How does GPT-2 Predict Acronyms? Extracting and Understanding a Circuit via Mechanistic Interpretability

Jorge García-Carrasco **Alejandro Maté** **Juan Trujillo**
Lucentia Research, Department of Software and Computing Systems, University of Alicante

Abstract

Transformer-based language models are treated as black-boxes because of their large number of parameters and complex internal interactions, which is a serious safety concern. Mechanistic Interpretability (MI) intends to reverse-engineer neural network behaviors in terms of human-understandable components. In this work, we focus on understanding how GPT-2 Small performs the task of predicting three-letter acronyms. Previous works in the MI field have focused so far on tasks that predict a single token. To the best of our knowledge, this is the first work that tries to mechanistically understand a behavior involving the prediction of multiple consecutive tokens. We discover that the prediction is performed by a circuit composed of 8 attention heads ($\sim 5\%$ of the total heads) which we classified in three groups according to their role. We also demonstrate that these heads concentrate the acronym prediction functionality. In addition, we mechanistically interpret the most relevant heads of the circuit and find out that they use positional information which is propagated via the causal mask mechanism. We expect this work to lay the foundation for understanding more complex behaviors involving multiple-token predictions.

1 INTRODUCTION

Scaling up the size of Language Models based on the Transformer architecture (Vaswani et al., 2017; Brown et al., 2020) has been shown to greatly improve its performance on a wide range of tasks. Because of this, the use of Large Language Models (LLMs) on high-impact fields such as in medicine (Thirunavukarasu et al., 2023; Zhang et al., 2023) is increasingly growing and is expected to keep growing even larger. However, these models are treated as black-boxes due to the fact that they have a large number of parameters and complex internal interactions, hampering our ability to understand its behavior. This is a considerable concern with regards to the safety of Artificial Intelligence (AI) systems, as using a model without knowing its internal heuristics or algorithms can derive in unexpected outcomes such as privacy issues (Li et al., 2023) or harmful behavior (Wei et al., 2023) among others.

Mechanistic Interpretability (MI) aims to tackle this issue by interpreting behaviors in terms of human-understandable algorithms or concepts (Elhage et al., 2021; Olsson et al., 2022; Elhage et al., 2022; Nanda et al., 2023). In other words, it tries to reverse-engineer the large amount of parameters that compose the model into understandable components, which essentially increases the trustworthiness of the model. MI has been successfully applied to explain different tasks on transformer-based models. For example, Wang et al. (2023) discover the circuit responsible for the Indirect Object Identification task (IOI) in GPT-2 Small, composed by 26 attention heads grouped in 7 different classes. Similarly, Hanna et al. (2023) use MI techniques to explain how GPT-2 Small performs the greater-than operation on a single task and test if the discovered circuit generalizes to other contexts. Likewise, Heimersheim and Janiak (2023) discovered how a smaller 4-layer transformer model predicted argument names on a docstring.

In summary, works like the ones mentioned above are proof that mechanistic analysis can be used to shine light into the inner workings of language models. As MI is a young field, the current focus is on developing methods and understanding behaviors on relatively smaller models to lay a solid foundation that will be used to interpret increasingly larger models. In fact, preliminary studies have already appeared discussing

Proceedings of the 27$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

whether current MI techniques are scalable to larger models, with optimistic results (Lieberum et al., 2023).

In this work, we contribute to the growing body of MI works by focusing on understanding how GPT-2 Small performs the task of predicting three-letter acronyms (e.g. `"The Chief Executive Officer"` → `"CEO"`). We have mainly chosen this task because it consists on predicting three consecutive tokens, in contrast to previous existing work which focused on single-token prediction. To the best of our knowledge, this is the first work that applies MI to understand a behavior involving the prediction of multiple tokens. Hence, we expect that the work presented here serves as a starting point for understanding more complex behaviors that involve predicting multiple tokens.

More specifically, we will adopt a *circuits* perspective (Elhage et al., 2021; Olah et al., 2020) and identify the components of the model that are responsible for the behavior under study via a series of systematic *activation patching* (Meng et al., 2022) experiments. Our contributions can be summarized as follows:

- We discover the circuit responsible for three-letter acronym prediction on GPT-2 Small. The circuit is composed by 8 attention heads ($\sim 5\%$ of GPT-2's heads) which we classified on three groups according to their role.
- We evaluate the circuit by ablating the rest of components of the model and show that the performance is preserved and even slightly improved when isolating the discovered 8-head circuit.
- We interpret the main components of the circuit, which we term *letter mover heads* by reverse-engineering their parameters.
- We also found that *letter mover heads* make use of positional information, mainly derived from the attention probabilities due to the causal mask mechanism instead of the positional embeddings.

The remainder of this paper is structured as follows. In Section 2, the required background and the problem statement are presented. Section 3 describes the procedure used to discover the circuit responsible for three-letter acronym prediction as well as the role of each component, followed by an evaluation of the circuit. Section 4 delves into mechanistically interpreting *letter mover heads* as well as studying how these heads use positional information. Finally, the conclusions about the work are presented in Section 5.

2 BACKGROUND

In this section we briefly present the transformer and the used notation, the task of study and how to evaluate the performance of the model on that task.

2.1 Model and Notation

GPT-2 Small (Radford et al., 2019) is a 117M parameter decoder-only transformer architecture composed by 12 transformer blocks containing 12 attention heads followed by an MLP, each component preceded by Layer Normalization (Ba et al., 2016). The input to the model is a sequence of N consecutive tokens which are embedded into $x_0 \in \mathbb{R}^{N\times d}$ via a learned embedding matrix $W_E \in \mathbb{R}^{V\times d}$, where V is the size of the vocabulary. Similarly, positional embeddings are added to x_0.

Following the notation presented in Elhage et al. (2021), x_0 is the initial value of the *residual stream*, where all the components of the model read from and write to. Specifically, if h_{ij} denotes the jth attention head at layer i, the ith attention layer will update the residual stream as $x_{i+1} = x_i + \sum_j h_{ij}(x_i)$ (omitting layer normalization). Each attention head is parameterized by the matrices $W_Q^{ij}, W_K^{ij}, W_V^{ij} \in \mathbb{R}^{d\times d/H}$ and $W_O^{ij} \in \mathbb{R}^{d/H\times d}$, where H is the number of heads in a single layer, which can be arranged into the QK and OV matrices $W_{QK}^{ij} = W_Q^{ij}W_K^{ij}$, $W_{OV}^{ij} = W_V^{ij}W_O^{ij}$. The QK matrix contains information about which tokens the head attends to, whereas the OV matrix is related to what the head writes into the residual stream.

Finally, the resulting vector is unembedded via a unembedding matrix, which in the case of GPT-2 is tied to the embedding matrix (i.e. $W_U = W_E^T$) to obtain a vector $y \in \mathbb{R}^{N\times V}$ where y_{ij} represents the logits of the jth token of the vocabulary for the prediction following the ith token of the sequence.

2.2 Task Description

We will focus on the task of predicting three-letter acronyms. To evaluate whether GPT-2 is able to properly perform this task or not, we curated a dataset of 800 acronyms. It is important to remark that this dataset will **not** be used to re-train the model, but to perform experiments and identify the underlying circuit associated to the task of study. In other words, our aim is to detect a circuit responsible for a concrete task on an LLM that has already been trained in a general, self-supervised way. Hence, in order to isolate the behavior of study and reduce the amount of noise, we made each acronym to meet the following characteristics:

- Each word must be composed by two tokens, the first being only composed by the capital letter and its preceding space (e.g. `"| C|ane|"`)
- The acronym must be tokenized by exactly three tokens, each for one letter of the acronym (e.g. `"|C|K|L|"`)

In order to build the dataset, we took a public list of the most frequently-used common nouns in English (Quintans, 2023) containing a total of 6775 nouns. However, building the dataset is not as easy as choosing three random words and tokenizing them according to our imposed characteristics: words have to be tokenized as GPT-2 naturally expects to stay in-distribution. GPT-2 uses byte-pair encoding (BPE) tokenization (Sennrich et al., 2016), a technique that tokenizes according to the most frequent substring. This means that common substrings/words such as `"ABC"` or `" Name"` are encoded as a single token, hence reducing the amount of possible nouns and acronyms to use on our dataset. Taking this into account, the building procedure was the following:

1. **Nouns:** We took the list of 6775 nouns and filtered out the words that did not meet the characteristics (i.e. each word of the acronym must be composed by two tokens, the first being only composed by the capital letter and its preceding space), leaving us with 381 nouns.
2. **Acronyms:** We tokenized the $P^R(26,3) = 26^3 = 17576$ possible 3-letter acronyms and checked which were naturally tokenized as three separate tokens, which reduced the amount of possible acronyms to 2740. As common nouns beginning with the letter `X`, `Q` or `U` are rare, we also excluded acronyms containing that letter, resulting in a total of 1154 possible combinations.
3. **Dataset:** Finally, we built the dataset by (i) sampling one of the 152 possible acronyms (e.g. `WVZ`) and randomly sampling three of the 381 nouns, one for every letter of the acronym (e.g. `Wreck`, `Vibe` and `Zipper`). Notice that we can build much more than 800 samples in this way, but we chose this size because of computational constraints. As a reference, Hanna et al. (2023) curated a dataset of 490 datapoints to properly identify a circuit.

In summary, this results in a dataset composed by prompts with the structure:

`"|The|C1|T1|C2|T2|C3|T3| (|A1|A2|A3|"`

where `Ci` is the token encoding the capital letter of the ith word (together with its preceding space), `Ti` is the remainder of the word, and `Ai` is the ith letter of the acronym. Therefore, the task consists on predicting `A1`, `A2` and `A3` given the previous context.

The reason for choosing a list of nouns was because (i) it is the common way to build acronyms and (ii) the type of word (noun, adjective, etc.) does not affect the result obtained. We can confirm these results since we have also experimented with synthetic words by taking a random token `"| A|"` where `A` can be any capital letter, followed by one to three random tokens containing just lowercase letters and found the same results that will be presented in this work.

Also, one important concern is that the model could have memorized popular acronyms (e.g. `The Central Processing Unit (CPU)`). However, the acronyms built with our procedure are rare (e.g. `The Wreck Vibe Zipper (WVZ)`). We took this decision to ensure that the model has not been trained with these samples, implying that the discovered circuit generalizes to samples outside of the training dataset and does not just memorize common acronyms.

2.3 Evaluation

In order to quantitatively evaluate the ability of GPT-2 on the task under study, we will compute the *logit difference* between the correct letter and the incorrect letter with the highest logits, for each of the three letters of the acronym:

$$\text{logit\_diff}_i = \text{logits}_{a_i} - \max_{l \in \mathcal{L} \setminus \{a_i\}} \text{logits}_l \qquad (1)$$

where a_i is the correct prediction for the ith letter of the acronym and $\mathcal{L}$ is the set of possible predictions, which in the case of acronym prediction, is the set of capital letters. GPT-2 has an average logit difference across every letter and sentence of the dataset of 2.22, which translates to an average $\sim 90.2\%$ probability difference. Overall, this result provides quantitative evidence supporting that GPT-2 is indeed able to perform three-letter acronym prediction.

3 A CIRCUIT FOR 3-LETTER ACRONYM PREDICTION

Now that the task has been clearly defined and checked that GPT-2 is indeed able to perform it, we will discover the circuit responsible for this behavior, evaluate it and understand the components that compose such circuit. The following experiments were performed by using both *PyTorch* (Paszke et al., 2019) and *TransformerLens* (Nanda and Bloom, 2022) with a 40GB A100 GPU. A repository containing the code

required to reproduce the experiments and figures can be found in `https://github.com/jgcarrasco/acronyms_paper`.

3.1 Discovering the circuit

In order to discover which components form the circuit responsible for three-letter acronym prediction, we will perform a systematic series of *activation patching* experiments, first presented in Meng et al. (2022). The idea of activation patching is to patch (i.e. replace) the activations of a given component with the activations obtained by running the model on a *corrupted prompt*. If the metric degrades when patching a component, it means that it is relevant to the task under study, therefore enabling us to locate the circuit.

In this case, we have carefully performed activation patching with three different types of corruption. For each of the ith letter prediction, we (i) randomly resample the ith word, (ii) randomly resample the words previous to the ith word and (iii) randomly resample the acronym letters previous to the ith letter. This will allow us to better track the flow of information and the role of each component. Table 1 shows the different types of corruption for the prediction of the third letter on a sample prompt. We will perform the corresponding activation patching experiments for each of the three letters on parallel.

Table 1: Example of prompt corruption for the third letter prediction $i = 3$ (this is also performed for $i = 1, 2$)

Corruption Type	Prompt
Original	`The Cane Knee Lender (CK`
Current Word	`The Cane Knee Tandem (CK`
Previous Words	`The Ego Icy Lender (CK`
Previous Letters	`The Cane Knee Lender (BJ`
All corruptions	`The Ego Icy Tandem (BJ`

3.1.1 Corrupting the Current Word

Fig. 1 shows the change in logit difference when patching the residual stream before each layer at every position, for each of the three letters of the acronym to predict. If patching the residual stream at a given position and layer considerably degrades the performance, then it implies that the activations stored at that specific step are important for the acronym prediction task. In this case, we can see that patching `Ci` at earlier layers does indeed drastically degrade the performance, with the logit difference dropping up to -5 units. We can also notice a shift from `Ci` to `A(i-1)` beginning at layer 8. This implies that there are some components that read information about `Ci`, move it to `A(i-1)` and then use it to perform the prediction of the next letter `Ai`.

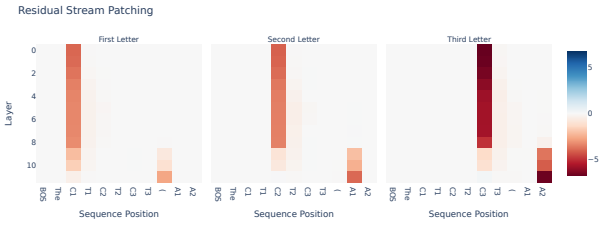

Figure 1: Patching the residual stream at every position and before every layer (corrupting the current word).

Once that we have tracked the flow of information, we can have a more fine-grained view by patching at the level of individual components, i.e. attention heads or MLPs. We performed activation patching experiments on MLPs and found that they were not relevant for acronym prediction. This was expected, as this task mostly requires moving information between token positions, which can only be performed by attention heads. Fig. 2 shows the result of patching the output of attention heads with the activations obtained by corrupting the current word. We are able to localize four heads that are relevant across the three predictions: `8.11`, `10.10`, `9.9` and `11.4`. It is also interesting to notice that the drop on performance is larger on the last letter than on the first. This is due to the fact that the model has more context (i.e. the two previous letters of the acronym) when predicting the last letter, which translates to the model being more confident on its prediction, which corresponds to a larger drop when patching.

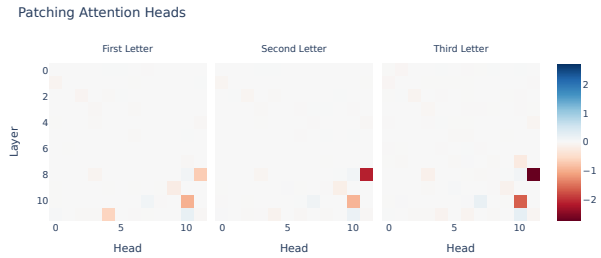

Figure 2: Patching the output of attention heads for every iteration (corrupting the current word).

Fig. 3 shows the distribution of the average attention paid from `A(i-1)` to the previous token positions for head `8.11`. It can clearly be seen that it mostly

attends from A(i-1) to Ci, strongly suggesting that these heads copy the information of the corresponding letter and use it to perform the prediction of the next letter of the acronym, so we term this heads as *letter mover heads*. The behavior of these heads will be extensively discussed on Section 4, after performing the remaining activation patching experiments. The attention patterns for the rest of letter mover heads can be seen in the Supplementary Materials.

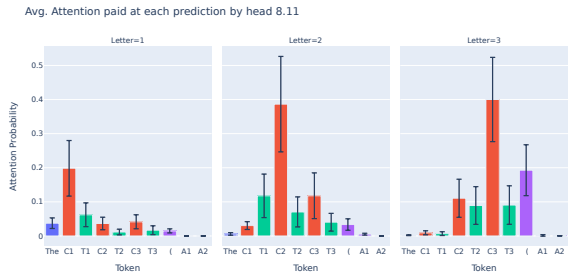

Figure 3: Average probability paid from A(i-1) to the previous token positions for head 8.11.

3.1.2 Corrupting the Previous Words

Fig. 4 shows the results of performing activation patching on the residual stream by corrupting the previous words. As expected, there is no effect on the prediction of the first letter because there are no previous words to corrupt. However, on the remaining letters, patching C(i-1) at earlier layers has a significant (although quite smaller than the previous) effect on predicting Ai, indicating that the circuit uses information about the previous words to perform the task. Concretely, it seems that the information is moved from C(i-1) to Ci on layers 1-2 and from there to A(i-1) at layer 5 and below. Interestingly, patching Ti around layers 5-11 slightly improves the performance of the circuit. We hypothesize that patching only this position may cause the model to become less confident on previous letters, essentially increasing the logit difference. As it is not the focus of the paper and the effect is minimal, we will not delve deeper into this fact.

In order to check which attention heads were responsible for this movement of information, we patched the output of attention heads at positions Ci and A(i-1). Fig. 5 shows that there are a diffuse set of attention heads responsible for moving information from C(i-1) to Ci, such as 4.11, 1.0 and 2.2. Further inspection of the attention patterns show that they are *previous token heads* (or fuzzy versions of it), i.e. heads that attend to the previous token position w.r.t. the current token and move information. Visualizations of the attention patterns of these heads can be found on the

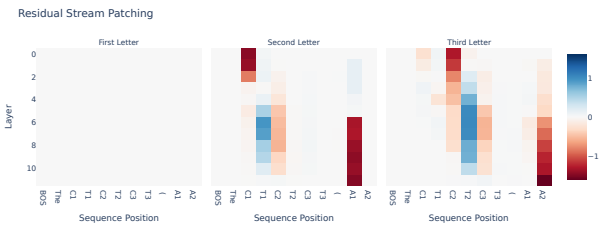

Figure 4: Patching the residual stream (corrupting previous words).

Supplementary Materials.

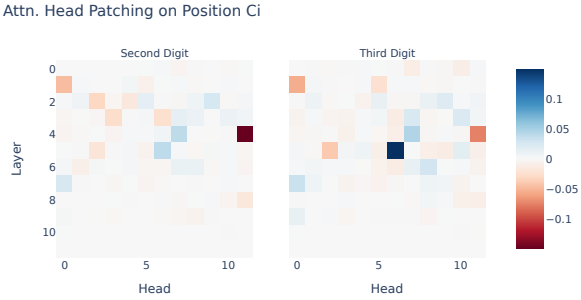

Figure 5: Patching the output of attention heads for every iteration at position Ci (corrupting the previous words).

On the other hand, Fig. 6 shows that heads 5.8, 8.11 and 10.10 are the most relevant in this patching experiment. Further inspection of the attention patterns reveals that they mostly attend to the T(i-1) and Ci tokens and move the information that was propagated to these positions via the previous token heads.

There are a few important aspects to remark from this patching experiment. The first is that the performance drop when patching individual components is significantly lower than on the previous experiment. Secondly, the computation is more diffuse, i.e. it is distributed across many components, specially when looking at the Ci position. As we will see in the next section, this is due to the fact that the model is able to obtain the exact same information (i.e. the capital letter of the previous word) by just attending to the previous predicted letter, which is considerably easier. Another interesting fact is that some letter mover heads are also present in this part of the computation, i.e. some heads have multiple roles or behaviors, which is a motif that has also been discovered on other works Heimersheim and Janiak (2023).

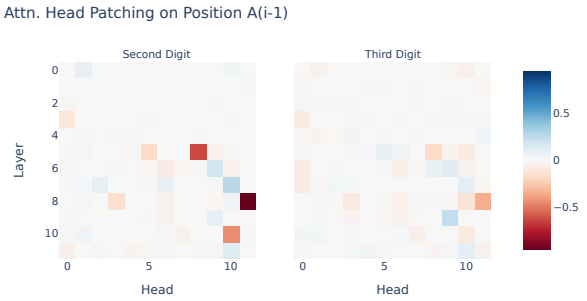

Figure 6: Patching the output of attention heads for every iteration at position `A(i-1)` (corrupting the previous words).

3.1.3 Corrupting Previous Predicted Letters

The results presented in Fig. 7 clearly show that the model uses information about the previous predicted letter to predict the next one, as patching the `A(i-1)` position causes a considerable performance drop (larger than the previous corruption method) across every layer. This provides even more evidence in favor of the previously presented hypothesis that letter mover heads obtain the same information via two paths: from `C(i-1)` via the combination of previous token heads and heads that move information to `Ci` and then to `A(i-1)`, and directly from `A(i-1)`.

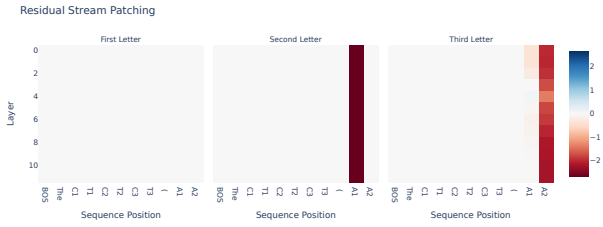

Figure 7: Patching the residual stream at every position and before every layer (corrupting the previous predicted letters).

To summarize, we have been able to discover the following circuit via a series of activation patching experiments:

- Heads `8.11`, `10.10`, `9.9` and `11.4`, termed Letter Mover Heads, attend mostly to the `Ci` token position from the `A(i-1)` token position and are the main responsible for acronym prediction on GPT-2.
- Letter Mover Heads use the previous predicted letter to attend to the correct token position and predict the next letter of the acronym.
- This information, although more faintly, is also propagated from `C(i-1)` to `Ci` via a set of fuzzy previous heads such as `4.11`, `1.0` and `2.2`, which is then moved from `Ci` to `A(i-1)` via heads `5.8`, `8.11` and `10.10`.

3.2 Circuit Evaluation

Now that we have defined a circuit, it is necessary to evaluate whether it is sufficient to effectively perform acronym prediction. In order to evaluate it, we will ablate every other component that is not part of the circuit. Specifically, we will perform mean-ablation, which consists on replacing the activation of a component with the mean activation obtained across all samples of the dataset. In this way, we only discard the information related to the task of study while keeping the rest. Fig. 8 shows the logit difference obtained when progressively adding heads to the circuit. We start with an empty circuit (i.e. ablating every head) to check that the model is unable to perform the task, obtaining negative values of the logit difference, as expected. Then, progressively adding Letter Mover Heads greatly improves performance, the most significant increase being on the third letter prediction, where adding just head `8.11` increases the average logit difference from -1 to 2 approximately. The logit difference keeps increasing by progressively adding the rest of components until we reach the baseline performance with just the 8 discovered heads.

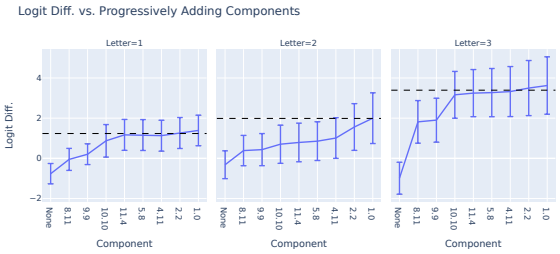

Figure 8: Logit Difference obtained by ablating everything and progressively adding components to the circuit. The dashed horizontal line represents the logit difference obtained with the complete model.

4 UNDERSTANDING LETTER MOVER HEADS

Now that we have discovered and evaluated the main circuit responsible for the task of three-letter acronym prediction on GPT-2, we will provide further evidence on how Letter Mover Heads work, which are the main components of the circuit.

4.1 What do Letter Mover Heads Copy?

We discovered that Letter Mover Heads mostly attend from `A(i-1)` to `Ci` and were the main responsible for the acronym prediction task. Because of this, we hypothesize that these heads directly increase the logits of the correct letter to predict. In order to give evidence about this, we will take a look at the weights of Letter Mover Heads and try to reverse-engineer their behavior.

Specifically, we will inspect the full OV circuit, obtained by retrieving the embeddings corresponding to the capital letter tokens and the capital letter tokens preceded by a space, passing them through the OV circuit of a Letter Mover Head and unembedding the resulting vector. This essentially tells us what would the head write into the residual stream if it attended perfectly to that token. Fig. 9 shows the full OV circuit for Letter Mover Head `8.11`, rearranging it in four different ways to check what it writes when fully attending to capital letters, with or without a preceding space. At first sight, one cannot draw any conclusion except that there is a slight diagonal when attending to capital letters preceded with a space (two rightmost plots.

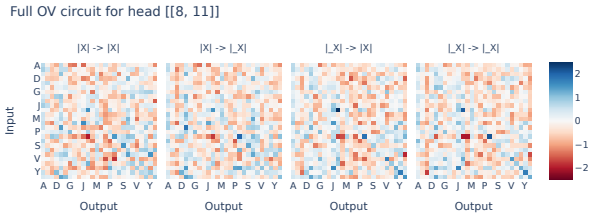

Figure 9: Full OV circuit for head `8.11`, for all capital letter tokens with/without a preceding space.

However, the pattern becomes much more clear when we plot the full OV circuit taking into account all Letter Mover Heads. As it can be seen in Fig. 10, there is now a clear diagonal pattern on the two rightmost plots. In other words, this implies that when Letter Mover Heads attend to a capital letter preceded with a space (which is exactly what `Ci` are), they translate it to the token corresponding to the same capital letter without a space (i.e. `Ai`) and write it into the residual stream. It is important to remark that during analysis we did not use any information from the dataset, i.e. it was purely performed by inspecting the weights.

We also studied the copying behavior by analyzing the relationship between the attention paid to `Ci` and the increase of the logits of `Ai` for different heads. Specifically, it can be seen on Fig. 11 that it pays more

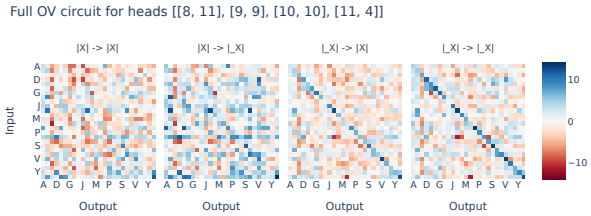

Figure 10: Sum of every Letter Mover Head OV circuit, for all capital letter tokens with/without a preceding space.

attention to `Ci` when predicting the ith digit, and that the value written along the logits of `Ai` increases with such attention.

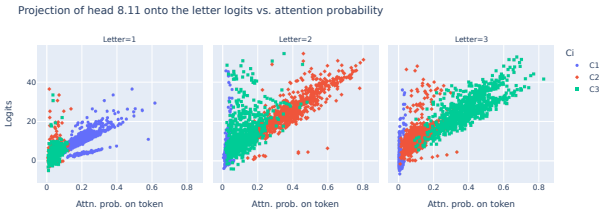

Figure 11: Projection of the output of head 8.11 along the logits correct letter vs. the attention probability paid to `Ci`.

4.2 Positional Information Experiments

We also hypothesized that Letter Mover Heads should use positional information to perform the final prediction (i.e. to attend to the first token of the *first/second/third* word), specially when predicting the first letter of the acronym, as there is no available information regarding the previously predicted letters of the acronym. In order to test this hypothesis, we first study the positional embeddings, as it is the most evident source of positional information of the model. Specifically, we swapped the positional embeddings of different pairs of `Ci` and checked if it had an effect on the attention pattern of Letter Mover Heads. Ideally, if a head relied in positional embeddings, swapping them should force them to attend a different letter. Fig. 12 shows the effect of swapping the positional embeddings of `C1` and `C3` on the attention probabilities of head `8.11`.

As it can be seen, the change in attention probabilities is negligible. We performed all the possible swapping combinations for every letter mover head and found similar results, implying that positional embeddings are not the main source of positional information for

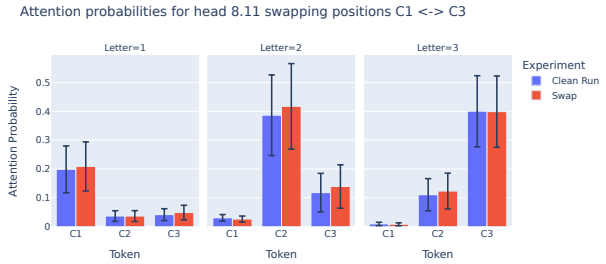

Figure 12: Attention probabilities on head `8.11` when swapping the positional embeddings of `C1` and `C3`.

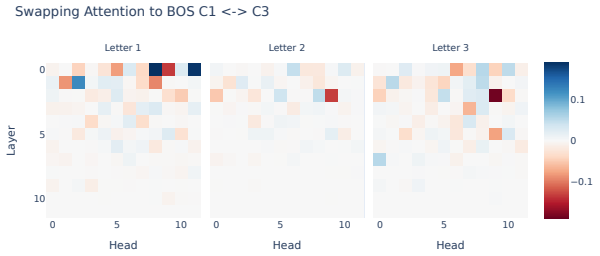

Figure 13: Change in logit difference obtained by swapping the attention paid to `BOS` from the `C1` and `C3` for every head in the model.

Letter Mover Heads. However, the model has to use some source of positional information to predict the first letter, so we looked for another possible source. It has been recently hypothesized (Heimersheim and Janiak, 2023) that models are able to derive positional information from attention probabilities. Specially, due to causal masking, the attention pattern paid to the Beggining of Sequence (`BOS`) token position generally decreases with the destination token position. Therefore, the model could infer the position of a certain token `Ci` by looking at the attention paid to the `BOS` token: a lower attention paid to this token position implies that the destination token is further from the start of the sentence, and vice versa.

Therefore, we patched the activations of each head by swapping their attention paid to the `BOS` token from tokens `Ci` and `Cj` for all possible combinations and measured the change in logit difference. Fig. 13 shows the results for $i = 1$, $j = 3$. Indeed, swapping the attention values does have an impact on the performance, meaning that letter mover heads are likely to use positional information derived from this mechanism, specially when predicting the first letter. There are also other heads that contribute positively. We hypothesize that these heads are writing on the opposite direction to avoid the model becoming overconfident (similar to negative name mover heads on Wang et al. (2023)). However, we leave this aspect as part of a future study, as this requires an extensive analysis.

In order to provide further evidence, we swapped the `BOS` attentions for those heads that had a negative impact of at least 1% in the previous experiment and visualized the change of attention pattern on letter mover heads. Specifically, Fig. 14 shows the attention probabilities paid to the `Ci` tokens on head `8.11` on the clean run, swapping the positional embeddings, swapping the `BOS` tokens and applying both swapping techniques. In general, swapping the `BOS` tokens has the most impact across all predictions, meaning that head `8.11` does indeed use positional information. As expected, the greatest difference can be found on predicting the first letter: swapping the positional embeddings and the `BOS` tokens of `C1` and `C3` changes the average prediction from `A1` to `A3`. It is also important to remark that, most of the change on the attention pattern, is caused by simply swapping two scalars on each of the patched heads, i.e. a slight change in the attention pattern of the patched heads causes a large impact on the attention probabilities of head `8.11`.

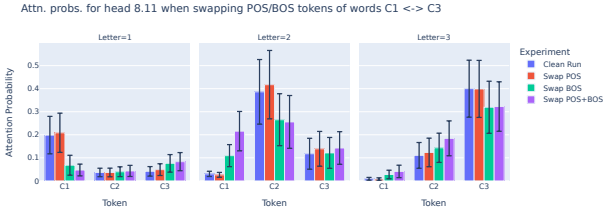

Figure 14: Change in attention when performing the `BOS` attention swapping experiment.

We also performed this experiment with every possible swapping combination and found two important aspects. First, the attention probabilities are mainly affected on the first letter prediction, aligned with our hypothesis that the model relies mainly on positional information (i.e. it has to look for the *first* capital letter). Comparatively, the second and third letter prediction rely on the previous predicted letter/s to gain some context. Second, we found that only swapping `C1` and `C3` had a considerable impact, probably due to the fact that the other two possible swaps are performed between tokens that are closer together, hence the degree of corruption is smaller. This phenomena also occured on the rest of letter mover heads. The experiment involving the rest of letter mover heads and swapping combinations are presented in the Supplementary Materials.

5 CONCLUSIONS

In this work, we identified the circuit responsible for the task of predicting three-letter acronyms on GPT-2 Small via a series of activation patching experiments. The discovered circuit was composed by 8 attention heads which we classified into three different groups according to their role. We showed that ablating every other head did preserve the performance, meaning that the task of acronym prediction does indeed rely on the discovered circuit. We also paid special attention to the most important heads of the circuit, which we termed *letter mover heads*, whose role is to attend to the capital letter of the ith word and copy its content for the ith letter prediction. We provided evidence of this behavior by studying their attention patterns, OV matrices and output activations. We also show that these heads use positional information and that this information is received not only by the positional embeddings, but from the attention patterns. Our experiments show that the positional information is derived from the attention paid to the `BOS` token, in accordance to what it was discovered in simpler models (Heimersheim and Janiak, 2023).

In summary, this is the first work that tries to mechanistically interpret a task involving multiple consecutive token using MI, laying the foundation for understanding more complex behaviors. Moreover, we strongly believe that MI will enable us to understand larger models, increasing the safety and trustworthiness of AI systems.

Acknowledgements

This work has been co-funded by the BALLADEER (PROMETEO/2021/088) project, a Big Data analytical platform for the diagnosis and treatment of Attention Deficit Hyperactivity Disorder (ADHD) featuring extended reality, funded by the *Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital (Generalitat Valenciana)* and the AETHER-UA project (PID2020-112540RB-C43), a smart data holistic approach for context-aware data analytics: smarter machine learning for business modelling and analytics, funded by the *Spanish Ministry of Science and Innovation*. Jorge García-Carrasco holds a predoctoral contract (CIACIF/2021/454) granted by the *Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital (Generalitat Valenciana)*.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. `https://transformer-circuits.pub/2021/framework/index.html`.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. `https://transformer-circuits.pub/2022/toy_model/index.html`.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023. `https://openreview.net/forum?id=p4PckNQR8k`.

Stefan Heimersheim and Jett Janiak. A circuit for Python docstrings in a 4-layer attention-only transformer. `https://www.alignmentforum.org/posts/u6KXXmKFbXfWzoAXn/a-circuit-for-python-docstrings-in-a-4-layer-attention-only`, 2023.

Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. Multi-step jailbreaking privacy attacks on chatGPT. In *The 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. `https://openreview.net/forum?id=ls4Pfsl2jZ`.

Tom Lieberum, Matthew Rahtz, János Kramár, Geoffrey Irving, Rohin Shah, and Vladimir Mikulik. Does circuit analysis interpretability scale? Evidence from multiple choice capabilities in Chinchilla. *arXiv preprint arXiv:2307.09458*, 2023.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. https://openreview.net/forum?id=-h6WAS6eE4.

Neel Nanda and Joseph Bloom. TransformerLens, 2022. https://github.com/neelnanda-io/TransformerLens.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. https://openreview.net/forum?id=9XFSbDPmdW.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Desi Quintans. The Great Noun List. https://www.desiquintans.com/nounlist, 2023.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature Medicine*, 29(8):1930–1940, Aug 2023. ISSN 1546-170X. doi: 10.1038/s41591-023-02448-8. URL https://doi.org/10.1038/s41591-023-02448-8.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023. URL https://openreview.net/forum?id=jA235JGM09.

Zhiyue Zhang, Hongyuan Mei, and Yanxun Xu. Continuous-time decision transformer for healthcare applications. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 206 of *Proceedings of Machine Learning Research*, pages 6245–6262. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/zhang23i.html.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes/No/Not Applicable] Yes, at Section 2.

(b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes/No/Not Applicable] Not applicable. The focus of this work is not on any algorithm but on discovering a circuit by using an already existing technique. However, we specify the sample size used on Section 3.

(c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes/No/Not Applicable] No, but it will be made public upon acceptance.

2. For any theoretical claim, check if you include:

(a) Statements of the full set of assumptions of all theoretical results. [Yes/No/Not Applicable] Not applicable. The work presented here is mainly empirical evidence.

(b) Complete proofs of all theoretical results. [Yes/No/Not Applicable] Not applicable.

(c) Clear explanations of any assumptions. [Yes/No/Not Applicable] Not applicable.

3. For all figures and tables that present empirical results, check if you include:

(a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes/No/Not Applicable] No, but it will be made publicly available upon acceptance.

(b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes/No/Not Applicable] Not applicable, as no training is performed. We study a pretrained model.

(c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes/No/Not Applicable] Yes

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes/No/Not Applicable] Yes, on the start of Section 3.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

(a) Citations of the creator If your work uses existing assets. [Yes/No/Not Applicable] Yes

(b) The license information of the assets, if applicable. [Yes/No/Not Applicable] Not applicable

(c) New assets either in the supplemental material or as a URL, if applicable. [Yes/No/Not Applicable] Yes

(d) Information about consent from data providers/curators. [Yes/No/Not Applicable] Not applicable

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes/No/Not Applicable] Not applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Yes/No/Not Applicable] Not applicable

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes/No/Not Applicable] Not applicable

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Yes/No/Not Applicable] Not applicable

A ATTENTION PATTERNS

This section contains additional attention pattern visualizations to support the findings described in Section 3. Fig. 15 shows the mean attention patterns for heads `1.0`, `2.2` and `4.11`, which were the main responsible of moving information from previous words. As it can be seen, these heads have the characteristic offset diagonal pattern of previous token heads, meaning that these heads attend to the previous token w.r.t. the current one and copy their information. On the other hand, heads `5.8`, `8.11` and `10.10` move the previous information into `A(i-1)` by attending to the tokens `T(i-1)` and `Ci`, as it can be seen on Figs. 16-17.

The attention probability histograms of the rest of letter mover heads `10.10`, `9.9` and `11.4` are shown on Figs. 17-19. Even though the histograms are noisier than the one associated to the main letter mover head `8.11`, it can clearly be seen that these heads generally pay more attention to the proper capital letter `Ci` compared with the other capital letters. We also see a high attention paid to `T(i-1)`, in particular on heads `9.9` and `10.10`. This is likely due to the fact that these heads (specially `10.10`) also perform the role of copying information about the previous capital letter, as previously-mentioned.

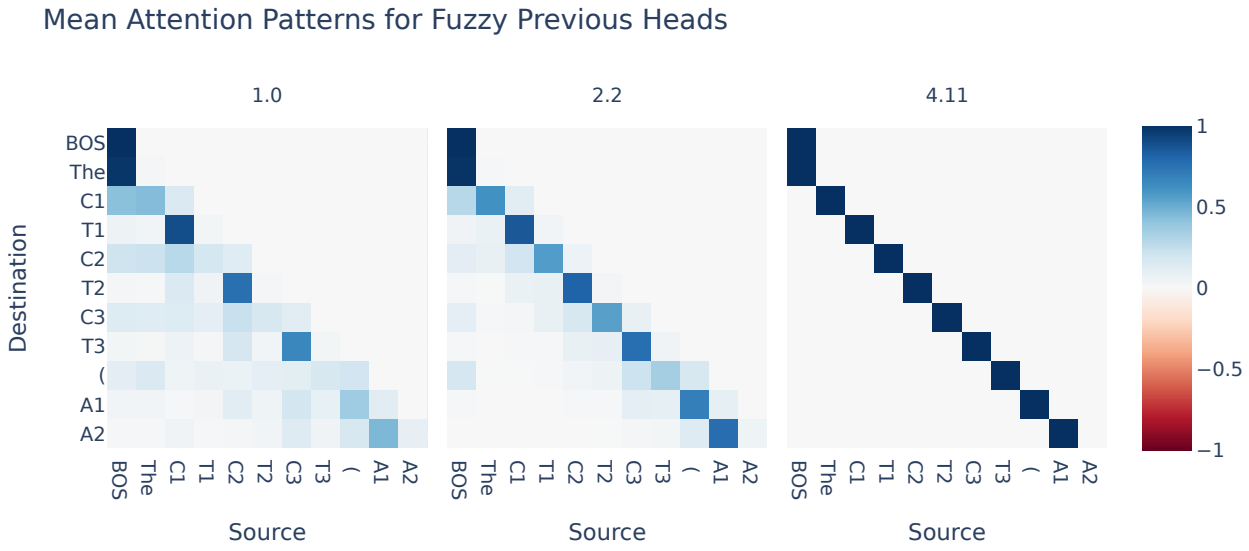

Figure 15: Mean attention patterns for the 3 heads on the circuit that move information from `C(i-1)` to `Ci`.

B POSITIONAL INFORMATION EXPERIMENTS

This section contains the remaining positional experiments (presented in Section 4.2) regarding the rest of possible swapping combinations and letter mover heads. Figs. 20 and 21 show the result of swapping the attention paid to `BOS` from the `C1` and `C2` tokens, and the `C2` and `C3` tokens respectively.

Figs. 22-24 show the difference in attention paid to the `Ci` tokens on the clean run, when swapping the positional embeddings, swapping the attention paid to the `BOS` token, and performing both swaps at the same time. This is visualized for every possible swap and letter mover head. As mentioned in the paper, it can be seen that the largest impact happens when performing the swapping operation on tokens `C1` and `C3`, specially on the first letter prediction, whereas the changes on the other swapping experiments are almost negligible.

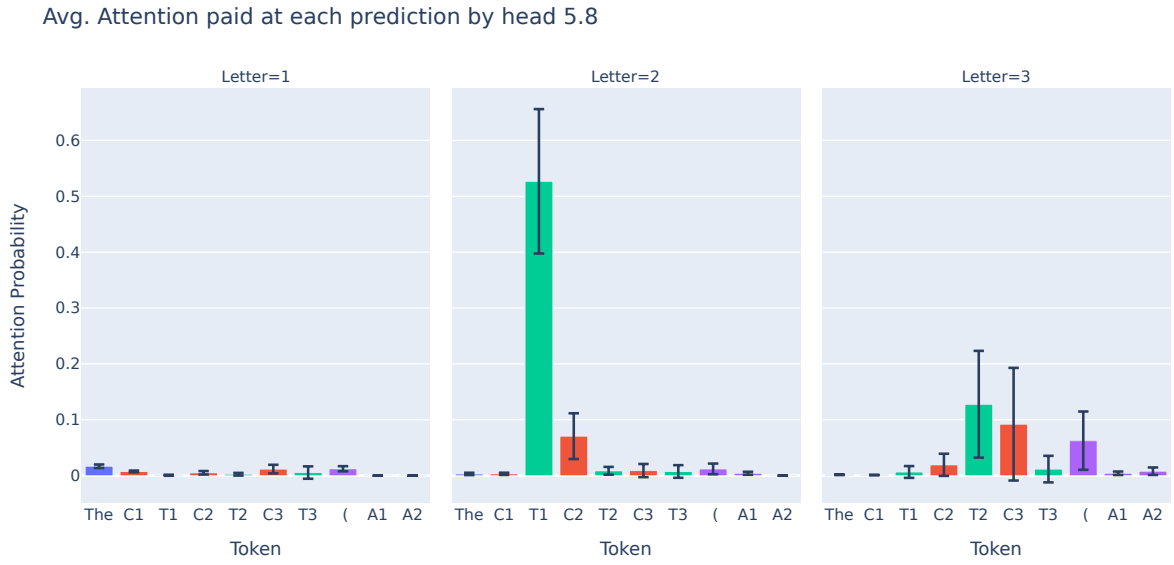

Figure 16: Average probability paid from `A(i-1)` to the previous token positions for head `5.8`.

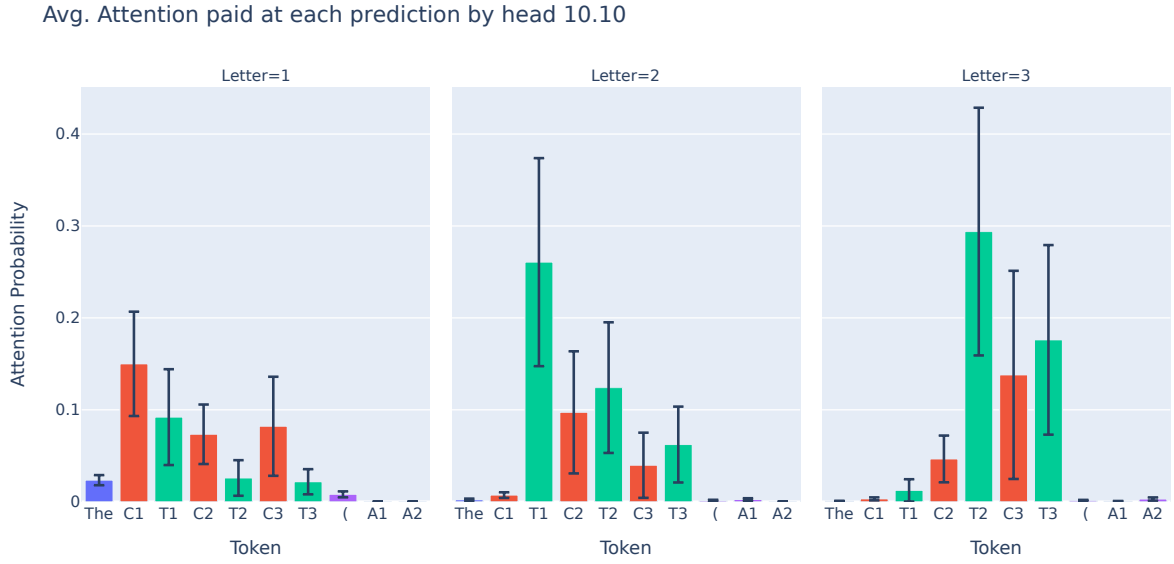

Figure 17: Average probability paid from `A(i-1)` to the previous token positions for head `10.10`.

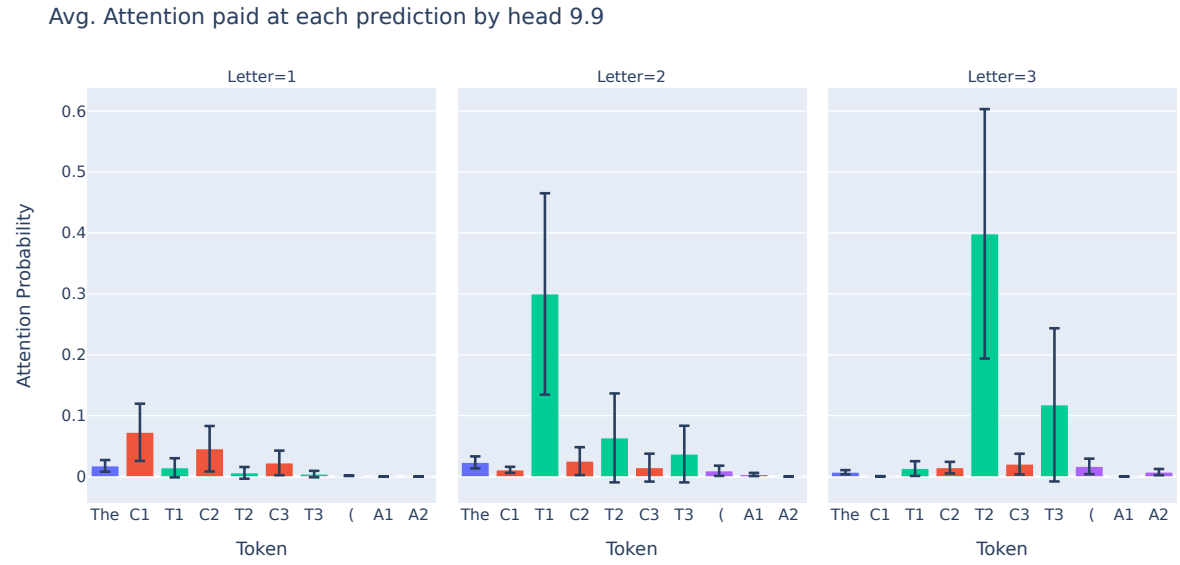

Figure 18: Average probability paid from `A(i-1)` to the previous token positions for head `9.9`.

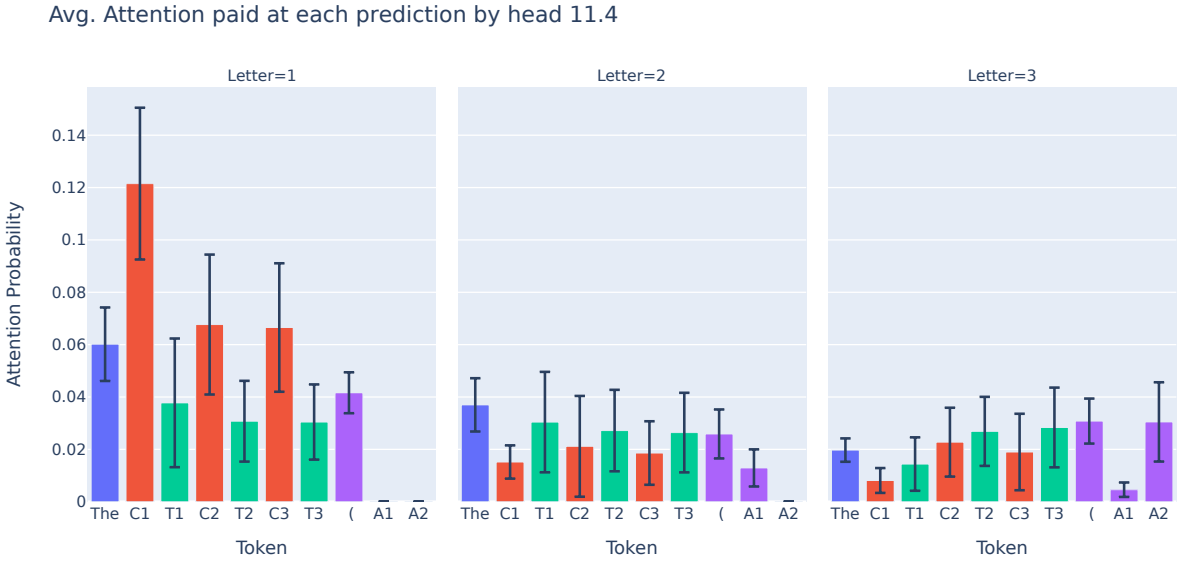

Figure 19: Average probability paid from `A(i-1)` to the previous token positions for head `11.4`.

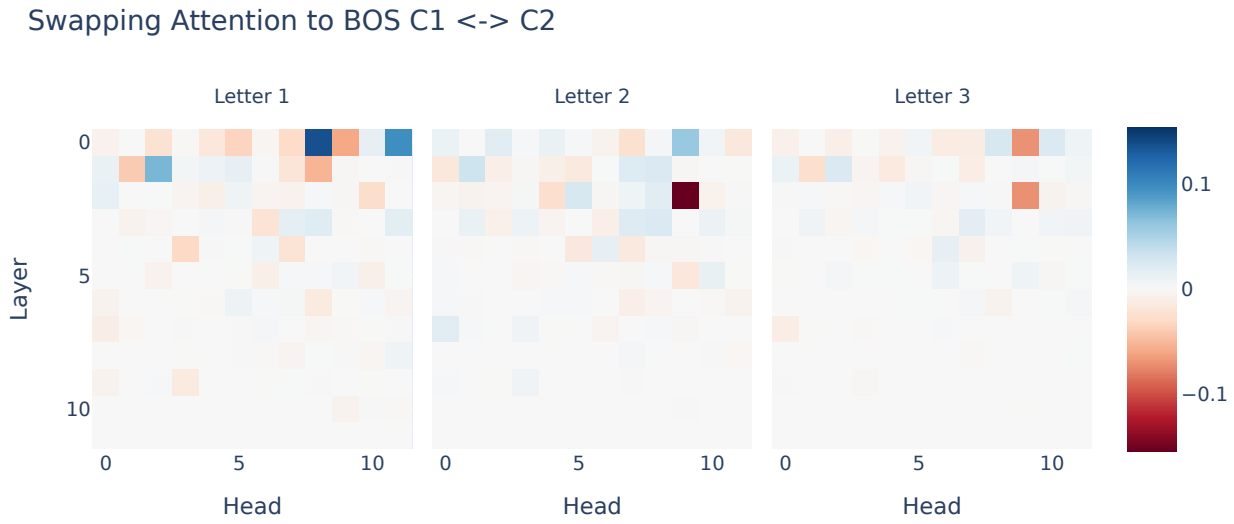

Figure 20: Change in logit difference obtained by swapping the attention paid to `BOS` from the `C1` and `C2` tokens for every head in the model.

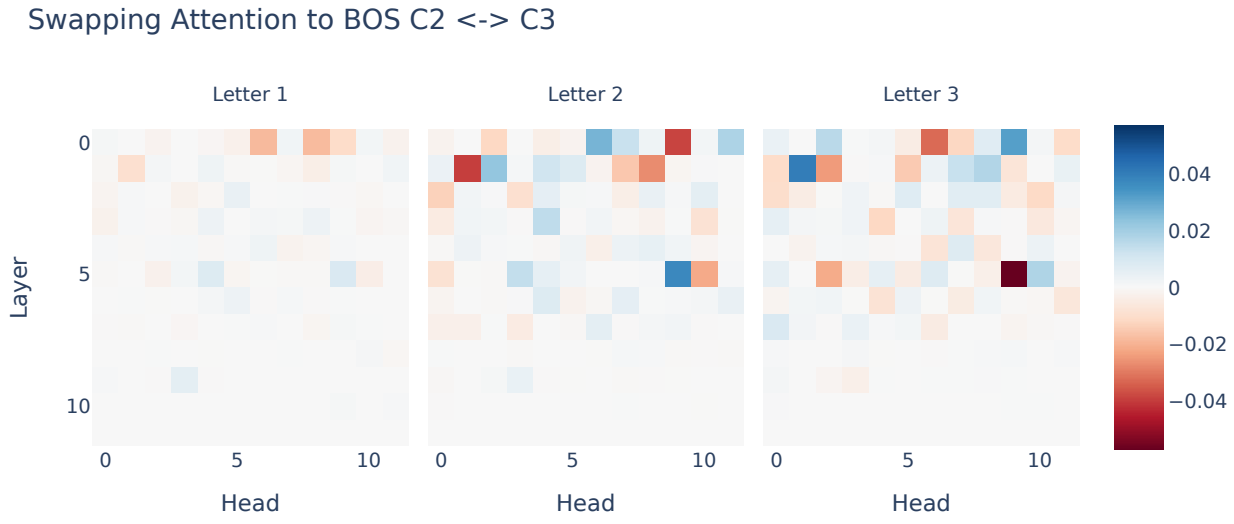

Figure 21: Change in logit difference obtained by swapping the attention paid to `BOS` from the `C2` and `C3` tokens for every head in the model.

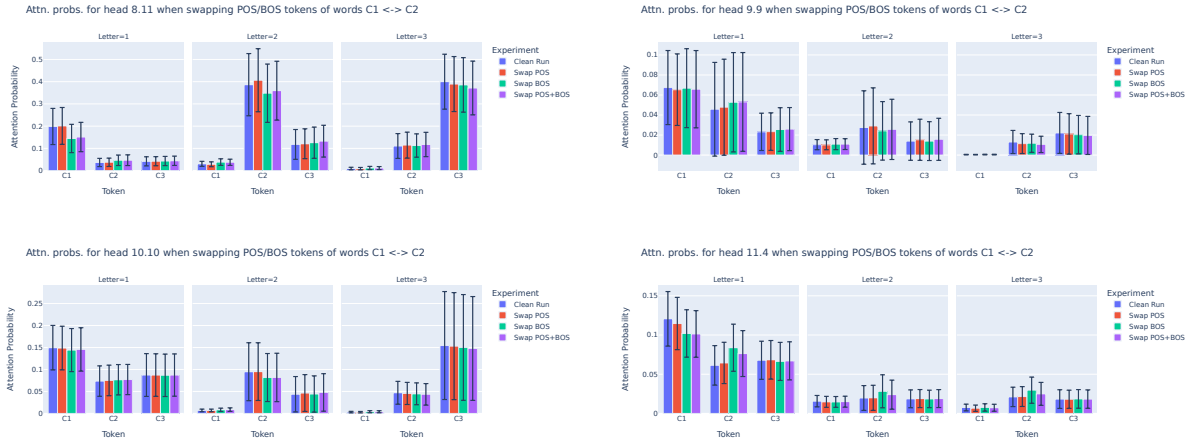

Figure 22: Effect of swapping the positional embeddings and/or attention to `BOS` of `C1` and `C2` on the attention paid to the capital letter tokens for each letter mover head.

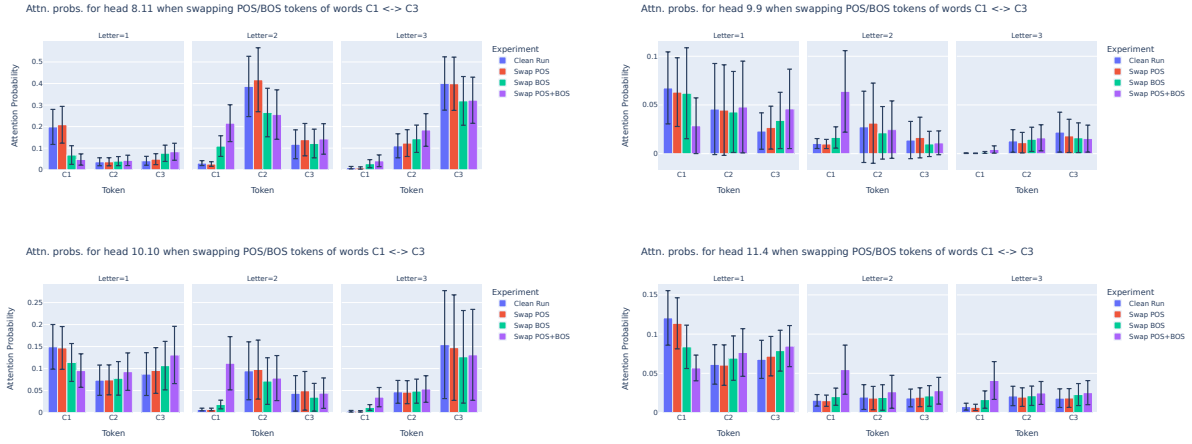

Figure 23: Effect of swapping the positional embeddings and/or attention to `BOS` of `C1` and `C3` on the attention paid to the capital letter tokens for each letter mover head.

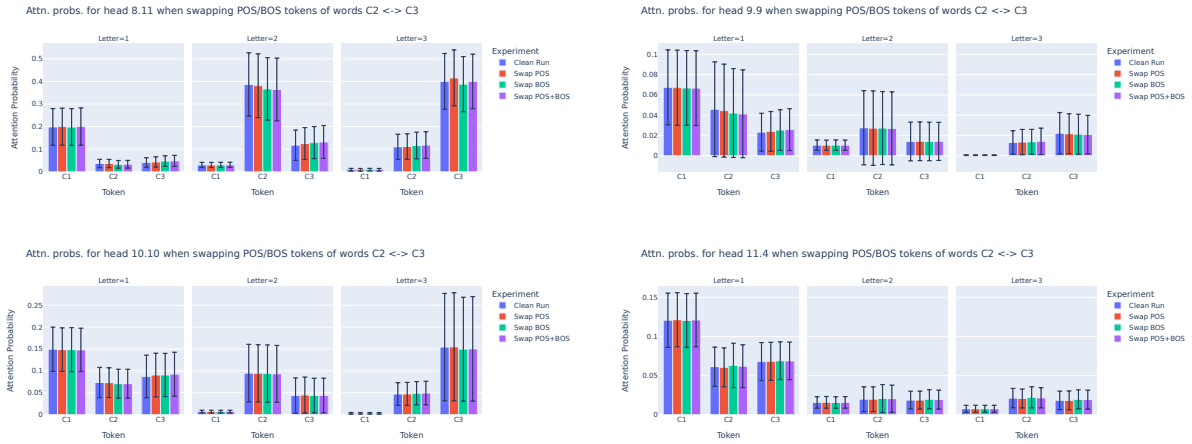

Figure 24: Effect of swapping the positional embeddings and/or attention to `BOS` of `C2` and `C3` on the attention paid to the capital letter tokens for each letter mover head.