ChunkSumm: Extending BERT for Long Document Summarization

Aman Jaiswal
Faculty of Computer Science
Dalhousie University
Halifax, Canada
aman.jaiswal@dal.ca

Juan Ramirez-Orta
Faculty of Computer Science
Dalhousie University
Halifax, Canada
juan.ramirez.orta@dal.ca

Evangelos Milios
Faculty of Computer Science
Dalhousie University
Halifax, Canada
eem@cs.dal.ca

***Abstract*—In this abstract, we introduce ChunkSumm, a method to produce extractive summaries for documents with length well above the well-established 512 token limit for BERT. We do so by chunking the input into blocks that have the maximum length accepted by BERT and then producing token-level predictions using a combination of the features produced by BERT and 1D-Convolutional layers. Our method can handle full documents with thousands of tokens on a single NVIDIA A100 GPU with ease. We test our method on the TalkSumm data set of extractive summaries of full academic papers and obtain promising results.**

***Index Terms*—extractive summarization, document-level natural language processing, pre-trained deep language models**

I. Introduction

Inside Natural Language Processing (NLP), Automatic Summarization is one of the oldest and most important tasks, which has received continued attention since the creation of the field in the late 50's [1], mainly because of the ever-increasing size of libraries since the advent of digital computers. The objective of the Automatic Summarization task is, given a document, to produce a shorter text with maximum information content, fluency and coherence. The summarization task can be classified into extractive and abstractive: in extractive summarization, the summary is composed exclusively of passages present in the original document; while in abstractive summarization, the summary can contain words that were not present in the original document.

Since the creation of GPT [2], pre-trained deep language models have revolutionized the field of NLP. Models like BERT [3], RoBERTa [4] and GPT-3 [5] have shown impressive generation and reasoning skills that have set new standards in what language models based on neural networks can do. The standard fine-tuning pipeline of the current state of the art in NLP allows to leverage the long training on huge data sets that these models had even in low-resource settings, resulting in impressive results in basically all the tasks across NLP.

However, the main limitation when using these models is their computational requirements: the immense amount of memory and processing they need to be effective restricts their usage when working at the document level and in low-resource devices, like cellphones.

The compute resources for this project were provided by Compute Canada.

In this work, we propose a model that can perform token-level extractive summarization on whole documents. Using a BERT-like model as backbone and combining the features it produces with 1-D Convolutional layers, it can produce fine-grained summaries based on sentences and even passages on standard hardware without sacrificing performance. Our model has the following advantages:

- It can process whole documents while running on standard hardware.
- It produces extractive summaries at the token level, which means that it can extract passages and even phrases from the input that are relevant for the summary.
- It introduces a novel training method based on chunking which can fine-tune pre-trained deep language models on sequences that are well above their input limits.

II. ChunkSumm

The main idea of our method is to fine-tune BERT and combine the features it produces with 1D-Convolutional layers to produce token-level predictions for full documents. However, using this approach directly is computationally unfeasible because full documents are sequences with thousands of words, and training a model like this would need an immense amount of both memory and documents with annotations for every token. To overcome these limitations, we propose a method composed of five steps, described in Figure 1.

A. Tokenization

The first step of our methodology is to transform the raw text into tokens that BERT can understand. This is accomplished with the standard AutoTokenizer for BERT included in the Transformers library [6], which can split documents of arbitrary length into the sub-word units that BERT was trained on and can also turn passages from them back into text.

B. Chunking, BERT Layers and Concatenation

The main contribution of our architecture is chunking the input into segments that can be fitted into BERT. Normally, a full document has thousands of words, so the standard length limit of 512 is the main limitation of the documents that can be processed with BERT. In our method, such a document

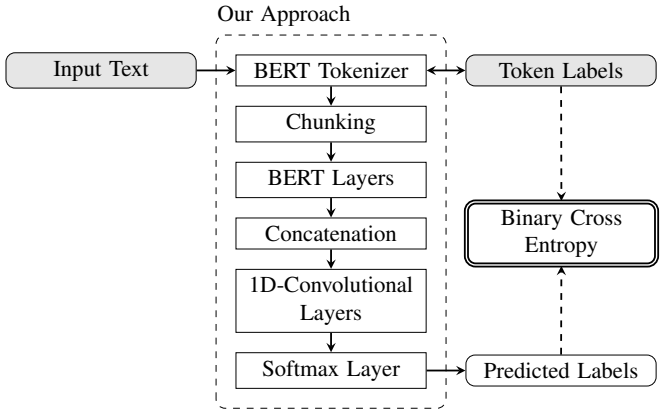

Fig. 1. Overview of our proposed architecture. First, the text is tokenized using the tokenizer included with BERT and the sentence-level labels are propagated to obtain a label for every token. Then, the tokens are chunked into blocks of 512 and processed using the Transformer layers from BERT. After all the chunks have been processed, they are concatenated and processed with 1D-Convolutional layers. The last part of our architecture is a Softmax layer that produces the token labels predicted by the model, which are compared against the true labels using Binary Cross Entropy.

would be split into contiguous, disjoint blocks of length 512, which can be fitted into BERT without problems.

Each one of the token chunks is then processed independently using the backbone of our architecture, which is a BERT-like model. There is no gradient propagation between the chunks, which makes the training stable and computationally feasible. An interesting idea is to decide if the backbone of the architecture should be fine-tuned or not: in our experiments, we found it beneficial to initialize the weights of the backbone by fine-tuning it with an auxiliary task, described below.

Once the chunks has been processed with the backbone model, they are concatenated together to recover the length of the original sequence: in this way, our architecture computes BERT-like features for each one of the tokens in the input using a reasonable amount of computational power without sacrificing performance. This process is shown in Figure 2.

C. Convolutional and Softmax Layers

Once the model has computed the features for each token in the input, the features are combined together using a stack of 1D-Convolutional and Activation layers that are then fed into a Softmax classifier on top of each token. The convolutional layers have a number of hyperparameters, but in our experiments, we found that the kernel size is the one that had the biggest impact during training.

The predictions produced by the Softmax layer can be then compared with the reference token labels via Cross-Entropy and optimized using standard Gradient Descent methods.

III. Experimental Setup

A. Data

The TalkSumm [7] data set is composed of 1,651 academic papers, each one with its extractive summary, which was computed using the fitted model from the paper. The data set is tokenized at the sentence level, which means that the extractive summary for each one of the documents is composed of whole sentences. The papers themselves are not shared in the data set, due to copyright reasons. Instead, they share the URLs where the papers can be downloaded, assuming that one has access to them. Then, the way they recommend to extract the text and sentences from the papers is using *science-parse* [8], which is a neural-based model for extracting and parsing text from scientific papers by AllenAI. We randomly split the data set into three disjoint subsets, shown in Table I.

TABLE I
Number of examples per data set

Data Set	Papers	Sentences
Train	1,001	225,584
Development	150	33,702
Test	500	113,278

B. Training

The proposed architecture can be trained end-to-end using two strategies: using whole sentences or using whole papers. These strategies, which are defined by the data units presented to the model during training, specify the number of data points observed and the potential number of optimization steps.

In sentence-level training, the objective is to predict if the tokens of a given sentence should be included in the summary or not, which is optimized using the Binary Cross Entropy between the predictions and the token labels of independent sentences of the paper. The main idea behind this strategy is that the words of the sentences that are included in summary are very different from the ones that are left out, even when just looking at isolated sentences. The main advantage of this is to bypass the computational overhead of chunking the complete document (which can contain thousands of tokens), but it has the drawback that the order of sentences in not preserved, which can make the task impossible to solve for some sentences.

On the other hand, in paper-level training, the objective is to processes the complete document while preserving the order of occurrence and outline of the document. This strategy is much more expensive than training at the sentence-level, but it takes into account the full context for every token selected in the summary. We hypothesize that this mode of training should benefit summarization tasks which require analyzing the whole structure of long, full documents. For example, when summarizing an academic paper, important extractions can be found in the abstract and conclusion, which are very far away from each other in the input sequence.

C. Experimental Results

We experimented with the four different variations derived from the mode of training and the state of backbone model, which could be either frozen or trainable. The number of trainable parameters for each model are shown in Table II.

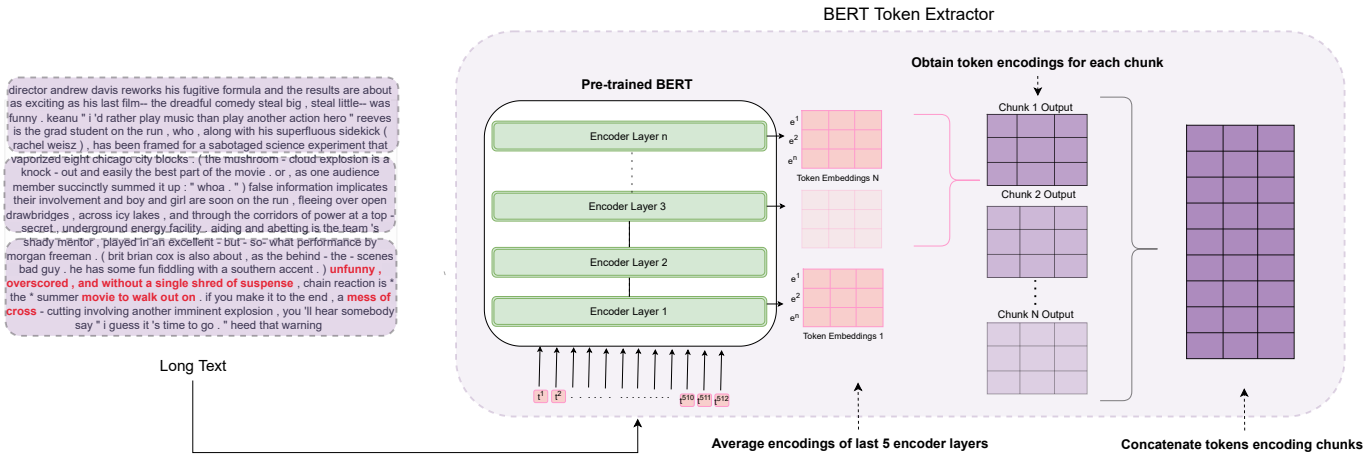

Fig. 2. Overview of our chunking methodology. First, the complete input is tokenized using the BERT tokenizer and split into chunks of size 512 tokens. Sequentially, The input chunks are processed by BERT to produce token embeddings for each chunk. Here, we utilize the average token embeddings of the last 5 layer of the BERT. Once every chunk is processed independently, they are concatenated over the token(time) dimension to produce the complete input embedding matrix.

TABLE II
NUMBER OF TRAINABLE PARAMETERS

Model	# Parameters BERT	# Parameters Convolutions	# Parameters Total
ChunkSumm-Paper-Frozen	–	19M	19M
ChunkSumm-Sentences-Frozen	–	19M	19M
ChunkSumm-Paper	128M	19M	147M
ChunkSumm-Sentences	128M	19M	147M

The ROUGE-1, ROUGE-2, ROUGE-L and AUC scores of all the models on the test set are shown in Table III. In paper-level training, the summarization performance improved significantly when the BERT parameters were fine-tuned along with the convolution parameters. We also observed that Sentence-level training can match the performance of paper-level training without the need for fine-tuning the BERT parameters. This suggests that we can obtain better summarization performance while training 128M fewer parameter and training with a strategy that uses significantly less memory.

TABLE III
RESULTS

Model	R-1	R-2	R-L	AUC
ChunkSumm-Paper-Frozen	0.02	0.00	0.02	0.89
ChunkSumm-Sentences-Frozen	**0.48**	**0.31**	**0.47**	0.78
ChunkSumm-Paper	0.45	0.26	0.42	0.89
ChunkSumm-Sentences	**0.48**	0.30	0.46	**0.91**

IV. CONCLUSION AND FUTURE WORK

Our results show that we can efficiently fine-tune BERT for long-text summarization using chunking and convolutions. Our method produces accurate token-level predictions for the complete document even while training only on isolated sentences.

For future work, we would like to explore more sophisticated strategies of aggregating the token probabilities to produce summaries.

REFERENCES

[1] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research and Development*, vol. 2, no. 2, pp. 159–165, 1958.

[2] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-training," *https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf*, 2018.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018. [Online]. Available: https://arxiv.org/pdf/1810.04805.pdf

[4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019. [Online]. Available: https://arxiv.org/pdf/1907.11692.pdf

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[6] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-Art Natural Language Processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[7] G. Lev, M. Shmueli-Scheuer, J. Herzig, A. Jerbi, and D. Konopnicki, "TalkSumm: A dataset and scalable annotation method for scientific paper summarization based on conference talks," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2125–2131. [Online]. Available: https://www.aclweb.org/anthology/P19-1204

[8] AllenAI, "Science parse," GitHub repository, https://github.com/allenai/science-parse, October 2019, visited on April 23, 2021.