SUMM^N: A Multi-Stage Summarization Framework for Long Input Dialogues and Documents

Yusen Zhang[♣] **Ansong Ni**[†] **Ziming Mao**[†] **Chen Henry Wu**[‡]
Chenguang Zhu[◇] **Budhaditya Deb**[◇] **Ahmed H. Awadallah**[◇]
Dragomir Radev[†] **Rui Zhang**[♣]
[♣] Penn State University [†] Yale University
[‡] Carnegie Mellon University [◇] Microsoft Research
{yfz5488,rmz5227}@psu.edu, {ansong.ni,dragomir.radev}@yale.edu

Abstract

Text summarization helps readers capture salient information from documents, news, interviews, and meetings. However, most state-of-the-art pretrained language models (LM) are unable to efficiently process long text for many summarization tasks. In this paper, we propose SUMM^N, a simple, flexible, and effective multi-stage framework for input texts that are longer than the maximum context length of typical pretrained LMs. SUMM^N first splits the data samples and generates a coarse summary in multiple stages and then produces the final fine-grained summary based on it. Our framework can process input text of arbitrary length by adjusting the number of stages, while keeping the LM input size fixed. Moreover, it can deal with both single-source documents and dialogues, and it can be used on top of different backbone abstractive summarization models. To the best of our knowledge, SUMM^N is the first multi-stage split-then-summarize framework for long input summarization. Our experiments demonstrate that SUMM^N outperforms previous state-of-the-art methods by improving ROUGE scores on three long meeting summarization datasets AMI, ICSI, and QMSum, two long TV series datasets from SummScreen, and a long document summarization dataset GovReport. Our data and code are available at https://github.com/psunlpgroup/Summ-N.

1 Introduction

Abstractive summarization helps readers capture salient information from various sources such as documents, news, interviews, and meetings. Previous work has primarily focused on short texts of news (Gehrmann et al., 2018; Zhang et al., 2019) and short conversations (Gliwa et al., 2019; Chen and Yang, 2021). Recently proposed longer dialogue and document summarization tasks (Zhong et al., 2021b; Huang et al., 2021; Chen et al., 2021; Zhu et al., 2021a) pose challenges for current large pretrained language models due to the time and memory complexity of training, as well as limited input lengths these models can consume.

A common method to handle long text reduces the input to a shorter one. This can be accomplished by truncating inputs (Lewis et al., 2020) or employing retrieve-then-summarize pipelines (Zhong et al., 2021b). However, these methods break the dependency of the context and decrease the number of tokens that the model can read, i.e., the receptive field of the model. The cutting-off model depends on the lead bias of the source text, while the retrieve-then-summarize models heavily rely on the independence of retrieved units (turns or sentences) which are usually scattered throughout the source text.

Another approach optimizes the attention mechanism in Transformers to accommodate longer inputs by reducing the impact of quadratic complexity of the attention process using Locality-sensitive hashing (LSH) attention (Kitaev et al., 2020) and Sinkhorn attention (Tay et al., 2020). Additionally, HMNet (Zhu et al., 2020) and HAT-BART (Rohde et al., 2021) use hierarchical self-attention to extend the input limitation of typical self-attention models. However, the simplified attention mechanism weakens the power of pretrained Transformer models, e.g., HMNet is not pretrained on external large-scaled unsupervised datasets as BART did.

In this paper, we propose SUMM^N, a multi-stage framework for long dialogue and document summarization. Figure 1 shows the structure of SUMM^N. First, it divides each source text into segments so that each can be completely fed into the backbone abstractive summarization model. Then, it matches each of them with the subset of target text using a ROUGE-based greedy algorithm. Next, each stage generates a coarse summary for each segment and concatenates them together as the input to the

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 1592 - 1604
May 22-27, 2022 ©2022 Association for Computational Linguistics

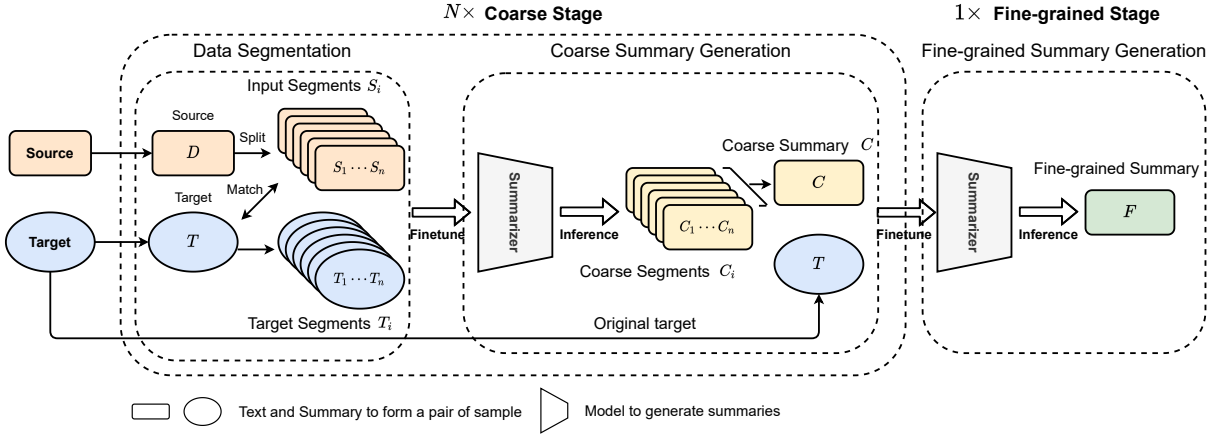

Figure 1: Workflow of the proposed SUMM^N framework. It contains N coarse stages and 1 fine-grained stage. At each coarse stage, source and target text is segmented and paired using a ROUGE-based greedy algorithm, and then a backbone summarization model is used to generate the summary for each segment. After multiple coarse stages, the last fine-graded stage produces the final summary output.

next stage. After multiple stages of compression and summarization, the final stage produces a fine-grained summary. The process expands the model context to the full reception field, meaning that the proposed model can read the full input no matter how long the input is. Additionally, retrieve-then-summarize pipelines (Zhang et al., 2019) extract sentences individually, leading to the loss of the context information for understanding utterances. By contrast, SUMM^N only cuts the source text at the end of each segment, so that the context of most sentences are retained.

It does not assume lead bias because each part of the source is fully used. In addition, in each stage, it leverages a backbone abstractive summarization model to recursively generate the summaries. Therefore, it enjoys the full power of the pretrained language models because the framework preserves the intact structure of Transformers.

SUMM^N is flexible to inputs with different lengths by adjusting the number of stages. SUMM^N can change the number of coarse stages according to the compression ratio between source and target, the input limit of the backbone model, and the input source length. We give the empirical formula to decide the number of needed stages for every tested dataset. Our experiments show that ROUGE increases on all datasets when increasing the number of stages from one to the appropriate number. Additionally, SUMM^N is flexible because it can be applied to different backbone summarization models. For example, we found that the ROUGE scores increase sharply on the AMI dataset when replacing the backbone BART model with T5 (Raffel et al., 2020) and PEGASUS (Zhang et al., 2019).

We conduct extensive experiments on long-input summarization datasets in multiple domains. The results demonstrate that the proposed model significantly outperforms previous state-of-the-art methods according to automatic and human evaluations on three long meeting summarization datasets (AMI, ICSI, QMSum) and one long TV series summarization dataset (SummScreen). It also achieves state-of-the-art performance on a long document summarization dataset (GovReport). These datasets include document summarization as well as both query-based and query-independent long dialogue summarization tasks.

Our contributions are: (1) We propose SUMM^N, a simple, flexible, and effective framework for long dialogue and document summarization. To the best of our knowledge, SUMM^N is the first multi-stage split-then-summarize framework to solve long text summarization tasks. (2) We evaluate SUMM^N on both dialogue and document domains and improve the baseline model by a large margin. (3) We analyze and compare the proposed framework with baselines and discuss its merits in detail.

2 Related Work

Long Document Summarization Long document summarization has been studied in multiple domains, such as news (Liu et al., 2021; Zhu et al., 2021b), patterns (Trappey et al., 2009), books (Kryściński et al., 2021; Wu et al., 2021), scientific publications (Qazvinian and Radev, 2008; Mao et al., 2021), and medical records (Cohan

et al., 2018). Gidiotis and Tsoumakas (2020) proposed a divide-and-conquer method by splitting the input into multiple segments, summarizing them separately, and combining the summary pieces. Grail et al. (2021) proposed a hierarchical neural model to process segmented input blocks. Compared with SUMM^N, these models only split the input once, implying the lack of flexibility when handling longer input.

The GovReport dataset was recently introduced containing documents with more than 9000 words, thus greatly challenging the capabilities of current models such as PEGASUS (Zhang et al., 2019), TLM (Pilault et al., 2020), and BIGBIRD (Zaheer et al., 2020). To handle this dataset, Huang et al. (2021) proposed head-wise positional strides to reduce the cost of the encoder-decoder attention. Similarly, models such as Longformer (Beltagy et al., 2020) and Reformer (Kitaev et al., 2020) adjust attention mechanisms in Transformers to consume longer inputs. However, these models sparsify the attention structure of the pretrained model to fit the longer source text. By contrast, SUMM^N is able to maintain the full structure of various pretrained models.

Long Dialogue Summarization Various models have also been proposed to handle long dialogue summarization. HMNet (Zhu et al., 2020) and HAT-BART (Rohde et al., 2021) leverage a two-level transformer-based model to obtain word level and sentence level representations. DialLM (Zhong et al., 2021a), Longformer-BART-arg (Fabbri et al., 2021) use finetuning or data augmentation to incorporate the external knowledge to maintain the accuracy of lengthy input. Different from these models, SUMM^N is a framework without modifying the structure of the backbone attention model.

Multi-Stage Text Generation Multiple multi-stage coarse-to-fine frameworks have been studied in many other text generation tasks, such as dialogue state tracking (Chen et al., 2020), neural story generation (Fan et al., 2018), and extractive summarization (Xu and Lapata, 2020). In a summarization task, a two-stage extract-and-summarize pipeline is commonly used (Zhang et al., 2019; Pilault et al., 2020; Zhao et al., 2020). However, unlike that work, our framework aims at long input summarization with fully abstractive intermediate summaries, meaning that SUMM^N can be viewed as a summarize-then-summarize pipeline.

3 Method

Figure 1 shows the workflow of SUMM^N. The workflow includes two types of stages, N coarse stages, and one fine-grained stage. Coarse stages include the data segmentation and coarse summary generation, while the fine-grained stage directly generates the summary as the final result. Besides, we have $N + 1$ separate models for each stage and each was separately trained. Our experiments show that the performance drops if different stages share the parameters (Section 4.2). SUMM^N can adjust and compute the number of coarse stages N according to the stats of dataset and model.

To formulate our task, we denote one sample of the source text as $D = \{D_1, D_2, \cdots, D_m\}$, where D_i indicates one sentence in a document or one turn in a dialogue. For query-based summarization, there is also a query Q. The goal is to generate a summary T, given D and the optional Q.

3.1 Data Segmentation

In long text summarization, the number of tokens in the source data usually exceeds the limit of the backbone summarization models, thus reducing the quality of the summary. To make sure that the model can capture information about all source tokens, we apply a segmentation algorithm for long input summarization datasets. First, we segment the source text so that the data input to the backbone model does not exceed the length limit. Then, we apply a greedy algorithm to find the best target summary that matches the source segments.

Source Segmentation Assume that the number of the maximum input tokens of the backbone model is K. To completely input the source information, we cut the input D (between sentences) into multiple segments, each of them containing fewer than K tokens. Given the input D, we will have n segments $S = \{S_1, S_2, \cdots, S_n\}$ where $S_i \in D$ is a segment in D. For query-based summarization tasks, we simply concatenate the query to the beginning of the S, i.e. $S_i \leftarrow Q \bigoplus S_i$. In both cases, the number of tokens in each segment is less than the hyper-parameter K.

Target Matching Segmenting the source text results in n source pieces S_i. We match each S_i with a target segment $T_i \in T$ to form the new pair (S_i, T_i) for the next step. We use a greedy algorithm for target matching. We first split T into separate sentences $T_s = \{T_{s_1}, T_{s_2}, \cdots, T_{s_k}\}$.

Algorithm 1 Greedy Target Matching

Input: $S_i, T_s = \{T_{s_1}, T_{s_2}, \cdots, T_{s_k}\}$
Output: (S_i, T_i)
  $T_i \leftarrow \Phi$
  loop
    $T_i' \leftarrow T_i$
    for $T_s' \in T_s - T_i$ **do**
      $\tau' \leftarrow \text{ROUGE}_1(S_i, T_i')$
      $\tau \leftarrow \text{ROUGE}_1(S_i, T_i \bigoplus T_s')$
      if $\tau' < \tau$ **then**
        $T_i' \leftarrow T_i \bigoplus T_s'$
      end if
    end for
    if $T_i' = T_i$ **then**
      Break the loop.
    else
      $T_i \leftarrow T_i'$
    end if
  end loop
  return (S_i, T_i)

Then, each segment S_i is matched with a subset of T_s such that the ROUGE-1 score between the subset and S_i is maximized. However, it is not feasible to find the optimal set due to the considerable running time. We apply a simple greedy approximation to find such a subset. From a null set T_i, we iteratively add to the subset the sentence with the highest ROUGE-1 gain between T_i and S_i. Algorithm 1 shows how we obtain the new training pair (S_i, T_i). $\bigoplus$ indicates the concatenation of sentences while keeping them in the same order as in the original text. We use ROUGE-1 as the matching criterion because the higher ROUGE-1 score usually implies higher scores on the other metrics such as ROUGE-2 or ROUGE-L, while ROUGE-1 enjoys lower time complexity compared with other ROUGE metrics.

This matching algorithm also ensures $T_i \neq \emptyset$ so that each S_i can be matched to at least one target sentence. A sentence $t \in T_s$ can be added to multiple subsets T_i because one sentence of summary may need the information from multiple segments.

3.2 Coarse Summary Generation

In coarse summary generation, we train a summarization model, that takes the segmented data as input. We first collect the training samples (S_i, T_i) generated by data segmentation to form a new dataset. This augments the source data to d_1/K times compared with the cut-off methods, where $d_1 = |D^1|$ indicates the averaged number of tokens of original source text. Thus, data segmentation helps the summarizer to better learn the task of the current stage. Additionally, because we incorporate the full input using segmentation, it does not rely on the leading bias in the cut-off method that only considers the first segment S_1. Afterward, we use these data to train a neural summarizer. This way, our model treats each part of the source text as equally important.

Given a source segment S_i and an optional query Q, we obtain the coarse summary segments using a backbone summarization model:

$$C_i^l = \text{SUMM}_l(Q, S_i)$$

Where $l \in [1, N]$ is the index of the current stage. Then, the n coarse summaries corresponding to the original source $S = \{S_1, S_2, \cdots, S_n\}$ are concatenated: $C^l = C_1^l \bigoplus C_2^l \bigoplus \cdots \bigoplus C_n^l$. We use C^l as the new source text of next stage, which compresses the input source data D^l. i.e. $D^{l+1} = C^l$. To pair with the D^{l+1}, the target to the next stage is copied from the original dataset, i.e. $T^{l+1} = T$.

The proposed framework is applicable to different backbone models $\text{SUMM}_l(*)$, such as BART (Lewis et al., 2020) and T5 (Raffel et al., 2020). We pick BART as the backbone model because it can best illustrate the benefits of our framework (Section 4.2).

3.3 Estimation of the Number of Coarse Stages N

The number of stages can be estimated by data stats and model characteristics. In SUMM^N, each coarse stage compresses the input to a shorter length. After N turns of coarse stages, the averaged length of source text is below K, the dataset is then fed into the fine-grained stage. Hence, the number of coarse stages can be computed by the following equation (details can be found in Appendix A):

$$\hat{N} = \lceil \frac{\log K - \log d_1}{\log c_1 - \log K} \rceil$$

where d_1 and c_1 are the average length of source text and coarse segments in stage 1. In Section 5.7 and Table 9, we demonstrate this estimation is close to the empirical number of coarse stages.

The greedy algorithm in SUMM^N for target matching is critical to the performance. Consider a duplication algorithm where each segment S_i is simply paired with the target T, i.e. $T_i = T$. Since the target text is longer than the text segmented by Algorithm 1, the generated summary of each coarse stage will be longer as well, leading to a lower compression speed and larger N. Besides,

Dataset	Type	Domain	Size	Source length	Target length	Query	$N+1$
AMI	Dialogue	Meetings	137	6007.7	296.6	✗	2
ICSI	Dialogue	Meetings	59	13317.3	488.5	✗	3
QMSum	Dialogue	Meetings	1808	9069.8	69.6	✓	2
SummScreen	Dialogue	TV shows	26851	6612.5	337.4	✗	2
GovReport	Document	Reports	19466	9409.4	553.4	✗	3

Table 1: The summarization datasets for evaluation. The source length and target length is the averaged number across the dataset. N indicates the number of coarse stages we use.

the duplication of the target will confuse the model, because some source segments will probably be paired with the same target, causing the model to generate duplicated content. Experiments (Table 7, "- stage 2" versus "- stage 2 & tar. seg.") show that ROUGE scores declines a lot when greedy target segment is replaced by the duplication algorithm .

3.4 Fine-Grained Summary Generation

When the input source of D^l is shorter than K, we can proceed to the fine-grained stage. In this stage, D^l is used to train a summarization model from scratch to obtain the final summary. The fine-grained stage works the same way as the vanilla backbone model. In fact, SUMM^N with $N = 0$ is the backbone summarizer. In the fine-grained stage, the model is directly trained on dataset (D^N, T) from the last coarse stage, and obtain the summary as the final output of SUMM^N:

$$F = \text{SUMM}_{N+1}(Q, D^N)$$

It is worth noting that, although source text may be shorter than 2 segments, i.e. $d^i \leq K$, we still add them in all stages, so that each summarization model can be trained on the full dataset.

4 Experiment Setup

We first list the datasets and metrics to evaluate the model. Then, we introduce the backbone model and baselines for comparisons. Finally, we present some implementation details.

4.1 Datasets and Metrics

Table 1 shows data statistics for the datasets[1].

AMI & ICSI (McCowan et al., 2005; Janin et al., 2003) are meeting scripts generated by Automatic Speech Recognition (ASR) systems. AMI is collected from product design meetings in a company while ICSI is collected from academic group meetings. Because the transcript is produced by ASR, there is a word error rate of 36% for AMI and 37% for ICSI.

QMSum (Zhong et al., 2021b) is a query-based meeting summarization dataset. It consists of meetings from three domains, including AMI and ICSI, and the committee meetings of the Welsh Parliament and the Parliament of Canada. Each query and sample are written by experts.

SummScreen (Chen et al., 2021) consists of community-contributed transcripts of television show episodes from The TVMegaSite, Inc. (TMS) and ForeverDream (FD). The summary of each transcript is the recap from TMS, or a recap of the FD shows from Wikipedia and TVMaze.

GovReport (Huang et al., 2021) is a large-scale long document summarization dataset with 19,466 long reports published by the U.S. Government Accountability Office on national policy issues.

We use ROUGE (Lin, 2004) as the automatic evaluation metric.[2] We split summary outputs into sentences to calculate the ROUGE-L score. If not specified, F1 scores are used in all results.

4.2 Backbone Model

We pick BART (Lewis et al., 2020) as our backbone summarization model because it performs well on short text summarization but not as good on longer texts, illustrating the benefits of our framework. Compared with other pretrained parameters, the BART-large model pretrained on the CNN/DM dataset yields the best performance (Zhang et al., 2021). So we use the BART-large-cnn parameter as a better starting point.

It is worth noting that we use separate backbone models for each stage and each was separately trained. We experimented with reusing the model parameters in multiple stages but obtained a lower

[1]Both QMSum and SummScreen can be accessed through SummerTime (Ni et al., 2021).

[2]We use `pyrouge`, a Python wrapper for the ROUGE: https://github.com/bheinzerling/pyrouge

	AMI			ICSI			QMSum-All			QMSum-Gold		
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L
PGNet	42.60	14.01	22.62*	35.89	6.92	15.67*	28.74	5.98	25.13	31.52	8.69	27.63
TopicSeg	51.53	12.23	25.47*	-	-	-	-	-	-	-	-	-
HMNET	52.36	18.63	24.00*	**45.97**	10.14	18.54*	32.29	8.67	28.17	36.06	11.36	31.27
TextRank	35.19	6.13	16.70*	30.72	4.69	12.97*	16.27	2.69	15.41	-	-	-
HAT-BART	52.27	20.15	50.57	43.98	10.83	41.36	-	-	-	-	-	-
DDAMS	53.15	**22.32**	25.67*	40.41	11.02	19.18*	-	-	-	-	-	-
SUMM^N	**53.44**	20.30	**51.39**	45.57	**11.49**	**43.32**	**34.03**	**9.28**	**29.48**	**40.20**	**15.32**	**35.62**

Table 2: ROUGE scores on three meeting summarizing tasks, AMI, ICSI, and QMSum. QMSum-ALL uses inputs with all turns while MSum-Gold uses inputs with only the gold turns. * denote the ROUGE-L scores without sentence split.

score, e.g. the ROUGE-1 score of stage 2 on the QMSum dataset decreases around two points if we use the best parameters of stage 1 summarizer as the starting point of training stage 2 summarizer. This is because the tasks of the different stages differ significantly. For instance, the input to the first stage of dialogue summarization is the dialogue turn while the input to the latter stages is the document.

4.3 Baselines

We compare the proposed framework with various baselines. **PGNet** (See et al., 2017) uses a pointer mechanism to copy the token from the training sample. **TopicSeg** (Li et al., 2019) is a multi-modal model jointly learning the segmentation and summarization. **HMNet** (Zhu et al., 2020) uses a hierarchical attention structure and cross-domain pre-training for meeting summarization. **TextRank** (Mihalcea and Tarau, 2004) is a graph-based ranking model for text processing. **HAT-BART** (Rohde et al., 2021) is a new hierarchical attention transformer-based architecture that outperforms standard Transformers. **DDAMS** (Feng et al., 2021) uses a relational graph to model the interaction between utterances by modeling different discourse relations.

For the SummScreen dataset, we use the neural and hybrid model scores reported by Chen et al. (2021). We rename these two baselines as **Longformer+ATT** and **NN+BM25+Neural** to clarify the difference between other baselines.

The baseline scores we report on GovReport are from the original paper (Huang et al., 2021). **BART Variant** indicates self-attention variants with full attention. **BART HEPOS** indicates encoder variants with head-wise positional strides (HEPOS) encoder-decoder attention.

4.4 Implementation Details

We fit all models into a single RTX A6000 GPU with a 48 GiB memory. We adopt the fairseq[3] implementation for BART. The learning rate is set to 2e-5 and the beam width is set to 2 for coarse stages and 10 for fine-grained stages. The maximum number of tokens in each batch is set to 2048. The maximum number of tokens in each source text is set to 1024 because we tried to extend the positional embeddings to 2048 or longer but obtained worse performance. We stop the coarse stage and start the fine-grained stage when the averaged source length is shorter than 2048 rather than 1024 to obtain a better performance (Section 5.7). For the output of each intermediate stage, we use <s> and </s> to separate each generated target segments C_i^l.

5 Results and Analysis

We discuss the evaluation results and effects of each component of SUMM^N in this section.

5.1 Overall Results

Meeting Summarization Table 2 shows the ROUGE scores on AMI, ICSI, and QMSum. Compared with the baseline models, SUMM^N achieves state-of-the-art results on almost all metrics. Specifically, SUMM^N improves SOTA on ICSI by **0.83**, and **1.96** ROUGE-2/L scores, improves SOTA on QMSum-Gold by **4.14**, **3.96**, and **4.35** ROUGE-1/2/L scores. These results demonstrate the effectiveness of SUMM^N on long dialogue summarization tasks.

[3]https://github.com/pytorch/fairseq

	SummScreen-FD			SummScreen-TMS		
	R1	R2	R-L	R1	R2	R-L
Longformer+ATT	25.90	4.20	23.80	42.90	**11.90**	41.60
NN+BM25+Neural	25.30	3.90	23.10	38.80	10.20	36.90
SUMM^N	**32.48**	**5.85**	**27.55**	**44.64**	11.87	**42.53**

Table 3: ROUGE scores on the SummScreen datasets including ForeverDreaming (SummScreen-FD) and TV MegaSite, Inc. (SummScreen-TMS).

	R-1	R-2	R-L
BART Variants			
Full (1024)	52.83	20.50	50.14
Stride (4096)	54.29	20.80	51.35
LIN. (3072)	44.84	13.87	41.94
LSH (4096)	54.75	21.36	51.27
Sinkhorn (5120)	55.45	21.45	52.48
BART HEPOS			
LSH (7168)	55.00	21.13	51.67
Sinkhorn (10240)	**56.86**	22.62	53.82
SUMM^N	56.77	**23.25**	**53.90**

Table 4: ROUGE scores on GovReport. For each baseline model, the number in parentheses is the maximum input length.

TV Series Summarization Table 3 shows ROUGE scores on SummScreen. SUMM^N outperforms on almost all metrics on two SummScreen datasets. Specifically, we improve **6.58**, **1.65**, and **3.75** ROUGE-1/2/L scores on the SummScreen-FD dataset. This result demonstrates the generalizability of SUMM^N over various domains including meetings and TV series.

Document Summarization Table 4 shows ROUGE scores on GoveReport. SUMM^N achieves state-of-the-art performance on ROUGE-2 and ROUGE-L, and compatible results on ROUGE-1. The results show that SUMM^N is applicable to both long dialogue and document summarization tasks.

5.2 Effects of Number of Stages

We also notice that the performance increases consistently when the number of stages goes up until the predefined number of stages. Figure 2 shows the ROUGE-1 scores of different tasks across stages. **Stage 1** indicates the model with only one coarse stage and no fine-grained stage. In this model, We directly use the first segment of the coarse summary as the output, i.e. C_1^1 of each sample. **Stage i** ($i > 1$) model contains $i - 1$ coarse stages and one fine-grained stage, the generated

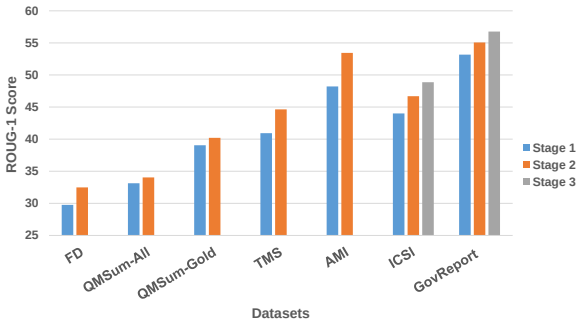

Figure 2: ROUGE-1 scores of various datasets at different stages. ICSI and GovReport have 3 stages, while the others have 2 stages. In all datasets, ROUGE-1 score increases with the increasing number of stages.

summary is from fine-grained summarization models, i.e. F.

Although stage 2 of SUMM^N on the ICSI dataset has already outperformed the baselines, the scores can be further improved by adding one more coarse stage. In fact, on all datasets, increasing the number of stages leads to a performance gain. This gain can be explained as the following: if the output of the current stage is longer than K tokens, adding one more coarse stage will help since the model will receive more information from the source text compared with simply truncating them. On the contrary, if the input is smaller than K, there is no need to add more stages, because there is only one segment.

5.3 Improvements over Backbone Models

SUMM^N also boosts the performance of a backbone model by a large margin. As shown in Table 5, it improves the BART-large model by **6.87**, **3.89**, **6.78** ROUGE-1/2/L on AMI. This indicates the capability of SUMM^N to boost the performance of a weak learner on long summarization tasks. In particular, when the backbone model is well pretrained on short input texts and performs well on short summarization tasks, SUMM^N could greatly increase the capability of the backbone model to process and read long source texts. Also, the backbone of SUMM^N can be easily replaced by some other models, and models do not necessarily have to be identical at every stage. For example, one can try different learners such as T5 as the backbone model and replace the model in stage 1 with a dialogue-to-document model.

		R1	R-2	R-L
AMI	Backbone	46.57	16.41	44.61
	SUMM^N	**53.44**	**20.30**	**51.39**
ICSI	Backbone	39.91	9.98	38.17
	SUMM^N	**45.57**	**11.49**	**43.32**
QMSum-All	Backbone	29.20	6.37	25.49
	SUMM^N	**34.03**	**9.28**	**29.48**
QMSum-Gold	Backbone	32.18	8.48	28.56
	SUMM^N	**40.20**	**15.32**	**35.62**

Table 5: Improvements of SUMM^N over backbone BART models on AMI, ICSI, and QMSum datasets.

		R-1	R-2	R-L	Input
BART-base	Backbone	41.54	13.80	38.75	1024
	SUMM^N	46.60	18.80	45.23	1024
T5-large	Backbone	47.81	16.06	45.77	512
	SUMM^N	51.85	19.40	49.94	512
PEGASUS-cnn_dailymail	Backbone	46.37	16.21	44.75	1024
	SUMM^N	50.15	19.07	48.28	1024

Table 6: ROUGE scores of different backbone models on AMI. For all backbone models with various maximum input lengths, ROUGE scores increase with the help of proposed framework. Input indicates the maximum number of tokens the model can take.

5.4 Generalizability over Backbone Models

To demonstrate our framework can generalize to different backbone summarization models, we replace the BART-large-cnn model in previous experiments with other neural summarization models including T5 (Raffel et al., 2020) and PEGASUS (Zhang et al., 2019) using Hugging Face. Table 6 shows the ROUGE scores of three different models that are trained and evaluated on AMI. In all models, SUMM^N improves the performance of backbone models by a large margin. For instance, although BART-base is a weaker summarizer compared with the BART-large model, the framework is still able to improve the ROUGE-1 score by 5.06.

5.5 Ablations

Table 7 shows the ablation study results of SUMM^N on the AMI test set. Removing stage 2 (using the first segment of the coarse summary C_1^1 as the generated summary) leads to a 5.23 ROUGE-1 score drop. Without data segmentation, the ROUGE-1 score decreases by 6.61 using the same fine-grained stage. Removing both stage 2 and target matching (use duplication algorithm instead) further decreases the performance. It even hurts the performance of the original BART model because the duplication of targets will introduce some biases towards the common part of the targets.

	R-1	R-2	R-L
SUMM^N	**53.44**	**20.30**	**51.39**
- stage 2	48.21	18.59	46.46
- data seg.	46.83	15.91	45.00
- stage 2 & tar. seg.	46.24	16.03	44.45
only BART	46.57	16.41	44.61

Table 7: Ablations on the test set of AMI. "- data seg." indicates removing data segmentation (the same as cut-off at limitation), "- tar. seg." indicates source segmentation paired with duplicated targets.

	AMI			ICSI		
	Read.	Conc.	Cove.	Read.	Conc.	Cove.
HMNet	3.93	4.05	4.15	3.21	3.33	3.84
SUMM^N	**4.45**	**4.13**	**4.23**	**4.12**	**3.55**	**4.06**

Table 8: Human evaluation scores. Read. indicates *Readability*, Conc. indicates *Conciseness*, and Cove. indicates *Coverage*.

5.6 Human Evaluation

We conduct a human evaluation to assess the following: *Readability* takes into account word and grammatical error rate to evaluate how fluent the summary language is; *Conciseness* measures how well the summary discards the redundant information; *Coverage* measures how well the summary covers each part of the dialogue.

We compare the results of SUMM^N and HMNet because HMNet is a baseline model with the good capability to read whole input. For each meeting in AMI and ICSI dataset, we ask 3 different annotators with English expertise to label the summaries. Each annotator was asked to read the meeting transcript, gold summaries, and generated summaries using the SummVis (Vig et al., 2021) toolkit. They were asked to rate each summary from 1 to 5 (higher is better) for each metric. We also shuffle the summaries of two models to reduce the bias.

Table 8 shows that SUMM^N achieves higher scores in *Readability*, *Conciseness*, and *Coverage* than HMNet in both AMI and ICSI dataset. Specifically, the *Readability* of SUMM^N greatly surpasses the baseline by around 0.5/1 point on AMI/ICSI dataset. This is because BART is well-pretrained and is able to generate more readable text and SUMM^N successfully maintains this capability.

	Avg. $\lvert D^i \rvert$	Avg. $\lvert C_i \rvert$	Comp. R	$\hat{N}_{val}$	N
Stage 1	7996.01	377.02	- -	1.41	2
Stage 2	3582.47	373.29	0.45	0.55	1
Stage 3	1517.02	492.89	0.42	-0.41	0

Table 9: Comparison of the empirical number of coarse stage N and the corresponding estimation $\hat{N}_{val}$ on the GovReport test set. Avg. $\lvert D^i \rvert$ and Avg. $\lvert C_i \rvert$ are averaged number of tokens in source text and coarse segments of stage i (Section 3.3). Comp. R is the compression rate R of the stage.

5.7 Intermediate Result Analysis

To gain more understanding of the multi-stage mechanism of SUMM^N, we analyze the number of coarse stages and the compression rate through statistics of intermediate stages.

Early Stopping of the Coarse Stage Although the ideal input of the final fine-grained stage should be shorter than K, the experiment results show that compressing input from $2K$ to $1K$ tokens usually hurts the performance of the model. This is probably because generating too many short segments which are hard to summarize confuses the model.

Thus, we increase the length of input to the final fine-grained stage from K to $2K$ to prevent noises in the training set. The modified formula to estimate the number of coarse stages $\hat{N}$ is shown as follows (details in Appendix A).

$$\hat{N}_{val} = \frac{1 + \log K - \log d_1}{\log c_1 - \log K}$$

$$\hat{N} = \lceil \hat{N}_{val} \rceil$$

Number of Coarse Stages To verify that our estimation $\hat{N}$ is close to the empirical number of coarse stages N, we use GovReport to compare the two as shown in Table 9. We choose this dataset because it contains the most number of samples among all five datasets, with completely three coarse stages as well.

Table 9 shows the empirical/estimated number of coarse stages. To clearly show the $\hat{N}$ value, we display the float number $\hat{N}_{val}$ as the estimated number, and N as the empirical number of "remaining coarse stages" (Table 1). As can be seen, $N = \hat{N} = \lceil \hat{N}_{val} \rceil$ holds for all stages, meaning that the estimated $\hat{N}$ is capable of estimating the correct N value. It is worth noting that, for stage 2 and stage 3, using this formula can also estimate "how many additional coarse stage do we need".

	Transformers	SUMM^N
Time	$O(n^2)$	$O(nK/(1-R))$
Gen. Tokens	$O(n)$	$O(n/(1-R))$

Table 10: Time complexity of inference and the number of tokens generated during inference (Gen. Tokens) by comparing Transformers and SUMM^N. n is the number of tokens in the source text. K is the maximum input length of the backbone model of SUMM^N. R is the averaged compression rate.

Compression Rate We analyze the change of compression rate across different stages. In SUMM^N, compression rate R_i is defined as the averaged source length of stage i divided by source length of stage $i-1$. As shown in Table 9, both compression rates in stage 2 and stage 3 of GovReport are around 0.4, this shows that the compression rate of SUMM^N across different stages are stable, meaning that the number of segments will decrease to around 40% of the previous stage steadily.

5.8 Time Complexity

Table 10 shows the time cost of inferring one sample using vanilla transformer versus SUMM^N. Although the SUMM^N needs to generate more tokens due to multi-stage pipeline, SUMM^N reduces the inference time from quadratic to lower, i.e., from $O(n^2)$ to $O(Cn)$, $C = K/(1-R)$. Regarding training the model, SUMM^N also need to infer $O(n)$ additional tokens on the train/dev/test sets (details in Appendix B).

6 Conclusion

In this paper, we propose SUMM^N, a simple, flexible, and effective framework for long dialogue and document summarization. It consists of multiple coarse stages and one fine-grained stage to iteratively compress the long source input. It enjoys the full power of backbone models while ensuring the full receptive field of the summarization model. We evaluate the model on various datasets and improve the baselines by a large margin.

Acknowledgement

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

References

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

Jiaao Chen and Diyi Yang. 2021. Structure-aware abstractive conversation summarization via discourse and action graphs. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1380–1391, Online. Association for Computational Linguistics.

Mingda Chen, Zewei Chu, Sam Wiseman, and Kevin Gimpel. 2021. Summscreen: A dataset for abstractive screenplay summarization. *arXiv preprint arXiv:2104.07091*.

Zhi Chen, Lu Chen, Zihan Xu, Yanbin Zhao, Su Zhu, and Kai Yu. 2020. Credit: Coarse-to-fine sequence generation for dialogue state tracking. *arXiv preprint arXiv:2009.10435*.

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.

Alexander Fabbri, Faiaz Rahman, Imad Rizvi, Borui Wang, Haoran Li, Yashar Mehdad, and Dragomir Radev. 2021. ConvoSumm: Conversation summarization benchmark and improved abstractive summarization with argument mining. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6866–6880, Online. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Xiachong Feng, Xiaocheng Feng, Bing Qin, and Xinwei Geng. 2021. Dialogue discourse-aware graph model and data augmentation for meeting summarization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 3808–3814. ijcai.org.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. 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.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China. Association for Computational Linguistics.

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.

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.

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 *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2021. Booksum: A collection of datasets for long-form narrative summarization. *arXiv preprint arXiv:2105.08209*.

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.

Manling Li, Lingyu Zhang, Heng Ji, and Richard J. Radke. 2019. Keep meeting summaries on topic:

Abstractive multi-modal meeting summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2190–2196, Florence, Italy. Association for Computational Linguistics.

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

Ziming Mao, Chen Henry Wu, Ansong Ni, Yusen Zhang, Rui Zhang, Tao Yu, Budhaditya Deb, Chenguang Zhu, Ahmed H Awadallah, and Dragomir Radev. 2021. Dyle: Dynamic latent extraction for abstractive long-input summarization. *arXiv preprint arXiv:2110.08168*.

Iain McCowan, Jean Carletta, Wessel Kraaij, Simone Ashby, S Bourban, M Flynn, M Guillemot, Thomas Hain, J Kadlec, Vasilis Karaiskos, et al. 2005. The ami meeting corpus. In *Proceedings of the 5th International Conference on Methods and Techniques in Behavioral Research*, volume 88, page 100. Citeseer.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Ansong Ni, Zhangir Azerbayev, Mutethia Mutuma, Troy Feng, Yusen Zhang, Tao Yu, Ahmed Hassan Awadallah, and Dragomir Radev. 2021. SummerTime: Text summarization toolkit for non-experts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 329–338, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Vahed Qazvinian and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 689–696, Manchester, UK. Coling 2008 Organizing Committee.

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.

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

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020. Sparse sinkhorn attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9438–9447. PMLR.

Amy JC Trappey, Charles V Trappey, and Chun-Yi Wu. 2009. Automatic patent document summarization for collaborative knowledge systems and services. *Journal of Systems Science and Systems Engineering*, 18(1):71–94.

Jesse Vig, Wojciech Kryscinski, Karan Goel, and Nazneen Rajani. 2021. SummVis: Interactive visual analysis of models, data, and evaluation for text summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 150–158, Online. Association for Computational Linguistics.

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*.

Yumo Xu and Mirella Lapata. 2020. Coarse-to-fine query focused multi-document summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3632–3645, Online. 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*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization.

Yusen Zhang, Ansong Ni, Tao Yu, Rui Zhang, Chenguang Zhu, Budhaditya Deb, Asli Celikyilmaz, Ahmed Hassan Awadallah, and Dragomir Radev. 2021. An exploratory study on long dialogue summarization: What works and what's next. *arXiv preprint arXiv:2109.04609*.

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, 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 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*, pages 5905–5921, Online. Association for Computational Linguistics.

Chenguang Zhu, Yang Liu, Jie Mei, and Michael Zeng. 2021a. 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, Online. Association for Computational Linguistics.

Chenguang Zhu, Ruochen Xu, Michael Zeng, and Xuedong Huang. 2020. A hierarchical network for abstractive meeting summarization with cross-domain pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 194–203, Online. Association for Computational Linguistics.

Chenguang Zhu, Ziyi Yang, Robert Gmyr, Michael Zeng, and Xuedong Huang. 2021b. Leveraging lead bias for zero-shot abstractive news summarization. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1462–1471.

A Computing the Number of Stages

With regard to text length, the source text of each stage needs to be compressed gradually to ensure that the summary with proper length can be generated in the final stage. Also, the compression level determines the required number of stages, which is a significant indicator of time cost.

Suppose one sample of the source of stage i contains $d_i = |D^i|$ words, while the source of next stage D_{i+1} contains $d_{i+1} = |D^{i+1}|$ words. Also because the input of next stage is the coarse summary of current stage, $D^{i+1} = C^i$, thus $d_{i+1} = |D^{i+1}| = |C^i|$. The maximum input length of the model is K, $c_i = \sum_{j=0}^{n} |C_j^i|/n$ indicates the averaged number of tokens in the segmented predictions. d_{i+1} can be expressed by the length of coarse summary which is the number of segment $\frac{d_i}{K}$ times the length of coarse segments c_i.

In each stage, we have:

$$d_{i+1} = \frac{d_i}{K} \times c_i$$

By iterating this equation for N time, the number of needed coarse stages N for a dataset can be decided in this way:

$$d_1 \times \prod_{i=1}^{N} \frac{c_i}{K} \leq K$$

Empirically, c_i are similar in different stages , thus we replace the production of c_i with c_1 to the N, i.e.

$$c_1^N \approx \prod_{i=1}^{N} c_i$$

Thus, the estimation of N value can be calculated as follows:

$$d_1 \times \frac{c_1^N}{K^N} \leq K$$

$$\hat{N} = \lceil \frac{\log K - \log d_1}{\log c_1 - \log K} \rceil$$

We also call c_i/K the compression rate of stage i, denoted as R_i. For target matching, the compression rate of duplication segmentation is 1 and greedy segmentation is less than 0.5. So that target segmentation algorithm helps reduce number of coarse stages.

After using the early stopping of coarse stage, the estimation formula changes as follows:

$$d_1 \times \frac{c_1^N}{K^N} \leq 2K$$

$$\hat{N} = \lceil \frac{1 + \log K - \log d_1}{\log c_1 - \log K} \rceil$$

B Time Complexity

Suppose the length of the input is n, by segmenting the source text into n/K segments, the time cost of forwarding of one segment is K^2, thus the total time cost of stage 1 is $n/K \times K^2 = nK$. Then, in the next stage, the length of the source text is reduced to nR, thus the time complexity of stage 2 is nKR. We can list the total time cost by adding them together:

$$T(n) = \sum_{i=0}^{\infty} nKR^i = \frac{nK}{1-R}$$

	ICSI
SUMM^N	The project manager opens the meeting by recapping the events of the previous meeting. The marketing expert presents the results of market research , which shows that users want a fancy-looking remote control that is easy to use and has a fancy look and feel. The user interface designer presents the user interface concept for the remote , which is based on the idea that a remote should be simple and user-friendly. The industrial designer presents about the internal components of a remote control. The group discusses using kinetic energy to power the device , using a simple battery for the LCD screen , and using an advanced chip for the advanced chip. The project manager closes the meeting , telling the team members what their tasks will be for the next meeting. · · · The Marketing Expert will research how to produce a remote that is technologically innovative. The User Interface Designer will look at how to make a remote out of wood or plastic with either a wooden or plastic cover. The Group will not work with teletext. There was a lack of information on the cost of components and materials.
Gold	The project manager opened the meeting and recapped the decisions made in the previous meeting. The marketing expert discussed his personal preferences for the design of the remote and presented the results of trend-watching reports , which indicated that there is a need for products which are fancy , innovative , easy to use , in dark colors , in recognizable shapes , and in a familiar material like wood. The user interface designer discussed the option to include speech recognition and which functions to include on the remote. The industrial designer discussed which options he preferred for the remote in terms of energy sources , casing , case supplements , buttons , and chips. The team then discussed and made decisions regarding energy sources , speech recognition , LCD screens , chips , case materials and colors, case shape and orientation , and button orientation.· · · The case covers will be available in wood or plastic. The case will be single curved. Whether to use kinetic energy or a conventional battery with a docking station which recharges the remote. Whether to implement an LCD screen on the remote. Choosing between an LCD screen or speech recognition. Using wood for the case.

Table 11: Sample output summary SUMM^N on the ICSI dataset. Tokens marked in grey indicate the out-of-boundary contents of truncation models. Brown tokens are some topic words (manually selected) emerged in the gold summary. Tokens marked in red indicate the concepts of out-of-boundary text.

Similarly, in training phrase, stage 1 generates $O(n)$ tokens while stage 2 generates $O(nR)$ tokens for each sample in train/dev/test set. We can list the total generated tokens by adding them together:

$$T(n) = \sum_{i=0}^{\infty} nR^i = \frac{n}{1-R}$$

Thus the time cost of forwarding reduces. For instance, the inference time of SummScreen-TMS dataset reduces to $1024/(1-0.27)/6420.64 = 21.8\%$, and GovReport dataset reduces to $1024/(1-0.43)/7890.46 = 22.8\%$ of original time cost, compared with $O(n^2)$ transformers. This shows the efficiency of SUMM^N. On the other hand, since the training phrase needs to generate the target for each sample in the train/dev/test set, the training time of SUMM^N also includes the additional generation of $O(\frac{n}{1-R})$ tokens for each sample in the dataset.

C Case Study

Table 11 shows a concrete sample summary generated by SUMM^N. It captures the topics of the source text and smoothly follows the outline of the gold summary. Also, SUMM^N is able to evenly generate the information of the whole summary, including the last part of source text which is truncated in the standard BART-large models.