Neural Abstractive Multi-Document Summarization: Hierarchical or Flat Structure?

Ye Ma, Lu Zong
Mathematical Sciences, Xi'an Jiaotong-Liverpool University, Suzhou, 215028, China
{ye.ma, lu.zong}@xjtlu.edu.cn

Abstract

With regards to WikiSum (Liu et al., 2018b) that empowers applicative explorations of Neural Multi-Document Summarization (MDS) to learn from large scale dataset, this study develops two hierarchical Transformers (HT) that describe both the cross-token and cross-document dependencies, at the same time allow extended length of input documents. By incorporating word- and paragraph-level multi-head attentions in the decoder based on the parallel and vertical architectures, the proposed parallel and vertical hierarchical Transformers (PHT &VHT) generate summaries utilizing context-aware word embeddings together with static and dynamics paragraph embeddings, respectively. A comprehensive evaluation is conducted on WikiSum to compare PHT &VHT with established models and to answer the question whether hierarchical structures offer more promising performances than flat structures in the MDS task. The results suggest that our hierarchical models generate summaries of higher quality by better capturing cross-document relationships, and save more memory spaces in comparison to flat-structure models. Moreover, we recommend PHT given its practical value of higher inference speed and greater memory-saving capacity. [1]

1 Introduction

With the promising results achieved by neural abstractive summarization on single documents (See et al., 2017; Cao et al., 2018; Liu et al., 2018a; Gehrmann et al., 2018), an increasing number of attempts are made to study abstractive multi-document summarization (MDS) using seq2seq models (Liu et al., 2018b; Lebanoff et al., 2018; Fabbri et al., 2019; Liu and Lapata, 2019). Compared with the single-document summarization, multi-document summarization places challenges in two primary aspects, that is representing large source documents and capturing cross-document relationships. To address the former issue, Liu et al. (2018b) adopts a two-stage approach by first selecting a list of important paragraphs from all documents in an extractive framework. Then a modified language model based on the Transformer-decoder with memory compressed attention (T-DMCA) is developed to conduct abstractive summarization after concatenating the extracted paragraphs to a flat sequence. Although the proposed flat structure of T-DMCA demonstrates both theoretical and practical soundness to learn long-term dependencies, it fails to implant the cross-document relationship in its summaries. On the other hand, the encoder-decoder structure that allows hierarchical inputs of multiple documents offers not only another solution to the long-text summarization problem (Li et al., 2018; Zhang et al., 2019; Liu and Lapata, 2019) but also allows cross-document information exchange in the produced summaries. In particular, Liu and Lapata (2019) proposes a Hierarchical Transformer with local and global encoder layers to represent cross-token and cross-paragraph information, which are both utilized later to enrich token embeddings. Summaries are then generated based on a vanilla Transformer (Vaswani et al., 2017) by concatenating enriched token embeddings from different documents to a flat sequence. Such Hierarchical Transformer though captures cross-document relationships, the essentially-flat Transformer it adopts fails to learn dependencies of sequences longer than 2000 tokens according to Liu et al. (2018b).

In this paper, we develop two novel hierarchical Transformers to address both the text-length and cross-document linkage problems in MDS. By introducing the word-level and paragraph-level multi-head attention mechanisms, our models are designed to learn both cross-token and cross-

[1] https://github.com/yema2018/wiki_sum

Proceedings of the 2nd International Workshop on Discourse Processing, pages 29–37
Suzhou, China, December 7, 2020.

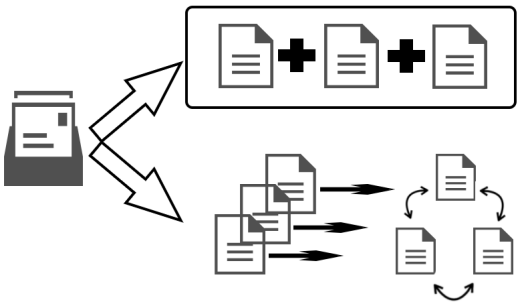

Figure 1: Flat structure (top) – concatenating documents to a flat sequence. Hierarchical structure (bottom)– hierarchical input and representation of documents + modeling cross-document relationships.

document relationships. The word- and paragraph-level context vectors are then jointly used to generate target sequences in order to abandon the flat structure, thus to mitigate the long-dependency problem. In detail, both of the proposed hierarchical architectures are based on the Transformer encoder-decoder model (Vaswani et al., 2017), with context-aware word embeddings obtained from a shared encoder and cross-token linkages described by the word-level multi-head attention mechanism in the decoder. The difference lies in the way that the document-level information is handled. Based on the *static* [2] paragraph embeddings computed from the context-aware word embeddings, the parallel hierarchical Transformer (PHT) models cross-document relationships with paragraph-level multi-head attention parallel to the word-level multi-head attention. The paragraph attentions are then used to normalize the word attentions. On the other hand, the vertical hierarchical Transformer (VHT) stacks the paragraph-level attention layer on top of the word-level attention layer in order to learn the latent relationship between paragraphs with *dynamic* [3] paragraph embeddings from the previous layer.

To evaluate the performance of the proposed models as well as to compare flat and hierarchical structures in the MDS task, we select several strong baselines covering abstractive models of flat strucuture (T-DMCA (Liu et al., 2018b) and Transformer-XL (Dai et al., 2019)) and of hierarchical structure (Liu's hierachical Transformer (Liu and Lapata, 2019)). A systematic analysis is conducted on the WikiSum dataset according to four criteria including the models' abilities of capturing cross-document relationships, ROUGE evaluation, human evaluation and computational efficiency. The results show that PHT&VHT outperform other baselines significantly with memory space.

[2] *static* means the embedding remains the same for different time steps in the decoder.

[3] *dynamic* means the embeddings are dynamic for different time steps in the decoder.

2 Related work

Neural multi-document summarization Regarding to extractive models, neural networks are the most widely-used approach to model in- and cross-document knowledge with the objective to minimize the distance between the selected sentence set and the gold summary (Cao et al., 2017; Li et al., 2017; Ma et al., 2016; Nallapati et al., 2016; Yasunaga et al., 2017). One representative study (Yasunaga et al., 2017) is to construct a graph of the document cluster based on the similarities between sentences. Graph Neural Network (GNN) (Kipf and Welling, 2016) is then employed to select salient sentences. Argued by Liu and Lapata (2019), self-attention is a better mechanism to learn the latent dependency among documents than GNNs. As for abstractive models, studies tend to extract important paragraphs from different documents followed by a abstractive seq2seq model to generate summaries (Liu et al., 2018b; Liu and Lapata, 2019; Fabbri et al., 2019). Additionally, Chu and Liu (2019) adopts an auto-encoder model to conduct MDS in an unsupervised way.

Hierarchical neural network Hierarchical neural document models are applied in various fields of NLP such as document auto-encoder (Li et al., 2015) or text classification (Yang et al., 2016). In the area of abstractive summarization, Li et al. (2018) extends a hierarchical RNN encoder-decoder (Lin et al., 2015) with the hybrid sentence-word attention. Instead of trainable attention machanisms, Fabbri et al. (2019) hires a hierarchical RNN with Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) to represent the relationship between sentences. Liu and Lapata (2019) proposes a hierarchical Transformer by incorporating a global self-attention to represent cross-document relationships. Moreover, Zhang et al. (2019) constructs a hierarchical BERT (Devlin et al., 2018) to learn the context relationships among sentences by using other sentences to generate the masked sentence.

3 Hierarchical Transformer

This paper proposes two hierarchical Transformers with parallel & vertical architectures, respectively. Section 3.1 explicitly explains the construction of the parallel hierarchical Transformer (PHT) and its application in MDS, whereas Section 3.2 places emphasis on explaining the structural differences between the vertical hierarchical Transformer (VHT) and PHT.

3.1 Parallel hierarchical Transformer

3.1.1 Encoder

As shown in Figure 2, the PHT encoder is shared by all paragraphs and consist of two major units, i.e. the transformer encoder and the **Multi-head Attention Pooling** layer, to obtain the token- and paragraph-embeddings. To be specific, context-aware word embeddings are first produced as the output of the transformer encoder based on the summation of word embeddings W and fixed positional encodings (Vaswani et al., 2017).

$$C_p = TransE(W_p + E_p) \tag{1}$$

where $C_p \in \mathbb{R}^{n \times d}$ denotes context-aware word embeddings in the paragraph p and n is the paragraph length. We select the fixed encoding method rather than other learning models given that the former has the capacity to deal with sequences of arbitrary length. The context-aware word embedding is then used to generate paragraph embeddings as well as being a part of inputs to the PHT decoder.

As the second step, the parallel architecture generates additional *static* paragraph embeddings to model cross-document relationships from the multi-head attention pooling:

$$head_p^i = HeadSplit(C_p W_1) \tag{2}$$

$$\phi_p^i = (Softmax(head_p^i W_2))^T head_p^i \tag{3}$$

$$\phi_p = W_3[\phi_p^0; \phi_p^1; \cdots] \tag{4}$$

$$\phi_p := layerNorm(\phi_p + FFN(\phi_p)) \tag{5}$$

where $W_1 \in \mathbb{R}^{d \times d}$, $W_2 \in \mathbb{R}^{d_{head} \times 1}$ and $W_3 \in \mathbb{R}^{d \times d}$ are linear transformation parameters, $head_p^i \in \mathbb{R}^{n \times d_{head}}$ and $\phi_p^i \in \mathbb{R}^{d_{head}}$ denote the i_{th} attention head and paragraph embedding. These head embeddings are concatenated and fed to a two-layer feed forward network (FFN) with Relu activation function after linear transformation. The paragraph embedding is another input to the decoder, together with the context-aware word embedding.

3.1.2 Decoder

The PHT decoder accepts three classes of inputs, namely the target summary, context-aware word embeddings in the p_{th} paragraph $C_p \in \mathbb{R}^{n \times d}$ where **n** is the length of the paragraph, and *static* paragraph embeddings $\Phi \in \mathbb{R}^{m \times d}$ where **m** is the number of paragraphs. Let $X^1 \in \mathbb{R}^{k \times d}$ denote the output of part I where **k** is the length of target sequence or the number of time steps. Note that both the word embedding and vocabulary in the decoder part I are shared with the encoder.

Paragraph embeddings are added with the ranking encoding R generated by the positional encoding function (Vaswani et al., 2017):[4]

$$\Phi := \Phi + R \tag{6}$$

Different from the token-level ranking encoding (Liu and Lapata, 2019), we intend to incorporate the positional information of paragraphs to their embeddings.

The PHT decoder consists of three parts. Similar to a vanilla Transformer (Vaswani et al., 2017), the first and last parts of the PHT decoder are the masked multi-head attention and the feed forward network, whereas the second part includes two parallel multi-head attentions to capture the inter-word and inter-paragraph relations.

Paragraph-level multi-head attention: This self-attention mechanism is to create paragraph-level context vectors that represent the latent cross-paragraph relationships. The query is the output of part I: X^1, whilst the key and value are *static* paragraph embeddings Φ:

$$X^{para}, A^{para} = MultiHead(X^1, \Phi, \Phi), \tag{7}$$

where $X^{para} \in \mathbb{R}^{k \times d}$ is the paragraph-level context vector and $A^{para} \in \mathbb{R}^{k \times m}$ denotes the attention weights of paragraphs[5]. Both X^{para} and A^{para} are comprised of representations of all time steps.

Word-level multi-head attention: This shared self-attention mechanism is to output word-level context vectors which represent the cross-token dependency for each paragraph. Since there are m paragraphs, so the mechanism is implemented m times at each time step. The query of self attention

[4] We directly use the ranked paragraphs provided by Liu and Lapata (2019)

[5] In this paper, average pooling is adopted to compute the final attention from multi-head attentions

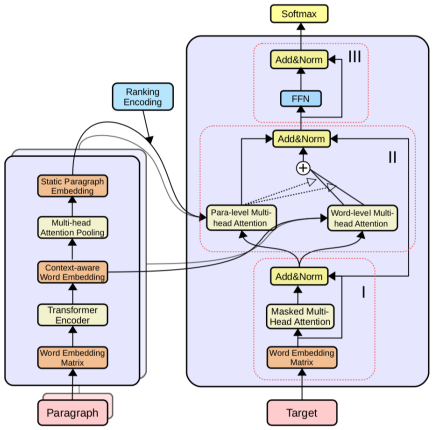

Figure 2: PHT –**shared** encoders on the left and the decoder on the right.

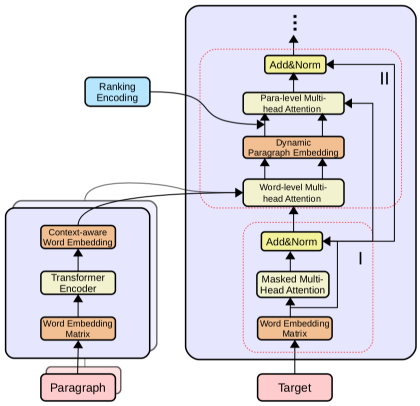

Figure 3: VHT – removing attention pooling in the encoder and using the vertical architecture in the decoder.

is X^1, whilst the key and value are context-aware word embeddings C_p.

$$X_p^{word} = MultiHead(X^1, C_p, C_p), \quad (8)$$

where $X_p^{word} \in \mathbb{R}^{k \times d}$ denotes the word-level context vectors of all time steps in the p_{th} paragraph.

The outputs $X^{word} \in \mathbb{R}^{k \times d \times m}$ are integrated by first being normalized by paragraph attentions A^{para}, then propagated to subsequent layers after summation.

$$X^{int} = X^{word} A^{para} \quad (9)$$

where the dimension of A^{para} is expanded to $\mathbb{R}^{k \times m \times 1}$ and matrices are multiplied in the last two dimensions so $X^{int} \in \mathbb{R}^{k \times d}$. The output of part II: X^2 is written as:

$$X^2 = LayerNorm(X^1 + X^{para} + X^{int}). \quad (10)$$

With the outputs of part II, we are able to proceed to part III and compute the final probability distributions.

3.2 Vertical hierarchical Transformer

The key difference between the parallel and the vertical architectures is the latter only passes context-aware word embeddings from the encoder to decoder part II without additional paragraph embeddings. Instead, the cross-document relationships in this architecture are modeled based on word-level context vectors by stacking the paragraph-level multi-head attention vertically on top of the word-level multi-head attention.

Vertical paragraph-level multi-head attention: Since the word-level context vectors $X_t^{word} \in \mathbb{R}^{m \times d}$ are the weighted summation of token embeddings in the paragraph at the t_{th} time step, the VHT decoder regards them as *dynamic* paragraph embeddings, opposite to the *static* paragraph embeddings in PHT. According to Figure 3, the dynamic paragraph embedding serves as the key and value of the vertical paragraph-level multi-head attention after adding the ranking embeddings, and the query remains as the output of part I after separating in the time dimension, i.e., $X_t^1 \in \mathbb{R}^{1 \times d}$.

$$X_t^{word} := X_t^{word} + R, \quad (11)$$

$$X_t^{para} = MultiHead(X_t^1, X_t^{word}, X_t^{word}), \quad (12)$$

where $X_t^{para} \in \mathbb{R}^{1 \times d}$ are concatenated to $X^{para} \in \mathbb{R}^{k \times d}$ along time steps before passed to decoder part III with X^1:

$$X^2 = LayerNorm(X^1 + X^{para}). \quad (13)$$

4 Experimental setup

4.1 WikiSum dataset

Data sparsity has been the bottleneck of Neural MDS models til the WikiSum dataset (Liu et al., 2018b) came along. In this study, we use the ranked version of WikiSum provided in Liu and Lapata (2019), in which each sample contains a short title, 40 ranked paragraphs with a maximum length of 100 tokens as source inputs, and a target summary with an average length of 140 tokens. Consistent with Liu and Lapata (2019), the dataset is split with 1,579,360 samples for training, 38,144 for validation and 38,205 for test. Subword tokenization (Bojanowski et al., 2017) is adopted to tokenize

our vocabulary to 32,000 subwords to better solve unseen words.

4.2 Training configuration

We apply a dropout rate of 0.3 to the output of each sub-layer and a warm-up Adam optimizer (Vaswani et al., 2017) with 16,000 warm-up steps. Given the limited computing resources (one 2080Ti), we stack 3-layers of encoder-decoder in both of our hierarchical Transformers with 256 hidden units, 1024 units in the feed-forward network and 4 headers. To demonstrate that our model has the potential to stack, 1-layer models are trained for comparison. All parameters are randomly initialized including token embeddings.

All multi-layer models are trained for approximately 600,000 steps, while single-layer models for approximately 300,000 steps. Checkpoints are saved per 20,000 steps and the best-performing checkpoint on the validation set is used to generate the final summary.

During the inference, the beam size is set as 5 and the average length normalization is used. The beam search is terminated til the length exceeds 200. In addition, we disallow repetition of trigrams and block two tokens (except the comma) before the current step to prevent degeneration situations such as *Mike is good at cooking and cooking*.

4.3 Baselines

We compare the proposed hierarchical Transformers with the following baselines of different modeling natures.

Extractive model

Lead is an extractive model that extracts the top K tokens from the concatenated sequence, given that K is the length of the corresponding gold summary. We combine paragraphs in order and place the title at the beginning of the concatenated sequence.

Abstractive model with flat structure

Flat Transformer (FT) is the vanilla Transformer encoder-decoder model (Vaswani et al., 2017). In this study, We adopt a 3-layers Transformer and truncate the flat sequence to 1600 tokens.

T-DMCA (Liu et al., 2018b) is a Transformer-decoder model that splits a concatenated sequence into segments, and uses a Memory Compressed Attention to exchange information among them. We construct this model with 3 layers and 256 hidden states. The top 3000 tokens are truncated as inputs.

Transformer-XL (Dai et al., 2019) is a language model that excels in handling excessively long sequences. This model improves the vanilla Transformer-decoder with the recurrent mechanism and relative positional encoding. We use 512 memory length and disable the adaptive softmax, with other hyper-parameters and token length remained the same as T-DMCA.

Abstractive model with hierarchical structure

Liu's Hierarchical Transformer (Liu's HT) (Liu and Lapata, 2019) uses a hierarchical structure to enrich tokens with information from other paragraphs before inputting to the flat Transformer. We use 3 local-attention layers and 3 global-attention layers introduced in Liu and Lapata (2019). Since this model is essentially based on the flat Transformer where token length should not exceed 2000, concatenated sequences are truncated to 1600 tokens.

Parallel & Vertical Hierarchical Transformer (PHT/VHT) are models proposed in this paper. To verify that the models could be improved with deeper architectures, we train two 1-layer models to compare with the 3-layer models. We extract the top 30 paragraphs with 100 tokens per paragraph as inputs, and concatenate the title before the first paragraph.

5 Results

5.1 The ability of capturing cross-document relationships

Cross-document relationships could be reflected by paragraph attentions. That is to say, if a model assigns higher attention weights to more important paragraphs and vice versa, the model is believed to have greater capacity of capturing cross-document relationships. To analytically assess the models' performance in this aspect, we use paragraph attentions of written summaries as the gold attention distribution, and its cosine similarity to the attention distribution of generated summaries as the evaluation metric. To model the paragraph attention of gold summaries, the normalized tf-idf similarities between the gold summary and each input paragraph are computed as the gold attention distribution. For non-hierarchical models, the summation of token weights in each paragraph are computed to indicate each paragraph's attention, whilst the hierarchical model returns the paragraph

attention distribution directly from its paragraph-level multi-head attention.

Table 1: Average cosine similarities between attention distributions of generated summaries and the gold attention distribution

Model	Cosine similarity
Flat Transformer	0.8143
T-DMCA	**0.8654**
Transformer-XL	0.8447
Liu's HT	0.8769
Vertical HT	**0.9142**
Parallel HT	0.8936

It is proved by Table 1 that hierarchical structures place significant improvements on the flat models in learning cross-document dependencies by assigning paragraph attentions in a way that is closer to the gold summaries. Moreover, VHT generates summaries of the greatest similarity 91.42% with the gold summaries, most likely due to its dynamic paragraph embedding architecture which allows more accurate representation of information that is continuously updated according to the changes of input targets.

5.2 ROUGE evaluation

In this section, we adopt a widely-used evaluation metrics ROUGE (Lin, 2004) to evaluate the MDS models. ROUGE-1 & -2 and ROUGE-L F_1 scores are reported in Table 2 assessing the informativeness and fluency of the summaries, respectively.

Table 2: Average ROUGE F_1 scores. The second and the third panels are models of the flat and hierarchical structures, respectively.

Model	R-1	R-2	R-L
Lead	36.40	16.66	32.95
FT	40.30	18.67	32.84
T-DMCA	41.09	19.78	33.31
Transformer-XL	41.11	19.81	33.72
Liu's HT	40.83	19.41	33.26
1-layer PHT	41.02	19.82	33.28
1-layer VHT	41.04	19.50	33.64
PHT	**41.99**	**20.44**	34.50
VHT	41.85	20.21	**34.61**

As shown in Table 2, the extractive model Lead exhibits overall inferior performance in comparison to the abstractive models, except that it produces a 0.11-higher ROUGE-L than the Flat Transformer. Although Liu's HT improves FT with a hierarchical structure, it fails to outperform the two extended flat models, i.e. T-DMCA and Transformer-XL, that are developed to learn with longer input of tokens. Moreover, T-DMCA and Transformer-XL, the two flat models based on the Transformer decoder, report comparable results in terms of the informativeness (ROUGE-1 & -2), whilst the latter outperforms the former by 0.41 in terms of the fluency (ROUGE-L).

Further, the proposed hierarchical Transformers show promising ROUGE results. Profited from the pure hierarchical structure that enlarges the input length of tokens, PHT & VHT outperform Liu's HT in all domains of the ROUGE test. Moreover, the models' potential to be deepened is suggested by enhanced results of the 3-layer architecture over the 1-layer architecture. The ultimate 3-layer PHT & VHT surpass T-DMCA and Transformer-XL, the two flat models that also handle long input sequences of 3,000 tokens. Between the parallel and vertical architectures, PHT appears to be more informative in its summaries as it produces the highest ROUGE-1 & -2 among all models, whilst VHT is more fluent with the highest ROUGE-L.

5.3 Human evaluation

To provide a better comparison between the hierarchical and the flat structures, we select 4 representative models with the best ROUGE performances, namely T-DMCA & Transformer-XL (flat structure), and PHT & VHT (hierarchical structure). The human evaluation is divided into two parts.

The first part is to score multi-document summaries from four perspectives, including (A) **Informativeness** (Does the summary include all important facts in the gold summary), (B) **Fluency** (Is the summary fluent and grammatically-correct), (C) **Conciseness** (Does the summary avoid repetition and redundancy), (D) **Factual consistency** (Does the summary avoid common sense mistakes such as wrong date, wrong location, or anything else against facts). We specify five levels ranging from *Very poor* (1) to *Very good* (5) to assess criteria (A)-(C), and three levels of *Much better* (2), *Better* (1), and *Hard to score* (0) to assess criteria (D). Twenty examples are randomly selected from generated summaries. As shown in Table 3, both Parallel and Vertical Hierarchical Transformer

Table 3: Human evaluation results

Model	Informativeness	Fluency	Conciseness	Factual consistency	Preference
T-DMCA	3.69	3.66	**3.82**	3.04	1
Transformer-XL	3.57	3.71	3.77	2.88	
PHT	4.11	**3.97**	3.81	3.28	2.92
VHT	**4.24**	3.87	3.81	**3.36**	

Table 4: Computational efficiency (Transformer-decoder is used to show that abandoning the encoder removes approximately one quarter of parameters from the encoder-decoder model.).

Model	Max Batch Size	Parameters (MB)	Validation Speed (s)
Flat Transformer	11	165.0	634
Transformer-decoder	-	**127.1**	-
T-DMCA	10	131.1	656
Transformer-XL	8	130.4	**489**
Liu's HT	11	190.8	639
Vertical HT	13	174.5	930
Parallel HT	**17**	182.4	648

bring significant improvements over T-DMCA and Transformer-XL in terms of informativeness, fluency and factual consistency, with the former being more fluent and the latter being more informative and fact-consistent[6]. In terms of conciseness, T-DMCA outperforms with a minor advantage in comparison to the other three models. In comparison to changing model architectures, it is believed that enlarging training data and using regularization rules in the inference are more effective in preventing repetitive generations.

The second part of human evaluation is a side-by-side preference test, which is comprised of thirty control groups of two sides. In each control group, Side A randomly places a summary generated by a flat model and side B places the corresponding summary generated by a hierarchical model. Assessors select their preferred side and briefly explain their reasons. Preference results show that the hierarchical class is approximately three times more likely to be chosen than the flat class, due to their overall accuracy and informativeness according to the assessors' comments.

[6]It is interesting to note that the human evaluation suggests opposite results to the ROUGE test in terms of PHT&VHT's informativeness and fluency. The authors choose to place more trust on the quantitative measure, i.e. ROUGE, as it represents the quality of the entire sample rather than a limited segment of it.

5.4 Computational efficiency

We assess the computational efficiency of the abstractive models in three aspects, namely the memory usage, parameter size and validation speed. We uniformly hire the 3-layers architecture and 1600 input tokens. In the experimental process, we increase the batch size until out of memory in a 2080ti GPU, and the model with the maximum batch size occupies the lowest memory space. To measure the parameter size, we count the number of parameters in the neural network. Finally, we run each trained model in the validation set (38,144 samples), and the average time consumed in each checkpoint is used to evaluate the efficiency of forward-propagating in the model.

According to Table 4, the hierarchical structure (the second panel) appears to be overall more memory-saving than the flat structure (the first panel), with higher requirements on the parameters. On the other hand, models based on the Transformer-decoder, i.e. Transformer-decoder, T-DMCA and Transformer-XL, demonstrate absolute superiority in reducing the parameter size. For the speed of forward-propagating, Transformer-XL dominates due to its recurrent mechanism, whereas VHT performs the worst in this aspect indicating the model's slow inference speed. Between the two proposed models, PHT is proven to outperform VHT in both the memory usage and inference

speed, due to its parallel, rather than sequential, computation of the word & paragraph-level attention mechanisms.

5.5 Conclusion

This paper proposes two pure hierarchical Transformers for MDS, namely the Parallel & Vertical Hierarchical Transformers (PHT & VHT). We experimentally confirm that hierarchical structure improves the quality of generated summaries over flat structure by better capturing cross-document relationships, at the same time saves more memory space. Given the similar performance of the two proposed models, we recommend PHT over VHT due to its practical value of higher inference speed and memory-saving capacity.

Acknowledgement

This work is supported by the XJTLU Key Programme Special Fund - Applied Technology (No. KSF-A-14).

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2017. Improving multi-document summarization via text classification. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, Melbourne, Australia. Association for Computational Linguistics.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 335–336, New York, NY, USA. ACM.

Eric Chu and Peter Liu. 2019. MeanSum: A neural model for unsupervised multi-document abstractive summarization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1223–1232, Long Beach, California, USA. PMLR.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, Florence, Italy. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Logan Lebanoff, Kaiqiang Song, and Fei Liu. 2018. 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, Brussels, Belgium. Association for Computational Linguistics.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *ACL*.

Piji Li, Zihao Wang, Wai Lam, Zhaochun Ren, and Lidong Bing. 2017. Salience estimation via variational auto-encoders for multi-document summarization. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving neural abstractive document summarization with structural regularization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4078–4087, Brussels, Belgium. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in*

Natural Language Processing, pages 899–907, Lisbon, Portugal. Association for Computational Linguistics.

Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. 2018a. Generative adversarial network for abstractive text summarization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, pages 8109–8110. AAAI Press.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018b. Generating wikipedia by summarizing long sequences. *CoRR*, abs/1801.10198.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Shulei Ma, Zhi-Hong Deng, and Yunlun Yang. 2016. An unsupervised multi-document summarization framework based on neural document model. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1514–1523, Osaka, Japan. The COLING 2016 Organizing Committee.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2016. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462, Vancouver, Canada. Association for Computational Linguistics.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy. Association for Computational Linguistics.