Mechanistic Interpretability of MiniGPT: Multilingual Analysis with Activation Patching

Louis Nicolas and Barbara Loletti

January 24, 2025

1 Introduction

Recent advances in natural language processing (NLP) have led to the creation of transformer-based models like GPT, which excel in text generation, comprehension, and reasoning. Despite their success, understanding their internal workings remains a challenge. Mechanistic interpretability aims to uncover how these models process inputs and generate coherent outputs.

This report explores GPT models through activation patching, a technique for analyzing how specific activations influence outputs. Using multilingual examples, we evaluate the model's ability to process and integrate data, visualizing results with heatmaps to identify key layers and tokens.

The goal is to provide insights into the model's interpretability, emphasizing its multilingual strengths and limitations while deepening our understanding of transformer-based language comprehension.

2 How MiniGPT Works

MiniGPT is a simplified implementation of the GPT model architecture designed for mechanistic interpretability and multilingual evaluation. MiniGPT uses a transformer-based architecture, specifically a medium-sized GPT-2 model, which is capable of generating and interpreting sequences in multiple languages. The key components of MiniGPT are:

- **Transformer Blocks:** MiniGPT employs a stack of transformer blocks, each consisting of a multi-head self-attention mechanism and feedforward layers. These blocks are responsible for capturing contextual relationships between tokens in a sequence.
- **Byte Pair Encoding (BPE):** Input sequences are tokenized using a BPE tokenizer, which encodes text into subword units, enabling the model to process diverse languages efficiently.
- **Pretrained Parameters:** The model leverages pretrained weights from a GPT-2 medium model, which include embeddings, positional encodings, and attention weights. This type of initialization allows the model to start with a strong foundation of linguistic knowledge.
- **Activation Patching:** During analysis, MiniGPT stores layer activations and performs activation patching to examine the influence of specific layers and tokens on the model's outputs.

The model processes input sequences by encoding them into embeddings, adding positional encodings, and passing the encoded inputs through transformer layers. The final logits are produced by a linear projection layer, which maps the hidden states to vocabulary probabilities.

2.1 Model Architecture

MiniGPT is a transformer-based language model implemented using a modular architecture inspired by the original GPT-2 model. The core implementation is found in `model.py` and for this project it has been chosen the GPT-2 medium model (350M parameters). This architecture in particular consists of:

- **Layer Count**: 24 transformer layers, organized into three distinct groups based on their functionality:

  - *Initial Layers (0 to 6):* They primarily handle low-level linguistic features such as token relationships and syntactic structures. They focus on capturing word-level dependencies and basic patterns in the input text.
  - *Middle Layers (7 to 16):* Responsible for more complex linguistic processing, they capture semantic relationships, context integration, and sentence-level coherence, ensuring the model can understand the broader context of the input.
  - *Final Layers (17 to 23):* They refine and consolidate the contextual understanding obtained from the earlier layers, focusing on generating high-quality outputs by synthesizing information and aligning it with the target language or task.

- **Attention Heads**: MiniGPT employs 16 attention heads per layer, each designed to focus on different aspects of the input sequence. These attention heads work in parallel to:
  - Capture token-to-token relationships at varying distances, from immediate neighbors to long-range dependencies.
  - Process multiple perspectives of the data simultaneously, enabling the model to recognize subtle and complex patterns in the text.
  - Assign varying degrees of importance to different parts of the input, ensuring relevant context is emphasized while less important details are downplayed.

  The multi-head mechanism allows the model to combine insights from all attention heads, resulting in a rich and nuanced understanding of the input.

- **Embedding Dimension**: 1024.

3 Code Implementation and Explanation

The MiniGPT project aims to test the ability of the model to predict the correct next word in English, French, Italian, and Spanish. By analyzing clean and corrupted inputs, it examines the model's language understanding and sensitivity to textual changes. The focus is on understanding MiniGPT's multilingual text processing and generation through mechanistic interpretability. Below, we outline the purpose and functionality of the key Python scripts used in the project.

3.1 Overview of Clean and Corrupted Inputs

The scripts for each language define clean and corrupted input sentences to analyze the model's interpretability. Clean sentences are grammatically correct, while corrupted ones alter a single word to test the model's sensitivity. The same example is translated into all four languages, resulting in the following inputs:

- **English:** ”The cat jumped nimbly from the table to the chair, landing with elegance in front of the open window.” (clean) vs. replacing ”elegance” with ”clumsiness” (corrupted).
- **Spanish:** ”El gato brinco rápido desde la mesa hacia la silla cayendo cerca de la ventana grande.” (clean) vs. replacing ”cayendo” with ”tropezando” (corrupted).
- **French:** ”Le chat sauta rapidement depuis la table vers la chaise tombant doucement.” (clean) vs. replacing ”doucement” with ”bruyamment” (corrupted).
- **Italian:** ”Il gatto salto agilmente dal tavolo alla sedia atterrando con eleganza.” (clean) vs. replacing ”eleganza” with ”goffaggine” (corrupted).

3.2 Tokenization and Input Preparation

The sentences are tokenized into meaningful subword units using the `BPETokenizer`, which enables the model to process a wide range of languages efficiently, using the following lines of code:

```
clean = tokenizer("The cat jumped nimbly from the table to the chair, landing with
elegance.").to(device)
corrupted = tokenizer("The cat jumped nimbly from the table to the chair, landing with
clumsiness.").to(device)
```

3.3 Iteration and Logits Computation for Specific Tokens

The scripts iterate over each specific token in order to compute logits and probabilities, enabling analysis of the model's output sensitivity. For each token, probabilities are calculated using the softmax function applied to the logits. The following code illustrates the process:

```
for specific_token in SPECIFIC_TOKENS:
    print(f"Processing token: {specific_token}")
    diff_matrix = np.zeros((n_layers, seq_length))

    for layer in range(n_layers):
        for pos in range(seq_length):
            logits_patched, _ = model(
                corrupted.to(device),
                patch_params=(layer, pos, clean_activations[f'layer_{layer}'][:, pos, :])
            )
            patched_probs = F.softmax(model.last_token_logits[0], dim=-1)
            diff_matrix[layer, pos] = patched_probs[token_id] - reference_probs[token_id]
```

This iterative process calculates the differences in probabilities between the clean and corrupted inputs, highlighting the contribution of individual tokens and layers to the model's predictions.

3.4 Activation Patching and Analysis

The core of the analysis is the activation patching mechanism, where activations from the clean input are used to replace corresponding activations in the corrupted input. This allows for detailed examination of the layers and tokens responsible for the model's outputs. The following snippet illustrates the setup:

```
for layer in range(n_layers):
    for pos in range(seq_length):
        logits_patched, _ = model(
            corrupted.to(device),
            patch_params=(layer, pos, clean_activations[f'layer_{layer}'][:, pos, :])
        )
        diff_matrix[layer, pos] = compute_difference(logits_patched, reference_logits)
```

The difference matrix is computed for each token and layer, quantifying the impact of specific activations on the model's predictions.

3.5 Visualization

To provide interpretable results, heatmaps are generated in order to visualize the impact of activation patching. These heatmaps highlight the most influential layers and tokens for specific outputs.

3.6 Language-Specific Implementations

Each language script follows the same structure, differing only in the input sentences and specific tokens analyzed. For example, the Spanish script evaluates tokens like "gato" and "ventana," while the French script focuses on "chat" and "fenêtre."

By standardizing the process across multiple languages, the implementation ensures consistent and comparable results, showcasing the multilingual capabilities of the MiniGPT model.

4 Analysis of Results and Observations

We analyze activation patching experiments on MiniGPT across English, French, Italian, and Spanish, with insights from heatmaps visualizing layer activations for clean and corrupted inputs.

4.1 Activation Patterns Across Layers and Multilingual Consistency

The heatmaps reveal that the final layers consolidate context most significantly across all languages, with activation patterns varying between languages and input conditions. Below are the key findings for each language.

4.1.1 English

For English, activations for tokens like ”elegance” (clean) and ”clumsiness” (corrupted) were diffused across layers rather than localized, highlighting distributed representations. The final layers refined predictions but showed limited token-specific differentiation between clean and corrupted inputs.

4.1.2 French

In French, tokens such as ”dou” (softly) and its corrupted counterpart ”bru” (noisily) exhibited moderate activations in the final layers, emphasizing their semantic influence. However, other tokens like ”ment” and ”y” showed more diffused activations, indicating reduced token-specific contributions. MiniGPT effectively processed French inputs while emphasizing localized activations for key tokens.

4.1.3 Italian

For Italian, the clean token ”eleganza” (elegance) showed localized activations in the final layers, while its corrupted counterpart ”goffaggine” (clumsiness) exhibited diffused activations. Tokens like ”go” and ”agg” reflected moderate activations across middle and final layers. This indicates heightened sensitivity to corruption in Italian compared to other languages.

4.1.4 Spanish

In Spanish, tokens like ”trope” (stumbled) and ”endo” (falling) showed significant activations in middle and final layers, whereas tokens like ”ay” and ”z” were less impactful. The distributed pattern highlights MiniGPT’s generalized multilingual processing but reduced sensitivity to specific corrupted tokens.

4.2 Role of Specific Tokens Across Languages

Across languages, MiniGPT exhibited a consistent pattern of distributed processing. Semantically significant tokens, such as ”elegance” in English and ”eleganza” in Italian, showed higher activations in final layers, while less impactful tokens demonstrated diffused patterns. This underscores the model’s reliance on generalized contextual processing, though token-specific sensitivity remains limited.

4.3 Fine-Tuning Potential

Fine-tuning on a curated multilingual corpus could improve MiniGPT’s sensitivity to subtle semantic shifts and enhance generalization to additional languages. This refinement, using parallel texts and multilingual contexts, could address biases from monolingual pretraining and enable stronger token-level prediction accuracy.

The provided `trainer.py` script facilitates fine-tuning, which could yield improvements in token sensitivity and multilingual contextual understanding.

4.4 General Observations and Limitations

While promising, MiniGPT has several limitations:

- Reliance on final layers may limit inter-layer contextual integration.
- Lack of explicit multilingual training results in slight inconsistencies for corrupted inputs, as seen in diffused activations for some tokens.
- The medium model size (350M parameters) constrains linguistic depth compared to larger models like GPT-2 XL.

These observations highlight areas for improvement, including scaling the model, enriching training data, and exploring advanced fine-tuning techniques.

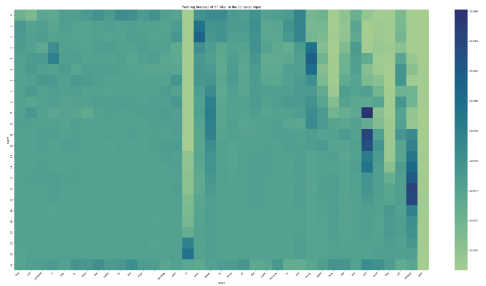
5 Conclusions

The analysis reveals valuable insights into MiniGPT's multilingual capabilities. Heatmaps show that while MiniGPT effectively consolidates semantic information in final layers, its reliance on distributed activations limits token-specific differentiation. The model demonstrates robust multilingual generalization but struggles with sensitivity to corrupted inputs.

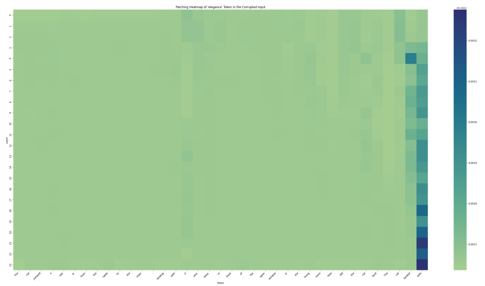
Fine-tuning MiniGPT on a diverse multilingual corpus offers potential for enhanced token sensitivity and prediction accuracy. Expanding the model's training data and architecture could further strengthen its ability to handle complex linguistic tasks. Overall, MiniGPT exemplifies the potential of transformer-based models in multilingual natural language processing, with opportunities for refinement to achieve even greater performance and interpretability.

A Heatmaps

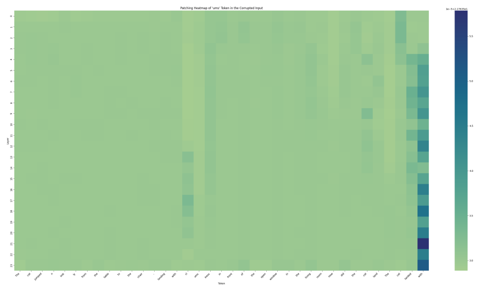
A.1 English



(a) ”cl” token’s heatmap

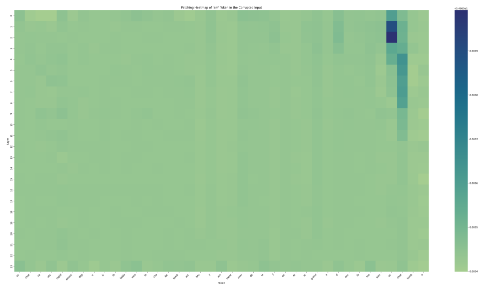


(b) ”elegance” token’s heatmap

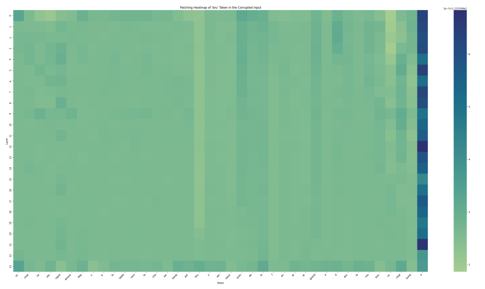


(c) ”ums” token’s heatmap

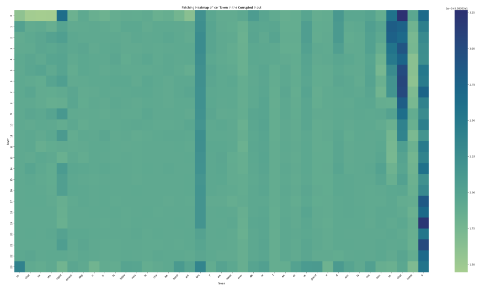
A.2 French



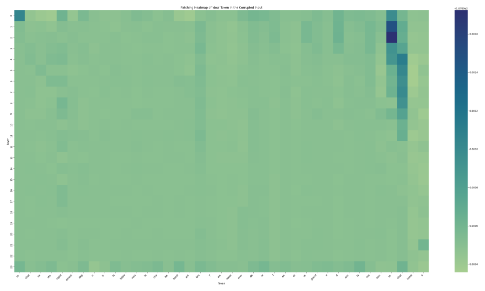
(d) ”am” token’s heatmap



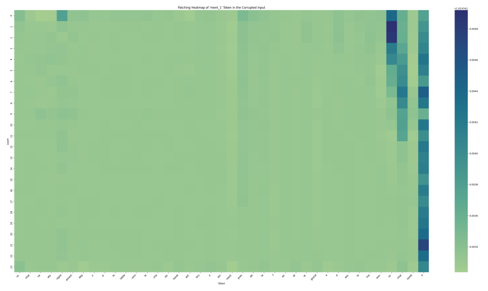
(e) ”bru” token’s heatmap

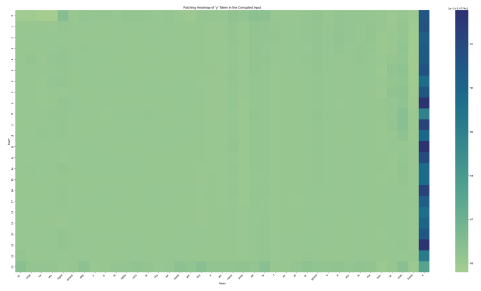


(f) ”ce” token’s heatmap

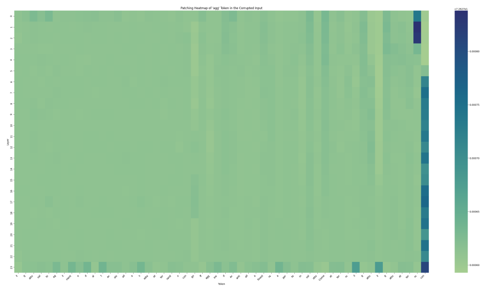


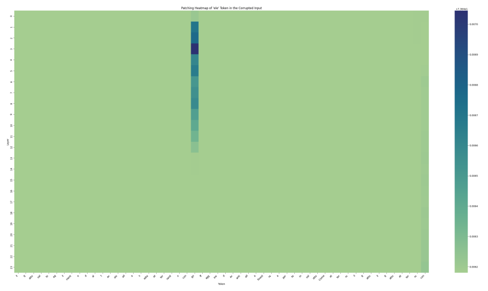
(g) ”dou” token’s heatmap

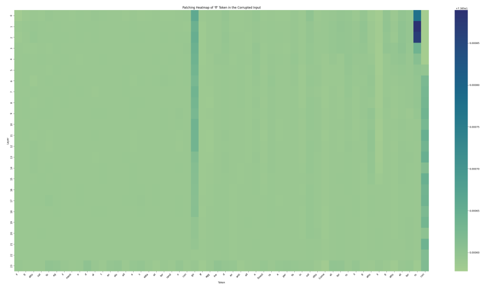

(h) ”ment” token’s heatmap

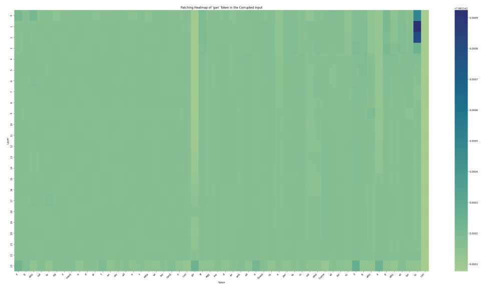

(i) ”y” token’s heatmap

A.3 Italian

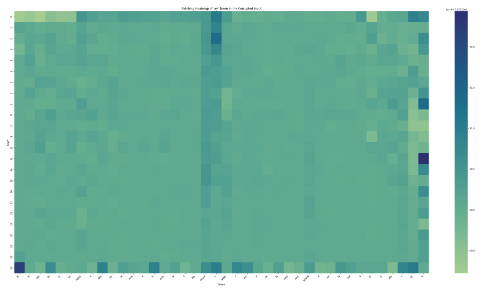

(j) ”agg” token’s heatmap

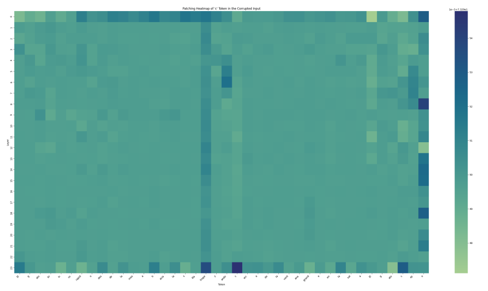

(k) ”ele” token’s heatmap

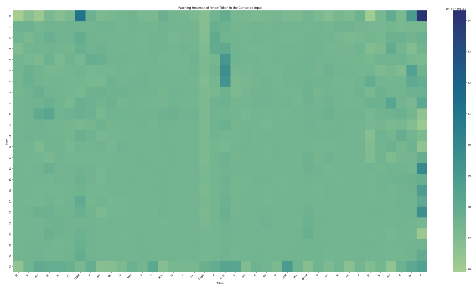

(l) ”ff” token’s heatmap

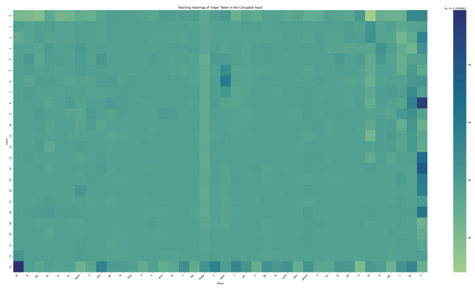

(m) ”gan” token’s heatmap

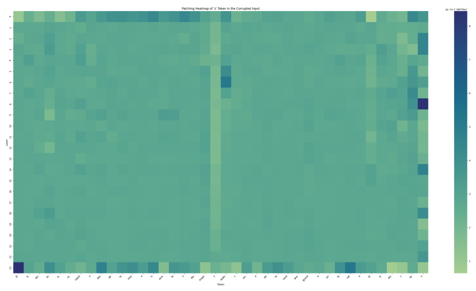
A.4 Spanish


(n) ”ay” token’s heatmap


(o) ”c” token’s heatmap


(p) "endo" token's heatmap


(q) "trope" token's heatmap


(r) "z" token's heatmap