基于SAE的金融LLM可解释性研究

# 技术路线图



# 语料库的选择

## 东方财富

FinGPT，中文语料库的使用，[open-compass/OpenFinData (github.com)](https://github.com/open-compass/OpenFinData)

OpenFinData是由东方财富与上海人工智能实验室联合发布的开源金融评测数据集。该数据集代表了最真实的产业场景需求，是目前场景最全、专业性最深的金融评测数据集。它基于东方财富实际金融业务的多样化丰富场景，旨在为金融科技领域的研究者和开发者提供一个高质量的数据资源。

优点：数据集模块多，缺点：数量少，为json文件，压缩大小为400KB，总共为1500条数据记录

## 轩辕开源

[Duxiaoman-DI/FinCorpus · Datasets at Hugging Face](https://huggingface.co/datasets/Duxiaoman-DI/FinCorpus)

中文金融数据集FinCorpus，包含上市公司公告、金融资讯、金融新闻和金融试题四个模块



## 最终选择

Anthropic在进行大模型可解释性分析时，使用The Pile数据集，并使用四千万的tokens来生成八十亿的数据集。所有在金融大模型领域进行可解释时，其金融语料库的内容需要丰富并具大规模的特点。

其中轩辕大模型开源的FinCorpus语料库，其数据丰富，包括上市公司公告、金融资讯、金融新闻以及金融试题，满足构建用于SAE训练的数据集的要求。

# 金融大模型的选择

## 基于GPT2微调

MLP层神经元为768个。

## Fingpt

基于ChatGLM2-6B进行微调。其MLP层神经元为4096个。

推理demo：[FinGPT/fingpt/FinGPT\_Sentiment\_Analysis\_v1/FinGPT\_v1.0/inferencing/infer.ipynb at master · AI4Finance-Foundation/FinGPT (github.com)](https://github.com/AI4Finance-Foundation/FinGPT/blob/master/fingpt/FinGPT_Sentiment_Analysis_v1/FinGPT_v1.0/inferencing/infer.ipynb)

## 轩辕

从头开始训练，和Fingpt类似的结构。不同在于fingpt-7B是在chatGLM/Llama2上微调，而轩辕-6B是采用类LLaMA架构，从零开始进行预训练的金融大模型。

## 最终模型的选取

需要考虑进行大模型可解释性分析时，模型结构对显存占用的消耗。

# SAE数据集的生成

## 使用HookedTransformer加载金融大模型

在SAE\_lens包中，是使用的Transformer\_lens包中的HookedTransformer来读取常见的预训练模型。而金融大模型不能直接通过HookedTransformer读取。

例如轩辕-6B模型，其使用Transformers包中的LlamaForCausalLM来读取其预训练模型，需要将LLaMA的权重转为HuggingFace的权重，并使用HookedTransformer来读取。

## 相关参考代码：

### 模型权重转换

[LLaMA.ipynb - Colab (google.com)](https://colab.research.google.com/github/TransformerLensOrg/TransformerLens/blob/main/demos/LLaMA.ipynb#scrollTo=sMkmqMjkVdMq)

转为LLaMA的权重为HuggingFace的权重，并使用HookedTransformer来读取。

# SAE用于可解释性分析



## 占用显存分析

### SAE模型占用显存



M=神经元数量，N=特征数量，F=数据精度。

总的显存占用=(M \* B \* F + M \* N \* F \* 2 + M \* B \* F + N \* B \* F + M \* F + N \* F)/(1024\*\*3)GB

之下表格为B=1024，F=4时的情况。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| M/N | 2\*\*12(4K) | 2\*\*15(32K) | 2\*\*17(131K) | 2\*\*20(1M) | 2\*\*22(4M) | 2\*\*25(34M) |
| 512 | 0.03 | 0.25 | 1.00 | 8.00 |  |  |
| 768 | 0.04 | 0.31 | 1.25 | 10.00 | 40.02 | 320.13 |
| 1024 | 0.05 | 0.38 | 1.50 | 12.01 | 48.02 | 384.13 |
| 4096 | \ | 1.15 | 4.53 | 36.03 | 144.04 | 1152.15 |

### 数据shuffle占用显存

sae\_lens的shuffle机制，并不是对全部数据进行统一shuffle，而是分批载入的，二合一后分为二

buffer\_size=store\_batch\_size\_prompts \* (n\_batches\_in\_buffer / 2) \* context\_size \* d\_model

buffer\_count=9+5\*2。

总的显存占用=buffer\_count \* buffer\_size





### 实际消耗

n\_batches\_in\_buffer简称为nbib，expansion\_factor简称为ef

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| model | 大小 | d\_model | ef | 激活 | dtype | batchsize | nbib | GPU |
| tiny-stories-1L-21M | 13M | 1024 | 16 | Relu | float32 | 4096 | 64 | 13.2 |
| 13M | 1024 | 16 | topk | float32 | 4096 | 64 | 13.6 |
| 13M | 1024 | 16 | Relu | float16 | 4096 | 64 |  |
| 13M | 1024 | 16 | topk | float16 | 4096 | 64 | 9.7 |
| 13M | 1024 | 16 | topk | float16 会导致loss变为nan | 1024 | 64 | 9.8 |
| 13M | 1024 | 16 | topk | float32 | 4096 | 32 | 9.6 |
| gpt2-small | 85M | 768 | 16 | topk | float32 | 4096 | 64 | 18.2 |
| 85M | 768 | 16 | topk | float32 | 4096 | 32 | 14.1 |
| 85M | 768 | 16 | topk | float32 | 1024 | 64 | 18.3 |

## SAE的变体

### gated SAE

### TopK SAE

## SAE评价指标分析

如何评估SAE，目前存在L0和Loss Recovered两个指标 。L0是SAE编码中非零元素的平均数。Loss Recovered是用重建的激活替换GPT或者其他大模型的原始激活并测量不完美重建的额外损失。这两个指标之间通常存在跷跷板效应，需要进行权衡。毕竟SAE会选择降低重建精度以增加稀疏性。

### Anthropic对特征的评价分析[1] ：

1. 手动检查：这些特征是否看起来可解释？
2. 特征密度：我们发现“实时”特征的数量和它们触发的标记百分比是极其有用的指南。（详情请参阅附录。）
3. 重构损失：自动编码器如何很好地重构 MLP 激活？最终目标是解释 MLP 层的功能，因此 MSE 损失应该很低。
4. 玩具模型：拥有玩具模型，可以在其中了解基本事实，从而可以清晰地评估自动编码器的性能。

基于以上分析，结合目前开源工具SAE\_lens，本研究拟采用以下几种评估指标对sae训练性能进行评估。（get\_downstream\_reconstruction\_metrics函数和get\_sparsity\_and\_variance\_metrics函数）

### SAE模型在重建任务中的指标

在金融大模型中使用自编码器（SAE）来替换某一层的激活，并通过前向钩子（forward hook）进行重建。

通过大模型计算原始 logits （logits：神经网络输出的未经过激活函数的原始分数）和交叉熵损失，然后定义一个标准的替换钩子函数，用于在前向传播过程中替换某一层的激活，最后使用这个钩子函数运行大模型，获取重构的 logits 和交叉熵损失，分析重构logits和原始logits之间的kl散度和交叉熵损失。

（1）KL散度

i.对原始 logits和重构logits进行softmax变换，得到原始分布original\_probs和重构分布new\_probs。

ii. 计算原始分布和重构分布的对数概率log\_original\_probs和log\_new\_probs

iii.计算kl散度。kl\_div = original\_probs \* (log\_original\_probs - log\_new\_probs)

（2）交叉熵损失（ce loss）

计算加入sae替换后的重构损失ce\_loss\_with\_sae和原始损失original\_ce\_loss。

### L2范数

L2范数由以下公式得到：

其中，表示第i个样本的输入激活向量。

该研究拟计算输入sae和输出激活的L2范数以及他们的比值作为评估指标。

### 稀疏度度量

（1）L0范数

L0范数用于表示非零特征的个数。

其中，表示第i个样本的第j个特征激活，是指示函数，当条件为真时取值为1，否则取值为0。即，L0范数表示非零特征的个数。

（2）L1范数

其中，表示第i个样本的第j个特征激活。即，L1范数表示特征激活的绝对值之和。

### 方差

（1）解释方差。

解释方差由以下公式得到。

其中，表示第i个样本的第j个输入激活，表示第i个样本的第j个输出激活, 表示第j个特征激活在所有样本中的平均值。

（2）均方误差

均方方差由以下公式得到。

其中，表示第i个样本的第j个输入激活，表示第i个样本的第j个输出激活。

**特征可解释性**

[sparse\_autoencoder/sparse\_autoencoder/explanations.py at main · openai/sparse\_autoencoder (github.com)](https://github.com/openai/sparse_autoencoder/blob/main/sparse_autoencoder/explanations.py) 提供了一种解释模型激活值的方法，使用 Trie 数据结构高效存储和查询模式，并通过显著性值计算来确定每个 token 的重要性。

核心功能说明

Trie 数据结构：用于高效存储和查询模式。Trie 的每个节点代表一个 token 或者特殊标识符（例如 \_ANY\_TOKEN），叶节点存储激活（\_SALIENCY\_KEY）。

激活值计算与存储：通过模型函数计算每个 token 序列的激活值，并存储在 Trie 中。显著性值计算：通过替换 token 并计算变化后的激活值来确定每个 token 的显著性。最小子序列确定：从 token 序列末尾开始，找到能解释激活值的最小子序列。

**重建可解释性**

如果我们只使用模型中我们可以解释的部分，我们会牺牲多少性能？

这建议将我们的下游损失和解释指标结合起来，通过使用我们的解释来模拟自动编码器潜伏，然后在解码后检查下游损失。该指标还有一个优点，即它以原则性的方式同时重视召回率和精确度，并且还更重视激活更密集的潜在变量的召回率。

# 特征可视化

## 相关参考代码

### github开源代码1：

[GitHub - callummcdougall/sae\_vis: Create feature-centric and prompt-centric visualizations for sparse autoencoders (like those from Anthropic&apos;s published research).](https://github.com/callummcdougall/sae_vis)

[文件夹 - Google 云端硬盘](https://drive.google.com/drive/folders/1sAF3Yv6NjVSjo4wu2Tmu8kMh8it6vhIb)

包含feature-centric的可视化，复现Anthropic的可视化，即我们查看一个特定feature并查看哪些tokens在该功能上触发最强。

以及以prompt-centric的可视化，查看一次特定prompt并根据各种不同的指标查看哪些feature在该prompt上触发最强。

### github开源代码2：

[sparse\_autoencoder/sae-viewer at main · openai/sparse\_autoencoder (github.com)](https://github.com/openai/sparse_autoencoder/tree/main/sae-viewer)

# 引导

## 相关参考代码

### Github开源代码1：

[SAELens/tutorials/using\_an\_sae\_as\_a\_steering\_vector.ipynb at main · jbloomAus/SAELens (github.com)](https://github.com/jbloomAus/SAELens/blob/main/tutorials/using_an_sae_as_a_steering_vector.ipynb)

# 参考文献

1. Bricken T, Templeton A, Batson J, et al. Towards monosemanticity: Decomposing language models with dictionary learning[J]. Transformer Circuits Thread, 2023, 2.