

调查和玩弄“转向矢量”帖子（论文） #1460

🔓 打开

Azeirah 打开了本期 2 days ago · 7 评论

译

阿泽拉评论 2 days ago · 编辑

我刚刚读了这篇[最近发布的关于转向矢量概念](#)的文章。

什么？

转向向量的想法是将一些预先计算的数据添加到您的推理中，以“引导”模型朝某个方向发展。IE，给它一定的“情绪”或“风格”。例如，您可以添加转向矢量“爱”，使您的LLM提供更多爱的输出。

Steering vector: "I talk about weddings constantly" - "I do not talk about weddings constantly" before attention layer 20 with coefficient +4^[16]

例：

| Unsteered completions | Steered completions |
|--|--|
| <p>I went up to my friend and said, "I'm sorry, I can't help you."</p> <p>"No," he said. "You're not."</p> <p>"Why not?" I asked. "Why do you think I can't help you?"</p> <p>He looked at</p> | <p>I went up to my friend and said, "I'm going to talk about the wedding in this episode of Wedding Season. I think it's a really good episode. It's about how you're supposed to talk about weddings."</p> <p>And I got a lot of people</p> |

更多细节

简而言之，转向矢量是特定层提示输出的快照。因此，例如，如果您提示“我喜欢狗”，则可以通过将网络的输出存储在您选择的层（例如第 2 层或第 10 层）来获得转向矢量。

使用转向矢量，您可以更改提示的“方向”。通过在后面的提示中添加转向向量，可以使模型更有可能输出与转向向量相关的内容，即，包括他们对狗的热爱。

当你提示“我最喜欢的动物是.....”时，你会得到各种各样的答案。狗是可能的，但猫，鸟或其他常见的家庭宠物也是如此。当你添加转向矢量时，它几乎可以保证输出它喜欢狗。

其效果类似，但不等同于直接向提示中添加其他令牌上下文。

论文中还有很多细节：

- 您可以在向量上使用（线性）数学。例如，如果你想让LLM更有可能谈论狗，你可以将狗的转向向量乘以一个因子。你也可以把它乘以0.5，让它更有可能谈论狗。

- 如果同时使用加法和减法，转向矢量效果最好，即：转向矢量=“爱”-“恨”
- 并非所有转向矢量都能按预期工作，例如“爱”-“恨”不能很好地工作，而“爱”-“恨”则能很好地工作。

潜在应用及研究方向



- 摆脱chatgpt训练模型中的“作为AI语言模型”结果。
- 微调的低成本替代方案
- 引导它走向以某些语言或格式（IE，JSON，法语等）说话的方向。作者无法使用它来让模型说法语，但这很可能是可能的。
- 也许可以使用它使LLM更轻松地遵循langchain提示的说明？IE，使骆马不太可能以对话方式与您交谈，而只需提供简单格式的输出，而没有对话绒毛。
- 提高性能：与其在提示中嵌入“友善和乐于助人”（这需要几个令牌），不如简单地添加一个执行相同任务的转向矢量
- 它可能充当提示模板的保存点？例如，如果您采用提示模板“您是一个有用的聊天机器人，它.....bla bla bla”并将其用作转向向量，您基本上可以强制LLM进入已经执行了该部分提示的状态。

我认为这里还有很多。



 10

 2

  SlyEcho 提到了这个问题 yesterday

[研究]转向矢量 #1472

 草稿

zrthxn 评论 yesterday

这是一个超级有趣的想法！我只是想澄清一下我是否正确理解了这一点。
所以，假设对于像“狗没事”这样的提示，你要做的是从（比如）第 12 个注意力层中获取上下文化的隐藏状态，并添加诸如“爱”之类的单词表示/嵌入。然后，结果输出类似于“狗真棒！**这是对的吗？**”

Also, do you think the reason this works is that word representations are additive? Like for example, a common one I remember is . And so when you add the word vector for "Love" in the above example, it sort of steers the model towards love. "king" - "man" + "woman" = "queen"



 1

SlyEcho commented yesterday

Collaborator

It adds some kind of bias to the context, but it is still a completion model, so "Dogs are okay" will still stay the same, but it may add " and I love them!".

They discovered that you need to have some kind difference, so not just adding "love" but better is ("love" - "hate"). You can get the opposite effect with ("hate" - "love") or with a negative coefficient.

I should also mention that it goes token-by-token, so I think in longer steering strings the tokens may need to be aligned for best results, but I need to research this.



Azeirah commented yesterday • edited ▾

Author

This is a super interesting idea! I just want to clarify if I understand this correctly. So, let's say for a prompt like "Dogs are okay", what you do is take the contextualised hidden state from the (say) 12th attention layer and add to it the word representation/embedding for something like "Love". And the resultant output is then something like, "Dogs are awesome!". **Is this correct?**

Also, do you think the reason this works is that word representations are additive? Like for example, a common one I remember is . And so when you add the word vector for "Love" in the above example, it sort of steers the model towards love. "king" - "man" + "woman" = "queen"

I'd strongly recommend reading the post linked in the first comment. This is all pretty new and there is very little understanding of why it works. Only how to do it.

So, let's say for a prompt like "Dogs are okay", what you do is take the contextualised hidden state from the (say) 12th attention layer and add to it the word representation/embedding for something like "Love". And the resultant output is then something like, "Dogs are awesome!". Is this correct?

How I understand it is that it is basically adding the model's understanding of a previous prompt and "adding" it to a later prompt. I think antropomorphizing here is the easiest way to conceptually understand it.

If you play with your dogs all day, it's very likely you're going to think about dogs in situations where dogs are not necessarily the most obvious thing to generally think of.

For example, I played with dogs all day which puts me in a dog-loving state of mind. A friend comes up to me and says "Let's go fishing tomorrow", it's likely you'll think of something like "I'll bring my dog". Even if that's not something you'll generally do.

This example is a bit far-fetched, but the analogy I make here is that

1. The steering-vector is your "state of mind" (dog-mind)
2. The conversation is the new prompt ("Do you want to go fishing tomorrow?")

Normal completion would be

"Yes I would like to go fishing"

With steering vector

"Yes, me and my dog would love to go fishing with you tomorrow."

That said, I think the name of "steering vector" is chosen very well. It steers the model to think in a certain way or direction. IE about love, dogs, coding, or whatever.

Also, do you think the reason this works is that word representations are additive? Like for example, a common one I remember is "king" - "man" + "woman" = "queen". And so when you add the word vector for "Love" in the above example, it sort of steers the model towards love.

Possibly, that's what the post is looking into as well. There isn't a clear answer yet. There are some theories that models are more linear than we initially expected, which means you can perform linear algebra on the vectors and predictably influence the behavior. Which is 100% what they're doing here.



SlyEcho commented yesterday

Collaborator

It is modifying the start of the context.

So the prompt is "Do you want to go fishing tomorrow?"

We calculate the steering vector from "I like dogs" – "I like cats" and get a vector like $[0, 0, \text{🐶}]$ where 🐶 represents some dogness value, and 0 because the other tokens are the same.

When the model is generating new tokens, now the attention mechanism when it is looking into the previous text the prompt now looks more like "Do you 🐶 want to go fishing tomorrow?"



Azeirah commented yesterday

Author

It is modifying the start of the context.

译

So the prompt is "Do you want to go fishing tomorrow?"

We calculate the steering vector from "I like dogs" – "I like cats" and get a vector like $[0, 0, \text{dog}]$ where dog represents some dogness value, and 0 because the other tokens are the same.

When the model is generating new tokens, now the attention mechanism when it is looking into the previous text the prompt now looks more like "Do you dogwant to go fishing tomorrow?"

This is not quite true.

Testing the hypothesis that we're "just injecting extra tokens"

There's a hypothesis that the steering vectors are just injecting extra tokens into the forward pass. In some situations, this makes sense. Given the prompt "I love you because", if we inject a wedding token into the first residual stream with a large coefficient, perhaps the model just "sees" the sentence " wedding love you because".

Tokens are a discrete quantity. You can't have more than one in a single position. You can't have three times wedding and then negative three times (space), on top of I. That's just not a thing which can be done using tokens.

However, consider the steering vector for "Anger"-"Calm" just before layer 20, with coefficient +10. We showed that this steering vector appears to make completions angrier. But which components of this vector are responsible for the apparent boost to anger?

Perhaps what matters is not so much the computational work done by transformer blocks 0 through 19, but the vector given by[25]

[source](#)



SlyEcho commented yesterday

Collaborator

Yes.

I maybe was not clear in how I wrote it. It does not insert new tokens, it modifies the ones in the prompt (or when generating).

When I wrote " 🐶 want" I meant that this is the token "want" with a little bit of dogness added to it.

We calculate the steering vector from "I like dogs" – "I like cats" and get a vector like $[0, 0, \text{🐶}]$ where 🐶 represents some dogness value, and 0 because the other tokens are the same.

This part is not quite right, now that I think about it. The "I like" should affect the "dog" and the "cat" as well, however when subtracting the two, its meaning is now probably something abstract about liking dogs versus liking cats.



SlyEcho commented yesterday

Collaborator

In my PR [#1472](#), I only get the example there kind of working, I'm not able to plant ideas of dogs or weddings into the model.

But there's still things I don't understand.



译

受让人

没有人分配

标签

暂无

项目

暂无

里程碑

无里程碑

发展

没有分支或拉取请求

3 名参与者

