Internship weekly report

Mathis Embit

MILES, LAMSADE

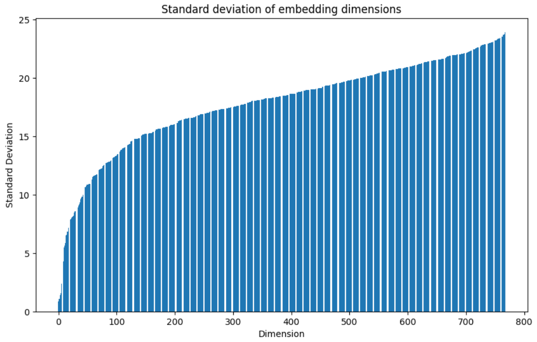
May 3, 2024

Outline

Introduction

Today: some experiments and questions.

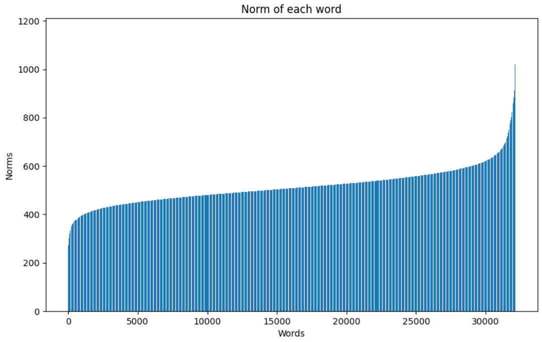
Standard deviation for each dimension



Standard deviation for each dimension. Mean = 18

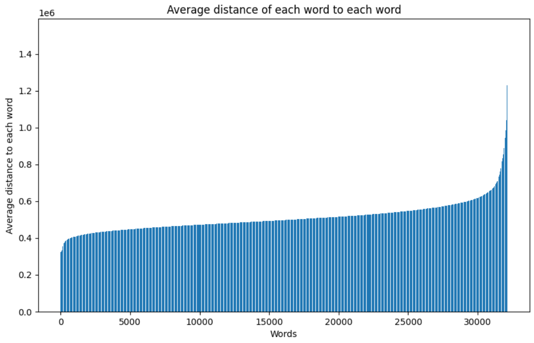
Conclusion: high compared to other models. We normalized gradient and choose a learning rate around 18.

L2 norm of each word in the vocabulary



L2 norm of each word in the vocabulary. Mean = 516

Average L2 distance of each word to every word of the vocabulary



Average L2 distance of each word to every word of the vocabulary. Mean = 510,000

Synonyms

Just to clarify

- soft prompting = continuous descent = optimizing over the embeddings
- hard prompting = discrete descent = optimizing over the tokens

Discrete descent examples

Let's fix:

- number of steps = 100
- number of loss evaluation among the candidates = 3
- initial prompt = ['<extra_id_0>', '<extra_id_1>', '<extra_id_2>', '<extra_id_3>', '<extra_id_4>', '<extra_id_5>', '<extra_id_6>', '<extra_id_7>', '<extra_id_8>', '<extra_id_9>', '<extra_id_10>', '<extra_id_11>', '<extra_id_12>', '<extra_id_13>', '<extra_id_14>', '<extra_id_15>', '<extra_id_16>', '<extra_id_17>', '<extra_id_18>', '<extra_id_19>', '_London', '_is', '_the', '_capital', '_of', '<extra_id_0>', '</s>']

Discrete descent examples

Let's make the number of prefix tokens vary.

Discrete descent examples

If **prefix length = 5** we get:

- Optimized prompt = `['_circulation', 'Char', 'rruption', 'haudiere', '_Alfred', '_London', '_is', '_the', '_capital', '_of', '<extra_id_0>', '</s>']`
- Output generated with the user prompt = "England."
- Output generated with the optimized prompt = "England."

Discrete descent examples

If **prefix length = 6** we get:

- Optimized prompt = `['mila', '_miscare', '_vizualiz', '_pamant', '_Sfant', '_departe', '_London', '_is', '_the', '_capital', '_of', '<extra_id_0>', '</s>']`
- Output generated with the user prompt = "England."
- Output generated with the optimized prompt = "France."

Conclusion

There is a sort of threshold for the number of prefix tokens.

Continuous descent examples

Let's fix:

- prefix length = 20
- number of steps = 100
- initial prompt = `['.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '_London', '_is', '_the', '_capital', '_of', '<extra_id_0>', '_', '.', '</s>']`

And the output generated with the user prompt = "the United Kingdom the United Kingdom the United Kingdom the United Kingdom the United"

Continuous descent examples

Let's make the learning rate vary.

Continuous descent examples

If **learning rate = 0.1** we get:

- Optimized prompt with the closest words = `['.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '_London', '_is', '_the', '_capital', '_of', '<extra_id_0>', '_', '.', '</s>']`
- Output generated with the optimized prompt = "England. England. England. England... England."
- Next token generated from the raw optimized embeddings = "the"

Continuous descent examples

If **learning rate = 1** we get:

- Optimized prompt with the closest words = `['.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '_London', '_is', '_the', '_capital', '_of', '<extra_id_0>', '_', '.', '</s>']`
- Output generated with the optimized prompt = "England. England. England. England... England."
- Next token generated from the raw optimized embeddings = "France"

Continuous descent examples

If **learning rate = 10** we get:

- Optimized prompt with the closest words = `['.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '_London', '_is', '_the', '_capital', '_of', '<extra_id_0>', '_', '.', '</s>']`
- Output generated with the optimized prompt = "England. England. England. England... England."
- Next token generated from the raw optimized embeddings = "France"

Continuous descent examples

If **learning rate = 100** we get:

- Optimized prompt with the closest words = `['farbe', 'cativa', '_2017.', 'Datorita', '_Audio', '_dorint', 'nique', '_interioare', '<extra_id_27>', '_France', '_cautare', 'ologi', '.', '_suprafete', '_summarize', '_adica', '_Nantes', '2.', 'haudiere', 'Biserica', '_London', '_is', '_the', '_capital', '_of', '<extra_id_0>', '_', '.', '</s>']`
- Output generated with the optimized prompt = "England."
- Next token generated from the raw optimized embeddings = "France"

Conclusions

- We are able to find embeddings that generates 'France' eventhough their closest words stay at '.'.
- Then, when the learning rate is high enough to make clost words change, those do not allow us to generate 'France'.

Interpretable?

Even when tweaking the hyperparameters a little bit more, i didn't manage to generate 'France' with the closest words from the raw embeddings:

- With nb steps = 100, lr = 1000 we get "Romania. Romania Romania. Romania.."
- With nb steps = 100, lr = 10000 we get "Romania."
- With nb steps = 500, lr = 1000 we get "the world. the world."
- With nb steps = 1000, lr = 100 we get "the UK."

Hence, not super interpretable.

Other ideas

Ideas I came across in **Universal and Transferable Adversarial Attacks on Aligned Language Models** [Zou+23]:

> *Soft prompting process is not reversible: optimized soft prompts will typically have no corresponding discrete tokenization[...]. However, there exist approaches that **leverage these continuous embeddings by continually projecting onto hard token assignments**. The Prompts Made Easy (PEZ) algorithm [Wen+23], for instance, uses a quantized optimization approach to adjust a continuous embedding via gradients taken at projected points, then additionally projects the final solution back into the hard prompt space. Alternatively, recent work also leverages Langevin dynamics sampling to sample from discrete prompts while leveraging continuous embeddings [Qin+22].*

Questions

- How should we initialize the placeholder tokens?
- In soft prompting, does projecting at each step is equivalent to choosing the token that minimize the loss the most?
- Is there a set of reachable labels given a number of placeholder tokens, a number of steps, a learning rate? If so, can we design an aligned model such that this set is the smallest possible?
- Is the optimal prompt syntactically correct or rather looks random? If it is syntactically correct we may add a likelihood term to the loss.
- Does optimizing for the generation of a single word makes sense? (one prompt per task so we can compare to fine-tuning)

LLM control theory

A paper I came across: **What's the Magic Word? A Control Theory of LLM Prompting** [Bha+24].

What they are interested in:

- formalization of LLM system, then what can we say mathematically
- is there some input u for every imposed state x_0 that steers an LLM to output any desired y? If yes, optimal/minimal length of u. If no, which y are reachable from which x_0.
- engineering: how to find u practically

Their goals:

- control properties of C-o-T
- learnability/computational cost of control
- composability of LLM systems
- are there controllable/uncontrollable subspaces?

Their contributions

- A theorem about self-attention I do not understand yet.
- They experimentally illustrate the reachabe set with the proportion of label they are able to generates with a prompt optimization process (greedy coordinate gradient from [Zou+23]).

However I think their framework does not take into account the training process. Hence I do not know what kind of hypothesis they put on their matrices.

References I

[Bha+24] Aman Bhargava et al. *What's the Magic Word? A Control Theory of LLM Prompting*. 2024. arXiv: 2310.04444 [cs.CL].

[Qin+22] Lianhui Qin et al. *COLD Decoding: Energy-based Constrained Text Generation with Langevin Dynamics*. 2022. arXiv: 2202.11705 [cs.CL].

[Wen+23] Yuxin Wen et al. *Hard Prompts Made Easy: Gradient-Based Discrete Optimization for Prompt Tuning and Discovery*. 2023. arXiv: 2302.03668 [cs.LG].

[Zou+23] Andy Zou et al. *Universal and Transferable Adversarial Attacks on Aligned Language Models*. 2023. arXiv: 2307.15043 [cs.CL].