Boosting Natural Language Generation from Instructions with Meta-Learning

Budhaditya Deb, Guoqing Zheng, Ahmed Hassan Awadallah
Microsoft Research
{budeb, zheng, hassanam}@microsoft.com

Abstract

Recent work has shown that language models (LMs) trained with multi-task *instructional learning* (MTIL) can solve diverse NLP tasks in zero- and few-shot settings with improved performance compared to prompt tuning. MTIL illustrates that LMs can extract and use information about the task from instructions beyond the surface patterns of the inputs and outputs. This suggests that meta-learning may further enhance the utilization of instructions for effective task transfer. In this paper we investigate whether meta-learning applied to MTIL can further improve generalization to unseen tasks in a zero-shot setting. Specifically, we propose to adapt meta-learning to MTIL in three directions: 1) Model Agnostic Meta Learning (MAML), 2) Hyper-Network (HNet) based adaptation to generate task specific parameters conditioned on instructions, and 3) an approach combining HNet and MAML. Through extensive experiments on the large scale Natural Instructions V2 dataset, we show that our proposed approaches significantly improve over strong baselines in zero-shot settings. In particular, meta-learning improves the effectiveness of instructions and is most impactful when the test tasks are strictly zero-shot (i.e. no similar tasks in the training set) and are "hard" for LMs, illustrating the potential of meta-learning for MTIL for out-of-distribution tasks.

1 Introduction

Given some basic instructions and a few demonstrations, humans are capable of conducting diverse tasks without any supervision. Can language models perform similarly on unseen tasks when trained with instructions? Specifically can such an approach work on complex generation tasks with relatively smaller language models (LMs)?

The recent advances in large LMs have shown tremendous potential in diverse AI applications and have the capability to change the way model developers and users interact with intelligent systems. The inherent representative power of such models has shown that diverse NLP tasks can be solved purely by appending prompts or demonstrations in context before a test input (Radford et al., 2019; Brown et al., 2020). This has led to the rise of prompt-based training (Liu et al., 2021) where even much smaller models trained on a large set of tasks in a multi-task setting with prompts, can behave similarly (Schick and Schütze, 2021).

A natural extension of the prompt tuning concept involves adding instructions about the task along with the demonstrations. Instructions are more informative than prompts and aid the language models to solve unseen tasks better. Instructions can have different forms, for example to convey a short task specific statement (e.g. "Provide a short summary for the following input") (Schick and Schütze, 2021b), or a natural language question ("How would you rephrase that in a few words?") (Sanh et al., 2022; Wei et al., 2022; Bach et al., 2022). However, for complex generation tasks, short instructions can be ambiguous and non-informative and thus need large LMs which can encode a much richer prior knowledge.

In contrast, (Wang et al., 2022) define instructions in the Natural Instructions V2 (NIV2) dataset comprising of detailed task descriptions, positive and negative examples, and explanations. Instructions in NIV2 are similar to annotation guidelines, and thus potentially more beneficial[1]. Using multi-task *instructional learning* (MTIL) on diverse tasks, (Wang et al., 2022) showed that even smaller models can be competitive with larger models on zero-shot generalization to unseen tasks.

Results in (Wang et al., 2022) illustrated that LMs can extract useful information from instructions beyond the surface patterns available in the prompts for solving a task. This suggests that

[1]The instructions in NIV2 are in-fact taken from annotation guidelines for each of the tasks

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 6792–6808
December 7-11, 2022

learning-to-learn or meta-learning paradigm can further enhance the utilization of instructions by learning about task at deeper levels. In this paper, we investigate how smaller LMs could best benefit from the natural instructions and whether meta-learning paradigms can further improve the zero-shot generalization ability of LMs in MTIL. Meta-learning has been shown to be effective in adapting knowledge with little supervision but to the best of our knowledge has not been adapted to MTIL in zero-shot settings.

Specifically, we explore two different meta-learning approaches. First we propose to adapt Model Agnostic Meta Learning (MAML) (Finn et al., 2017) for MTIL, an optimization based approach. Second, we explore hyper-network (HNet) (Ha et al., 2017) based MTIL, a black-box approach. HNet introduces an auxiliary LM which encodes instructions to produce task specific parameters which are added to the main LM parameters to generate a task specific LM at prediction time. In addition, we evaluate a third approach which combines the two into a HNet-MAML by training the HNet model using MAML.

We conduct extensive experiments specifically designed to test the generalization ability of LMs trained with instructions under different zero shot conditions. We use two sets of training tasks from the NIV2 dataset: 1) all natural language tasks and 2) natural language generation tasks. We evaluate the models for two sets of held out **generation tasks** conveying different levels of zero-shot generalization ability: 1) **weak generalization** set with a random selection of generation tasks with potential overlap of categories with training tasks and 2) **strong generalization** set (or strict zero-shot conditions) using summarization and title generation tasks with no overlap in categories from the training tasks. We further investigate the task sets under difficulty levels of *easy*, *medium*, and *hard* based on their baseline ROUGE scores.

The main conclusion from our study is that under strict zero-shot conditions, meta-learning with instructions significantly improves the performance. The improvements become more significant for the strong generalization task set and when the task difficulty level is hard (i.e. tasks where the LM struggles to generate correct outputs in zero-shot setting). Moreover, meta-learning increases the effectiveness of instructions under all conditions. While both MAML and HNet models show improvements over the baselines, HNet (along with its MAML extension) by explicitly enforcing the use of instructions through task specific conditioning of parameters, results in larger gains. In summary, the main contributions of the paper are two-fold. First, we adapt meta-learning approaches to MTIL. Second, we study their efficacy and show significant improvements under strict zero-shot conditions.

2 Related Work

Learning from instructions: An extension of the basic prompt-based in-context learning is appending task specific instructions with prompts. Several recent works which include FLAN (Wei et al., 2022), T0 (Sanh et al., 2022) and (Reif et al., 2021), train a large LM in a multi-task setting with instructions. *InstructGPT* (Ouyang et al., 2022) takes slightly different approach by training the GPT3 model (Brown et al., 2020) with human annotated dataset of demonstrations of desired user intents and use reinforcement learning to improve the model to follow such instructions. Yet another direction called pattern-exploiting training (PET) (Schick and Schütze, 2021a; Schick and Schütze, 2021) combines the idea of formulating instructions as cloze questions and show that even small LMs can be good few-shot learners and work with language generation.

Meta-learning for language generation: Meta learning has been applied in several language generation settings such as (Lin and Lee, 2020) to induce persona in a chatbot, (Mi et al., 2019) for task oriented dialog systems, (Gu et al., 2018) for low resource machine translation, and (Chen and Shuai, 2021) for abstractive summarization in a low-resource transfer learning but do not use instructions for zero-shot transfer. Our MTIL scenario is closely related to MetaICL (Min et al., 2022) which applies multi-task learning in-context in a K-shot setting for classification tasks, but differs in that it is a k-shot in-context scenario and does not use instructions or meta-learning optimization. While these works are related, to the best of our knowledge, meta-learning has not been used to generalize to unseen generation tasks in zero shot settings using instructions and thus the paper provides several novel insights and approaches.

Hyper-Networks (HNet) in NLP applications: (Karimi Mahabadi et al., 2021) use HNet to train LMs in a multi-task setting with adapters and (von Oswald et al., 2020) propose a contin-

ual learning framework with HNets conditioned on unique task IDs to reduce catastrophic forgetting. HNets have been used for input conditioning a decoder in (Ivison and Peters, 2022) which produces a unique decoder for each input, and thus is similar to our approach. However these the approaches are not strictly applicable in our zero-shot scenario or in general NLP tasks with task descriptions in natural language.

Language model editing: Our HNet based approach is based on the architecture in (Cao et al., 2021) which uses it to edit factual knowledge in LMs. While the architecture is similar, we use the HNet to encode task specific instructions and is intended for controlling task-level LM behavior unlike the micro-behavior targeted in (Cao et al., 2021). Similar to ours and (Cao et al., 2021), Bayesian hyper networks (Krueger et al., 2018) linearizes the number of parameters for predictions by constraining the HNet outputs to scale and shift parameters. (Sinitsin et al., 2020; Mitchell et al., 2022) propose Meta Learning approaches for editing errors in a neural network but is not directly applicable for MTIL in a zero-shot setting.

MTIL: Finally, the work most closely related to this paper is the T*k*-Instruct model from (Wang et al., 2022) which fine tunes a T5 model (Raffel et al., 2020) with instructions, which we use as the baseline. We use the same dataset and training settings as T*k*-Instruct but instead use the pretrained BART model (Lewis et al., 2020) as it is task agnostic compared to T5 (T5 may not represent a true zero-shot setting). In addition, we enhance this model with meta-learning and consider significantly different training, evaluation, and model settings to test zero-shot generalization resulting in unique contributions and conclusions orthogonal to the findings in (Wang et al., 2022).

3 Problem Setup

In this section we briefly outline our problem settings and baselines used in this paper.

3.1 Natural Instructions V2 Dataset

We use the Natural Instructions V2 (NIV2) dataset (Wang et al., 2022)[2] to investigate meta-learning approaches for instructional learning. The NIV2 is a meta-dataset with over 1600 tasks.

In NIV2, each task contains instructions and multiple training instances with input and output. The instructions consist of: 1) **Categories** (classification, summarization etc.), 2) **Short description** (a short sentence about the task), 3) **Long description** (a detailed description of the task similar to annotation guidelines), 4) **Positive examples** (inputs with correct outputs), 5) **Negative examples** (inputs with *incorrect* outputs), and 6) **Explanations** for the positive or negative examples.

(Wang et al., 2022) train a pretrained T5 language model (Raffel et al., 2020) on input-output pairs with instructions (T*k*-Instruct) appended before the input in a multi-task setting. During testing, held out unseen tasks are predicted by appending similar instructions to the test input. (Wang et al., 2022) provide detailed ablations and baseline comparisons with related models showing the impact of instructions. Following the results there, we only use the task descriptions and positive examples in this study as negative examples and explanations were not shown to have any positive contributions.

3.2 Baseline Model with Standard Training

Based on results in (Wang et al., 2022) where T*k*-Instruct was shown to comfortably beat much larger T5, GPT3, InstructGPT3, and T0 models, we use the T*k*-Instruct setting as our baseline, i.e. we train a pre-trained encoder-decoder LM on multiple tasks with instructions. We also explored appending the instructions before the decoder sequence but did not find any improvements. However, we did observe that by pre-pending a special prefix to the decoder (we use "[Output]:") improves the overall prediction performance. We refer to this model as the **standard training** model.

For our base LM, we use the pretrained BART model (Lewis et al., 2020) as it is task agnostic compared to T5[3] and thus represents a stronger zero-shot setting. Interested readers should refer to the (Wang et al., 2022) paper for detailed ablations specific to the NIV2 dataset and the T5 model.

3.3 Evaluation Settings

We focus specifically on the zero-shot generalization on generation tasks. While the general settings remain similar to (Wang et al., 2022) we consider some specific settings to illustrate the generalization capabilities of models to different tasks.

For training, we use two sets of tasks 1) All EN tasks in the NIV2 dataset and 2) Generation tasks.

[2]https://instructions.apps.allenai.org/

[3]Publicly available T5 models are pre-trained on a multi-task mixture of unsupervised and supervised tasks.

For evaluation, we consider two sets of generation tasks with different zero-shot levels : 1) weak generalization set using a random set of generation tasks with potential similarity to the training tasks and 2) strong generalization set using tasks from summarization and title generation categories with no overlap with the training tasks. The list of evaluation tasks with short descriptions are provided in the appendix in Figures 11 and 12.

We further divide the evaluation tasks into difficulty levels of "easy", "medium" and "hard" based on the ROUGE scores from the baseline model (low scores indicate out-of distribution and difficult tasks) to see to what extent meta-learning helps in improving performance of the out-of-distribution tasks.

4 Meta-Learning with Instructions

Training on a large number of diverse tasks and testing on unseen tasks lend itself to the paradigm of learning-to-learn or meta-learning, which has been successfully applied for generalizing in both zero- and few- shot scenarios. Task meta-data in the form of instructions can also provide discriminative information about the task process in addition to the surface patterns of the input and output strings. We investigate whether meta-learning can aid such learning and adapt three approaches to MTIL.

4.1 Standard Training + MAML

We adapt Model Agnostic Meta Learning (MAML) (Finn et al., 2017) to instructional learning of LMs as a way to generalize to unseen tasks by training on large number of diverse tasks.

The standard training with MAML is described in Algorithm 1 in the appendix. At any training iteration, we sample two different sets of k tasks for MAML meta-train and meta-test steps. We uniformly sample across tasks to maximize the diversity of tasks in each batch. The data format is same as the standard training. Since we test zero-shot conditions, we do not have any test time optimization typically employed in MAML.

4.2 Standard Training + HNET

Both standard and MAML training do not explicitly enforce the use of instructions during decoding. The model can thus minimize the loss simply by ignoring the instruction part of the encoder by attending to the input and output texts. This can lead to sub-optimal use of the instructions.

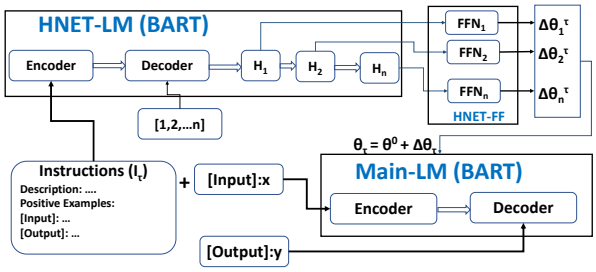

Figure 1: Encoding instructions using a Hyper-network

We propose a hyper-network (HNet) based architecture (Ha et al., 2017) to produce task specific model parameters conditioned on the instructions. The HNet architecture consists of an auxiliary LM (HNet-LM) along with the Main-LM. The HNet-LM produces task specific parameters of the main LM by conditioning on instructions.

In particular, we adapt a specific type of HNet architecture from (Cao et al., 2021) which predicts the *delta-parameters* of the Main-LM, which are then added to the Main-LM parameters to produce the task specific LM. This preserves the parameters in the Main-LM utilizing the shared generation capability of the LM while specializing in task specific behavior. However, there are some specific differences based on our requirements for instructional training, which are described next.

4.2.1 The HNet Language Model (HNet-LM)

Since the input to the HNet model is text, we use a pretrained encoder-decoder LM (BART in this paper) to encode the instructions[4] and use the decoder's hidden states for conditioning the layer specific parameters of the Main-LM.

In (Cao et al., 2021) the last hidden state from an LSTM is used for conditioning the parameters of the main model. To increase the effective bandwidth of the HNet while keeping the number of parameters same, we use the last N hidden states (for N layers of the main LM). This simple trick allows the model to independently attend to and condition each layer on the input instructions while still keeping the model parameters the same.

The HNet-LM takes instruction I_τ for a task τ and sequence of decoder indexes d_n as input and produces N hidden states $h_\tau(n)$. The decoder index sequence we use is simply $1, ..., n$. Decoder indexes provide different inputs to the decoder to influence the generation of distinct parameters for each layer of the main LM. This is not strictly re-

[4] In contrast (Cao et al., 2021) used an untrained LSTM.

quired as the position embeddings can in principle drive the input/output of the HNet to produce different parameters for each target layer. However, we found that adding different decoder indexes improves the performance as it provides additional differentiation to the decoder.

The HNet-LM can produce the parameters of all or a subset of layers of the Main-LM. We experimented with three settings: encoder, decoder, and both. The best metrics are obtained when the HNet is used to generate parameters of the decoder of the Main-LM, which is what we use in reporting the results.

4.2.2 HNet-FF Projections

Next, the output hidden states from the HNet-LM decoder are projected to the Main-LM parameter space using the HNet Feed Forward (HNet-FF) layer consisting of one dense layer with $Tanh$ activation. Let ψ denote the parameters of the HNet-LM, π_n the parameters of the N HNet-FF layers, and d_n an input index to the decoder.

The $h_\tau(n)$ is then projected using the FF network to five parameter vectors $\alpha_n, \beta_n, \gamma_n, \delta_n, \eta_n$ as given in Equation 2. Finally the delta parameters $\Delta\theta_\tau$ of the main LM layers are generated using equation 3. Here σ denotes the sigmoid function and σ' is the softmax. The projection process is equivalent to (Cao et al., 2021) but slightly differs in that we do not use the gradients of the Main-LM parameters as we target a zero-shot scenario and during test time, the target labels are not available. The model is illustrated in Figure 1 and Algorithm 2 in the appendix.

$$h_\tau(n) = HNetLM(\psi; I_\tau, d_n) \quad (1)$$

$$\alpha_n, \beta_n, \gamma_n, \delta_n, \eta_n \leftarrow HNetFF_n(\pi_n; h_\tau(n)) \quad (2)$$

$$\Delta\theta_\tau = \sigma(\eta) \cdot (\sigma'(\alpha)\gamma^T + \sigma'(\beta)\delta^T) \quad (3)$$

$$\theta_\tau = \Theta_0 + \Delta\theta_\tau \quad (4)$$

4.2.3 Alternating Training Schedule

The HNet-LM along with the Main-LM can be jointly trained end-to-end. We tested with different configurations such as partially or fully freezing the Main-LM, but best metrics were achieved with the Main-LM fully trained along with the HNet. However, with this comes convergence issues and significantly increased training cost. First, the training can be unstable when both the HNet and Main-LMs are updated, requiring low learning rates. Second, joint training requires twice as much memory and computation.

We address both issues by using an alternating training schedule for the HNet and Main LMs, where we freeze one of the LMs for a few steps while updating the other and vice-versa. Thus, in the backward step only one of the LM's parameters are updated, which leads to lower memory requirements, stable loss convergence, and better metrics. Moreover, during the HNet training phase (i.e., when the Main-LM is fixed), the model is forced to update the HNet parameters using the instructions as input thus allowing itself to learn instruction-specific patterns. The HNet based model training is described in Algorithm 2 in the appendix.

4.3 Standard Training + HNet + MAML

Since the HNet model comprising of the HNet-LM, HNet-FF, and Main-LM is end-to-end differentiable, we can employ MAML training loop for this combined network. Here, inside the HNet loop, we employ the alternate training schedules in the MAML inner loop, while the outer training loop remains the same. The HNet-MAML training is described in Algorithm 3 in the appendix.

5 Experiments

5.1 Experimental Setup

We use the NIV2 dataset in all our experiments in a multi-task setting similar to prior work (Wang et al., 2022) for training LMs to interpret instructions. I.e. we train on a large number of tasks with instructions and analyze the zero-shot generalization capability on held-out evaluation tasks.

We first split the instances in NIV2 dataset for each each task into training, validation, and evaluation sets consisting of 80%, 10%, and 10% split respectively. We filter out any non-English tasks for this study, which leads to around 1000 tasks. We consider two sets of training and evaluation tasks which are intended to better understand the generalization capability of LM with instructions. We first create two tasks sets as follows:

1) All-tasks set which has all EN tasks from NIV2 dataset consisting of around 1000 tasks.

2) Generation-tasks set which consists of language generation tasks by filtering out task categories such as classification, sequence tagging, multiple choice QA, etc. This consists of around 400 tasks.

Next we randomly split (90%/10%) the Generation task set into training and evaluation tasks. Additionally the tasks from the held out strong gen-

eralization set (described next) are also removed from the training sets. After filtering, we have 916 tasks in the All-tasks training set and 306 tasks in the Generation-task training set.

For evaluation, we consider two language generation task sets which differ in the level of dissimilarity from the training tasks and thus test different levels of generalization ability of the trained LMs.

Weak generalization evaluation set (81 tasks): This set (also used as a validation set) comprises of 10% of tasks randomly split from the Generation task set. Since it is a random split, this set has overlap in the categories with the training set.

Strong generalization evaluation set (33 tasks): This set consists of tasks with no overlap in categories with either the training or the validation sets. We select text compression categories of Summarization and Title Generation tasks as they require complex understanding of language components, and are sufficiently distinct from training tasks (majority of which are Q/A generation, text manipulation etc.).

While both the evaluations sets are held-out during training, second set tests a stricter level of zero-shot generalizability to unseen tasks.

Instruction components: We use four configurations of the instruction components appended to the input of train instances: **1) None** uses the short task description, **2) Desc** uses the long task description. **3) PosEx** adds one positive example of input/output pairs, and **4) Desc + PosEx** includes both the task description and one positive example.

Training parameters: For training, we randomly select a maximum of 100 instances from each task, and train for around 20 epochs[5]. We use the pre-trained BART-base model for most of our experiments and analysis (BART-large numbers are also briefly reported). We use maximum context lengths of 1024 (encoder) and 128 (decoder) tokens. We do not truncate the sequences but instead filter out sequences which do not fit in the context length. We make sure that all models and configurations get exposed and evaluated on the same data by pre-filtering out instances for the longest input configuration (Desc + PosEx). All models are trained on a single NVIDIA-V100 GPU with 32GB memory, and takes around 24-72 hours (depending on the model complexity and data size) to train. More training details are provided in the appendix.

[5]Similar to (Wang et al., 2022), we see that more instances or epochs can overfit on the training data and lead to poor zero-shot generalization.

For evaluation, we follow the NIV2 settings where a maximum of 100 instances per task was used for evaluation to reduce overhead for large scale experimentation. This leads to around 3000 instances for the smaller strong generalization set, and around 5000 for the larger weak generalization set with multiple references per instances. We generate predictions using greedy decoding and compute ROUGE-2F and ROUGE-L (both follow similar trends with minor differences). However, we found that ROUGE-2F was slightly more robust with less variance in the zero-shot scenario, when in the initial stages of training, the model frequently copies the input prompt/instruction to output giving unnaturally high ROUGE-L scores. ROUGE-2F was less susceptible to such a scenario and thus we use ROUGE-2F metrics for our analysis (ROUGE-L is reported in the appendix). We compute both the overall metrics as well as the task level metrics and analyze them in detail.

For implementation, we adapted code from several open source packages: HuggingFace Transformers (Wolf et al., 2020), learn2learn (Arnold et al., 2020), Higher (Grefenstette et al., 2019), and KnowledgeEditor (Cao et al., 2021)[6]. Our adaptations and complete training and evaluation scripts will be open sourced for further research.

5.2 Summary of Results

We first provide a short summary of conclusions from our study before diving into the details.

Task descriptions: Descriptions improve performance when used by itself. When used with positive examples, it improves the performance, for the strong generalization set. For the weak set, positive examples are sufficient.

Training sets: The all-tasks set has better performance even though the generation-set is more similar to the evaluation tasks. I.e. it is better to train on a larger number of diverse tasks for better zero-shot generalization.

Evaluation sets: For the strong generalization set, instructions and meta-learning improve the performance. For the weak set, meta-learning improves performance only when using task descriptions without demonstrations or with the smaller training set.

Model performance: Both MAML and HNet models improve performance but HNet with task specific encoding is better. HNet-MAML improves

[6]https://github.com/nicola-decao/KnowledgeEditor

Evaluation Tasks	Instructions	Standard	MAML	HNET	HNET_MAML
BART BASE + Generation Tasks Train Set (306)					
Strong Generalization Eval Set: Summarization + Title Generation (33 Tasks)	None	0.101	**0.1078**	0.0963	0.0922
	Desc	0.11	**0.1112**	0.1071	0.1022
	PosEx	0.1168	0.1123	**0.1189**	0.1157
	Desc + PosEx	0.1237	0.1269	0.1282	**0.1306**
Weak Generalization Eval Set: Random Split (81 Tasks)	None	0.1	0.0964	0.0973	**0.1029**
	Desc	0.1004	0.1021	0.1028	**0.1045**
	PosEx	**0.1353**	0.1312	**0.1353**	0.1324
	Desc + PosEx	**0.1198**	0.1154	0.1167	0.1169
BART BASE + All Tasks Train Set (916)					
Strong Generalization Eval Set: Summarization + Title Generation (33 Tasks)	None	0.0196	0.0162	0.0359	**0.0395**
	Desc	0.1135	0.1148	0.1196	**0.1219**
	PosEx	0.1301	0.1311	**0.1332**	0.1297
	Desc + PosEx	0.1302	0.1321	0.1337	**0.137**
Weak Generalization Eval Set: Random Split (81 Tasks)	None	0.0921	0.0943	0.0961	**0.1**
	Desc	0.106	0.1037	0.1036	**0.108**
	PosEx	**0.1369**	0.1335	0.1327	0.1326
	Desc + PosEx	0.1158	0.1171	**0.124**	0.1218

Figure 2: Metrics (ROUGE-2F) with different models and instruction fields on the two training and evaluation sets. Instructions improve the performance specially for the strong generation set. Best metrics are obtained with **HNet-MAML** using **Desc+PosEx**.

the performance by almost 30% overall in the strong generation set, showing the effect of twin enhancements.

Task difficulty level: Meta-learning significantly improves performance (by almost 100%) for "hard" tasks, with HNet-MAML having the best performance. Meta-learning also significantly increases the impact of using instructions: impact increases from 250% for standard to 1500% using HNet-MAML.

5.3 Performance of Different Models

Baseline metrics on the standard training with ablations on instruction components, and a short ablation study comparing the different models are presented in the appendix in Figures 8 and 9.

Here, we compare the overall performance of the standard model with the meta-learning approaches in Figure 2 which reports the overall ROUGE scores for the different evaluation sets.

Role of instructions: Instructions improve metrics for the strong generalization set across all the models and for both the training sets. As the instructions get more detailed (None through Desc+PosEx) we see improved performance with the more complex models. When no instructions are used (None), MAML performs the best. When entire instruction is used (PosEx+Desc) HNet-MAML has the best performance. The results illustrate that instructions are more effective with meta-learning, particularly for the strong generalization set. However, for the weak generalization set, the surface patterns of the positive examples are sufficient and task descriptions do not improve the generalization capability. In addition, the standard training achieves the best performance for the weak set showing that meta-learning is not as effective for weak generalization conditions.

Training sets: Overall, the performance is better with the larger training set but mostly for the strong generalization set. For the weaker set, the smaller training set of just 306 tasks is competitive with the larger set (achieved with just the positive examples and standard training), showing that task similarity in train and evaluation does matter. However, under stricter zero-shot conditions it is better to have a more diverse and larger set of training tasks.

Strong generalization eval set: We see improved performance of MAML and HNet variants for the strong generalization set for both sets of training tasks. When instructions are most detailed (Desc + PosEx), HNet-MAML has the best performance with both the training sets. The results illustrate the effectiveness of creating task specific model parameters through the HNet especially under strict zero shot settings. Moreover, HNet-MAML has the best performance showing that the twin optimizations utilize instructions and generalize better to out-of-distribution tasks.

Weak generalization eval set: For the weak set (when the model has seen similar tasks in training), task descriptions are not useful and best metrics are achieved using just the positive examples across all models. It is also interesting to note that the smaller train set has comparable metrics with the larger set, showing that the set can learn mostly from the set of tasks in the generation train set. Moreover, results here show that standard training from (Wang et al., 2022) is a strong baseline and thus further distinguishes the performance of meta-learning in utilizing instructions under strict zero-shot conditions.

5.4 Effect of Instructions on Tasks with Different Difficulty Levels

Next, we analyze how the models utilize the different instruction components by breaking down the performance to different difficulty levels in Figure 3. We compute the ROUGE scores for each task separately, and deem tasks whose metrics from the baseline (standard training) are low as *hard* and ones with high as *easy*. Then we sort the tasks by the scores and divide them into three difficulty groups: "easy", "medium" and "hard". For the strong generalization set of summarization and title

	Difficulty	None	Desc	PosEx	Desc + PosEx
Standard	Hard	0.0118	103.01%	437.03%	642.19%
	Medium	0.0417	25.38%	84.72%	78.01%
	Easy	0.1847	30.67%	19.31%	29.02%
	Overall	0.0794	53.02%	180.36%	249.74%
MAML	Hard	0.0109	130.72%	856.93%	944.96%
	Medium	0.0436	37.34%	80.77%	138.71%
	Easy	0.2212	8.91%	-12.61%	-2.76%
	Overall	0.0919	58.99%	308.36%	360.30%
HNET	Hard	0.0088	209.36%	1290.04%	1598.32%
	Medium	0.0413	9.51%	43.69%	36.04%
	Easy	0.215	9.29%	1.14%	14.53%
	Overall	0.08841	**76.05%**	**444.95%**	549.63%
HNET-MAML	Hard	0.007273	79.69%	737.02%	4217.57%
	Medium	0.038718	36.34%	190.14%	146.61%
	Easy	0.236009	24.13%	15.35%	-6.22%
	Overall	0.094	46.72%	314.17%	**1452.65%**

Figure 3: The % improvements with different instruction components. Tasks are split into easy/medium/hard based on the ROUGE-2F scores from standard training with the None setting. Instructions help the **hard tasks** across all models with best results using **HNet-MAML**.

Evaluation Tasks	Difficulty	MAML	HNET	HNET_MAML
BART BASE + Generation Tasks Train Set (306)				
Strong Generalization Eval Set: Summarization + Title Generation (33 Tasks)	Hard	42.87%	67.29%	**90.69%**
	Medium	9.51%	8.32%	**13.27%**
	Easy	**-0.37%**	-1.57%	-1.60%
	Overall	17.34%	24.68%	**34.12%**
Weak Generalization Eval Set: Random Split (81 Tasks)	Hard	60.10%	81.18%	**88.73%**
	Medium	8.14%	0.34%	**17.18%**
	Easy	-4.00%	-7.19%	**-1.87%**
	Overall	21.41%	24.78%	**34.68%**
BART BASE + All Tasks Train Set (916)				
Strong Generalization Eval Set: Summarization + Title Generation (33 Tasks)	Hard	79.68%	11.23%	**87.35%**
	Medium	1.99%	**13.55%**	6.65%
	Easy	-1.24%	**-0.16%**	-5.37%
	Overall	26.81%	8.21%	**29.54%**
Weak Generalization Eval Set: Random Split (81 Tasks)	Hard	15.54%	33.86%	**62.79%**
	Medium	8.06%	15.21%	**49.62%**
	Easy	-0.11%	**1.59%**	-2.20%
	Overall	7.83%	16.89%	**36.74%**

Figure 4: % differences from standard training with other models for the two training and evaluation sets. **HNet-MAML** and **hard tasks** have the largest improvements across the different train and eval sets.

generation tasks (33 tasks), we have 11 tasks while for the weak generalization set (81 tasks), we have 27 tasks per group. We report the % change in the metrics with different instruction components for each model with the "None" setting as the baseline in Figure 3.

Instructions help hard tasks: Figure 3 shows that instructions have the biggest impact on the hard tasks and when both positive examples and descriptions are used. For example, we get an improvement of 642% with standard training and 4000% for Hnet-MAML using instructions (compared to the None setting). For easy tasks, instructions can even lead to regression in metrics. For example, we see a -6.22% regression with HNet-MAML model.

Meta-learning increases the effectiveness of instructions: Figure 3 also shows that meta-learning can significantly boost performance. Overall, HNet-MAML is able to increase the performance by 1452% using instructions over the None setting compared to around 250% for standard, 360% for MAML, and 549% for HNet. Thus, while MAML is able to utilize the instructions better than standard training, HNet by explicitly conditioning the parameters on instructions can improve it further for out-of-distribution tasks where the additional information from instructions are most useful. In addition, we see that the twin meta-learning optimizations with HNet and MAML maximize the utilization of instructions.

5.5 Detailed Task Level Analysis of Models

Next we analyze the metrics across the two training and evaluation sets for the different models. Here we employ the full instruction (Desc+PosEx) and use the metrics from standard training. We divide the tasks into the three difficulty levels of easy/medium/hard and report the % changes with other models in Figure 4.

It is interesting to contrast Figure 4 with the overall metrics reported in Figure 2. While the overall metrics show small ROUGE differences, breaking down the metrics at the task difficulty levels illustrate significant differences. This is due to the large range of ROUGE scores across tasks, which can normalize the overall differences.

Train sets: In Figure 4, we see that relative performance improvement with meta-learning approaches are similar for the two training sets across the models even if the absolute numbers differ. HNet-MAML has the best performance across the two train sets with an overall improvement of around 30% compared to standard training.

Eval sets: For the two eval sets, when broken down into task difficulty levels, we surprisingly see that meta-learning models are better for both the weak and the strong sets. This is because even in the weak set, there are hard tasks for which the meta-learning models improve the generalization. Moreover, the hard tasks get the highest improvements using HNet-MAML and shows the impact of twin optimization on out-of-distribution tasks inside both weak and strong generalization sets.

Per-task metrics: We report the per-task metrics for the three groups in Figure 5 for the strong generalization set. The standard model does really

Hard Tasks	Standard	MAML	HNET	HNET_MAML
task569_recipe_nlg_text_generation	0.001	226.32%	10.53%	131.58%
task613_politifact_text_generation	0.001	0.00%	0.00%	0.00%
task620_ohsumed_medical_subject_headings	0.0038	0.00%	0.00%	0.00%
task1290_xsum_summarization	0.009	1.90%	51.43%	10.48%
task1342_amazon_us_reviews_title	0.0092	109.78%	14.13%	46.74%
task743_eurlex_summarization	0.0105	23.35%	26.95%	17.96%
task1357_xlsum_summary_generation	0.0153	-15.69%	-28.76%	1.96%
task589_amazonfood_summary_text_generation	0.0167	112.22%	100.00%	210.00%
task510_reddit_tifu_title_summarization	0.0203	-5.42%	572.41%	574.88%
task511_reddit_tifu_long_text_summarization	0.0236	8.05%	8.90%	-1.69%
task500_scruples_anecdotes_title_generation	0.039	11.03%	-15.38%	5.64%
	0.0135	42.87%	67.29%	**90.69%**
Medium Tasks				
task1540_parsed_pdfs_summarization	0.0464	-11.05%	-1.52%	3.05%
task418_persent_title_generation	0.0525	16.38%	1.94%	20.26%
task219_rocstories_title_answer_generation	0.0582	52.41%	57.56%	35.05%
task1639_doqa2.1_travel_text_summarization	0.0592	-19.17%	-32.75%	-6.23%
task1358_xlsum_title_generation	0.0626	9.55%	17.68%	9.24%
task1356_xlsum_title_generation	0.0628	5.74%	7.60%	-17.06%
task1586_scifact_title_generation	0.069	-5.71%	-31.71%	-17.57%
task618_amazonreview_summary	0.07	17.97%	10.00%	39.24%
task1161_coda19_title_generation	0.0772	10.43%	0.29%	7.39%
task288_gigaword_summarization	0.079	26.17%	30.83%	24.74%
task899_freebase_qa_topic_generation	0.0878	1.94%	31.66%	47.84%
	0.0658	9.51%	8.32%	**13.27%**
Easy Tasks				
task1637_doqa2.1_cooking_text_summarization	0.0949	-16.46%	-3.25%	-5.24%
task1572_samsum_summary	0.0954	-18.76%	-4.11%	-0.32%
task668_extreme_abstract_summarization	0.1038	7.51%	-13.68%	8.38%
task619_ohsumed_abstract_title_generation	0.1153	0.94%	-1.03%	-16.74%
task1659_title_generation	0.1165	17.95%	0.96%	-7.89%
task1638_doqa2.1_movies_text_summarization	0.1771	11.91%	12.93%	11.80%
task1355_sent_comp_summarization	0.2075	0.10%	5.16%	0.24%
task1499_dstc3_summarization	0.3023	-1.49%	-4.83%	-4.50%
task1340_msr_text_compression_compression	0.5043	-1.30%	-4.43%	-3.99%
task769_qed_summarization	0.5239	-0.38%	-4.12%	5.12%
task645_summarization	0.6015	-4.07%	-0.81%	-4.46%
	0.2584	**-0.37%**	-1.57%	-1.60%

Figure 5: % differences listed for individual tasks divided into easy/medium/hard difficulty levels. Results show that MAML, HNet and HNet-MAML models have significant improvements for the hard tasks)

poorly in some of the hard tasks. Essentially these are tasks where the patterns are significantly different from the training tasks and the model is unable to generalize to the new instruction patterns. This is where meta-learning and in particular the twin optimization of MAML and HNet significantly improve the scores. This improvement can be the key factor whether zero-shot based predictions can be used practically and can result in big difference in how users perceive the model quality.

BART-Large: We also report the task level metrics for BART-Large in Figure 6. Meta-learning has a higher impact with BART-large compared to BART-Base. However, the performance for HNet and HNet-MAML is mixed. HNet-MAML with BART-large is difficult to train on a single GPU due to high memory requirements (requiring small batch sizes), which might have reduced its effectiveness (See appendix on the training parameters for different models and further discussion).

6 Conclusions

In this paper we investigate whether meta-learning applied to multi-task *instructional learning* (MTIL) can boost the generalizability of LMs to unseen tasks in a zero-shot setting. Specifically, we evaluate MTIL in three directions with MAML, HNet and HNet-MAML. To test the generalization ability, we consider two sets of training and evaluation task sets and through extensive experiments on the NIV2 dataset, show that meta-learning can significantly boost the performance by increasing the effectiveness of instructions particularly under strict zero shot conditions and for "hard" tasks.

Evaluation Tasks	Difficulty	MAML	HNET	HNET_MAML
BART LARGE + All Tasks Train Set (916)				
Strong Generalization Eval Set: Summarization + Title Generation (33 Tasks)	Hard	15.68%	**101.31%**	80.53%
	Medium	**8.96%**	-0.92%	-2.17%
	Easy	-5.95%	-6.71%	-7.68%
	Overall	6.23%	**31.23%**	23.56%
Weak Generalization Eval Set: Random Split (81 Tasks)	Hard	176.08%	116.76%	**316.10%**
	Medium	11.30%	**17.48%**	5.37%
	Easy	**0.50%**	-1.59%	-7.77%
	Overall	62.63%	44.22%	**104.57%**

Figure 6: % differences from standard training with other models with **BART-Large**. **HNet** has the best performance for the strong generalization set.

While the models perform relatively well under zero-shot conditions, the performance is far from fully supervised models. It remains to be seen at what point we can match fully supervised models (for example using a k-shot setting). In addition, the impact of HNet-MAML on the BART-Large model was lower. It will be interesting to see how meta-learning scales with model sizes and whether the additional bandwidth from larger models can negate the impact of meta-learning in encoding and utilizing instructions. This is subject of future work.

7 Limitations

There are several limitations of the proposed meta-learning based approaches in its present form.

- Computation and memory overhead: Meta-learning approaches have higher resource requirements which can limit the usage specially for larger models. For example with BART-large, the HNet-MAML model on a single GPU is inefficient to train since we have to use small batch sizes which leads to lower performance.
- Regressions with easy tasks: We see some regression in metrics for the easy tasks. Further analysis and research is needed to understand the factors and improve the models such that model enhancements are uniform across tasks.
- Hyper-parameter tuning: Meta learning models have more hyper-parameters and thus

might be more difficult to tune than the standard training approach.

- Overall zero-shot performance: The zero-shot performance even with the best meta-learning approaches is quite far from state-of-the-art results. It will be interesting to see at what point (e.g. with k-shot learning) the performance can match a fully supervised model.

References

Sébastien M R Arnold, Praateek Mahajan, Debajyoti Datta, Ian Bunner, and Konstantinos Saitas Zarkias. 2020. learn2learn: A library for Meta-Learning research.

Stephen H. Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M SAIFUL BARI, Thibault Févry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Mike Tian-Jian Jiang, and Alexander M. Rush. 2022. Promptsource: An integrated development environment and repository for natural language prompts. *ArXiv*, abs/2202.01279.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Nicola De Cao, W. Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *EMNLP*.

Yi-Syuan Chen and Hong-Han Shuai. 2021. Meta-transfer learning for low-resource abstractive summarization. *ArXiv*, abs/2102.09397.

Chelsea Finn, P. Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. 2019. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*.

Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and Victor O. K. Li. 2018. Meta-learning for low-resource neural machine translation. In *EMNLP*.

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. Hypernetworks. *ArXiv*, abs/1609.09106.

Hamish Ivison and Matthew E. Peters. 2022. Hyperdecoders: Instance-specific decoders for multi-task nlp.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 565–576, Online. Association for Computational Linguistics.

David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. 2018. Bayesian hypernetworks.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

Chien-Fu Lin and Hung-Yi Lee. 2020. Master of puppets: Model-agnostic meta-learning via pre-trained parameters for natural language generation. *Neurips Workshop on Meta Learning*, abs/2110.15943.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ArXiv*, abs/2107.13586.

Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. 2019. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. *ArXiv*, abs/1905.05644.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022. MetaICL: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, Seattle, United States. Association for Computational Linguistics.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast model editing at scale. In *International Conference on Learning Representations*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton,

Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, abs/1910.10683.

Emily Reif, Daphne Ippolito, Ann Yuan, Andy Coenen, Chris Callison-Burch, and Jason Wei. 2021. A recipe for arbitrary text style transfer with large language models. *ArXiv*, abs/2109.03910.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*.

Timo Schick and Hinrich Schütze. 2021b. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *NAACL-HLT*, pages 2339–2352.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *International Conference on Learning Representations*.

Johannes von Oswald, Christian Henning, Benjamin F. Grewe, and João Sacramento. 2020. Continual learning with hypernetworks. In *International Conference on Learning Representations*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddhartha Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Hannaneh Hajishirzi, Noah A. Smith, and Daniel Khashabi. 2022. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *ArXiv*, abs/2104.08773.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Model Details

A.1 Standard Training with MAML

The training loop for MAML with instructions is given in Algorithm 1. At each step, we sample two sets of tasks for the MAML fast-adaptation and meta-update steps. To increase the generalization of the process, we force tasks to be unique across the two sets. This forces the use of different sets of instructions for the two MAML steps. We also sample tasks uniformly to maximize the diversity of tasks the model gets exposed to during training. These factors improve the performance over a proportionate sampling method.

For MAML training, we consider the following hyper-parameters: inner and outer learning rates, the inner batch sizes, the number of inner loop iterations, and the number of tasks in the loop. We use first-order MAML as it takes significantly more memory to maintain the second order gradients and fit large batches in a single GPU.

We conducted ablations studies to find the best hyper-parameters for MAML. It is important to note that the hyper-parameters are constrained by memory. For example if we increase the batch

size, or the number of tasks, we have to reduce the number of inner steps to keep the training memory within the GPU memory limits. Based on our ablations we use 3 inner steps, a batch size of 10, and 2 tasks per MAML step. We use a inner learning rate of $5e-3$ and an outer learning rate of $5e-4$ for all the experiments. For BART-Large we use a batch size of 4. With gradient accumulation steps we can increase the effective batch sizes to up-to around 400 for BART-base and 200 for Bart-Large.

Algorithm 1 MAML Loop with Instructions

$MAML(\theta_{MainLM}; I_\tau, x, y)$
  Sample K Support and Target tasks τ_s, τ_t
  $\Theta_0 \leftarrow \theta$
  for $k \in 0 : K_{tasks}$ **do**
    for $n \in 0 : N_{steps}$ **do**
      $\Theta_{n+1,k} \leftarrow GD\mathcal{L}(\Theta_{n,k}, , (I_{\tau_s}, x, y)$
    end for
  end for
  $\theta = \Theta_N - \beta \nabla_{\Theta_0} \sum_k \mathcal{L}(\Theta_{N,k}; (I_{\tau_t}, x, y))$

A.2 Standard Training with HNET

Standard training with HNet is described in Algorithm 2. The HNet model consists of HNet-LM, HNet-FFs and the Main-LM. We use the BART model for both the HNet-LM and Main-LM. While potentially any pretrained text encoder could have been used for the HNet-LM, we keep the two LMs the same. This is mostly a practical consideration which allows us to use a single tokenizer to process the text. This also maintains the uniformity of input across models (for examples instructions fed into the HNet-LM vs. Main-LM) and allows flexibility in the use the tokenized text in either LMs.

Algorithm 2 HNet with Instructions

$HNET(\theta_{MainLM}, \Phi_{HNetLM}; I_\tau, x, y)$
  Sample a task τ and mini-batch from the task
  $\Delta\theta_\tau \leftarrow \Phi(I_\tau)$
  $\theta_\tau \leftarrow \theta_{Main} + \Delta\theta_\tau$
  if Alternating **then**
    if $steps\%k = 0$, Freeze θ **then**
      $\Phi' \leftarrow \mathcal{L}_{GD}(\Phi; \theta_\tau(I_\tau, x, y))$
    elseFreeze Φ
      $\theta' \leftarrow \mathcal{L}_{GD}(\theta; \theta_\tau(I_\tau, x, y))$
    end if
  else
    $\theta', \Phi' \leftarrow \mathcal{L}_{GD}(\Phi, \theta; \theta_\tau)(I_\tau, x, y))$
  end if

During training, each instance for a given task produces a unique task specific LM. This prevents training in batch-mode when the batch consists of a random set of tasks, and considerably slows down the training. To speed up training, we always create a batch of instances from the same task such that at any step there is a single task specific LM through which we do the forward and backward steps. While this reduces the metrics to an extent, it leads to much faster training of the HNet models.

Ablation type		ROUGE-2F	ROUGE-L
Different target layers, Alternating Opt	Decoder	**0.1282**	**0.2479**
	Encoder	0.1268	0.2476
	Both	0.128	0.245
Different target layers, Joint Optimization	Decoder	0.1269	0.2411
	Encoder	0.1258	0.2385
	Both	0.125	0.2378
Last vs sequence of hidden states	FALSE	**0.1282**	**0.2479**
	TRUE	0.1194	0.2331
Different Hidden dimensions	**32**	0.1229	0.2409
	64	**0.1289**	0.2456
	128	0.1282	**0.2479**
	256	0.1244	0.2366
	512	0.1221	0.2359
	1024	0.1182	0.2308
Number of Alternating steps	**1**	0.1253	0.2428
	5	0.126	0.2418
	10	**0.1282**	**0.2479**
	20	0.128	0.2436
	25	0.1255	0.2443
	50	0.1274	0.2442
	100	0.1229	0.2388

Figure 7: Ablation studies with HNET model.

We conducted extensive ablations with the HNet model to understand the best hyper-parameters. The ablations are based on the following hyper-parameters and configurations, shown in Figure 7 and discussed below.

- HNet target layers: Since the Main-LM has two transformer stacks in the encoder and decoder, we consider predicting the parameters of encoder, decoder, or both using the HNet. Targeting the decoder has the best performance.
- Last vs. sequence of output hidden states from HNet-LM: Since the HNet-LM is also an encoder-decoder model, we compare using last hidden state vs. the sequence of hidden states for projecting to the Main-LM parameter space. We find that sequence of hidden steps has better performance.
- HNet hidden dimension: The dimension of the hidden layer for the FF projections for each layer. Our ablations show that the best metrics are achieved with a dimension size of 128.
- Training Schedule: Joint vs. Alternating. We found that the alternating schedule has better performance. Having a schedule frequency of 10 steps leads to the best results.

Algorithm 3 HNET-MAML with Instructions

$HNETMAML(\theta_{MainLM}, \Phi_{HNetLM}; I_\tau, x, y)$
  $\Theta_0 \leftarrow [\theta_0, \Phi_0]$
  Sample Support and Target tasks τ_s, τ_t
  Sample mini-batches from τ_s, τ_t
  for $n \in 0 : N$ **do**
    $\theta_{n+1}, \Phi_{n+1} \leftarrow HNET(\theta_n, \Phi_n; I_{\tau_s}, x, y)$
  end for
  $\Theta_0 \leftarrow \theta_N - \beta \nabla_{\theta_0} HNET(\theta_n, \Phi_n; I_{\tau_t}, x, y)$

A.3 HNet-MAML model

The HNet-MAML model is described in Algorithm 3. Here, the input model to the MAML loop is the HNet model comprising of the HNet-LM, HNet-FFs and the Main-LM. Within the MAML inner steps, each individual step uses the HNet inner step (i.e. first project the instructions to the Main-LM parameter space and then generate using the Main-LM). Similar to the HNet training, we use the alternating training schedule for the HNet inner loop. The HNet's inner alternating frequency takes into account the number of inner MAML steps, gradient accumulation steps and extra steps due to the fast adaptation and meta-update steps of the outer MAML loop to ensure that only one of the LMs are updated during the inner and outer MAML loops.

For BART-base we use an inner batch size of 10, 3 inner steps and 2 tasks per MAML loop. For BART-Large, we use an inner batch size of 2, with 3 inner steps and 2 tasks (one each for MAML train and test steps). As with the HNET model, to enable batch processing, we sample instances in each step from the same task. Using gradient accumulation, with BART-base models we are able to train with batch sizes up-to 200. While this is lower than the other models, it is still significantly high enough to get good training convergence. However, with BART-large this number reduces significantly to around 50. This might be one of the reasons why we do not get similar improvements with BART-Large models. Since the training complexity of HNet-MAML is higher compared to the other models, we only train the model for 10 epochs. In contrast, we train the other models for 20 epochs.

B Experimental settings

In the main section of the paper we have summarized the settings for our experiments. Here we provide some more details.

We use the BART encoder for both the HNet and the Main-LM models. We use a maximum context length of 1024 for the encoder and 128 for the decoder. Note that in MTIL, the inputs to the encoder includes the task description, one positive example and the training instance with an input and output text. This significantly increases the context length of inputs compared to traditional supervised training. In addition, we consider tasks such as summarization, whose inputs are typically long. Thus some of the instances of training and evaluation cannot be fitted into the chosen context windows.

We do not truncate the inputs. Instead we remove those instances which exceed the maximum context lengths. I.e. we compute the tokenized lengths of instances with the longest input configuration (Instruction = Desc + Positive Example) and remove all instances which exceed the context lengths of encoder (we do the same for the decoder max length of 128). Thus for all configurations, the training, validation and the evaluation instances remain the same irrespective of the input instruction configurations. Some tasks such as CNN-Daily mail summarization get completely filtered by this process due to the long input as well as the long instructions. For others it is partial, but since we use a maximum of 100 instances per task during training (much lower than total instances available),

	Model	ROUGE-2F	ROUGE-L
1	**Standard** + PosEx	0.0992	0.1904
2	**Standard** + Desc + PosEx	0.1012	0.2044
3	**MAML** + PosEx	0.1031	0.1948
4	**HNET_Joint** + PosEx	0.1095	0.2081
5	**HNET_Alternate** + PosEx	0.1136	0.2173
6	**MAML** + Desc + PosEx	0.1141	0.2171
7	**HNET_Alt** + Desc + PosEx	0.117	0.2199
8	**HNET_Alt_HidSeq** + PosEx	0.1194	0.222
9	**HNET_Alt_HidSeq** + Desc + PosEx	0.1226	0.2315

Figure 8: Ablation studies on the validation set for different models. The proposed model enhancements improve the performance.

Train Tasks		Strong Generalization set: Summarization + Title Generation (33 Tasks)		Weak Generalization Set: Random Split (81 Tasks)	
	Instructions	ROUGE-2F	ROUGE-L	ROUGE-2F	ROUGE-L
Generation Tasks (~306)	None	0.101	0.1965	0.1	0.2248
	Desc	0.11	0.2045	0.1004	0.2291
	PosEx	0.1168	0.2269	**0.1353**	**0.3089**
	Desc + PosEx	**0.1237**	**0.2368**	0.1198	0.2875
All Tasks (~916)	None	0.0196	0.0548	0.0921	0.2222
	Desc	0.1305	0.2291	0.106	0.2497
	PosEx	0.1301	0.249	**0.1369**	**0.3102**
	Desc + PosEx	**0.1302**	**0.2521**	0.1158	0.2852

Figure 9: Baseline metrics with standard training and different instructional fields. Instructions improve the performance over the *None* setting. Best metrics are obtained when using Desc + PosEx.

each task has similar number of instances. For the all tasks training set, we have a total of around 120k train instances, while we have around 52k for the smaller generation task sets.

C Baseline Metrics with Standard Training

Here, we report the baseline performance using the standard training approach, and different instruction components in Figure 9.

Role of task description: Long task descriptions by themselves are useful ("Desc" is better than "None") for both train task sets. Task description has a bigger impact when using the all-tasks set of 916 tasks. This is possibly because the short task description becomes ambiguous (e.g. several question generation tasks may have the same short description) across more number of tasks.

Weak vs. strong generalization eval set: Descriptions improve generalization but only for the strong generalization set. For the weak set, the best metrics are achieved by using one positive example.

Tasks vs number of instances: Using a larger number of tasks leads to better performance as seen Figure 9 (all tasks set is better than the generation set). The effect is not just due to more data in the all-task set. We tested with similar data sizes between the two training sets and found similar results.

BART Large: Performance significantly improves for the the larger LM but the overall trends remain the same.

D Pilot Studies with Different Models

We conducted a pilot study with around 100 train tasks and 10% of the validation data and compared metrics for different models under different settings and hyper-parameters. Here the aim was to find the ideal hyper-parameters as well as quickly validate if the proposed solutions work as expected, before doing more detailed analysis. Moreover, we wanted to use approximately similar settings for different models as this is likely to be the case in a zero shot setting, where we would not have validation datasets to tune our hyperparameters.

We summarize the best performance from different models in this pilot study in 8. We see that using both the task description as well as a positive example has the best performance across all models. For the HNet based model, we see that using the training schedule of alternately freezing the Main and HNet language model has better performance than jointly training both the networks. We also see that using the sequence of hidden states instead of the last hidden from the HNET decoder leads to better performance as expected.

Hard Tasks	Standard-R2F	Standard-RL	MAML-R2F	MAML-RL	HNET-R2F	HNET-RL	HNET_MAML-R2F	HNET_MAML-RL
task569_recipe_nlg_text_generation	0.001	0.1174	226.32%	30.07%	10.53%	22.01%	131.58%	-5.61%
task613_politifact_text_generation	0.001	0.0146	0.00%	8.60%	0.00%	-5.45%	0.00%	5.54%
task620_ohsumed_medical_subject_headings_answer_generation	0.0038	0.0695	0.00%	32.19%	0.00%	98.63%	0.00%	71.23%
task1290_xsum_summarization	0.009	0.0433	1.90%	-1.59%	51.43%	3.63%	10.48%	-8.40%
task1342_amazon_us_reviews_title	0.0092	0.0506	109.78%	3.95%	14.13%	5.14%	46.74%	14.23%
task743_eurlex_summarization	0.0105	0.1131	23.35%	0.85%	26.95%	42.19%	17.96%	3.08%
task1357_xlsum_summary_generation	0.0153	0.1169	-15.69%	4.79%	-28.76%	3.85%	1.96%	-4.45%
task589_amazonfood_summary_text_generation	0.0167	0.0941	112.22%	15.01%	100.00%	30.48%	210.00%	29.56%
task510_reddit_tifu_title_summarization	0.0203	0.0662	-5.42%	11.48%	572.41%	294.41%	574.88%	297.89%
task511_reddit_tifu_long_text_summarization	0.0236	0.1238	8.05%	3.47%	8.90%	5.01%	-1.69%	8.00%
task500_scruples_anecdotes_title_generation	0.039	0.1331	11.03%	9.84%	-15.38%	0.60%	5.64%	10.89%
	0.0135	0.0856	**42.87%**	**10.79%**	**67.29%**	**45.50%**	**90.69%**	**38.36%**
Medium Tasks								
task1540_parsed_pdfs_summarization	0.0464	0.1533	-11.05%	1.77%	-1.52%	-2.17%	3.05%	-4.22%
task418_persent_title_generation	0.0525	0.1753	16.38%	0.65%	1.94%	4.50%	20.26%	4.44%
task219_rocstories_title_answer_generation	0.0582	0.1654	52.41%	30.17%	57.56%	55.99%	35.05%	36.88%
task1639_doqa2.1_travel_text_summarization	0.0592	0.1256	-19.17%	-4.42%	-32.75%	-13.40%	-6.23%	-7.32%
task1358_xlsum_title_generation	0.0626	0.1925	9.55%	8.54%	17.68%	10.06%	9.24%	10.44%
task1356_xlsum_title_generation	0.0628	0.1839	5.74%	15.92%	7.60%	20.22%	-17.06%	-5.81%
task1586_scifact_title_generation	0.069	0.1635	-5.71%	-0.99%	-31.71%	-16.53%	-17.57%	-9.87%
task618_amazonreview_summary_text_generation	0.07	0.1621	17.97%	4.58%	10.00%	-3.27%	39.24%	13.73%
task1161_coda19_title_generation	0.0772	0.2154	10.43%	-1.28%	0.29%	2.32%	7.39%	5.99%
task288_gigaword_summarization	0.079	0.1836	26.17%	8.36%	30.83%	5.80%	24.74%	5.39%
task899_freebase_qa_topic_generation	0.0878	0.2476	1.94%	-1.37%	31.66%	23.83%	47.84%	21.37%
	0.0658	0.1789	**9.51%**	**5.63%**	**8.32%**	**7.94%**	**13.27%**	**6.45%**
Easy Tasks								
task1637_doqa2.1_cooking_text_summarization	0.0949	0.2666	-16.46%	-11.90%	-3.25%	-7.84%	-5.24%	-2.92%
task1572_samsum_summary	0.0954	0.2193	-18.76%	-3.75%	-4.11%	0.08%	-0.32%	2.25%
task668_extreme_abstract_summarization	0.1038	0.2265	7.51%	2.91%	-13.68%	-10.20%	8.38%	-0.09%
task619_ohsumed_abstract_title_generation	0.1153	0.2222	0.94%	3.20%	-1.03%	-1.79%	-16.74%	-5.28%
task1659_title_generation	0.1165	0.2123	17.95%	9.95%	0.95%	2.66%	-7.89%	-6.08%
task1638_doqa2.1_movies_text_summarization	0.1771	0.3216	11.91%	5.38%	12.93%	10.42%	11.80%	9.14%
task1355_sent_comp_summarization	0.2075	0.2756	0.10%	0.44%	5.16%	4.54%	0.24%	0.76%
task1499_dstc3_summarization	0.3023	0.4532	-1.49%	-2.54%	-4.83%	-4.41%	-4.50%	-5.54%
task1340_msr_text_compression_compression	0.5043	0.6965	-1.30%	-0.97%	-4.43%	-3.24%	-3.99%	-2.92%
task769_qed_summarization	0.5239	0.6518	-0.38%	-1.11%	-4.12%	-3.06%	5.12%	1.92%
task645_summarization	0.6015	0.8414	-4.07%	-0.44%	-0.81%	-0.70%	-4.46%	-6.82%
	0.2584	0.3988	**-0.37%**	**0.11%**	**-1.57%**	**-1.23%**	**-1.60%**	**-1.41%**

Figure 10: % differences listed for individual tasks divided into easy/medium/hard difficulty levels. Results show that MAML, HNet and HNet-MAML models have significant improvements for the difficult tasks)

Valid Task Names	Short Description
task003_mctaco_question_generation_event_duration	Writing questions that involve commonsense understanding of "event duration".
task005_mctaco_wrong_answer_generation_event_duration	Writing an implausible answer to the given "event duration" question.
task007_mctaco_answer_generation_transient_stationary	Answering questions that involve commonsense understanding of "transient vs. stationary" events.
task014_mctaco_wrong_answer_generation_absolute_timepoint	Writing an implausible answer to the provided "absolute timepoint" question.
task029_winogrande_full_object	Creating a pair of fill in the blank question-answer pairs on objects.
task030_winogrande_full_person	Creating a pair of fill in the blank questions on persons.
task034_winogrande_question_modification_object	Modifying a fill in the blank question on objects.
task051_multirc_correct_answer_single_sentence	Generating correct answer to single-sentence questions.
task054_multirc_write_correct_answer	Writing a Correct Answer for a Reading Comprehension Task.
task060_ropes_question_generation	Constructing questions regarding relations in the given paragraph.
task074_squad1.1_question_generation	Generate questions based on SQuAD 1.1.
task080_piqa_answer_generation	Generate a solution to a goal regarding physical knowledge about the world.
task084_babi_t1_single_supporting_fact_identify_relevant_fact	Given a question and an answer, identify the relevant piece of evidence.
task102_commongen_sentence_generation	Given a collection of concepts, use them in a coherent sentence.
task112_asset_simple_sentence_identification	Given two excerpts of text, choose the one that is simpler and easier to understand by non-native speakers.
task127_scan_long_text_generation_action_command_all	Given a sequence of actions, provide its natural language command.
task131_scan_long_text_generation_action_command_long	Given a long sequence of actions, provide its natural language command.
task138_detoxifying-lms_classification_fluency	Given a prompt and two completions, determine which completion is more fluent.
task156_codah_classification_adversarial	Given a prompt, select the completion that is the most plausible.
task191_hotpotqa_question_generation	Given a set of context, supporting facts and an answer, generate the question asked based on them.
task193_duorc_question_generation	Generate a question based on a given plot.
task216_rocstories_correct_answer_generation	Given the title and the first four sentences of a five sentence story, write a correct story ending.
task221_rocstories_two_choice_classification	Given three sentences and title of a five sentence story, choose which two sentences from the options given will complete the story.
task235_iirc_question_from_subtext_answer_generation	Given a context statement, further information on a linked term in the statement, and an answer term, generate a question that can use the information provided to obtain the given answer
task247_dream_answer_generation	Given a conversation and a question, answer the question based on the conversation.
task270_csrg_counterfactual_context_generation	Given premise, initial context with ending, and new counterfactual ending, generate counterfactual context which supports the new story ending.
task281_points_of_correspondence	Find the entity or event that is in common between the given three sentences.
task283_dream_incorrect_answer_generation	Given a conversation and a question, write an incorrect answer to the question.
task287_casehold_legal_incorrect_answer_generation	Given a prompt from a judicial decision and multiple potential holdings, choose one of the incorrect options.
task344_hybridqa_answer_generation	Given a question, answer the question based on your knowledge.
task361_spolin_yesand_prompt_response_classification	Given a prompt and a response, classify whether the response is "yes, ands" type
task385_socialiqa_incorrect_answer_generation	You're given a context, a question, three options. Your task is to return an incorrect answer from the option.
task405_narrativeqa_question_generation	Given a plot summary, create questions that can be answered based on it
task456_matres_intention_classification	Given a context and a verb, answer if the given verb is about an intention or not
task460_qasper_answer_generation	Given a context and a question, answer the question based on the context.
task471_haspart_answer_generation	Generating entity which is in has-part-relationship with input entity
task568_circa_question_generation	Given an answer, Predict the question.
task580_socialiqa_answer_generation	Given a context, a question and three options; provide correct answer for the question based on the context.
task581_socialiqa_question_generation	Generate a question based on the given context and an answer.
task591_sciq_answer_generation	Given a scientific question, generate a correct answer to the given question
task595_mocha_answer_generation	Generating answers to MOCHA questions
task619_ohsumed_abstract_title_generation	Generating title to Ohsumed dataset abstracts
task620_ohsumed_medical_subject_headings_answer_generation	Generating MESH terms to Ohsumed dataset abstracts
task621_ohsumed_yes_no_numerical_answer_generation	Generating Yes/No answer to Ohsumed dataset questions
task672_nummersense	Given a cloze question, identify the missing numerical value
task672_amazon_and_yelp_summarization_dataset_summarization	Generating summaries to amazon/yelp reviews
task741_lhoestq_answer_generation_place	Given a passage and a question, answer the question based on the passage to output a particular place or position of something.
task748_glucose_reverse_cause_event_detection	Given a story and a selected sentence, find an event that is directly caused or made possible by that sentence
task770_pawsx_english_text_modification	Given a sentence in English, provide an equivalent paraphrase in said language
task820_protoqa_answer_generation	Given a question, generate a relevant answer to the question
task849_pubmedqa_answer_generation	Generating answer from context and question (based on pubmed_QA)
task860_prost_mcq_generation	Generating MCQs
task870_msmarco_answer_generation	Generating answers based on natural language passage and related query from MS MARCO
task886_quail_question_generation	Generating questions based on passages
task897_freebase_qa_topic_question_generation	Generate question for the given topic
task959_e2e_nlg_text_generation_identify	Identify the named entity that is the subject of the excerpt.
task964_librispeech_asr_text_auto_completion	Text Auto Completion of partial English sentences
task1161_coda19_title_generation	Given a paragraph from a research paper, your task is to generate the title of the paper
task1217_atomic_answer_generation	Given a sentence, fill in the blank with a plausible word.
task1355_sent_comp_summarization	Given text generate summary about the text
task1358_xlsum_title_generation	Generates title for the text in xlsum
task1359_numer_sense_answer_generation	Generates answer to numer sense
task1364_hans_answer_generation	Generating answers (based on Hans)
task1368_healthfact_sentence_generation	Generate a claim based on a given paragraph
task1369_healthfact_sentence_generation	Generate an explanation for a claim based on a given paragraph
task1380_quarel_correct_option_generation	Given a sentence and a question, choose the correct option number instead of exact answer based on the sentence.
task1400_obqa_incorrect_answer_generation	Given a fact and question, generate an incorrect answer to the question
task1415_youtube_caption_corrections_grammar_correction	Given a set of closed captions (from `youtube_caption_corrections`), produce a grammatically correct version of those captions
task1437_doqa_cooking_question_generation	Given a paragraph about cooking, and a set of conversational question answers about the paragraph, generate a relevant question to the topic of the paragraph
task1482_gene_extraction_chemprot_dataset	Given a sentence from the ChemProt dataset, return the list of tokens that mentions of protein.
task1486_cell_extraction_anem_dataset	Given a sentence from the AnEM dataset, return the list of tokens that mentions of cells in the body.
task1487_organism_substance_extraction_anem_dataset	Given a sentence from the AnEM dataset, return the list of tokens that mentions of organs in the body.
task1515_imppres_longtextgeneration	Given a premise, generate hypothesis
task1517_limit_classfication	Classifying sentence based on the condition that it contains a motion of a physical entity or not.
task1555_scitail_answer_generation	Generating answers to SciTail Sentence-Questions
task1566_propara_structured_text_generation	Generate entities from given text
task1590_diplomacy_text_generation	Text generation based on diplomacy_detection
task1600_smcalflow_sentence_generation	Given a agents' reply, generate a users' utterance
task1603_smcalflow_sentence_generation	Given a user utterance, generate agents' utterance
task1608_xquad_en_answer_generation	Generating answers to xquad en questions
task1609_xquad_en_question_generation	Generating questions (based on xquad en)
task1656_gooaq_answer_generation	short_answer generation for given question
task1665_trainglecopa_question_generation	Generating a Question for the given premise from traingleCOPA dataset
task1711_poki_text_generation	Given a title, generate a short poem that should look like written by a kid.
task1729_personachat_generate_next	Generate the next utterance in a conversation

Figure 11: Weak generalization evaluation set: List of tasks with the short task descriptions for the weak generalization set of 81 generation tasks

Eval Task Names	Short Description
task219_rocstories_title_answer_generation	Given a five sentence story, generate an appropriate title for the story.
task288_gigaword_summarization	Given a text of article, generate a title for the article.
task418_persent_title_generation	Given a document, generate a short title of the document.
task500_scruples_anecdotes_title_generation	Given a real-life anecdote of a complex ethical situation, generate a title that describes the main event/root cause of the situation
task510_reddit_tifu_title_summarization	Given the text of a social media post, generate a title summarizing the post
task511_reddit_tifu_long_text_summarization	Given the text of a social media post, generate a short summary the post
task522_news_editorial_summary	Given an article text, select spans of text that show a summary of the thesis of the article.
task569_recipe_nlg_text_generation	Predict the title given its required ingredients and directions.
task589_amazonfood_summary_text_generation	Given a review of amazon's food product, you have to generate the summary of the review.
task613_politifact_text_generation	Given a statement from a politifact.com you task is to generate the subject of discussion of the statement.
task618_amazonreview_summary_text_generation	Given an Amazon product review your task is to generate the summary of the review.
task619_ohsumed_abstract_title_generation	Generating title to Ohsumed dataset abstracts
task620_ohsumed_medical_subject_headings_answer_generation	Generating MESH terms to Ohsumed dataset abstracts
task645_summarization	Generating summary for Data
task668_extreme_abstract_summarization	Generate a summary of this abstract.
task672_amazon_and_yelp_summarization_dataset_summarization	Generating summaries to amazon/yelp reviews
task743_eurlex_summarization	Generate headline (summary) for legal act article
task769_qed_summarization	Generating titles for passage
task899_freebase_qa_topic_generation	Generate the specific topic for a given question
task1161_coda19_title_generation	Given a paragraph from a research paper, your task is to generate the title of the paper
task1290_xsum_summarization	Given an article, summarize it.
task1291_multi_news_summarization	Given some news, summarize them.
task1340_msr_text_compression_compression	Generating Compressed text based on MSR dataset
task1342_amazon_us_reviews_title	Generating Title for Amazon US review dataset
task1355_sent_comp_summarization	Given text generate summary about the text
task1356_xlsum_title_generation	Generating title for the text in xlsum
task1357_xlsum_summary_generation	Generating summary for the text in xlsum
task1358_xlsum_title_generation	Generates title for the text in xlsum
task1499_dstc3_summarization	Summarization of conversations in DSTC 3
task1540_parsed_pdfs_summarization	Given a text, generate a title for it
task1553_cnn_dailymail_summarization	Generating summary to news articles
task1572_samsum_summary	Generate a summary of given conversations
task1586_scifact_title_generation	Title Generation
task1637_doqa2.1_cooking_text_summarization	Generating title from text (based on DoQA 2.1 cooking data)
task1638_doqa2.1_movies_text_summarization	Generating title from text (based on DoQA 2.1 movie data)
task1639_doqa2.1_travel_text_summarization	Generating title from text (based on DoQA 2.1 travel data)
task1658_billsum_summarization	Generating summary (based on billsum)
task1659_title_generation	Generating Title (based on billsum)

Figure 12: Strong generalization evaluation set: List of tasks with the short task descriptions for the strong generalization set of 33 generation tasks from summarization and title generation categories