Task6 miniWatson-Report1

23 组：何青蓉 徐轶琦

一 项目介绍

Watson 是 IBM 自然语言处理领域的智能问答系统。本小组计划基于 Seq2Seq 模型，利用淘宝客服对话数据，实现一个 miniWatson 聊天机器人。

人机对话模型是人工智能研究领域的热门话题，聊天机器人有着广泛的应用，例如客户服务应用和在线帮助。传统的聊天机器人主要是基于检索式的模型——提前建立一个问答库，根据用户输入的问题检索相应的答案，这样的模型输出的答案是固定不变的，且问答系统无法回答语料库中未出现的问题。随着深度神经网络研究的发展，基于深度学习的生成式模型让问答系统能够产生更加灵活的答案，回答语料库中未出现的问题，使人机交互更加灵活。

本项目实现 miniWatson 采用的是生成式模型——基于 RNN 的 Seq2Seq 序列到序列模型。Seq2Seq 属于 encoder-decoder 结构的一种，利用两个 RNN，一个 RNN 作为 encoder，负责将输入的序列压缩成指定长度的向量，这个向量表征了序列的语义；另一个 RNN 作为 decoder，负责根据语义向量生成指定的序列。

二 实验方法

本项目的实现分为以下几个阶段：数据预处理、定义模型、训练模型、效果测试。

第一阶段完成了数据预处理部分，并通过阅读文献和相关代码对模型定义与训练有了初步了解。下面介绍数据预处理的步骤方法。

1、创建句对 Pairs

将中文语料处理为问题（query）和答案（response）组成的句对（pairs）。

2、创建词典 Voc

输入是一个句对，每个句子都是词的序列，但是机器学习只能处理数值，因此我们需

要建立词到数字 ID 的映射。

（1）预处理

- 去除停用词；
- 去除长度大于 MAX_LENGTH 的句子。

（2）词典类 class Voc

创建一个类 class Voc，保存词到 ID 的映射，同时反向保存从 ID 到词的映射，记录每个词出现的次数以及总共出现的词的个数，并实现以下功能：

- 增加词：addWord；
- 增加句子：addSentence；
- 去除低频词：trim；
- 去除包含低频词的句子。

3、创建 Tensor

为了加快训练速度，需要一次处理一组共 batch 条数据。

（1）生成一组共 batch 个固定长度为 max_length 的句子的技巧

- Padding：把短的句子补上 0，使得输入的一组共 batch 个句子（即词 ID 序列）长度均为 max_length。

（2）处理过程

从所有 pairs 中随机选择 batch 个句子（问和答分开处理）作为一组，把句子的词变成 ID，padding 处理为固定长度的 list，得到 batch 个长度为 max_length 的 list。为了使 t 时刻 batch 个数据在内存中是连续的，需要将其转置为(max_length,batch)（图 1）。

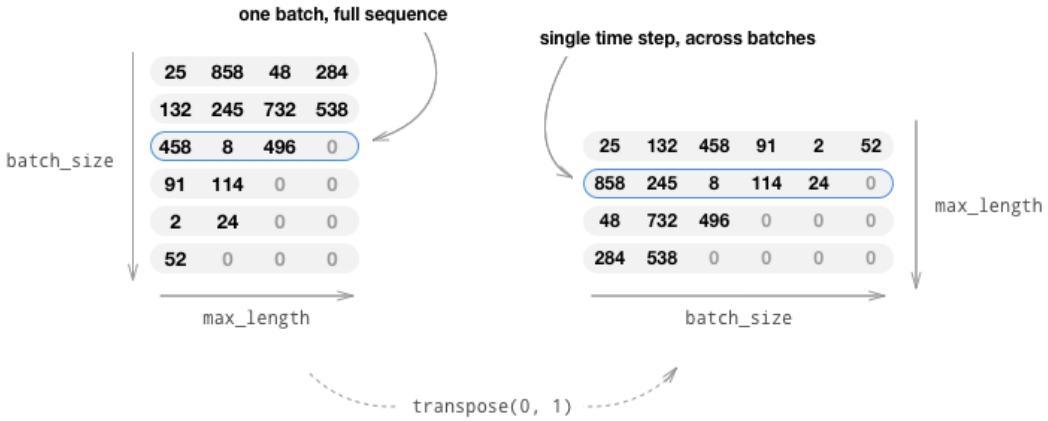

图 1 (batch,max_length)转置为(max_length,batch)

①问句处理

输入：随机选择的 batch 个 pairs 中的问句。

输出：

• input_variable：大小为(max_length,batch)的 tensor，表示 max_length 个长度为 batch 的 list；

• lengths：大小为 batch 的 list，表示 batch 个句子每个句子的实际长度，即 padding 前的句子长度。

②答句处理

输入：随机选择的 batch 个 pairs 中的回答。

输出：

• target_variable：大小为(max_length,batch)的 tensor，表示 max_length 个长度为 batch 的 list；

• mask：大小为(max_length,batch)的 tensor，0-1 变量记录每个位置是否为 padding；

• max_target_len：batch 个答句中最长句子的长度（依据此长度对其他句子 padding）。

三 第一阶段成果展示

1、分词后的 pair<question,answer>：

```
['538405926243 买 二份 有没有 少点 呀', '亲亲 真的 不好意思 我们 已经 是 优惠价 了 呢 小本生意 请亲 谅解']
['那 就 等 你们 处理 喽', '好 的 亲退 了']
['那 我 不 喜欢', '颜色 的话 一般 茶刀 茶针 和 二合一 的话 都 是 红木 檀 和 黑木 檀 哦']
['不是 免 运费', '本店 茶具 订单 满 99 包邮除 宁夏 青海 内蒙古 海南 新疆 西藏 满 39 包邮']
['好吃 吗', '好吃 的']
['为什么 迟迟 不 给 我 发货', '实在 抱歉 呢 由于 订单 量 大 您 的 订单 本来 安排 今天 发货 的 呢']
['对 谢谢', '小店 尽快 给 您 发出 哦']
['3 组送 什么', '拍 2 组送 2 包 湿巾 3 组 也 是 2 包']
['那 我 马上 拍', '记得 勾选 优惠券 哦']
['每 一样 都 要点 要 二百个', '单个 的话 价格 都 是 最低 了 哦 都 是 亏本 促销 的 只是 为了 前期 冲 销量 的 5488816028685471076129665473934863645472597857 39 这款 单买 可以 再 便宜 鲜肉 蜜枣 豆沙 最低 53 蛋黄 肉 粽 最低 63 元 哦']
['百世', '好 的']
['⊙。⊙ 哦 好 的', '优惠券 有效期 至 8 月 31 日 谢谢 客官 支持 小店 哦']
['目前 我 只有 这些', '亲 只有 这些 齐全 的 才能 开']
['可是 这个 没有 到 啊', '应该 这 两天 会到 的 这个 会 联系 下 快递']
['单 怎么 下', '您 提交 以后 哪里 可以 修改 数量 的']
['吃 起来 和 新鲜 的 有 很大 差别 嘛 保鲜', '吃 起来 也 很 松脆']
['杭州 桐庐 几天 到', '一般 1 - 2 天']
['别 给 我 像 发货 一样 的 慢', '不会 的 呢']
['我 买 的 东西 发货 了 没 怎么 看不见 物流', '亲亲 实在 抱歉 仓库 那边 说 配货 的 时候 椒盐 味纸 皮 核桃 缺货 了 呢 新 的 一批 明天 才 到货 哦 到货 后 第一 时间 给 您 打包 哦 亲亲 不要 着急 哦']
['嗯 呢 要 不到 不了 啊', '嗯 嗯 已经 给 您 修改 好 了 哦']
```

2、准备传入 Seq2Seq 模型的 tensor

随机选择 batch=5 个 pair：

- 问句的 tensor：

```
input_variable: tensor([[ 204,  276,  255,   43,   43],
        [ 593,   44,  460,    6,    2],
        [ 255,  185,   28,  665,    0],
        [ 276,   30, 1159,  519,    0],
        [ 185,  242, 7828, 2283,    0],
        [ 150,   81,   28,  144,    0],
        [ 597,  270,  908,    2,    0],
        [  30, 1131,  356,    0,    0],
        [  22,  986, 3188,    0,    0],
        [ 142,  347,    2,    0,    0],
        [ 111,    2,    0,    0,    0],
        [ 215,    0,    0,    0,    0],
        [ 490,    0,    0,    0,    0],
        [ 276,    0,    0,    0,    0],
        [  16,    0,    0,    0,    0],
        [   2,    0,    0,    0,    0]])
lengths: tensor([16, 11, 10,  7,  2])
```

- 回答的 tensor：

```
target_variable: tensor([[ 204,  369,   27, 3504,  204],
        [ 204,   16,   28,  366,  204],
        [  27,   17,  136,   17,  136],
        [  28,  593,    2,    2,    2],
        [   2,   72,    0,    0,    0],
        [   0,  436,    0,    0,    0],
        [   0,   16,    0,    0,    0],
        [   0,  180,    0,    0,    0],
        [   0,  998,    0,    0,    0],
        [   0,   65,    0,    0,    0],
        [   0,   72,    0,    0,    0],
        [   0,  346,    0,    0,    0],
        [   0,   99,    0,    0,    0],
        [   0,   72,    0,    0,    0],
        [   0,  438,    0,    0,    0],
        [   0,   72,    0,    0,    0],
        [   0,  358,    0,    0,    0],
        [   0, 2878,    0,    0,    0],
        [   0,  142,    0,    0,    0],
        [   0,  874,    0,    0,    0],
        [   0,  150,    0,    0,    0],
        [   0,   12,    0,    0,    0],
        [   0,  339,    0,    0,    0],
        [   0,   72,    0,    0,    0],
        [   0,  360,    0,    0,    0],
        [   0,  110,    0,    0,    0],
        [   0,   72,    0,    0,    0],
        [   0,  438,    0,    0,    0],
        [   0,  110,    0,    0,    0],
        [   0,   62,    0,    0,    0],
        [   0,    2,    0,    0,    0]])
mask: tensor([[ True,  True,  True,  True,  True],
        [ True,  True,  True,  True,  True],
        [ True,  True,  True,  True,  True],
        [ True,  True,  True,  True,  True],
        [ True,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False],
        [False,  True, False, False, False]])
max_target_len: 31
```

四 后续任务

（一）定义模型

1、Encoder

使用多层的 Gated Recurrent Unit（GRU）作为 Encoder，遍历每个词（Token），每个时刻的输入是上一个时刻的隐状态和输入，产生一个输出和新的隐状态，作为下一个时刻的输入隐状态。把最后一个时刻的隐状态作为 Decoder 的初始隐状态。最终返回所有时刻的输出和最后时刻的隐状态。

2、Decoder

每个时刻的输入是上一个时刻的隐状态和上一个时刻的输出。初始隐状态是 Encoder 最后时刻的隐状态。利用 RNN 计算新的隐状态和输出的第一个词，接着用新的新状态和第一个词计算第二个词，以此类推直到遇到结束符。

利用 Attention 机制，在 Decoder 进行 t 时刻计算的时候，除了 t-1 时刻的隐状态、当前时刻的输入，还会参考 Encoder 所有时刻的输入。

（二）训练模型

- 一个 batch 传入 Encoder；
- Encoder 最后时刻的隐状态作为 Decoder 的初始隐状态，Decoder 每次处理一个时刻的 forward 计算，把上个时刻“正确”的词作为当前输入（teacher forcing）或者把上一个时刻的输出作为当前时刻的输入；
- 计算 loss：交叉熵；
- 反向计算梯度；
- 梯度裁剪（gradient clipping）；
- 更新模型参数。

（三）效果测试

与 miniWatson 聊天：用 Decoder 生成一个响应。

- 贪心解码（Greedy decoding）算法；
- Beam-Search 算法。