Progress Report

1. Novelty
    a. Code summarization (in this case automatic documentation generation) is a popular research task that involves translating from source code to natural language descriptions of the source code. Transformers are considered the state-of-the-art model for tasks involving sequential data, and is seeing increased use in code summarization. It is generally thought in the field of source code language modeling that structural information (e.g., from the AST) is useful in representing source code. **From what I've seen, there has been limited work on the incorporation of structural information in Transformers for documentation generation and several recent papers on source code language modeling mention this as a possible extension to their work.** Ahmad et al. 2020 briefly makes an attempt while Chirkova and Troshin 2020 conduct a fairly broad and deep investigation into incorporating ASTs in Transformers, but does evaluate their approaches on documentation generation.
2. Value to User Community
    a. Most of the benefit will be to researchers in this topic (documentation generation). Specifically, I will be trying out a combination of techniques that is relatively novel, as seen by the lack of existing work on this specific subtopic. The main research question will be something along the lines of "Does explicitly incorporating structural information to Transformers improve their performance in documentation generation?" By evaluating and comparing these techniques against existing ones, we will have a better idea of the impact of structural information on source code summarization using Transformers.
    b. Another benefit (to researchers in this topic) may be increasing awareness and use of improved dataset processing practices described by Leclair and McMillan 2019, practices which, according to the authors, improve the reproducibility of research in this topic.
3. Dataset(s)
    a. Python 150k
        i. This includes parsed ASTs and their source files. Parsing ASTs was a concern of mine, so it's convenient that this dataset has already serialized them as JSON. A large part of the challenge will be figuring out how to ingest/represent the ASTs in the model.
        ii. Chirkova and Troshin use this dataset and describe specifically how they use it. They do evaluate on a method name generation task in their paper, which is similar to documentation generation, so I will try to follow their procedures for processing this dataset.
    b. Ahmad et al. 2020 dataset and preprocessing (less likely)
        i. Working with this dataset may make it easier to compare to this work, but their procedure has data leakage issues. Also, from a brief examination, I don't think they included their AST parsing logic in their git repo, so I would have to do more preprocessing work on this dataset.
4. Comparison Subjects:
    a. Ahmad et al. 2020

        i. Same task; I'll also be trying to build on their Transformer model. If I use their data preprocessing procedures, I can potentially directly compare my results with theirs. Otherwise, I'll need to adapt their provided code and run their model on the dataset I use for my model. This will basically be running a text-only Transformer instead of a text + AST transformer, so aside from the additional training time, it shouldn't be too terrible. This would also serve as a natural ablation comparison.
        ii. I should probably keep track of model parameter count and other architectural differences as well as training time differences. Time (and compute resources) permitting, I could vary model size and compare performance.
    b. Chirkova and Troshin 2020
        i. They use different tasks, but I'm trying to follow their methodology for adapting the Ahmad et al. Transformer and some improved data preprocessing practices they demonstrate.
        ii. As far as I know, these authors didn't make their code available, so I may try to reach out if I run into issues/confusion.

5. Deliverables
    a. Written work:
        i. Courseworks
            1. This should include a table comparing performance on standard metrics (ROUGE, METEOR, etc.) for at least one variant of the text-only Transformer and one variant of the AST-augmented Transformer. Other variants include:
                a. AST representation variants
                b. Parameter counts/sequence lengths
    b. Code:
        i. github, ideally structured and documented so that using it is as easy as a model imported from some standard lib (e.g. pytorch, hugging face). I liked that CodeBERT, which I used for my midterm paper, once installed, was able to fine-tune and run using 2 (very long) lines in the terminal. I'd like to provide a similar experience (though I won't be offering a pretrained model).
    c. Dataset:
        i. github for dataset extraction code
        ii. potentially zenodo for the actual dataset if I end up doing anything noteworthy to it, otherwise it is an existing dataset
6. Resources
    a. Dataset:
        i. Python 150k: https://eth-sri.github.io/py150
    b. Code:
        i. Ahmad et al. 2020 model and related code: https://github.com/wasiahmad/NeuralCodeSum