Saucissonnage of Long Sequences into a Multi-encoder for Neural Text Summarization with Transformers

Jessica López Espejel, Gaël de Chalendar, Jorge Garcia Flores, Ivan Vladimir Meza Ruiz, Thierry Charnois

Saucissonnage of Long Sequences into a Multi-encoder for Neural Text Summarization with Transformers

Jessica López Espejel[*,**], Gaël de Chalendar[*]
Jorge Garcia Flores[**], Ivan Meza[***], Thierry Charnois[**]

[*]Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
{jessica.lopez-espejel, gael.de-chalendar}@cea.fr,
[**]CNRS-LIPN-Université Paris 13, France
{jgflores, charnois}@lipn.univ-paris13.fr
[***]IIMAS, UNAM, Mexico
ivanvladimir@turing.iimas.unam.mx

Abstract. Transformer deep models have gained lots of attraction in Neural Text Summarization. The problem with existing Transformer-based systems is that they truncate documents considerably before feeding them to the network. In this paper, we are particularly interested in biomedical long text summarization. However, current input sequences are far shorter than the average length of biomedical articles. To handle this problem, we propose two improvements to the original Transformer model that allow a faster training of long sequences without penalizing the summary quality. First, we split the input between four encoders to focus attention on smaller segments of the input. Second, we use end-chunk task training at the decoder level for progressive fast decoding. We evaluate our proposed architecture on PubMed, a well-known biomedical dataset. The comparison with competitive baselines shows that our approach: (1) allows reading large input sequences, (2) reduces the training time considerably, and (3) slightly improves the quality of generated summaries.

1 Introduction

Recently, Transformer neural networks (Vaswani et al., 2017) have become the predominant model for a wide variety of *Natural Language Processing* (NLP) tasks. Transformers are based on an attention mechanism that allows focusing on the positional value of each element of a sequence. Contrarily to *Long Short Term Memory* (LSTMs) proposed by Hochreiter and Schmidhuber (1997), Transformers are not recurrent and allow reading input sequences in a parallel way. Transformers help avoiding the vanishing gradient problem, thus outperforming LSTMs and many other architectures for Deep NLP tasks (Wood, 2020). The initially proposed transformer architecture is based on an encoder and a decoder. The encoder represents both the input words sequence and the positional information in a single encoding vector to be fed later into the decoder. The latter transforms the encoded vector into a sequence of per-word probabilities.

In this work, we propose an improvement of the Transformers model for *Neural Text Summarization* (NTS), addressing two main problems in this task. The first one is the training time of the model. Depending on the number of available GPUs, this phase could take from few days to few weeks, even with transformers' ability to process sequential input parallelly. The second one is the size of the encoded document (i.e., input word sequence). With the birth of deep learning, the length of the input sequence (L_{input}) fed into the model has been significantly reduced compared to extractive predecessor systems from the 2000s (García Flores et al., 2009). For example, some LSTM-based summarization systems (Cohan et al., 2018; See et al., 2017) truncate the source document to 2000 and 400 tokens, respectively. The maximum input length for a Transformer-based NTS system is $L_{input} = 1024$ tokens (Zhang et al., 2020). However, scientific articles are much larger. For instance, the biomedical dataset built by Cohan et al. (2018) has, on average, 3016 tokens in each article.

Our main contribution is to modify the transformers model to reduce the training time while increasing the input document size for summarization. To do so, we propose improving the original Transformer architecture by increasing the number of encoders from 1 to 4 and secondly splitting the input sequence between them. Experiments show that our architecture reduces training time without penalizing the quality of generated summaries. Our second contribution intervenes at the decoding level. Instead of presenting the gold summary to the decoder all at once, we present it chunk by chunk to learn progressively. This contribution is inspired by Hoang et al. (2019) that presents the gold summary to the decoder token by token. Our approach makes a compromise between learning progressively and fast.

In French, *saucissonner* means to "slice" a dry-cured sausage. In their original Transformer paper (Vaswani et al., 2017), the authors noted that self-attention models tend to perform better with shorter input sequences. Our approach's underlying rationale is to "slice" (*saucissonner*) longer sequences. It distributes the slices between the encoders to reduce the training time and consider a larger part of the documents to summarize. Our results show that our approach: (1) summarizes documents having up to $L_{input} = 2000$ tokens, (2) reduces the training time process, and (3) produces summaries with higher scores than the original Transformer architecture.

The rest of the paper is organized as follows. In Section 2, we describe the most popular recurrent and transformers based approaches for the NTS task. Section 3 presents our improved version of the original Transformer to train faster by processing longer source documents. Section 4 presents the dataset used for evaluation, baselines, experimental framework, and the implementation details. Section 5 provides an insight into the results, with a comprehensive discussion. Finally, we conclude in Section 6 and provide some perspectives.

2 Related work

Automatic summarization is an NLP task whose objective is to automatically produce a summary concentrating the most important information from a long source document or a document collection (Mani, 2001). Extractive summaries simply copy the most relevant sentences from the source document, while abstractive summaries reformulate and paraphrase the main information from the source document into new sentences.

The first approaches based on extractive summarization, assume that the most important words are those repeated most frequently (Luhn, 1958; Sparck, 1972). The most important such methods include probabilistic models such as *Probabilistic Context-Free Grammars* (PCFG) (Rahman et al., 2001; Knight and Marcu, 2002), *Markov Models* (MM), and *Hidden Markov Models* (HMM) (Chen and Withgott, 1992; Jing and McKeown, 1999; Conroy and O'leary, 2001).

Automatic summarization evolved later, and extractive based methods relied on machine learning to tackle the NTS as a classification problem, where new techniques were used, such as Naive Bayes (Thu, 2014; Ramanujam and Kaliappan, 2016), Clustering (ShivaKumar and Soumya, 2015), and *Support Vector Machine* (SVM) (Schilder and Kondadadi, 2008; Begun et al., 2009). However, work is still needed to improve the automatic generation of summaries, especially with the new challenges that arose along with neural networks.

The Deep Learning methodological leap has shifted research efforts into abstractive models based on neural architectures. Deep models require a considerable amount of data for efficient training and high GPU computational resources. Therefore, abstractive NTS has been gaining momentum (See et al., 2017; Cohan et al., 2018) with the introduction of *Recurrent Neural Networks* (RNNs), in particular LSTMs (Hochreiter and Schmidhuber, 1997) and *Gated Recurrent Units* (GRUs) (Cho et al., 2014). For instance, Rush et al. (2015) introduced pioneer research to generate abstractive summarization at the sentence level. Further, Nallapati et al. (2016) adapted an off-the-shelf attentional encoder-decoder RNN that was initially proposed for translation (Rush et al., 2015) to tackle the summarization task. In recent years, Transformers Neural Networks (Vaswani et al., 2017) has been gaining increased interest for summarization in the scientific community. Therefore, a new wave of pre-trained NTS systems saw the light (Narayan et al., 2018; Kim et al., 2019; Zhang and Tetreault, 2019; Fabbri et al., 2019; Zhang et al., 2020). Inspired by Transformers, several language models usable for automatic summarization have been developed, such as BERT (*Bidirectional Encoder Representations from Transformers*) (Devlin et al., 2019), MASS (Song et al., 2019), UniML (Dong et al., 2019), T5 (Raffel et al., 2020), etc.

As stated above, our concern in this work is to reduce the training time while making it possible to summarize longer source documents in the context of a basic Transfomer NTS system without any pre-training whatsoever. Therefore, we compare our system with two state-of-the-art approaches. The first one was proposed by Cohan et al. (2018). It is an LSTM-based approach with the longest size of summarized source documents ($L_{input} = 2000$ tokens). Authors of the same paper introduced the PubMed dataset of biomedical documents, having each an average of 3016 tokens. We used this dataset for our experiments insofar length of documents is convenient to test our multi-encoder model on long sequences. The second approach is called PEGASUS (*Pre-training with Extracted Gap-sentences for Abstractive Summarization*) (Zhang et al., 2020). PEGASUS is a Transformer Neural Network pre-trained on massive text corpora. It is based on an encoder and a decoder architecture. We compare our approach with the non-pre-trained $Transformer_{BASE}$ model from the PEGASUS paper. For the sake of comparability, we also do not pre-train our model in order to explicitly evaluate the modified Transformer architecture.

3 Segmentation of long sequences into a multi-encoder

3.1 The original Transformer network

Our Transformer baseline (presented in Figure 1) was introduced by Vaswani et al. (2017). It is a deep neural network based on an attention mechanism (Equation 1) and is based on an encoder and a decoder. The encoder is a set of six layers, where each layer contains two sub-layers: a multi-head attention layer and a feed-forward network. The decoder also has six layers but is different from the encoder in two aspects. First, it has an additional multi-head attention sub-layer, and second, the self-attention sub-layer is modified to avoid attending subsequent positions. Each sub-layer of the encoder and the decoder is followed by a residual connection and a normalization layer. Below, we call this baseline $Transformer_{ORIGINAL}$.

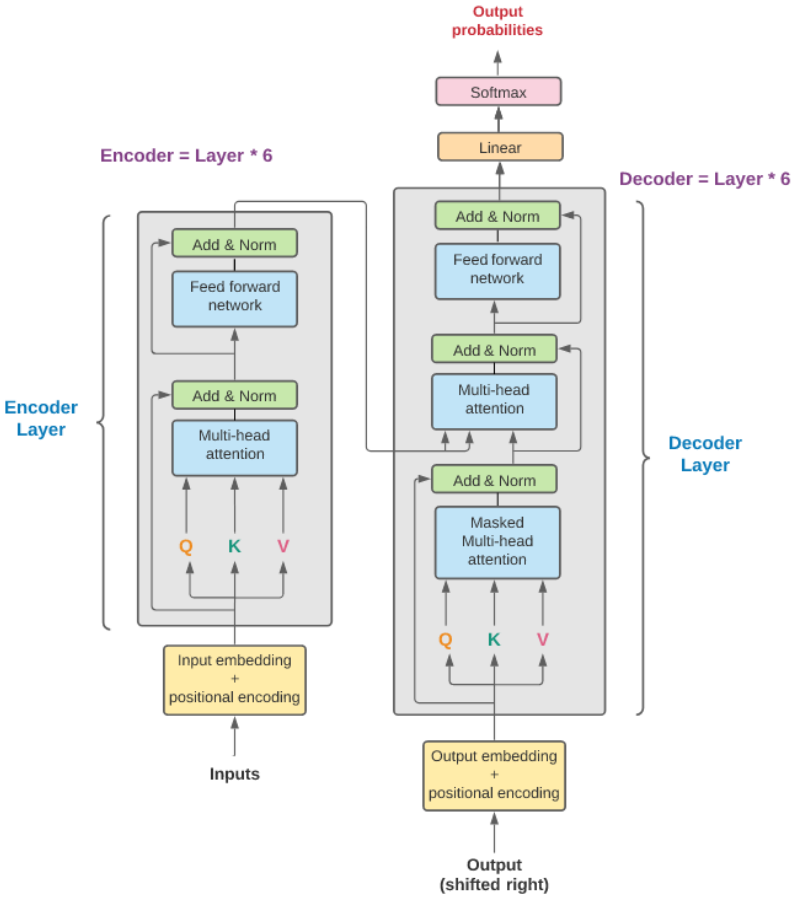

FIG. 1 – *Transformer architecture (Vaswani et al., 2017)*

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

where Q are queries, K are keys, V are values, and d_k is the dimension of Q and K.

There are three different ways to handle multi-head attention in Transformers: (1) at the encoding level only, (2) at the decoding level only, or (3) at both the encoding and the decoding levels. Multi-head attention is a set of self-attentions (heads) concatenated as shown in Equation 2.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \quad (2)$$

where:

- h is the number of heads
- W^O is the weight matrix of Queries (Q), Values (V), or Keys (K).
- $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

3.2 The multi-encoder Transformer model

Our contributions to the Transformers model are organized in two parts. First, we explain the slicing of long sequences for multi encoding (Section 3.2.1), and later, we present the end-chunk task training improvement (Section 3.2.2).

3.2.1 Slicing long sequences for multi-encoding

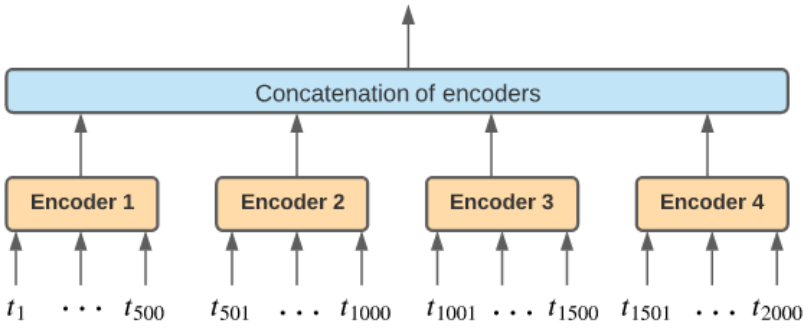

FIG. 2 – *Segmentation of a long input sequence ($L_{input} = 2000$ tokens) and feeding it to the multi-encoder layer*

Figure 2 shows our improved architecture of the original Transformer model that targets long input sequences. Instead of feeding the whole sequence to one encoder, as in the original Transformer, we split the input into four chunks and feed each chunk to a different encoder. Contrarily to $Transformer_{ORIGINAL}$ that uses one encoder with eight self-attention heads for the whole input, we use four encoders with eight self-attention heads each (see Figure 3). This choice is motivated by works such as Fabbri et al. (2019) and Zhang et al. (2020) that achieved competitive results compared to state of the art by using sequences having $L_{input} =$

500 and $L_{input} = 512$ tokens as input for the encoder, respectively. Since we experiment with 2000 tokens long sequences, it is fair to use four encoders, each with $L_{input} = 500$ tokens.

This improvement tries to cope with Transformer's attention under-performance with long input sequences. Four encoders with eight multi-attention heads each would improve the Transformer processing of long sequences and reduce the training time without penalizing the quality of generated summaries (see Section 5). As it is clearly stated in the original Transformer paper (Vaswani et al., 2017), self-attention models tend to perform better with shorter token sequences. Thus, we suppose that splitting long sequences and distributing them in the multi-encoder layer would reduce the training time and improve the processing of long source documents. Our goal here is to measure the impact of a substantial modification in the Transformer architecture before escalating to an intensive pre-training scenario, like in the PEGASUS approach (Zhang et al., 2020). The underlying rationale of both improvements from this section is to better focus the model's attention on shorter sequences with this *saucissonnge* process.

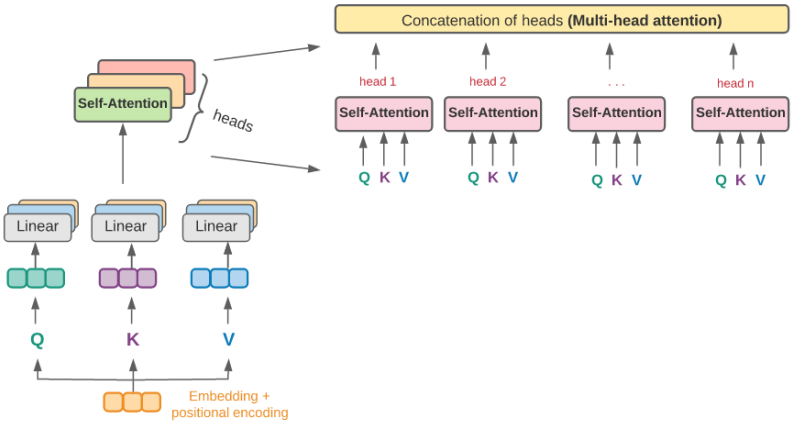

FIG. 3 – *Multi-head attention in each encoder*

Figure 3 shows multi-head attention layers in each of the four encoders of our multi-encoder architecture. This input of multi-head attention encoders is a slice of the source text's embeddings and positional encodings. This input can be linearly transformed to get queries (Q), keys (K), and values (V) matrices from the original mono-encoder Transformer model. Each attention head uses different linear transformations to represent words. For this reason, different heads can learn different relationships between words. As a consequence of using a multi-encoding schema with four encoders, our model contains 32 attention heads compared to 8 in the original Transformer. We hypothesize that this increase in the number of attention heads would lead our multi-encoder Transformer to learn different relationships between words. While most of the Transformer models read 512 or at most 1024 (Zhang et al., 2020) tokens, we use input sequences having $L_{input} = 2000$ tokens, which is closer to the average size of 3016 tokens of medical documents from the PubMed dataset (Cohan et al., 2018). However, this multi-encoding schema is only applied at the encoder level, not at the decoder level, where sequences are short enough to generate summaries of the desired length (216 tokens long).

3.2.2 End-chunk task training

End task training was introduced by Hoang et al. (2019). The goal of their approach was to adapt a generic pre-trained text generation Transformer to the NTS task. End task training is an extra training step that aims to constrain the neural network's loss function to maximize the log-likelihood probability of generating pertinent summary given the reference summary. We adapted the equation from Hoang et al. (2019) to receive sequences of tokens (or chunks) instead of a token-by-token flow. We call this improvement *end-chunk task training (ECTT)*. While the approach of Hoang et al. (2019) feeds increasing 1-token differential sentences, we feed chunks of cs (chunk size) tokens. The new loss function $\mathcal{L}_{ECTT}$ is provided in Equation 3.

$$\mathcal{L}_{ECTT} = -\sum_{i=0}^{cn-1} log\ P(\{x^s\}_0^{(i+1)\times cs-1}|\{x^a\}_0^{M-1}) \tag{3}$$

where:

- M is the number of tokens in the article
- cn is the number of chunks into which the summary is divided
- cs is the chunk size, such that $cs = N/cn$, where N is the number of tokens in the summary
- x^s is a token from the summary
- x^a is a token from the article
- $\{x^s\}_0^{(i+1)\times cs-1}$ is the summary chunk, counting from the first token (0) until the token number $(i+1) \times cs - 1$
- $\{x^a\}_0^{M-1}$ is the input article

4 Experimental framework

4.1 Evaluation dataset

Experiments were done using PubMed, a dataset collected by Cohan et al. (2018) from the well-known PubMed scientific repository (`PubMed.gov`). This dataset is composed of 130397 documents, where 117108 are in the training set, 6631 documents are in the validation set, and 6658 documents are in the test set. We used the validation set to tune hyper-parameters during the training process and the test set to get the final summaries.

4.2 Baselines

Experiments have been conducted with strong baselines, described as follows:

- $\mathbf{Transformer_{ORIGINAL}}$ - This baseline is a mono-encoder architecture described in Subsection 3.1. Inspired by Gehrmann et al. (2018), we use 4 layers contrarily to the initially proposed 6-layers architecture (Vaswani et al., 2017).

- $\mathbf{Transformer_{BASE}}$ - is the version of $PEGASUS_{BASE}$ without pre-training (Zhang et al., 2020). The architecture of this model has $L = 12$, $H = 768$, $F = 4096$, and $A = 16$, where L is the number of layers in the encoder and the decoder, H is the hidden

size, F is the feed-forward layer size, and A is the number of self-attention heads.

- **LSTM** - is an NTS system proposed by Cohan et al. (2018). It is based on a hierarchical encoder to model the discourse structure of documents. Both the encoder and the decoder are implemented as Long Short Term Memory (LSTM) networks.

4.3 Implementation details and evaluation metric

In our experiments, we truncate scientific documents to their first 2000 tokens to evaluate our model's performance on long sequences. Generated summaries are 216 tokens long, and only the most frequent 100,000 tokens are kept in the vocabulary.

We implemented $Transformer_{ORIGINAL}$ and our multi-encoder Transformer with Keras (Chollet et al., 2015). However, we report results of $Transformer_{BASE}$ from the Table 2 of the PEGASUS paper. We trained the models on 8 GPUs of Nvidia Quadro P5000 with 16GB of RAM capacity. We used sinusoidal positional encoding following Vaswani et al. (2017). For optimization, we used Adam algorithm with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $batch\ size = 32$. Besides, we used beam search with $\alpha = 0.8$ and $beam\ size = 6$.

The $Transformer_{ORIGINAL}$ and our multi-encoder Transformer were trained for 300 epochs. The end-chunk stage lasts ten epochs in both approaches. The optimal chunk size in this stage was empirically set to $cs = 27$ tokens.

We used ROUGE (Lin, 2004) approach to evaluate generated summaries and thus compare the proposed multi-encoder architecture with our implementation of $Transformer_{ORIGINAL}$ and $Transformer_{BASE}$ from the PEGASUS paper. The ROUGE method is based on lexical overlaps between tokens and phrases in the reference (human-written) summaries and the generated ones. We report scores with ROUGE-1 (unigrams), ROUGE-2 (bigrams), and ROUGE-L (longest common sub-sequence). Equally, we report the training time of the multi-encoder architecture versus the mono-encoder one to compare their performance in terms of computational time. Note that we do not report the training time of $Transformer_{BASE}$ (Zhang et al., 2020) since it was trained on TPUs instead of GPUs, and the authors did not provide any information about execution time in their paper.

5 Results and discussion

Results in Table 1 show that our multi-encoder model outperforms the $Transformer_{BASE}$ from the PEGASUS paper (Zhang et al., 2020) and $Transformer_{ORIGINAL}$ from Vaswani et al. (2017) in terms of ROUGE-1 and ROUGE-2 scores, while it gives slightly lower scores in terms of ROUGE-L compared to $Transformer_{BASE}$.

Our model can read sequences with $L_{input} = 2000$ tokens. This number is closer to the average size of a standard biomedical article (3016 tokens) than the input length used by other state-of-the-art approaches. To the best of our knowledge, there is no Transformer-based NTS system able to read such length in the biomedical domain. The longest input text was used by $PEGASUS_{LARGE}$ (Zhang et al., 2020), a pre-trained model with $L_{input} = 1024$ tokens. Note that the non-pre-trained $Transformer_{BASE}$ uses sequences of length $L_{input} = 512$.

The LSTM-based approach (Cohan et al., 2018) (called $LSTM$ in Subsection 4.2) is still the best NTS system producing summaries with the highest ROUGE scores. This system can process 2000 tokens source documents, with ROUGE scores of R1/R2/RL = 39.93/15.37/35.21.

	$Transformer$ $ORIGINAL$ + end-chunk	Multi-encoder Transformer + end-chunk (ours)	$Transformer$ $BASE$
Number of encoders	1	4	1
Number of layers	4	4	12
Input length (L_{input})	2000 tokens	2000 tokens	512 tokens
Training time per epoch	58mn21s	33mn15s	NA
ROUGE-1	32.7	**34.11**	33.94
ROUGE-2	7.11	**7.68**	7.43
ROUGE-L	18.14	18.56	**19.02**

TAB. 1 – *ROUGE scores of the $Transformer_{ORIGINAL}$, the multi-encoder Transformer (ours), and $Transformer_{BASE}$ from the PEGASUS paper. Best results are in bold.*

Automatic summary generated with our approach

```
urothelial carcinoma of the bladder is a rare and aggressive
malignancy with a poor prognosis . there are only a few re-
ports of radical cystectomy for bladder carcinoma . we report
the case of a 61 year old male patient who presented to our
urology clinic with an initial diagnosis of bladder cancer ,
the patient was treated with radical nephroureterectomy and
adjuvant chemotherapy . the outcome was excellent for all pa-
tients and there was no evidence of local recurrence or distant
metastases . in this article we review the current litera-
ture on the use of neoadjuvant chemotherapy as the first line
treatment for patients presenting with bladder carcinomas
```

Human reference summary for the same scientific article

```
small cell carcinoma of the urinary bladder is an extremely
aggressive and rare tumor . even though small cell carcinoma
most commonly arises from the lungs there are several reports
of small cell carcinoma in extrapulmonary sites . due to its
low frequency there is no well - established management for
this disease . we report the case of a 61 year - old man with
small cell carcinoma of the bladder who underwent radical cys-
tectomy following neoadjuvant chemotherapy . we also reviewed
the literature for the optimal treatment strategy
```

Our multi-encoder Transformer model runs almost two times faster than the original Transformer baseline ($Transformer_{ORIGINAL}$). According to our quantitative results and our qualitative analysis of the generated summaries, using a multi-encoder layer in the transformer neural network architecture helps capturing the most important tokens from the input. This finding is intuitive insofar we increase the number of head attentions when using more than an encoder at a time. In general, each encoder has eight self-attention heads. However, in our experiments, we have four encoders, leading to a total of 32 self-attention heads. Therefore, with more self-attentions, the encoders of the model are able to capture the most relevant tokens from the article, while we use a different number of self-attentions in the decoder.

Equally important, the second stage of training, called end-chunk task training, improves our summaries' quality. Indeed, learning chunk by chunk helps feeding information progressively and relatively fast to the decoder. This training is inspired by how humans get knowledge incrementally over time.

6 Conclusion and further work

In this work, we proposed two improvements for Neural Text Summarization using Transformers (Vaswani et al., 2017), intended to improve the model's performance with long input sequences and reduce training time without penalizing the quality of the generated summaries. The first contribution is to use a multi-encoder and to *saucissonner* (or slice) a long input sequence to process each slice with one of the four encoders composing our multi-encoder architecture. The second contribution is to modify the End Task Training technique (Hoang et al., 2019) at the decoder level to process phrase chunks instead of individual words.

We evaluate these improvements on the medical articles summarization task, using the PubMed (Cohan et al., 2018) dataset. Results show that our improved model was able to:

1. Reduce the training time to almost half compared to the original Transformer model,
2. Extend the size of the input source document to almost the double of the state-of-the-art Transformer-based summarization system (Zhang et al., 2020)
3. Slightly improve the ROUGE scores of lexical similarity between the generated summaries and their corresponding human reference abstracts

Pre-training was intentionally left aside to analyze the exact impact of our improvements in the model. Further experiments will aim to pre-train our model with a large medical dataset (such as the Covid-19 virus corpus (House, 2020)), and feed 2000 tokens long source documents to establish a comparison with other pre-trained models such as $PEGASUS_{BASE}$ and $PEGASUS_{LARGE}$ (Zhang et al., 2020), whose longest length source document is 512, and 1024 tokens, respectively. Both PEGASUS models are pre-trained on large text corpora. Furthermore, a more detailed ablation analysis will be done to evaluate each contribution independently.

References

Begun, N., M. A. Fattah, and F. Ren (2009). Automatic text summarization using support vector machine. *International Journal of Innovative Computing, Information and Control 5*, 1987–1996.

Chen, F. R. and M. Withgott (1992). The use of emphasis to automatically summarize a spoken discourse. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 229–232.

Cho, K., D. Bahdanai, F. Bougares, H. Schwenk, and Y. Bengio (2014). Learning phrase representation using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734. Association for Computational Linguistics.

Chollet, F. et al. (2015). Keras. `https://github.com/fchollet/keras`.

Cohan, A., F. Dernoncourt, D. S. Kim, T. Bui, S. Kim, W. Chang, and N. 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)*, New Orleans, Louisiana, pp. 615–621. Association for Computational Linguistics.

Conroy, J. M. and D. P. O'leary (2001). Text summarization via hidden markov models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, New York, NY, USA, pp. 406–407. Association for Computing Machinery.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019). 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)*, Minneapolis, Minnesota, pp. 4171–4186. Association for Computational Linguistics.

Dong, L., N. Yang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon (2019). Unified language model pre-training for natural language understanding and generation. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*.

Fabbri, A., I. Li, T. She, S. Li, and D. Radev (2019). Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, pp. 1074–1084. Association for Computational Linguistics.

García Flores, J., O. Ferret, and G. de Chalendar (2009). Summarizing through sense concentration and contextual exploration rules: the CHORAL system at TAC 2009. In *Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*.

Gehrmann, S., Y. Deng, and A. M. Rush (2018). Bottom-up abstractive summarization. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii (Eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 4098–4109. Association for Computational Linguistics.

Hoang, A., A. Bosselut, A. Çelikyilmaz, and Y. Choi (2019). Efficient adaptation of pretrained transformers for abstractive summarization. *CoRR abs/1906.00138*, online.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural Computation 9(8)*, 1735–1780.

House, T. W. (2020). Covid-19 open research dataset challenge (cord-19).

Jing, H. and K. McKeown (1999). The decomposition of human-written summary sentences. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pp. 129–136. Association for Computing Machinery.

Kim, B., H. Kim, and G. Kim (2019). Abstractive summarization of Reddit posts with multi-level memory networks. 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)*, Minneapolis, Minnesota, pp. 2519–2531. Association for Computational Linguistics.

Knight, K. and D. Marcu (2002). Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence 139*, 91–107.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, Barcelona, Spain, pp. 74–81.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research Development 2*(2), 159–165.

Mani, I. (2001). *Automatic Summarization*. John Benjamins Publishing.

Nallapati, R., B. Zhou, C. Dos Santos, and B. Xiang (2016). 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.

Narayan, S., S. B. Cohen, and M. Lapata (2018). 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.

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. 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.

Rahman, A. F. R., H. Alam, R. Hartono, and K. Ariyoshi (2001). Automatic summarization of web content to smaller display devices. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pp. 1064–1068.

Ramanujam, N. and M. Kaliappan (2016). An automatic multidocument text summarization approach based on naïve bayesian classifier using timestamp strategy. In *TheScientificWorld-Journal*.

Rush, A., S. Chopra, and J. Weston (2015). A neural attention model for sentence summarization. *Association for Computational Linguistics Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 379–389.

Schilder, F. and R. Kondadadi (2008). Fastsum: Fast and accurate query-based multi-document summarization. In *ACL*.

See, A., P. Liu, and C. Manning (2017). Get to the point: Summarization with pointer-generator networks. *Trans. Amer. Math. Soc. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1073–1083.

ShivaKumar, K. and R. Soumya (2015). Text summarization using clustering technique and svm technique. *International Journal of Applied Engineering Research 10*, 25511–25519.

Song, K., X. Tan, T. Qin, J. Lu, and T.-Y. Liu (2019). Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pp. 5926–5936.

Sparck, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation 28(1)*, 11–21.

Thu, H. (2014). An optimization text summarization method based on naive bayes and topic word for single syllable language. *Applied mathematical sciences 8*, 99–115.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc.

Wood, T. (2020). Transformer neural network. `https://deepai.org/machine-learning-glossary-and-terms/transformer-neural-network`. Accessed: 2020-12-12.

Zhang, J., Y. Zhao, M. Saleh, and P. J. Liu (2020). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. `https://arxiv.org/abs/1912.08777`.

Zhang, R. and J. Tetreault (2019). This email could save your life: Introducing the task of email subject line generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, pp. 446–456. Association for Computational Linguistics.

Résumé

Cet article présente une amélioration d'une approche de résumé automatique de textes avec des réseaux de neurones de type Transformers. Notre méthode consiste à augmenter le nombre d'encodeurs du modèle en découpant l'entrée entre eux afin de concentrer l'attention du modèle sur des sous parties du texte. En plus, notre méthode favorise l'apprentissage progressive en présentant les résumés au décodeur partie par partie jusqu'à la consommation de toute la séquence. Les résultats obtenus sont encourageants en comparant avec des méthodes compétetive de l'état de l'art.