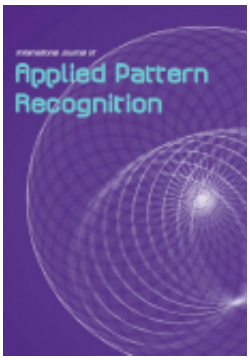


International Journal of Applied Pattern Recognition

ISSN online: 2049-8888 - ISSN print: 2049-887X
https://www.inderscience.com/ijapr

Efficient post-training pruning of dialogue summarisation based on sensitivity evaluation

Yupeng Liu, Yuhao Zhang, He Sun, Xiaochen Zhang

DOI: 10.1504/IJAPR.2026.10079730

Article History:

Received:	22 November 2025
Last revised:	23 March 2026
Accepted:	16 April 2026
Published online:	14 July 2026

Efficient post-training pruning of dialogue summarisation based on sensitivity evaluation

Yupeng Liu, Yuhao Zhang and He Sun

Harbin University of Science and Technology,
No. 52 Xuefu Road, Nangang District,
Harbin 150080, Heilongjiang Province, China
Email: flyeaglelyp@hrbust.edu.cn
Email: 1472490772@qq.com
Email: 1279224279@qq.com

Xiaochen Zhang*

Heilongjiang Institute of Technology,
No. 999 Hongqi Street, Daowai District,
Harbin 150050, Heilongjiang Province, China
Email: zxc161616@126.com
*Corresponding author

Abstract: We propose an efficient post-training structured pruning framework for dialogue summarisation that directly optimises under hardware FLOPs constraints, eliminating the need for full-model retraining. Sensitivity of attention heads and FFN filters is estimated via the Hessian matrix trace approximated by the Hutchinson algorithm, guiding a FLOPs-constrained mask search followed by intra-layer greedy rearrangement. A layer-wise Huber regression step then fine-tunes binary masks to real values to restore output fidelity. Experiments on five benchmark datasets show that the framework reduces FLOPs by 30%–40% with less than 1% degradation across BLEU, ROUGE, METEOR, and PARENT metrics, achieving up to 1.52× inference acceleration.

Keywords: sensitivity evaluation; post-training pruning; FLOPs-constrained pruning; dialogue summarisation.

Reference to this paper should be made as follows: Liu, Y., Zhang, Y., Sun, H. and Zhang, X. (2026) 'Efficient post-training pruning of dialogue summarisation based on sensitivity evaluation', *Int. J. Applied Pattern Recognition*, Vol. 8, No. 3, pp.1–18.

Biographical notes: Yupeng Liu is a Professor and Postdoctoral Fellow at Harbin University of Science and Technology. His research interests include natural language processing, machine translation, intelligent healthcare systems, and quantum computing. He serves as a reviewer for multiple SCI journals, has won top awards in international machine translation competitions, and has published over 30 academic papers.

Yuhao Zhang is a graduate student at Harbin University of Science and Technology. His research focuses on deep learning, structured pruning of Transformer models, and efficient natural language processing. He has participated in multiple research projects on AI model optimisation, with rich practical experience in model compression and deployment.

He Sun is a graduate student at Harbin University of Science and Technology, majoring in software engineering. His research interests cover graph neural networks, model optimisation, and applied pattern recognition. He has participated in several research projects on efficient deep learning, with solid experience in model implementation.

Xiaochen Zhang is an Associate Professor with a PhD degree at Heilongjiang Institute of Technology. His research interests include pattern recognition, natural language processing, and efficient deep learning. He has rich experience in academic research and project development, and has published high-quality papers in related international journals.

1 Introduction

Dialogue summarisation models extract key information from conversational texts, and have been widely applied in online customer service, intelligent dialogue systems, news summarisation and other fields to facilitate users' quick comprehension of conversational core content. Current dialogue summarisation models are mostly built on transformer architectures, such as BERT (Devlin et al., 2019) and GPT (Radford and Narasimhan, 2018), which enables them to generate accurate and concise summaries with promising performance by integrating large-scale pre-training techniques for better understanding of conversational content. However, these models suffer from high storage demands and significant inference latency, making it a persistent challenge to deploy them efficiently according to the computing power constraints of hardware devices Model pruning has emerged as an effective solution to this issue, as it can drastically reduce model inference time (Hou et al., 2020; Michel et al., 2019; Lin et al., 2020). Nevertheless, existing pruning methods face practical limitations that hinder their real-world application:

1 Pruned models require full retraining or the pruning configuration has to be learned during the training phase, which increases training time by dozens of times and incurs enormous computational overhead (Lin et al., 2020).

2 A large number of dynamic components need to be introduced during model deployment, which complicates the pruning process and necessitates additional hyperparameter tuning (Mao et al., 2020).

3 Such methods cannot be directly adapted to hardware computing power constraints, as they either rely on regularisation to control model sparsity or adopt fixed model architectures irrelevant to user-specific settings (Liu et al., 2021).

To address the above problems, this paper proposes an adaptive structured pruning method for dialogue summarisation models, which takes attention heads and filters as the minimal pruning units. The method takes a pre-trained dialogue summarisation model, sample data and hardware FLOPs constraints as inputs, and outputs a pruned model that is ready for fast deployment. The key design principle is to decouple pruning from retraining: sensitivity is assessed on the converged model using a computationally tractable Hessian approximation, the pruning configuration is then optimised to respect hardware FLOPs budgets while accounting for intra-layer functional dependencies, and performance is recovered through a lightweight layer-wise regression step rather than

global backpropagation. This design enables the entire process to complete in under 0.1 hours, compared to the 15–93 hours required by retraining-based methods.

The innovations are:

1 a hardware-aware pruning search that treats FLOPs as a hard constraint, enabling direct adaptation to heterogeneous deployment targets

2 an intra-layer mask rearrangement strategy based on Hessian trace approximation that captures functional redundancy within each transformer layer

3 a layer-wise Huber regression fine-tuning step that restores model accuracy without global retraining, reducing recovery time from hours to minutes.

2 Related work

Pruning methods are categorised by paradigm into unstructured pruning and structured pruning; unstructured pruning targets individual neurons, while structured pruning prunes hierarchical model structures to enable actual inference acceleration with only minor performance impact. By evaluation criterion, pruning falls into amplitude-based pruning; first-order Taylor approximation pruning and second-order Taylor approximation pruning, which respectively rely on weight absolute values, gradient guidance and Hessian matrix guidance for weight evaluation.

Amplitude-based pruning (Gale et al., 2019) cuts down model storage yet induces severe accuracy loss. First-order Taylor expansion (Sanh et al., 2020) and Hessian matrix-based weight evaluation (Kurtic et al., 2022) serve as more refined pruning criteria in comparison. In the field of model compression pipelines, Han et al. (2016) proposed a three-stage deep compression method combining pruning, quantisation and Huffman coding, which delivers a high compression ratio but demands extensive manual debugging. The lottery hypothesis (Frankle and Carbin, 2019; Chen et al., 2020, 2021; Prasanna et al., 2020) posits that dense models contain high-performance sub-networks, and subsequent relevant studies focus on automatically identifying such sub-networks during the training phase to cut down training costs.

For transformer-specific pruning, LadaBERT (Mao et al., 2020) integrates pruning, matrix factorisation and knowledge distillation to realise iterative model compression; DynaBERT (Hou et al., 2020) adapts model size by adjusting width and depth adaptively; Voita et al. (2019) put forward a multi-head attention (MHA) pruning method based on differentiable relaxation. In convolutional neural networks, pruning techniques have achieved 90% parameter reduction and 3–5 times FLOPs reduction on AlexNet and VGG-16, with almost no performance loss (Han et al., 2015). However, mainstream transformer pruning methods necessitate retraining or the learning of pruning configurations during training, leading to up to ten times increase in training time and massive computational overhead (Lagunas et al., 2021; Xia et al., 2022).

Existing pruning research can reduce model storage requirements and boost inference speed to a certain degree, but all of them call for extensive manual hyperparameter debugging and neglect hardware computing power constraints, resulting in poor hardware adaptability. In contrast, this paper centres on hardware FLOPs limitations, conducts model pruning according to the requirements of different hardware devices, and avoids

full-model retraining and manual hyperparameter adjustment, thus obtaining pruned models with low time complexity.

3 Efficient pruning method based on the post-trained model

3.1 Preliminaries and notation

For clarity, we briefly define key technical terms used throughout this paper. Floating-point operations (FLOPs) measure the computational cost of a forward pass and serve as our hardware budget proxy. The Hessian matrix **H** contains second-order partial derivatives of the loss with respect to model parameters, capturing local curvature; its trace tr(**H**) aggregates diagonal curvature information efficiently. The Hutchinson algorithm estimates the matrix trace stochastically without forming **H** explicitly, by computing $\mathbb{E}[z^T \mathbf{H} z]$ for random vectors z. The Hadamard product ($\circ$) denotes element-wise multiplication between two vectors or matrices. Huber regression minimises a loss function that is quadratic for small residuals and linear for large ones, providing robustness to outliers compared to standard least-squares regression.

3.2 Overall framework

For completeness, we briefly introduce only the transformer components needed for our pruning formulation. Our dialogue summarisation model is based on a standard transformer architecture, in which each block contains a MHA module followed by a feed-forward network (FFN). In our method, an attention head in MHA and an FFN filter are treated as the basic structural pruning units.

Each MHA contains multiple independent heads:

$$MHA(x) = \sum_{i=1}^{H} Att_i(x) \tag{1}$$

where the Att_i is the i^{th} dot-product head and x is the input sequence. Each FFN consists of multiple filters:

$$FFN(x) = \sum_{i=1}^{N} \left(W_{:,i}^{(2)} \sigma \left(W_{i,:}^{(1)} x + b_i^{(1)} \right) \right) + b^{(2)} \tag{2}$$

where $W^{(1)} \in \mathbb{R}^{\mathrm{d}_{ff} \times \mathrm{d}_{\text{model}}}$, $W^{(2)} \in \mathbb{R}^{\mathrm{d}_{\text{model}} \times \mathrm{d}_{ff}}$ are weight matrices, $b^{(1)}$ and $b^{(2)}$ are bias terms, and σ denotes the activation function.

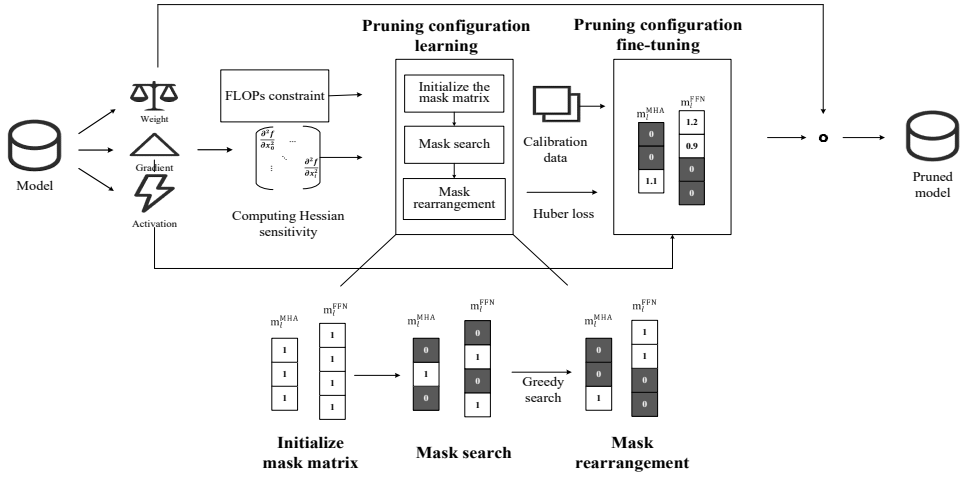
To formulate pruning, we introduce binary mask variables for the attention heads and FFN filters in layer l:

$$MHA\left(\mathrm{x}; m_l^{MHA}\right) = \sum\nolimits_{i=1}^{H} m_{l,i}^{MHA} \circ Att_i(x) \tag{3}$$

$$FFN\left(\mathrm{x}; m_l^{FFN}\right) = \left(\sum\nolimits_{i=1}^{N} m_{l,i}^{FFN} \circ W_{:,i}^{(2)} \sigma \left(W_{i,:}^{(1)} x + b_i^{(1)} \right) \right) + b^{(2)} \tag{4}$$

Here, $m_l^{MHA} = [m_{l,1}^{MHA}, m_{l,2}^{MHA}, ..., m_{l,H}^{MHA}]^T \in \{0,1\}^H$ and $m_l^{FFN} = [m_{l,1}^{FFN}, m_{l,2}^{FFN}, ..., m_{l,H}^{FFN}]^T \in \{0,1\}^N$ denote the binary mask vectors for the attention heads and *FFN* filters in the l^{th} layer, respectively. The symbol ◦ denotes the Hadamard product (element-wise multiplication). Setting $m_{l.i}^{MHA}$ or $m_{l.i}^{FFN}$ to 0 is equivalent to pruning the i^{th} attention head or filter. In a standard transformer with L blocks, there are a total of $L(H + N)$ mask variables.

Figure 1 Pruning process



The pruning process is shown in Figure 1. The framework inputs are: the dialogue summarisation model; calibration dataset; FLOPs constraint of hardware device; the model has converged on the training dataset. The pruning process is divided into the two stages:

1 A preliminary search of the mask matrix is performed using approximate Hessian matrices and FLOPs constraint, and then the intra-layer masks are rearranged through greedy search to restore the model configuration.

2 Use Huber regression to fine-tune the mask matrix layer-by-layer to ensure that the output of each layer is restored.

3.3 The learning of the pruning configurations

1 Initialisation of the mask matrix: model pruning can be quantified by the error caused by the pruning weights. To this end, we formulate dialogue summarisation pruning as the following optimisation problem:

$$\delta\mathcal{L} = \mathcal{L}(x; \boldsymbol{m}) - \mathcal{L}(x; \mathbf{1}) \tag{5}$$

where $\delta\mathcal{L}$ denotes the error caused by the pruning, x denotes the model input, $\boldsymbol{m} \in \{0, 1\}^{L(H+N)}$ is the unified mask vector over all transformer blocks (as defined in Section 3.1, concatenating $m^{MHA} \in \mathbb{R}^{LH}$ and $m^{FFN} \in \mathbb{R}^{LN}$), and $\mathbf{1}$ denotes the

all-ones vector corresponding to the unpruned state. By applying Taylor expansion to equation (5) at the point $\boldsymbol{m} = \mathbf{1}$, we obtain:

$$\mathcal{L}(m) \approx \mathcal{L}(\mathbf{1}) + \mathrm{g}^T(\boldsymbol{m} - \mathbf{1}) + \frac{1}{2}(\boldsymbol{m} - \mathbf{1})^T \mathbf{H}(\boldsymbol{m} - \mathbf{1}) \tag{6}$$

where $\mathrm{g} = \mathbb{E}\left[\left.\frac{\partial \mathcal{L}}{\partial m}\right|_{\boldsymbol{m}=1}\right]$ and $\mathbf{H} = \mathbb{E}\left[\left.\frac{\partial^2 \mathcal{L}}{\partial m^2}\right|_{\boldsymbol{m}=1}\right]$. Since the pruned model is already trained and converged, we can assume that the gradient is close to 0, i.e., $\mathrm{g}^T(\mathbf{m} - \mathbf{1}) \approx 0$. Furthermore, since $\mathcal{L}(\mathbf{1})$ is constant, we can approximate the error induced by changing the mask as:

$$\delta\mathcal{L} \approx \sum_i g_i \delta m_i + \frac{1}{2}\sum_i h_{ii}\delta m_i^2 + \frac{1}{2}\sum_{i \neq j} h_{ij}\delta m_i \delta m_j + O\left(\delta_m{}^3\right) \tag{7}$$

where $g_i = \frac{\partial \mathcal{L}}{\partial m_i}$, $h_{ij} = \frac{\partial^2 \mathcal{L}}{\partial m_i \partial m_j}$ and $O(\delta_m{}^3)$ denotes higher-order error terms. Here, $\delta_{mi} = (m_i - 1)$ represents the deviation of the mask from its initial value. Assuming no correlation between the mask variables, the errors induced by pruning each unit are considered independent. Consequently, the sensitivity of the i^{th} unit (where m_i is set to 0) is:

$$\delta\mathcal{L}_i \approx \frac{1}{2}h_{ii}(0-1)^2 = \frac{1}{2}h_{ii} \tag{8}$$

Here, the subscript i indexes a single structural unit (either an attention head $m_{l,i}^{MHA}$ or an *FNN* filter $m_{l,i}^{FNN}$) within the flattened mask vector $\boldsymbol{m}$, and h_{ii} is the corresponding diagonal element of the Hessian $\mathbf{H}$.

According to equation (8), the loss error is mainly determined by the diagonal elements of the Hessian matrix $\mathbf{H}$ with respect to the masks. However, computing $\mathbf{H}$ is computationally costly due to second-order parameter derivatives. For large models, it may exceed resource limits and suffer from numerical instability caused by strong nonlinearity or saddle points. Furthermore, the memory footprint of $\mathbf{H}$ grows linearly with the number of parameters.

In this paper, we use the matrix trace to approximate $\mathbf{H}$ for evaluating the sensitivity of attention heads and filters, and employ the Hutchinson algorithm for efficient trace estimation. For the i^{th} attention head or filter, the Hessian-based sensitivity is defined as:

$$\mathcal{E}_i^{\mathbf{H}} = \mathbb{E}\left[tr\left(\frac{\partial^2 \mathcal{L}(x, \boldsymbol{m})}{\partial \mathrm{m}_i^2}\right)\right] \tag{9}$$

where tr is the trace operator, $\mathcal{L}$ is the model loss function, m is the set of all considered mask variables, and x is the calibration data. A higher value of $\mathcal{E}_i^{\mathbf{H}}$ indicates an increase in the local curvature of the loss function, implying that the unit is more sensitive.

2 Mask search: search the mask matrix based on sensitivity evaluation. To incorporate hardware efficiency, we define F_{head} and F_{filter} as the number of FLOPs required by a single attention head and a single filter, respectively. These constants allow the search process to strictly adhere to the total FLOPs constraint $\boldsymbol{C}$ by calculating the trade-off between pruning different types of structural units. The detailed procedure is outlined in Table 1.

Table 1 Pseudocode for mask search under FLOPs constraint

Input: FLOPs constraint C, mask matrix m^{MHA} and m^{FFN}

1: for k_1 from 0 to LH:
2: HI = indices of k_1 least important heads
3: $k_2 = LN - \lfloor (\text{C} - k_1 F_{\text{head}})/F_{\text{filter}} \rfloor$
4: FI = indices of k_2 least important filters
5: $R[n]$ = (HI, FI)
6: **end**
7: HI^*, $\text{FI}^* = \text{R}[n^*]$
8: m^{MHA} $[\text{HI}^*] = 0$
9: m^{FFN} $[\text{FI}^*] = 0$

Output: $\text{m}^* = (\text{m}^{\text{MHA}}, \text{m}^{\text{FFN}})$

In the pseudocode, k_1 denotes the number of the pruned attention heads, k_2 denotes the number of the pruned filters, HI stands for the indices of the pruned attention heads, and FI denotes the indices of the pruned filters. In line 3, the number of the pruned filter is determined based on the formula $\lfloor (C - nF_{\text{head}})/F_{\text{filter}} \rfloor$, where F_{head} is FLOPs per head and F_{filter} is FLOPs per filter. R[n] records the current remaining indices of the heads and filters. Line 7 obtains the attention heads and filter indices that need to be pruned. $m^{MHA}[\text{HI}^*]$ and $m^{FFN}[\text{FI}^*]$ meaning pruning the heads and filters, respectively. After the mask search, the optimal mask m^* is obtained.

3 Mask rearrangement: although the mask search identifies a pruning solution with minimum sensitivity, it might not be optimal as it overlooks functional interactions between attention heads and filters within the same layer. Since there may be two attention heads (filters) serving a similar function within a layer, pruning only one of them may not affect the model performance. However, when both are pruned, the model accuracy may significantly decrease.

To address this, we initialise the optimisation using prior-stage results. Firstly, we set the number of pruned attention heads and filters per layer identical to the previous stage. Given the mask matrix m^* obtained from the search stage, for the l^{th} layer, we constrain $\|m_l\|_0 = \|m_l^*\|_0$. Then, using the mask m as the starting point, we record the output results for each layer and employ a greedy search strategy to solve this problem.

Specifically:

1 Assuming no interaction between the mask variables across the different layers.

2 Assuming the predetermined number of the pruned attention heads (filters) in each layer. Furthermore, equation (8) is further expressed on each layer as:

$$\mathrm{m}_l = \arg\min_{\mathrm{m}_l} (1-\mathrm{m}_l)^{\mathrm{T}} H_l (1-\mathrm{m}_l) \tag{10}$$

where H_l represents the layer-wise Hessian matrix corresponding to the mask variables in the l^{th} layer.

3.4 Fine-tuning of pruning configuration

In this section, we will further optimise the mask matrix by adjusting the values with 1 to any real values in $\hat{m}$. To mitigate the computational overhead of standard fine-tuning (Xia et al., 2022), we employ Huber regression for efficient layer-wise mask optimisation. Taking *MHA* layer as an example, the problem of minimising the fine-tuning error is represented as:

$$\arg\min_{m_l^{MHA}} \left\| \left(x + MHA\left(x; m_l^{MHA}\right) \right) - \left(x' + MHA\left(x'; \mathbf{1}\right) \right) \right\|_2^2 \tag{11}$$

$$s.t.\ Z\left(m_l^{MHA}\right) = Z\left(\hat{m}_l^{MHA}\right) \tag{12}$$

where x, x' are input hidden states, $\mathbf{1}$ is the unpruned state vector, and $Z(m) = \{i|m_i = 0\}$ defines the zero-index set to preserve the sparsity pattern during fine-tuning. Then, the optimisation problem is rewritten as:

$$\arg\min_{m_l^{MHA}} Huber\_loss\left(\left|\mathrm{A}m_l - \mathrm{b}\right|\right) \tag{13}$$

$$\begin{aligned} A &:= \left[m_{l,1}^{MHA}\mathrm{Att}_1(x), ..., m_{l,\mathrm{H}}^{MHA}\mathrm{Att}_H(x) \right] \\ b &:= \left(x' + \sum_{h=1}^{H} \mathrm{Att}_h(x') \right) - x \end{aligned} \tag{14}$$

where $A \in \mathbb{R}^{T\times H}$ is the stacked output of H attention heads over T tokens, and $m_l \in \mathbb{R}^H$ is the real-valued mask vector being optimised, matrix A denotes the output activation of the attention heads of the pruned model with binary masks, and vector b is the difference between the output activations of the two models.

4 Experiment

4.1 Experiment setup

1 Environment: PyTorch 1.13.1, CUDA 11.7, Python 3.8.10, NVIDIA RTX 3080 (10GB), Intel Core i5-13600KF.

2 Key hyperparameters: calibration set: 128 samples; Hutchinson sampling: 100 steps; Huber regression: learning rate 5e-4, delta 1.0, 50 optimisation steps; inference: batch size 256, max length 512, beam width 4.

4.2 Experiment setting

The experiment setting involved two models as shown in Table 2, utilising the datasets specified in Table 3, and the baselines shown in Table 4. Performance evaluation of the pruned models was conducted using BLEU, ROUGE, METEOR, and PARENT metric for evaluating factual consistency.

Table 2 Model description

Model	*Description*
LBART (Lewis et al., 2020)	A pre-trained model based on BART (Lewis et al., 2020), fine-tuned for dialogue summarisation task, with a total of 140 million parameters
DSMC (Chen and Yang, 2020)	A non-pre-trained model based on transformer, with approximately 70 million parameters

Table 3 Dataset description

Dataset	*Description*
AMI (Carletta et al., 2006)	A multimodal dataset comprising 100 hours of conference recordings. The corpus has been manually annotated, covering various tasks including orthographic transcription, named entities and dialogue acts as discourse attributes, summarisation, emotion, as well as some head and gesture actions.
QMSum (Zhong et al., 2021)	Query-based multi-domain conference summarisation, comprising 1,808 query-summary pairs from 232 conferences
SAMSum (Gliwa et al., 2019)	This dataset consists of a high-quality dialogue corpus, manually annotated, and accompanied by abstract summarisation
GupShup (Fabbri et al., 2021)	The dataset comprises over 6,800 Hindi-English dialogues along with their corresponding human annotations and Hindi-English dialog summarisation
ConvoSumm (Mehnaz et al., 2021)	The dataset collected is based on a question-opinion-argument framework, covering various forms of online dialogues such as news comments, discussion forums, community Q&A forums, and e-mail discussions.

4.3 Comparison experiment

Figure 2 shows the comparative experiment of LBART and DSMC model on SAMSum using our pruning method and baselines. In Figure 2, the percentage represents the FLOPs constraint. Since the baseline methods do not inherently support FLOPs as a direct pruning constraint, we implemented a calibration process to ensure a fair comparison. Specifically, for baselines relying on sparsity ratios or regularisation (e.g., SLIP, EBERT), we performed a hyperparameter sweep – systematically adjusting their specific pruning thresholds or penalty coefficients – until the resulting pruned models reached FLOPs levels equivalent to our target constraints (70%, 60%, and 50%) within a marginal error of ±1%. For methods with fixed architectures (e.g., Sajjad), we selected the configuration that most closely aligned with our hardware budget. This alignment ensures that performance metrics are benchmarked under near-identical computational costs across all evaluated methods. Flop 66.7%, SLIP 65.60%, Sajjad 66.70%, DynaBERT 75%, and our proposed method with 70% FLOPs (Ours 70%) all maintain nearly the same scores as the original model. Meanwhile, it can be observed that DynaBERT 50% and EBERT 50% exhibit significant performance declines.

Table 4 Baseline system

Pruning method	*Classification*	*Main idea*
SLIP (Lin et al., 2020)	Structured	Imposes penalties on entire residual modules within Transformer, directing them towards an identity mapping during the pruning process.
EBERT (Liu et al., 2021)	Structured	Dynamically determining and pruning unimportant structures in feedforward networks
Sajjad (Sajjad et al., 2023)	Structured	Applied to pre-trained models, several layer-wise pruning strategies have been proposed.
DynaBERT (Hou et al., 2020)	Structured	This method allows flexible adjustment of width/depth according to task requirements, to meet the hardware performance demands of edge devices.
BMP (Lagunas et al., 2021)	Structured	Partitioning the matrix into fixed-size blocks and exploring the weight matrix based on a block-by-block mode.
CoFi (Xia et al., 2022)	Semi-structured	This method incorporates both the coarse-grained and fine-grained modules, utilising the different granularity masks to control the pruning degree for each parameter.
Flop (Wang et al., 2020)	Unstructured	Structured pruning approach for large language models that removes less important structures to improve efficiency while preserving performance.

4.4 Training and inference time

4.4.1 Training time analysis

We selected the top-performing pruning methods, including DynaBERT (Hou et al., 2020), EBERT (Liu et al., 2021), BMP (Lagunas et al., 2021), and CoFi (Xia et al., 2022), and analysed their end-to-end retraining costs. As shown in Figure 3, these methods require 15–93 hours for retraining. In contrast, our method completes in 0.1 hours.

4.4.2 Inference time analysis

Figure 4 shows the inference time of LBART model on QMSum, SAMSum, GupShup, and ConvoSumm with FLOPs constrained to 60% (i.e., ensuring that the decrease in the multiple evaluation metrics does not exceed 1%). With a batch size of 256, the average acceleration is 1.44 times, with a maximum of up to 1.52 times.

4.4.3 Model size and storage analysis

To provide a comprehensive evaluation of the pruning efficiency, we report the model size reduction along-side computational savings. Table 5 presents the parameter count and storage requirements for both base-line and pruned models.

Figure 2 Comparative experiment on LBART and DSMC model (see online version for colours)

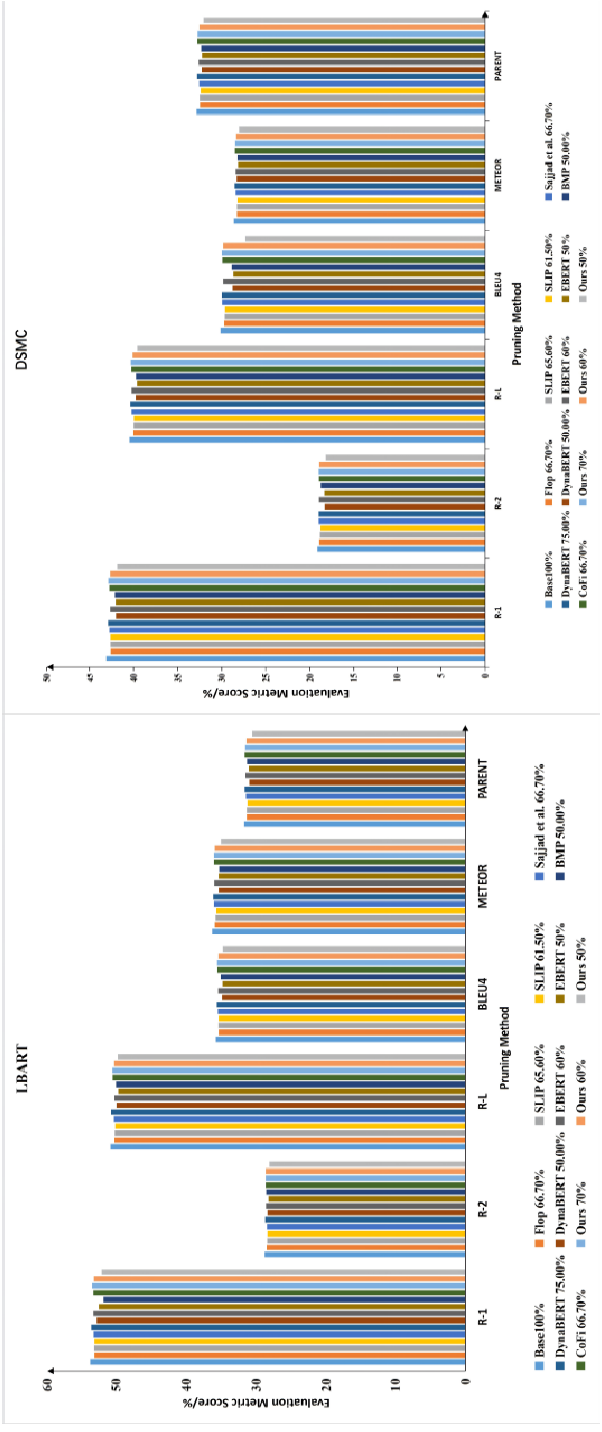


Figure 3 Retraining cost (see online version for colours)

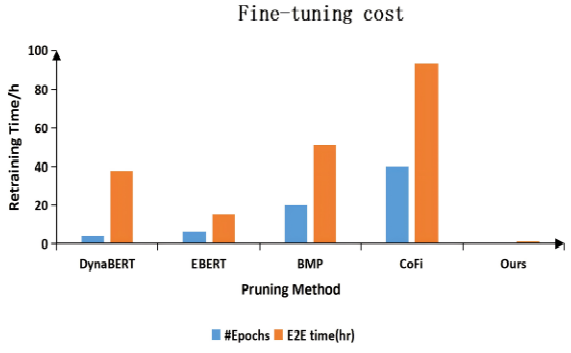


Figure 4 Inference acceleration on LBART model (see online version for colours)

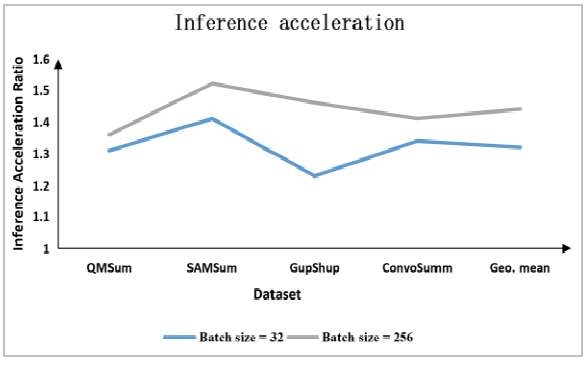


Table 5 Model size comparison before and after pruning

Model	*FLOPs constraint*	*Parameters (M)*	*Storage size (MB)*	*Reduction*
LBART (baseline)	100%	140.0	560.0	-
LBART (pruned)	70%	98.0	392.0	30.0%
LBART (pruned)	60%	84.0	336.0	40.0%
LBART (pruned)	50%	70.0	280.0	50.0%
DSMC (baseline)	100%	70.0	280.0	-
DSMC (pruned)	70%	49.0	196.0	30.0%
DSMC (pruned)	60%	42.0	168.0	40.0%
DSMC (pruned)	50%	35.0	140.0	50.0%

Notes: Storage size is calculated assuming FP32 precision (4 bytes per parameter). The reduction percentage represents the decrease in both parameter count and storage requirements compared to the baseline model.

As shown in Table 5, the pruned models achieve storage reductions proportional to the FLOPs constraints. At 60% FLOPs (the optimal operating point with < 1% performance degradation), LBART reduces from 560 MB to 336 MB (40% reduction), while DSMC reduces from 280 MB to 168 MB (40% reduction). This demonstrates that our structured pruning method effectively removes redundant parameters while

maintaining model performance, making the pruned models more suitable for deployment on resource-constrained devices.

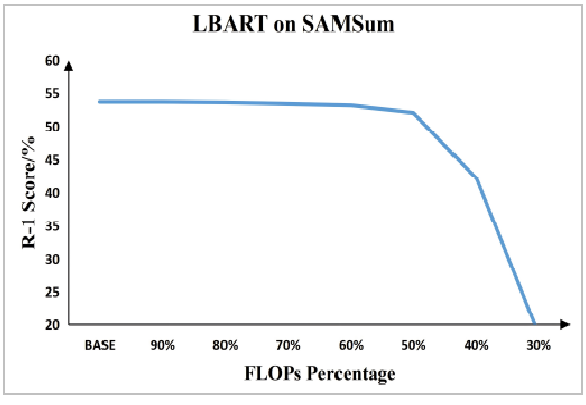
4.5 Comprehensive performance evaluation

Figure 6 shows the various evaluation metrics of DSMC and LBART models under the different FLOPs constraints on AMI, SAMSum, and QMSum.

It can be observed that both DSMC and LBART models maintain the performance of evaluation metrics R-1, R-2, R-L, BLEU4, METEOR, and PARENT within a 1% decrease when constrained to only 60% of the original model's FLOPs. Additionally, when decreasing FLOPs in increments of 10%, the evaluation metrics show a stable decreasing trend with each decrease of FLOPs. At 90% FLOPs constraint, the models exhibit almost no performance decrease. However, a significant decrease occurs when FLOPs constraint reaches 50%.

To further explore the compression lower limit of FLOPs, this study uses LBART model on SAMSum with R-1 as evaluation metric, progressively reducing FLOPs. The results are shown in Figure 5.

Figure 5 LBART model performance under different FLOPs (see online version for colours)



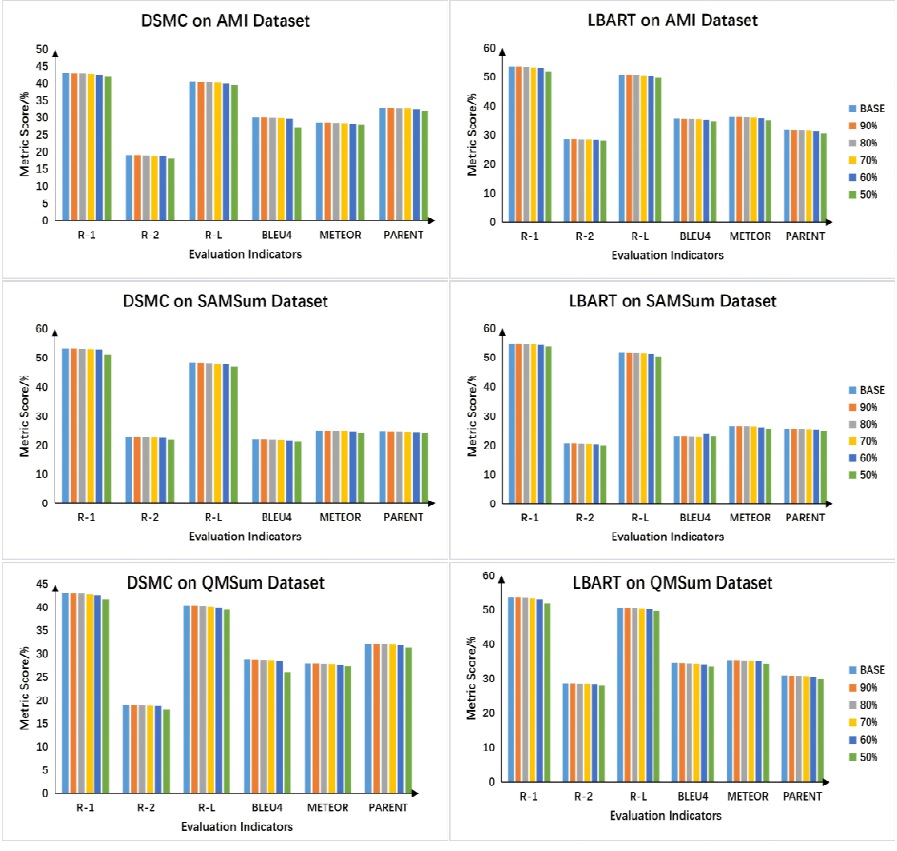
Through the experiment, it was found that after FLOPs dropped below 50%, there was a cliff-like drop in the model performance. When FLOPs constraint was set to 30%, the R-1 score could only maintain 34% of the original model performance.

4.6 Ablation experiment

To validate the effectiveness of mask rearrangement and mask fine-tuning in restoring model performance, ablation experiments were designed. Table 6 presents the results of ablating mask rearrangement and mask fine-tuning for LBART with 60% FLOPs constraint on SAMSum dataset. The experiment results show that both stages contribute to restoring model performance across all evaluation metrics. Specifically, mask re-arrangement provides an average improvement of +0.58 points, while mask fine-tuning contributes a further +1.28 points, confirming it as the most critical recovery stage. Notably, the fine-tuned model achieves performance within 1% of the

baseline across ROUGE, BLEU, METEOR, and PARENT metrics, demonstrating the effectiveness of our layer-wise Huber regression approach.

Figure 6 Performance under different FLOPs constraints on AMI, SAMSum, and QMSum datasets (see online version for colours)



4.7 Qualitative analysis

To further illustrate the impact of the proposed pruning and fine-tuning stages on the model's actual performance, we conducted a qualitative comparison using a sample from the SAMSum dataset. Table 7 presents the summaries generated by the original LBART model, the pruned model (at 60% FLOPs) without fine-tuning, and the final pruned model after layer-wise Huber regression fine-tuning.

As demonstrated in Table 7, the original model provides a fluid and factually complete summary. The pruned model without fine-tuning (Mask search + Re-ordering) retains core keywords like '3 PM' but suffers from significant readability issues and 'choppy' syntax, which aligns with the lower ROUGE scores observed in the ablation study. After applying the Huber regression fine-tuning, the model recovers its ability to generate grammatically correct and factually consistent sentences. This qualitative evidence suggests that while pruning removes structural redundancy, the fine-tuning

method is essential for restoring the linguistic nuance and logical coherence required for high-quality dialogue summarisation.

Table 6 Ablation experiment

Method	*R-1*	*R-2*	*R-L*	*BLEU-4*	*METEOR*	*PARENT*	*Avg. improvement*
Baseline	53.70	28.79	50.81	24.35	31.42	28.67	-
Mask search	51.46	26.82	48.19	22.18	29.31	26.54	-
+Rearrangement	52.04	27.41	48.71	22.76	29.89	27.12	+0.58
+Fine-tuning	53.18	28.61	50.27	24.12	31.18	28.41	+1.28

Notes: All metrics are reported as percentages. Avg. Improvement represents the mean improvement across all six metrics compared to the Mask Search baseline.

Table 7 Qualitative comparison of dialogue summaries (LBART, 60% FLOPs)

Model	*Summary output*
Original model	Amanda informs Ben that the meeting has been rescheduled to 3 PM today. Ben confirms he will be there on time.
Pruned (no fine-tuning)	Amanda says meeting is 3 PM. Ben will be.
Pruned + Fine-tuned	Amanda tells Ben the meeting is rescheduled to 3 PM today. Ben confirms he can attend.

5 Conclusions

This paper presents an adaptive structured pruning method for dialogue summarisation that enables efficient hardware deployment through direct FLOPs constraints. Unlike traditional methods requiring exhaustive retraining, our post-training framework maintains accuracy while drastically reducing computational overhead.

Experimentally, sensitivity-based mask search under explicit FLOPs constraints yields near-original performance at 60% FLOPs; intra-layer mask rearrangement provides an average +0.58 improvement in ROUGE scores; and Huber regression fine-tuning contributes a further +1.28 improvement, confirming it as the most critical recovery stage. Together, these components achieve up to 1.52× inference acceleration on real hardware while completing the entire pruning pipeline in 0.1 hours.

However, a critical 'performance cliff' emerges below 50% FLOPs, with evaluation metrics dropping sharply and nonlinearly – indicating that uniform pruning encounters a representational bottleneck at high compression ratios.

To resolve this, future research will explore layer-wise adaptive FLOPs allocation, dynamically distributing sparsity by each layer's sensitivity and semantic coherence contribution instead of global constraints. We also aim to integrate global semantic information into fine-tuning to preserve long-range dependencies, potentially extending the compression limit beyond 50%.

Declarations

All authors declare that they have no conflicts of interest.

References

Carletta, J., Ashby, S., Bourban, S. et al. (2006) 'The AMI meeting corpus: a pre-announcement', in Renals, S. and Bengio, S. (Eds.): *Machine Learning for Multi-modal Interaction: Second International Workshop, MLMI 2005*, Revised Selected Papers, Springer, Berlin, Heidelberg, LNCS 3869, pp.28–39, DOI: 10.1007/11677482_3.

Chen, J. and Yang, D. (2020) 'Multi-view sequence-to-sequence models with conversational structure for abstractive dialogue summarization', *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, Online: Association for Computational Linguistics, pp.4106–4118, DOI: 10.18653/v1/2020.emnlp-main.336.

Chen, T., Frankle, J., Chang, S. et al. (2020) 'The lottery ticket hypothesis for pre-trained BERT networks', *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, Curran Associates, Inc., Red Hook, NY, pp.15834–15846.

Chen, X., Cheng, Y., Wang, S. et al. (2021) 'EarlyBERT: efficient BERT training via early-bird lottery tickets', *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online: Association for Computational Linguistics, pp.2195–2207, DOI: 10.18653/v1/2021.acl-long.171.

Devlin, J., Chang, M.W., Lee, K. et al. (2019) 'BERT: pre-training of deep bidirectional transformers for language understanding', *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long and Short Papers)*, Association for Computational Linguistics, Minneapolis, MN, USA, pp.4171–4186, DOI: 10.18653/v1/N19-1423.

Fabbri, A.R., Rahman, F., Rizvi, I. et al. (2021) 'ConvoSumm: conversation summarization benchmark and improved abstractive summarization with argument mining', *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online: Association for Computational Linguistics, pp.6866–6880, DOI: 10.18653/v1/2021.acl-long.535.

Frankle, J. and Carbin, M. (2019) 'The lottery ticket hypothesis: finding sparse, trainable neural networks', *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, ICLR/OpenReview, New Orleans, LA, USA.

Gale, T., Elsen, E. and Hooker, S. (2019) *The State of Sparsity in Deep Neural Networks* [EB/OL] 25 February [online] https://arxiv.org/abs/1902.09574 (accessed 25 May 2026).

Gliwa, B., Mochol, I., Biesek, M. et al. (2019) 'SAMSum corpus: a human-annotated dialogue dataset for abstractive summarization', *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, Association for Computational Linguistics, Hong Kong, China, pp.70–79, DOI: 10.18653/v1/D19-5409.

Han, S., Mao, H. and Dally, W.J. (2016) 'Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding', *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016)*, ICLR/OpenReview, San Juan, Puerto Rico.

Han, S., Pool, J., Tran, J. et al. (2015) 'Learning both weights and connections for efficient neural networks', *Advances in Neural Information Processing Systems 28 (NeurIPS 2015)*, Curran Associates, Inc., Red Hook, NY, pp.1135–1143.

Hou, L., Huang, Z., Shang, L. et al. (2020) 'DynaBERT: dynamic BERT with adaptive width and depth', *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, Curran Associates, Inc., Red Hook, NY, pp.9782–9793.

Kurtic, E., Campos, D., Nguyen, T. et al. (2022) 'The optimal BERT surgeon: scalable and accurate second-order pruning for large language models', *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, pp.4163–4181, DOI: 10.18653/v1/2022.emnlp-main.279.

Lagunas, F., Charlaix, E., Sanh, V. et al. (2021) 'Block pruning for faster transformers', *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Association for Computational Linguistics, Dominican Republic, pp.10619–10629, DOI: 10.18653/v1/2021.emnlp-main.829.

Lewis, M., Liu, Y., Goyal, N. et al. (2020) 'BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension', *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, pp.7871–7880, DOI: 10.18653/v1/2020.acl-main.703.

Lin, Z., Liu, J.Z., Yang, Z. et al. (2020) 'Pruning redundant mappings in transformer models via spectral-normalized identity prior', *Findings of the Association for Computational Linguistics: EMNLP 2020*, Online: Association for Computational Linguistics, pp.719–730, DOI: 10.18653/v1/2020.findings-emnlp.64.

Liu, Z., Li, F., Li, G. et al. (2021) 'EBERT: efficient BERT inference with dynamic structured pruning', *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, Online: Association for Computational Linguistics, pp.4814–4823, DOI: 10.18653/v1/2021.findings-acl.425.

Mao, Y., Wang, Y., Wu, C. et al. (2020) 'LadaBERT: lightweight adaptation of BERT through hybrid model compression', *Proceedings of the 28th International Conference on Computational Linguistics*, International Committee on Computational Linguistics, Barcelona, Spain, pp.3225–3234, DOI: 10.18653/v1/2020.coling-main.287.

Mehnaz, L., Mahata, D., Gosangi, R. et al. (2021) 'GupShup: summarizing open-domain code-switched conversations', *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Association for Computational Linguistics, Dominican Republic, pp.6177–6192, DOI: 10.18653/v1/2021.emnlp-main.499.

Michel, P., Levy, O. and Neubig, G. (2019) 'Are sixteen heads really better than one?', *Advances in Neural Information Processing Systems 32 (Neu-rIPS 2019)*, Curran Associates, Inc., Red Hook, NY, pp.14014–14024.

Prasanna, S., Rogers, A. and Rumshisky, A. (2020) 'When BERT plays the lottery, all tickets are winning', *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, Online: Association for Computational Linguistics, pp.3208–3229, DOI: 10.18653/v1/2020.emnlp-main.259.

Radford, A. and Narasimhan, K. (2018) *Improving Language Understanding by Generative Pre-training*, OpenAI, San Francisco.

Sajjad, H., Dalvi, F., Durrani, N. et al. (2023) 'On the effect of dropping layers of pre-trained transformer models', *Computer Speech & Language*, Vol. 77, p.101429, DOI: 10.1016/j.csl.2022.101429.

Sanh, V., Wolf, T. and Rush, A.M. (2020) 'Movement pruning: adaptive sparsity by fine-tuning', *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, Curran Associates, Inc., Red Hook, NY, pp.20378–20389.

Voita, E., Talbot, D., Moiseev, F. et al. (2019) 'Analyzing multi-head self-attention: specialized heads do the heavy lifting, the rest can be pruned', *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Florence, Italy, pp.5797–5808, DOI: 10.18653/v1/P19-1580.

Wang, Z., Wohlwend, J. and Lei, T. (2020) 'Structured pruning of large language models', *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, Online: Association for Computational Linguistics, pp.6151–6162, DOI: 10.18653/v1/2020.emnlp-main.496.

Xia, M., Zhong, Z. and Chen, D. (2022) 'Structured pruning learns compact and accurate models', *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Dublin, Ireland, pp.1513–1528, DOI: 10.18653/v1/2022.acl-long.107.

Zhong, M., Yin, D., Yu, T. et al. (2021) 'QMSum: a new benchmark for query-based multi-domain meeting summarization', *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online: Association for Computational Linguistics, pp.5905–5921, DOI: 10.18653/v1/2021.naacl-main.472.