自然语言处理

实验报告

姓　　名　王丹丹

学　　号　2201870

专　　业　计算机科学与技术

班　　级　计硕 2204 班

2023 年 1 月 10 日

硬件环境（CPU/GPU）：

CPU：Intel(R) Core(TM) i5-9300H　GeForce GTX 1650 显卡：4G

操作系统：

Windows 10

采用的框架、工具、语言：

框架、工具：PyTorch　Python 3.7

语言：Python

任务描述/问题定义：

现实生活中的许多人间都设置了客服，可以对软件相关应用进行咨询，而全部使用人工客服，会产生大量的人力成本。基于此在人工客服之前，我们常常会见到聊天机器人与我们对话，为我们提供一些解决方案。准备相应的语料库，基于模型 Seq2seq 实现实现聊天机器人，进行简单的聊天。

采用的模型：

Seq2seq 模型：

主要流程：

1、处理语料、文本序列化

2、使用对语料处理后的序列，准备数据集和 Dataloader

3、构造模型:完成编码器、解码器，完成 Seq2seq 模型

4、完成训练逻辑，开始训练

5、评估模型

相关参数设置：

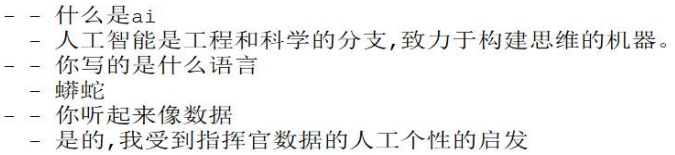
lr = 0.001

teacher_forcing_ratio = 0.7

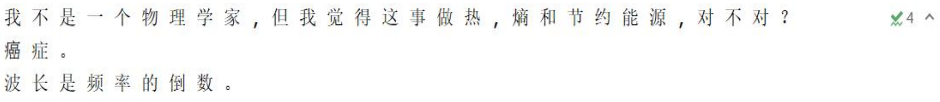
语料来源：

https://github.com/gunthercox/chatterbot-corpus/tree/master/chatterbot_corpus/data/chinese

原始语料库部分截图：存放在 chinses.txt.



经过分词处理后的语料库部分截图：存放在 input_by_word.txt 和 target_by_word.txt.



一、实验内容

动手完成一个简单任务，比如机器翻译等等，并尝试理解模型的原理，通过已知方法改善模型性能。

二、框架、工具等的安装及配置过程

（1）查看电脑可以支持的 cuda 的最高版本，使用命令：nvidia-smi，如图所示，

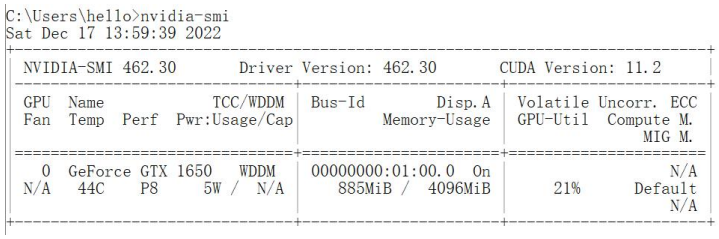


图 1 应用 nvidia-smi 查看 cuda 版本

（2）创建一个虚拟环境，命令：conda create -n pytorch python=3.7，如图 2 所示，

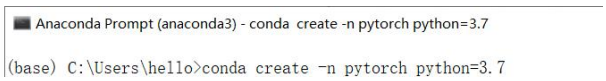


图 2 应用 conda create -n pytorch python=3.7 创建虚拟环境

（3）进入该环境，使用命令：conda activate pytorch1，如图所示，

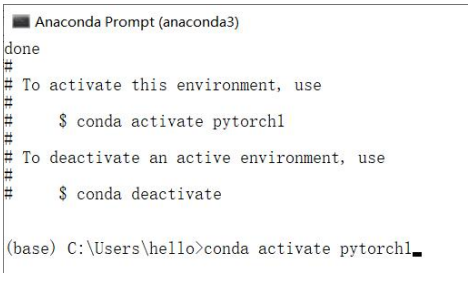


图 3 应用 conda activate pytorch1 进入虚拟环境

（4）在 Pytorch 官网（https://pytorch.org）选择合适的版本 Pytorch 进行安装，以及安装的部分过程：

```
(pytorch1) C:\Users\hello>conda install pytorch==1.0.0 torchvision==0.2.1 cuda100 -c pytorch
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Collecting package metadata (repodata.json): done
Solving environment: done
```

图 4 安装 Pytorch，具体命令需官网查询

```
## Package Plan ##

  environment location: D:\Program Files (x86)\anaconda3\envs\pytorch1

  added / updated specs:
    - cuda100
    - pytorch==1.0.0
    - torchvision==0.2.1


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    cuda100-1.0                |                0           2 KB  pytorch
    pytorch-1.0.0              | py3.7_cuda100_cudnn7_1       654.7 MB  pytorch
    torchvision-0.2.1          |           py_2              37 KB  pytorch
    ------------------------------------------------------------
                                           Total:       654.8 MB

The following NEW packages will be INSTALLED:
```

图 5 安装 Pytorch 的部分过程

（5）安装相关的包和库，命令：pip install 包/库名，部分截图如下

Files (x86) › anaconda3 › envs › pytorch1 › Lib › site-packages

名称	修改日期	类型
__pycache__	2022/12/17 19:35	文件夹
_distutils_hack	2022/12/17 16:41	文件夹
absl	2022/12/17 19:35	文件夹
absl_py-1.3.0.dist-info	2022/12/17 19:35	文件夹
astor	2022/12/17 19:35	文件夹
astor-0.8.1.dist-info	2022/12/17 19:35	文件夹
boto3	2022/12/17 19:08	文件夹
boto3-1.26.31.dist-info	2022/12/17 19:08	文件夹
botocore	2022/12/17 19:08	文件夹
botocore-1.29.32.dist-info	2022/12/17 19:08	文件夹
brotli	2022/12/17 19:08	文件夹

图 6 安装的部分依赖包或者库

二、模型介绍

1、模型简介：

Seq2Seq 模型主要是由一个编码器和一个解码器组成，编码器接收到输入，对输入进行理解，得到一个输出：语义向量，解码器对输入的语义向量进行相应处理，得到输出序列。主要的优势就是可以在输入与输出不等长的情况下有良好的结果。如图所示为 Seq2Seq 模型的结构：

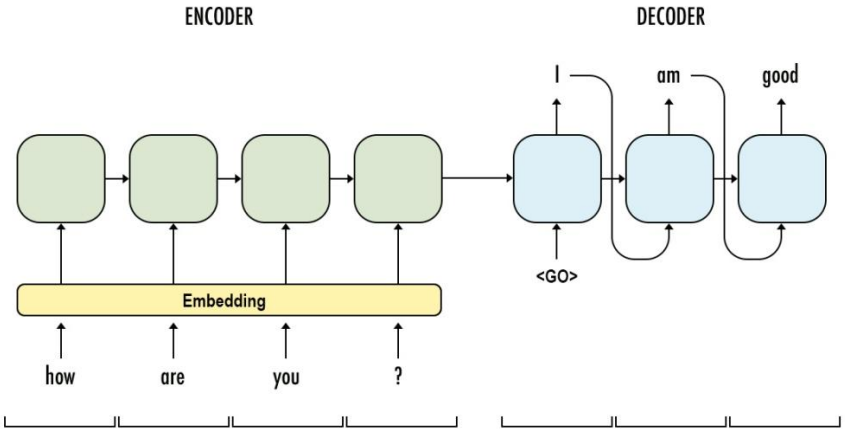


图 7 Seq2Seq 模型的结构

Seq2Seq 模型的结构主要有三种，它们之间的主要不同是在于解码器的不同。第一种的结构图 1 如下所示，此种结构将编码器得到的输出序列当做初始隐藏层输入，后续只是接受上一个隐藏层状态。

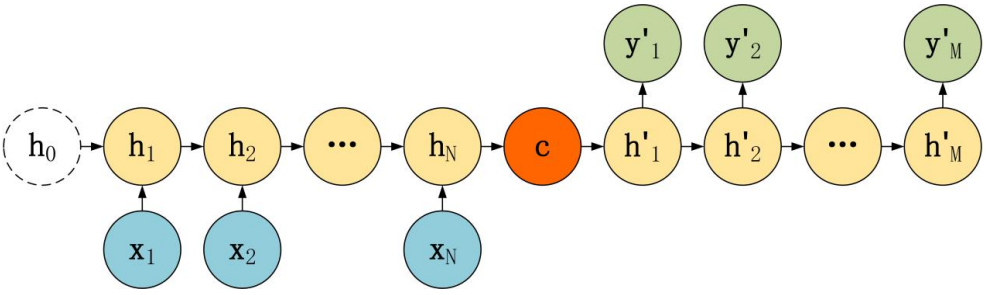


图 8 第一种 Seq2Seq 模型结构

第二种的结构图如下所示，解码器有自己的初始输入，将编码器得到的输出序列当做每一层的输入，而不是解码器的初始隐藏层输入。

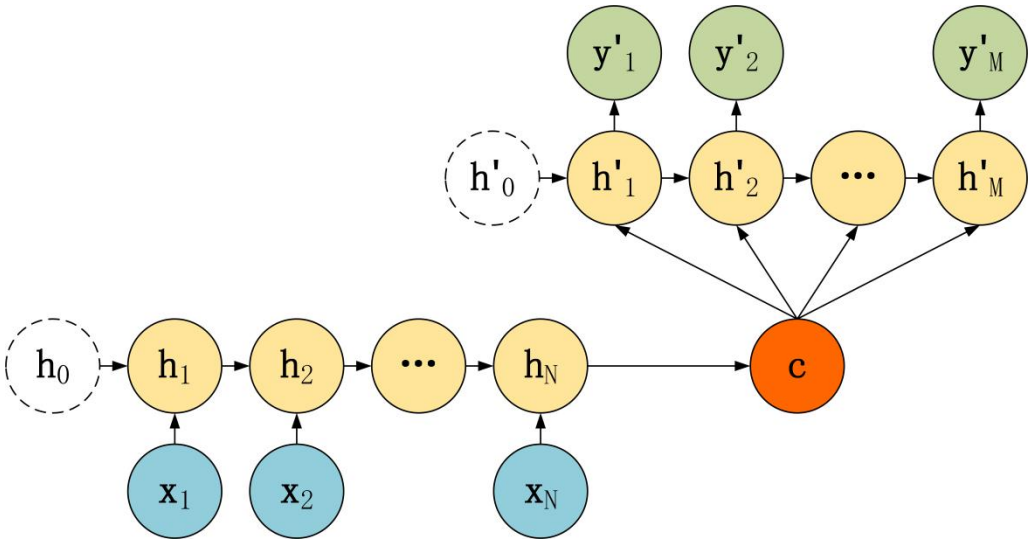


图 9 第二种 Seq2Seq 模型结构

第三种模型的结构如下图所示，同样，解码器有自己的初始输入，将编码器得到的输出序列当做每一层的输入，与第二种不同的是多了上一个神经元的输出，即每个每一个神经元的输入由三部分组成。

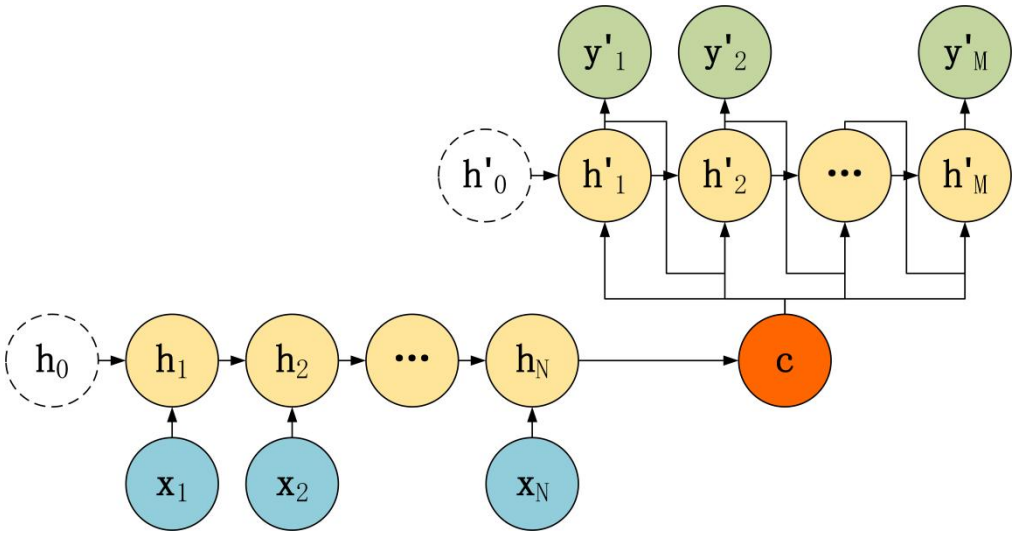


图 10 第三种 Seq2Seq 模型结构

2、Teacher Forcing 机制：

Teacher Forcing 机制主要用于训练阶段，主要功能就是缓解因上一个神经元输出错误造成一连串的错误，即防止一步错步步错的问题。

三、代码

代码结构：

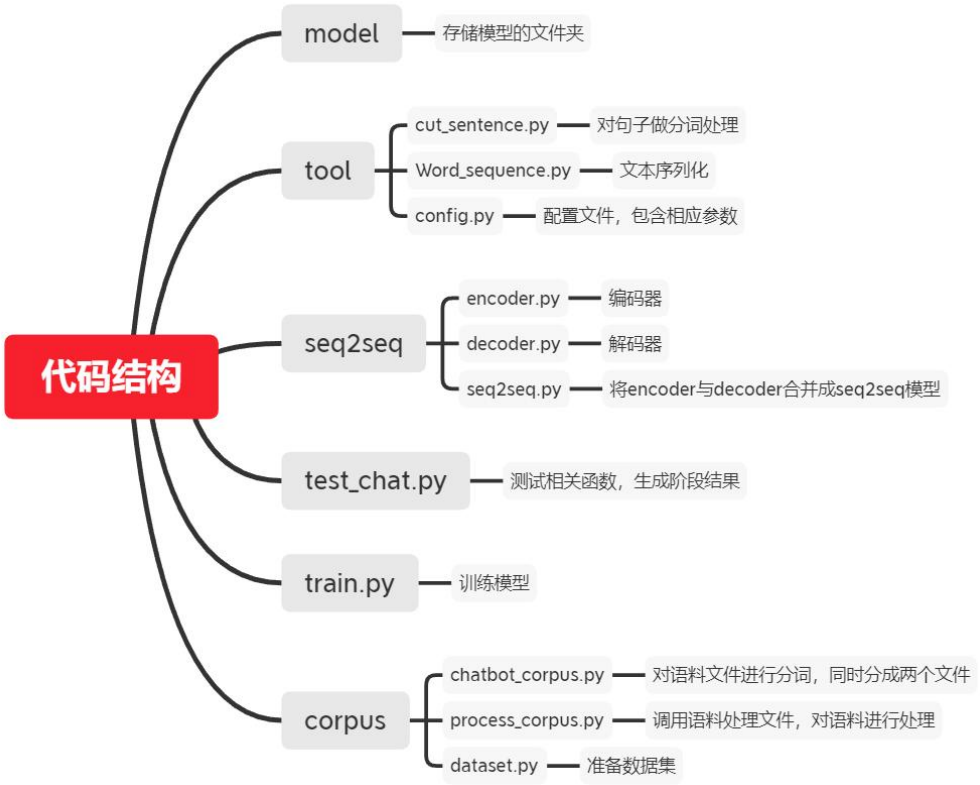


图 11 代码结构思维导图

四、训练曲线

图为 10epoch 的运行结果图，可以看见损失还是比较大的。图和图是 10epoch 的运行结果图，前 10 次迭代的损失函数还是比较大的，经过训练后，后 10 次迭代的损失函数程逐渐减小趋势，均在 0.4-0.2 间可见训练是有效的。

```
epoch:0 idx:0    loss:4.982: 100%|##########| 1/1 [00:01<00:00,  1.71s/it]
epoch:1 idx:0    loss:4.909: 100%|##########| 1/1 [00:00<00:00, 20.89it/s]
epoch:2 idx:0    loss:4.828: 100%|##########| 1/1 [00:00<00:00, 17.59it/s]
epoch:3 idx:0    loss:4.731: 100%|##########| 1/1 [00:00<00:00, 19.66it/s]
epoch:4 idx:0    loss:4.562: 100%|##########| 1/1 [00:00<00:00, 21.80it/s]
epoch:5 idx:0    loss:4.356: 100%|##########| 1/1 [00:00<00:00, 19.28it/s]
epoch:6 idx:0    loss:4.172: 100%|##########| 1/1 [00:00<00:00, 21.80it/s]
epoch:7 idx:0    loss:4.468: 100%|##########| 1/1 [00:00<00:00, 16.71it/s]
epoch:8 idx:0    loss:4.008: 100%|##########| 1/1 [00:00<00:00, 20.05it/s]
epoch:9 idx:0    loss:3.983: 100%|##########| 1/1 [00:00<00:00, 19.66it/s]
```

图 12 训练结果截图 10epoch

```
epoch:0 idx:0    loss:4.992: 100%|##########| 1/1 [00:00<00:00,  2.94it/s]
epoch:1 idx:0    loss:4.840: 100%|##########| 1/1 [00:00<00:00, 19.66it/s]
epoch:2 idx:0    loss:4.695: 100%|##########| 1/1 [00:00<00:00, 20.89it/s]
epoch:3 idx:0    loss:4.494: 100%|##########| 1/1 [00:00<00:00, 20.46it/s]
epoch:4 idx:0    loss:4.237: 100%|##########| 1/1 [00:00<00:00, 21.80it/s]
epoch:5 idx:0    loss:4.342: 100%|##########| 1/1 [00:00<00:00, 21.33it/s]
epoch:6 idx:0    loss:4.245: 100%|##########| 1/1 [00:00<00:00, 23.32it/s]
epoch:7 idx:0    loss:4.151: 100%|##########| 1/1 [00:00<00:00, 17.59it/s]
epoch:8 idx:0    loss:4.061: 100%|##########| 1/1 [00:00<00:00, 18.92it/s]
epoch:9 idx:0    loss:3.973: 100%|##########| 1/1 [00:00<00:00, 17.59it/s]
```

图 13(a) 训练结果截图 200epoch

```
epoch:190   idx:0    loss:0.321: 100%|##########| 1/1 [00:00<00:00, 20.05it/s]
epoch:191   idx:0    loss:0.315: 100%|##########| 1/1 [00:00<00:00, 18.92it/s]
epoch:192   idx:0    loss:0.310: 100%|##########| 1/1 [00:00<00:00, 20.89it/s]
epoch:193   idx:0    loss:0.304: 100%|##########| 1/1 [00:00<00:00, 21.33it/s]
epoch:194   idx:0    loss:0.299: 100%|##########| 1/1 [00:00<00:00, 20.89it/s]
epoch:195   idx:0    loss:0.293: 100%|##########| 1/1 [00:00<00:00, 20.05it/s]
epoch:196   idx:0    loss:0.288: 100%|##########| 1/1 [00:00<00:00, 19.28it/s]
epoch:197   idx:0    loss:0.283: 100%|##########| 1/1 [00:00<00:00, 19.66it/s]
epoch:198   idx:0    loss:0.278: 100%|##########| 1/1 [00:00<00:00, 19.28it/s]
epoch:199   idx:0    loss:0.273: 100%|##########| 1/1 [00:00<00:00, 18.92it/s]
```

图 13(b) 训练结果截图 200epoch

如图为 epoch-loss 训练曲线图，从图中可以看到整体趋势是随着训练轮数的增加损失函数在逐渐减小，但是幅度不大，并且存在不同程度的波动问题。后续可以进行模型优化，同时增加语料库的内容，

相信可以提高模型的性能，得到更好的 epoch-loss 训练曲线图。考虑通过加入 attention 机制来优化模型，当然也可以使用调整参数等方法。

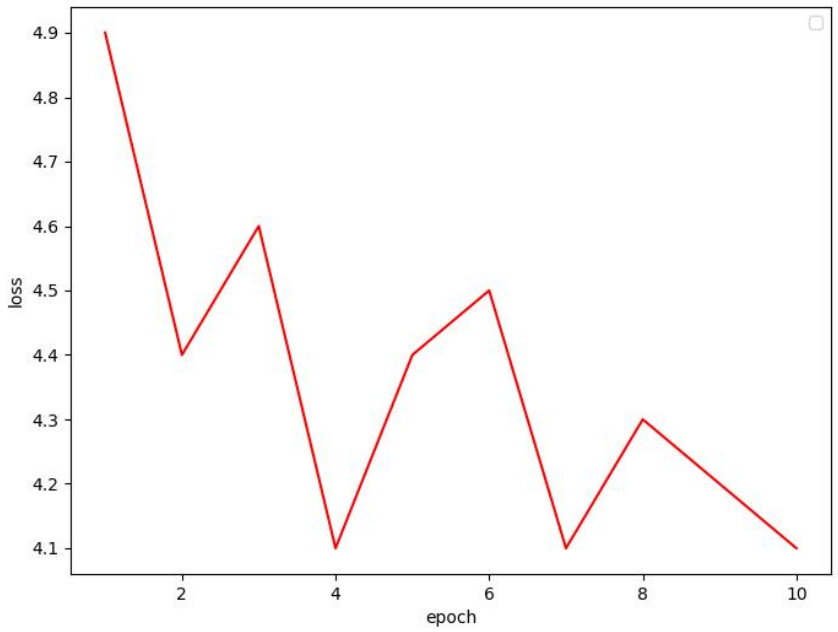


图 14 epoch-loss 训练曲线图（10epoch）

如图 5 为运行结果截图，可以在一定程度上进行闲聊，但聊天的前后相关性有待提高。

```
I：你好啊
you：好久不见啊最近怎样
I：挺好的，你呢
you：还好啊挺好的
```

图 15 运行截图

五、实验总结

（1）该实验主要是收集原始语料，对其做分词、文本序列化、准备数据集，在数据集的基础之上完成基于 seq2seq 模型的聊天机器人。

（2）在实验中尝试采用了 Teacher Forcing 机制，还可以尝试使用 Attenton 机制。

（3）实验性能比较一般，这可能是数据集够大造成的，可以增加语料库语料，或者并入已有的语料库。

（4）因为预训练模型已经有比较成熟的了，例如 bert，也可以尝试使用预训练来提高聊天相关性并降低训练的损失函数。