

SciBERTSUM: Extractive Summarization for Scientific Documents

Athar Sefid¹ and C Lee Giles^{1,2}

¹ Computer Science and Engineering, Pennsylvania State University, University, PA, 16802, USA

² Information Sciences & Technology Department, Pennsylvania State University, University Park, PA, 16802, USA

atharsefid@gmail.com, clg20@psu.edu

Abstract. The summarization literature focuses on the summarization of news articles. The news articles in the CNN-DailyMail are relatively short documents with about 30 sentences per document on average. We introduce SciBERTSUM, our summarization framework designed for the summarization of long documents like scientific papers with more than 500 sentences. SciBERTSUM extends BERTSUM to long documents by 1) adding a section embedding layer to include section information in the sentence vector and 2) applying a sparse attention mechanism where each sentences will attend locally to nearby sentences and only a small number of sentences attend globally to all other sentences. We used slides generated by the authors of scientific papers as reference summaries since they contain the technical details from the paper. The results show the superiority of our model in terms of ROUGE scores.³

1 Introduction

Automatic summarization frameworks condense an input document into shorter text consisting of the main points in that document. Neural networks have achieved state of the art results for both paradigms of abstractive summarization [4, 19] and extractive summarization [15, 18]. While extractive models are factually more consistent with the content in the input document, abstractive models can be novel and less redundant. Most of the existing methods are used on news datasets [8, 16] where the input document is relatively short and normally less than 30 sentences long. Summarization of long documents such as scientific papers is different from a short article summarization since it requires more memory and computational power to encode the full document and model the relationship between the sentences.

Natural language processing applications have completely been revolutionized with the advent of pre-trained models. Pre-trained language models are easy to incorporate and don't require much-labeled data to deal with, which makes it appropriate for many problems such as prediction, transfer learning, and feature extraction. Bidirectional Encoder Representations from Transformers (BERT) [6] have combined both word and sentence representations into a single very large Transformer [23]. This has shown superior results on many NLP tasks such as question answering and text generation.

³ The code is available at <https://github.com/atharsefid/SciBERTSUM>.

BERT was trained on large amounts of data with the objective of predicting the masked tokens and the next sentence and it can be fine-tuned for various task-specific objectives [14].

Language models such as BERT [6] or SciBERT [1] have improved many language based tasks, especially with SciBERT for science related documents. The impact of BERT on extractive summarization was due to BERTSUM. BERTSUM extended BERT from a two-sentence language model to one that covers all sentences in a document. BERTSUM model with a full attention layer can capture the document-level features. However, full attention is not efficient for the summarization of long documents such as scientific papers which have more than 500 sentences. Here we propose an extractive transformer based summarizer for longer documents such as scientific articles with multiple sections.

The contributions of our model are:

- Design a section embedding layer to the embedding module of BERTSUM where all tokens in the same section are embedded with the same embedding token. This is crucial for the embedding of long documents with multiple sections in a hierarchical structure.
- Employ a sparse inter-sentence attentional model with local and global attention schemes where each sentence will attend locally to nearby sentences and some random sentences attend globally to all other sentences in the document.
- Devise summarization modules for scientific articles using the presentation slides as the ground-truth summaries. The slides contain the technical details from the paper and usually follow the structure of the papers.

2 Related Work

2.1 Summarization

We believe summarizing scientific articles is more challenging than summarizing generic text since such articles have a hierarchical structure [9]. They contain technical terms and formulas [24], and much valuable content can be embedded in figures, tables, and algorithms [3].

Scientific article summarization has been less investigated compared to news articles summarization [5, 19]. This seems to be mainly due to the lack of training data for full scientific articles. Types of reference summaries for scientific articles are:

- Abstract: Most of the traditional summarization methods use the abstract as the reference summary of the paper. However, abstracts are extremely compressed versions of a papers and usually do not have enough space to include all of the contributions [7].
- Citation-based: These types of summaries integrate the authors' highlights in the abstract of the paper with citation context of the citing papers which in some ways reflects the impact in the paper of the research community [24].
- Speaker Transcript: Many conference proceedings/ workshops require the authors to verbally present their work. TalkSum [10] uses the transcript of these presentations as a summary of the scientific article. However, the transcripts in the TalkSum data set are often noisy and can not be readily used as reference summaries.

Presentation slides for a paper are a different class of summaries that intend to cover in some way the important content of the entire paper, sometimes section by section. They contain the main highlights and also valuable images/tables. They are not as noisy as speaker transcripts and are becoming more available as more conferences are providing slides that go with their papers. We used the PS5k dataset [20, 21] to build our summarizer.

2.2 Transformer Based Summarization

Pre-trained language models such as BART [11] produce state-of-the-art results on the summarization tasks. However, they are often used on short news articles such as XSum [17] or CNN-DailyMail [16] datasets. These models are not designed for scientific articles and their space/computational complexity grows quadratically with the size of the input.

HIBERT [25] is an extractive summarizer that learns context aware sentence representations using multiple layers of transformers. Here, 15% of the full sentences are masked (replaced with a single [mask] token) with the goal to predict the sentence embedding of the masked sentences. BERTSUM [12] is another BERT style extractive summarizer that extends BERT to multiple sentences by expanding the positional embedding and using interval segmentation embeddings to distinguish multiple sentences within a document. Sotudeh et al. [22] added section information to the objective function of BERTSUM so it could optimize both the sentence prediction and section prediction tasks in a multi-task setting. However, most of these transformer-based extractive summarizers do not scale for long documents with thousands of tokens nor can they be applied to many full scientific documents.

3 Method - SciBERTSUM

Most of the previous language models such as BERT are employed as encoders for short pieces of text usually covering at most 2 sentences. The summarization task besides other NLP tasks (e.g. predicting entailment relationship, question answering) requires a wide coverage of the full document containing multiple sections and many sentences. We propose a document encoder based on BERT. Our encoder model will help build sentence representations by stacking multiple transformer layers on top of sentence vectors to model the inter sentence relations in the full document.

Our SciBERTSUM model is an extension of BERTSUM and can generate sentence embeddings for all sentences in a full document with multiple sections. Our model applies a linear sparse attention mechanism between sentences to represent inter sentence relations and it outperforms BERTSUM on our dataset.

4 Language Model Architecture

To explain the architecture of our language model, we first explain how we generate the sentence embeddings by adding section information to sentences and then we explain how our sparse attention mechanism helps us process the full document efficiently.

4.1 Embedding Layer

The embedding layer of BERT [6] applies the byte-pair encoding to tokenize the text. It adds [CLS] tokens to the beginning of the sequences and [SEP] tokens to separate the first and second sentences in the sequence. The embedding representation of the [CLS] token is the representation of the full sequence and is used for sentence classification tasks.

BERT combines 1) Semantic embedding (the meaning of the token) 2) Positional embedding (the position of the token in the sequence), 3) Segmentation embedding (the embedding layer to distinguish the first and the second sentence in the sequence) to form the embedding of a token in a sequence.

BERTSUM [12] extends BERT to multiple sentences by adding [CLS] tokens to the beginning of all sentences. It changes the segmentation embedding to distinguish odd and even sentences. The embedding model is depicted in Figure 1. The green boxes are the segmentation embeddings. Light greens are the embedding for odd sentences and dark green boxes are for even sentences. **BERTSUM extended the positional embedding of BERT beyond 512 tokens to cover all tokens of the input document.**

The sentence embeddings are the embedding of the [CLS] tokens which are the combination of semantic, segment, and position embeddings. The Positional encoding is the sinusoid embedding from Vaswani et. al [23].

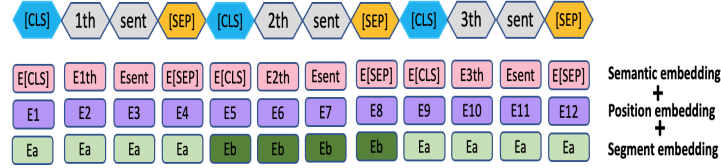


Fig. 1: BERTSUM architecture covering multiple sentence. Each sentence has a [CLS] token at the beginning.

Long documents especially long scholarly articles contain multiple sections. The section of the document is important in the selection of salient sentences. For instance, the sentences in the ‘*acknowledgment*’ section are less important compared to other sections like ‘*abstract*’ or ‘*results*’. **We enhance BERTSUM by adding section embedding as shown in Figure 2.** The sentence embeddings (E_{sents}) are the combination of the section, semantic, position, and segmentation embeddings. The section embeddings are the blue boxes in Figure 2. All of the tokens of the sentences in the first section are embedded by dark blue and the tokens of sentences in the second section are embedded in light blue. Each section has the same segmentation embedding as in BERTSUM.

$$E_{sents} = Semantic + Position + Segment + Section \quad (1)$$

To overcome the memory issue where we cannot load the full document with the full position embedding in the memory. We get the sentence vectors section by section. We

can load the maximum 3072 tokens to the memory based on experiments on an Nvidia GPU with 11,019 MiB memory capacity.

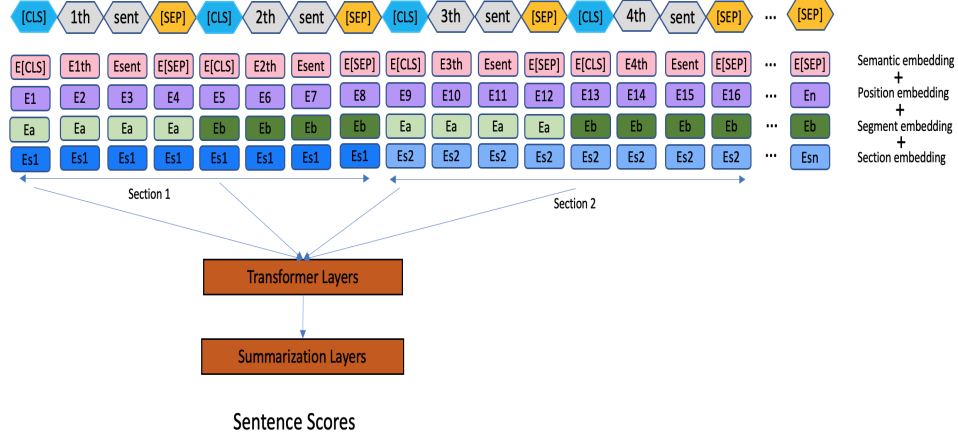


Fig. 2: SciBERTSUM architecture

4.2 Attention Mechanism

BERTSUM applies multiple layers of full dot-product attention. However, applying full attention on the sentence vectors is expensive where the number of sentences in documents is large. Scientific documents can have more than 500 sentences. Therefore, extracting document-level features is expensive for long scientific articles with a full attention layer. Therefore, we introduce a lightweight attention mechanism inspired by LongFormer [2]. The LongFormer language model applies sparse attention between tokens to learn the embedding of the masked tokens. We apply their attention mechanism at the **sentence level** where each sentence will fully attend locally to the nearby sentences and some sentences will attend globally to all sentences in the document.

This attention mechanism will help model select salient sentences locally from the window and at some random and selected positions sentences will attend to all other sentences to identify the salient sentences that are globally important regardless of the section they belong to. The attention window in Figure 3 is 2 which means each sentence will attend to 2 sentences before and after it and in Figure 4 sentences 2 and 7 (marked with *) are attending to all other sentences.

Applying window-based local attention requires a few preprocessing steps. We list the main steps in the following section.

Building the Attention Matrix Since we are processing multiple documents in batch mode and each document has a different length, we fix the number of sentences and

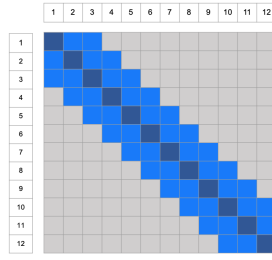


Fig. 3: Local attention

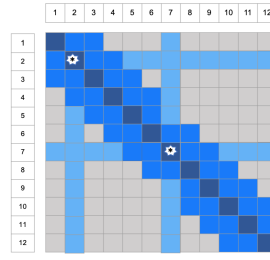


Fig. 4: Local and global attention

make the length of the documents to be a multiple of the attention window. Therefore the following steps are required to process the document:

1. Padding to document size: the document size is fixed to 500 sentences for scientific documents in our corpus.
2. Padding to attention window: the length of the document must be a multiple window size to be able to apply the sliding window attention mechanism.
3. Building attention matrix: the attention matrix has a value of 0 for the padded sentences, a value of 1 for the local attention, and 2 specifies the combination of both local and global attention. Figure 5 shows an attention matrix for a batch of size 3. This batch contains 3 documents with 6, 2, and 6 sentences respectively. For example, the first document attends locally at positions [1, 2, 3, 4, 5, 6] and attends globally at position 4.

	Sent count											
Batch size	[
	1	1	1	2	1	1	0	0	0	0	0	0
	2	1	0	0	0	0	0	0	0	0	0	0
	1	2	1	2	1	1	0	0	0	0	0	0

Fig. 5: Attention Mask Matrix

Calculating Attention Value Here we list the steps for calculating the local attention. The global attention follows the same approach by adjusting the sentences vectors.

1. Three linear layers are applied on the sentence vectors to generate the query, key and value vectors

$$E_q = W_q * E_s + b_q \quad (2)$$

$$E_k = W_k * E_s + b_k \quad (3)$$

$$E_v = W_v * E_s + b_v \quad (4)$$

where E_s is the sentence embedding from the embedding layer that embeds the section information. Here b is the bias term and W matrices are learned in the training phase in order to generate E_q , E_k , and E_v which are respectively the query, key, and value embeddings.

2. For the second step the query is normalized by the square root of the head dimension

$$E_q = E_q / \sqrt{heads}. \quad (5)$$

3. The attention scores are calculated by a sliding query and key matrix multiplication on all chunks of the attention-window size

$$S_{attn}[i] = E_q[i] * E_v[i] \quad (6)$$

where $E_q[i]$ and $E_v[i]$ are the query and value embeddings of window i , and $s_{attn}[i]$ is the attention score for that window.

4. The values of the attention scores at the padding positions are set to 0 to ignore values at these locations so that

$$S_{attn}[padIndex] = 0. \quad (7)$$

5. *Softmax* is applied to attention scores to generate the attention probabilities

$$P_{attn} = Softmax(S_{attn}). \quad (8)$$

6. Finally, the attention probabilities are multiplied by the value vectors chunk by chunk in a sliding window

$$out[i] = P_{attn}[i] * E_v[i]. \quad (9)$$

4.3 Transformer Layer

Our sparse attention mechanism is applied in each layer of transformation [23] and the inputs to the first layer of transformer are the sentence embeddings that include the section information

$$\tilde{h}^l = h^{l-1} + normalize(SparseAttention(h^{l-1})) \quad (10)$$

$$h^l = PositionwiseFeedForward(\tilde{h}^l) \quad (11)$$

where $h^0 = E_{sents}$. We apply a sparse attention mechanism here instead of full attention.

5 Sentence Extractor

To generate the final sentence score, we combined the sentence embedding from the language module with a list of features necessary for the score prediction. Section 5.1 elaborates on the list of features applied to generate the sentence scores. These features depend on the document embedding calculated as in 5.2.

5.1 Sentence Features

The features used to predict the final scores are:

1. Length: number of characters in the sentence i

$$E_{length} = ReLU(Linear(Embedding(length[i]))). \quad (12)$$

2. Position: position of the sentence (i) in the document

$$E_{position} = ReLU(Linear(Embedding(i))). \quad (13)$$

3. Section: section of the sentence i in the document

$$E_{section} = ReLU(Linear(Embedding(section[i]))). \quad (14)$$

Each of the embedding layers is a simple lookup table that stores embeddings of a fixed dictionary and size. The size of the Embedding layers is d . The linear layer applies a linear transformation to the input data x ($y = Wx^T + b$).

4. Correlations: the sentence correlations embed the correlation between sentences. The correlation embeddings help the model to identify sentences with a high degree of correlation to other sentences and then exclude them.

$$Correlation = \tanh(E_{sents} \times W_c \times E_{sents}^T), \quad (15)$$

$$E_{correlation} = ReLU(Linear(Correlation \times E_{sents})) \quad (16)$$

where $W_c \in R^{d \times d}$ is the learned correlation matrix and $Correlation \in R^{n \times n}$ and $E_{Correlation} \in R^{n \times d}$.

5. Saliency: the saliency embedding will embed the importance of sentence vectors with respect to the document embedding. The saliency weight matrix $W_s \in R^{d \times d}$ is learned in the training phase.

$$Saliency = \tanh(E_{sents} \times W_s \times E_D^T), \quad (17)$$

$$E_{Saliency} = ReLU(Linear(Saliency * E_{sents})) \quad (18)$$

where $W_s \in R^{d \times d}$ is the learned saliency matrix and $Saliency \in R^{n \times 1}$. The document embedding (E_D) is the weighted average of the sentence embeddings as explained in 5.2

5.2 Document Embedding

The document encoder is simply the weighted average of the sentence vectors:

$$Weight = Softmax(E_{sents} \times W_{sents}) \quad (19)$$

where $E_{sents} \in R^{n \times d}$ and $W_{sents} \in R^{d \times 1}$. The weights are initialized randomly and will be learned during the training process. Therefore, $Weight \in R^{n \times 1}$ are the weights of the sentences.

Therefore, the embedding of document D is:

$$E_D = \frac{1}{n} \sum_{i=1}^n Weight[i] * E_{sents[i]} \quad (20)$$

where the terms are defined above.

5.3 Score Predictor

The score prediction module concatenates all of the features and feeds them to a linear layer to generate the final scores. Our cross-entropy loss evaluates the difference between the prediction and the ground truth scores. We also evaluated the loss factored by the rewards to see if the model makes better predictions using reinforcement learning (in section 6)

$$p(y_i) = Linear(E_{sent} + E_{length} + E_{position} + E_{section} + E_{correlation} + E_{saliency}) \quad (21)$$

where $p(y_i)$ is the probability of adding sentence i to the summary and the linear layer format is $Wx^T + b$ and $x \in R^{1 \times d}$ and $w \in R^{1 \times d}$.

6 Reinforcement Learning

Ground truth summaries are abstractive summaries that cover the important content from the input documents. Extractive summarization frameworks need conversion of abstractive summaries to extractive 0/1 labels and they maximize the likelihood of the 0/1 ground truth labels for sentences. The Objective function is to minimize this negative log likelihood

$$loss = - \sum_{i=1}^n \log[p(y_i)]. \quad (22)$$

The objective in Eq. 22 maximizes the correct prediction of 0/1 labels where $p(y_i)$ is the probability of label y_i for sentence i . However, the evaluation of summaries are based on the similarity of selected sentences to the abstractive summaries evaluated by ROUGE scores. Therefore, in the training phase we are minimizing the cross entropy loss while in the test phase we evaluate the ROUGE scores [18].

To mitigate the discrepancy between the train and test objectives, Narayan et. al [18] suggest using ROUGE scores in the a reinforced setting to factor the pure cross-entropy loss

$$\nabla loss \simeq -r(y) \sum_{i=1}^n \nabla \log[p(y_i)] \quad (23)$$

where $r(y)$ is the average of ROUGE-F1 and ROUGE-F2 scores.

Since there are multiple collection of sentences or candidate summaries that all could have reasonably high ROUGE scores, they suggest training the model with a selection of good candidate summaries. Therefore, if a candidate summary (made by extractive labels) have high overlap with the abstractive summary, the model wants to predict those labels since the *loss* will be higher.

7 Experimental Results

7.1 Hardware

Three NVIDIA GPUs (GeForce RTX 2080 Ti) with 11019MiB on memory were used. The batch size is set to 1 because of the size of the input document. Since we could not have a large batch size, we accumulate the gradients for 10 steps and then update the parameters. Learning rate is one of the most important hyper-parameters to tune. We used the NOAM scheduler to adjust the learning rate while training and apply gradient-clipping to prevent exploding gradients.

7.2 Experiments

Table 1 shows different values for the size of the local attention window and the ratio of the sentences that will attend globally to all other sentences in the document. The results show that increasing the window size for local attention and global attention ratio will improve the ROUGE recall scores. We can set the size of global and local attentions based on the hardware available at hand. Our model converges faster with a larger attention window and more global attentions.

The effect of reinforcement learning on our model is shown in Table 2. The reinforcement learning does not improve the results on our dataset mainly due to reducing the bias toward the position and length of the sentences.

Our results outperform many of the tested extractive and abstractive models as seen in Table 3. The sentence scores of BERTSUM in Table 3 are generated chunk by chunk since this model is not designed for extractive summarization of long documents. The BART and T5 summaries are generated section by section since they were developed for short sequences and have problems with long documents.

Table 4 shows the effect of trigram blocking in our dataset. If we block adding a sentence if it has a shared trigram, the results are not improved (first row). We also tried allowing some shared trigram in the sentence. For example, the third row in table 4 only blocks sentences if there are more than 5 shared trigrams with the current summary. We see that the results get worse with trigram blocking. It shows that the scores predicted by

the model are good enough to understand if adding a sentence with shared tokens can improve the ROUGE scores.

Table 1: Tuning the local attention window size and ratio of global attentions. The evaluation is based on the ROUGE recall scores. The summary limit is 20% of the size of the input document

Local Window Size	Global Ratio(%)	ROUGE-1	ROUGE-2	ROUGE-L
6	-	56.854	19.692	41.210
10	-	58.854	20.392	41.810
20	-	59.06	20.77	42.00
30	-	58.989	20.664	42.031
40	-	58.97	20.44	41.91
50	-	59.408	21.099	42.232
40	20	59.47	21.11	42.34
40	40	59.72	21.45	42.77
50	20	59.829	21.479	42.973
50	40	59.714	21.498	43.057

Table 2: Applying the reinforcement learning suggested in [18] does not improve ROUGE scores.

Local Window Size	Global Ratio(%)	Reinforced	ROUGE-1	ROUGE-2	ROUGE-L
20	-	No	59.06	20.77	42.00
20	-	Yes	55.27	16.40	38.54
30	-	No	58.989	20.664	42.031
30	-	Yes	55.38	16.57	38.72

8 Conclusions and Future Work

We created an extractive summarization framework, SciBERTSUM, based on BERTSUM for long documents with multiple sections (e.g. scientific papers). We generate sentence vectors based on their sections. The section information is important for the summarization task since sentences in the abstract or method sections are more important compared to the acknowledgement parts. To build a computationally efficient model that scales linearly with the number of sentences in the document, we employed the sparse attention mechanism of LongFormer [2] to embed the inter sentence relations. All sentences attend to a limited number of sentences before and after the current sentence

Table 3: Comparison with baselines based on ROUGE recall.

MODEL	ROUGE-1	ROUGE-2	ROUGE-L
Lead20%	37.68	6.62	15.90
TextRank [13]	38.87	9.28	19.75
SummaRuNNer [15]	45.04	11.67	23.03
BART (section-based)	46.34	11.14	29.85
T5 (section-based)	44.72	10.23	29.63
BERTSUM	52.34	15.06	36.87
SciBERTSUM	59.714	21.498	43.057

Table 4: ROUGE results for tri-gram blocking.

Local Window Size	Global Ratio(%)	Tri-gram count	ROUGE-1	ROUGE-2	ROUGE-L
40	40	tri-grams >0	51.27	14.57	33.34
40	40	tri-grams >3	57.28	19.10	39.49
40	40	tri-grams >5	58.49	20.18	40.90
40	40	no-blocking	59.72	21.45	42.77

and only a small number of random sentences attend globally to all other sentences. Our model is computationally efficient and improves the ROUGE scores on the dataset of paper-slide pairs.

Future work could be applying our model on existing summarization datasets and other long scholarly documents. It would also be interesting to see whether the SciBERT language model, which is pre-trained on scientific text, will give improved performance.

9 Acknowledgement

Partial support from the National Science Foundation is gratefully acknowledged.

References

1. Beltagy, I., Lo, K., Cohan, A.: SciBERT: A pretrained language model for scientific text. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 3615–3620. Association for Computational Linguistics, Hong Kong, China (Nov 2019). <https://doi.org/10.18653/v1/D19-1371>, <https://aclanthology.org/D19-1371>
2. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150 (2020)
3. Bhatia, S., Mitra, P.: Summarizing figures, tables, and algorithms in scientific publications to augment search results. ACM Trans. Inf. Syst. **30**(1) (Mar 2012). <https://doi.org/10.1145/2094072.2094075>, <https://doi.org/10.1145/2094072.2094075>

4. Celikyilmaz, A., Bosselut, A., He, X., Choi, Y.: Deep communicating agents for abstractive summarization. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 1662–1675. Association for Computational Linguistics, New Orleans, Louisiana (Jun 2018). <https://doi.org/10.18653/v1/N18-1150>, <https://aclanthology.org/N18-1150>
5. Cheng, J., Lapata, M.: Neural summarization by extracting sentences and words. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 484–494. Association for Computational Linguistics, Berlin, Germany (Aug 2016). <https://doi.org/10.18653/v1/P16-1046>, <https://www.aclweb.org/anthology/P16-1046>
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). <https://doi.org/10.18653/v1/N19-1423>, <https://www.aclweb.org/anthology/N19-1423>
7. Elkiss, A., Shen, S., Fader, A., Erkan, G., States, D., Radev, D.: Blind men and elephants: What do citation summaries tell us about a research article? *Journal of the American Society for Information Science and Technology* **59**(1), 51–62 (2008)
8. Grusky, M., Naaman, M., Artzi, Y.: Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 708–719. Association for Computational Linguistics, New Orleans, Louisiana (Jun 2018). <https://doi.org/10.18653/v1/N18-1065>, <https://aclanthology.org/N18-1065>
9. Ibrahim Altmami, N., El Bachir Menai, M.: Automatic summarization of scientific articles: A survey. *Journal of King Saud University - Computer and Information Sciences* (2020). <https://doi.org/https://doi.org/10.1016/j.jksuci.2020.04.020>, <https://www.sciencedirect.com/science/article/pii/S1319157820303554>
10. Lev, G., Shmueli-Scheuer, M., Herzig, J., Jerbi, A., Konopnicki, D.: Talksumm: A dataset and scalable annotation method for scientific paper summarization based on conference talks. *arXiv preprint arXiv:1906.01351* (2019)
11. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: 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. pp. 7871–7880. Association for Computational Linguistics, Online (Jul 2020). <https://doi.org/10.18653/v1/2020.acl-main.703>, <https://www.aclweb.org/anthology/2020.acl-main.703>
12. Liu, Y., Lapata, M.: Text summarization with pretrained encoders. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 3730–3740. Association for Computational Linguistics, Hong Kong, China (Nov 2019). <https://doi.org/10.18653/v1/D19-1387>, <https://www.aclweb.org/anthology/D19-1387>
13. Mihalcea, R., Tarau, P.: Textrank: Bringing order into text. In: Proceedings of the 2004 conference on empirical methods in natural language processing. pp. 404–411 (2004)
14. Mosbach, M., Andriushchenko, M., Klakow, D.: On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884* (2020)

15. Nallapati, R., Zhai, F., Zhou, B.: Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In: *Thirty-First AAAI Conference on Artificial Intelligence* (2017)
16. Nallapati, R., Zhou, B., dos Santos, C., Güçlçehre, Ç., Xiang, B.: Abstractive text summarization using sequence-to-sequence RNNs and beyond. In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. pp. 280–290. Association for Computational Linguistics, Berlin, Germany (Aug 2016). <https://doi.org/10.18653/v1/K16-1028>, <https://www.aclweb.org/anthology/K16-1028>
17. Narayan, S., Cohen, S.B., Lapata, M.: Don’t give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium (2018)
18. Narayan, S., Cohen, S.B., Lapata, M.: Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636* (2018)
19. See, A., Liu, P.J., Manning, C.D.: 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)*. pp. 1073–1083. Association for Computational Linguistics, Vancouver, Canada (Jul 2017). <https://doi.org/10.18653/v1/P17-1099>, <https://www.aclweb.org/anthology/P17-1099>
20. Sefid, A., Mitra, P., Giles, L.: Slidegen: an abstractive section-based slide generator for scholarly documents. In: *Proceedings of the 21st ACM Symposium on Document Engineering*. pp. 1–4 (2021)
21. Sefid, A., Mitra, P., Wu, J., Giles, C.L.: Extractive research slide generation using windowed labeling ranking. In: *Proceedings of the Second Workshop on Scholarly Document Processing*. pp. 91–96. Association for Computational Linguistics, Online (Jun 2021). <https://doi.org/10.18653/v1/2021.sdp-1.11>, <https://aclanthology.org/2021.sdp-1.11>
22. Sotudeh Gharebagh, S., Cohan, A., Goharian, N.: GUIR @ LongSumm 2020: Learning to generate long summaries from scientific documents. In: *Proceedings of the First Workshop on Scholarly Document Processing*. pp. 356–361. Association for Computational Linguistics, Online (Nov 2020). <https://doi.org/10.18653/v1/2020.sdp-1.41>, <https://www.aclweb.org/anthology/2020.sdp-1.41>
23. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems* 30, pp. 5998–6008. Curran Associates, Inc. (2017), <http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>
24. Yasunaga, M., Kasai, J., Zhang, R., Fabbri, A.R., Li, I., Friedman, D., Radev, D.R.: Scisumnet: A large annotated corpus and content-impact models for scientific paper summarization with citation networks. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. vol. 33, pp. 7386–7393 (2019)
25. Zhang, X., Wei, F., Zhou, M.: Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. *arXiv preprint arXiv:1905.06566* (2019)