实验报告

4

4.1 A

3. 损失曲线下降，说明模型预测越来越接近真实标签，训练有效。损失曲线与理论公式一一对应，数值即为交叉熵。
4. 你的 `train()` 已经正确使用了 `torch.nn.utils.clip_grad_norm_` 实现梯度裁剪：

```
loss.backward()
nn.utils.clip_grad_norm_(model.parameters(), clip)
optim.step()
```

其中 `clip` 就是裁剪阈值（如 5.0），用于抑制梯度爆炸。
只需确保 `clip` 参数合理（如 1.0～5.0），即可防止梯度过大导致训练不稳定。

简要说明：

- `clip_grad_norm_` 会将所有参数的梯度范数裁剪到不超过 `clip`，防止梯度爆炸。
- 你可以通过命令行参数 `--clip 5.0` 控制阈值。

无需额外改写，当前实现已满足实验要求。
5. 不同温度（temperature）采样时，softmax logits 会被 T 缩放，影响生成文本的多样性和确定性：

- T < 1（如 0.5）：分布更尖锐，概率最大的字符更容易被选中，生成文本趋于重复、保守，创造性低。
- T = 1：标准 softmax，平衡多样性与合理性。
- T > 1（如 1.3）：分布更平坦，低概率字符被采样的机会增加，文本更有创造性但也更容易出现语法混乱或无意义内容。
  这与 §2.4 的理论一致：温度调节了输出分布的“熵”，影响生成文本的多样性和可控性。实际运行时你会发现，低温度输出重复性高，高温度输出更丰富但偶有乱码。

4.2 B

1. 只用最后的 h_T，意味着每个序列只在最后一步预测下一个字符，训练信号只来自每个序列的最后一个字符，导致每个样本的有效监督较少，训练效率较低。
   如果改为“全时间步预测”，即每一步 h_t 都预测 x_{t+1}，则每个序列能产生 T 个监督信号，显著提高训练信号密度，加快收敛速度。这也是现代语言模型常用的做法。
   换言之：
   只用 h_T，训练信号稀疏，效率低。
   全时间步预测，训练信号密集，模型学得更快更好。