Meta-Training Methods in Natural Language Processing

Tianjian Li

Johns Hopkins University

July 2, 2022

Abstract

Meta learning(i.e. learning to learn) is a learning paradigm where models are not directly trained on a specific task but rather trained on instructions on how to perform these tasks. There have been an emergent trend of applying meta learning methods to Natural Language Processing. This note will cover some of the canonical meta learning methods including **zero-shot** methods such as T0[11], where a pretrained T5[7] model is fine-tuned on various tasks; FLAN[12] where a very large pretrained language model(137B parameters) is fine-tuned on various tasks with instructions; MetaICL[9], where a pretrained language model is finetuned to perform in-context learning. This note will also cover **few shot** meta learning methods and intermediate fine-tuning such as STILT[2] and Multi-Task Fine-Tuning[8],[13] which co-finetunes a low resource task with additional self-supervised objectives or rich data tasks.

1 Zero-Shot

1.1 T0

T0[11] hypothesizes that the reason why large language model have reasonable zero-shot performance is that they are training implicitly under a **multitask** setting. To verify their hypothesis, the authors of T0 conducted a set of experiments by explicitly multitask training a pretrained language model. The authors mainly studies two questions

1. Can multitask training improve performance on unseen tasks?

   The answer is yes. T0[12] outperforms T5+LM[7] on unseen tasks significantly. Moreover, T0 outperforms GPT-3[5] on 9/11 unseen tasks. The author also finds out that scaling the number of tasks during such a intermediate multitask fine-tuning phase leads to better performance.

2. Can we vary the choice of prompts to get a a model that is robust to prompt wording?

   The answer is yes. Increasing the number of prompts does yield better performance and the variance is generally lower(more robust) if we increase the number of prompts. Note that the variance is sometimes the lowest when using no prompts. The author concludes by saying that "Adding more datasets consistently leads to higher median performance but does not always reduce interquartile range for held-out tasks"

1.2 FLAN: Instruction Tuning

FLAN stands for Finetuned LAnguage Net. FLAN[12] is very similar to T0[11], in that they both pre-finetune a large pretrained langauge model on a collection of tasks. On a higher level intuition, this work(FLAN) is actually telling the model to understand the "instructions", so that given a new task, we can leverage that the model understands instructions to give it new instructions for the new task to achieve better performance. Compared with T0, FLAN uses a much larger model(137B vs 11B), and uses a decoder-only Transformer

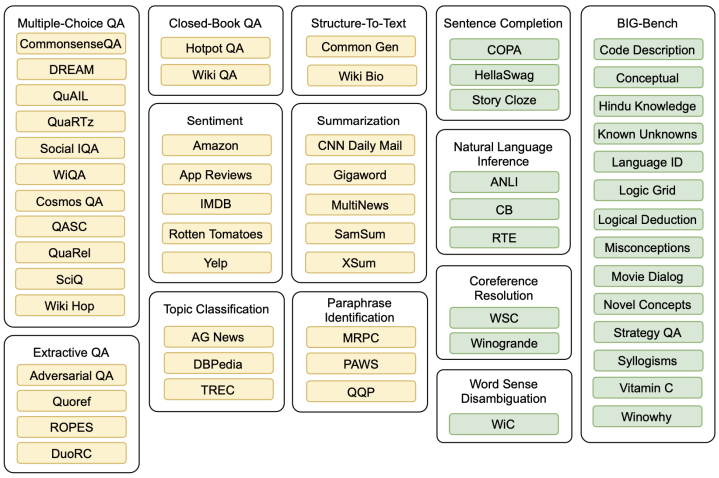

Figure 1: detailed datasets of T0, figure pasted from [12]

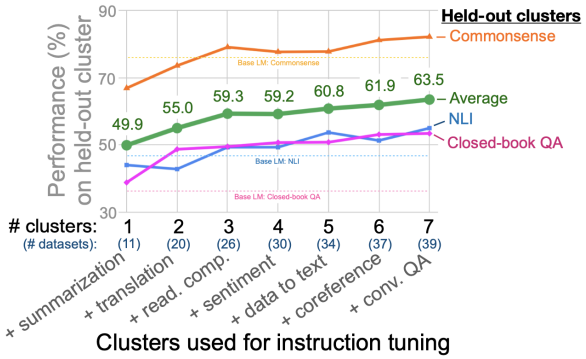

Figure 2: detailed datasets of T0, figure pasted from [12]

model as the base model as opposed to the T5 model with adopts a encoder-decoder Transformer as their base model.

The experimental results of FLAN echoes with the results of T0. Notably the number of tasks and the size of the model contribute most to the zero-shot performance. From the following figure we can see that adding more datasets consistently improve the zero-shot performance on held-out tasks.

Some major drawbacks of FLAN multitasking is that it is not useful in everytask, especially when the held-out task is similar to the mask language modeling objective used during pretraining. From figure 3 we can see that FLAN instruction tuning only works when the model size is large than 8B. The authors extrapolate that "potentially all model capacity is used to learn the mixture of instruction tuning tasks" (instead of learning the semantic meaning of instructions).

1.3 MetaICL

In Context Learning(ICL) have been popularized since the release of the famous GPT-3[5] model, where a pretrained language model(usually very large, GPT-3 uses 175 billion parameters) is given a few examples, concatenated with the actual input at the end to generate an answer. The model learns structured and semantic knowledge of the task from the given examples. MetaICL[9] aims to let the pretrained language model(in this case, a GPT-2[4] model with 770M parameters) learn how to perform ICL through feeding it

	Meta-training	Inference
Task	*C meta-training* tasks	An unseen *target* task
Data given	Training examples $\mathcal{T}_i = \{(x_j^i, y_j^i)\}_{j=1}^{N_i}$, $\forall i \in [1, C]$ $(N_i \gg k)$	Training examples $(x_1, y_1), \cdots, (x_k, y_k)$, Test input x
Objective	For each iteration, 1. Sample task $i \in [1, C]$ 2. Sample $k+1$ examples from $\mathcal{T}_i$: $(x_1, y_1), \cdots, (x_{k+1}, y_{k+1})$ 3. Maximize $P(y_{k+1}\|x_1, y_1, \cdots, x_k, y_k, x_{k+1})$	$\text{argmax}_{c \in \mathcal{C}} P(c\|x_1, y_1, \cdots, x_k, y_k, x)$

Figure 3: overview of metaICL, figure pasted from [9]

ICL examples of different tasks.

One major edge of MetaICL over other zero-shot frameworks is that MetaICL does not rely on task reformatting, which results in high variance in template engineering[10] but rather use all datasets in their as-is condition. The major results are:

1. MetaICL outperforms GPT-3[5], T0[11] and FLAN[12] under zero-shot settings.
2. The gains over zero-shot transfer are particularly significant with meta-training tasks and target tasks are **dissimilar**.

Experimental results show that:

- MetaICL significantly outperforms baselines in most settings, it only marginally outperforms Multi-task 0-shot in the QA→QA setting, as an exception. This is likely because the meta-training and target tasks are relatively similar
- Gains over Multi-task 0-shot are more significant on target tasks in **unseen domains**.
- Fine-tuning with meta-train yields the best results with the only exception with the downstream task is QA(both non-QA to QA and QA to QA).
- Number of tasks and task diversity does a lot of the heavy lifting, which echos the finding of T0 and FLAN.

Again I would like to reiterate that all of the downstream tasks falls into the NLI category. Further investigations are required to prove that meta-training method works for **zero-shot natural language generation**.

2 Few-Shot

2.1 Intermediate Fine-Tuning

2.1.1 STILT

STILT stands for Supplementary Training on Intermediate Labeled data Tasks. It aims to add an intermediate phase between the standard pre-train and fine-tune paradigm. Experiments are usually conducted in the following form: First a language model is pre-trained with an unsupervised objective like masked language modeling[3]; Then the model is further trained(or intermediate trained) on resourceful labelled data; Finally, the model is fine-tuned further on the target task and evaluated.

The author evaluated this new intermediate-training paradigm on BERT[3], GPT[4] and a variant of ELMo[1] and evaluated on the GLUE benchmark. Since GPT and ELMo are a little bit outdated, I will only show the results of BERT with STILT in figure 4 and 5, representing the validation set adn test set, respectively. We can see that largest gains are usually on the MNLI dataset. However, even after manually selecting the best intermediate task according to the dev set still only results in better performance in 4 tasks out of 9 total downstream tasks. Either indicating that this method either is not effective on smaller are more naive

Training Set Size	Avg	A.Ex	CoLA *8.5k*	SST *67k*	MRPC *3.7k*	QQP *364k*	STS *7k*	MNLI *393k*	QNLI *108k*	RTE *2.5k*	WNLI *634*
Development Set Scores											
BERT	80.8	78.4	**62.1**	92.5	89.0/92.3	**91.5/88.5**	90.3/90.1	**86.2**	89.4	70.0	56.3
BERT→QQP	80.9	78.5	56.8	93.1	88.7/92.0	~~91.5/88.5~~	90.9/90.7	86.1	89.5	74.7	56.3
BERT→MNLI	82.4	80.5	59.8	**93.2**	**89.5/92.3**	91.4/88.4	**91.0/90.8**	~~86.2~~	**90.5**	**83.4**	56.3
BERT→SNLI	81.4	79.2	57.0	92.7	88.5/91.7	91.4/88.4	90.7/90.6	86.1	89.8	80.1	56.3
BERT→Real/Fake	77.4	74.3	52.4	92.1	82.8/88.5	90.8/87.5	88.7/88.6	84.5	88.0	59.6	56.3
BERT, Best of Each	**82.6**	**80.8**	**62.1**	**93.2**	**89.5/92.3**	**91.5/88.5**	**91.0/90.8**	**86.2**	**90.5**	**83.4**	56.3

Figure 4: Results of STILT on validation set, figure pasted from [2]

Test Set Scores											
BERT	80.4	79.4	60.5	**94.9**	85.4/89.3	**89.3/72.1**	87.6/86.5	**86.3**	**91.1**	70.1	65.1
BERT on STILTs	**81.8**	**81.4**	**62.1**	94.3	**89.8/86.7**	89.4/71.9	**88.7/88.3**	86.0	**91.1**	**80.1**	65.1

Figure 5: Results of STILT, figure pasted from [2]

models, or only works on specific datasets. However, this method works better when target data is limited, rendering intermediate training more effective as a data augmentation technique rather than a general improvement of parameter initialization.

The author also conducted experiments of different multitask learning techniques and find out that sequential learning yields the best result when the model backbone is GPT. The authors conclude that "naive multitask learning appears to yield worse performance than STILTs, at potentially greater computational cost."

2.1.2 Multilingual STILT

The authors also conducted STILT under a multilingual transfer learning setting[6], where the model is intermediate-trained and fine-tuned on English data only, but evaluated on different languages. The authors state that English STILT yields "Moderate improvements on question-answering target tasks. MNLI, SQuAD and HellaSwag achieve the best overall results as intermediate tasks, while multi-task intermediate offers small additional improvements."

Specifically speaking, SQuADv2(with MLM) yields largest gain(+1.9/+2.1) on PAWS-X. Multitask without MLM yields largest gain on NER. SQuADv1.1 yields largest gain on XQuAD and MLQA, note that here SQuADv1.1 serves as both the intermediate task and training set of XQuAD and MLQA. The authors hypothesize that the reason why SQuADv1.1 makes such a good intermediate task is due to the baseline XQuAD and MLQA is undertrained.

Two things that did not work out are 1) Mixing the Pre-train multilingual MLM objective, and 2) Translating the intermediate task dataset to other languages.

2.2 Multi-Task Learning

Another line of research is Multi-Task Learning, which jointly trains the desired task and one or a few auxiliary tasks to boost the performance. Usually, MTL is explored for a specific task(e.g. summarization, classification) and few studies have investigated which auxiliary task is useful for which type of tasks. Here I will introduce two papers, one is [13], which studies the pros and cons of MTL opposed to intermediate fine-tuning from a downstream task performance perspective; The other one is [8], which performs MTL for a specific task: abstractive summarization.

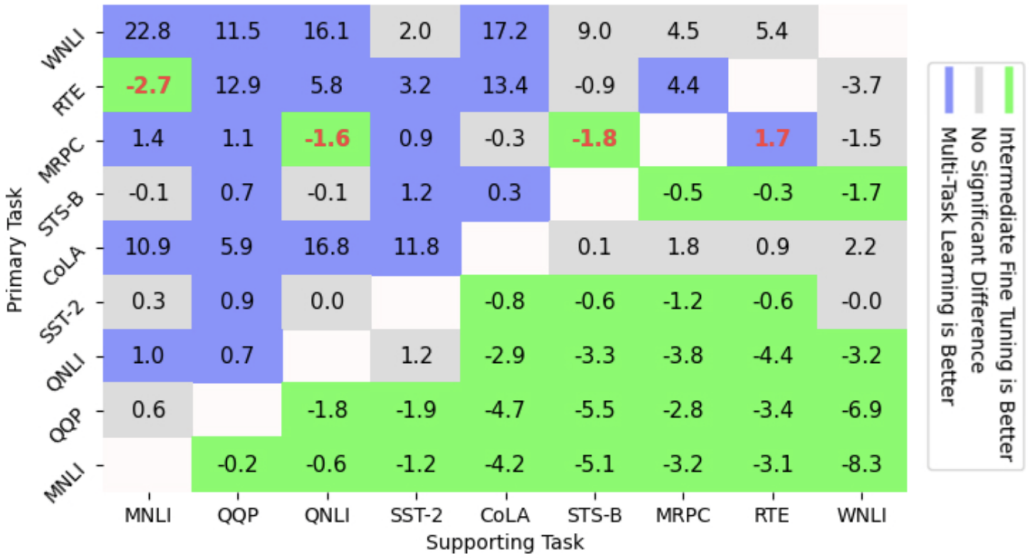

Figure 6: MTL VS Intermediate FT, figure pasted from [13]

Dataset	Citation	Training Size
MNLI	Williams et al. (2018)	392,662
QQP	No citation, link here	363,846
QNLI	Levesque et al. (2011)	104,743
SST-2	Socher et al. (2013)	67,349
CoLA	Warstadt et al. (2018)	8,551
STS-B	Cer et al. (2017)	5,749
MRPC	Dolan and Brockett (2005)	3,668
RTE	Dagan et al. (2006)*	2,490
WNLI	Levesque et al. (2011)	635

Figure 7: Detailed sizes of GLUE benchmark datasets, figure pasted from [13]

2.2.1 Multi-Task Learning vs Intermediate Fine-Tuning

This paper compares three different methods: Intermediate Fine-Tuning, pairwise MTL and full MTL. The backbone model is a pretrained DistilRoBERTa.

Experiment results(figure 6) clearly shows that MTL holds its superiority when the data of the downstream task is insufficient because the tasks are sorted from the smallest(WNLI) to the largest(MNLI). Another findings of the authors is that using full MTL, which jointly trains the target dataset along with every other dataset is usually worse than the average performance of using only intermediate FT and the average performance of using only MTL. Therefore even if we did not carefully select between Intermediate FT and MTL and carefully select which auxiliary task to use, full MTL is still worse than the average performance.

2.2.2 MTL for Abstractive Summarization

There are two different flavors of summarization: extractive and abstractive. The former aims to extract key spans and sentences from a document to reconstruct a summary. The latter aims to generate a summary from scratch while keeping the semantic meaning of the original document. Therefore abstractive summarization can be decomposed into extractive summarization and paraphrasing. This paper[8] leverages this and and jointly trains an abstractive summarization objective along with extractive summarization, para-

Tasks	R1	R2	RL
Single Task (A)	36.08	10.94	31.57
A E	29.99	8.80	24.80
A C	35.46	10.76	30.81
A P	**36.75**	**12.13**	**32.30**
A C P	36.28	11.59	31.58
A E C	29.19	8.69	25.20
ALL	30.31	9.60	27.97

Figure 8: T5 with MTL: C = Concept Labels, P = Paraphrase Detection, E = Extractive Summary, figure pasted from [8]

phrase detection and also language modeling.

To verify their proposed simple-yet-effective MTL framework, the authors experimented on T5[7] and BERT[3]. Here I will only paste the results of T5. Clearly, Paraphrase detection as the auxiliary task yields the largest gains on T5(also on BERT) when jointly trained for a abstractive summarization objective. Moreover, using auxiliary tasks more or less improve upon the naive single objective training when it comes to the realm of summarization, probably because that, as I have stated before, summarization is a composition of extract and paraphrase. However, when training data is scares, MTL does not yield performance gains.

References

[1] Matthew E. Peters et al. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://aclanthology.org/N18-1202.

[2] Jason Phang, Thibault Févry, and Samuel R. Bowman. "Sentence Encoders on STILTs: Supplementary Training on Intermediate Labeled-data Tasks". In: *CoRR* abs/1811.01088 (2018). arXiv: 1811.01088. URL: http://arxiv.org/abs/1811.01088.

[3] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[4] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2019).

[5] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[6] Jason Phang et al. "English Intermediate-Task Training Improves Zero-Shot Cross-Lingual Transfer Too". In: *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*. Suzhou, China: Association for Computational Linguistics, Dec. 2020, pp. 557–575. URL: https://aclanthology.org/2020.aacl-main.56.

[7] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[8] Ahmed Magooda, Diane Litman, and Mohamed Elaraby. "Exploring Multitask Learning for Low-Resource Abstractive Summarization". In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 1652–1661. DOI: 10.18653/v1/2021.findings-emnlp.142. URL: https://aclanthology.org/2021.findings-emnlp.142.

[9] Sewon Min et al. "MetaICL: Learning to Learn In Context". In: *NAACL-HLT*. 2022.

[10] Swaroop Mishra et al. "Reframing Instructional Prompts to GPTk's Language". In: *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 589–612. DOI: 10.18653/v1/2022.findings-acl.50. URL: https://aclanthology.org/2022.findings-acl.50.

[11] Victor Sanh et al. "Multitask Prompted Training Enables Zero-Shot Task Generalization". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=9Vrb9D0WI4.

[12] Jason Wei et al. "Finetuned Language Models are Zero-Shot Learners". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=gEZrGCozdqR.

[13] Orion Weller, Kevin Seppi, and Matt Gardner. "When to Use Multi-Task Learning vs Intermediate Fine-Tuning for Pre-Trained Encoder Transfer Learning". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 272–282. DOI: 10.18653/v1/2022.acl-short.30. URL: https://aclanthology.org/2022.acl-short.30.