A GPT 2 Model on the Browser | Close but no Cigar

8:54:00 AM CODE, GPT2, JAVASCRIPT, MACHINE LEARING, TENSORFLOW JS, WEB
NO COMMENTS

I recently built an app that generates horror stories based on user provided input. It has branching pre-written options and randomness that pushes each story to be unique, even when the same actions are taken. You can check it out at feverdrea.me! I have a desire to expand it, but each additional branch to the story will about double the amount I need to write, and adding more options and random branches requires me to write a bunch of new segments as well. For reference, in total it was about 14,000 words and expanding the length of the story would about double the amount I need to write.

So I put the expansion on the backburner and moved onto other projects. Months later, after sending a friend feverdrea.me, they sent me a link to AI Dungeon, a game based on a GPT-2 model with some great generated writing. It immediately got me wondering if I could apply the same approach to generate unique stories for each person experiencing feverdrea.me!

I immediately got to researching how I could build a GPT-2 model. To be clear, I am not a ML developer, I'm an application developer. The closest I came was taking an image recognition course in college and opening the tensorflow page for about 20 seconds before nodding my head resolutely, and closing it. Which left me at square -1 when starting with this project.

Here is what I found. Btw, I would recommend holding off on jumping into creating a model without reading this, but you can find the colabatory notebook with my added steps here.

The Background

available language models. On top of that it offers you the ability to retrain it on a new set of data making it easy to create new models for specific purposes. For my project I managed to retrain the model with about 80KB of my own writing.

Open AI has released multiple models you can use or retrain with varying sizes ranging from around 130MB up to 1.5GB. Though I wanted to use the largest model it would have taken my GPU far too long to do so. Moreover, the collab edit does limit you to generating new model with the small and medium (355MB) models.

As I looked through samples of people using GPT-2 I noticed that almost every application I could find of gpt2 online, had a server evaluating the model. While I could have done this, I didn't want to spin up a backend in order to limit maintenance costs. I also don't care about people re-using my model so exposing it by sending it to the browser wasn't a concern for me.

Here was my core challenge, the methods I could find to generate GPT-2 models were for subsequent use in Python. Evidently pushing out a 100MB+ resource on web for users to use as a toy isn't a common use case for developers. My original plan was to try and use WebAssembly to get the model to run, but while you can run Python in interpreters written for WebAssembly[1][2], it would require me to push out not only my model and the logic to run it, but all the dependencies needed for the model as well (not fun to download, or for me to deploy)[3].

Enter ~~today's sponsor~~(wait a second, this isn't LTT, I don't have sponsors) TensorFlow and TensorFlow JS (TFjs). TensorFlow is platform that gives you tooling to build, generate, deploy, and run machine learning models. TensorFlow JS is a library within the TensorFlow toolkit that lets you build and run machine learning models in a JavaScript environment. I figured that if I could convert my model to TFjs I could then figure out a way to shrink it and create a good experience for users.

Well, it worked... sorta... let's dive into the approach and results.

The Approach

I settled on taking an incremental approach, to first focus on making a model, then converting it, getting it to run on my browser and finally optimizing it.

Model Generation

I could run model generation/finetuning locally, but I found this collab edit by Max Woolf that let me finetune the model via my browser on Google's servers (free real estate my friends). The instructions were abundantly clear and made it easy to upload my test data and run model generation. I would recommend reading a bit on how to format and cleanse your data as I spent a good number of iterations trying to improve my results by adjusting the format of my input data. For example, I realized that the removal "filler" lines with alliteration helped improve the comprehensibility of the model, and the addition of |<endoftext>| between segments of my story made it easier for the model to generate similarly sized story segments.

Original text sample:

```
You realize you hadn't fed it and start to look for the kibble you left in the kitchen. As you ram into the bag it sways and leans forward and then sways back towards you...
Click-clack, click-clack
Your six arms grab your prey, and you puncture a hole in its abdomen...
A guttural shriek erupts from your mouth, but it sounds nothing like you...
You realize that everyone else is staring at you and stop your screaming...
You hear the feet of hundreds of thousands just like you pound in unison...
```

Formatted text sample:

```
Something buzzes by your face, you stare down, to see a giant fly...
<|endoftext|>
Your six arms grab your prey, and you puncture a hole in its abdomen...
<|endoftext|>
You gingerly step forward, trying to avoid the debris that coats the floor...
<|endoftext|>
In excitement you pickup the pace, following your hands that guide you...
<|endoftext|>
You step back, trying to escape the horrors unleashed upon you in the room...
<|endoftext|>
```

I first tried the small dataset. but ended up switching to the medium set (355MB) for the last run. As I felt the results were slightly better. Here is some sample output for you to judge.

Small:

```
Your eyes are tipped forward and you blink away tears, you had missed your dad...
<|endoftext|>
You turn back to the other end of town and hear the funeral...
<|endoftext|>
You take the lead of the only chair in the room next to you...
<|endoftext|>
You feel the woman say "If you have to choose between me and your task, you make...
<|endoftext|>
You feel the woman leave the room, almost out of the mission window...
```

Medium:

```
you run but are stopped by a tall, burly figure. It's snarl is a mixture...
<|endoftext|>
Your caroms are too loud to ignore and you slow down even more to try...
<|endoftext|>
you run up to the counter and you pull the door open. A pair of straw...
<|endoftext|>
You catch a twinge of it that the shoplifters have been seen here...
<|endoftext|>
you run to your room and close the door behind you. The world...
<|endoftext|>
You unpack and pull down the duvet covering, it falls to the floor with a thump...
```

Conversion to TFjs (wololo)

With a Python model ready to go I got into converting it to JS, I ended up expanding on the colaboratory notebook to include these steps, you can find all the code for this section in there. I had seen some posts around the internet asking how to do this, but none of them were complete, either they were focused on just getting to TensorFlow js models or they would use the pre-generated models [1][2]. As a result the code may not be the cleanest way to do the conversion and I welcome feedback!

In short, I first had to convert the model to a pytorch model. I then converted that into a TensorFlow model. Finally I was able to convert my TensorFlow model to a TensorflowJS model. The first two steps were done with the transformers from hugging face, while the last

was done with one of the tools from TensorFlow.

With my model generated, I was able to download it (it was 500 MB whether I used a small or medium gpt2 model, o.0) and add it into a html page I was running locally. I found this easiest to do with firefox as it makes it easy to enable the loading of a local file into your webpage using the privacy.file_unique_origin flag (make sure to turn this back off after you are done testing). Once again, if you want to do this yourself, check out the colaboratory page.

With the process I used out of the way, lets get into the results.

The Result

You can see that the output from the python model above made sense thematically, but isn't fully coherent. This wasn't an issue for me as the stories in feverdrea.me are kind of kooky and unhinged.

I managed to to build my model, convert it into a TensorFlow JS model, get it to load on my browser, and even evaluate it there. Here, is where my limited understanding of Tensorflow and ML have stopped me in my tracks. After digging through the docs, I still don't know what the parameters I am passing in mean. Moreover the results returned to me are an array of floats which I don't know how to interpret.

Here is the input I am passing into the model:

```
input = tf.tensor([70,1,1,1,1], [1,5], 'int32');
const predictions = model.predict(input, [1,1]);
```

And here is an example of the output:

```
» predictions2[0].dataSync()
← Float32Array [ -0.8916228413581848, -0.13784034550189972,
  0.32917171716690063, -0.33279359340667725,
  -0.041885871440172195, -0.2832326292991638,
  0.46863073110580444, -0.37489521503448486, 0.2765974998474121,
  0.42103514075279236, 7670 more… ]
```

One of the last things I found, but didn't build on, was this example on github passing data into a BERT language model in TensorFlow JS. While this is a different model, the process looks promising and the processing and parsing they are doing is probably what I am missing in my usage. I also saw that hugging face has tokenizers for doing this for Bert models, so it could be worth making a distilBert model instead of a GPT-2 one.

Finally, if I manage to get the JS model working and prove it's value, I would next need to research how to minimize the size of it so it can load quickly.

Hopefully this is helpful to someone else, and they can put together and share whatever steps I am missing to make this a reality. If you find anything, please let me know in the comments, I would love to come back and update feverdrea.me with it!

Newer Post Older Post

0 comments:

Post a Comment

To leave a comment, click the button below to sign in with Google.

SIGN IN WITH GOOGLE

Copyright © 2023 Bit by Bit

Powered by Blogger.