

DYLE: Dynamic Latent Extraction for Abstractive Long-Input Summarization

Ziming Mao^{*1} Chen Henry Wu^{*2} Ansong Ni¹ Yusen Zhang³

Rui Zhang³ Tao Yu⁴ Budhaditya Deb⁵

Chenguang Zhu⁵ Ahmed H. Awadallah⁵ Dragomir Radev¹

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 text summarization. However, they still struggle with long-input summarization. In this paper, we present a new approach for long-input summarization: Dynamic Latent Extraction for Abstractive Summarization. We jointly train an extractor with an abstractor and treat the extracted text snippets as the latent variable. We propose extractive oracles to provide the extractor with a strong learning signal. We introduce consistency loss, which encourages the extractor to approximate the averaged dynamic weights predicted by the generator. We conduct extensive tests on two long-input summarization datasets, GovReport (document) and QMSum (dialogue). Our model significantly outperforms the current state-of-the-art, including a 6.21 ROUGE-2 improvement on GovReport and a 2.13 ROUGE-1 improvement on QMSum. Further analysis shows that the dynamic weights make our generation process highly interpretable. Our code will be publicly available upon publication. ¹

1 Introduction

Transformer-based (Vaswani et al., 2017) pre-trained language models (PLMs), e.g., 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 mechanism (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 for long input summarization (Huang et al., 2021): models need to capture information scattered across the long input while maintaining a low computational cost.

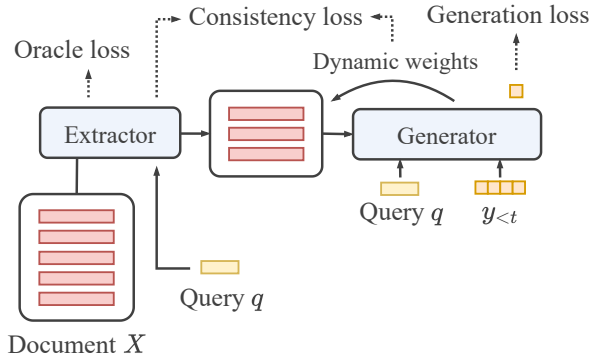


Figure 1: Graphical overview of our approach. The input is a document X and an optional query q , where each $x \in X$ is a sentence, and the output is a summary y of length T .

Prior models tackled long input summarization mostly following 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 based on 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). Third, models are proposed to divide source text into sections (Gidiotis and Tsoumakas, 2020; Wu 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) attempt to improve summarization by capturing sentence or discourse level dependencies. We elaborate on these four directions and their drawbacks in Section 5.

We believe that the extract-then-generate approach mimics the way a person would handle long-input summarization: identify important information in the text and then summarize them. This approach reduces the source inputs to a fixed

^{*}Equal Contributions.

¹<https://github.com/Yale-LILY/DYLE>

pre-set length, which addresses the main challenge of model not being able to handle longer input beyond a certain limit. However, previous separately-trained extract-then-generate approaches are limited as they break contextual dependencies between extracted chunks and suffer from cascaded errors from the extractor to the generator. Though various Reinforcement Learning techniques are introduced to bridge the two steps, it has noticeable drawbacks (explored in Section 2.3), and we argue that the nature of 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). We **jointly train** an extractor with an abstractor **using the extracted text snippets as the latent variable**. For an output token, we compute its probability conditioned on each input snippet *separately*, and its generation probability is computed by *marginalizing* over all input snippets under a learned distribution assigned by the extractor. The distribution over the input snippets is conditioned on the previously generated tokens. Output tokens are **dynamically generated** at each time step.

We propose to optimize the extractor using two surrogate losses. First, we compute the *extractive oracle* based on the gold output using greedy search. These oracle snippets are used as targets to optimize the extractor. Second, we propose the *consistency loss*, which encourages the extractor to approximate the averaged dynamic weights predicted by the generator.

We conducted experiments on two long-input summarization datasets: GovReport (Huang et al., 2021) for long document summarization and QM-Sum (Zhong et al., 2021b) for long dialogue summarization. Our method achieves state-of-the-art results on both datasets and significantly outperforms the currently best baselines. These experiments demonstrate the generalizability of our model to multiple long-input summarization tasks of different domains. The dynamic weights in our model improve the interpretability of the generation process and help denoise the extraction by down-weighting irrelevant text snippets. Our contributions are as follows.

- We introduce dynamic latent extraction for the abstractive long-input summarization task, a new approach that better captures information in the long input, allows interpretable dynamic weights, and reduces computational

complexity.

- We propose multiple auxiliary optimizations: extractive oracle as a learning signal for the extractor, consistency loss that bridges extraction and generation, hybrid training methods that furthers generalizability of the extractor.
- Experimental results show that our approach achieves state-of-the-art results on two long input summarization datasets in documents and dialogues. We also conducted detailed analysis on the interpretability of our model.

2 Our Approach

An overview of our approach is shown in Figure 1. In Section 2.1, we formulate our task and the extractor-generator framework. In Section 2.2, we introduce our parameterization of the extractor for long inputs. The extractor module could be optimized with the consistency loss and the oracle loss (Section 2.4). The overall training objective is summarized in Section 2.5.

2.1 Extractor-Generator Framework

We start by formulating the task of our interest. The input consists of L text snippets, $X = (x_1, \dots, x_L)$, and an optional query q for query-based summarization tasks. In long-input summarization, the number of text snippets, L , is usually very large. The output is a summary y of length T . For the dialogue summarization task, dialogue turns (utterances by each speaker) are used as snippets. For the long document summarization task, we tokenize the input into sentences and use the sentences as snippets. The goal is to learn a model that generates the sequence of summary tokens y given the input snippets X and the previously generated tokens $y_{<t}$:

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

The extractor-generator framework is based on the assumption that salient information useful for summarization only occupies a small portion of the input, which is a sensible assumption given the long input length. Specifically, the *extractor* takes the query and the document as input and outputs a score $s_i = E_{\eta}(q, x_i)$ for each text snippet x_i . Here η denotes the extractor parameters. To extract K

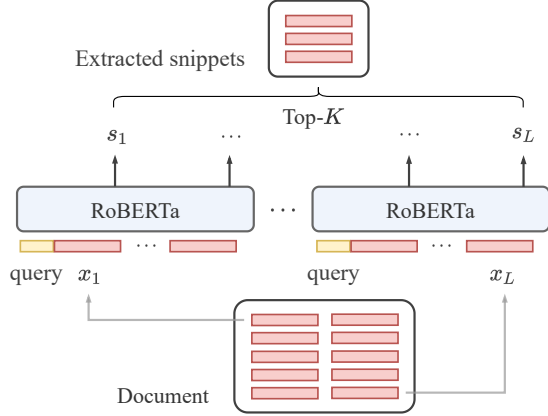


Figure 2: Extractor for long inputs. We divide the document into chunks, each containing neighboring snippets. A shared RoBERTa model encodes each chunk independently.

snippets X_K from the document X , we take the K snippets with highest scores:

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

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

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

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 2.3 and Section 2.4.

2.2 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 within the chunk it belongs to. We project the encoded vectors to scalar scores $s_i = E_\eta(q, x_i)$ using an MLP.

2.3 Generator with Dynamic Weights

A simple way to use the extracted snippets is to concatenate them into a single sequence and feed this sequence to a seq2seq generator. However, when applied to long-input summarization, it faces

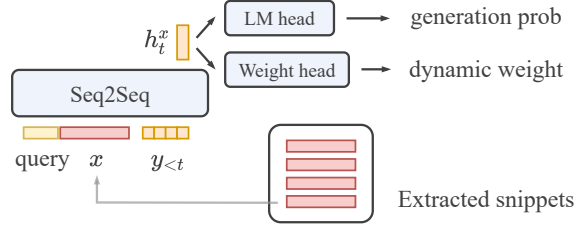


Figure 3: At each decoding time step, our generator predicts the dynamic weight and the generation probability for each extracted snippet.

two challenges. The first challenge is that the extraction operation, i.e., top- K in Eq. (1), is non-differentiable. One approach is to adopt RL-based optimizations (Chen and Bansal, 2018; Bae et al., 2019). However, this approach has three drawbacks if applied to long input summarization. Firstly, Reinforcement Learning for large action spaces (i.e., extracting K out of L snippets when L is very large) has high variances. The second challenge is that we cannot interpret how the generator utilizes the extracted snippets. For example, one may want to know whether the generator is leveraging extracted information at each decoding time step. Thirdly, current methods in fine-tuning extract-then-abstract models with RL either uses sentence-level ROUGE (Chen and Bansal, 2018) or summary-level ROUGE (Bae et al., 2019) as 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 as the training reward leads to the sparsity of the training signal, and longer input makes this approach even harder to train.

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 is independent of the decoding time step, generator assigns different dynamic scores at different time steps. The dynamic weights make the decoding process interpretable, and it also provides training signals for the extractor using what we term as the *consistency loss*.

Generator formulation The overview of the generator is shown in Figure 3. Specifically, for each extracted snippet x , the generator predicts the generation probability $P_\theta(y_t|q, x, y_{<t})$ based on this snippet and a dynamic weight $P_\theta(x|q, X_K, y_{<t})$ for this snippet. 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 h_t^x . For Transformers (Vaswani et al., 2017) and encoder-decoder with attention models (Bahdanau et al., 2015), 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 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 h_t^x to a scalar logit l^x , and $P_\theta(\cdot|q, X, y_{<t})$ is defined as $\text{softmax}(\{l^x\}_{x \in X})$. The generation probability is computed 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 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}^θ to the extractor parameters since top- K is non-differentiable.

Consistency loss We also leverage the dynamic weights to provide 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 can average the dynamic weights over all 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 weight 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

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

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.

2.4 Leveraging Extractive Oracles

For long-input summarization, the extracted snippets X_K used during training are important for the stable optimization. Instead of using the definition of X_K in Eq. (1), which is adopted at test time, we propose to leverage *extractive oracles* during training.

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 are 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. Specifically, it is possible that there are other text snippets in the source input that are omitted in the greedy oracle extraction but could still help the generation. This way, hybrid training allows our model to capture a greater variety of source text snippets.

Oracle loss The extractive oracles X_o are used as a supervision signal for the extraction part of our model. The oracle loss $\mathcal{L}_{oracle}^\eta$ 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

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

2.5 Training Objective

The overall training objective of our method is

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

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.

3 Experiments

3.1 Datasets and Baselines

QMSum (Zhong et al., 2021c) is a benchmark for query-based multi-domain meeting summarization. It consists of meetings from three domains, including AMI and ICSI, and the committee meetings of the Welsh Parliament and Parliament of Canada. The meetings in this dataset comprise of a large number of turns uttered by multiple speakers.

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). The GovReport authors also note that salient information is more scattered across the documents. A comparison of the datasets is presented in Table 1.

Baselines We used reported baselines. The baselines for the QMSum paper came from (Zhong et al., 2021a). The baseline used for GovReport came from the original paper (Huang et al., 2021), which experiments with multiple encoder self-attention, encoder-decoder attention patterns on BART-large.

3.2 Implementation Details

The extractor is initialized with the pretrained Roberta-base model (Liu et al., 2019). The generator is initialized with the BART-large (Lewis et al., 2020a) model. Both models used Hugging Face implementations. Training is done with the Adam optimizer. We apply gradient checkpointing (Chen et al., 2016) to both the extractor and the generator to save memory. Each experiment is run on a single NVIDIA Quadro RTX 8000 GPU. We set the batch size as 8 (1 sample per forward pass with gradient accumulation step set to 8).

ROUGE (Lin, 2004) is used as the automatic evaluation metrics throughout all experiments. We split the sentence in each generated summary to obtain the full ROUGE-L scores.

3.3 Automatic Evaluation

For automatic evaluation, we report ROUGE-1, ROUGE-2, and ROUGE-L. The results are summarized in Table 2 and Table 3.

Our model DYLE achieved state-of-the-art performance on both datasets. On the GovReport dataset, we achieved **4.15** ROUGE-1 improvement, **6.21** ROUGE-2 improvement, and **4.00** ROUGE-L improvement compared to the currently best model.

On the QMSum dataset, we achieved **2.13** ROUGE-1 improvement, **1.04** ROUGE-2 improvement, and **1.93** ROUGE-L improvement compared to the state-of-the-art model HMNet.

These results show that DYLE can be applied to both the long document summarization and long dialogue summarization tasks.

3.4 Evaluation of Auxiliary Optimizations

We conduct detailed 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) oracle loss. The results are summarized in Table 4. Without the hybrid training optimization, only the extractive oracles will be used to train the generator.

We see that excluding either of the hybrid training, consistency loss, and 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 there

Dataset	Src. length	Tgt. length
Document		
GovReport (Huang et al., 2021)	9409	553
PubMed (Cohan et al., 2018)	3049	202
ArXiv (Cohan et al., 2018)	6030	273
BillSum (Kornilova and Eidelman, 2019)	1813	208
BigPatent (Sharma et al., 2019)	3573	117
Dialogue		
QMSum (Zhong et al., 2021c)	9070	69
AMI (Carletta et al., 2005)	6008	297
ICSI (Janin et al., 2003)	13317	489
MediaSum (Zhu et al., 2021)	1554	14
SummScreen (Zhu et al., 2021)	6613	337

Table 1: Comparison of Document and Dialogue Summarization Dataset.

	R-1	R-2	R-L		R-1	R-2	R-L
Bart-large variants				GovReport			
Full (1024)	52.83	20.50	50.14	Full	61.01	28.83	57.82
Stride (4096)	54.29	20.80	51.35	w/o hybrid	60.89	28.28	57.31
LIN. (3072)	44.84	13.87	41.94	w/o consistency	60.59	28.48	57.49
LSH (4096)	54.75	21.36	51.27	w/o oracle	57.57	25.92	53.14
Sinkhorn (5120)	55.45	21.45	52.48	QMSum			
Bart-large + HEPOS				Full	34.42	9.71	30.10
LSH (7168)	55.00	21.13	51.67	w/o hybrid	31.77	8.33	28.37
Sinkhorn (10240)	56.86	22.62	53.82	w/o consistency	32.51	8.77	28.94
DYLE (ours)	61.01	28.83	57.82	w/o oracle	32.13	8.38	28.63

Table 2: Automatic evaluation on GovReport, where R stands for ROUGE metric

Table 4: Experiments on auxiliary optimizations.

	R-1	R-2	R-L
Baseline with Locator			
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
With DialogLM Pretraining			
DialogLM (5120)	34.02	9.19	29.77
DialogLM - Sparse (8192)	33.69	9.32	30.01
DYLE (ours)	34.42	9.71	30.10

Table 3: Automatic evaluation on QMSum.

could still be text snippets that can further improve the summarization but are not included in the greedily extracted oracles.

3.5 Analysis of Extracted Snippets

We are interested in the amount of salient information passed to the generator. To investigate, we report the decomposed precision and recall of ROUGE scores in Table 5. We observe that the extracted snippets have much higher recall than the generated summaries, and the generated summaries

have higher precision. It suggests that one way to improve the overall performance is to increase the information coverage (i.e., recall) of the extractor and to more accurately identify the salient snippets (i.e., precision) in the generator.

4 Discussion

Capacity to Summarize Longer Input This paper demonstrates the effectiveness of latent extraction for abstractive summarization, in both long document summarization and long dialogue summarization. The first-step extraction picks out salient information from the long input, thereby greatly extending the input length that the model can handle. Previous Bart-large-based baselines on GovReport, even with sparse encoder self-attention and encoder-decoder attention, are only able to process up to 10,240 tokens. However, our model can handle the long input at its full length.

Interpretability of Dynamic Weights Our approach is more interpretable than sparse attention and two-step extraction-generation pipeline methods. Specifically, *dynamic weights* in the generator show how the information is used throughout the

	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 summary	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 summary	29.78	45.64	34.42	8.39	13.06	9.71	26.14	39.70	30.10

Table 5: Precision-recall decomposition of ROUGE scores.

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 index of the extracted top- K snippets and the decoding time step, 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. By contrast, weights are uniformly distributed for random summaries.

Comparison with RAG The generator of our method is related to but differ 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. However, unlike our *dynamic* weights, the weights in RAG remains *static* during decoding. Specifically, using our notations, RAG decomposes the generation probability 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. Furthermore, RAG retrieves a set of documents, whereas our extractor extracts text snippets from the input.

Effect of K in top- K 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 model performance generally increases as the value of K

	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 6: Experiments on different values of K .

	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 7: Experiments of feeding extractive oracles to generator.

increases. Results are summarized in Table 6. Due to the limitation of computational resources, the largest K value we tried is 25.

Extractor performance We feed the extractive oracle to the generator. The results are summarized in Table 7. We observe that extractive oracle contains more salient information than the text snippets extracted by extractor.

Future Directions One future direction is to adapt our approach to other long input generation tasks, such as open-domain question answering and response generation in multi-turn dialogue systems when the dialogue history is long.

5 Related Work

Sparse Attention Mechanism The full attention mechanism has a quadratic dependency on mem-

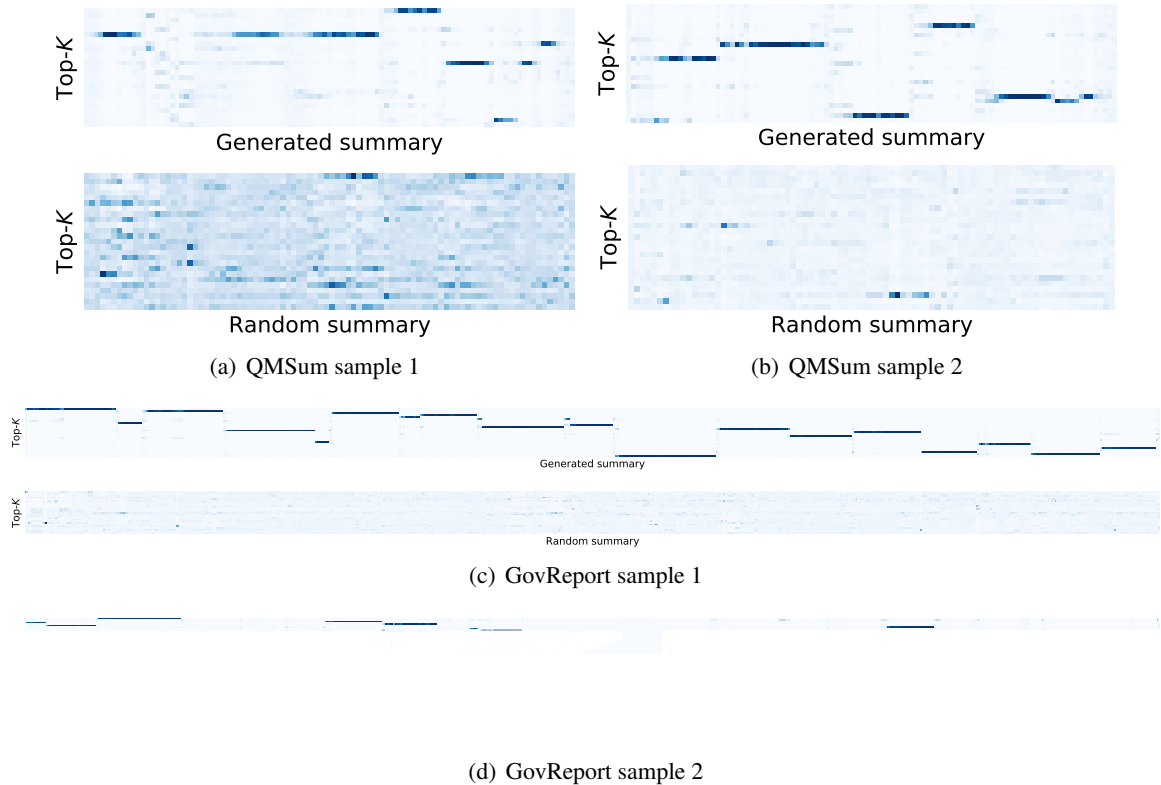


Figure 4: Dynamic weight visualization. We visualized the dynamic weight matrices of the *generated summary* and a *random summary* from other samples in the validation set. x -axis: decoding time step; y -axis: index of the extracted top- K snippets. Darker squares stand for higher weights.

ory. Prior research works have proposed different sparse attention mechanisms that reduce the memory cost. Longformer (Beltagy et al., 2020) uses a dilated sliding window of blocks and task-motivated global attention patterns. BigBird (Zaheer et al., 2020) treats attention patterns as a graph sparsification problem and employs sliding window and random blocks to simplify attention complexity. Reformer (Kitaev et al., 2020) makes use of the locality-sensitive hashing to reduce the memory complexity. 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-abstract Method The model first extracts salient text snippets from the input, followed by rewriting them abstractively to generate a concise overall summary. Most two-stage summarization approaches (Zhang et al., 2019; Lebanoff et al., 2019; Xu and Durrett, 2019; Bajaj et al., 2021) are trained separately, which suffer from information loss due to the cascaded errors. Some

approaches attempt to reduce that loss by bridging the two stages. Chen and Bansal (2018) adopts an extract-then-rewrite approach using Reinforcement Learning with a sentence-level policy gradient method. Bae et al. (2019) improves it by using summary-level policy gradient. In addition to the drawbacks explained in Section 2.3, our model is different as we jointly train an extract-then-abstract model for summarization using latent variables.

Divide-and-conquer Approach A common approach in long input summarization is divide-and-conquer (Gidiotis and Tsoumakos, 2020; Grail et al., 2021). This approach breaks a long input into multiple parts, which are summarized separately and later combined to produce a final complete summary. However, these models do not capture the contextual dependencies across parts and need to assume a certain structure of the input (such as paper sections).

Hierarchical Models Various hierarchical models have been proposed to handle the longer inputs. Cohan et al. (2018)’s model consists of a hierarchical encoder that models the document dis-

course structure and an attentive discourse-aware decoder to generate the summary. 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.

Latent Retrieval in Open-domain QA RAG (Lewis et al., 2020b) used a parametric memory of the Transformer and a non-parametric memory of Wikipedia vector index. It trains these components in a probabilistic model end-to-end. REALM (Guu et al., 2020) augments language model pre-training with a latent knowledge retriever. Both these papers are on open-domain Question Answering and little attention has been given to long input summarization using latent variables.

6 Conclusions

In this paper, we propose the first framework that **jointly trains** an extract-then-abstract model with latent extraction for long input summarization. We demonstrate its effectiveness by testing on the GovReport and QMSum datasets. Our model significantly outperforms the current state-of-the-art on both, while having the advantages of being able to process arbitrary long input, low memory cost, and interpretable generator weights.

Acknowledgment

The authors would like to thank Ming Zhong and Yixin Liu for their discussions. 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 *3rd International Conference on Learning Representations, 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 Dotterer, 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. Association for Computational Linguistics.

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

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 the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 675–686. Association for Computational Linguistics.

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 the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

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*, 28:3029–3040.

Quentin Grail, Julien Perez, and Eric Gaussier. 2021. [Globalizing BERT-based transformer architectures for long document summarization](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1792–1810, Online. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. [REALM: retrieval-augmented language model pre-training](#). *CoRR*, abs/2002.08909.

- Luyang Huang, Shuyang Cao, Nikolaus Nova Parulian, Heng Ji, and Lu Wang. 2021. [Efficient attentions for long document summarization](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1419–1436. Association for Computational Linguistics.
- 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 *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP'03)*, volume 1, pages I–I. IEEE.
- Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. [Reformer: The efficient transformer](#). In *International Conference on Learning Representations*.
- 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 the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2175–2189, Florence, Italy. Association for Computational Linguistics.
- 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 *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Chin-Yew Lin. 2004. [ROUGE: A package for automatic evaluation of summaries](#). In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- 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*.
- 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 the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213.
- 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*.
- Jiacheng Xu and Greg Durrett. 2019. [Neural extractive text summarization with syntactic compression](#). 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 3292–3303, Hong Kong, China. 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. 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)*, pages 789–797.

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 the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5905–5921. Association for Computational Linguistics.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. 2021c. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921.

Chenguang Zhu, Yang Liu, Jie Mei, and Michael Zeng. 2021. Mediasum: A large-scale media interview dataset for dialogue summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5927–5934.

Chenguang Zhu, Ruochen Xu, Michael Zeng, and Xuedong Huang. 2020. A hierarchical network for abstractive meeting summarization with cross-domain pretraining. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 194–203.

A Additional Dynamic Weight Visualization

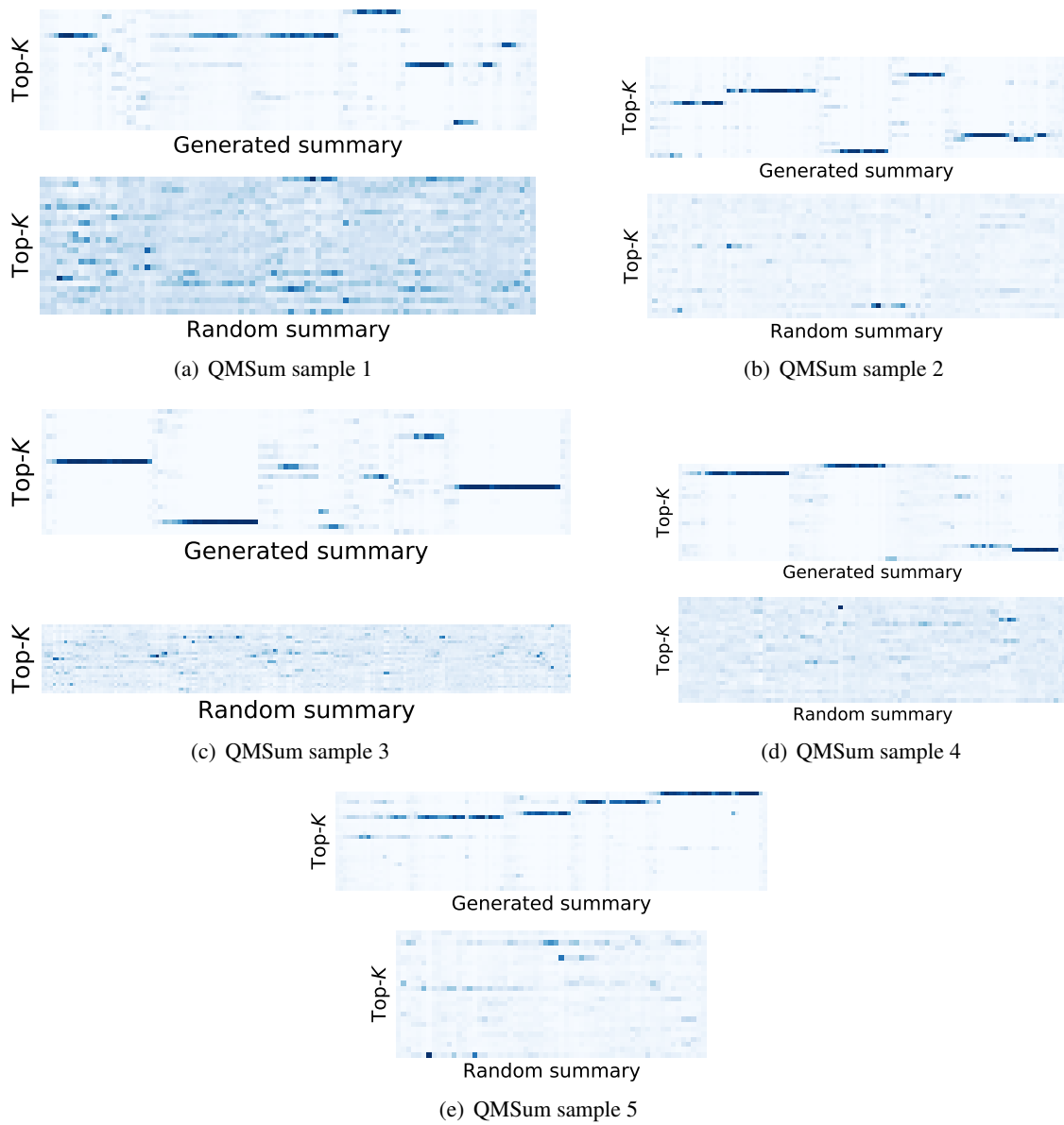
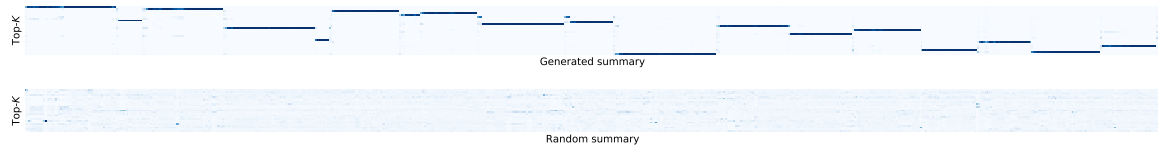


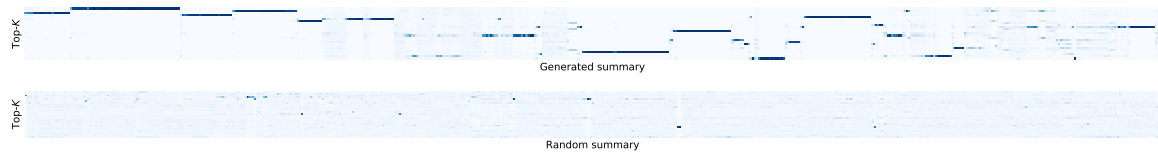
Figure 5: Dynamic weights visualization on QMSum.



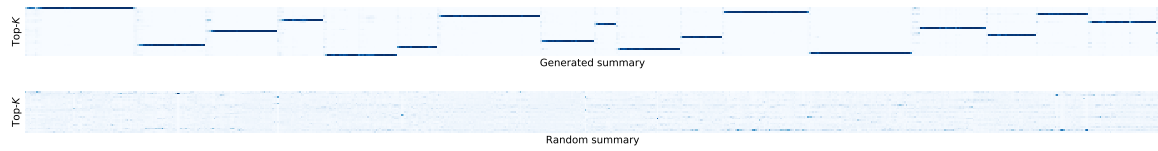
(a) GovReport sample 1



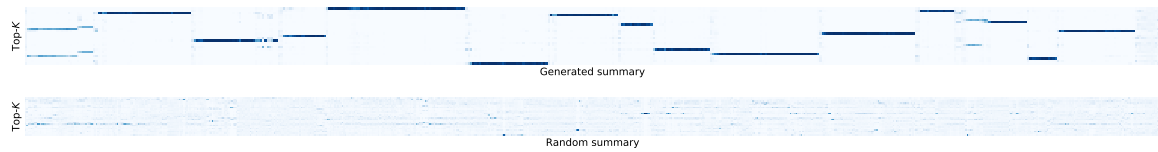
(b) GovReport sample 2



(c) GovReport sample 3



(d) GovReport sample 4



(e) GovReport sample 5

Figure 6: Dynamic weights visualization on GovReport.