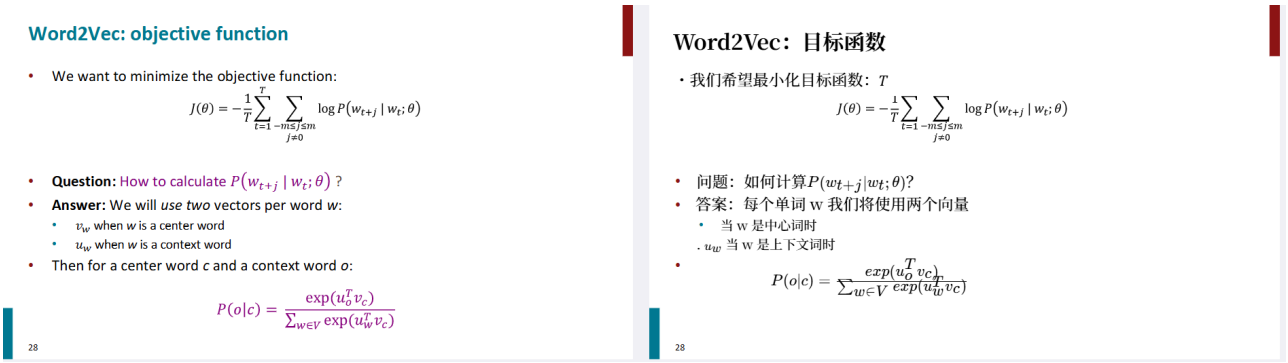
第一讲：简介与词向量

word2vec

- 一个用于学习词向量的框架
- 

- 一个单词两个向量：作为中心词、上下文词

第二讲：词向量、词义和神经网络分类器

- 朴素softmax计算量大，因此使用带负采样的SG模型
- 基于窗口的共现矩阵，维度大，因此降维，还可以缩放
- 词向量评估：内在、外在
- 一个词含义很多：线性叠加
- 命名实体识别（NER）
- 交叉熵损失

作业1

- **基于计数的词向量：共现矩阵**

展平：嵌套的列表推导式

```
all_words = [word for document in corpus for word in document]
```

去重：用set元素唯一的特性

共现矩阵编写：提取文档单词和词数，从而初始化共现矩阵，for循环构建

降维：sklearn.decomposition.TruncatedSVD（可截断SVD）

绘制图形：查看the Matplotlib gallery，找到与你想要的图形相似的示例，然后改编它们提供的代码

共现矩阵可视化：需要对返回的向量进行归一化，使所有向量分布在单位圆附近（这样 “距离” 就能反映方向上的相似度）

- **基于预测的词向量：GloVe和Word2Vec**

词向量可用于类比推理

第三讲：神经网络学习

链式法则：

- 传播时有个雅可比矩阵为对角矩阵，可以转化为与向量的阿达马积（逐元素相乘）
- 前面内容（上游梯度）还可以计算复用
- 算梯度时考虑形状匹配
- 形状匹配便于梯度，雅可比形式便于应用链式法则

反向传播：

- 计算图
- 对于单个节点，接收上游梯度，计算局部梯度，输出下游梯度（多个输入，多个下游梯度）
- 传播的时候都是数字
- 一次性计算所有梯度（计算复用）
- 手动梯度检查：数值梯度（检查各层是否正确实现）

第四讲：依存句法分析（Dependency Parsing）

句法结构：一致性与依存关系

- 模型需要理解句子结构才能正确解读语言
- 依赖路径有助于提取语义解释
- 依存句法假定，句法结构由词汇项之间的关系构成，通常是被称为依存关系的二元非对称关系（“箭头”）

依存语法与树库

基于转换的依存句法分析

神经依存句法分析

作业2

word2vec

机器学习和神经网络

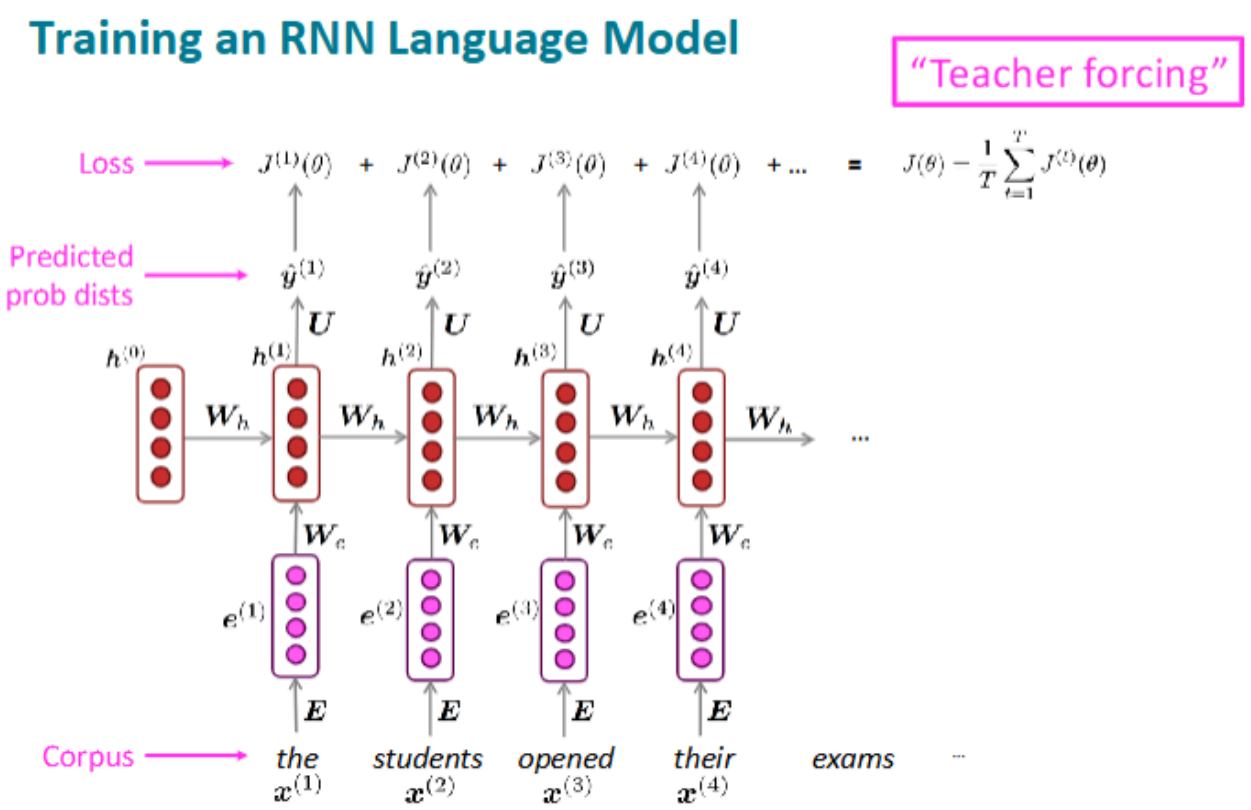
基于神经转换的依存句法分析

- 有个stack（初始有个root），有个buffer（初始为句子）
- 每次进行三个转换中的一个（可用神经网络来判断用哪个转换）
- 有两个转换需要确定依存关系
- 直至栈还剩root

- ```
    if transition == "S":
        if self.buffer:
            self.stack.append(self.buffer.pop(0))
    elif transition == "LA":
        dependent = self.stack.pop(-2)
        head = self.stack[-1]
        self.dependencies.append((head, dependent))
    elif transition == "RA":
        dependent = self.stack.pop()
        head = self.stack[-1]
        self.dependencies.append((head, dependent))
  ```

第五讲：LMs和RNNs

语言建模是预测下一个单词是什么的任务

- n元语法语言模型
- 固定窗口的神经语言模型
- RNN：生成滚动输出
- 

- 梯度爆炸解决：梯度裁剪：如果梯度的范数大于某个阈值，则在应用随机梯度下降（SGD）更新之前将其按比例缩小
- 梯度消失解决：LSTM

第六讲：LSTM和神经机器翻译

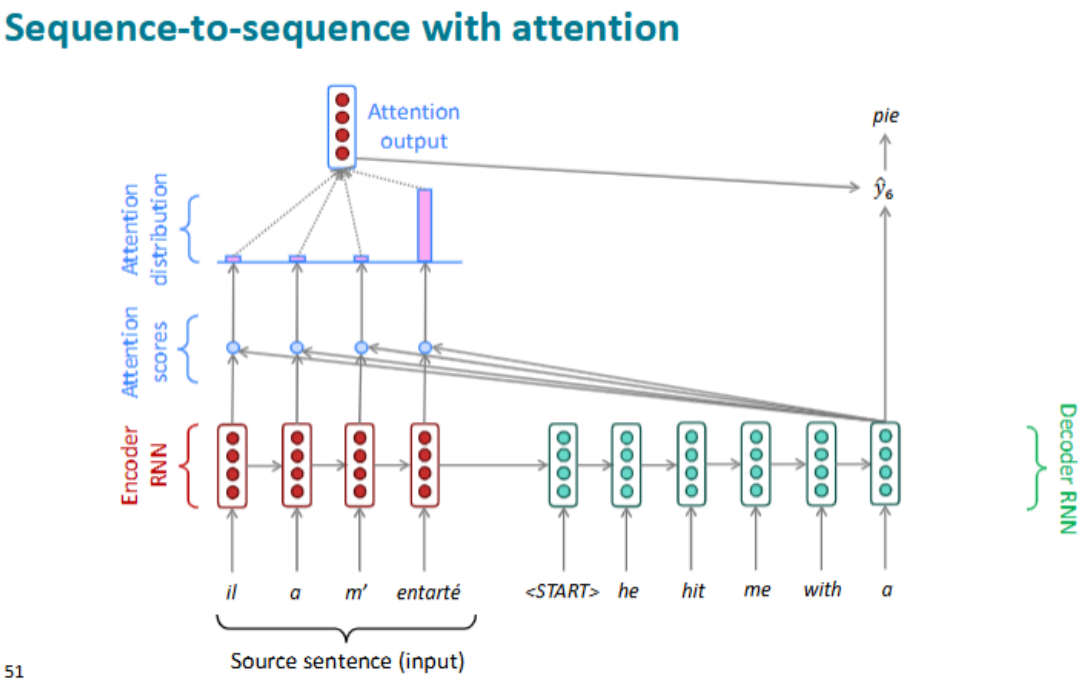
残差连接、密集连接、highway连接
  ```

双向RNN：适用于能够获得完整输入序列，不适用于语言建模（只能获得左侧语境）

多层RNN

机器翻译

- SMT、NMT
- 编码器-解码器NMT
- 带注意力机制的seq2seq

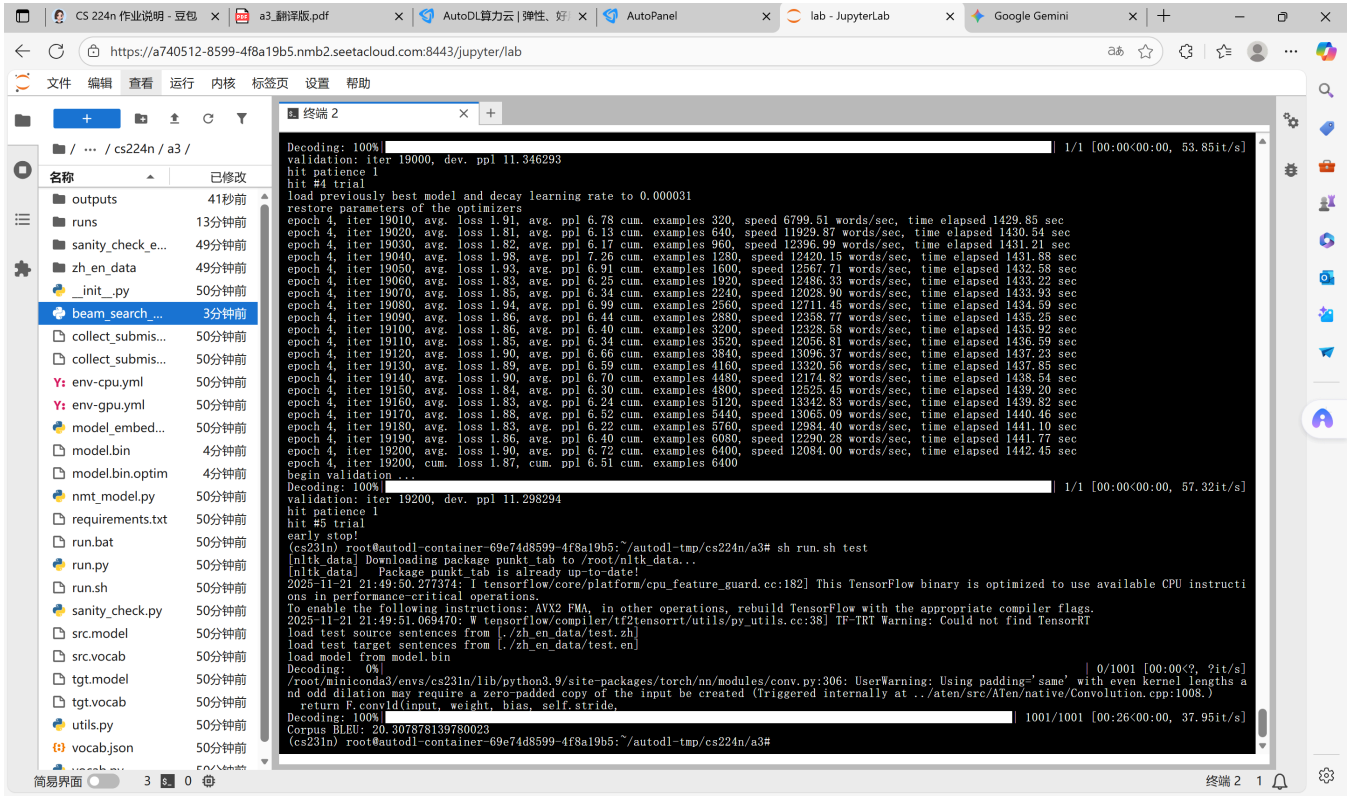


作业3

pytorch中，LSTM/RNN和CNN对输入数据的处理维度要求不一样，使用时要注意转换

维度转换：`torch.permute(enc_hiddens, (1, 0, 2))`

作为输入给解码器时，需要去掉最后一个`<end>`



第八讲：transformer

大部分模型前归一化，但BERT后归一化

操作次数为n平方，一般序列长度为512，无法处理真正长的上下文；进而RNN回归，例如RWKV、Mamba

这个成本可以接受，实际上也是这样用的

第九讲：预训练

第十讲：后训练

作业4

RoPE

- 动机：绝对位置编码在长上下文下表现不佳，因为推理时的位置编码分布与训练时差异极大
- 改进：相对位置编码可缓解分布不匹配
- 让 token 的编码向量包含 “与其他 token 的相对位置信息”，相当于模型只需要学习 “相对距离 k 的语义意义”

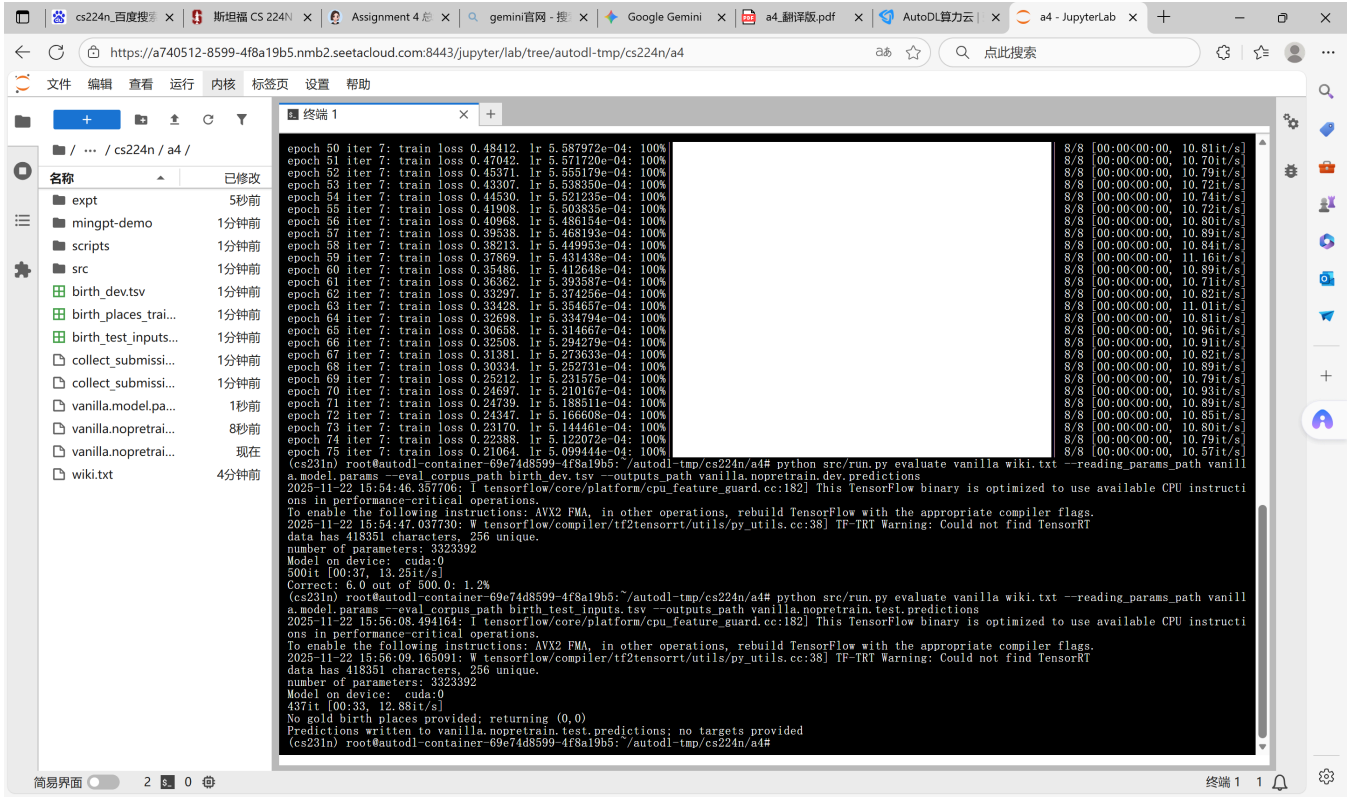
- 绝对位置编码是 “死记硬背每个位置的编码”，长上下文会遇到 “没背过的编码”；RoPE 是 “学习位置间的相对关系”，不管上下文多长，只要相对关系见过，就能正确处理
- 细节：旋转角度θ是位置t的函数

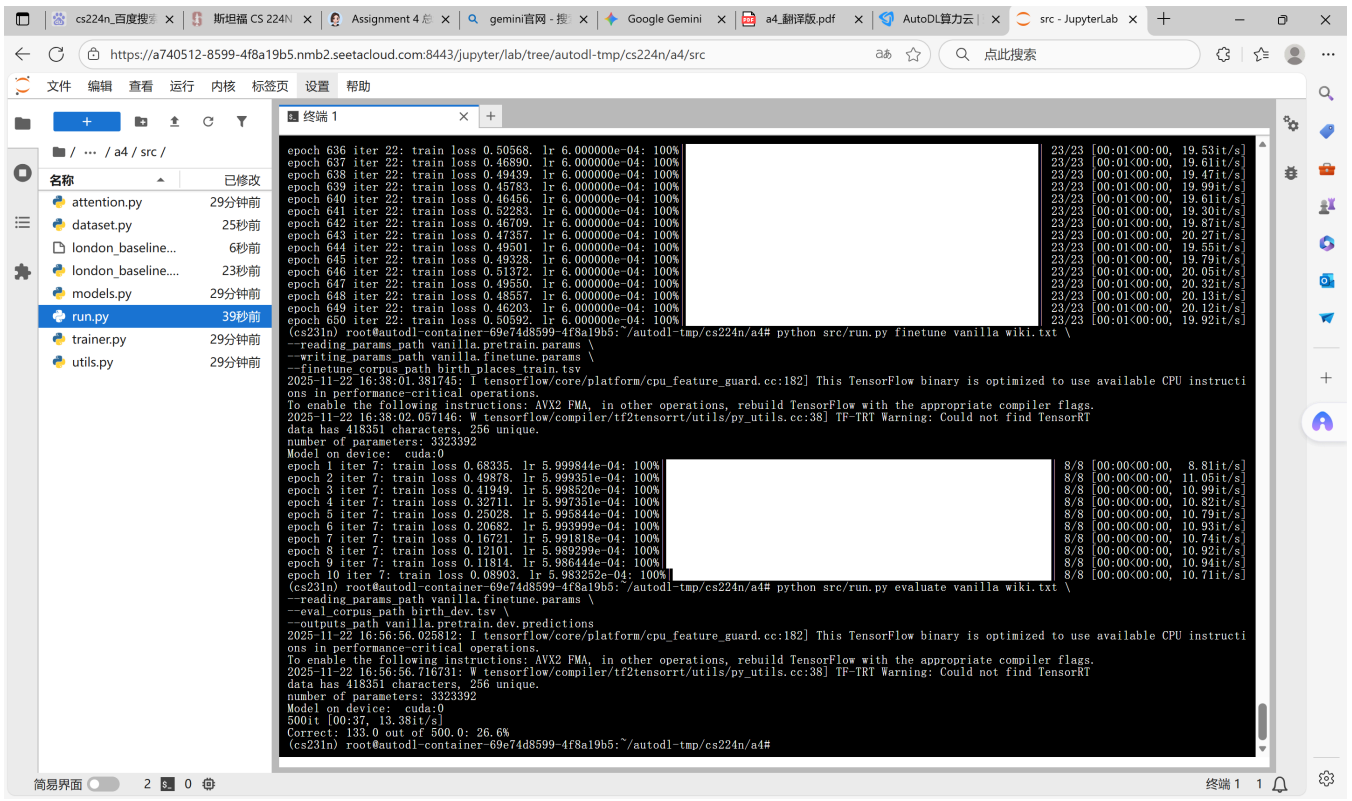
For a d dimensional feature, RoPE is applied to each pair of features with an angle θ_i defined as $\theta_i = 10000^{-2(i-1)/d}$, $i \in \{1, 2, \ldots, d/2\}$.

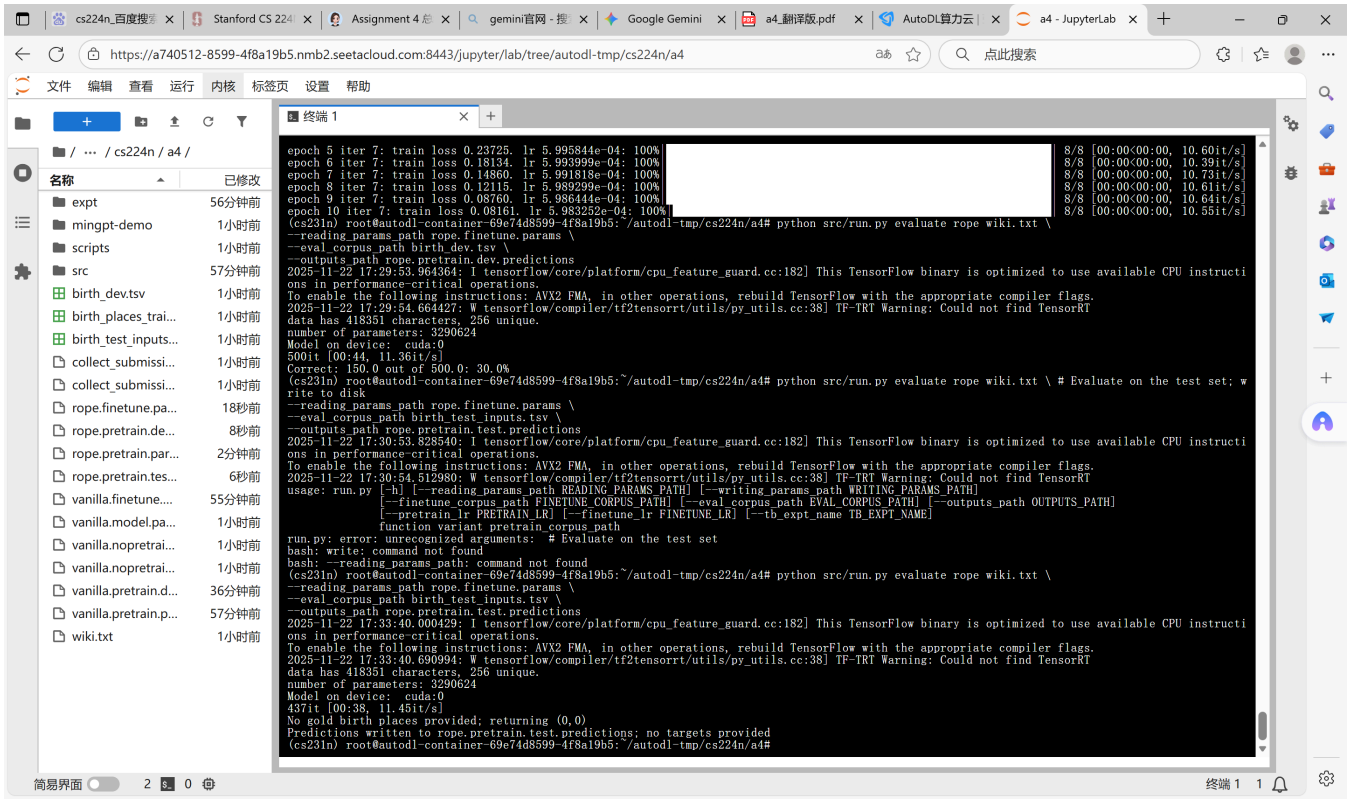
$$\begin{pmatrix} \cos t\theta_1 & -\sin t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin t\theta_1 & \cos t\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos t\theta_2 & -\sin t\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin t\theta_2 & \cos t\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos t\theta_{d/2} & -\sin t\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin t\theta_{d/2} & \cos t\theta_{d/2} \end{pmatrix} \begin{pmatrix} x_t^{(1)} \\ x_t^{(2)} \\ x_t^{(3)} \\ x_t^{(4)} \\ \vdots \\ x_t^{(d-1)} \\ x_t^{(d)} \end{pmatrix} \tag{3}$$

Finally, instead of adding the positional embeddings to the input embeddings, RoPE is applied to the key and query vectors for each head in the attention block for all the Transformer layers.

- 位置 t_1 和 t_2 处向量的RoPE嵌入点积仅依赖于相对位置 $t_1 - t_2$
- **具体流程如下：**
  1. **输入：** 你的文本经过 Embedding 层和一些 Transformer Block。
  2. **线性变换：** 数据经过 Linear 层生成了 Query (Q), Key (K), Value (V)。
  3. **应用 RoPE (关键点)：** 在拿 Q 和 K 去计算点积（Dot Product）之前，**必须先通过这个函数进行旋转**。
     - *注意：通常只对 Q 和 K 用，不对 V 用。*
  4. **注意力计算：** 使用旋转后的 Q' 和 K' 计算注意力：$\text{Softmax}(Q' \cdot (K')^T)$。




src - JupyterLab
https://a740512-8599-4f8a19b5.nmb2.seetacloud.com:8443/jupyter/lab/tree/autodl-tmp/cs224n/a4/src
文件 编辑 查看 运行 内核 标签页 设置 帮助
/ a4 / src /
名称 已修改
attention.py 29分钟前
dataset.py 25秒前
london_baseline... 6秒前
london_baseline.... 23秒前
models.py 29分钟前
run.py 39秒前
trainer.py 29分钟前
utils.py 29分钟前
终端 1
epoch 636 iter 22: train loss 0.50568. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 19.53it/s]
epoch 637 iter 22: train loss 0.46890. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 19.61it/s]
epoch 638 iter 22: train loss 0.49439. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 19.47it/s]
epoch 639 iter 22: train loss 0.45783. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 19.99it/s]
epoch 640 iter 22: train loss 0.46456. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 19.61it/s]
epoch 641 iter 22: train loss 0.52283. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 19.30it/s]
epoch 642 iter 22: train loss 0.46709. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 19.87it/s]
epoch 643 iter 22: train loss 0.47357. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 20.27it/s]
epoch 644 iter 22: train loss 0.49501. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 19.55it/s]
epoch 645 iter 22: train loss 0.49328. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 19.79it/s]
epoch 646 iter 22: train loss 0.51372. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 20.05it/s]
epoch 647 iter 22: train loss 0.49550. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 20.32it/s]
epoch 648 iter 22: train loss 0.48557. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 20.13it/s]
epoch 649 iter 22: train loss 0.46203. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 20.12it/s]
epoch 650 iter 22: train loss 0.50592. lr 6.000000e-04: 100% 23/23 [00:01<00:00, 19.92it/s]
(cs231n) root@autodl-container-69e74d8599-4f8a19b5:~/autodl-tmp/cs224n/a4# 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
2025-11-22 16:38:01.381745: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2025-11-22 16:38:02.057146: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
data has 418351 characters, 256 unique.
number of parameters: 3323392
Model on device: cuda:0
epoch 1 iter 7: train loss 0.68335. lr 5.999844e-04: 100% 8/8 [00:00<00:00, 8.81it/s]
epoch 2 iter 7: train loss 0.49878. lr 5.999351e-04: 100% 8/8 [00:00<00:00, 11.05it/s]
epoch 3 iter 7: train loss 0.41949. lr 5.998520e-04: 100% 8/8 [00:00<00:00, 10.99it/s]
epoch 4 iter 7: train loss 0.32711. lr 5.997351e-04: 100% 8/8 [00:00<00:00, 10.82it/s]
epoch 5 iter 7: train loss 0.25028. lr 5.995844e-04: 100% 8/8 [00:00<00:00, 10.79it/s]
epoch 6 iter 7: train loss 0.20682. lr 5.993999e-04: 100% 8/8 [00:00<00:00, 10.93it/s]
epoch 7 iter 7: train loss 0.16721. lr 5.991818e-04: 100% 8/8 [00:00<00:00, 10.74it/s]
epoch 8 iter 7: train loss 0.12101. lr 5.989299e-04: 100% 8/8 [00:00<00:00, 10.92it/s]
epoch 9 iter 7: train loss 0.11814. lr 5.986444e-04: 100% 8/8 [00:00<00:00, 10.94it/s]
epoch 10 iter 7: train loss 0.08903. lr 5.983252e-04: 100% 8/8 [00:00<00:00, 10.71it/s]
(cs231n) root@autodl-container-69e74d8599-4f8a19b5:~/autodl-tmp/cs224n/a4# 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
2025-11-22 16:56:56.025812: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2025-11-22 16:56:56.716731: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
data has 418351 characters, 256 unique.
number of parameters: 3323392
Model on device: cuda:0
500it [00:37, 13.38it/s]
Correct: 133.0 out of 500.0: 26.6%
(cs231n) root@autodl-container-69e74d8599-4f8a19b5:~/autodl-tmp/cs224n/a4#
简易界面
终端 1


a4 - JupyterLab
https://a740512-8599-4f8a19b5.nmb2.seetacloud.com:8443/jupyter/lab/tree/autodl-tmp/cs224n/a4
文件 编辑 查看 运行 内核 标签页 设置 帮助
/ cs224n / a4 /
名称 已修改
expt 56分钟前
mingpt-demo 1小时前
scripts 1小时前
src 57分钟前
birth_dev.tsv 1小时前
birth_places_trai... 1小时前
birth_test_inputs... 1小时前
collect_submissi... 1小时前
collect_submissi... 1小时前
rope.finetune.pa... 18秒前
rope.pretrain.de... 8秒前
rope.pretrain.par... 2分钟前
rope.pretrain.tes... 6秒前
vanilla.finetune.... 55分钟前
vanilla.model.pa... 1小时前
vanilla.nopretrai... 1小时前
vanilla.nopretrai... 1小时前
vanilla.pretrain.d... 36分钟前
vanilla.pretrain.p... 57分钟前
wiki.txt 1小时前
终端 1
epoch 5 iter 7: train loss 0.23725. lr 5.995844e-04: 100% 8/8 [00:00<00:00, 10.60it/s]
epoch 6 iter 7: train loss 0.18134. lr 5.993999e-04: 100% 8/8 [00:00<00:00, 10.39it/s]
epoch 7 iter 7: train loss 0.14860. lr 5.991818e-04: 100% 8/8 [00:00<00:00, 10.73it/s]
epoch 8 iter 7: train loss 0.12115. lr 5.989299e-04: 100% 8/8 [00:00<00:00, 10.61it/s]
epoch 9 iter 7: train loss 0.08760. lr 5.986444e-04: 100% 8/8 [00:00<00:00, 10.64it/s]
epoch 10 iter 7: train loss 0.08161. lr 5.983252e-04: 100% 8/8 [00:00<00:00, 10.55it/s]
(cs231n) root@autodl-container-69e74d8599-4f8a19b5:~/autodl-tmp/cs224n/a4# python src/run.py evaluate rope wiki.txt \
--reading_params_path rope.finetune.params \
--eval_corpus_path birth_dev.tsv \
--outputs_path rope.pretrain.dev.predictions
2025-11-22 17:29:53.964364: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2025-11-22 17:29:54.664427: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
data has 418351 characters, 256 unique.
number of parameters: 3290624
Model on device: cuda:0
500it [00:44, 11.36it/s]
Correct: 150.0 out of 500.0: 30.0%
(cs231n) root@autodl-container-69e74d8599-4f8a19b5:~/autodl-tmp/cs224n/a4# python src/run.py evaluate rope wiki.txt \ # Evaluate on the test set; write to disk
--reading_params_path rope.finetune.params \
--eval_corpus_path birth_test_inputs.tsv \
--outputs_path rope.pretrain.test.predictions
2025-11-22 17:30:53.828540: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2025-11-22 17:30:54.512980: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
usage: run.py [-h] [--reading_params_path READING_PARAMS_PATH] [--writing_params_path WRITING_PARAMS_PATH]
[--finetune_corpus_path FINETUNE_CORPUS_PATH] [--eval_corpus_path EVAL_CORPUS_PATH] [--outputs_path OUTPUTS_PATH]
[--pretrain_lr PRETRAIN_LR] [--finetune_lr FINETUNE_LR] [--tb_expt_name TB_EXPT_NAME]
function variant pretrain_corpus_path
run.py: error: unrecognized arguments: # Evaluate on the test set
bash: write: command not found
bash: --reading_params_path: command not found
(cs231n) root@autodl-container-69e74d8599-4f8a19b5:~/autodl-tmp/cs224n/a4# python src/run.py evaluate rope wiki.txt \
--reading_params_path rope.finetune.params \
--eval_corpus_path birth_test_inputs.tsv \
--outputs_path rope.pretrain.test.predictions
2025-11-22 17:33:40.000429: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2025-11-22 17:33:40.690994: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
data has 418351 characters, 256 unique.
number of parameters: 3290624
Model on device: cuda:0
437it [00:38, 11.45it/s]
No gold birth places provided; returning (0,0)
Predictions written to rope.pretrain.test.predictions; no targets provided
(cs231n) root@autodl-container-69e74d8599-4f8a19b5:~/autodl-tmp/cs224n/a4#
简易界面
终端 1

书面回答 from ai

CS 224n Assignment 3: Written Solutions

1. Neural Machine Translation with RNNs

(g) The Effect of Masks in Attention

Question: Explain what effect the masks have on the entire attention computation and why it is necessary to use the masks in this way.

Answer:

掩码（Masks）在注意力计算中的主要作用是防止解码器（Decoder）“关注”到源句子中的填充标记（pad tokens）。

在计算注意力分布 α_t 时，我们通过 softmax 函数对注意力分数 e_t 进行归一化。如果不使用掩码，填充位置的分数 $e_{t,i}$ 会参与 softmax 计算，导致模型分配非零的概率给无意义的填充符，这会“稀释”分配给真实单词的注意力权重，并使上下文向量 a_t 包含无用的填充信息。通过将填充位置的分数设置为极小的负数（如 $-\infty$），softmax 后的概率将变为 0，从而确保模型只关注实际的源句子内容。

(h) BLEU Score Report

Answer:

(注：此题需要您在本地或虚拟机上运行完测试脚本后填写具体数值。以下为示例格式)

Corpus BLEU Score: [在此填写您的分数，例如 22.5]

(The score should be larger than 18.0)

(i) Attention Mechanisms Comparison

i. Dot Product vs. Multiplicative **Attention**

- **优势 (Advantage of Dot Product):** 计算效率更高且空间复杂度更低。点积注意力 ($e_{t,i} = s_t^T h_i$) 不需要额外的权重矩阵 W，减少了参数数量和计算量。
- **劣势 (Disadvantage of Dot Product):** 灵活性较差。它要求解码器隐藏状态 s_t 和编码器隐藏状态 h_i 必须具有相同的维度。如果维度不同，无法直接进行点积计算。

ii. Additive vs. Multiplicative Attention

- **优势 (Advantage of Additive):** 在高维空间或维度差异较大时表现通常更稳健。加性注意力 ($e_{t,i} = v^T \tanh(W_1 h_i + W_2 s_t)$) 通过非线性变换 ($\tanh$) 解耦了源和目标的表示，在早期 NMT 研究中表现往往略优于乘性注意力。
- **劣势 (Disadvantage of Additive):** 计算效率较低。它涉及更复杂的运算（矩阵乘法后接激活函数再接向量乘法），不如乘性注意力（主要是高度优化的矩阵乘法）那样容易并行化和加速。

2. Analyzing NMT Systems

(a) 1D Convolutional Layer for Chinese

Question: How could adding a 1D Convolutional layer ... help our NMT system?

Answer:

中文是一种基于字符的语言，很多词汇（Words）是由多个字符（Characters）组成的。例如，“电” (Electricity) 和“脑” (Brain) 组合在一起形成 “电脑” (Computer)。

在嵌入层（Embedding Layer）之后、双向编码器（Bidirectional Encoder）之前加入 1D 卷积层，可以起到类似 N-gram 特征提取的作用。卷积核可以捕捉局部上下文，将相邻的字符（字）聚合为更高级的语义单元（词或短语表示）。这使得 LSTM 编码器能够处理具有更丰富语义信息的输入，而不是仅处理单个字符的嵌入，从而更容易捕捉长距离依赖关系并理解复合词的含义。

(b) Error Analysis

i. Error: `sentenced to theft`

- **Reason:** 模型混淆了动词结构和名词短语。源句中“被判处”（sentenced to/convicted of）后面紧跟“盗窃罪名”（theft crime/charges）和“成立”（established/proven）。模型可能错误地将“盗窃”直接作为“判处”的直接宾语（sentenced to theft），而不是理解为“被判盗窃罪名成立”（convicted of theft）。这是句法结构解析的错误。
- **Fix:** 增加对复杂句法结构的训练数据，或者使用依赖句法分析（Dependency Parsing）作为辅助特征输入到编码器中，以帮助模型理解“判处...成立”这一搭配。

ii. Error: `the resources have been exhausted and resources have been exhausted`

- **Reason:** 这是神经机器翻译（尤其是 RNN 模型）中常见的“重复生成”问题（Repetition）。当模型在解码过程中“忘记”已经翻译过某些信息（如“资源已经用尽”），或者注意力机制在某些源标记上滞留时，就会产生重复。
- **Fix:** 引入覆盖率惩罚（Coverage Penalty）。在损失函数中加入一项，惩罚解码器多次关注同一个源词，或者强制模型在解码结束前覆盖所有源词。

iii. Error: `today's day`

- **Reason:** 翻译错误，未能识别专有名词。源词“国殇日”（National Mourning Day）是一个特定的节日/实体。模型可能没有在词表中见过这个词（OOV），或者将其拆解为“国”（Country）、“殇”（Death/Wound）、“日”（Day/Date）处理，导致生成了语义不明的 "today's day"。
- **Fix:** 使用更细粒度的子词单元（Subword units, 如 Byte Pair Encoding）来更好地处理罕见词；或者引入指针/复制机制（Copy Mechanism），允许模型在遇到未知实体时直接从源句复制或查表翻译。

iv. Error: `it's not wrong`

- **Reason:** 习语/俗语翻译错误。源句“唔做唔错”是粤语俗语，意为“不做就不会做错”（Do nothing, make no mistakes）。模型按字面意思翻译（唔=not, 做=do, 错=wrong），未能理解其作为谚语的整体隐喻含义。
- **Fix:** 在训练数据中增加包含成语、俗语及其对应意译的平行语料；或者建立一个外部的短语表（Phrase Table）/记忆库，专门用于检索和替换常见的固定表达。

(c) BLEU Score Calculation

i. Two References (r_1, r_2)

- **Reference 1 (r_1):** resources have to be sufficient and they have to be predictable (Length $r = 11$)
- **Reference 2 (r_2):** adequate and predictable resources are required (Length $r = 6$)

Candidate 1 (c_1): there is a need for adequate and predictable resources

- **Unigrams (p_1):**
  - Present in refs: *adequate, and, predictable, resources* (4 words)
  - Total length: 9 words
  - $p_1 = 4/9$img
- **Bigrams (p_2):**
  - Present in refs: *adequate and* (r_2), *and predictable* (r_1, r_2), *predictable resources* (r_2) (3 bigrams)
  - Total bigrams: 8
  - $p_2 = 3/8$img
- **Brevity Penalty (BP):**
  - Length $c = 9$.
  - Closest reference length: r_1 (11, diff=2) vs r_2 (6, diff=3). Closest is 11.
  - Since $c < r$, $BP = e^{1-11/9} = e^{-2/9} \approx 0.8007$.
- **BLEU (c_1):**
  - $Score = 0.8007 \times \exp(0.5 \ln(4/9) + 0.5 \ln(3/8))$img
  - $Score = 0.8007 \times \exp(0.5(-0.8109) + 0.5(-0.9808))$img
  - $Score = 0.8007 \times \exp(-0.8959) \approx 0.8007 \times 0.4082 \approx \mathbf{0.327}$img

Candidate 2 (c_2): resources be sufficient and predictable to

- **Unigrams (p_1):**
  - Present in refs: *resources, be, sufficient, and, predictable, to* (All 6 appear in r_1)
  - $p_1 = 6/6 = 1.0$img
- **Bigrams (p_2):**
  - Matches: *be sufficient* (r_1), *sufficient and* (r_1), *and predictable* (r_2)
  - Note: *resources be* (No), *predictable to* (No).
  - Total bigrams: 5. Matches: 3.
  - $p_2 = 3/5 = 0.6$img
- **Brevity Penalty (BP):**
  - Length $c = 6$.
  - Closest reference length: r_2 (6).
  - Since $c \geq r$, $BP = 1.0$.
- **BLEU (c_2):**

- $Score = 1.0 \times \exp(0.5 \ln(1.0) + 0.5 \ln(0.6))$img
- $Score = 1.0 \times \sqrt{1 \times 0.6} \approx \mathbf{0.775}$img

Comparison:

c_2 (0.775) has a much higher BLEU score than c_1 (0.327).

Do I agree? No. c_1 is a grammatically correct and meaningful English sentence that captures the meaning well. c_2 ("resources be sufficient and predictable to") is grammatically broken and ends abruptly. BLEU favors c_2 here because it heavily rewards high n-gram overlap with r_1 and short length matches r_2, failing to penalize the grammatical incoherence.

ii. Single Reference (r_2 only)

- **Reference (r_2):** adequate and predictable resources are required (Length $r = 6$)

Candidate 1 (c_1): there is a need for adequate and predictable resources (Len=9)

- p_1: *adequate, and, predictable, resources* -> 4/9.
- p_2: *adequate* and, *and predictable, predictable resources* -> 3/8.
- BP: $c(9) > r(6) \rightarrow BP = 1.0$.
- **BLEU (c_1):** $1.0 \times \exp(0.5 \ln(4/9) + 0.5 \ln(3/8)) \approx \mathbf{0.408}$img

Candidate 2 (c_2): resources be sufficient and predictable to (Len=6)

- p_1: *and, predictable, resources* -> 3/6 = 0.5.
- p_2: *and predictable* -> 1/5 = 0.2.
- BP: $c(6) = r(6) \rightarrow BP = 1.0$.
- **BLEU (c_2):** $1.0 \times \exp(0.5 \ln(0.5) + 0.5 \ln(0.2)) = \sqrt{0.1} \approx \mathbf{0.316}$img

Comparison:

Now c_1 (0.408) scores higher than c_2 (0.316).

Do I agree? Yes, this ranking is better. c_1 is indeed the better translation. However, the scores are still quite low because strict n-gram matching misses that "there is a need for" is synonymous with "are required".

iii. Problems with Single Reference

使用单一参考译文进行评估是有问题的，因为语言具有多样性（Ambiguity and Diversity）。同一个源句子可以有多种完全正确但措辞不同的翻译方式。

如上例所示，如果模型生成的翻译 (c_1) 准确地表达了意思，但使用的词汇或句式与唯一的参考译文 (r_2) 不同，BLEU 分数会惩罚这种“不匹配”。当有多个参考译文时，BLEU 更有可能在某一个参考中找到匹配的 n-gram，从而更公平地评估翻译质量。单一参考会导致对有效意译的低估（Underestimation of valid paraphrases）。

iv. Advantages and Disadvantages of BLEU

- **Advantages:**
    1. **自动化与低成本 (Automatic & Cheap):** 计算极其快速，无需雇佣昂贵的人类专家，适合在训练过程中频繁评估。
    2. **语言无关性 (Language Independent):** 不依赖于特定的语言学知识或模型，适用于任何语言对。
- **Disadvantages:**

1. **忽略语义与连贯性 (Ignores Meaning & Coherence):** 仅基于 n-gram 表面匹配，无法检测同义词替换（除非多参考），也不能很好地判断句子的语法结构或逻辑连贯性（如 c_2 在 i 题中的高分）。
2. **权重分配平均 (Blind to Importance):** 将所有单词视为同等重要，忽略了关键词（如核心动词、否定词）的错误对句意造成的巨大影响。

(d) Beam Search (General Concept)

(Note: Specific examples require your TensorBoard logs, but here is the qualitative analysis)

i. Quality Improvement:

通常情况下，翻译质量会随着训练迭代次数的增加而提高。

- **Early Iterations (e.g., 200):** 模型可能输出乱码、重复词或完全不相关的词。
- **Mid Iterations (e.g., 3000):** 模型开始捕捉基本的句法结构，但在生僻词或长句逻辑上可能出错。
- **Final Iteration:** 翻译应当流畅且在语义上忠实于原文。

ii. Beam Search Hypotheses Comparison:

Beam Search 产生的不同假设（Hypotheses）通常是相似的，但在局部措辞或标点上略有不同。

- **Rank 1 (Best):** 通常是概率最高的，语法最通顺。
- **Lower Ranks:** 可能会有同义词替换，或者在句子结尾处出现不同的截断/长度。有时较低排名的假设可能包含更生僻但准确的词，或者包含一些重复/

CS 224N Winter 2025 Assignment 4 - 解答指南

这份文档包含了 Assignment 4 中书面问题（数学推导与分析部分）的参考解答思路。请根据题目要求，结合数学公式进行完整表述。

1. Attention Exploration (14 points)

这一部分探讨自注意力机制的数学特性，以及为什么多头注意力（Multi-head Attention）通常优于单头注意力。

(a) Copying in attention (3 points)

i. 概率分布集中的条件 (2 points)

题目问在什么条件下，注意力权重 α 会几乎全部集中在某一个特定的 j 上（即 $\alpha_j \gg \sum_{i \neq j} \alpha_i$）。

- 解答思路:

  注意力权重由 Softmax 计算得出：$\alpha_i = \frac{\exp(k_i^\top q)}{\sum \exp(k_j^\top q)}$。

  为了使 α_j 接近 1，分子 $\exp(k_j^\top q)$ 必须远大于其他项。这意味着点积 $k_j^\top q$ 必须显著大于其他的点积 $k_i^\top q$（对于所有 $i \neq j$）。

- 参考描述:

  查询向量 q 必须与键向量 k_j 高度相似（点积大），同时与其他键向量 $k_i (i \neq j)$ 正交或差异很大（点积小）。

ii. 输出 c 的形式 (1 point)

- 解答思路:

  $c = \sum v_i \alpha_i$。如果 $\alpha_j \approx 1$ 且其他 $\alpha_i \approx 0$。

- 参考描述:

  输出向量 c 将近似等于值向量 v_j（即 $c \approx v_j$）。

(b) An average of two (2 points)

题目: 假设我们要让 $c \approx \frac{1}{2}(v_a + v_b)$。已知所有 k_i 正交且为单位向量。求 q 的表达式。

- 解答思路:

  我们需要 $\alpha_a \approx \alpha_b \approx 0.5$，且其他权重接近 0。

  这意味着 $k_a^\top q \approx k_b^\top q$，且这两个值都要远大于其他 $k_i^\top q$。

  由于 k 是正交单位向量，如果我们设 $q = \beta(k_a + k_b)$（其中 β 是一个较大的标量），那么：

  $k_a^\top q = \beta(1 + 0) = \beta$

  $k_b^\top q = \beta(0 + 1) = \beta$

  $k_i^\top q = 0$ (对于 $i \neq a, b$)

  经过 Softmax，当 β 很大时，$\exp(\beta)$ 远大于 $\exp(0)$，权重集中在 a 和 b 上。

- 参考答案:

  $q = \beta(k_a + k_b)$，其中 β 是一个足够大的标量。

(c) Drawbacks of single-headed attention (5 points)

i. 设计 q (2 points)

假设 $k_i \sim \mathcal{N}(\mu_i, \alpha I)$，$\mu_i$ 正交单位向量，Σ_i 极小。设计 q 使 $c \approx \frac{1}{2}(v_a + v_b)$。

- 解答思路:

  当方差极小时，$k_i \approx \mu_i$。

  利用 (b) 的结论，我们可以利用均值向量 μ 来设计 q。

- 参考答案:

  $q = \beta(\mu_a + \mu_b)$，其中 β 很大。因为 k_i 集中在 μ_i 附近，且 μ_i 正交，该查询向量能同时“激活” k_a 和 k_b。

ii. 较大扰动的影响 (3 points)

假设 $\Sigma_a = \alpha I + \frac{1}{2}(\mu_a \mu_a^\top)$，导致 k_a 在 μ_a 方向上的模长变化很大。

- 解答思路:

  现在 k_a 的模长是不确定的（方差大），可能很大也可能很小。

  计算点积 $k_a^\top q$ 时，由于 q 包含 μ_a 分量，点积的值会随 k_a 的模长剧烈波动。

  Softmax 对数值大小非常敏感。

  - 如果采样到的 k_a 模长很大，$k_a^\top q$ 就会比 $k_b^\top q$ 大很多，导致 $\alpha_a \approx 1, \alpha_b \approx 0$，输出 $c \approx v_a$。
  - 如果采样到的 k_a 模长很小，$k_a^\top q$ 可能小于 $k_b^\top q$，导致 $\alpha_a \ll \alpha_b$，输出 $c \approx v_b$。

- 参考描述:

  输出 c 将在 v_a 和 v_b 之间剧烈波动（高方差）。单头注意力无法在键向量模长变化较大时稳定地保持“平均”两个向量的比例，通常会坍缩到其中某一个上。

(d) Benefits of multi-headed attention (3 points)

i. 设计 q_1, q_2 (1 point)

多头注意力输出 $\frac{1}{2}(c_1 + c_2)$。设计 q_1, q_2 使总输出为 $\frac{1}{2}(v_a + v_b)$。

- 解答思路:

  与其让一个头同时关注两个（像 (c) 那样不稳定），不如让每个头各关注一个。

  让 Head 1 关注 a：$q_1 = \beta\mu_a \to c_1 \approx v_a$。

  让 Head 2 关注 b：$q_2 = \beta\mu_b \to c_2 \approx v_b$。

  平均后得到 $0.5(v_a + v_b)$。

- 参考答案:

  $q_1 = \beta\mu_a, \quad q_2 = \beta\mu_b$。

ii. 扰动下的定性分析 (2 points)

同样假设 k_a 在 μ_a 方向方差大。

- **解答思路**:
  - **Head 1 ($q_1 = \mu_a$)**: 关注 k_a。由于 k_a 模长波动，虽然它主要还是关注 a（假设 $k_a^\top q_1$ 仍大于其他噪声），但其 Softmax 的尖锐程度可能会变，不过既然目标只是 v_a，只要它是最大的就好。或者，如果 k_a 变小，可能混入噪声；如果变大，就更确信是 v_a。总体来说 $c_1 \approx v_a$（可能会有一些波动，但方向是对的）。
  - **Head 2 ($q_2 = \mu_b$)**: 关注 k_b。关键点在于 k_a 对 q_2 的影响。因为 k_a 主要在 μ_a 方向，而 q_2 在 μ_b 方向，且 $\mu_a \perp \mu_b$，所以 $k_a^\top q_2 \approx 0$。这意味着 k_a 的模长变化**不影响** Head 2。c_2 将非常稳定地等于 v_b。
- 参考描述:

  c_2 将保持稳定（因为 q_2 与 k_a 正交），而 c_1 可能受 k_a 模长影响。但最终的平均值 $\frac{1}{2}(c_1 + c_2)$ 中，至少有一半的内容（v_b 部分）是完全稳定的。相比单头注意力在 v_a 和 v_b 之间大幅摇摆，多头注意力能保证 v_a 和 v_b 均被“捕获”，只是混合比例的方差会比单头情况小很多，或者至少 v_b 成分不会丢失。

(e) Summary (1 point)

- 参考总结:

  多头注意力允许模型将不同的关注点“解耦”到不同的头上。即使某些键向量存在噪声或扰动，各头可以独立运作，互不干扰（正交性），从而使聚合后的结果比试图用单个查询向量同时平衡多个目标更鲁棒。

2. Position Embeddings Exploration (6 points)

(a) Permuting the input (4 points)

i. 证明 $Z_{perm} = PZ$ (3 points)

- **解答思路**:

  1. Attention 层：$A = \text{softmax}(QK^\top)$。

     $X_{perm} = PX$。

     $Q_{perm} = (PX)W_Q = PQ$。

     $K_{perm} = (PX)W_K = PK$。

     $V_{perm} = (PX)W_V = PV$。

Attention Logits: $Q_{perm}K_{perm}^\top = (PQ)(PK)^\top = PQK^\top P^\top$。

Attention Weights: $\text{softmax}(P(QK^\top)P^\top) = P\text{softmax}(QK^\top)P^\top = PAP^\top$ (利用题目给的性质)。

Output $H_{perm} = A_{perm}V_{perm} = (PAP^\top)(PV) = PA(P^\top P)V = PAV = PH$ (因为 P 是置换矩阵，正交，$P^\top P = I$)。

2. Feed Forward 层：是逐点操作 (Point-wise)。

$Z_{perm} = \text{FFN}(H_{perm}) = \text{FFN}(PH)$。

利用性质 $\text{ReLU}(PA) = P\text{ReLU}(A)$，线性变换同理。

所以 $Z_{perm} = PZ$。

ii. 这种性质的问题 (1 point)

- 参考答案:

  这说明如果没有位置编码，Transformer 是**置换等变（permutation equivariant）**的（甚至在做文本分类等池化任务时是置换不变的）。这意味着模型无法区分词序。例如 "The dog bit the man" 和 "The man bit the dog" 对模型来说是一样的，这对于理解自然语言（严重依赖语序）是灾难性的。

(b) Position Embeddings (2 points)

i. 是否解决问题？ (1 point)

- 参考答案:

  是的。加入位置编码后，$X_{pos} = X + \Phi$。每个位置的 token 向量加上了独特的位置向量。此时若交换输入词的顺序，词向量 x_i 会加上不同的位置向量 p_j，导致输入矩阵不再仅仅是简单的行置换，破坏了上述的置换等变性，使模型能感知顺序。

ii. 不同 token 的位置编码是否可能相同？ (1 point)

- 参考答案:

  不会。位置编码由不同频率的 $\sin$ 和 $\cos$ 函数组成。只要位置索引 $t_1 \neq t_2$，在维度 d 足够大的情况下，由正弦/余弦函数构成的向量组合是唯一的（类似于二进制编码的连续版本），不会出现碰撞。

3. (g) Rotary Positional Embeddings (RoPE) Derivations (3 points)

这是作业第 8-9 页关于 RoPE 的数学推导部分。

i. RoPE 的复数形式与矩阵形式等价性 (2 points)

题目: 展示 Eq 3 (矩阵旋转形式) 中的元素可以由 Eq 4 (复数乘法形式) 得到。

- **解答思路**:
  - 复数形式: 考虑二维向量 (x_1, x_2) 对应的复数 $z = x_1 + ix_2$。

    RoPE 操作对应于将 z 乘以 $e^{it\theta} = \cos(t\theta) + i\sin(t\theta)$。

    计算乘积：

    $(x_1 + ix_2)(\cos(t\theta) + i\sin(t\theta))$

    $= (x_1\cos(t\theta) - x_2\sin(t\theta)) + i(x_1\sin(t\theta) + x_2\cos(t\theta))$

- 矩阵形式:

  题目中的 Eq 3 对二维向量的旋转矩阵为:

  $$R = \begin{pmatrix} \cos(t\theta) & -\sin(t\theta) \\ \sin(t\theta) & \cos(t\theta) \end{pmatrix}$$

  计算 $R \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$:

  $$\begin{pmatrix} x_1\cos(t\theta) - x_2\sin(t\theta) \\ x_1\sin(t\theta) + x_2\cos(t\theta) \end{pmatrix}$$

- 结论:

  可以看到矩阵运算结果的第一项对应复数运算的实部，第二项对应虚部。因此两者是等价的。

ii. 相对位置证明 (1 point)

题目: 证明 $\langle RoPE(z_1, t_1), RoPE(z_2, t_2) \rangle = \langle RoPE(z_1, t_1 - t_2), RoPE(z_2, 0) \rangle$。

- **解答思路**:
  - 利用复数内积的性质：对于二维实向量 u, v，其点积等于它们对应复数 z_u, z_v 的 $\mathrm{Re}(\bar{z}_u z_v)$（其中 $\bar{z}$ 是共轭复数）。
  - 设 z_1 在 t_1 处的 RoPE 变换为 $z_1' = z_1 e^{it_1\theta}$。
  - 设 z_2 在 t_2 处的 RoPE 变换为 $z_2' = z_2 e^{it_2\theta}$。
  - 计算点积:

    $\langle z_1', z_2' \rangle = \mathrm{Re}(\overline{z_1 e^{it_1\theta}} \cdot z_2 e^{it_2\theta})$

    $= \mathrm{Re}(\bar{z}_1 e^{-it_1\theta} \cdot z_2 e^{it_2\theta})$

    $= \mathrm{Re}(\bar{z}_1 z_2 e^{i(t_2 - t_1)\theta})$
  - 现在看等式右边 $\langle RoPE(z_1, t_1 - t_2), RoPE(z_2, 0) \rangle$:

    $z_a = z_1 e^{i(t_1 - t_2)\theta}$

    $z_b = z_2 e^{i(0)\theta} = z_2$

    $\langle z_a, z_b \rangle = \mathrm{Re}(\overline{z_1 e^{i(t_1 - t_2)\theta}} \cdot z_2)$

    $= \mathrm{Re}(\bar{z}_1 e^{-i(t_1 - t_2)\theta} \cdot z_2)$

    $= \mathrm{Re}(\bar{z}_1 z_2 e^{i(t_2 - t_1)\theta})$
  - **结论**: 左右两边相等，得证。这说明 RoPE 的注意力分数只依赖于相对位置 $(t_2 - t_1)$。

4. Considerations in pretrained knowledge (5 points)

这是基于代码实验后的反思部分。

(a) 预训练模型为何能达到 >10% 准确率 (1 point)

- 参考答案:

  预训练任务（在 Wikipedia 上训练）使模型接触并“记忆”了大量的世界知识（事实），这些知识被隐式存储在模型的权重中。当进行微调提问“某人出生在哪里”时，模型可以从权重中提取这些预先学到的知识，而不仅仅依赖微调数据集中提供的少数样本。非预训练模型从未见过这些名字和地点的关联，因此只能瞎猜。

(b) "幻觉" (Hallucinations) 的隐患 (2 points)

模型输出无法区分是“检索到的事实”还是“编造的”。

- **理由 1**: **安全性/误导风险**。
  - *例子*: 在医疗问答中，模型可能编造一个错误的药物副作用或治疗方案，用户无法分辨真假，可能导致健康受损。
- **理由 2**: **可解释性与调试困难**。
  - *例子*: 当模型回答错误时（例如说“奥巴马出生在法国”），开发者很难确定这是因为训练数据中有错误信息，还是模型未能正确回忆知识，亦或是模型完全在胡编乱造（幻觉）。

(c) 对未见过的名字的策略及隐患 (2 points)

- 策略:

  模型可能会利用名字的统计特征或子词（subword）关联进行猜测。例如，看到法语名字可能会猜“巴黎”，看到听起来像某些族裔的名字就猜对应的国家首都。或者直接输出训练集中出现频率最高的城市（如“伦敦”）。

- 隐患:

  这会导致偏见（Bias）和刻板印象。如果模型仅仅依据名字的种族特征来推断出生地，不仅准确率低，还会强化种族与地点的刻板联系，在实际应用中可能导致冒犯或歧视性的结果。