Dynamic Sliding Window Modeling for Abstractive Meeting Summarization

Zhengyuan Liu†‡, Nancy F. Chen†‡
†Institute for Infocomm Research, A*STAR, Singapore ‡CNRS@CREATE, Singapore
liu.zhengyuan,nfychen@i2r.a-star.edu.sg

Abstract

Summarizing spoken content using neural approaches has raised emerging research interest lately, as sequence-to-sequence approaches have improved abstractive summarization performance. However, summarizing long meeting transcripts remains challenging. Meetings are multi-party spoken discussions where information is topically diffuse, making it harder for neural models to distill and cover essential content. Such meeting summarization tasks cannot readily benefit from pre-trained language models, which typically have input length limitations. In this work, we take advantage of the intuition that the topical structure of meetings tends to correlate with the meeting agendas. Inspired by this phenomenon, we propose a dynamic sliding window strategy to elegantly decompose the long source content of meetings into smaller contextualized semantic chunks for more resourceful modeling, and propose two methods without additional trainable parameters for context boundary prediction. Experimental results show that the proposed framework achieves state-of-the-art abstractive summarization performance on the AMI corpus and obtains higher factual consistency on competitive baselines.

Index Terms: Spoken language summarization, dialogue summarization

1. Introduction

Abstractive summarization is one of the most challenging tasks in natural language processing, involving context understanding, information collecting, and language generation [1]. Various neural models, from pointer-generator networks [2] to pre-trained language models [3], have been proposed and brought significant improvement on the generation quality, especially in fluency and readability [4]. Although most prior work focuses on monologues like news [5, 6], summarizing dialogues is gaining research attraction [7, 8, 9, 10]. Meetings are a type of multi-party conversation [11] that set a high bar for information gathering, consolidation, and distilling. A typical meeting conversation covers multiple sub-topics and the average turn number is much higher than that of short conversations like social chats [12] and inquiry-answering [9]. Neural language generation quality also tends to degrade when faced with massive contextual information [13]. Moreover, when adopting Transformer-based pre-trained language models like BERT [14] and BART [3], the transcript length exceeds their maximum positional embedding limitations. Text truncation, the default approach to bypass such limitations, is ill-suited in the use case of meeting summarization, as it often results in irreversible loss of useful context. In recent studies of summarizing long monologic documents, researchers have proposed efficient Transformer attention schemes [15], multi-block aggregation [16], and static sliding window approaches [9, 17]. However, most previous approaches need to modify the original Transformer architecture and introduce additional training parameters, and it is not straightforward to adopt such methods on the off-the-shelf language backbones.

Meetings usually consist of a set of agenda items [11], where each item focuses on a specific topic. Thus the entire conversation is inherently organized at the topic level. This is also reflected in the human-written summaries, which condense key information for each discussion point. Therefore, we postulate the divide-and-conquer strategy can be useful to tackle the aforementioned challenges of abstractive meeting summarization. More specifically, we aim to decompose the long transcript into multiple segments, and obtain the final output by aggregating all summary snippets of each segment. In this work, we propose to enhance a sequence-to-sequence summarizer with a dynamic sliding window strategy in order to tackle the length limitation of Transformer-based language models. Unlike conventional fixed sliding window approaches [14], our proposed framework can dynamically decide the start position of each context segment during the generation process. To this end, we introduce two parameter-free methods for automatic context text boundary prediction, which can be readily integrated into the existing language backbones without any layer modification and additional trainable parameters: (1) a learning-based approach that directly generates the supporting utterance; (2) an attention-based approach that uses decoder-to-encoder cross-attention scores as the context span pointer. While the proposed methods are inspired by the organizational features of meeting transcripts, the generality of them is not limited to the meeting domain, and is in principle applicable to any task that might face input length limitations when exploiting pre-trained language models. Experimental results on the representative corpus AMI [11] show that the summary generation benefits from our dynamic sliding window strategy and achieves state-of-the-art performance without additional in-domain pre-training and discourse information. Further sample analysis demonstrates that the enhanced framework performs better than the baseline model when considering factual consistency.

2. Related Work

Text summarization is mainly studied in abstractive and extractive paradigms [18]. For extractive summarization, traditional methods study various linguistic and statistical features via lexical [19] and graph-based modeling [20]. Much progress has been made by contextualized neural extractors [21, 22]. While abstractive summarization is challenging, its performance has achieved substantial improvements in the news domain [5]: sequence-to-sequence models were introduced for fluent generation [1], pointer-generator network [2] elegantly handles out-of-vocabulary issues, and large-scale pre-trained language models bring further performance improvement [3, 4].

Recently, neural summarization for spoken languages has become an emerging research area [12]. With the abundance of automatic meeting transcripts and practical value of real-world use cases, meeting summarization attracts much attention.
In this section, we elaborate on our framework for meeting summarization with the dynamic sliding window strategy. Since meetings are usually taken with a set of sub-topics, we require grasping useful contextual information across the entire conversation, as the supporting segments for each abstractive summary sentence. Based on this extractive annotation, we obtained the start/end supporting utterance indices in the conversation of each summary sentence. Then we sorted them by their occurrence order in the source content, and observed that 80% of sentences in reference summaries match the same agenda order as in the source content. This finding suggests that when humans write a summary, they might refer to the source content and record the key points sequentially.

3.2. Segmentation Analysis

When adopting a divide-and-conquer strategy, only a part of the conversation will be extracted. To assess whether sufficient context information is provided by the segmented source content, we calculate the word level coverage with extractive annotations (see Figure 1) in the AMI corpus. With the start/end supporting utterance indices described in Section 3.1, we split the original transcripts into a number of segments. Then word-level recall is calculated between each summary sentence and its corresponding segment. The overall word-level coverage between summary snippets and their corresponding source segments is 74%, which is substantial since the abstractive summaries are often written via sentence paraphrasing and introducing some novel words [2, 31].

3.3. Summary Conversion

As the final output under a divide-and-conquer strategy is produced by aggregating summary snippets, to build the training ground-truth, we split each reference summary into multiple parts based on the analysis in Section 3.1 and Section 3.2. For each summary sentence, we constructed its context segment by using the corresponding extractive annotation. Then we ordered the summary sentences by their occurrence indices in the source conversation, and merged adjacent summary sentences to summary snippets if their context segments had a certain overlap. For summary sentences without extractive annotation, we reorganized them via calculating word-level overlap with existing content segments. Furthermore, we observed that some summary snippets started with a pronoun that refers to a precedent personal named entity. To maximize semantic integrity when producing one snippet, we used coreference resolution on the original summary; if one summary snippet starts with a pronoun that refers to a precedent personal named entity. To maximize semantic integrity when producing one snippet, we used coreference resolution on the original summary. When adopting a divide-and-conquer strategy, only a part of the conversation will be extracted. To assess whether sufficient context information is provided by the segmented source content, we calculate the word level coverage with extractive annotations (see Figure 1) in the AMI corpus. With the start/end supporting utterance indices described in Section 3.1, we split the original transcripts into a number of segments. Then word-level recall is calculated between each summary sentence and its corresponding segment. The overall word-level coverage between summary snippets and their corresponding source segments is 74%, which is substantial since the abstractive summaries are often written via sentence paraphrasing and introducing some novel words [2, 31].

3.4. Proposed Neural Meeting Summarizer

In this section, we elaborate on our framework for meeting summarization with the dynamic sliding window strategy.

\[ \text{Average Source Content Length:} \quad \text{Sentence Level:} \quad 3.59 \quad 3.86 \quad 2.44 \quad 17.55 \]
\[ \text{Word Level:} \quad 45.8 \quad 54.65 \quad 45.54 \quad 323.3 \]

Table 1: Data statistics of the news summarization benchmarks and the AMI meeting corpus.
4.1. Dynamic Sliding Window

The sliding window splits a long input into a number of shorter spans, processes the spans in order (e.g., from left to right), and adopts aggregation for the final output. It is a straightforward but useful method that is commonly applied to long sequence encoding [14], and it is generally controlled by two parameters: window size denotes the context span size, and stride size is the amount of movement at each sliding step. In previous work, these two parameters are often fixed. Here we propose a dynamic sliding window strategy, where the stride size at each sliding step is predicted by the model. More specifically, given the window size $k$, at each sliding step, one index $j$ is selected (equals stride size is $j$) in the range of $[0, k - 1]$ as the start position of the next window.

4.2. Base Neural Architecture

We use the Transformer-based sequence-to-sequence architecture [32], and select the large-scale pre-trained language backbone ‘BART-base’ [3] for model initialization. The encoder consists of 6 stacked Transformer layers, where each layer has two sub-components: a multi-head self-attention layer and a position-wise feed-forward layer. Between the two sub-components, residual connection and layer normalization are used. The $i$-th encoding layer is formulated as:

$$
\tilde{h}^i = \text{LayerNorm}(h^{i-1} + \text{MHAttention}(h^{i-1})) \quad (1)
$$

$$
h^i = \text{LayerNorm}(\tilde{h}^i + \text{FFN}(\tilde{h}^i)) \quad (2)
$$

where $h^i$ is the $i$-th layer input, MHAttention, FFN, and LayerNorm are multi-head attention, feed-forward, and layer normalization, respectively.

The decoder consists of 6 stacked Transformer layers as well. In addition to the two encoding sub-components, the decoder performs another multi-head attention over the previous decoding hidden states, and all the encoded representations (i.e., cross-attention). Then, the decoder generates tokens in an autoregressive manner from left to right.

4.3. Adopting Dynamic Sliding Window for Meeting Summarization

The proposed dynamic sliding window is a general design that can be applied to various neural architectures, and its overview is shown in Figure 2. In our summarization setting, the source content is a sequence with $n$ tokens $C = \{w_1, w_2, ..., w_n\}$, and the summary is composed of $m$ snippets $S = \{s_1, s_2, ..., s_m\}$. In a traditional sequence-to-sequence process, the whole source content $C$ is fed to a model as input, and the output summary is generated at one decoding stage. With the dynamic sliding window strategy, the encoding-decoding process will be conducted in a number of steps, as shown in Figure 2. Given the window size $k$, at $i$-th sliding step, the start token index is $i_{left}$, and the end token index $i_{right}$ is $i_{left} + k$. Then the input sequence of encoder at $i$-th step is $c_i = \{w_{i_{left}}, ..., w_{i_{right}}\}$.

After encoding, the contextualized hidden representation $h_i = \{h_{i_{left}}, ..., h_{i_{right}}\}$ is fed to the decoder to generate a summary snippet, and all snippets are merged to form the final output.

4.4. Context Boundary Prediction

To achieve dynamic sliding prediction, one approach is to directly predict the stride size as in [33], where reinforcement learning is used to decide the value, and Schuller et al., [17] proposed to generate a special indicator for the moving operation. In this paper, we introduce two approaches that can be easily integrated into existing language models without any additional neural components: (1) a learning-based method Retrospective and (2) an attention-based method Cross-Attention.

Retrospective Since the supporting utterance list of each summary snippet is labeled, one way to obtain the dynamic stride step is to encourage the model to learn context boundary generation via the sequence-to-sequence process. More specifically, at $i$-th step, the decoder not only generates the summary snippet $s_i$, but also the last supporting utterance of its context segment that is described in Section 3.1 (they are concatenated with a special token ‘[SEP]’). Therefore, each predicted snippet $s_i$ is formulated as $\langle s \rangle, t_0, t_1, ..., t_j, [SEP], t_0, t_1, ..., t_k, \langle s \rangle$, where $t^i$ and $t^j$ denote the summary and the supporting utterance span respectively. The next context window will start from the boundary span $t^j$ (see Figure 2, Method I).

Cross-Attention While the Retrospective approach does not require any additional trainable parameter, it needs a few more decoding steps for boundary generation. Here we propose another parameter-free method that has the same inference complexity as the original summarization process. As the Transformer decoder conducts a cross-attention on the encoded representation [32], we use the attentive salience as the boundary indicator. More specifically, we first normalize the cross-attention matrices extracted from top decoding layers with max-pooling, and then take the last utterance span that has the attentive score above average as the prediction of current context.
Table 2: ROUGE F1 scores on the AMI test set from baseline models and our framework. * Results are reported as in [28].

<table>
<thead>
<tr>
<th></th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstractive baselines*: Seq2Seq+Attention</td>
<td>53.42</td>
<td>22.57</td>
<td>28.52</td>
</tr>
<tr>
<td></td>
<td>BART-Truncate</td>
<td>43.51</td>
<td>17.52</td>
</tr>
<tr>
<td></td>
<td>BART-SW-Retrospective (Ours)</td>
<td>52.83</td>
<td>21.77</td>
</tr>
<tr>
<td></td>
<td>BART-SW-CrossAttention (Ours)</td>
<td>52.91</td>
<td>21.91</td>
</tr>
</tbody>
</table>

boundary. Since the predicted supporting utterance/extracted attentive scores indicate the most salient span at the current generation step on the right, it is used to determine the amount of sliding movement (see Figure 2, Method II).

After all sliding steps, generated snippets \{s_1, s_2, ..., s_m\} are concatenated as the final output summary S, and duplicate sentences will be removed in a post-processing step.

5. Experimental Results and Analysis

5.1. Configuration

The proposed framework was implemented using PyTorch. Learning rate was set at 2e-5, and AdamW optimizer was applied. We trained each model for 20 epochs, and selected best checkpoints based on ROUGE-2 validation score [34]. Input sequences were processed with the sub-word tokenization scheme as in [3], and repeated output sentences were removed.

Based on the summary conversion described in Section 3.3, we obtained 598/131/143 snippet-level samples as training, validation, and test set. At the training stage, to simulate the input noise during inference time, we randomly added k adjacent utterances (5 ≤ k ≤ 15) in each context chunk. At the testing stage, for each sample, the generation process of our BART-SW-Retrospective model starts with the first chunk of a default window size, which is initialized as 1024. Then we used the sliding prediction described in Section 4.3 to obtain the next chunk, until the entire transcript was processed. We also assessed the BART-SW-Fixed model by fixing all window size at 1024, the BART-Truncate model by truncating the transcript of 1024 tokens, and the BART-SW-GoldSeg model by using the gold context segmentation.

5.2. Results on the AMI Corpus

Following previous work [28, 35], we used the ROUGE score [34] for evaluation, and reported ROUGE-1 (R-1), ROUGE-2 (R-2) and ROUGE-L (R-L). We selected strong baselines for extensive comparison including Pointer-Generator [2], Sentence-Gated [7], TopicSeg [25], HMNet [35], and DDA-GCN [28]. As shown in Table 2, for meeting summarization, abstractive approaches generally perform much better than extractive ones, due to the need of information collecting and paraphrasing. Our proposed models BART-SW-Retrospective and BART-SW-CrossAttention obtain similar generation scores, and they both outperform models with a fixed window (BART-SW-Fixed) and with text truncation (BART-Truncate), and this shows the effectiveness of adopting sliding window strategy for long transcript summarization. Moreover, our framework achieves the comparable performance of the contemporary state-of-the-art models, while those models require to modify the original Transformer architecture [8], or use additional in-domain pre-training and discourse information [28].

5.3. Factual Consistency Analysis

We first conducted a text quality analysis on the generated summaries across models. As shown in Figure 3, based on the strong generation capability of the language backbone, all BART-based models can produce fluent and grammatically correct summaries. While the model with text truncation achieves relatively acceptable ROUGE scores (as reported in Table 2), it produces sentences that are factually inconsistent with the source content, as shown in Figure 3. We speculate that this is caused by over-fitting the limited training samples, and we observed that some phrases that are irrelevant to the source content are repeated in summary generation. In contrast, as our proposed framework can produce the final summary based on relevant context segments without truncation, it performs better than the base model considering factual consistency. We observed a 37% relative reduction of inconsistency errors on the AMI test set via human evaluation. Additionally, we conducted a stride prediction assessment. For the Retrospective method described in Section 4.3, the predicted context boundaries are expected to be located closely to the ground-truth. With the best checkpoint, we observed that the average utterance-level distance between gold boundary span and model prediction is 2.7 (48 characters), indicating that the model is able to predict the correct start position at each sliding step.

6. Conclusion

In this work, to tackle the challenges of summarizing lengthy meeting transcripts, we introduced a dynamic sliding window strategy to resourcefully decompose the long source content to smaller contextualized semantic chunks for summarization modeling, and proposed two methods without any additional training parameters for context boundary prediction. Our experiments on the AMI corpus achieved state-of-the-art abstractive summarization performance, and we also obtained higher factual consistency on competitive baselines.
7. References


