Assignment on Mechanistic Interpretability of Transformers

Guillem Soler

January 2025

1 Introduction

The implementation described here builds upon the GPT-2 model from Andrej Karpathy's `minGPT` repository. This project explores the activation patching technique for analyzing model behavior and understanding its predictions. Below, we discuss the key steps and modifications made to the base implementation.

2 Tokenization

The GPT-2 model uses a Byte-Pair Encoding (BPE) tokenizer to segment input text into subword units or words. The tokenizer provided in `minGPT` was used to tokenize both clean and corrupted text inputs. An example of tokenization:

```
from mingpt.bpe import BPETokenizer

tokenizer = BPETokenizer()
clean_tokens = tokenizer(clean_text).to(device)
corrupted_tokens = tokenizer(corrupted_text).to(device)
assert clean_tokens.shape == corrupted_tokens.shape, "Texts
    must have same number of tokens"
```

Listing 1: Tokenizing input text

3 Code Modifications and Features

The following key features were implemented:

3.1 Storing Layer Activations

A boolean flag `save_activations` was added to the `forward` function of the transformer model. When enabled, this stores the activations of each layer and position for later use.

```
if save_activations:
    self.stored_activations = []
...
if save_activations:
    self.stored_activations.append(x.detach().clone())
```

Listing 2: Saving activations in the forward function

3.2 Activation Patching

Activation patching involves replacing corrupted activations at specific layers and positions with clean activations and measuring its impact on the model's output. This helps identify critical layers and positions contributing to predictions.

```
if patch_layer is not None and patch_position is not None
    and clean_activations is not None:
    if layer_idx == patch_layer:
        x_new = x.clone()
        x_new[:, patch_position, :] = clean_activations[
            layer_idx][:, patch_position, :].clone()
        x = x_new
```

Listing 3: Applying activation patching in forward function

3.3 Logit Analysis for the Last Token

To analyze the model's predictions, the logits of the last token were stored after each forward pass.

```
x = self.transformer.ln_f(x)
logits = self.lm_head(x)
self.last_token_logits = logits[:, -1, :]
```

Listing 4: Storing logits of the last token

3.4 Difference Matrix Computation

A nested loop iterates over all layers and positions, performing activation patching for each combination. The difference in logits for specific tokens is computed and stored in a matrix.

```
difference_matrix = torch.zeros((num_layers, num_positions))
for layer in range(num_layers):
    for position in range(num_positions):
        with torch.no_grad():
            _, _ = model(
                corrupted_tokens,
```

```
                patch_layer=layer,
                patch_position=position,
                clean_activations=clean_activations
            )
            patched_logits = model.last_token_logits
            current_diff = patched_logits[0, smith_index] -
                patched_logits[0, jones_index]
            difference_matrix[layer, position] =
                current_diff.item()
```

Listing 5: Calculating logit differences

4 Visualization

The resulting difference matrix is visualized to understand the impact of each layer and position on the model's predictions.

```
plt.imshow(difference_matrix, cmap='gray', aspect='auto',
    interpolation='nearest')
cbar = plt.colorbar()
cbar.set_label('Logit Difference', fontsize=12)
plt.title("Impact of Activation Patching on Token Prediction
    ")
plt.xlabel("Token Position")
plt.ylabel("Layer")
plt.show()
```

Listing 6: Plotting the difference matrix

This implementation demonstrates how activation patching and logit analysis can provide insights into the inner workings of GPT-2, helping identify critical activations and potential sources of prediction errors.

5 Analysis of the Leaning Tower Example

The difference matrix visualized in Figure 1 provides insights into how interventions in different layers and positions affect the model's predictions for the `clean_text`:

> The main feature of the leaning Tower of Pisa is that it is perfectly leaning. The tower of Pisa is perfectly

and the `corrupted_text`:

> The main feature of the straight Tower of Pisa is that it is perfectly leaning. The tower of Pisa is perfectly

Similar to the Smith/Jones example, we analyze a case where a well-known fact is altered. Here, we introduce an incorrect description of the Tower of Pisa as "straight" instead of its famous "leaning" characteristic. This parallel structure allows us to examine how the model handles factual corrections in different contexts.

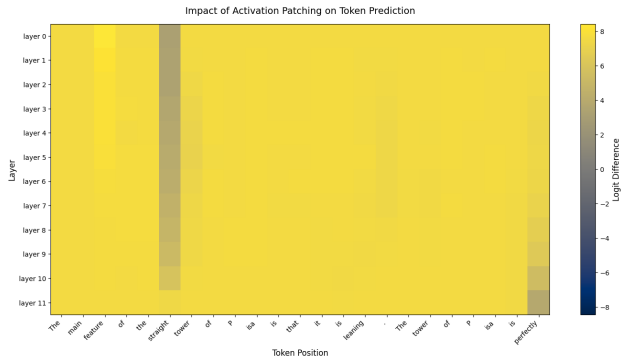

Figure 1: Difference matrix for activation patching on the Leaning Tower example.

5.1 Observations

The attention mechanism and information flow in the model reveals several key patterns:

- Similar to the Smith/Jones case, the first column (initial tokens) shows minimal impact on predictions, as these embeddings maintain identical values in both clean and corrupted contexts. This confirms the model's consistent behavior across different examples when handling unmodified prefix tokens.

- The most pronounced effects appear around the intervention point (`straight/leaning`), visualized by the darker region in the first layers at this position. This mirrors the behavior observed in the Smith/Jones example, where modifying embeddings of the critical token significantly influenced the prediction trajectory. However, as we can see the logit difference is lower than in the example seen.

- The middle sequence tokens (`of`, `Pisa`, `is`) demonstrate minimal intervention effects, suggesting these contextual elements play a secondary role in the factual correction process. This pattern reinforces our understanding of how the model prioritizes semantically significant tokens.

- In the final positions, particularly at `perfectly`, the deeper layers (10-11) show anticipatory behavior towards the correct prediction, evidenced

by the subtle color variations. This aligns with our previous observations about how final layer embeddings contribute to sequence prediction. However, it is also lower, this probably is because in the same sentence we are giving two different informations 'leaning' and 'straight' but the model 'knows' that leaning is the correct one and that's why the difference is smaller than in the example Jones/Smith.

This analysis strengthens our understanding of the model's correction mechanisms by demonstrating consistent patterns across different examples. The parallel between the Tower of Pisa and Smith/Jones cases suggests a generalizable framework for how the model handles factual corrections.

In the following example we are going to try a little bit different example to test the boundaries of this correction mechanism and investigate how the model handles cases with multiple competing facts or more nuanced contextual dependencies.

6 Analysis of the Eiffel Tower Example

The difference matrix visualized in Figure 2 provides insights into how interventions in different layers and positions affect the model's predictions for the `clean_text`:

> The Paris Eiffel Tower, a landmark in Paris, is situated in the country of

and the `corrupted_text`:

> The Milan Eiffel Tower, a landmark in Paris, is situated in the country of

6.1 Experimental Design

This experiment was designed to see how the model processes and reconciles conflicting geographical information. The approach leverages the Eiffel Tower's strong geographical association with France and Paris in the model's knowledge base. We constructed two versions of the text:

- A clean version reinforcing the correct location ("Paris Eiffel Tower")
- A corrupted version introducing geographical conflict ("Milan Eiffel Tower")

The key aspect of this design lies in exploiting the left-to-right processing nature of language models. In the corrupted text, we deliberately position the conflicting location ("Milan") before the confirming information ("landmark in Paris"), creating a scenario where the model must navigate between:

1. Trusting its pre-existing knowledge about the Eiffel Tower's location

2. Following the sequential information as presented in the text

This setup(we dit it also with Pisa tower) is particularly revealing because the model encounters incorrect information (Milan) before the correct contextual cue (Paris). Due to the unidirectional nature of the model's processing, any influence from early tokens persists throughout the sequence, forcing the model to either maintain or override its initial geographical interpretation based on its knowledge base and subsequent context.

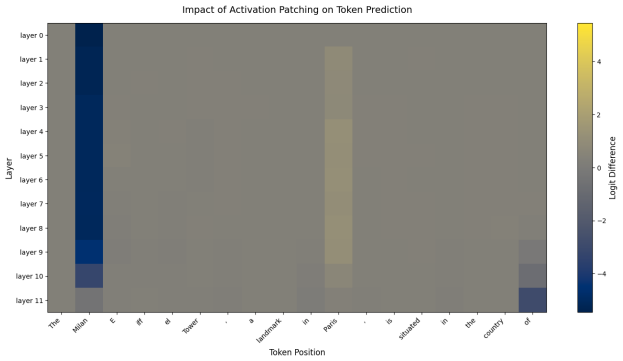

Figure 2: Difference matrix for activation patching on the Eiffel Tower example.

6.2 Observations

The attention mechanism reveals several distinctive patterns, including a behavior not observed in previous examples:

- The most striking feature is the intense negative logit difference (dark blue) concentrated in the early layers at `position 1` (`Milan`). This represents a much stronger correction signal than seen in the previous example, suggesting the model strongly rejects the inconsistency.
- Interestingly, we observe a subtle positive logit difference (lighter shading) around the `Paris` token position, indicating the model might be reinforcing the correct location information to counteract the earlier geographical error.
- The final positions show small but noticeable differences in the deeper layers (10-11), suggesting the model is preparing to generate the correct country name despite the earlier conflicting information.

6.3 Positive Logit Difference

This example reveals a particularly interesting pattern not observed in previous cases - a distinct positive logit difference (lighter shading) around the `Paris`

token position. This feature is especially noteworthy when contrasted with the strong negative signal at `Milan`:

- While the model shows a strong rejection of the incorrect location (`Milan`) through a dark vertical pattern, it simultaneously produces a lighter verification signal when encountering the correct location (`Paris`). We talk about verification and rejection in base of the logit difference, the difference in Milan is negative and in Paris is positive.
- This "verification stripe" suggests the model actively seeks confirming evidence for its geographical knowledge, rather than simply rejecting incorrect information
- The presence of both patterns - rejection at `Milan` and verification at `Paris` - suggest a sophisticated mechanism for handling the correct information where the model seems to validate the correct information.

The presence of this verification pattern raises interesting questions about how models might employ similar validation mechanisms in other domains where factual consistency is crucial.

7 Conclusion

The analysis of the Tower of Pisa and Eiffel Tower examples reveals key insights into GPT-2's mechanisms for handling factual inconsistencies:

- Knowledge Robustness: The model effectively rejects incorrect information (e.g., `Milan`) while reinforcing accurate data (`Paris`) through distinct logit patterns.
- Contextual Validation: It actively verifies correct context even when preceded by conflicting cues, demonstrating a sophisticated correction mechanism.
- Layer Contributions: Deeper layers play a critical role in aligning predictions with factual knowledge.
- Consistency Across Examples: The observed patterns suggest a general framework for transformers in resolving conflicts between input sequences and pre-existing knowledge.

This assignment has provided valuable insights into transformer model behavior through activation patching analysis. By examining how models handle inconsistencies and factual corrections. These probing techniques offer a window into the decision-making process, revealing how they weigh and reconcile conflicting information.