Learning to Generate Task-Specific Adapters from Task Description

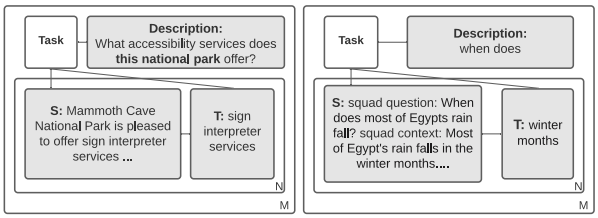
Qinyuan Ye **Xiang Ren**
University of Southern California
{qinyuany,xiangren}@usc.edu

Abstract

Pre-trained text-to-text transformers such as BART have achieved impressive performance across a range of NLP tasks. Recent study further shows that they can learn to generalize to novel tasks, by including task descriptions as part of the source sequence and training the model with (source, target) examples. At test time, these fine-tuned models can make inferences on new tasks using the new task descriptions as part of the input. However, this approach has potential limitations, as the model learns to solve individual (source, target) examples (*i.e.*, at the *instance* level), instead of learning to solve tasks by taking all examples within a task as a whole (*i.e.*, at the *task* level). To this end, we introduce HYPTER, a framework that improves text-to-text transformer's generalization ability to unseen tasks by training a hypernetwork to generate task-specific, light-weight adapters from task descriptions. Experiments on ZEST dataset and a synthetic SQuAD dataset demonstrate that HYPTER improves upon fine-tuning baselines. Notably, when using BART-Large as the main network, HYPTER brings 11.3% comparative improvement on ZEST dataset.[1]

1 Introduction

Pre-trained text-to-text models (Raffel et al., 2020; Lewis et al., 2020) provide a unified formulation and off-the-shelf weights for a variety of NLP tasks, such as question answering (Khashabi et al., 2020) and commonsense reasoning (Bosselut et al., 2019). In addition to their strong performance, text-to-text models naturally support generalizing to novel tasks, by incorporating task description as part of the source sequence and fine-tuning the model with (source, target) examples (Weller et al., 2020). At inference time, the model is required to perform unseen tasks with the source sequence containing new task descriptions.



(a) Zero-shot Learning from Task Description, ZEST dataset (Weller et al., 2020)

(b) Synthetic Version of SQuAD, (Rajpurkar et al., 2016)

Figure 1: Instead of learning from (source, target) examples, in this paper we study the problem of *learning from task descriptions* (Weller et al., 2020). The train set contains M tasks, and the i-th task contains N_i examples of (s, t) pairs in text format. During test time, the learned model is required to directly make inferences on a new task given a task description.

While this initial attempt shows positive results, there are two potential limitations for the direct fine-tuning approach. (1) Predictions can be sensitive to the task descriptions (or "prompts") that are heuristically designed (Jiang et al., 2020). Paraphrasing the task description may lead to performance downgrade. (2) The model still learns from individual (source, target) examples, instead of learning to solve tasks at a higher level, by explicitly taking multiple examples within a task as a whole (see Fig. 1). Meanwhile, applying existing zero-shot learning methods that supports task-level learning to text-to-text transformers is non-trivial. Methods designed specifically for classification problems, such as prototypical networks (Snell et al., 2017), cannot be directly applied to text-to-text models. Moreover, given the large size of text-to-text models, generating parameters for a whole model from the task description (Jin et al., 2020) is infeasible.

In this work, we follow the settings in (Weller et al., 2020) and aim to improve a model's generalization ability to unseen tasks by better incorporating task descriptions and using a task-level training procedure. We introduce HYPTER, a frame-

[1]Code and data can be found at https://github.com/INK-USC/hypter.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Short Papers), pages 646–653
August 1–6, 2021.

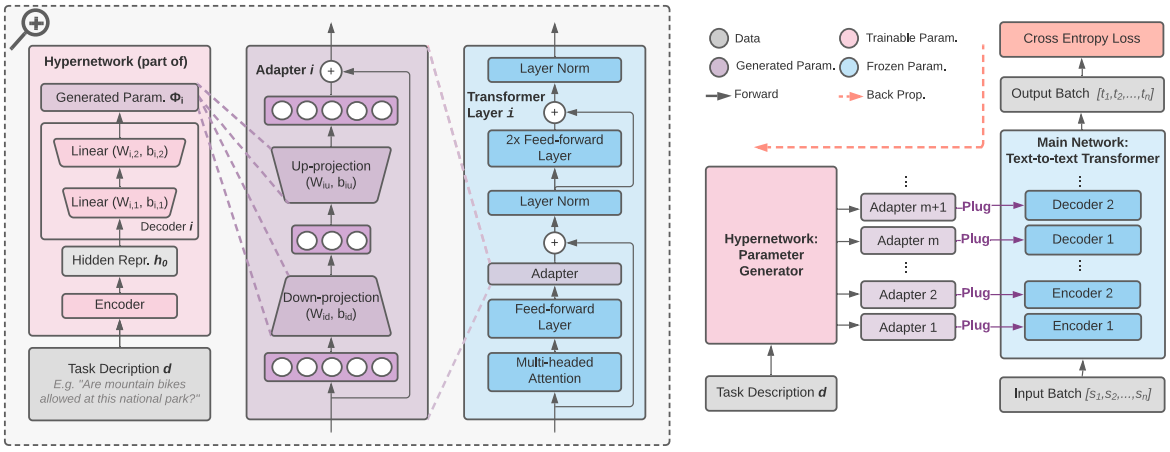


Figure 2: **Illustration of HYPTER Framework.** **Left:** A hypernetwork generates parameter ϕ_i for task-specific adapter i that is plugged to transformer layer i in the text-to-text model. **Right:** The adapted main network is evaluated on a task $(d, \mathcal{D})$. The final cross entropy loss is back-propagated to update the hypernetwork.

work that employs a hypernetwork (Ha et al., 2017) to dynamically generate task-specific parameters (*i.e.*, adapters) from task descriptions. Adapters (Houlsby et al., 2019) are light-weight modules that can be inserted into transformer layers for *parameter-efficient* adaptation. Such formulation also effectively enables learning at the *task* level, by learning to generate appropriate parameters for a task, and examine the model's competence on each task using multiple examples within that task. This is in contrast to learning at the *instance* level, by learning to generate the correct output for one specific input sequence.

We apply HYPTER to two datasets: ZEST (Weller et al., 2020) and a synthetic version of SQuAD (Rajpurkar et al., 2016). We demonstrate that HYPTER improves upon direct fine-tuning baselines. Notably, training with HYPTER achieves 0.45% absolute improvement (11.3% comparative improvement) in Competence@90 metric on ZEST, when BART-Large is used as the main network.

2 Problem Definition

We study the problem of *learning from task description* (Weller et al., 2020), and aim to improve models' competence on unseen tasks at the inference time. Formally, a task is denoted as a tuple of $(d, \mathcal{D})$, where d is the natural language description of the task, and $\mathcal{D} = \{(s_1, t_1), ..., (s_n, t_n)\}$ contains (source, target) examples of this task (See Fig. 1). In our text-to-text formulation, both s_i and t_i are text sequences. At train time, both d and $\mathcal{D}$ are available, while at test time, an unseen description d is given, and the model is expected to predict the correct t given input s without further training.

For instance, in the ZEST dataset (Weller et al., 2020), a train task description can be "*Are mountain bikes allowed at this national park?*", while $\mathcal{D}$ contains twenty paragraphs for different national parks and twenty corresponding answers. During test time, a novel task may be "*Are there fish in this national park that live in caves?*", and the model is asked to directly make inferences.

3 Background: Adapters

Our work is built on adapters (Houlsby et al., 2019), light-weight modules that can be placed into transformer layers for parameter-efficient transfer learning. In the original paper, the main model is frozen during training, while only layer norm and adapter parameters are learnable. In this paper, we adopt a simplified design compared to the original paper (see Fig. 2 (Left)) – In each transformer layer, exactly one adapter module will be added after the multi-headed attention. One adapter module contains two linear layers separated by an non-linearity activation layer. We use $(\mathbf{W}_{id}, \mathbf{b}_{id})$ to denote the down-projection parameters for the adapter in transformer layer i, and $(\mathbf{W}_{iu}, \mathbf{b}_{iu})$ for the up-projection parameters.

4 Method

Overview. Fig. 2 provides an illustration of our HYPTER framework. HYPTER has two major parts: (1) A main network, which is a pre-trained text-to-text model. We instantiate the main network with BART-Base/Large (Lewis et al., 2020). (2) A hyper-

network, which generates adapters to be plugged into the main network. Fig. 2 (Left) contains a detailed illustration of how adapter parameters are generated and how adapter layers are incorporated into one transformer layer.

Hypernetwork. The hypernetwork consists of an encoder and multiple decoders. The encoder maps the task description d to a latent representation $\mathbf{h}_0$, while the decoders use $\mathbf{h}_0$ to generate adapter parameters ϕ. In our work we instantiated the encoder with a RoBERTa-Base model (Liu et al., 2019), *i.e.*, $\mathbf{h}_0 = \text{RoBERTa}(d)$. For a text-to-text model with n transformer layers, the hypernetwork contains n decoders. Decoder i uses $\mathbf{h}_0$ as input, and outputs adapter parameters ϕ_i for transformer layer i, *i.e.*, $\mathbf{h}_{i,1} = \text{ReLU}(\mathbf{W}_{i,1}\mathbf{h_0} + \mathbf{b}_{i,1})$, $\phi_i = \mathbf{W}_{i,2}\mathbf{h}_{i,1} + \mathbf{b}_{i,2}$. Here $\mathbf{W}_{i,1}, \mathbf{b}_{i,1}, \mathbf{W}_{i,2}, \mathbf{b}_{i,2}$ are trainable parameters. The generated parameters ϕ_i are sliced and reshaped to become parameters $[\mathbf{W}_{id}, \mathbf{b}_{id}, \mathbf{W}_{iu}, \mathbf{b}_{iu}]$ used in the adapter i.

Model Training. We adopt a training schedule where we first train the main network, then train the hypernetwork while the main network is frozen. Conceptually, the first stage ensures that the main network captures the general ability across different tasks; the second stage allows the hypernetwork to learn to adapt the main network to a specific task. During the first stage the text-to-text model is fine-tuned with all $(\text{Concat}(d, s), t)$ examples in the training set. Here $\text{Concat}(d, s)$ means the concatenation of task description d and input s. The learned main network from this stage also serves as the baseline method.

During the second stage, we sample a task $(d, \mathcal{D})$ from the training set and sample a mini-batch of (s, t) examples from $\mathcal{D}$. Given a description d, the hypernetwork generates adapter parameters ϕ_i. We insert the resulting adapter layers into the main network, and compute the cross entropy loss L of generating t given input $\text{Concat}(d, s)$. The loss is end-to-end differentiable and is back-propagated to update the hypernetwork, while the main network is frozen. See Fig. 2 (Right) for illustration. This second stage of training effectively enables learning at the *task* level. The loss L characterizes the model's competence in the task $(d, \mathcal{D})$. Therefore, by optimizing L, the model is trained to *solve tasks*.

Model Inference. At test time the model is given an unseen task description d. The hypernetwork generates description-dependent adapter parameters, similar to the procedure during training. In this way, we obtain a model that is capable of making inferences for this new task.

5 Experiments

5.1 Experiment Setup

Datasets. We use two datasets that fit our setup. The first one is Zero-shot Learning from Task Descriptions dataset (ZEST, Weller et al. 2020), which formulates task descriptions as generalized questions, and provides multiple source-target examples for each question. The performance is evaluated with a novel metric: "Competence@K", along with mean F1 score. Competence@K is the percentage of all tasks for which the model achieves mean F1 score higher than K. For example, Competence@90=5 suggests that 5% of all tasks can be solved with mean F1 better than 90%. We report dev set performance, and hidden test set performance obtained from ZEST's official leaderboard.

We construct the second dataset from SQuAD v1 (Rajpurkar et al., 2016) to simulate the problem setting in this paper. We refer to this dataset as Synthetic SQuAD. Specifically, we construct tasks from the original SQuAD train set according to "question type", the bi-gram containing the central question word (*e.g.*, what, when). For example, "when does" questions are considered as a task, and "what country" questions are considered as another task. These bi-grams are used as "task descriptions". We select the 100 most frequent question types in SQuAD train set, and randomly subsample 64 examples from each type to formulate our dataset. We then randomly split the 100 types into 80/10/10 for train/dev/test. In addition, we select examples that fall into the 10 test question types from Natural Questions (Kwiatkowski et al., 2019) and NewsQA (Trischler et al., 2017), and use these as out-of-domain test examples. Performance is evaluated with mean F1. We include the list of question types and more details about this dataset in Appendix A.

Baseline. To demonstrate the efficacy of the HYPTER framework, we compare it to just its first half – the main text-to-text transformer model that we obtain after the first stage of training. This is identical to the fine-tuning baseline method in (Weller et al., 2020), and there are no other applicable baselines to the best of our knowledge.

Model	Mean-F1	C@75	C@90
Bart-Base	28.44 (±1.58)	5.76 (±2.10)	0.74 (±0.00)
+ HYPTER	**28.96** (±1.15)	**6.32** (±2.02)*	**1.08** (±0.62)
Bart-Large (reported)	40	13	8
Bart-Large	41.17 (±1.16)	15.74 (±2.16)	7.17 (±1.66)
+ HYPTER	**41.65** (±1.34)	**16.41** (±2.15)*	**7.62** (±1.66)*

Table 1: **Performance on ZEST Dev Set.** "C@75/90" refers to Competence@75/90 metric. We report mean and standard deviation over 7 runs. $*$ indicates statistical significance in a two-tailed paired t-test ($p < 0.05$).

Model	Mean-F1	C@75	C@90
Bart-Base	31.97	**7.03**	2.23
+ HYPTER	**32.32**	6.72	**2.53**
Bart-Large (reported)	37.93	11.19	3.96
Bart-Large	40.13	10.91	3.98
+ HYPTER	**40.41**	**11.35**	**4.43**

Table 2: **Performance on ZEST Test Set.** Performance obtained from ZEST official leaderboard[2].

Training Details. For each method, we train the model 7 times using different random seeds, and we report average and standard deviation. We discuss other training details, including hyperparameters, in Appendix B. Notably, we ensure all baseline models will not benefit from additional training, by tuning the number of epochs and using early stopping based on dev performance. This ensures the improvement brought by HYPTER is not due to additional training.

5.2 Results

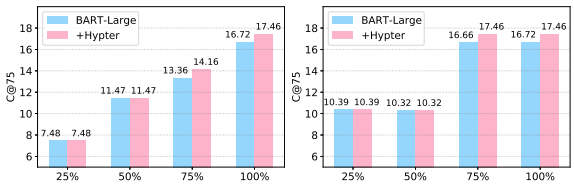
Main Results. We present the results for ZEST in Table 1-2 and results for Synthetic SQuAD in Table 3. On ZEST test set, we observe that the Competence@90 metric is improved from 3.98 to 4.43 when using BART-Large, yielding an 11.3% relative improvement. When BART-Base is used, C@90 is improved from 2.23 to 2.53. This demonstrates that by learning to solve tasks with HYPTER, the model's generalization ability to unseen tasks is improved. On Synthetic SQuAD dataset, we observe 0.74% improvement with BART-Base and 0.41% improvement with BART-Large. Additionally, models trained with HYPTER achieves comparable or better performance on out-of-domain test sets, suggesting the learned task-solving ability is generalizable to new test distribution.[3] It is a known issue that evaluating zero-shot performance can be tricky. We tried our best to reduce the randomness and instability by using different random seeds. In Table 1 and Table 3, we demonstrate that performance improvement is significant (p<0.05) in multiple settings, *e.g.*, on ZEST dev set when C@75 metric is used.

[2] https://leaderboard.allenai.org/zest/submissions/public

[3] Unexpectedly, in Table 3 we observe that performance of BART-Large on NewsQA is worse than that of BART-Base. We suspect that BART-Large may have overfit the SQuAD train set during the first stage of fine-tuning.

Model	SQuAD	NQ	NewsQA
Bart-Base	74.79 (±0.91)	49.78 (±0.95)	56.37 (±0.90)
+ HYPTER	**75.53** (±0.68)*	**50.39** (±1.01)*	**56.41** (±0.85)
Bart-Large	79.32 (±0.34)	59.21 (±0.89)	55.41 (±0.54)
+ HYPTER	**79.73** (±0.50)	**59.58** (±0.57)	**55.60** (±0.90)

Table 3: **Performance on Synthetic SQuAD dataset.** We report mean and standard deviation over 7 runs. NQ and NewsQA serve as out-of-domain test data.



(a) # Tasks (b) # Examples per Task

Figure 3: Competence@75 Performance on ZEST Dev when less training data is used.

Model Behavior Analysis on ZEST. ZEST dataset provides a comprehensive analysis protocol by splitting tasks into different generalization types (base, paraphrase, composition, semantic flips, and output structure) and defining four error types (recall, precision, partial, and other). Compared to the BART-Large fine-tuning baseline, our model achieves better performance in "base" and "paraphrase" categories in the ZEST official test set. We also manually inspected dev set predictions produced by the baseline and our model. We found the predictions corrected by our method span across the four error types. In particular, the proposed method flipped two "n/a" predictions into the correct answers in the task "Which royalty was this dog breed popular with?" ("base" category), reducing the recall errors and improving the competence metric. We do not observe more granular model behavioral patterns beyond this point.

Study of Data Efficiency. We study whether HYPTER is effective when trained with (1) fewer tasks, while the number of examples per task is unchanged; (2) fewer examples per task, while the number of total tasks is kept constant. We experiment with ZEST and BART-Large, and show the performance in Fig. 3. We observe that HYPTER is effective when trained with 75%/100% tasks, but does not improve performance with fewer tasks. This is reasonable since HYPTER learns at the *task*

level (taking one task as an "example"), and 50% of the tasks may be insufficient. We also observe performance improvement with 75%/100% examples per task, but not with fewer examples. This suggests sufficient number of examples per task is necessary for HYPTER to generate effective adapters.

6 Related Work

Zero-shot Learning with Transformers. Zero-shot learning (ZSL) has been explored for various NLP tasks, including text classification (Yin et al., 2019), entity linking (Logeswaran et al., 2019) and entity typing (Obeidat et al., 2019). Several works study cross-task transfer by unifying the input-output format, *e.g.*, relation extraction as machine reading comprehension (Levy et al., 2017), named entity recognition as machine reading comprehension (Li et al., 2020). Such formulation allows generalization to unseen relation or named entity types at test time. Learning from task descriptions (Weller et al., 2020) and instructions (Mishra et al., 2021) can be considered as a sub-category in zero-shot learning, with the goal of generalizing to unseen tasks during inference.

Adapters for Transformers. Houlsby et al. (2019) proposed adapter layers for parameter-efficient transfer learning in NLP. Adapter layers, which adopt a bottleneck architecture with two linear layers, are added after each multi-headed attention layer and each feed-foward layer in a pre-trained transformer. Adapters have been recently applied to multi-lingual settings, with successes in NER, QA and commonsense reasoning (Pfeiffer et al., 2020; Philip et al., 2020; Artetxe et al., 2020).

Hypernetworks and Contextual Parameter Generators. Hypernetwork (Ha et al., 2017) is a broad concept of "using one network to generate the weights for another network". This concept has been broadly applied to visual reasoning (Perez et al., 2018), zero-shot image classification (Jin et al., 2020), etc. Closely related to our work, UDapter (Üstün et al., 2020) studies multilingual dependency parsing by generating adapter parameters. Our work is more generalizable as we do not restrict task format (dependency parsing v.s. general text-to-text tasks) or relations between subtasks (cross-lingual tasks v.s. tasks with text-form descriptions).

7 Conclusion

In this paper, we introduced HYPTER, a framework to improve text-to-text transformer's generalization ability to unseen tasks. HYPTER enhances task-specific abilities by inserting adapters generated with a hypernetwork, meanwhile it maintains the model's general task-solving ability by freezing main model parameters. We demonstrated the effectiveness of HYPTER on two datasets. Future work may explore teaching models with compositional instructions using HYPTER, or propose robust fine-tuning methods that help the model generalize to unseen data. It is also necessary to construct a large dataset of diverse NLP tasks to facilitate future research in this direction.

Acknowledgments

This research is supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Contract No. 2019-19051600007, the DARPA MCS program under Contract No. N660011924033 with the United States Office Of Naval Research, the Defense Advanced Research Projects Agency with award W911NF-19-20271, and NSF SMA 18-29268. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. We would like to thank anonymous reviewers and collaborators in USC INK research lab for their constructive feedback.

References

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019

shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13, Hong Kong, China. Association for Computational Linguistics.

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799, Long Beach, California, USA. PMLR.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Tian Jin, Zhun Liu, Shengjia Yan, Alexandre Eichenberger, and Louis-Philippe Morency. 2020. Language to network: Conditional parameter adaptation with natural language descriptions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6994–7007, Online. Association for Computational Linguistics.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3449–3460, Florence, Italy. Association for Computational Linguistics.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hanna Hajishirzi. 2021. Natural instructions: Benchmarking generalization to new tasks from natural language instructions. *ArXiv*, abs/2104.08773.

Rasha Obeidat, Xiaoli Fern, Hamed Shahbazi, and Prasad Tadepalli. 2019. Description-based zero-shot fine-grained entity typing. 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)*, pages 807–814, Minneapolis, Minnesota. Association for Computational Linguistics.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3942–3951. AAAI Press.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Jerin Philip, Alexandre Berard, Matthias Gallé, and Laurent Besacier. 2020. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages

4465–4470, Online. Association for Computational Linguistics.

Colin Raffel, Noam 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. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4077–4087.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online. Association for Computational Linguistics.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 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.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3914–3923, Hong Kong, China. Association for Computational Linguistics.

A Dataset Details

ZEST. ZEST dataset is released at https://ai2-datasets.s3-us-west-2.amazonaws.com/zest/zest.zip. ZEST leaderboard is hosted at https://leaderboard.allenai.org/zest/submissions/public.

Synthetic SQuAD. We build our synthetic dataset from the processed version of SQuAD, Natural Questions and NewsQA in MRQA Shared Task 2019 (Fisch et al., 2019) (https://mrqa.github.io/2019/). We provide the script to reconstruct the data we use in our released code. We list the bigrams we use to formulate synthetic tasks and their train/dev/test partition in Listing 1.

Listing 1: Train/Dev/Test Partition in Synthetic SQuAD dataset.

```
"train": ["why were", "what years", "who said", "what percent", "when did", "where do", "who is", "how are", "what decade", "how does", "how long", "where was", "what has", "which two", "who was", "who were", "where are", "where does", "what did", "how far", "what organization", "what does", "what group", "what would", "how did", "who has", "who created", "how many", "what name", "what types", "what two", "which city", "who are", "how is", "what event", "what are", "what century", "what area", "whom did", "why was", "who wrote", "why are", "where is", "how old", "when is", "what caused", "who did", "where did", "what happened", "what state", "what kind", "what time", "what famous", "what's the", "what day", "what is", "what company", "what were", "why do", "what new", "what date", "what do", "what color", "which group", "what country", "how can", "what have", "where can", "what period", "which year", "when was", "what other", "what happens", "was the", "what was", "which of", "when were", "what sort", "what city", "what year"],
"dev": ["what month", "why is", "what part", "what term", "how was", "how were", "how do", "who led", "which country", "when does"],
"test": ["where were", "what political", "what religion", "why did", "what type", "what language", "who had", "what percentage", "what can", "how much"]
```

B Training Details

We use transformers (Wolf et al., 2020) for all our experiments. All experiments are done with one single GPU. We use NVIDIA Quadro RTX 8000, NVIDIA Quadro RTX 6000, or NVIDIA GeForce RTX 2080 Ti, depending on availability.

For text-to-text model fine-tuning, we select learning rate from {1e-5, 3e-5, 5e-5}, and select the total number of epochs from {5, 10, 15, 20, 30} for ZEST and {10, 20, 30, 50, 100} for synthetic SQuAD. We use a fixed batch size of 32.

For hypernetwork training, we train up to 100 epochs (one epoch here refers to an iteration over all tasks). We update the hypernetwork every b tasks, and we call b as task batch size. When learning from one task, we sample b' examples within this task, and we call b' as the example batch size. We greedily and sequentially select adapter width d from {4,8,16,32}, learning rate α from {3e-6, 1e-5, 3e-5, 1e-4}, b from {4,8,16,32}, b' from {4,8,16,32}, based on dev set performance.

C Additional Baseline

Another reasonable baseline is to fine-tune a text-to-text model together with randomly initialized adapters plugged in it. We experiment with this method using BART-Large and list the performance in Table 4. We do not observe significant differences between the two methods (p=0.8840 for C@75, p=0.8118 for C@90 in two-tailed paired t-test).

Model	Mean-F1	C@75	C@90
Bart-Large	41.17 (±1.16)	15.74 (±2.16)	7.17 (±1.66)
Bart-Large with Adapters	39.76 (±1.26)	15.61 (±1.14)	6.96 (±1.15)

Table 4: Performance comparison when adapters are plugged / not plugged during fine-tuning.

D Dev Set Performance of Models Submitted to ZEST Leaderboard

In Table 5 we present the dev performance of models submitted to the leaderboard. The submitted models are the "first-runs" in the 7-run series, as we add the 7-run experiments and significance test later on, following a reviewer's suggestion.

Model	Mean-F1	C@75	C@90
Bart-Base	29.72	7.87	**4.05**
+ HYPTER	**29.81**	**8.67**	**4.05**
Bart-Large (reported)	40	13	8
Bart-Large	42.10	16.72	8.85
+ HYPTER	**43.50**	**17.46**	**9.64**

Table 5: Dev set performance of models submitted to ZEST leaderboard.

E Discussion

It is worth noting that the efficacy of HYPTER is at the cost of introducing new parameters in the hypernetwork. To generate adapter parameters, more parameters are introduced and trained in the hypernetwork. One may achieve better generalization ability to unseen tasks with larger pre-trained models with billions of parameters. In this case, we consider HYPTER as an alternative by augmenting a medium-sized pre-trained model with a hypernetwork. Meanwhile, we highlight our contribution to be the concept of generating task-specific adapters from descriptions and HYPTER's task-level training procedure.