Human Interpretation and Exploitation of Self-attention Patterns in Transformers: A Case Study in Extractive Summarization

Raymond Li, Wen Xiao, Lanjun Wang, Giuseppe Carenni

1 University of British Columbia, Vancouver, Canada
2 Tianjin University, Tianjin, China
{raymondl, xiaowen3, carenini}@cs.ubc.ca
wanglanjun@tju.edu.cn

Abstract
The transformer multi-head self-attention mechanism has been thoroughly investigated recently. On one hand, researchers are interested in understanding why and how transformers work. On the other hand, they propose new attention augmentation methods to make transformers more accurate, efficient and interpretable. In this paper, we synergize these two lines of research in a human-in-the-loop pipeline to first find important task-specific attention patterns. Then those patterns are applied, not only to the original model, but also to smaller models, as a human-guided knowledge distillation process. The benefits of our pipeline are demonstrated in a case study with the extractive summarization task. After finding three meaningful attention patterns in the popular BERTSum model, experiments indicate that when we inject such patterns, both the original and the smaller model show improvements in performance and arguably interpretability.

1 Introduction

With transformer-based models (Vaswani et al. 2017) dominating the leaderboard for many key NLP tasks like summarization (Liu and Lapata 2019) and sentiment analysis (Adhikari et al. 2019), their core multi-head self-attention mechanism has been thoroughly investigated recently. In particular, to explain why and how transformers work, researchers analyze the learnt self-attention matrices of language models or task-specific models (e.g., Raganato and Tiedemann 2018; Vig and Belinkov 2019), with Voita et al. (2019) for instance, exploring the patterns of attention heads in neural machine translation during pruning. Instead of visualization on trained or fine-tuned models, in a recent work, Li et al. (2021) introduce a visual analytic framework, T3-Vis, to help researchers to better train and fine-tune transformer models, by providing valuable insights about the model’s intrinsic properties and behaviours.

Meanwhile, a parallel line of research has shown that injecting linguistic/positional information into the attention matrices is useful for reducing the size of the model while keeping competitive results, and even improve the performance in some cases. This can be done by either enforcing fixed attention patterns, like Raganato, Scherrer, and Tiedemann (2020) do with positional attention in a machine translator, and Xiao, Huber, and Carenni (2020) with fixed discourse tree attention in a summarizer; or alternatively by guiding the attention weights through more flexible masking strategies, like Yang et al. (2018); Fan et al. (2021) do by using masks to enforce locality within a fixed-window, and others have done to inject discourse information (Mihaylov and Frank 2019), syntactical dependencies (Bai et al. 2021), and world knowledge (Liu et al. 2020). Interestingly, all the above strategies can also be applied to the original model through injecting the patterns into newly added attention heads using techniques such as Projected Attention Layers (PAL) (Stickland and Murray 2019), in which case while the size of the model is not reduced the injected information will be more flexibly and effectively integrated into the original model.

In this paper, we propose and test a novel human-in-the-loop pipeline, that to the best of our knowledge is the first attempt trying to synergize research on analyzing self-attention, along with work on injecting information into attention matrices. Initially, an NLP expert analyzes the attention heads of a transformer model using an interactive visual interface (Li et al. 2021) to identify potentially meaningful and useful patterns/relations. Then, those patterns are evaluated on the validation set to confirm their global relevance. Finally, the patterns found to be useful are applied back to the original model, making the resulting model more accurate and/or more interpretable, because those patterns become inherent and transparent properties of such model. Besides, those patterns can also be applied to models with smaller size, in what can be regarded as a human-guided, interpretable knowledge distillation process. Typically, in knowledge distillation a small or focused model is obtained from a huge or general purpose model by automatically distilling the most useful knowledge, with the resulting model being still however a black box, because it is often unclear what information has been distilled and what has not (Gou et al. 2021). In contrast, in our pipeline, a human can find attention patterns that are useful for the task-specific model to make predictions, and then apply those to the smaller model, improving its performance, with less parameters as well as

* Equal Contribution.
† Corresponding Author. This work has been done when Lanjun was with Huawei Canada Technologies Co. Ltd.

* https://github.com/raymondzmc/T3-Vis

https://github.com/raymondzmc/T3-Vis
better interpretability.

In order to test the feasibility and potential benefits of our approach, we run a case study on the extractive summarization task, in particular using the popular BERTSum model (Liu and Lapata 2019), and we find that: (i) For some of the important heads, the patterns they learn do have practical meaning, either lexical, local or positional. For instance, matching token (i.e. the trend on attending to other tokens with the same id) is an important clue for the summarization model. (ii) By applying the patterns back into the original model through PAL, the resulting model can achieve better task performance and stronger interpretability. (iii) Remarkably, the human-guided knowledge distilled model performs much better than the vanilla transformer (baseline), and can be competitive even with fine-tuned BERT distilled models, which by distilling all aspects of the model have an a priori substantial advantage over our technique that only distils attention patterns.

2 Related Work

2.1 Attention Analysis in Transformers

Head attention matrices have been thoroughly investigated, often with the aid of visualization tools (Vig 2019; Hoover, Strobelt, and Gehrmann 2020), because they reveal how transformers work when applied to specific tasks, with most studies focusing on language models. For instance, Vig and Belinkov (2019) visually explore attention patterns in BERT and GPT-2, analyzing their alignment with syntax. Clark et al. (2019) instead evaluate the attention heads by finding the most attended to tokens with respect to a given one, and recognize that while certain attention heads do specialize to specific dependency relations, no individual attention head captures the entire dependency structure. Furthermore, Kovaleva et al. (2019) categorize BERT’s attention heads based on the patterns (e.g. diagonal, heterogeneous), and find that some heads correspond to core frame-semantic relations. More recently, Zhao and Bethard (2020) look at how BERT’s attention changes after fine-tuning by focusing on tokens within the negation scope, while Guan et al. (2020) categorize attention heads through unsupervised clustering based on aggregated attention patterns.

Other tasks, besides language modelling, have been studied. For example, Raganato and Tiedemann (2018) investigate the linguistic information learned by the transformer encoder when trained for machine translation (MT), and similarly, Voita et al. (2019) characterize the functions of the attention heads in the MT model (positional, syntactic, rare words), and evaluate the importance of those head functions. Finally, the work by Xiao, Huber, and Carenini (2021) shows that transformer-based summarizers learn discourse information implicitly in the attention heads. In addition, Li et al. (2021) propose a visual interface that shows the attention maps as well as the importance score of each head in an interactive way, allowing deeper and specialized exploration on all the attention heads. In this paper, we aim to find task-specific important attention patterns, but differently from previous work which just identify and categorize attention patterns, we propose a pipeline to leverage these patterns in improving models’ performance and interpretability by using the new visual interface, $T^3$-Vis (Li et al. 2021).

2.2 Attention Augmentation

There are two widely used ways to augment attention: fixed and masking. For fixed attention patterns, Raganato, Scherrer, and Tiedemann (2020) use fixed positional patterns in MT models and demonstrate benefits for low-resource scenarios, while Tay et al. (2020) show that even a random attention matrix is competitive with the scaled dot-product attention in standard transformers. Later on, Xiao, Huber, and Carenini (2020) expand on their work by using embedded RST-style discourse trees as fixed attention matrices and show their effectiveness in extractive summarization. For masking attention patterns, Yang et al. (2018) model localness through casting a learnable Gaussian mask; Fan et al. (2021) apply either static or dynamic attention masks to better model local semantic features; Ainslie et al. (2020) and Beltagy, Peters, and Cohan (2020) use attention masks to create sparse attention matrices that improve the scalability on long documents. By comparison, while in all these previous works the applied attention patterns (either fixed or masking) are pre-defined based on prior intuitions, in this paper we propose a strategy to find and assess important attention patterns interactively. More recently, Yin et al. (2021) explore the alignment between human attention and model’s learnt attentions in the machine translation domain, and further propose a guided attention strategy to make them more aligned. However, more task specific and fine-grained annotations are required to have the human attentions. Instead, in our pipeline, we explore the effective interpretable patterns directly from the model, and guide the models with those patterns, which not only better guarantees the effectiveness of the attention patterns, but also allowing researchers to understand how the model works.

2.3 Knowledge Distillation in NLP

Knowledge distillation (KD) (Hinton, Vinyals, and Dean 2015) aims to compress a large teacher model into a smaller student model. This is achieved by training the student network to mimic the behaviors of the teacher model in order to obtain a competitive performance. Specifically, the type of knowledge distilled in KD can be categorized into three types (Gou et al. 2021): response-based (e.g., DistilBert (Sanh et al. 2019)), feature-based (e.g., TinyBERT (Jiao et al. 2020), MobileBERT (Sun et al. 2020), MiniLM (Wang et al. 2020)), and relation-based (e.g., Multi-head Graph Distillation (Seunghyun Lee 2019)). Our approach can be considered as of type relation-based because we exploit the relations between tokens in the attention weights. Importantly, the key difference from existing KD works, is that in our approach knowledge is explicitly extracted from the teacher model by a human, and such knowledge becomes a transparent property of the student model, arguably increasing its interpretability.

2.4 Extractive Summarization

Extractive summarization is the task to pick the most representative sentences as the summary for the given docu-
ment(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), can deliver good performances, but why and how such models work so well still remains an open question. In this paper, we run a case study for our proposed pipeline on the extractive summarization task, exploring the discovery, assessment and application of useful attention patterns in the context of the transformer-based BERTSum model (Liu and Lapata 2019).

3 Proposed Generic Pipeline

In this section, we will briefly describe the proposed pipeline (Figure 1). Specifically, given a trained model (the left part in Figure 1), NLP experts are supposed to first extract important patterns on the visual interface (Li et al. 2021) (middle part in Figure 1) by following these three steps.

Step 1: Estimate the importance scores for all the heads on the validation set, and find important heads that stand out.

Step 2: Recognize special patterns in the important attention heads.

Step 3: Evaluate and validate the pattern to confirm global relevance.

Once the important patterns are identified, there are two common approaches - fixed and masking - to apply them as constraints to the attention matrices in the transformer-based neural models (see §3.2). The pipeline also recommends two scenarios to apply the patterns: the first one is to enhance the original model, while the second one is to train a new model in which the patterns are enforced. Additional potential ways to make use of the patterns are left as future work.

3.1 Extract Patterns from Attention

Step 1: Estimate Attention 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. Motivated by Molchanov et al. (2019); Michel, Levy, and Neubig (2019), we assume the importance of each head to be independent to avoid an NP-hard combinatorial search. In principle, the importance of an attention head \(I(h)\) can be defined as the loss increment on the validation set when removing it during inference.

\[
I(h) = \sum_{(x,y) \in X} \left( L(y|x, H - h) - L(y|x, H) \right)
\]

where \(L(y|x, H)\) is the loss on the sample \(x\) with label \(y\) with all the heads \(H\), while \(L(y|x, H - h)\) is the loss with the heads except \(h\). However, this leave-one-out strategy for computing importance scores, requiring \(|H|\) times inferences (on the validation set), is extremely time consuming and not scalable to large datasets. Thus, following previous works (Michel, Levy, and Neubig 2019; Molchanov et al. 2019), we explore approximate importance score estimation methods, which only require a single forward and backward pass for each example. As different methods are suitable for different tasks, we have considered the following three popular estimation methods.

Head Sensitivity: Proposed by Michel, Levy, and Neubig (2019), a mask variable is applied to the output of each individual attention head, where its gradient sensitivity is used as a proxy score for the importance.

Layer-wise Relevance Propagation (LPR): Designed to compute the contributions of individual pixels for image classification (Bach et al. 2015). LRP is adapted by Voita et al. (2019) for the transformer model to estimate the attention head relevance to the prediction.

Taylor Estimation: Molchanov et al. (2019) propose using the Taylor expansion to estimate the error induced from removing a parameter from the model. In our work, we use the first-order expansion to avoid the overhead from computing the Hessian, where the gradient with respect to validation loss is summed over all parameters of an attention head to estimate its importance.

In the pipeline, NLP experts can select the most proper estimation method for their own task either based on prior knowledge (e.g. Head Sensitivity should be selected in MT following Michel, Levy, and Neubig (2019)), or empirically, by verifying which of the three approximate estimation methods best aligns (e.g. by cosine similarity) with ‘gold head importance scores’ computed by leave-one-out (Eq. 1) on the validation set.

Step 2: Find Attention Patterns

In this step, the human expert should start with the most important heads and visually inspect their attention distributions looking for patterns. Here, we define a pattern very broadly as a predicate \(P\) that can be verified on 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 should satisfy two criteria: 1) occur consistently among relevant to-donations; 2) be interpretable by human experts to be beneficial for the downstream tasks.

For example, in previous works on analyzing the attention heads of pretrained language models (Vig and Belinkov 2019; Kovaleva et al. 2019), some attention heads show a high correlation with position or linguistic properties (like syntactic dependency).

Step 3: Evaluate Attention Patterns

When a specific interesting pattern is uncovered from visualizing the attention heads, the next step is to confirm its global relevance on each head by empirically measuring the proportion of total attentions from the head aligned with the pattern aggregated over data samples. By the evaluation on each single head over the whole validation set, the NLP expert can verify if the pattern generally exists across different data samples, instead of only appearing by chance on certain data that the expert happened to look at.

Specifically, we define the global relevance (GR) of an attention pattern \(P\) for the attention head \(h\) as follows:

\[
\text{GR}(P, h) = \frac{\sum_{x \in X} gr(x, P, h)}{|X|}
\]

\[
gr(x, P, h) = \sum_{i} \sum_{j} \alpha_{i,j} \cdot \mathbf{1}_{P(x_i, x_j)}
\]
where $gr(x, P, h)$ denotes the global relevance of a pattern $P$ for head $h$ on a single data sample $x$, and to validate the generality, $GR(P, h)$ is then computed as the average $gr(x, P, h)$ over the validation set. The attention value from token $x_i$ to $x_j$ on the head $h$, denoted $\alpha_{ij}^h$, is aggregated if and only if $P(x_i, x_j)$ holds. Note that $\sum_i \sum_j |x_i| \alpha_{ij}^h = |x|$ due to the property of attention matrices.

Given a pattern $P$, it will be kept if there exists at least one significantly relevant head. There are several ways to decide whether the relevant head exists based on GR, e.g., setting a threshold. Here, we suggest to use one-tailed one-sample t-test on each head $h^*$ with the null hypothesis as: $GR(P, h^*) < GR(P, h)$, where $GR(P, h)$ is the average of $GR(P, h)$ over $H$. The p-value is usually set as 0.01. If there is at least one head rejecting this null hypothesis, i.e. showing its significantly higher relevance than most other heads, we keep the pattern $P$ for further applications.

### 3.2 Apply Patterns

Once important and interpretable patterns have been identified, they can be injected into a transformer model by either fixing or masking the attention weights prior to the softmax function. Although the two strategies are very similar with respect to what they can achieve, as we will see in the case study they can be more or less appropriate depending on the nature of the pattern that needs to be applied. For fixed attention weights, the attention logits in the scaled-dot-product attention is replaced with a fixed (possibly input dependent) matrix such that

$$\text{FixAttn}(V, X) = \sigma(F(X))V$$  \hspace{1cm} (4)

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 specific pattern. Similarly, a pattern can also be applied by casting a mask over the attention

weights computed from the key and query vectors, such that

$$\text{MaskAttn}(Q, K, V, X) = \sigma(M(X) + QK^T) V$$  \hspace{1cm} (5)

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.

In practice, patterns can be applied in at least two scenarios (i) to enhance the original model on which they are discovered aiming to improve its accuracy and interpretability (ii) to improve smaller models, in what can be regarded as a human-guided distillation process. In the first scenario, although patterns can be directly injected into the pretrained encoder, determining to which heads the patterns should be applied to requires extensive hyperparameter search and risk overfitting. Instead, we opt to inject the patterns via additional attention heads through techniques such as the Projected Attention Layers (Stickland and Murray 2019). As for the second scenario, the patterns are simply applied on the heads (one per head) for each layer, and the new models are trained from scratch.

### 4 Case Study: Extractive Summarization

#### 4.1 Trained Summarization Model

In our case study, we adopt the architecture of the popular BERTSum model (Liu and Lapata 2019). The model first obtains the contextualized sentence representation from the pretrained BERT encoder, and uses a single-layer binary classifier to score the sentences for summary selection. More specifically, a pair of BOS, EOS tokens are inserted before and after each sentence to indicate the segment boundary, and during prediction, the last hidden state of the BOS token of each sentence is used as the sentence representation. We apply our pipeline on the BERTSum model trained

---

2The figures of (a) and (b) in the ‘Extract Patterns’ step are captured from the visual interface (Li et al. 2021).

3We re-implemented the model using the BERT class from hugging face. Results are comparable with original paper with details in Appendix A.
on the CNN/DM dataset (Hermann et al. 2015; See, Liu, and Manning 2017), a widely used summarization dataset and as evaluation metrics we use standard ROUGE scores (Lin 2004).

4.2 Extract Patterns from Attentions

The process of extracting patterns in the extractive summarization case study is shown in Figure 2.

Find Important Heads To identify the most appropriate head importance estimation method for extractive summarization, we evaluate the three proxy scores (Sensitivity, LRP and Taylor) against the leave-one-out head importance score by using the selection process outlined in §3.1. As shown in Table 1, the Taylor Estimation (Molchanov et al. 2019) is the most aligned with leave-one-out in term of cosine similarity and therefore it is used in the the rest of our analysis.

| Method            | Sensitivity | LRP  | Taylor |
|-------------------|-------------|------|--------|
| Cosine Similarity | 0.6         | 0.64 | 0.83   |

Table 1: Cosine similarity between the leave-one-out loss increment and the three estimation methods.

The resulting estimated importance score heatmap of all heads is shown 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, which is inline with the findings in Michel, Levy, and Neubig (2019).

Find and Evaluate Patterns With help from a visual interface, we analyze the attention distributions of the most important heads by looking for human interpretable relationships encoded in the attention weights. From this analysis, three specific types of patterns appear to be present in the most important heads. Those patterns are then evaluated on the validation set to assess their global relevance on each head.

Matching Token (Green in Figure 2) We observe that the attention weights of some important heads appear to exhibit an "attending to matching tokens" pattern. Specifically, the attention value \( \alpha_{i,j}^h \) between input tokens \( x_i \) and \( x_j \) on head \( h \) is high whenever \( x_i = x_j \). For example, as shown in Figure 2 (b), 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. 2), 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 with (See the Green box in Figure 2 (c)). Interestingly, these heads are prominent in the importance heatmap, which suggests the matching token pattern is critical for the summarization task.

Intra-Sentence (Olive in Figure 2) For some heads, the attentions among tokens tend to be localized within the sentence boundaries, as shown on an example in Figure 2 (b).

To evaluate this pattern, GR is computed with \( P(x_i, x_j) \) holding if and only if \( x_i \) and \( x_j \) occurs within the same sentence boundary. This reveals that such pattern appears frequently, especially in the mid to upper layers of the transformer encoder. (See Figure 2 (c))

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

4.3 Apply Patterns to Summarizers

After uncovering potentially important patterns and confirming their relevance, we apply them to transformer-based summarizers through masking and fixing the attention weights. Aiming for more general ‘cross-dataset’ insights, in addition to the CNN/DM dataset, on which we trained the model that we use to extract the patterns, we also evaluate the benefit of the patterns when applied to the NYT-50 dataset (Sandhaus 2008).

Method The patterns identified from our analysis can be applied on 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:

\[
m_{i,j}(m) = \begin{cases} 
1 & (x_i = x_j) \land (\text{freq}(x_i) = 1) \\
0 & \text{otherwise}
\end{cases}
\]  

(6)

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

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

\[
m_{i,j}(s) = \begin{cases} 
1 & \text{SameSent}(x_i, x_j) \\
0 & \text{otherwise}
\end{cases}
\]  

(7)

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

\[
f_{i,j}^{(-1)} = \begin{cases} 
1 & j = i-1 \\
0 & \text{otherwise}
\end{cases}
\]  

(8)

And \( f_{i,j}^{(+1)} \) being the same, but equal to 1 for \( j = i+1 \). We opt for 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

\footnote{The figures of (a) and (b) from Figure 2 are captured from the visual interface (Li et al. 2021).}
Figure 2: Example of Extracting Patterns in the extractive summarization case study. Important heads are found ((a), up left). Then three pattern types are identified ((b), up right): Matching Token, Intra-Sentence and Positional (-1, +1), shown in Green, Olive, and Blue respectively. Finally, each pattern is evaluated with GR on all of the heads ((c), bottom). For better visualization, we only label one head with significantly larger GR in (c) for each pattern.

Table 2: ROUGE F-scores of PAL with pretrained models on corresponding test sets.

| 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 (Ours)   | 42.58 | 20.05 | 39.10 | 48.74 | 29.60 | 41.11 |

Table 2: ROUGE F-scores of PAL with pretrained models on corresponding test sets.

Guided Knowledge Injection into Pre-trained Models
We first experiment with injecting the patterns back into the pre-trained BERTSum summarizer. In particular, we apply them through additional attention heads in the form of a Projected Attention Layer (PAL)(Stickland and Murray 2019), along with the existing parameters of the original model.

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 applied 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 applying the patterns. Results in Table 2 indicate applying the patterns in the PAL (+PAL(Ours)) 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. This suggests that following our pipeline can boost model performance, as well as its interpretability, as the model follows meaningful patterns. Interestingly, visualizing the head importance scores reveals that the PAL heads with patterns applied are significantly more important (by 2 orders of magnitude) than the PAL heads without patterns applied.

Knowledge Distillation for Vanilla Transformers
In a second round of experiments, we apply the three kinds of...
patterns on a simpler and smaller summarizer, namely a non-pretrained 6-layer 8-head transformer architecture (Vaswani et al. 2017) and compare against the vanilla summarizer baseline. Since this can be seen as a form of human-guided distillation, we also compare against two popular SOTA distillation models from recent works: DistilBERT (Sanh et al. 2019) and TinyBERT (Jiao et al. 2020). Specifically, we compare against the 6-layer 12-head variant of these two models. As these models use the embedding-layer from BERT (TinyBERT uses projected embeddings but maintains the same dimensions), we also use the pretrained embeddings for a fair comparison.

Under both settings, each of the four patterns (with 2 as positional patterns) is applied in a separate attention head across all layers in the model. Note that since the goal of these experiments is to assess the benefits of the patterns, we do not perform extensive hyperparameters search when applying these patterns (e.g. on which layer, on how many heads, etc.).

As shown in Table 3, for CNN/DM the pattern-infused models significantly outperform the baseline vanilla summarizer under all three settings (6-8, 6-12, and 6-12 w/ BERT embeddings), and can also beat the two distilled models with the same setting, which convincingly demonstrates the utility of applied patterns. As for the cross-dataset experiments on the NYT-50 dataset, our models significantly improves over the vanilla transformer and DistilBERT, but lags behind TinyBERT in performance. Since TinyBERT also distilled the layer output along with the attention weights and predicted distribution of BERT, it has a substantial advantage over our technique that only distils attention patterns. We suspect that the additional knowledge helped TinyBERT in capturing long term dependencies required by NYT-50.

Overall, with the specific patterns applied, our models are arguably more interpretable than both vanilla transformers and distilled models, as we certainly know the information encoded in each masked/fixed attention heads. Additionally, as was the case for the enhanced pre-trained model, the attention heads expressing the patterns tend to have higher importance scores than the other heads, suggesting that such patterns are effectively leveraged by the model.

To study the contribution of individual patterns, we perform an ablation study by applying all combinations of patterns on CNN/DM using the transformer model with 6 layers and 8 heads. According to Table 4, we observe that applying matching token and intra-sentence together achieves the strongest improvement on the performance among all combinations, only slightly lower than applying all patterns. Meanwhile, the gains from applying them 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 applying only the positional patterns causes the performance to degrade, it works better when combined with the other patterns. The reason why this happens is unclear and further study are left as future work.

**Discussion on Interpretability** Arguably, with human-interpretable patterns enhanced in the model for a specific downstream task, we gain a better intuition about the model’s behaviour and the type of features that are useful for the respective task, i.e. extractive summarization in the case study.

### 5 Conclusion and Future Work

In this paper, we propose a generic human-in-the-loop pipeline, which synergizes 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. To be specific, a human expert analyzes the attention heads of a task-specific model, identify and verify potentially meaningful patterns and finally inject the patterns back into a the original model or in smaller models. By running a case study on the extractive summarization task, we show the potential benefits of our pipeline. We do find meaningful patterns in some important heads, and the relationships encoded in the patterns help us understand the features used by the model when performing summarization. Furthermore, by applying the patterns into the original as well as smaller models, the performance and interpretability improve in both cases.

One very promising direction for future work is to ap-

---

7An illustrative example is shown in Appendix B.2
ply our generic pipeline to other NLP tasks, like machine translation, whose models are known to contain redundancy among the heads (Michel, Levy, and Neubig 2019; Voita et al. 2019). It would also be worth exploring to verify whether the important patterns from one task can be transferable to another task (e.g. extractive summarization to abstractive summarization, or to discourse parsing, etc.), in order to better understand the connections between different tasks. In addition to the apparent patterns that human experts can identify through visualization of the attention, our pipeline can be further expanded to consider deeper linguistic features, i.e. human experts could propose linguistic patterns that may be relevant for a task based on prior knowledge (e.g. discourse may be important for summarization), and then evaluate those patterns, to apply them back to the model once validated.

References

Adhikari, A.; Ram, A.; Tang, R.; and Lin, J. 2019. Docbert: Bert for document classification. arXiv preprint arXiv:1904.08398.

Ainslie, J.; Ontanon, S.; Alberti, C.; Cvicek, V.; Fisher, Z.; Pham, P.; Ravula, A.; Sanghai, S.; Wang, Q.; and Yang, L. 2020. ETC: Encoding Long and Structured Inputs in Transformers. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 268–284. Online: Association for Computational Linguistics.

Bach, S.; Binder, A.; Montavon, G.; Klauschen, F.; Müller, K.-R.; and Samek, W. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one, 10(7): e0130140.

Bai, J.; Wang, Y.; Chen, Y.; Yang, Y.; Bai, J.; Yu, J.; and Tong, Y. 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, 3011–3020. Online: Association for Computational Linguistics.

Belshaw, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The Long-Document Transformer. arXiv:2004.05150.

Clark, K.; Khandelwal, U.; Levy, O.; and Manning, C. D. 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, 276–286. Florence, Italy: Association for Computational Linguistics.

Fan, Z.; Gong, Y.; Liu, D.; Wei, Z.; Wang, S.; Jiao, J.; Duan, N.; Zhang, R.; and Huang, X. 2021. Mask Attention Networks: Rethinking and Strengthen Transformer. arXiv preprint arXiv:2103.13597.

Gou, J.; Yu, B.; Maybank, S. J.; and Tao, D. 2021. Knowledge distillation: A survey. International Journal of Computer Vision, 1–31.

Guan, Y.; Leng, J.; Li, C.; Chen, Q.; and Guo, M. 2020. How Far Does BERT Look At: Distance-based Clustering and Analysis of BERT’s Attention. In Proceedings of the 28th International Conference on Computational Linguistics, 3853–3860. Barcelona, Spain (Online): International Committee on Computational Linguistics.

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching Machines to Read and Comprehend. In Cortes, C.; Lawrence, N.; Lee, D.; Sugiyama, M.; and Garnett, R., eds., Advances in Neural Information Processing Systems, volume 28. Curran Associates, Inc.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. In NIPS Deep Learning and Representation Learning Workshop.

Hoover, B.; Strobl, H.; and Gehrmann, S. 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, 187–196. Online: Association for Computational Linguistics.

Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; and Liu, Q. 2020. TinyBERT: Distilling BERT for Natural Language Understanding. In Findings of the Association for Computational Linguistics: EMNLP 2020, 4163–4174. Online: Association for Computational Linguistics.

Jones, K. S. 1972. A statistical interpretation of term specificity and its application in retrieval. Journal of documentation.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In ICLR (Poster).

Kovaleva, O.; Romanov, A.; Rogers, A.; and Rumshisky, A. 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), 4365–4374. Hong Kong, China: Association for Computational Linguistics.

Li, R.; Xiao, W.; Wang, L.; Jiang, H.; and Carenini, G. 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, 220–230. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In Text Summarization Branches Out, 74–81. Barcelona, Spain: Association for Computational Linguistics.

Liu, W.; Zhou, P.; Zhao, Z.; Wang, Z.; Ju, Q.; Deng, H.; and Wang, P. 2020. K-bert: Enabling language representation with knowledge graph. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2901–2908.

Liu, Y.; and Lapata, M. 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), 3730–3740. Hong Kong, China: Association for Computational Linguistics.
Michel, P.; Levy, O.; and Neubig, G. 2019. Are Sixteen Heads Really Better than One? In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d’Alché-Buc, F.; Fox, E.; and Garnett, R., eds., Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc.

Mihaylov, T.; and Frank, A. 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), 2541–2552. Hong Kong, China: Association for Computational Linguistics.

Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; and Kautz, J. 2019. Importance estimation for neural network pruning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 11264–11272.

Nallapati, R.; Zhai, F.; and Zhou, B. 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.

Paulus, R.; Xiong, C.; and Socher, R. 2018. A Deep Reinforced Model for Abstractive Summarization. In International Conference on Learning Representations.

Raganato, A.; Scherrer, Y.; and Tiedemann, J. 2020. Fixed Encoder Self-Attention Patterns in Transformer-Based Machine Translation. In Findings of the Association for Computational Linguistics: EMNLP 2020, 556–568. Online: Association for Computational Linguistics.

Raganato, A.; and Tiedemann, J. 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, 287–297. Brussels, Belgium: Association for Computational Linguistics.

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

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. ArXiv, abs/1910.01108.

See, A.; Liu, P. J.; and Manning, C. D. 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), 1073–1083. Vancouver, Canada: Association for Computational Linguistics.

Seunghyun Lee, B. C. S. 2019. Graph-based Knowledge Distillation by Multi-head Attention Network. In British Machine Vision Conference (BMVC).

Stickland, A. C.; and Murray, I. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In International Conference on Machine Learning, 5986–5995. PMLR.

Sun, Z.; Yu, H.; Song, X.; Liu, R.; Yang, Y.; and Zhou, D. 2020. MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2158–2170. Online: Association for Computational Linguistics.

Tay, Y.; Bahri, D.; Metzler, D.; Juan, D.-C.; Zhao, Z.; and Zheng, C. 2020. Synthesizer: Rethinking self-attention in transformer models. arXiv preprint arXiv:2005.00743.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc.

Vig, J. 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, 37–42. Florence, Italy: Association for Computational Linguistics.

Vig, J.; and Belinkov, Y. 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, 63–76. Florence, Italy: Association for Computational Linguistics.

Voita, E.; Talbot, D.; Moiseev, F.; Sennrich, R.; and Titov, I. 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, 5797–5808. Florence, Italy: Association for Computational Linguistics.

Wang, W.; Wei, F.; Dong, L.; Bao, H.; Yang, N.; and Zhou, M. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., Advances in Neural Information Processing Systems, volume 33, 5776–5788. Curran Associates, Inc.

Xiao, W.; Huber, P.; and Carenini, G. 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, 124–134. Online: Association for Computational Linguistics.

Xiao, W.; Huber, P.; and Carenini, G. 2021. Predicting Discourse Trees from Transformer-based Neural Summarizers. CoRR, abs/2104.07058.

Xu, J.; and Durrett, G. 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), 3292–3303. Hong Kong, China: Association for Computational Linguistics.

Xu, J.; Gan, Z.; Cheng, Y.; and Liu, J. 2020. Discourse-Aware Neural Extractive Text Summarization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 5021–5031. Online: Association for Computational Linguistics.
Yang, B.; Tu, Z.; Wong, D. F.; Meng, F.; Chao, L. S.; and Zhang, T. 2018. Modeling Localness for Self-Attention Networks. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 4449–4458. Brussels, Belgium: Association for Computational Linguistics.

Yin, K.; Fernandes, P.; Pruthi, D.; Chaudhary, A.; Martins, A. F. T.; and Neubig, G. 2021. Do Context-Aware Translation Models Pay the Right Attention? 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), 788–801. Online: Association for Computational Linguistics.

Zhang, X.; Wei, F.; and Zhou, M. 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, 5059–5069. Florence, Italy: Association for Computational Linguistics.

Zhao, Y.; and Bethard, S. 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, 4729–4747. Online: Association for Computational Linguistics.

Zhong, M.; Liu, P.; Chen, Y.; Wang, D.; Qiu, X.; and Huang, X. 2020. Extractive Summarization as Text Matching. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 6197–6208. Online: Association for Computational Linguistics.

A Experiment Settings

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, Liu, and Manning (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, Wei, and Zhou 2019; Xu et al. 2020), we create the NY 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, Zhai, and Zhou 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 2015) ($\beta_1 = 0.9, \beta_2 = 0.999$) 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 Head Importance of Pattern-Injected Models

B.1 Projected Attention Layers

We visualize the important scores of the PAL heads for BERTSum trained on CNN/DM (Fig. 3), where there are four heads added to each BERT layer via residual connection. Figure 3a shows the normalized importance score of the PAL heads without any patterns applied, where the model is opting to use almost entirely the representation from the BERT layers. In Figure 3b, 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 applied, indicating that the features encoded in our patterns are indeed being utilized by the models in addition to the existing pretrained representations.

B.2 Distilled Model

Similarly, we also visualize the head importance score (Fig. 4) 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 applied 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.

C Summarization with Trigram Blocking

C.1 Trigram Blocking

In our experiments, we follow previous work (Paulus, Xiong, and Socher 2018; Liu and Lapata 2019) in evaluating the models in two ways: with and without the trigram blocking trick applied. Normally at inference time, the summary is formed by selecting sentences with the highest prediction scores, but with the trick applied, sentences containing the same trigrams as already-selected sentences will not be selected. Trigram blocking 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).

C.2 Guided Knowledge Injection into pre-trained Models

In Table 5, we show all the results on both datasets with the Trigram Block trick applied. While the same trend is present as with pattern applied PALs, the model with Trigram Block seems to work better on both datasets.
Figure 3: PAL head importance without (a) and with our patterns (b), such that: Matching Token (Green), Intra Sentence (Olive) and Positional (Blue) (-1, +1).

Figure 4: Head importance heatmap for 6-layer 8-head transformer model, where the first four heads are injected with the patterns: Matching Token (Green), Intra Sentence (Olive) and Positional (Blue) (-1, +1).
Table 5: ROUGE F scores of PAL with pretrained models on corresponding test sets with Trigram Blocking Trick.

| Model          | CNN/DM R-1 | CNN/DM R-2 | CNN/DM R-L | NYT-50 R-1 | NYT-50 R-2 | NYT-50 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 (Ours)   | **43.07**   | **20.12**   | **39.50**   | **48.25**   | **29.10**   | **40.70**   |

Table 6: The overall results on the knowledge distillation experiments with two settings - the setting of the original transformer (Vaswani et al. 2017) (6 layers, 8 heads), and the setting of the distilled models (Sanh et al. 2019; Jiao et al. 2020) (6 layers, 12 heads), with Trigram Blocking applied.

| Model                                | CNN/DM R-1 | CNN/DM R-2 | CNN/DM R-L | NYT-50 R-1 | NYT-50 R-2 | NYT-50 R-L |
|--------------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|
| Transformer 6 Layer 8 Head            | 41.07       | 18.41       | 37.45       | 45.13       | 26.05       | 37.52       |
| Ours 6 Layer 8 Head                  | **41.93**   | **19.04**   | **38.37**   | **46.36**   | **27.03**   | **38.58**   |
| Transformer 6 Layer 12 Head           | 41.12       | 18.42       | 37.50       | 45.35       | 26.23       | 37.71       |
| Ours 6 Layer 12 Head (with BERT Embeddings) | **42.01**   | **19.12**   | **38.44**   | **46.09**   | **26.84**   | **38.35**   |
| Transformer 6 Layer 12 Head (with BERT Embeddings) | 41.38       | 18.65       | 37.79       | 45.35       | 26.20       | 37.66       |
| DistillBERT 6 Layer 12 Head           | 40.84       | 18.22       | 37.23       | 44.77       | 25.77       | 37.21       |
| TinyBERT 6 Layer 12 Head              | 42.16       | **19.35**   | 38.61       | **46.97**   | **27.70**   | **39.24**   |
| Ours 6 Layer 12 Head (with BERT Embeddings) | **42.24**   | **19.35**   | **38.68**   | **46.27**   | **27.02**   | **38.49**   |

C.3 Knowledge Distillation for Vanilla Transformers

In Table 6, we show the experimental results of the knowledge distillation models with trigram blocking applied, and Table 7 shows the ablation results on the CNN/DM dataset with Trigram Blocking. In line with the observations in Sec.4.3, our human-guided knowledge distilled models work better than all the other models on all of the settings on the CNN/DM dataset, while better than all models except TinyBERT on the NYT dataset. As for the ablation study, the matching token pattern brings a higher performance increment compared with the result without Trigram Block, and overall, with all of the patterns applied the performance is the best.

Table 7: Ablation study on the CNN/DM dataset with the basic transformer setting with Trigram Blocking trick.

| Model                  | 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 |