University *of Ljubljana*
Faculty *of Computer and Information Science*

Slovenian Instruction-based Corpus Generation

Jernej Ulčakar, Jon Kuhar, Bella Muradian

Abstract

This study explores the utilization of web scraping techniques to extract and analyze data from online forums. Through scraping scripts, we gathered a substantial corpus of forum posts and comments. This corpus was then parsed with scripts.

Keywords
LLM,Slovene,Training

Advisors: Slavko Žitnik

Introduction

Large Language Models (LLMs) have shown great promise as highly capable AI assistants that excel in complex reasoning tasks requiring expert knowledge across a wide range of fields, including in specialized domains such as programming and creative writing. They enable interaction with humans through intuitive chat interfaces, which has led to rapid and widespread adoption among the general public.

We fine tune a model on conversations from slovene forums such as Med.Over.Net, slo-tech and Reddit Slovenia. We collected this data using WinHTTracker scraper and python for parsing.

For the base model we used Alpaca-lora, the *low-rank adaptation* (LoRA)**??** of the Alpaca**??** model. This model is itself a fine-tune of the Llama**??** model. It is fine-tuned with a large set of conversations and was relatively cheap to make. The LoRA version is even more concise, allowing for very cheap further training, doable on consumer hardware. The model should have a basic understanding of the slovenian language because Llama's training dataset contains 4.5% wikipedia dumps from June-August 2022. These dumps cover 20 languages, including slovenian.

We published the code and the model weights on github.

Related Works

Llama 2 (2023) [**?**] is a collection of pretrained and fine-tuned large language models (LLMs) ranging in scale from 7 billion to 70 billion parameters. They introduced LLMs that are fine-tuned, called Llama 2-Chat, and are optimized for dialogue use cases.

BLOOM (2023) [**?**] is a collection of pretrained and fine-tuned A 176B-parameter open-access language model designed and built thanks to a collaboration of hundreds of researchers. BLOOM is a decoder-only Transformer language model that was trained on the ROOTS corpus [**?**], a dataset comprising hundreds of sources in 46 natural and 13 programming languages (59 in total). BLOOM achieves competitive performance on a wide variety of benchmarks, with stronger results after undergoing multitask prompted fine-tuning. To facilitate future research and applications using LLMs.

Training models (2022) [**?**] this paper contains instructions in how to fine-tune language models with user intent on a wide range of tasks by using human feedback. Starting with a set of labeler-written prompts and prompts submitted through the OpenAI API, they collected a dataset of labeler demonstrations of the desired model behavior, which they later used to fine-tune GPT-3 using supervised learning and the collect the dataset of rankings of model outputs. These are used to further fine-tune the model using reinforcement learning from human feedback. The model names are InstructGPT. In human evaluations on our prompt distribution, outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3, despite having 100x fewer parameters. Moreover, InstructGPT models show improvements in truthfulness and reductions in toxic output generation while having minimal performance regressions on public NLP datasets. Even though InstructGPT still makes simple mistakes, their results show that fine-tuning with human feedback is a promising direction for aligning language models with human intent.

Methods

Scraping

We used WinHTTrack, a powerful web scraping tool, to scrape data from three online forums: Med.Over.Net, slo-tech.com and forum.over.net.

We configured WinHTTrack to navigate through the forum websites systematically, crawling through the existing links to access all posts and comments.

Each website contains more than just discussions with comments, for example user pages, about info, links to outside of the page. To ensure efficient parsing, we needed to analize the structure of each forum's urls to make sure only the urls corresponding to forum threads are crawled. An example of the process for the site slotech.com is in .

We made sure to respect the forums' `robots.txt`. This is done automatically by WinHTTrack and actually simplified the process as all the rulesets prevented scraping of data we didn't want to scrape anyway.

Scraping slotech.com forums

Slo-tech is the biggest slovenian computer forum that has existed for over two decades. On the front page of the forum we see the split into many subforums in the format of `/forum/{id}`. The subforums are not listed in increasing order and not all ids are taken. The lowest id forum is "Zvok in slika", `/forum/4`. There are two special subforums that are not scraped as they link to threads in other subforums, "Teme zadnjih 24 ur" (Threads made in the last 24 hours) and "Neodgovorjene teme" (Threads without responses).

The next pages on the forum are accessed with `/forum/{id}/1`, `/forum/{id}/2` and so on. `/forum/{id}/0` is also theoretically valid for the access of the first page but this is not used inside of the pages for the scraper to pick up.

The threads have url structure `/forum/t{id}`. They contain up to 50 messages. Next pages are accessed with `/forum/t{id}/{n}`. *n* represents the number of the next message shown so page 2 is `/49`, page 3 is `/99` and so on. For large pages the navigation includes `/konec` to jump to the last page. The first page is linked to as `/0`. We set up the scraper to ignore the `/0` and `/konec` as they lead to duplicated pages.

At the bottom of threads there is usually a table of related posts. These do not necessarily link to the same subforum This is a bit of a problem if we want to be selective about scraping a specific subforum. There is no info in the url of a thread to indicate the subforum it is in.

The robots.txt of slo-tech permits accessing all forum pages but not user pages `/profili` and the lists of user threads `/u{id}`, user comments `/c{id}` and threads contributed to by a user `/s{id}`. This is fine for us as we do not need to scrape user profiles.

Important parameters used for the scraper, the initial urls and scan rules, are available in the github repository in the `scripts/scraping` directory.

Parsing data

Similarly to scraping, each forum had a different structure in the html. To parse this structure we developed parsers in python using the `BeautifulSoup` library. We use this on each thread to extract the users, dates, content and quoted content. Not all of this data is going to be used, but it was still collected for completeness. The threads are collected into a json file. The code used for parsing is available in the `scripts/parsing` directory and the parsed files are in `results/parsing`.

Generating instructions

The original training of the Alpaca model was done by fine-tuning on a list of 52k data points in the format of:

```
{
    "instruction": "...",
    "input": "...",
    "output": "..."
}
```

Our fine-tuning script will also use this format to keep the original training script with minimal modifications.

Because many threads start with a question, we decided to format our instruction set such that the title and original posts are combined as the instruction. Because many of the first responses directly address and answer the question we use these as the output.

The code used for transforming our parsed data into this format is in `scripts/generate`, the instruction set is in `results/generate`.

Training

The training script was only slightly modified from the original Alpaca-lora's training script. Since we wanted to keep the lora focused on instruction responses, the initial weights were set to those from Alpaca-lora. We used the following hyperparameters:

batch_size=128, micro_batch_size=4, num_epochs=3,
learning_rate=0.0003, cutoff_len=256, val_set_size=2000,
lora_r=8, lora_alpha=16, lora_dropout=0.05

Evaluation

We used the human evaluation instructions from Self-instruct**??**, translating a small set to slovene and evaluating it.

Results

The scraping statistics are in table **??**. We expected to get rate limited but that only happened late when enough threads were already scraped.

The results of data parsing are in table **??**. The data was parsed well. The only junk remaining was inside the posted content of each user as well as the users type of speech (this includes typos and accent-like typing). Generating instructions was also without issues.

The training was done on vast.ai. The GPU we used was an RTX 6000Ada with 81.4 TFlops and 48GB of VRAM. The VRAM was underutilized, only using up to 10GB while training. This means we could have used a less performant GPU without running out of memory.

In total, training took 3h for 300 iterations.

Discussion

Use the Discussion section to objectively evaluate your work, do not just put praise on everything you did, be critical and exposes flaws and weaknesses of your solution. You can also explain what you would do differently if you would be able to start again and what upgrades could be done on the project in the future.

References