model description” from the user instruction.



PROMPT2MODEL

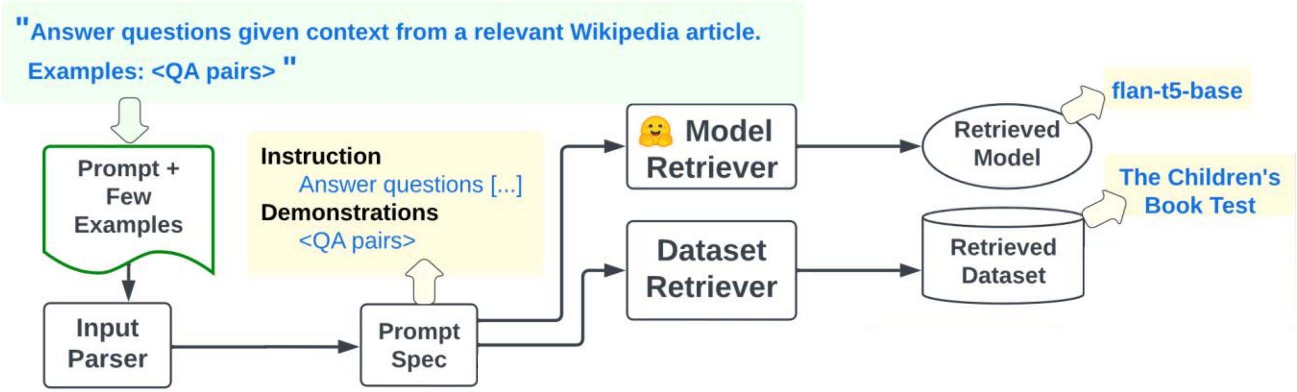
- Step 2: Select the model.
  - Compare this model description with the descriptions of models on HuggingFace, and select the model with highest similarity (BM25 metric)



[Viswanathan and Zhao et al., 2023]

PROMPT2MODEL

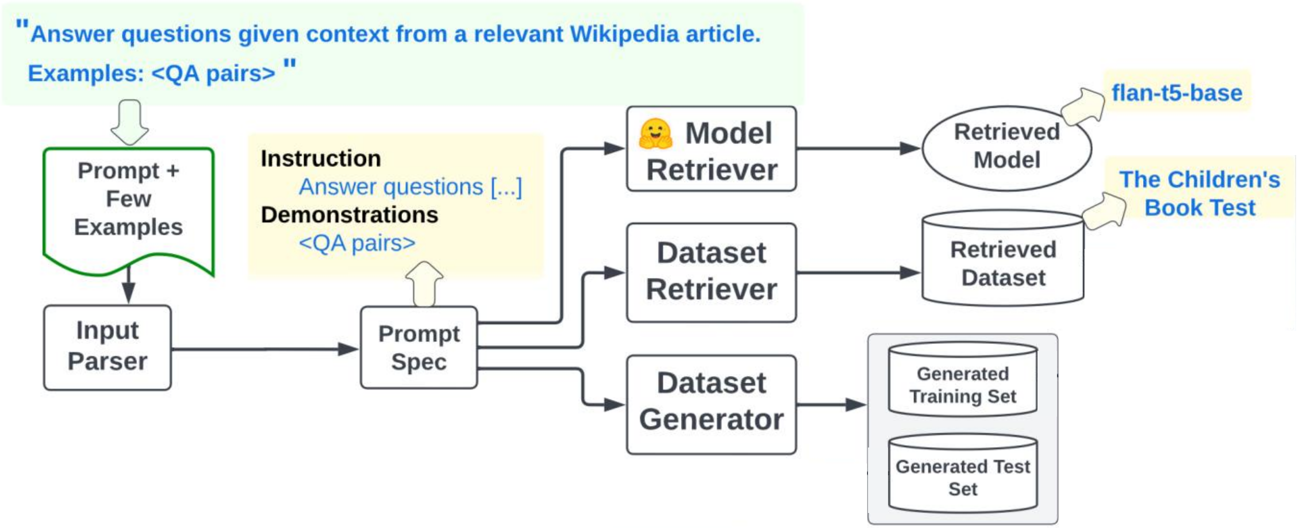
- Step 3: Select the dataset.
  - Rank datasets in HuggingFace according to relevance to the user instruction + optional examples. The top-25 datasets are returned.



[Viswanathan and Zhao et al., 2023]

PROMPT2MODEL

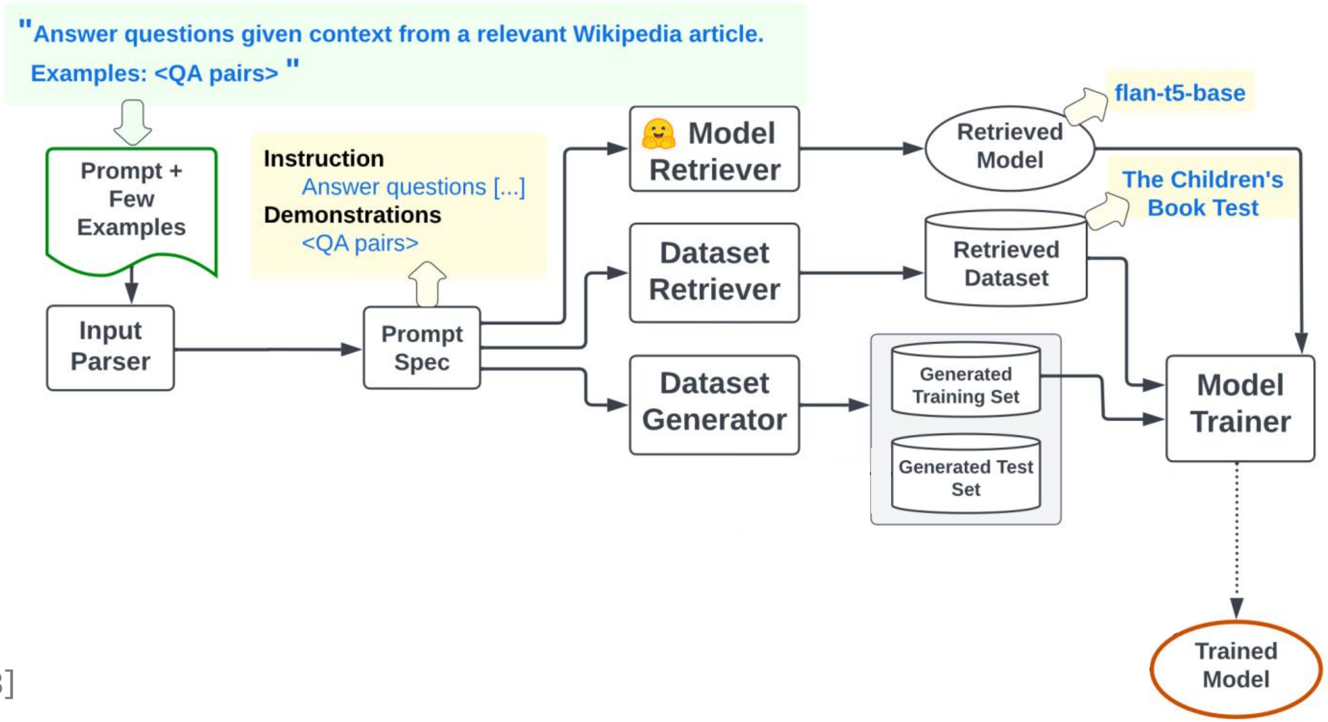
- Step 4: Generate new examples.
  - Use teacher model to generate new examples from the user instruction + optional examples.



[Viswanathan and Zhao et al., 2023]

PROMPT2MODEL

- Step 5: Fine-tune the student model on the retrieved and generated data.



[Viswanathan and Zhao et al., 2023]

PROMPT2MODEL

- Viswanathan and Zhao et al. (2023) evaluated their method on 3 datasets:
  - SQuAD (span retrieval),
  - MCoNaLa (Japanese natural language to code),
  - Temporal (temporal information normalization)
    - E.g., “October 23rd, 1999” -> “1999-10-23”.

Method	SQuAD (EM)	MCoNaLa (ChrF++)	Temporal (ChrF++)
Prompt2Model	61.5	13.1	55.2
w/o Model Ret.	61.5	15.8	55.2
w/o Data Ret.	50.2	16.6	N/A
gpt-3.5-turbo	42.1	37.3	30.7

[Viswanathan and Zhao et al., 2023]

PROMPT2MODEL

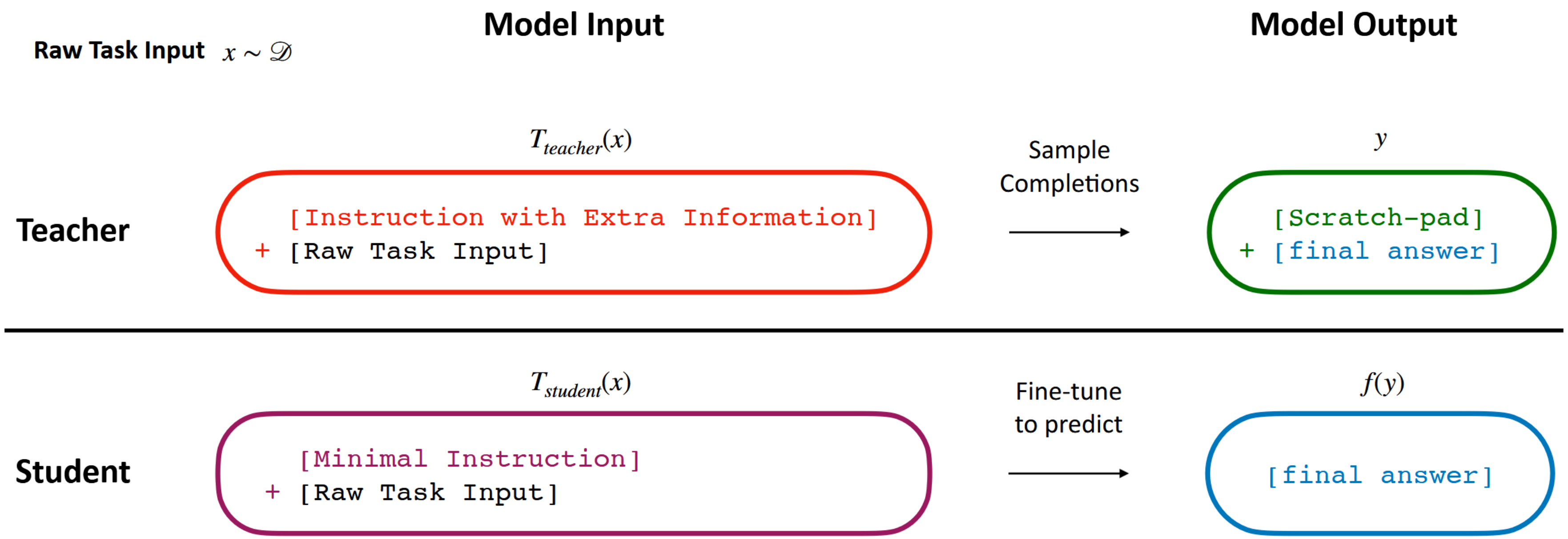
- They found that for SQuAD and Temporal, the distilled model outperformed the teacher model (`gpt-3.5-turbo`).
  - The retrieved models for these two tasks was `flan-T5`, which is 700 times smaller than `gpt-3.5-turbo`.
  - But not so for MCoNaLa.

Method	**SQuAD** (EM)	**MCoNaLa** (ChrF++)	**Temporal** (ChrF++)
Prompt2Model	61.5	13.1	55.2
w/o Model Ret.	61.5	15.8	55.2
w/o Data Ret.	50.2	16.6	N/A
`gpt-3.5-turbo`	42.1	37.3	30.7

[Viswanathan and Zhao et al., 2023]

CONTEXT DISTILLATION

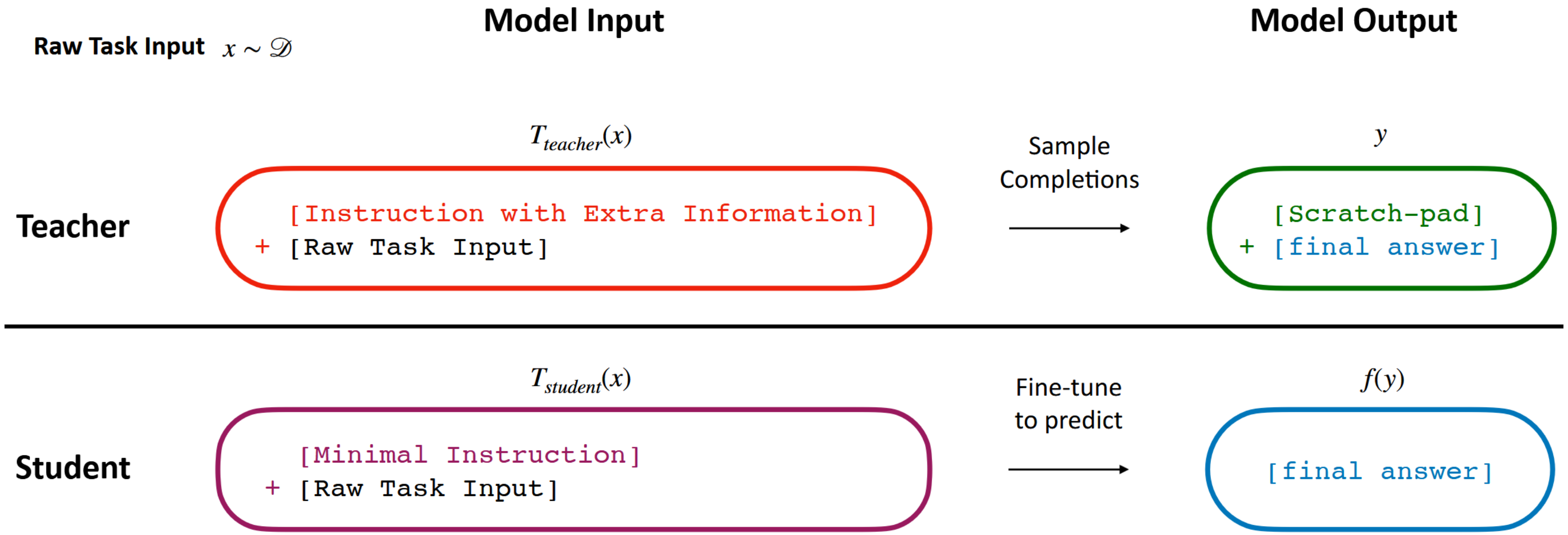
- For many tasks, language models benefit from long detailed instructions, in-context examples, and chain-of-thought.
- Can we teach a model "internalize" this extra information,
  - So that it can perform as well without it?



[Snell et al., 2022]

CONTEXT DISTILLATION

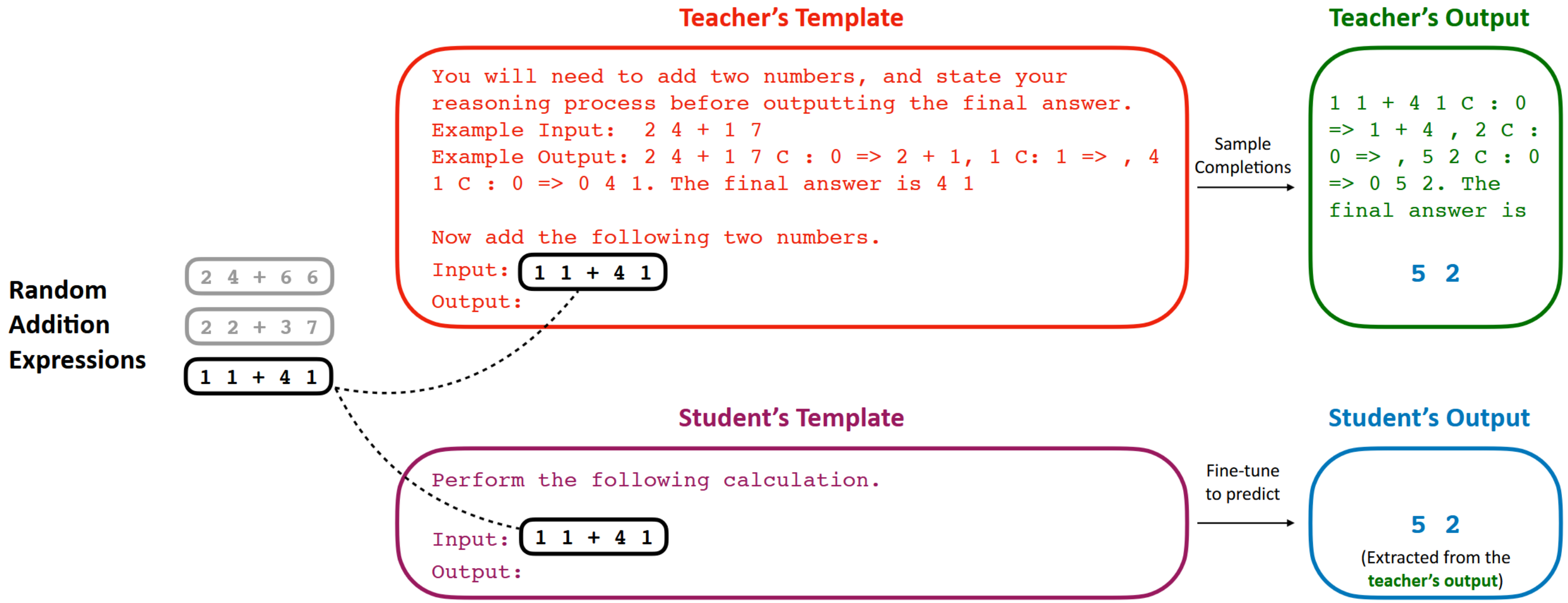
- Snell et al. (2022) call this approach context distillation.
- A teacher model is prompted with extra information (such as few-shot examples, CoT, additional instructions) to produce an output.
- The student model is trained on this output without the extra information.



[Snell et al., 2022]

CONTEXT DISTILLATION

- An example where the student is taught to internalize scratchpad (analogous to CoT):
- By removing the CoT, we are effectively augmenting the teacher model to be able to compute the answer without CoT.



[Snell et al., 2022]

CONTEXT DISTILLATION

- Snell et al. (2022) avoid using soft targets for distillation, since the vocabulary of LLMs is very large (50k-100k).
- Instead, they approximate soft target training by empirically sampling 100 tokens from each logit vector.
  - Then each training example for the student model consists of an "approximate" soft target

    (i.e., the histogram of tokens from the 100 token examples).

CONTEXT DISTILLATION

- In one of their experiments, they use `Incoder-6.7B` fine-tuned on text-to-SQL code generation as the teacher model.
  - The student model is the same as the teacher, except without in-context examples.
- This approach could be used to distill models with large context sizes into models with smaller sizes.

Model	4 Examples	8 Examples
Teacher	27.7	28.2
Pre-distill Student	0.3	0.3
Post-distill Student	**22.1**	**27.9**
Direct Gradient Descent	13.4	18.9

[Snell et al., 2022]

CONTEXT DISTILLATION

- In another experiment, they use `T5-small` (60M parameters) as the teacher model, which has been fine-tuned on the addition task with scratchpad.
  - The student model is the same model without scratchpad.

	Teach	Pre-Dist	Post-Dist
8 Digit Addition Accuracy %	93	0	**95**

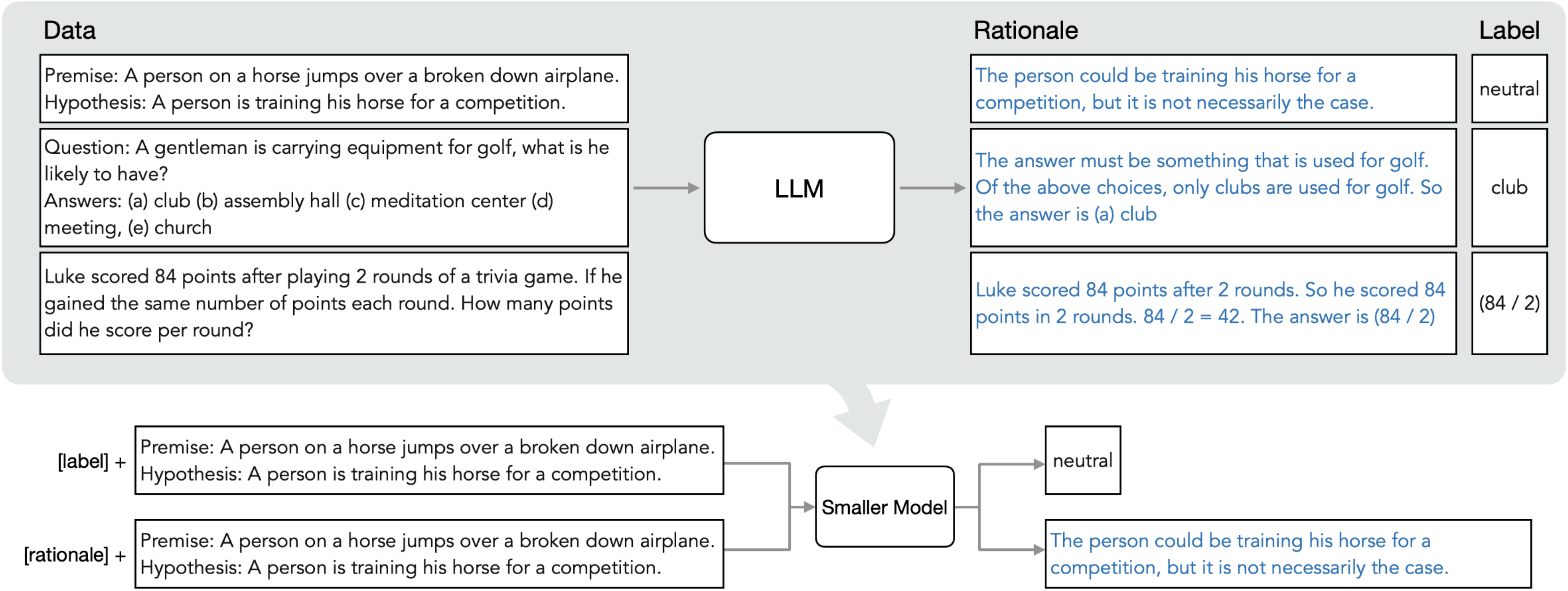
- They did not experiment with larger models, or test whether their approach could be used in a non-task-specific setting.
- This is an example application of distillation where the goal is not model compression.

[Snell et al., 2022]

DISTILL STEP-BY-STEP

- Hsieh et al. (2023) proposed a similar approach which they called "distilling step-by-step."
- In contrast with Snell et al. (2022), they used a much larger teacher model (`PaLM-540B`; Chaudury et al., 2022) to train a small student model (`T5-770M`; Raffel et al., 2020).
- They use the teacher model to generate for each input example:
  - A CoT rationale, as well as the output label.
- Then they train the student model in a *multi-task setting*:
  - If the input example has the word "`[label]`" prepended to it, the model is trained to predict the output label.
  - If the input example has the word "`[rationale]`" prepended to it, the model is trained to predict the rationale.

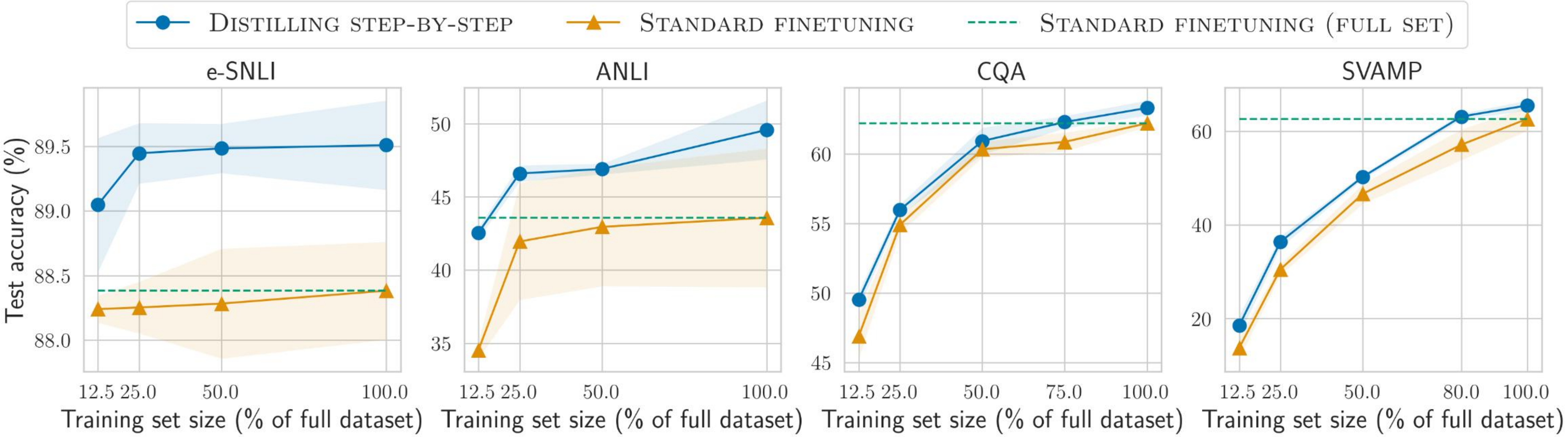
DISTILL STEP-BY-STEP



[Hsieh et al., 2023]

DISTILL STEP-BY-STEP

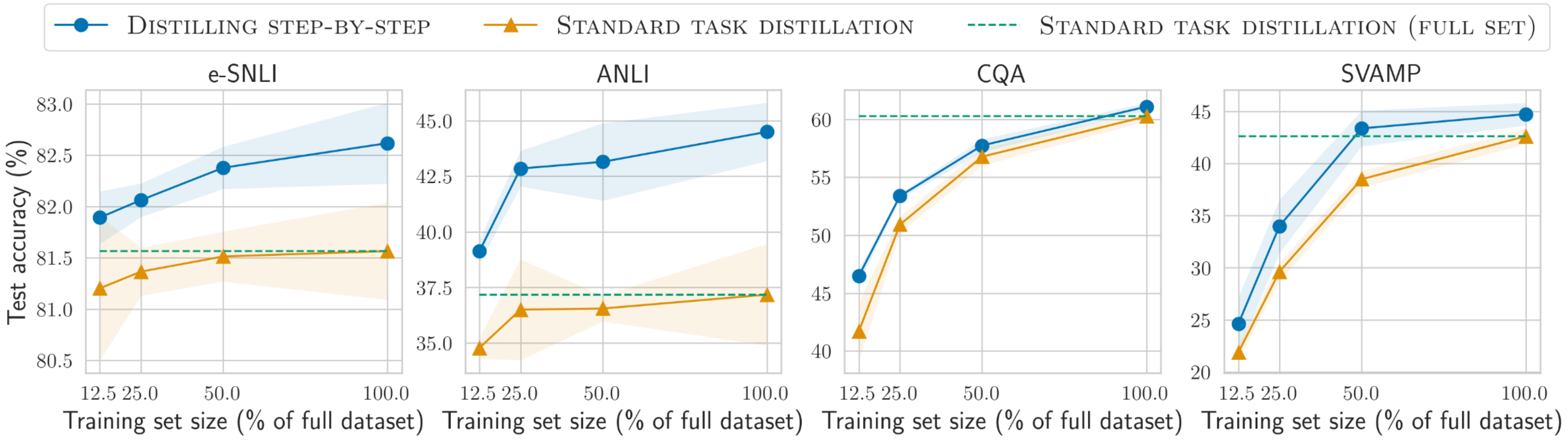
- They trained each student model using 12.5% of a task-specific dataset.
- They compared against full fine-tuning on the same amount of data.



[Hsieh et al., 2023]

DISTILL STEP-BY-STEP

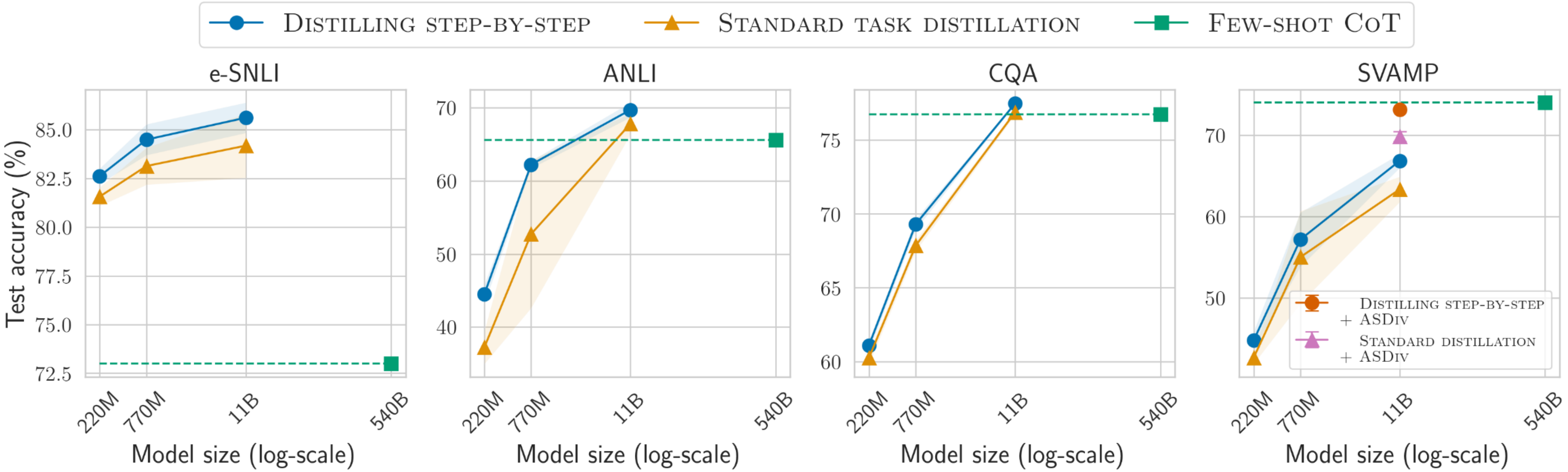
- They also compared against standard knowledge distillation (i.e., without rationales).



[Hsieh et al., 2023]

DISTILL STEP-BY-STEP

- They also compared with the teacher model using few-shot CoT prompting,
  - While varying the size of the student model.
  - (the teacher model was not fine-tuned on any of these datasets)



[Hsieh et al., 2023]

MODEL COMPRESSION SUMMARY

- We have concluded our discussion of model compression, including the three high-level approaches: quantization, pruning, and distillation.
- All three are able to produce smaller models that require less memory and computation time.
- Model compression do come at a cost to accuracy,
  - And further research is needed to better study their differences in behavior (as compared to larger models).
  - E.g., out-of-distribution performance, hallucinations, alignment, etc.

QUESTIONS?