AWESOME: GPU Memory-constrained Long Document Summarization using Memory Mechanism and Global Salient Content

Shuyang Cao and **Lu Wang**
Computer Science and Engineering
University of Michigan
Ann Arbor, MI
{caoshuy, wangluxy}@umich.edu

Abstract

Long document summarization systems are critical for domains with lengthy and jargon-laden text, yet they present significant challenges to researchers and developers with limited computing resources. Existing solutions mainly focus on efficient attentions or divide-and-conquer strategies. The former reduces theoretical time complexity, but is still memory-heavy. The latter methods sacrifice global context, leading to uninformative and incoherent summaries. This work aims to leverage the memory-efficient nature of divide-and-conquer methods while preserving global context. Concretely, our framework AWESOME uses two novel mechanisms: (1) *External memory mechanisms* track previously encoded document segments and their corresponding summaries, to enhance global document understanding and summary coherence. (2) *Global salient content* is further identified beforehand to augment each document segment to support its summarization. Extensive experiments on diverse genres of text, including government reports, meeting transcripts, screenplays, scientific papers, and novels, show that AWESOME produces summaries with improved informativeness, faithfulness, and coherence than competitive baselines on longer documents, while having a smaller GPU memory footprint.

1 Introduction

Large pre-trained transformer models have demonstrated impressive performance across popular abstractive summarization benchmarks (Lewis et al., 2020; Raffel et al., 2020). Yet, transformer's quadratic **memory complexity** presents challenges for summarizing long documents with more than hundreds of words, such as scientific papers and investigation reports (Cohan et al., 2018; Huang et al., 2021), making it infeasible for researchers and developers with limited hardware resources (e.g., GPUs with insufficient memories) to contribute to this important research field.

The NLP community has made several innovations to address the long document challenge. Prior work divides a document into smaller chunks and summarizes each separately (Gidiotis and Tsoumakas, 2020), reduces the complexity of attention calculations (Beltagy et al., 2020), and removes unimportant content before running an abstractor (Pilault et al., 2020). In terms of memory efficiency, divide-and-conquer methods obtain the most significant advantage (Moro and Ragazzi, 2022). However, information outside of a document segment and their corresponding summaries become inaccessible, leading to uninformative and incoherent summaries. Unsurprisingly, state-of-the-art performance is obtained by models that can maintain global context, e.g., by combining global attentions with local attentions in transformer-based summarization models (Zaheer et al., 2021; Phang et al., 2022). Yet, they still require a large GPU memory footprint in practice.[1] Though large language models like GPT-4 (OpenAI, 2023) are trained to handle up to 128K tokens, the privacy and security of data transmitted and shared through the API remain concerning, particularly in sectors dealing with sensitive information, e.g., clinical notes. Local model development can bolster privacy and security; however, limited computational resources in these scenarios necessitate the exploration of efficient modeling techniques.

Therefore, this work aims to address the problem of long document summarization using constrained resources, specifically focusing on *constrained GPU memory*. We propose **AWESOME**[2], which is built on the memory-efficient divide-and-conquer approach, and Augmented With Estimated Salient cOntent and MEmory mechanism. In essence,

[1]These approaches require a GPU memory of >40GB to process documents with over 8K tokens, while the most cost-effective GPUs only have 24GB of memory (Li, 2022).

[2]Our code is publicly available at https://shuyangcao.github.io/projects/awesome/.

AWESOME maintains global context of both the source document and the summary generated so far with a limited memory usage, to enhance summary informativeness, faithfulness, and coherence.

First, **external memory mechanism** is used on the encoder side of AWESOME to store information as it sequentially reads document segments. This maintains relevant context for improving document understanding and salient content detection, thus promoting summary informativeness and faithfulness. Another memory is applied on the decoder side to boost generation coherence by tracking the partial summaries generated for previous document segments. Importantly, to ensure the GPU memory efficiency of AWESOME, we *curb gradients from propagating to other document and summary segments* and only implement the external memory in a *limited* number of layers.

Second, AWESOME incorporates **global salient content** selected by an efficiently trained extractor through (1) direct text concatenation, or (2) inserting their key-value matrices into attention calculation. This lets the summarizer be aware of important topics at a global level, to enhance salience estimation and summary informativeness.

We experiment with five popular long-input benchmarks of different genres: investigation reports in GovReport (Huang et al., 2021), meeting transcripts in QMSum (Zhong et al., 2021), TV screenplays in SummScreen (Chen et al., 2022), scientific papers in arXiv (Cohan et al., 2018), and fictions in BookSum (Kryscinski et al., 2022). First, on all the five datasets, all AWESOME variants uniformly outperform Se3 (Moro and Ragazzi, 2022), the divide-and-conquer baseline, on summary informativeness as evaluated by ROUGE (Lin, 2004) and on coherence as measured by DiscoScore (Zhao et al., 2022) and a metric based on entity graphs (Guinaudeau and Strube, 2013)—both metrics are highly correlated with human judgment, according to Zhao et al. (2022). Second, AWESOME with memory mechanisms also improves summary faithfulness over Se3 on GovReport, according to SummaC (Laban et al., 2022), an entailment-based faithfulness metric. Lastly, compared with more memory-intensive models that also maintain global context, such as Phang et al. (2022) and Liu et al. (2022), AWESOME achieves better or comparable automatic scores for informativeness, coherence, and faithfulness on GovReport (Huang et al., 2021). On BookSum which comprises the lengthiest documents and summaries among the five datasets, AWESOME produces more informative and coherence outputs than recent models.

Approach	In→Out	Enc	Enc←Dec	Dec
Efficient Attention	$x \rightarrow y$	■	■	●
Extract-Abstract	$x_e \rightarrow y$	□	★	●
Dynamic Weight	$x \rightarrow y$	□	■ + ★	●
Divide-Conquer	$x_i \rightarrow y_i$	□	□	○

Table 1: Existing approaches to long document summarization (§2.1). **In→Out**: Longer inputs ($|x| > |x_e| > |x_i|$) or outputs ($|y| > |y_i|$) produce more nodes in the computation graph, thus the higher memory consumption. **Enc**: Encoder accessing partial documents (□) hurts document understanding, compared to reading the full text (■). **Enc←Dec**: Decoder reading the full document (■) or pre-identified salient content (★) enhances summary informativeness, compared to a segment (□). **Dec**: Decoder accessing previously generated summary content (●) is crucial for generation coherence than reading a current summary segment only (○).

Our contributions are summarized as follows:

1. We conduct a comprehensive study of existing approaches to long document summarization, revealing that the number of tokens involved in training computation significantly affects GPU memory usage.

2. We design AWESOME based on the GPU memory-efficient divide-and-conquer approach. AWESOME leverages the memory mechanism and global salient content augmentation to compensate the context loss due to the divide-and-conquer process.

3. We experiment with diverse long-document summarization datasets, showing the effectiveness of AWESOME.

2 Related Work

2.1 Efficient Long Document Summarization

We categorize existing efficient long document summarization models into four major types, as summarized in Table 1. The model **input** can be an original document, extracted important segments of the document, or a document segment, which are denoted as x, x_e, or x_i (for the i-th segment), and typically, $|x| > |x_e| > |x_i|$. The **output** can be the full summary y or a summary segment y_i (for x_i), where $|y| > |y_i|$. Importantly, longer inputs and outputs expand larger computation graph,

leading to higher GPU memory usage. Moreover, we analyze both the **document context** and the **summary context** used by each approach when generating summaries. Specifically, we check (1) full vs. partial documents that are consumed to obtain the encoder representations (**Enc**); (2) full vs. partial encoder representations that are attended by the decoder (**Enc**←**Dec**); and (3) full vs. partial output that is accessed by the decoder (**Dec**).

Efficient attentions are designed to reduce the quadratic complexity of the original transformer architecture (Vaswani et al., 2017) and maintain full encoding context by combining global attentions with local attentions built on sliding windows (Beltagy et al., 2020; Zaheer et al., 2021), text blocks (Phang et al., 2022; Tay et al., 2020), or clusters of similar tokens (Kitaev et al., 2020; Roy et al., 2021). Besides the aforementioned attention variants designed for self-attentions, recent work has reduced the memory usage of decoder cross attentions by distributing encoder outputs to different attention heads (Huang et al., 2021) or selecting attendable encoder outputs via KNN search (Bertsch et al., 2023). Despite the reduced complexity, efficient attention-based systems effectively require reading the full document x to generate a summary y during model training and thus still need huge GPU memory that scales with the input length.

Extract-then-abstract systems circumvent the long sequence challenge by first identifying the salient segments, x_e (e.g., sentences), using an extractor, and then running an abstractor over x_e to produce the final summary (Pilault et al., 2020; Liu and Lapata, 2019; Zhao et al., 2020). However, the extracted segments may contain incomplete and out-of-context information that leads to incomprehensible and unfaithful summaries.

To mitigate the error propagation issue of a two-stage approach, recent studies bridge the extractor and abstractor via **dynamic weights** over document segments. Rather than feeding the extracted segments directly to the abstractor, at each summary decoding step, DYLE (Mao et al., 2022) first predicts an output token distribution for each segment separately, and then aggregates over all the extracted segments as weighted by their extraction salience. PageSum (Liu et al., 2022) further alleviates context loss by averaging decoder output representations conditioned on all document segments. Though their abstractor processes each document segment x_i separately, jointly training the extractor and the abstractor still requires loading the full document x into the GPU memory.

Divide-and-conquer systems split a document into multiple non-overlapping segments and summarize each segment separately, as done in Gidiotis and Tsoumakas (2020) and Se3 (Moro and Ragazzi, 2022). Summ^N (Zhang et al., 2022) uses an additional summarization stage to further condense the segmented summaries. As each document segment x_i is summarized separately, the divide-and-conquer approach's fixed GPU memory footprint is independent from the document length. This fits well with our goal of long document summarization with limited memory. However, without access to other parts of the document and their summaries, the summarizer struggles for content salience estimation in each isolated segment, and generates incoherent outputs when piecing together summaries. Though Wu et al. (2021) concatenate previously generated summaries as part of the input, a complicated strategy is required for training sample construction.

AWESOME is built on the memory-efficient divide-and-conquer approach, and improves summary informativeness, coherence, and faithfulness by using newly designed external memories for accumulating salient information from other document segments and their generated summaries. We further augment AWESOME with global salient content to provide important topics at the document level, when summarizing each segment.

2.2 Memory and Content Augmentation

Different memory mechanisms have been studied for long-range text *understanding* tasks. For instance, Transformer-XL (Dai et al., 2019) caches intermediate representations produced in the last document segment and attends over these representations. Compressive Transformer (Rae et al., 2020) further increases the context range by compressing the oldest cached representations. To simulate memory reading and writing, Recurrent Memory Transformer (Bulatov et al., 2022) includes extra memory vectors in each text segment and passes their corresponding output vectors to the next segment. Instead of using a memory with a fixed size, Memorizing Transformer (Wu et al., 2022a) stores all prior representations as key-value pairs, and performs an approximate kNN lookup to retrieve representations to augment the current segment. However, existing work on memory mechanisms focuses on language modeling, while *incorporating memory mechanisms into the decoding process for*

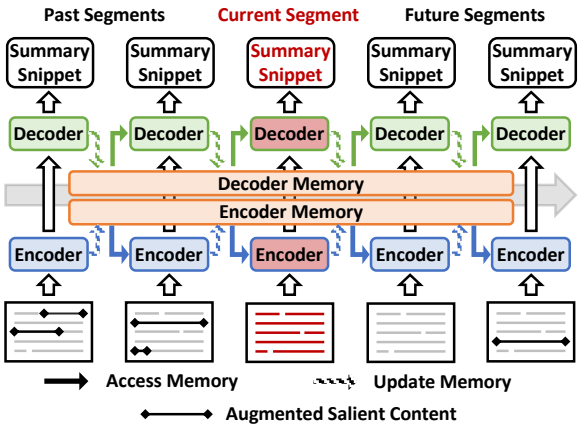

Figure 1: Illustration of AWESOME. Encoder and decoder **memories** can be accessed any time and updated after reading each document segment and generating the corresponding summary. They accumulate global context that improves summary informativeness and coherence (§3.1). When encoding each segment, global salient content from other segments (lines with ♦-shaped ends, from both past and future) are provided to further assist salience estimation (§3.2).

generation tasks is nontrivial as it requires updating both decoding states (e.g., beams) and memory states. Our work is the first to leverage parametric memory mechanisms and content augmentation to incorporate global context for the purpose of memory-efficient long document summarization.

3 External Memory and Global Salient Content Augmentation

The architecture of AWESOME (Figure 1) is based on Se3 (Moro and Ragazzi, 2022), where a document is summarized segment by segment, with the final summary obtained by concatenating the resultant summaries. Document sentences are split into segments with up to 768 tokens each, while reference summary sentences are assigned to their most overlapping segment to create the oracle summary, as detailed in Appendix A. Following Longformer (Beltagy et al., 2020), we initialize the encoder and decoder parameters from BART (Lewis et al., 2020). AWESOME preserves the global context and builds communications across segments with minimal GPU memory increase, by (1) employing external memories in *both the encoder and the decoder* to gather relevant information (§3.1), and (2) augmenting *the encoder* with salient content from other segments (§3.2).

3.1 External Memory Mechanisms

We design two external memory mechanisms to efficiently enable the information flow from prior segments to the current segment. Specifically, each memory module maintains a matrix $M \in \mathbb{R}^{m \times d}$, where $m = 1024$ is the memory size and $d = 1024$ is the hidden state dimension of BART. M is updated after encoding each document segment and then passed to the next segment. We denote the memory matrix after the t-th segment as M^t. Each layer of the encoder and decoder can be equipped with one such external memory. Below we describe two mechanisms to update M^t and incorporate it in both the encoding and decoding processes. The layer index in the formulas is omitted for simplicity.

Compressive Memory. For each document segment, compression-based memory caches its input vectors to be fed into self-attention calculation. Since storing the input vectors as-is requires the memory usage m to scale linearly with the context length, we dedicate half of M^t to store the compressed memory, with a compression ratio of r. With H^t_{inp} denoting the matrix that contains input vectors to the transformer self-attention, the memory compression and update processes are:

$$M^{t-1}_c, M^{t-1}_u = M^{t-1}[:\frac{m}{2}], M^{t-1}[\frac{m}{2}:] \quad (1)$$

$$M'_u = \text{concat}(M^{t-1}_u, \text{SG}(H^t_{inp})) \quad (2)$$

$$M'_c = \text{compress}(M'_u[:-\frac{m}{2}]) \quad (3)$$

$$M^t_u = M'_u[-\frac{m}{2}:] \quad (4)$$

$$M^t_c = \text{concat}(M^{t-1}_c, M'_c)[-\frac{m}{2}:] \quad (5)$$

$$M^t = \text{concat}(M^t_c, M^t_u) \quad (6)$$

where $\text{SG}(\cdot)$ denotes stopping the gradient backpropagation to lower GPU usage, and $\text{compress}(\cdot)$ performs convolutions with their stride and kernel size set to the compression ratio r. r is set to 5 after tuning on the development sets.

Next, to leverage the memory from the previous segment in summarizing the current segment, M^{t-1} is concatenated with the inputs to the self-attentions to obtain the key-value matrices:

$$H^t_{mem} = \text{concat}(M^{t-1}, H^t_{inp}) \quad (7)$$

$$H^t_{self} = \text{Attn}(\underbrace{H^t_{inp}}_{query}, \underbrace{H^t_{mem}}_{key}, \underbrace{H^t_{mem}}_{value}) \quad (8)$$

where H^t_{self} is the output of the self-attention.

Our compression-based memory is adopted from Compressive Transformer (Rae et al., 2020), a decoder-only model for language modeling. We are the first to apply it to both the encoder and the decoder of a Transformer model and on long document summarization tasks.

Compressive memory favors recency, particularly the previous segment and its summary, potentially causing older relevant history to be lost during compression.

Attentive Memory. To mitigate the recency bias by compressive memory, we further investigate an attention-based memory updating mechanism, to selectively include content in M^t. First, the memory is additionally accompanied by an extra cross-attention in each of the encoder and decoder layers, specialized in retrieving relevant information from M^t. Following prior study (Lei et al., 2020) that uses memories in video captioning, we update M^t with a gate matrix G^t to control the amount of content to be updated:

$$M^t = G^t \odot U^t + (1 - G^t) \odot M^{t-1} \tag{9}$$

where $\odot$ denotes the element-wise product and U^t is the matrix containing vectors to update the memory. U^t and G^t are obtained as follows:

$$U^t = \tanh(W_{u1} M^{t-1} + W_{u2} S^t) \tag{10}$$

$$G^t = \sigma(W_{g1} M^{t-1} + W_{g2} S^t) \tag{11}$$

$$S^t = \mathrm{Attn}(\underbrace{M^{t-1}}_{query}, \underbrace{\mathrm{SG}(H^t_{self})}_{key}, \underbrace{\mathrm{SG}(H^t_{self})}_{value}) \tag{12}$$

where W_* are learnable matrices, S^t synthesizes the current segment via an attention calculation, and $\mathrm{SG}(\cdot)$ indicates stopping the gradient back-propagation. In each encoder and decoder layer, an extra cross-attention is inserted after the self-attention, where M^{t-1} is attended and incorporated into the current segment's summarization process.

Unlike our approach, the memory in Lei et al. (2020) does not employ gradient stopping. This omission eliminates the memory efficiency gained from the divide-and-conquer strategy, leading to comparable high memory usage as the efficient attention strategy.[3] While their memory is suitable for generating short image captions, our design with gradient stopping is crucial for efficient long document summarization.

[3]Without gradient stopping, the model fails to complete training with 48GB GPU memory.

Selective Installation of External Memory. To mitigate the GPU memory overhead incurred by external memory, we selectively add external memory to specific layers. Due to the high computational cost of exhaustively searching for the optimal layer or combination of layers for each dataset, we divide the layers of BART into four groups, each comprising three layers. We test the performance of installing external memory in each group separately and select the group that shows the best overall performance on the validation set of GovReport.[4] The last three layers are chosen for attentive memory, while the first three layers are selected for compressive memory. We believe the limited adaptability of compressive memory prevents its effective application in the latter layers.

3.2 Global Salient Content Augmentation

The memory mechanisms only grant access to prior content in the documents, yet subsequent context can also help with salience estimation, e.g., elaborating the pros and cons of a proposed solution makes it necessary to introduce the problem and the solution. Moreover, memories store content implicitly, so it is unclear whether relevant information can be stored and retrieved effectively. Therefore, we inform the system of a document's important sentences, which are pre-identified by a separately-trained extractor. The details of extractor training can be found in Appendix D. After extracting important sentences in a document, we study two methods of injecting them into the summarizer.

Text Concatenation. For each segment, we include the extracted sentences in the following way to prioritize long-term context. We start with the "outermost" extracted sentences, i.e., the earliest sentence in the past segments and the last sentence in the future segments, and repeat this process until the input has reached the maximum length accepted by the positional encoding of the model (1024 for BART).[5] To differentiate the content in the current segment from the added sentences, we prefix the current segment and the added sentences from before/after the current segment with "`Current chunk:`", "`Previous important sentences:`", and "`Next important sentences:`", respectively. Text concatenation is easy to implement and most compatible with the source modality, but the mem-

[4]Selective installation reduces GPU memory usage by approximately 9GB.

[5]Other inclusion strategies can be explored in future work.

Dataset	# Samples Train	Dev	Test	# Word Doc	Summ
GovReport	17,516	974	973	9,409	553
QMSum	1,257	272	279	9,070	70
SummScreen	18,915	1,795	1,793	6,421	381
arXiv	203,037	6,436	6,440	6,030	273
BookSum	314	45	46	143,301	1,294

Table 2: Statistics of datasets used in our experiments.

ory usage increase is quadratic to the length of the augmented content.

Key-value Vectors. To circumvent the quadratic memory increase, we join the key-value representations of tokens in important sentences in the encoder self-attentions, and directly inject them into the summarizer encoder. The memory increase is only linear to the augmented content's length.

Concretely, the summarizer encoder first encodes all document segments and obtains the representations (i.e., encoder outputs) of tokens belonging to the extracted important sentences. During training, the token representations of these sentences are concatenated with the key-value matrices in the encoder self-attentions while the query matrix remains in its original form. Up to 1024 tokens are concatenated via the same inclusion method for text concatenation, to prioritize the outermost sentences. A similar idea has been used by Memorizing Transformer (Wu et al., 2022a) to include retrieved text representations from past segments for long-form language modeling. Our method differs in two aspects. First, we extract representations from *future segments*, which are crucial for accurately identifying salient content. Second, we apply *a learnable projection* to the augmented representations prior to key-value concatenation. This process is crucial in improving compatibility with the original key-value matrices.

4 Experimental Setups

Datasets. We conduct experiments on GovReport (Huang et al., 2021), QMSum (Zhong et al., 2021), SummScreen (Chen et al., 2022), arXiv (Cohan et al., 2018), and BookSum (Kryscinski et al., 2022). The average input lengths of these datasets range from 6K to 143K (Table 2).

Experiment Setups and Comparisons. Our main experiments are conducted with a *GPU memory constraint of 27GB*. For each model, we truncate the input such that its maximum GPU memory usage during training does not exceed the constraint when gradient checkpointing (Chen et al., 2016) is *disabled*. The constraint is specifically chosen such that the baselines perform reasonably. Appendix C.2 provides information on the maximum number of input tokens that can conform to the constraint for other models.

For baselines, in addition to the divide-and-conquer **Se3** model (Moro and Ragazzi, 2022), we compare with state-of-the-art or popular long document summarization systems including **Block-Attn** (Phang et al., 2022),[6] **Longformer** (Beltagy et al., 2020), **LongT5** (Guo et al., 2022), and **Unlimiformer** (Bertsch et al., 2023). We also include an extract-then-abstract model (**Extract-Abstract**) and **PageSum** (Liu et al., 2022) that leverages dynamic weights, as discussed in §2. All models are initialized from `BART-large`, except for LongT5 that is pre-trained on long-form data. Details of baseline models are reported in Appendix D.

Evaluation Metrics. We evaluate summary ***informativeness*** using ROUGE (Lin, 2004). To measure ***coherence***, we use DiscoScore (Zhao et al., 2022) (**Disco**), a reference-based metric that evaluates discourse coherence by comparing focus (e.g., nouns) frequency and semantics between the system summary and the reference. We also report a graph-based reference-free coherence metric (Guinaudeau and Strube, 2013) (**Ent Graph**), which measures the connectivity of summary sentences linked by entities, reflecting the coherence of topic transitions. For summary ***faithfulness***, a recent model-based faithfulness metric, **SummaC** (Laban et al., 2022), is used.[7]

Finally, we show the maximum size of allocated **GPU memory** by each model during training.

5 Results

We report results by all AWESOME variants and comparison models on **GovReport** in Table 3. Compared with Se3, AWESOME variants *consistently achieve better performance* on both *ROUGE* and *coherence* scores, indicating the importance of maintaining global context for accurate salience estimation of local content and enforcing coherent transitions across segment-level summaries. This

[6]While Phang et al. (2022) introduce a new pre-trained model, we only incorporate their proposed block attentions into BART for a fair comparison.

[7]We only evaluate SummaC on GovReport, as the less formal formats or the domains of other datasets degrade the sentence-level NLI model of SummaC.

Model	R-1 ↑	R-2 ↑	R-L ↑	SummaC ↑	Disco ↓	Ent Graph ↑	GPU Mem ↓
Se3	46.56	23.22	44.36	14.71	7.37	1.41	11.1
BlockAttn	57.46	26.78	54.82	20.43	5.91	2.05	25.6
Longformer	57.40	26.92	54.70	20.39	5.68	2.05	25.3
LongT5	54.21	24.87	51.06	13.34	4.81	1.56	25.4
Unlimiformer	56.35	25.94	53.83	6.05	5.36	1.96	27.0
Extract-Abstract	56.89	24.76	54.26	**22.07**	4.03	**2.09**	13.2
PageSum	56.80	23.26	54.11	6.82	**3.04**	1.88	24.9
AWESOME *using External Memory Only*							
Compressive	56.30†	26.94†	53.77†	15.85	5.04†	2.04†	12.5
Attentive (Attn)	58.44*	27.71*	55.98*	18.98†	3.62†	1.98†	14.0
AWESOME *using Global Salient Content Only*							
Text-concat (Txt)	56.65†	27.68*	54.11†	12.23	5.05†	**2.09**†	12.0
Key-value Vectors	55.02†	26.39†	52.41†	11.52	4.75†	1.75†	14.3
AWESOME (Attn + Txt)	**58.76***	**28.18***	**56.05***	19.22†	3.86†	2.03†	14.8

Table 3: Results on GovReport. The best and second best results per metric are **bolded** and underlined. AWESOME with attentive memory only and its full version that additionally uses salient content through text concatenation obtain the highest ROUGE scores and are comparable or better on faithfulness (SummaC) and coherence (Disco & Ent Graph) than base model Se3. $*$: our model is better than all comparisons with approximation randomization test ($p < 0.0005$); †: our model is better than Se3 ($p < 0.0005$).

can also be demonstrated by the sample outputs in Table 4. Summaries generated by Se3 tend to introduce noun phrases without context, as Se3 fails to plan at a global level. On faithfulness, AWESOME with attentive memory improves SummaC over Se3, while *only* augmenting AWESOME with global salient content hurts faithfulness. Inspecting the model outputs, we find that using attentive memory improves understanding concepts of long-term dependencies, e.g., connecting a strategy with its related information that appears earlier in the report.

Of the two types of external memory mechanisms, *attentive memory outperforms compression-based memory on all metrics* except the entity graph, which highlights the advantage of adaptively updating the stored context. Meanwhile, *directly concatenating salient content with the input yields higher ROUGE scores* than injecting key-value vectors into the attention calculation, though the latter is less memory-intensive. We believe natural language-based augmentation better interleaves with the document segment, echoing the findings by prior work on using retrieval for question answering (Wu et al., 2022b).

Importantly, *under a strict GPU memory constraint,* AWESOME *with external memory mechanisms and global salient content augmentation achieves the best ROUGE scores* among all models, while obtaining competitive results on other measures. Though efficient attention models and PageSum can perform remarkably when given higher-capacity GPUs as in the original work, they generate less informative summaries when truncation is required to comply with the memory constraint, emphasizing the importance of studying memory-efficient long document summarization models. Furthermore, AWESOME only creates a small GPU memory overhead of less than 4GB, enhancing the model performance efficiently.

Se3: VA has taken a number of actions to address deficiencies GAO found in wait-time measurement and implementation of its scheduling policy. For wait-time measurement, these actions included changes to the wait-time measurement definitions, provision and documentation of scheduler training, and improved oversight through audits, all of which have been in a state of flux for the past 6 years. On July 12, 2019, VA provided GAO additional updates on efforts to implement **GAO's related recommendations**.

AWESOME: GAO **recommended** that VA either clarify its scheduling policy to better define the desired date, or identify clearer wait-time measures that are not subject to interpretation and prone to scheduler error. VA concurred with the **recommendation**, which GAO has identified as among those **recommendations** that warrant priority attention. VA has taken a number of actions to address **GAO's recommendations** regarding deficiencies GAO found in wait-time measurement and implementation of its scheduling policy.

Table 4: Summary snippets generated by Se3 and AWESOME. AWESOME's summary is more coherent, with natural transitions surrounding "**GAO's recommendation**", while Se3 abruptly introduces the topic.

On **QMSum** (Table 5), AWESOME with attention-based memory outperforms all comparisons on ROUGE scores. While our models' sum-

Model	R-1 ↑	R-2 ↑	R-L ↑	Disco ↓	GPU ↓
Se3	29.28	10.51	25.93	0.77	8.1
BlockAttn	30.76	8.26	26.49	0.50	22.8
Longformer	29.18	7.82	24.94	3.07	26.5
LongT5	31.88	10.07	27.82	0.44	25.4
Unlimiformer	30.57	8.82	26.89	0.49	26.9
Extract-Abstract	17.63	5.65	16.02	4.02	10.3
PageSum	29.55	7.38	26.11	**0.31**	21.5
AWESOME					
Attn Only	32.02†	**12.02**	28.16*	0.69	12.9
Attn + Txt	**32.05**†	10.53	**28.31**	0.63	13.3

Table 5: Results on meeting transcripts in QMSum. Equipped with attentive memory only, AWESOME achieves the better ROUGE scores than baselines. Adding extracted salient content does not further boost the performance, due to the low performance of the extractor on dialog data.

Model	R-1 ↑	R-2 ↑	R-L ↑	Ent G ↑	GPU ↓
Se3	38.09	11.30	36.56	0.50	11.3
BlockAttn	32.01	8.99	30.90	**1.61**	25.7
Longformer	42.78	**13.21**	41.34	0.97	25.3
LongT5	42.03	12.67	40.76	1.03	25.4
Unlimiformer	35.17	11.98	34.28	1.33	27.0
Extract-Abstract	19.95	5.58	19.70	0.06	13.1
AWESOME					
Attn Only	**46.05**†	13.09†	**44.21**†	0.81†	13.2
Attn + Txt	45.30†	12.63†	43.51†	0.90†	14.2

Table 6: Results on TV transcripts in SummScreen. We report Ent Graph instead of DiscoScore, as DiscoScore encounters errors when identifying focus. AWESOME with the attentive memory obtains the best R1 and RL scores, while the low accuracy of the extracted salient content leads to performance drop of the summarizer.

maries are more coherent than the summaries by Se3, as measured by DiscoScore, the differences among all models are less pronounced compared to the ones on GovReport. This is because QMSum contains shorter summaries than GovReport (69 vs. 553), thus involving fewer topic transitions. We also find that the extractor performs poorly on QMSum, leading to degraded ROUGE-2 result after augmenting our model with the extracted salient content. Specifically, the F1 score of the extractor on the test set is only 1.29, as opposed to 27.85 on GovReport. This trend is similarly observed on **SummScreen** (Table 6), where the extract-then-abstract method performs poorly and adding extracted content leads to performance drop of AWESOME due to the low performance of the extractor. Meanwhile, AWESOME with the attentive memory is able to obtain the best ROUGE-1 and ROUGE-L scores.

Model	R-1 ↑	R-2 ↑	R-L ↑	Disco ↓	GPU ↓
Se3	40.74	17.96	36.87	1.33	12.8
BlockAttn	**49.12**	**21.69**	**44.40**	1.77	25.7
Longformer	48.59	21.45	43.99	2.17	25.2
LongT5	48.25	20.74	43.41	0.97	25.5
Unlimiformer	47.78	20.58	43.22	1.22	26.8
Extract-Abstract	42.37	16.43	38.62	1.03	15.3
PageSum	46.01	18.77	41.55	**0.88**	26.2
AWESOME					
Attn Only	42.51†	18.96†	38.56†	1.30	16.0
Attn + Txt	44.20†	18.89†	40.07†	1.32	16.5

Table 7: Results on arXiv papers. AWESOME variants again outperform Se3. For 80% of the arXiv documents, efficient attention models and PageSum can fully train on their first halves, covering 90% of the salient content that appear in the references (Huang et al., 2021), thus the better ROUGE scores than models encoding smaller segments.

Model	R-1 ↑	R-2 ↑	R-L ↑	Disco ↓	GPU ↓
Se3	40.78	10.16	39.77	10.46	11.5
BlockAttn	23.45	3.09	22.09	190.27	25.7
Longformer	20.20	2.45	18.55	204.48	25.3
LongT5	33.15	6.74	32.62	24.24	25.5
Unlimiformer	38.09	9.55	37.41	47.72	27.0
AWESOME (Attn)	**41.11**	**10.63**	**40.20**	**10.36**	24.0

Table 8: Results on novels in BookSum. AWESOME with attentive memory in all layers achieves the best performance on all metrics. Methods requiring external extractors are not included due to the computational cost of building extractive oracles for long novels.

On arXiv, models that use efficient attentions obtain the higher ROUGE scores, because truncating arXiv documents has little effect on summary generation—arXiv articles have the most **uneven** distributions of salient content, where only about 10% of new salient bigrams are located in the second halves of the documents (Huang et al., 2021).

Finally, experiments on **BookSum** show that the divide-and-conquer method produces better summaries for long novels, while our method can further boost its performance (Table 8). However, we find it necessary to incorporate external memory into all layers, suggesting a more complex interaction of external memory with the summarization process for novel plots. Unlike other document types tested, novel plots are typically sequential with less redundancy, which reduces the necessity of the memory mechanism.

Among all datasets, global salient content augmentation performs better on datasets rich in knowledge-dense factual statements, such as Gov-

Report and arXiv. We observe that the majority of sentences extracted from GovReport and arXiv are standalone statements comprehensible with little to no contextual support. By contrast, sentences extracted from other datasets typically demand integration with their original context for full clarity. Our global salient content augmentation mechanism leverages these extracted sentences without their own context, which reduces its effectiveness on datasets such as SummScreen and QMSum. We will explore methods that allow contextualization of global salient content in future work.

More experimental findings are presented in Appendix B. Notably, with the same input length, AWESOME still achieve competitive performance (Table 11), despite using less GPU memory and running faster than most models (Figure 3).

Human Evaluation. We ask three fluent English speakers that have extensive NLP data annotation experience to examine the outputs by BlockAttn, Se3, and AWESOME using attentive memory and text concatenation for content augmentation on GovReport. 25 GovReport documents are randomly selected,[8] each summarized by all three systems. Outputs from different systems are randomly shuffled and displayed. For **each summary sentence**, the annotators give binary labels on where the sentence is **coherent**—it uses natural transitions to logically connect with the previous content, and does not contradict any prior statement. The annotators also compare each summary sentence with the document and check if it is **faithful**, i.e., it can be verified and entailed from the document. We further ask the annotators to rank the summaries generated by the three systems based on their **informativeness**—how well the summary captures the salient content of the document.

As seen in Figure 2, AWESOME's summaries are rated by human judges to be more coherent, faithful, and informative than Se3, again evidencing the importance of incorporating global context. Though BlockAttn produces the most coherent summaries, it has more faithfulness errors and is less informative, due to document truncation under a constrained memory.

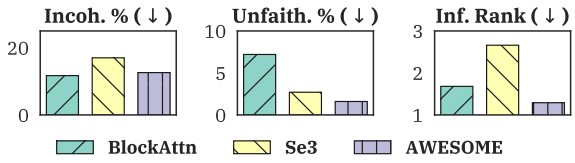

Figure 2: Percentages of system summary sentences that are incoherent (**Incoh.**) and unfaithful (**Unfaith.**) on GovReport, as rated by human. The average rankings of the informativeness (**Inf. Rank**) of system outputs are also reported. Though summaries by BlockAttn are more coherent, AWESOME generates more faithful and informative outputs. Krippendorff's α: 0.50/0.49/0.57.

6 Conclusion

We present AWESOME for summarizing long documents in a memory-constrained setting. Based on the divide-and-conquer strategy, AWESOME uses two mechanisms to gather global context and improve summary quality. First, external memories on the encoder and decoder are employed to track previously read document content and the corresponding summaries. Second, the encoder is informed of global salient content predicted by an extractor via text or representation concatenation. On five summarization datasets, AWESOME generates summaries with better informativeness, faithfulness, and coherence than a baseline divide-and-conquer system. Under the same memory constraint, AWESOME outperforms competitive models that leverage efficient attentions or dynamic extraction to preserve global context, highlighting its effectiveness in supplying global context.

Acknowledgments

This work is supported in part by the National Science Foundation through grant IIS-2046016. Shuyang Cao is supported by a Bloomberg Data Science Ph.D. Fellowship. We thank all reviewers for their useful feedback.

Limitations

AWESOME's external memory mechanism is restricted to operating solely from past segments to the current segment. This means that the model does not leverage the information contained in future segments, which can be relevant for a comprehensive understanding of the current segment. To address this limitation, we have designed the global salient content augmentation mechanism to cover context from the future segments, yet more advanced solutions can be explored in future work. For example, on the encoder, making the external memory bidirectional is a potential approach.

[8] We focus on GovReport, as its documents are well formatted and easier for annotators to follow.

While being memory-efficient, the external memory mechanism of AWESOME necessitates a longer running time due to its recurrent nature. The need for recurrent computations may lead to increased processing requirements, which could impact real-time applications or scenarios where rapid responses are crucial. The running times of different models are provided in Appendix B.4 for reference. Although our model is slower than that of LongT5 and Se3, it still outperforms several other competitive models in terms of speed, and we will investigate methods for reducing the running time in future work.

The scope of our human evaluation is limited due to practical considerations. Expanding the scale of our human assessment would be highly time-intensive, given the need for referring to long documents. This limitation is a common challenge encountered in annotating lengthy texts, and many long document summarization studies opt to only include automatic evaluations. However, we recognize that increasing the number of samples in our human evaluation could provide stronger empirical support for the efficacy of our approach.

Ethical Considerations

We anticipate that one of the major use cases of AWESOME is to allow ordinary users who have computing devices with limited memory to quickly understand government policies and other types of long documents. However, we recognize that the system generated summaries might not comprehensively cover the salient content that is essential for correctly understanding the policies, causing risks ranging from capital loss to legal liability. Moreover, system summaries might contain statements that cannot be verified through the document, which further adds to the risks of real-world deployment. We suggest developers who intend to use our model for real-world application carefully study the outputs by our model before the actual deployment.

References

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

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. 2023. Unlimiformer: Long-range transformers with unlimited length input.

Aydar Bulatov, Yuri Kuratov, and Mikhail Burtsev. 2022. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*.

Mingda Chen, Zewei Chu, Sam Wiseman, and Kevin Gimpel. 2022. SummScreen: A dataset for abstractive screenplay summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8602–8615, Dublin, Ireland. Association for Computational Linguistics.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*.

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.

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.

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.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 93–103, Sofia, Bulgaria. 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.

Luyang Huang, Shuyang Cao, Nikolaus 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, pages 1419–1436, Online. Association for Computational Linguistics.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Wojciech Kryscinski, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2022. BOOKSUM: A collection of datasets for long-form narrative summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6536–6558, Abu Dhabi, United Arab Emirates. 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.

Jie Lei, Liwei Wang, Yelong Shen, Dong Yu, Tamara Berg, and Mohit Bansal. 2020. MART: Memory-augmented recurrent transformer for coherent video paragraph captioning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2603–2614, Online. 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.

Chuan Li. 2022. Best gpu for deep learning in 2022 (so far).

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.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. 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. *arXiv preprint arXiv:1907.11692*.

Yixin Liu, Ansong Ni, Linyong Nan, Budhaditya Deb, Chenguang Zhu, Ahmed H. Awadallah, and Dragomir Radev. 2022. Leveraging locality in abstractive text summarization.

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 documents in low-resource regimes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11085–11093.

OpenAI. 2023. Gpt-4 technical report.

Maxime Peyrard, Teresa Botschen, and Iryna Gurevych. 2017. Learning to score system summaries for better content selection evaluation. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 74–84, Copenhagen, Denmark. Association for Computational Linguistics.

Jason Phang, Yao Zhao, and Peter J Liu. 2022. Investigating efficiently extending transformers for long input summarization. *arXiv preprint arXiv:2208.04347*.

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.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

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.

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.

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

Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, and Christian Szegedy. 2022a. Memorizing transformers. In *International Conference on Learning Representations*.

Yuxiang Wu, Yu Zhao, Baotian Hu, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2022b. An efficient memory-augmented transformer for knowledge-intensive nlp tasks.

Wen Xiao and Giuseppe Carenini. 2020. Systematically exploring redundancy reduction in summarizing long documents. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 516–528, Suzhou, China. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2021. Big bird: Transformers for longer sequences.

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.

Wei Zhao, Michael Strube, and Steffen Eger. 2022. Discoscore: Evaluating text generation with bert and discourse coherence.

Yao Zhao, Mohammad Saleh, and Peter J Liu. 2020. Seal: Segment-wise extractive-abstractive long-form text summarization. *arXiv preprint arXiv:2006.10213*.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. 2021. 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, Online. Association for Computational Linguistics.

A Divide-and-Conquer Architecture

We choose Se3 (Moro and Ragazzi, 2022) as our base divide-and-conquer architecture because it can be applied to any document-summary pair. In order to create divide-and-conquer training data for summarization, for each document-summary pair, the document is first divided into segments (§A.1) and each summary sentence is then assigned to a document segment as part of the generation target (§A.2).

A.1 Document Segmentation

The length of each document segment is between 512 and 768 tokens. During segmentation, the algorithm loops through all document sentences, as shown in Algorithm 1. A document sentence will be added to the current segment if the segment contains less than 512 tokens. The current segment will be finalized if the current segment contains more than 768 tokens or the current sentence is more semantically similar to the next pseudo segment than the current segment, where the next pseudo segment is created by including future sentences until reaching 512 tokens. To measure the similarity between the current sentence and a segment, we use the average cosine similarity between the representation of the current sentence and representations of the sentences in the segment. Sentence representations are obtained using Sentence Transformer (Reimers and Gurevych, 2019) with the `all-roberta-large-v1` model.

A.2 Target Assignment

For each sentence in the reference summary, we calculate its ROUGE scores with the document segments. The sentence will then be assigned to the document segment with which yields the highest ROUGE-1 and ROUGE-2 scores.

B Additional Results

B.1 Entity Graph Results

We show the entity graph scores on datasets other than GovReport in Table 9.

B.2 Effects of Encoder Memory and Decoder Memory

We conduct ablation studies on the usage of the encoder and decoder memories used by AWESOME. As shown in Table 10, taking out the external memory from the decoder significantly affects summary

Algorithm 1: Document Segmentation

Data: Input document doc; Segment min, max length l_{min}, l_{max}

```
segs ← [];
currSeg ← [];
foreach sent in doc do
    if len(currSeg) < l_min then
        currSeg ← currSeg + [sent];
    end
    else if len(currSeg) > l_max then
        segs ← segs + [currSeg];
        currSeg ← [sent];
    end
    else
        nextSeg ← pseudoSegment;
        if sim(nextSeg, sent) >
          sim(currSeg, sent) then
            segs ← segs + [currSeg];
            currSeg ← [sent];
        end
        else
            currSeg ← currSeg + [sent]
        end
    end
end
segs ← segs + [currSeg];
return segs
```

Model	QMSum	arXiv	BookSum
Se3	0.47	0.73	1.42
BlockAttn	0.56	0.92	2.64
Longformer	0.49	0.96	2.90
LongT5	0.30	0.76	1.98
Unlimiformer	0.59	0.94	2.69
Extract-Abstract	1.07	1.11	-
PageSum	0.54	0.91	-
AWESOME			
Attn Only	0.47	0.80	1.33
Attn + Txt	0.56	0.99	-

Table 9: Results of the entity graph metric on experimented datasets.

Model	R-2 ↑	R-L ↑	SC ↑	Disco ↓	GPU ↓
AWESOME (Attn)	**27.71**	**55.98**	**18.98**	**3.62**	14.0
w/o Dec Mem	27.63	54.60	12.99	4.50	13.0
w/o Dec & Enc Mem	23.22	44.36	14.71	7.37	11.1

Table 10: Effects of encoder and decoder memories on AWESOME on GovReport. SC: SummaC. Both types of memories contribute to the summary coherence, while the decoder memory is more important for faithfulness and the encoder memory advances the summary informativeness more significantly.

faithfulness and coherence, while the informativeness measures remain comparable. This indicates that the information from past summary segments tracked by the decoder memory is crucial for producing coherent transitions. Furthermore, the decoder memory mechanism may also store information relevant to the key topics or entities, which promotes the understanding of their mentions in the current summary segment and boosts faithfulness. The encoder memory allows comparing content in the current segment versus its past context, which is crucial for salience estimation. Therefore, removing encoder memory results in a more significant drop in ROUGE scores.

B.3 Performance w/ the Constrained Input Length

Besides experiments with constrained GPU memory, we also examine model performance when training with the same input length (16384 tokens) on GovReport. Gradient checkpointing is allowed when the model is using more than 48GB of GPU memory. Results are reported in Table 11. With shorter training data, AWESOME remains competitive on summary informativeness and coherence, while maintaining a low GPU memory usage.

Model	R-1 ↑	R-2 ↑	R-L ↑	Ent Prec ↑	SummaC ↑	Disco ↓	Ent Graph ↑	GPU Mem ↓
BlockAttn	57.69	26.92	55.00	**97.86**	**23.98**	4.61	**2.06**	48.0+
Longformer	57.61	26.88	54.93	97.80	23.42	5.95	2.05	48.0+
LongT5	55.23	25.54	52.57	96.57	17.52	4.49	1.76	48.0+
PageSum	**59.36**	26.59	**56.44**	88.87	2.25	**2.43**	1.87	48.0+
AWESOME	58.69	**28.07**	55.99	97.76	19.42	3.82	2.01	14.8

Table 11: Results on GovReport when models are trained with up to 16384 tokens. The best and second best results per metric are **bolded** and underlined. When more than 48GB of GPU memory is required, gradient checkpointing is enabled. AWESOME achieves comparable informativeness and coherence, while using much less GPU memory.

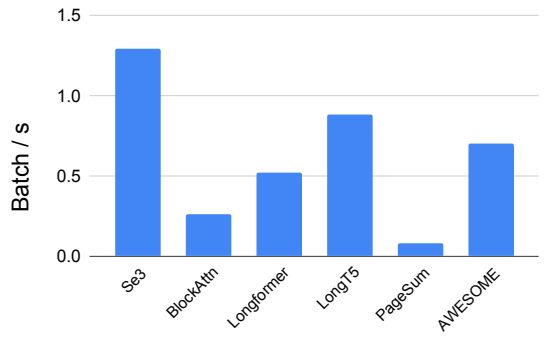

Figure 3: Running time (batch per second) of each model. A higher number of batches processed per second indicates a faster running speed. All models use a batch size of 1 and the input is truncated to 16384 tokens.

Model	Unigram	Bigram	Trigram
Se3	38.56	79.34	91.35
BlockAttn	38.10	77.48	89.61
Longformer	36.79	76.56	88.51
LongT5	44.21	85.10	95.71
Unlimiformer	36.29	75.09	89.82
Extract-Abstract	34.80	76.26	91.24
PageSum	40.44	86.51	99.04
AWESOME			
Attn Only	38.06	78.76	90.98
Attn + Txt	37.19	77.97	90.39

Table 12: Unique n-gram ratios of summaries generated by different models on GovReport. Redundancy of summaries generated by AWESOME is comparable to that of other models.

B.4 Running Time

We compare the model running time on GovReport (Figure 3). The input document is truncated to 16384 tokens and each model is separately train for 1000 steps with a batch size of 1. No other computation-heavy program is running at the same time. While AWESOME take longer time to complete training than Se3, it is still the third fastest model.

B.5 Redundancy Evaluation

We measure the unique n-gram ratios of summaries generated by different models on GovReport. The unique n-gram ratio is calculated by dividing the count of unique n-grams by the total number of n-grams (Peyrard et al., 2017; Xiao and Carenini, 2020), and a lower unique n-gram ratio indicates a higher level of redundancy. As shown in Table 12, the redundancy of summaries generated by our models is comparable to that of other comparisons. While LongT5 and PageSum have the lowest redundancy, they have lower informativeness and faithfulness as rated by other metrics.

B.6 Abstractiveness Evaluation

We measure the density of model outputs on GovReport. The density quantifies how well the word sequence of a summary can be described as a series of extractions (Grusky et al., 2018). A lower density indicates a higher abstractivenss. Compared to Se3 (Table 13), summaries generated by our models are more extractive, while they are more abstractive than summaries generated by BlockAttn and Longformer.

C Dataset Details

C.1 Statistics

We conduct experiments on five long document summarization datasets with diverse genres. **GovReport** (Huang et al., 2021) contains long reports and their summaries written by government research agencies. **QMSum** (Zhong et al., 2021) is a query-focused long meeting transcript summarization dataset, with summary-worthy content spread over the documents. We prepend the query to all segments. We further use a screenplay summarization dataset, **SummScreen** (Chen et al., 2022), which contains the transcripts of TV series. The TMS subset, with more samples and

Model	Density
Se3	65.94
BlockAttn	125.46
Longformer	113.97
LongT5	67.02
Unlimiformer	42.06
Extract-Abstract	32.88
PageSum	28.40
AWESOME	
Attn Only	82.18
Attn + Txt	88.67

Table 13: Densities of summaries generated by different models on GovReport. AWESOME produces summaries with higher abstractiveness than BlockAttn and Longformer.

Model	Dataset				
	Gov	arXiv	QMSum	SumScrn	Book
Se3	50x	50x	50x	50x	50x
Ext-Abs †	1x (∞)	1x (∞)	1x (∞)	1x (∞)	-
BlockAttn	6x	6x	8x	6x	6x
Longformer	8x	8x	8x	8x	8x
LongT5	6x	6x	6x	6x	6x
Unlimiformer	2x	2x	2x	2x	2x
PageSum	3x	5x	2x	-	-
AWESOME	50x	50x	50x	50x	50x

Table 14: Truncation thresholds (multiply by 1024) used by each model on different datasets to comply with the memory constraint during training. †: For the extract-then-abstract model, the abstractor has a maximum input length of 1024, while the extractor can consume all sentences in the document.

longer summaries, is selected. Moreover, we experiment with the scientific papers and their abstracts from **arXiv** (Cohan et al., 2018). Finally, we test our models on summarizing *full* novels in **BookSum** (Kryscinski et al., 2022). For all datasets, we use the official train/dev/test splits if their original data files are released.

For GovReport[9], QMSum[10], and SummScreen (Chen et al., 2022), we use the data released by the original papers. For arXiv, we use the version provided by Huggingface Datasets.[11] As the original data files for BookSum are not released due to summary copyright, we use the version reproduced by Unlimiformer (Bertsch et al., 2023).

C.2 Input Truncation

In our main experiments, we employ a GPU memory constraint of 27GB. As some baseline models require the input length to be a multiplier of 1024, setting a constraint of 24GB, a more common number, would lead to further truncation and significant performance drop.

To fit models into our memory constraint, we truncate the model inputs. The truncation thresholds used by each model on different datasets are shown in Table 14. Although Se3 and AWESOME theoretically maintain a consistent GPU memory consumption during training regardless of the number of input tokens processed, we have chosen to restrict the maximum number of input tokens in a training sample to 51200 for reasonable training time.

D Implementation Details

Baselines. BlockAttn and Longformer use block-wise attentions (Phang et al., 2022) and sliding-window attentions (Beltagy et al., 2020), where a global token can attend to and be attended by all tokens, while other tokens can only attend to tokens in the same block or window. LongT5 (Guo et al., 2022) is a sliding-window attention model pre-trained on long sequences, and Unlimiformer (Bertsch et al., 2023) extends BART by selecting input tokens to be attended to via KNN searching. For the extract-then-abstract approach, we use the same extractor as in the global salient content augmentation of our model, and the abstractor takes as input oracle extracted sentences during training. Lastly, PageSum (Liu et al., 2022) synthesizes the output representations given by different document segments with dynamic weights.

Extractor. The extractor first uses a RoBERTa (Liu et al., 2019) to encode each sentence and takes the average of the final layer's outputs as the sentence representation. It then applies a self-attention on top of all sentence representations. The resulting representations are converted to extraction scores after applying a multi-layer perception with one hidden layer. The extractor is trained with **oracle extractive labels** that are constructed by greedily searching for document sentences that maximize the sum of ROUGE-1 and ROUGE-2 scores, compared against the reference summary. We do not compute ROUGE-L as in DYLE (Mao et al., 2022), because finding the longest common subsequence is computationally expensive and does not yield performance gain.

[9] https://gov-report-data.github.io/

[10] https://github.com/Yale-LILY/QMSum

[11] https://huggingface.co/datasets/scientific_papers

Training Parameters. We train all models with a maximum learning rate of 5×10^{-5}, except that LongT5 is trained with a maximum learning rate of 1×10^{-4}. We use a running batch size of 1 and apply gradient accumulation to achieve an effective batch size of 8. The numbers of training epochs are 3, 9, 6, 2, 10 on GovReport, QMSum, SummScreen, arXiv, and BookSum, with warmup steps of 300, 100, 300, 1000, and 40. Due to the computational cost of training long document summarization, each model is trained for a single run.

Model Size. AWESOME is based on `BART-large`[12] and has 708 millions of parameters.

Computing Infrastructure. All experiments are conducted on RTX A6000 GPUs.

Evaluation Metrics. For ROUGE (Lin, 2004), we use the Python implementation by Google.[13] The official code for DiscoScore (Zhao et al., 2022) is used[14], which also provides an implementation of the Ent Graph metric (Guinaudeau and Strube, 2013). We implement the entity precision measure ourselves and run the official code for SummaC (Laban et al., 2022).[15] All metrics used are open-source and can be distributed for research purposes.

Usage of AI Assistants. The authors use Copilot to assist coding. ChatGPT is used to fix grammatical errors during writing.

E Human Evaluation Details

The three annotators in our human evaluation are all US college students and they have taken undergraduate-level or graduate-level natural language processing courses. Before starting the annotation, the goal of the annotation is explained and the instruction is presented to the annotators. The annotators are fairly compensated ($12/hr). Figure 4 shows the detailed instruction for evaluation on GovReport.

[12] https://huggingface.co/facebook/bart-large
[13] https://pypi.org/project/rouge-score/
[14] https://github.com/AIPHES/DiscoScore
[15] https://github.com/tingofurro/summac

Instructions

During this task, you will be given a report and three different summaries for the report. Additionally, the reference summary will be provided for comparison. You will evaluate the quality of each of the three summaries by **three** axes: *coherence*, *faithfulness*, and *informativeness*.

For coherence and faithfulness, you are asked to provide binary 0/1 labels for each sentence. For informativeness, you also need to rank the six summaries from 1 (best) to 3 (worse). Ties are allowed.

Coherence

For each summary sentence, please indicate whether the sentence is coherent. You should consider both local and global coherence.

Local: the sentence is logically connected with the previous sentence (i.e., no abrupt change of subject and show clear discourse relation) and language-wise using natural transitions.

Global: the sentence does not contradict all prior statements and discourse cues. Examples of globally incoherent sentences include:

- Contradicting statements, such as arguing against government intervention in the market but then advocating for it.
- Disrupting the established order of discussion, such as discussing topic Z before topics X and Y, while the order has been set to X, Y, Z by a prior sentence.
- Omitting a point that should be discussed per a prior sentence.

Faithfulness

For each summary sentence, please indicate whether the sentence is faithful to the report. A sentence is faithful if its content can be verified by the report. You can also compare the sentence with the reference summary to determine whether the sentence is faithful. Here are some examples of faithfulness errors:

- Attributing an abbreviation or acronym to the wrong full name, or vice versa.
- Describing an action or policy as being completed or established when it has not, or describing it as not having been completed or established when it has.
- Making a statement that does not appear in the document.

Informativeness

Compare the system summaries against the reference, and rank summaries from most informative to least informative.

- If the summaries contain information outside the reference, it doesn't count.
- A summary can be long but only contain information not covered in the reference, and won't be ranked higher

Figure 4: Human evaluation instruction for evaluation on GovReport.