Meeting — Summary LLM

Bas Doners, Dennis Fok, Finn-Ole Höner

January 18, 2024

Terms

target string The string that we want to "re-create"

input embedding $n_{tokens} \times n_{vocab}$ input embedding

flat input embedding $1 \times n_{vocab}$ input embedding

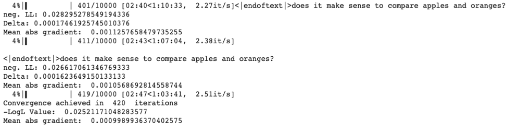
Can we speed things up? i

- flat input embedding of target string helps
- GPU helps a lot
- RMSProp instead of ADAM helps
  - Adjust learning rate based on input embedding and generation technique
- n-Beam-search is slightly slower than greedy search, but not by factor n (finding better steps that take longer to compute?)
- Stop optimization based on hitting the target sequence instead? About half the iterations

Can we speed things up? ii

- Is there a way to move to "conditional likelihoods", once we found one correct token?
- How stable is the optimization?

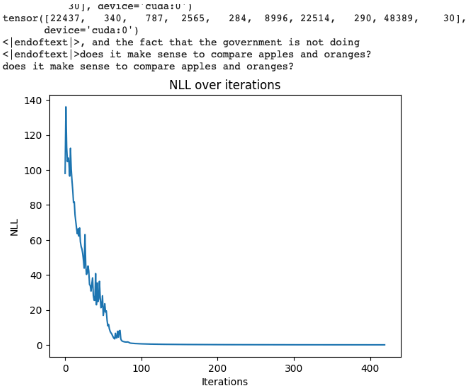
“Does it make sense to compare apples and oranges?”, flat input embedding, RMSProp(lr=1e-1), T4GPU



```
  4%|▌         | 401/10000 [02:40<1:10:33,  2.27it/s]<|endoftext|>does it make sense to compare apples and oranges?
neg. LL: 0.028295278549194336
Delta: 0.00017461925745010376
Mean abs gradient:  0.0011257658479735255
  4%|▌         | 411/10000 [02:43<1:07:04,  2.38it/s]

<|endoftext|>does it make sense to compare apples and oranges?
neg. LL: 0.026617061346769333
Delta: 0.00016236491501331333
Mean abs gradient:  0.0010568692814558744
  4%|▌         | 419/10000 [02:47<1:03:41,  2.51it/s]
Convergence achieved in  420  iterations
-LogL Value:  0.025211710482835777
Mean abs gradient:  0.0009989936370402575
```

Figure 1: Convergence



```
30], device='cuda:0')
tensor([22437,   340,   787,  2565,   284,  8996, 22514,   290, 48389,    30],
       device='cuda:0')
<|endoftext|>, and the fact that the government is not doing
<|endoftext|>does it make sense to compare apples and oranges?
does it make sense to compare apples and oranges?
```


Figure 2: NLL

Generation technique?

- We run into problems if the generation technique samples: Then we cannot track the likelihood anymore (inplace gradient change)
- Likely beam search with many beams is the most balanced option?
- Contrastive search, which penalizes repeated tokens, does not work as it relies on sampling behind the scenes
- Maybe we can penalize the likelihood itself for repetitions? Or solve it through restrictions on the optimization?

Number of parameters?

- With flat input embedding, we have n_{vocab} (768 for GPT-2) parameters.
- Many hyperparameters: Starting values, optimizer, generation settings, LLM
- Maybe we can reduce the number of parameters by having a smaller layer in front of the flat input-embedding. This smaller layer plus the weights connecting it to the input embedding could be less than n_{vocab}. (Backwards convolution?)

Other

- Need to read up on the Google Colab resources, whether it is suitable for computing larger datasets
- GPU on Snellius (would that be feasible)?
- GitHub
- Can we compare the likelihood across different settings? (Likely not?)
- Difference to "Reverse Prompt-Engineering"