Learning to Generate Task-Specific Adapters from Task Description

Qinyuan Ye

Xiang Ren

University of Southern California - Information Sciences Institution

INK Lab @ USC-ISI

inklab.usc.edu

Problem Setting

- Learning from Task Descriptions (Weller et al. 2020)
  - A task is defined as a tuple of (d, D)
  - d is the task description
  - $D = \{(s_1, t_1), (s_2, t_2), \dots, (s_n, t_n)\}$ contains (source, target) examples of that task

task description d

What accessibility services does this national park offer?

source s

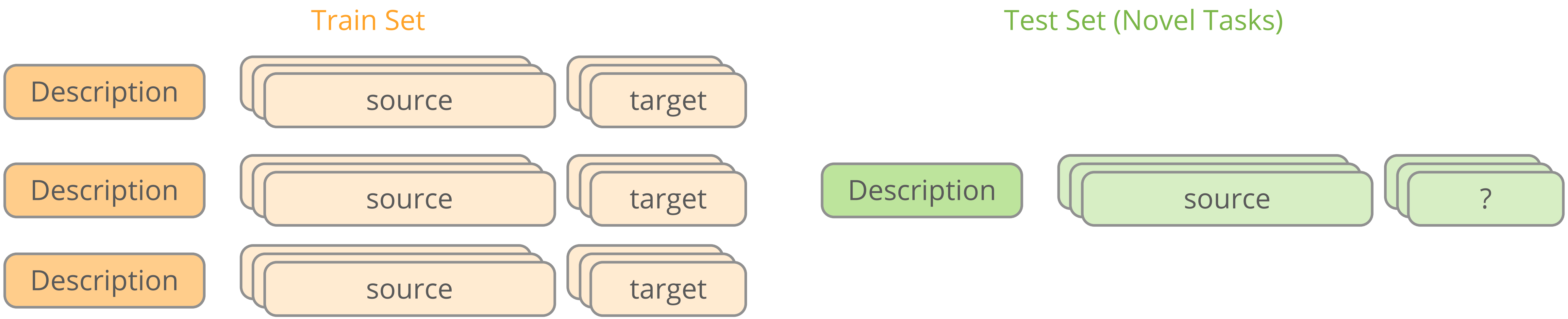
Select picnic tables and vault toilets along Dunes Drive are also wheelchair-accessible.

target t

wheelchair-accessible

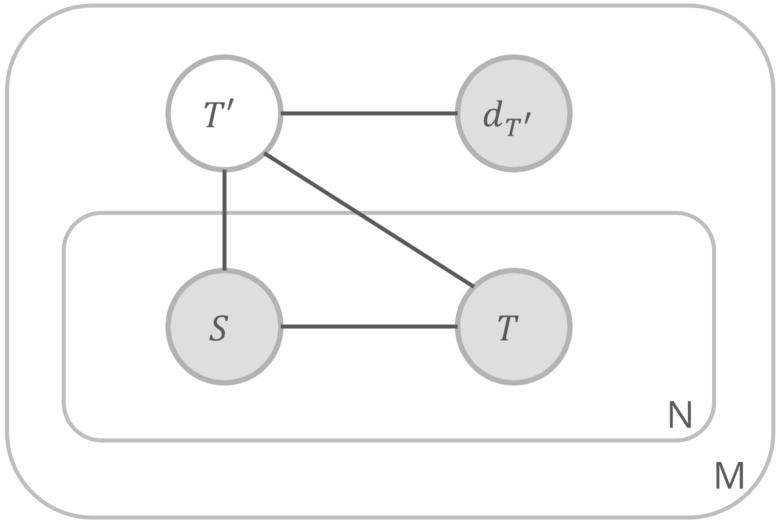
Problem Setting

- Learning from Task Descriptions ([Weller et al. 2020](#))
  - A task is defined as a tuple of (d, D)
  - During training time, a set of train tasks, with both d and D, are available.
  - During test time, an unseen description d is given, and the model is expected to predict the correct t given the input s, without further training.

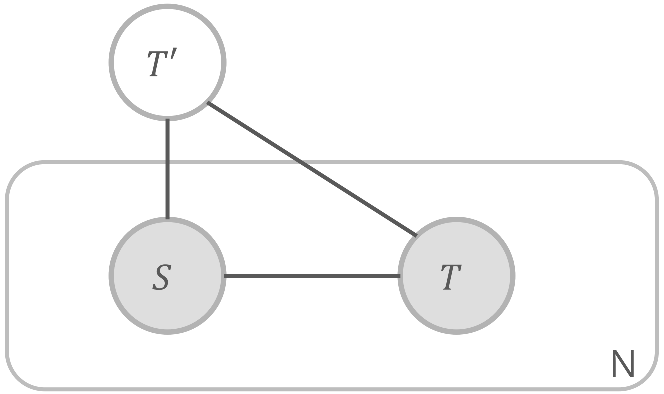


Problem Setting

Learning from Task Descriptions



Learning from Examples



Prior Work and Limitation

- Solution 1: Include task description in the input sequence (*i.e.*, using the description as a “prompt”)

task description

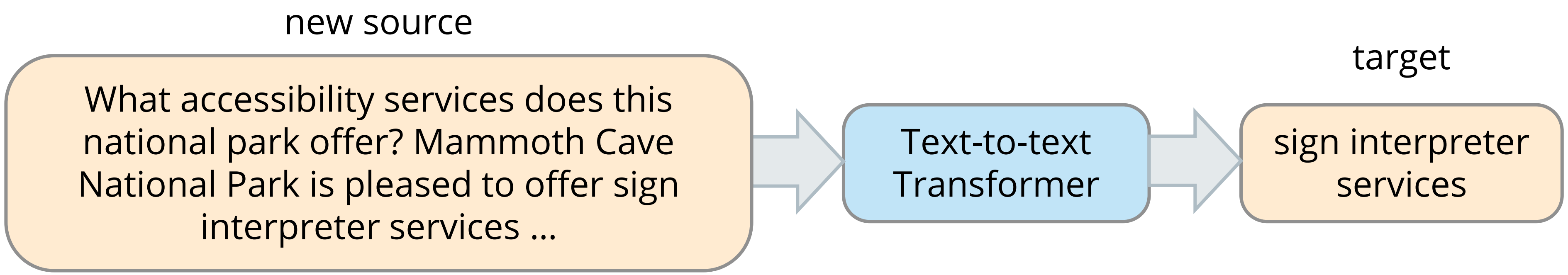
What accessibility services does this national park offer?

source

Mammoth Cave National Park is pleased to offer sign interpreter services …

Prior Work and Limitation

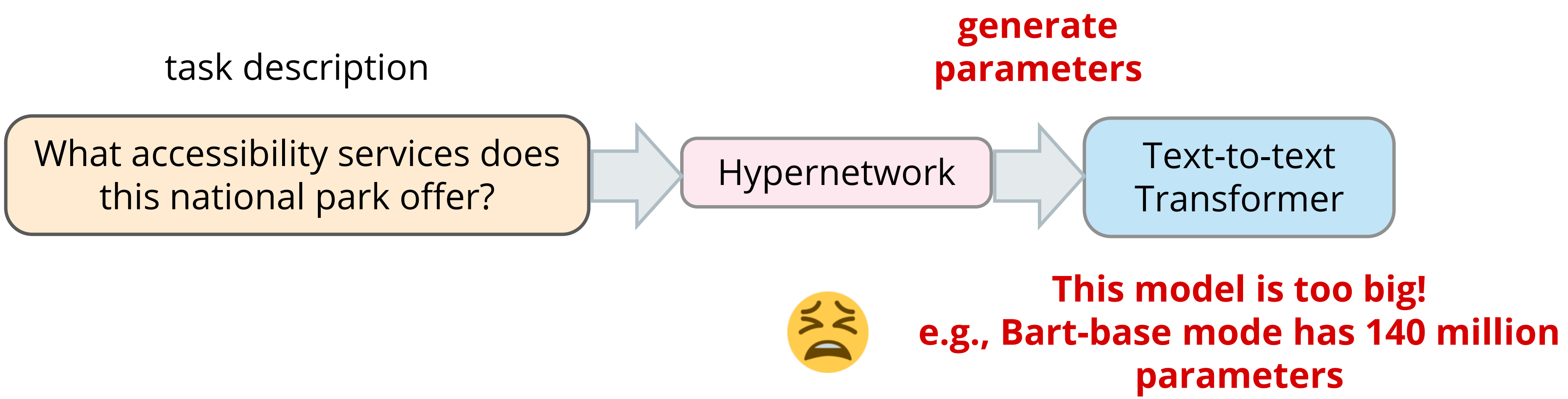
- Solution 1: Include task description in the input sequence (*i.e.*, using the description as a "prompt")



Reducing the "learning from task description" problem back to a "learning from examples" problem?!

Prior Work and Limitation

- Solution 2: Apply existing zero-shot learning methods to the model
    - Our task cannot use classical zero-shot learning methods such as **prototypical network** (Snell et al., 2017).
    - **Hypernetwork** approaches that generates all model parameters (Jin et al., 2020) is infeasible.



Prior Work and Limitation

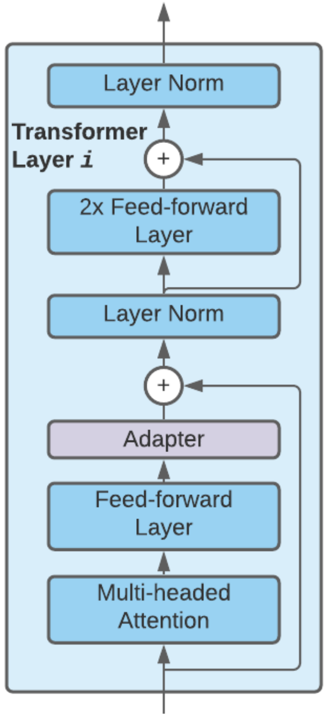
- Solution 1: Using task description as “prompt”
  - ✔ Straight-forward
  - ✕ The model is still “learning from examples”
- Solution 2: Hypernetwork approaches that generate model parameters
  - ✔ Indeed “learning from task descriptions”
  - ✕ Text-to-text transformers are too large to generate

prompt-based model + generate only a few task-specific parameters?

Hypter: use a hypernetwork to generate adapter layers

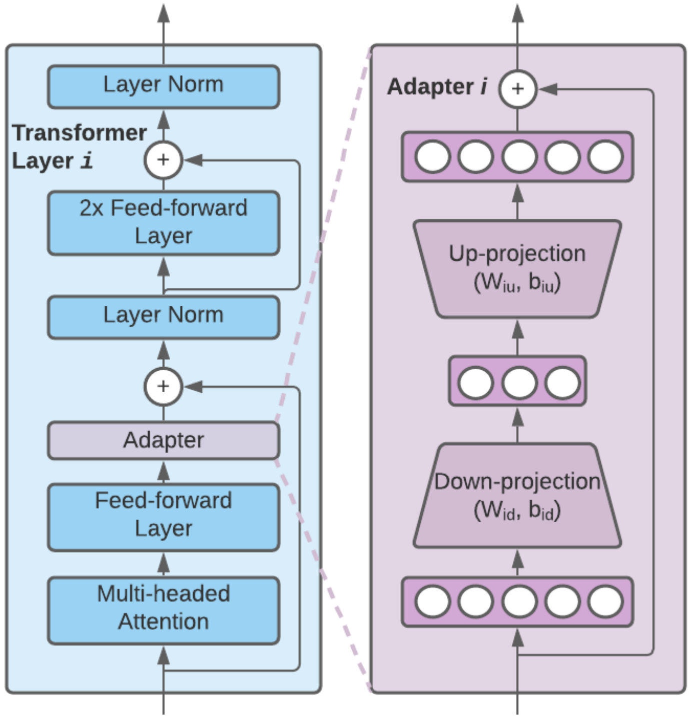
Background: Adapters

- Adapters ([Houlsby et al., 2019]) are light-weight modules that can be inserted into transformer layers for parameter-efficient transfer learning.



Background: Adapters

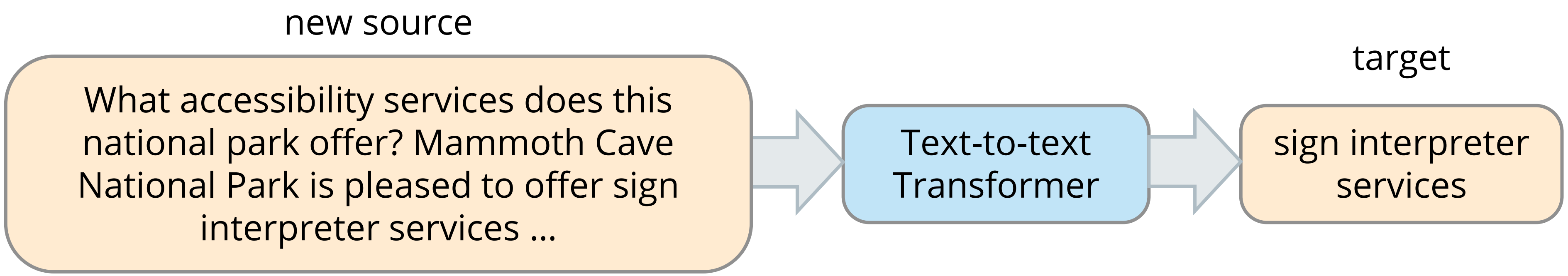
- **Adapters (Houlsby et al., 2019) are light-weight modules that can be inserted into transformer layers for parameter-efficient transfer learning.**
- One adapter module has one down-projection layer and one up-projection layer.
- During learning, the main network is frozen, while only adapter parameters are trainable.
- E.g., adding 2% extra adapter parameters to BERT and fine-tuning the model on SQuAD will give 90.4% F1 score.



** We used a simpler design compared to the original paper.*

Hypter: Using a Hypernetwork to Generate Adapters

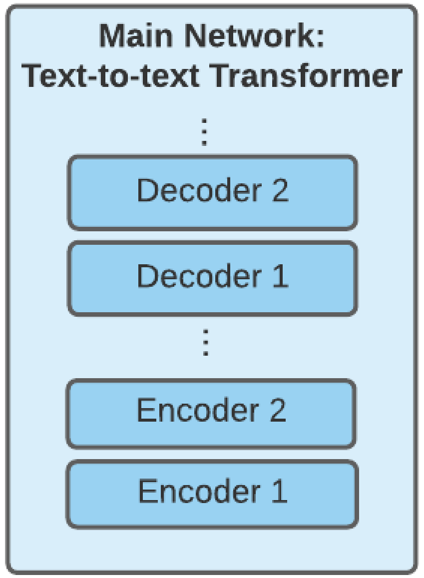
Stage 1: Text-to-text Model Fine-tuning (the same as Solution 1)



Hypter: Using a Hypernetwork to Generate Adapters

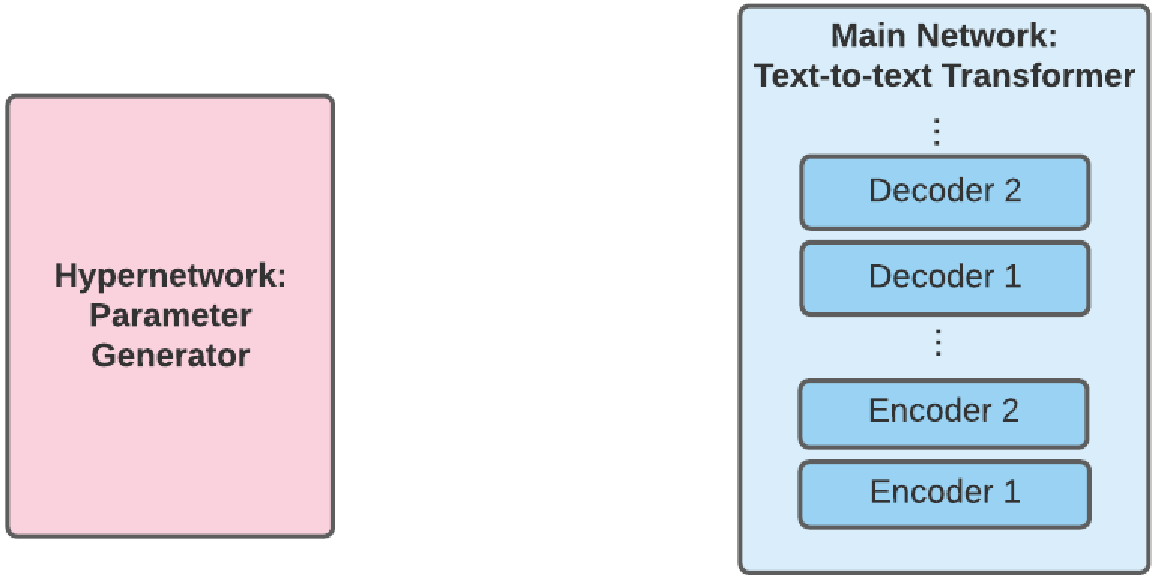
Stage 2: Train a hypernetwork to generate task-specific adapters using task description

Model from Stage 1



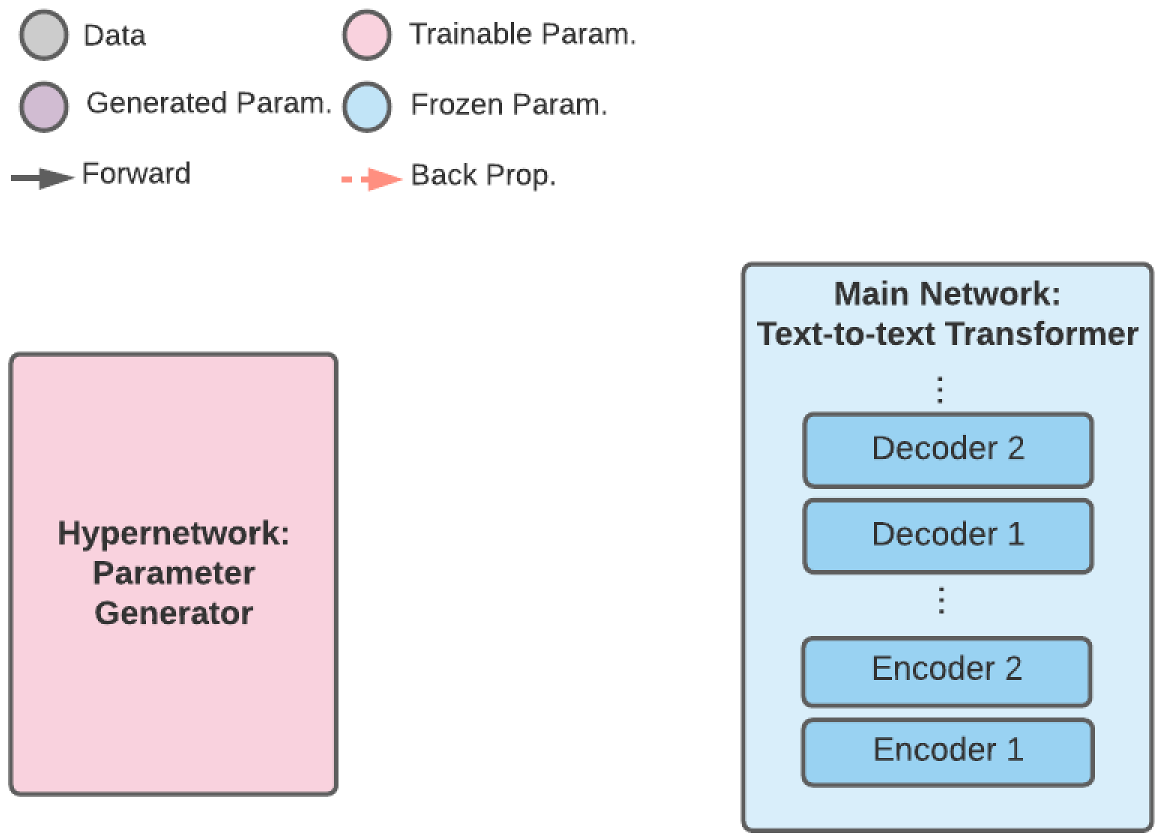
Hypter: Using a Hypernetwork to Generate Adapters

Stage 2: Train a hypernetwork to generate task-specific adapters using task description



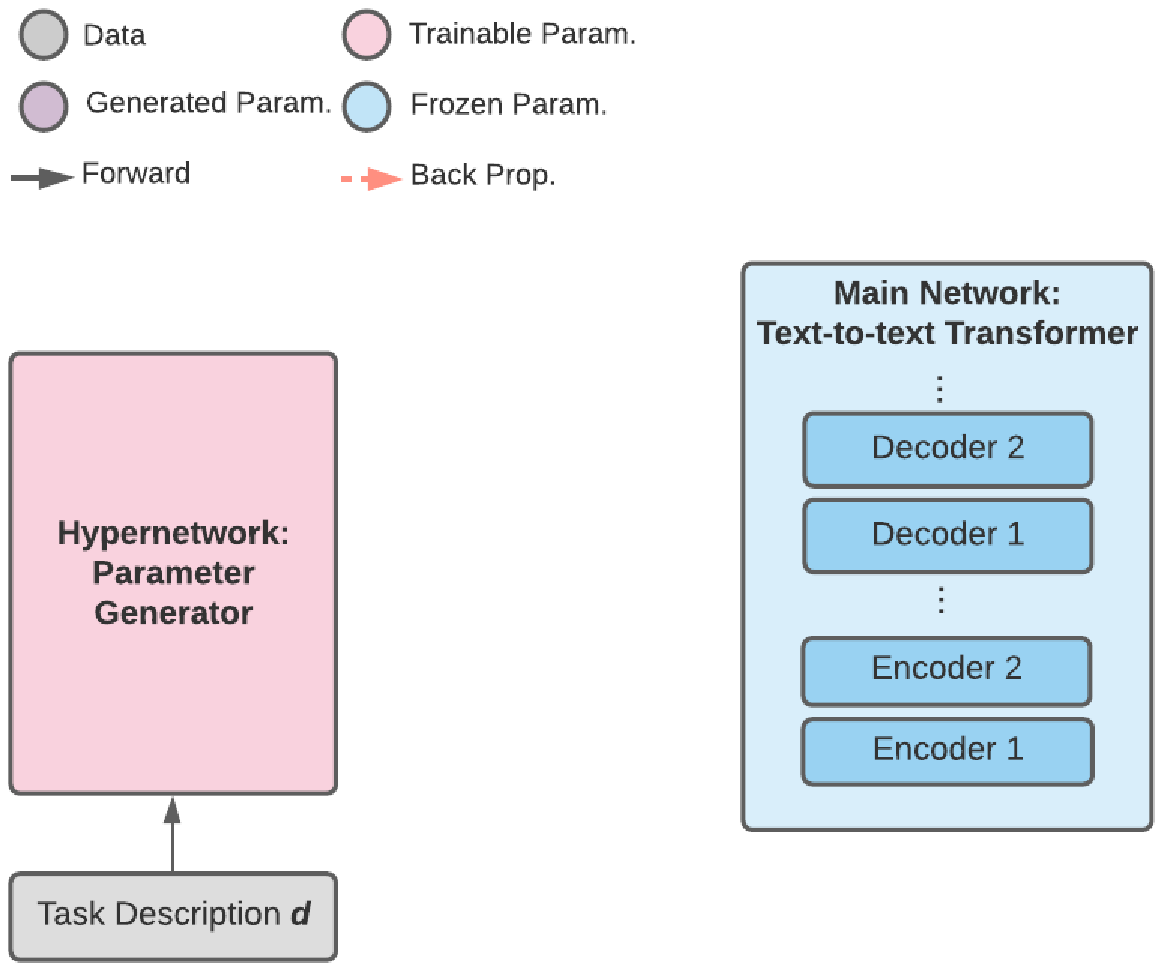
Hypter: Using a Hypernetwork to Generate Adapters

Stage 2: Train a hypernetwork to generate task-specific adapters using task description



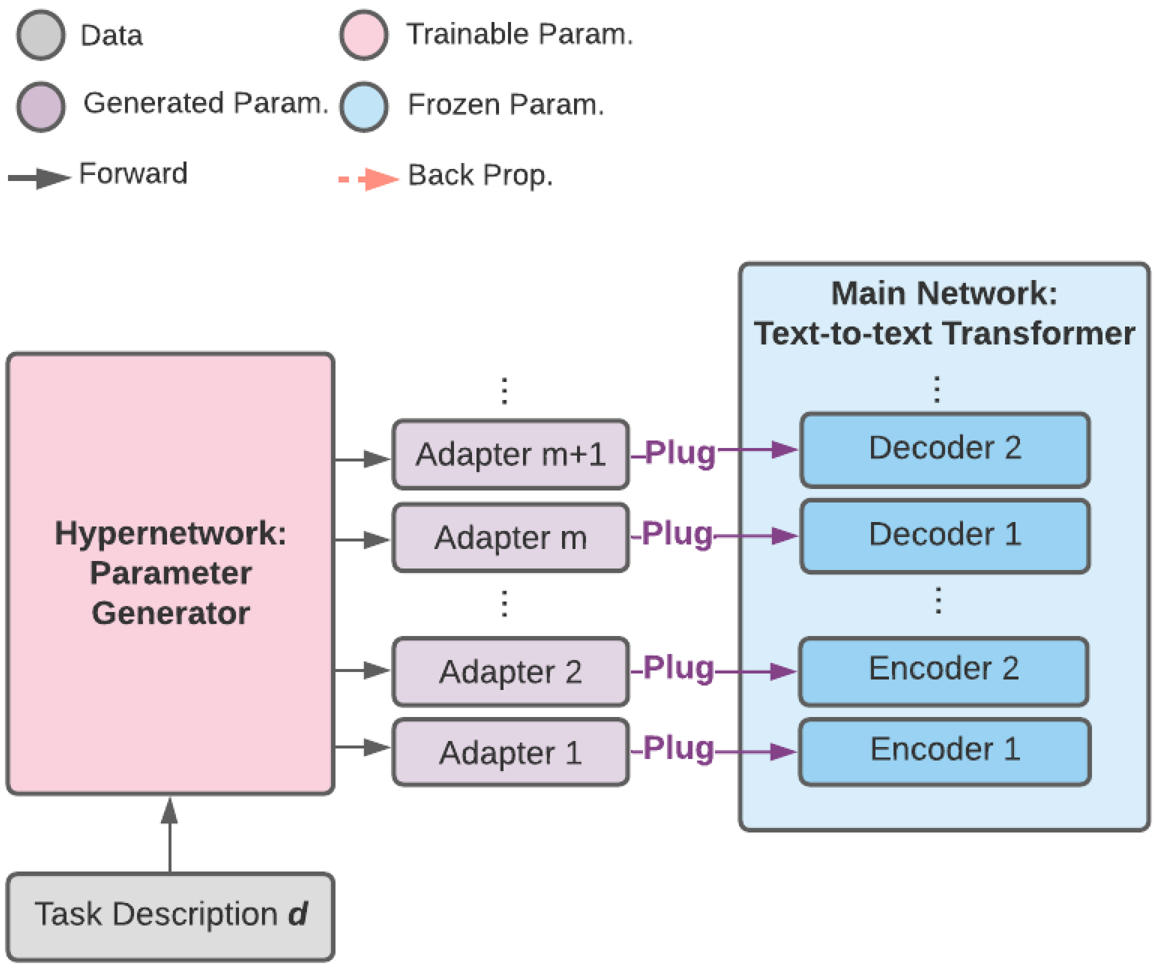
Hypter: Using a Hypernetwork to Generate Adapters

Stage 2: Train a hypernetwork to generate task-specific adapters using task description



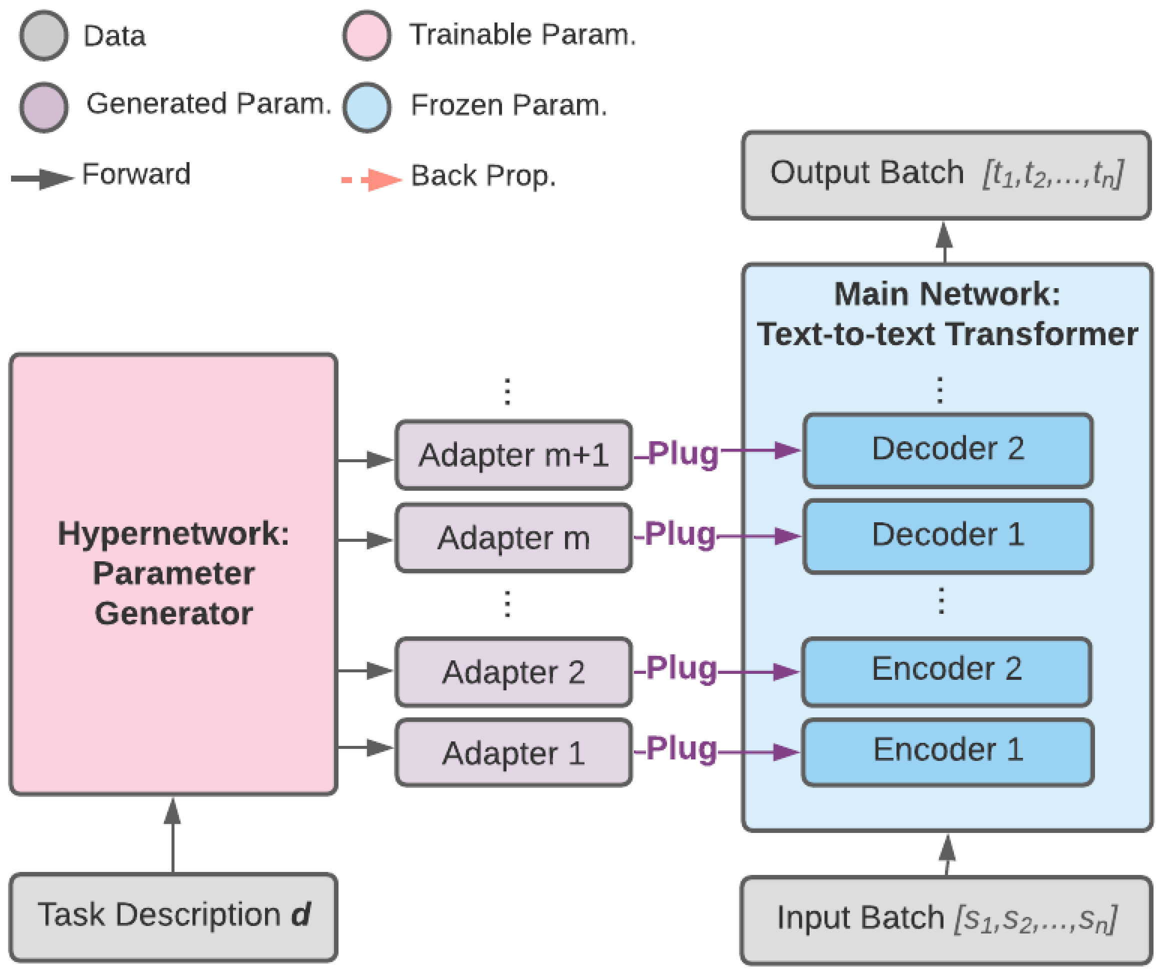
Hypter: Using a Hypernetwork to Generate Adapters

Stage 2: Train a hypernetwork to generate task-specific adapters using task description



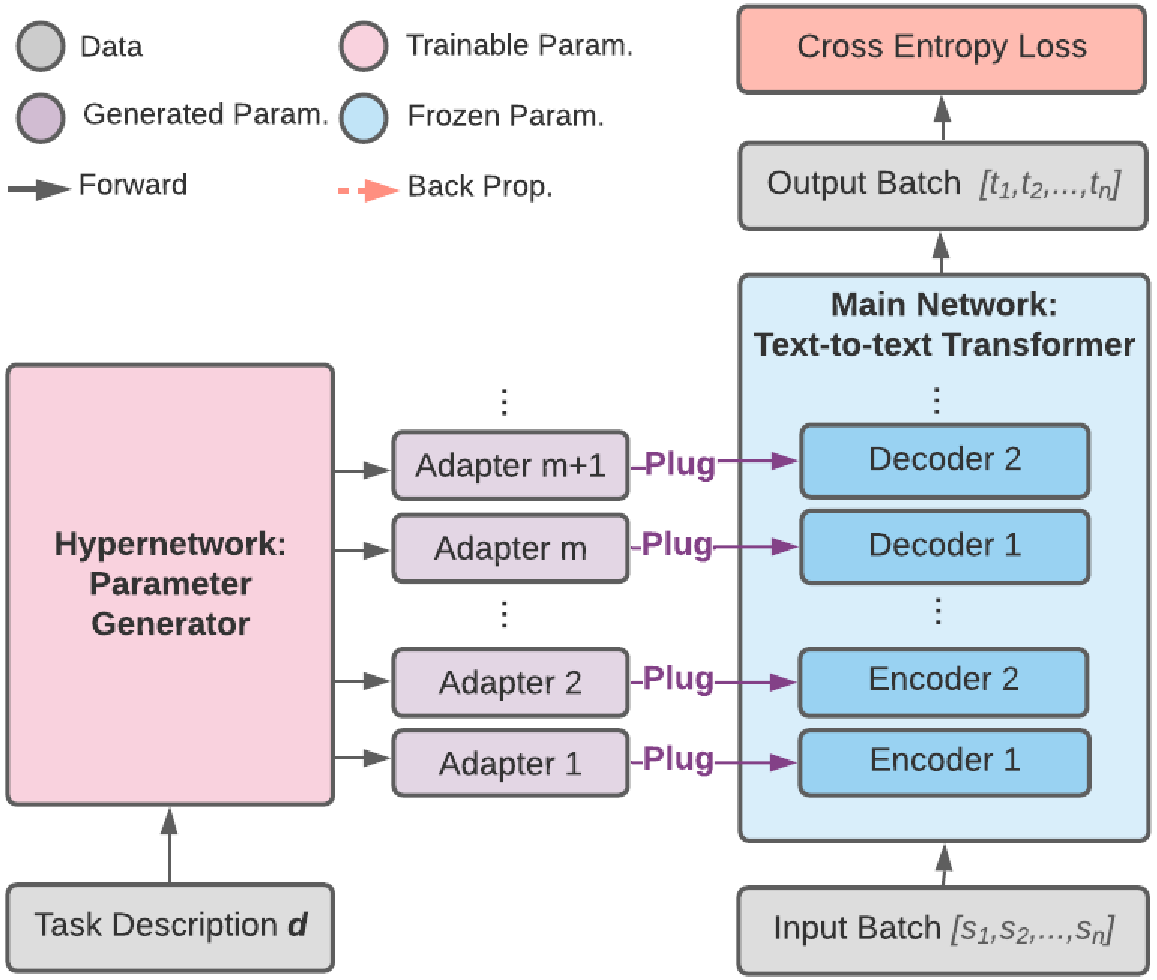
Hypter: Using a Hypernetwork to Generate Adapters

Stage 2: Train a hypernetwork to generate task-specific adapters using task description



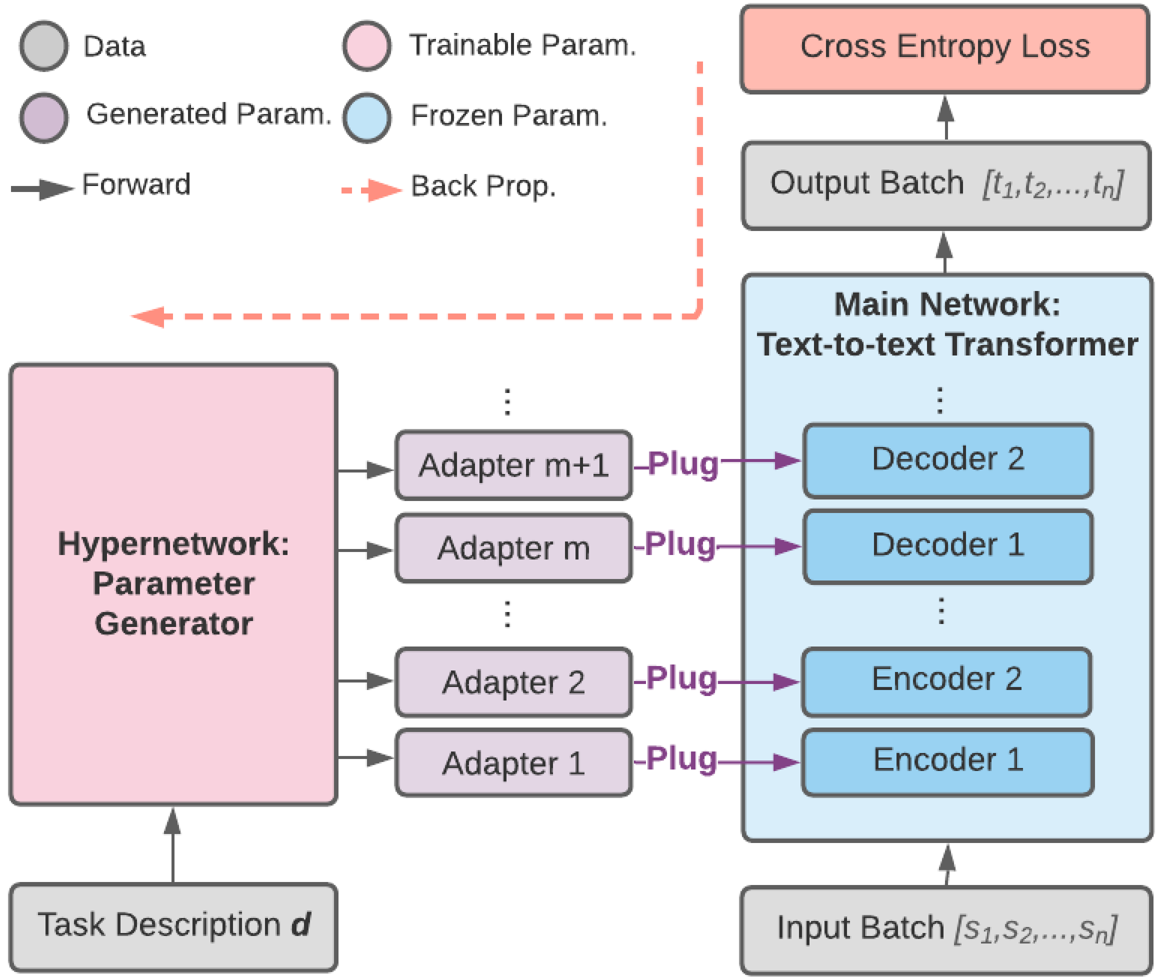
Hypter: Using a Hypernetwork to Generate Adapters

Stage 2: Train a hypernetwork to generate task-specific adapters using task description



Hypter: Using a Hypernetwork to Generate Adapters

Stage 2: Train a hypernetwork to generate task-specific adapters using task description



Datasets

- Zero-shot Learning from Task Descriptions, ZEST (Weller et al. 2020)
  - Task descriptions are generalized questions, e.g., “what accessibility services does this national park offer?”
  - Evaluate with Competence@K metric, and mean F1 score.

Train Set	Test Set
Are fires ever restricted at this national park?	What accessibility services does this national park offer?
Are there always spots on this dog breed?	Are there any controversial actions that this president approved of?
Did this president deal with any illnesses as a child?	Did humans cross breed this dog breed into existence?

Datasets

- Repurposed SQuAD Dataset
  - We repurpose the SQuAD dataset (Rajpurkar et al, 2016) to simulate our problem setting.
  - We construct tasks according to “question type”, the bigram containing the question word.
  - All “who said” questions are considered as a task, and “who said” is the task description.
  - We additionally use NewsQA (Trischler et al., 2017) and Natural Questions (Kwiatkowski et al., 2019) examples as out-of-domain test examples.

Train Set			Test Set		
why were	what years	who said	where were	what religion	how much

Results

ZEST Dataset (Official Test Set)

	C@75	C@90
Bart-Large	10.91	3.98
+ Hypter	11.35 (+4%)	4.43 (+11%)

Better generalization to novel tasks!

Repurposed SQuAD Dataset (Test Set)

	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

** Indicates statistical significance in a two-tailed paired t-test (p<0.05)*

Better generalization to out-of-domain inputs!

Conclusions

- **We introduced Hypter, a framework ...**
  - to improve text-to-text transformer's generalization ability to unseen tasks
  - by generating adapter parameters using a hypernetwork and enabling task-level learning

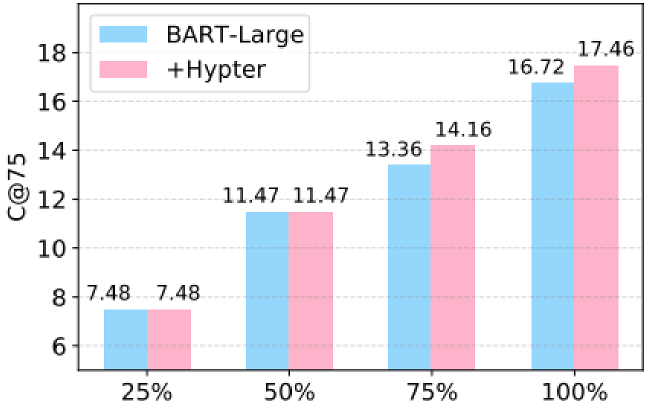
- **Hypter can ...**
  - bring up to 11.3% comparative improvement on ZEST test set
  - demonstrate better generalization to out-of-domain inputs

- **Checkout our code at https://github.com/INK-USC/hypter 😃**

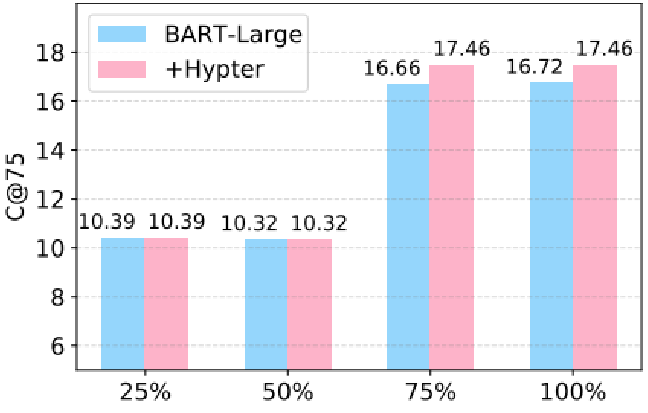
Analysis

Backup Slides

Taking ZEST train dataset and controlling …
(a) number of tasks (b) examples per task



(a) # Tasks



(b) # Examples per Task

Sufficient number of tasks and sufficient number of examples in each task are both necessary.

Concurrent Work

Backup Slides

- **Prefix-tuning** (Li et al., 2021), **Prompt-tuning** (Lester et al., 2021)
  - New alternatives to adapters that are more parameter-efficient and powerful!
  - We can adapt our approach to train a hypernetwork to generate soft prefixes/prompts.
- **HyperFormer** (Mahabadi et al., 2021)
  - Generate adapter parameters based on task, adapter position and layer ID.
  - Enable information sharing across tasks.
  - Improved performance on GLUE tasks while using only 0.29% parameters per task!
- **Natural Instructions** (Mishra et al., 2021)
  - Instructions, prompts and descriptions are very similar concepts.
  - Natural Instructions provides a more comprehensive and realistic testbed for “learning from task description” problem.