DYLE: Dynamic Latent Extraction for Abstractive Long-Input Summarization

Ziming Mao[*1] **Chen Henry Wu**[*2] **Ansong Ni**[1] **Yusen Zhang**[3]
Rui Zhang[3] **Tao Yu**[4] **Budhaditya Deb**[5]
Chenguang Zhu[5] **Ahmed H. Awadallah**[5] **Dragomir Radev**[1]
[1] Yale University [2] Carnegie Mellon University [3] Penn State University
[4] The University of Hong Kong [5] Microsoft Research
ziming.mao@yale.edu, henrychenwu@cmu.edu

Abstract

Transformer-based models have achieved state-of-the-art performance on short-input summarization. However, they still struggle with summarizing longer text. In this paper, we present DYLE, a novel dynamic latent extraction approach for abstractive long-input summarization. DYLE jointly trains an extractor and a generator and treats the extracted text snippets as the latent variable, allowing dynamic snippet-level attention weights during decoding. To provide adequate supervision, we propose simple yet effective heuristics for oracle extraction as well as a consistency loss term, which encourages the extractor to approximate the averaged dynamic weights predicted by the generator. We evaluate our method on different long-document and long-dialogue summarization tasks: GovReport, QMSum, and arXiv. Experiment results show that DYLE outperforms all existing methods on GovReport and QMSum, with gains up to 6.1 ROUGE, while yielding strong results on arXiv. Further analysis shows that the proposed dynamic weights provide interpretability of our generation process.[1]

1 Introduction

Transformer-based (Vaswani et al., 2017) pre-trained language models (PLMs) such as BART (Lewis et al., 2020a) and T5 (Raffel et al., 2020), have achieved state-of-the-art performance on short text summarization. However, due to the high memory complexity of the full self-attention (Tay et al., 2020a), PLMs still struggle to handle long inputs (Rohde et al., 2021). *Model efficiency* and *summary quality* present a pair of challenges (Huang et al., 2021): models need to capture information scattered across the long input while maintaining a low computational cost.

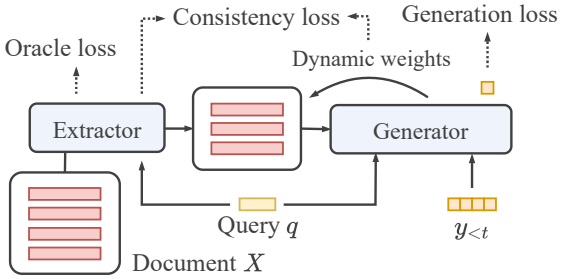


Figure 1: Overview of our approach. The input is a document X (each $x \in X$ is a sentence) and an optional query q, and the output is a summary y.

Prior models tackled long input summarization mostly in four ways. First, *sparse attention* (Child et al., 2019; Beltagy et al., 2020; Tay et al., 2020b) is used to reduce the memory complexity of the Transformers so that they can attend to more tokens. Second, *extract-then-generate* methods extract salient texts from the input and then summarize the extracted texts. Extractors are either independently trained with full supervision (Zhong et al., 2021b) or optimized using reinforcement learning (Williams, 1992; Chen and Bansal, 2018; Bae et al., 2019; Bražinskas et al., 2021). Third, models are proposed to divide source text into sections (Gidiotis and Tsoumakas, 2020; Wu et al., 2021; Liu et al., 2021) which are individually summarized and combined to form a full summary. Fourth, hierarchical models (Rohde et al., 2021; Zhu et al., 2020) improve summarization by capturing sentence or discourse level dependencies. We elaborate on these four directions and their limitations in Section 2.

We believe that the extract-then-generate approach mimics how a person would handle long-input summarization: first identify important pieces of information in the text and then summarize them (Kiyoumarsi, 2015; Sun et al., 2020). The extract-then-generate framework is based on the assumption that salient information useful for summarization only occupies a small portion of the input,

[*] Equal Contributions.
[1] Our code is available at: https://github.com/Yale-LILY/DYLE

which is a sensible assumption given the long input length. This approach shortens the source input to a pre-set length, which addresses the main challenge of the model not being able to handle longer input beyond a certain limit. However, previous separately-trained extract-then-generate approaches are limited as they suffer from cascaded errors from the extractor to the generator. Though various reinforcement learning techniques are introduced to bridge the two steps, they have noticeable drawbacks (discussed in Section 3.3), and we argue that the long input makes this approach suboptimal.

In this paper, we propose a new approach for long-input summarization: *Dynamic Latent Extraction for Abstractive Summarization* (DYLE). DYLE *jointly trains* the extractor and the generator and keeps the extracted text snippets latent. For an output token, DYLE compute its probability conditioned on each input snippet *separately*, and its generation probability is computed by *marginalizing* over all the input snippets under a learned *dynamic weights* assigned by the generator conditioned on the previously generated tokens.

We optimize the extractor with two surrogate 代理 losses. First, we compute the *extractive oracle* based on the reference summary with a greedy search over the best ROUGE scores. These oracle snippets are used as targets for the extractor learning signal. Moreover, we propose *consistency loss* to encourage the extractor to approximate its own predicted weights on the snippet to the averaged dynamic weights predicted by the generator.

We conducted experiments on three long-input summarization datasets: GovReport (Huang et al., 2021) and arXiv (Cohan et al., 2018) for long-document summarization, and QMSum (Zhong et al., 2021b) for long-dialogue summarization. Our method DYLE largely outperforms existing methods on GovReport and QMSum, while achieving strong results on arXiv. Notably, DYLE yields gains of 4.2/6.1/4.0 of ROUGE-1/2/L points over the previous best method on GovReport. These experiments demonstrate the generalizability of DYLE to multiple long-input summarization tasks.

We summarize our contributions as follows:

- We introduce DYLE, a dynamic latent extraction approach for abstractive long-input summarization. DYLE better captures information in the long input and reduces computational cost;
- We propose multiple auxiliary optimizations for the effective training of DYLE: 1) extractive oracle as a learning signal for the extractor; 2) consistency loss that bridges extraction and generation; 3) hybrid training methods that make the extraction more robust;
- Experimental results show that DYLE largely outperforms the state-of-the-art on two long input summarization datasets. We also conducted a detailed analysis that shows dynamic weights improve model interpretability.

2 Related Work

We introduce in detail the four main categories of methods in recent work to address long-input summarization tasks.

Sparse attention mechanism The full attention mechanism has a quadratic memory cost. Prior research works have proposed different sparse attention mechanisms to reduce the memory cost. Longformer (Beltagy et al., 2020) uses a dilated sliding window of blocks and global attention patterns. BigBird (Zaheer et al., 2020) employs sliding windows and random blocks. Reformer (Kitaev et al., 2020) uses the locality-sensitive hashing. In addition to optimizing the encoder self-attention, Huang et al. (2021) proposes head-wise positional strides to reduce the cost of the encoder-decoder attention. However, sparse attention diminishes the benefits of pretraining and sacrifices parts of the receptive field.

Extract-then-generate method This method extracts salient text snippets from the input, followed by generating an overall summary. Most of these approaches are trained separately (Zhang et al., 2019; Lebanoff et al., 2019; Xu and Durrett, 2019; Bajaj et al., 2021; Zhang et al., 2021b), which suffer from information loss as we pass the extracted snippets to the generator. Some approaches attempt to reduce that loss by bridging the two stages. Chen and Bansal (2018) adopts reinforcement learning (RL) with a sentence-level policy gradient. Bae et al. (2019) proposes summary-level policy gradient. Using RL suffers from various drawbacks on long input texts, which will be elaborated in Section 3.3. DYLE is different as we jointly train an extract-then-generate model for summarization using latent variables.

Divide-and-conquer approach A common approach in long input summarization is divide-and-

conquer (Gidiotis and Tsoumakas, 2020; Grail et al., 2021; Zhang et al., 2021a). It breaks a long input into multiple parts, which are summarized separately and combined to produce a final summary. However, these models do not capture the contextual dependencies across parts and assume that the input has certain structure.

Hierarchical models Various hierarchical models have been proposed to handle the longer inputs. Cohan et al. (2018) models the document discourse structure with a hierarchical encoder and a discourse-aware decoder. HAT-Bart (Rohde et al., 2021) proposes a new Hierarchical Attention Transformer-based architecture that attempts to capture sentence and paragraph-level information. HMNet (Zhu et al., 2020) builds a hierarchical structure that includes discourse-level information and speaker roles. However, these models focus mainly on model performance and not on reducing the memory and computational cost.

3 Our Approach

An overview of our approach is shown in Figure 1. In Section 3.1, we formulate our task and the extractor-generator framework. In Section 3.2, we introduce our parameterization of the extractor for long inputs. In Section 3.3, we introduce generator formulation and the novel consistency loss. The extractor module is both optimized with the consistency loss and the oracle loss, which we elaborate on in Section 3.4. The overall training objective is summarized in Section 3.5.

3.1 Extractor-Generator Framework

In the long-input summarization task, the input consists of L text snippets, $X = (x_1, \ldots, x_L)$, and an optional query q if a query is paired with a summary. In long-input summarization, the number of text snippets, L, could be potentially large. The output is a summary y of length T. For the dialogue summarization task, dialogue utterances by each speaker are used as snippets. For documents, we tokenize the input into sentences and use each sentence as a snippet. The goal is to learn a model that generates a sequence of summary tokens y given the input snippets X and the previously generated tokens $y_{<t}$:

$$P_\theta(y|q, X) = \prod_{t=1}^{T} P_\theta(y_t|q, X, y_{<t})$$

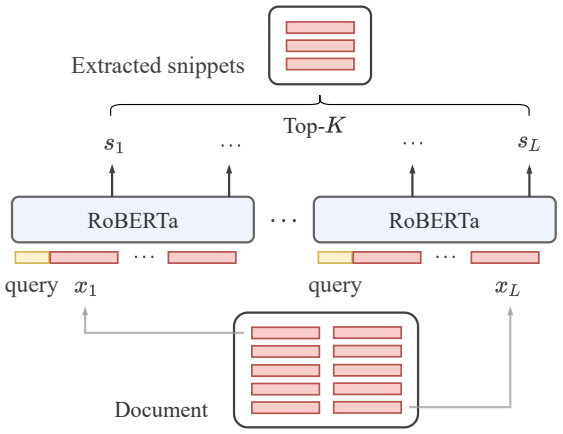


Figure 2: Long-input extractor. We divide the document into chunks, each containing consecutive snippets. A shared RoBERTa encodes each chunk independently.

The *extractor* takes the query and the source text as input and outputs a score $s_i = E_\eta(q, x_i)$ for each text snippet x_i. Here η is the extractor parameters. We extract K snippets X_K from the document X based on their scores:

$$X_K = \text{top-}K(E_\eta(q, x_i), x_i \in X) \quad (1)$$

After retrieving X_K from X, the extractor-generator framework models the output probability by replacing X with X_K, i.e.,

$$\begin{aligned} P_\theta(y|q, X) &= P_\theta(y|q, X_K) \\ &= \prod_{t=1}^{T} P_\theta(y_t|q, X_K, y_{<t}) \end{aligned} \quad (2)$$

Note that the top-K operation in Eq. (1) is non-differentiable, and we do not propagate gradients through top-K; instead, we propose methods to optimize the extractor in Section 3.3 and Section 3.4.

3.2 Extractor for Long Inputs

An interesting research question is how to design the extractor for long inputs. Limited by GPU memory, it is impractical to concatenate all snippets and encode them with a large pre-trained language model. As shown in Figure 2, we group consecutive snippets into *chunks*. We concatenate the query q with each chunk and compute the encoded vector for each snippet independently within the chunk it belongs to. We project the encoded vectors to scalar scores $s_i = E_\eta(q, x_i)$ using an MLP.

3.3 Generator with Dynamic Weights

Challenges An extract-then-generate model faces two challenges in long-input summarization.

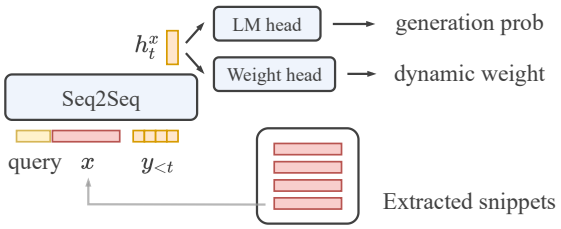


Figure 3: At each decoding time step, our generator predicts the dynamic weight and the generation probability for each extracted snippet.

The first challenge is that the extraction operation (top-K in Eq. (1)) is non-differentiable. One approach is to adopt RL-based optimizations (Chen and Bansal, 2018; Bae et al., 2019), which has two drawbacks. First, reinforcement learning for large action spaces (i.e., extracting K out of L snippets when L is very large) has high variances. Second, current methods mostly use sentence-level ROUGE (Chen and Bansal, 2018) or summary-level ROUGE (Bae et al., 2019) as training rewards. Using sentence-level ROUGE could potentially select sentences with overlapping contents (Narayan et al., 2018), resulting in redundant final summaries. Using a summary-level ROUGE leads to the sparsity of the training signal, and longer input makes this approach harder to train. The second challenge is interpretability: one might want to know whether the generator is leveraging the extracted information at each decoding time step.

To address these challenges, we propose a generator that dynamically assigns weights to every extracted snippet at each time step. Different from the extractor scores, which are independent of the decoding time step, the generator assigns different dynamic scores at different time steps. Dynamic weights make the decoding process interpretable and help denoise the extraction by down-weighting irrelevant snippets. It also provides training signals for the extractor using *consistency loss*.

Generator formulation The overview of the generator is shown in Figure 3. For each extracted snippet x, the generator predicts the generation probability $P_\theta(y_t|q, x, y_{<t})$ on this snippet and a dynamic weight $P_\theta(x|q, X_K, y_{<t})$ for this snippet. The independent encoding of each extracted snippet saves memory because the snippets do not need to attend to each other. Without loss of generality, we assume that $P_\theta(\cdot|q, x, y_{<t})$ is computed by first mapping the input $(q, x, y_{<t})$ to a contextualized representation vector $\boldsymbol{h}_t^x$. For Transformers (Vaswani et al., 2017) and encoder-decoder with attention models (Bahdanau et al., 2015), $\boldsymbol{h}_t^x$ is usually the model's output before the final language model head. The generation probability $P_\theta(y_t|q, x, y_{<t})$ is computed by feeding $\boldsymbol{h}_t^x$ into the language model head. For the dynamic weight $P_\theta(x|q, X_K, y_{<t})$, we adopt a separate MLP to map each $\boldsymbol{h}_t^x$ to a scalar logit l^x, and $P_\theta(\cdot|q, X, y_{<t})$ is defined as $\text{softmax}(\{l^x\}_{x\in X})$. We compute the generation probability by marginalizing over all extracted snippets:

$$P_\theta(y|q, X_K) = \prod_{t=1}^{T} \sum_{x\in X_K} P_\theta(y_t|q, x, y_{<t}) P_\theta(x|q, X_K, y_{<t}) \quad (3)$$

The dynamic weight $P_\theta(x|q, X_K, y_{<t})$ at each decoding time step t allows us to interpret how the generator utilizes the extracted snippets. For example, a larger weight to a particular snippet indicates the larger importance of the snippet to the current decoding time step. The generation loss is defined as the NLL of the gold summary:

$$\mathcal{L}_{gen}^{\theta} = -\log P_\theta(y|q, X_K) \quad (4)$$

where $P_\theta(y|q, X_K)$ is defined in Eq. (2). Here we do not propagate gradients of $\mathcal{L}_{gen}^{\theta}$ to the extractor parameters since top-K is non-differentiable. Instead, methods to optimize the extractor are described in Section 3.3 and Section 3.4.

Consistency loss We also leverage the dynamic weights to provide a training signal for the extractor. Since the dynamic weight of a snippet can be interpreted as the importance of the snippet at a particular time step, we average the dynamic weights over all the decoding steps and view the averaged weight as the overall importance of the snippet. Based on this intuition, we propose what we term as *consistency loss*, which measures the distance between the averaged dynamic weights distribution and the extractor distribution. We want these two distributions to be close on an *arbitrary* subset of X. For simplicity, we take X_K as the subset and define the consistency loss as

$$\mathcal{L}_{consist}^{\eta} = \text{KL}\Big[\frac{1}{T}\sum_{t=1}^{T} P_\theta(\cdot|q, X_K, y_{<t}) \;||\; \text{softmax}(E_\eta(q, x_i), x_i \in X_K)\Big] \quad (5)$$

Note that the consistency loss is superscripted with the extractor's parameters η, which means that we do not compute gradients for the generator's parameters θ. Since we want the distributional distance to be small on an *arbitrary* subset of X, we do not propagate gradients through the top-K operator.

3.4 Leveraging Extractive Oracles

For long-input summarization, the extracted snippets X_K used during training are important for stable optimization. Instead of using X_K defined in Eq. (1), we propose to leverage *extractive oracles* during training. No extractive oracles are used during test time.

Greedy search for extractive oracles *Extractive oracles* denote a set of selected text snippets whose concatenation maximizes the evaluation metric given the gold summary. We implement the extractive oracle using greedy search. Specifically, we start with an empty set, and we iteratively select a snippet from the input such that the concatenation of that snippet and the already selected snippets maximizes the average of ROUGE-1, ROUGE-2, and ROUGE-L scores given the gold summary. We denote the extractive oracles as X_o.

Hybrid training We leverage the extractive oracles to define X_K used during training. If the number of oracles equals or exceeds K, we define X_K as the first K oracle snippets. If the number of oracles is less than K, we define X_K as the union of X_o and the top snippets ranked by the extractor that is not appearing in X_o. Such hybrid training has two benefits. First, compared with X_K defined in Eq. (1), it provides higher-quality inputs to the generator. Second, it reduces the reliance on the oracle and improves the generalizability of our model beyond the training set, as other text snippets omitted in the greedy search might help the generation.

Oracle loss The extractive oracles X_o are used as a supervision signal for the extraction part of our model. The oracle loss $\mathcal{L}^{\eta}_{oracle}$ is computed from the cross-entropy loss between all chunks in the extractor selected set and the extractive oracle. Formally, the oracle loss is computed as

$$\mathcal{L}^{\eta}_{oracle} = -\frac{1}{|X_o|}\sum_{x\in X_o}\log\frac{e^{E_\eta(q,x)}}{\sum_{x_i\in X}e^{E_\eta(q,x_i)}} \quad (6)$$

Dataset	Query	Format	Src. leng.	Tgt. leng.
GovReport	✗	Doc.	9,409	553
arXiv	✗	Doc.	6,030	273
QMSum	✓	Dial.	9,070	69

Table 1: Comparison of evaluation benchmarks.

3.5 Training Objective

The overall training objective of our method is

$$\mathcal{L}^{\theta,\eta} = \lambda_g\mathcal{L}^{\theta}_{gen} + \lambda_o\mathcal{L}^{\eta}_{oracle} + \lambda_c\mathcal{L}^{\eta}_{consist} \quad (7)$$

where λ_g, λ_o, and λ_c are hyperparameters to balance the loss components. Gradients are computed for the superscripted parameters. Specifically, the extractor is solely optimized with the consistency loss and the oracle loss, and the generator is solely optimized with the generation loss.

4 Experiment Setups

4.1 Datasets

We consider the following long-input abstractive summarization datasets as evaluation benchmarks:[2]

QMSum (Zhong et al., 2021b) is a benchmark for query-based multi-domain meeting summarization. It consists of meetings from three domains: AMI (Carletta et al., 2005), ICSI (Janin et al., 2003), and committee meetings of the Welsh Parliament and Parliament of Canada;

GovReport (Huang et al., 2021) is a large-scale long document summarization dataset, consisting of about 19.5k U.S. government reports with expert-written abstractive summaries; GovReport is a good benchmark as it contains significantly longer documents (average 9.4k words) and summaries (553 words) than other long document datasets, such as ArXiv, PubMed (Cohan et al., 2018), BillSum (Kornilova and Eidelman, 2019), and BigPatent (Sharma et al., 2019);

arXiv (Cohan et al., 2018) is a dataset of scientific articles from arXiv. Abstracts of the articles are used as the target summary. ArXiv is chosen over PubMed (Cohan et al., 2018) as arXiv contains longer articles compared to PubMed.

A detailed comparison of the datasets used can be found in Table 1.

[2] QMSum and arXiv can be accessed through SummerTime (Ni et al., 2021a).

	R-1	R-2	R-L
BART(1024)	52.83	20.50	50.14
BART w/ sparse attn.			
Stride (4096)	54.29	20.80	51.35
LIN. (3072)	44.84	13.87	41.94
LSH (4096)	54.75	21.36	51.27
Sinkhorn (5120)	55.45	21.45	52.48
BART w/ sparse attn. + HEPOS			
LSH (7168)	55.00	21.13	51.67
Sinkhorn (10240)	56.86	22.62	53.82
DYLE (dynamic)	**61.01**	**28.83**	**57.82**

Table 2: Results on GovReport, where R stands for the ROUGE metric and the number in the brackets denotes maximum input sequence length of the model.

	R-1	R-2	R-L
Locator as extractor			
PGNet (2048)	28.74	5.98	25.13
Bart-large (3072)	32.16	8.01	27.72
HMNet (8192)	32.29	8.67	28.17
Longformer (8192)	31.60	7.80	20.50
UNILM-base (5120)	29.14	6.25	25.46
UNILM-CP (5120)	29.19	6.73	25.52
UniLM with DialogLM pretraining			
DialogLM (5120)	34.02	9.19	29.77
DialogLM - Sparse (8192)	33.69	9.32	30.01
DYLE (dynamic)	**34.42**	**9.71**	**30.10**

Table 3: Results on QMSum. The baseline performance numbers are from Zhong et al. (2021a).

4.2 Baselines and Implementation

Baselines for Comparisons We compare DYLE with the previous state-of-the-art methods on the aforementioned three datasets. More specifically: 1) For GovReport, we report the performance from the original paper, which uses various encoder self-attention and the proposed HEPOS encoder-decoder attention; 2) For QMSum, we compare with Zhong et al. (2021a), the current SoTA and other baselines mentioned in that work; 3) For arXiv, we include the results from the best performing models in previous works, including ExtSum-LG (Xiao and Carenini, 2019), PEGASUS (Zhang et al., 2020), DANCER (Gidiotis and Tsoumakas, 2020), BigBird (Zaheer et al., 2020), HEPOS + LSH (Huang et al., 2021), HAT-BART (Rohde et al., 2021), Longformer (Beltagy et al., 2020), and SSN-DM (Cui and Hu, 2021). Note that those baselines spans over different strategies to handle long input, such as sparse-attention (HEPOS, BigBird, Longformer), hierarchical attention (HAT-BART), extract-then-generate (Locator + different generators).

	R-1	R-2	R-L
Prior Work			
ExtSum-LG (dynamic)	44.01	17.79	39.09
PEGASUS (3072)	44.21	16.95	38.83
DANCER-PEGASUS (dynamic)	45.01	17.60	40.56
BigBird-PEGASUS (3072)	46.63	19.02	41.77
LSH (7168)	**48.24**	**20.26**	41.78
HAT-BART (3072)	46.68	19.07	**42.17**
LED-large (16384)	46.63	19.62	41.83
SSN-DM (dynamic)	45.03	19.03	32.58
DYLE (dynamic)	46.41	17.95	41.54

Table 4: Results on arXiv.

	R-1	R-2	R-L
GovReport			
Full	**61.01**	**28.83**	**57.82**
w/o hybrid	60.89	28.28	57.31
w/o consistency	60.59	28.48	57.49
w/o oracle	57.57	25.92	53.14
QMSum			
Full	**34.42**	**9.71**	**30.10**
w/o hybrid	31.77	8.33	28.37
w/o consistency	32.51	8.77	28.94
w/o oracle	32.13	8.38	28.63

Table 5: Ablation study for auxiliary optimizations.

4.3 Implementation Details

Pretrained-LM The extractor is initialized with `RoBERTa-base` (Liu et al., 2019) weights. The generator is initialized with `BART-large` (Lewis et al., 2020a) weights. We use the Adam optimizer and set the extractor learning rate to 5e-5 and the generator learning rate to 5e-6.

Hyperparameters λ_g, λ_o, and λ_c are the coefficients for the generation loss, oracle loss, and the consistency loss respectively. For λ_g and λ_o, we did a 2-step binary search between 0 and 2. For λ_c, we did a 3-step binary search between 0 and 10. For the QMSum dataset, we used $\lambda_g = 1$, $\lambda_o = 1$, $\lambda_c = 1$. For the GovReport dataset, we used $\lambda_g = 0.5$, $\lambda_o = 1$, $\lambda_c = 1$. For the ArXiv dataset, we used $\lambda_g = 0.5$, $\lambda_o = 1$, $\lambda_c = 5$.

Hardware We apply gradient checkpointing (Chen et al., 2016) to save the GPU memory. Each experiment is run on one NVIDIA Quadro RTX 8000 GPU. The effective batch size is set to 8.

5 Experiment Results

5.1 Main Results

The evaluation results are summarized in Table 2, Table 3, and Table 4. For GovReport, DYLE yields

		ROUGE-1			ROUGE-2			ROUGE-L		
		P	R	F1	P	R	F1	P	R	F1
GovReport	Extracted snippets	48.98	**73.40**	57.56	24.20	**36.59**	28.53	46.28	**69.25**	54.35
	Generated summaries	**63.16**	61.61	**61.01**	**29.85**	29.10	**28.83**	**59.88**	58.35	**57.82**
QMSum	Extracted snippets	4.25	**76.90**	7.74	1.36	**28.41**	2.49	3.99	**72.83**	7.26
	Generated summaries	**29.78**	45.64	**34.42**	**8.39**	13.06	**9.71**	**26.14**	39.70	**30.10**

Table 6: Precision-recall decomposition of ROUGE scores of extracted snippets and generated summaries.

gains of 4.15/6.21/4.00 of ROUGE-1/2/L scores compared to the previous best method. Experiments on GovReport show that DYLE is performant over prior sparse attention approaches.

On QMSum, DYLE yields the new state-of-the-art ROUGE-1/2/L scores of 34.42/9.71/30.10, outperforms UniLM with DialogLM pretraining. Comparing DYLE with locator-based models on the QMSum dataset shows that DYLE outperforms prior extract-then-generate approaches where the locator is independently trained with intermediate annotated text spans. This shows the effectiveness of DYLE's joint training approach. These results show that DYLE can be applied to both the long document summarization and long dialogue summarization tasks. DYLE's better performance can be attributed to lowered information loss between the extraction and the generation steps and its ability to handle input of a much longer length.

We notice that while DYLE largely outperforms the LSH baseline (Huang et al., 2021) on the GovReport dataset, it underperforms the LSH baseline on arXiv. We posit two reasons. First, the input of the GovReport is much longer than that of arXiv. Most, if not all, of the sentences in the arXiv input article can be processed by the LSH model. Second, the summaries of the arXiv dataset are more abstractive than those of GovReport. It is possible that individually extracted text snippet is not the best linguistic unit for generating output tokens. It is our future work to explore the optimal input unit for an extract-then-generate approach. Nevertheless, DYLE outperforms other extraction-based approaches (*e.g.,* SSN-DM (Cui and Hu, 2021)) and divide-and-conquer approaches (*e.g.,* DANCER (Gidiotis and Tsoumakas, 2020)).

5.2 Evaluation of Auxiliary Optimizations

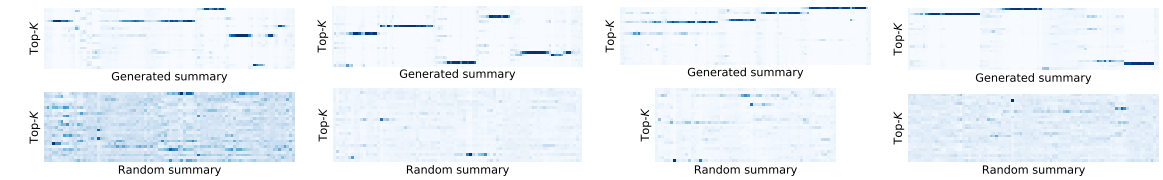
We conduct ablation studies to investigate the effectiveness of the auxiliary optimizations we introduced. Specifically, we report the full model's performance after removing 1) hybrid training, 2) consistency loss, 3) extractive oracle loss. In our default model, the consistency loss is computed on the combination of the extracted snippets and oracle snippets; in the "w/o hybrid" experiment, the consistency loss is only computed on the set of oracle snippets; in "w/o consistency" experiment, the consistency loss is not computed. The results are summarized in Table 5. Note that without the hybrid training optimization, only the extractive oracles will be used to train the generator. When the consistency loss is not calculated, the extractor and the generator can be viewed as being trained independently with the extractive oracles.

We see that excluding either of the hybrid training, consistency loss, or oracle loss optimization leads to a performance drop. Training the model without the supervision of the oracle leads to the greatest decrease in model performance, showing the importance of good supervision for the extractor. Removing the consistency loss also decreases the model performance. This shows that the consistency loss allows the extractor to better learn to select salient snippets from the input text and enables DYLE to generalize better to the test set.

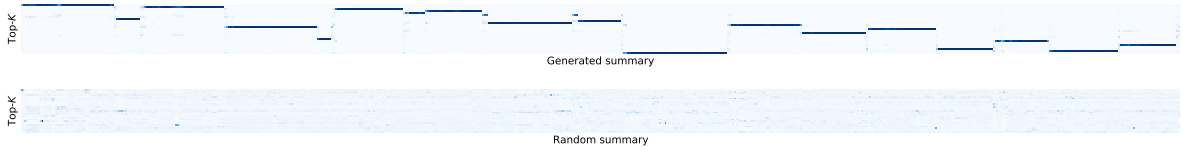
6 Analysis and Discussion

Analysis of extracted snippets We are interested in the amount of salient information passed to the generator. To investigate this, we report the decomposed precision and recall of ROUGE scores in Table 6. We observe that the extracted snippets have much higher recall than the generated summaries, while the generated summaries have higher precision. This suggests that to improve the overall performance, we can increase the information coverage (i.e., recall) of the extractor and improve the accuracy of the generator in identifying the salient snippets (i.e., precision).

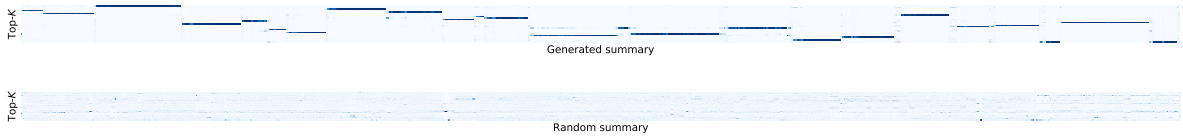
Interpretability of dynamic weights Our approach is more interpretable than sparse attention and two-step extraction-generation pipeline meth-



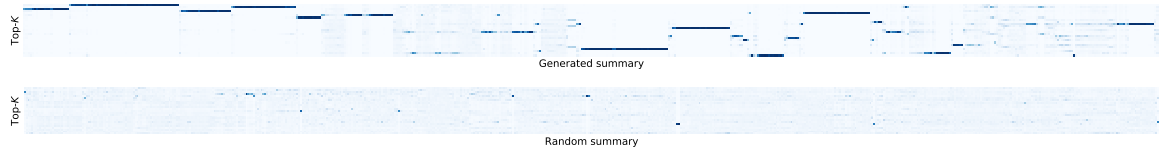
(a) QMSum sample 1 (b) QMSum sample 2 (c) QMSum sample 3 (d) QMSum sample 4



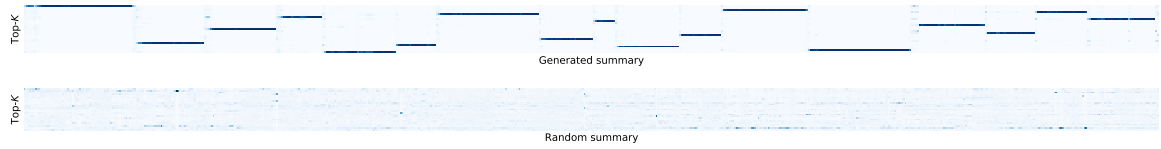
(e) GovReport sample 1



(f) GovReport sample 2



(g) GovReport sample 3



(h) GovReport sample 4

Figure 4: Dynamic weight visualization. We visualized the dynamic weight matrices of the *generated summary* and a *random summary* from other samples in the validation set. x-axis: decoding time step; y-axis: index of the extracted top-K snippets. Darker squares stand for higher weights. More examples can be found in Appendix A.

ods. Specifically, *dynamic weights* in the generator shows how the information is used throughout the decoding process. In Figure 4, we visualize the dynamic weights for the extracted snippets assigned by the generator during decoding. In each subfigure, we visualize the dynamic weight matrices of the *generated summary* and a *random summary* from other samples in the validation set. The x-axis and y-axis represent the decoding time step and the index of the extracted top-K snippets, respectively. Darker squares denote higher weights. For each generated summary, we observe multiple consecutive high-weight areas, indicating alignments between the extracted snippets and the generated summary. In contrast, weights are uniformly distributed for random summaries. Interestingly, we observe that, on QMSum, fewer sentences are considered when generating the summaries. Our explanation for this observation is that QMSum is a query-based dataset, where the queried information is more concentrated in a few snippets. By contrast, we find that a larger number of snippets are used on the GovReport dataset as seen in Figure 4, as GovReport is a general summarization dataset.

Effect of number of extracted snippets To evaluate the effect of number of extracted snippets on model performance, we vary the value of K of top-K in Eq. (1) and test it on both the GovReport and QMSum datasets. We observe that the model performance generally increases as the value of K increases. This is expected as more extracted snippets provide the generator with more information to form a final summary. The results are summa-

	R-1	R-2	R-L
GovReport			
K=25	**61.01**	**28.83**	**57.82**
K=20	59.25	27.46	55.74
K=15	58.55	26.95	54.89
K=10	54.98	24.10	51.25
QMSum			
K=25	**34.42**	**9.71**	**30.10**
K=20	33.10	8.69	29.62
K=15	31.78	8.36	28.31
K=10	33.30	9.18	29.53

Table 7: Comparing model performance with different values of K on the GovReport and QMSum dataset

	R-1	R-2	R-L
GovReport			
Extractor Output	61.01	28.83	57.82
Oracle	**68.02**	**39.16**	**65.29**
QMSum			
Extractor Output	34.42	9.71	30.10
Oracle	**39.80**	**14.74**	**36.06**

Table 8: Feeding extractive oracles to generator. "Oracle" is computed based on the gold summary; thus, it is a soft upper-bound of the extractor's performance.

rized in Table 7. Due to the limit of GPU memory, the largest K value we tried is 25.

Effect of consistency loss We evaluate the effect of consistency loss on extractor performance. Note that removing the consistency loss means that the extractor and the generator are independently trained. The results are presented in Table 5 as part of the ablation study. Removing the consistency loss leads to worse model performance. We observe that the consistency loss helps the model better learn the importance of the selected text snippets useful for the generation.

Extractor performance compared with extractive oracles We feed the extractive oracles to the generator. The results are summarized in Table 8. We observe that extractive oracles contain more salient information than the text snippets extracted by the extractor. Feeding the extractive oracle to the generator indicates the upper bound of the extractor performance. However, we observe that the gap between the performance of using the extractive oracle and using the extractor output is relatively small.

Comparison with RAG The generator of our method is related to but differs significantly from Retrieval-Augmented Generation (RAG) (Lewis et al., 2020b). The similarity only lies in the idea of marginalization over a set of text snippets, which is shown to be useful in question answering as well (Ni et al., 2021b). However, unlike our *dynamic* weights, the weights in RAG remains *static* during decoding. In our notations, RAG's generation probability can be formulated as:

$$\begin{aligned} P_\theta(y|q, X_K) &= \prod_{t=1}^{T} P_\theta(y_t|q, X_K, y_{<t}) \\ &= \prod_{t=1}^{T} \sum_{x \in X_K} P_\theta(y_t|q, x, y_{<t}) P_\theta(x|q, X_K) \end{aligned} \tag{8}$$

The static weight $P_\theta(x|q, X_K)$ in Eq. 8 is computed based on q and X_K, while our dynamic weight $P_\theta(x|q, X_K, y_{<t})$ is additionally conditioned on the already generated tokens.

Limitations and future directions We acknowledge that joint training of the extractor and the generator cannot eliminate information loss, which might be addressed by combining DYLE and sparse attention to encode longer snippets. Though formulated for long-input summarization, DYLE can be applied to general long-input generation tasks where information is scattered across the input, e.g., open-domain question answering and multi-turn dialogue systems with long dialogue history.

7 Conclusions

In this paper, we propose the first framework that jointly trains an extract-then-generate model with latent extraction. The first-step extraction picks out salient information from the long input, thereby extending the input length that the model can handle. Our novel joint training method addresses the challenge of information loss associated with the prior extract-then-generate approaches. Our model largely outperforms the current state-of-the-art on GovReport and QMSum, while achieving strong results on arXiv. Lastly, DYLE has the advantages of being able to process arbitrarily long input with a lower memory cost and interpretable generator weights.

Acknowledgment

The authors would like to thank Yixin Liu and Ming Zhong for the discussions. We also would like to thank the anonymous reviewers for their helpful comments. This work is supported in part by a grant from Microsoft Research.

References

Sanghwan Bae, Taeuk Kim, Jihoon Kim, and Sang-goo Lee. 2019. Summary level training of sentence rewriting for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 10–20.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ahsaas Bajaj, Pavitra Dangati, Kalpesh Krishna, Pradhiksha Ashok Kumar, Rheeya Uppaal, Bradford Windsor, Eliot Brenner, Dominic Dotterrer, Rajarshi Das, and Andrew McCallum. 2021. Long document summarization in a low resource setting using pretrained language models. In *Proceedings of the ACL-IJCNLP 2021 Student Research Workshop, ACL 2021, Online, JUli 5-10, 2021*, pages 71–80.

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

Arthur Bražinskas, Mirella Lapata, and Ivan Titov. 2021. Learning opinion summarizers by selecting informative reviews. *arXiv e-prints*, pages arXiv–2109.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Mael Guillemot, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, et al. 2005. The ami meeting corpus: A pre-announcement. In *International workshop on machine learning for multimodal interaction*, pages 28–39. Springer.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost. *CoRR*, abs/1604.06174.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of ACL 2018, Long Papers*, pages 675–686.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of NAACL-HLT 2018, Short Papers*, pages 615–621, New Orleans, Louisiana.

Peng Cui and Le Hu. 2021. Sliding selector network with dynamic memory for extractive summarization of long documents. In *Proceedings of NAACL-HLT 2021*, pages 5881–5891.

Alexios Gidiotis and Grigorios Tsoumakas. 2020. A divide-and-conquer approach to the summarization of long documents. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28.

Quentin Grail, Julien Perez, and Eric Gaussier. 2021. Globalizing BERT-based transformer architectures for long document summarization. In *Proceedings of EACL 2021*, pages 1792–1810, Online.

Luyang Huang, Shuyang Cao, Nikolaus Nova Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1419–1436.

Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, et al. 2003. The icsi meeting corpus. In *ICASSP 2003.*, volume 1, pages I–I. IEEE.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *ICLR*.

Farshad Kiyoumarsi. 2015. Evaluation of automatic text summarizations based on human summaries. *Procedia-Social and Behavioral Sciences*, 192:83–91.

Anastassia Kornilova and Vladimir Eidelman. 2019. Billsum: A corpus for automatic summarization of us legislation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 48–56.

Logan Lebanoff, Kaiqiang Song, Franck Dernoncourt, Doo Soon Kim, Seokhwan Kim, Walter Chang, and Fei Liu. 2019. Scoring sentence singletons and pairs for abstractive summarization. In *Proceedings of ACL 2019*, pages 2175–2189, Florence, Italy.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of ACL 2020, Online, July 5-10, 2020*, pages 7871–7880.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*.

Yang Liu, Chenguang Zhu, and Michael Zeng. 2021. End-to-end segmentation-based news summarization. *arXiv preprint arXiv:2110.07850*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *NAACL-HLT*.

Ansong Ni, Zhangir Azerbayev, Mutethia Mutuma, Troy Feng, Yusen Zhang, Tao Yu, Ahmed Hassan Awadallah, and Dragomir Radev. 2021a. SummerTime: Text summarization toolkit for non-experts. In *Proceedings of EMNLP 2021: System Demonstrations*, pages 329–338.

Ansong Ni, Matt Gardner, and Pradeep Dasigi. 2021b. Mitigating false-negative contexts in multi-document question answering with retrieval marginalization. In *Proceedings of EMNLP 2021*, pages 6149–6161.

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

Tobias Rohde, Xiaoxia Wu, and Yinhan Liu. 2021. Hierarchical learning for generation with long source sequences. *arXiv preprint arXiv:2104.07545*.

Eva Sharma, Chen Li, and Lu Wang. 2019. Bigpatent: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of ACL 2019*, pages 2204–2213.

Xiaofei Sun, Chun Fan, Zijun Sun, Yuxian Meng, Fei Wu, and Jiwei Li. 2020. Summarize, outline, and elaborate: Long-text generation via hierarchical supervision from extractive summaries. *arXiv preprint arXiv:2010.07074*.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020a. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020b. Efficient transformers: A survey. *CoRR*, abs/2009.06732.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256.

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nissan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. 2021. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*.

Wen Xiao and Giuseppe Carenini. 2019. Extractive summarization of long documents by combining global and local context. In *Proceedings of EMNLP-IJCNLP 2019*, pages 3011–3021.

Jiacheng Xu and Greg Durrett. 2019. Neural extractive text summarization with syntactic compression. In *Proceedings of EMNLP-IJCNLP 2019*.

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. In *NeurIPS*.

Haoyu Zhang, Jingjing Cai, Jianjun Xu, and Ji Wang. 2019. Pretraining-based natural language generation for text summarization. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*.

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 Hassan Awadallah, Dragomir R. Radev, and Rui Zhang. 2021a. Summˆn: A multi-stage summarization framework for long input dialogues and documents. *CoRR*, abs/2110.10150.

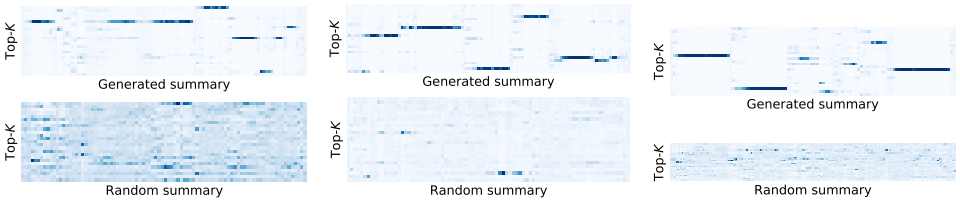
Yusen Zhang, Ansong Ni, Tao Yu, Rui Zhang, Chenguang Zhu, Budhaditya Deb, Asli Celikyilmaz, Ahmed Hassan Awadallah, and Dragomir Radev. 2021b. An exploratory study on long dialogue summarization: What works and what's next. In *EMNLP 2021: Findings*.

Ming Zhong, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021a. Dialoglm: Pre-trained model for long dialogue understanding and summarization. *arXiv preprint arXiv:2109.02492*.

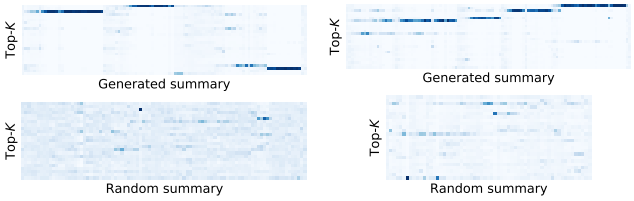
Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir R. Radev. 2021b. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of NAACL-HLT 2021, Online, June 6-11, 2021*.

Chenguang Zhu, Ruochen Xu, Michael Zeng, and Xuedong Huang. 2020. A hierarchical network for abstractive meeting summarization with cross-domain pretraining. In *Proceedings of EMNLP 2020: Findings*.

A Additional Dynamic Weight Visualization

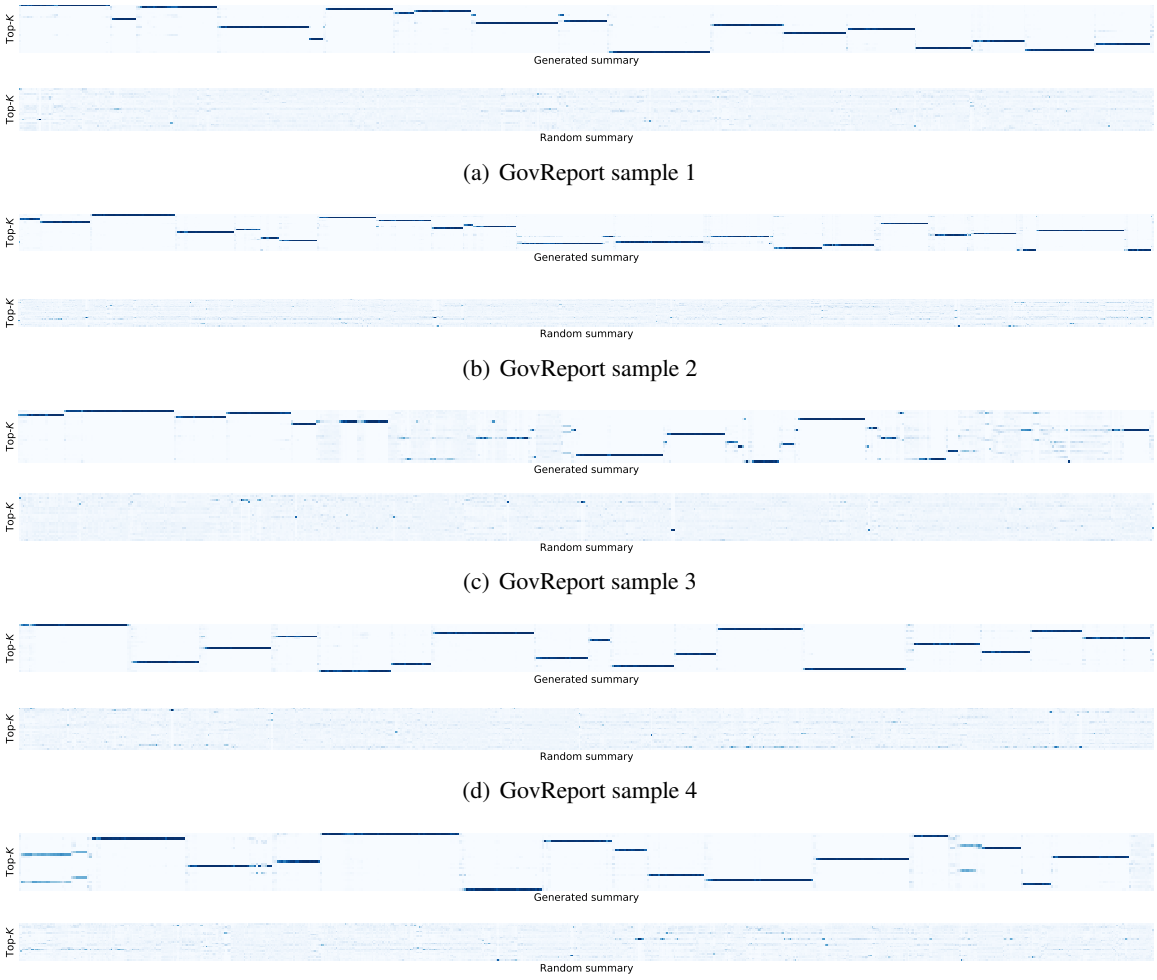


(a) QMSum sample 1 (b) QMSum sample 2 (c) QMSum sample 3



(d) QMSum sample 4 (e) QMSum sample 5

Figure 5: Dynamic weights visualization on QMSum.



(a) GovReport sample 1

(b) GovReport sample 2

(c) GovReport sample 3

(d) GovReport sample 4

(e) GovReport sample 5

Figure 6: Dynamic weights visualization on GovReport.