Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)

On Conditional and Compositional Language Model Differentiable Prompting

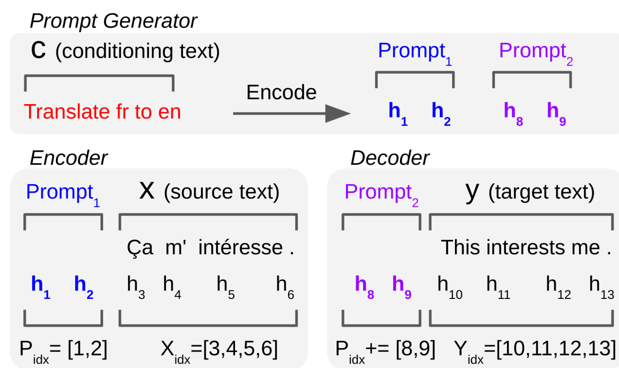
Jonathan Pilault[1]*, Can Liu[2], Mohit Bansal[3], Markus Dreyer[2]
[1]Mila - Québec AI Institute, Polytechnique Montréal
[2]Amazon Alexa,
[3]University of North Carolina at Chapel Hill

Abstract

Prompts have been shown to be an effective method to adapt a frozen Pretrained Language Model (PLM) to perform well on downstream tasks. Prompts can be represented by a human-engineered word sequence or by a learned continuous embedding. In this work, we investigate conditional and compositional differentiable prompting. We propose a new model, Prompt Production System (PROPS), which learns to *transform* task instructions or input metadata, into continuous prompts that elicit task-specific outputs from the PLM. Our model uses a modular network structure based on our neural formulation of Production Systems, which allows the model to learn discrete *rules* – neural functions that learn to specialize in transforming particular prompt input patterns, making it suitable for compositional transfer learning and few-shot learning. We present extensive empirical and theoretical analysis and show that PROPS consistently surpasses other PLM adaptation techniques, and often improves upon fully fine-tuned models, on compositional generalization tasks, controllable summarization and multilingual translation, while needing fewer trainable parameters.

1 Introduction

Humans have a remarkable ability to solve cognitive tasks by following instructions and using prior relevant experiences. Can a machine do the same? Recent research has shown that textual task instructions [Efrat and Levy, 2020] appended to a Pretrained Language Model's (PLM) input can yield successful results in several NLP tasks such as classification [Schick and Schütze, 2021], image captioning [Tsimpoukelli *et al.*, 2021] and question-answering [Mishra *et al.*, 2021] *without fine-tuning the full model*. However, PLMs do not always accurately grasp the meaning of textual prompts [Mishra *et al.*, 2021] and often display high sensitivity to the word formulation [Jiang *et al.*, 2020] of instructions. Further, unless the model exceeds billions of parameters, full-model fine-tuning still typically outperforms human-engineered prompts for frozen PLMs [Lester *et al.*, 2021]. To mitigate such issues, differentiable and continuous prompting techniques [Li and Liang, 2021] have emerged as a viable alternative (illustrated in Figure 1a). However, current techniques optimize prompts solely based on the task learning objective without leveraging information embedded in textual prompts.



(a) Prepending generated differentiable prompts.

Figure 1: PROPS is a differentiable and conditional prompt generator that outputs a sequence of vectors that is prepended to PLM hidden states at positions $\in P_{\text{idx}}$ as seen in (a).

We argue that simple task instructions (e.g., "Translate English to French"), along with other already available or easily extractable metadata, such as the text category of an input sample (e.g., "sports"), can be valuable conditioning context for a model to *generate* a continuous prompt. The prompt would be specific to the particular task and input, and would elicit task- and input-specific output from a frozen PLM. We propose a model that can, given a PLM, be trained to *transform* task- and input-specific text into conditional continuous prompt embeddings. The advantage of using text conditioning as input to a continuous prompt generator is that our model can, once trained, repurpose previously unseen instructions from previously trained tasks, without having to retrain a PLM. By sharing aspects in the task instructions that were observed during training of related tasks, the continuous prompts of new tasks are more suitable for transfer learning and few-shot learning. With our method, we seek to combine the advantages of simple task descriptions, which provide useful con-

*Work completed during an internship at Amazon Research.

text, and learned continuous prompts, which work well even with smaller PLMs. In addition, we propose to enhance our model's compositional ability for few-shot and transfer learning by using a modular network structure, based on our own neural formulation of Production Systems (ProdSys, [Newell, 1972]).

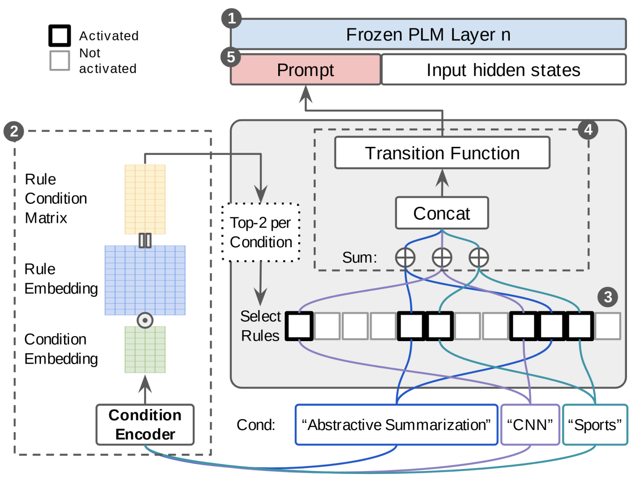


Figure 2: PROPS architecture with top-k=2 rule selection as an example. "Cond"= conditioning text, example inputs to PROPS.

ProdSys has several desirable properties. First, the ability to reason in a propositional format (if - then) allows the disentanglement and *modularization of knowledge* about a task or input sample. Second, using the fact that propositions are discrete and independent from each other, the system can combine productions and *compose acquired knowledge*. We hypothesize that such properties are desirable to improve generalization, transfer of learning, and reusability of prior task descriptions when generating prompts. ProdSys models symbolic knowledge in the form of "IF" (condition or situation or task) - "THEN" (action or instructions or response) rules, called productions, and a set of mechanisms for matching and applying such rules. Generally, ProdSys contains four basic components [Nilsson, 2010]: ① a long-term memory, a knowledge base that contains general information independent of the current situation/task and which persists over time and that may be used as a condition; ② a working memory that contains temporary information about the current situation or task and that also operates as a sparse rule selector; ③ a rule memory that contains a collection of productions; ④ an inference engine that maps a condition C_j to an instruction I_i in the form $C_j \xrightarrow{R_i} I_j$, for selected productions R_i.

We draw a parallel between the process of generating continuous prompts (differentiable instructions for the PLM) from textual conditioning (conditions) and rule productions. Specifically, as shown in Fig. 2: a frozen PLM can be used as a persistent knowledge base [Petroni *et al.*, 2019] or ① a long-term memory[1]; a Condition Encoder that maps textual conditioning sequences to a fixed-sized vector to create the Rule-Condition matrix works similarly to ② a working memory by encoding and sparsely matching conditions and rules; the selected differentiable rules, the summation operation and transition function allows the model to map a condition to differentiable instructions and work as ③ a rule memory and ④ an inference engine; finally, instructions ⑤ are, in our case, differentiable prompts that we prepend to a frozen PLM's hidden states. Such a system, which we call Prompt Production System (PROPS), shares the four main building blocks of ProdSys, which allows it to conditionally generate prompts while compartmentalizing neural propositional knowledge via its condition-rule selection mechanism. As we will discover, such mechanism equips PROPS with an ability to reuse and re-purpose knowledge from examples of the same tasks or instructions of intermediate and jointly trained tasks that allow the model to better control text generation. This construction allows PROPS to surpass all other prompt generator baselines, and often fully fine-tuned models, on summarization, translation, and semantic parsing benchmarks.

Our contributions are the following:

- We provide strong Conditional Prompt Generator baselines (Section 3.1) on multiple tasks and datasets, showing the advantage of conditioning continuous prompts.
- We propose a new architecture, called PROPS, a prompt generator based on ProdSys (Section 3.2).
- We demonstrate PROPS 's ability to compose knowledge in the four settings described above in both the full-data regime and low-resource settings.
- To better understand PROPS, we provide theoretical and empirical analysis in Appendix D.

We evaluate our methods in four situations as illustrated in Fig. 3 to check if the following can be composed: (1) input text from training set input text segments in Section 4.2, (2) tasks when knowledge is transferred between all tasks during training in Section 4.3, (3) metadata about the input data in Section 4.4, (4) all of the above combined in Section 4.5.

2 Related Works

Differentiable Prompting has seen increased interest lately due to the method's ease of use and compelling performance. Unlike prior work, PROPS is a conditional prompt generator. There are two differentiable prompting methods that are close to PROPS. The first is PTR [Han *et al.*, 2021] that breaks an input into a series of predicates of first order logic. However, unlike PROPS, sub-prompts are manually designed and PTR is mainly useful for classification tasks. Concurrently to our work, PGT [Li *et al.*, 2022] also attempts to transfer prompts learned from multiple representative support text generation tasks (e.g. paraphrasing, question answering). Instead of using an attention mechanism to select task-specific prior prompts, PROPS compartmentalizes and reconstructs prior prompt knowledge using differentiable rules, allowing module sharing across tasks. Using fewer support tasks, PROPS outperforms PGT on XSum on the fully-supervised and few-shot learning settings.

Modular Networks are a type of neural architecture that splits computation into specialized and independent neural

[1]The neural long-term memory is used in two ways: the conditions can be hidden states from the previous PLM layer and the gradient updates are dependent of the PLM pretrained weights.

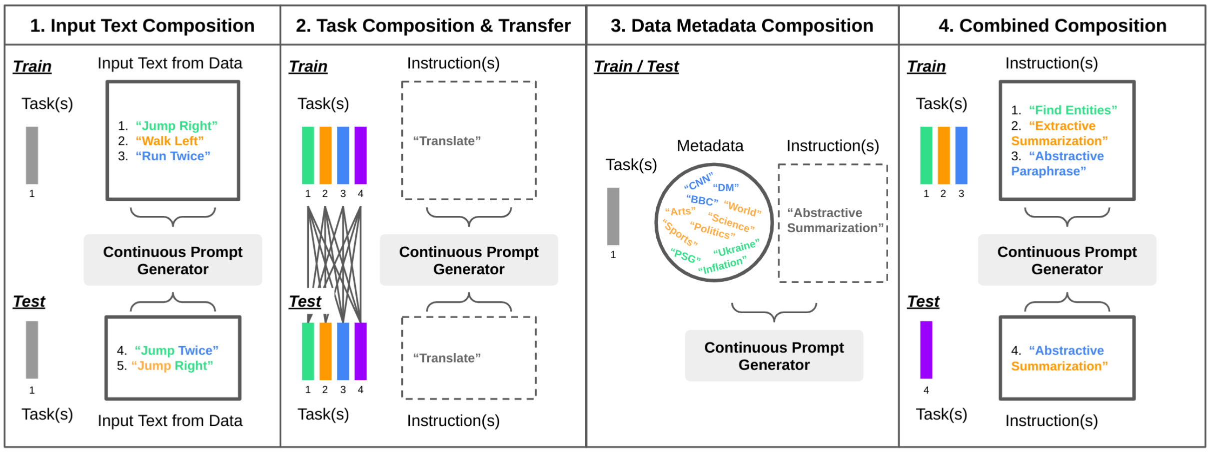


Figure 3: Outline of the capabilities investigated by training composable inputs, tasks and/or metadata and tested on different combinations. The **colored** artifacts in the solid line circles or squares represent information that can be composed for controlled text generation.

sub-systems (i.e., modules). Modular Networks borrow from conditional computation methods, where modules are chosen dynamically given a certain criterion. Such module specialization has been shown empirically to improve generalization to unseen inputs [Kirsch *et al.*, 2018] or tasks [Goyal *et al.*, 2021]. In a multitask learning setting, several recent approaches have leveraged modularity to decompose knowledge [Ostapenko and Charlin, 2021] such that it could be reused for a new task. Similar to Ponti *et al.*, our method also assumes that tasks can be decomposed into a set of modules that are shareable and reusable across tasks. PROPS differs in three ways: (1) we use both tasks and inputs to condition computation; (2) sub-module selection is informed by external information (text conditioning); (3) we use differentiable prompts on a PLM instead of adapters.

3 Methodology

We now present the two components on which our method is based. We will first provide an overview of Conditional Prompt Generators (CPG), linking conditional continuous prompts and adapters. Then, we discuss PROPS, our novel construction of CPGs using Neural Production Systems.

3.1 Conditional Prompt Generators (CPG)

Prompt Generators such as Prefix-Tuning [Li and Liang, 2021], Prompt-Tuning [Lester *et al.*, 2021] and P-Tuning [Liu *et al.*, 2021] are all *Unconditional Prompt Generation (UPG)* methods used to adapt PLMs. Generally, prompt vectors $\boldsymbol{P}_k, \boldsymbol{P}_v \in \mathbb{R}^{T_P \times d}$, of length T_P and dimension d, are concatenated to the key $\boldsymbol{K}$ and value $\boldsymbol{V}$ vectors to modulate the frozen PLM attention output of each head H. We get for a query $\boldsymbol{Q}_t$ at step t:

$$\begin{aligned} H &= \text{Attn}\big(\boldsymbol{Q}_t, (\boldsymbol{P}_k \| \boldsymbol{K}), (\boldsymbol{P}_v \| \boldsymbol{V})\big) \\ &= \sigma\big(\boldsymbol{Q}_t (\boldsymbol{P}_k \| \boldsymbol{K})^\top\big)(\boldsymbol{P}_v \| \boldsymbol{V}) \\ &= \alpha \cdot \sigma(\boldsymbol{Q}_t \boldsymbol{P}_k^\top)\boldsymbol{P}_v + (1-\alpha) \cdot \sigma(\boldsymbol{Q}_t \boldsymbol{K}^\top)\boldsymbol{V} \\ &= \alpha \cdot \underbrace{\text{Attn}(\boldsymbol{Q}_t, \boldsymbol{P}_k, \boldsymbol{P}_v)}_{\text{Learnable task adaptation}} + (1-\alpha) \cdot \underbrace{\text{Attn}(\boldsymbol{Q}_t, \boldsymbol{K}, \boldsymbol{V})}_{\text{Frozen PLM attention}}, \end{aligned} \tag{1}$$

where σ is the softmax function and $\alpha = \alpha(\boldsymbol{Q}_t, \boldsymbol{K}, \boldsymbol{P}_k)$ is a learnable gating function:

$$\alpha = \frac{\sum_i \exp(\boldsymbol{Q}_t \boldsymbol{P}_k^\top)_i}{\sum_i \exp(\boldsymbol{Q}_t \boldsymbol{P}_k^\top)_i + \sum_j \exp(\boldsymbol{Q}_t \boldsymbol{K}^\top)_j}. \tag{2}$$

Conditional Prompt Generation, as in PROPS, is achieved by plugging condition-dependent prompt vectors $[\boldsymbol{P}_k, \boldsymbol{P}_v] = [\boldsymbol{P}_k(c_t), \boldsymbol{P}_v(c_t)] = \boldsymbol{P}(c_t)$ in Equation 1, where $\boldsymbol{P}_k, \boldsymbol{P}_v$ is created by splitting the last dimension of $\boldsymbol{P}(\boldsymbol{c}_t) \in \mathbb{R}^{T_P \times 2d}$ in two and $\boldsymbol{c}_t$ is the word embedding at the t^{th} condition token.

	Setting	Conditioning text (C) examples	Task
1	Input Composition	"{Look, Opposite, Left, Thrice}" + +"{and}"+"{Jump, Right, Thrice}"	cg
2	Task Composition	"Translate fr to en"+ [French Text], "Translate es to en"+ [Spanish Text]	nmt
3	Metadata Composition	"{cnn, dm, bbc}" + "{sports, politics, business, local, ...}"	sum
4	Instructions, tasks, metadata	"{Find Entities, Paraphrase, Summarization}" + [metadata]	sum

Table 1: Conditioning text inputted to the prompt generator. **cg** = compositional generalization, **nmt** = multilingual translation and **sum** = summarization.

Conditions $C \in \mathcal{C}$ represent posterior knowledge about a task or dataset as well as instructions. The Conditions or conditioning text are inputs to the CPGs. We provide examples in Table 1 following the different evaluation settings depicted in Fig 3. In certain cases, it can be the input data itself as in the compositional generalization task (e.g.: input "walk right twice and walk left" has the output "turn_right walk turn_right walk turn_left walk"). The conditioning text is easily extractable. For news summarization, we extracted the news outlet and the article type. For the topic-focused news summarization, additional inputs such as the topic is provided with the dataset. Before going through a word embedding layer, each condition is a textual description or an instruction about an example, a type of an example, a task or a dataset.

Instructions were written by the authors. See Appendix A for more details.

3.2 Prompt Production Systems (PROPS)

PROPS is inspired by Neural Production Systems (NPS, Goyal and Bengio), the deep learning version of ProdSys. NPS is an end-to-end model that constructs object- or condition-centric representations on which independent learnable production rules are applied. PROPS also has independent learnable production rules but there are three main differences with NPS. First, rules are applied to sub-sequences of variable lengths instead of a single vector. Second, PROPS is tied to a PLM through gradient updates and the appendage of prompt vectors to the input hidden states whereas NPS was not built to interact with a PLM. Third, our rule selection mechanism has a different design that allows rule overlap amongst conditions of a single instance. Unlike NPS that uses MLPs, our rules are attention heads [Vaswani *et al.*, 2017]. PROPS consists of N separately differentiable rules, $\{\boldsymbol{R}_1, \boldsymbol{R}_2, .., \boldsymbol{R}_N\}$ that are each represented by $\boldsymbol{R}_i = (\vec{\boldsymbol{R}}_i, \widetilde{\mathrm{H}}_i)$, where $\vec{\boldsymbol{R}}_i$ is a rule embedding vector and $\widetilde{\mathrm{H}}_i$ is an attention head. As summarized in Algorithm 1, each condition sequence $\mathcal{S}_C = \langle \boldsymbol{c}_t | t \in \{1, \ldots, T_C\} \rangle_C$, where T_C is the sequence length of condition $C \in \mathcal{C}$ and $\mathcal{C}$ is the set of conditions, is first encoded by a Condition Encoder $f(\cdot)$. We use a light-weight attentive max pooling layer [Wu *et al.*, 2020] to encode condition sequences into a fix sized representation. Each condition vector $\vec{\boldsymbol{C}}$ is then concatenated together to create a Condition Embedding matrix $[\boldsymbol{E}] \in \mathbb{R}^{|\mathcal{C}| \times d}$, where d is the embedding dimension. Similarly, the learnable Rule Embedding matrix $[\boldsymbol{R}] \in \mathbb{R}^{n \times d}$, for n rules (i.e., attention heads), is created by concatenating each rule representation $\vec{\boldsymbol{R}}_i$. The rule-condition matrix is formed from the dot product operation $[\boldsymbol{E}] \cdot [\boldsymbol{R}]^\top \in \mathbb{R}^{|\mathcal{C}| \times n}$ and provides a template that maps a condition to a set of rules. We use a Gumbel Top-k, our extension[2] of the Gumbel Softmax , to choose k rules out of n for any of the conditions in $\mathcal{C}$. As discussed in Section 4.5, the ability to choose rules allows PROPS to compose modules. Typically $k = \frac{n}{2} > 1$ is the best choice of chosen rules as it allows a maximum combination of sub-modules (see Appendix D.1 and D.2 for in-depth theoretical and empirical analysis). The rest of the steps are similar to a Transformer layer's. However, instead of using all attention heads $\widetilde{\mathrm{H}}_{1 \ldots n}$ to transform a sequence, condition sub-sequences use a varying subset of k heads. At step t, our prompt vector $\boldsymbol{P}_t$ is generated in the following way:

$$\boldsymbol{P}_t(\boldsymbol{c}_t) = \boldsymbol{W}^o \sum_{r \in \{r\}} \widetilde{\mathrm{H}}_r(\mathrm{LN}(\boldsymbol{c}_t)), \tag{3}$$

where LN is layer normalization , $\boldsymbol{c}_t$ is the condition word embedding at step t, $g(\cdot)$ is the transition function (i.e., intermediate Transformer layer), $\boldsymbol{W}^o$ is a learnable weight matrix, $\|$ is the concatenation operator, $\{r\}$ is the set of indices identifying selected rules $\boldsymbol{P}_t$ can be applied L times for an L layer PROPS. Note that each PROPS *layer shares weights* to allow greater parameter efficiency and knowledge reusability.

[2]The argmax of a Gumbel softmax would only point to one rule. To select multiple rules we apply a differentiable top-k operator.

4 Main Results

In the next sections, we discuss our experimental setup and evaluation results across the four settings seen in Fig. 3 for both full data and low resource settings. Raffel *et al.* observe significant gaps between full-model fine-tuning (FFT) and adapters [Houlsby *et al.*, 2019; Bapna and Firat, 2019] on both summarization and translation tasks. The gap increases with increasing training data size. Recently, several methods claimed results on-par with FFT on NLU benchmarks [Guo *et al.*, 2021; Hu *et al.*, 2021] such as GLUE [Wang *et al.*, 2018] or on relatively simple CNLG tasks such as E2E [Novikova *et al.*, 2017]. However, our results indicate that such PLM adaptation may not generalize well across various architectures and for more complex tasks, high-resource benchmarks or large multi-task datasets. As demonstrated in this section (details below), PROPS outperforms all other model adaptation techniques on **three high-resource benchmarks, two datasets in the low resource knowledge transfer setting and one high resource multilingual translation dataset**.

4.1 General Setup

Model Name	adapted layers	condition n/a	condition emb	condition txt	shared weights	select rule
ADAPTER [1]	100%	√				
CA-MTL [2]	50%		√			
PREFIX [3]	100%					
PREFIX++ *	100%			√		
TRSF-P *	33%			√		
S-TRSF-P *	33%			√	√	
PROPS	33%			√	√	√

Table 2: Baselines. [1][Lin *et al.*, 2020], [2][Pilault *et al.*, 2021], [3][Li and Liang, 2021]. *Our implementation. hid=hidden states; emb=task embedding; txt=textual input.

Baselines: We compare PROPS to four main model categories: fully fine-tuned (FFT) models, adapters where trainable layers are inserted between frozen PLM layers, UPG and CPG described in Section 3. A comparison of baselines, including Transformer-Prompt (TRSF-P) and Transformer-Prompt with shared weights (S-TRSF-P) generators, is found in Table 2 where we show the percentage of PLM layers adapted, whether the PLM hidden states or the attention is adapted, the types of conditions used (none, embeddings or text prompts). The ADAPTER model uses trainable MLPs inserted in the PLM to transform hidden states. CA-MTL is type of multitask adapter that uses task embedding to conditionally adapt hidden states and the attention of the PLM. PREFIX++ is exactly the same as PREFIX however we additionally append conditions textual prompt to the input text (e.g.: "Abstractive Summarization, BBC, Sports" or "Jump Twice Jump Right"). The baselines allow us to measure the effects of conditions (PREFIX vs. TRSF-P or PREFIX++), the affects of sharing weights (TRSF-P vs. S-TRSF-P) and the affects of rule selection (S-TRSF-P vs PROPS). We also provide side by side comparisons of PREFIX, S-TRSF-P and PROPS in Figure 8 of the Appendix. More details are found in Appendix G.

Datasets, Training and Evaluation: We study four Conditional Natural Language Generation (CNLG) datasets SCAN [Lake and Baroni, 2018], Europarl [Koehn, 2005], XSum [Narayan *et al.*, 2018] and Topic-CNN-DM [Mrini *et al.*, 2021a]. The datasets and tasks for our experiment were chosen since text conditioning from metadata and instructions is similar to controllable text generation. We describe our datasets, training and evaluation setup in Appendix E. *The code and datasets will be made publicly available.*

4.2 Can CPG Modularization Better Compose Instructions by Composing the Input Text?

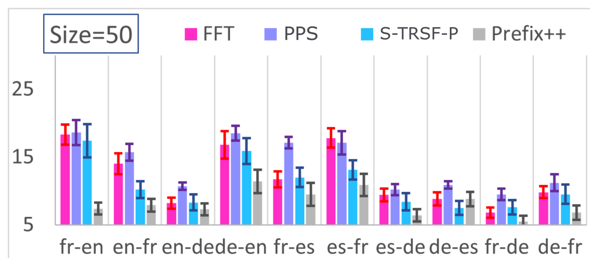
Model	Add Jump	Add Turn Left	Jump Around Right	Length	Avg. MCD
T5 [1]	98.3	69.2	99.9	5.2	10.1
PREFIX	83.8	66.7	91.0	5.9	7.8
PREFIX++	85.1	68.1	92.3	6.4	8.5
S-TRSF-P	90.9	70.3	93.6	9.3	9.9
PROPS	**99.2**	**72.9**	**100**	**11.4**	**12.6**

Table 3: Compositional Generalization test accuracy from exact match on SCAN. [1]Results from Furrer *et al.*.

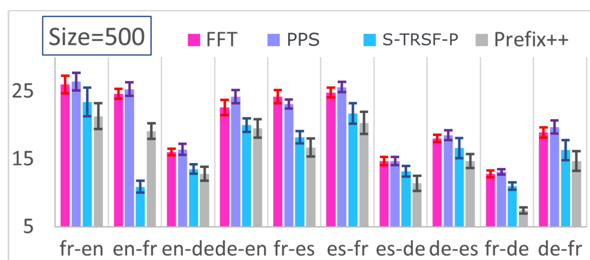
In the first type of evaluation depicted in Fig. 3, we test whether modular CPGs such as PROPS can compose instructions from the input. We fist want to test if the model learns from training inputs such as "jump right" and "walk left", how to properly resolve unseen combinations "jump left" or "walk right" at test time. We use a compositional generalization task made from the synthetic dataset SCAN [Lake and Baroni, 2018]. The SCAN dataset is constructed from a rule-based grammar that explicitly composes primitive commands. Lake and Baroni proposed various data splits to measure *Compositional Generalization* by testing on commands constructed with new primitives, excluded complex command sequences or extended length (length generalization). All adaptation techniques are built on top of a pretrained T5 model. In the "Add Jump" split for example, the training set includes all of the compositional tasks excluding "Jump", which is only seen in isolation, and the testing set includes all compositions using "Jump". The inductive bias in PROPS allows it to modularize knowledge via a sparse selection of attention heads. Experimental results in Table 3 show that PROPS's internal knowledge structure is also composable. Not only does PROPS surpass all other adaptation methods across splits by a large margin, **it also improves over T5 FFT on five out of seven cases**, including on the more difficult Maximum Compound Divergence (MCD) datasets [Keysers *et al.*, 2020]. We follow the training setup of Furrer *et al.*.

4.3 Can CPG Modularization Better Transfer and Compose Jointly Trained Tasks?

Now, we turn our attention to the second evaluation case in Fig. 3: task composition. We hypothesize that modular CPG networks such as PROPS will better segregate and compose knowledge from other tasks trained jointly. We select the Europarl [Koehn, 2005] multilingual dataset since it contains numerous parallel language corpora. The Europarl multilingual corpora is also interesting since our instructions can leverage "bridge tasks". For example, we can translate English to French (en $\rightarrow$ fr) at test time if we only trained on English to German (en $\rightarrow$ de) and German to French (de $\rightarrow$ fr) pairs (i.e., translate en to de and then de to fr to get en to fr). Further, it was shown that multilingual translation can benefit from positive transfer from one language (task) to the other. As previously discussed, it is typically easier to outperform fully-finetuned models with parameter efficient adaptation methods in the low-resource setting. However, it becomes increasingly difficult as the data size scales [Raffel *et al.*, 2020]. We evaluate the performance $\overrightarrow{\text{yy-xx}}$ and $\overleftarrow{\text{yy-xx}}$ where yy, xx $\in$ {de,fr,en,es} and yy $\neq$ xx. With most multi-task learning frameworks, it is paramount to maximize positive transfer while minimizing negative transfers. Since PROPS is able to choose up to k rules to apply on a condition, multiple tasks can choose either common shared rules or independent rules, we believe this mechanism helps enhance positive transfer (shared rules), and mitigate negative transfer (independent rules). We present our MBART [Liu *et al.*, 2020] based multilingual translation results below.



(a) Europarl trained with 50 samples



(b) Europarl trained with 500 samples

Figure 4: Few-shot multitask. **PPS = PROPS**. BLEU shown for 10 language pairs. Error bars are standard deviations of 3 random seeds.

The few-shot case: In this experiment, each of the 10 language pairs include 50 or 500 training samples. The task instruction have the form "Translate {source} to {target}". For CA-MTL, each translation pair generates different adapter weights from its task embedding. We do not run experiments for ADAPTER since Pilault *et al.* show that CA-MTL is better equipped to handle multiple jointly trained tasks. In Figure 4, we notice that both Condition Prompt Generators (PROPS and S-TRSF-P) perform much better than PREFIX++. PROPS improves over all other methods on both average performance and standard deviation, including FFT, for **9 out of 10 tasks** when each pair has 50 training examples, and for **8 out of 10 tasks** when each pair has 500 training examples.

Model	trained params	Europarl $\overrightarrow{\text{fr-en}}$	$\overleftarrow{\text{fr-en}}$	$\overrightarrow{\text{de-en}}$	$\overleftarrow{\text{de-en}}$
mBART	100%	40.10	39.54	37.32	30.19
ADAPTER	14%	27.22	26.56	24.25	20.10
CA-MTL	13%	36.81	36.74	33.07	26.17
PREFIX	15%	27.51	26.24	26.01	20.87
PREFIX++	15%	28.75	27.40	26.93	21.38
S-TRSF-P	**10%**	36.76	36.66	33.02	26.24
PROPS	**10%**	**38.17**†	**38.09**†	**35.20**†	**28.39**†

Table 4: Multilingual translation test BLEU results. Results labelled with † are significantly better than all other adapters based on pairwise significance testing [Koehn, 2004] with p = 0.01.

Full data multilingual experiments: Each language pair direction has 1M training and 100K testing examples. In the full data multitask regime, we see that fully finetuned mBART has an edge over adaptation methods. Our results demonstrate that almost all PLM adaptation struggle for a 4M combined dataset size. On average, PROPS is 1.8% lower than FFT but also 1.8% higher than the second highest performing adaptation method. Comparing the results of PROPS and S-TRSF-P to PREFIX++, it is clear that conditional prompt generation performs better than unconidional prompt generation. This is also true when we compare ADAPTER to CA-MTL (Figure 9), indicating that task-conditioned PLM adaptation is better suited at transferring knowledge. We start noticing that PROPS's modularization of knowledge from learnable rules and the control structure has clear benefits over S-TRSF-P.

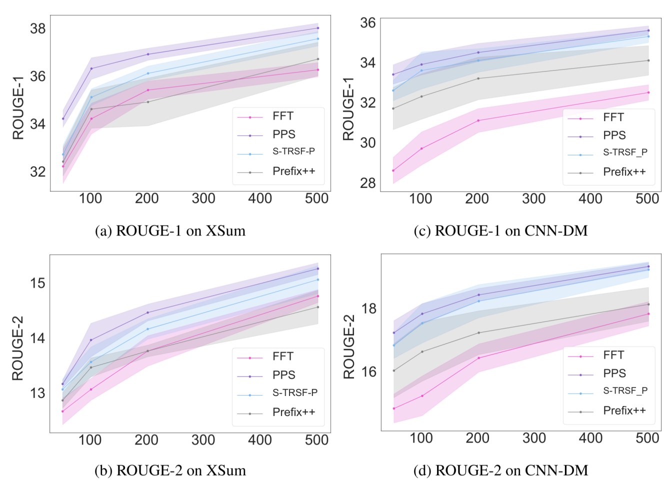
4.4 Does Conditioning Prompt Generators on Data Metadata Improve Performance?

In this section, we assess PROPS and our baseline's performance when text conditioning is the news article metadata (third evaluation case). Our results are in Table 5. The instructions remain unchanged (i.e., "Abstractive Summarization") for each dataset and example. However, the metadata such as news outlet, article type or news topic (for Topic-CNN-DM) change for every example. XSum-OOT is a new dataset based on XSum where the training set consists of specific types of articles (i.e., "world", "politics", "business", "sports") and the test set is composed of the remaining news categories (i.e, "health", "tech"). XSum-OOT tests if models can generalize to unseen news categories and extrapolate from seen data metadata.

Overall, we notice that PROPS, other CPGs and even PREFIX++ all perform better with the added information. We then observe that PROPS consistently surpasses ADAPTER, PREFIX++ and our Conditional Prompt Generation baselines (TRSF-P and S-TRSF-P). We note that PROPS and S-TRSF-P adapt 33% of the layers with 5% fewer trained parameters, 50 fewer prompt vectors when compared to PREFIX. **PROPS is the only adaptation technique that is consistently on par or outperforms FFT**. For close to 4 times less parameters, PROPS consistently outperforms TRSF-P with statistical significance (except on Meteor scores), surprisingly offering a 1.5% relative improvement over TRSF-P on averaged scores across the three datasets. When the task requires more control on the generated output text, we see that PROPS rule selection mechanism boosts relative performance by close to 4% on Topic-CNN-DM [Mrini *et al.*, 2021a] over S-TRSF-P. The improvement in the controlled generation task shows that the separate transformation of conditions help disentangle representations of the topic and other accompanying conditions.

4.5 Can we Compose Instructions, Metadata and Pre-Learned Support Task Knowledge?

In this section, we test the affects of pre-learning specific Conditional Language Generation support tasks along with their instructions. At test time, we evaluate our methods on Abstractive Summarization using an instruction that was textually composed from the pre-learned task instructions. We handpick three support tasks that are related to summarization. Indeed, summarization can be decomposed into several arguably simpler tasks [Pilault *et al.*, 2020]: (1) **NLU -** understand and interpret a source document, (2) **Extract -** highlight and prioritize important points, (3) **Paraphrase -** coherently reformulate key messages. We describe a technique to induce transfer and composition via text conditioning. We vary the *overlap of words* describing a support task and a target task, and measure the effects on target task zero-shot and few-shot performance. In this setting, support tasks are trained jointly and the target task is evaluated separately. We provide examples of how to create target task descriptions from support task descriptions with certain word overlaps, which is color coded for clarity in Table 8. Note that we did not search over multiple instructions in the experiments below. The experimental setup and the support tasks datasets used or created are discussed in Appendix E.



(a) ROUGE-1 on XSum

(c) ROUGE-1 on CNN-DM

(b) ROUGE-2 on XSum

(d) ROUGE-2 on CNN-DM

Figure 5: Few-shot transfer. **PPS = PROPS**. Axis show average ROUGE scores vs. number of training examples. Error bands represent min/max of three runs with random seeds.

	trained params	prompt length	XSum R1	XSum R2	XSum RL	XSum-OOT R1	XSum-OOT R2	XSum-OOT RL	Topic-CNN-DM R1	Topic-CNN-DM R2	Topic-CNN-DM RL	Topic-CNN-DM MET
Full model fine-tuning												
BART [1]	100%	—	45.14	22.27	37.25	39.53	16.88	31.55	39.38	22.41	35.10	29.93
Adaptation without conditions												
ADAPTER❣	14%	—	43.29	20.75	34.91	38.51	15.76	30.28	37.19	20.90	34.01	27.89
PREFIX [2]	15%	200	43.80	20.93	36.05	39.41	16.87	31.47	38.86	21.44	34.81	28.41
Adaptation with conditions												
PREFIX++	15%	200	43.90	20.98	36.14	39.52	16.87	31.49	38.87	21.52	35.02	28.59
TRSF-P	36%	150	44.39	21.41	36.26	39.82	17.06	31.71	39.32	23.06	36.25	**32.07**
S-TRSF-P	**10%**	150	44.25	21.36	36.10	39.53	16.95	31.49	38.93	22.02	35.38	29.77
PROPS	**10%**	150	**44.67**†	**21.64**†	**36.52**†	**40.03**†	**17.25**†	**31.90**†	**39.94**†	**23.77**†	**36.90**†	32.05

Table 5: Summarization results. ❣: Applying method of Lin *et al.* on BART. **best** and 2nd best results indicated. [1]XSum results from Lewis *et al.* and Topic-CNN-DM results from Mrini *et al.*. [2]XSum and XSum-OOT results from Li and Liang. Results labelled with † are significantly better than all other adapters with p = 0.05. MET=Meteor.

The few-shot case: For this experiment, we use the same set-up as in the zero-shot section, except our target task is fine-tuned on a few examples. The intermediate task prompts used in this section are presented in Table 8 in the "simple" task instruction column. We also ran experiments on Topic-CNN-DM to see if our results generalized to other datasets. Our few-shot summarization results are shown in Figure 5. We see that PROPS average performance is above that of other BART based techniques while also maintaining a lower spread between maximum and minimum values. This means that PROPS better leverages prior task instructions and knowledge to initialize new task prompts. Again, for controlable generation (Topic-CNN-DM), PROPS displays close to a **5 and 2 ROUGE-1 point and 3 and 2 ROUGE-2 point improvement** over FFT and PREFIX++ respectively.

The zero-shot case: We first test if pretrained task and *"Simple task instructions"* in Table 8 are enough to solve our target task without any additional fine-tuning. CPG modularization approaches such as PROPS should improve the composition of task instructions by reusing words and phrases found in support tasks and further enhance PROPS's rule reusability and knowledge composition ability. From our results in the *"Simple"* column of Table 6, we first notice that PROPS tops S-TRSF-P, our strongest baseline, by 0.8 R1 or a 4% relative improvement.

Method	Task Instructions: Simple	Task Instructions: Detailed	Δ R
	R1 / R2	***R1 / R2***	***R1 / R2***
BART FFT	20.9$_{\pm 1.1}$ / **4.0**$_{\pm 0.2}$	21.4$_{\pm 0.7}$ / 4.2$_{\pm 0.1}$	**+0.5** / +0.3
PREFIX	19.0$_{\pm 0.5}$ / 3.5$_{\pm 0.1}$	19.1$_{\pm 0.5}$ / 3.5$_{\pm 0.1}$	+0.1 / +0.0
S-TRSF-P	21.0$_{\pm 0.8}$ / 3.9$_{\pm 0.1}$	21.2$_{\pm 0.5}$ / 4.1$_{\pm 0.1}$	+0.3 / +0.1
PROPS	**21.8**$_{\pm 0.6}$ / **4.0**$_{\pm 0.1}$	**22.3**$_{\pm 0.5}$ / **4.4**$_{\pm 0.1}$	**+0.5 / +0.4**

Table 6: Zero-shot test ROUGE 1 and ROUGE 2 scores on XSum when pretrained with "simple" or "detailed" task instructions. +/- numbers in grey are the standard deviation (STD).

Improving task instructions: We evaluate if it helps to improve the task instructions content and support-target word overlap. The detailed target instruction contains word segments from all three support tasks whereas the simple target instruction is composed from only two tasks. As shown in Table 8, *"Detailed task instructions"* allow us to fully connect each support task with the target task, while providing additional and potentially relevant information. Further, the detailed target instruction are lengthier and allow greater overlap between words describing the support tasks as well.

Our *hypothesis* is that greater overlap and lengthier instruction will improve target task performance by better inducing prior knowledge composition. Task description composition by assembling task descriptions works in conjunction with module composition (see explanation in Appendix C and an empirical analysis in Appendix D.3). Our results suggest that all models benefit from the improved task instruction as observed in the *"Detailed"* column of Table 6. Specifically, PROPS gets the largest % Δ gain when moving from "Simple" to "Detailed" instruction, with a 10% increase in R2 while maintaining the same standard deviation.

5 Conclusion

In this work, we evaluated conditional and composable architectures to conditionally generate differentiable continuous prompts. Our new modular prompt generator PROPS with sparse rule selection outperforms strong Transformer-based prompt generator baselines. PROPS excels in the ability to distill knowledge from the conditioned task and example metadata, and often surpasses the fully fine-tuned model, with fewer trainable parameters than other prompt generators. In general, PROPS performed well on three tasks – summarization, translation, and semantic parsing, with three different models (BART, mBART, T5). PROPS provided even larger improvements in controlled CNLG, where disentangling topic information from other conditions is important. The improvement is even larger for task which requires more controlled generation. PROPS showed ability to compose knowledge from bridge tasks to enhance transfer learning and improve unseen tasks in few-shot setting both from a mechanistic and induced standpoint. Finally, via both theoretical (see Appendix D) and empirical analysis, we showed how rules are sparsely selected and composed, the key ingredients that make PROPS more successful and generalizable.

References

[Ando and Zhang, 2005] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(61):1817–1853, 2005.

[Bapna and Firat, 2019] Ankur Bapna and Orhan Firat. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conference on EMNLP and the 9th International Joint Conference on Natural Language Processing*, pages 1538–1548, Hong Kong, China, November 2019. Association for Computational Linguistics.

[Cheng and Lapata, 2016] Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany, August 2016. Association for Computational Linguistics.

[Efrat and Levy, 2020] Avia Efrat and Omer Levy. The turking test: Can language models understand instructions? *CoRR*, abs/2010.11982, 2020.

[Furrer *et al.*, 2020] Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Scharli. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *ArXiv*, abs/2007.08970, 2020.

[Goyal and Bengio, 2021] Anirudh Goyal and Yoshua Bengio. Neural production systems. In *Advances in Neural Information Processing Systems*, 2021.

[Goyal *et al.*, 2021] Anirudh Goyal, A. Lamb, J. Hoffmann, Shagun Sodhani, S. Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. In *International Conference on Learning Representations*, 2021.

[Guo *et al.*, 2021] Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Proceedings of ACL*, 2021.

[Han *et al.*, 2021] Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. PTR: prompt tuning with rules for text classification. *CoRR*, abs/2105.11259, 2021.

[Houlsby *et al.*, 2019] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 09–15 Jun 2019.

[Hu *et al.*, 2021] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

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

[Keysers *et al.*, 2020] Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*, 2020.

[Kirsch *et al.*, 2018] Louis Kirsch, Julius Kunze, and David Barber. Modular networks: Learning to decompose neural computation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[Koehn, 2004] Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[Koehn, 2005] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X: Papers*, pages 79–86, Phuket, Thailand, September 13-15 2005.

[Lake and Baroni, 2018] Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*, 2018.

[Lester *et al.*, 2021] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[Lewis *et al.*, 2020] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the ACL*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[Li and Liang, 2021] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the ACL and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August 2021. Association for Computational Linguistics.

[Li *et al.*, 2022] Junyi Li, Tianyi Tang, Jian-Yun Nie, Ji-Rong Wen, and Xin Zhao. Learning to transfer prompts for text generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3506–3518, Seattle, United States, July 2022. Association for Computational Linguistics.

[Lin *et al.*, 2020] Zhaojiang Lin, Andrea Madotto, and Pascale Fung. Exploring versatile generative language model via parameter-efficient transfer learning. In *Findings of EMNLP 2020*, pages 441–459, Online, November 2020. Association for Computational Linguistics.

[Liu *et al.*, 2020] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020.

[Liu *et al.*, 2021] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *CoRR*, abs/2103.10385, 2021.

[Mishra *et al.*, 2021] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Natural instructions: Benchmarking generalization to new tasks from natural language instructions. *CoRR*, abs/2104.08773, 2021.

[Mrini *et al.*, 2021a] Khalil Mrini, Can Liu, and Markus Dreyer. Rewards with negative examples for reinforced topic-focused abstractive summarization. In *Proceedings of the Third Workshop on New Frontiers in Summarization*, pages 33–38, Online and in Dominican Republic, November 2021. Association for Computational Linguistics.

[Mrini *et al.*, 2021b] Khalil Mrini, Can Liu, and Markus Dreyer. Rewards with negative examples for reinforced topic-focused abstractive summarization. In *Proceedings of the Third Workshop on New Frontiers in Summarization*, pages 33–38, Online and in Dominican Republic, November 2021. Association for Computational Linguistics.

[Narayan *et al.*, 2018] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of EMNLP*, 2018.

[Newell, 1972] Allen. Newell. *Human problem solving*. Prentice-Hall, Englewood Cliffs, N. J, 1972.

[Nilsson, 2010] Nils J. Nilsson. *The Quest for Artificial Intelligence: A History of Ideas and Achievements*. Cambridge University Press, Cambridge, UK, 2010.

[Novikova *et al.*, 2017] Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany, August 2017.

[Ostapenko and Charlin, 2021] Oleksiy Ostapenko and Laurent Charlin. Continual learning via local module composition. In *Advances in Neural Information Processing Systems*, 2021.

[Petroni *et al.*, 2019] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on EMNLP and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics.

[Pilault *et al.*, 2020] Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Chris Pal. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on EMNLP*, pages 9308–9319, Online, November 2020. Association for Computational Linguistics.

[Pilault *et al.*, 2021] Jonathan Pilault, Amine El hattami, and Christopher Pal. Conditionally adaptive multi-task learning: Improving transfer learning in nlp using fewer parameters & less data. In *International Conference on Learning Representations*, 2021.

[Ponti *et al.*, 2023] Edoardo Maria Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. Combining parameter-efficient modules for task-level generalisation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 687–702, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[Raffel *et al.*, 2020] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[Schick and Schütze, 2021] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the ACL: Main Volume*, pages 255–269, Online, April 2021. Association for Computational Linguistics.

[Touretzky and Hinton, 1988] David S. Touretzky and Geoffrey E. Hinton. A distributed connectionist production system. *Cognitive Science*, 12(3):423–466, 1988.

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

[Tsimpoukelli *et al.*, 2021] Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *CoRR*, abs/2106.13884, 2021.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[Wagner, 2020] C.G. Wagner. *A First Course in Enumerative Combinatorics*. Pure and Applied Undergraduate Texts. American Mathematical Society, 2020.

[Wang *et al.*, 2018] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[Wu *et al.*, 2020] Chuhan Wu, Fangzhao Wu, Tao Qi, Xiaohui Cui, and Yongfeng Huang. Attentive pooling with learnable norms for text representation. In *Proceedings of the 58th Annual Meeting of the ACL*, pages 2961–2970, Online, July 2020. Association for Computational Linguistics.