****

|  |  |
| --- | --- |
| SKY-LABS CAPSTONE PROJECT | Abstract  We finetuned the T5Codebase model for LaTeX-to-Python translation, focusing on data augmentation to improve accuracy. Various experiments yielded incremental gains despite challenges like CUDA errors and time constraints.  Saad, Ahmed, Irfan, Quindeel  **26/08/2024** |

**Current Approach:**

**We are using the t5codebase model, because the problem is restricted to a small subset of equations, also since the task of latex to python is a sequence to sequence task we prefer then encoder-decoder architectures over the other approaches. We are enriching our training dataset by generating augmented equations, our current accuracy is the result of training the model on about 100,000 examples. To generalize better we have included a large variety of augmented equations by combining many different equation types given in the training data.**

****

**Approaches Experimented Upon:**

1. **We tried mixture of experts, found a potential base model that could work for latex part but wasn’t able to load there were some errors its name was “Chan-Y/Florence-2-LaTex”. It was finetuned for mathematical latex code generation, while we wanted to use the t5codebase as the source model.**
2. **Data corruption where we corrupted the latex expressions and kept the python code same so that it may learn to generate code even for faulty latex expressions but the results were sub-optimal.**
3. **Using larger models i.e., t5codelarge and smaller models t5codesmall, the results on both the variants were bad. The smaller couldn’t learn much and the bigger one had to learn a lot.**
4. **Adding noise, we changed the braces, cos-sin, removed random spaces in both the latex and python codes, this helped in increasing the accuracy by 7-9% if the noise was about 10-15%.**
5. **We tried to generate high quality data using chat-gpt prompts, that were highly optimized, included seed values and few shot examples for generating latex and python code. We made about 5000 examples in 5 hours and didn’t continue because it was too time consuming and the generation speed of gpt is slow after 5pm.**
6. **We also tried different prompts in codet5base tokenizing and inference times, we found that using bigger prompts resulted in garbage results, one sentence prompts like convert latex to python worked.**
7. **We also used the llama-7b to convert the augmented equation’s python code into multiple lines. Since it was generated using sympy we had long statements in the return line. That caused the missing braces errors about 100 during inference. It took about 5 hours to process 5000 examples and the results were trash, in some cases it even gave text like “yes I will help you in the given task ….”.**
8. **We tried giving different data sizes to the model and learned that giving 100,000 examples can increase the accuracy about 1-2% when you are struck like at 30,000 data. Increasing it even more didn’t cause any significant changes.**
9. **We tried another approach which was a long one where we finetuned first on the “code\_search\_net” python dataset to make mode focus more on python data, then we finetuned it on large data of latex equations “Kyudan/arXiv\_latex”. Afterwards we trained on the latex to python data (only the training part). We found out that this approach increased the accuracy from 57% to about 65%. We could use the datasets of python and latex equations fully as there were cuda errors and time constraints but it is an interesting thing to test how much further can this be improved.**
10. **We experimented upon training the model like a baby, we gave it simple latex equations alongside python code, then a combination of 2 equation types and so on till 4 types. Our results weren’t great, since we did that long before we made the augmented data quality better. It should have better results but we couldn’t try it later on due to time constraints.**
11. **We had an interesting idea to split the latex equations into parts alongside the python code and then let the model learn the chunks but it was too complex and the splitting couldn’t be made accurately on the data we had because a lot of equations had function definition and a long return statement.**
12. **At the inference stage, we did postprocessing, and during that in the last meeting with sky-labs we had an idea about using the llama 7b when there were errors in generated code like missing braces etc. There was an interesting issue or bug, the llama model corrected the mistakes it even corrected wrong function definitions, some evaluations in the derivatives, but when it returned the code string and we executed it again, there was a random indent error. We tried saving code in json files and checking it if it had any errors in indentation but there were none somehow this interaction between exec() and the generated code from llama wasn’t working. We even tried to remove spaces in the string returned but they were still there.**
13. **After playing a lot with the synthetic data we found that if we didn’t use the training data to create more data we have bad results, i.e., the distribution had to be same. Moreover, showing too much diverse data ended up in model learning nothing.**
14. **We even tried finetuning models with quantization (peft and lora) but the training time was large so we didn’t proceed with that approach.**
15. **We also tried using different scores like bleu, but the issue was during backpropagation we have cuda out of memory errors.**
16. **Another approach we used was the custom losses, when there were a lot of braces errors, we designed a loss that counted the braces and if there was a mismatch we added an additional penalty, and its effect was controlled by alpha, we kept alpha = 0.7 (more effect of original loss and lees effect of this loss) and didn’t achieve any improvements, when we changed it to 0.4 it had great improvements on the outputs. Moreover, there was another idea similar to this where we used exec() to check whether the function has any errors or not, we tried it once by giving a 10 penalty when ever the function couldn’t be executed, since it was too hard we got trash outputs when inferencing we wanted to try using something like a 4 or 5 but didn’t had time.**
17. **We also thought about maybe training the tokenizer a weird thing chat-gpt told me about to better handle latex expressions, we had the data of latex equations but didn’t go ahead with this approach since it didn’t seem much logical.**

****

**There might be more things to tell but even we don’t remember all of them since there were so many ideas and so less time to try them all out. Thank you for this amazing opportunity.**

****