Document Summarization with Text Segmentation

Lesly Miculicich
leslym@microsoft.com
Microsoft AI
Bellevue, Washinton, USA

Benjamin Han∗
dingjung.han@microsoft.com
Microsoft AI
Bellevue, Washinton, USA

ABSTRACT
In this paper, we exploit the innate document segment structure for improving the extractive summarization task. We build two text segmentation models and find the most optimal strategy to introduce their output predictions in an extractive summarization model. Experimental results on a corpus of scientific articles show that extractive summarization benefits from using a highly accurate segmentation method. In particular, most of the improvement is in documents where the most relevant information is not at the beginning thus, we conclude that segmentation helps in reducing the lead bias problem.

CCS CONCEPTS
• Computer systems organization → Embedded systems; Redundancy; Robotics; • Networks → Network reliability.

KEYWORDS
text segmentation, extractive summarization

ACM Reference Format:

1 INTRODUCTION
Extractive summarization offers the ability to retrieve relevant sentences in a document. The applications of this technology can speed up organization’s work and could serve as pre-processing step for other document understanding tasks in which the input is too long to be easily processed.

Documents have an internal structure that can be explicit like content tables, or implicit like change of topics. The exploitation of these structures is pertinent for summarization for helping to locate the most relevant information in the document. Thus our objective is to automatically detect text segments and incorporate this information in an extractive summarization model to boost its accuracy. Given that the automatic extraction of document structures have challenges in real application scenarios, we aim at detecting implicit changes of topics in the text.

We explore two state-of-the-art models for text segmentation, one trained with supervised learning and the other is unsupervised. Supervised methods exhibit better accuracy however are limited by the feature characteristics of the training data such as domain, genre, and distribution; thus they have difficulties for generalizing to unseen data. Unsupervised methods work in a more generic manner at the cost of less correctness over in-domain data. We use the prediction of these methods in an extractive summarization model. We assess a comprehensive set of strategies for integrating segment data in the summarization model, and show that segmentation increases the quality of the summarization model on a corpus of scientific papers [2]. However, the effectiveness depends on the accuracy of the segmentation model.

This paper is organized as follows: In Section 2, we described the related work; in Sections 3 and 4, we define our models for text segmentation and summarization respectively; in Section 5, we report the dataset and metrics used for the evaluation and the results are shown in Section 6. Finally, Section 7 contains our conclusions and future work.

2 RELATED WORK
On unsupervised segmentation, one of the first successful approaches was TextTiling [5]. It utilizes similarity scores between adjacent sentences to decide whether there is a change of segments. The original work uses sentence similarity based on word frequency. Later work updated this method by using cosine similarity between sentence embeddings [14, 16]. On supervised segmentation, the later work uses sequence-to-sequence learning, for instance [9] uses a sequence encoder and, for each token, predicts whether there is a new segment or not. In [6], the authors use an RNN based pointer-network, the RNN has as many time-steps as segments, at each time step, it selects a token which indicates the end of a segment.

Text segmentation has been successfully applied to automatic summarization. Most of this work is evaluated on meeting transcripts. The main idea is to split the documents on segments and then summarize each segment. This can be implemented in a pipeline (first segment then summarize) or in an end-to-end fashion [9, 10]. More related to the present work, [13] proposed an extractive summarization model that includes information for segments. However, they use Oracle document sections, here, we train a segmentation model to predict the sections and then we integrate this information in the summarization model.

3 TEXT SEGMENTATION
Text segmentation is the task of dividing a text in meaningful parts such topics or sections. In this work, we present an unsupervised and a supervised method for document segmentation. Both approaches have state-of-the-art results on topic segmentation for
meeting transcripts. Here, we adapt and optimize them for section detection in text documents. The task is defined as follows: Given a set of input sentences \( S = \{S_1, \ldots, S_M\} \) with an underlying segment structure; the objective is to predict a sequence \( Y = \{y_1, \ldots, y_M\} \) where \( y_i \) is a binary value indicating whether \( S_i \) is the beginning of a new segment.

### 3.1 Unsupervised Segmentation
We based our model on the unsupervised segmentation approach proposed in [14]. It is a modified version of TextTiling [5] that detects topic changes with a similarity score based on BERT embeddings [3]. We first compute the representations for every sentence. Then, we divide the document in overlapping windows and perform max pooling to get the window representation. We compute cosine similarity among adjacent windows and derive segment boundaries where the semantic similarity is lower than a given threshold. We adjust the window size, and the similarity threshold parameters using a validation set.

### 3.2 Supervised Segmentation
We use the supervised segmentation approach proposed in [17]. We divide the document in overlapping windows, and each window is encoded with a transformer network [15]. The segmentation is performed as a sequence labeling task, where each token in the sequence is assigned with a binary label to indicate whether it is the start of a new segment. We initialize the weights of the model with the pretrained model DeltaLM [11], and adjust the window size, stride size, and the classification threshold parameters using a validation set.

### 4 EXTRACTIVE SUMMARIZATION

Extractive summarization is the task of finding the subset of sentences in a document that best summarize it. Following [8], we define extractive summarization as a sequence labeling task. Given a set of input sentences \( S = \{S_1, \ldots, S_M\} \) the objective is to assign a label \( y_i \in \{0, 1\} \) to each \( S_i \), indicating whether the sentence should be included in the summary. The model is composed by two transformer encoders: word-level and sentence-level. Each document is tokenized and encoded with the word-level transformer. We introduce a special to token [CLS] at the end of each sentence. The output vector corresponding to this token serves as the sentence representation. All sentence representations plus their corresponding positions are input to a secondary sentence-level transformer. The word-level transformer is initialized with the pretrained model DeltaLM [11] whereas the sentence-level transformer is initialized with random values, and it is composed of only 2 layers. We use binary-cross entropy to train the model.

In order to manage long input sequences, the documents are chunked into equal size block. Each chunk is encoded with the word-level transformer independently. Then the output of the chunks are concatenated and the rest of the model is the same.

#### 4.1 Integrating Segment Information
We deem adequate to integrate the segment information in the sentence-level transformer. The referred segment information can be either the segment position in the document or the segment semantic representation. Both are relevant and serve different purposes, one learns the location of the relevant information, and the other spots the relevant segment depending on its content.

4.1.1 Segment position encoding. We use a learned positional encoding with a maximum of 10 segments. To avoid position bias, we applied normalization to the number of segments in the document as follows: \( \text{pos}_i = i + \text{maxseg}/(n_{\text{seg}} + 1) \), where \( \text{pos}_i \) is the final position, \( i \) is the segment index, \( \text{maxseg} = 10 \), and \( n_{\text{seg}} \) is the number of segments in the document.

4.1.2 Segment Embedding. We calculate the segment embedding by applying pooling to its tokens embeddings. We used maximum, minimum, and mean pooling. Preliminary experiments showed that mean pooling has better results, thus we use it.

4.1.3 Segment Position HiStruct. Following [13], we encode positions in a hierarchical manner by summing the segment position and sentence relative position in the corresponding segment. According the original experiments in the paper, using learnable embeddings and summing performed the best, thus we apply the same strategy.

We integrated the segment information by either adding or concatenation the segment representations.

### 5 DATA AND METRICS
For our experiments, we use Arxiv dataset [2]. It contains scientific articles with annotation of sections and sentences. It is composed of 203,037 samples for training, 6,436 for validation, and 6,440 for testing. We use sections as segment markers. Table 1 shows the statistics of sentences per section in the training set.

#### 5.1 Summarization
We use ROUGE score [7] for evaluating the summarization models. It measures the n-gram overlapping between the predicted summary and a reference. We report the F1 score of uni-grams (R1), bi-grams (R2), and the longest matching sequence (RL).

#### 5.2 Text Segmentation
Two standard evaluation metrics are used to evaluate text segmentation: Pk [1] and WinDiff [12]. Pk represents the probability that a randomly chosen pair of words at a distance of \( k \) is inconsistently classified; that is, for one segmentation the pair lies within the same segment, while for another the pair spans across segment boundaries. This is implemented by using a sliding window of size set to half of the average true segment length, and counting how many

<table>
<thead>
<tr>
<th>Statistics</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. sentences per summary</td>
<td>11</td>
</tr>
<tr>
<td>Avg. sections per document</td>
<td>5.5</td>
</tr>
<tr>
<td>Avg. sentences per document</td>
<td>130</td>
</tr>
<tr>
<td>Max. sentences per summary</td>
<td>110</td>
</tr>
<tr>
<td>Max. sections per document</td>
<td>60</td>
</tr>
<tr>
<td>Max. sentences per document</td>
<td>1268</td>
</tr>
</tbody>
</table>

Table 1: Sections and sentences statistics on Arxiv data-set.
times the predictions differ from the reference. This probability can be further decomposed into two conditional probabilities: the miss and the false alarm probabilities. WinDiff is a modification of Pk where the algorithm slides a fixed-sized window across the text and penalizes whenever the number of predicted boundaries within the window does not match the true number of boundaries within the same window.

5.3 Lead bias
Following [4], we calculate the R1 score on three different label distributions – D-early, D-middle and D-last – which are obtained by first sorting documents by the average sentence position of the positive labeled sentences: D-early are the first 100 documents, D-middle are the middle 100 documents, and D-late are the last 100 documents.

6 EXPERIMENTAL RESULTS
In this section, we describe the results of both unsupervised and supervised text segmentation methods. We also compare the different methods for integrating text segmentation on the extractive summarization model based on ground truth segments. Finally, we use the best integration method to report final results on summarization.

6.1 Unsupervised Segmentation
We tuned the hyper-parameters of the model using 15 documents from the validation set. We evaluate a window size in the range [1, 5, 10, 15] and threshold in [0.4, 0.5, 0.6] (see Table 2). We picked 0.5 and 5 as the threshold and the window size respectively.

We include two simple baselines models: a Random method that places segment boundaries uniformly at random, and an Even method that places boundaries every k sentences. The results are shown in Table 3.

6.2 Supervised Segmentation
In the supervised segmentation model, a document is processed by sliding windows. The widow’s size is the number of sentences to be processed in a chunk. We tested sizes in the range of [10, 40]. Figure 1(a) shows the Pk score in the development set. The best score was obtained with a window of 20. Similarly, the stride is the number of overlapping sentences in the windows. We evaluated values in the range of [3, 12]. Figure 1(b) shows the Pk score in the development set with a window size of 20. The best score was obtained with a stride of 7, the best value is consistent for all window sizes. Finally, the model outputs a score in the range of [0, 1] for each sentence. The score threshold determines whether the sentence is the start of a new segment. Figure 1(c) shows the WinDiff scores for different thresholds on the development set obtained with the best model. The optimum threshold value is 0.35. The final results are shown in Table 3.

6.3 Extractive Summarization with Segmentation
Table 4 shows the ROUGE scores of different methods of integrating segmentation information on the extractive summarization model (ExtSum). As the objective is to evaluate the best integration method, we use the ground truth annotation for segments, named Oracle. We found that both segment position and segment embedding help to improve the model. We also found that concatenating the segment information to the sentences input works better than adding it. Finally, using a flat position embedding have equal or slightly better results that using a hierarchical position as purposed in [13].

We compare the results of unsupervised and supervised methods with current SOTA [13] (Table 5). We use the best model: ExtSum with concatenation of segment position and embedding. We also include the results of ExtSum with oracle segmentation to define the upper limit of this method. We measure the ROUGE scores together with the the lead bias metrics D-early, D-middle and D-late described in Section 5.3. The unsupervised method rather decrease the scores for summarization. The low accuracy of the unsupervised segmentation could be adding noise instead of helping. The supervised segmentation method show improvement above all metrics, in particular in the middle and late part of the document as shown by D-middle and D-late. To further analyze this results, we plot the relative position of extracted sentences in the document (Figure 2). We can see that adding segmentation information makes the ExtSum model extract less sentences from the middle part, and more from the early and late parts. This is consistent with Arxiv dataset, where the introduction and conclusions contains the most relevant information for the summary.
Figure 1: Hyper-parameter tuning for the supervised segmentation model. (a) Window; (b) Stride; (c) Threshold.

<table>
<thead>
<tr>
<th>Model</th>
<th>R1</th>
<th>R2</th>
<th>RL</th>
<th>D-early R1</th>
<th>D-middle R1</th>
<th>D-late R1</th>
</tr>
</thead>
<tbody>
<tr>
<td>HiStruct+ (with Longformer-base 28k tok.) [13]</td>
<td>45.22</td>
<td>17.67</td>
<td>40.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ExtSum (with DeltaLM and chunked input 25K tok.)</td>
<td>48.91</td>
<td>20.62</td>
<td>43.85</td>
<td>48.9</td>
<td>55.9</td>
<td>47.6</td>
</tr>
<tr>
<td>ExtSum + unsupervised segmentation</td>
<td>48.63</td>
<td>20.41</td>
<td>43.55</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ExtSum + supervised segmentation</td>
<td>49.11(+0.20)</td>
<td>20.68(+0.06)</td>
<td>44.01(+0.16)</td>
<td>49.1(+0.2)</td>
<td>56.5(+0.6)</td>
<td>48.0(+0.4)</td>
</tr>
<tr>
<td>ExtSum + oracle segmentation</td>
<td>49.49(+0.68)</td>
<td>21.04(+0.42)</td>
<td>44.34(+0.49)</td>
<td>49.4(+0.5)</td>
<td>56.8(+0.9)</td>
<td>48.1(+0.5)</td>
</tr>
</tbody>
</table>

Table 5: Evaluation results of using text segmentation in extractive summarization on Arxiv test-set

Figure 2: Relative position of summary sentences in the document.

7 CONCLUSIONS

In this paper, we evaluated two text segmentation methods to detect sections in documents. One unsupervised method that uses similarity scores between adjacent text blocks, and a supervised method that detects sentence by sentence when a new section starts. We combine these predictions in an extractive summarization model to boost its accuracy. We evaluated a series of strategies to combine the segmentation information on summarization. We show that the supervised segment predictions improve the ROUGE scores of the summarization. According to our analysis, segmentation helps to detect the sentences of most relevant sections in the dataset: introduction and conclusions (at beginning and end). However, the maximum improvement we can get using this method is about 1 point ROUGE score, as shown by our experiments using Oracle sections. This improvement is significant but limited. Future work could include more elaborate information from the document, like hierarchical structure, section titles, and discourse information. Also, an end-to-end method to combine both summarization and segmentation.

REFERENCES


