Two-Stage Movie Script Summarization: An Efficient Method For Low-Resource Long Document Summarization

Dongqi Pu*, **Xudong Hong***†, **Pin-Jie Lin***, **Ernie Chang** and **Vera Demberg**
†Max Planck Institute for Informatics
Saarland University, Saarland Informatics Campus, Germany
{dongqipu,pinjie}@lst.uni-saarland.de
{xhong,cychang,vera}@coli.uni-saarland.de

Abstract

The Creative Summarization Shared Task at COLING 2022 aspires to generate summaries given long-form texts from creative writing. This paper presents the system architecture and the results of our participation in the Scriptbase track that focuses on generating movie plots given movie scripts. The core innovation in our model employs a two-stage hierarchical architecture for movie script summarization. In the first stage, a heuristic extraction method is applied to extract actions and essential dialogues, which reduces the average length of input movie scripts by 66% from about 24K to 8K tokens. In the second stage, a state-of-the-art encoder-decoder model, Longformer-Encoder-Decoder (LED), is trained with effective fine-tuning methods, *BitFit* and *NoisyTune*. Evaluations on the unseen test set indicate that our system outperforms both zero-shot LED baselines as well as other participants on various automatic metrics and ranks 1st in the Scriptbase track[1].

1 Introduction

The goal of the Creative Summarization Shared Task 2022 (ASCW@COLING'22) is to automatically generate summaries based on long-form creative texts like literature, movie scripts, or TV screenplays. This task is encouraged by practical settings in part by the condition to reflect the information that is realistically available in real-world natural language generation task – realistic texts like movie scripts and plot summaries can be *prohibitively long* (See *Movie Script* and *Plot Summary* in Figure 1).

Current dominant neural approaches to long document summarization (Zhang et al., 2020; Xiao et al., 2022; Guo et al., 2022) mainly embrace neural sequence-to-sequence architectures consisting of an encoder-decoder setup where the entire input sequence is first encoded before decoding the output sequence autoregressively. While the encoder-decoder architecture is triumphant in natural language generation tasks (Peng et al., 2020a; Akermi et al., 2020; Erdem et al., 2022; Qian et al., 2022), it is not without its challenges, some of which are exacerbated in this shared task.

In particular, the challenges in this shared task stem from the inherent characteristics of the corpus, which consists of not only texts (i.e. scripts of 24K tokens) that are much longer than the context length of current state-of-the-art long document summarizers (e.g. 16K in LongT5 (Guo et al., 2022)), but also requires that the decoder be exploited of its long-term attention capabilities to an extreme extent and generate up to summaries of 1K tokens. The excessively long decoding time made it impossible to experiment with different model architectures and perform extensive hyperparameter tuning for model selection. Therefore, the optimization space during training time is severely limited; and the inference step becomes highly time-consuming for experimentation during the trial and error process.

To this end, our system employs a two-stage procedure for movie script to movie plot summarization. In the first stage, we propose to heuristically extract sentences that effectively reduce the average input length from 24K to 8K tokens. Next, we propose to improve upon the Longformer-based encoder-decoder model (LED) (Beltagy et al., 2020) by coupling it with two effective fine-tuning methods, i.e., BitFit (Ben Zaken et al., 2022) where only the bias-terms (0.09% of the parameters) of the model are being updated and NoisyTune (Wu et al., 2022) where employs a matrix-level perturbation strategy to increase the variation amplitude of the parameters.

∗ These authors contributed equally to this work.

[1]Source code and pre-trained models are available at: https://github.com/tony-hong/script-2-story

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 57–66
July 5–10, 2020.
https://doi.org/10.26615/978-954-452-056-4_009

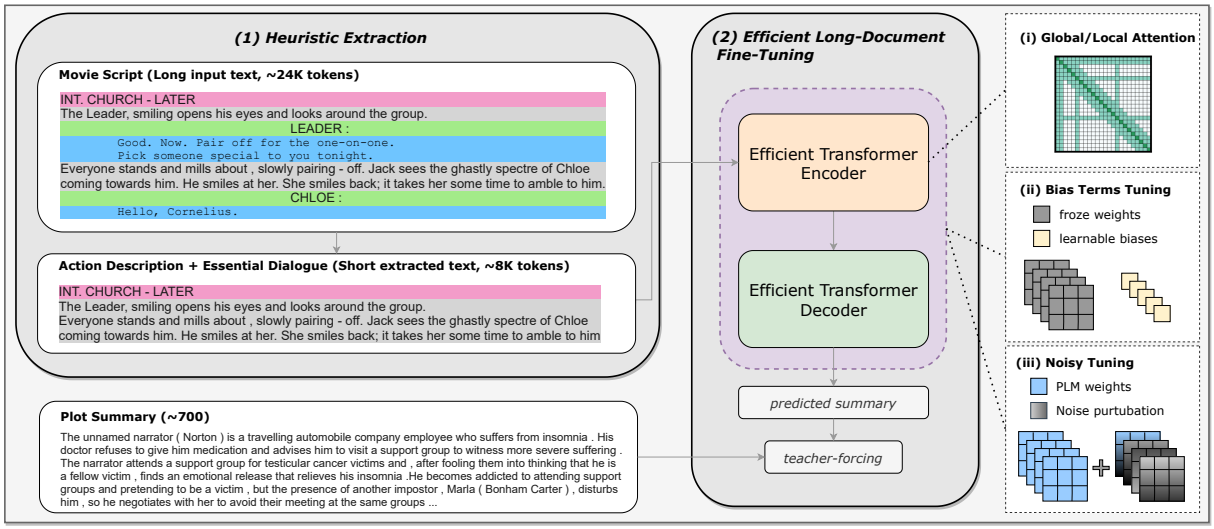

Figure 1: Diagram to depict the pipeline. Our system achieves efficient text summarization by (1) heuristics extraction method to compress long input text into a relatively short input sequence, and (2) efficient long-document fine-tuning. In the first stage, the heuristics extraction method takes (a) long movie scripts with an average length of 24K and compress them into 8K. (b) Following that, the transformer encoder with (i) global sliding winding attention takes movie scripts up to 8K tokens and is jointly fine-tuned with the decoder using (ii) bias-terms tuning or (iii) noisy tuning techniques to generate a long movie summarization (1K) in a computationally efficient manner.

Further, we start to tease apart the intricate relationship between decoding beam size, model performance, and maximum encoder and decoder lengths in our ablation studies (see Section 3.6). We examine the trade-offs between performance and decoding runtime, and empirically find that beam search size = 4 is the most suitable.

To summarize, our contributions are as follows:

1. We describe the long-form challenge in this shared task as a challenge that impacts not only the model performance but also the model training and evaluation.

2. We propose a two-stage solution to solving this challenge by first reducing its average input length, and then incorporating the Longformer architecture with either a simple yet effective finetuning technique (BitFit) or a matrix-wise perturbing method for finetuning (NoisyTune).

3. We are the first to apply the transformer-based model for summarization on this dataset (Scriptbase), and our model ranks 1st on metrics including ROUGE, BERTScore, and N-gram diversities.

2 Our Approach

Our method is a two-stage summarization method where the 1st stage is a heuristic extraction method (Sec. 2.1) and the 2nd stage is neural seq2seq summarization model (Sec. 2.2).

2.1 Heuristic Extraction Method

Because the average input length (24K) is way beyond the maximum context length of SOTA long-text summarization models, e.g., 16K in LongT5 (Guo et al., 2022). We are required to first reduce input text length by applying heuristics about what parts of the input to drop.

Specifically, for each of the movie scripts, which consist of two elements: (1) *action* (red and grey in Figure 1*(1)*), descriptions of events or expressions that can be heard by the audiences; (2) *dialogue* (green and blue in Figure 1*(1)*), conversations between characters. We first identify all the titles and texts of actions and dialogues. Then we extract the titles and texts of all actions with regular expression because they deliver essential information about the movies. However, some important concepts are only in the dialogues (Gorinski and Lapata, 2015). According to the narrative structures of movie scripts (Lee et al., 2021), when a new character occurs, the first few dialogues contain introductory concepts about this character. So our heuristics also include these essential dialogues in our input.

2.2 Long Document Encoder-Decoder

Bottleneck of Transformer Transformer-based models, based on the multi-head self-attention (MHSA) mechanism (Vaswani et al., 2017), are allowed to simultaneously attend to the con-

Dataset	R-1	R-2	R-L	BS-P	BS-R	BS-F1	S	1-G	2-G
LED-1024	13.68	1.25	12.77	43.22	39.24	40.99	0.00	30.24	72.11
LED-4096	14.16	1.30	12.99	42.45	41.37	41.79	0.00	30.92	72.73
LED-16384	14.92	1.46	13.73	42.98	42.38	42.58	0.00	33.57	74.85
MovING	41.44	8.23	39.63	51.63	52.33	51.94	**4.76**	16.21	48.27
UdS	**46.39**	11.52	**44.08**	57.03	56.72	56.86	2.69	34.56	76.04
UdS NoisyTune	46.34	**11.58**	44.05	**57.23**	**56.80**	**57.00**	2.50	**35.20**	**76.36**
UdS BitFit	45.76	**11.58**	43.80	56.90	56.20	56.53	3.01	31.67	74.59

Table 1: Results of the proposed model on unseen test set compared to other systems using automatic metrics including ROUGE-1 F1 (R-1), ROUGE-2 F1 (R-2), ROUGE-L F1 (R-L), BERTScore-Precision (BS-P), BERTScore-Recall (BS-R), BERTScore-F1 (BS-F1), SummaCZS (S), Novel 1-grams (1-G) and Novel 2-grams (2-G).

text at different positions from different representation subspaces: $\mathrm{MHSA}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = [\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_h]\mathbf{W}^O + \mathbf{B}^O$ where $\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i \in \mathbb{R}^{t\times d}$ are the input attention matrices, t is the sequence length, d_m is the model embedding dimension, $\mathbf{A}_i$ is the i-th attention head, h is the number of attention heads, and $\mathbf{W}^O \in \mathbb{R}^{d_m\times d_m}$ is the parameter matrix and $\mathbf{B}^O \in \mathbb{R}^{t\times d_m}$ is the bias term. Each attention head is defined as:

$$\mathbf{A}_i = \sigma\underbrace{(\frac{\mathbf{Q}_i(\mathbf{K}_i)^{\mathrm{T}}}{\sqrt{d}})}_{\Phi}\mathbf{V}_i \tag{1}$$

where

$$\mathbf{Q}_i = \mathbf{X}\mathbf{W}_i^Q + \mathbf{B}_i^Q,$$
$$\mathbf{K}_i = \mathbf{X}\mathbf{W}_i^K + \mathbf{B}_i^K,$$
$$\mathbf{V}_i = \mathbf{X}\mathbf{W}_i^V + \mathbf{B}_i^V,$$

$\mathbf{X} \in \mathbb{R}^{t\times d_m}$ represents input embedding, $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d_m\times d}$, $\mathbf{B}_i^Q, \mathbf{B}_i^K, \mathbf{B}_i^V \in \mathbb{R}^{t\times d}$ are bias terms. $d = d_m/h$. σ is the *softmax* function. The input to the *softmax* function can be represented by $\Phi \in \mathbb{R}^{t\times t}$. Because the computational complexity of Φ is $O(t^2)$, which is very expensive, it becomes the main bottleneck of the Transformer model for dealing with long sequences.

Sparse Attention Due to the problem of the original Transformer's attention described above, for long documents, the common practice of previous works (Qiu et al., 2020; Pilault et al., 2020) is to slice the long sequence document into different blocks (which is usually limited to 512 tokens). The downside is that there is no interactive information between the sliced blocks, which causes valuable knowledge loss. Moreover, reducing the input sequence length does not inherently change the algorithm's complexity. In our task, we apply Longformer (Beltagy et al., 2020) to alleviate the computational problem by introducing a sparse attention mechanism consisting of three parts: sliding window attention, dilated sliding window attention, and global sliding window attention.

To be specific, for sliding windows, the $query$ at each location attends only to the $keys$ of the adjacent w locations, which is suitable for capturing the shallow local information. For the dilated sliding window, the $query$ of each position also attends to the $keys$ on w positions, but the position of interest is not adjacent but discontinuous. Dilated sliding attention can attend to non-proximity tokens, which is more suitable for capturing long-distance dependency.

Global attention is identical to the ordinary attention mechanism but only for specific tokens. A token with global attention is associated with every input token. Local tokens attend to the tokens in their own sliding window, and also to all global tokens. The essence of Sparse Attention is to reduce the number of tokens used to compute attention scores, thereby reducing the computational complexity.

2.3 Efficient Fine-Tuning

Most PLMs are highly likely to overfit on the pre-trained data because of the huge amount of parameters. When there is a large domain gap between pre-training and fine-tuning data, the model's parameters are difficult to adjust effectively during fine-tuning (Gao et al., 2021), because: (1) the parameters adjust only slightly during fine-tuning, which is often not sufficient to bridge the domain gap; (2) there is very limited training data for low-resource tasks, making it even harder to adjust many over-fitted parameters.

Parameter Variation To alleviate the first problem, we apply the NoisyTune (Wu et al., 2022)

Size	EFT	R-1	R-2	R-L
original				
2048		30.10	4.50	11.80
full		4.60	1.80	2.70
heuristics				
2048		30.40	4.80	12.10
full		13.60	3.30	6.30
LED				
base		40.80	9.75	16.50
base	BitFit	36.93	8.71	15.42
large		41.51	9.78	16.20
large	BitFit	40.41	10.45	16.37

Table 2: Results of the proposed model on validation set compared to other systems using automatic metrics including ROUGE-1 F1 (R-1), ROUGE-2 F2 (R-2), ROUGE-L F1 (R-L). **EFT** means efficient fine-tuning.

which employs a matrix-level perturbation strategy to increase the variation amplitude of the parameters to adapt the PLMs faster to the target domain on our low-resource data.

Parameter Efficiency Although previous work applied efficient Transformers strategies to reduce the theoretical complexity of the self-attention in long document summarization, how to efficiently utilize PLMs and adapt to new domain data is not explored. Mainly, how to fine-tune large PLMs under exceptionally low-resource settings (1K training samples in our case) using limited hardware resources (Nvidia V100 with 32GB memory) is not well explored. To experiment with parameter efficient fine-tuning method, we also apply BitFit (Ben Zaken et al., 2022), a method that only fine-tunes the bias terms (i.e. $\mathbf{B}^O, \mathbf{B}_i^Q, \mathbf{B}_i^K, \mathbf{B}_i^V$), on a pre-trained LED model checkpoint[2]. By reducing the number of trainable parameters, we aim to increase the fine-tuning speed.

3 Experiments and Analysis

3.1 Experimental Setup

We build our system using HuggingFace transformers (Wolf et al., 2019) and train LED on the training split of the Scriptbase dataset. We choose the checkpoint before over-fitting for evaluation. We limit the output length between 512 and 1024 tokens. For the rest, we follow the configuration from Longformer (Beltagy et al., 2020).

All experiments is optimized using AdamW (Loshchilov and Hutter, 2017) (where β_1 was 0.9, β_2 was 0.99, ϵ was 1e-8) and the initial learning rate is set to 5e-5 with weight decay of 0.01. The number of warm-up steps is 512. We enable mixed precision during training and evaluation to save memory for larger batch size. We use ROUGE in evaluation on validation split. All ROUGE scores are multiplied by 100.

3.2 Hidden Test Submission

For the test submission, we train our models on the training and validation splits of the Scriptbase dataset following the organizers' instructions. We train all the models either in pure fine-tuning or coupling with *NoisyTune* or *BitFit* method. Hence, we obtain 3 systems: `UdS`, `UdS NoisyTune` and `UdS BitFit` respectively.

For the sake of fairness, the results on the unseen test set are released by the organizing committee as shown in Table 1. Compared with the official baseline models, all candidate models have improved in various metrics. Particularly, `UdS NoisyTune` that introduces noise during fine-tuning performs the best overall. Among them, 6 of the 9 evaluation metrics achieved the best performance. The competitor's model (MovING) outperforms ours only on the SummaCZS metric (Laban et al., 2022), which is an evaluation metric that focuses on inconsistency in summaries. Furthermore, `UdS BitFit` that applies the BitFit algorithm to fine-tune only 0.09% of the parameters is very close to the `UdS` performance, but its more significant advantages lie in fewer computational parameters and shorter training time.

3.3 Baseline Comparison

Unfortunately, no previous work reports standard summarization metrics (like ROUGE) on the Scriptbase dataset (Gorinski and Lapata, 2015; Papalampidi et al., 2019, 2021; Lee et al., 2021). We therefore create a *naive* baseline by copying first M tokens from source sequence. We apply this baseline to both the original movie scripts (*original*) and the extracted texts using our heuristics (*heuristics*). As we increase M from 128 by a factor of 2 up until the length of the source text, we observe that the recall of ROUGE-1/2/L increase and the precision decrease as expected. We thus report the *naive* baseline with the highest F1 (2048) and with input full length in Table 2. Results show that the *naive* baseline using our heuristics outperforms the

[2]https://huggingface.co/allenai/led-base-16384

Input	Avg. Len.	Input Len.	%	R-2-P	R-2-R	R-2-F1
all	24106	8192	100%	10.46	11.34	10.26
all	24106	16384	100%	11.26	12.05	11.02
dialogue	15818	8192	66%	10.67	11.71	10.52
dialogue	15818	16384	66%	10.39	12.78	10.90
ours	8288	8192	34%	10.47	11.91	10.52

Table 3: Results of the model trained on data created with our heuristics extraction method compared to other extraction methods.

Models	Output
Gold	MI6 sends James Bond, agent 007, into the field to spy on a terrorist arms bazaar on the Russian border. Via television, MI6 and the Royal Navy identify several wanted men, including American "techno-terrorist" Henry Gupta, who is buying a GPS encoder made by the U.S. military...
BART	The film is being shot by the British pilot, as he attempts to escape, but is shot by a group of a helicopter. He is killed by a helicopter pilot, the helicopter. The pilot from the pilot who has a helicopter which the helicopter pilot who is killed...
T5	James Bond is a former MI6 operative, now working for MI-6. He was assigned to spy on Russian President Vladimir Vladimirovich Kirillov's election campaign. Bond is assigned to investigate the uranium scandal in Vladivostok....
Our	James Bond is a British Secret Service agent, nicknamed "Bond", who works for MI6. He is recruited by M, the head of the agency's counter-espionage unit, to investigate the disappearance of highly enriched uranium from a UK nuclear reactor...

Table 4: Case study of model-generated output. More examples can be found in our GitHub repo.

baseline using original text under the optimal setting (2048) and under the full length setting (full)[3].

3.4 Model Selection

Prior state-of-the-art Transformer models either fail to handle such long-range movie scripts or perform slowly during training time. For example, the maximum input length of BART (Lewis et al., 2020), PEGASUS (Zhang et al., 2020), T5 (Raffel et al., 2020) and BigBird (Zaheer et al., 2020) are all less than 4096. Among BART, T5 and LED, our benchmark comparison shows that LED achieves strong performance compared to BART by 2.4 points and is 8× times faster than T5 at the training stage. We thus develop our system based on LED. The detailed comparison is in Appendix A. We extend the original LED with two recent fine-tuning methods, BitFit and NoisyTune. Both model variations with these techniques achieve strong results on R-2 F1 scores. In addition, we set the encoding length as 8192 and the decoding length 1024 with beam size 4 (BS) for all our experiments. Figure 5 shows that UdS BitFit stops the performance improvement when $BS > 4$, but with a huge time complexity. Hence, we choose $BS = 4$.

[3]Figure 5 in Appendix B further shows that our submitted system outperforms all *naive* baselines by a large margin.

3.5 Case Study

Table 4 shows the first two sentences of the movie summary of "Tomorrow Never Dies" in the dataset. Where Gold is the human answer, we compared the output of our model with the current SOAT model BART and T5. We find that our model can effectively capture proper nouns in movies, such as characters, organizations, locations, etc., and the generated sentences are more in line with a reasonable story logic. However, the BART model seems to only be able to focus on a certain part of the plot in the movie and cannot summarize the movie well. T5 model often generates sentences that contradict the truth and has difficulty handling transitions between sentences.

3.6 Ablation Study

Input Data To further show the effectiveness of our heuristics extraction method, we conduct an ablation study where we train our best summarization model with three different inputs: the full movie scripts (all), the actions and dialogues that are selected by our heuristics (ours), and the dialogues that are omitted by our heuristics (dialogue). We also experiment with two input lengths (8192 and 16384). Results in Table 3 demonstrate that our heuristics extraction method reduces the input length significantly down to 33% of the original length with only 4.5% performance loss compared to the model using full scripts (all).

Performance and Decoding Time Trade-off To understand the dependency between model performance and runtime, we conduct the ablation study of testing the evaluation runtime when varying the beam size (BS). Figure 2 illustrates that UdS BitFit stops showing significant improvement in performance after $BS = 4$ but takes more decoding time than the total training time (red dotted line). Using large BS from 5 to 8 requires additional 1 to 6 hours yet only obtains 0 to 2% performance gain. This indicates the decoding with large BS

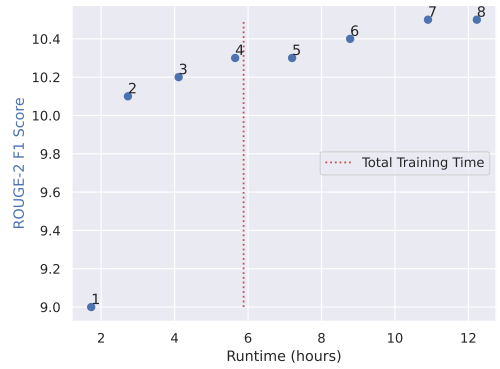

Figure 2: Performance (blue points) and evaluation runtime (hours) of UdS BitFit when varying in beam sizes (BS) from 1 to 8. The points are labeled with their BS. The total training time is marked by red dotted line. All UdS BitFit models are evaluated on the validation set.

is extremely expensive but unnecessary. We also provide further analysis to understand the impact of encoding and decoding lengths on performance and runtime in Appendix B.

4 Related Work

4.1 Efficient Transformers

Transformer-based models (Vaswani et al., 2017) are widely applied for text generation problems, but the $O(n^2)$ complexity of the attention calculation makes long document text generation computationally expensive and prohibitive. Various strategies have been proposed to ameliorate this issue (Correia et al., 2019; Child et al., 2019; Beltagy et al., 2020; Guo et al., 2022; Tay et al., 2020; Ainslie et al., 2020; Zaheer et al., 2020; Wang et al., 2020; Peng et al., 2020b; Dai et al., 2019). Most of these proposals demonstrate efficiency of their model on Long Range Arena (LRA), a benchmark of six simple tasks to evaluate the efficiency of different Transformers (Tay et al., 2021). However, only one of these tasks (Path-X) has an input length of 16K which is much longer than the input lengths of the other five tasks (mostly below 10K), and most of these methods failed on Path-X. Thus it is unclear whether the good performance on LRA can be transferred to more realistic downstream tasks like long-document summarization.

4.2 Long Document Summarization

Long document summarization is a trending natural language generation task. Existing solutions are principally divided into two directions: The first is a multi-stage strategy that reduces long input sequences while minimizing the loss of important details (Moro and Ragazzi, 2022; Zhang et al., 2022). The second improves the internal representation of the summarization model to process longer inputs efficiently (Zhang et al., 2020; Xiao et al., 2022; Mao et al., 2022; Cao and Wang, 2022). However, the above strategies are either domain-specific or pre-training corrections. Few people have explored effective fine-tuning strategies for long sequence large models in text summarization tasks, and the main content of our work is to fill this gap.

5 Conclusion

In this paper, we present the details of our system, which ranks 1st in the Scriptbase track on various metrics, including ROUGE, BERTScore, and N-gram diversities. We show that the proposed approach involving a two-stage solution results in competitive and efficient performance for long-form text encoding and generation. In addition, we deliver analysis and ablation studies for the components within our proposed techniques, which allows us to draw further conclusions about decoding configurations and vocabulary sampling. Lastly, we argue that more work can be done to speed up the model selection process, which impacts the model performance and model training and evaluation.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 948878). Xudong Hong is supported by the International Max Planck Research School for Computer Science (IMPRS-CS) of Max-Planck Institute for Informatics (MPI-INF).

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference*

on Empirical Methods in Natural Language Processing (EMNLP), pages 268–284, Online. Association for Computational Linguistics.

Imen Akermi, Johannes Heinecke, and Frédéric Herledan. 2020. Transformer based natural language generation for question-answering. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 349–359, Dublin, Ireland. Association for Computational Linguistics.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Shuyang Cao and Lu Wang. 2022. HIBRIDS: Attention with hierarchical biases for structure-aware long document summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 786–807, Dublin, Ireland. Association for Computational Linguistics.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. 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 2174–2184, Hong Kong, China. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Erkut Erdem, Menekse Kuyu, Semih Yagcioglu, Anette Frank, Letitia Parcalabescu, Barbara Plank, Andrii Babii, Oleksii Turuta, Aykut Erdem, Iacer Calixto, et al. 2022. Neural natural language generation: A survey on multilinguality, multimodality, controllability and learning. *Journal of Artificial Intelligence Research*, 73:1131–1207.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. 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 3816–3830, Online. Association for Computational Linguistics.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1066–1076, Denver, Colorado. Association for Computational Linguistics.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. LongT5: Efficient text-to-text transformer for long sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736, Seattle, United States. Association for Computational Linguistics.

Philippe Laban, Tobias Schnabel, Paul N. Bennett, and Marti A. Hearst. 2022. SummaC: Re-visiting NLI-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, 10:163–177.

Myungji Lee, Hongseok Kwon, Jaehun Shin, Won-Kee Lee, Baikjin Jung, and Jong-Hyeok Lee. 2021. Transformer-based screenplay summarization using augmented learning representation with dialogue information. In *Proceedings of the Third Workshop on Narrative Understanding*, pages 56–61, Virtual. Association for Computational Linguistics.

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

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Ziming Mao, Chen Henry Wu, Ansong Ni, Yusen Zhang, Rui Zhang, Tao Yu, Budhaditya Deb, Chenguang Zhu, Ahmed Awadallah, and Dragomir Radev. 2022. DYLE: Dynamic latent extraction for abstractive long-input summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1687–1698, Dublin, Ireland. Association for Computational Linguistics.

Gianluca Moro and Luca Ragazzi. 2022. Semantic self-segmentation for abstractive summarization of long legal documents in low-resource regimes. In *Proceedings of the Thirty-Six AAAI Conference on Artificial Intelligence, Virtual*, volume 22.

Pinelopi Papalampidi, Frank Keller, and Mirella Lapata. 2019. Movie plot analysis via turning point identification. 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 1707–1717, Hong Kong, China. Association for Computational Linguistics.

Pinelopi Papalampidi, Frank Keller, and Mirella Lapata. 2021. Movie summarization via sparse graph construction. In *AAAI*.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020a. Few-shot natural language generation for task-oriented dialog. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 172–182, Online. Association for Computational Linguistics.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. 2020b. Random feature attention. In *International Conference on Learning Representations*.

Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Chris Pal. 2020. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9308–9319, Online. Association for Computational Linguistics.

Jing Qian, Li Dong, Yelong Shen, Furu Wei, and Weizhu Chen. 2022. Controllable natural language generation with contrastive prefixes. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2912–2924, Dublin, Ireland. Association for Computational Linguistics.

Jiezhong Qiu, Hao Ma, Omer Levy, Wen-tau Yih, Sinong Wang, and Jie Tang. 2020. Blockwise self-attention for long document understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2555–2565, Online. Association for Computational Linguistics.

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

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR.

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, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2022. NoisyTune: A little noise can help you finetune pretrained language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 680–685, Dublin, Ireland. Association for Computational Linguistics.

Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. 2022. PRIMERA: Pyramid-based masked sentence pre-training for multi-document summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5245–5263, Dublin, Ireland. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297.

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

Yusen Zhang, Ansong Ni, Ziming Mao, Chen Henry Wu, Chenguang Zhu, Budhaditya Deb, Ahmed Awadallah, Dragomir Radev, and Rui Zhang. 2022. Summ^n: A multi-stage summarization framework for long input dialogues and documents. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1592–1604, Dublin, Ireland. Association for Computational Linguistics.

A Appendix: Benchmark Comparison

Table 5 compares current state-of-the-art Transformer-based models to find the optimal model architecture. For BART, we extend input length up to 5120 until total memory footprints can be loaded into one Nvidia Tesla V100 with

Architecure	# Tok.	T	R-2-F1
BART	5K	1	6.30
T5	8K	16	8.60
LED	8K	2	**8.70**

Table 5: Comparison of model architectures. BART, T5 and LED are trained for 10 epochs. $\#Tok.$: the number of tokens as the maximum input length to the encoder. T: the training time in hours. All models are in `base` size and evaluated on validation set and use decoding length 1024.

32 GB. Because BART is not designed to handle such long-form sequences, we extend the size of position embeddings by initializing from the pre-trained position embeddings in BART. For T5 and LED, we use the input length ($\#Tok.$) 8192 to encode whole movie scripts. Lastly, we fine-tune all models for 10 epochs and report the ROUGE-2 F1 score (R-2-F1) on the validation set. Based on the finding of the ablation study on beam size (Sec. 3.6), all experiments use $BS = 4$.

Among the current SOTA models: BART, T5 and LED, we obtain 6.3 and 8.6 and 8.7 on R-2 F1 scores respectively. For the performance, LED outperforms BART and T5 by 2.4 and 0.1 points. The demand of high complexity attention computation in BART limits the maximum input length and thus fails to generalise well on the long-form movie summarization. On the other hand, employing to global sliding window attention, LED does not need to shrink its input context length and greatly benefits from 16-bit mixed precision training by which take only 2 hours. In contrast, fine-tuning T5 requires 32-bit training which results in $8\times$ slower than LED at training phase. Based on the performance and training efficiency advantages, we leverage LED for the development of our long-form summarization system on CreativeSumm'22.

B Appendix: Runtime Performance

To further understand the effects of different encoding and decoding lengths on model performance to select models and its inference efficiency, we design the controlled trails to test the impact of encoding and decoding lengths. We use our performant UdS NoisyTune and evaluate model on validation set using $BS = 4$.

Impact of Encoding Lengths Figure 3 shows the dependency between performance (blue) and runtime (red) on various encoding length. It is

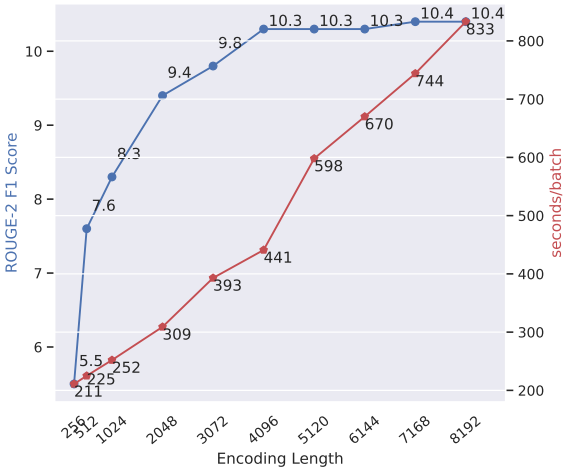

Figure 3: Performance (blue) and decoding runtime (red) when varying in encoding lengths. We report the ROUGE-2 F1 score and inference time on various input lengths from 256 up to 8192.

worth noting that the model is to be performant only when it encodes a long enough movie script, such as $8K$. More importantly, we reduced input length to 4094 with only 0.1 performance drops.

However, the runtime varying in different encoding lengths and scales linearly as encoding length increases. These results suggest that the current Transformer-based encoder-decoder model with quadratic-form attention would be even worse at inference.

Impact of Decoding Lengths Unlike the encoding length, the decoding length plays a more critical role in the inference time. The performance gradually drops caused by shorter decoding length which is making sense as the length of gold summary is around $1K$.

Notably, runtime varying in decoding length follows a quadratic trend. By shrinking decoding length to 768, we find that the summarizer obtains $1.6\times$ speedups at inference time while keeping 96% performance. In addition, by reducing decoding length to 512, our model achieves $3.3\times$ speedups and keeps 89% performance on the R-2 F1 score. Our result indicates that reducing decoding length to 75% or 50% of the original decoding length significantly improves its inference efficiency without much performance drops.

Impact of Beam Size To understand the dependency between model performance and runtime when varying beam sizes (BS), Figure 5 illustrates

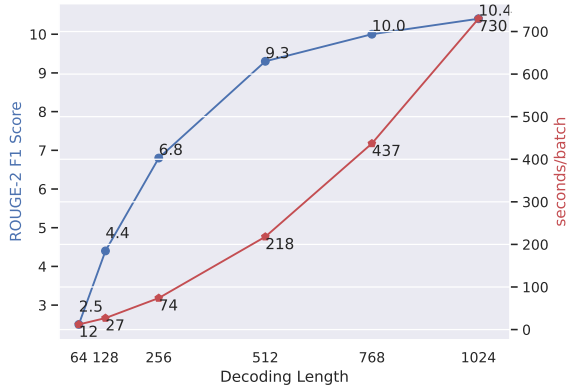

Figure 4: Performance (blue) and decoding runtime (red) when varying in decoding lengths. We report the ROUGE-2 F1 score and inference time on various output lengths from 64 up to 1024.

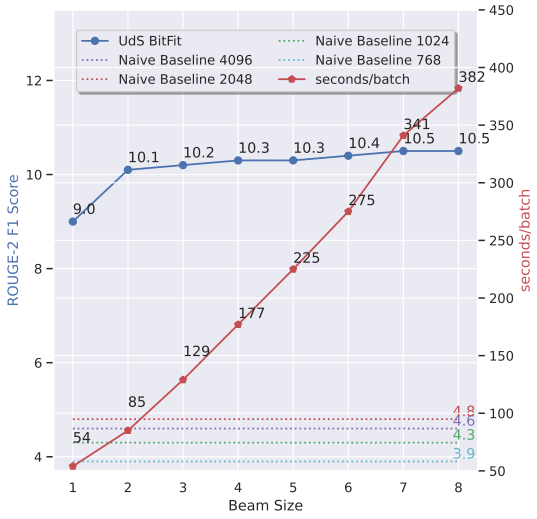

Figure 5: Performance (blue) and runtime (red) of UdS BitFit when varying in beam sizes. We evaluate UdS BitFit on various beam size (BS) from 1 to 8. All models are evaluated on validation set.

the performance between our system (blue) , *naive* baselines (dotted lines) and inference speed (red). As choosing $BS = 2$, UdS BitFit gains 3.4 points improvement compared to the model using greedy search ($BS = 1$) on ROUGE-2 F1 score. In addition, our UdS BitFit using large BS from 4 to 8 achieve the best performance 14.9 and significantly outperforms the *naive* baseline 2048 by 10.1 points on the ROUGE-2 F1 score. The result suggests $BS = 4$ is sufficient to obtain the most performant result.

However, the runtime scales linearly with the BS increasing. For instance, UdS BitFit stops showing the improvement in performance after $BS = 4$ but takes more computational cost. Using large BS from 5 to 8 requires around 5 to 8 minutes for merely one mini-batch (size=1), which shows the decoding with large BS is extremely expensive.

C Appendix: Background

C.1 Efficient Transformers

Transformer-based models (Vaswani et al., 2017) are widely applied for text generation problems but the $O(n^2)$ complexity of the self-attention makes long document text generation computationally expensive and prohibitive. Various strategies have been proposed to address this issue: **1.** Complexity can be reduced by restricting the global attention to local patterns. (Correia et al., 2019) learn shorter attention patterns for different heads and different layers, (Child et al., 2019; Beltagy et al., 2020; Guo et al., 2022) use random, stride or fixed local attention patterns, (Tay et al., 2020) use learnable attention patterns improve the memory efficiency of the attention module. **2.** (Ainslie et al., 2020; Zaheer et al., 2020) use memory/downsampling methods. **3.** Complexity can also be reduced by approximating self-attention using low-rank decomposition (Wang et al., 2020) or kernels (Peng et al., 2020b). **4.** The context of transformers can also be encoded into a fixed sized hidden state (Dai et al., 2019).