# Human Guided Exploitation of Interpretable Attention Patterns in Summarization and Topic Segmentation

**Raymond Li[†], Wen Xiao[†], Linzi Xing[†], Lanjun Wang[‡*], Gabriel Murray[§], Giuseppe Carenini[†]**

[†]University of British Columbia, Vancouver, BC, Canada

[‡]Tianjin University, Tianjin, China

[§]University of Fraser Valley, Abbotsford, BC, Canada

{raymondl, xiaowen3, lzxing, carenini}@cs.ubc.ca

wanglanjun@tju.edu.cn gabriel.murray@ufv.ca

## Abstract

The multi-head self-attention mechanism of the transformer model has been thoroughly investigated recently. In one vein of study, researchers are interested in understanding why and how transformers work. In another vein, researchers propose new attention augmentation methods to make transformers more accurate, efficient and interpretable. In this paper, we combine these two lines of research in a human-in-the-loop pipeline to first discover important task-specific attention patterns. Then those patterns are injected, not only to smaller models, but also to the original model. The benefits of our pipeline and discovered patterns are demonstrated in two case studies with extractive summarization and topic segmentation. After discovering interpretable patterns in BERT-based models fine-tuned for the two downstream tasks, experiments indicate that when we inject the patterns into attention heads, the models show considerable improvements in accuracy and efficiency.

## 1 Introduction

With transformer-based models (Vaswani et al., 2017) dominating the leaderboard for many key NLP tasks such as summarization (Liu and Lapata, 2019), topic segmentation (Lukasik et al., 2020), and sentiment analysis (Adhikari et al., 2019), their core multi-head self-attention mechanism has also been thoroughly investigated. In particular, to explain why and how transformers work, researchers have analyzed the learnt self-attention matrices of trained transformer models (e.g., Raganato and Tiedemann (2018); Kovaleva et al. (2019)), with Vig and Belinkov (2019) for instance, exploring the attention patterns in BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019), as well as analyzing their alignment with syntax.

Meanwhile, a parallel line of research has explored injecting predefined patterns into attention matrices of transformers in an attempt to reduce the run-time complexity of self-attention while maintaining competitive accuracy. This can be done by either replacing the attention weights with a fixed matrix (Raganato et al., 2020; Tay et al., 2021; Xiao et al., 2020); or alternatively by guiding the attention weights through more flexible masking strategies (Mihaylov and Frank, 2019; Child et al., 2019; Guo et al., 2019; Li et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Bai et al., 2021).

In this work, we propose and test a novel human-in-the-loop pipeline that to the best of our knowledge is the first attempt to combine research on analyzing self-attention with work on injecting patterns into attention matrices. To start, human users visually explore the attention matrices of transformers to identify task-specific patterns that could be formalized as a predicate. After quantitatively evaluating the patterns on the validation set, they can be injected into attention heads of transformer models to simultaneously improve task accuracy and make the model more efficient by sparsifying the attention matrices[1]. This is in contrast to previous work that mostly focuses on the trade-off between two metrics.

In both scenarios, we argue the interpretability of the resulting model is improved. We provide a justification of our claim based on the Predictive, Descriptive, and Relevant (PDR) framework proposed by Murdoch et al. (2019). Specifically, by injecting human-interpretable patterns into the model, we increase the model's descriptive accuracy by explicitly encoding useful relationships between input tokens in the attention weights while simultaneously improving the predictive accuracy in task performance. Further, the patterns are relevant for the problem since they are discovered in the human-in-the-loop process and are verified to

[*] Corresponding author.

[1] The implementation of our work is publicly available at: https://github.com/raymondzmc/Attention-Pattern-Exploitation

December 7-11, 2022 ©2022 Association for Computational Linguistics

be important for the task.

In order to test the feasibility and potential benefits of our approach, we run two case studies on the tasks of extractive summarization and topic segmentation using BERT-based models, and we find that: *(i)* Some of the important heads do have patterns with interpretable meaning, either lexical, local or positional. For instance, the matching token (i.e. the trend of attending to other tokens with the same id) is an important clue for the summarization model. *(ii)* We show that when the discovered patterns are injected to the attention heads of transformer models, both the task accuracy and efficiency of the model can be significantly improved. *(iii)* Additionally, we also propose a strategy to improve the performance of pretrained transformer models by injecting patterns through PALs.

## 2 Related Work

In §2.1 and §2.2, we describe the two lines of research that our work aims to combine. §2.3 summarizes recent trends on enhancing the interpretability of neural NLP models, while §2.4 introduces the two NLP tasks used for our case studies.

### 2.1 Attention Analysis in Transformers

Various works have investigated the attention head matrices in transformers (Raganato and Tiedemann, 2018; Clark et al., 2019; Kovaleva et al., 2019; Zhao and Bethard, 2020; Xiao et al., 2021), often with the aid of visualization tools (Vig, 2019; Hoover et al., 2020; Li et al., 2021). For examples, Vig and Belinkov (2019) visually explore attention patterns in BERT and GPT-2, analyzing their alignment with syntax. While Voita et al. (2019) characterize the functions of the attention heads in Machine Translation (MT) models (positional, syntactic, and rare words), and evaluate the importance of those head functions. More recently, Bian et al. (2021) find the redundancy in BERT's attention patterns to be both phase-independent (pretrained and fine-tuned) and task-agnostic. Lastly, Huber and Carenini (2022) infer discourse structures from the attention patterns of language models (BERT and BART), and find discourse information to be consistently captured in the same heads even when fine-tuned for different tasks. In this paper, we also aim to find task-specific important attention patterns, but in contrast to previous work that identifies and categorizes attention patterns, we propose a pipeline to leverage these patterns in improving models' performance and interpretability.

### 2.2 Attention Augmentation

We organize the related work on augmenting attention matrices into two categories. In the first category, attention weights are completely replaced with a fixed matrix. For example, Raganato et al. (2020) use fixed positional patterns in MT models and demonstrate benefits for low-resource scenarios, while Tay et al. (2021) replace the weights computed using dot-product self-attention with a random matrix, and report comparable performance with standard transformers. Later on, Xiao et al. (2020) expand their work by using embedded RST-style discourse trees as fixed attention matrices and show the effectiveness of discourse-based attention matrices for extractive summarization. In contrast, in the second category of attention augmentation works, masks are applied on top of the attention weights to either inject linguistic information (Yang et al., 2018; Mihaylov and Frank, 2019) or improve the efficiency of self-attention via fixed patterns (Child et al., 2019; Guo et al., 2019; Li et al., 2019; Ainslie et al., 2020). Just to describe a few prominent examples, Strubell et al. (2018) use bi-affine attention to learn syntactic dependencies in attention heads, and Bai et al. (2021) inject syntactic structures into BERT through extra attention layers. Concurrently, while Beltagy et al. (2020) use diagonal/vertical/horizontal patterns to respectively model local and global context, Zaheer et al. (2020) add patterns randomly by drawing inspiration from graph theory. In comparison, while in all previous works the designing of pre-defined patterns requires extensive trial and error, and only improve upon either the accuracy or efficiency at the expense of the other, we explore a strategy of discovering and assessing important attention patterns interactively in this paper. Not only do the discovered patterns help improve performance in terms of both accuracy and efficiency, they also reveal valuable insights regarding the internal workings of pretrained language models.

### 2.3 Model Interpretability

In the context of Machine Learning, interpretability can be defined as the description of the internals of a model in a way that is understandable to humans (Gilpin et al., 2018). With the rise of deep learning, various techniques have been proposed to interpret the inner workings of neural NLP models. For example, probing classifiers are often used for finding linguistic or knowledge information learned by neural networks (Conneau et al., 2018; Tenney et al., 2019; Pimentel et al., 2020; Voita and Titov, 2020; Hou and Sachan, 2021; Aghazadeh et al., 2022), while behaviour testing aims at understanding how models behave through inferences under different controlled settings (McCoy et al., 2019; Ross and Pavlick, 2019; Ribeiro et al., 2020; Koh et al., 2021; Goel et al., 2021). In contrast, our work is an example of making interpretability an inherent attribute of the neural models (e.g. Chen and Ji (2020); Hu et al. (2021)), with human-distinguishable patterns revealing insights regarding a subset of parameters in the model.

### 2.4 NLP Tasks used in the two Case Studies

Extractive summarization is the task of picking the most representative sentences as the summary for the given document(s). Current state-of-the-art models, which are mostly based on large-scale pretrained language models (Liu and Lapata, 2019; Zhong et al., 2020; Jia et al., 2020; Ruan et al., 2022), can deliver good performance, but why and how such models work so well still remain an open question. In our case study, we adopt the popular BERTSum (Liu and Lapata, 2019).

Topic segmentation is the task of breaking stretches of running text into smaller topical-coherent segments consisting of one or more sentences addressing a common topic. Recently, more research work frames the task in the supervised learning paradigm and uses neural models such as Bi-LSTMs (Koshorek et al., 2018; Xing et al., 2020) and transformer (Glavas and Somasundaran, 2020; Lo et al., 2021) as the backbone, due to the availability of large-scale labeled benchmarks sampled from *Wikipedia*. These proposed neural topic segmentation models achieve state-of-the-art performance on monologue text by formulating the problem as a sequence labeling task, where the predicted label of each sentence indicates whether or not it is the end of a segment. In our case study, we adopt Cross-Segment BERT (Lukasik et al., 2020).

## 3 Proposed Generic Pipeline

As an overview, we first briefly describe the proposed pipeline (Figure 1). Specifically, given a trained model, users are asked to first discover important patterns using the visual interface (Li et al., 2021) by following three steps:
*Step 1* (§3.1.1): Estimate the importance scores for all the heads on the validation set, and find important heads that stand out.
*Step 2* (§3.1.2): Discover relevant patterns in the important heads, using criteria described in §3.1.2.
*Step 3* (§3.1.3): Evaluate and validate the patterns to confirm their global relevance.

Once the important patterns are identified, there are two common approaches (i.e. fixing and masking) to inject them as constraints to the attention matrices in the transformer-based neural models (see §3.2). The pipeline also enables two scenarios, in which injecting the patterns can be beneficial: the first one is to train a new model with the patterns injected, while the second one is to enhance the original model.

### 3.1 Discover Patterns from Attention

In this section we provide details of the three steps for discovering patterns from the attention heads. The three steps are illustrated in Figure 1 (B).

#### 3.1.1 Step 1: Estimate Head Importance

Although the multi-head self attention mechanism in transformers allows the model to learn multiple types of relationships between input representations across a single hidden layer, the importance of the individual attention heads can vary depending on the downstream tasks. In practice, we propose the use of scalable gradient-based methods (Michel et al., 2019; Voita et al., 2019; Molchanov et al., 2019) for an efficient estimation of head importance, and take the top-K heads at each layer to find important patterns for the task (§3.1.2). Note that $K$ can be adjusted based on the availability of human users and the size of the model.

#### 3.1.2 Step 2: Find Attention Patterns

Once the the most important heads are identified, their attention distributions are inspected to look for patterns.

We define an attention pattern to be interpretable *iff* it can be modeled as a predicate $P$ between any pair of input tokens $(x_i, x_j)$. For instance, the positional pattern ‘preceding token’ would be true if $x_i$ appears before $x_j$. Candidate patterns can be discovered following two criteria: 1) they are beneficial for the downstream task; 2) they occur consistently among relevant tokens.

#### 3.1.3 Step 3: Evaluate Attention Patterns

With a pattern discovered in §3.1.2, this step confirms the pattern’s global relevance by empirically

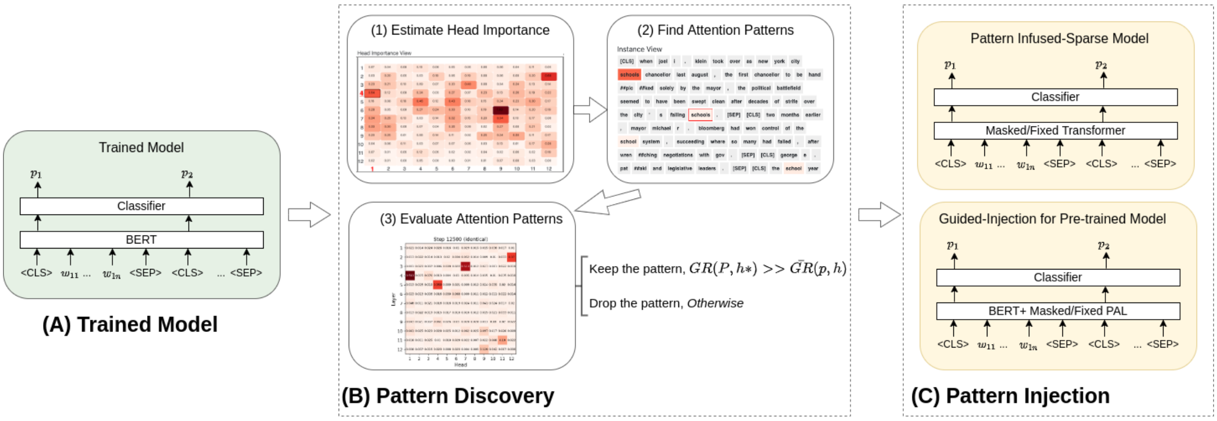

Trained Model
$p_1$
$p_2$
Classifier
BERT
<CLS> $w_{11}$ ... $w_{1n}$ <SEP> <CLS> ... <SEP>
(A) Trained Model
(1) Estimate Head Importance
(2) Find Attention Patterns
(3) Evaluate Attention Patterns
Keep the pattern, $GR(P, h*) >> \vec{GR}(p, h)$
Drop the pattern, Otherwise
(B) Pattern Discovery
Pattern Infused-Sparse Model
Classifier
Masked/Fixed Transformer
Guided-Injection for Pre-trained Model
Classifier
BERT+ Masked/Fixed PAL
(C) Pattern Injection

Figure 1: The overview of our proposed generic pipeline. Given **(A)** a trained model for a specific task, our pipeline can be divided into two main parts: **(B)** pattern discovery and **(C)** pattern injection.

measuring the proportion of attention values aligning with the pattern. For each attention head, the associated predicate is evaluated over the entire validation set to ensure the pattern is not appearing by chance on the certain data that the user happen to look at.

Specifically, we define the global relevance (GR) of a pattern $P$ for a head $h$ as follows:

$$\text{GR}(P,h) = \frac{1}{|X|}\sum_{x\in X}\frac{\sum_i^{|x|}\sum_j^{|x|}\alpha_{i,j}^{(x,h)}\cdot \mathbb{1}_{P(x_i,x_j)}}{|x|} \tag{1}$$

where the attention value from the token $x_i$ to $x_j$ on the head $h$ for an input sample $x$, denoted as $\alpha_{i,j}^{(x,h)}$, is aggregated if and only if $P(x_i, x_j)$ holds. To validate a pattern's generality, the relevance is averaged over the validation set $X$.

## 3.2 Inject Patterns

As illustrated in Figure 1 (C), after extracting the patterns following the three steps in §3.1, we propose to inject the patterns into attention matrices with two methods (§3.2.1), and discuss two practical scenarios (§3.2.2) where they can be beneficial for the downstream tasks.

### 3.2.1 Methods for injecting Patterns

In this work, we inject the discovered patterns by either fixing or masking the attention weights prior to the softmax function. For fixed attention weights, the attention logits in the scaled-dot-product attention is replaced with a fixed (possibly input dependent) matrix such that:

$$\text{FixAttn}(V, X) = \sigma(F^{(P)}(X))V \tag{2}$$

where $\sigma$ is the softmax operation, $V$ is the value vectors, and $F(X) \in [0, 1]$ computes a binary matrix from the input sequence $X$ based on the predicated $P$ for the specific pattern. Similarly, a pattern can also be injected by casting a mask over the attention weights computed from the key and query vectors, as:

$$\text{MaskAttn}(Q, K, V, X) = \sigma(M^{(P)}(X)+QK^T)V \tag{3}$$

where $M(X) \in [0, -\infty)$ computes the desired behaviour in the same fashion as $F(X)$, and is added to the attention logits to approximate the multiplication of the attention distribution by a weight.

Although the two methods are very similar with respect to the improvement they contribute (see §4), masking allows more flexibility and is generally used for patterns with a large number of applicable tokens, while fixing is more rigid and better suited for a small number of applicable tokens.

### 3.2.2 Scenarios for Injecting Patterns

In practice, patterns can be injected in at least two scenarios: (i) injecting patterns directly into the attention heads of transformer-based models, and (ii) injecting patterns into pretrained transformer models using techniques such as the Projected Attention Layers (Stickland and Murray, 2019). We conduct case studies for these two scenarios in §4.

# 4 Case Studies

In this section, we demonstrate the effectiveness of our pipeline in two NLP tasks (extractive summarization and topic segmentation) and discuss our findings in detail.

## 4.1 Models for Tasks

We adopt the popular BERTSum (Liu and Lapata, 2019) for extractive summarization. With the con-

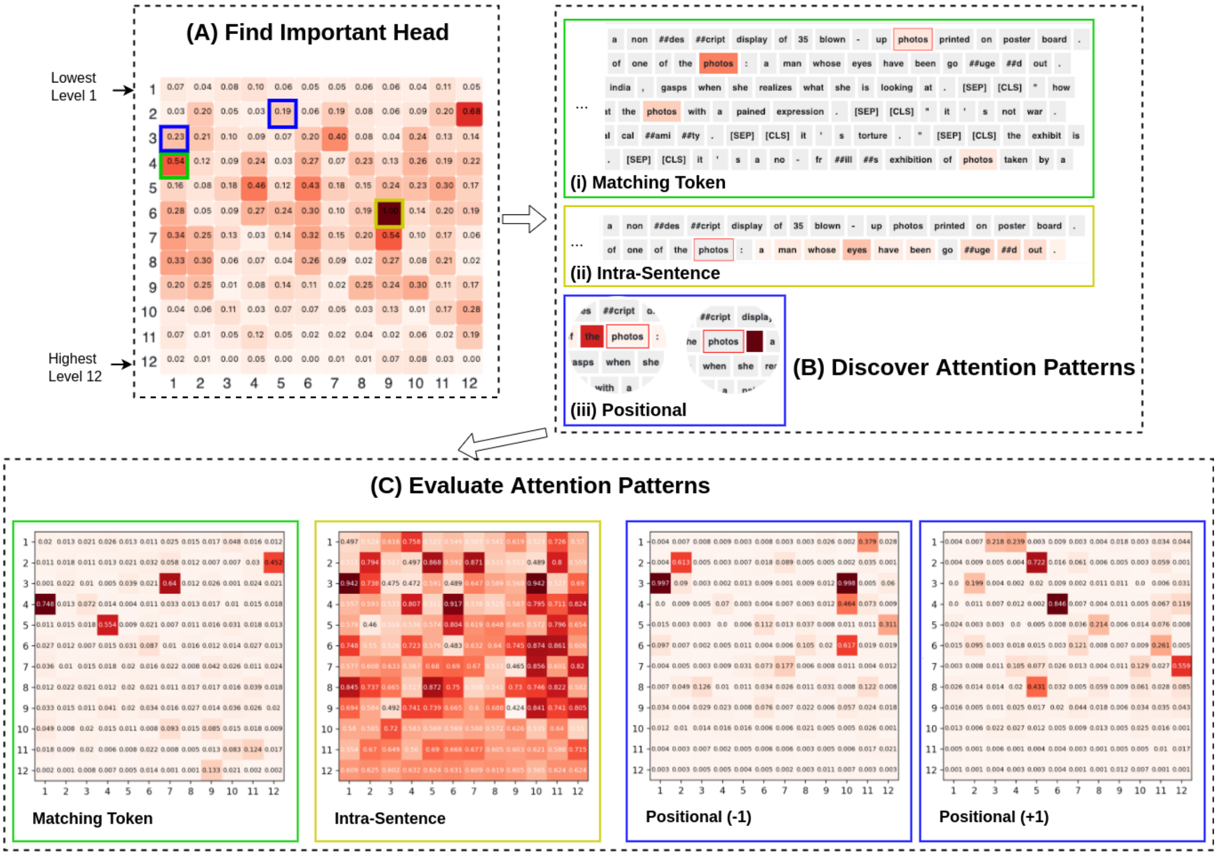


Figure 2: Example of Pattern Extraction in the extractive summarization case study.[2] **(A)** We first find important heads, before **(B)** identifying the three interpretable patterns (highlighted in Green, Olive and Blue, respectively): (i) Matching token, (ii) Intra-Sentence, and (iii) Positional.. Finally, **(C)** each pattern is evaluated with the global relevance score (GR) on all of the attention heads. For the purpose of illustration, we display one attention head with significantly larger GR in for each of the three identified patterns.

textualized representation from BERT, the model uses a binary classifier to predict whether each sentence belongs in the summary. We train the model on the CNN/DM dataset (See et al., 2017), and use ROUGE (Lin, 2004) as the evaluation metric.

We adopt Cross-Segment BERT (Lukasik et al., 2020) for topic segmentation, where a candidate segment boundary is first represented by its left and right context, and then passed through a binary classifier to predict whether the candidate is a topical segment boundary. The model is trained on the WikiSection dataset (Arnold et al., 2019), and the F1-score is used as evaluation metric for validation.

## 4.2 Discover Patterns from Attentions

Using the two models from §4.1, as we discover that similar attention patterns exist in the important heads for both tasks, the two case studies are presented together. Without loss of generality, we will use extractive summarization as the running example task (Figure 2) to illustrate the process of pattern discovery. We also apply the same process to topic segmentation.

### 4.2.1 Find Important Heads

We adapt the Taylor expansion method (Molchanov et al., 2019) as a proxy score for the head importance estimation. Following Li et al. (2021), we use the first-order expansion to avoid the overhead from computing the Hessian, where the gradient w.r.t. the validation loss is summed over all parameters of an attention head to estimate its importance.

The importance score heatmap of all heads is visualized in Figure 2 (A), revealing that head importance is not uniformly distributed, i.e. a small number of heads play a dominant role for the summarization task, as observed in Michel et al. (2019).

### 4.2.2 Discover and Evaluate Patterns

To discover task-specific patterns, we analyze the top-3 most important heads of each layer, and

[2] **(A)** and **(B)** of Figure 2 are captured from the visual interface presented in Li et al. (2021).

look for human-interpretable relationships encoded in the attention weights. In practice, we use the instance-level interactions provided by the visual framework (Li et al., 2021), and randomly select 5 validation examples per task for our analysis. The entire process takes less than one hour to complete for each task, where we manually examine the attention weights for less than half of the tokens for each example. It is worth noting that detailed analysis regarding the trade-off between human cost and pattern recall would require extensive user studies beyond the scope of this work.

After discovering patterns, we assess the global relevance of each patterns on the validation set, where the pattern is kept only if the corresponding predicate $P$ exists in at least one significantly relevant head. In our case studies, we use the 3-sigma rule to determine the significance of a pattern. Specifically, patterns with at least one head over 3 standard deviations above the GR mean (over all the heads) are kept for further applications.

After verifying on the validation set, we discover three patterns consistently existing in both tasks (over 50% of important heads). This suggests that important patterns are generalizable across multiple NLP tasks, which is consistent with the findings in Bian et al. (2021). Further analysis also shows that the attention patterns are consistent after fine-tuning, where we report an average Jensen-Shannon Divergence of 0.01 between the attention distributions of BERTSum across 3 random seeds. We hope our findings provide motivation for the in-depth study of pattern importance in different NLP tasks. Lastly, while it may be argued that this step of the pipeline can be automated by directly evaluating the importance and relevance of predefined patterns (e.g. syntax, discourse) based on intuitions, as indicated below, our interactive approach allows the discovery of interpretable patterns which otherwise would be hard to define due to the infinite search space of possible patterns. Next, we describe the three discovered patterns in detail.

**Matching Token (Green in Figure 2)** This pattern describes the "attending to matching tokens" behaviour, where the attention value $\alpha_{i,j}^h$ between input tokens $x_i$ and $x_j$ on the head $h$ is high whenever $x_i = x_j$. For example, as shown in Figure 2 (i), the token "photo" mostly attends to other appearances of the token "photo" in the input sequence. To evaluate whether this pattern has a large global relevance for any head, we only consider tokens that appear at least twice within a single documents, and compute GR (Eq. 1), in which $P(x_i, x_j)$ holds if and only if $x_i = x_j$, i.e. $\mathbb{1}_{P(x_i,x_j)} = (\mathbb{1}_{\text{freq}(x_i)>1}) \times (\mathbb{1}_{x_i=x_j})$.

The evaluation results show that there are several heads for which the matching token pattern has high global relevance (See the Green box in Figure 2). Interestingly, these heads are more prominent (in the importance heatmap) for the extractive summarization task, suggesting this pattern is especially important for summarization models during inference.

**Intra-Sentence/Context (Olive in Figure 2)** This pattern describes the behaviour of only attending to tokens within a text span. For summarization, these heads will focus on attending tokens within the same sentence (Figure 2 (ii)). Similarly, the same heads in topic segmentation models will focus on attending tokens within the same context (left or right). To evaluate this pattern, GR is computed with $P(x_i, x_j)$ holding *iff* $x_i$ and $x_j$ occur within the same text span. Figure 2 (C) reveals that this pattern appears more frequently in the mid to upper layers of the transformer encoder.

**Positional (Blue in Figure 2)** Similar to Kovaleva et al. (2019), we also observe "positional heads", which focus specifically on either the preceding or following tokens, i.e., either $\alpha_{i,i-1}^h$ or $\alpha_{i,i+1}^h$ have high values (Figure 2 (iii)). To evaluate this pattern, GR is computed with $P(x_i, x_j)$ holding *iff* $j = i - 1$ for preceding postional heads and $j = i + 1$ for succeeding positional heads. The pattern is verified to exist in the lower layers of the encoder, shown in the blue boxes of Figure 2 (C).

**Other Patterns** In addition to the three patterns mentioned above, we also observe heads that focus on attending to special tokens (e.g. [CLS], [SEP]) or punctuations (e.g. periods). However, we find that attention heads with this behaviour are generally less important for the task (outside top-3), and therefore omitted them from the next step of our pipeline.

On the other hand, we also find uninterpretable attention patterns in some of the important heads of each layer. As hypothesized by previous works (Clark et al., 2019), these attention heads might be performing complex linguistic operations in combination with other heads. We leave the verification, interpretation and the efficient injection of these patterns into the models as a direction for future work.

### 4.3 Injecting Patterns to Models

After uncovering potentially important patterns and confirming their relevance, we inject them to transformer-based models for the task of summarization and topic segmentation through masking and fixing the attention weights. While we only perform the pattern discovery process on the CNN/DM and WikiSection datasets, we inject the discovered patterns to two other datasets (NYT-50 (Sandhaus, 2008) for summarization and Wiki-727K (Arnold et al., 2019) for topic segmentation) to demonstrate that our discovered patterns are generalizable in "cross-dataset" settings[3].

#### 4.3.1 Method for Fixing and Masking

The patterns identified from our analysis can be injected into an attention head through masking or fixing its corresponding attention weight matrix. Specifically, for the matching token pattern, we apply an attention mask which enforces that when a token appears more than once in the document, it should attend only to other occurrences of itself:

$$M_{i,j}^{(m)} = \begin{cases} 1 & (x_i = x_j) \vee (\text{freq}(x_i) = 1) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where the constraint is removed for tokens occurring only once in the document.

Similarly, for intra-sentence/intra-context attention, the attention mask specifies that only tokens within the same boundary can attend to each others, where:

$$M_{i,j}^{(s)} = \begin{cases} 1 & \text{SameBoundary}(x_i, x_j) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Lastly, we use a fixed attention matrix to encode the two positional patterns with:

$$F_{i,j}^{(-1)} = \begin{cases} 1 & j = i - 1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

With $F_{i,j}^{(+1)}$ being the same, but equal to 1 for $j = i+1$. We use fixed attention matrices for these patterns to save computational overhead since it has the same effect as applying the mask (each row is a one-hot vector). This is similar to the method proposed by Raganato et al. (2020), but we only fix for the preceding and succeeding token patterns.

#### 4.3.2 Pattern-Infused Sparse Transformers

In the first round of experiments, we inject the four patterns on smaller transformer models to demonstrate their effectiveness on both tasks. Since the goal of these experiments is to assess the benefits brought by these patterns, we do not perform extensive hyper-parameter search when injecting these patterns (e.g. on which layer, etc.).

Under both settings, each of the four patterns (including two positional patterns) is injected in a separate attention head across all layers in the model. Motivated by studies on the trade-off between sparsity ratio and task performance, we adopt the sparsity ratio used by previous works (Shi et al., 2021; Wang et al., 2022): $\rho = 1 - |M|/N^2$, where $|M|$ denotes the number of non-zero elements in the attention mask, and $N$ denotes the length of the example. Given the sparsity $\rho$, the complexity of self-attention is thus reduced to $\mathcal{O}\big((1-\rho)n^2\big)$ (Shi et al., 2021). To investigate how the sparsity ratio affects the performance of our pattern-infused models, we experiment with different number of heads to inject our patterns, where the sparsity ratio increases along with the number of heads (with patterns).

As shown in Table 1, our pattern-infused models outperform the plain transformer models for both the CNN/DM and NYT-50 datasets under all three settings (6 Layer 8 Heads, 6 Layer 12 Heads, and 6 Layer 12 Heads with BERT embeddings). Similarly for topic segmentation, results also show that the pattern-injection approach substantially outperforms the vanilla transformer across all metrics. It is worth emphasizing that the performance gain is slightly higher for summarization models. When normalized by the ROUGE scores of extractive oracle summaries[4], the pattern-infused summarization models achieve an average $15\%$ improvement over the baselines, while the topic-segmentation models achieve a $12\%$ improvement over the baselines. In-line with prior work (McCoy et al., 2020), we also find that the performance is consistent across random seeds, where we report an extremely low standard deviation of 0.03 (ROUGE) and 0.002 (F1) for extractive summarization and topic segmentation, respectively. Overall, the results from our experiments convincingly demonstrates the benefits of our approach and the generalizability of the

[3] Results shown in Sec. 4 are without the Trigram Blocking trick, and more results *with* it are in Appendix D

[4] As reported by Liu and Lapata (2019), the ROUGE scores (R-1/R-2/R-L) of the oracle upper bound for CNN/DM and NYT-50 are respectively, 52.59/31.24/48.87 and 49.18/33.24/46.02.

| Model | Sparsity ($\rho$) | Extractive Summarization | | | | | | Topic Segmentation | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CNN/DM | | | NYT-50 | | | WikiSection | | | Wiki-727K | | |
| | | R-1 | R-2 | R-L | R-1 | R-2 | R-L | P | R | F-1 | P | R | F-1 |
| 6 Layer 8 Heads | | | | | | | | | | | | | |
| Transformer | 0 | 40.50 | 18.22 | 36.94 | 45.86 | 26.83 | 38.23 | 0.698 | 0.647 | 0.671 | 0.647 | 0.243 | 0.353 |
| +Patterns (4/8) | 0.43 | **41.42** | **18.94** | **37.92** | 47.11 | 27.89 | 39.34 | 0.744 | **0.711** | **0.727** | 0.624 | **0.318** | **0.421** |
| +Patterns (8/8) | 0.86 | 41.37 | 18.92 | 37.89 | **47.17** | **27.94** | **39.37** | **0.771** | 0.666 | 0.714 | **0.668** | 0.274 | 0.395 |
| 6 Layer 12 Heads | | | | | | | | | | | | | |
| Transformer | 0 | 40.53 | 18.22 | 36.98 | 46.07 | 27.01 | 38.42 | 0.681 | 0.680 | 0.681 | 0.687 | 0.255 | 0.372 |
| +Patterns (4/12) | 0.29 | 41.58 | 19.10 | 38.10 | 46.84 | 27.68 | 39.08 | 0.752 | **0.717** | **0.734** | 0.643 | **0.350** | **0.453** |
| +Patterns (8/12) | 0.58 | 41.66 | 19.17 | 38.17 | 47.15 | **27.95** | **39.38** | **0.757** | 0.701 | 0.730 | 0.655 | 0.342 | 0.450 |
| +Patterns (12/12) | 0.86 | **41.68** | **19.16** | **38.19** | **47.17** | 27.94 | **39.38** | 0.756 | 0.702 | 0.728 | **0.663** | 0.318 | 0.430 |
| 6 Layer 12 Heads (with BERT Embeddings) | | | | | | | | | | | | | |
| Transformer | 0 | 40.74 | 18.40 | 37.20 | 46.07 | 26.98 | 38.37 | 0.738 | 0.674 | 0.704 | 0.665 | 0.415 | 0.511 |
| +Patterns (4/12) | 0.29 | 41.49 | 19.07 | 37.99 | 47.02 | 27.83 | 39.21 | **0.782** | 0.715 | 0.747 | **0.674** | **0.423** | **0.520** |
| +Patterns (8/12) | 0.58 | 41.57 | 19.11 | 38.08 | 47.16 | **27.96** | **39.40** | 0.760 | **0.737** | **0.748** | 0.665 | 0.421 | 0.515 |
| +Patterns (12/12) | 0.86 | **41.61** | **19.15** | **38.14** | **47.17** | 27.95 | 39.37 | 0.761 | 0.731 | 0.745 | 0.666 | 0.367 | 0.473 |

Table 1: Results for the two tasks (four datasets) under different settings, where we report the average performance across the top-3 checkpoints. The parenthesis (e.g. 4/8) denotes the number of heads with patterns injected, while sparsity ($\rho$) is computed from the average of the 4 datasets.

| Model | R-1 | R-2 | R-L |
|---|---|---|---|
| Transformer | 40.50 | 18.22 | 36.94 |
| + match (m) | +0.03 | +0.12 | +0.07 |
| + intra (i) | +0.05 | +0.06 | +0.12 |
| + pos (p) | -0.16 | -0.17 | -0.13 |
| + m +i | +0.84 | +0.65 | +0.91 |
| + m +p | +0.07 | +0.11 | +0.11 |
| + i + p | -0.01 | +0.03 | +0.07 |
| + all | **+0.92** | **+0.72** | **+0.98** |

Table 2: Ablation study on the CNN/DM dataset with the 6 Layer 8 Head transformer setting.

patterns discovered by our pipeline.

In addition, while a higher sparsity ratio causes a slight decrease in performance under some scenarios, we find that even with a ratio of 0.86, our model still significantly outperforms the vanilla transformer across all settings. This is in contrast to the findings by previous work (Child et al., 2019; Guo et al., 2019; Li et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Shi et al., 2021), where the high sparsity ratio from fixed patterns often results in performance degradation from the vanilla transformer. These findings from our work provide crucial insights for designing more energy efficient models in the future.

Overall, with the discovered patterns injected, our models are arguably more interpretable than plain transformers on both tasks, as we know with certainty the information encoded in each masked/fixed attention heads. To further justify our claim of interpretability, the attention heads with patterns injected tend to have higher importance scores than the other heads[5], suggesting that such patterns are effectively leveraged by the model.

To study the contribution of individual patterns, we perform an ablation study by injecting all combinations of patterns on CNN/DM using the transformer model with 6 layers and 8 heads[6]. From Table 2, we observe that injecting matching token and intra-sentence together achieves the strongest improvement in accuracy among all combinations, only slightly lower than injecting all patterns. Meanwhile, the gains from injecting patterns separately are only marginal. One intriguing explanation is that these two patterns allows the model to learn sentence-level features based on term frequency (plausibly similar to TF-IDF (Jones, 1972)), where higher scores are assigned to sentences containing frequently appearing tokens. Additionally, although injecting *only* the positional patterns causes the performance to degrade, it works better when combined with the two other patterns. We hypothesize that positional patterns need to be combined with patterns with more global context in order to be more effectively utilized.

#### 4.3.3 Guided Pattern Injection into Pre-trained Models

We then experiment with injecting the patterns back into the pre-trained transformer encoder. In particular, we inject them through additional attention heads in the form of a Projected Attention Layer (PAL) (Stickland and Murray, 2019), along with the parameters of the original model. Details regarding PALs are described in Appendix A.

[5] An illustrative example is shown in Appendix C.1

[6] Ablation study results for topic segmentation (WikiSection) can be found in Appendix E

| Model | CNN/DM (in-dataset) | | | NYT-50 (cross-dataset) | | |
|---|---|---|---|---|---|---|
| | R-1 | R-2 | R-L | R-1 | R-2 | R-L |
| BERTSum | 42.33 | 19.88 | 38.86 | 48.37 | 29.25 | 40.72 |
| + PAL | 42.34 | 19.88 | 38.86 | 48.56 | 29.41 | 40.91 |
| + PAL + Patterns | **42.58** | **20.05** | **39.10** | **48.74** | **29.60** | **41.11** |

Table 3: ROUGE F-scores of PAL with pretrained models for extractive summarization. All metrics were significantly better than the baselines with a confidence level of 99% according to the Bootstrap Significance test (Dror et al., 2018).

The hidden size of our PALs is 256, which consists of 4 additional attention heads ($d_k = d_v = d_q = 64$). PAL is added in each of the 12 BERT layers, where our patterns are injected in the 4 PAL attention heads. To ensure the changes in performance are due to the patterns rather than the additional parameters, we also compare against adding PAL without injecting the patterns.

Results in Table 3 indicate that injecting the patterns in PAL (+PAL+Patterns) surprisingly improves BERTSum's performance on both datasets, where the performance gains on the NYT-50 are similar (or even slightly better) than on the in-domain CNN/DM dataset, supporting the generality of the discovered patterns. Additionally, as it was the case for the transformers with patterns injected, visualizing the head importance scores reveals that the PAL heads with patterns injected are significantly more important (by two orders of magnitude) than the PAL heads without patterns injected[7], indicating that the interpretable patterns are important features during model inference.

In summary, the key aim of our experiments was to verify consistent improvements over our own baselines under the same settings in order to probe the benefits (effectiveness and efficiency) of the discovered patterns for the task. Therefore, we do not perform extensive tuning to achieve the same results reported by Liu and Lapata (2019).

## 5 Conclusion and Future Work

In this paper, we propose a generic human-in-the-loop pipeline, which combines two popular research directions, where the findings from an analysis of the multi-head self-attention mechanism in transformers can be utilized to create more accurate and interpretable transformer models. A human analyzes the attention heads of a task-specific model, discovers and verifies potentially meaningful patterns, and injects them into the attention heads of models. By running a case study on two NLP tasks, we show the effectiveness of our pipeline. We do discover meaningful patterns in some important heads, and the relationships encoded in the patterns help us understand the features used by the model for both tasks. Furthermore, by injecting the patterns into the smaller models and the original model, the performance and interpretability get improved in both cases.

As future work, we plan to apply our pipeline to other NLP tasks (e.g. language modeling, abstractive summarization) and explore and verify whether the important patterns from one task can be transferable to another task. Similarly, we also plan to apply our pipeline to different model variants to examine and compare the patterns encoded in the attention weights. In the long term, our pipeline could be naturally automated by replacing the pattern discovery step with evaluating predefined linguistic patterns. However, assessing the efficiency gains from injecting such patterns (requiring ground-truth annotations) would require more in-depth studies beyond the scope of this paper. Finally, since human factors are an important aspect of interpretability, we plan to conduct extensive user studies across different NLP tasks and model sizes to examine the trade-off between human-cost and the coverage of discovered patterns.

## Limitations

The scope of our case studies is limited to English datasets consisting of long documents for BERT-based models. Additionally, we only adopt the visual interface proposed by Li et al. (2021) due to its support for long documents, and leave the design and implementation of additional visualization techniques as a venue for future work.

## Acknowledgements

We thank all reviewers for providing valuable feedbacks and suggestions for us to incorporate into the final version. The authors are supported by an NSERC Discovery Grant [RGPIN-2018-06806] This work is also supported in part by the Institute for Computing, Information and Cognitive Systems (ICICS) at the University of British Columbia.

## References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*.

[7] An illustrative example is shown in Appendix C.2

Ehsan Aghazadeh, Mohsen Fayyaz, and Yadollah Yaghoobzadeh. 2022. Metaphors in pre-trained language models: Probing and generalization across datasets and languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2037–2050, Dublin, Ireland. Association for Computational Linguistics.

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, Online. Association for Computational Linguistics.

Sebastian Arnold, Rudolf Schneider, Philippe Cudré-Mauroux, Felix A. Gers, and Alexander Löser. 2019. SECTOR: A neural model for coherent topic segmentation and classification. *Transactions of the Association for Computational Linguistics*, 7:169–184.

Jiangang Bai, Yujing Wang, Yiren Chen, Yaming Yang, Jing Bai, Jing Yu, and Yunhai Tong. 2021. Syntax-BERT: Improving pre-trained transformers with syntax trees. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3011–3020, Online. Association for Computational Linguistics.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Yuchen Bian, Jiaji Huang, Xingyu Cai, Jiahong Yuan, and Kenneth Church. 2021. On attention redundancy: A comprehensive study. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 930–945, Online. Association for Computational Linguistics.

Hanjie Chen and Yangfeng Ji. 2020. Learning variational word masks to improve the interpretability of neural text classifiers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4236–4251, Online. Association for Computational Linguistics.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE.

Goran Glavas and Swapna Somasundaran. 2020. Two-level transformer and auxiliary coherence modeling for improved text segmentation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, pages 2306–2315.

Karan Goel, Nazneen Fatema Rajani, Jesse Vig, Zachary Taschdjian, Mohit Bansal, and Christopher Ré. 2021. Robustness gym: Unifying the NLP evaluation landscape. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 42–55, Online. Association for Computational Linguistics.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-transformer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1315–1325, Minneapolis, Minnesota. Association for Computational Linguistics.

Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. 2020. exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 187–196, Online. Association for Computational Linguistics.

Yifan Hou and Mrinmaya Sachan. 2021. Bird's eye: Probing for linguistic graph structures with a simple information-theoretic approach. In *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)*, pages 1844–1859, Online. Association for Computational Linguistics.

Xiang Hu, Haitao Mi, Zujie Wen, Yafang Wang, Yi Su, Jing Zheng, and Gerard de Melo. 2021. R2D2: Recursive transformer based on differentiable tree for interpretable hierarchical language modeling. In *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)*, pages 4897–4908, Online. Association for Computational Linguistics.

Patrick Huber and Giuseppe Carenini. 2022. Towards understanding large-scale discourse structures in pre-trained and fine-tuned language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2376–2394, Seattle, United States. Association for Computational Linguistics.

Ruipeng Jia, Yanan Cao, Hengzhu Tang, Fang Fang, Cong Cao, and Shi Wang. 2020. Neural extractive summarization with hierarchical attentive heterogeneous graph network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3622–3631, Online. Association for Computational Linguistics.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. 2021. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR.

Omri Koshorek, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant. 2018. Text segmentation as a supervised learning task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 469–473, New Orleans, Louisiana. Association for Computational Linguistics.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. 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 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Raymond Li, Wen Xiao, Lanjun Wang, Hyeju Jang, and Giuseppe Carenini. 2021. T3-vis: visual analytic for training and fine-tuning transformers in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 220–230, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems*, 32.

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.

Yang Liu and Mirella Lapata. 2019. 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 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Kelvin Lo, Yuan Jin, Weicong Tan, Ming Liu, Lan Du, and Wray Buntine. 2021. Transformer over pre-trained transformer for neural text segmentation with enhanced topic coherence. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3334–3340, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Michal Lukasik, Boris Dadachev, Kishore Papineni, and Gonçalo Simões. 2020. Text segmentation by cross segment attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4707–4716, Online. Association for Computational Linguistics.

R. Thomas McCoy, Junghyun Min, and Tal Linzen. 2020. BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 217–227, Online. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of* *the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Todor Mihaylov and Anette Frank. 2019. Discourse-aware semantic self-attention for narrative reading comprehension. 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 2541–2552, Hong Kong, China. Association for Computational Linguistics.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11264–11272.

W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. 2019. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2020. Fixed encoder self-attention patterns in transformer-based machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 556–568, Online. Association for Computational Linguistics.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium. Association for Computational Linguistics.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2018. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8119–8127.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Alexis Ross and Ellie Pavlick. 2019. How well do NLI models capture verb veridicality? 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 2230–2240, Hong Kong, China. Association for Computational Linguistics.

Qian Ruan, Malte Ostendorff, and Georg Rehm. 2022. HiStruct+: Improving extractive text summarization with hierarchical structure information. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1292–1308, Dublin, Ireland. Association for Computational Linguistics.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. 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, Vancouver, Canada. Association for Computational Linguistics.

Han Shi, Jiahui Gao, Xiaozhe Ren, Hang Xu, Xiaodan Liang, Zhenguo Li, and James Tin-Yau Kwok. 2021. Sparsebert: Rethinking the importance analysis in self-attention. In *International Conference on Machine Learning*, pages 9547–9557. PMLR.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995. PMLR.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2021. Synthesizer: Rethinking self-attention for transformer models. In *International Conference on Machine Learning*, pages 10183–10192. PMLR.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. 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, Florence, Italy. Association for Computational Linguistics.

Elena Voita and Ivan Titov. 2020. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.

Zihan Wang, Jiuxiang Gu, Jason Kuen, Handong Zhao, Vlad Morariu, Ruiyi Zhang, Ani Nenkova, Tong Sun, and Jingbo Shang. 2022. Learning adaptive axis attentions in fine-tuning: Beyond fixed sparse attention patterns. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 916–925, Dublin, Ireland. Association for Computational Linguistics.

Wen Xiao, Patrick Huber, and Giuseppe Carenini. 2020. Do we really need that many parameters in transformer for extractive summarization? discourse can help ! In *Proceedings of the First Workshop on Computational Approaches to Discourse*, pages 124–134, Online. Association for Computational Linguistics.

Wen Xiao, Patrick Huber, and Giuseppe Carenini. 2021. Predicting discourse trees from transformer-based neural summarizers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4139–4152, Online. Association for Computational Linguistics.

Linzi Xing, Brad Hackinen, Giuseppe Carenini, and Francesco Trebbi. 2020. Improving context modeling in neural topic segmentation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 626–636, Suzhou, China. Association for Computational Linguistics.

Jiacheng Xu and Greg Durrett. 2019. Neural extractive text summarization with syntactic compression. 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 3292–3303, Hong Kong, China. Association for Computational Linguistics.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Discourse-aware neural extractive text summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5021–5031, Online. Association for Computational Linguistics.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4449–4458, Brussels, Belgium. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297.

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.

Yiyun Zhao and Steven Bethard. 2020. How does BERT's attention change when you fine-tune? an analysis methodology and a case study in negation scope. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4729–4747, Online. Association for Computational Linguistics.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online. Association for Computational Linguistics.

# A Projected Attention Layer (PAL)

Projected Attention Layer (PAL) proposed by Stickland and Murray (2019) as adaptor modules for the pretrained model. Similar to the design in the original work, the PAL layer runs "parallel" with the pretrained encoder layer where their respective output are added in a residual manner (Rebuffi et al., 2018), such that:

$$\mathbf{h}^{l+1} = \mathrm{LN}\big(\mathbf{h}^{l} + \mathrm{SA}(\mathbf{h}^{l}) + \mathrm{PAL}(\mathbf{h}^{l})\big) \quad (7)$$

where LN denotes LayerNorm, and SA is the self-attention layer in the pretrained encoder. In each PAL layer, the hidden size of pretrained layer is first reduced via linear projection and passed through its own self-attention layer before transformed back into the original hidden size.

# B Experiment Settings

## B.1 Extraction Summarization

The dataset CNN/DM consists of news articles and multi-sentence highlights as summaries. In our work, we used the non-anonymized version processed by See et al. (2017) while following the standard dataset split that contains 287,226 training examples, 13,368 validation examples and 11,490 test examples [8]. Following previous work (Xu and Durrett, 2019; Zhang et al., 2019; Xu et al., 2020), we create the NYT-50 dataset from the New York Times Annotated Corpus by removing the documents whose summaries are shorter than 50 words, and use the data split that consists of 137,778 training examples, 17,222 validation examples and 17,223 test examples. In both datasets, we use the same data pre-processing steps from previous work (Liu and Lapata, 2019; Xu et al., 2020), and obtain sentence-level oracle labels for extractive summarization by greedily select sentences that maximizes the ROUGE evaluation metric (Nallapati et al., 2017). During training and inference, the documents are truncated to 512 and 800 tokens, respectively, for the CNN/DM and NYT-50 datasets.

During training, we use the ADAM optimizer (Kingma and Ba, 2014) ($\beta_1 = 0.9$, $\beta_2 = 0.999$) following the same learning rate scheduler used in (Liu and Lapata, 2019). We train all our models for a total of $50,000$ steps where the validation loss is evaluated every $1,000$ steps for selecting the top-3 checkpoints. We perform all our experiments on a combination of NVIDIA GTX 1080 Ti and V100 under the single GPU setting, where the true batch size is set to 36 with gradient accumulation per step is set to 9 or 3 for 1080 Ti and V100 respectively due to memory constraints.

## B.2 Topic Segmentation

The dataset WikiSection (Arnold et al., 2019) consists of Wikipedia documents with distinct section and subsection headings indicating topic boundaries. In our work, we use the largest English subset in city domain (*en_city*) consisting of 19.5k documents, and use the same 70/10/20 (train/dev/test) split setting used by the authors[9]. Similarly, Wiki-727 (Koshorek et al., 2018) consists of 727,000 open-domain documents from English Wikipedia, where we use the 80/10/10 (train/dev/test) split setting used by the authors[10]. During training and inference, the context length for the left and right windows are both set to 128 tokens.

During training, we use the AdamW optimizer (Loshchilov and Hutter, 2019) ($\beta_1 = 0.9$, $\beta_2 = 0.999$) following the same learning rate scheduler used in Lukasik et al. (2020). Due to the significant size difference between the two datasets, we trained the model on WikiSection for five epochs and on Wiki-727 for one epoch, where validation process is executed every $2,500$ steps to select the top-3 checkpoints. We perform all our experiments on a combination of NVIDIA GTX 1080 Ti and V100 under the single GPU setting, where the true batch size is set to 64 with gradient accumulation per step is set to 4 or 1 for 1080 Ti and V100 respectively due to memory constraints.

# C Head Importance of Pattern-Injected Models

## C.1 Attention Heads

Similarly, We also visualize the head importance score (Figure 3) using the 6-layer 12-head model on NYT-50, where the first four heads (index 1-4) of each layer are injected with our patterns (matching token, intra-sentence and positional, respectively). From this example, we can see that the heads with patterns injected are considered to be more important across almost all layers, with the most important head being the intra-sentence head in the last layer. This fits our intuition since the output of the last layer is used as the sentence-representation for the classifier.

[8] https://github.com/abisee/cnn-dailymail

[9] https://github.com/seastianarnold/WikiSection

[10] https://github.com/koomri/text-segmentation

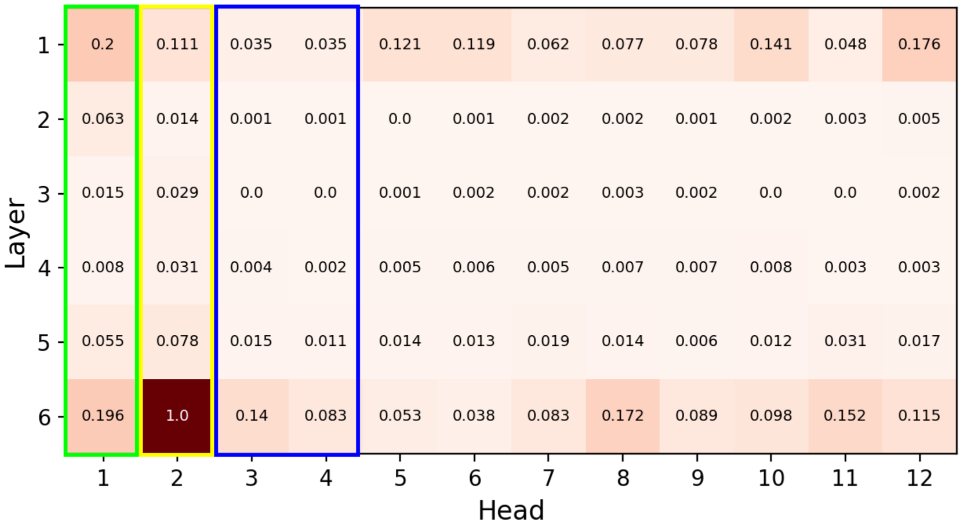


Figure 3: Importance heatmap for the 6-layer 12-head model. Head 1-4 are injected with the patterns, where the highlighted boxes represent Matching Token (Green), Intra Sentence (Olive) and Positional (Blue) (-1, +1).

### C.2 Projected Attention Layer Heads

We visualize the important scores of the PAL heads for BERTSum trained on CNN/DM (Figure 4), where there are four heads added to each BERT layer via residual connection. Figure 4b shows the normalized importance score of the PAL heads without any patterns injected, where the model is opting to use almost entirely the representation from the BERT layers. In Figure 4a, where each of the four PAL heads are injected with our patterns, we can see that importance score significantly increased from the score without the patterns injected, indicating that the features encoded in our patterns are indeed being utilized by the models in addition to the existing pretrained representations.

## D Summarization with Trigram Blocking

| Model | CNN/DM | | | NYT-50 | | |
|---|---|---|---|---|---|---|
| | R-1 | R-2 | R-L | R-1 | R-2 | R-L |
| 6 Layer 8 Head | | | | | | |
| Transformer | 41.07 | 18.41 | 37.45 | 45.13 | 26.05 | 37.52 |
| +Patterns (4/8) | **41.93** | **19.04** | **38.37** | **46.36** | **27.03** | **38.58** |
| 6 Layer 12 Head | | | | | | |
| Transformer | 41.12 | 18.42 | 37.50 | 45.35 | 26.23 | 37.71 |
| +Patterns (4/12) | **42.01** | **19.12** | **38.44** | **46.09** | **26.84** | **38.35** |
| 6 Layer 12 Head (with BERT Embeddings) | | | | | | |
| Transformer | 41.38 | 18.65 | 37.79 | 45.35 | 26.20 | 37.66 |
| +Patterns (4/12) | **42.24** | **19.35** | **38.68** | **46.27** | **27.02** | **38.49** |

Table 4: The results for the summarization experiments under three settings with Trigram Blocking applied.

### D.1 Trigram Blocking

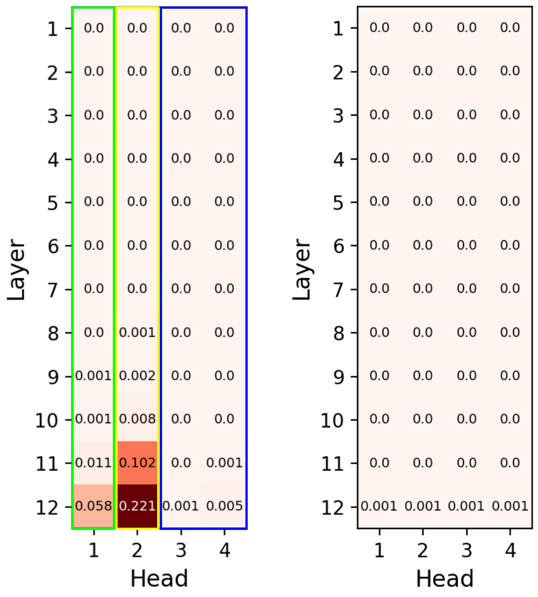
In our experiments, we follow previous work (Paulus et al., 2018; Liu and Lapata, 2019) in evaluating the models in two ways: with and without the trigram blocking. At inference time, the summary is usually formed by selecting sentences with the highest prediction scores. However, with the trigram blocking trick, sentences with overlapping trigram will not be selected. This trick has been shown to be an effective method to deal with redundancy on some dataset (e.g. CNN/DM), but may cause performance drop in others (e.g. Pubmed and arXiv).

| Model | w Tri-block | | |
|---|---|---|---|
| | R-1 | R-2 | R-L |
| Transformer | 41.07 | 18.41 | 37.45 |
| + match (m) | +0.67 | +0.54 | +0.71 |
| + intra (i) | +0.20 | +0.12 | +0.27 |
| + pos (p) | -0.13 | -0.13 | -0.10 |
| + m +i | +0.84 | +0.57 | +0.89 |
| + m +p | +0.46 | +0.38 | +0.52 |
| + i + p | +0.27 | +0.20 | +0.34 |
| + all | **+0.86** | **+0.63** | **+0.92** |

Table 5: Ablation study on the CNN/DM dataset (6-layer 8-head) with Trigram Blocking applied.

### D.2 Pattern-Infused Sparse Transformers

In Table 4, we show the trigram blocking results of the sparse transformer models on both summarization datasets, and Table 5 shows the trigram blocking results for pattern ablation experiment on the CNN/DM dataset. In line with §4.3.2, our pattern-infused models work better than all the other models on all of the settings on both dataset. As for the ablation study, we see a higher performance gain with the matching-token pattern when trigram blocking is applied, where the best performing model is still the one with all patterns applied.



(a) PAL with patterns (b) PAL without patterns

Figure 4: PAL head importance with (a) and without (b) patterns injected, where the highlighted boxes represent Matching Token (Green), Intra Sentence (Olive) and Positional (Blue) (-1, +1)

### D.3 Guided Knowledge Injection into Pre-trained Models

Table 6 shows that the results with trigram blocking. We find the performance gain from the patterns to be higher for CNN/DM and lower for NYT-50.

| Model | CNN/DM | | | NYT-50 | | |
|---|---|---|---|---|---|---|
| | R-1 | R-2 | R-L | R-1 | R-2 | R-L |
| BERTSum | 42.97 | 20.09 | 39.43 | 47.58 | 28.40 | 39.95 |
| + PAL | 42.96 | 20.07 | 39.41 | 47.78 | 28.56 | 40.15 |
| + PAL + Patterns | **43.07** | **20.12** | **39.50** | **48.25** | **29.10** | **40.70** |

Table 6: ROUGE F-scores of pretrained models with PAL when trigram blocking is applied.

## E Pattern Ablation for Topic Segmentation

Table 7 shows that applying all 3 types of patterns leads to the highest performance gain in F-1 score. This is inline with the ablation results for extractive summarization.

| Model | P | R | F-1 |
|---|---|---|---|
| Transformer | 0.698 | 0.647 | 0.671 |
| + match (m) | +0.035 | +0.064 | +0.051 |
| + intra (i) | +0.006 | -0.006 | +0.000 |
| + pos (p) | +0.022 | -0.022 | -0.002 |
| + m +i | +0.023 | **+0.070** | +0.047 |
| + m +p | +0.040 | +0.059 | +0.050 |
| + i + p | +0.030 | -0.030 | -0.004 |
| + all | **+0.046** | +0.064 | **+0.056** |

Table 7: Ablation study results on the WikiSection dataset with the 6-layer 8-head setting.