Promptmanteau (by mentaleap)

port·man·teau is a word blending the sounds and combining the meanings of two others, for example motel (from ‘motor’ and ‘hotel’) or brunch (from ‘breakfast’ and ‘lunch’).

Participants

- Amir Sarid
- Bary Levy
- Dan Barzily
- Edo Arad
- Gal Hyams
- Geva Kipper
- Guy Dar
- Itay Yona
- Yossi Gandelsman

Introduction

We researched prompt tuning on GPT-2 for various tasks, with our main conclusion being that **the embedding space for prompt tuning tasks is convex.**
We tried several iterations of training on the same prompt tuning task, each reaching different results, and then checked different convex combinations and saw that they reached a similar success rate in those same tasks. This implies that the different possible ways to solve this task might all come from a single convex set of valid solutions, and allows us to generate many various solutions that all achieve similar results on the task at hand.

Motivation

Prompt-tuning [1] is a technique for instructing the model toward a specific task in “its own language” (the embedding space). The idea is to prepend yet-to-be-used tokens, and learn embeddings for them (also known as soft prompts) while freezing the rest of the model parameters. This technique is successful and mysterious and according to several attempts to interpret soft-prompt it is considered difficult [2,3].

In the context of Mechanistic Interpretability (MI), prompt-tuning offers an interesting ‘hook’, as most of the research is static / passive: you are given a set of weights / architecture and your goal is to learn something about it. Prompt-Tuning gives a dynamic angle to the problem, it allows a quick model fitting for a task of our choice. Moreover, soft-prompts are interesting since they are exactly on the intersection between human-interpretable input and the network algorithm - on an equal footing with natural language tokens, but also serve as a means to affect the network’s algorithm. We believe having both static and dynamic tools is crucial for MI [4].

Research Diary

We used GPT2 for simplicity and speed considerations.

Attempt #1

Our first motivation to interpret soft-prompt is a technique called “Speaking Probes” [5], conducted by one of our members. The idea is to use parameters of the network as part of the input, and let the model work out an explanation using natural language prompts such as: “What is the meaning of <neuron 2>”. The Speaking Probes code replaces under the hood the <neuron 2> with the parameters (or vectors) in question. Even though this idea failed, we moved to other similar questions with the aim of understanding properties of soft-prompts.

Other attempts

We started with a sentiment analysis task for rotten tomatoes with multiple soft-tokens (10 & 20). The model failed to learn the task successfully. We managed to find a simple task (+1) which works for small numbers represented as a single token (the numbers from 1- 500). We shrunk the number of tokens down the three and the model actually managed to solve the problem. We tried changing the order of the token (which works quite well), to repeat the token multiple times (hoping the model will apply +2) but it failed, we tried using only a subset of the three tokens (it partially worked, seems like a single token was enough to reduce the result to numeric), and it also tried to sum all the tokens (this also partially worked, it has interestingly only worked when we summed the difference between the before and after training embeddings!).

Successful attempt

Eventually we chose to study the +1 task, on GPT2 model. We trained a single token to represent the plus_1_task reaching 100% accuracy. It is important to note that the model naturally doesn’t output the next number of the input. We repeated the training several times ending up with several different tokens representing the same task. We tried out convex combinations for the task-tokens and for our supersize the model predicted the next number!

Results

Our main result is that prompts seem to lie in a convex set. That means the vectors that represent this task can be added together as a convex sum to yield a different token which is also instructing the model to conduct the same task. We managed to show that for a variety of tasks: +1, +2, identity (+0), IMDB sentiment-analysis. Using multiple single-task tokens (one to four tokens). Furthermore, the convex sums tokens show only a minor reduction in performance of the task! We also made sure the learned tokens are not identical or similar using cosine similarity (this method works also for orthogonal task vectors).

We believe these results extend beyond GPT2, and might also work with multiple-tokens per task.

Some numeric results of our analysis:

Sentiment Analysis

- token1_imdb: 83.90%
- token2_imdb: 81.50%
- combination 1 **[0.797 0.2033]**: 63.10%
- combination 2 **[0.358 0.642]**: 59.00%

Identity

- token01: 99.60%
- token02: 99.20%
- combination 1 **[0.707 0.293]**: 95.00%
- combination 2 **[0.743 0.257]**: 96.20%

+1

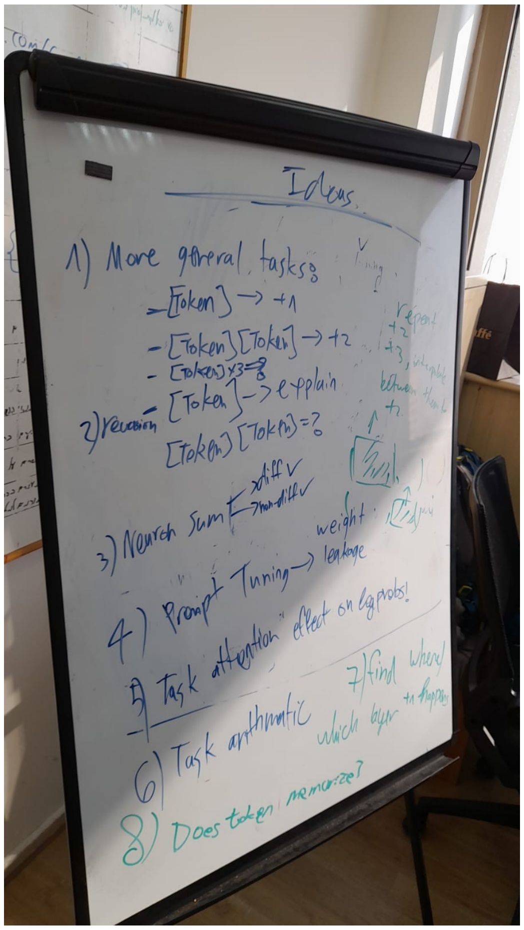
- token11: 69.60%
- token12: 95.20%
- token13: 99.80%
- token14: 88.00%
- combination 1 **[0.370 0.229 0.217 0.184]**: 92.80%
- combination 2 **[0.09 0.263 0.413 0.231]**: 99.40%

+2

- token21: 77.00%
- token22: 77.60%
- token23: 84.40%
- combination 1 **[0.439 0.075 0.485]**: 50.80%
- combination 2 **[0.334 0.235 0.431]**: 41.40%

Further directions

1. An interesting question we had was to measure the dimensionality of the task convex space but adding orthogonality regularization. We decided not to follow this direction because of time considerations.
2. A different thing we wanted to study is a representation for a generalizable task, such as a token that adds one, but also when repeated adds two.
3. [token] [num] => [num+1]
4. [token] [token] [num] => [num+2]
5. Fitting the embedding accordingly might yield a token which could be repeated n times to express the instruction of +n.
6. We also wanted to fit a token for the task of “Explain the following” and then run:
7. [token] [token], the model would need to explain the meaning of the explanation token. This recursive structure interested us.
8. We wanted to use prompt-leakage to maybe extract the meaning of the prepended tokens by using a phrase such as “ the beginning of this sentence means”.
9. Compositionality: it would be cool to find a way to compose different token-tasks and end up with a task reflecting both tasks! We had ideas on how to follow this question but we did not have enough time. The idea is common in Computer Vision [6].


Ideas.
1) More general tasks:
- [Token] → +1
- [Token] [Token] → +2
- [Token] x3 = ?
repeat +2 +3, interpolate between them to +2
2) recursion - [Token] → explain
[Token] [Token] = ?
3) Neuron Sum → diff ✓ → non-diff ✓
weight
4) Prompt Tuning → leakage
5) Task attention effect on logprobs!
6) Task arithmetic
7) find where which layer +1 happens
8) Does token memorize?

References

[1] The Power of Scale for Parameter-Efficient Prompt Tuning https://arxiv.org/abs/2104.08691

[2] Prompt Waywardness: The Curious Case of Discretized Interpretation of Continuous Prompts, https://arxiv.org/abs/2112.08348

[3] Interpretable Soft Prompts https://learnprompting.org/docs/trainable/discretized

[4] Software Reverse Engineering and Mechanistic Interpretability, https://www.neelnanda.io/mechanistic-interpretability/reverse-engineering

[5] Speaking probes, https://link.medium.com/tfcdeAOzDwb

[6] MyStyle: A Personalized Generative Prior, https://arxiv.org/abs/2203.17272