Tracing Typos in LLMs: My Attempt at Understanding How Models Correct Misspellings

by **Ivan Dostal**
2nd Feb 2025

Because this is your first post, this post is awaiting moderator approval.

This blogpost was created as a part of the AI Safety Fundamentals course by BlueDot Impact. All of the code can be found on my GitHub.

TLDR: I tried to uncover if there are specific components in language models that enable typo correction. I identified a subword merging head in the first layer of Llama-3.2-1B that plays a crucial role in the process. In this blog post, I'll walk through my thought process and findings.

Motivation

Large language models are getting significantly more capable every month, but we still don't know how they work inside. If a model generates an incorrect answer, we currently have (almost) no way of explaining why it did so. Mechanistic interpretability tries to solve this problem. And while we probably won't be able to completely reverse engineer large language models anytime soon, interpreting smaller parts of the models is still very valuable since these findings often generalize to larger models and provide valuable insights into how they work.

I was reading a lot of mechanistic interpretability papers over the past year, but I have never actually done anything myself. So I finally gave it a try and explored how typo correction works in LLMs.

Initial exploration

I started by manually testing sentences with typos and looking at their attention patterns. I suspected an early attention head might specialize in typo correction. And sure enough, I found one, though it wasn't exclusively dedicated to typo correction. I found a *subword merging head*[1]. It moves information between tokens that belong to the same word, but were tokenized into multiple tokens. This happens when a word is rare and lacks a dedicated token or, as in my case, when a typo alters its structure.

1. I later discovered that these are already known (for example here), but I couldn't find much research on them.

Attention head ablation

To test whether this head played a role in typo reconstruction, I tried generating text with this head ablated (I zeroed out its output during the forward pass). Without ablation, when given the prompt:

> "<|begin_of_text|>What is the meaning of the word 'comptuer'? It means"

the model completed the sentence correctly:

> "What is the meaning of the word 'comptuer'? It means a computer. It is a machine that can do calculations."

But when I ablated the subword merging head, the model completely failed:

> "What is the meaning of the word 'comptuer'? It means to be a part of something. It is a verb."

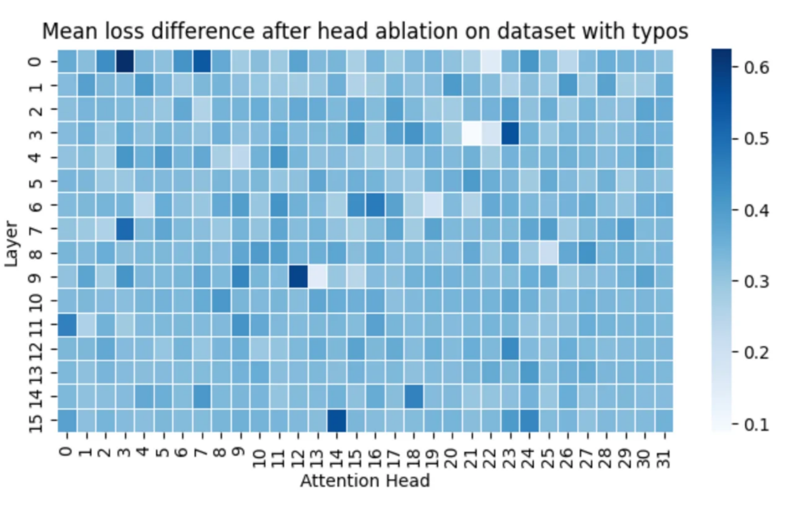
Interestingly, ablating this head had no noticeable effect on text without typos. This suggests that while the head isn't crucial for general language processing, it plays a specialized role in handling fragmented tokens—whether due to typos or rare words that lack a dedicated token.

Is there more to it?

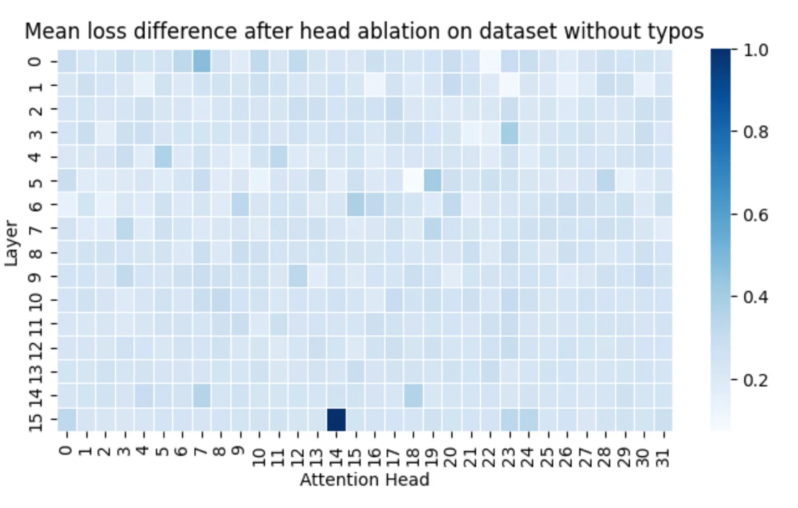
To further verify my findings and check for other important components I might have overlooked, I created a small dataset of sentences with the following structure

> "Text: <sentence_with_typos>, Corrected: <same_sentence_without_typos>"

and then measured mean loss across these samples when ablating individual heads one by one.



I also repeated the same process with the same text, but without typos.



As expected, the subword merging head **L0H3** had the largest impact on loss. Beyond **L0H3**, several other heads also significantly affect loss. Head **L15H14** is an induction head, which explains why it influences loss in both cases. Interestingly, a few other heads also seem important, but I wasn't able to determine their exact function.

Using a better metric

I later realized that loss isn't the best metric for this task, as it is influenced by many other factors than just typo correction success. The loss is computed as a cross entropy loss averaged over the whole sequence. In the format I used earlier, reducing loss would require the model to predict misspelled words in the first half of the sequence.

Another reason why the format isn't ideal is that the model probably has some complex mechanisms of correcting the typos based on surrounding words and broader context. I want to simplify things and only focus on typo correction without any additional context, just based on the word with a typo itself.

To improve this, I transitioned into using **logit difference**. Logit difference, a common metric in mechanistic interpretability, measures the gap between the logit for the correct token and the logit for the incorrect one. But what counts as the incorrect token in this task?

I created a simpler dataset of samples in the following format:

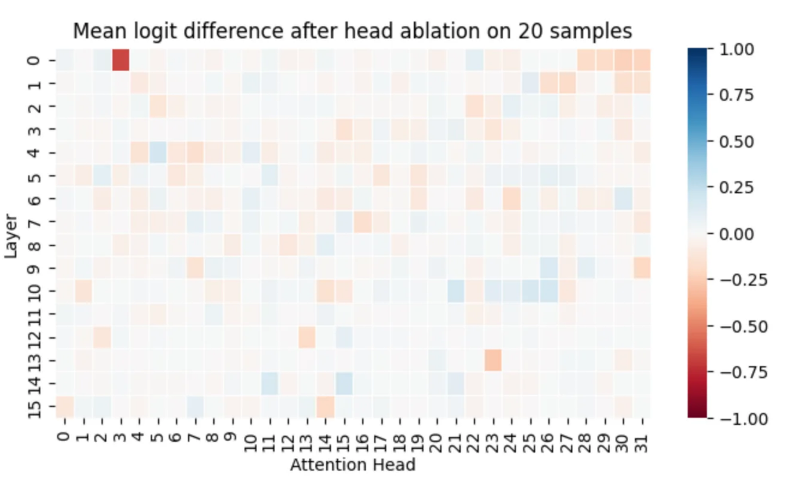
> "incorrect: <word_with_a_typo>[2], correct:"

Now if the model realizes what the typo is, it predicts the corrected word (correct token). Otherwise it repeats the misspelled word again (incorrect token is the first token of the misspelled word[3]). With this format I can measure how good the model is at correcting typos. I only used samples where the model correctly predicts the right word in the base setting, so that I could see which components are crucial for this task. If I ablate a head and the logit difference decreases, it means that the model is likelier to predict the incorrect answer, so the ablated head had a positive influence on the task.

2. I experimented with different types of typos but ultimately chose to swap the first two letters in all samples of this dataset. This approach keeps all original letters intact, unlike letter addition or deletion, making it easier for the model to correct without relying on additional context.

3. This format is still not perfect, because the model occasionally generates the correct word, but split into multiple subtokens.

I ran the experiment from above again and ablated all attention heads one by one and plotted the change in logit difference.



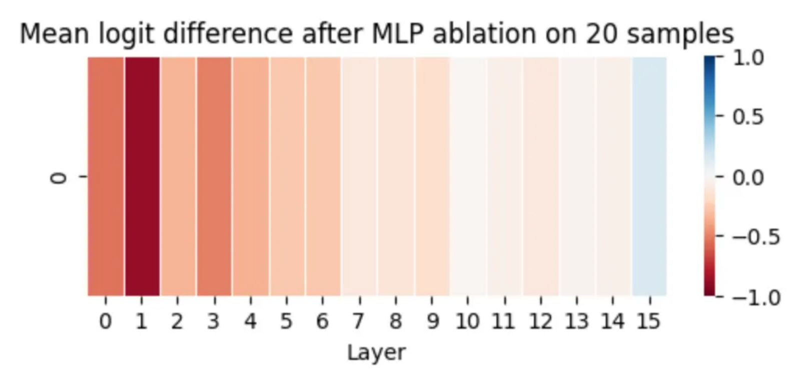
With this setup, only the subword merging head (L0H3) shows a significant impact[4]. That suggests, that the previous plot with loss was a bit misleading and that the loss was probably influenced by other factors. It seems that without additional surrounding context, the typo correction is really done mainly by this head[5].

4. The second most impactful head, L13H23, functions as an induction head in these samples. This makes sense, as some tokens repeat within the sequence.

5. However, it's possible that I missed some important heads. Models often have backup heads for critical tasks, meaning that if one head was ablated, another might have compensated for its function. This could lead to some heads not appearing as crucial in the plot.

MLP ablation

So far I only experimented with attention heads. But what if MLP layers also contribute to this task? To check this, I also tried ablating MLP layers.



At first glance it may look like this also plays a major role in typo correction, but as it turns out, the first few MLP layers are really important and ablating them degrades the general performance of the model across all tasks.

What is copied by the subword merging head?

The above seems like a strong evidence that typo correction in the examined model (in a setting without a broad context) is done mostly by a single attention head. But what exactly is this head doing? The hypothesis is that it moves information from previous tokens to the current token, if they are part of the same word. For example, with the word '*computer*' misspelled and tokenized as |*com*|*pt*|*uer*|, the attention head would move information about the preceding tokens (|*com*| and |*pt*|) to the position of the token |*uer*|.

I wanted to test whether this was truly the case. The most straightforward way to do that was **logit lens**. Logit lens is a technique that applies the final unembedding to intermediate activations inside the model. It skips all following layers and directly predicts tokens from the given activations.

I hoped that by unembedding the outputs of the subword merging attention head at the last token of a given word, the previous subword tokens would be the most probable prediction. Unfortunately, this was rarely the case. However, the predictions weren't completely unrelated. After inspecting a lot of different samples, I discovered that most of the time, the predicted tokens start with the same letter as the previous token ends[6].

6. This might not generalize to all kind of typos - in the later experiments I tested only typos with the first two letters of a word switched.

Here is an example of that with the word *weather*:

```
Inspected text:
<|begin_of_text|>|incorrect|:| ew|ather|,| correct|:

Undembedding the output of L0H3 at token position 4 (token |ather|)
10 most probable tokens:
TOKEN | Weld|, logit value: 0.1449
TOKEN | jur|, logit value: 0.1370
TOKEN | welded|, logit value: 0.1232
TOKEN |jur|, logit value: 0.1227
TOKEN | weld|, logit value: 0.1164
TOKEN | wrap|, logit value: 0.1142
TOKEN | ward|, logit value: 0.1126
TOKEN | Wald|, logit value: 0.1126
TOKEN | wallet|, logit value: 0.1075
TOKEN | Warrior|, logit value: 0.1059
```

While this isn't the neat and clean result I had hoped for, it still suggests that some "meaning" of the previous token is copied to the current token position. It would be interesting to further explore this with SAEs and see if any interpretable features come up.

Conclusion

In hindsight, it makes sense that when the surrounding context is limited, ablating the subword merging head hinders the model's ability to correct typos. This head plays a key role in reconstructing misspelled words, particularly when there's insufficient context to rely on. By transferring information between subword tokens, it helps the model "reassemble" the word even when parts of it are misspelled, making typo correction more effective.

This project was my first mechanistic interpretability project and also my first blog post, so I'd love to hear any feedback or suggestions for improvement.

---

1. ^ I later discovered that these are already known (for example here), but I couldn't find much research on them.
2. ^ I experimented with different types of typos but ultimately chose to swap the first two letters in all samples of this dataset. This approach keeps all original letters intact, unlike letter addition or deletion, making it easier for the model to correct without relying on additional context.
3. ^ This format is still not perfect, because the model occasionally generates the correct word, but split into multiple subtokens.
4. ^ The second most impactful head, L13H23, functions as an induction head in these samples. This makes sense, as some tokens repeat within the sequence.
5. ^ However, it's possible that I missed some important heads. Models often have backup heads for critical tasks, meaning that if one head was ablated, another might have compensated for its function. This could lead to some heads not appearing as crucial in the plot.
6. ^ This might not generalize to all kind of typos - in the later experiments I tested only typos with the first two letters of a word switched.