Thesis

Albert Ramus Sidenius Garde (s183969)

March 12, 2024

Contents

Chapter 1

Introduction

Large Language Models (LLMs) based on the transformer architecture [Vaswani et al., 2023] have shown exceptional performance across a range of tasks, yet their complexity renders their inner workings opaque. This opacity challenges our ability to understand, trust, and safely deploy these models in real-world applications. To rectify this, the overlapping fields of Explainable AI (XAI) and Mechanistic Interpretability (MI) have emerged. The former focuses on developing methods to explain the predictions of LLMs, while the latter focuses on understanding the inner workings of LLMs. Transformer models contain both attention and multi-layer perceptron (MLP) layers, and the latter is the focus of this thesis. Most attempts at interpreting MLP neurons have focused on understanding the behaviour of individual neurons, but as demonstrated in Elhage et al. [2022], the behaviour of individual neurons often doesn't map onto human understandable concepts. Bricken et al. [2023] shows a possible way forward by suggesting the features of SAEs (Sparse Autoencoders) as alternative units of interpretability. This thesis aims to apply the N2G [Foote et al., 2023] method to the features of SAEs, with the goal of understanding the behaviour of these features and the potential for using this understanding to interpret the behaviour of LLMs. If SAEs truly do provide more interpretable features and N2G truly does provide a useful representation of feature behaviour, we would expect this to be reflected when comparing N2G graphs for individual neurons against those for features. This means that the results of this thesis will inform the usefulness of these two methods.

Info on what comparisons we expect to make.

Chapter 2

Theory

2.1 Preliminaries

Before we can dive into the specifics of the methods used in this thesis, we need to establish some basic concepts and terminology. Firstly, throughout this section, we assume there is some fixed transformer language model [Vaswani et al., 2023] which we want to interpret. We will refer to this as the *original language model.* We will also assume that we have a set of text samples which we use to train and test the methods. We will refer to these as the *dataset.*

Next we introduce the unit of interpretability, namely *features.* In the most general form, a feature is any function of the activations of a model, but in the context of this thesis we will focus on two specific types. These are individual neurons or activations which can be read directly from the model, and neurons in the hidden layer of a sparse autoencoder [Conmy et al., 2023]. We will generally refer to these as *neurons* and *features* or *features of an SAE* respectively.

Since we interpret features based on their value on various inputs, it is useful to have notation for this. Taking inspiration from Foote et al. [2023] but diverging significantly, we use the function

$$a \in \mathcal{F} \times \mathcal{T} \times \mathbb{N} \to \mathbb{R}$$

where $\mathcal{F}$ is the set of possible features and $\mathcal{T}$ is the set of possible token strings. a then takes a feature, a token string, and an index, and returns the value of the feature on the token string at the given index. For example, if f is the 423rd neuron in the second MLP layer of the `gpt2-small` model, s is the token string `"<|BOS|>","The"," quick"," brown"," fox"` then $a(f, s, 3)$ would be the value of the 423rd neuron on the token " `brown`" when `gpt2-small` processes the string "`The quick brown fox`".

2.2 MAS

The first method we will use is that of *maximum activating samples.* It consists of finding the samples which maximally activate a given feature. The hope is that these samples will give us some insight into what the feature is doing. Mathematically this can be written as

$$\underset{\substack{s \in \mathcal{D} \\ \text{top } k}}{\arg\max} \left(\max_{i \leq |s|} \left(a(f, s, i) \right) \right)$$

for some feature $f \in \mathcal{F}$ and dataset $\mathcal{D} \subseteq \mathcal{T}$. This method has faced criticism [Bolukbasi et al., 2021] and should probably not be used in isolation, but it can be a useful starting point for understanding the behaviour of a feature.

Calculation of the maximum activating samples is rather simple. Simply iterate over the dataset and for each of the n features of interest keep track of the k samples which maximally activate the feature. Here "keep track" means storing the token that causes the activation and surrounding c tokens of context along with the activations on that context. This can be done for the entire model with a single pass over the dataset. The more of the dataset is used, the "better" the found samples will be, but the computational cost will of course increase linearly.

2.3 Neuron2Graph

The second method of interest is the *Neuron2Graph* as described in Foote et al. [2023]. It attempts to build a graph model of the behaviours of a feature by finding a set of patterns which activate the feature. Each pattern consists of a string of tokens (an n-gram) with the possibility of a special ignore token which signal that the token at that position does not matter. It does this on the basis of maximum activating samples, s so it is dependent on that method and can be argued to share some of its weaknesses.

though it doesn't have to

To run this method for a particular feature $f \in \mathcal{F}$, we need a set of highly activating samples $\mathcal{S} \subseteq \mathcal{T}$. For sample $s \in \mathcal{S}$, we identify a *pivot token* e, which is the most activating token in the sample, and perform 3 steps: pruning, saliencey detection, and augmentation.

2.3.1 Pruning

For a token string $s \in \mathcal{S}$, and pivot token e, pruning consists of finding the smallest substring of s that ends in e and still sufficiently activates f. What "sufficiently activates" means is a parameter of the method, but in the original paper it is defined as causing an activation at least half of original activation. This removes context that is irrelevant to the activation of f. We call the pruned string s'

2.3.2 Saliency detection

Here we find the most important tokens in the pruned string s'. This is done by replacing each token in the pruned string with a padding token and finding the change in activation. If the change is large, the token is important. How large the change needs to be is another parameter of the model. Once this step is done, we have a set B of important tokens in s'.

2.3.3 Augmentation

Given a pruned string s' and a set of important tokens B in that string, augmentation is the process of finding nearby nearby strings that activate f similarly to s'. To do this, we replace each $b \in B$ with other "similar" tokens and see whether the resulting string activates f sufficiently. What counts as sufficient is yet another parameter, while "similar" tokens are found using a helper model (`distilbert-base-uncased` in the original paper) that is asked to predict replacements for b given the rest of s' as context. All alternative strings that activates f sufficiently are stored.

2.3.4 Graph building

After performing the 3 previous steps on all strings in $\mathcal{S}$, We have a set $\mathcal{S}' \subseteq \mathcal{T}$ of pruned and augmented strings that all activate f highly. In order to make predictions we must build a model from these strings. This is done by creating a trie T by working backwards through each string. The first nodes after the root of T are the activating tokens in the strings of S'. The rest of the nodes are the tokens in the strings of S', so that each path from the root to a leaf represents a string in S'. At the end of each of these paths through T, we add an end node storing the activation of f on the represented string. To predict the activation of f on the last token of a new string s, we start from the root of T and the last token of s. We then traverse the trie, going backwards through s following any node that matches the current token of s, with the special ignore tokens matching any token. If we reach an end node, we return the activation stored there. If at some point no node matches the current token of s, we return 0.

This gives us a quantative measure of how good a model of the feature the graph is. It also allows us to create a visual representation of the feature behaviour. To do this, we create a new graph from T where all ignore nodes are removed, and nodes representing the same token on the same layer are collapsed. We refer to both this representation and the original trie T as the *feature graph*.

Bibliography

Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. An Interpretability Illusion for BERT, April 2021. URL http://arxiv.org/abs/2104.07143. arXiv:2104.07143 [cs].

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. *Transformer Circuits Thread*, 2023.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards Automated Circuit Discovery for Mechanistic Interpretability, October 2023. URL http://arxiv.org/abs/2304.14997. arXiv:2304.14997 [cs].

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.

Alex Foote, Neel Nanda, Esben Kran, Ioannis Konstas, Shay Cohen, and Fazl Barez. Neuron to Graph: Interpreting Language Model Neurons at Scale, May 2023. URL http://arxiv.org/abs/2305.19911. arXiv:2305.19911 [cs].

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, August 2023. URL http://arxiv.org/abs/1706.03762. arXiv:1706.03762 [cs].