A Mathematical Framework for Transformer Circuits

Wenjie Li
liwj2022@shanghaitech.edu

Abstract

In this paper we will study transformers with two layers or less which have only attention blocks. Our aim is to discover simple algorithmic patterns, motifs, or frameworks that can subsequently be applied to larger and more complex models. We find that by conceptualizing the operation of transformers in a new but mathematically equivalent way, we are able to make sense of these small models and gain significant understanding of how they operate internally. Of particular note, we find that specific attention heads that we term "induction heads" can explain in-context learning in these small models, and that these heads only develop in models with at least two attention layers. Then I conducted some experiments on specific data.

1 Introduction

1.1 The Research Problem

In this paper, we attempt to take initial, very preliminary steps towards interpret transformers. Using toy models like a transformer with only 2 attention layers and no mlp layer, we hope to make sense of these small models and gain significant understanding of how they operate internally.

A previous project, the *Distill Circuit's thread*, focused on interpreting vision models[1], but so far there hasn't been a comparable project for transformers or language models.

1.2 Importance and Background

Transformer language models are an emerging technology that is gaining increasingly broad real-world use, for example in systems like GPT-4, Codex and similar models. However, as these models scale, their open-endedness and high capacity creates an increasing scope for unexpected and sometimes harmful behaviors. Especially with the release of ChatGPT, more and more people are now paying attention to AI safety issues. Such concerns came to a head when Prof.Hinton announced he was leaving google[2] and warns over dangers of chatbots. Just after many notable signatories signed an open letter asking 'all AI labs to immediately pause for at least 6 months'[3], there is another letter from CAIS with signatures from AI experts, journalists, policymakers and the public to state that mitigating the risk of extinction from AI should be a global priority alongside other societal-scale risks such as pandemics and nuclear war.[4]

One avenue for addressing these issues is Mechanistic Interpretability, attempting to reverse engineer the detailed computations performed by transformers, similar to how a programmer might try to reverse engineer complicated binaries into human-readable source code. If this were possible, it could potentially provide a more systematic approach to explaining current safety problems, identifying

[1] https://distill.pub/2020/circuits/

[2] https://www.nytimes.com/2023/05/01/technology/ai-google-chatbot-engineer-quits-hinton.html

[3] https://futureoflife.org/open-letter/pause-giant-ai-experiments/

[4] https://www.safe.ai/statement-on-ai-risk

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

new ones, and perhaps even anticipating the safety problems of powerful future models that have not yet been built.

2 Methodology

Given the incredible complexity and size of modern LLM, we think it's more fruitful to start with the simplest models that can give us some hints and guides which can then subsequently be applied to complex models. Therefore, in this paper, we will focus on "toy transformers" with some necessary simplifications . More specifically, we focus on "attention-only" transformers, which do not have MLP layers. Also, We do not consider biases, although a model with biases can actually be simulated without them by folding them into the weights and creating a dimension that is always one.

2.1 Residual Stream

Residual stream is the most fundamental concept in our framework. It's the sum of all previous outputs of layers of the model, and also the input to each next layer. The initial value of the residual stream is denoted x_0 in the below diagram, and x_i are later values of the residual stream. The residual stream has shape '$[batch, seq_{len}, d_{model}]$' (where d_{model} is the length of a single embedding vector).

2.2 QK&OV Circuits

To analyse how transformer works, We separate the whole working procedure into two independent circuits, the "Query-Key (QK) circuit" and "the "Output-Value (OV) circuit." We'll talk about them soon in the following sections. As an appetizer, you could think that QK circuit tells how much a given query token "wants" to attend to a given key token and OV circuit tells how a given token will affect the output logits if attended to.

2.3 Q&K&V Composition

To analyse 2-layer attention-only transformer, we introduce a new kind of composition when describing the QK&OV circuits in them. We will talk about it with more details in the corresponding section.

3 Highlights

First, instead of using the traditional paradigm to write the calculation process with matrix multiplication, the author use tensor product to transform the whole question into another equivalent but more easy-to-analyse form. This is also the key reason why we can then use QK&OV circuits separately in the following experiments and view the whole working procedure of attention head into two independent stages. Another creative idea I think is to consider the attention matrix of each head separately and multiply them with each one's own small weight matrix before adding them all into the residual stream, instead of following the previous way which will concatenate all the patterns together and then multiply with a big weight matrix. This helps us analyse each attention head independently which is of great help for us to find the induction head later. Also, the QK&OV circuits is brilliant idea which helps us easily separate the "attention part" and "output part" in the attention head when trying to interpret it.

Then using this framework, the paper shows us some interesting results: 1. Zero-layer transformers model bigram statistics. 2. One-layer attention-only transformers are an ensemble of bigram and "skip-trigram" which implements a kind of very simple in-context learning. 3. Two-layer attention-only transformers can implement much more complex algorithms using compositions of attention heads. Notably, two layer models use attention head composition to create "induction heads", a very general in-context learning algorithm.

4 Results

All the code including some additional attempts can be found in my github repo.[5]

4.1 Zero-Layer Transformers

Zero-layer transformer is an ultimate simplification we used to find out what a transformer is doing without MLP layer and attention heads. The claim is that such transformer which only involve the embedding and unembedding is approximating bigram frequencies.

In such case, all we have now is a linear map from the input token ‘t‘ to a probability distribution over the token following ‘t‘. The map can be written as $t \to t^T W_E W_U$, which indicates that the output only depend on the token that is one position before it and has no relation with all the earlier tokens. Therefore, the best we can do is to have this map approximate the true frequency of bigrams starting with ‘t‘, which appear in the training data. For example, the map should assign high probability to "Obama" if the current token is "Barack".

4.2 One-Layer Attention-Only Transformers

Adding a single attention layer into the zero-layer model, we can get a very interesting finding, where a large amount of attention heads is doing "copying information".

First, we decompose the residual path into "Direct Path" and "Attention Terms".

$$T = \mathrm{Id} \otimes W_U W_E \quad + \sum_{h \in H} A^h \otimes \left(W_U W_{OV}^h W_E \right)$$

As we said before, the left term represents the "bigram approximation" shown in zero-layer transformer. "Copying" ability is brought by the right term, i.e. the attention part. This is found out by decomposing the attention head into two seperate circuits named QK&OV circuits and visualizing the attention patterns.

I’ll use the following example to show what does "copying" mean. When an attention head observes a token (e.g. "R") which is very likely to be followed by some tokens(e.g. "alph" where "Ralph" is the name of a person), then this head would attend back to that token. This can found by looking at the attention pattern and the QK circuits. What’s more, after the head knows where to attend, the OV circuit, which decides how a token would change the destination logits if attended to, will directly modify the logits by assigning a large probability to that token. (e.g. the prediction would be "alph", which is simply copied from the previous token) More specifically, the OV circuit sets things up so that tokens, if attended to by the head, increase the probability of that token. Such ability can be seen as a very primitive version of "in-context" learning shown in LLM like GPT-4, and such attention head is kind of mimicing the "induction heads" we will talk about in the two-layer transformer.

The reason I call it "primitive" is that why a token attend to another is depending on learned statistics in hand like whether one token is indeed plausibly following another in the training data that the model has already seen. (i.e. similar to the bigram frequency in the zero-layer model, so it’s more like a look-up table instead of really "learning")

The author also gives an addition try which use eigenvectors and eigenvalues of OV&QK circuits to capture the copying phenomenon in one-layer models. I personally don’t fully understand that part so I won’t cover that here.

4.3 Two-layer Attention-Only Transformers

We have already introduced the mathematical framework to view a transformer into different paths and proved that an attention head can be anylysed using independent and separable circuits, where the OK circuit tells us how the attention pattern is computed and OV circuits tell us how a token can change the output logits if being attended to. Here we will still apply this framework but with a new technique introduced named "attention heads composition". Here I will only talk about K-composition which is the most common and important composition in toy transformers. To write the QK circuit of

[5] https://github.com/andyisokay/cs282-ML

a head in a two-layer model, remember that the input to the second-layer is the output of the first layer. Therefore, we can get how an attention head in the second layer modify the residual stream when its key is using the output of a head in the first layer while its query is using the original input token from the direct path (i.e. how can the head in the second layer modify the residual stream).

$$\text{K-Composition} = \sum_{h_k \in H_1} \text{Id} \otimes A^{h_k} \otimes \left(W_E^T W_{QK}^h W_{OV}^{h_k} W_E \right)$$

Besides K-Composition, we also have V-Composition and Q-Composition which is using the output from last layer as its query instead of key, and get its key from the direct path. Since K-Composition contributes the highest marginal contribution in our example, I won't spend more words on the other two.

With these attention heads composition, a two-layer attention-only transformer can implement much more complex algorithms. For example, we can find that some heads in the second layer begin to show the ability of "in-context" learning. We call them "induction heads".

4.4 Induction Heads & In-Context Learning

Induction heads can search over the context for previous examples of the present token and then do prediction based on that. This is just like how we human do when coping with text tasks and I think it's the most interesting ability shown in language models. I would put my experiment result here which indeed shows that some heads in the second layer are doing in-context learning.

Figure 1: "Induction"

Figure 2: The attention pattern of "induction head" (Head 4)

As shown in the above figures, we generate a random sequence and then repeat it once to let the model has somewhere to attend to and hence test if it can do in-context learning. More specifically, let's look at the purple square in figure one which denotes the current destination token. We can tell that, from the green highlights as well as the attention pattern from figure 2, it's exciting that "INC" is searching back and looking at the token "neat" which is just next to a previous occured "INC" which is very similar to how we human do when learning the context information. I would like to leave a very short comment on the weaker attention on the token "<|endoftext|>, it can be understanded that a way for transfomer to have a rest when there's nothing else necessary to attend to."

Moreover, by using random sequence, we can prevent model from simply memorizing bigram frequencies in the training data which is kind of cheating. With this interesting result, it shows that our two-layer transformer knows where there are similar cases while looking back from the current destination token. Moreover, we find that while some heads in the second layer are doing things like induction, some heads in the first layer are acting like "previous token follower" which means it always attend to the token that is one position before it.

First, let's denote the follower head in the first layer as H_1 and H_2 for the head in the second layer. Then, remember we said that K-Composition use the original token from the direct path as its query, and the output from a head in the first layer as its key. Therefore, suppose we have the sequence as "A, B, ... A" and we want to predict the next token with the query A. According to the induction behavior, H_2 will search back to get the context information of A. Since the only tool it can use is the key, i.e. the output from H_1, it's plausible for me that, H_1 should act like a "previous token follower" because by doing so, it can convey the information that "B follows A" to H_2 which will then make a prediction based on the context information it receives. As for the whole circuit that go through the

Figure 3: "previous token follower"

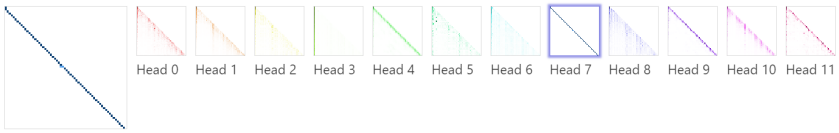

Figure 4: The attention pattern of "follower head" (Head 7)

two heads, we also have another name for it, "the induction circuit". As we can see in the figure 3 & 4, the empirical results where I generated some random tokens which is the same to that in figure 1 & 2 show that, each token is indeed keeping track to the token before it which agrees with our analysis under the framework.

5 Further Thoughts

Question 1 - why couldn't an induction head be formed in a one-layer model?

Answer - My intuition is that, since this would require a head which attends to a key position where there is a composite information from the previous layer, as we said before, the "previous token follower" which tell the induction head that "who is following me" through the K composition and the induction circuit. Therefore, this can only appear when there are more than two attention layers in transformer. With this mathematical framework, we can now think about the situation with more layers and analysis more interesting behaviors. There is some following papers that introduce a very exciting phenomenon named "Grokking" which appeared in very large language models. "Grokking"[6] is related to the "Emergent Ability" of LLMs which means that some ability suddenly show up as we make the model more complex. Neel, who creates the TransformerLens Library, has been doing a lot of interesting research on this topic recently based on the framework we introduced here.

Question 2 - Can we really take the residual stream as a "linear map"?

Answer - No. However, I believe the key point here is that, by leaving the mlp layer and layernorm, we indeed find some general circuits which can explain how a toy "transformer" works and also can be extended to a larger transformer at least to some extend. We hope this to be a good start point which has been proved to be after the following works published. BTW, although the softmax indeed makes attention pattern include effects from other tokens, if another key token has a high attention score, softmax inhibits this pair. However, this inhibition is symmetric across positions, so it cannot systematically favour the token "next" to the relevant one.

References

[1] Elhage, et al., "A Mathematical Framework for Transformer Circuits", Transformer Circuits Thread, 2021.

[6] https://arxiv.org/abs/2301.05217