PROMPTSUM: PLANNING WITH MIXED PROMPTS FOR PARAMETER-EFFICIENT CONTROLLABLE ABSTRACTIVE SUMMARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Prompt tuning (PT), a parameter-efficient technique that only tunes the additional prompt embeddings while keeping the backbone pre-trained language model frozen, has shown promising results in language understanding tasks, especially in low-resource scenarios. However, effective prompt design methods suitable for generation tasks such as summarization are still lacking. At the same time, summarization guided through instructions (discrete prompts) can achieve a desirable double objective of higher quality and controllability in summary generation. Towards a triple goal of strong summarization performance with parameter-efficiency, data-efficiency and controllability, we introduce PromptSum, a method combining PT with a multi-task objective and discrete entity prompts for abstractive summarization. Our model achieves state-of-the-art results on several popular few-shot benchmarks as well as a strong level of controllability through entities, all while only tuning several orders of magnitude less parameters.

1 INTRODUCTION

Pre-training large-scale language models (LMs) and adapting them to downstream tasks through fine-tuning has become the dominant paradigm in NLP (Raffel et al., 2019; Radford et al., 2019; Lewis et al., 2019), including in summarization (Zhang et al., 2020). However, full-model fine-tuning requires storing an entire set of weights for each downstream task, prohibiting simultaneous multi-task inference as the models become larger (Chowdhery et al., 2022; Thoppilan et al., 2022). Also, full model fine-tuning requires careful considerations to avoid overfitting, especially when the dataset is small as in most few-shot tasks (Kaplan et al., 2020).

Prompt engineering has recently gained popularity as a low-cost alternative for adapting LMs to downstream tasks. Such prompts are generally constructed either by finding suitable discrete task instructions (e.g., "TL;DR" for summarization) with possibly a few examples (Brown et al., 2020; Schick & Schütze, 2021) or by tuning embeddings of *soft* prompts (Li & Liang, 2021; Lester et al., 2021). As a specific conditioning of the LM, prompt-based adaptation has achieved comparable (or even better) performance to standard fine-tuning in low-resource scenarios, while using very few or even *zero* learnable parameters (Brown et al., 2020; Li & Liang, 2021; Lester et al., 2021).

Specifically, soft prompt tuning (Lester et al., 2021) adapts the nature of discrete prompts to continuous soft prompts whose embeddings are *learned* with backpropagation. This way, the model is offered greater flexibility and capacity in learning prompts, as opposed to manually finding an effective discrete prompt. Soft prompts have brought great success in many language understanding tasks (Qin & Eisner, 2021; Gu et al., 2022; Vu et al., 2022; Su et al., 2022), and have also shown promising results in language generation (Li & Liang, 2021; Qin & Joty, 2022). Despite this, one important drawback of soft prompt tuning is that it lacks the human-explainable nature of discrete text prompts, thus sacrifices user controllability.

On the other hand, for a complex and less constrained generation task like abstractive summarization, guiding the model with additional discrete signals, such as entities or keywords, can significantly enhance the performance as it helps the generative model to be faithful and on topic (Dou et al., 2020). An entity chain (sequence of entities) can be viewed as a high-level *content plan* that can bootstrap the generation process (Liu & Chen, 2021; Puduppully et al., 2019). Entities are also

a strong proxy for topic saliency (Barzilay & Lapata, 2005). Intuitively, when a model is provided with extra information on the important entities to incorporate into the output summaries, it is expected to be more accurate and controllable, and the summary will more likely cover the important facts. Similar to Neuro-Symbolic approaches (Garcez et al., 2022), we try to approximate the human reasoning process of plannning content with entities first before drafting the summaries. Controllability is an essential aspect in summarization as it is a subjective task, where several different yet valid summaries can be generated from the same source, for instance when different aspects are considered (Ahuja et al., 2022; Liu et al., 2020).

Recent work in summarization has indeed incorporated high-level content planning or controllability. CTRLSum (He et al., 2020) takes a discrete instruction as input to enable control on several summarization aspects such as output entities or summary length. Narayan et al. (2021; 2022) propose FROST that trains the same model to first generate an entity chain, and then a summary by conditioning on the entity chain in an auto-regressive manner. They also adapt PEGASUS (Zhang et al., 2020) pre-training objective to include an entity chain during pre-training. This leads to both higher ROUGE scores and a powerful mechanism to control summary generation, which allows for reducing hallucinations by dropping entities not present in the source. Nevertheless, these methods require tuning the entire language model, on the entire dataset.

In this work, our goal is to combine the best of both worlds. On the one hand, we leverage soft prompts to achieve parameter-efficiency in data-efficient setups. On the other hand, we use discrete prompts to induce controllable summarization. Our mixture-of-prompts model maintains good performance while operating under triple constraints of parameter-efficiency, data-efficiency and controllability. Our model is multi-tasking, and can generate either the summary entity chain or the summary itself. To control which task to activate, we use a dedicated soft prompt for each task. After prompting with the entity chain soft prompt, we re-use the predicted entity chain for summary generation. This enables great flexibility, as depending on the setup, the user can input desired entities to condition on. We demonstrate the strength of our approach on four major summarization benchmarks: CNN/DM (Hermann et al., 2015), XSum (Narayan et al., 2018), BillSum (Kornilova & Eidelman, 2019) and SAMSum (Gliwa et al., 2019), and achieve state-of-the-art on several few-shot scenarios. We also conduct extensive qualitative analysis for utilizing our new prompting mechanism for controllable summary generation and show that we can reduce hallucinations.

Our main contributions in this paper are the following:

- We are the first to propose using a mixture of discrete and soft prompts for generation tasks. We empirically show its effectiveness on abstractive summarization.
- We scale our approach by introducing a multi-task prompt-tuning pre-training framework.
- We demonstrate the performance of our method by reaching state-of-the-art few-shot ROUGE summarization results.
- Using the input entity chain, we propose the first *light-weight controllable summarization model*, which shows impressive levels of controllability of the summary *even with few-shot supervision*.

2 Related Work

Prompt-based Learning (PL) In general, to learn new tasks, PL prepends a task-specific template or prompt to the original input (Liu et al., 2021a). Since Brown et al. (2020) showed that a frozen GPT-3 model can achieve promising results on various few-shot tasks through manually designed prompts, many efforts have been made in PL. While early efforts mainly focus on designing *discrete* prompts (Schick & Schütze, 2021; Gao et al., 2020), more recent work attempts to learn trainable parameters, *soft* prompts (Li & Liang, 2021; Liu et al., 2021b; Lester et al., 2021; Qin & Joty, 2021), showing impressive performance on a variety of tasks.

Few-shot Summarization Few-shot learning remains under-explored in abstractive summarization. PEGASUS (Zhang et al., 2020), which pre-trains the model on HugeNews with the *gap-sentence generation objective* to predict salient sentences from the rest of the document, achieves very strong 100-shot performance on several datasets, including XSum. WikiTransfer (Fabbri et al., 2020) proposes to construct pseudo-samples from Wikipedia to fine-tune pre-trained models before few-shot fine-tuning. This approach achieves great progress in zero-shot and few-shot summarization, but at the cost of using external data and performing an intermediate fine-tuning phase. Bi

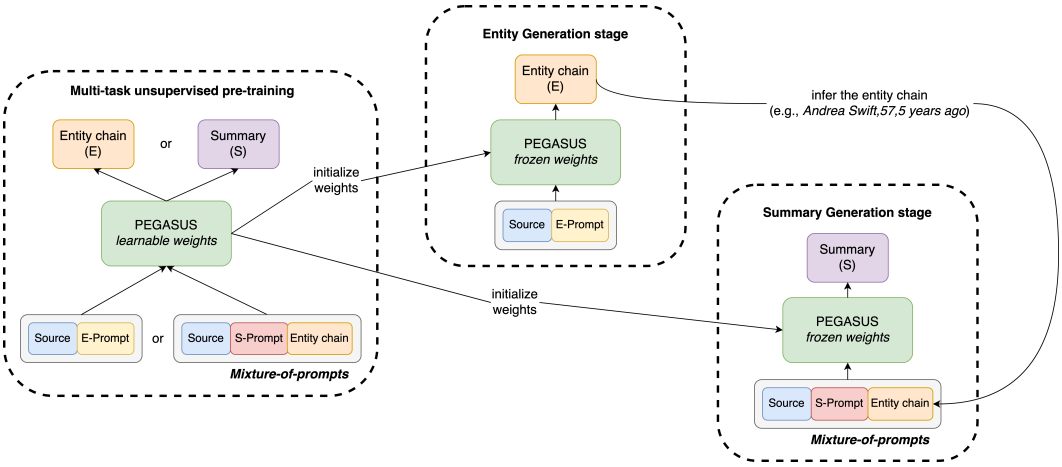

Figure 1: **PromptSum training.** In the **pre-training** stage (left), the model alternates between the entity-chain prediction and summary prediction tasks, both with pseudo-labels and each with a dedicated input soft prompt (*E-Prompt* for entity and *S-Prompt* for summary). Besides, all model weights are updated in pre-training. In the **Entity Generation** stage, PEGASUS weights (initialized from pre-training) are kept frozen, and *E-Prompt* is further optimized for entity chain generation with the available entity-chain supervision. In the **Summary Generation** stage, PEGASUS weights are also initialized from pre-training and kept frozen, and this time *S-Prompt* is tuned for summary generation with the available summary labels. Besides, for summary generation, the model also conditions on predicted entity chains from the first prompt tuning stage.

et al. (2021) adds auxiliary tasks such as object prediction and entailment to the cross-entropy training objective to boost few-shot summarization performance. PSP (Liu et al., 2022b) pre-trains soft prompts and then fine-tunes them for summarization, outperforming the base BART (Lewis et al., 2019) on few-shot benchmarks by a good margin.

Guidance and Planning for Summarization CTRLSum (He et al., 2020) trains the summarization model with input keywords which allow users to input any desired keywords at inference, enabling better controllability on the summarization process. GSum (Dou et al., 2020) uses a second encoder to leverage external signals, such as keywords or salient sentences predicted by an extractive summarization model like MatchSum (Zhong et al., 2020), to guide the base encoder-decoder abstractive summarization model, achieving state-of-the-art results. FROST (Narayan et al., 2021), which is the most comparable to our work, modifies the PEGASUS (Zhang et al., 2020) summarization pre-training objective to condition on entity chains. However, they condition through the *decoder* by making the model generate first the entity chain, then the summary; while we input the entity chain to the *encoder*. Besides, we use a mix of soft and discrete prompts for summary generation. To the best of our knowledge, there is no work proposing a controllable summarization model while being parameter-efficient or data-efficient.

3 Method

3.1 Problem Formulation

From an input text X, we divide the process of generating a summary Y into two subtasks:

- *Entity Generation*: generate a chain of salient (discrete) entities E from the input text X.
- *Summary Generation*: generate the summary Y conditioned on both X and E.

By dividing the traditional summarization process into two stages, we aim to mimic a human's thinking process of generating summaries by first planning out the content with a sequence of discrete entities before writing down the final summary (Cao et al., 2018). The predicted entities in *Entity Generation* will also aid *Summary Generation* performance by bootstrapping salient content

selection. Following this more practical two-stage design, we can better control the summaries with different entity chains (§5.2). Moreover, the model's performance on the few-shot setup is enhanced as the two-stage summarization process is more generalizable to unseen data points (§4.2).

3.2 MULTI-TASK PROMPT TUNING

To learn both the subtasks for summarization with the least amount of parameters and simplest architecture, we employ *multi-task* prompt tuning, where we use the same Pre-trained Language Model (PLM) for both tasks but with different prompts. We decide to use PEGASUS (Zhang et al., 2020) as the backbone PLM, due to its strong generalization across many summarization tasks. Given a training dataset $D^{tr} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ for a task T and a PLM θ, prompt tuning or PT (Lester et al., 2021) prepends the input text X_i with a sequence of tunable prompt tokens P, while all other parameters remain frozen during training.

For the *Entity Generation* task (i.e., predicting salient entities E_i from input X_i), we first formulate a learning dataset $D_E^{tr} = \{(X_1, E_1), \ldots, (X_n, E_n)\}$ where E_i represents the entity chain tagged from the ground truth summary Y_i. Then, to prompt-tune, we use soft prompt tokens P_E, parameterized by prompt embeddings ϕ_E, optimized through gradient descent with the following objective:

$$\mathcal{L}_{\phi_E} = \mathcal{L}(\phi_E, D_E^{tr}) = -\sum_{i=1}^{n} \log(p(E_i|[P_E, X_i], \phi_E, \theta)) \tag{1}$$

For *Summary Generation*, generating summaries Y_i from input X_i and entity chain E_i, we formulate the training dataset $D_S^{tr} = \{(X_1, E_1', Y_1), \ldots, (X_n, E_n', Y_n)\}$, where E_i' is inferred from the prompt-tuned model in the previous stage. Then, we use both soft prompt tokens P_S and discrete prompt E_i' to generate the summary Y_i. Similar to the first prompt tuning stage, only prompt embeddings ϕ_S are optimized through gradient descent through the following objective:

$$\mathcal{L}_{\phi_S} = \mathcal{L}(\phi_S, D_S^{tr}) = -\sum_{i=1}^{n} \log(p(Y_i|[P_S, X_i, E_i'], \phi_S, \theta)) \tag{2}$$

3.3 MODEL TRAINING

Pre-training To better help the model learn each task with the least number of trainable parameters as a soft prompt, we perform pre-training. Similarly to PEGASUS (Zhang et al., 2020), we leverage the C4 dataset for pre-training, first introduced in Raffel et al. (2019). We initialize our model, named *PromptSum*, from the PEGASUS checkpoint pre-trained on C4 and HugeNews, and perform our multi-task prompt tuning pre-training on the *realnewslike* subset of C4, containing 13M samples. We construct pre-training labels that are similar to Narayan et al. (2021): labels for *Summary Generation* follow the *gap-sentences generation* pre-training objective: they are salient sentences removed from each document to form a pseudo-summary. Labels for *Entity Generation* simply consist of the tagged entities found in each pseudo-summary. We initialize soft prompt embeddings for both tasks from PEGASUS embeddings of the most frequent tokens. During pre-training, within each batch, each pre-training document is used once for entity chain generation, and also once for summary generation, each time with the corresponding soft prompt being inputted to the model. The final pre-training loss is simply the sum of each subtask loss: $\mathcal{L}_\phi = \mathcal{L}_{\phi_E} + \mathcal{L}_{\phi_S}$

Fine-tuning Fine-tuning the model on labeled data follows a two-stage approach. First, we extract entity chains from the available ground truth summaries, and train the soft prompt for *Entity Generation*. Then, we infer the entity chain, and train the soft prompt for *Summary Generation*. For each tuning stage, the model is intialized with weights from pre-training, and the corresponding soft prompt is also initialized from its pre-training weights. Besides, during both pre-training and *Summary Generation*, we found it beneficial to use *teacher forcing for entity chains*: at training time, i.e., we input the ground truth entity chain to the model, instead of the predicted one. Not only was such teacher forcing leading to higher performance, but it also sped up training due to avoiding generation of entity chains on the whole training set. At inference, predicted entity chains are first inferred on the whole validation or test set. We refer to Fig. 1 for an overview of the whole training process of PromptSum, with multi-task pre-training on the left, and the prompt-tuning stages in the center (entity) and right (summary).

Dataset	# Data points			# Words		# Tokens (PEGASUS)		# Entities		New summary n-grams (%)		
	Train	Val	Test	Doc.	Summ.	Doc.	Summ.	Doc.	Summ.	1-grams	2-grams	3-grams
CNN/DM	287,113	13,334	11,490	786.68	55.06	851.53	64.57	64.13	5.71	12.07	51.05	71.38
XSum	204,045	11,332	11,334	414.51	22.96	456.96	26.01	38.26	2.85	33.98	83.33	95.52
BillSum	17,055 (ours)	1,894 (ours)	3,269	1,659.13	203.88	1,759.92	209.19	109.05	14.01	10.22	37.77	54.87
SAMSum	14,732	818	819	124.07	23.42	133.07	25.66	11.45	3.25	33.88	79.02	90.10

Table 1: **Statistics** on the datasets that we used for experiments. *Doc.* is short for document, and *Summ.* for summary. Tokens counts are calculated based on PEGASUS tokenization.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Before our pre-training on C4, we initialize PromptSum with the *google/pegasus-large* checkpoint pre-trained on HugeNews and C4. The checkpoint is downloaded from HuggingFace *transformers* library (Wolf et al., 2020). To detect entities as ground-truth, the *spacy* library is used. The datasets used are downloaded with the HuggingFace *datasets* library (Lhoest et al., 2021).

Following PEGASUS (Zhang et al., 2020), we use Adafactor (Shazeer & Stern, 2018) as optimizer for all experiments. We pre-train for 400k steps, using an effective batch size of 256 and data parallelism over multiple GPUs, evaluating every 5k steps. We will release our pre-trained checkpoint publicly. We use 300 tokens per soft prompt, each with embedding size 1024 (as per PEGASUS), leading to a total of 614,400 trainable parameters (sum over both soft prompts) during fine-tuning. We concatenate each soft prompt to the right of the source document, as we observed a slight performance gain compared to concatenating left.

We fine-tune PromptSum on four popular summarization benchmarks:

- **CNN-DailyMail** (Hermann et al., 2015) consists of 93k and 220k news articles from the CNN and DailyMail newspapers, respectively, with corresponding highlighted bullet points serving as summaries. We follow the non anonymized version from See et al. (2017).
- **XSum** (Narayan et al., 2018) is the task of extreme summarization: compressing an entire news article into a single, highly abstractive sentence. It is made of 227k BBC articles from 2010-2017.
- **BillSum** (Kornilova & Eidelman, 2019) contains 22k US Congressional bills and human-written reference summaries from several sessions of the American Congress.
- **SAMSum** (Gliwa et al., 2019) is a corpus of 16k daily-life conversations. The input is significantly shorter than the other datasets, and it is as abstractive as XSum.

We refer the reader to Table 1 for detailed statistics on each dataset. Datasets were selected to cover multiple domains (news, legal, dialogue), abstractiveness levels, and compression ratios.

In few-shot fine-tuning experiments, we fine-tune each task for 60 epochs and evaluate every epoch, using a learning rate of 5e-3 for all experiments. We subsample three random pairs of training and validation sets of the corresponding few-shot size from the training set, and fine-tune a model on each pair, then report results on the test averaged over the three models. For *Entity Generation*, we early stop on the F-1 score of the generated entity chain, and on the mean ROUGE of generated summaries for the *Summary Generation* task.

In the full-shot experiments, we train the model for 5 epochs for each task on CNN/DM and XSum, 20 epochs on BillSum and 30 epochs on SAMSum. We found that in this setup, results could drastically vary depending on the choice of learning rate, and perform a grid search on the validation set over $\{0.0005, 0.005, 0.05, 0.5, 5.0\}$ to find a good learning rate for each dataset. We settle on 0.005 for XSum and SAMSum, 5.0 for CNN/DM and 0.5 for BillSum. We use an effective batch size of 256 and validate every 500 optimization steps on CNN/DM and XSum, every 50 steps on BillSum and SAMSum.

4.2 RESULTS

We evaluate PromptSum summarization performance using the standard ROUGE metric (Lin, 2004), in its three commonly used versions of ROUGE-1, ROUGE-2, and ROUGE-L; in all three supervision scenarios: zero-shot, few-shot (with 1, 10, and 100 labels) and full-shot.

Dataset	Model	Trainable Params (M)	0-shot			1-shot			10-shot			100-shot		
			R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
CNN/DM	T5-large (ours)	783	29.11	8.71	26.97	29.88	8.99	27.60	34.47	11.63	31.76	36.47	13.32	33.63
	BART-large (ours)	406	23.63	2.45	19.94	24.65	3.99	21.57	30.01	5.87	26.31	32.96	7.47	29.24
	PEGASUS-large (Zhang et al., 2020)	568	32.90	13.28	29.38	_	_	_	37.25	15.84	33.49	40.28	18.21	37.03
	PEGASUS-large (ours)	568	**35.15**	**13.93**	**31.15**	**35.09**	**13.89**	**31.01**	36.47	15.06	32.39	39.30	16.56	35.81
	WikiTransfer (Fabbri et al., 2020)	336	_	_	_	_	_	_	39.39	**16.92**	**36.00**	**42.08**	**18.93**	**38.83**
	BART-auxiliary tasks (Bi et al., 2021)	336	_	_	_	_	_	_	**39.50**	16.67	25.15	40.73	17.84	26.73
	PromptSum	0.6	28.35	8.37	24.53	29.46	9.13	25.62	31.82	11.09	27.89	37.79	15.75	34.40
XSum	T5-large (ours)	783	22.02	4.40	14.8	22.32	4.52	14.96	25.26	6.10	16.92	27.90	7.92	19.54
	BART-large (ours)	406	18.87	2.11	11.91	19.32	2.61	12.49	23.05	4.40	16.11	28.22	7.44	20.99
	PEGASUS-large (Zhang et al., 2020)	568	19.27	3.00	12.72	_	_	_	19.39	3.45	14.02	39.07	16.44	31.27
	PEGASUS-large (ours)	568	18.81	2.87	12.90	18.81	2.88	12.92	29.36	10.04	22.38	**41.72**	**18.82**	33.38
	WikiTransfer (Fabbri et al., 2020)	336	_	_	_	_	_	_	**35.17**	**12.76**	**26.80**	37.26	14.20	28.85
	UL2 20B (Tay et al., 2022)	20,000	_	_	_	25.50	8.60	19.80	_	_	_	_	_	_
	PSP (Liu et al., 2022b)	0.2	_	_	_	_	_	_	_	_	_	32.50	10.83	25.03
	PromptSum	0.6	**30.12**	**11.34**	**21.81**	**30.23**	**11.43**	**21.94**	31.56	12.04	23.16	**41.73**	18.63	**33.55**
BillSum	T5-large (ours)	783	24.10	6.69	15.31	20.19	5.92	12.50	36.35	12.32	20.28	41.13	19.83	26.56
	BART-large (ours)	406	22.87	3.43	12.19	25.67	4.69	14.45	35.24	8.74	18.85	39.65	12.16	21.82
	PEGASUS-large (Zhang et al., 2020)	568	**41.02**	**17.44**	**25.24**	_	_	_	**40.48**	**18.49**	**27.27**	44.78	**26.40**	**34.40**
	PEGASUS-large (ours)	568	37.74	15.57	23.33	**38.58**	**16.19**	**23.85**	39.46	18.18	25.80	**46.01**	25.57	32.89
	PromptSum	0.6	34.08	12.62	22.07	34.06	12.52	22.20	38.04	16.22	24.95	43.96	23.30	30.81
SAMSum	T5-large (ours)	783	21.02	4.46	16.62	21.30	4.49	16.68	30.28	**10.56**	21.53	34.01	4.73	27.83
	BART-large (ours)	406	22.07	2.56	14.24	18.77	2.07	12.62	31.42	7.27	23.45	37.05	10.49	27.37
	PEGASUS-large (ours)	568	**26.34**	**6.14**	**20.48**	**26.34***	**6.14***	**20.48***	**32.93**	**10.65**	**26.15**	**41.97**	**17.56**	**33.67**
	PromptSum	0.6	23.94	5.41	18.42	23.48	5.23	18.25	27.24	7.32	21.37	39.77	16.84	32.99

Table 2: **Zero-shot & few-shot ROUGE results** on the test set of several summarization datasets for PromptSum and other leading **full-weights** (top sub-blocks) and **parameter-efficient** (bottom sub-blocks) summarization models. Few-shot results are averaged over three random seeds.
* The results are the same as in 0-shot because we validate before the first epoch in few-shot setups.

Model	Trainable Params (M)	CNN/DM			XSum			BillSum			SAMSum		
		R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
T5-large Raffel et al. (2019)	783	42.50	20.68	39.75	_	_	_	_	_	_	_	_	_
BART-large Lewis et al. (2019)	406	44.16	21.28	40.90	45.14	22.27	37.25	_	_	_	_	_	_
PEGASUS-large Zhang et al. (2020)	568	44.17	21.47	41.11	47.21	24.56	39.25	**57.31**	**40.19**	**45.82**	**52.01***	**27.35***	**43.27***
CTRLSum He et al. (2020)	406	**45.65**	**22.35**	**42.50**	_	_	_	_	_	_	_	_	_
FROST Narayan et al. (2021)	568	45.11	22.11	42.01	**47.80**	**25.06**	**39.76**	_	_	_	_	_	_
Prefix-Tuning Li & Liang (2021)	0.4	_	_	_	42.92	20.03	35.05	_	_	_	_	_	_
PromptSum	0.6	38.16	17.67	35.08	43.12	19.96	35.18	45.20	29.19	35.24	47.43	23.50	39.48

Table 3: **Full-shot ROUGE results** on the four summarization test sets. The bottom block corresponds to **parameter-efficient** methods. * means our run.

We report zero-shot and mean few-shot results in Table 2, with corresponding standard deviations available in Appendix A. We compare against prior summarization work tuning all parameters: T5 (Raffel et al., 2019), BART (Lewis et al., 2019), PEGASUS (Zhang et al., 2020), WikiTransfer (Fabbri et al., 2020), BART with auxiliary tasks (Bi et al., 2021), UL2 (Tay et al., 2022); as well as a parameter-efficient method: PSP (Liu et al., 2022b).[1] As seen, compared to T5 and BART, which suffer from over-fitting when there are few labels, PEGASUS provides largely superior few-shot performance due to its summarization-specific pre-training objective, which motivates our choice of backbone PLM. Despite containing three order of magnitude less parameters than the former type of models, PromptSum achieves several state-of-the-art results on XSum few-shot, and wildly outperforms the comparabale parameter-efficient PSP on XSum. When not achieving SOTA, it is consistently a strong few-shot summarization baseline. The fact that PromptSum outperforms UL2 (Tay et al., 2022) on XSum 1-shot echoes recent work arguing for parameter-efficient fine-tuning instead of in-context learning with extremely large models (Liu et al., 2022a).

We show full-shot results in Table 3. PromptSum outperforms comparable parameter-efficient method Prefix-Tuning (Li & Liang, 2021) on XSum, and is a few ROUGE points behind leading full-weights summarization models such as FROST (Narayan et al., 2021). We also include an entity-specific evaluation in Appendix B, where we obtain with spacy entities from summaries generated by PromptSum and compare against ground-truth entity chains.

[1] We could not evaluate FROST in few-shot due to some compatibility issues with HuggingFace.

Model	Freezing weights?	Pre-training?	Entity chain?	Oracle entities?	Trainable Params (M)	CNN/DM		XSum		BillSum		SAMSum	
						100-shot	Full-shot	100-shot	Full-shot	100-shot	Full-shot	100-shot	Full-shot
Fine-tuning (PEGASUS-large)	✗	✗	✗	✗	568	16.56	20.51	18.82	**23.76**	25.57	37.29	17.56	27.35
Soft PT / full weights	✗	✗	✗	✗	568	16.11	21.14	19.01	**23.75**	25.30	37.75	17.52	**27.59**
Soft PT / full weights (pre-trained)	✗	✓	✗	✗	568	15.73	21.14	**19.81**	**23.74**	**28.23**	**38.05**	18.42	27.15
PromptSum / full weights (no pre-training)	✗	✗	✓	✗	568	8.51	11.21	10.61	17.95	15.58	34.08	14.49	21.89
PromptSum / full weights	✗	✓	✓	✗	568	15.55	17.06	18.82	20.91	26.89	30.70	16.60	22.48
PromptSum-oracle / full weights	✗	✓	✓	✓	568	16.39	**21.25**	17.88	23.34	27.89	36.77	18.33	**27.67**
Soft PT	✓	✗	✗	✗	0.3	16.37	18.84	18.56	21.23	18.40	29.61	13.90	21.20
Soft PT (pre-trained)	✓	✓	✗	✗	0.3	**16.67**	19.53	19.30	21.22	24.37	31.23	**18.59**	24.88
PromptSum (no pre-training)	✓	✗	✓	✗	0.6	10.52	18.06	13.11	20.13	13.40	28.22	11.63	16.02
PromptSum	✓	✓	✓	✗	0.6	15.75	17.67	18.63	19.96	23.30	29.19	16.79	23.50
PromptSum-oracle	✓	✓	✓	✓	0.6	16.55	19.34	18.79	21.00	22.58	30.80	17.95	25.16

Table 4: **Model analysis (ROUGE-2 results)**. PT refers to prompt tuning. We experiment with removing PromptSum components such as freezing weights, pretraining, or using the entity chain. We report **ROUGE-2** results on CNN/DM and XSum test sets, both in 100-shot and full-shot. ROUGE-1 and ROUGE-L results are in Appendix C. 100-shot results are averaged over three random seeds.

4.3 Model Analysis

To better understand the contribution of each design choice to PromptSum performance, we experiment with multiple variants of the model. Specifically, we relax each of the following components:

- **Freezing weights**: whether to freeze the backbone PLM weights or not.
- **Pre-training**: whether to perform the pre-training described in §3.3.
- **Entity chain**: whether to use the entity chain. Removing it from PromptSum falls back to standard soft prompt tuning (PT).
- **Oracle entities**: when using the entity chain, whether to use the ground truth entities. This is only applicable to PromptSum and not to the soft prompt tuning (PT) baseline.

We synthesize results from all these model ablations in Table 4 over all datasets, both in 100-shot and full-shot. We draw several important conclusions from this table. First, it is interesting to see that simply adding a learnable soft prompt can improve performance compared to standard fine-tuning (almost +3 ROUGE-2 on BillSum 100-shot). Then, standard soft prompt tuning only moderately benefits from the pre-training in full-shot setups (except for SAMSum). However, the gain is larger in few-shot: +0.74 for XSum, compared to -0.01 in full-shot. Next, we confirm that freezing the model weights maintains most of the performance, especially in few-shot, motivating our choice to work under the parameter-efficiency constraint. In fact, *PromptSum / full weights* performs *worse* than PromptSum on average. Lastly, unlike soft PT, PromptSum relies heavily on the pre-training and its performance drops by a lot when not using our pre-trained checkpoint, even when tuning all weights, and in both few-shot and full-shot. Our introduced prompt multi-task pre-training is necessary to maintain performance of parameter-efficient controllable summarization. In the following section, we showcase PromptSum capacity to control summaries, a key aspect of our model not reflected in the performance comparison with Soft PT and Fine-tuning variants from Table 4.

5 Summary Control

5.1 Causal Overview

As PromptSum incorporates entity chains as discrete prompts, it is able to explicitly control the summary generation by altering the entity chains. From a causal perspective demonstrated in Fig. 2, traditional models without entity planning only follow *link a*, where the output Y is conditioned only on the input text X. PromptSum, however, provides a more controllable and realistic two-stage process: first, PromptSum with the entity prompt predicts the entity chain E from the input text X. Then, the output summary Y is predicted given both X and E, utilizing *link c* combined with *link a*, which allows for additional controllability.

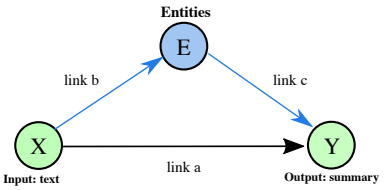

Figure 2: PromptSum's causal graph for controllability.

Model	Train Data	Success Rate(%)			
		K=1	K=2	K=5	Oracle entities
PromptSum	Full	80.5	**73.3**	**51.1**	**52.0**
CTRLsum		**91.3**	25.5	3.3	26.0
PromptSum	100-shot	**76.6**	**65.7**	**47.5**	**50.0**
– no pretraining		21.2	13.8	16.2	30.0
CTRLsum		44.9	28.1	4.2	26.0

Table 5: **Controllability** results on CNN/DM. Success rate is the percentage of generated summaries that mention all entities in the given entity-chain. $K = n$ denotes entity chain including n unique entities randomly selected from source input. *Oracle entities* denotes entity chain extracted from ground truth summary.

Dataset	Entities	Controlled	Hal. %	Mean R	F-1
CNN/DM	Non-Hal.	N	2.5	25.2	35.5
	Hal.	N	9.5	**26.2**	**33.3**
	Hal.	Y	**3.6**	24.9	32.9
XSum	Non-Hal.	N	14.8	19.5	37.4
	Hal.	N	62.7	**20.9**	**32.3**
	Hal.	Y	**48.8**	19.3	28.8
BillSum	Non-Hal.	N	28.1	30.5	33.2
	Hal.	N	39.2	**32.6**	**27.3**
	Hal.	Y	**31.9**	31.9	26.8
SAMSum	Non-Hal.	N	0	14.9	54.0
	Hal.	N	22.7	14.6	41.0
	Hal.	Y	**4.6**	**15.0**	**46.5**

Table 6: **Hallucination** results on each dataset with PromptSum trained in 100-shot. **Hal. %** stands for Hallucination, and lower is better. **F-1** is computed between generated entities and entities in ground truth summaries.

5.2 Intervention on the Entity Chains

To test controllability on entities (the strength of *link c*), we perform the following experiments with intervention on the entity chain: We sample 100 documents from test set of CNN/DM. On these documents, we randomly choose K entities from source texts or use oracle entities extracted from ground truth summaries, and compute the success rate that all the selected entities appear in the generated summaries. We report controllability results of PromptSum and CTRLsum (He et al., 2020) both on 100-shots and full-shot of CNN/DM in Table 5. We can see that PromptSum is significantly more controllable than CTRLsum in 100-shots for all settings, and in full shots when $K \geq 2$. When K increases, CTRLsum quickly loses its ability to condition on all entities while PromptSum is able to maintain a strong performance. This shows that PromptSum is significantly better at conditioning on multi-entities than CTRLsum while having three order of magnitude less parameters. Besides, PromptSum without pre-training struggles to effectively condition on entities, reaffirming our conclusion in Section 4.3 that pre-training is critical for PromptSum.

To better understand how PromptSum conditions on multiple entities, we show some qualitative examples in Table 7. The examples demonstrate that PromptSum can produce summaries focusing on different aspects, conditioned on the given entity chain, while still producing fluent, factual and informative summaries. This controllability has the potential of being useful in many downstream tasks, such as producing summaries on different topics, expanding summary candidates, and controlling hallucinations. We will next investigate controlling hallucinations as a critical use case for abstractive summarization.

5.3 Controlling Hallucinations

As hallucinations, *i.e.,* generating entities that do not appear in the inputs, are becoming an important threat to the factuality in summarization. Detecting (Maynez et al., 2020), preventing (Narayan et al., 2021) and analyzing (Cao et al., 2022) hallucinations remain major open problems in abstractive summarization research. We now show that the controllable PromptSum model has the potential of mitigating hallucinations, even under strong parameter-efficiency and data-efficiency constraints.

To test this hypothesis, we conduct the following experiments: given the prompts learned from 100-shot entity and summary tuning, we divide the test set $D^{te} = \{(X_1, E'_1, Y_1), \ldots (X_n, E'_n, Y_n)\}$ into two subsets: a set where the predicted salient entities E' are not hallucinated (D^{te}_{nh}), and another set where at least one entity of E' is hallucinated (D^{te}_{h}). Then, we perform summarization inference on both sets as-is and document the percentage of hallucinated entities in the predicted summaries. To test PromptSum's ability to *reduce hallucinations*, we also infer on D^{te}_{h} with non-hallucinated entity chains, which are produced by manually removing hallucinated entities from E'. If the hypothesis holds, the hallucination percentage should decrease after such controlling.

The hallucination experiment results are shown in Table 6. On all four datasets, the set D^{te}_{h}, where the prompt-tuned tagger generated hallucinated entities, contains a significantly higher percentage of hallucinations in generated summaries. This behavior shows that the summarization model has

Entity Chain	Summary
Ground Truth	Three passengers report a loss of consciousness on SkyWest flight . But officials say there is no evidence of a pressurization problem .
Buffalo	The flight attendant told passengers that the plane was going to make an emergency landing in Buffalo because of a "pressurization issue," according to a passenger on board
Cunningham, 84	Cunningham, who was on board with 84 other passengers, said the flight attendant told her to administer oxygen to a woman near the middle of the plane.
SkyWest Airlines, FAA	A SkyWest Airlines spokeswoman said in a statement that the airline is cooperating with the FAA's investigation into the incident, but declined to comment further.
Buffalo, 84, FAA	SkyWest Flight 5622 made an emergency landing in Buffalo after a number of passengers became ill on board the plane, which was carrying 84 people. The FAA said it is aware of media...
Ground Truth	Joel Parker, 33, was riding the bus in St Johns County, Florida . Police said he threatened the driver and was disruptive during the ride ...
33, St Johns County	Police said the 33-year-old was about to get off the Sunshine Bus in St Johns County when he asked the driver if he would like a Snickers bar.
250, Joel Parker	He posted a $250 bond and was issued a trespass warning, but he is never allowed to use the bus again, according to police. Joel Parker was arrested for ...
none	The driver told police that the man had been "disruptive" during the ride, and when he asked him if he would like a Snickers bar, he threatened to ...

Table 7: **Controllability** qualitative examples on CNN/DM 100-shot. **Ground Truth**: target summaries.

explicitly learned to include entity prompts in generated summaries, which further supports the observations in §5.2. Then, after manually removing the hallucinated entities from entity chain prompts E' (shown as Controlled Y in the table), the hallucination percentage decreased significantly (by 5.9 to 18.1 points) on all four datasets, showing promising abilities to control hallucinations.

6 Discussion

To better understand the quality of the summaries generated by PromptSum, we asked five humans to compare 50 random outputs generated by PromptSum and PEGASUS on CNN/DM, with both models trained on 100 data points. The evaluators are volunteers, who are graduate students with professional English competency.

Summary aspect evaluated	Informative	Factually consistent	Relevant	Fluent	Coherent
PEGASUS / PromptSum / Tie	19.6 / 23.4 / 7.0	11.2 / 11.6 / 20.0	21.0 / 22.6 / 6.4	17.6 / 18.0 / 14.4	15.6 / 15.2 / 19.2
Human agreement*	62.8%	64.8%	62.8%	59.6%	52.4%

Table 8: **Human evaluation** on CNN/DM, comparing PEGASUS with PromptSum in 100-shot.
* We follow the procedure in (Durmus et al., 2020) to compute human agreement.

Results of this human evaluation are shown in Table 8. Both models are on par with regards to factual consistency, fluency and coherence (as shown by similar scores and high number of ties), but summaries produced by PromptSum are more informative and more relevant. We believe that these results are due to the additional entity chain, which PromptSum conditions on, making the generated summary more likely to cover key aspects embodied through entities.

We also investigate the impact of PromptSum on the abstractive nature of generated summaries. As seen in Appendix D, intriguingly, soft prompt-tuning makes summaries more extractive, but PromptSum pushes back up abstractiveness on CNN/DM, BillSum and SAMSum.

PromptSum is the first model enabling controllability in summarization while being parameter-efficient and using few-shot supervision. Its performance is competitive in few-shot, especially on XSum, where it reaches state-of-the-art. In full-shot, PromptSum remains behind leading models tuning all parameters. This is inline with the findings from (Lester et al., 2021) that prompt tuning requires a larger backbone model (10B parameters) to match full-shot fine-tuning performance. PromptSum provides an easy-to-use and strong control mechanism of the summary through the input entity chain, outperforming CTRLSum in few-shot and some full-shot setups. Besides, pruning hallucinated entities from the input chain drastically reduces hallucinations in the output summary.

As we saw in §4.3 our tailored pre-training is necessary to unlock PromptSum's full capacity. We highlight that our introduced pre-training using soft prompts could be extended to other parameter-efficient multi-tasking. Future work includes studying the use of other signals than entities (e.g., salient sentences) in the discrete prompt.

REFERENCES

Ojas Ahuja, Jiacheng Xu, Akshay Gupta, Kevin Horecka, and Greg Durrett. Aspectnews: Aspect-oriented summarization of news documents. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6494–6506, 2022.

Regina Barzilay and Mirella Lapata. Modeling local coherence: An entity-based approach. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pp. 141–148, Ann Arbor, Michigan, 2005. Association for Computational Linguistics.

Qiwei Bi, Haoyuan Li, and Hanfang Yang. Boosting few-shot abstractive summarization with auxiliary tasks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 2888–2893, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Meng Cao, Yue Dong, and Jackie Chi Kit Cheung. Hallucinated but factual! inspecting the factuality of hallucinations in abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3340–3354, 2022.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 152–161, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1015. URL https://aclanthology.org/P18-1015.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. Gsum: A general framework for guided neural abstractive summarization. *arXiv preprint arXiv:2010.08014*, 2020.

Esin Durmus, He He, and Mona Diab. Feqa: A question answering evaluation framework for faithfulness assessment in abstractive summarization. *arXiv preprint arXiv:2005.03754*, 2020.

Alexander R Fabbri, Simeng Han, Haoyuan Li, Haoran Li, Marjan Ghazvininejad, Shafiq Joty, Dragomir Radev, and Yashar Mehdad. Improving zero and few-shot abstractive summarization with intermediate fine-tuning and data augmentation. *arXiv preprint arXiv:2010.12836*, 2020.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Luis C Lamb, Leo de Penning, BV Illuminoo, Hoifung Poon, and COPPE Gerson Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. *Neuro-Symbolic Artificial Intelligence: The State of the Art*, 342:1, 2022.

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

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. PPT: pre-trained prompt tuning for few-shot learning. In *ACL (1)*, pp. 8410–8423. Association for Computational Linguistics, 2022.

Junxian He, Wojciech Kryściński, Bryan McCann, Nazneen Rajani, and Caiming Xiong. Ctrlsum: Towards generic controllable text summarization. *arXiv preprint arXiv:2012.04281*, 2020.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Anastassia Kornilova and Vlad Eidelman. Billsum: A corpus for automatic summarization of us legislation. *arXiv preprint arXiv:1910.00523*, 2019.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, et al. Datasets: A community library for natural language processing. *arXiv preprint arXiv:2109.02846*, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *arXiv preprint arXiv:2205.05638*, 2022a.

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

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021b. URL https://arxiv.org/abs/2103.10385.

Xiaochen Liu, Yu Bai, Jiawei Li, Yinan Hu, and Yang Gao. Psp: Pre-trained soft prompts for few-shot abstractive summarization. *arXiv preprint arXiv:2204.04413*, 2022b.

Zhengyuan Liu and Nancy F Chen. Controllable neural dialogue summarization with personal named entity planning. *EMNLP*, 2021.

Zhengyuan Liu, Ke Shi, and Nancy F Chen. Conditional neural generation using sub-aspect functions for extractive news summarization. *EMNLP Findings*, 2020.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*, 2020.

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. *arXiv preprint arXiv:1808.08745*, 2018.

Shashi Narayan, Yao Zhao, Joshua Maynez, Gonçalo Simões, Vitaly Nikolaev, and Ryan McDonald. Planning with learned entity prompts for abstractive summarization. *Transactions of the Association for Computational Linguistics*, 9:1475–1492, 2021.

Shashi Narayan, Gonçalo Simões, Yao Zhao, Joshua Maynez, Dipanjan Das, Michael Collins, and Mirella Lapata. A well-composed text is half done! composition sampling for diverse conditional generation. *arXiv preprint arXiv:2203.15108*, 2022.

Ratish Puduppully, Li Dong, and Mirella Lapata. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 6908–6915, 2019.

Chengwei Qin and Shafiq Joty. Lfpt5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. *arXiv preprint arXiv:2110.07298*, 2021.

Chengwei Qin and Shafiq Joty. LFPT5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=HCRVf71PMF.

Guanghui Qin and Jason Eisner. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*, 2021.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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. *arXiv preprint arXiv:1910.10683*, 2019.

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 Association for Computational Linguistics: Main Volume*, pp. 255–269, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.20. URL https://aclanthology.org/2021.eacl-main.20.

Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, Lei Hou, Maosong Sun, and Jie Zhou. On transferability of prompt tuning for natural language processing. In *NAACL-HLT*, pp. 3949–3969. Association for Computational Linguistics, 2022.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Neil Houlsby, and Donald Metzler. Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*, 2022.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou', and Daniel Cer. Spot: Better frozen model adaptation through soft prompt transfer. In *ACL (1)*, pp. 5039–5059. Association for Computational Linguistics, 2022.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pp. 11328–11339. PMLR, 2020.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Extractive summarization as text matching. *arXiv preprint arXiv:2004.08795*, 2020.

A STANDARD DEVIATIONS

In Table 9, we report the standard deviations over the three random seeds used in each few-shot setup, for our models from Table 2: baseline T5-large, BART-large, and PEGASUS-large models, as well as PromptSum.

Dataset	**Model**	**1-shot**			**10-shot**			**100-shot**		
		R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
CNN/DM	T5-large (ours)	1.76	0.56	1.50	0.95	0.55	0.84	0.26	0.14	0.14
	BART-large (ours)	3.73	0.94	3.07	1.88	0.70	0.98	0.23	0.21	0.72
	PEGASUS-large (ours)	0.06	0.06	0.08	0.43	0.19	0.42	0.38	0.14	0.38
	PromptSum	0.63	0.42	0.60	0.63	0.45	0.63	0.82	0.59	0.75
XSum	T5-large (ours)	0.41	0.16	0.21	1.46	1.00	1.45	1.59	0.52	0.91
	BART-large (ours)	0.79	0.87	1.01	4.17	2.11	3.06	0.25	0.11	0.19
	PEGASUS-large (ours)	0.01	0.01	0.03	5.77	4.14	5.17	0.91	0.69	0.82
	PromptSum	0.08	0.12	0.19	0.85	0.49	0.72	0.19	0.29	0.40
BillSum	T5-large (ours)	9.71	2.86	5.89	2.22	5.06	3.72	0.53	0.31	0.47
	BART-large (ours)	4.21	1.59	2.00	2.61	1.01	0.71	1.59	0.34	0.74
	PEGASUS-large (ours)	1.69	0.91	0.69	1.52	0.20	0.34	0.67	0.12	0.26
	PromptSum	0.36	0.24	0.12	0.25	0.21	0.31	0.85	1.09	0.42
SAMSum	T5-large (ours)	0.38	0.20	0.33	0.71	0.33	0.33	2.64	0.41	1.15
	BART-large (ours)	5.70	0.85	2.80	3.50	1.79	1.76	0.53	0.69	1.04
	PEGASUS-large (ours)	0.00	0.00	0.00	1.76	1.06	1.32	1.64	0.91	1.08
	PromptSum	0.22	0.04	0.14	0.43	0.11	0.54	0.72	0.34	0.59

Table 9: **Few-shot standard deviations**. For each dataset and in each few-shot setup, we train a model on three different subsets of the training set of the corresponding few-shot size, and report the standard deviation over the tree models for ROUGE-1, ROUGE-2 and ROUGE-L.

As expected, standard deviations in 100-shot are significantly lower than in 10-shot. We noticed extreme variance in some 1-shot cases.

B ENTITY EVALUATION

In Table 10, we conduct an entity-focused evaluation over all our model variants introduced in Table 4. Specifically, we single out the entity chain from generated summaries with spacy, and compare it with the entity-chain from the ground truth using Precision, Recall, and F1-Score.

Dataset	Model	Freezing weights?	Pre-training?	Entity chain?	Oracle entities?	Trainable Params (M)	100-shot			Full-shot		
							Precision	Recall	F1	Precision	Recall	F1
CNN/DM	Fine-tuning	✗	✗	✗	✗	568	38.15	37.41	35.27	41.61	43.96	40.49
	Soft prompt-tuning / full weights	✗	✗	✗	✗	568	37.84	37.26	35.24	42.79	44.95	**41.63**
	Soft prompt-tuning / full weights (pre-trained)	✗	✓	✗	✗	568	38.71	35.86	34.78	42.61	**45.08**	**41.62**
	PromptSum no pre-training / full weights	✗	✗	✓	✗	568	25.65	20.73	20.90	24.17	17.42	18.47
	PromptSum / full weights	✗	✓	✓	✗	568	**40.28**	34.61	34.74	**47.36**	28.13	32.91
	PromptSum-oracle / full weights	✗	✓	✓	✓	568	36.49	**39.92**	**35.62**	42.68	**45.10**	**41.69**
	Soft prompt-tuning	✓	✗	✗	✗	0.3	36.90	37.08	34.43	40.18	39.73	37.45
	Soft prompt-tuning (pre-trained)	✓	✓	✗	✗	0.3	37.94	35.77	34.23	41.00	40.68	38.45
	PromptSum no pre-training	✓	✗	✓	✗	0.6	29.08	24.53	24.17	42.23	38.21	37.61
	PromptSum	✓	✓	✓	✗	0.6	**40.36**	35.19	35.15	45.70	36.25	38.10
	PromptSum-oracle	✓	✓	✓	✓	0.6	37.99	35.40	34.06	42.74	39.09	38.35
XSum	Fine-tuning	✗	✗	✗	✗	568	37.74	36.98	35.72	46.03	43.09	42.84
	Soft prompt-tuning / full weights	✗	✗	✗	✗	568	38.05	**37.21**	35.95	46.11	**43.55**	**43.12**
	Soft prompt-tuning / full weights (pre-trained)	✗	✓	✗	✗	568	42.50	36.93	**37.82**	**46.22**	43.25	43.04
	PromptSum no pre-training / full weights	✗	✗	✓	✗	568	22.20	21.94	20.82	31.66	29.52	28.97
	PromptSum / full weights	✗	✓	✓	✗	568	**43.82**	34.32	36.74	45.99	35.60	38.32
	PromptSum-oracle / full weights	✗	✓	✓	✓	568	38.55	35.27	35.14	45.80	43.15	42.73
	Soft prompt-tuning	✓	✗	✗	✗	0.3	35.89	38.46	34.98	40.98	38.96	38.26
	Soft prompt-tuning (pre-trained)	✓	✓	✗	✗	0.3	42.92	35.17	36.97	42.41	38.70	38.80
	PromptSum no pre-training	✓	✗	✓	✗	0.6	25.79	26.43	24.59	38.50	37.18	36.22
	PromptSum	✓	✓	✓	✗	0.6	43.56	33.83	36.32	45.84	35.78	38.55
	PromptSum-oracle	✓	✓	✓	✓	0.6	40.28	33.96	35.17	41.76	38.19	38.21
BillSum	Fine-tuning	✗	✗	✗	✗	568	48.80	42.81	41.37	58.98	54.81	52.41
	Soft prompt-tuning / full weights	✗	✗	✗	✗	568	48.27	42.48	40.97	60.61	54.87	53.14
	Soft prompt-tuning / full weights (pre-trained)	✗	✓	✗	✗	568	50.75	44.09	42.73	61.15	**55.55**	**53.73**
	PromptSum no pre-training / full weights	✗	✗	✓	✗	568	32.58	27.69	26.66	60.49	41.13	44.45
	PromptSum / full weights	✗	✓	✓	✗	568	**52.89**	38.20	40.08	45.73	30.61	32.26
	PromptSum-oracle / full weights	✗	✓	✓	✓	568	50.09	**44.47**	**42.86**	**62.54**	52.38	52.30
	Soft prompt-tuning	✓	✗	✗	✗	0.3	37.54	24.33	25.82	54.98	42.53	43.04
	Soft prompt-tuning (pre-trained)	✓	✓	✗	✗	0.3	42.03	30.45	31.05	56.16	41.38	42.62
	PromptSum no pre-training	✓	✗	✓	✗	0.6	29.06	16.52	18.09	51.61	42.46	42.03
	PromptSum	✓	✓	✓	✗	0.6	42.02	30.67	31.22	52.39	33.91	36.95
	PromptSum-oracle	✓	✓	✓	✓	0.6	38.48	31.02	29.92	60.23	37.65	41.59
SAMSum	Fine-tuning	✗	✗	✗	✗	568	66.12	57.92	58.57	66.90	66.17	63.71
	Soft prompt-tuning / full weights	✗	✗	✗	✗	568	65.47	57.97	58.34	**71.53**	69.65	**68.08**
	Soft prompt-tuning / full weights (pre-trained)	✗	✓	✗	✗	568	67.40	59.34	60.02	69.80	69.85	67.38
	PromptSum no pre-training / full weights	✗	✗	✓	✗	568	58.30	48.33	49.73	54.85	49.99	49.67
	PromptSum / full weights	✗	✓	✓	✗	568	67.22	58.11	59.29	70.25	61.42	62.51
	PromptSum-oracle / full weights	✗	✓	✓	✓	568	65.91	**61.22**	**60.57**	70.34	**70.64**	**68.14**
	Soft prompt-tuning	✓	✗	✗	✗	0.3	62.92	50.91	53.06	64.95	53.64	55.75
	Soft prompt-tuning (pre-trained)	✓	✓	✗	✗	0.3	**67.55**	52.95	56.26	64.90	59.84	59.45
	PromptSum no pre-training	✓	✗	✓	✗	0.6	61.52	47.58	50.26	60.37	48.37	50.46
	PromptSum	✓	✓	✓	✗	0.6	**67.57**	53.67	56.90	66.88	59.38	59.94
	PromptSum-oracle	✓	✓	✓	✓	0.6	65.86	53.14	55.54	69.46	65.26	64.67

Table 10: **Entity evaluation** on the test set of all datasets. PromptSum and its variants (PromptSum-oracle, Prompt tune weights) tend to outperform other categories of model like Fine-tuning and Soft prompt-tuning.

C ANALYSIS RESULTS WITH ROUGE-1 AND ROUGE-L

In Table 11 and Table 12, we conduct the same model ablation as in Table 4 but report results with the ROUGE-1 and ROUGE-L metrics, respectively.

Model	Freezing weights?	Pre-training?	Entity chain?	Oracle entities?	Trainable Params (M)	CNN/DM		XSum		BillSum		SAMSum	
						100-shot	Full-shot	100-shot	Full-shot	100-shot	Full-shot	100-shot	Full-shot
Fine-tuning (PEGASUS-large)	✗	✗	✗	✗	568	39.30	43.26	41.72	**46.69**	46.01	55.04	41.97	52.01
Soft PT / full weights	✗	✗	✗	✗	568	39.37	44.15	42.04	**46.77**	46.64	55.70	42.53	51.76
Soft PT / full weights (pre-trained)	✗	✓	✗	✗	568	38.92	44.23	**43.34**	**46.85**	48.31	**55.82**	**43.30**	51.77
PromptSum / full weights (no pre-training)	✗	✗	✓	✗	568	29.16	29.80	30.69	39.90	38.22	50.84	38.22	43.74
PromptSum / full weights	✗	✓	✓	✗	568	37.57	38.71	41.99	43.85	45.61	48.39	40.79	47.53
PromptSum-oracle / full weights	✗	✓	✓	✓	568	**39.93**	**44.34**	41.24	46.43	**48.75**	53.81	**43.38**	**52.21**
Soft PT	✓	✗	✗	✗	0.3	38.60	40.96	40.78	44.45	37.23	45.80	35.79	44.64
Soft PT (pre-trained)	✓	✓	✗	✗	0.3	39.05	41.66	42.53	44.35	44.33	47.11	41.09	48.59
PromptSum (no pre-training)	✓	✗	✓	✗	0.6	31.67	39.83	34.31	43.29	32.27	46.56	32.36	39.95
PromptSum	✓	✓	✓	✗	0.6	37.79	38.16	41.73	43.12	43.96	45.20	39.43	47.43
PromptSum-oracle	✓	✓	✓	✓	0.6	38.66	40.92	41.91	44.19	43.12	44.48	41.10	48.60

Table 11: **Model analysis (ROUGE-1 results)**.

Model	Freezing weights?	Pre-training?	Entity chain?	Oracle entities?	Trainable Params (M)	CNN/DM		XSum		BillSum		SAMSum	
						100-shot	Full-shot	100-shot	Full-shot	100-shot	Full-shot	100-shot	Full-shot
Fine-tuning (PEGASUS-large)	✗	✗	✗	✗	568	35.81	39.92	33.38	**38.57**	33.24	42.95	33.67	43.27
Soft PT / full weights	✗	✗	✗	✗	568	**36.14**	40.86	33.71	38.47	32.66	43.45	33.80	43.53
Soft PT / full weights (pre-trained)	✗	✓	✗	✗	568	35.64	**41.02**	**34.88**	**38.58**	**34.52**	**43.73**	**34.99**	43.36
PromptSum / full weights (no pre-training)	✗	✗	✓	✗	568	26.45	27.30	23.66	32.15	24.62	39.79	30.31	36.60
PromptSum / full weights	✗	✓	✓	✗	568	34.13	35.68	33.86	35.88	33.32	36.64	32.98	38.74
PromptSum-oracle / full weights	✗	✓	✓	✓	568	35.93	**41.03**	32.84	38.16	34.25	42.45	34.81	**43.76**
Soft PT	✓	✗	✗	✗	0.3	34.96	37.65	32.48	36.08	26.47	35.94	29.28	37.29
Soft PT (pre-trained)	✓	✓	✗	✗	0.3	35.62	38.40	34.26	36.24	31.04	37.19	34.55	40.83
PromptSum (no pre-training)	✓	✗	✓	✗	0.6	28.39	36.52	26.75	34.89	22.69	35.03	26.30	32.46
PromptSum	✓	✓	✓	✗	0.6	34.40	35.08	33.55	35.18	30.81	35.24	32.86	39.48
PromptSum-oracle	✓	✓	✓	✓	0.6	35.21	37.81	33.66	36.02	30.81	36.42	33.96	40.99

Table 12: **Model analysis (ROUGE-L results)**.

D ABSTRACTIVENESS

In Fig. 3, we analyze the level of abstractiveness in generated summaries through percentage counts of novel n-grams. Interestingly, Soft prompt-tuning is less abstractive than the Fine-tuning baseline on CNN/DM, BillSum, and SAMSum, but PromptSum pushes back abstractiveness level higher. On the very abstractive XSum dataset, we do not observe such pattern, but all models are already very abstractive.

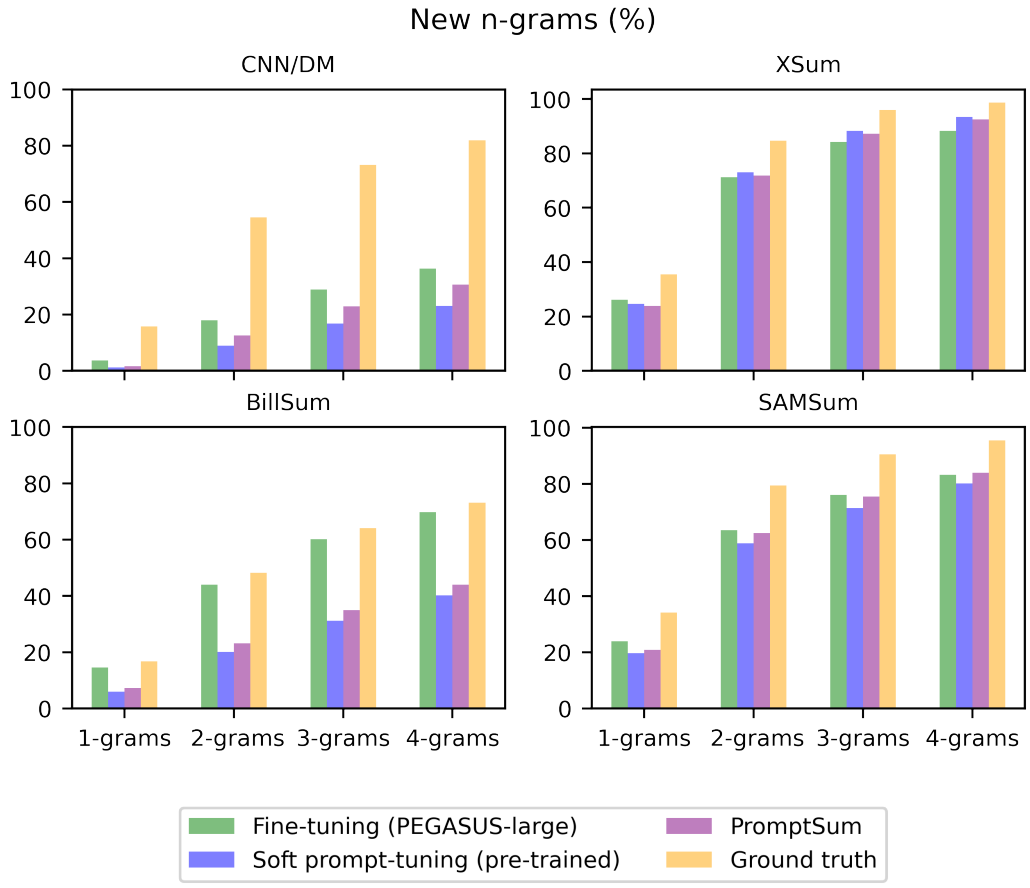

Figure 3: **Abstractiveness.** We show the percentage of new n-grams (being present in the output summary but not in the source text) for n=1,2,3,4 on the test of each dataset for Fine-tuning, Soft prompt-tuning (with our pre-training), and PromptSum, all trained in 100-shot. Results are averaged over three random seeds.