Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.037083

ARTICLE

A Method of Integrating Length Constraints into Encoder-Decoder Transformer for Abstractive Text Summarization

Ngoc-Khuong Nguyen[1,2], Dac-Nhuong Le[1], Viet-Ha Nguyen[2] and Anh-Cuong Le[3,*]

[1]Faculty of Information Technology, Haiphong University, Haiphong, 180000, Vietnam

[2]Faculty of Information Technology, VNU University of Engineering and Technology, Hanoi, 10000, Vietnam

[3]Natural Language Processing and Knowledge Discovery Laboratory, Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam

*Corresponding Author: Anh-Cuong Le. Email: leanhcuong@tdtu.edu.vn

Received: 22 October 2022 Accepted: 20 February 2023 Published: 26 January 2024

ABSTRACT

Text summarization aims to generate a concise version of the original text. The longer the summary text is, the more detailed it will be from the original text, and this depends on the intended use. Therefore, the problem of generating summary texts with desired lengths is a vital task to put the research into practice. To solve this problem, in this paper, we propose a new method to integrate the desired length of the summarized text into the encoder-decoder model for the abstractive text summarization problem. This length parameter is integrated into the encoding phase at each self-attention step and the decoding process by preserving the remaining length for calculating head-attention in the generation process and using it as length embeddings added to the word embeddings. We conducted experiments for the proposed model on the two data sets, Cable News Network (CNN) Daily and NEWSROOM, with different desired output lengths. The obtained results show the proposed model's effectiveness compared with related studies.

KEYWORDS

Length controllable; abstractive text summarization; length embedding

1 Introduction

There are two common approaches to dealing with the text summarization problem: extractive text summarization (ETS) and abstractive text summarization (ATS). While ATS aims to generate a new text to summarize the main content of the original text [1–4], the purpose of ETS is to extract important sentences from the original text. The former approach seems natural and appropriate and is similar to what humans do when summarizing a text, thus making it an exciting subject to study. Furthermore, the AST is a highly customizable approach, especially for the length of the generated summary text.

It was shown that it is a practical need to generate a summary text with different levels of abstraction [5]. Therefore, the problem of text summarization with a desired length is an important

problem that has attracted much interes in the summarization community. This problem is called the controllable abstractive summarization we are dealing with.

A well-known approach for AST is to apply deep learning models for generating an output sequence from an input sequence, which consists of two components, the encoder and decoder components. The first component is responsible for encoding information from the original text, while the second one is responsible for generating the summarized sequence [6]. The commonly used models are Long Short-Term Memory (LSTM) and Transformer [7]. Related studies solve this problem according to a general principle that includes the length constraint in the generation process. The standard technique used is that it tries to stop early in the decoding process. A typical method was shown in Rush et al. [8], where the proposed model assigns a negative number to generated words that exceed the desired length limit. This approach is entirely mechanical and does not provide timely information about the desired length of the output at all stages of the encoding and decoding processes. In other studies, Length Embedding (LE) [9] encodes the remaining length as an additional input length for the LSTM of the decoder, and Length Attention (LA) [10] proposes a Length Attention mechanism to make a priority in length control. These studies have used the desired length information more flexibly by creating a length context vector and using it in the attention mechanism. However, it will not be effective when the training data is sparse because it contains more parameters to use this length of information.

Several other studies (e.g., Length-controllable Prototype Abstractive Summarization (LPAS) [11]) try to use the desired length information in the encoding process to select the corresponding number of most important words for the encoding process, thereby influencing the generated summary. Takase et al. [12] used the desired length as added information in the encoding processing, extending the sinusoidal positional encoding formula given in [13].

In this paper, we also aim to use the information about the desired length of the output sequence in both the encoder and decoder components of the sequence-to-sequence (seq2seq) model. However, unlike the previous studies, in our proposed model, the encoding of length information is dynamically integrated into the model more intrinsically. We consider the desired length a valuable information source, and it is equally used in encoding along with word embeddings and position embeddings. During the decoding process, we will dynamically encode the desired length of information. We will gradually reduce this information so that the completion of output generation happens naturally. We also use length embeddings at the decoder as added information to word embeddings for generating output words.

The novelty of our approach is to use as much as possible of the desired length by integrating it into both the encoder and decoder, unifying their use as an embedding vector integrated with word embeddings to derive the information for generating the output text of the desired length. The idea is to utilize the characteristics of deep learning models, in which the information is filtered by itself based on the model and big data. If the information is added in many different aspects through the learning process, it will make a useful contribution to producing the desired output. Our proposed model is based on the Transformer model for the Sequence to Sequence type of problem, in which we integrate the additional information about the desired output length.

In summary, our proposal represents the following new contributions:

- First, we uniformly represented the desired length information by an embedding vector. We combined it with the word vectors and position vectors in both the encoder and decoder components. In addition, we suggest using an additional Head Attention to control the remaining length of the output when referring to the desired length. Our approach helps the

proposed model contain more information about the desired length in various encoding and decoding aspects than in other studies such as in [9,10,12,13].
- Second, our proposal uses only the encoder-decoder transformer model and does not use additional modules as in the study [11]. The proposed model utilizes the characteristics of deep learning models and big data to filter helpful information to produce the desired output.

The rest of the paper is organized as follows. Section 2 describes an abstractive summarization problem with a length constraint and presents some technical details of the Transformer model to supplement the description of the new technical proposals. Section 3 describes in detail the proposals for integrating the desired length information into the proposed model's encoder and decoder. Section 4 gives the experiments and results of the proposed model when compared to previous models. Finally, the contribution of this paper is summarized in conclusion.

2 Length-Controllable Summarization with Transformer Model

2.1 The Problem Description

The seq2seq model for the summarization problem takes an input text as a sequence of words and generates a summary text that is also considered as a sequence of words. In Length-controllable summarization, the model takes the source document $x = (x_0, x_1, \ldots, x_{n_{input}})$ and a desired length l as input and the summary $y = (y_0, y_1, \ldots, y_{m_{onput}})$ as output. Note that $x+m$ and y_n are the special tokens, denoted by "*eos*" that are used to determine the ending of a sentence. In general, to solve this problem, models are built to estimate the conditional probability $p\,(x, l)$, defined as follows:

$$p\,(y|x, l) = \prod_{t}^{n} p\,(y_t|y_1, y_2, \ldots, y_{t-1}, x, l) \tag{1}$$

Like most other studies, we use the Transformer model in this work. The model is trained on pairs (original text, summary text); here we will use the length of the summary text for the parameter l.

The conventional transformer model for the abstractive summarization problem is quite clear. The transformer consists of two components: the encoder component, which has the goal of encoding the input text, and the decoder component, which is responsible for generating the word string that is the result of the summary.

The encoder takes as an input a sequence of vectors containing information about the corresponding words of the input text. The transformer uses self-attention and multi-head mechanisms to encode the context information of each word. As the result, the encoder maps an input sequence of symbol representations $(x_1, \ldots, x_{n_{input}})$ to a sequence of continuous representations $z = (z_1, \ldots, z_{n_{input}})$. And then, given z, the decoder generates an output sequence $(y_0, y_1, \ldots, y_{m_{onput}})$ of symbols, one element at a time. At each step, the model is auto regressive and consumes the previously generated symbols as additional input when generating the next.

The goal is to efficiently integrate information about the desired length of the output into the encoding and decoding processes so that the resulting summary text satisfies both the summary quality and the desired length.

2.2 Some Basics of Seq2Seq Transformer Model

Transformers draw inspiration from the encoder-decoder architecture found in RNNs because of their attention mechanism. It can handle sequence-to-sequence (seq2seq) tasks while removing the sequential component. A Transformer, unlike an RNN, does not perform data processing in sequential

order, allowing greater parallelization and faster training. The Transformer model's architecture, including the encoder and decoder for the seq2seq problem, is illustrated in Fig. 1.

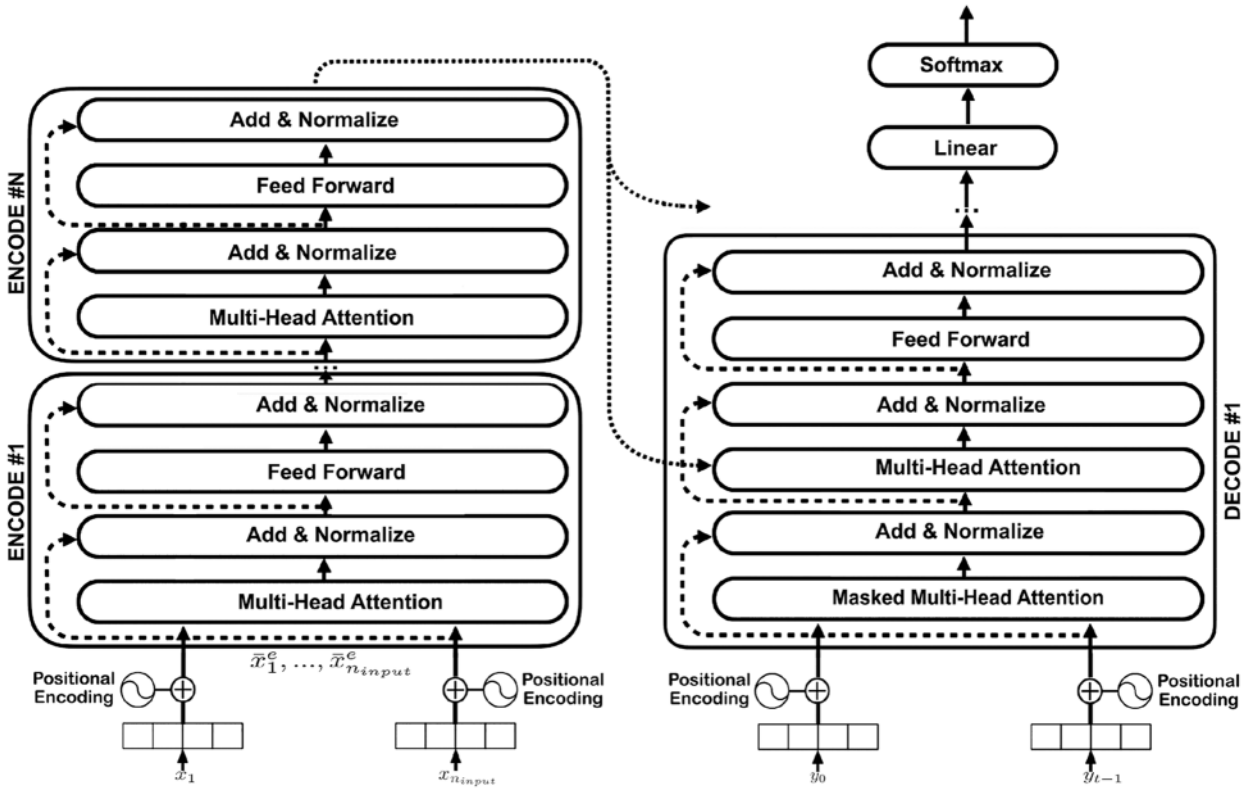

Figure 1: Overview of transformer sequence to sequence

We list here some important notations in the formulas used in the Transformer model.

$x^e = x_1^e, \ldots, x_{n_{input}}^e$ where x_i^e is the word embedding of the word i^{th} of input sequence.

$\overline{x}^e = \overline{x}_1^e, \ldots, \overline{x}_{n_{input}}^e$ where $\overline{x}_i^e$ positional embedding i^{th} of the input sequence.

$y^e = y_1^e, \ldots, y_{m_{output}}^e$ where y_i^e: word embedding i^{th} of the output sequence.

$\overline{y}^e = \overline{y}_1^e, \ldots, \overline{y}_{m_{output}}^e$ where $\overline{y}_i^e$: positional embedding $i^{\{th\}}$ of the output sequence.

N: number of stacks in the encoder, the number of stacks in the decoder.

d_{model}: length of word embedding vector.

d_k: length of query vector, length of the key vector.

h: number of attention heads in multi-head attention.

n_{vol}: vocabulary size.

The encoder

The encoder includes N layers, each of which has two sub-layers. The first is the most important layer, which is a multi-head self-attention mechanism. The second one is a position-wise fully connected feed-forward network.

The decoder

The decoder is designed with the same number of layers as the encoder. In addition to the two sub-layers, as in every encoder layer, the decoder inserts a more sub-layer, which performs multi-head attention over the output of the encoder stack.

Attention

The seft-attention value is estimated by the following equation:

$$Seft\_Att\,(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

where

$$Q = \overline{x}^e \times W^Q \in \mathbb{R}^{n_{input} \times d_{model}} \tag{3}$$

$$K = \overline{x}^e \times W^K \in \mathbb{R}^{n_{input} \times d_{model}} \tag{4}$$

$$V = \overline{x}^e \times W^V \in \mathbb{R}^{n_{input} \times d_{model}} \tag{5}$$

Multi-head attention allows the model to attend to different aspects of context-based representation. The multi-head attention for stack i in the encoder is calculated by:

$$Z_i^e = Multihead\_Att_i^e(Q,K,V) = Concat(head_{i,1}^e, head_{i,2}^e, \ldots, head_{i,h}^e)W_i^{O,e} \tag{6}$$

$$z_{i,j}^e = head_{i,j}^e = Self\_Att(QW_{i,j}^{Q,e}, KW_{i,j}^{K,e}, VW_{i,j}^{V,e}); j \in 1, \ldots, h \tag{7}$$

which $W_{i,j}^{Q,e} \in \mathbb{R}^{d_{model}*d_k}$, $W_{i,j}^{K,e} \in \mathbb{R}^{d_{model}*d_k}$, $W_{i,j}^{V,e} \in \mathbb{R}^{d_{model}*d_v}$, $W_{i,j}^{O,e} \in \mathbb{R}^{hd_v*d_{model}}$

Similarly, for the decoder multi-head attention for layer i in the decoder, we have that

$$Z_i^d = Multihead\_Att_i^d(Q,K,V) = Concat(head_{i,1}^d, head_{i,2}^d, \ldots, head_{i,h}^d)W_i^{O,d} \tag{8}$$

where

$$head_{i,j}^d = Self\_Att(QW_{i,j}^{Q,d}, KW_{i,j}^{K,d}, VW_{i,j}^{V,d}) \tag{9}$$

Masked multi-head attention only exists in the decoder. For the masked multi-head for layer i in the decoder during training, we have that

$$Z^T = MaskedMultihead\_Att_i^T(Q,K,V) = Concat(head_{i,1}^{dm}, head_{i,2}^{dm}, \ldots, head_{i,h}^{dm})W_i^{O,dm} \tag{10}$$

where

$$head_{i,j}^{dm} = softmax\left(Mask\left(\frac{\left(QW_{i,j}^{Q,dm}\right)\left(KW_{i,j}^{K,dm}\right)^T}{\sqrt{d_k}}\right)\right)\left(VW_{i,j}^{V,dm}\right) \tag{11}$$

Here, Mask(m) is a function of a square matrix m, and it holds that $m' = Mask(m)$ if

$$m_{ij}' = \begin{cases} m_{ij} & \text{if } \ i \geq j \\ -\infty & \text{otherwise} \end{cases} \tag{12}$$

For the masked multi-head for layer i in the decoder during inference, the mask function is removed. That is to say, the MaskedMultiHead layer is the same as the MultiHead layer during inference.

$$Z^I = MaskedMultihead\_Att_i^I(Q,K,V) = Concat(head_{i,1}^{dm}, head_{i,2}^{dm}, \ldots, head_{i,h}^{dm})W_i^{O,dm} \tag{13}$$

where

$$head_{i,j}^{dm} = Self\_Att(QW_{i,j}^{Q,dm}, KW_{i,j}^{K,dm}, VW_{i,j}^{V,dm}) \tag{14}$$

LayerNorm is a normalized layer that does not change the shape of tensors. We use residual connections and apply layer normalization to determine the output value of layer i in the encoder and decoder, i.e.,

$$\overline{Z} = \text{LayerNorm}\left(\overline{Z} + Z\right) \tag{15}$$

Note that in all the formulas, the notation W is used to denote the parameter of the model. In the last layer of the decoder, we usually use a linear layer and then go through the soft-max layer to generate the corresponding output.

3 Our Proposed Model

In this section, we will present our proposal about how to integrate information on the desired length of the output into the encoder and decoder of the basic model presented in the previous section (see Fig. 2). The algorithm schema of the model is shown in Fig. 3, which shows the information about the desired length used in different aspects. The length embedding is integrated with word embedding in both the encoder and decoder. The length-controllable head attention is calculated in the Multi-Head Attention calculation step by Eq. (24). This length-controllable head attention is calculated based on the desired length and remaining length. It is used to control the output length.

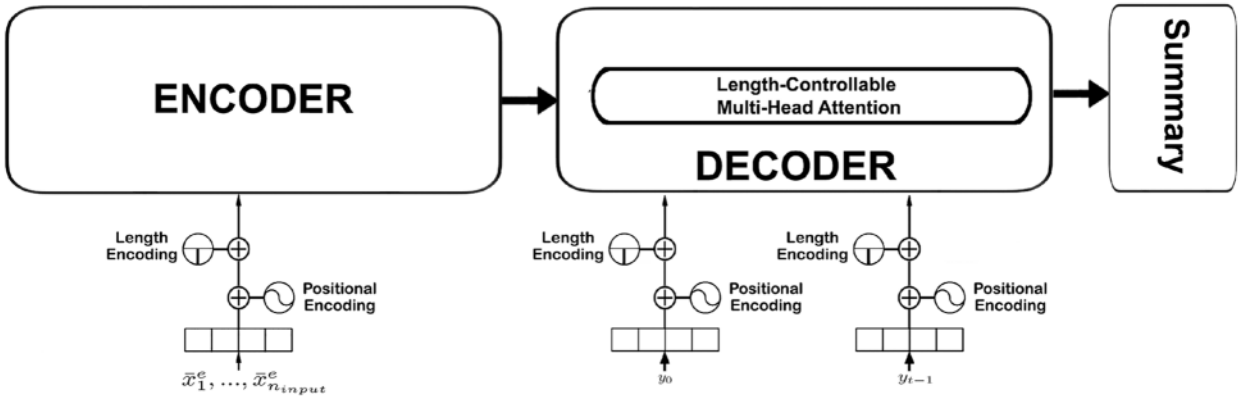

Figure 2: Our proposed model

3.1 Integrating Length Information into Encoder

As mentioned above, the input of the encoder is a sequence of vectors, which are calculated from the embeddings of input words and of their positions.

To use the desired length information l of the output, we first represent it by a vector and then add it with the word embeddings and word position embeddings.

Suppose that o_l^e denotes a one-hot vector of length l with l in the range [1, m_{output}], the length-embedding $LE^e(x^e)$ will be calculated as:

$$LE^e(x^e) = o_l^e \times W^l \in \mathbb{R}^{m_{output} \times d_{model}} \tag{16}$$

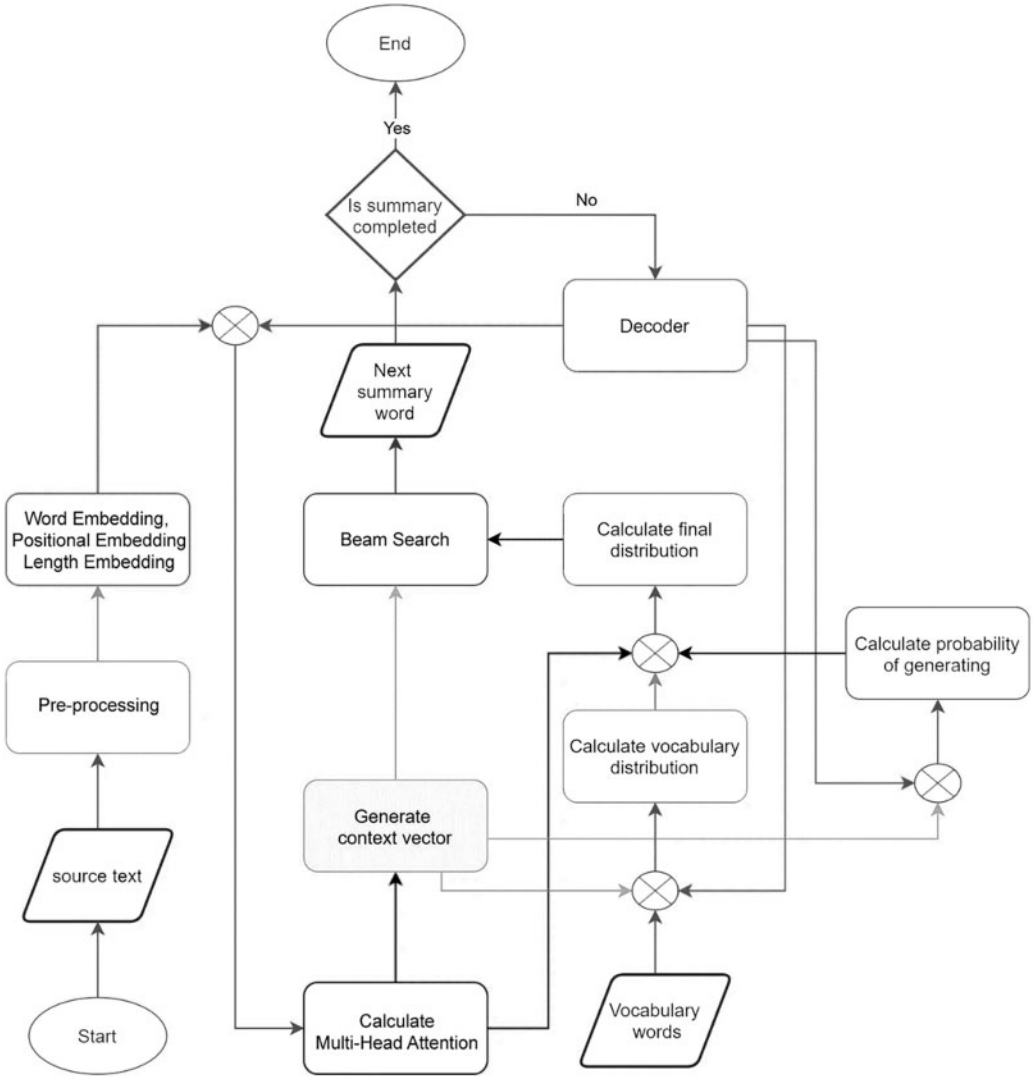

Figure 3: The algorithm flow charts of the proposed model

Then the input representation of $\overline{x}^e$ is a combination of Word Embedding, Positional Embedding, and Length Embedding. It is calculated as:

$$\overline{x}^e = PE\left(x^e\right) + LE^e\left(x^e\right) + x^e \in \mathbb{R}^{n_{input} \times d_{model}} \tag{17}$$

3.2 Integrating Length Information into Decoder

The desired length l of the output sequence is used to add information to the words y_i through a discretization of levels in the segment $[1, l]$. Then o_t^d is a one-hot representation of length t with t in the range $[1, l]$. Then the length-embedding $LE^d(y^e)$ will be calculated as:

$$LE^d(y^e) = o_t^d \times W^l \in \mathbb{R}^{l \times d_{model}} \tag{18}$$

The input representation of $\overline{y}^e$ is now added Length Embedding together with Word Embedding and Positional Embedding as:

$$\overline{y}^e = PE\left(y^e\right) + LE^d\left(y^e\right) + y^e \in \mathbb{R}^{n_{input} \times d_{model}} \tag{19}$$

At the decoding step, the remaining length information l_t at each decoding step is used at each cross-attention calculation step in Eq. (32), specified as follows:

$$\overline{Z}_1^{dm} = Multihead\_Att_i^d\left(\overline{Z}_1^d, \overline{Z}_N^{er}, \overline{Z}_N^{er}\right) = Concat\left(head_{i,1}^{dm}, head_{i,2}^{dm}, \ldots, head_{i,h}^{dm}, head_{i,l_t}^{dm}\right) W_i^{O,dm} \tag{20}$$

where

$$head_{i,j}^{dm} = Self\_Att\left(\overline{Z}_1^d W_{i,j}^{Q,dm}, \overline{Z}_N^{er} W_{i,j}^{K,dm}, \overline{Z}_N^{er} W_{i,j}^{V,dm}\right) \tag{21}$$

Especially, we propose to use one more head-attention $head_{i,l_t}^{dm}$ to control the remaining length of the output sequence, which we call the length head attention. The smaller the remaining length l_t, the more attention will be paid to the remaining length limit l_t. This allows the model to control the output length during generation. The value of head-attention $head_{i,l_t}^{dm}$ is calculated based on normalizing the value of head-attention $head_{i,j}^{dm}$ with $j \in 1 \ldots h$.

$$head_{i,l_t}^{dm} = \sum_{i=1}^{l} \overline{\alpha}_{t,i} \overline{Z}_i^d \tag{22}$$

$$\overline{\alpha}_{t,i} = \frac{\alpha_{t,i}^{'}}{\sum_{j=1}^{l} \alpha_{t,j}^{'}} \tag{23}$$

$$\alpha_{t,i}^{'} = (l + 1 - l_t) \sum_{j=1}^{h} head_{i,j}^{dm} \tag{24}$$

3.3 Training

Here we started to apply self-attention at fist stack in the encoder $Z_1^e \in \mathbb{R}^{n_{input} \times hd_v}$ as follows:

$$Z_1^e = Multihead\_Att_1^e\left(\overline{x}^e, \overline{x}^e, \overline{x}^e\right) \tag{25}$$

We use residual connection and apply layer normalization $\overline{Z}_1^e \in \mathbb{R}^{n_{input} \times hd_v}$

$$\overline{Z}_1^e = LayerNorm\left(\overline{x} + Z_1^e\right) \tag{26}$$

Then we do feed forward neural network $F_1^e \in \mathbb{R}^{n_{input} \times hd_v}$

$$F_1^e = FFN_1^e\left(\overline{x} + \overline{Z}_1^e\right) \tag{27}$$

We use residual connection and apply layer normalization $\overline{Z}_1^{er} \in \mathbb{R}^{n_{input} \times hd_v}$

$$\overline{Z}_1^{er} = LayerNorm\left(\overline{Z}_1^e + F_1^e\right) \tag{28}$$

$\overline{Z}_1^{er}$ will be used as the input to the next stack (stack 2) in the encoder $Z_2^e \in \mathbb{R}^{n_{input} \times hd_v}$

$$Z_2^e = Multihead\_Att_1^e\left(\overline{Z}_1^{er}, \overline{Z}_1^{er}, \overline{Z}_1^{er}\right) \tag{29}$$

We iterate this process N times and get $Z_4^e \in \mathbb{R}^{n_{input} \times hd_v}$ from the stack N in the encoder. This encoded embeddings Z_4^e will be used for each of the stacks in the decoder. We then look at the decoder.

We then look at the decoder. We have the translation output, which is the translation of input, consisting of m_{output} words. The output matrix $y \in \mathbb{R}^{m_{output} \times n_{vol}}$ has $y = \{y\_1, \ldots, y_{m_{output}}\}$. Each y_k is a one-hot encoded row vector of length n_{vol} for the corresponding word.

The input to the decoder the right shifted of y is called y'. Concretely $y' = y_0, y_1, \ldots, y_{m_{output}-1}$, and y_0 could just simply be a zero vector. Give y' we wish the decoder could generate y. Similar to the encoder, we have the output embedding matrix $y^e \in \mathbb{R}^{n_{output} \times d_{model}}$:

$$y^e = Input\_embedding\,(y) = \sqrt{d_{model}} y' W_{embedding} \in \mathbb{R}^{m_{output} \times d_{model}} \tag{30}$$

$$\overline{y}^e = Positional\_encodding\,(y^e) = PE\,(y^e) + y^e \in \mathbb{R}^{n_{input} \times d_{model}} \tag{31}$$

Then we started to apply self-attention. However, to prevent the current word input from peaking at the following words, we used masked multi-head attention instead of the ordinary multi-head attention we used in the encoder. Here is the intuition. During training, the entire right shifted output y' was fed. Without the mask, after applying self-attention, word y_i will have attention to word y_j for $j > i$. This is unwanted. By applying mask, the attention of word y_i to word y_j for $j > i$ will be 0. We have $Z_1^d \in \mathbb{R}^{m_{output} \times hd_v}$

$$Z_1^d = MaskedMultihead\_Att_1^T\left(Q, K, V, \overline{y}^e\right) \tag{32}$$

We use residual connection and apply layer normalization $\overline{Z}_1^d \in \mathbb{R}^{m_{output} \times d_{model}}$

$$\overline{Z}_1^d = LayerNorm\left(\overline{y}^e + Z_1^d\right) \tag{33}$$

Then we use queries from $\overline{Z}_1^d$, keys and values from $\overline{Z}_N^{er}$. We then do multi-head attention trying to find the attention of the words to the embeddings from the encoder $\overline{Z}_1^{dm}$

$$\overline{Z}_1^{dm} = Multihead\_Att_i^d\left(\overline{Z}_1^d, \overline{Z}_N^{er}, \overline{Z}_N^{er}\right) \tag{34}$$

We use residual connection and apply layer normalization $\overline{Z}_1^{ddm} \in \mathbb{R}^{n_{output} \times d_{model}}$

$$\overline{Z}_1^{ddm} = LayerNorm\left(\overline{Z}_1^d + \overline{Z}_1^{dm}\right) \tag{35}$$

Then we do feed forward neural network $F_1^d \in \mathbb{R}^{m_{output} \times d_{model}}$

$$F_1^d = FFN_1^d\left(\overline{Z}_1^{ddm}\right) \tag{36}$$

We use residual connection and apply layer normalization $\overline{Z}_1^{dddm} \in \mathbb{R}^{m_{output} \times d_{model}}$

$$\overline{Z}_1^{dddm} = LayerNorm\left(\overline{Z}_1^{ddm} + F_1^d\right) \tag{37}$$

$\overline{Z}_1^{dddm}$ will be used as the input to the next stack (stack 2) in the decoder, $\overline{Z}_1^{dddm} \in \mathbb{R}^{m_{output} \times hd_v}$

$$Z_2^d = Multihead\_Att_2^T\left(\overline{Z}_1^{dddm}, \overline{Z}_1^{dddm}, \overline{Z}_1^{dddm}\right) \tag{38}$$

We iterate this process N times and get $\overline{Z}_1^{dddm} \in \mathbb{R}^{m_{output} \times d_{model}}$ from the stack N in the decoder. This encoded embeddings $\overline{Z}_1^{dddm}$ will be used for linear transformation followed by SoftMax to generate the

output probabilities for the decoded word sequence. Concretely, we have $Z \in \mathbb{R}^{m_{output} \times n_{vol}}$ were

$$Z = \overline{Z}_1^{dddm} W^S + b^s \tag{39}$$

We then apply SoftMax to get the probability matrix $P \in \mathbb{R}^{m_{output} \times n_{vol}}$, where

$$\mathrm{P} = \mathrm{softmax(Z)} \tag{40}$$

Finally, we could calculate the cross entropy loss using y and P. More specifically, each row of y_{output} and P are y_i and $p_{i,}$, respectively.

$$L = \sum_{i=1}^{m_{output}} CrossEntropy\,(y_i, p_i) \tag{41}$$

4 Experiment

4.1 Data

To evaluate the effectiveness of the proposed model, we conducted experiments on the popular datasets CNN/Daily Mail (CNNDM) and NEWSROOM, which were widely used in related studies such as in [7,9,11].

Structured CNN/Daily Mail (CNNDM) data consists of text pairs and the summary text of that texts. This data includes 286,817 pairs of training texts, 13,368 pairs of evaluation texts, and 11,487 pairs of test texts.

NEWSROOM has 1.3 million text pairs and corresponding summaries. We randomly selected 973,042 text pairs for training data, 30,000 text pairs as the evaluation data set, and 106,394 text pairs for model testing data [6,14].

4.2 Experimental Models

Our proposal is indicated by integrating different aspects of the desired output length information into the encoder and decoder of the transformer model. Our proposed model adds desired length information to both the encoder and the decoder, so we will build different test models to test the effectiveness of these suggestions. Furthermore, how we represent and integrate desired length information into our Seq2Seq Transformer architecture is also different so we will compare the experimental results obtained with those of related studies. Specifically, we build the following test models:

- The proposed model with fully integrated information of the desired length, including the length head attention at the decoder and the length embedding at both the encoder and decoder. We call this model "the full proposed model."
- The proposed without integrating desired length information at the encoder. We call it "the proposed model 1". This model uses the length head attention and length embeddings at the decoder without the length embedding at the encoder.
- "The proposed model 2" contains the information about the desired length in two aspects: the length head attention at the decoder and the length embedding at the encoder.

In summary, Table 1 illustrates the three models with varying utilization of the desired length information.

In the experimental process, we used the same network configuration parameters on the two corpora. The model uses the 300-dimensional Glove Embeddings set. The encoder-decoder uses a Transformer with 6 blocks, with 8 heads on the data and 1 head length. The dimension of the Feed Forward Network (FFN) network is 2048, d_{model} is set to 510. We use true Adam's optimization with the scheduled learning rate suggested in the study of Vaswani et al. [13]. The learning rate is fixed at 0.0001, and mini-batches of size 16 are used. Gradient clipping is also used with a maximum gradient norm of 2.0. For all datasets, the vocabulary consists of 50K words and is shared between the source and target. During training, we run 35 epochs for the CNN/Daily Mail dataset and 20 epochs for the Newsroom dataset. During testing, we set the size of a beam to 5. The model sets the lexical sizes of the input and output of the model to 100,000 and 1,000, respectively.

Table 1: The three models with different aspects of the desired length information

	Length embeddings at the encoder	Length embeddings at the decoder	Length head attention at the decoder
The proposed model 1		X	X
The proposed model 2	X		X
The full proposed model	X	X	X

4.3 Results

Similar to previous related studies, we also performed experiments with different desired output lengths, including three values of 30, 50 and 70.

We used the ROUGE scores (F1), including ROUGE-1(R1), ROUGE-2 (R-2), and ROUGE-L (R-L), as the evaluation metrics [15]. We used the files2rouge toolkit for calculating the ROUGE scores[1].

At first, Figs. 4–9 show the results of our experimental models. They represent a unified assessment that the full proposed model outperforms the rest of the models in different criteria: on different measures Rouge-1, Rouge-2, and Rouge-L, as well as for different desired lengths 30, 50, and 70. The superiority of the complete proposed model is also confirmed for both data sets, CNNDM and NEWSROOM.

Note that all three models (i.e., the full model, model 1, and model 2) use the length head attention. However, our proposed full model uses information about the desired length, such as length embedding in both the encoder and decoder. Whereas model 1 contains only the length embedding in the decoder, model 2 contains only the length embedding in the decoder. The full model's use of desired length information in various aspects explains why it yields better results than model 1 and model 2.

[1] https://github.com/pltrdy/les2rouge.

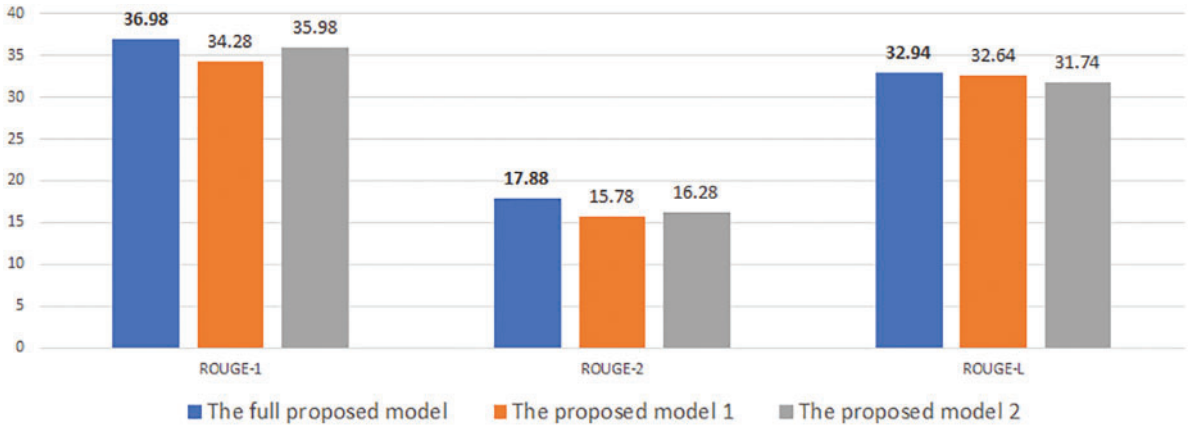

Figure 4: Experimental results of our model with desired length is 30 on CNN/DM

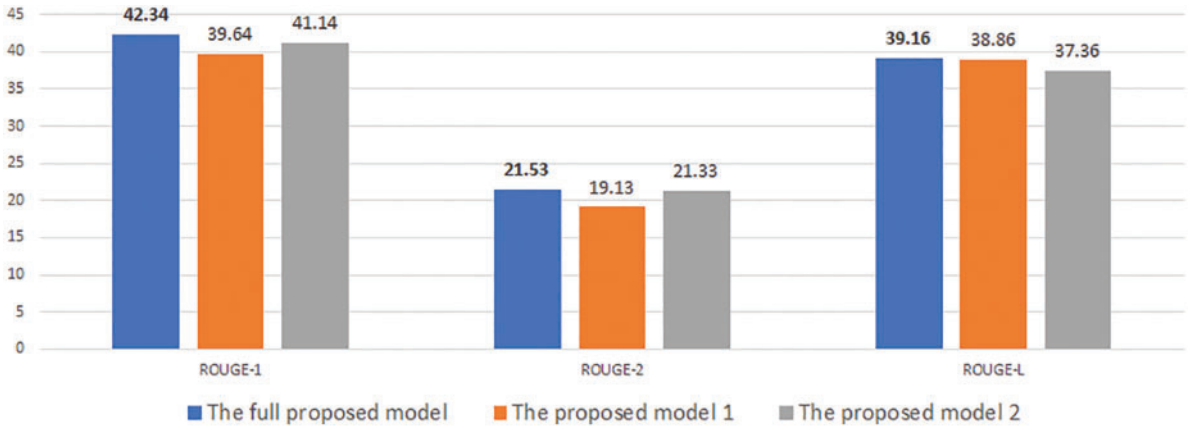

Figure 5: Experimental results of our model with desired length is 50 on CNN/DM

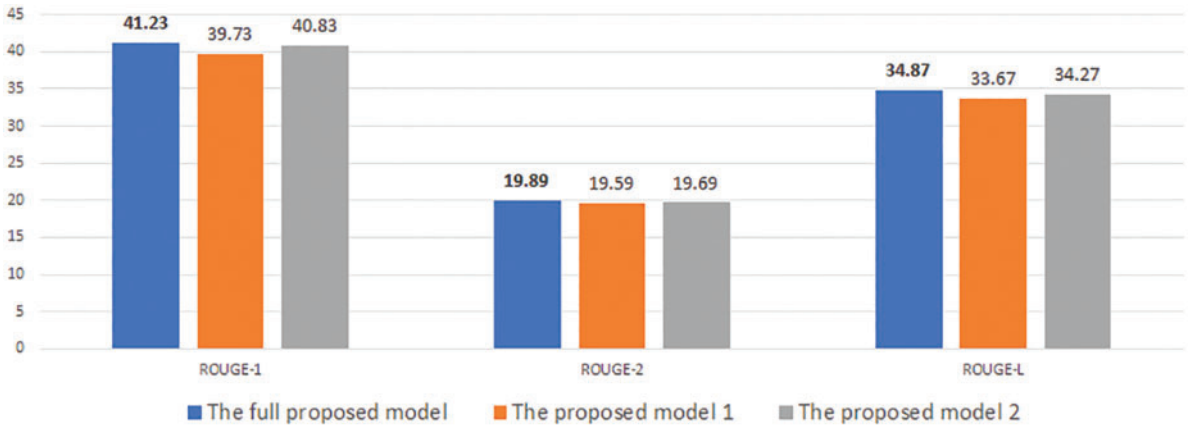

Figure 6: Experimental results of our model with desired length is 70 on CNN/DM

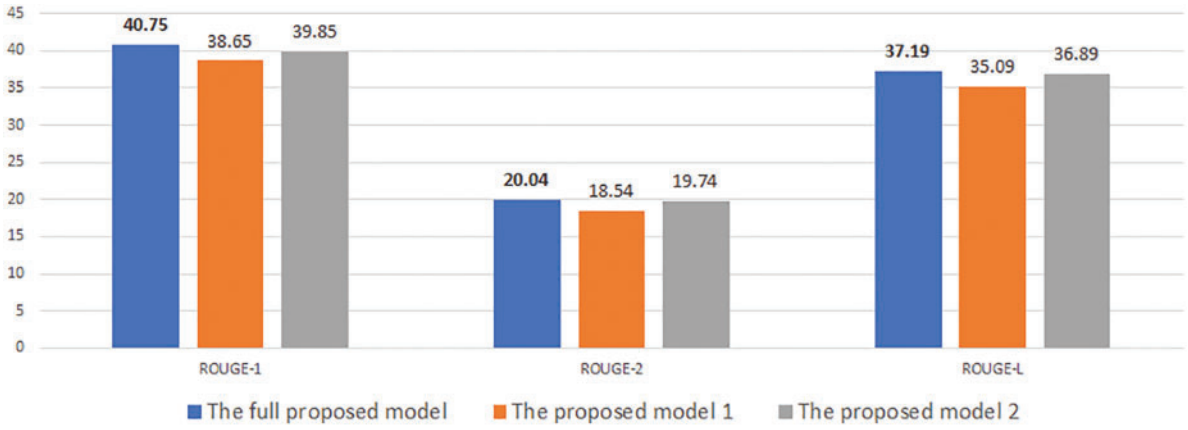

Figure 7: Experimental results of our model with desired length is 30 on NEWROOMS

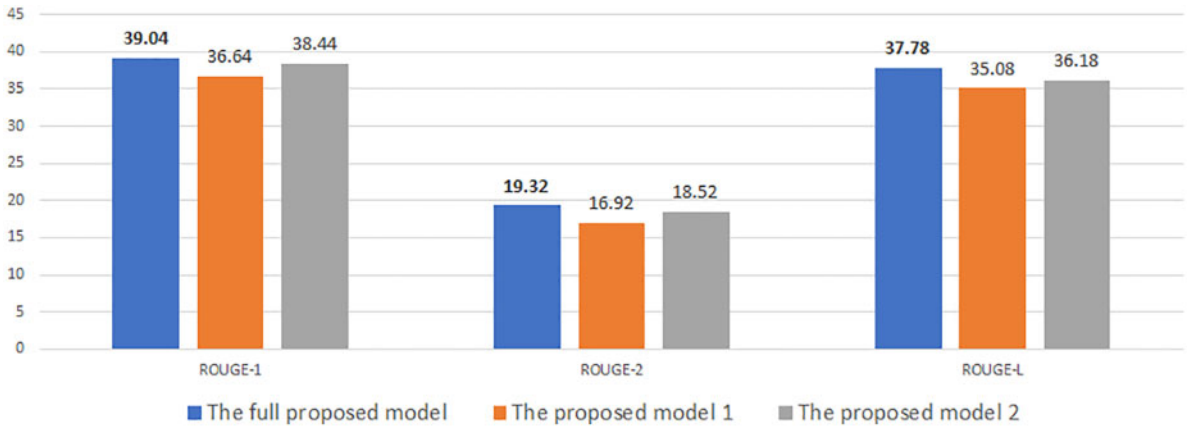

Figure 8: Experimental results of our model with desired length is 50 on NEWROOMS

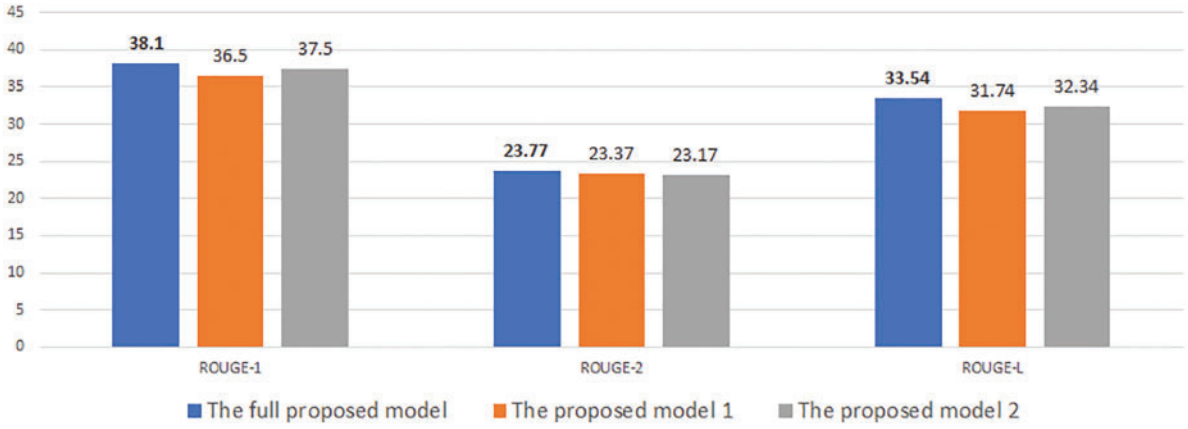

Figure 9: Experimental results of our model with desired length is 70 on NEWROOMS

To compare with previous models, we choose four typical recent models, and they are performed experimentally on the same data sets. In Figs. 10–15, we denote LE for the study of Kikuchi et al. [9],

which encodes input length embeddings to the decoder. We denote LA for the study of Yu et al. [16], which uses the desired length as attention for decoding. Two other studies using the method of extracting information from input text based on the desired length before decoding are [11,17], denoted as Length Control (LC) and LPAS, respectively.

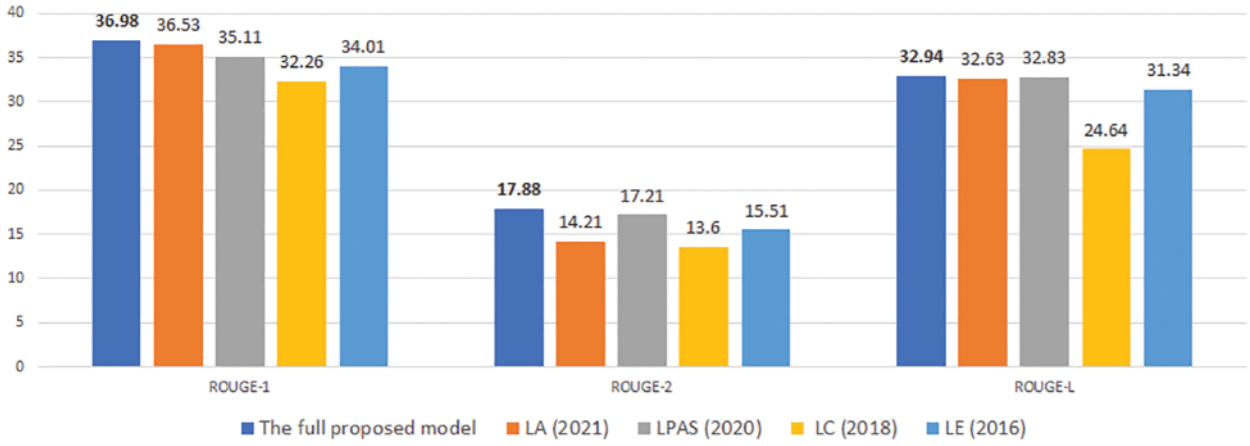

Figure 10: An experimental comparison of our model with previous studies on CNN/DM with desired length is 30

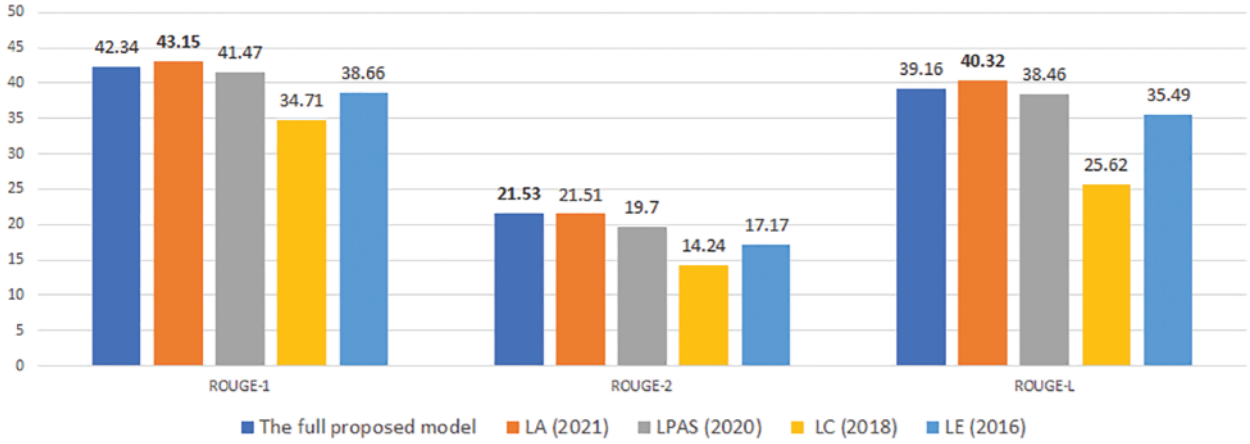

Figure 11: An experimental comparison of our model with previous studies on CNN/DM with desired length is 50

Experimental results in Figs. 10–15 show that our proposed model gives competitive results with previously published research models.

For the NEWSROM corpus, which is considered a more diverse corpus in terms of summary style because it is synthesized from many different news sites, the experimental results in Figs. 13–15 show the results. In the model, we proposed to give competitive results on the ROUGE-1 and ROUGE-L measures in the model. However, ROUGE-2 in the LPAS model is much better, which shows that our proposed model has an excellent ability to link important words but could be a better ability to link important phrases. The LPAS model gives a better possibility. However, the approach of the LPAS model is very different; LPAS is formed based on LA, which is a model that combines the author's

proposed unit of attention with a PAULUS model [18], which has achieved outstanding results in the text summarization problem. Besides that, LPAS includes a separate stage to extract the list of important words/phrases containing the content of the original text, so this method is more complex and takes more time. At the same time, our solution represents a fully automated process using a length constraint to extract important information within the model itself. Therefore, it is also an open issue for us to study in the future to improve the efficiency of the proposed model.

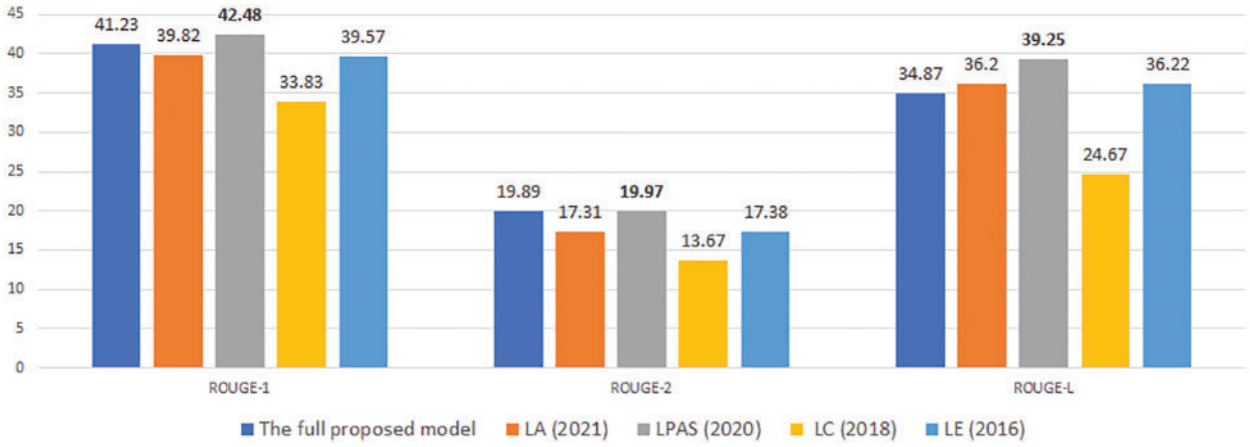

Figure 12: An experimental comparison of our model with previous studies on CNN/DM with desired length is 70

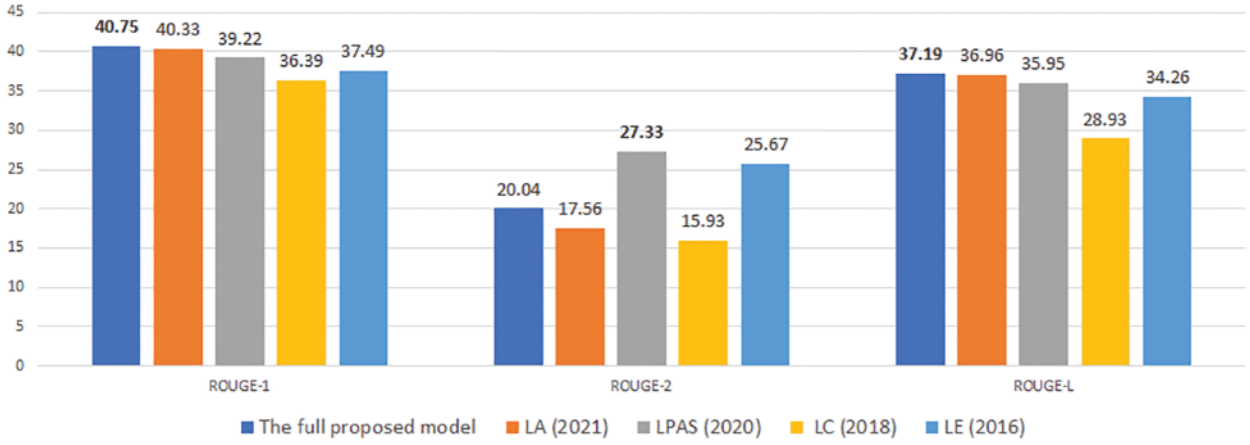

Figure 13: An experimental comparison of our model with previous studies on NEWROOMS with desired length is 30

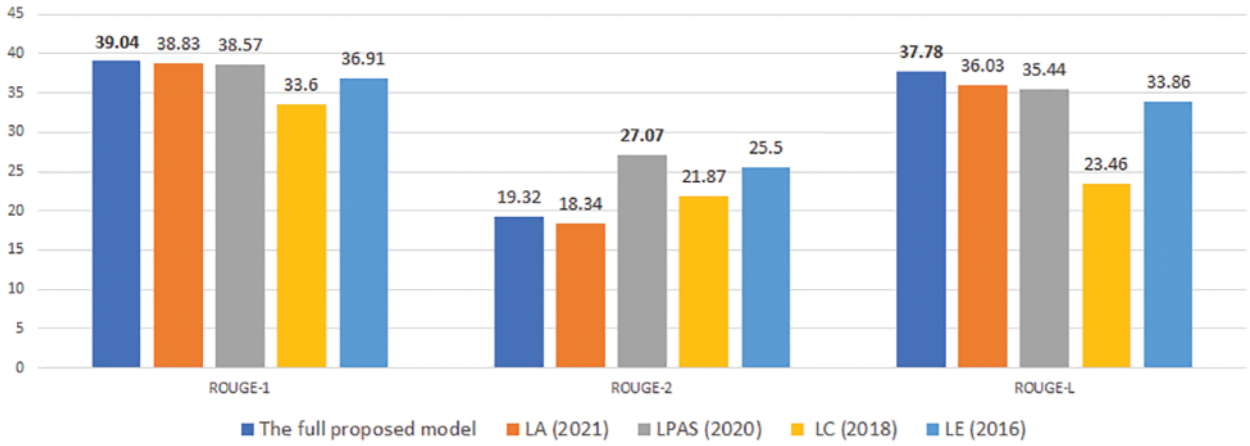

Figure 14: An experimental comparison of our model with previous studies on NEWROOMS with desired length is 50

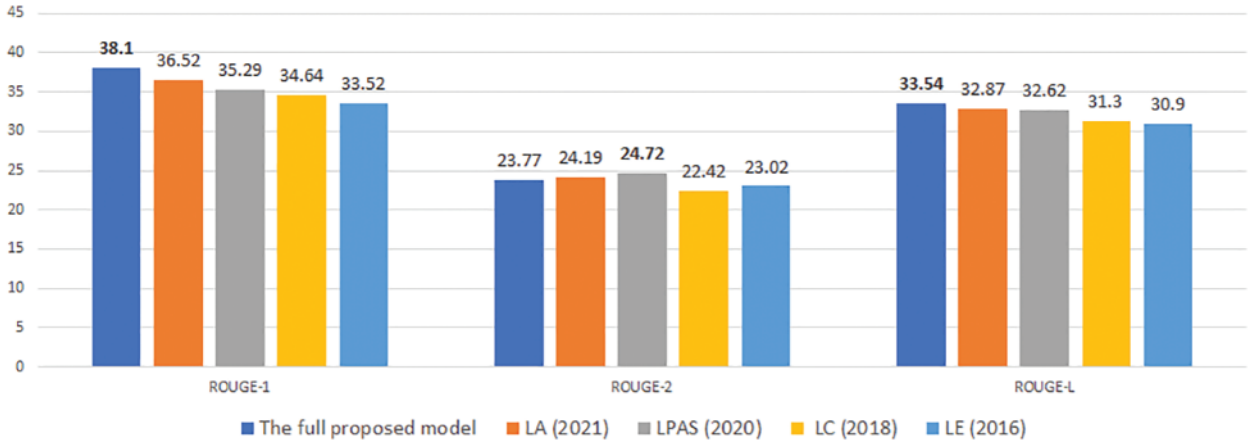

Figure 15: An experimental comparison of our model with previous studies on NEWROOMS with desired length is 70

5 Conclusion

In this paper, we have presented our new approach to the problem of Length-Controllable Abstractive Summarization. We proposed to use the desired length constraint information with various aspects in both the encoder and decoder of the sequence-to-sequence transformer model. Experimental results on different configurations of the proposed model have proved that our proposal is efficient; each additional component has added helpful information to the model and contributed to increasing the accuracy of the output. Our experiment also shows that the proposed model achieves competitive results with the related summarization methods.

In future work, we will continue to improve the proposed model by combining it with the advantages of the approach to extracting important words at the encoder step described in [19] and hope to create more efficient models.

Acknowledgement: None.

Funding Statement: This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant Number 102.05-2020.26.

Author Contributions: Proposing ideas, solutions and formulation, writing the article: Ngoc-Khuong Nguyen, Anh-Cuong Le, Viet-Ha Nguyen; Conducting the experiment: Ngoc-Khuong Nguyen, Dac-Nhuong Le. All authors reviewed the results and approved the final version of the manuscripts.

Availability of Data and Materials: The data that support the findings of this study are openly available at https://github.com/abisee/cnn-dailymail for the CNNDM dataset and https://github.com/lil-lab/newsroom for the NEWROOM dataset.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. See, P. J. Liu and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proc. of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics*, Vancouver, Canada, 2017.

[2] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed *et al.,* "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics*, 2020.

[3] Z. Liu, K. Shi and N. Chen, "Conditional neural generation using sub-aspect functions for extractive news summarization," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.

[4] N. Moratanch and S. Chitrakala, "Anaphora resolved abstractive text summarization (AR-ATS) system," *Multimedia Tools and Applications*, vol. 82, pp. 4569–4597, 2023. https://doi.org/10.1007/s11042-022-13299-9

[5] G. Xu and X. Hu, "Multi-dimensional attention based spatial-temporal networks for traffic forecasting," *Wireless Communications and Mobile Computing*, vol. 2022, pp. 1–13, 2022.

[6] N. Moratanch and S. Moratanch, "A novel framework for semantic oriented abstractive text summarization," *Journal of Web Engineering*, vol. 8, pp. 675–716, 2018.

[7] Y. Atri, S. Pramanick, V. Goyal and T. Chakraborty, "See, hear, read: Leveraging multimodality with guided attention for abstractive text summarization," *Knowledge-Based Systems*, vol. 227, pp. 107–152, 2021.

[8] A. M. Rush, S. Chopra and J. Weston, "A neural attention model for abstractive sentence summarization," in *Proc. of the 2015 Conf. on Empirical Methods in Natural Language Processing, Association for Computational Linguistics*, Lisbon, Portugal, 2015.

[9] Y. Kikuchi, G. Neubig, R. Sasano, H. Takamura and M. Okumura, "Controlling output length in neural encoder-decoders," in *Proc. of the 2016 Conf. on Empirical Methods in Natural Language Processing*, Austin, Texas, 2016.

[10] T. Yu, D. Su, W. Dai and P. Fung, "Dimsum @LaySumm 20: BART-based approach for scientific document summarization," in *Proc. of the 58th Annual Meeting of the Association for Computational Linguistics*, Canada, 2020.

[11] I. Saito, K. Nishida, K. Nishida, A. Otsuka, H. Asano *et al.,* "Length-controllable abstractive summarization by guiding with summary prototype," in *Proc. of the 60th Annual Meeting of the Association for Computational Linguistics*, Dublin, Ireland, 2020.

[12] S. Takase and N. Okazaki, "Positional encoding to control output sequence length," in *Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, 2019.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones *et al.,* "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, 2017.

[14] M. Grusky, M. Naaman and Y. Artzi, "Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies," in *Proc. of the 2018 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, New Orleans, Louisiana, 2018.

[15] C. Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, 2004.

[16] T. Yu, Z. Liu and P. Fung, "AdaptSum: Towards low-resource domain adaptation for abstractive summarization," in *Proc. of the 2021 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5892–5908, 2021.

[17] Y. Liu, Z. Luo and K. Zhu, "Controlling length in abstractive summarization using a convolutional neural network," in *Proc. of the 2018 Conf. on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018.

[18] R. Paulus, C. Xiong and R. Socher, "A deep reinforced model for abstractive summarization," arXiv:1705.04304, 2017.

[19] I. Saito, K. Nishida, K. Nishida and J. Tomita, "Abstractive summarization with combination of pre-trained sequence-to-sequence and saliency models," arXiv:2003.13028, 2020.