作业 4 语言模型

本次作业分成两部分：基于 RNNs 的神经机器翻译（*Neural Machine Translation with RNNs*）和 NMT 系统分析（*Analyzing NMT Systems*）。第一部分优先聚焦代码和实现，第二部分完全由需要分析和手写的问题组成。如果困在第一部分，你通常可以完成第二部分因为两个部分相互独立。请注意 NMT 系统比之前搭建的神经网络更复杂，在 GPU 上需要训练大约 2 小时。因此，我们强烈建议你尽早开始完成本次作业。最后，NMT 系统的概念和实现有一点技巧性，因此如果你在完成过程中遇到困难，请寻求 TAs 的帮助。

1. *Neural Machine Translation with RNNs*（45 分）

机器翻译的目标是将一个句子从源语言（例如中文普通话）转换到目标语言（例如英语）。本次作业，我们将实现一个带注意力的序列到序列（Seq2Seq）的网络以构建一个 NMT 系统。本部分，我们描述提出的 NMT 系统的训练过程，该系统使用了双向 LSTM 编码器和单向 LSTM 解码器。

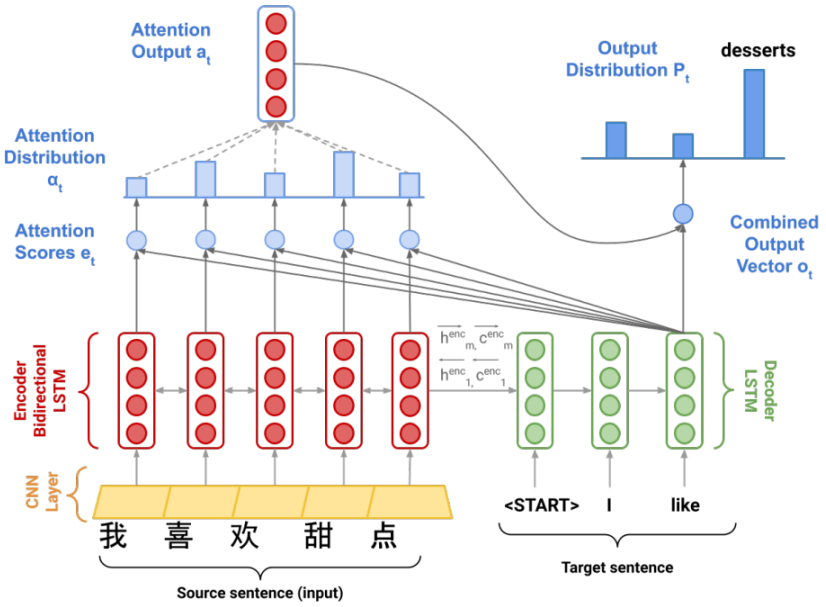

图 1：使用乘性注意力的 Seq2Seq 模型，在解码器的第三步展示。隐藏状态$\boldsymbol{h}_i^{enc}$和单元状态$\boldsymbol{c}_i^{enc}$在下文定义。

模型描述（训练过程）

给定源语言的一个句子，从嵌入矩阵（**embeddings matrix**）中查找字符或单词的嵌入，得到$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m (\boldsymbol{x}_i \in \mathbb{R}^{e\times 1})$，其中$m$是源句子的长度，$e$是嵌入大小。将嵌入输入卷积层（**convolutional layer**），并维持其形状。将卷积层输出作为双向编码器（**bidirectional encoder**）的输入，得到前向 LSTM 和反向 LSTM 的隐藏状态和单元状态。前向和反向的版本拼接到一起得到隐藏状态$\boldsymbol{h}_i^{enc}$和单元状态$\boldsymbol{c}_i^{enc}$：

$$\mathbf{h}_i^{\text{enc}} = [\overleftarrow{\mathbf{h}_i^{\text{enc}}}; \overrightarrow{\mathbf{h}_i^{\text{enc}}}] \text{ where } \mathbf{h}_i^{\text{enc}} \in \mathbb{R}^{2h\times 1}, \overleftarrow{\mathbf{h}_i^{\text{enc}}}, \overrightarrow{\mathbf{h}_i^{\text{enc}}} \in \mathbb{R}^{h\times 1} \qquad 1 \le i \le m \qquad (1)$$

$$\mathbf{c}_i^{\text{enc}} = [\overleftarrow{\mathbf{c}_i^{\text{enc}}}; \overrightarrow{\mathbf{c}_i^{\text{enc}}}] \text{ where } \mathbf{c}_i^{\text{enc}} \in \mathbb{R}^{2h\times 1}, \overleftarrow{\mathbf{c}_i^{\text{enc}}}, \overrightarrow{\mathbf{c}_i^{\text{enc}}} \in \mathbb{R}^{h\times 1} \qquad 1 \le i \le m \qquad (2)$$

然后将解码器（**decoders**）的第一个隐藏状态$\boldsymbol{h}_0^{dec}$和单元状态$\boldsymbol{c}_0^{dec}$初始化成编码器最后一个隐藏状态和单元状态的线性映射。

$$\mathbf{h}_0^{\text{dec}} = \mathbf{W}_h[\overleftarrow{\mathbf{h}_1^{\text{enc}}}; \overrightarrow{\mathbf{h}_m^{\text{enc}}}] \text{ where } \mathbf{h}_0^{\text{dec}} \in \mathbb{R}^{h\times 1}, \mathbf{W}_h \in \mathbb{R}^{h\times 2h} \quad (3)$$

$$\mathbf{c}_0^{\text{dec}} = \mathbf{W}_c[\overleftarrow{\mathbf{c}_1^{\text{enc}}}; \overrightarrow{\mathbf{c}_m^{\text{enc}}}] \text{ where } \mathbf{c}_0^{\text{dec}} \in \mathbb{R}^{h\times 1}, \mathbf{W}_c \in \mathbb{R}^{h\times 2h} \quad (4)$$

解码器初始化后，现在输入一个目标句子。在第 t 步，为第 t 个子词查找嵌入，$\boldsymbol{y}_t \in \mathbb{R}^{e\times 1}$。然后将$\boldsymbol{y}_t$和上一步的混合输出向量$\boldsymbol{o}_{t-1} \in \mathbb{R}^{h\times 1}$拼接（后面解释）以产生$\overline{\boldsymbol{y}}_t \in \mathbb{R}^{(e+h)\times 1}$。注意对第一个目标子词（例如开始词元）$\boldsymbol{o}_0$是一个零向量。然后将$\overline{\boldsymbol{y}}_t$作为解码器的输入。

$$\mathbf{h}_t^{\text{dec}}, \mathbf{c}_t^{\text{dec}} = \text{Decoder}(\overline{\mathbf{y}}_t, \mathbf{h}_{t-1}^{\text{dec}}, \mathbf{c}_{t-1}^{\text{dec}}) \text{ where } \mathbf{h}_t^{\text{dec}} \in \mathbb{R}^{h\times 1}, \mathbf{c}_t^{\text{dec}} \in \mathbb{R}^{h\times 1} \quad (5)$$

$$(6)$$

然后使用$\boldsymbol{h}_t^{dec}$在$\boldsymbol{h}_1^{enc},...,\boldsymbol{h}_m^{enc}$上计算乘性注意力：

$$\mathbf{e}_{t,i} = (\mathbf{h}_t^{\text{dec}})^T \mathbf{W}_{\text{attProj}} \mathbf{h}_i^{\text{enc}} \text{ where } \mathbf{e}_t \in \mathbb{R}^{m\times 1}, \mathbf{W}_{\text{attProj}} \in \mathbb{R}^{h\times 2h} \quad 1 \le i \le m \quad (7)$$

$$\alpha_t = \text{softmax}(\mathbf{e}_t) \text{ where } \alpha_t \in \mathbb{R}^{m\times 1} \quad (8)$$

$$\mathbf{a}_t = \sum_{i=1}^{m} \alpha_{t,i} \mathbf{h}_i^{\text{enc}} \text{ where } \mathbf{a}_t \in \mathbb{R}^{2h\times 1} \quad (9)$$

$e_{t,\,i}$是一个标量，是$\boldsymbol{e}_t \in R^{m\times 1}$的第 i 个元素，使用第 t 步解码器的隐藏状态$h_t^{dec} \in R^{h\times 1}$，注意力投影矩阵$W_{attProj} \in \mathbb{R}^{h\times 2h}$，第 i 步编码器的隐藏层状态$h_i^{enc} \in R^{2h\times 1}$这些变量计算得到。现在拼接注意力输出$\boldsymbol{a}_t$和解码器隐藏状态$\boldsymbol{h}_t^{dec}$，传递给一个线性层，tanh 和 dropout 以得到混合输出向量$\boldsymbol{o}_t$。

$$\mathbf{u}_t = [\mathbf{a}_t; \mathbf{h}_t^{\text{dec}}] \text{ where } \mathbf{u}_t \in \mathbb{R}^{3h\times 1} \quad (10)$$

$$\mathbf{v}_t = \mathbf{W}_u \mathbf{u}_t \text{ where } \mathbf{v}_t \in \mathbb{R}^{h\times 1}, \mathbf{W}_u \in \mathbb{R}^{h\times 3h} \quad (11)$$

$$\mathbf{o}_t = \text{dropout}(\tanh(\mathbf{v}_t)) \text{ where } \mathbf{o}_t \in \mathbb{R}^{h\times 1} \quad (12)$$

随后在第 t 步计算所有子词的概率分布$\boldsymbol{P}_t$：

$$\mathbf{P}_t = \text{softmax}(\mathbf{W}_{\text{vocab}} \mathbf{o}_t) \text{ where } \mathbf{P}_t \in \mathbb{R}^{V_t\times 1}, \mathbf{W}_{\text{vocab}} \in \mathbb{R}^{V_t\times h} \quad (13)$$

V_t是目标词汇表的大小。最后为了训练神经网络，可以计算$\boldsymbol{P}_t$和$\boldsymbol{g}_t$之间的交叉熵损失，其中$\boldsymbol{g}_t$是第 t 步目标子词的独热向量：

$$J_t(\theta) = \text{CrossEntropy}(\mathbf{P}_t, \mathbf{g}_t) \quad (14)$$

θ 是模型的所有参数，$J_t(\theta)$ 是第 t 步解码器的损失。理解上述过程后，可以开始实现中文普通话到英语的 NMT 系统。

虚拟机设置（略）

本地机器运行模型代码，请在命令行输入以下指令创建虚拟环境：
conda env create --file local_env.yml

实现和手写问题

(a) 批量内句子长度一致化，一致化后的长度为批量内句子的最大长度。在 utils.py 内实现 pad_sents 函数，该函数生成填充后的句子；(2 分)

(b) 在 model_embeddings.py 中实现__init__函数，该函数初始化必要的起源和目标的嵌入；(3 分)

(c) 在 nmt_model.py 中实现__init__函数，初始化 NMT 系统必要的模型层（LSTM, CNN, 映射和 dropout）；(4 分)

(d) 在 nmt_model.py 中实现编码器函数。该函数将填充后的源句子转换成张量 X，生成 $\boldsymbol{h}_1^{enc},...,\boldsymbol{h}_m^{enc}$，并计算解码器的初始隐藏状态 $\boldsymbol{h}_0^{dec}$ 和单元状态 $\boldsymbol{c}_0^{dec}$。可以运行以下命令执行非完全正确性检测：(8 分)

python sanity_check.py 1d

```
(local_nmt) PS F:\CV\NLP\coding\a4\student> python .\sanity_check.py 1d
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\69134\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Running Sanity Check for Question 1d: Encode
------------------------------------------------------------------------------
D:\miniconda3\envs\local_nmt\lib\site-packages\torch\nn\modules\conv.py:306: UserWarning: Using padding='same' with even kernel
y of the input be created (Triggered internally at C:\cb\pytorch_1000000000000\work\aten\src\ATen\native\Convolution.cpp:1009.)
  return F.conv1d(input, weight, bias, self.stride,
enc_hiddens Sanity Checks Passed!
dec_init_state[0] Sanity Checks Passed!
dec_init_state[1] Sanity Checks Passed!
------------------------------------------------------------------------------
All Sanity Checks Passed for Question 1d: Encode!
------------------------------------------------------------------------------
```

(e) 在 nmt_model.py 中实现解码器函数。该函数构建 $\bar{\boldsymbol{y}}$ 并对输入在每一个时间步执行步进函数。可以运行以下命令执行不完全正确性检测：(8 分)

python sanity_check.py 1e

```
(local_nmt) PS F:\CV\NLP\coding\a4\student> python .\sanity_check.py 1e
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\69134\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
--------------------------------------------------------------------------------
Running Sanity Check for Question 1e: Decode
--------------------------------------------------------------------------------
torch.Size([23, 5, 2])
combined_outputs Sanity Checks Passed!
--------------------------------------------------------------------------------
All Sanity Checks Passed for Question 1e: Decode!
--------------------------------------------------------------------------------
```

(f) 在 nmt_model.py 中实现步进函数。该函数在每一个时间步使用解码器的 LSTM 单元，计算出目标子词的编码$\boldsymbol{h}_t^{dec}$，注意力分数$\boldsymbol{e}_t$，注意力分布α_t，注意力输出$\boldsymbol{a}_t$以及最后结合后的输出$\boldsymbol{o}_t$。可以运行以下命令执行不完全正确性检测：(10 分)

python sanity_check.py 1f

```
(local_nmt) PS F:\CV\NLP\coding\a4\student> python .\sanity_check.py 1f
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\69134\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
--------------------------------------------------------------------------------
Running Sanity Check for Question 1f: Step
--------------------------------------------------------------------------------
F:\CV\NLP\coding\a4\student\nmt_model.py:375: UserWarning: Implicit dimension choice for softmax has been deprecated. Change the call to include dim=X as an argument.
  alpha_t = nn.functional.softmax(e_t)
dec_state[0] Sanity Checks Passed!
dec_state[1] Sanity Checks Passed!
combined_output  Sanity Checks Passed!
e_t Sanity Checks Passed!
--------------------------------------------------------------------------------
All Sanity Checks Passed for Question 1f: Step!
--------------------------------------------------------------------------------
```

(g) (手写) nmt_model.py 中的 generate_sent_masks 函数生成一个名为 enc_masks 的张量。该张量的形状为（批量大小，最大源句子长度），在输入的”<pad>”词元的对应位置为 1，非”<pad>”词元的对应位置为 0。查看 step()函数注意力计算过程中该掩码是如何使用的。（311-312 行）。首先解释（大约三句话）掩码在整个注意力计算中的效果。然后解释为什么有必要采用这种掩码方式（一到两句话）。

My answer:
将’<pad>’词元所在位置的初始权重设置成负无穷，经过 softmax 后的权重趋于 0，从而基本消除’<pad>’嵌入对注意力输出的影响；’<pad>’词元缺少语义信息。

接下来执行以下命令在本地训练模型：
sh run.sh train_local
(Windows) run.bat train_local

可以运行以下命令在更少数据上更快地训练：
sh run.sh train_debug
(Windows) run.bat debug

为便于监控和调试，启动代码在训练过程中使用 TensorBoard 记录损失和困惑度。TensorBoard 提供了日志记录和训练信息可视化工具。可以在 conda 环境中运行下列命令开启 TensorBoard：

tensorboard –logdir=runs

正常情况下应该在初始迭代中看到明显的损失下降。一旦确定代码不崩溃（直至第 10 或第 20 次迭代），切换到虚拟环境。

接下来，在虚拟环境中安装必要的包：

pip install -r gpu_requirements.txt

最后，创建一个新的 tmux session。具体而言，运行下列指令创建一个名为 nmt 的 tmux session：

tmux new -s nmt

一旦虚拟环境配置完成，且位于 tmux session 中，执行

sh run.sh train

(Windows) run.bat train

一旦知晓代码正确运行，可以从 session 中分离并关闭 ssh 服务器连接。为了从 sesssion 中分离，运行：

tmux detach

可以通过 ssh 到服务器并关联 tmux session 以返回到训练的模型：

tmux a -t nmt

(h) （手写）一旦模型完成训练（在虚拟环境下大约需要 2h），执行以下命令测试模型：

sh run.sh test

(Windows) run.bat test

请汇报模型在语料库上的 BLEU 分数。该分数应大于 18。

```
(base) root@autodl-container-a04e11b652-2609ac97:~/lee/a4/student# sh run.sh test
load test source sentences from [./zh_en_data/test.zh]
load test target sentences from [./zh_en_data/test.en]
load model from model.bin
Decoding:   0%|                                                                                                | 0/1001 [00:00<?, ?it/s]
/root/miniconda3/lib/python3.8/site-packages/torch/nn/modules/conv.py:298: UserWarning: Using padding='same' with even kernel lengths and odd dilation may req
uire a zero-padded copy of the input be created (Triggered internally at  ../aten/src/ATen/native/Convolution.cpp:744.)
  return F.conv1d(input, weight, bias, self.stride,
Decoding: 100%|████████████████████████████████████████████████████████████████████████████████████████████| 1001/1001 [00:32<00:00, 30.98it/s]
Corpus BLEU: 19.60398618099794
```

(i) （手写）在课堂上学习了点积注意力，乘性注意力和加性注意力。作为回顾，点积注意力是 $\boldsymbol{e}_{t,i} = \boldsymbol{s}_t^T \boldsymbol{h}_i$，乘性注意力是 $\boldsymbol{e}_{t,i} = \boldsymbol{s}_t^T \boldsymbol{W} \boldsymbol{h}_i$，加性注意力是 $\boldsymbol{e}_{t,i} = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}_t)$。

1. (2 分) 解释点积注意力相比于乘性注意力的一个优点和缺点。
2. (2 分) 解释加性注意力相比于乘性注意力的一个优点和缺点。

My answer:

1. 优点：没有需要学习的参数，计算复杂度低；缺点：要求输入词嵌入和隐藏向量的长度一致，参数灵活性降低；
2. 优点：融入源句子不同位置的关联信息；缺点：计算复杂度高。

2. 分析 NMT 系统（25 分）

(a) 查阅 src.vocab 中关于源语言词汇表短语和单词的一些例子。当将中文普通话句子编码

成词汇表中的“块”时，分词器将句子映射成词汇表中的一系列项，每一项由一个或多个字符组成（多亏了 sentencepiece 分词器，使得原始文本即使没有空格也能完成分割）。给定这一信息，那么通过在嵌入层后和将词嵌入传入双向编码器前添加一个一维卷积层是如何改善 NMT 系统的？提示：每一个中文字符要么是一整个词，要么是词中的一个语素。以“电”，“脑”，“电脑”各自的意义为例。字符“电”（电能）和“脑”（大脑）组合成的短语“电脑”表示计算机。（3 分）

My answer:
一维卷积层能够提取序列的特征，捕获单语素及语素组合的语义信息，弥补分词器的分词失误，获得更符合上下文的词嵌入。

(b) 这里提供一系列在 NMT 模型输出中找到的错误。对每一个包含参考英文翻译和 NMT 英文翻译的例子，请：
1. 找出 NMT 翻译中的错误
2. 提供模型出错的可能原因（特定的语言结构或特定的模型限制）
3. 描述改进 NMT 系统以改正观察到的错误的一种可能途径。对于一个错误可能有不止一种途径。例如，途径可能是调整隐藏层大小或改变注意力机制。

以下是需要按照上述描述分析的翻译。只需分析每个句子带下划线部分的错误。剩余部分确保不需要了解普通话就能回答这些问题。只需知道英文即可。但是，如果需要关于源句子的额外信息，请访问 https://www.archchinese.com/chinese_english_dictionary.html 以查询词语。尽情搜索训练数据文件以对特定字符的出现频率产生更进一步的理解。

1. （2 分）源句子：贼人其后被警方拘捕及被判处盗窃罪名成立。
参考翻译：the culprits were subsequently arrested and convicted.
NMT 翻译：the culprit was subsequently arrested and sentenced to theft.

My answer:
错误：名词复数误用成单数
可能原因：“贼人”对应的英文单复数不明确
可能的改进途径：增加训练数据中单复数用法示例

2. （2 分）源句子：几乎已经没有地方容纳这些人，资源已经用尽。
参考翻译：there is almost no space to accommodate these people, and resources have run out.
NMT 翻译：the resources have been exhausted and resources have been exhausted.

My answer:
错误：主被动错误
可能原因：未注意到源句子使用主动结构
可能的改进途径：使用加性注意力

3. （2 分）源句子: 当局已经宣布今天是国殇日。
参考翻译：authorities have announced a national mourning today.

NMT 翻译：the administration has announced today's day.

My answer:
错误：翻译缺失
可能原因："国殇日"属于生僻词
可能的改进途径：增加训练数据中生僻词样本

4. （2 分）源句子: 俗语有云:"唔做唔错"。
参考翻译："act not, err not", so a saying goes.
NMT 翻译：as the saying goes, " it's not wrong. "

My answer:
错误：语义错误
可能原因：模型上下文理解能力不足
可能的改进途径：增加隐藏层大小；使用双向 RNN

(c) BLEU 分数是自动评估 NMT 系统最常用的指标。BLEU 通常在整个测试集上计算得到，但在此处仅考虑定义在单个例子上的 BLEU。假定有一个源句子 s，k 个参考翻译 r1, … ,rk 组成的集合，和一个候选翻译 c。为了计算 c 的 BLEU 分数，首先计算修改后的 n-gram 准确率 p_n，对 $n=1,2,3,4$，n 是 n-gram 中的 n：

$$p_n = \frac{\sum_{\text{ngram}\in\mathbf{c}} \min\left(\max_{i=1,\dots,k} \text{Count}_{\mathbf{r}_i}(\text{ngram}),\ \text{Count}_{\mathbf{c}}(\text{ngram})\right)}{\sum_{\text{ngram}\in\mathbf{c}} \text{Count}_{\mathbf{c}}(\text{ngram})} \tag{15}$$

对于每一个出现在候选翻译 c 中的 n-gram，计算该 n-gram 在任意一个参考翻译中出现的最大次数，其上限是该 n-gram 在 c 中出现的次数（分子部分）。将分子除以 n-gram 在 c 中出现的次数（分母部分）。

接下来，计算简洁性惩罚 BP。记 len(c)为 c 的长度，len(r)为最接近 len(c)的参考翻译的长度（在两个距离相等的参考翻译长度的情况下，选择较短的那个参考翻译的长度作为标准）。

$$BP = \begin{cases} 1 & \text{if } len(c) \geq len(r) \\ \exp\left(1 - \frac{len(r)}{len(c)}\right) & \text{otherwise} \end{cases} \tag{16}$$

最后，候选翻译 c 关于 r1, r2, …, rk 的 BLEU 分数为：

$$BLEU = BP \times \exp\left(\sum_{n=1}^{4} \lambda_n \log p_n\right) \tag{17}$$

其中 λ1, λ2, λ3, λ4 是和为 1 的权重，log 是自然对数。

1. （5 分）考虑这个例子：
源句子 s：**需要有充足和可预测的资源**。
参考翻译 r1：resources have to be sufficient and they have to be predictable

参考翻译 r2：adequate and predictable resources are required
NMT 翻译 c1：there is a need for adequate and predictable resources
NMT 翻译 c2：resources be sufficient and predictable to

请计算 c1 和 c2 的 BLEU 分数。λ1= λ2=0.5，λ3=λ4=0（这意味着忽略 3-grams 和 4-grams，从而无需计算 p3 和 p4），当计算 BLEU 分数时，提供计算过程（例如，计算出的 p1, p2, len(c), len(r)和 BP）。注意 BLEU 可以表示在 0~1 或 0~100。代码使用了 0~100 范围，然而在本问题中使用 0~1 范围。答案保留 3 位小数。

My answer:

$$p_n = \frac{\sum_{\text{ngram}\in\mathbf{c}} \min\left(\max_{i=1,\ldots,k} \text{Count}_{\mathbf{r}_i}(\text{ngram}),\ \text{Count}_{\mathbf{c}}(\text{ngram})\right)}{\sum_{\text{ngram}\in\mathbf{c}} \text{Count}_{\mathbf{c}}(\text{ngram})} \tag{15}$$

1-gram：

	c1	r1	r2
there	1	0	0
is	1	0	0
a	1	0	0
need	1	0	0
for	1	0	0
adequate	1	0	1
and	1	1	1
predictable	1	1	1
resources	1	1	1

	c1	r1	r2
resources	1	1	1
be	1	2	0
sufficient	1	1	0
and	1	1	1
predictable	1	1	1
to	1	2	0

2-gram：

	c1	r1	r2
there is	1	0	0
is a	1	0	0
a need	1	0	0
need for	1	0	0
for adequate	1	0	0
adequate and	1	0	1
and predictable	1	0	1
predictable resources	1	0	1

	c1	r1	r2
resources be	1	0	0
be sufficient	1	1	0
sufficient and	1	1	0
and predictable	1	0	1
predictable to	1	0	0

对 c1：

$p_1 = \frac{4}{9},\ p_2 = \frac{3}{8}$

对 c2：

$p_1 = \frac{6}{6} = 1,\ p_2 = \frac{3}{5}$

len(r)=min(len(r1), len(r2)) = min(11, 6) = 6
len(c1) = 9, len(c2) = 6
BP1 = BP2 = 1

$$BLEU = BP \times \exp\left(\sum_{n=1}^{4} \lambda_n \log p_n\right) \tag{17}$$

c_1: $BLEU = 1 \times \exp\left(0.5 \times \log\left(\frac{4}{9}\right) + 0.5 \times \log\left(\frac{3}{8}\right)\right) \approx 0.678$

c_2: $BLEU = 1 \times \exp\left(0.5 \times \log(1) + 0.5 \times \log\left(\frac{3}{5}\right)\right) \approx 0.895$

根据 BLEU 分数，两个 NMT 翻译哪个更好？是否认为确实是更好的翻译？

My answer:
c2 优于 c1；c2 存在语法错误且语句不完整，实际翻译效果劣于 c1

2. （5 分）假定硬盘损坏，参考翻译 r1 丢失。请重新计算 c1 和 c2 的 BLEU 分数，这一次仅关于 r2。现在哪一个参考翻译获得更高的 BLEU 分数？是否认为确实是更好的翻译？

My answer:
对 c1：

$p_1 = \frac{4}{9},\ p_2 = \frac{3}{8}$

对 c2：

$p_1 = \frac{3}{6} = \frac{1}{2},\ p_2 = \frac{1}{5}$

len(r)=len(r2) = 6, len(c1) = 9, len(c2) = 6
BP1 = BP2 = 1

c_1': $BLEU = 1 \times \exp\left(0.5 \times \log\left(\frac{4}{9}\right) + 0.5 \times \log\left(\frac{3}{8}\right)\right) \approx 0.678$

c_2': $BLEU = 1 \times \exp\left(0.5 \times \log\left(\frac{1}{2}\right) + 0.5 \times \log\left(\frac{1}{5}\right)\right) \approx 0.606$

c1 获得更高的 BLEU 分数；c1 无语法错误，语句完整且符合原意，翻译效果更好。

3. （2 分）由于数据有限，NMT 系统经常只能在单一参考翻译上做评估。请用几句话解释为什么这可能造成问题。在解释中，讨论相比于单一参考翻译，多个参考翻译下 BLEU 评价指标是如何评估 NMT 的翻译质量的？

My answer:
单一参考翻译评估具有一定的主观性，且违背翻译自身的多样性；多参考翻译下的 BLEU 通过统计候选翻译的 n-gram 在多个参考翻译中的最大出现频率并与候选翻译中的出现频率比较，反映候选翻译和参考翻译的整体匹配度，相对更客观、准确、全面。

4. （2 分）列举 BLEU 作为机器翻译的评价指标，相比人类评估的两个优点和两个缺点。

My answer:
优点：自动评估；可重复性强
缺点：评价相对粗略，不适合诗歌等复杂翻译任务；依赖参考翻译，评估结果容易受参考翻译质量的影响