PROGRESS REPORT

The Initial plan was to come up with a language model that generates sequences which are valid molecules. For this purpose I built a language model with 2 layers of LSTMs and a time distributed dense layer. This model was not efficient as there was lots of noise introduced due to padding.To compensate for the padding, the best alternative was to use embeddings. So an embedding layer was introduced in the model. But instead of using LSTMS, GRUS were used as suggested in the paper.The Keras LSTM implementation did not produce good results due to technical reasons which were solved in the GRU Pytorch implementation.The code was refactored so as to fit the paper, this model was much better than the former, it not only did train quickly but also generated valid molecules with high accuracy. The next approach was to how to put it in our use case.
As suggested by my guide, I started working on GAN based approaches. This was not as fruitful as expected since the research in this area was a bit old (3 years) and not all the papers provided any code to back up their claims of better performance. They were all experimenting with lots of future potential but cannot be directly used in our use case.
However in the recent past (2019 December) a paper was published with promising results but it was not entirely of our use case but has promising approaches so I am yet to read this paper. (If I restart my work on GANs.)
Since GANS did not quite meet the standards, I started looking for alternatives and worked on Reinforcement Learning methods. As it was the best big shot after GANs.
Fortunately there was a really good paper which performed very well and also addressed something similar to our use case. I took ideas from their Reinforcement methods. Wasting no time I absorbed all the concepts and implemented this paper according to our use case therefore made few changes and the model was the best model to be implemented.
The model was perfectly fitting our case hence the next approach was to make this model better. The language model used in this paper was remastered before a year hence there was no scope of a huge improvement except for upscaling (adding new GRU layers) and retrain but this would produce a very minute improvement or none at all. The reinforcement approach was concrete, only change we can try is tweaking few hyperparameters again the authors have performed that so no much scope there. Under the hood the model uses molecular fingerprints for molecules. It is the fingerprint which decides the sequences to be made so the fingerprint generation is very important. The model currently uses RDkit's morgan fingerprint with a radius of 4, I believe the fingerprint generation could be improved. So my next research was to make a better fingerprint generator and come up with a unique approach.
Molecular fingerprints are not as popular ; they are only used in molecule property predictions so the research was quite limited to GNNs and RNNS, I read too many papers in this topic to point all of them. It was clearly visible that Graph Neural Networks would be the go to approach since chemical molecules and graph networks have homologous structures. Nodes as atoms and edges as bonds. I did not implement all the papers due to the fact that they were way too many, so I decided to only implement SOTA but it was not quite clear which was the SOTA, since there is no particular dataset like ImageNet for Image classification but recently a dataset

called MoleculeNet was curated. One model which achieves SOTA in many subtasks was earmarked for implementation.
All of these models are property prediction tasks so the model can not be directly used as our use case requires a fingerprint.
So the model was tweaked but the hindrance is the lack of trained weights as the authors have not open sourced it and did not respond to any of my mails.So this approach remains stagnant.

As the current NLP Domain uses Transformers everywhere, I researched in the transformer field as well, both for a better language model and if possible a molecular feature extractor model. As our model needs both.
Language models using Transformers which use SMILES have been published in the recent past but they did not have any code or weights. Training a Transformer model in BERT fashion from scratch is not feasible as the paper claims it took them 2 weeks on a decent GPU. But there were few works on molecular predictions which we can alter to extract features. and more importantly this uses transformers. Two significant ones are MAT and SMILES Transformers. Both were implemented and both had to be tweaked under the hood to get a fingerprint like a (n-dimensional vector) as output instead of scalar 1 dimensional output.
Fingerprint Model Implementations

1. MAT
2. SMILES Transformer
3. Naive GNN

From the results it was pretty much evident that MAT Paper was a better one owing to the use of Graph Transformers instead of Transformers. However there are few issues running it on google colab. I have opened a github issue on the same but no one has responded. Recently I have been in contact with the paper's author to whom I told my approach. I am yet to get a reply from him.
Numerous other works like drug target Interactions, chemical reaction predictions were explored but they were not as promising as it was not our use case.
Now the current work is to somehow solve the colab error which has no solutions on stack overflow or github and rewrite the fingerprint similarity score calculator function and also change the MAT input formats.

References:
All reference links added as hyperlinks.