**Investigating the Effect of Training Data Order on Small Language Model Fact Retention during fine-tuning - or “order is all you need”**

**Project Description:**

This project explores how the ordering of training data affects fact retention in small language models (LLMs) during fine-tuning task. Specifically, we aim to test whether information presented at the beginning or end of a training dataset is more likely to be retained and retrieved accurately by a model. While transformer-based models process data in segments during training, it remains unclear whether global ordering across an entire corpus significantly impacts memorization or downstream performance.

**Training data –**   
our base corpus was a data set of 5000 sentences structured as “Q <question> A: <Answer>”.   
the base corpus was taken from (Harel explain shortly from and how you created it)

In addition, we created 30 made up questions, with 2 options for fictitious answers for each question.  
the questions and answers are made up and make little sense to make sure the results aren’t affected from previous bias of the model. Indeed, the answers are given very low probability and rank as can be seen below.

This template was structured and chosen to mimic real finetuning tasks, such a chatbots fine tuning.

The answers were all made of a single token, to make the evaluation and training processes clearer and to prevent imbalances in the data.

**Training process general notes -**

We finetuned three base model of different sizes (160m, 410m, 1b).   
the training took place only on the <answer> token, to focus on fact learning and storage, without changing the model’s world knowledge.

**Experiment 1 – early vs late comparison.**

**training process –**

For each model, we fine-tuned two different versions, one with the fictitious dataset appearing before the base corpus (early variant), and a second where it appears after (late variant).

**Evaluation process and metrics -**

For both the early and late finetuned version we tested several metrics measuring how saucerful it is in retrieving the expected fake answer from the training data –

Average rank, average probability, and percteage of the answer appearing in the top 1,5,10,50 and 100 tokens, ranked by probability.

The result for each model can be shown below –

|  |  |  |  |
| --- | --- | --- | --- |
| Base model (160/410/1b) | Base model | early | late |
| Average rank | 200 | 3 | 2 |
| Average probability | 0.0001 | 0.4 | 0.6 |
| Top1 | 0% | 20% | 50% |
| Top5 | … | … | … |
| Top10 | .. | … | … |
| Top50 | … | … | … |
| Top100 | … | … | … |

Result analysis –

Lets see lol

**Experiment 2 – contradicting facts –**

**training process –**   
For each base model, we fine-tune two variants:

* Variant 1: prepend the corpus with the ficticious question with answers A1 and append the corpus with contradictory answers A2.
* Variant 2: reverse the order of A2 and append A1.

 **Evaluation process and metrics -**

For each varient, and for both the early and late answers (which are different for each variant) we tested again the rank, probability and whether the answer appeared in the top 1,5,10,50 and 100 tokens.

We took this approach to prevent possible skewed results caused by possible preconceived biases of the model. If we for example had accidently placed a real fact only at the beginning, the output would have invalid for our purposes. Placing every answer both at the end and the beginning of the corpus cancels out this possible problem. (maybe this whole paragraph is a footnote)

We then aggregated the result of both variants early and late answers to see the averages, with the following results –

|  |  |  |
| --- | --- | --- |
| Base model (160/410/1b) | Late facts | Early facts |
| Average rank | 3 | 2 |
| Average probability | 0.4 | 0.6 |
| Top1 | 20% | 50% |
| Top5 | … | …. |
| Top10 | … | … |
| Top50 | … | … |
| Top100 | … | … |

**Implications –**

If early better –

Place higher quality data and data from quality sources earlier in fine tuning data, but further research is required to come to a definite conclusion

If same –

The order of the data doesn’t seem to matter but further research is required to come to a definite conclusion

**Future steps -**

While our research provide preliminary findings and results, many additional questions remain –

Will the results stay the same on models of larger size?

Will the results stay the same for different model architectures? Further research on image generation applications such as diffusion and flow matching models is needed.

How will the result be affected on training the model from scratch instead of finetune missions?

Is earlier always better? Does data that appear in the middle still prove more prominent the data which appears late?

Test the effect on different phrases of the same question. Will asking the same question in a slightly different wording change our observations?

How much data is required to overcome this effect? Will enough data samples in base corpus be able to mitigate the results?

**Appendix –**

**Failed attempts –**

At first, we planned to train the models which we used in the paper from scratch, but we quickly learned that we don’t have sufficient compute nor time. Any attempts to train from scratch smaller model produced very bad results which we believed would have rendered any result irrelevant.

We then proceeded to fine tune the models using basic statements, i.e. “the capital of France is Paris” instead of “Q: what is the capital of France? A: paris”. However, despite many different attempts and configurations our attempts seemed to either overfit the training data, destroying the model’s world knowledge, or provide very bad results in the training data. We couldn’t reach a version we deemed to have an appropriate tradeoff between the two. It is possible that larger models with more parameters might prove to be more resilient to such changes, and this is a question for further research.

**Data creation process details -**

Harel you’re up

**Code repo –**

https://github.com/HarelBS/NLP\_Project/tree/main