Learnings from fine-tuning LLM on my Telegram messages

27 Nov 2023

For most people I interact with, I’m just another text-based program for the most of the time. If input and output are so simple, could I be replaced by the model? For this to work, the model would need to not only understand my writing style but also know a lot about me. The best source for this is my Telegram messenger, as I use it daily and it contains almost everything about my thoughts and actions in the form of chat histories.

Approach

The most straightforward approach would be to extract all my messages, load them into ChatGPT’s context, and instruct it to use this information to mimic my style when responding to new messages. However, this approach is limited by the context window size, requiring me to preprocess messages to extract key points. As I want to avoid this hassle, perhaps Retrieval Augmented Generation (RAG) could be used to pull necessary information when needed. However from my experience, retrieving from diverse data like chat sessions usually needs a supervised fine-tuning of the retrieval model, and I’m not keen on creating such a dataset. So, fine-tuning seems like the best option. It’s ideal for several reasons: it should capture my writing style and potentially accumulate knowledge from all my messages without having to select what’s important.

OpenAI offers fine-tuning capabilities, but as I’ll be using my private messages, I don’t want to use any third-party fine-tuning services. So, I need to choose a base model. According to the Hugging Face Open LLM Leaderboard, one of the top smaller models (≤13B parameters) is Mistral 7B. It even outperforms Llama 2 13B. Now, the question is whether LoRA is sufficient or if full fine-tuning is necessary. Various comparisons [1] [2] suggests that LoRA is a bit worse than full fine-tuning but still fine most of the time. However, for specific tasks like mine (Russian language + chat), I found a paper, where researchers conducted Llama instruction fine-tuning in Chinese, similar in complexity to my goal. They found that LoRA-based tuning on a base model without prior instruction tuning is less effective than full fine-tuning. Yet, LoRA-based tuning on a model already fine-tuned for instructions can yield comparable results. In my case, this means either full fine-tuning on a base model or LoRA on a model already fine-tuned for chatting in Russian. Since I couldn't find a model fine-tuned for Russian chat, I'll try LoRA on a model fine-tuned for English chat, like the fine-tuned Mistral model Dolphin.

So, the plan is:

1. Start with LoRA on top of Dolphin, the English chat fine-tuned Mistral
2. If quality is not sufficient, try full fine-tuning on Mistral

Data preparation

One unique aspect of messaging in apps like Telegram, compared to emails, is the conversational flow. Messages don't usually alternate one-by-one between you and your contact. Instead, you often find yourself sending a couple of messages in a row, followed by several responses from the other person. These messages are generally short, too. I wanted to preserve this natural conversational style in my data.

Telegram offers a built-in feature to export all chats into JSON. After some filtering and grouping messages into sessions, I've compiled data from the last five years of using Telegram. This resulted in 15,789 sessions from 466 chats, with an average session length of 8.51 messages. For structuring the data, I've chosen the ChatML prompt format. Here's a sample session (translated from Russian):

```
<|im_start|>John Smith
>>> damn, can't get around the 135 time limit
>>> trying to do everything super optimally, but no luck<|im_end|>
<|im_start|>Alexander Smirnov
>>> yeah same
>>> you still going with the same idea?<|im_end|>
<|im_start|>John Smith
>>> dunno, I think we're on the same page
>>> as you said
>>> going with the reversed string in a try and trying to find
something there
>>> seems like real shit because z function ruins
everything………………………<|im_end|>
<|im_start|>Alexander Smirnov
>>> don't get where z comes into this<|im_end|>
<|im_start|>John Smith
>>> dunno seems like I'm doing everything iteratively anyway, but
yeah gotta reverse some strings to build the z function
>>> and it's just a random solution
>>> from discussions<|im_end|>
<|im_start|>Alexander Smirnov
>>> got it<|im_end|>
```

▶ original

My data collator ensures that the loss is only calculated based on someone’s response. Predicting who will speak next is relatively straightforward, and we don’t want the model to focus on learning that. Therefore, parts of the conversation where the loss is calculated are highlighted in bold.

You might notice that not only my responses but also those of others are used for loss calculation. This is deliberate. By doing this, the model will be able to role-play not only as me but also as my frequent conversational partners!

Evaluation plan

I will test models by having chats in two ways. First, the model will pretend to be me and I will be chatting with myself from the perspective of my different friends. Then, I’ll chat as myself while the model acts as my friends. My conversation starter will always be the same 2 messages: “hey” and “what’s up?” (in Russian, “прив” and “как дела?”). Generated phrases and persons as the model acts who from will be **highlighted**. All conversations initially will be held in Russian and may be accessed by clicking on the ‘original’ details button. For testing I will be using oobabooga/text-generation-webui.

In the beginning, I want to explore how the generic conversation fine-tuned Mistral model deals with that task without any prior training from my side.

▶ Friend 1 vs **Alexander Smirnov**
▶ original

▶ Alexander Smirnov vs **Friend 1**
▶ original

Ok, it is capable of forming coherent sentences. The most noticeable problem is its lack of awareness regarding the context of the conversations which leads to bland and generic replies. The messages lacked any distinct style, feeling quite basic. Another issue is that the model’s Russian is poor. This is expected, as the model is too small to generalize well to languages other than its primary one, English. Additionally, the model tended to be overly proactive, ending almost every sentence with a question, which isn’t how real people typically communicate in messengers.

Let’s try to fix all of these!

LoRA

LoRA offers a low-effort approach in terms of both the training pipeline and hardware requirements. It trains around 1% of the total weights. I chose a 1024 sequence length and a batch size of 8. The training, which consumed 20GB of VRAM on an RTX 3090, took three epochs and lasted for 5.5 hours. For this, I used vast.ai, where the GPU cost was $0.362 per hour, totaling $2 for the entire training, excluding time spent on experiments and bug fixes.

Here are the results:

▶ Friend 1 vs **Alexander Smirnov**
▶ original

▶ Friend 2 vs **Alexander Smirnov**
▶ original

▶ Friend 3 vs **Alexander Smirnov**
▶ original

▶ Alexander Smirnov vs **Friend 1**
▶ original

▶ Alexander Smirnov vs **Friend 2**
▶ original

▶ Alexander Smirnov vs **Friend 3**
▶ original

This is much better. It definitely captures the style of the person it's responding on behalf of. It also identifies the most common topics discussed between specific pairs of people. For example, with friend 2, the focus is clearly more on work. However, the grammar is still off, and it loses the context of the conversation quickly. I'm pretty confident that LoRA would work with reasonable quality in English, and full fine-tuning might not be necessary. But, since Russian isn't the model's native language, let's try full fine-tuning.

Full fine-tuning

Full fine-tuning is more challenging due to the need for multi-GPU training. Popular methods include either ZeRO & DeepSpeed [3] or

FSDP [4], with FSDP essentially being a ZeRO3 [5]. I decided to go with FSDP.

While implementing the training pipeline, I referred to the Stanford Alpaca fine-tuning code and Anton Bacaj’s Mistral fine-tuning code.

Using a half-precision FSDP full shard with a 1024 sequence length and a micro batch size of 2 required 63GB of VRAM on each of the eight A100 80 GB GPUs. The training, lasting three epochs, took just 20 minutes. The total cost for the VM was $8.88 per hour, resulting in $3, not including the time for experiments and bug fixes.

Conversations:

▶ Friend 1 vs **Alexander Smirnov**
▶ original

▶ Friend 2 vs **Alexander Smirnov**
▶ original

▶ Friend 3 vs **Alexander Smirnov**
▶ original

▶ Alexander Smirnov vs **Friend 1**
▶ original

▶ Alexander Smirnov vs **Friend 2**
▶ original

▶ Alexander Smirnov vs **Friend 3**
▶ original

Conversations have become more interesting and engaging, although there’s still a risk of losing context. Russian language performance has improved, but errors still occur. I believe that before fine-tuning for a specific task with limited data, like mine, it would be beneficial to first fine-tune the model unsupervised on a large corpus of Russian texts. Additionally, incorporating common conversation partners’ names as separate tokens might enhance the quality.

I wouldn’t say it has turned out to be significantly better than LoRA. It might be more effective to focus solely on a single person and calculate the loss based only on my responses (or someone else’s), instead of trying to learn about each and every conversational partner.

Closing thoughts

Certainly, I had to cherry-pick the results, not because most of the model’s replies were inadequate, but because many were simple responses like “I’ll call you later,” “busy,” and “ok,” which are naturally frequent in conversations. Despite this, it’s clear that the model excels in mimicking the style of the person it’s impersonating. It also captures the commonly discussed topics between two people. However, it significantly lacks context in conversations. Responding to queries like “yo, so?” or “what are your plans for the weekend” is challenging without having full context. Perhaps utilizing a system like Rewind, which captures everything the user does across the computer, could be beneficial.

Code

You can find code for this project as well as instructions on how to replicate it yourself on your own Telegram dump in my github repo. Training logs may be accessed on WandB.

---

1. Fine-Tuning LLMs: LoRA or Full-Parameter? An in-depth Analysis with Llama 2 (anyscale.com/blog) | ↩
2. LoRA results in 4-6% lower performance compared to full fine-tuning (github.com/huggingface) | ↩
3. How to Choose Which ZeRO Stage and Offloads To Use For Best Performance (huggingface.co/docs) | ↩
4. Introducing PyTorch Fully Sharded Data Parallel (FSDP) API (pytorch.org/blog) | ↩
5. It’s 2023. Is PyTorch’s FSDP the best choice for training large models? (openmmlab.medium.com) | ↩
