Discourse-Aware Prompt Design for Text Generation

Anonymous ACL submission

Abstract

Current efficient fine-tuning methods (e.g., adapters (Houlsby et al., 2019), prefix-tuning (Li and Liang, 2021a), etc.) have optimized conditional text generation via training a small set of extra parameters of the neural language model, while freezing the rest for efficiency. While showing strong performance on some generation tasks, they don't generalize across all generation tasks. In this work, we show that prompt based conditional text generation can be improved with simple and efficient methods that simulate modeling the discourse structure of human written text. We introduce two key design choices: First, we show that a higher-level discourse structure of human written text can be modelled with *hierarchical blocking* on prefix parameters. It enables spanning different parts of the input and output text and yields more coherent output generations. Second, we propose sparse prefix tuning by introducing *attention sparsity* on the prefix parameters at different layers of the network and learn sparse transformations on the softmax-function, respectively. We find that sparse attention enables the prefix-tuning to better control of the input contents (salient facts) yielding more efficient tuning of the prefix-parameters. Our experiments show that structured design of prefix parameters can yield more coherent, faithful and relevant generations than baseline prefix-tuning on all generation tasks and perform at par with fine-tuning while being more efficient.[1]

1 Introduction

Recent advances in pre-trained langauge models (PLMs) (Lewis et al., 2020; Raffel et al., 2020; Radford et al., 2019) have made great impact on text generation research, especially when they are fine-tuned on downstream tasks such as summarization, data-to-text generation, long-question answering, etc. Consequent research have shown that PLMs' impact can further be improved when trained with more parameters, on more data and with more compute (GPT-3 (Brown et al., 2020), Megatron (Kharya and Alvi, 2021)). On the flip side, storing larger LMs or fully fine-tuning them (updating all the parameters) on downstream tasks usually causes resource or over-fitting issues.

To mitigate fine-tuning issues, recent work have proposed *prompt-based learning* (Liu et al., 2021a), which focus on learning textual prompts to steer PLMs' continuation towards desired output while keeping the model parameters frozen. While providing strong control of the PLMs, such prompt engineering could be time consuming requiring manual crafting. There is a growing research direction under prompt learning towards lightweight fine-tuning (Houlsby et al., 2019; Lester et al., 2021), which update only a small number of existing or extra parameters while keeping the rest of the pre-trained parameters frozen. Among them is *prefix-tuning* (Li and Liang, 2021b), which focuses on text generation tasks. It prepends tunable continuous task-specific prompt vectors called *prefix*es to the input and only trains these continuous prompts during fine-tuning. Although prefix-tuning can yield comparable results to full fine-tuning on some generation tasks, it did not generalize well to known generation tasks like abstractive summarization.

In this work, we focus on prefix-tuning and propose approaches to improve its generalization on text generation tasks. We investigate efficient design choices considering text generation challenges to close the gap with the full fine-tuning while providing evidences to answer the following questions: (1) *Do different parts of the transformer network process the prefix parameters more efficiently?*; (2) *Do prefix parameters capture high-level discourse structure of the input text?*; (3) *Can constraining prefix attention distribution to be structurally sparse enable better transfer of task features?*

To address (1), we conduct empirical analysis on

[1] All supporting code will be publicly released.

prefix-tuned BART (Lewis et al., 2020), by varying the size of prefix parameters at the encoder and decoder networks on text generation tasks. We find that the prefix parameters at higher layers impact the performance the most, while sparse prefixes can be sufficient at the lower layers (§ 6.1).

Motivated by this finding and to address (2), we introduce **discourse-aware prompting** via **hierarchical blocking** of prefix parameters. Previous text generation work (e.g., abstractive summarization) has shown that abstraction can be better modeled with hierarchically structured architectures (Liu and Lapata, 2019; Fabbri et al., 2019; Xiao et al., 2021). To simulate a hierarchical discourse structure while *only* tuning additional prefix parameters, we first split the input and output text into segments and then assign sets of prefix parameters to each segment at different layers. With this structure, a set of prefixes can only be reached by their designated input or output segments during self-attention. We argue that for conditional generation tasks with hierarchically structured blocking of prefixes, we can simulate the structure of human writing styles: in input text each paragraph is a distinct section of related sentences and in output text (e.g., summary) each output sentence outlines salient concepts. Thus, a set of prefixes designated to each input and output segment at different layers can learn levels of abstractions from each section. We show strong performance improvements over baseline prefix tuning, yielding comparable results to full fine-tuning in all generation tasks in § 6.2.

Inspired by these findings, we address (3) by introducing a suite of **sparse attention** alternatives to standard full-attention matrix. Prior work have shown that sparsity in self-attention not only improves training efficiency, but also focusing on *salient features* while pushing down unrelated features and relations can provide better control for the model. This improves language modeling (Sukhbaatar et al., 2021; Wang et al., 2020), language understanding (Shi et al., 2021; Cui et al., 2019) and text generation (Zaheer et al., 2020; Li et al., 2021; Liu et al., 2021b; Manakul and Gales, 2021). Motivated by this, we **introduce sparsity into the self-attention** by substituting the softmax function with a sparse alternative under encoder prefix-tuning without introducing any additional model parameters. Our quantitative and human evaluations as well as spectral analysis (to analyze if sparse prefix-tuned models can encode important features better than dense models) collectively yield that sparse attention enables better control of the input contents (salient facts) yielding more efficient tuning of the prefix-parameters (§ 6.3).

Efficient tuning of PLMs offers a promising new direction for many NLP tasks including text generation, which we study in this work. Our results support our hypothesis that **prompt design with hierarchical structure and sparsity in prefix parameters**: (*i*) generate more coherent and faithful text than baseline prefix-tuning across several summarization and structure-to-text generation tasks on quantitative and human evaluation metrics, (*ii*) perform at par with fine-tuning on most tasks while being more efficient at training time, (*iii*) outperform all the baselines in low-resource settings.

2 Related Work

Prompt Tuning. Recent years have observed several efficient methods for prompt-based tuning of large-scale PLMs (Liu et al., 2021a). These range from prompt engineering (Petroni et al., 2019; Cui et al., 2021), to more advanced approaches such as prompt ensembling (Mao et al., 2021), composition (Han et al., 2021), or prompt-aware training methods (Lester et al., 2021; Gu et al., 2021). Li and Liang (2021a) propose *prefix-tuning* and show strong results on some text generation tasks, leaving room for further generalization. Here, we build directly upon the prefix-tuning from Li and Liang (2021a), showing where it falls short and providing several discourse-aware prompt design approaches. We find that the prefix-tuning struggles with encoding of salient concepts that constraint generation models require. This setting bears similarities to discourse modeling, which we discuss below.

Discourse Modeling. A large family of methods make architectural design choices to teach models about the overall document discourse structure (Marcu, 1997; Barzilay and Lee, 2004; Barzilay and Lapata, 2008; Li and Hovy, 2014) to improve the summarization task. Recent work investigate different architectures to model the discourse structure via: structured attention (Cohan et al., 2018a), graph based methods (Dong et al., 2021), or hierarchical encoders (Pasunuru et al., 2021). We simulate the discourse structure of text via hierarchical prefix structure and propose discourse-aware prompt-design for efficient PLM tuning.

Sparse Language Models. Most work on sparsity in transformers aim at improving the time and

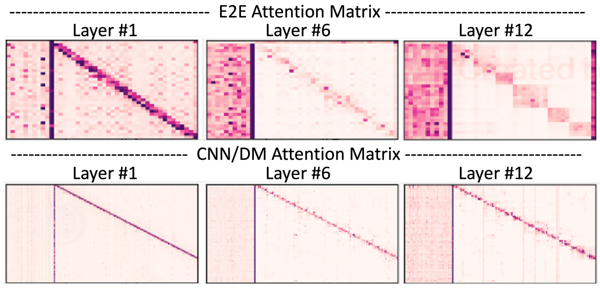


Figure 1: Encoder self-attention matrices $\boldsymbol{A}$ from layers 1, 6 and 12 of prefix-tuned models showing query attention scores (on y-axis) over all prefix+inputs keys (on x-axis). Top row are matrices of models on E2E dataset where the first 10 features on x-axis are prefix features, and bottom row are on CNN/DM dataset where first 100 features are prefix parameters.

space bottleneck of dense transformers (Tay et al., 2021). Work on text generation imbue sparsity to improve coherence, fluency, n-gram diversity and reduce repetition. These work range from: sparse methods on posterior vocabulary distributions at inference time (Fan et al., 2018; Holtzman et al., 2020), sparse attention mechanisms (Cui et al., 2019; Liu et al., 2021b; Shi et al., 2021; Sukhbaatar et al., 2021), modified softmax Martins et al. (2020), or loss functions (Welleck et al., 2020) to improve LM coherence. Following these work, we introduce sparsity on the attention matrix of prefix+input features to improve the knowledge transferred to downstream text generation tasks and generating more relevant and coherent text.

3 Preliminaries

We build our models on the encoder-decoder Transformer architecture (Vaswani et al., 2017; Lewis et al., 2020), with a stack of layers composed of a multi-head self-attention and feedforward network (FFN) sublayers. A decoder usually has another multi-head cross-attention module between the self-attention and FFN, which we omit for simplicity.

Self-Attention. The output of each timestep t is the hidden state $h_t^l \in \mathbb{R}^d$ at layer l, which is then projected to key $k_t^l = W_k^l h_t^l$, value $v_t^l = W_v^l h_t^l$, and query $q_t^l = W_q^l h_t^l$ vectors. We focus on a single layer and omit the layer index l for brevity. The W_q^l,W_k^l, W_v^l are parameters learnt to project inputs to queries, keys and values. Context information is obtained through attention $a_{ti} \in \boldsymbol{A}$ distribution: a_{ti}=Softmax$\left(q_t^\top k_i\right)$ to create the output $o_t = W_o \sum_{i=1}^{T} a_{ti} v_i$, where i,t=$1 \cdots T$ and l=$1 \cdots L$.

Prefix-Tuning. Extending text-based prompt tuning methods (Liu et al., 2021a), prefix-tuning (Li and Liang, 2021b) introduces task-specific prompt parameters. At each layer, it prepends P tunable prefix parameters as additional keys $k^p \in \mathbb{R}^{L\times d}$ and values $v^p \in \mathbb{R}^{L\times d}$ to multi-head self-attention:

$$
\begin{aligned}
a_{tn} &= \underset{i\in 1\ldots T,\, j\in 1\ldots P}{\mathrm{Softmax}} \left(q_t^\top [k_j^p, k_i]\right) \\
o_t &= W_o \left(\textstyle\sum_{j=1}^{P} a_{tj} v_j^p + \sum_{i=P+1}^{T} a_{ti} v_i\right).
\end{aligned} \tag{1}
$$

$[,]$ indicates concatenation, n=$1\cdots(P{+}T)$. During training only the parameters corresponding to the prefix keys W_k^p and values W_v^p are initialized and the same objective function as finetuning is used.

4 Discourse Aware Prompt Design

Visualizing prompt impact. To motivate the discourse-aware prompt design, we investigate the impact of prefix-parameters on transformer models during prefix-tuning. We first analyze the attention behaviour similar to (Sun and Lu, 2020). We prefix-tune two BART-LARGE models, one on structure-to-text generation task with E2E dataset (Duš ek et al., 2019), and another on summarization with CNN/DM (Hermann et al., 2015). For E2E we use 10-prefixes (the first 10 keys are from prefix parameters) and 100-prefixes for CNN/DM[2]. In Figure 1, we plot the encoder self-attention distributions $\boldsymbol{A}$ for different layers averaging over all head vectors. The x-axis represent the keys (k_i) while y-axis denote the queries (q_t). For attention matrices of all the layers, see Appendix A.4 Figure 6. The attention scores show stronger relations with the prefix-keys in the E2E model compared to CNN/DM, where the prefixes exhibit weaker relations compared to the input keys. We attribute this to a few issues which we investigate in this work:

Modeling hierarchical structure. Firstly, during prefix-tuning, the model should not only focus on learning the task specific semantics, but also the models should learn the corresponding discourse structure of the downstream task datasets. To model the intrinsic structure of input text, biasing transformer models with a type of hierarchy has been shown to improve the generation performance. For example, previous work (Cohan et al., 2018b; Liu and Lapata, 2019) learns the discourse structure of human written text (e.g., the beginning, body, conclusion paragraphs, topic shifts, etc.) with hierarchically structured transformers to capture the salient aspects in the input text necessary for improved performance in summarization.

[2]The length of per instance input tokens is 680 in CNN/DM and 26 in E2E dataset, so we use less prefix parameters.

With probing experiments Jawahar et al. (2019) show that BERT (Devlin et al., 2019) captures surface features and phrase-level information in the lower layers, syntactic features in the middle and semantic features and long-distance dependencies at the higher layers. Motivated by these, we introduce variations of **hierarchical blocking** on prefix parameters at different layers of the network and investigate their impact on text generation with qualitative and quantitative experiments.

Introducing sparsity. Secondly, the weaker prefix attention in longer inputs (Figure 1-CNN/DM attention matrices) may imply that the attention neglects important connections, and potentially disturbed by many unrelated words. This issue can be attributed to the softmax function at attention score calculation (Laha et al., 2018; Cui et al., 2019). Softmax produces attention distribution with dense dependencies between words, and fails to assign near/exactly zero probability to less meaningful relations. Thus, the model neglects to pay more attention to important connections while also being easily disturbed by many unrelated words (Cui et al., 2019). This issue is more pronounced in tasks like abstractive summarization, since only a handful of salient input aspects is needed to compose a coherent summary. Sparse attention mechanisms (Liu et al., 2021b; Shi et al., 2021) can remedy this issue by learning to avoid attending to the content unrelated to the query. We introduce soft-attention blocking on the prefix and input parameters to put emphasis on important prefixes and tokens.

We introduce below a suite of blocking schemes and sparsity as sketched in Figure 2. Each block represents attention matrix $\mathbf{A}\in\mathbb{R}^{T\times(P+T)}$, with each key-feature (column) denoting to attention weights $a_{t,n}$ of P prefix and T input-key features.

4.1 Prefix Blocking

As shown in Figure 2-(b) and (e), the two variations of prefix-blocking we introduce here are a type of structural bias we imbue the models to simulate high-level discourse structure of documents:

(*i*) Uniform Blocking (UniBlock): We first split the sequence of input tokens into segments. We allocate different sets of prefix parameters to each segment and apply blocking on the rest of the prefix parameters. In baseline prefix-tuning, a query of a token can bind with all the prefix and input key and value parameters, while in the uniform blocked prefix-tuning, the query of a token in the

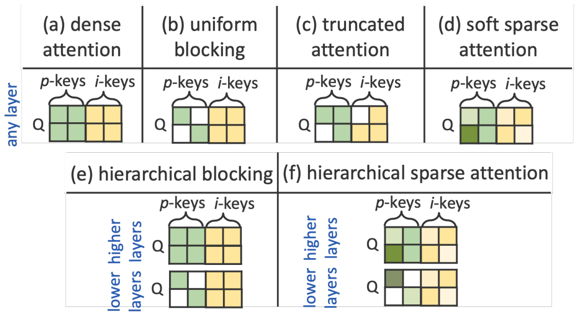


Figure 2: Attention matrices $\boldsymbol{A}$ of prefix-tuning models representing different prefix design patterns. p-keys and i-keys denote P prefix and T input keys. Sparsity of attention scores are indicated by color gradations. White cells in any row represent blocked input prefix parameters for the query.

input or output segment can bind with all input key and values but only with the designated prefix key and value vectors. For example, if P=100 prefix parameters are introduced and we split the input tokens into 2 segments, the first 50 prefix keys k_j^p and values v_j^p (j=1..50) can only be bound with the query vectors of input tokens from the first input segment and so on. We only apply blocking to the prefix parameters and let all inputs tokens attend to each other, see Figure 2-(b). In uniform blocking, we use the same blocking schema at each layer.

(*ii*) Hierarchical Blocking (HierBlock): To bias the prefix parameters with a form of hierarchy, we use the uniform prefix-blocking on the lower layers of the transformer, while we let all tokens attend to all prefixes at the top layers as shown in Figure 2-(e). The attention matrix of the top layers is same as the standard prefix-tuning of (Li and Liang, 2021b) where no blocking on prefixes is applied.

4.2 Sparse Attention Prefix-Tuning

To train a prefix-tuning model that learns to highlight important input content, we introduce four sparse attention design options for the encoder.

(*a*) Truncated Sparse Attention (TruncSA): We apply top-p sampling on both the prefix and input keys as follows: we first add all the row elements of the attention matrix, namely the attention scores contributing from all the queries, then normalize across all key-features, which yields a key-feature impact row vector $\tilde{\mathbf{a}} \in \mathbb{R}^{(P+T)}$ and $\tilde{a}_t \in \tilde{\mathbf{a}}$:

$$\bar{a}_n = \sum_i^T a_{t,i} \quad \tilde{a}_t = \bar{a}_t / (\sum_n^{(P+T)} \bar{a}_n) \qquad (2)$$

Using top-p sampling (Holtzman et al., 2020) we truncate the feature key scores $\tilde{\mathbf{a}}$ and use the top-p portion of the probably mass in each key attention score. We create a binary mask for each key feature via $mask(\tilde{\mathbf{a}})$ = top-$p(\tilde{\mathbf{a}}, \tau)$ by assigning 1.0

to the keys that the top-p sampling has selected, 0 otherwise and threshold parameter τ controls sparsity. Lastly, we broadcast point-wise multiplication between the sparse mask and the attention matrix $\mathbf{A}$ to obtain the top-p sparse attention matrix $\tilde{\mathbf{A}} = mask(\tilde{\mathbf{a}}) \odot \mathbf{A}$, as sketched in Figure 2-(c).

Our top-p sampling is similar to using dropout on randomly selected features of the network during training while controlling the dropout rate with a user-defined threshold to compensate for overfitting and performance. Although top-p sparse attention provides automatic control over attention sparsity, truncation completely masks some features. Next, we show how to dynamically learn to apply *soft*-sparsity via sampling from a distribution.

***(b) Soft* Sparse Attention (SoftSA)**: Influencing the attention distribution with a stochastic mask to attend to salient tokens can potentially help build higher quality sparse attention for text modeling. Several work investigate novel approaches to learn the sparsity in attention matrix (Li et al., 2021; Roy et al., 2021; Shi et al., 2021) using a sampling method to formulate the right amount of sparsity. They associate the attention scores $a_{t,i}$ with each position (t, i) in $\mathbf{A}$ and define a sampling distribution to learn the attention mask during training as sketched in Figure 2-(d). Similarly, we define relaxed Bernoulli distribution as a sampler to construct our stochastic mask. Since sampling from Bernoulli distribution is not differentible, we use the Gumbel Softmax reparameterization trick (Jang et al., 2017) with gumbel-softmax:

$$\tilde{a}_{tn} = \underset{n \in 1 \ldots (P+T)}{\text{Softmax}} \left(a_{t,n}, g, \tau \right) \tag{3}$$

where $g{=}-\log(-\log(u))$ is an independent Gumbel noise generated from the uniform distribution $u \sim U(0, 1)$ and τ is a temperature. As τ approaches zero, the gumbel output approaches to a discrete distribution in {0, 1}, becomes identical to those from the Bernoulli distribution.

(*c* & *d*) Hierarchical Sparse Attention: To simulate an intrinsic discourse structure of the input text, similar to the hierarchical blocking in § 4.1, we apply sparsity on the parameters only at the lower layers. We train hierarchical models with the dense attention at the higher layers, and apply (*c*) *truncated* (**HTruncSA**) or (*d*) *soft* sparse attention (**HSoftSA**) at the lower layers (see Figure 2-(f)).

(*e*) Hierarchical Blocking with Sparse Attention (HierBlock+SoftSA): The hierarchical blocking models we introduced in § 4.1 puts restrictions on the prefix parameters that input tokens can bind with at different layers of the network. To analyze the impact of ensemble of prefix blocking and sparsity, we introduce sparsity to the hierarchically blocked prefix-tuning models. We apply soft sparsity (SoftSA) on the lower layers of the network attention matrices of HierBlock models and keep the higher layer attention matrices dense.

Dataset	Domain	#Data
Summarization		Train/Val/Test
XSum (2018)	News	204K/11K/11K
CNN/DM (2015)	News	287K/13K/11K
Wikihow (2018)	DIY	157K/5.6K/5.6k
SAMSum (2019)	Dialog	15.7K/<1K/<1K
Pubmed (2018b)	Clinical	203K/6K/6K
Structure to Text (S2T)		
E2E (2017; 2019)	Reviews	33K/4K/4.7K
DART (2021)	Reviews	63K/7K/12.5K

Table 1: Datasets used in the experiments.

5 Experiment Setup

Methods. All of the models are based on BART-LARGE (Lewis et al., 2020), though our methods can be applied to any architecture. We compare our discourse aware prefix-tuning approaches to full parameter fine-tuning and baseline prefix-tuning (Li and Liang, 2021a). Finetuning updates all the LM parameters, while all prefix-tuning models freeze LM parameters and only update the prefix parameters. Baseline prefix-tuning models update prefix parameters at each layer (full-stack) of the transformers using dense attention while our proposed models use variations of sparse and blocked attention at different layers of the network. We choose the best models on validation dataset during training and repeat each experiment ~3-5 times with different random seeds and report the average results. For details of the setup see Appendix A.1.

Datasets. We conduct experiments across six datasets on two tasks: abstractive summarization and structure-to-text (S2T) generation. We present a summary of the datasets in Table 1 and provide more details about the datasets in Appendix A.2.

Metrics. For all the tasks and datasets we use the n-gram match metrics: ROUGE-1/2/L, and report human evaluations to compare the results of the models on various qualitative evaluation criteria.

6 Experiment Results

6.1 Are all prefix-parameters useful?

Finding: Prefix-tuning models encode diverse but task specific features at each layer differently, while

Method	XSum R1/R2/RL	CNN/DM R1/R2/RL	PubMed R1/R2/RL	Wikihow R1/R2/RL	SAMSum R1/R2/RL	Avg. -
Finetune	**43.37/20.55/35.31**	**42.46/19.78/29.56**	**40.51/15.50/23.93**	**41.61/17.76/32.40**	51.02/25.70/41.58	**32.07**
Prefix-tune	42.31/19.28/34.37	42.31/19.47/28.94	34.07/12.58/20.38	37.32/14.37/27.17	**51.98/27.37/43.28**	30.78
Prefix-tune with Blocking						
UniBlock	42.91/**19.64**/34.61	42.04/19.50/29.16	38.93/14.24/22.63	**39.31/16.60/30.85**	51.07/26.50/42.53	30.82
HierBlock	**42.99**/19.56/**34.76**	**43.04/20.21/29.73**	**39.10/14.50/22.91**	39.10/16.42/30.59	**51.53/26.83/42.94**	**31.66**

Table 2: **Prefix blocking** experiment results in comparison to finetuning and prefix-tuning on **summarization** tasks. Uniform (UniBlock) and Hierarchical (HierBlock) prefix blocking represent models which use prefix-blocking at different layers of the network (§ 4.1). The top-two best results across models are bolded.

the top-layer prefixes encode abstract features.

Analysis: Earlier work (Jiang et al., 2021; Elazar et al., 2021; Yates et al., 2021) suggests that some layers of the transformers are better than others at producing representations that are useful for a given task. To investigate if similar patterns show up in prefix-tuned models, we take XSum dataset and train models with prefix parameters only at the top layers, the bottom layers, and at a single layer.

Layers	Rouge-1/2/L
Top (8-12)	40.1/16.8/31.4
Low (1-7)	33.7/13.1/26.9
All (1-12)	41.2/18.4/33.4

Table 3: Validation Rouge scores of prefix-tuned models on XSum using only top/low layers.

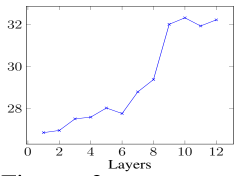


Figure 3: Validation **Rouge-L** on single-layer prefix-tuning with XSum.

We show layer-specific prefix-tuned models' validation performance results in Table 3. The 'Top' layers model is tuned with only the top-layer prefix parameters (i.e., top 4 layers have additional prefix parameters), the 'Low' layers model uses only the lower-layer prefix-parameters (i.e., bottom 7 layers have additional prefix parameters) and 'All' layers prefix parameters is same as baseline prefix-tuning. On inspection, we see a large performance gap between the models trained with top/lower layers up to 6.4 Rouge-1 scores, while we obtain the best performance when we tune all-layer prefix parameters. We see similar patterns on the SAMSum dialog summarization and E2E structure to text generation tasks (details in Appendix A.5). We also build models when prefix parameters are used at a single layer of the network. Our analysis on single layers in Figure 3 suggest that the top layer prefixes can encode summary related abstract information.

6.2 Are hierarchical prompts effective?

Finding: Hierarchical design of prefix parameters can yield more robust information transfer in text generation outperforming baseline prefix-tuning.

Analysis: To bias prefix parameters with the structure of the input documents to learn discourse related representations (as discussed in §4), we experiment with two hierarchical structures: *uniform* (UniBlock) and *hierarchical* (HierBlock) from § 4.1. In Table 2 we report the performance of our models in comparison to fine-tuning and baseline prefix-tuning on abstractive summarization tasks. Our results indicate that prefix-blocking models improve over the baseline prefix-tuning on all summarization tasks by up to +1.0 ROUGE score overall, and even outperforming fine-tuning on CNN/DM dataset. Especially for PubMed and Wikihow, which are considered long document summarization tasks, structure in prefixes improves learning better semantic representations of the downstream tasks compared to baseline models with no structural bias. In addition, the performance gap is larger in news article summarization compared to SAMSum conversational summarization dataset. The reason behind this may be that SAMSum input tokens are shorter and hierarchical discourse structure is not prominent as much as in the long document encoding tasks. We further observe that hierarchical blocking on prefixes also helps for structure-to-text tasks, though the performance impact of structural bias is more prominent in summarization tasks. We show detailed results of structure-to-text tasks and provide samples of generated outputs in Appendix A.6.

6.3 Does sparse attention help prefix-tuning?

Finding: With hierarchically structured sparsity training, prefix tuning show more sparse patterns at the lower layers. Sparse prefix parameters at lower layers, and dense at higher layers enable more efficient tuning of the prefix-parameters.

Spectrum Analysis: To investigate if our sparse models do in fact learn sparse representations, we conduct spectrum analysis on the encoder attention matrix **A** zooming in on the prefix parameters[3]. To

[3] A similar spectrum analysis has been used to prove the sparsity of the attention matrix in Linformer (Wang et al., 2020), a sparse transformer.

analyze the variation of attention scores we calculate the principal components of the attention scores of prefix parameters[4]. We observe that the spectrum distribution of prefixes in lower layers is more skewed than in higher layers, meaning that, in lower layers, more information is concentrated in the largest singular values and the rank of **A** is lower. With sparse attention at the lower layers and dense attention at the top layers, the prefix-tuned models can encode salient features controlling the generation. Details on spectrum analysis are provided in Appendix A.7 and Figure 7.

Sparsity Analysis: To further support the findings from the spectrum analysis, we investigate the impact of sparsity on the performance of the prefix-tuning models. For a fair comparison, we also apply attention sparsity on the finetuned models. We build prefix-tuning models with (*a*) Truncated Sparse Attention (TruncSA), (*b*) Soft Sparse Attention (SoftSA), (*c*) Hierarchical TruncSA (HTruncSA), with top-p sparsity at the lower layers, and dense attention at the top layers, (*d*) Hierarchical Soft Sparse Attention (HSoftSA), with soft sparse attention at the lower layers but dense at top layers.

We show the ROUGE-L results in Table 4. We observe that when sparsity is used on the prefix-parameters, the prefix-tuned models learn to encode more salient features about the summarization task and outperform baseline all-dense prefix-tuning models on all datasets. The performance improvements are more pronounced on long document summarization tasks such as Pubmed, reaching more than 2.0 ROUGE score improvements. Comparing all layers sparse models of (*a*) and (*b*) to hierarchically biased sparsity models of (*c*) and (*d*), we observe improvements with the hierarchically structured sparse prefix-tuning models. More details on quantitative analysis are provided in Appendix A.7 and Table 11.

6.4 Does sparsity on hierarchically blocked prefixes further improve performance?

Finding: The most performance gains are obtained when sparsity constraints are applied on the hierarchically blocked prefixes (Table 5).

Analysis: Recall from the earlier discussions in §6.2 that, if we apply blocking on the lower layered prefixes, while we let all tokens attend to all prefixes at the top layers (HierBlock models), we observe significant performance improvements. On separate set of ablations in §6.3, we also observe that if we introduce sparsity at different layers of the network, the sparse parameters influence the performance compared to the dense prefix tuned parameters at all layers. We now introduce sparsity on the hierarchically blocked prefix-models, combining the best hierarchically blocked prefix-tuned models with the sparse attention.

In Table 5 we show results of our hierarchical prefix blocking (HierBlock) model against hierarchical prefix blocking model with soft sparse attention (HierBlock+SoftSA). To build the HierBlock+SoftSA models, we apply soft sparsity at the lower layers with blocked prefix parameters, while the top layers use dense prefixes with all tokens attending to all prefixes. In Table 5 we repeat the results of the last row from Table 2 for easy comparison. We observe performance improvements on almost all the summarization tasks: XSum, CNN/DM, SAMSum, PubMED. We find that HierBlock+SoftSA models show significant improvements on SAMSum (±1.3; $p<1 \times 10^{-4}$). On the structure to text generation tasks the sparsity on hierarchical blocking

Method	XSum	CNN	PubMed	Wikihow	SAMSum
Finetune					
Dense	**35.31**	**29.56**	**23.93**	**32.40**	**41.58**
(*a*) TruncSA	34.90	28.36	20.90	27.88	41.46
(*b*) SoftSA	35.34	29.32	23.73	**32.50**	41.42
Prefix-tune					
Dense	34.37	34.37	20.38	27.17	43.28
(*a*) TruncSA	35.14	29.59	22.60	27.63	43.31
(*b*) SoftSA	35.14	**29.64**	22.66	**27.70**	**43.80**
(*c*) HTruncSA	**35.26**	28.54	**22.75**	27.59	43.57
(*d*) HSoftSA	35.20	29.69	22.70	27.66	**43.73**

Table 4: Sparse Attention experiment **ROUGE-L** results on Finetuning, and Prefix-tuning using dense and soft sparse attention designs in §6.3. Best performing finetune and sparse prefix-tune model results are bolded within each block.

Dataset	**HierBlock** R1/R2/RL	**HierBlock+SoftSA** R1/R2/RL
Summarization		
XSum	42.99/19.56/34.76	**43.26/19.89/34.99**
CNN/DM	43.04/20.21/29.73	**43.04/20.24/29.81**
PubMED	**39.10/14.50/22.91**	**39.10**/14.46/**22.91**
Wikihow	**39.10/16.42/30.59**	38.45/16.35/30.54
SAMSum	51.53/26.83/42.94	**52.91/27.56/43.63**
Structure to Text		
E2E	**72.10/43.79/51.27**	71.61/43.49/50.85
DART	74.44/48.18/56.72	**74.62/48.60/57.03**

Table 5: What happens when we introduce sparsity to hierarchically blocked prompt design? Experiment results comparing dense and sparse prefix-tuning with structurally biased prefix design (via hierarchical blocking) on various text generation tasks. The best results across models are bolded.

[4] Eigenvalues capture the variation of the attention scores distribution along different principal components.

Loses %	Wins % matches			
Faithfullness	PT	HSoftSA	HB+SoftSA	HB
PT		**64.0**	52.9	**74.7**
HSoftSA	36.0		46.4	**65.3**
HB+SoftSA	47.6	53.6		58.0
HB	25.3	34.7	42.0	
Overall	PT	HSoftSA	HB+SoftSA	HB
PT		**67.5**	48.3	**63.7**
HSoftSA	32.5		47.3	55.2
HB+SoftSA	51.7	52.7		**57.3**
HB	36.3	44.8	42.7	

Table 6: Human evaluation results on *Faithfullness* (top) and *Overall* (bottom) ratings. PT: Prefixtune, HSoftSA: Hierarchical Soft Attention, HB: HierBlock, HB+SoftSA: HierBlock with Soft Sparse Attention. Bold win %s indicate significance ($p < .05$).

helps on some datasets (with E2E), though both HierBlock and HierBlock+SoftSA perform better than baseline prefix-tuning models (see App. Table 9). More details are provided in Appendix A.8.

6.5 Do human evals. support our claims?

Finding: Humans generally prefer generated text from hierarchically blocked prefix-tuned models over all other models, find overall quality of generations indistinguishable from fine-tuning.
Analysis: To evaluate the generated text from our proposed methods against baseline models, we ask human annotators to rate generations on five criteria: *faithfulness* (consistent with the context), *relevance* (captures key-points), *grammaticality*, *coherence* (form a cohesive whole), and *overall quality* (helpful, informative). Table 6 shows the results of the study on faithfulness, and overall metrics. The columns show the percentage of wins of the model against its opponent on a given row. Our Hierarhical Blocking (HierBlock) and Hierarchical Soft Sparse Attention (HSoftSA) models beat prefix-tuning and HierBlock significantly ($p < .05$) beats most of our sparse models on all axes including factuality. In Table 12 we provide comparisons with fine-tuning and observe that HierBlock models perform as good as finetuning on all criteria. More details about the evaluation setup as well as results on all the criteria comparing against fine-tuning and prefix-tuning can be found in Appendix A.10.

6.6 Which structural features are harder to transfer in low-resource settings?

Finding: In low-resource settings, hierarchically designed sparse prefix parameters can efficiently transfer knowledge and represent the semantics and structure of input text yielding more accurate output generations.
Analysis: We simulate a low-resource setting by randomly sampling $k\%$ (k=5,10,25,50) from the training dataset of two summarization tasks: XSum on news, and Wikihow on DIY domains (see train data sizes in Table 1). We use the same hyperparameter settings as our previous models detailed in § 5. We compare our approach to finetuning and prefix-tuning under low-resource settings.

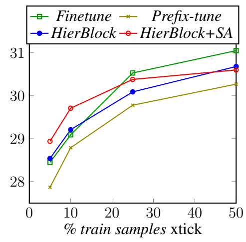


Figure 4: Average ROUGE-L scores on **low-resource settings**.

In Figure 4 on the right, we plot ROUGE-L averaging scores of models trained on XSUM and Wikihow. Our structured prefix-tuned models, HierBlock (blue) and its sparse extension which uses sparse features, HierBlock+SA (red) outperforms fine-tuned (green) and prefix-tuned models (olive), while using the same number of parameters in low resources settings (when <50% training samples are used). Although HierBlock models show consistent performance, on low-resource settings HierBlock-SA performance is more stable. (See Appendix A.11 for more details.)

7 Conclusion and Limitations

We have described simple but effective prompt design options for prefix-tuning of text generation tasks. We enrich prefix parameters with structural biases by way of: prefix-blocking at different layers of the network, sparsity on prefix-parameters and an ensemble of both biases. We show with quantitative and human evaluations on metrics such as coherence and faithfullness that discourse aware prefix designs outperforms baseline prefix-tuning across all text generation tasks even at low data settings and perform at par with finetuning.

We note a few limitations of our work: (1) our experiments are limited by available datasets, and only evaluated on limited closed domain text generation tasks; (2) we focused on efficient prefix-tuning, while ensemble of different efficient tuning models can boost performance even further; (3) we conduct experiments with ~300M parameter models as in past work, but it will be valuable for future work to scale to larger models which may exhibit more coherent generations.

8 Ethics Statement

In this work we propose a new encoder-decoder modeling architecture and build several models to benchmark our new architecture with baseline architectures on several open source text generation datasets.

Intended use. Our architecture is designed to build models of abstractive document summarization and table summarization. Potentially our architecture could be used to train models for summarizing any type of datasets (e.g., any documents, textual conversational dialogues, blog posts, reports, meetings, legal forms, etc.) to further improve the productivity and efficiency of the users in their daily activities without needing to read/listen to long documents/conversations/meetings.

Failure mode. Even though our models yield factually consistent summaries, as judged by us and raters, they can still generate factually inconsistent summaries or sometimes hallucinate information that the source document does not include. This might be due to the bias or noise in the training data. Model builders wanting to use our architecture to build models on their datasets should build models with consideration of intellectual properties and privacy rights.

Misuse Potential. We note the models to be built with our architecture should be used with careful consideration especially if used to build summarization models. The generated summaries produced by our models are not controlled and use generative approaches, therefore, they could generate unreliable text. Researchers working on abstractive summarization should focus on generating factually correct, ethical and reliable text. If our models are trained on news datasets, a careful consideration should be made on factuality of the generated text and measures have been taken to prevent model hallucinations.

References

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1).

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL*.

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

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. 2021. DialogSum: A real-life scenario dialogue summarization dataset. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5062–5074, Online. Association for Computational Linguistics.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, W. Chang, and Nazli Goharian. 2018a. A discourse-aware attention model for abstractive summarization of long documents. In *NAACL*.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018b. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2019. Fine-tune BERT with sparse self-attention mechanism. 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 3548–3553, Hong Kong, China. Association for Computational Linguistics.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1835–1845, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Yue Dong, Andrei Mircea, and Jackie Chi Kit Cheung. 2021. Discourse-aware unsupervised summarization for long scientific documents. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1089–1102, Online. Association for Computational Linguistics.

Ondř ej Duš ek, David M. Howcroft, and Verena Rieser. 2019. Semantic noise matters for neural natural language generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 421–426, Tokyo, Japan. Association for Computational Linguistics.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *ACL*.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *arXiv preprint arXiv:1911.12237*.

Xiaodong Gu, Kang Min Yoo, and Sang-Woo Lee. 2021. Response generation with context-aware prompt learning. *CoRR*.

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

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration.

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.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics.

Yichen Jiang, Asli Celikyilmaz, Paul Smolensky, Paul Soulos, Sudha Rao, Hamid Palangi, Roland Fernandez, Caitlin Smith, Mohit Bansal, and Jianfeng Gao. 2021. Enriching transformers with structured tensor-product representations for abstractive summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online. Association for Computational Linguistics.

P. Kharya and A. Alvi. 2021. Using deepspeed and megatron to train megatron-turing nlg 530b, the world's largest and most powerful generative language model.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset.

Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. 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 540–551, Hong Kong, China. Association for Computational Linguistics.

Anirban Laha, Saneem A. Chemmengath, Priyanka Agrawal, Mitesh M. Khapra, Karthik Sankaranarayanan, and H. G. Ramaswamy. 2018. On controllable sparse alternatives to softmax. In *NeurIPS*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP*.

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.

Haoran Li, Arash Einolghozati, Srini Iyer, Bhargavi Paranjape, Yashar Mehdad, Sonal Gupta, and Marjan Ghazvininejad. 2021. Ease: Extractive-abstractive summarization with explanations. *ArXiv*, abs/2105.06982.

Jiwei Li and Eduard H Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Xiang Lisa Li and Percy Liang. 2021a. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the ACL*.

Xiang Lisa Li and Percy Liang. 2021b. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language*

Processing (Volume 1: Long Papers), pages 4582–4597, Online. Association for Computational Linguistics.

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

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. *CoRR*.

Ye Liu, Jian-Guo Zhang, Yao Wan, Congying Xia, Lifang He, and Philip S. Yu. 2021b. Hetformer: Heterogeneous transformer with sparse attention for long-text extractive summarization.

Potsawee Manakul and Mark Gales. 2021. Long-span summarization via local attention and content selection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6026–6041, Online. Association for Computational Linguistics.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen tau Yih, and Madian Khabsa. 2021. Unipelt: A unified framework for parameter-efficient language model tuning. *ArXiv*, abs/2110.07577.

Daniel Marcu. 1997. From discourse structures to text summaries. In *Intelligent Scalable Text Summarization*.

Pedro Henrique Martins, Zita Marinho, and André F. T. Martins. 2020. Sparse text generation. In *EMNLP*.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Murori Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. Dart: Open-domain structured data record to text generation. *arXiv preprint arXiv:2007.02871*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Jekaterina Novikova, Ondrej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Saarbrücken, Germany. ArXiv:1706.09254.

Ramakanth Pasunuru, Asli Celikyilmaz, Michel Galley, Chenyan Xiong, Yizhe Zhang, Mohit Bansal, and Jianfeng Gao. 2021. Data augmentation for abstractive query-focused multi-document summarization. In *AAAI*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

PurdueOWL. 2019. Journalism and journalistic writing: The inverted pyramid structure.

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

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.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient Content-Based Sparse Attention with Routing Transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

A. See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Han Shi, Jiahui Gao, Xiaozhe Ren, Hang Xu, Xiaodan Liang, Zhenguo Li, and James T. Kwok. 2021. Sparsebert: Rethinking the importance analysis in self-attention. In *ICML*.

Sainbayar Sukhbaatar, Da Ju, Spencer Poff, Stephen Roller, Arthur Szlam, Jason Weston, and Angela Fan. 2021. Not all memories are created equal: Learning to forget by expiring.

Xiaobing Sun and Wei Lu. 2020. Understanding attention for text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3418–3428, Online. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training.

Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. 2021. Primer: Pyramid-based masked sentence pre-training for multi-document summarization.

Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained transformers for text ranking: Bert and beyond. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 1154–1156, New York, NY, USA. Association for Computing Machinery.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences.

A Appendix

Context: How to straighten a curly wig?

It is the best method for both straightening and curling a wig. It is also the best method for synthetic wigs that may be more delicate to heated styling tools.
Allow the hair to fall straight down without resting on anything. Tripods work brilliantly if you fit a mannequin head on them.
Boil about 1.5L for a long wig and 1L or less for a short wig. The heat of the water will soften the plastic fibers and will pull them down. Use a wide soft-bristled brush.
For extremely curly wigs, it will take more than one treatment and a lot of time. It does no damage to the wig and can be done while the wig is still wet. This is a permanent style; you will need to re-curl the wig if you don't want it straight anymore. You're basically remolding plastic into a new shape, so will need to mold it back if you want the curls again.

Output Summaries:

Human : consider trying the hot water method. place the wig on a stand. boil a pot full of water. pour it carefully over the wig, making sure that it completely covers it. avoid combing or brushing a wig when it is wet. wait until it is dry. repeat the method if necessary.
Finetune: understand that this method is best for very curly wigs. place the wig on a flat surface. boil water in a large pot. remove the pot from the heat and gently brush the wig with a wide-bristled brush. repeat the process until the entire wig is curled.
Prefixtune: put the wig on a tripod. boil water in a large pot. brush the wig with a soft brush. repeat the process until the hair is completely straight. repeat as needed. repeat until the wig is fully straightened. repeat this process for longer wigs.
SoftSA: wash the wig with warm water. place the wig on a tripod. boil the wig. brush the wig down with a soft-bristled brush. let the wig air dry for a few minutes. repeat the process if you want the hair to stay straight.
HSoftSA: understand the benefits of this method. lay the wig flat on a flat surface. boil water. brush the wig with a soft-bristled brush. let the hair air dry. repeat the process as needed. re-curl the wig if necessary.
HierBlock: put the wig in a bowl of warm water. place the wig on a tripod. pour the water over the wig. brush the hair with a soft-bristled brush. repeat as needed.
HierBlock+SoftSA: wash the wig with warm water. put the wig on a tripod. boil the water. brush the wig down with a soft-bristled brush. let the hair air dry. repeat the process with the wig if you want it to stay straight.

Figure 5: Model Generated Output Text on Wikihow Dataset. The red colored text indicates factual errors, repetitions, and incoherent text.

A.1 Hyperparameters (Cont. from § 5)

We fit our BART-Large models to their respective datasets with the hyperparameters shown in Table 7. Encoder/decoder block sizes indicate the size of the segments we split the input/output tokens. For instance, if the encoder block size is 2, we split the input tokens into two segments. Each segment has designated set of prefixes which can vary at each

Parameter	Xsum	CNN/DM	PubMed	Wikihow	SAMSum	E2E	DART
learning rate	5e-05	5e-05	5e-05	5e-05	5e-05	5e-05	5e-05
# epochs	30	30	30	30	20	10	10
batch size	8	8	8	8	16	16	16
prefix-length	100	50	50	100	10	10	10
beamsize	5	5	5	5	5	5	5
Hierarchical Blocking							
encoder block size	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3	1,2,3
decoder block size	1,2	1,2	1,2	1,2	1,2	1,2	1,2
Sparse Attention							
top-p	95.%	95.%	95.%	95.%	95.%	95.%	95.%
τ (top-p)	1.0,0.1	1.0,0.1	1.0,0.1	1.0,0.1	1.0,0.1	1.0,0.1	1.0,0.1
τ (soft attn.)	1.0,0.1,0.01	1.0,0.1,0.01	1.0,0.1,0.01	1.0,0.1,0.01	1.0,0.1,0.01	1.0,0.1,0.01	1.0,0.1,0.01

Table 7: Hyperparameters of different prefix-tuned models.

Corpus	Version	License	Citation	Link
XSum	v1	MIT	Narayan et al. (2018)	https://github.com/EdinburghNLP/XSum
CNN/DM	v1	MIT	Hermann et al. (2015)	https://github.com/abisee/cnn-dailymail
PubMed	v1	Creative Commons	Cohan et al. (2018b)	https://github.com/armancohan/long-summarization
WikiHow	v1	CC-BY-NC-SA	Koupaee and Wang (2018)	https://github.com/mahnazkoupaee/WikiHow-Dataset
SAMSum	v1	CC BY-NC-ND 4.0	Gliwa et al. (2019)	https://github.com/giancolu/Samsung-dataset
E2E	v1	CC4.0-BY-SA	Duš ek et al. (2019)	https://github.com/tuetschek/e2e-cleaning
DART	v1	MIT	Nan et al. (2021)	https://github.com/Yale-LILY/dart

Table 8: Additional documentation of scientific artifacts used in our paper.

layer. In hierarchical blocking models (HierBlock) we segment the lower layers, so the prefixes are blocked for different segments, while at the top layers no segmentation or blocking is applied. We use at most two segments in the output text since the text generations tasks we investigate in this work contain much shorter output tokens compared to the input tokens.

A.2 Dataset Details (Cont. from §5)

All datasets are in English language. The summarization datasets range from extreme abstractive summarization with XSum (Narayan et al., 2018) to summarize documents into one summary sentence, conversational summarization using SAMSum dataset (Gliwa et al., 2019), long clinical document summarization with PubMed (Cohan et al., 2018b)[5] and DIY domain with Wikihow (Koupaee and Wang, 2018), and commonly used CNN/DM (Hermann et al., 2015; See et al., 2017) news article summarization dataset with an ”Inverted Pyramid” (PurdueOWL, 2019) document structure (Kryscinski et al., 2019). We also investigate S2T datasets on customer reviewers including E2E (Novikova et al., 2017; Duš ek et al., 2019) and DART (Nan et al., 2021) with each input being a semantic RDF triple set derived from data records in tables and sentence descriptions that cover all facts in the triple set.

XSum (Narayan et al., 2018) is a collection of 227k BBC News articles ranging from 2010 to 2017. The dataset covers a wide range of subjects. The single-sentence summaries are written by professionals.

CNN/DailyMail (Hermann et al., 2015) dataset contains 93k news articles extracted from CNN News, and around 220k articles extracted from the Daily Mail newspapers. The summaries are human written bullet point text which are provided in the same source documents. In our experiments we use the non-anonymized version, which is commonly used in summarization research papers.

PubMed (Cohan et al., 2018b) is a long document dataset of 215K scientific publications from PubMed. The task is to generate the abstract from the paper body.

WikiHow (Koupaee and Wang, 2018) is a large-scale dataset of 200K instructions from the online WikiHow.com website. Each instance consists of multiple instruction-step paragraphs and an accompanying summary sentence of each paragraph. The task is to generate the concatenated summary-sentences from the paragraphs.

SAMSum (Gliwa et al., 2019) is a multi-turn dialog corpus of 16K chat dialogues and manually

[5] We acknowledge that the source of dataset is the NLM Catalog, and the citations used in Pubmed corpus may not reflect the most current/accurate data available from NLM, which is updated regularly.

annotated summaries. The task is to generate an abstractive summary of the dialog with coherent discourse structure of the original dialog.

E2E (Duš ek et al., 2019) is a structured data to natural langauge summary dataset that provides information about restaurants. The structured inputs consists of different attributes (slots) such as name, type of food or area and their values. It contains 50K instances of diverse descriptions of the structured input introducing challenges, such as open vocabulary, complex syntactic structures and diverse discourse phenomena.

DART (Nan et al., 2021) is a text generation dataset for open-domain structured data-record to text generation. It consists of 82K examples from variety of domains. The inputs are in semantic RDF triple set form which are derived from data records in tables and tree ontology of the schema. The output generations are human annotated with sentence descriptions that cover all facts in the triple set.

Licence details In our experiments, we use several datasets (as detailed above) from public resources . Table 8 summarizes the licences. All data are solely used for research purposes.

A.3 Compute Infrastructure and Run time

Each experiment runs on a single machine with 8 GPUs. Depending on the training dataset size, summarization models require from 5.5 hours to 18 hours to train. The structure-to-text datasets are much smaller which takes less than 4 hours. All fine-tuned models follow the BART-large transformer architecture with a total of 12 layers, 1024 hidden dimensions, and 406M parameters. The prefix-models increase the parameters size of fine-tune models by 0.1% up to 2% depending on the number of prefix parameters. See hyperparameters details in Appendix A.1.

A.4 Visualization of Prefix Parameters (Cont. from § 4

To analyze the attention behaviour (similar to (Sun and Lu, 2020)) we plot the attention matrix of the prefix-tuned models focusing on the prefix parameters. We use a prefix-tuned BART-Large (12-layer stacked transformer) on two tasks: structure-to-text generation on E2E (Duš ek et al., 2019) and summarization on CNN/DM (Hermann et al., 2015). In Figure 6, we plot the encoder self-attention distributions $\boldsymbol{A}$ for different layers averaging over head vectors. The x-axis represent the keys (k_i) while y-axis denote the queries (q_t).

Method	E2E R1/R2/RL	DART R1/R2/RL
Finetune	71.12/42.87/49.61	73.92/47.98/56.44
Prefix-tune	**71.65/43.18/50.50**	**74.48/48.42/56.70**
Prefix-tune with Blocking		
UniBlock	71.27/42.81/47.80	74.37/48.00/55.94
HierBlock	**72.10/43.79/51.27**	**74.44/48.18/56.72**

Table 9: **Blocked prompt design** experiment results in comparison to finetuning and prefix-tuning on **structure-to-text** tasks. The top-two best results across models are bolded.

A.5 Are All Prefix Parameters Useful? (Cont. from § 6.1)

We investigate the influence of prefix parameters on different layers of the network. For this experiments we trained BART-Large and introduced prefix parameters only at the top layers, lower layers and all layers (this is same as baseline prefix-tuning models). On XSum dataset, we observed a large performance gap between the models trained with top/lower layers, while we obtain the best performance when we tune all-layer prefix parameters (in Table 3 in the main text). Here, we investigate if similar perforamance gains are observed on dialog summarization (SAMSum) and data to text generation (E2E) tasks.

We show the performance scores of our experiments on validation datasets in Table 10. We observe similar results as the analysis on XSum dataset. Top layers prefix parameters learn salient features related to the task, though using prefixes at all layers yields better performance.

Method	SAMSum R1/R2/RL	E2E R1/R2/RL
Top (8-12)	50.61/24.24/40.95	66.37/38.10/50.31
Low (1-7)	41.87/19.98/34.12	62.32/33.60/46.22
All (1-12)	52.56/26.93/42.96	67.18/39.71/50.31

Table 10: Validation Rouge scores of prefix- tuned models on SAMSum (from the summarization task) and E2E (from the structure to text task) datasets using only the top/low layers.

A.6 Investigation of Hierarchical Prompt Design (Cont. from § 6.2)

We investigate if blocking prefixes helps for structure-to-text tasks. Table 9 shows the results. Similar to summarization experiments in § 6.2, we observe improvements with hierarchical blocking on structure-to-text datasets, though the structural bias we introduce prefix-tuning models with hierar-

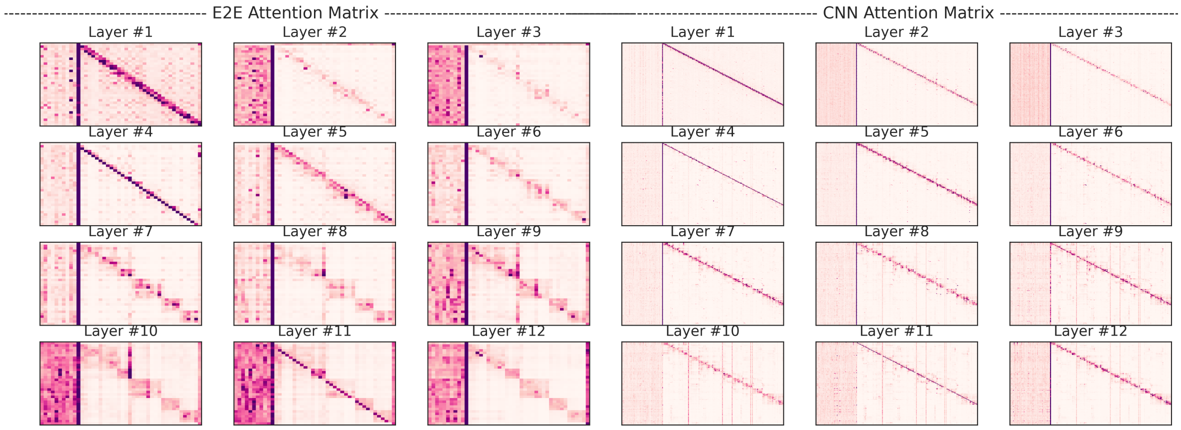


Figure 6: Encoder self-attention matrices $\boldsymbol{A}$ of prefix-tuned models indicating the query attention scores over all keys (prefix+inputs) on the y-axis. The scores are averaged over all heads. The left block is for E2E dataset where the first 10 features represent prefix features, while CNN/DM dataset on the right with first 100 features represent the prefixes.

chical blocking is more prominent in summarization tasks. The models with blocked prefix discourse structure have outperformed finetuning on both tasks by up to 1.0 ROUGE score. We attribute this to potential overfitting of finetuning models on rather smaller size downstream task datasets (compared to summarization tasks, E2E and DART datasets are much smaller in size (Table 1)). We conclude from these results that the prefix models tuned with structurally biased additional set of parameters can yield more robust information transfer outperforming finetuning models.

In Figure 5 we show the output summaries generated by some of our best discourse aware prefix-tuned models in comparison to baseline fine-tuned and prefix-tuned models.

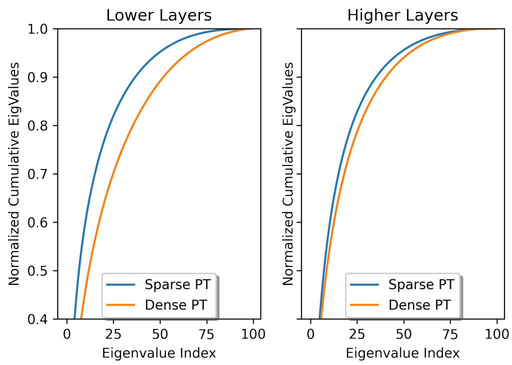


Figure 7: Spectrum analysis of the self-attention matrix comparing the baseline (Dense) and our Sparse Prefix-Tuned (PT) transformer model zooming in on prefix parameters of size 100. The Y-axis is the normalized cumulative singular value of the self-attantion matrix $\mathbf{A}$, and the X-axis the index of largest eigenvalue. The results are based on BART-Large on XSum dataset. The left plots the averages of all $\mathbf{A}$ on the lower layers, while right plots averages over higher layers. Spectrum distribution of prefixes in lower layers is more skewed than in higher layers, meaning that, in lower layers, more information is concentrated in the largest singular values and the rank of $\mathbf{A}$ is lower.

A.7 Investigation of the Impact of Sparsity (Cont. from § 6.3)

Spectrum Analysis: We conduct spectrum analysis of the encoder attention matrix $\mathbf{A}$ zooming in on the prefix parameters to investigate if our sparse models do in fact learn sparse representations. A similar spectrum analysis has been used to prove the sparsity of the attention matrix in Linformer (Wang et al., 2020), a sparse transformer. Our goal is to analyze the principal components of the subspace that captures the variation of the attention scores in prefix parameters. The eigenvalues capture the variation of the attention scores distribution along different principal components. The higher the elbow in the spectrum graph, the less parameters are used and the model learns to represent the inputs with only the salient terms ignoring superfluous details.

For our spectrum analysis, we compare the baseline prefix-tuning, which encodes a *dense* attention matrix everywhere in the network (Dense PT) against one of our sparse prefix-tuned models with truncated attention matrix (Sparse PT), as we explained in § 4.2-(a), using top-p sampling. Both models are a 12-layer stacked transformer (BART-Large) trained on XSum extreme summarization task. We apply singular value decomposition into $\mathbf{A}$ across different layers and different heads of the model, and plot the normalized cumulative singular value averaged over 1000 sentences. We compare

Method	Xsum R1/R2/RL	CNN/DM R1/R2/RL	PubMed R1/R2/RL	Wikihow R1/R2/RL	SAMSum R1/R2/RL
Finetune					
Dense	**43.37/20.55/35.31**	**42.46/19.78/29.56**	**40.51/15.50/23.93**	**41.61/17.76/32.40**	**51.02/25.70/41.58**
(*a*) TruncSA	43.10/20.17/34.90	40.58/18.80/28.36	36.01/12.46/20.90	35.89/14.02/27.88	50.38/25.55/41.46
(*b*) SoftSA	43.34/20.42/35.34	41.97/19.44/29.32	40.32/15.31/23/73	**41.55/17.80/32.50**	50.51/25.71/41.42
Prefix-tune					
Dense	42.31/19.28/34.37	42.31/19.28/34.37	34.07/12.58/20.38	37.32/14.37/27.17	51.98/27.37/43.28
(*a*) TruncSA	43.12/19.97/35.14	42.80/20.01/29.59	38.57/14.08/22.60	37.74/14.81/27.63	52.13/27.41/43.31
(*b*) SoftSA	43.08/20.04/35.14	**42.88/20.10/29.64**	39.00/14.40/22.66	**37.77/14.85/27.70**	52.48/**27.93/43.80**
(*c*) HTruncSA	**43.19**/20.14/**35.26**	42.74/19.98/29.54	**39.04/14.40/22.75**	37.70/14.74/27.59	52.55/27.87/43.57
(*d*) HSoftSA	43.15/**20.16**/35.20	42.83/20.05/29.59	**39.04/14.40**/22.70	37.73/14.85/27.66	**52.71/27.93/43.73**

Table 11: Sparse Attention experiment results on Finetuning, and Prefix-tuning using Truncated (TruncSA) and Bernoulli Sampling soft attention (SoftSA) and Hierarchical Truncated (HTruncSA) and Soft Attention (HSoftSA) for Prefix-Tuning. Each model is repeated 3 times and the average results are reported. Best performing finetune and sparse prefix-tune model results are bolded within each block.

the models' sparsity patterns at the top and at the lower layers separately as shown in Figure 7. The two figures exhibit a long-tail spectrum distribution across layers and heads. This implies that most of the information of matrix **A** can be recovered from the first few largest singular values. We observe that the spectrum distribution in lower layers is more skewed than in higher layers, meaning that, in lower layers, more information is concentrated in the largest singular values and the rank of **A** is lower. With sparse attention at the lower layers and dense attention at the top layers, the prefix-tuned models can encode salient features controlling the generation.

Sparsity Analysis: In Table 11 we show the ROUGE-1, ROUGE-2 and ROUGE-L scores of fine-tuned and prefix-tuned models comparing dense and sparse attention impact. We observe that when sparsity is used on the prefix-parameters, the prefix-tuned models outperform dense counterparts. The performance improvements are more pronounced on long document summarization tasks such as Pubmed and Wikihow, reaching up to 4.0 ROUGE-1 and 2.0 ROUGE-L score improvements.

A.8 Investigation of the Impact of Sparsity on Hierarchically Blocked Prefixes (Cont. from § 6.4)

In Table 5 we showed ROUGE-L results of our hierarchical prefix blocking (HierBlock) model against hierarchical prefix blocking model with soft sparse attention (HierBlock+SoftSA). We observe improvements on performance on almost all the summarization tasks including news summarization (XSum and CNN/DM), dialog summarization (SAMSum), and clinical document summarization (PubMED). We find that HierBlock+SoftSA models show significant improvements on dialog summarization (SAMSum) ($\pm$1.3; p<1 $\times$ 10^{-4}). On the structure to text generation tasks the sparsity on hierarchical blocking helps on some datasets (with E2E), though both HierBlock and HierBlock+SoftSA perform better than the baseline prefix-tuning models (see Table 9).

In fact, our results from the hierarchical sparsity models (HSoftSA) in Table 4 as well as the hierarchical blocking models (HierBlock+SoftSA) in Table 5 on SAMSum dataset is not surprising: From the original SAMSum work (Gliwa et al., 2019) and a very recent dialog summarization work (Chen et al., 2021), we know that the main difficulties of summarizing the dialogues originates partially from the inherent discourse structures in multi-turn dialogues and that models lacking this property perform poorly. Both the hierarchical blocking structures and sparsity on the prefix-parameters can enrich the models with the discourse structure it thrives to generate summaries.

A.9 Automatic Evaluations (Cont. from § 5)

For model evaluations we use ROUGE-1/2/L using Python rouge-score 0.0.4 version licensed under the Apache 2.0 License. We use the default ROUGE script rouge.py from the GEM evaluation shared task.

A.10 Human Evaluations (Cont. from § 6.5)

We perform human evaluations to establish that our model's ROUGE improvements are correlated with human judgments. We compare the generations from four models: baseline prefix-tune (PT), Hierarchically Blocked PT (HierBlock/HB), Hierarchical Soft Sparse Attention PT (HSoftSA) and the ensemble of the blocked sparse model (Hi-

Criteria	**Prefixtune**	**HierBlock**		**Prefixtune**	**HSoftSA**		**Prefixtune**	**HierBlock+SoftSA**	
	wins	wins	same	wins	wins	same	wins	wins	same
factuality	20	**59**	34	27	48	39	40	45	28
relevance	33	48	32	25	39	50	46	40	27
gramaticality	27	40	46	18	42	54	45	31	37
coherence	31	42	40	29	54	40	43	37	33
overall	33	58	22	26	**54**	34	44	41	28
Criteria	**HierBlock**	**HSoftSA**		**HierBlock**	**HierBlock+SoftSA**		**HSoftSA**	**HierBlock+SoftSA**	
	wins	wins	same	wins	wins	same	wins	wins	same
factuality	47	25	41	40	29	44	37	32	44
relevance	42	39	32	40	28	45	39	36	38
gramaticality	34	32	47	28	23	62	42	28	43
coherence	39	37	37	31	27	55	41	37	35
overall	48	39	26	47	35	31	39	35	39
Criteria	**Finetune**	**HierBlock**		**Finetune**	**HierBlock+SoftSA**				
	wins	wins	same	wins	wins	same			
factuality	14	31	43	8	7	10			
relevance	14	36	38	8	6	11			
gramaticality	20	19	49	7	7	11			
coherence	23	18	47	9	7	9			
overall	22	40	26	10	6	9			

Table 12: Head-to-Head comparison of human evaluations on random subset of Wikihow dataset.

Document	Human Summary	Model-A	Model-B	Faithfullness	Relevance	Grammatically	Coherence	Overall

Table 13: Human annotation screen as used in spread-sheet format.

erBlock+SoftSA). We use the following as evaluation criteria for generated summaries, which we inlcude in the instructions for the annotators.

Faithfulness: Are the details in the summary fully consistent with the details in the source document? The summary must not change any details from the source document. The summary also must not hallucinate any information that is not in the source document.

Relevance: Does the summary capture the key points of the text? Are only the important aspects contained in the summary? Is there any extra/irrelevant information?

Grammaticality: Considers the grammatical quality of each individual sentence in the summary. For each sentence, does it sound natural and grammatically correct?

Coherence: Does the summary form a cohesive, coherent whole? Is it well-written, well-structured and well-organized? Is it easy to follow? It should not be a heap of related information, but should build from sentence to sentence to a coherent body of information about a topic.

Overall Quality: Given the input context, is the summary satisfactory? Does the summary provide good quality information to the user? Is it helpful, informative and detailed enough given the information that's contained in the text? Which summary of the two do you prefer best overall?

Annotator Details: Human annotation was conducted by 9 professional raters (7 linguist raters, 1 linguist subject-matter-expert and 1 linguist) employed by an outsourcing company handling content moderation. All raters are monolingual native speakers of English; 6 have a minimum of high school degree or equivalent and 3 have a bachelor's degree. Raters received compensation starting at $18 per hour (which is close to 2.5 minimum wage in the state where the raters are located) and were also provided with Premium Differential as part of their contracts. Each rater conducted between 44 and 175 pairwise evaluations. Data collection protocol was reviewed by a expert reviewers and received expedited approval as the data presented to the raters did not contain any sensitive or integrity-violating content. Participant consent was obtained as part of the non-disclosure agreement signed by each rater employee upon hire. All raters have also signed a sensitive content agreement that outlined the types of content they may encounter as part of their employment, associated potential risks and information and wellness resources provided by the outsourcing company to its employees.

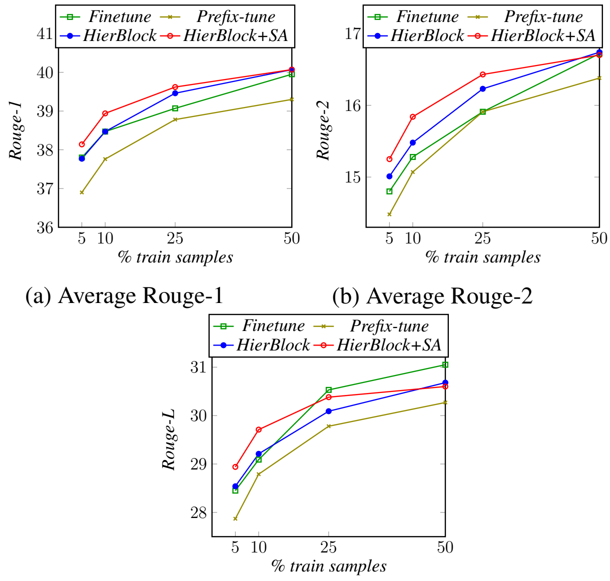
Human Evaluation Procedure: We randomly select 50 samples from the Wikihow test set and ask 9 trained judges to evaluate them on the 5 criteria defined above. We perform head-to-head evaluation (more common in DUC style evaluations), where judges are shown the original document, the ground truth summary and two model summaries in random order. The judges are then asked to compare two model summaries based on each of the five criteria. In each case, a judge either has the option to choose a model summary that ranks higher on a given criterion (i.e., respond by identifying the winning summary), or assert that both summaries are similar given the criterion and rate the comparison as "same". The evaluation of each pair of summaries across all 5 criteria takes on average between 5 and 10 minutes to complete. The raters were shown the data, as shown in Table **??**, to be rated in a spread sheet, where each line contained multiple columns in sequence: document, human written summary, model-A generated summary, model-B generation summary, and five additional columns indicating faithfulness, relevance, gramaticality, coherence, overall quality. The headers of the columns were clearly stated. The rates enter a/b/same in each corresponding cell when comparing summaries head-to-head based on each criteria.

Human Evaluation Results: In Table 12 we show head-to-head evaluation scores on all five metrics showing wins from each model as well as when both are selected as equal. Each sub-table compare a different model. Our Hierarhical Blocking (HierBlock) and Hierarchical Soft Sparse Attention (HSoftSA) models beat prefix-tuning and HierBlock significantly ($p < .05$) beats most of our sparse models on all axes including factuality. In

On a small data annotation, we also compare two of our best models HierBlock and HierBlock+SoftSA againts best finetuning model generations, which are shown in the same Table 12. We observe that in most cases both of our models are prefered as good as finetuning on all criteria, except on overall, the HierBlock summaries are ranked much higher than fine-tuning models.

A.11 Low-data settings (Cont. from § 6.6)

In Figure 8, we plot the ROUGE-1, ROUGE-2 and ROUGE-L scores averaging scores from two summarization tasks (XSUM and Wikihow). Our structured prefix parameter tuned models, HierBlock (blue) and its sparse extension which uses sparse features, HierBlock+SA (red) outperforms Prefix-tuned models (olive), while using the same number of parameters in low resources settings (when



(a) Average Rouge-1 (b) Average Rouge-2

(c) Average Rouge-L

Figure 8: Quantitative analysis on **low-resource settings**. The charts show **average** of ROUGE-1, ROUGE-2, ROUGE-L scores from models trained on two summarization tasks: XSUM and Wikihow. Our structurally biased parameter tuned HierBlock (blue) and HierBlock+SA (red) consistently outperforms the baseline Finetuned (green) and Prefix-tuned models (olive) when <50% training data is used.

<50% training samples are used). Both models outperform Finetuned models (green) on ROUGE-1 and ROUGE-2 metrics (Figure 8-(a)&(b)). While the HierBlock models show consistent performance, we conclude that on low-resource settings HierBlock-SA performance is more stable.