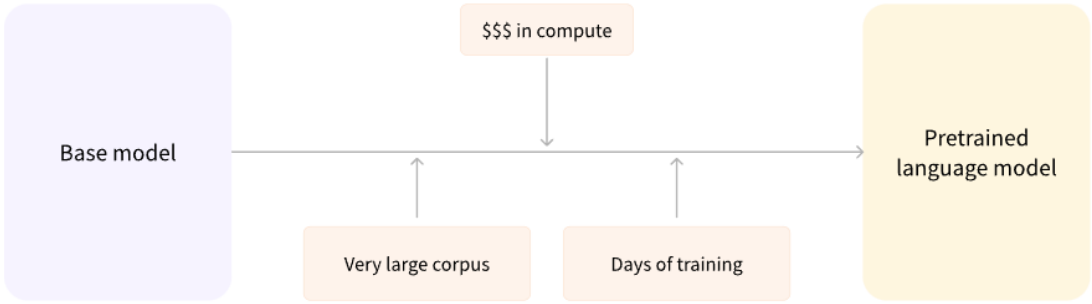
Lab2 实验报告

高健翔 521021910197

一．实验设计：

采用 transformer 库中预训练的 GPT2 模型(属于解码模型)，在自行构造的数据集上进行迁移学习，转化为面向代码生成的模型，相当于将已经模型已经获得的知识转移到了新的领域，即本实验中的代码生成。



由于预训练的模型已经在大规模语料中进行训练，再使用代码语料进行微调，达到的效果可能比从头训练一个还好。

二．训练方法：

首先尝试将源代码放在自己电脑上运行，发现耗费时间较长且内存不够，于是将代码上传到 kaggle 上，采取训练一轮后停下，查看 log 并调整一些参数，然后在原来的基础上继续训练的方式。

由于原代码只提供了 java 和 python 两种语言的训练方式，我在将这两种语言进行训练之后又在数据集中添加了 cpp 语言进行训练，数据集通过 python 代码利用 transformer 库提供的分词器将 java，python，cpp 代码由 token 编码为 input_id，输入内容如下图所示：


```
{"token_ids": [220, 220, 220, 1782, 2073, 1391, 198, 220, 220, 220, 220, 220, 220, 220, 220, 220, 220, 220, 1441, 6376, 26,
  198, 220, 220, 220, 220, 220, 220, 220, 1782, 198, 220, 220, 220, 1782, 198, 220, 220, 220, 3373, 14176, 262, 938, 5002,
 198, 220, 220, 220, 611, 7, 14421, 11405, 5145, 3258, 13, 28920, 3419, 11405, 5240, 58, 28968, 10, 16, 60, 6624, 1988,
 19953, 198, 220, 220, 220, 220, 220, 220, 220, 220, 1441, 11677, 10, 16, 26, 198, 220, 220, 220, 1782, 198, 220, 220, 220,
 3373, 1988, 373, 407, 1043, 11, 1441, 532, 16, 198, 220, 220, 220, 1441, 532, 16, 26, 198, 92, 198, 198, 35343, 198, 1635,
  2488, 65, 3796, 4738, 5254, 329, 10627, 2854, 618, 281, 7177, 1595, 470, 3994, 281, 5002, 198, 16208, 198, 30388, 645,
 62, 13966, 495, 1198, 62, 41989, 39893, 198, 220, 220, 220, 20512, 3804, 796, 2081, 26, 198, 220, 220, 220, 493, 43720,
 62, 22510, 11, 43720, 62, 8367, 11, 6376, 11, 997, 62, 41989, 796, 8576, 26, 198, 220, 220, 220, 14367, 3712, 31364, 27,
 600, 29, 5240, 26, 198, 220, 220, 220, 981, 7, 22510, 62, 41989, 438, 19953, 198, 220, 220, 220, 220, 220], "label":
 "C-Plus-Plus"}
```


然后由 dataset 类根据语言类型贴上对应的标签<java><cpp><python>对其进行编码之后与 input_id 和 eos 进行拼接。

以下是对于代码的修改以适应上述的变化：

```
os.makedirs(output_path, exist_ok=True)
if os.listdir(output_path) and not restore_training:
    out = input(
        "Output directory ({}) already exists and is not empty, you wanna remove it before start? (y/n)".format(
            output_path))
    if out == "y":
        shutil.rmtree(output_path)
        os.makedirs(output_path, exist_ok=True)
    # else:
    #     raise ValueError("Output directory ({}) already exists and is not empty".format(
    #         output_path))
```

注释掉了是否删除./tmp 文件夹的逻辑，kaggle 好像无法应对这种运行时输入的情况.

```python
if path == "Python":
    source_files = glob.glob(f'{path}/**/*.py', recursive=True)
elif path == "Java":
    source_files = glob.glob(f'{path}/**/*.java', recursive=True)
else:
    source_files = glob.glob(f'{path}/**/*.cpp', recursive=True)
for each_src in tqdm(source_files)\
        :
    with open(each_src, "r", encoding="utf-8") as f:
        code_content = f.read()
        encoded = gpt2_tok.encode(code_content)
        for i in range(len(encoded) // args.stride):
            seg = encoded[i * args.stride:i * args.stride + args.segment_len]
            if path not in segments:
                segments[path] = []
            segments[path].append(json.dumps({"token_ids": seg, "label": path}))
```

在由源文件 tokens 转化为编码的时候(encode)添加 c++语言的部分

```python
def _build(self, file_path, model):
    with open(file_path) as f:
        for line in tqdm(f):
            example = json.loads(line.strip())
            if example["label"].lower() == "python":
                encoded_plus = model.tokenizer.encode_plus(
                    model.tokenize("<python>") + example["token_ids"] + [model.eos_token_id],
                    max_length=model.max_seq_length)
            elif example["label"].lower() == "java":
                encoded_plus = model.tokenizer.encode_plus(
                    model.tokenize("<java>") + example["token_ids"] + [model.eos_token_id],
                    max_length=model.max_seq_length)
            else:  # example["label"].lower() == "c-plus-plus":
                encoded_plus = model.tokenizer.encode_plus(
                    model.tokenize("<cpp>") + example["token_ids"] + [model.eos_token_id],
                    max_length=model.max_seq_length)
            self.inputs.append(encoded_plus.data)
```

在 build dataset 过程中添加 C++数据集

```python
#initialize model by model name (the same as used in transformers lib)
model = GPTSingleHead("./model/0_GPTSingleHead/", max_seq_length=args.max_seq_length)
# model = GPTSingleHead()
# model = GPT2LMHeadModel.from_pretrained("./tmp/model/distilgpt2_fine_tuned_coder/0_GPTSingleHead/")
#add special tokens for controlling code generation by different programming language
model.add_special_words({"pad_token": "<pad>", "additional_special_tokens": ["<python>", "<java>", "<cpp>"]})
#load training dataset
file_path = dataset_folder + "train.jsonl"
train_dataset = SrcCodeDataset(file_path, model, cache_path=os.path.join(".cache", output_path, "train"))
```

在初始化过程中添加<cpp>标签，并在训练一轮过后更改加载 model 的路径——从本地训练过的模型进行加载

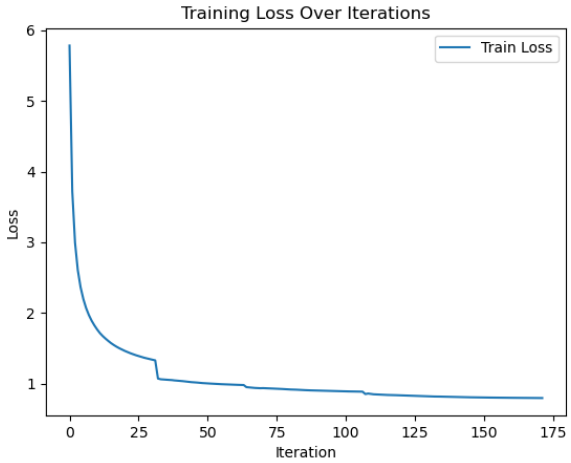
三．训练过程及结果：

1. 在自己电脑上运行的过程中，观察 log 发现 loss 有一个大幅降低的过程，因此在 kaggle 第一轮训练过程中适当调大学习率，减小评估的频繁程度，期望能尽快到达局部最小值附近，之后观察第一遍训练的 log，进行参数调整，尝试增大/减小 batchsize，增加评估的频繁程度，使得 loss 尽快收敛。

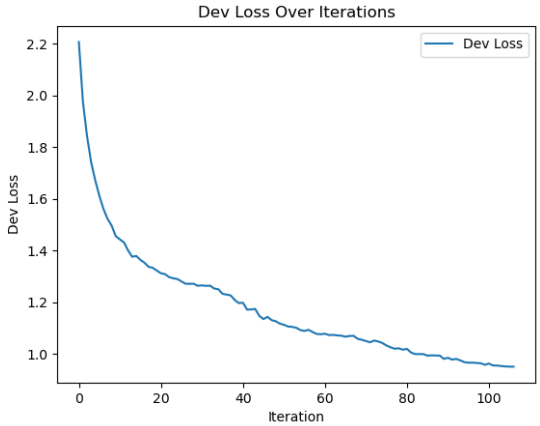
2. 由于原本代码只将 loss 写入 log 文件，因此我额外写了一份代码将 loss 值提取出来并绘制图像，代码如下：

```python
import re
import matplotlib.pyplot as plt

trainlosses = []
devloss = []
per = []
for i in range(1, 4):
    stri = "./p+j/log" + str(i) + ".out"
    with open(stri, 'r') as file:
        for line in file:
            # 使用正则表达式匹配包含"train_loss"的行
            match = re.search(r'train_loss = ([0-9.]+)', line)
            match2 = re.search(r'dev_loss = ([0-9.]+)', line)
            match3 = re.search(r'\'perplexity\': ([0-9.]+)', line)
            if match:
                # 提取train_loss的值并添加到数组中
                train_loss = float(match.group(1))
                trainlosses.append(train_loss)
            if match2:
                devl = float(match2.group(1))
                devloss.append(devl)
            if match3:
                pe = float(match3.group(1))
                per.append(pe)

plt.plot(trainlosses, label='Train Loss')
plt.xlabel('Iteration')
plt.ylabel('Loss')
plt.title('Training Loss Over Iterations')
plt.legend()
plt.show()

plt.plot(devloss, label='Dev Loss')
plt.xlabel('Iteration')
plt.ylabel('Dev Loss')
plt.title('Dev Loss Over Iterations')
plt.legend()
plt.show()
```

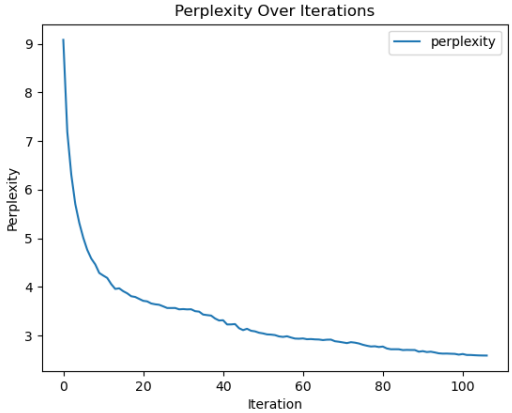
以下是对于只在<java><python>数据集上进行训练和在三种语言上进行训练的 loss 曲线：



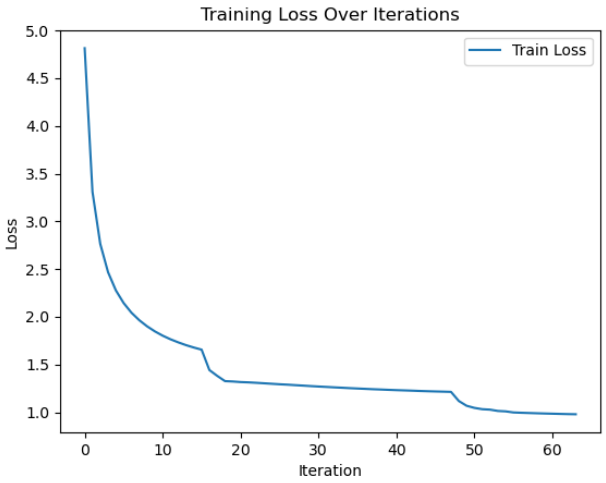
在<java><python>语料进行训练的 trainloss 曲线



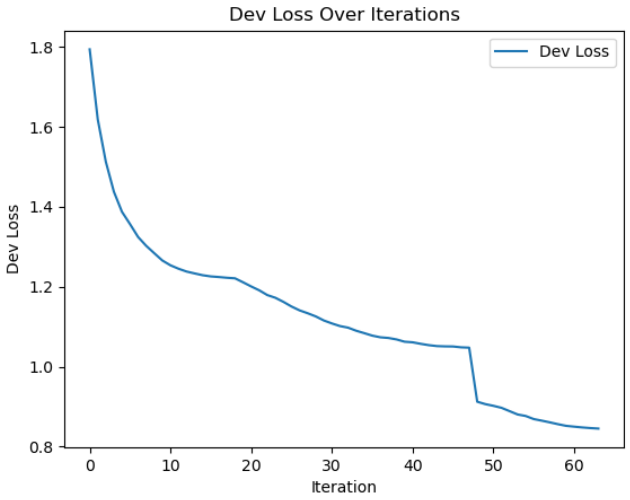
在<java><python>语料进行训练的 devloss 曲线



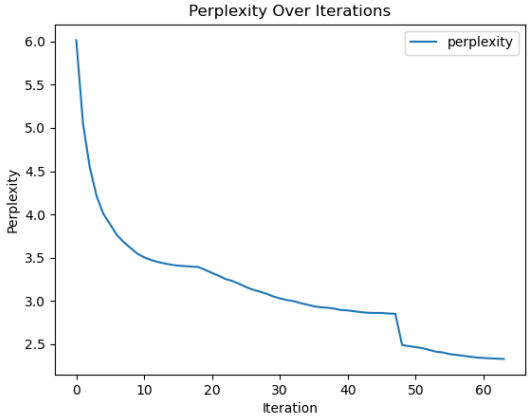
在<java><python>语料进行训练的 perplexity 曲线



在三种语言语料库进行训练的 trainloss 曲线



在三种语言语料库进行训练的 devloss 曲线



在三种语言语料库进行训练的 perplexity 曲线

对于 loss 曲线的分析：

1. 如图，可以看出 loss 曲线并不光滑，原因是我在 kaggle 上训练到第二轮的时候就会出现内存不足的情况，因此每训练一轮就会中止然后调整参数在原有的基础上继续训练，因此会出现突变的情况。
2. 在对<python><java>两种语料进行训练的过程中，出现了 perplexity 与 devloss

抖动的情况，集中在第二轮的训练过程，观察 log 发现，我在对于这批数据的训练过程中第一轮到第二轮并没有调整参数，可能原有的 batch_size 与学习率在这一阶段已经不合适，造成了在训练集上的过度拟合。

3. 基本上 loss 与模糊程度都处于下降状态，事实上，在最后一轮的训练过程中发现 loss 的下降已经非常缓慢，由于 kaggle 限制 gpu 运行时间，我不能再次调整参数以判断 loss 的究竟能收敛到什么程度，以下是运行的最终结果

```
dev_loss = 0.850175      ||          dev_eval_scores = {'perplexity': 2.340055227279663}
train_loss = 0.7981406450271606
```

<python><java>

```
dev_loss = 0.844905      ||          dev_eval_scores = {'perplexity': 2.327756881713867}
train_loss = 0.9783708453178406
```

<python><java><cpp>

3. 运行效果展示：

```
You are using cpp now. Enter the context code (exit or change_lang)
>>> int a
F:\anaconda\envs\gpt2\Lib\site-packages\transformers\generation\configu
  warnings.warn(
The attention mask and the pad token id were not set. As a consequence,
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated 0: <cpp>  int a, int b) {
        if (a < b) {
                a = b;
                b = b;
            } else {
                a = b;
                b = b;
             }
         }
        return a;
    }
```

```
You are using cpp now. Enter the context code (exit or change_lang)
>>> #include
F:\anaconda\envs\gpt2\Lib\site-packages\transformers\generation\configu
  warnings.warn(
The attention mask and the pad token id were not set. As a consequence,
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated 0: <cpp>  #include <iostream>  /// for IO operations
#include <vector>    /// for std::vector
```

```
>>> java
You are using java now. Enter the context code
>>> Scanner input= new Scanner(System.in)
F:\anaconda\envs\gpt2\Lib\site-packages\transformers\generation\configuratio
  warnings.warn(
The attention mask and the pad token id were not set. As a consequence, you
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated 0: <java>  Scanner input= new Scanner(System.in) {
        int n = input.nextInt();
        System.out.println("Enter the number of elements in the array: ");
        int[] arr = new int[n];
        int n = input.nextInt();
        int[] expected = {-1, -1, -1};
        assertEquals(expected, arr[n]);
    }
You are using java now. Enter the context code (exit or change_lang)
>>>
```

```
Generated 0: <python>  def sum_of_series(num_of_terms:int, power: int) -> int:
        """
        >>> solution(10)
        10
        >>> solution(20)
        30
        >>> solution(50)
        30
        >>> solution(50.0)
        30
        >>> solution(50.0)
        30
        """
        return sum_of_series(num_of_terms)
```

对于 python 和 cpp 代码的生成结果并不理想，可能是因为我训练次数有限，并且为了提高速度，我减小了 seq_len，导致 python 函数之后基本上读入很多注释，以及 cpp 文件读入很多#DEFINE 和#include 的内容。