a5

a.i

α 可以被解释为一个类别概率分布，因为它满足类别概率分布的两个基本性质：

1. **非负性**：$\alpha_i = \frac{\exp(k_i^\top q)}{\sum_{j=1}^{n} \exp(k_j^\top q)}$，由于指数函数 $\exp(x)$ 的值始终为正，因此 $\alpha_i \geq 0$ 对所有 i 都成立。
2. **归一性**：$\sum_{i=1}^{n} \alpha_i = \sum_{i=1}^{n} \frac{\exp(k_i^\top q)}{\sum_{j=1}^{n} \exp(k_j^\top q)} = \frac{\sum_{i=1}^{n} \exp(k_i^\top q)}{\sum_{j=1}^{n} \exp(k_j^\top q)} = 1$。这说明 α 的所有元素的总和等于 1。

因此，α 符合概率分布的定义，可以被解释为一个表示每个值向量 v_i 的“选择概率”的类别概率分布。

a.ii

1. **查询向量 q 与某个键向量 k_j 的相似性极高**：
   α_i 的计算公式是

   $$\alpha_i = \frac{\exp(k_i^\top q)}{\sum_{j=1}^{n} \exp(k_j^\top q)}$$

   当 $k_j^\top q \gg k_i^\top q$（对于所有 $i \neq j$）时，$\exp(k_j^\top q)$ 的值将远大于其他项的指数值。结果是 α_j 接近 1，而其他 α_i 接近 0。
2. **键向量彼此差异明显**：
   当所有键向量 $\{k_1, \ldots, k_n\}$ 彼此间差异较大（即它们的方向或大小有明显区分）时，某个键向量 k_j 与查询向量 q 的相似性将显著高于其他键向量。这有助于形成尖锐的类别分布。
3. **查询向量 q 的强选择性**：
   如果 q 的方向与某个键 k_j 的方向几乎完全对齐，而与其他键的方向偏离，则注意力权重会集中在 k_j 上。

a.iii

在上述条件下（$\alpha_j \gg \sum_{i \neq j} \alpha_i$），输出 c 可以描述为：

$$c = \sum_{i=1}^{n} v_i \alpha_i$$

由于 α_j 接近 1，而其他 α_i （对于 $i \neq j$）接近 0，因此 c 将近似等于对应的值向量 v_j：

$$c \approx v_j$$

a.iv

直观上，这意味着当查询向量 q 与某个键向量 k_j 极为相似时，注意力机制会将几乎所有的权重分配给 v_j，从而使输出 c 几乎完全等于 v_j。换句话说，注意力机制在这种情况下表现为一种"选择性复制"行为，仅关注与查询最相关的值向量。

b.i

1. 表达 v_a 和 v_b

- 根据题目假设，v_a 位于子空间 A，由正交基 $\{a_1, a_2, \ldots, a_m\}$ 张成，因此：

$$v_a = c_1 a_1 + c_2 a_2 + \cdots + c_m a_m = \sum_{i=1}^{m} c_i a_i$$

  其中 c_i 是标量系数。

- 同样，v_b 位于子空间 B，由正交基 $\{b_1, b_2, \ldots, b_p\}$ 张成，因此：

$$v_b = d_1 b_1 + d_2 b_2 + \cdots + d_p b_p = \sum_{j=1}^{p} d_j b_j$$

  其中 d_j 是标量系数。

- 由于 A 和 B 是正交的，基向量满足 $a_i^\top b_j = 0$ 对所有 i, j 成立。

2. 表达 $s = v_a + v_b$

- 将 v_a 和 v_b 加起来得到 s：

$$s = v_a + v_b = \sum_{i=1}^{m} c_i a_i + \sum_{j=1}^{p} d_j b_j$$

3. 构造矩阵 M

- 我们希望矩阵 M 提取 v_a，即 $Ms = v_a$。由于 v_a 完全位于子空间 A，我们可以设计矩阵 M 来保留 A 的部分，屏蔽 B 的部分。
- 构造 M 为以下形式：

$$M = \sum_{i=1}^{m} a_i a_i^\top$$

这里，$a_i a_i^\top$ 是一个对称矩阵，用于将输入向量投影到基向量 a_i 的方向。

4. 验证 $Ms = v_a$

- 计算 Ms：

$$Ms = M(v_a + v_b) = Mv_a + Mv_b$$

- 首先计算 Mv_a：

$$Mv_a = \left(\sum_{i=1}^{m} a_i a_i^\top \right) \left(\sum_{k=1}^{m} c_k a_k \right)$$

展开后：

$$Mv_a = \sum_{i=1}^{m} \sum_{k=1}^{m} c_k (a_i a_i^\top a_k)$$

由于 $\{a_i\}$ 是正交基，$a_i^\top a_k = 0$（当 $i \neq k$）且 $a_i^\top a_i = 1$，因此只有当 $i = k$ 时有非零贡献：

$$Mv_a = \sum_{i=1}^{m} c_i a_i = v_a$$

- 再计算 Mv_b：

$$Mv_b = \left(\sum_{i=1}^{m} a_i a_i^\top \right) \left(\sum_{j=1}^{p} d_j b_j \right)$$

展开后：

$$Mv_b = \sum_{i=1}^{m} \sum_{j=1}^{p} d_j (a_i a_i^\top b_j)$$

由于 $a_i^\top b_j = 0$ 对所有 i, j 成立，因此：

$$Mv_b = 0$$

- 因此，$Ms = Mv_a + Mv_b = v_a + 0 = v_a$。

5. 结论

我们构造的矩阵 $M = \sum_{i=1}^{m} a_i a_i^\top$ 成功提取了子空间 A 中的向量 v_a，并屏蔽了子空间 B 中的向量 v_b。因此，$Ms = v_a$ 对任意 $v_a \in A$ 和 $v_b \in B$ 成立。

补充：

$$a_i^\top a_i = \|a_i\|^2 = 1$$

1. **正交性**：
   - 正交基的每对不同向量 a_i, a_j 满足 $a_i^\top a_j = 0$（当 $i \neq j$ 时）。
   - 这意味着基向量之间是相互正交的。
2. **单位范数**：
   - 每个基向量 a_i 的范数为 1，即 $\|a_i\| = 1$。
   - 向量的范数的平方等于它自身的点积：$\|a_i\|^2 = a_i^\top a_i$。

因此，对于每个 a_i：

$$a_i^\top a_i = \|a_i\|^2 = 1$$

b.ii

1. 注意力的公式回顾

注意力的输出 c 的定义为：

$$c = \sum_{i=1}^{n} v_i \alpha_i$$

其中：

$$\alpha_i = \frac{\exp(k_i^\top q)}{\sum_{j=1}^{n} \exp(k_j^\top q)}$$

α_i 是键向量 k_i 和查询向量 q 的相似性所决定的注意力权重。

2. 设定目标

我们希望 $c \approx \frac{1}{2}(v_a + v_b)$。这意味着：

1. 注意力权重需要满足：$\alpha_a \approx \alpha_b \approx \frac{1}{2}$，且 $\alpha_i = 0$ 对于其他 i。
2. 这要求查询向量 q 必须与 k_a 和 k_b 的相似度相等（或非常接近），同时对其他键向量的相似度为零。

3. 利用正交性假设

根据题目假设：

1. 所有键向量是正交的，即 $k_i^\top k_j = 0$ 对于 $i \neq j$；
2. 所有键向量的范数为 1，即 $\|k_i\| = 1$。

正交性意味着 q 的相似度（即 $k_i^\top q$）可以单独控制，不受其他键向量的干扰。

4. 构造查询向量 q

为了满足 $\alpha_a \approx \alpha_b \approx \frac{1}{2}$，我们需要：

$$k_a^\top q = k_b^\top q$$

并且这些相似度需要显著大于其他键向量的相似度（如 $k_i^\top q = 0$ 对于 $i \notin \{a, b\}$）。

一种满足这些条件的 q 是：

$$q = \beta(k_a + k_b)$$

因为：

1. $k_a^\top q = \beta k_a^\top k_a + \beta k_a^\top k_b = \beta + 0 = \beta$；
2. $k_b^\top q = \beta k_b^\top k_a + \beta k_b^\top k_b = 0 + \beta = \beta$；
3. 对于其他键向量 k_i $(i \notin a, b)$，由于正交性：

   $k_i^\top q = \beta k_i^\top k_a + \beta k_i^\top k_b = 0 + 0 = 0$

因此，$q = \beta(k_a + k_b)$ 使 $k_a^\top q = k_b^\top q$，同时对其他键向量的相似度为零。

5. 验证注意力权重

计算注意力权重 α_a 和 α_b：

$\alpha_a = \frac{\exp(k_a^\top q)}{\exp(k_a^\top q)+\exp(k_b^\top q)+\ldots+\exp(k_j^\top q)} = \frac{\exp(\beta)}{\exp(\beta)+\exp(\beta)+\ldots+1}$

若要$\alpha_a \approx \frac{1}{2}$，则$\beta \gg 0$

对于α_b同理

6. 结论

通过$q = \beta(k_a + k_b), \beta \gg 0$，则可以得到$c \approx v_a\alpha_a + v_b\alpha_b \approx \frac{1}{2}(v_a + v_b)$

c.i

1. 问题回顾

注意力机制输出的向量 c 的公式为：

$$c = \sum_{i=1}^{n} \alpha_i v_i$$

其中：

$$\alpha_i = \frac{\exp(k_i^\top q)}{\sum_{j=1}^{n} \exp(k_j^\top q)}$$

α_i 是注意力权重，取决于键向量 k_i 与查询向量 q 的点积 $k_i^\top q$。

目标是设计一个 q，使得注意力权重满足：

$$\alpha_a \approx \alpha_b \approx \frac{1}{2}, \quad \alpha_i \approx 0 \quad (\forall i \notin \{a, b\}),$$

从而得到：

$$c \approx \frac{1}{2}(v_a + v_b).$$

2. 约束条件分析

1. **协方差矩阵假设**：
   - 键向量 k_i 从分布 $k_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ 中采样，其中 $\Sigma_i = \alpha I$，且 α 极小。
   - 这意味着 k_i 将非常接近其均值 μ_i，即：

$$k_i \approx \mu_i.$$

2. **均值正交性**：
   - 均值向量 μ_i 是正交的，满足 $\mu_i^\top \mu_j = 0\,(i \neq j)$，且 $\|\mu_i\| = 1$。
   - 正交性允许我们独立地调整 q 对 μ_a 和 μ_b 的相似性，而不影响其他 μ_i。

3. 构造查询向量 q

为了让 $\alpha_a \approx \alpha_b \approx \frac{1}{2}$，我们需要确保：

$$\mu_a^\top q = \mu_b^\top q, \quad \mu_i^\top q \ll \mu_a^\top q \quad (\forall i \notin \{a, b\}).$$

一种简单的选择是将 q 构造为：

$$q = \beta(\mu_a + \mu_b)$$

4.验证

- **与 μ_a和 μ_b 的相似性：**

$$\mu_a^\top q = \beta \mu_a^\top (\mu_a + \mu_b) = \beta \mu_a^\top \mu_a + \beta \mu_a^\top \mu_b = \beta + 0 = \beta.$$

同理：

$$\mu_b^\top q = \beta \mu_b^\top (\mu_a + \mu_b) = \beta \mu_b^\top \mu_a + \beta \mu_b^\top \mu_b = 0 + \beta = \beta.$$

- **与其他μ_i的相似性**：
  对于
  $i \notin \{a, b\}$：$\mu_i^\top q = \beta \mu_i^\top (\mu_a + \mu_b) = \beta \mu_i^\top \mu_a + \beta \mu_i^\top \mu_b = 0 + 0 = 0.$

注意力权重计算：

- 对于 i=a 或 i=b：

  $$\exp(\mu_a^\top q) = \exp(\beta), \quad \exp(\mu_b^\top q) = \exp(\beta).$$

- 对于$i \notin \{a, b\}$：

  $$\exp(\mu_i^\top q) = \exp(0) = 1.$$

$\alpha_a = \frac{\exp(\mu_a^\top q)}{\exp(\mu_a^\top q) + \exp(\mu_b^\top q) + \ldots + \exp(\mu_j^\top q)} = \frac{\exp(\beta)}{\exp(\beta) + \exp(\beta) + \ldots + 1}$

若要$\alpha_a \approx \frac{1}{2}$，则$\beta \gg 0$

对于α_b同理

6. 结论

通过$q = \beta(\mu_a + \mu_b), \beta \gg 0$，则可以得到$c \approx v_a \alpha_a + v_b \alpha_b \approx \frac{1}{2}(v_a + v_b)$

c.ii

1. 问题回顾

- 键向量 k_i 从分布 $\mathcal{N}(\mu_i, \Sigma_i)$ 中采样。
- k_a 的协方差矩阵为 $\Sigma_a = \alpha I + \frac{1}{2}(\mu_a \mu_a^\top)$，因此 k_a 在方向 μ_a 上可能有较大的方差，其大小可能比其他键向量更大或更小。
- 其他键向量 k_i 的协方差矩阵为 $\Sigma_i = \alpha I \ (i \neq a)$，因此它们紧密分布在各自的均值附近，方差较小。

查询向量在第 i 部分定义为：

$$q = \beta(\mu_a + \mu_b), \quad \beta \gg 0,$$

输出结果为：

$$c = \alpha_a v_a + \alpha_b v_b + \sum_{i \neq a,b} \alpha_i v_i.$$

2. 多次采样对键向量的影响

2.1 k_a 的变化

- 由于 $\Sigma_a = \alpha I + \frac{1}{2}(\mu_a \mu_a^\top)$，$k_a$ 的可能值范围不仅与 μ_a 的方向一致，还可能因方差增加而在大小上发生显著变化。
- $k_a^\top q$ 的点积因此受到两方面的影响：
  1. k_a 的方向接近 μ_a，因此点积主要由 $\mu_a^\top q$ 决定；
  2. 由于 k_a 的大小变化，点积的结果可能波动较大。

2.2 其他键向量的变化

- 对于其他键向量 k_i （$i \neq a, b$），因为它们的协方差矩阵是 $\Sigma_i = \alpha I$，方差较小，因此这些键向量紧密分布在各自的均值 μ_i 附近，大小和方向变化都较小。

2.3 k_b 的变化

- k_b 的分布与 $\Sigma_b = \alpha I$ 一致，因此大小和方向变化幅度都较小，与第 i 部分假设一致。

3. 多次采样下 c 的行为

3.1 注意力权重 α_i 的变化

注意力权重由公式决定：

$$\alpha_i = \frac{\exp(k_i^\top q)}{\sum_{j=1}^{n} \exp(k_j^\top q)}.$$

1. **对于 α_a 和 α_b：**
   - $k_a^\top q \approx \mu_a^\top q + 波动项$，波动项来源于 k_a 的大小变化（由协方差 Σ_a 导致）。
   - $k_b^\top q$ 的变化较小，因为 k_b 的大小和方向稳定。
   - 当 k_a 的大小显著偏大或偏小时，α_a 和 α_b 会发生较大偏移。
2. **对于 α_i $(i \neq a, b)$ ：**
   - 因为 $k_i^\top q$ 主要取决于小方差扰动，α_i 的值基本保持较小，且不显著影响输出 c。

3.2 输出向量 c 的变化

由于 α_a 和 α_b 的波动，c 的值会发生变化：

$$c = \alpha_a v_a + \alpha_b v_b + \sum_{i \neq a,b} \alpha_i v_i.$$

- 理想情况下，第 i 部分的结果为：
  $$c \approx \frac{1}{2}(v_a + v_b),$$
  但由于 k_a 的大小波动，α_a 和 α_b 的分布可能偏离均匀分布（$\frac{1}{2}$）。
- 如果 k_a 的大小比预期更大，则 $\alpha_a > \frac{1}{2}$，导致 c 偏向 v_a；
- 如果 k_a 的大小比预期更小，则 $\alpha_a < \frac{1}{2}$，导致 c 偏向 v_b。

4. 与第 i 部分的结果对比

在第 i 部分，假设所有键向量的协方差矩阵为 αI，波动很小，因此注意力权重分布稳定，输出 $c \approx \frac{1}{2}(v_a + v_b)$。

在高方差条件下：

1. 由于 k_a 的大小变化导致注意力权重 α_a 和 α_b 波动；
2. 输出 c 的值偏离理想的均值结果，可能更接近 v_a 或 v_b，具体取决于采样。

5. c 的方差分析

输出 c 的方差主要来源于 k_a 的波动：

1. 当 Σ_a 的方差分量较大时，k_a 的大小波动剧烈，导致 α_a 和 α_b 偏离理想值；
2. c 的方差会增加，因为 $\alpha_a v_a$ 和 $\alpha_b v_b$ 的波动性显著增强。

6. 结论

在高方差条件下，输出 c 的行为随采样而显著波动，偏离理想值 $\frac{1}{2}(v_a + v_b)$。其方差增加，主要由 k_a 的大小波动引起。相比于第 i 部分的结果，这种波动破坏了注意力权重的稳定性，导致输出结果变得不可靠。

d.i

在之前相同的假设下，我们可以设计 q_1 和 q_2，使得其中一个复制 v_a，另一个复制 v_b。由于所有键向量都与其均值非常接近，并且均值是正交的，按照问题 (a) 部分的解释，我们将查询向量表达为：

$$q_1 = \beta\mu_a, \quad q_2 = \beta\mu_b, \quad \text{其中 } \beta \gg 0。$$

这将得到以下结果（因为均值是正交的）：

$$c_1 \approx v_a, \quad c_2 \approx v_b。$$

由于多头注意力的输出只是两个值的平均值，因此可以看到：

$$c \approx \frac{1}{2}(v_a + v_b)。$$

d.ii

关于问题 (c) 部分 ii. 的内容，如果我们选择 $q_1 = \beta\mu_a$ 和 $q_2 = \beta\mu_b$，则可以得到以下结果（注意，所有其他键-查询点积将变得微不足道）：

$$k_a^\top q_1 = \gamma\mu_a^\top \beta\mu_a = \gamma\beta, \quad \text{其中 } \beta \gg 0;$$
$$k_b^\top q_2 = \mu_b^\top \beta\mu_b = \beta, \quad \text{其中 } \beta \gg 0。$$

我们可以求出 α 的值（再次注意，当 β 较大时，所有其他键-查询点积将变得不显著）：

$$\alpha_{a1} \approx \frac{\exp(\gamma\beta)}{\exp(\gamma\beta)} \approx 1; \quad \alpha_{b2} \approx \frac{\exp(\beta)}{\exp(\beta)} \approx 1。$$

由于我们可以说对于任何$i \neq a, \alpha_{i1} \approx 0$ 并且 $i \neq b, \alpha_{i2} \approx 0$ 并且对于任何，因此很容易看出：

$$c_1 \approx v_a, \quad c_2 \approx v_b。$$

这意味着最终输出将始终近似于值的平均值：

$$c \approx \frac{1}{2}(v_a + v_b)。$$

对于不同的键向量采样，输出 c 的均值保持稳定，即虽然单个头采样键向量出现较大波动，但是影响仅限于单个头，并且对单个头最后的输出结果影响较小，从而对整个多头注意力影响更小，多头注意力的结果变得更加健壮

代码部分

a.

训练过程

```
[6]: # you can download this file at https://github.com/karpathy/char-rnn/blob/master/data/tinyshakespeare/input.txt
text = open('input.txt', 'r').read() # don't worry we won't run out of file handles
train_dataset = CharDataset(text, block_size) # one line of poem is roughly 50 characters

data has 1115394 characters, 65 unique.

[7]: from mingpt.model import GPT, GPTConfig
mconf = GPTConfig(train_dataset.vocab_size, train_dataset.block_size,
                  n_layer=8, n_head=8, n_embd=512)
model = GPT(mconf)

11/18/2024 21:06:12 - INFO - mingpt.model -   number of parameters: 2.535219e+07

[8]: from mingpt.trainer import Trainer, TrainerConfig

# initialize a trainer instance and kick off training
tconf = TrainerConfig(max_epochs=2, batch_size=512, learning_rate=6e-4,
                      lr_decay=True, warmup_tokens=512*20, final_tokens=2*len(train_dataset)*block_size,
                      num_workers=4)
trainer = Trainer(model, train_dataset, None, tconf)
trainer.train()

epoch 1 iter 2178: train loss 0.34671. lr 3.000169e-04: 100%|██████████| 2179/2179 [15:14<00:00,  2.38it/s]
epoch 2 iter 2178: train loss 0.18294. lr 6.000000e-05: 100%|██████████| 2179/2179 [15:15<00:00,  2.38it/s]
```

训练结果

```python
[9]: # alright, let's sample some character-level Shakespeare
from mingpt.utils import sample

context = "O God, O God!"
x = torch.tensor([train_dataset.stoi[s] for s in context], dtype=torch.long)[None,...].to(trainer.device)
y = sample(model, x, 2000, temperature=1.0, sample=True, top_k=10)[0]
completion = ''.join([train_dataset.itos[int(i)] for i in y])
print(completion)
```

```
O God, O God! that e'er this tongue of mine,
That laid the sentence of dread banishment
On yon proud man, should take it off again
With words of sooth! O that I were as great
As is my grief, or lesser than my name!
Or that I could forget what I have been,
Or not remember what  I must be now!
Swell'st thou, proud heart? I'll give thee scope to beat,
Since foes have scope to beat both thee and me.

DUKE OF AUMERLE:
Northumberland comes back from Bolingbroke.

KING RICHARD II:
What must the king do now? must he submit?
The king shall do it: must he be deposed?
The king shall be contented: must he lose
The name of king? o' God's name, let it go:
I'll give my jewels for a set of beads,
My gorgeous palace for a hermitage,
My gay apparel for an almsman's gown,
My figured goblets for a dish of wood,
My sceptre for a palmer's walking staff,
My subjects for a pair of carved saints
And my large kingdom for a little grave,
A little little grave, an obscure grave;
Or I'll be buried in the king's highway,
Some way of common trade, where subjects' feet
May hourly trample on their sovereign's head;
For on my heart they tread now whilst I live;
And buried once, why not upon my head?
Aumerle, thou weep'st, my tender-hearted cousin!
We'll make foul weather with despised tears;
Our sighs and they shall lodge the summer corn,
And make a dearth in this revolting land.
Or shall we play the wantons with our woes,
And make some pretty match with shedding tears?
As thus, to drop them still upon one place,
Till they have fretted us a pair of graves
Within the earth; and, therein laid,--there lies
Two kinsmen digg'd their graves with weeping eyes.
Would not this ill do well? Well, well, I see
I talk but idly, and you laugh at me.
Most mighty prince, my Lord Northumberland,
What says King Bolingbroke? will his majesty
Give Richard leave to live till Richard die?
You make a leg, and Bolingbroke says ay.

NORTHUMBERLAND:
My lord, in the base court he doth attend
To slugge a this shame, not by the disguised,
B
```

```python
[ ]: # well that was fun
```

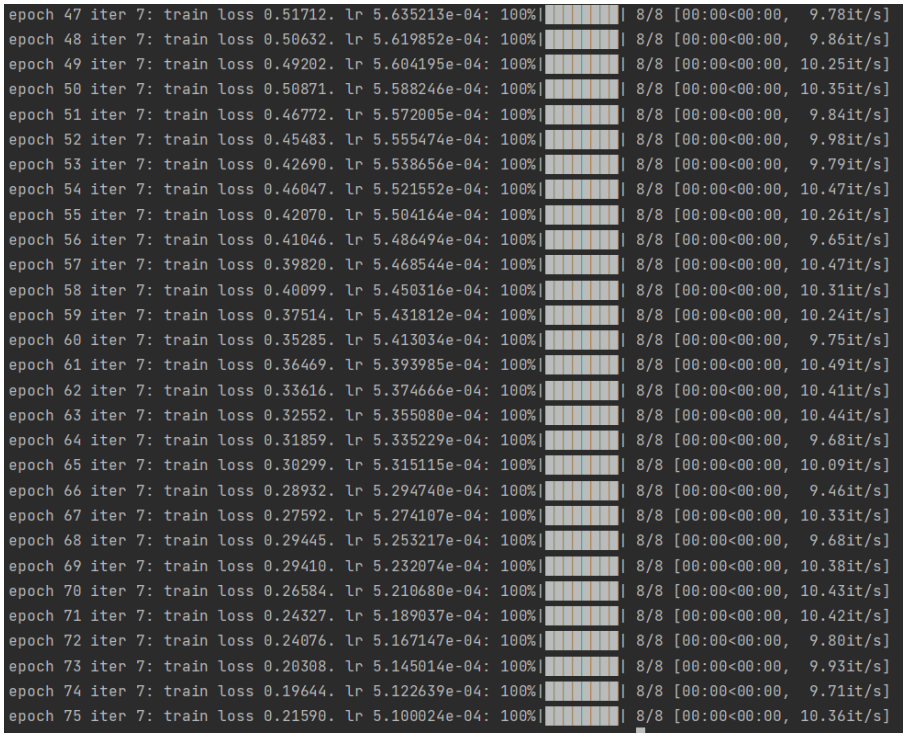
b.

打印数据

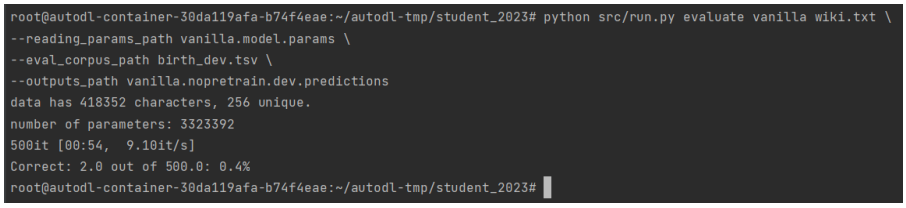
```
root@autodl-container-6c3e4aa234-d86e7afd:~/autodl-tmp/student_2023# python src/dataset.py namedata
data has 418352 characters, 256 unique.
x: Where was Khatchig Mouradian born??Lebanon??□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
y: □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□??Lebanon??□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
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??□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

d.

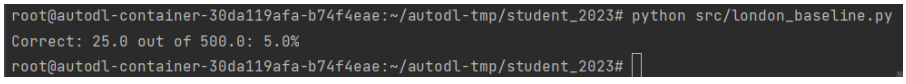
训练过程



开发集上结果：准确率0.4%



全猜London结果：准确率5%



e.

文本掩盖打印结果：

```
root@autodl-container-7145478fa9-2858b787:~/autodl-tmp/student_2023# python src/dataset.py charcorruption
data has 418352 characters, 256 unique.
x: K⁇tchig Mouradian. Khatchig Mourad⁇ha□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
y: ⁇tchig Mouradian. Khatchig Mourad⁇ha□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
x: Ja⁇Studer. Jacob Henry Studer (26 February 1840 Col⁇cob Henry □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
y: a⁇Studer. Jacob Henry Studer (26 February 1840 Col⁇cob Henry □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
x: John Stephen. ⁇, Steph⁇Born in Glasgow□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
y: ohn Stephen. ⁇, Steph⁇Born in Glasgow□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
x: Georgina Willis. Georgina W⁇m director who was born in Aus⁇illis is an award winning fil□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
y: eorgina Willis. Georgina W⁇m director who was born in Aus⁇illis is an award winning fil□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
root@autodl-container-7145478fa9-2858b787:~/autodl-tmp/student_2023#
```

f.

预训练模型结果：

```
epoch 634 iter 22: train loss 0.50648. lr 6.000000e-04: 100%|██████████| 23/23 [00:01<00:00, 20.49it/s]
epoch 635 iter 22: train loss 0.51053. lr 6.000000e-04: 100%|██████████| 23/23 [00:01<00:00, 19.74it/s]
epoch 636 iter 22: train loss 0.49269. lr 6.000000e-04: 100%|██████████| 23/23 [00:01<00:00, 19.90it/s]
epoch 637 iter 22: train loss 0.49810. lr 6.000000e-04: 100%|██████████| 23/23 [00:01<00:00, 19.65it/s]
epoch 638 iter 22: train loss 0.51153. lr 6.000000e-04: 100%|██████████| 23/23 [00:01<00:00, 20.44it/s]
epoch 639 iter 22: train loss 0.49012. lr 6.000000e-04: 100%|██████████| 23/23 [00:01<00:00, 20.58it/s]
epoch 640 iter 22: train loss 0.52894. lr 6.000000e-04: 100%|██████████| 23/23 [00:01<00:00, 19.73it/s]
epoch 641 iter 22: train loss 0.53671. lr 6.000000e-04: 100%|██████████| 23/23 [00:01<00:00, 20.54it/s]
epoch 642 iter 22: train loss 0.52466. lr 6.000000e-04: 100%|██████████| 23/23 [00:01<00:00, 20.55it/s]
epoch 643 iter 22: train loss 0.50676. lr 6.000000e-04: 100%|██████████| 23/23 [00:01<00:00, 19.78it/s]
epoch 644 iter 22: train loss 0.51826. lr 6.000000e-04: 100%|██████████| 23/23 [00:01<00:00, 20.32it/s]
epoch 645 iter 22: train loss 0.51585. lr 6.000000e-04: 100%|██████████| 23/23 [00:01<00:00, 19.97it/s]
epoch 646 iter 22: train loss 0.52380. lr 6.013186e-04: 100%|██████████| 23/23 [00:01<00:00, 19.98it/s]
epoch 647 iter 22: train loss 0.51551. lr 6.296935e-04: 100%|██████████| 23/23 [00:01<00:00, 20.44it/s]
epoch 648 iter 22: train loss 0.50998. lr 6.586443e-04: 100%|██████████| 23/23 [00:01<00:00, 19.73it/s]
epoch 649 iter 22: train loss 0.51337. lr 6.881640e-04: 100%|██████████| 23/23 [00:01<00:00, 19.96it/s]
```

微调模型结果：

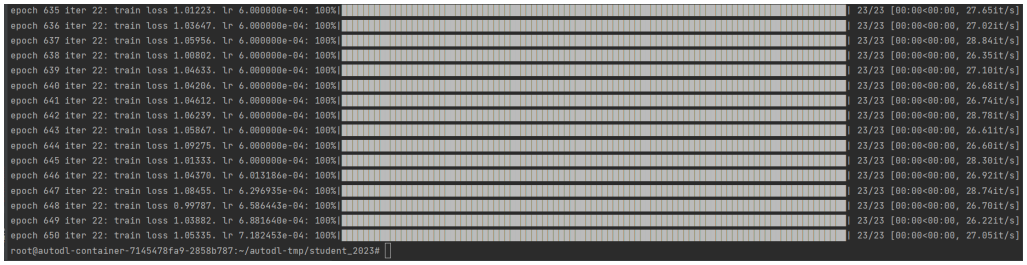
```
data has 418352 characters, 256 unique.
number of parameters: 3323392
epoch 1 iter 7: train loss 0.74723. lr 5.999844e-04: 100%|██████████| 8/8 [00:01<00:00,  7.07it/s]
epoch 2 iter 7: train loss 0.61550. lr 5.999351e-04: 100%|██████████| 8/8 [00:00<00:00, 11.51it/s]
epoch 3 iter 7: train loss 0.53251. lr 5.998521e-04: 100%|██████████| 8/8 [00:00<00:00, 10.31it/s]
epoch 4 iter 7: train loss 0.46716. lr 5.997352e-04: 100%|██████████| 8/8 [00:00<00:00, 10.93it/s]
epoch 5 iter 7: train loss 0.40949. lr 5.995847e-04: 100%|██████████| 8/8 [00:00<00:00, 11.50it/s]
epoch 6 iter 7: train loss 0.36283. lr 5.994004e-04: 100%|██████████| 8/8 [00:00<00:00, 10.55it/s]
epoch 7 iter 7: train loss 0.31523. lr 5.991823e-04: 100%|██████████| 8/8 [00:00<00:00, 10.79it/s]
epoch 8 iter 7: train loss 0.28917. lr 5.989306e-04: 100%|██████████| 8/8 [00:00<00:00, 10.38it/s]
epoch 9 iter 7: train loss 0.24821. lr 5.986453e-04: 100%|██████████| 8/8 [00:00<00:00, 10.65it/s]
epoch 10 iter 7: train loss 0.21684. lr 5.983263e-04: 100%|██████████| 8/8 [00:00<00:00, 11.32it/s]
```

开发集上的准确率：11.60%

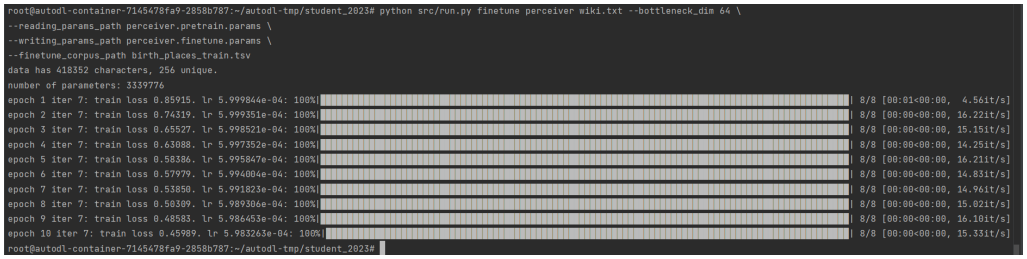
```
root@autodl-container-7145478fa9-2858b787:~/autodl-tmp/student_2023# python src/run.py evaluate vanilla wiki.txt \
--reading_params_path vanilla.finetune.params \
--eval_corpus_path birth_dev.tsv \
--outputs_path vanilla.pretrain.dev.predictions
data has 418352 characters, 256 unique.
number of parameters: 3323392
500it [01:01,  8.08it/s]
Correct: 58.0 out of 500.0: 11.600000000000001%
root@autodl-container-7145478fa9-2858b787:~/autodl-tmp/student_2023#
```

g.

预训练模型结果：



微调模型结果：



开发集上的准确率：6.20%

ii.

1. Vanilla 模型的时间复杂度

Vanilla Transformer 模型的核心计算是 **自注意力机制**，其复杂度主要由以下部分决定：

计算 Query, Key, Value 矩阵：

- 对于输入序列 $X \in \mathbb{R}^{\ell \times d}$，计算 Q, K, V 的复杂度为：

$$\mathcal{O}(\ell \cdot d^2)$$

(这里 d 是嵌入维度）。

计算注意力分数矩阵：

- 分数矩阵大小为 $\ell \times \ell$，计算复杂度为：

$$\mathcal{O}(\ell^2 \cdot d)$$

每个层的复杂度：

- 每一层 Transformer 的总复杂度为：

$$\mathcal{O}(\ell \cdot d^2 + \ell^2 \cdot d)$$

整个模型（L 层）的复杂度：

- 假设总层数为 L，则整体复杂度为：

$$\mathcal{O}(L \cdot (\ell \cdot d^2 + \ell^2 \cdot d))$$

2. Perceiver 模型的时间复杂度

Perceiver 模型引入了 **降维（Down Projection）** 和 **恢复维度（Up Projection）** 操作，这些操作改变了计算注意力的复杂度。

2.1 降维模块（Down Projection Block）

1. **输入序列长度从 ℓ 降到瓶颈维度 m**：
    - 在降维过程中，`DownProjectBlock` 执行交叉注意力：

- 对于输入序列 $X \in \mathbb{R}^{\ell \times d}$和 $C \in \mathbb{R}^{m \times d}$

  计算 Q 的复杂度为：

  $\mathcal{O}(m \cdot d^2)$

  计算 K,V 的复杂度为：

$$\mathcal{O}(\ell \cdot d^2)$$

2.2中间的 Transformer 块

在降维之后，序列长度为 m。这些块执行常规的自注意力操作，其复杂度为：

$$\mathcal{O}(m \cdot d^2 + m^2 \cdot d)$$

2.3 恢复维度模块（Up Projection Block）

1. **从瓶颈维度 m 恢复到原始长度 ℓ**：

- 对于输入序列 $X \in \mathbb{R}^{\ell \times d}$和 $Y \in \mathbb{R}^{m \times d}$

  计算 Q 的复杂度为：

  $\mathcal{O}(\ell \cdot d^2)$
  计算 K,V 的复杂度为：

  $\mathcal{O}(m \cdot d^2)$

总复杂度为：

$$\mathcal{O}(\ell \cdot d^2) + \mathcal{O}(m \cdot d^2) + (L-2) \cdot O(m \cdot d^2 + m^2 \cdot d) + \mathcal{O}(\ell \cdot d^2) + \mathcal{O}(m \cdot d^2)$$

3

a.

1. 预训练模型的优势：知识迁移

预训练 模型通过在大规模语料库（如 Wikipedia）上进行训练，学到了丰富的世界知识。这包括：

- 常见的人名与地名的关联。
- 语义上下文的基本理解能力。
- 自然语言中的统计模式（如哪些名字更可能与哪些地点相关联）。

因此，在微调阶段，预训练模型已经拥有了丰富的先验知识，可以更高效地学习预测任务。

2. 非预训练模型的劣势：从零开始学习

非预训练 模型从零开始训练，仅依赖于微调数据（即训练集）。由于缺乏对人名和地名之间关系的任何先验知识，它需要：

- 从有限的训练数据中学习。
- 推断所有模式（如名字与地点的关联），这在数据稀疏时极为困难。

微调数据通常较小，无法为模型提供足够的信息来建立这些复杂的关联关系。

3. 数据稀疏性的影响

- **预训练模型**：即使微调数据中某些名字或地点很少出现，预训练模型可能已经在预训练语料库中见过这些名字或地点，因此可以借助这些知识进行准确预测。
- **非预训练模型**：由于完全依赖于微调数据，如果某些名字或地点在训练集中未出现或出现频率很低，模型很难准确预测。

b.

1. 系统应用的潜在担忧

原因 1：错误信息的传播

当模型捏造了错误的预测时，用户界面系统可能将错误的信息展示给用户，这会导致错误信息传播。

- **例子**：
    - 假设用户询问 "Where was Albert Jones born?"（一个冷门历史人物）。
    - 模型捏造了答案 "Paris"，但实际出生地是 "Berlin"。
    - 这种错误可能会误导用户，特别是在需要高准确率的领域（如历史研究或教育）。

原因 2：用户对模型可信度的怀疑

用户无法判断模型的预测是否是基于可靠的知识还是完全捏造的，这会削弱用户对系统的信任。

- **例子**：

- 一个知识问答系统回答了 "Where was Jane Doe born?" 为 "London"。
- 如果用户发现其他权威来源并未提到相关信息，可能会质疑系统的可信性，甚至认为所有答案都不可靠。

2. 为什么这种行为对系统应用构成挑战

原因 1 的扩展：错误传播的风险

- 这对医疗、法律等高风险场景尤为关键。例如，错误地预测某人出生地可能影响移民申请或法律案件的处理。

原因 2 的扩展：透明性问题

- 用户需要了解系统为何做出某个预测，而当前 NLP 系统缺乏解释能力。缺乏透明性会让用户对系统的整体表现产生怀疑。

c.

1. 模型可能的策略

1.1 数据分布驱动的猜测

- 模型可能会简单地输出训练数据集中最频繁的出生地。
- **例子**：如果训练集中 "London" 出现最多，则对于未见名字，模型直接猜测 "London"。

1.2 基于字符特征的推断

- 模型可能分析名字的字符模式和语言特征，尝试将其映射到一个它认为合理的地理区域。
- **例子**：名字 "Maria" 可能被映射到 "Spain" 或 "Italy"，因为这些地区的人名中更常见此模式。

1.3 随机生成

- 在没有明确相关性的情况下，模型可能会随机输出一个地名作为出生地预测。

2. 为什么这种行为可能引发担忧？

担忧 (除b部分展示的以外)：隐性偏见

- 模型可能基于训练数据中的不平衡分布，输出具有隐性偏见的预测。
- **例子**：
  - 如果训练集中大部分数据来自英语语料，模型可能对非英语名字做出错误的猜测，忽略其他文化背景。
  - 假设用户查询 "Where was Wei Zhang born?"，模型可能错误地输出 "New York"，因为它对非英语地区的代表性不足。