1. Attention exploration

- (a) 提示：参考[slides]中注意力机制的相关内容。
  - (1) 作业中式(1)已经写得很明白了，这是一个模糊查询，我们不能直接通过查询向量q精确匹配到某个键向量k，只能赋予每个键一定的概率分布权重（即α_{ij}），得到最终的输出结果。
  - (2) 根据式(2)的计算方法，如果查询向量q与某个键k_i的相似度非常高（点积值很大），且q与其他的键基本垂直（点积值为零），那么就会使得α_i极大。
  - (3) 此时c基本近似等于v_i
  - (4) 直觉上就是单词的表示越相近，注意力权重就会越高，得到的注意力输出就越接近那个单词。（感觉在把一句废话换着方式说了好几遍）
- (b) 只考虑两个值向量的特殊情况，探究注意力机制的深层含义。
  - (1) 有人可能会觉得如果只是将值向量根据注意力得分取加权和，很难从这个结果中挖掘原先值向量的信息，事实上不然，但是这里做了一个非常强的假定，即两个值向量v_a, v_b是来自相互垂直的向量空间的：

$$
\begin{aligned}
&v_a \in \text{span}\{a_1, a_2, \ldots, a_m\} \Rightarrow v_a = \sum_{i=1}^{m} c_i a_i \\
&v_b \in \text{span}\{b_1, b_2, \ldots, b_p\} \Rightarrow v_b = \sum_{j=1}^{p} d_i b_i \\
&\text{where} \begin{cases} a_i^\top b_j = 0 & \forall i = 1, \ldots, m; \forall j = 1, \ldots, p \\ a_i^\top a_j = 0 & \forall i = 1, \ldots, m \\ b_i^\top b_j = 0 & \forall j = 1, \ldots, p \end{cases}
\end{aligned} \tag{a5.1.1}
$$

根据秩一矩阵的构造方法，假定M具有如下的形式：

$$
M = \sum_{i=1}^{m} \lambda_i a_i a_i^\top \tag{a5.1.2}
$$

其中$\lambda_i, i = 1, \ldots, m$是待定系数，则有如下推导：

$$
\begin{aligned}
Ms = v_a &\iff M(v_a + v_b) = v_a \\
&\iff \left(\sum_{i=1}^{m} \lambda_i a_i a_i^\top \right) \left(\sum_{i=1}^{m} c_i a_i + \sum_{j=1}^{p} d_i b_i \right) = \sum_{i=1}^{m} c_i a_i \\
&\iff \sum_{i=1}^{m} \lambda_i c_i a_i a_i^\top a_i = \sum_{i=1}^{m} c_i a_i \quad \text{(orthogonal property)} \\
&\iff \sum_{i=1}^{m} (\lambda_i c_i a_i^\top a_i) a_i = \sum_{i=1}^{m} c_i a_i \\
&\Longrightarrow \lambda_i c_i a_i^\top a_i = c_i \\
&\Longrightarrow \lambda_i = \frac{1}{a_i^\top a_i} \quad i = 1, \ldots, m
\end{aligned} \tag{a5.1.3}
$$

综上所述：

$$
M = \sum_{i=1}^{m} \frac{a_i a_i^\top}{a_i^\top a_i} \tag{a5.1.4}
$$

  - 本质上就是找一个q使得$k_a^\top q = k_b^\top q$，则可知$q^\top (k_a - k_b) = 0$，找一个与$k_a - k_b$垂直的q就完事了（表达式应该怎么写呢？）。
- (c) 探究单头注意力机制的缺陷：
  - (1) 因为协方差矩阵很小，因此可以近似用μ_i来替换k_i，因此等价于找一个q与$(\mu_a - \mu_b)$垂直即可。

- (2) 容易想到，如果存在一个明显很大的键向量k_a，那么单头注意力机制得到的权重就没有什么意义了，因为加权和之后基本就还是指向k_a的方向。

- (d) 探究多头注意力机制的优势：

  这里的意思是说，给两个查询向量q_1和q_2，分别计算单头注意力得到权重c_1和c_2，然后取$c=(c_1+c_2)/2$作为最终结果即可。

  - (1) 这个就没那么显然了，要求有下式的条件成立：

$$
\begin{aligned}
&\alpha_1^a+\alpha_2^a=\alpha_1^b+\alpha_2^b\\
\Longleftrightarrow&\frac{\exp(k_a^\top q_1)}{\exp(k_a^\top q_1)+\exp(k_b^\top q_1)}+\frac{\exp(k_a^\top q_2)}{\exp(k_a^\top q_2)+\exp(k_b^\top q_2)}=\frac{\exp(k_b^\top q_1)}{\exp(k_a^\top q_1)+\exp(k_b^\top q_1)}+\frac{\exp(k_b^\top q_2)}{\exp(k_a^\top q_2)+\exp(k_b^\top q_2)}\\
\Longleftrightarrow&\frac{\exp(k_a^\top q_1)-\exp(k_b^\top q_1)}{\exp(k_a^\top q_1)+\exp(k_b^\top q_1)}+\frac{\exp(k_a^\top q_2)-\exp(k_b^\top q_2)}{\exp(k_a^\top q_2)+\exp(k_b^\top q_2)}=0\\
\Longleftrightarrow&[\exp(k_a^\top(q_1+q_2))+\exp(k_a^\top q_1+k_b^\top q_2)-\exp(k_b^\top q_1+k_a^\top q_2)-\exp(k_b^\top(q_1+q_2))]\\
&+[\exp(k_a^\top(q_1+q_2))+\exp(k_b^\top q_1+k_a^\top q_2)-\exp(k_a^\top q_1+k_b^\top q_2)-\exp(k_b^\top(q_1+q_2))]=0\\
\Longleftrightarrow&\exp(k_a^\top(q_1+q_2))=\exp(k_b^\top(q_1+q_2))\\
\Longleftrightarrow&k_a^\top(q_1+q_2)=k_b^\top(q_1+q_2)\\
\Longleftrightarrow&(k_a-k_b)^\top(q_1+q_2)=0
\end{aligned}
\tag{a5.1.5}
$$

    刚好消掉了交叉项，那么结论就是找到q_1,q_2使得它们的和与k_a-k_b垂直，这里用μ_a和μ_2近似，就是跟$\mu_a-\mu_b$垂直。

  - (2) 实话说没怎么搞明白是什么意思，虽然增加了协方差，但是$\mu_a-\mu_b$依然可以近似表示k_a-k_b，而且理论上偏差值比没有协方差的情况要小一些（因为协方差都是正数，所以相减相当于抵消了一些偏差）。

    我觉得可能就是想说在多头注意力的情况下，可以缓解$(c.2)$的问题，因为对输出的注意力权重进行了均衡。

2. Pretrained Transformer models and knowledge access

本次代码实验是GPT模型的预训练和微调，GPT模型定义的代码已经完全写好了，要完成的只是数据处理、注意力机制定义、运行与报告部分的代码。

注意代码里有不少读取文件的默认代码可能出错，需要设置文件编码类型。

实话说这个任务有点离谱，居然是根据人名预测出生地，虽说的确不同地区的人名是可以做一些区分，但未免也太牵强了。

本题的代码借鉴自GitHub@Mr-maoge的解法，需要至少8G以上的显存才能跑通，因为缺少计算资源无法跑通代码（经测试，可以调小batch size使得在低显存耗用的情况下通过代码测试，但是无法获得正确的结果）。

虽然代码很难跑通得到结果，但是其中的GPT模型代码以及两种注意力机制的实现代码是值得学习的。

- (a) 阅读`play_char.ipynb`，看代码说明里应该还有`play_math.ipynb`，`play_image.ipynb`，`play_word.ipynb`，有谁知道几个在哪儿可以找到，到时候踢我一下。

- (b) 运行`python src/dataset.py namedata`得到以下输出：

```
data has 418352 characters, 256 unique.
x: Where was Khatchig Mouradian born???Lebanon??
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
y: □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□??Lebanon??
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□
x: Where was Jacob Henry Studer born???Columbus??
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
y: □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□??Columbus??
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
x: Where was John Stephen born???Glasgow??
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□
y: □□□□□□□□□□□□□□□□□□□□□□□□□□□??Glasgow??
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□
x: Where was Georgina Willis born???Australia??
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□
y: □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□??Australia??
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□
```

双问号表示`MASK_CHAR`，正方形表示`PAD_CHAR`。

- (c) 编写`run.py`中相关代码块，注意如果出现`trainer.py`中有`pipeline`的报错信息，将`num_workers`取0来避免。（从这边往下PC机就跑不通了）
- (d) 运行下面的脚本：

```
# Train on the names dataset
python src/run.py finetune vanilla wiki.txt --writing_params_path
vanilla.model.params --finetune_corpus_path birth_places_train.tsv
# Evaluate on the dev set, writing out predictions
python src/run.py evaluate vanilla wiki.txt --reading_params_path
vanilla.model.params --eval_corpus_path birth_dev.tsv --outputs_path
vanilla.nopretrain.dev.predictions
# Evaluate on the test set, writing out predictions
python src/run.py evaluate vanilla wiki.txt --reading_params_path
vanilla.model.params --eval_corpus_path birth_test_inputs.tsv --outputs_path
vanilla.nopretrain.test.predictions
```

- (e) 运行`python src/dataset.py charcorruption`
- (f) 运行下面的脚本：

```
# Pretrain the model
python src/run.py pretrain vanilla wiki.txt --writing_params_path
vanilla.pretrain.params
# Finetune the model
python src/run.py finetune vanilla wiki.txt --reading_params_path
vanilla.pretrain.params --writing_params_path vanilla.finetune.params --
finetune_corpus_path birth_places_train.tsv
# Evaluate on the dev set; write to disk
python src/run.py evaluate vanilla wiki.txt --reading_params_path
vanilla.finetune.params --eval_corpus_path birth_dev.tsv --outputs_path
vanilla.pretrain.dev.predictions
# Evaluate on the test set; write to disk
python src/run.py evaluate vanilla wiki.txt --reading_params_path
vanilla.finetune.params --eval_corpus_path birth_test_inputs.tsv --
outputs_path vanilla.pretrain.test.predictions
```

- (g) 运行下面的脚本：

```
# Pretrain the model
python src/run.py pretrain synthesizer wiki.txt --writing_params_path
synthesizer.pretrain.params

# Finetune the model
python src/run.py finetune synthesizer wiki.txt --reading_params_path
synthesizer.pretrain.params --writing_params_path
synthesizer.finetune.params --finetune_corpus_path birth_places_train.tsv

# Evaluate on the dev set; write to disk
python src/run.py evaluate synthesizer wiki.txt --reading_params_path
synthesizer.finetune.params --eval_corpus_path birth_dev.tsv --outputs_path
synthesizer.pretrain.dev.predictions

# Evaluate on the test set; write to disk
python src/run.py evaluate synthesizer wiki.txt --reading_params_path
synthesizer.finetune.params --eval_corpus_path birth_test_inputs.tsv --
outputs_path synthesizer.pretrain.test.predictions
```

记录一下synthesizer注意力（提出论文）的原理：

- 设$X \in \mathbb{R}^{l\times d}$，其中$l$的块大小（序列长度），$d$是词向量温度，$d/h$是每个注意力头的维度，$Q, K, V \in \mathbb{R}^{d\times d/h}$跟自注意力中的三个矩阵一样，则自注意力头的输出为：

$$Y_i = \text{softmax}\left(\frac{(XQ_i)(XK_i)^\top}{\sqrt{d/h}}\right)(XV_i) \in \mathbb{R}^{l\times d/h} \qquad \text{(a5.2.1)}$$

  接着将各个自注意力头拼接起来：

$$Y = [Y_1; \ldots; Y_h]A \in \mathbb{R}^{l\times d} \qquad \text{(a5.2.2)}$$

- 本题实现的是上面的一个变体：

$$Y_i = \text{softmax}(\text{ReLU}(XA_i + b_1)B_i + b_2)(XV_i) \qquad \text{(a5.2.3)}$$

  其中$A_i \in \mathbb{R}^{d\times d/h}, B \in \mathbb{R}^{d/h\times l}, V_i \in \mathbb{R}^{d\times d/h}$

  可以作这样的解释：

  ① $(XQ_i)(XK_i)^\top \in \mathbb{R}^{l\times l}$是注意力得分；

  ② synthesizer变体则避免计算所有成对的这种点积，而是直接通过将每个自注意力头的d维向量映射到$l \times l$的注意力得分矩阵。

3. Considerations in pretrained knowledge

- (a) 预训练模型结果比非预训练模型结果好不是理所当然的吗，硬要说就是首先找到了一个比较好的初始解开始迭代，因而可以收敛到更好地解。实际情况，不微调只有0.02，微调了之后是0.22
- (b) 人无法辨别出机器到底是检索还是在瞎猜，这可能会使得机器的可解释性下降，无法用于实际应用。测试集中几乎所有人名都没有在训练集中出现过，但是只看姓氏或者名字的话还是有迹可循的，所以机器也并非完全是在瞎猜。
- (c) 模型瞎猜肯定会导致应用的可信度下降呗，不是很能理解这种应用有啥用。