Key Phrase Aware Multi-Head Attention for Abstractive Summarization

Shuaiqi Liu[1], Jiannong Cao, Ruosong Yang, and Zhiyuan Wen

Department of Computing, The Hong Kong Polytechnic University, Hong Kong

Abstract

Automatic text summarization techniques can produce a concise summary of one or more text documents and support people to process the textual information more efficiently. The abstractive summarization methods aim to generate novel sentences as a summary covering salient content from input documents. Compared with previous RNN-based abstractive summarization models, the transformer-based models employ the self-attention mechanism to capture dependencies in documents, and they can generate better summaries. But existing works have not considered key phrases in determining self-attention weights of the transformer-based summarization model. Consequently, some of the tokens within key phrases only receive small attention weights, which can affect completely encoding key phrases that convey the salient ideas of input documents. In this paper, we propose the Key Phrase Aware Transformer (KPAT), a model with the highlighting mechanism in the encoder to assign greater attention weights for tokens within key phrases. Specifically, we first build the block diagonal highlighting matrix to indicate key phrases' positions and their importance scores. To combine the self-attention weights with the phrases' importance, we design two structures of highlighting attention for each head and the multi-head highlighting attention. Besides, the block-wise linear transformation on the highlighting matrix is adopted to adjust the scale of phrases' importance scores. The experimental results on two datasets from different summarization tasks and domains show that our proposed model significantly outperforms the competitive baseline models.

*Corresponding author.
Email address: `cssqliu@comp.polyu.edu.hk` (Shuaiqi Liu)

Keywords: Text summarization, Abstractive summarization, Key phrase extraction, Deep Learning

1. Introduction

Nowadays, people are suffering from the information explosion, which refers to the rapid increase in the amount of published information or data. With the rapid development of online services, including social media, search engines, news websites, and electronic preprint websites, people can easily obtain massive textual information. However, it also brings challenges for people to process their acquired texts. It would be a heavy burden to read through all the text content and find out the parts they are interested in.

During the COVID-19 pandemic, hundreds of thousands of scientific articles about the pandemic were published[1], and a flood of news articles about this epidemic also swept most news websites. People were drowned in the torrent of coronavirus papers and news articles, while a large amount of information is not what people care about or are interested in. There is an urgent need to develop advanced tools to help people efficiently process the massive and fast-growing textual information[2].

The automatic text summarization techniques, which aim to produce a concise summary of one or more text documents, can be adopted to alleviate the above problem. On the one hand, high-quality summaries can help people efficiently obtain the key information in original documents. On the other hand, people can first read the summary to determine if one document is worth further reading, which enables people to quickly filter out undesired documents and save a lot of time and effort.

Previous text summarization methods can be generally classified into two categories, namely extractive methods and abstractive methods. The extractive methods select important sentences from input documents to form the summary, while the abstractive methods aim to generate novel sentences as summaries.

[1]https://www.nature.com/articles/d41586-020-03564-y

[2]https://www.sciencemag.org/news/2020/05/scientists-are-drowning-covid-19-papers-can-new-tools-keep-them-afloat

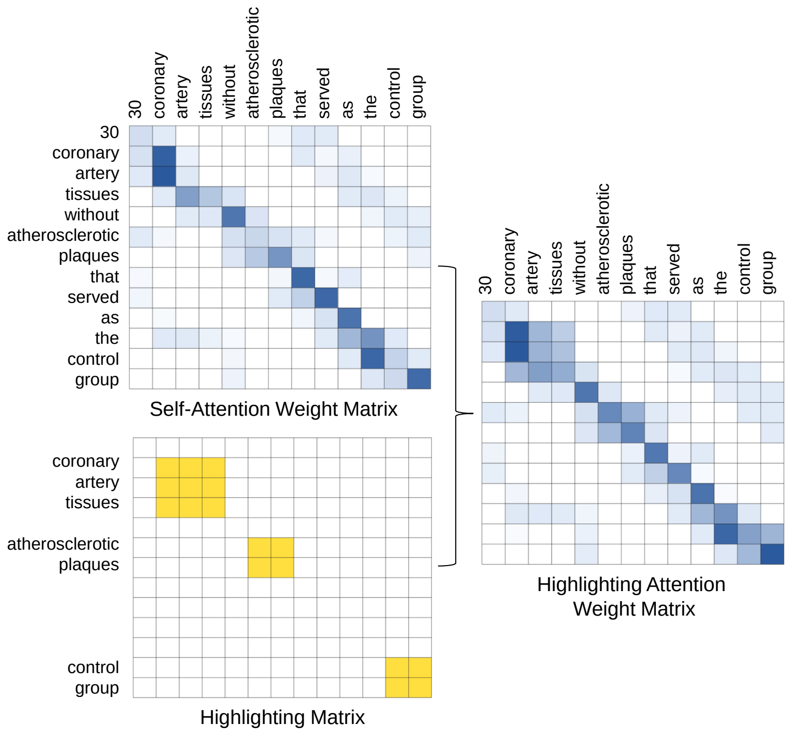

Figure 1: The highlighting mechanism assigns greater attention weights for tokens within key phrases indicated by the highlighting matrix.

This paper focuses on the abstractive summarization models, which also require capturing the salient content from input documents. Compared with the previous RNN-based abstractive summarization models, transformer-based models [16, 28, 29] employ the self-attention mechanism to capture dependencies in input documents, and they can generate better summaries.

Calculating attention weights is a crucial step in the self-attention mechanism. Input documents usually contain some key phrases that convey the salient ideas of input documents. Existing works have not considered key phrases in determining attention weights of self-attention. Key phrases are usually composed of multiple tokens, which should be highly related and serve as a complete grammatical unit in input documents. When testing the existing transformer-based models, we observe some of the tokens within key phrases only receive small attention weights, which can affect completely encoding key phrases and the salient ideas they convey.

In this paper, we propose the Key Phrase Aware Transformer (KPAT), an abstractive summarization model with the highlighting mechanism in the encoder. As shown in Fig.1, the highlighting mechanism assigns greater attention weights for tokens within key phrases. And there are three parts of the highlighting mechanism, including the highlighting matrix, the highlighting attention for each head, and the multi-head highlighting attention.

Our work is inspired by previous studies in education and psychology that indicate key phrases are important for people to understand [39, 18] and summarize [4, 9] the given documents. Highlighting key phrases can help people with dyslexia improve comprehension [39, 18]. Yue et al. [53] suggest a potential benefit of highlighting can be it makes use of a cognitive bias named the Von Restorff effect [49, 37]. The highlighted portion of text stands out from the surrounding non-highlighted text, which makes it more memorable [53]. Their findings can be instructive to improve the attention mechanism in summarization models.

We build a highlighting matrix for each input token sequence to indicate key phrases' positions in the attention weight matrix and phrases' importance scores. Besides, the block-wise linear transformation is adopted on the highlighting matrix to adjust the scale of phrases' importance scores. To combine the self-attention weights with the phrases' importance, we propose two structures of highlighting attention for each head in the KPAT model. After comparing the effects of adopting the highlighting attention in the different numbers of heads and layers, we discover adopting it in a subset of heads surpass adopting it in all heads.

In our experiments, we train and evaluate our model on a multi-document summarization (MDS) dataset named Multi-News [13] and a single document summarization (SDS) dataset named Pub-Med [11]. The automatic evaluation results show that our proposed model significantly improves the ROUGE scores [26] of generated summaries. The results of human evaluation also confirm our model can improve the informativeness of generated model. These experimental results verify the effectiveness of our proposed methods on different summarization tasks (MDS and SDS) and datasets from different domains (news articles and biomedical academic literature).

The rest of this paper is organized as follows. We list our objectives and contribution in Section 2. Section 3 discusses related work and Section 4 briefly introduces the original transformer model. We present our proposed method in Section 5 and our settings of experiments in Section 7. Our experimental results are reported and analyzed in Section 8. Finally, Section 9 concludes this paper and discusses our future work.

2. Objectives and contribution

In this work, our motivation is to enhance the transformer-based abstractive summarization model's ability to completely encoding key phrases that usually convey the salient ideas of input documents. Specifically, we have three objectives:

- To extract key phrases from input documents and score their importance.
- To combine attention weights with the phrase importance.
- To verify the effectiveness of our method on different summarization tasks and datasets from different domains.

The contribution of this work is threefold:

- We present the highlighting mechanism that assigns greater attention weights for tokens within key phrases.
- We propose two structures of highlighting attention for each head and the multi-head highlighting attention to combine attention weights with key phrases' importance.
- Our proposed model significantly outperforms the competitive baseline models on different summarization tasks and datasets from different domains.

3. Related work

3.1. Automatic text summarization

Automatic text summarization aims to reduce the length of input text while preserving the meaning. Previous text summarization methods can be generally classified into two categories: extractive summarization and abstractive summarization.

The extractive summarization methods select a subset of sentences from input documents to form summaries, which maximize the coverage of salient content in input documents while minimizing the redundancy. In the past decades, extractive methods have been extensively studied [12, 32, 30, 42, 33, 3]. But the extracted summaries suffer from problems of coherence and readability [50, 52].

In contrast, the abstractive summarization methods capture and represent the semantic information of input documents and then generate novel sentences as summaries. Compared with extractive methods, the abstractive methods can approximate human-written summaries by merging and compressing information from multiple sentences and generating new expressions not contained in input documents [17, 27].

Some released large-scale datasets for single document summarization (SDS) [20, 35, 11, 44] and multi-document summarization (MDS) [28, 13, 31] make it possible to train large neural models for abstractive summarization.

Previous encoder-decoder models [40, 34, 38, 8] equipped with the attention mechanism [2] have achieved great performance on abstractive summarization. However, they were found to miss some important content in input documents [24, 51]. How to retain the key information of input documents in the generated summaries has received increasing attention in the past few years.

Some previous works focus on improving the copy mechanism. Gehrmann et al. [16] utilize the attention masks to restrict copying phrases from the selected parts of an input document. Xu et al. [51] add words' centrality score to the linearly transformed encoding hidden state when calculating the copy distribution.

Several papers also explore the potential of enhancing the encoder. Li et al. [24, 25] extend the pointer-generator-based models [43] with a separate LSTM-based encoder to get the keywords' representation and then combine it with the sentence representation. In this work, we explore the potential of leveraging phrase importance as guidance to adjust attention weights in the multi-head self-attention of the transformer encoder.

3.2. Key phrase extraction

Key phrase extraction is the task of identifying a set of representative phrases consisting of multiple words from a document. The extracted phrases should reflect the key aspects of an input document [36]. Previous automatic key phrase extraction methods usually contain two steps. First, selecting the candidate phrases and then determining key phrases by using unsupervised or supervised algorithms. Since there are usually no key phrase labels in summarization datasets, we only focus on unsupervised extraction methods, including a statistics-based method and some graph-based ranking methods.

Tf-idf [41] is a widely used statistics-based key phrase extraction method. It scores the candidate phrases according to the following formulas:

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D) \tag{1a}$$

$$\text{tf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{1b}$$

$$\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{1c}$$

The term frequency (tf) describes the frequency of the term t in one sample d and can be calculated by Eq. (1b). In Eq. (1c), the inverse document frequency (idf) is defined as the logarithm of the quotient, which is obtained by dividing the total number of samples N in dataset D by the number of samples containing the term t. Candidate phrases can be ranked according to their tf–idf score, which equals the product of tf and idf. The phrases with the top-N tf–idf scores will be selected as key phrases of each sample.

Graph-based ranking methods first create a graph for a document, in which vertexes represent candidate phrases and edges connect related candidate phrases. Different methods have their ways to score candidate phrases based on the graph and then sort the candidates to select key phrases with top scores. The commonly used graph-based methods include TextRank [32], TopicRank [6], and PositionRank [15]. They first conduct tokenization and part-of-speech tagging on input documents as pre-processing and then utilize a syntactic filter to keep only nouns and adjectives as candidates. In this work, we adopt the TopicRank [6] and PositionRank [15], which significantly outperform the TextRank [32] on many key phrase extraction datasets of news articles and academic literature [6, 15].

PositionRank [15] builds the graph based on words' co-occurrence relations. Based on the idea that key phrases generally occur on positions very close to the beginning of a document or occur frequently, it adopts the position-biased PageRank algorithm [7] to assign larger probabilities to words that are found early or frequently in a given document. In the post-processing phase, adjacent candidate words are concatenated to reconstruct key phrases composed of multiple words. The phrases' scores will be assigned as the sum of words' scores, and the phrases with top scores will be selected as key phrases.

TopicRank [6] groups the candidate phrases into topics by clustering and build the complete graph, where topics are vertices and edges are weighted according to the semantic relations between topics. Topics are scored and ranked by the TextRank algorithm [32]. For each of the most important topics, one of the candidate phrases clustered into that topic will be selected as a key phrase.

4. Preliminaries

4.1. Encoder and decoder in transformer

The transformer model [47] follows the encoder-decoder structure. The encoder maps input sequences to continuous representations. Given the representations of inputs, the decoder is responsible for generating the output sequences.

The transformer encoder consists of N identical layers, and each of them has two sub-layers. The first sub-layer conducts the multi-head self-attention mechanism and the second sub-layer is a position-wise fully connected feed-forward network. The outputs of stacked sub-layers are connected with the residual connection [19] and normalized with layer normalization [1].

$$\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x)) \tag{2}$$

The decoder is also composed of identical N layers. The multi-head self-attention sub-layers mask subsequent positions in attention weight matrices. Compared with the encoder layers, the decoder layers add a sub-layer, which performs the encoder-decoder attention over the output of the encoder and that of the multi-head self-attention sub-layer in the decoder.

4.2. Attention in transformer

The transformer model [47] adopts the multi-head attention in both the encoder and decoder. The multi-head attention mechanism relies on the scaled dot-product attention on each head, which operates on a query Q, a key K, and a value V:

$$\mathrm{Attention}(Q, K, V) = W^m V \tag{3a}$$

$$W^m = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}}) \tag{3b}$$

where $W^m \in \mathbb{R}^{n \times n}$, and d_k is the dimensionality of key.

The multi-head attention employs the scaled dot-product attention on h heads.

$$\begin{aligned} &\mathrm{MultiHead}(Q, K, V) = \mathrm{Heads}W^o \\ &\mathrm{Heads} = \mathrm{Concat}(\mathrm{Head}_1, ..., \mathrm{Head}_h) \\ &\mathrm{Head_i} = \mathrm{Attention}(Q, K, V) \end{aligned} \tag{4}$$

where the matrix Head_i is calculated by Eq. (3a). The results of all the heads will be concatenated and then projected through a feed-forward layer, whose parameter matrix is $W^o \in \mathbb{R}^{hd_v \times d_{model}}$.

In the self-attention layers, all the keys, values, and queries come from the output of the previous layer. While in the encoder-decoder attention layers, the queries come from the previous decoder layer, and the keys and values are from the output of the encoder [47].

5. Proposed method

Our proposed method includes several steps, including data pre-processing, key phrase extraction, building highlighting matrix, and summarization, as depicted in Fig. 2. The procedures of data pre-processing and key phrase extraction will be presented in sub-section 5.1. And sub-section 5.2 will introduce our KPAT model, which comprises

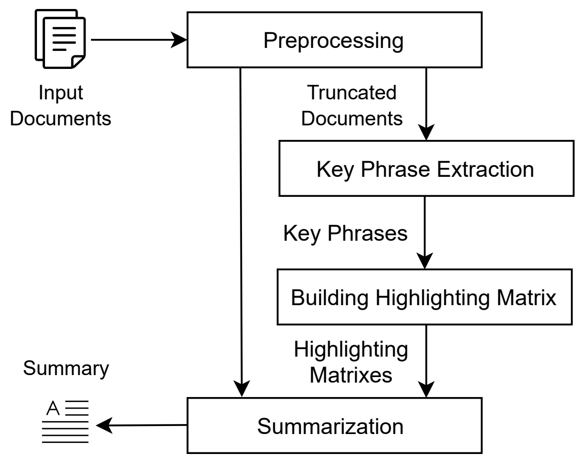

Figure 2: The workflow of proposed method.

the highlighting matrix, the highlighting attention for each head, and the multi-head highlighting attention mechanism.

5.1. Data preparation

We need to prepare the dataset for training and evaluating the proposed summarization model. Each input example of our KPAT model contains the truncated articles, key phrases, and their importance scores. As introduced in sub-section 5.1.1, we first pre-process input documents. And then, the automatic key phrase extraction method can be utilized to assess the phrase importance and select the phrases with top importance scores in sub-section 5.1.2.

5.1.1. Input documents pre-processing

The input documents need to be pre-processed to meet the requirements of neural summarization models. Considering not all the content in these documents is related to the text summarization task, we need to remove irrelevant content. For example, figures and tables should be removed and only preserve the text content.

Since the input length of the neural summarization model is usually limited and shorter than that of input documents, we still need to truncate input documents. For the MDS dataset, we can truncate input documents within each example. In some SDS datasets, single input documents contain multiple parts. And we can truncate these parts considering their contribution to the summary. Except for the length, we also need to change the format of input text content. For example, we need to lowercase all tokens and perform sentence and word tokenization.

More specific operations should depend on the nature of the dataset and the requirements of the summarization model. We will discuss these operations in sub-section 7.1.

5.1.2. Phrase importance assessment

This paper aims to enhance the transformer model's ability to completely encoding key phrases that convey the salient ideas of input documents. As a prerequisite, the phrases' importance should be assessed, and key phrases should be identified. Since there are usually no key phrase labels in the summarization datasets, we utilize unsupervised extraction methods discussed in sub-section 3.2 to score phrases and select the phrases with top scores as key phrases. After removing stopwords, we tried a statistics-based method named tf-idf[3][41], and two graph-based ranking methods, namely the TopicRank [6], and the PositionRank[4] [15].

For each input example comprising one or more documents, we adopt the extractors mentioned above to identify key phrases and use the L2 normalized scores of key phrases as their importance scores. We only select the bigrams and trigrams since longer phrases are sparse and more likely to be compressed in summaries.

After the step of key phrase extraction, we build the highlighting matrix based on the extracted key phrases and their importance scores, and the details will be illustrated in sub-section 5.2.2. We also conduct the experiments to compare the effects of adopting different key phrase extractors and selecting different numbers of key phrases. The experimental results will be reported and analyzed in sub-section 8.4.

5.2. Key phrase aware transformer model

In this section, we introduce the Key Phrase Aware Transformer (KPAT), a model with the highlighting mechanism. We first present the architecture of the KPAT model.

[3]We calculate the tf-idf score by the library named scikit-learn https://scikit-learn.org/stable/index.html
[4]We adopt the implementations of TopicRank and PositionRank from https://github.com/boudinfl/pke

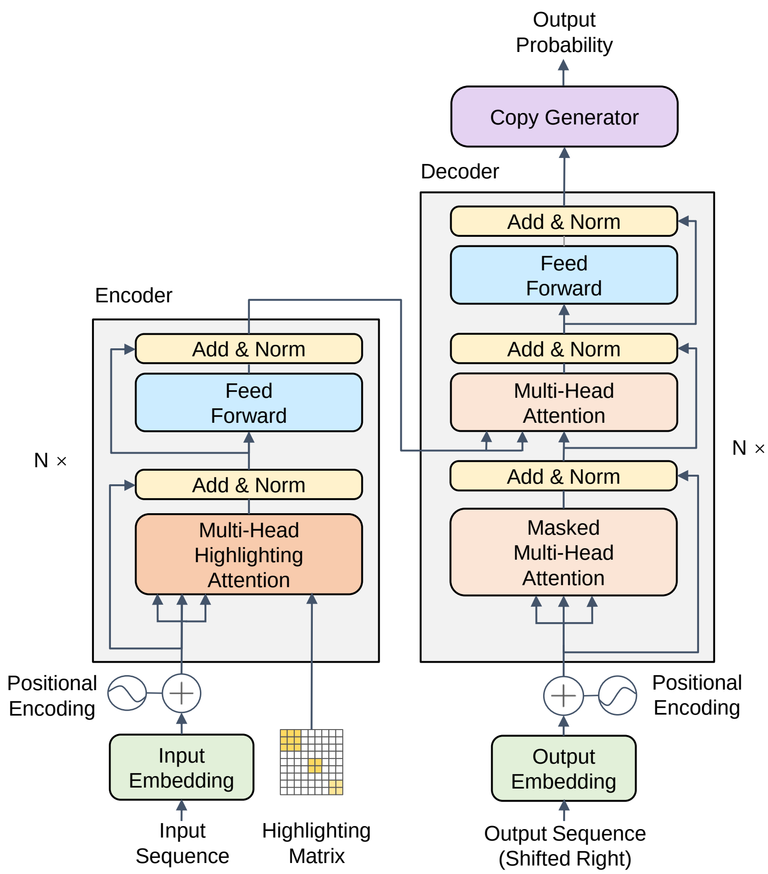

Figure 3: The architecture of the Key Phrase Aware Transformer (KPAT) model.

And then, three key components in the highlighting mechanism, including the highlighting matrix, the highlighting attention for each head, and the multi-head highlighting attention, will be introduced separately.

5.2.1. Model architecture

The KPAT model follows the encoder-decoder structure. In this paper, we mainly focus on the encoder part, since our motivation is to augments the transformer's ability to encode key phrases in input documents. And our decoder follows the copy-transformer model in [16, 13]. Fig. 3 depicts the architecture of the KPAT model.

The encoder of the KPAT model consists of N identical layers. Each encoder layer has two sub-layers: the multi-head highlighting attention layer and the position-wise fully connected feed-forward network. The original transformer model [47] adopts the multi-head self-attention layer in each encoder layer. While in the KPAT model, we replace the multi-head self-attention layers with the multi-head highlighting attention layers, which will be presented in sub-section 5.2.4. Each multi-head highlighting attention layer contains h heads and employs the highlighting attention on p highlighted heads. We depict the highlighting attention in sub-section 5.2.3. The inputs of these encoder layers contain both the output of the previous layer and the highlighting matrix.

5.2.2. *Highlighting matrix*

The first step of the highlighting mechanism is to build a highlighting matrix for each input example based on the results of key phrase extraction. The highlighting matrix can indicate key phrases' positions in the attention weight matrix and the phrases' importance scores.

As described in sub-section 5.1.1, the input example in the MDS dataset is the concatenation of multiple truncated articles, and the example of the SDS dataset can be the truncated single document. Each input example can be represented as an input sequence $(t_1, ..., t_n)$ containing *n* tokens. We use $(p_1, ..., p_k)$ and $(s_1, ..., s_k)$ to denote key phrases and their importance scores. For each input example, we build the highlighting matrix $H \in \mathbb{R}^{n \times n}$ with the same shape as the attention weight matrix.

Assuming a phrase p_r contains *b* tokens in the input sequence $p_r = (x_a, ..., x_{a+b})$, the phrase's importance score s_r is added to the elements $H_{i,j}$, where $i = a, \ldots, a + b, j = a, \ldots, a + b$, in the highlighting matrix. The phrases may be overlapping or nested, and the token t_i may be contained in c phrases $(p_r, ..., p_{r+c})$, whose importance scores are $(s_r, ..., s_{r+c})$. The element H_{ii} is assigned as the maximum value of the c phrases' importance scores. Finally, we will get a block diagonal matrix as the highlighting matrix $H = \mathrm{diag}(H_1, H_2, ..., H_t)$, in which the main-diagonal blocks are square matrices and all off-diagonal blocks are zero matrices, as depicted in Fig. 1.

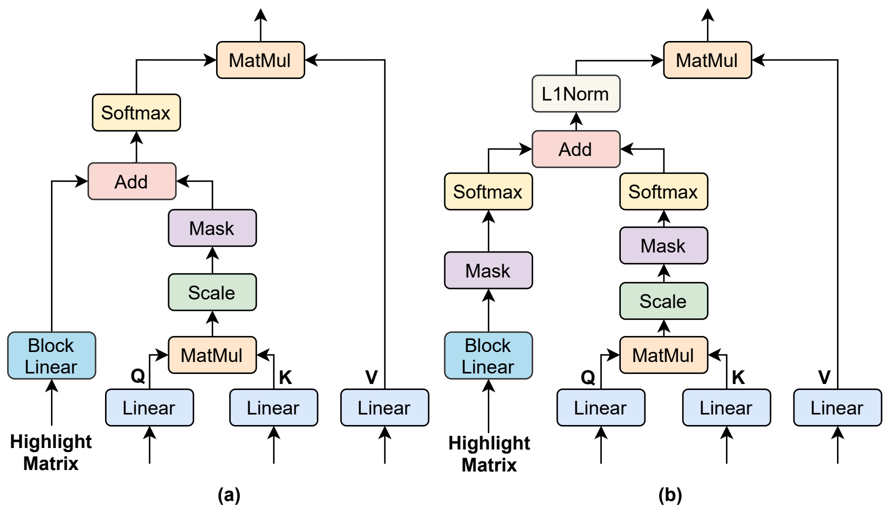

Figure 4: An overview of the proposed highlighting attentions, namely (a) the weighted highlighting attention and (b) the additive highlighting attention.

5.2.3. *Highlighting attention*

The highlighting attention is the crucial component in our model for adjusting attention weights according to the phrase importance. For the head *m*, the original transformer model [47] adopts Eq. (3b) to calculate the scaled dot-product attention.

We propose two structures of highlighting attention, namely the weighted highlighting attention and the additive highlighting attention, to replace the scaled dot-product attention. Two structures of highlighting attention are compared in Fig. 4.

The highlighting attention adjusts attention weights according to the phrase importance. And the highlighting matrix H can be used to determine which elements in the attention weight matrix should be increased.

The weighted highlighting attention mainly modifies Eq. (3b) to calculate the attention weight matrix W^m for the head *m*.

$$W^m = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}} + H^m) \tag{5a}$$

$$H^m = \mathrm{block\_linear}(H) \tag{5b}$$

$$H^m = \mathrm{diag}(\mathrm{linear}(H_1), ..., \mathrm{linear}(H_t)) \tag{5c}$$

Since the softmax function applies the exponential function to each input element and normalizes them through dividing by the sum of all these exponentials. Eq. (6) indicates the additive operation in Eq. (5a) can be identical to calculating the weighted average, so we name it the weighted highlighting attention.

$$\mathrm{softmax}(z_i + b_i) = \frac{e^{b_i} e^{z_i}}{\sum_{j=1}^{n} e^{b_j} e^{z_j}} \quad i = 1, \ldots, n \tag{6}$$

The additive highlighting attention is also designed to adjust the attention weight matrix W^m. The block diagonal highlighting matrix H will be transformed by Eq. (5c). And the result H^m will be normalized by the softmax function[5] and added into the original attention weight matrix $W^{m\prime}$ calculated by Eq. (3b). After that, the matrix W_b^m produced by Eq. (7b) will be normalized along the dimension, where the softmax function is computed, to ensure the sum of elements in this dimension equals one.

$$W^m_{:,\,j} = \frac{W^m_{b\ :,\,j}}{||W^m_{b\ :,\,j}||_1} \quad j = 1, \ldots, n \tag{7a}$$

$$W_b^m = W^{m\prime} + \mathrm{softmax}(H^m) \tag{7b}$$

[5]Since the number of key phrases is limited, and the highlighting matrix can be sparse, we mask the zero elements and only conduct the softmax operation on the nonzero elements.

5.2.4. Multi-head highlighting attention

In our model, the encoder with d_{model} consists of N layers and h heads. Each encoder layer contains the multi-head highlighting attention as a sub-layer. We proposed the multi-head highlighting attention mechanism, which employs the highlighting attention on p highlighted heads and the scaled dot-product attention on the rest of $(h-p)$ normal heads.

$$\begin{aligned} &\mathrm{MultiHead}(Q,K,V) = \mathrm{Heads}W^o \\ &\mathrm{Heads} = \mathrm{Concat}(\mathrm{Head}_1, ..., \mathrm{Head}_h) \\ &\mathrm{Head_i} = \mathrm{Attention}(Q,K,V) \end{aligned} \tag{8}$$

where the projection is a parameter matrix $W^o \in \mathbb{R}^{hd_v \times d_{model}}$. Eq. (3a) calculates the matrix Head_i. The attention weight matrix W of the highlighted heads can be calculated by Eq. (5a) or Eq. (7a), and that of the normal heads can be calculated by Eq. (3b). The results of all the heads will be concatenated and then projected through a feed-forward layer.

6. Datasets

We train and evaluate our model on a MDS dataset named Multi-News [13] and a SDS dataset named Pub-Med [11] to verify the effectiveness of our proposed methods on different summarization tasks (MDS and SDS) and datasets from different domains (news articles and biomedical academic literature).

Multi-News [13] contains summaries of news articles collected from the website newser.com. In this MDS dataset, each example includes multiple news articles collected from diverse news sources about the same event and a summary written by professional editors.

Cohan et al. [11] collected the scientific papers from PubMed and built up a SDS dataset named PubMed. The scientific papers are usually long documents, and the abstracts in these papers can be used as the ground truth summaries. We find the original PubMed dataset fails to separate the abstracts from body sections in some examples. So we remove abstracts from the body sections to avoid target sequences appear in input sequences.

Table 1 summarizes the statistical information of these two datasets. Since Multi-News is a MDS dataset, the length of the input document is calculated on the concatenation of all the input documents in each example. We find the input documents in the PubMed dataset are notably longer than that of the Multi-News dataset. Additionally, biomedical literature's format and content organization are quite different from news articles, so we need to adopt different data pre-processing operations on these two datasets.

Table 1: Statistical information of the two datasets. ”Pairs” denotes the number of examples. And ”Words” denotes the average number of words in the input documents and ground truth summaries

Dataset	**Pairs**	**Words (Doc)**	**Words (Summary)**
Multi-News	56K	2,103	264
PubMed	133K	3,016	203

Table 2: The percentage of examples contain common section names in the PubMed dataset.

Sections	**Train**	**Val**	**Test**
Introduction	78.7%	75.1%	76.3%
Discussion	70.8%	68.5%	69.6%
Result	62.3%	54.9%	56.4%
Conclusion	56.1%	54.2%	55.2%
Methods	58.9%	52.3%	54.0%
Case report	24.4%	28.3%	28.8%
Analysis	24.3%	21.4%	21.6%

7. Experiments

7.1. Data pre-processing

To prepare the text data for training and evaluating the summarization model, we need to remove irrelevant content, filter out some outliers, change the format and length of text content, and split the dataset into training, validation, and test subsets. We lowercase all tokens in two datasets and perform sentence and word tokenization using

NLTK [5]. More specific operations should depend on the nature of the dataset and the requirements of the summarization model.

For the Multi-News dataset, we follow the settings of data preparation in [13], only keep examples with 2-10 input documents per summary. For the neural abstractive models, we take the first 500/S tokens from each article for the example with S articles. If some input documents are shorter than 500/S, we follow [13] and iteratively adjust the quota for each document until reaching the 500-token limit. And then, we concatenate the truncated articles within one example into a single document. We follow [13] to split the dataset into training (80%), validation (10%), and test (10%) sets.

For the PubMed dataset, we follow the settings in [11], first filter out the outliers which are excessively long or too short or do not contain an abstract. In each document, figures and tables are removed. Math formulas and citation markers are normalized with special tokens to preserve only the text content.

Considering the academic papers usually contain multiple sections and each of them contributes differently to the abstract, we need strategies to pre-process these sections differently. We find the sections that appear after the conclusion section, like acknowledgments, conflict of interest, and sponsorship, do not contribute to the content in the abstract, so we should remove these sections.

We count the section names in these papers and find the most common sections, including introduction, discussion, results, conclusion, methods, case report, and analysis. Table 2 summarizes the percentages of examples containing these common sections. Since the concatenation of these common sections can be excessively long, and the input length of neural summarization models is usually limited, we still need to truncate these sections. We first count the number of common sections included in each paper. If one paper contains S common sections, we truncate each common section to 1000/S tokens. If some of the sections are shorter than 1000/S tokens, the excess quota will be equally distributed to other sections.

These truncated sections within one example are concatenated into a single document as the input. We do not find significant performance improvement when increasing input length from 1000 to 2000 tokens, while training the neural models with a larger input size is more time-consuming. Following the settings in [11], we split the dataset into training (90%), validation (5%), and test (5%) sets.

7.2. Experimental setting

We adopt a 4-layer encoder and a 4-layer decoder to build the KPAT model. Each layer has eight attention heads. Both the word embedding size and hidden size are set as 512. The maximum size of the vocabulary is set as 50000 as default. We also use label smoothing [46] with smoothing factor 0.1 and dropout [45] with probability 0.2. The optimizer is Adam [21] with learning rate 2, β_1=0.9 and β_2=0.998. We also adopt the learning rate warmup over the first 8,000 steps and decay as in [47]. During decoding, we use beam search with a beam size of 5. And trigram blocking is used to reduce repetitions. We implement our model with OpenNMT-py [22]. All the models are trained on one NVIDIA QUADRO RTX 8000 GPU.

7.3. Baselines

We compare our proposed KPAT model with the following comparative methods. These methods can be roughly divided into two categories, namely the extractive methods and abstractive methods.

7.3.1. Extractive methods

LexRank and TextRank[6] [12, 32] are two graph-based ranking methods that can be used for extractive summarization. They first build a sentence similarity graph and adopt the idea of PageRank [7] to scores sentences based on the graph. And then, they sort these sentences in descending order of their score and select the top-ranked sentences to form a summary.

Tf-idf scores of words within a sentence can be summed to measure the sentence's importance. An extractive summarization method [10] is built based on this idea. And we use it as a baseline to compare with introducing tf-idf into our abstractive method.

BertExt [30] stacks inter-sentence Transformer layers on top of the pre-trained BERT-base model to capture document-level features. It inserts [CLS] token at the start of

[6]We utilize the implementation of the LexRank model from https://pypi.org/project/lexrank/ and that of the TextRank model from https://radimrehurek.com/gensim_3.8.3/summarization/summariser.html

each sentence and uses the representation of [CLS] from the top layer of the BERT model as sentence representation. We follow settings in [30] and fine-tune the BERT model and inter-sentence transformer layers jointly on the training sets of two datasets.

7.3.2. Abstractive methods

PG, PG-MMR are the pointer-generator network based models reported by Lebanoff et al. [23]. The pointer-generator network [43] allows both copying words via pointing and generating words from a fixed vocabulary. It utilizes the coverage mechanism to discourage repetition.

Hi-MAP [13] expands the existing pointer-generator network into a hierarchical network and calculates sentence-level Maximal Marginal Relevance (MMR) score for each sentence. The attention distribution of tokens within one sentence is multiplied by the MMR score of the sentence to which they belong.

DAA [11] extends the pointer-generator network with discourse-aware attention. It consists of a hierarchical encoder modeling the discourse structure of each input document and an attentive discourse-aware decoder.

CopyTransformer reported in [16, 13] adds the copy mechanism [43] to a 4-layer transformer model. The decoder of our proposed model follows its architecture.

SAGCopy [51] adds words' centrality score to the linearly transformed encoding hidden state when calculating the copy distribution. And it introduces this copy mechanism into the transformer model for abstractive summarization.

BertAbs [30] adopts the pre-trained BERT-base model as the encoder and randomly initializes a decoder comprising six transformer layers. We adopt the settings in [30] and fine-tune the encoder and the decoder on the training sets of two datasets.

7.4. Evaluation metrics

We use the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) F_1 scores [26] as the automatic evaluation metrics. Specifically, we report the overlap of unigrams (R-1), bigrams (R-2), and skip-bigram with unigrams (R-SU) between system-generated summaries and gold references provided by summarization datasets.

ROUGE-N is a statistic on n-gram co-occurring in both a candidate summary and a set of reference summaries. And it can be calculated as follows:

$$\text{R-N}_{\text{r}} = \frac{\sum_{S\in\text{ref}} \sum_{\text{gram}_N \in S} \text{Count}_{\text{m}}(\text{gram}_N)}{\sum_{S\in\text{ref}} \sum_{\text{gram}_N \in S} \text{Count}(\text{gram}_N)} \tag{9a}$$

$$\text{R-N}_{\text{p}} = \frac{\sum_{S\in\text{ref}} \sum_{\text{gram}_N \in S} \text{Count}_{\text{m}}(\text{gram}_N)}{\sum_{S\in\text{cand}} \sum_{\text{gram}_N \in S} \text{Count}(\text{gram}_N)} \tag{9b}$$

$$\text{R-N}_{\text{F1}} = \frac{2 \times \text{R-N}_{\text{p}} \times \text{R-N}_{\text{r}}}{\text{R-N}_{\text{p}} + \text{R-N}_{\text{r}}} \tag{9c}$$

Where *N* stands for the length of the n-gram. The result of $\text{Count}_{\text{m}}(\text{gram}_N)$ is the maximum number of n-grams co-occurring in both a candidate summary and a set of reference summaries. R-N_{r}, R-N_{p}, and R-N_{F1} represent the recall, precision, and F1 score of ROUGE-N. We report the F1 scores of ROUGE-1 (R-1) and ROUGE-2 (R-2) in the following tables, which reflect the coverage of unigrams and bigrams. They can be regarded as means of assessing the informativeness of the generated summaries, compared with the human-written summaries.

ROUGE-SU is a statistic on skip-bigram with unigrams co-occurrence. A skip-bigram is an ordered pair of words in a sentence allowing for arbitrary gaps between them. Given a sentence comprising multiple words $\text{sent}_i = [w_1, w_2, ..., w_n]$ in a candidate summary. The pair of words within the sentence (w_{j1}, w_{j2}) is a skip-bigram if $j1 < j2$. ROUGE-S does not require consecutive matching but is still sensitive to word order [26]. It counts all in-order matching word pairs and can be computed as follows:

$$\text{R-S}_{\text{r}} = \frac{\text{SKIP}(X, Y)}{C(m, 2)} \tag{10a}$$

$$\text{R-S}_{\text{p}} = \frac{\text{SKIP}(X, Y)}{C(n, 2)} \tag{10b}$$

$$\text{R-S}_{\text{F1}} = \frac{(1+\beta^2) R_{\text{skip}} P_{\text{skip}}}{R_{\text{skip}} + \beta^2 P_{\text{skip}}} \tag{10c}$$

$$\text{SKIP}(X, Y) = \sum_{S\in X} \sum_{\text{s-gram}_i \in S} \text{Count}_{\text{m}}(\text{s-gram}_i) \tag{10d}$$

Where s-gram_{i} is the i-th skip-bigram, *m* and *n* stand for the length of reference summary X and generated candidate summary Y. $C(m,2)$ and $C(n,2)$ are the numbers of skip-bigrams in X and Y. $\mathrm{SKIP}(X,Y)$ is the number of matched skip-bigrams between X and Y.

However, ROUGE-S does not consider the generated sentences may not include any word pair co-occurring with its references. ROUGE-SU extends the ROUGE-S by adding unigram as a counting unit. It can be implemented by adding a marker at the beginning of candidate and reference sentences [26].

$$\text{ROUGE-SU}(X,Y) = \text{ROUGE-S}(X^+,Y^+) \tag{11a}$$

$$\mathrm{SKIP}(X^+,Y^+) = \mathrm{SKIP}(X,Y) + \text{Uni-CNT} \tag{11b}$$

$$\text{Uni-CNT} = \sum_{S\in X}\sum_{\mathrm{1gram}_i\in S}\mathrm{Count_m}(\mathrm{1gram}_i) \tag{11c}$$

Where X^+ and Y^+ denote the reference and the candidate adding a start token. Uni-CNT is the maximum number of unigrams co-occurring in both the candidate and the reference summary.

8. Results and discussion

8.1. Automatic evaluation results

For the Multi-News dataset, the results of LexRank, TextRank, PG, PG-MMR, Hi-MAP, and CopyTransformer on the Multi-News test set follow Fabbri et al. [13]. For the PubMed dataset, we train and evaluate all the models since we choose a different truncation strategy compared with the original scheme provided by Cohan et al. [11] and remove the abstracts from body sections in some examples that fail to separate the abstract and body sections.

Two additional extractive baselines are evaluated in our experiments. A tf-idf based extractive method [10] is adopted as a baseline to compare with introducing the tf-idf score into our abstractive model. We also fine-tune and evaluate a BERT-based extractive method [30], which is more powerful than those unsupervised extractive baselines mentioned above.

In addition to the abstractive baselines mentioned in [13, 11], we also evaluate some additional abstractive baselines on two datasets. A BERT-based abstractive summarization method [30] is fine-tuned on our training sets. Another transformer-based abstractive method named SAGCopy [51] discussed in subsection 7.3 is also trained and evaluated on these two datasets.

Table 3: Evaluation results on the Multi-News test set.

Method	**R-1**	**R-2**	**R-SU**
LexRank	38.27	12.70	13.20
TextRank	38.44	13.10	13.50
tf-idf	38.68	12.09	13.54
BertExt	44.27	15.09	17.44
PG	41.85	12.91	16.46
PG-MMR	40.55	12.36	15.87
Hi-MAP	43.47	14.89	17.41
BertAbs	42.21	15.14	16.33
SAGCopy	43.98	15.21	17.65
CopyTransformer	43.57	14.03	17.37
KPAT (Weighted)	**45.30**	**15.96**	**18.62**
KPAT (Additive)	44.37	15.55	17.77

Table 4: Evaluation results on the PubMed test set.

Method	**R-1**	**R-2**	**R-SU**
LexRank	35.78	14.75	11.35
TextRank	36.41	14.97	11.90
tf-idf	33.67	9.18	10.74
BertExt	37.72	13.95	12.48
PG	38.37	13.59	14.72
DAA	38.95	15.41	15.63
BertAbs	39.29	15.59	15.84
SAGCopy	38.66	15.24	15.35
CopyTransformer	38.81	14.99	15.39
KPAT (Weighted)	**40.04**	**15.82**	**16.24**
KPAT (Additive)	39.67	15.61	15.94

Table 3 and Table 4 summarize the automatic evaluation results on the test sets of Multi-News and PubMed. The ”KPAT (Weighted)” denotes the KPAT model equipped with the weighted highlighting attention mechanism on each head, and the "KPAT (Additive)" represents the KPAT model equipped with the additive highlighting attention mechanism. Our proposed model significantly outperforms these baseline models on all metrics. These results prove the effectiveness of the highlighting mechanism on different summarization tasks (MDS and SDS) and datasets from different domains (news articles and biomedical academic literature). Besides, the weighted highlighting attention is more favorable compared with the additive highlighting attention.

Table 5: Human evaluation results on the Multi-News test set. "Win" represents the generated summary of our KPAT model is better than that of CopyTransformer in one aspect.

	Win	**Lose**	**Tie**	**kappa**
Informativeness	46.5%	21.5%	32.0%	0.664
Fluency	29.5%	26.0%	44.5%	0.639
Non-Redundancy	27.5%	25.5%	47.0%	0.624

Table 6: Human evaluation results on the PubMed test set. "Win" represents the generated summary of our KPAT model is better than that of CopyTransformer in one aspect.

	Win	**Lose**	**Tie**	**Kappa**
Informativeness	43.0%	19.5%	37.5%	0.659
Fluency	27.0%	25.0%	48.0%	0.622
Non-Redundancy	23.5%	19.0%	57.5%	0.631

8.2. Human evaluation results

In addition to automatic evaluation, we performed the human evaluation to compare the generated summaries in terms of informativeness (the coverage of information from input documents), fluency (content organization and grammatical correctness), and non-redundancy (less repetitive information). We randomly select 50 samples from the test sets of Multi-News and PubMed respectively. Four annotators are required to compare two models' generated summaries that are presented anonymously. We assess their agreements by Fleiss' kappa [14].

The evaluation results in Table 5 and Table 6 suggest our proposed model significantly outperforms the CopyTransformer in terms of informativeness and is comparative in terms of fluency and non-redundancy on these two datasets.

8.3. *Impact of the multi-head highlighting attention*

We compare the effects of adopting the weighted highlighting attention in different numbers of heads and layers in the encoder of our proposed model. In this experiment, we adopt the weighted highlighting attention mechanism on each head of our KPAT model. The results on the test set of Multi-News are summarized in Table 7. It reveals that adopting it in a quarter of the heads and half of layers achieves the best performance. We discover adopting highlighting attention in a subset of heads surpass adopting it in all heads. Applying the multi-head highlighting attention on all layers of the encoder is also not optimal. One possible reason is that the different heads and layers in the transformer encoder attend to different types of information.

Multi-head attention in the transformer model [47] is designed for jointly attending to information from different representation sub-spaces. Voita et al. [48] find the heads in transformer model trained on the neural machine translation dataset have one or more specialized functions and focus on different types of information, including the adjacent tokens, syntactic relations, and rare words. Adopting the highlighting attention in all heads and layers may affect the transformer-based model to encode other types of useful information and lead to performance degradation.

Table 7: Evaluation results on highlighting different numbers of heads and layers.

KPAT (Weighted)	**R-1**	**R-2**	**R-SU**
1/4 Heads 1/2 Layers	**45.30**	**15.96**	**18.62**
1/2 Heads 1/2 Layers	44.61	15.60	18.16
All Heads 1/2 Layers	44.42	15.36	17.92
1/4 Heads All Layers	44.58	15.43	18.02
1/2 Heads All Layers	44.67	15.54	18.11
All Heads All Layers	44.35	15.23	17.90

8.4. *Impact of the key phrase extractor*

We compare the performance of introducing different key phrase extractors' results and different numbers of key phrases into our proposed model. Since there are usually no key phrase labels in the summarization datasets, we only focus on unsupervised extraction methods. We adopt and compare the tf-idf [41] based extractor and two graph-based ranking methods: TopicRank [6] and PositionRank [15]. The extracted key phrases and their importance scores can be used to build the highlighting matrices and then integrated into our abstractive summarization model.

The evaluation results in Table 8 and Table 9 suggest that introducing key phrases extracted by the PositionRank algorithm can achieve the best results on the two datasets. As discussed in sub-section 3.2, PositionRank assigns larger probabilities to words found early or frequently in a given document. It can meet the phenomenon that key-phrases generally occur on positions close to the beginning of a document and occur frequently [15].

When it comes to the number of key phrases for each example, selecting the top-10 key phrases performs well on the PubMed dataset. Considering PubMed is a SDS dataset, ten key phrases can be enough for one input document. But it seems not enough for the multiple input documents in each example since the multiple input documents may contain more information and key phrases. And we discover selecting the top-20 key phrases performs better on the Multi-News dataset.

Table 8: Impact of different key phrases selection settings on the Multi-News test set.

Key phrase extractor	R-1	R-2	R-SU
tf-idf (top-10)	44.56	15.63	18.00
tf-idf (top-20)	44.84	15.80	18.21
TopicRank (top-10)	44.53	15.29	17.97
TopicRank (top-20)	45.24	15.93	18.56
PositionRank (top-10)	44.70	15.73	18.12
PositionRank (top-20)	**45.30**	**15.96**	**18.62**

8.5. Ablation study

The ablation studies aim to validate the effectiveness of individual components in our proposed model. Table 10 and Table 11 summarize the results of ablation studies on the two datasets. The results confirm that incorporating the highlighting attention and the block-wise linear transformation on the block diagonal highlighting matrix is beneficial for both single document summarization and multi-document summarization.

Table 9: Impact of different key phrases selection settings on the PubMed test set.

Key phrase extractor	R-1	R-2	R-SU
tf-idf (top-10)	39.88	15.70	16.06
tf-idf (top-20)	39.43	15.55	15.96
TopicRank (top-10)	39.48	15.73	15.91
TopicRank (top-20)	39.28	15.58	15.85
PositionRank (top-10)	**40.04**	**15.82**	**16.24**
PositionRank (top-20)	39.64	15.70	15.99

Table 10: Ablation study on the Multi-News test set. ”block linear” denotes the block-wise linear transformation on the block diagonal highlighting matrix

	R-1	R-2	R-SU
KPAT model	**45.30**	**15.96**	**18.62**
w/o block linear	44.62	15.57	18.06
w/o highlighting attention	43.57	14.03	17.37
w/o self-attention	42.54	14.40	16.54

Table 11: Ablation study on the PubMed test set. ”block linear” denotes the block-wise linear transformation on the block diagonal highlighting matrix

	R-1	R-2	R-SU
KPAT model	**40.04**	**15.82**	**16.24**
w/o block linear	39.68	15.62	15.95
w/o highlighting attention	38.81	14.99	15.39
w/o self-attention	37.63	14.87	15.14

We also tried replacing the self-attention in a quarter of the heads and half of layers with the highlighting matrices directly. And the performance degradation reveals that it is important to combine the attention weight with the phrase importance.

9. Conclusion and future work

In this paper, we propose the Key Phrase Aware Transformer (KPAT), a novel abstractive summarization model with the highlighting mechanism in the encoder to assign greater attention weights for tokens within key phrases. The highlighting mechanism mainly comprises three parts: the highlighting matrix, the highlighting attention, and the multi-head highlighting attention. We build a block diagonal highlighting matrix for each input token sequence and adopt the block-wise linear transformation on the highlighting matrix to adjust the scale of phrases' importance scores. For each head in the KPAT model, we propose and compare two structures of highlighting attention. Besides, we also compare the effects of adopting the highlighting attention in different numbers of heads and layers in the encoder of our KPAT model. The experimental results exhibit the effectiveness of our proposed model on different summarization tasks and datasets from different domains. In future work, we intend to incorporate multi-granularity features of input documents, including the phrase-level, sentence-level, paragraph-level, and document-level features, into the transformer-based summarization models and evaluate them on different datasets.

References

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[3] R. C. Belwal, S. Rai, and A. Gupta. Text summarization using topic-based vector space model and semantic measure. *Information Processing & Management*, 58 (3):102536, 2021.

[4] A. Benzer, A. Sefer, Z. Ören, and S. Konuk. A student-focused study: Strategy of text summary writing and assessment rubric. *Education & Science/Egitim ve Bilim*, 41(186), 2016.

[5] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[6] A. Bougouin, F. Boudin, and B. Daille. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International joint conference on natural language processing (IJCNLP)*, pages 543–551, 2013.

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[8] S. Chopra, M. Auli, and A. M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *HLT-NAACL*, 2016.

[9] M.-h. Chou. Implementing keyword and question generation approaches in teaching efl summary writing. *English Language Teaching*, 5(12):36–41, 2012.

[10] H. Christian, M. P. Agus, and D. Suhartono. Single document automatic text summarization using term frequency-inverse document frequency (tf-idf). *ComTech: Computer, Mathematics and Engineering Applications*, 7(4):285–294, 2016.

[11] A. Cohan, F. Dernoncourt, D. S. Kim, T. Bui, S. Kim, W. Chang, and N. Goharian. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of NAACL-HLT*, pages 615–621, 2018.

[12] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479, 2004.

[13] A. R. Fabbri, I. Li, T. She, S. Li, and D. Radev. 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*, pages 1074–1084, 2019.

[14] J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[15] C. Florescu and C. Caragea. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115, 2017.

[16] S. Gehrmann, Y. Deng, and A. M. Rush. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, 2018.

[17] S. Gupta and S. Gupta. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 121:49–65, 2019.

[18] S. Hargreaves and J. Crabb. *Study Skills for Students with Dyslexia: Support for Specific Learning Differences (SpLDs)*. Sage, 2016.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE Computer Society, 2016.

[20] K. M. Hermann, T. Kočiskỳ, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1693–1701, 2015.

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[22] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, 2017.

[23] L. Lebanoff, K. Song, and F. Liu. Adapting the neural encoder-decoder framework from single to multi-document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4131–4141, 2018.

[24] C. Li, W. Xu, S. Li, and S. Gao. Guiding generation for abstractive text summarization based on key information guide network. In *NAACL-HLT*, 2018.

[25] H. Li, J. Zhu, J. Zhang, C. Zong, and X. He. Keywords-guided abstractive sentence summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8196–8203, 2020.

[26] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[27] H. Lin and V. Ng. Abstractive summarization: A survey of the state of the art. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9815–9822, 2019.

[28] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

[29] Y. Liu and M. Lapata. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, 2019.

[30] Y. Liu and M. Lapata. 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)*, pages 3721–3731, 2019.

[31] Y. Lu, Y. Dong, and L. Charlin. Multi-xscience: A large-scale dataset for extreme multi-document summarization of scientific articles. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8068–8074, 2020.

[32] R. Mihalcea and P. Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004.

[33] B. Mutlu, E. A. Sezer, and M. A. Akcayol. Candidate sentence selection for extractive text summarization. *Information Processing & Management*, 57(6): 102359, 2020.

[34] R. Nallapati, B. Zhou, C. D. Santos, Çaglar Gülçehre, and B. Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*, 2016.

[35] C. Napoles, M. R. Gormley, and B. Van Durme. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100, 2012.

[36] E. Papagiannopoulou and G. Tsoumakas. A review of keyphrase extraction. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(2):e1339, 2020.

[37] A. Parker, E. Wilding, and C. Akerman. The von restorff effect in visual object recognition memory in humans and monkeys: The role of frontal/perirhinal interaction. *Journal of cognitive neuroscience*, 10(6):691–703, 1998.

[38] R. Paulus, C. Xiong, and R. Socher. A deep reinforced model for abstractive summarization. *ArXiv*, abs/1705.04304, 2018.

[39] L. Rello, H. Saggion, and R. Baeza-Yates. Keyword highlighting improves comprehension for people with dyslexia. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 30–37, 2014.

[40] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, 2015.

[41] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[42] Y. Sankarasubramaniam, K. Ramanathan, and S. Ghosh. Text summarization using wikipedia. *Information Processing & Management*, 50(3):443–461, 2014.

[43] A. See, P. J. Liu, and C. D. Manning. 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)*, pages 1073–1083, 2017.

[44] E. Sharma, C. Li, and L. Wang. Bigpatent: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213, 2019.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[46] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[48] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, 2019.

[49] H. Von Restorff. Über die wirkung von bereichsbildungen im spurenfeld. *Psychologische Forschung*, 18(1):299–342, 1933.

[50] X. Wang, M. Nishino, T. Hirao, K. Sudoh, and M. Nagata. Exploring text links for coherent multi-document summarization. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 213–223, 2016.

[51] S. Xu, H. Li, P. Yuan, Y. Wu, X. He, and B. Zhou. Self-attention guided copy mechanism for abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1355–1362, 2020.

[52] J.-g. Yao, X. Wan, and J. Xiao. Recent advances in document summarization. *Knowledge and Information Systems*, 53(2):297–336, 2017.

[53] C. L. Yue, B. C. Storm, N. Kornell, and E. L. Bjork. Highlighting and its relation to distributed study and students' metacognitive beliefs. *Educational Psychology Review*, 27(1):69–78, 2015.