HW1: MSP Pre-training for Code

CS597LMZ

1 Goal

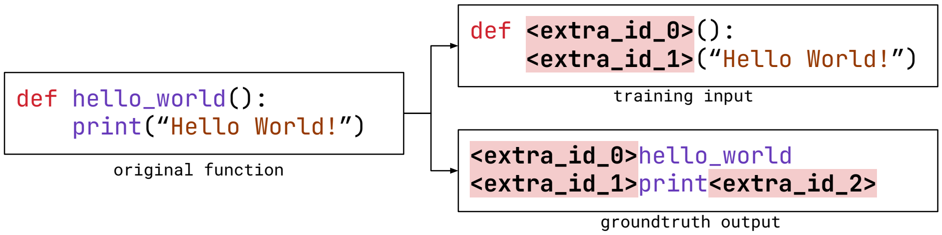
In this assignment you will train a code model using the **Masked Span Prediction** (MSP) objective used in T5 and CodeT5 encoder-decoder models. The goal is to transform the base T5 model to a code version by pre-training on code data.

1.1 Masked Span Prediction

We focus our training procedure to use Masked Span Prediction (MSP) commonly applied in encoder-decoder model. The procedure of MSP involves randomly replacing sequences of the training sample with mask tokens. The goal of the training is to then recover the original groundtruth of all the tokens which has been masked out.

Slighty more formally, given a sequence of tokens $X = \{x_1, ..., x_n\}$, random sequences of tokens are replaced with masked span tokens to produce $X_{masked} = \{x_1, ..., \texttt{<SPAN>}, x_n\}$. Let $M = \{m_1, ..., m_k\}$ be the tokens masked out, $M_{<g} = \{m_1, m_2, ..., m_{g-1}\}$ be token sequence predicted so far where $g \leq k$, P be the predictor (model) which outputs the probability of a token. The MSP loss function can be described as:

$$\mathcal{L}_{MSP} = -\frac{1}{k}\sum_{i=1}^{k} log\left(P\left(m_i \mid X_{masked},\ M_{<i}\right)\right) \tag{1}$$



```
def hello_world():
    print("Hello World!")
```

```
def <extra_id_0>():
    <extra_id_1>("Hello World!")
```

```
<extra_id_0>hello_world
<extra_id_1>print<extra_id_2>
```

Figure 1: Masked Span Prediction example

Figure 1 shows concretely what we expect the MSP objective input to be for this assignment. We first start with the original function (which contains a complete code. Note for your training input you may also include the function docstring as well). We then separate it into the training input by replacing random spans of tokens with span tokens (i.e., <extra_id_0> and <extra_id_1>). Note that the value 0 and 1 at the end of <extra_id_ indicates the order of the span token (from beginning to end) which allows the model to indicate in the output which span they are referring to. We also obtain the groundtruth output by keeping the tokens we masked out previously. We also add the span tokens which as mentioned before indicates which span its referring to and the tokens that directly follow it are the groundtruth original span. The goal of training is to then guide the model towards being able to recover the masked out tokens when given only the masked out training input.

While there could be many different ways to implement MSP (i.e., which spans to pick to mask out, how many span tokens should there be – these are all interesting research questions!), in this homework, you can randomly pick the span locations as well as the length of the span. The only requirement is to ensure that the percent of tokens you masked out is around 15% of the total input tokens. Note: since we are using specific models, please follow the training input and groundtruth output format shown in Figure 1.

2 Instructions

To complete this assignment, following the steps described below to complete both the training and inference task.

2.1 Training

- First use the Jupyter Notebook file provided in the homework assignment GitHub: https://github.com/uiuc-cs598lmz-s24/hw1/blob/main/pretraining.ipynb and follow the initial setup steps. Ensure you have changed the hardware accelerator to GPU if are using colab. Note: if you are using colab for the first time please read through the colab faq and/or follow some initial guides.

- The initial setup steps include downloading the training file used for pre-training. In this homework, we will be using a subset of the **CodeSearchNet**: https://zenodo.org/records/7908468, which contains code functions with both function body and docstring. Due to the limitation of computing resources, we will only be using one training file which contains 30,000 functions. Following, we provide a simple function `grab_raw_dataset` to extract the raw data from the training file. Please take a moment and play around with the raw dataset to understand the different fields in the raw dataset that you will use to construct the MSP training dataset.

- Next, load the base model - T5 (`T5-small`) and the corresponding tokenizer. In the notebook, we have provided the function `load_base_model` to do that. Feel free to also play around with the base model to understand the usage and the limitation, especially when given code as input.

- Implement the `prepare_dataset` function that turns the raw dataset into a MSP training dataset (see Section 1.1 for more detail). In this homework assignment, we aim to implement the 15% masked rate. That is to say, in the training dataset, approximately 15% of the tokens should be masked out and the model is trained to recover these masked tokens. In short, the format of the training data should include the `input_ids` – tokenized input for the model (with random spans masked out) and `labels` – the tokenized groundtruth of the masked out tokens that the model aims to recover during training. Ensure the final crafted dataset follows the format of the `Dataset` class provided.

- Using the base model and the training dataset, produce a pre-trained model. We have provided the simple training function, however feel free to modify the hyperparameters or apply other model training optimizations as you see fit. Note: you might want to save the trained model in a more non-ephemeral location in case colab removes your runtime.

2.2 Inference

- First, lets download the inference dataset task here: https://github.com/uiuc-cs598lmz-s24/hw1/blob/main/hw_1_inference_dataset.jsonl. The inference dataset is for code generation and it contains 1,000 input and desired output pair. Run the provided `grab_inference_dataset` to extract the inference dataset. The inference task is to perform code completion (in particular, code infilling) to fill-in code in the middle of the function when given the prefix and suffix.

- Implement the code infilling inference function `code_infill`. In this function you should format the input in the inference dataset to fit the expected input format of the model expects to perform the task successfully (hint: think about how the training task is done and replicate the input format also for inference). To generate the model output, you should use greedy decoding with sampling temperature = 0. Note you may choose to implement some post processing steps to further help the model (e.g., you can segment the model output to remove any irrelevant tokens)

- Execute both the base T5 model as well as your trained model on the inference task and report their results. For this assignment, we will use edit distance (specifically, levenshtein) to evaluate the models. Please use the same inference function for both base and trained model.

- Note: the trained model should achieve lower edit distance score than the base model (lower is better). If after this step your trained model has higher edit distance, please check your training procedure (e.g., hyperparameters) and dataset creation step (e.g., did you correctly emulate the same inference task as the your training dataset?) to ensure everything is correct. If after that the score is still worse than the base model, please record the result and explain a bit why the result is worse in the report.

2.3 Report

- Describe the Masked Span Prediction training procedure and how you implemented the dataset
- Log the inference accuracy of both the base model and the trained model
- Provide an explanation on why it performed well or why it did not perform well on the inference task.

3 Deliverables & Grading

You are expected to upload a zip file without any folders. The name of the zip file should be your NetID. The zip file should contain only the following:

- (16pt = 10pt + 6pt) `code.py` including both the filled out training implementation and inference implementation (Note: if working on colab you can save the notebook as a python file)
- `result.md` logging both the base model and trained model on the inference task. The format should be two lines:

  ```
  Base Model: {edit_distance}
  Trained Model: {edit_distance}
  ```

- (4 pt) `report.pdf` maximum one page report to answer the corresponding questions (see Section 2.3)

Warning: Not following the above format will result in deduction in marks.

4 Resources

- Note this project can be completed on google colab, see this for more detail: https://research.google.com/colaboratory/faq.html However you are more than welcome to use any GPUs you have access to
- Please read through the CodeT5 paper for more detail about the training objective: https://arxiv.org/pdf/2109.00859.pdf
- Base T5 model: https://huggingface.co/docs/transformers/model_doc/t5