ML-HW-3

Q1：Chat template comparison

实验现象：

```
========== 存在chat template ==========
 <bos><start_of_turn>user
请说明监督学习和无监督学习之间的主要区别，要求用200个字作答。<end_of_turn
<start_of_turn>model

========== Output ==========
 监督学习和无监督学习是机器学习中两种不同的学习方法。

**监督学习**：
* **输入：**包含标签数据，即已知结果。
* **目标：**学习模型，以便在新的输入数据上预测结果。
* **应用：**分类、回归、目标识别等。
* **例子：**图像分类、文本分类、预测房价。

**无监督学习**：
* **输入：**没有标签数据，只能从数据本身中学习。
* **目标：**发现数据中的模式和结构，例如聚类、降维、异常检测等。
* **应用：**聚类分析、降维、异常检测等。

简单来说，监督学习需要已知结果，而无监督学习则不需要。
========== Coherence Score：8.2235 ==========


========== 不存在chat template ==========
 请说明监督学习和无监督学习之间的主要区别，要求用200个字作答。
========== Output ==========
 **监督学习**：利用已有的 labeled 数据，训练模型，预测未知数据。例如，图像分
**无监督学习**：利用未标记的数据，寻找数据中的结构和模式。例如，聚类、降维

总结来说，监督学习需要已有的标签，而无监督学习不需要标签。监督学习是预测任
========== Coherence Score：7.4847 ==========
```

========== with template ==========
<bos><start_of_turn>user
Please tell me about the key differences between supervised learning and uns
<start_of_turn>model

========== Output ==========
Supervised and unsupervised learning are two fundamental types of machine

Supervised learning involves training a model on labeled data, where eacl

Unsupervised learning, on the other hand, uses unlabeled data. The mode

Key differences:

* **Labeling:** Supervised uses labeled data, unsupervised doesn't.
* **Goal:** Supervised aims to predict outputs, unsupervised aims to discover
* **Applications:** Supervised: classification, regression, supervised tasks; U

Both types are powerful tools with their own strengths and weaknesses, and t
========== Coherence Score : 6.0734 ==========

========== without template ==========
Please tell me about the key differences between supervised learning and un
========== Output ==========
Supervised Learning:

* **Labeled data:** Uses data with known outputs (labels) to train models.
* **Goal:** Predict the output for new, unseen data.
* **Examples:** Image classification, spam detection, predicting house prices

Unsupervised Learning:

* **Unlabeled data:** Uses data without known outputs to discover patterns.
* **Goal:** Explore data, identify clusters, or reduce dimensionality.
* **Examples:** Customer segmentation, anomaly detection, dimensionality r

```
In essence, supervised learning learns from labeled examples to make predict
========== Coherence Score : 4.2210 ==========
```

实验观察：

1. 存在模板和不存在模板的区别非常明显，存在模板时得分显著更高，回答也更加切合题目（如存在模板时明确指出了 Key differences）
2. 中文回答和英文回答存在评分差异，中文的评分要显著高于英文。这个点估计是评分的这个cross-encoder/ms-marco-MiniLM-L-6-v2本身对中文的评判不是那么专业，并且中文在简短信息的表达上要比英文有优势所导致的。其实光看回答切合程度来说，英文的要更好一些。

Q2: Multi-turn Conversations

```
You: Could you continue and name yet another color from the rainbow?
=== Prompt with chat template format inputted to the model on round 3 ===
<bos><start_of_turn>user
Name a color in a rainbow, please just answer in a word without any emoji.<er
<start_of_turn>model
Indigo<end_of_turn>
<start_of_turn>user
That's great! Now, could you tell me another color that I can find in a rainbow?
<start_of_turn>model
Orange<end_of_turn>
<start_of_turn>user
Could you continue and name yet another color from the rainbow?<end_of_tur
<start_of_turn>model
```

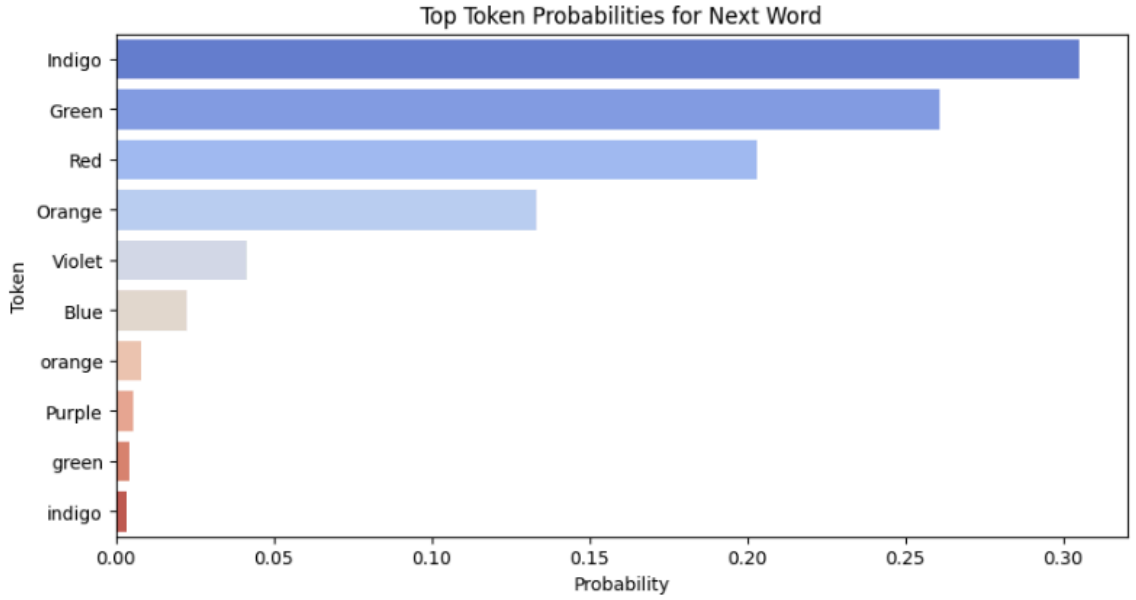
相比于这个多轮对话的问题，我个人发现一个更有趣的现象，也就是**带引号与否引发的输出差异**。这里如果直接输入问题【Name a color in a rainbow, please just answer in a word without any emoji.】那么可以看到第一张图得到的概率分布中，indigo的概率最高。

```
Name a color in a rainbow, please just answer in a word without any emoji.<end_of_turn>
<start_of_turn>model

================================================
<ipython-input-24-262378176447>:44: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hu

  sns.barplot(x=top_probs, y=top_tokens, palette="coolwarm")
```



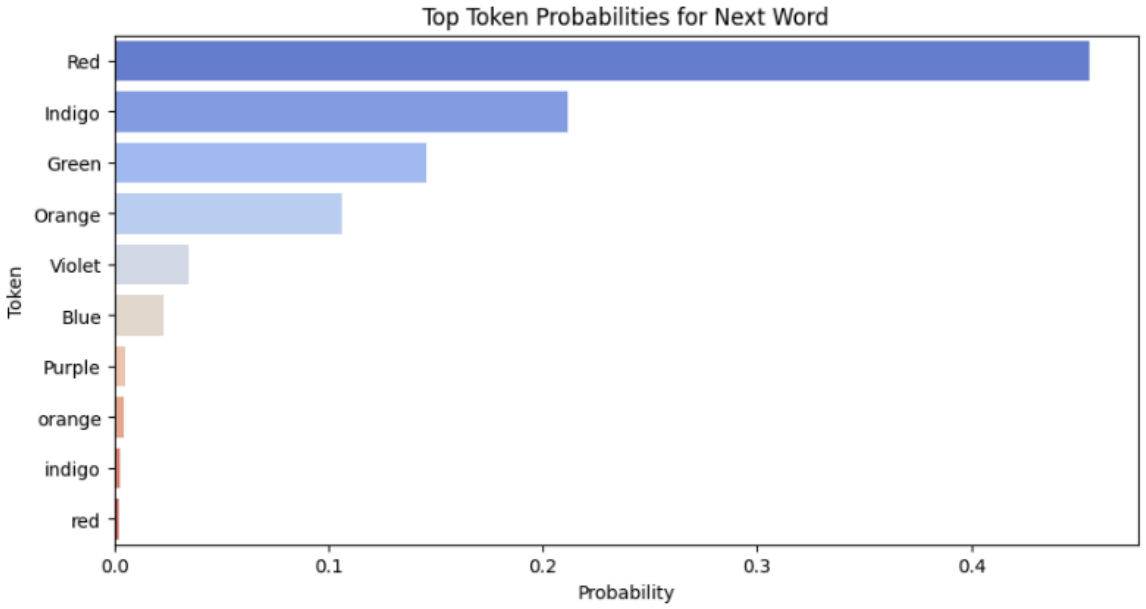
有趣的是，如果你输入的问题是【“Name a color in a rainbow, please just answer in a word without any emoji.”】，那么这个问题的概率分布发生了非常大的改变，不仅是red的概率变成了最高，下面各项的概率也发生了改变。

```
“Name a color in a rainbow, please just answer in a word without any emoji.”<end_of_turn>
<start_of_turn>model

================================================
<ipython-input-22-262378176447>:44: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to

  sns.barplot(x=top_probs, y=top_tokens, palette="coolwarm")
```



需要注意的是这里的随机性是关闭的，多次实验也是同样的结果。

在这里应该就能理解为什么需要使用 patchscore 的方法来研究网络对某一个token的理解了。连这种符号都能显著改变LLM对一个问题的回答，那么你直接通过修改token的形式来观察网络对这个token的反应，那肯定是不行的。

Q3: Tokenization

实验结果：

```
Token: I, token index: 235285
Token: _love, token index: 2182
Token: _taking, token index: 4998
Token: _a, token index: 476
Token: _Machine, token index: 13403
Token: _Learning, token index: 14715
Token: _course, token index: 3205
Token: _by, token index: 731
Token: _Professor, token index: 11325
Token: _Hung, token index: 18809
```

```
Token: -, token index: 235290
Token: yi, token index: 12636
Token: _Lee, token index: 9201
Token: ,, token index: 235269
Token: _What, token index: 2439
Token: _about, token index: 1105
Token: _you, token index: 692
Token: ?, token index: 235336
```

这个没有什么特别值得深入说的，这个应该是 embedding 的第一步，先把自然语言转换为 token_id，再把token_id转换为vector.

一个简便的可视化网站是：https://platform.openai.com/tokenizer

Q4： Autoregressive Generation

```
prompt = f"Generate a paraphrase of the sentence 'Professor Hung-yi Lee is
```

采样方法	参数数值	self-BLEU评分
top-k	k = 2	0.2975
top-p	p = 0.6	0.4596
top-k	k = 5	0.1190
top-p	p = 0.99	0.0864

self-BLEU 衡量了一组文本的多样性。分值越低，代表其文本的多样性越高。

top-k 和 top-p 代表两种不同的采样方式。这个也解释了当时第2次作业中，temperature 的真正作用机制。

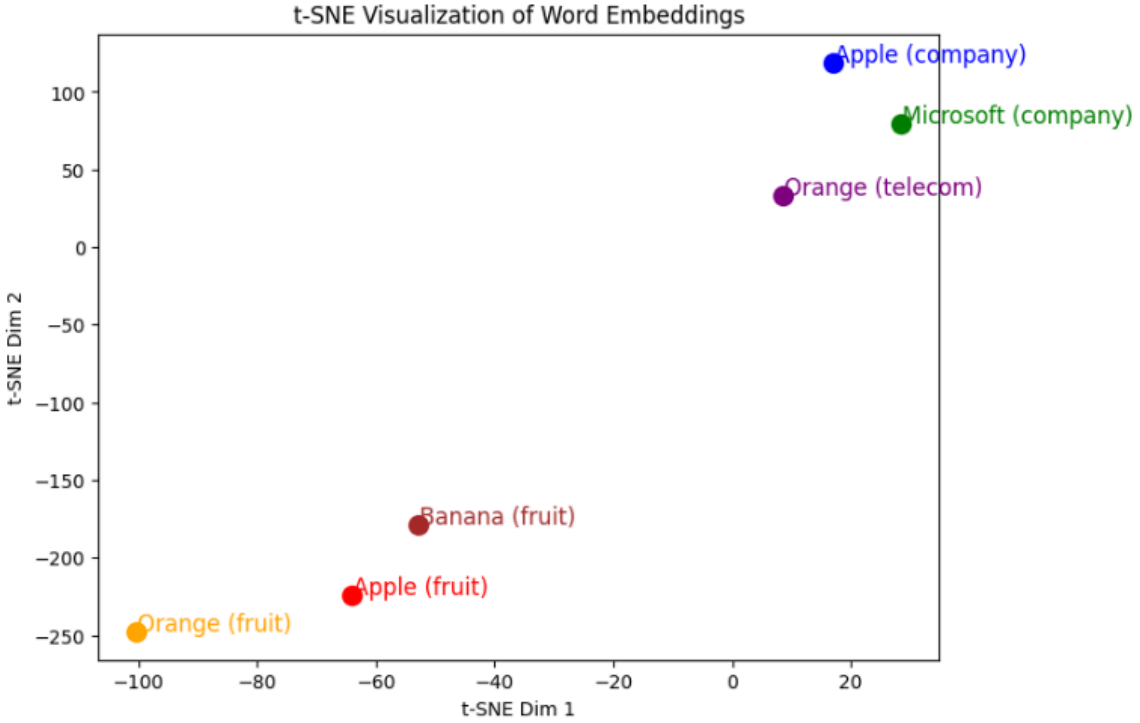
top-k 从概率排名前k个的token中选择输出；

top-p 则是从概率最高的token开始选取，从高往低把token的概率全部加和起来，直到超过p值时，其中的所有token就是待选集；

可以看到，k 和 p 的值越高，多样性就越丰富。

t-SNE

```
sentences = [
    "I ate a fresh apple.",  # Apple (fruit)
    "Apple released the new iPhone.",  # Apple (company)
    "I peeled an orange and ate it.",  # Orange (fruit)
    "The Orange network has great coverage.",  # Orange (telecom)
    "Microsoft announced a new update.",  # Microsoft (company)
    "Banana is my favorite fruit.",  # Banana (fruit)
]
```



这是一个很有趣的实验。我们刚刚看过LLM会把句子中的词汇拆解为Token，但是一些Token在不同句子中意义是不同的，这个实验就观察了LLM对这个句子的“理解”。

正如课程中所说，LLM对某个Token的理解实际上是一个layer中neuron的分布情况。那么这个实验的基本原理就是提取了这个句子输入后，该模型最后一个layer的embedding情况。

但是还有一个问题是，layer的维数是非常高的，直接可视化是不可行的，因此**引入了一个叫做t-SNE的模块，这个模块能够将高维信息的相关性保留至低维信息**。

于是我们能够看到，表示【科技公司】这一概念的apple和微软、orange是非常相关的。这体现了“LLM对某个Token的**理解**实际上是一个**layer中neuron的分布情况**”这一观点。

Attention Weight

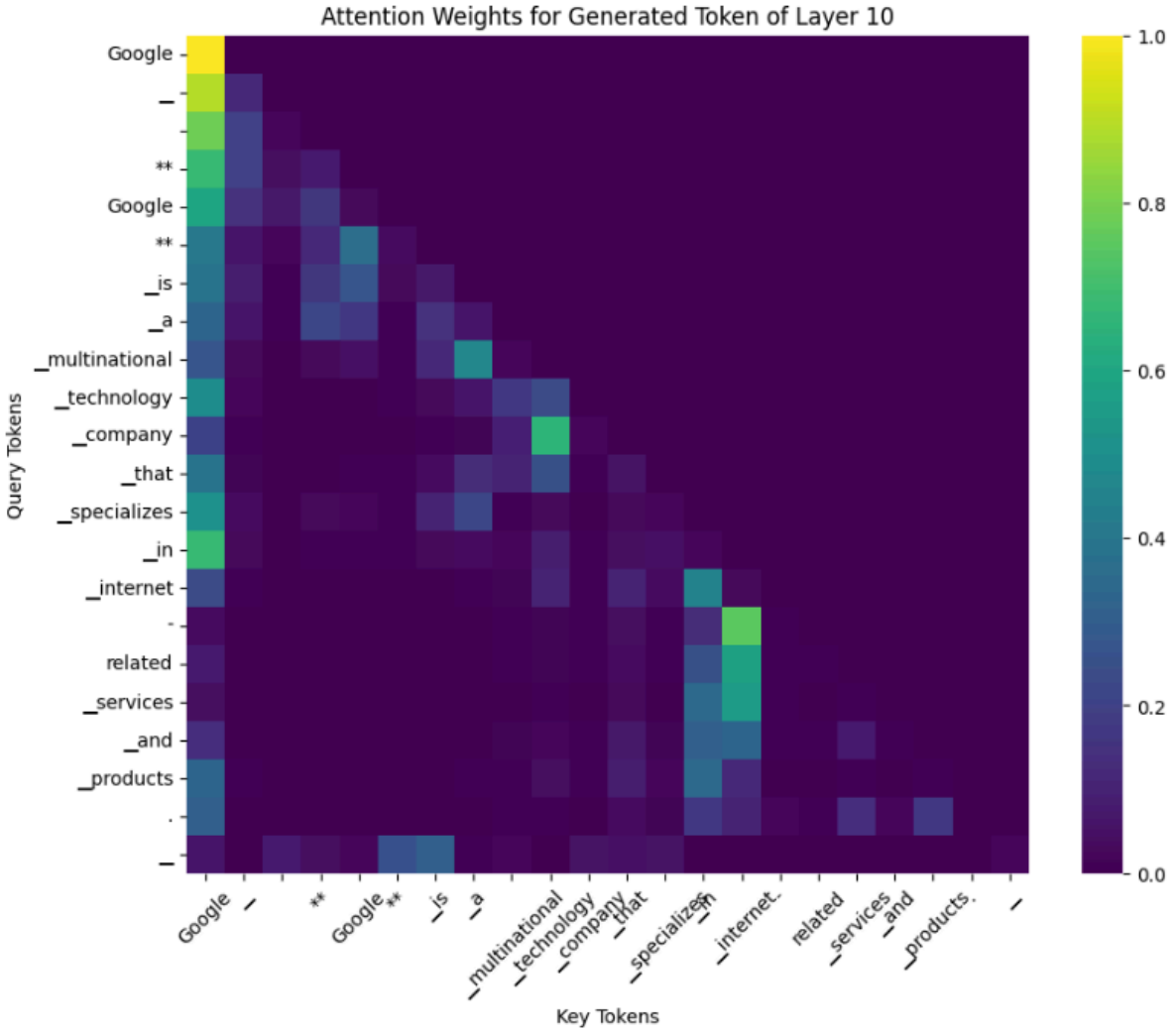
这个实验中可以看到不同深度的层级中，self-attention的三个关键变量的变化情况。

query tokens：表示当时正在处理的token；

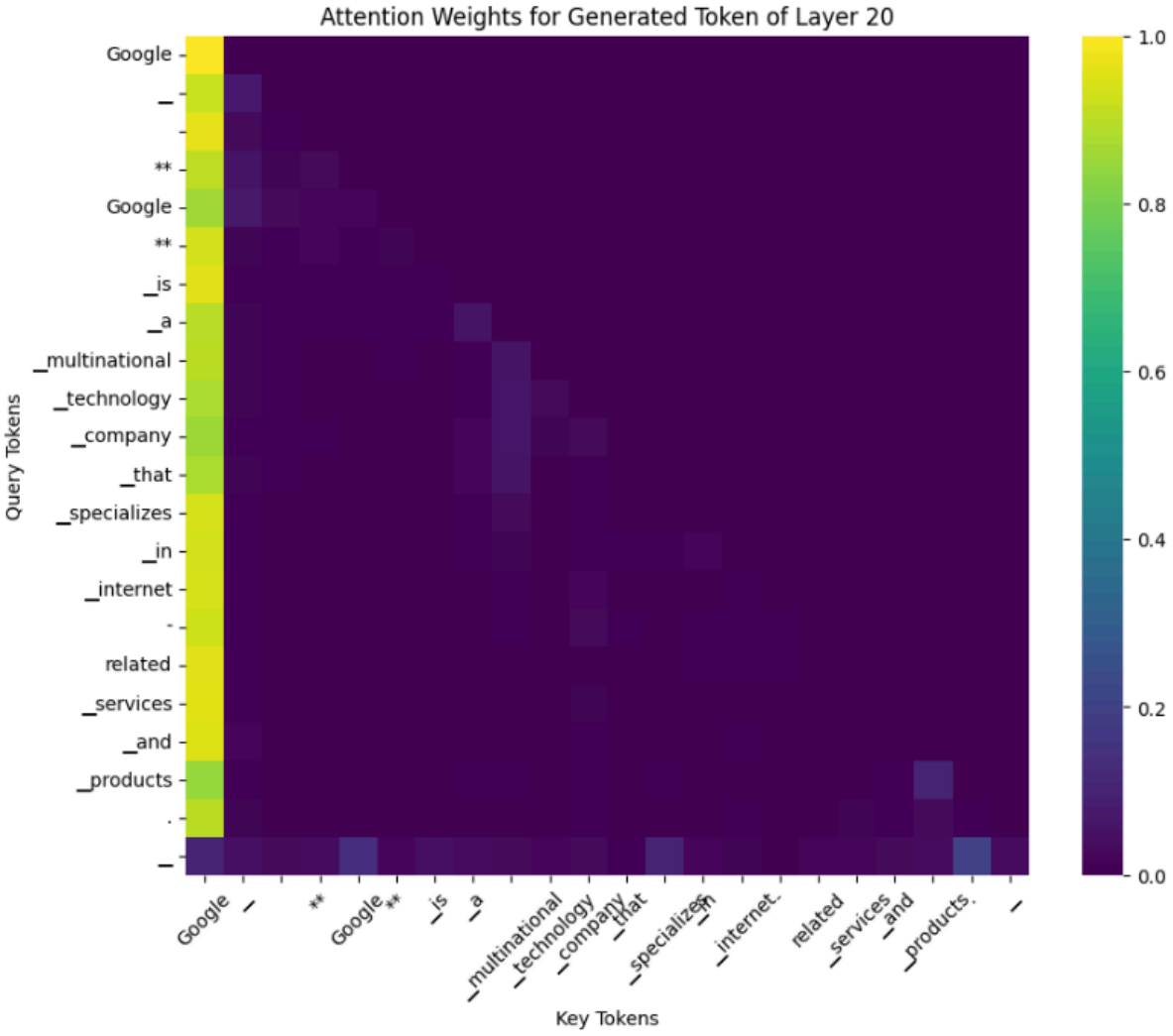
key tokens：当前token对其他token的关注程度；

以及右侧的attention数值。

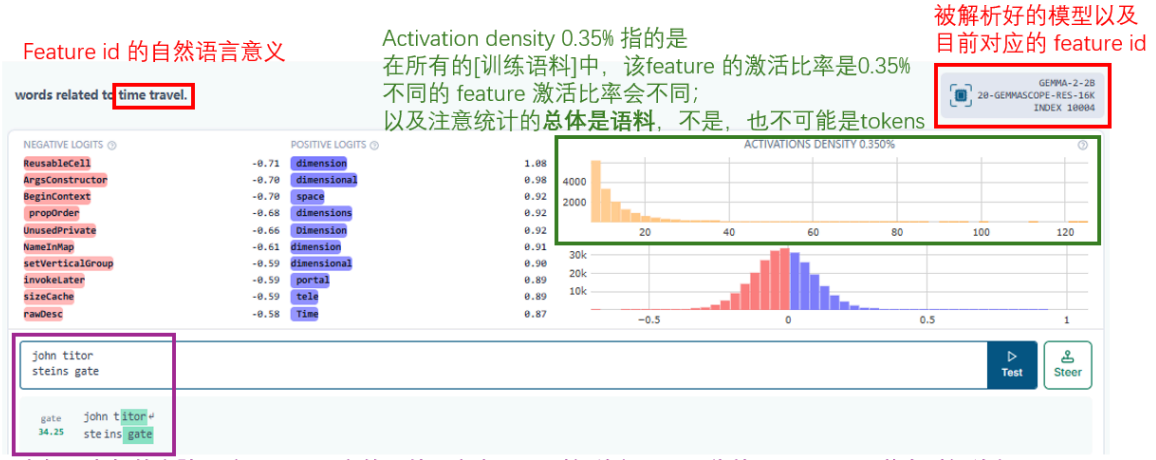
例如layer10的时候，最明显的是 company 这个tokens对multinational这个词的关注程度是最高的。这是因为 multinational 这个token经常和company一起出现。



而到了layer20的时候，所有的tokens都对Google这个词的关注达到了最高，也就是这句话的所有词汇都是围绕着google这个词展开的，也就是我们最终要达到的效果。



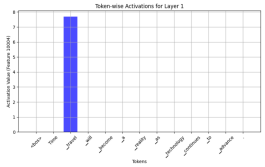
基于SAE的 activation score



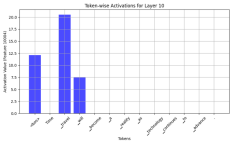
一个相当有趣的实验，该feature对应的自然语言意义是 时间旅行，如果你输入john titor (著名时间旅行meme) 和 steins gate (著名时间旅行动漫) 则会有 activation 值。
并且最重要的是单独的 titor 和 单独的 gate 是没有 activation 值的。
从技术上来说，我们不可能统计所有 token，尤其是不可能统计他们的 permutation，然而通过引入功能向量这种概念，我们得以通过观察neuron的激活组合来找到其背后对应的自然语言意义。

某一层的 Activation Distribution

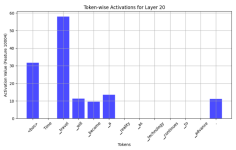
对于一句话”Time travel will become a reality as technology continues to advance.”我们逐层观察这句话中每个token对于时间旅行feature的activation distribution。



层数1



层数10



层数20

层数25

这个现象和self-attention那个章节中，随着层数的深入，所有词汇针对于中心词的key值将逐渐变高的现象是完全一致的。在这里，随着层数的深入，所有token针对特定feature的activation值逐渐变高。