Gridspace IAP 2023
Project 2

January 16, 2023

Week 2: Language and Lexers

Due Friday 20th Jan, Presentations Monday 22nd

Fine Tune GPT-3

This week we focused on linguistics, and the complex problem of language representation. Computers understand numbers, and in their essence, ML programs are trained to pick up on patterns in vectors. It then follows, that our ability to successfully transform a sentence into a vector representation without loss of the information we wish to encode, is a large part of what determines the success of our model.

For this week, we want to push you to play with some of the state of the art Large Language Models in the field. The key takeaway from this project should be just how easy it is to build off these, and just how powerful having free access to these highly trained models is.

GPT-3 (of which ChatGPT is a derivative) is amongst the most famous of these. It is a generative model, trained to predict the next word in a sentence, given the sentence preceding it. It works through a 'prompt' and 'response' method, and can be fine-tuned on any dataset formatted as

```
{"prompt": "<text>", "completion": "<ideal response>"}
{"prompt": "<text>", "completion": "<ideal response>"}
```

```
{"prompt": "<text>", "completion": "<ideal response>"}
    ...
```

ChatGPT and InstructGPT work by the same prompt:response principle.

Exercises

Code File: Colab Notebook

Exercise 1: Prepare Data and Fine Tune model

Choose an application to fine tune GPT-3 on, where you may be able to find data of the prompt:response format. A great example of this could be jokes, which often take this form. Perhaps you want a specific type of humour, or styling?

Source your dataset. As always, the option to use Harper Valley is open to you (What if you prompt with a dialog act and it generates a sentence?), however, feel free to find or curate your own. You may wish to do this step first, and choose or alter your application based on what data is available.

Split your dataset into training and test data. The exact percentage split we leave to your own judgement.

Load your dataset and process it into the ”prompt”:”response” form given above, then use the training procedure written by OpenAI to fine tune your model.

Exercise 2: Evaluate and Discuss

Test it! Feed it unseen prompts. What did it capture well, what did it mess up on, and are there any failure modes you can identify? Where does it make use of your fine-tuning, and where is it still generating based on the original weights (i.e., the fine tuning did not affect that part of the model and so the weights are unchanged from the base model).

Construct new prompts. Show us some of your best/favourite examples, both of successes and failures.

Develop some metrics to observe the quality of your model. Has it lost some speech quality during fine tuning? Use ROUGE and BLEU scores to determine the grammatical/structural quality of the output responses, and similarity metrics (e.g. edit distance, or more nuanced variants) to compare them to the ideal response.

Exercise 3: Improve

Make at least one improvement. Can you process your data differently? Can you find more data? Can you re-factor your prompts or responses to better target your desired behaviour?

Optional Additional Exercise: Train your own language model!

This can be any LM you like (feel free to design your own if you're feeling up to it!). However, a suggested one would be taking Hugging Face's implementation of distilBERT, and training it on your corpus (or fine-tuning a pre-trained version). Whilst we will not reach the level of understanding that GPT-3 demonstrates, it is reasonable to train your own language model on a small set of data, to mimic human sentence generation. Can you make one write a poem?

We will not be providing starter code, but instead linking to several tutorials on this topic. It is a fun exercise and one that can nurture a greater appreciation for how these LLMs capture sentence information.

How a language model is trained:

- BERT (Bi-directional Encoder Representation with Transformers. Kind of a dumb name.) is an 'out-of-the-box' model. You can load it from scratch, which is the model architecture but not trained on any data, or you can laod a pretrained model that has already been trained on large amounts of text data.

- A new model will have no concept of language, think of it as a baby. Whatever you train it with, it will take as law, and learn the laws of its training set, probably in ways you don't expect (especially for small datasets). This has the advantage of more power to train a model of language from scratch.

- Pretrained models have the advantage of being trained on far larger datasets, and far more compute power, than any individual can hope to achieve. As a result, it already knows the basics of language and grammar. Disadvantages include less flexibility in how you want to apply it (you are dealing with an adolescent, it thinks it knows it all). However, fine tuning such a model can achieve some amazing results, by leveraging the knowledge it already has and simply encouraging it to specialize on a new dataset.

- Pick your route, depending on your goals. Do you want to train a baby to learn your rules and your rules alone? Or would you rather take a teenager and nudge them towards your task?

Materials Provided

- GPT-3 Fine Tuning Starter Code
- Tutorial links for LM training and fine-tuning

Readings

- https://huggingface.co/blog/how-to-train
- https://huggingface.co/transformers/v4.8.2/training.html