Gregory Knapp

CS401R

Final Merging Write Up

All of Section 1 decided to work together to try and get the merging to work. To do this, we split up the members into 3 teams, one to set up the fine-tuned GPT2 for quest generation, another to implement the templates for the quests, and one to connect it to Neo4J. The team breakdown was: Nathan and Braden for the finetuned GPT2, Adam, Trisha and Jackson on the templating, and Abby and I on the Neo4J querying. The process seemed simple enough but as we started to implement it, a lot of little but annoying problems came up.

We started in class to try and put it all together. Everyone sent me their code over Slack and I connected my computer to the screen to try and put it together. The first thing we ran into was dependency problems. It was likely mostly since I still have Python 2 on my computer, but even things like spacy were giving me a hard time when I tried to install it. I did find that working in a fresh virtual environment made things a lot easier and I was able to get the dependencies working that way.

The next problem was dealing with the large file size of the model. Nathan worked out using Git-lfs and gitignore to get the files uploaded and saved to the repo which worked out well but took time to sort that out since without the model we could not even do the first step of the pipeline.

The biggest issue of them all however had to do with the model. I was assigned to do the assembly coding and worked a lot without outside of class since we did not finish then and ran into a really big problem trying to load Braden’s model. On my Windows computer at home, everything was working just fine, I could load the model into a Huggingface Pipeline and things worked great. As soon as I tried it on a Linux or Mac (I used my laptop and one of the cs lab machines), for some reason the Pipeline just refused to work. It wanted the model to be stored on a Huggingface account and demanded a user auth token which was not build into the config json file. It made even less sense since it was working just fine on Windows at home but did not work on Abbi’s windows computer when she tried it. I think in the long run, it would be best to retrain a model, but save it as a pt file or a tar file and load the checkpoint in PyTorch, or some other method. The Huggingface pipeline is great for the pre-trained models, but really was frustrating for a fine-tuned custom model, although I don’t think it was supposed to be this hard.

Since the third issue prevented me from getting the whole pipeline working on my laptop and time was running out, I just decided to get it working at home and screen record an instance of it doing what we intended it to. The output was simple, but it replaced the name with a value from Abbi’s knowledge graph.

Looking back as the person who tried to put everything together, while it was a good experience to try and use different people’s code to make the whole pipeline work, I think we would have had more success if we had each person try to implement the pipeline themselves using the code they wrote since everyone’s code did not work well together. For example, Adam’s template generator generated templated with the values in all caps (PERSON, ORG, LOC), but Abbi’s knowledge graph filling code was not made with this in mind. Even though we adjusted the output, the code still was not very good at handling the output of Adam’s spacy pipeline and sometimes crashed because the Neo4J queries did not know what to search for. It also struggled with handling multiple tags in the templates, another side affect of the two parts of the pipeline not really linking up.

There were also a lot of things that I did not think about before that came up along the way as well. For example, the game text box does cannot handle capital letters, so any text sent into the chatbot goes in lower case. I learned that Braden’s GPT2 did not like the lowercase input and would often create very messy output when it got it, likely because it was assuming it was already in the middle of a sentence as opposed to generating from the beginning. On a related note, when the model received lower-case input, it would generate pretty much all lowercase output. That normally would be fine, but Adam’s spacy pipeline would look for upper-case words to identify as people, places, or organizations, which made Braden’s model output useless. To combat this, I just hard coded a line to capitalize the first word of the input which improved the output and at least generated some proper nouns that spacy found; just another little inconvenience. Similarly, after each step of the pipeline, the original output string would get distorted and filled with random spaces and newlines, either from gpt2 or spacy, so I found myself constantly parsing and trying to clean up the output (which was not very successful as the video shows).

All in all, it was nice to have some form of the pipeline there, but it is not stable at all which I think is since it is made up of parts that are not well-suited to be assembled. For the sake of a learning experience in a class, that is absolutely okay and I’m glad that we worked on it as a section. There were a lot of great ideas and methods put into the different parts of it, they just all don’t exactly click with each other, and it will probably be easier to try and rework parts from scratch or an earlier stage than it would be to try and force the code in its current state to work better.

Edit 4/13: After everyone’s work in class today, the 2 main problems above were fixed. Braden uploaded the model to Huggingface and now it all works without using GitLFS and Abbi refined the querying code and it has way less crashes, mainly just when the fine tuned model tries to include WoW specific place holders like < name > or < race > in the output. With those two problems fixed, the pipeline works much more consistently (and across the different operating systems) and quickly, at least in comparison to what how long I thought it would take initially.

Work time: 15 hours between the work of the different groups and my work assembling things at home.

Code: Everything in its current state should be pushed to the 2 GitHub Repositories.

Example of the pipeline processing: (print statements in server.py)

GPT2 Output: Kevin how are you going to teach the beast and we can get him back? We want her back first but she doesn't know what to do.

Fixed template:   <PERSON1> how are you going to teach the beast and we can get him back ? We want her back first b ut she does n't know what to do .

Final Quest:  5->Plr: Aziraphale how are you going to teach the beast and we can get him back ? We want her back first b ut she does n't know what to do