复旦大学计算机科学技术学院

2019-2020 学年第一学期期末论文课程评分表

课程名称： **自然语言处理** **课程代码：** COMP130141.01

开课院系： 计算机科学技术学院

学生姓名： 宁晨然 **学号：** 17307130178 **专业：** 计算机科学与技术

论文名称：Seq2seq 模型构建电影对话机器人

（以上由学生填写）

成绩：

论文评语（教师填写）：

任课教师签名：

日　　期：

摘要

本文使用了 seq2seq[1] 模型结合 attention 的机制，使用康奈尔大学电影学院对白语料库作为数据集，训练 60000 迭代次数得到了一个对话机器人。输入对话的上文句子，输出对话的下文句子。交叉熵误差从最开始的 261.1714 降到了 42.9374。最终对话机器人能够回复简单的英文对话。

```
me: Hi
Ning: hi .
me: Hello!
Ning: hello .
me: How are you?
Ning: fine .
me: How do you do?
Ning: i 'm a little good .
me: Are you happy today?
Ning: no .
me: What's your name?
Ning: i 'm not .
me: Where are you from?
Ning: south .
me: thank you
Ning: you 're a little late .
me: I'm sorry.
Ning: why ?
me: I'm late.
Ning: what ?
me: ok
Ning: i 'm not sure i 'm not .
me: How old are you?
Ning: i 'm a little .
me: How old are you?
Ning: thirty-five .
```

图 1　基本问题

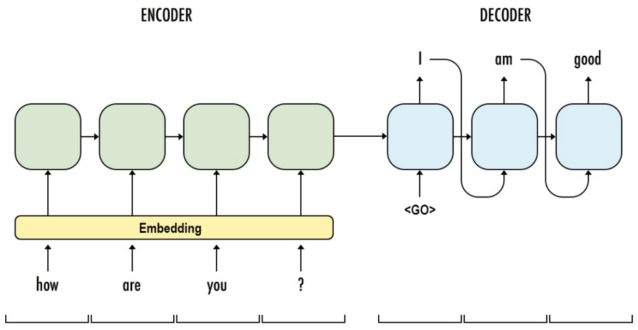

图 2

关键词：seq2seq；聊天机器人；attention；GRU

目　录

第 1 章　引入

1.1　创意来源

我们时常看到科幻片中的未来机器人风趣幽默地与人类谈话，不仅实时应答，还俏皮可爱。本人一直是个狂热电影迷，受到《星际穿越》的 TARS 的启发，我想着手做一个聊天机器人。该电影中的 TARS（图 1.1）可以随意调节幽默指数，最终为了保护人类撒谎牺牲，令人印象深刻。而搜集资料发现聊天机器人 chatbot 在 NLP 的研究已经很广泛，虽不是一个特新颖的话题，但仍在研究的火热阶段，故我选此题。

图 1.1　诺兰导演电影《星际穿越》机器人 TARS

1.2　简要步骤

- 爬取英文电影对白，筛选处理语料库
- 利用对白台词生成训练集和测试集
- 训练集用 seq2seq 模型训练对话机器人
- 测试准确度和损失，评估模型

第 2 章　数据处理

2.1　数据来源

所谓近朱者赤近墨者黑。我查阅了许多现有的聊天机器人模型，分析了其中的使用背景和训练集。比如，常见的问答系统，包括基于用户体验的便利操作聊天机器人 Siri 和小爱同学，是根据用户操作关键词给出响应；保险公司、医疗对话、淘宝商家的机器人，是基于专业知识甚至专家系统给出专业应答；而 QQ 小冰聊天，是基于 QQ 用户的群聊习惯提出有趣话题…但是都应证了**数据集决定了聊天内容的质量。**

一开始决定使用我的电脑客户端 QQ 的聊天记录，检查导出 txt 发现三个问题决定不采用：1. 聊天记录数据集基本只有 2017 的 7-9 月三个月的时间，不够健壮。2. 聊天内容很松散，没有成型语句较多，口语成分严重。3. 许多对话并不是一问一答经典对话。4. 涉及隐私。故决定放弃 QQ 聊天（或微信聊天）。

而热衷于电影的我立马想到了电影对白：1. 字幕文件易于获取。2. 电影对白质量更高，成型句子多、表意更清晰、场景冲突明确。3. 对白易于分割为一问一答的数据集。所以选择英文电影对白作为数据集。但本人爬虫太小白，没有爬虫经验爬取字幕网站（虽然网站很多资源也多），搜集资料中发现**康奈尔电影学院的对白语料库**[2]。包含大量英文原生对白，质量极高如表（ 2.1）。

表 2.1　Cornell Movie–Dialogs Corpus

File	Content
movie_titles_metadata.txt	电影标题信息
movie_characters_metadata.txt	电影角色信息
movie_lines.txt	对话实际语料
movie_conversations.txt	对话结构

2.2　预处理

原始数据用” +++$+++ ” 作为分隔符，重点文件是 movie_conversations.txt 和 movie_lines.txt。由于我想要使用的模型是 seq2seq 的结构（类似翻译任务，可变

长），故需要 1:1 的来回对话系统，在处理前有几个疑虑：

1. 电影对话中通常一个人会说多个句子，另一人才会回复。

2. 回复内容可能基于上一人说的所有句子，也可能基于其中一句。

3. 对话通常受语境（对话情景）、角色风格、电影主题的影响。

本人确实能力有限，只能作出假设简化模型，对白不考虑语境和角色；对话内容的回复基于 attention 机制，尽量寻找想要回复的部分。故预处理的方式就是，读取每个对话：假设对话有 L1,L2,L3 三句，则生成 (question,answer) 组：（L1,L2),(L2,L3)。考虑到**可能出现异常符号**，其中对于每句话需要分词后重组，使用 Tokenize 函数 2.1。该函数用于分词，在后续输入和建立词汇库也会用到。

```
def tokenize(sentence):
    sentence = sentence.lower()
    sentence = re.sub(r"(<u>)|(</u>)", r" ", sentence)
    sentence = re.sub(r"[^a-zA-Z',.!?-]+", r" ", sentence)
    if TOKENIZER == 'nltk':
        words = nltk.word_tokenize(sentence)
        pattern = re.compile(r'[-\.\']{2,}')
        words = [word for word in words if not re.match(pattern, word)]
        return words
```

图 2.1　Tokenize 函数

分词需要用正则表达式去除原始语料中的”<u>”,”</u>” 和非正规字符部分。使用 nltk 的分词器 word_tokenize，且清理出空白词语。生成 word 的 list。之后使用 sentence = ' '.join(tokenize(sentence_id[dialogue_ids[i]])) 重组原句子（保留了标点符号）。

生成的语料库则由处理后的单词和符号组成，存储为 dialogue_corpus。每行内容为 question:answer（1：1），中间由” +++$+++ ” 分隔。样本如 2.2。

```
1  can we make this quick ? roxanne korrine and andrew barrett are having an incredibly horrenc
2  well , i thought we 'd start with pronunciation , if that 's okay with you . +++$+++ not the
3  not the hacking and gagging and spitting part . please . +++$+++ okay then how 'bout we try
4  you 're asking me out . that 's so cute . what 's your name again ? +++$+++ forget it .
5  no , no , it 's my fault we did n't have a proper introduction - +++$+++ cameron .
```

图 2.2　dialogue_corpus 语料

2.3 词汇语料库

由于使用词向量 word2vec 模型，需要先建立词汇语料库，统计词频和词典。从上面预处理的语料库中，先分词后，统计出现了哪些词语，且每个词语的出现频率（这部分简单）。之后的重点在于建立词语与 index 的映射：1. 首先考虑到词语出现频率较低的词汇可以直接删减，便于后面向量空间维数降低。2. 再考虑到词语到 index 的映射过程需要三张 dict,word2count,word2index,index2word。index 唯一标注一个词汇。Vocabulary 的 word2index 如图 2.3。

```
13  are 12
14  having 13
15  an 14
16  incredibly 15
17  horrendous 16
18  public 17
19  break- 18
```

图 2.3

经过删减后的词汇再筛去无效对白，共有 173279 个对话（一问一答）。词汇库有词汇量 32107 个词语。其中对白的保留率为 0.9430，词汇的保留率为 0.7155。

再建立训练集和测试集，取测试集为数据集随机的 1/10。训练过程中不是单个 sequence 训练，为了提高效率使用 batchsize=64。这样一共有 2437 个 batch。对于每个 batch 中，可以分为 input 和 target 两个部分，每个部分中包含 batchsize 个句子。需要再对句子进行 index 映射，转换为 input 和 target 的 index 矩阵。这样一个 input 矩阵对应一个 target 矩阵构成了整个数据集。再随机抽取 1/10 作为测试集，剩下的作为训练集。

2.4 发现的问题

在之前就考虑到了电影独白作为语料库可能的问题，在处理好所有的词汇和句子数据后，确实出现了下面的问题：

首先，句子的长短，短的很短，长的很长。比如很多回答会直接用”yeah”,”I don’t know”,”Huh”,”fine” 这种单词回答，这类回答还比较合理，日常生活中确实就是这么回答的。但是长句子的问题就比较严重了，比如 i ’m kidding . you know how sometimes you just become this persona ? and you do n’t know how to quit ? +++$+++ no。

这个句子前面的问句有开玩笑（应该是对上一句的回复）、提问 1、提问 2

（对于提问 1 的进一步阐述），但是后面的答句只有 no。而我们直接看对话，其实也并不知道这个对话的场景和语境，所以对于 quit/become this persona 等动词的语义甚至并不知道，这个时候学习出来的回答 no 可能只是概率较大的可能性，而并非基于前面的问句。

同理还有反过来的结构，前面句子很短，后面句子很长。great +++$+++ would you mind getting me a drink , cameron ? 其实后面的句子应该是一个问句部分，因为对话总是互相抛问题，逻辑上并不是前者生成了后者，因为后面这么多单词也不是从一个 great 生成出来的。且同理，没有语境和上下文，我们人类无法得知 get me a drink 和 cameron 的语义，让机器学习这个例子或许没有太大用处。

还有一个疑惑就是，比如 exactly so , you going to bogey lowenbrau 's thing on saturday ? +++$+++ hopefully . 这个句子是一个典型的问答句子，学习效果应该较好，但是 hopefully 的回答对象应该是 you go ?，其他的特别是 exactly so 对于回答的作用几乎没有（因为是对上一个问题的回答)。所以后面的 seq2seq 模型才会需要继续改进为带有 attention 机制的 seq2seq。

第 3 章　模型描述

原任务可以分割为几个步骤：1. 未知长度的英文句子输入后分词。2. 根据分词的词向量 (word2vec) 作为输入。3. 根据输入序列用 encoder 求出隐藏语义特征。4. 根据隐藏语义特征 decoder 求出输出序列。5. 求解交叉熵 loss 函数并训练模型。

3.1　seq2seq

原任务是不定长的句子输入序列，给出一个不定长的输出序列。这个时候的任务有点类似于翻译任务，比如中译英中的序列长度就是不固定的。采用 seq2seq 模型的要领在于 encoder 和 decoder 的选择，我选择了的 GRU 作为其中的单位。seq2seq 模型，这里谈一下我的理解。

seq2seq 中，由于给出的输入和输出的词语序列不固定长度，所以对于每次的 batch 训练需要统一最大长度，如果不够的地方使用 <pad> 填充。如图 3.1，Encoder 用于解析句子的语义，最终的输出是 c（隐藏语义），而右边部分的 Decoder 则根据语义 c 解析生成输出。中间的 GRU 权重、词向量权重等均为学习对象。

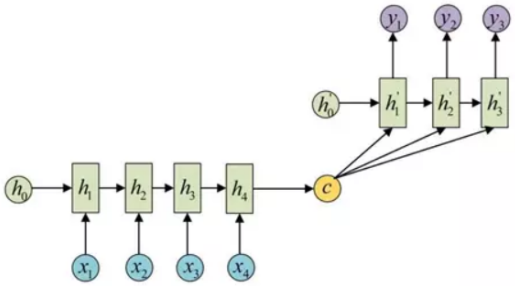

图 3.1

3.2　GRU

任务是 seq2seq 的模型，基本单位结构是 encoder 和 decoder，更基本的单位就是其中的原始网络结构。其实可以使用 RNN/LSTM/GRU，后来对比资料发现：

首先，为了让句子序列能够全部被记忆，肯定需要实现序列的有记忆性的网

络，比如 RNN。RNN 能够很好的捕捉语言句子中的长句子依赖关系，但是句子一旦变长前面的依赖关系也会减弱。结合上课知识点我知道 RNN 有梯度弥散和梯度消失的可能性，所以没有采用。再者，RNN 的改进版本可以使用 LSTM 长期记忆网络，采用了输入门、遗忘门、输出门和内部记忆单元（我也只是浅尝辄止略看懂），遗忘门可以控制了记忆单元中多大信息可以被遗忘，网络更容易学习到序列之间的长期依赖。然后，再到 GRU 门控循环单元，经过 LSTM 简化可以得到，资料显示训练效果和 LSTM 差不多但是计算速度更快，故采用了 GRU。

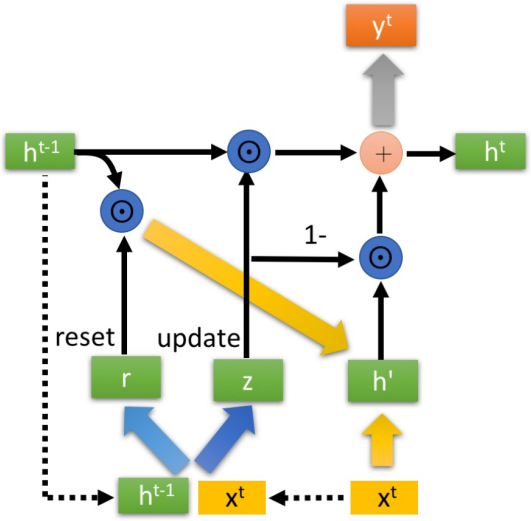

图 3.2

这里谈一下我对 GRU 的浅理解：因为句子序列需要长期记忆模式、且有时候需要遗忘某些特征，根据下面公式，其中 z_t 表示更新门，决定信息的遗忘和添加；R_t 表示重置门，用于决定丢弃程度。

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{3-1}$$

$$r_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{3-2}$$

$$\hat{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{3-3}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \hat{h}_t \tag{3-4}$$

3.3 Encoder

Encoder 部分，输入的矩阵是 batch 封装的训练集，对于每个句子：1. 句子分词。2. 分词映射到词汇表中的 index。3. 转换为词向量。其中词向量的转换过程也是需要学习权重矩阵的过程。前向传播过程由多个 GRU 组成，GRU 的长度由本次 batch 中最长 input 长度决定，即 input_lens。

Embedding 转换矩阵为 input_size*hidden_size，每个 GRU 的权重矩阵也是 hidden_size*hidden_size。

3.4 Decoder

Decoder 部分，输入的矩阵是隐藏语义矩阵，每个 GRU 的权重矩阵是 hidden_size*hidden_size，最终的输出转换矩阵为 hidden_size*hidden_size。

3.5 loss 计算

这里的 loss 函数使用了交叉熵，公式如下：

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^{T} e^{a_k}} \tag{3-5}$$

$$L = -\sum_{j=1}^{T} y_j \log s_j \tag{3-6}$$

3.6 反向传播

设定学习率为 0.0001，迭代次数为 60000。并且每 200epoch 就保存一次状态，输出测试集上的平均误差。

3.7 Attention 的改进

Attention 部分的机制其实我没有怎么看懂原理，但大致了解了整个原理和核心思想。首先针对之前提出的问答问题，在翻译问题中也出现了，翻译的句子

中有一些成分明显是人类注意力更高的地方。在 seq2seq 问题中，电影对白的前句子可能由多个句子构成，并且包含了对于上一个问题的回答，但是人类回答问题的机制是集中注意力到了后面的问询句子或者后面的回复。

上面的 seq2seq 有一定的缺陷，就是如果 encoder 和很长，信息量都会被压缩为一个固定长度的隐藏语义向量，意味着信息可能丢失。查阅资料发现翻译问题中 sentence 长度增加，decoder 的翻译效果也明显变差。

注意力模型有三个步骤：

- 计算注意力得分
- 进行标准化处理
- 结合注意力得分和隐状态值计算上下文状态 C

第 4 章　训练结果

图 4.1和 4.2分别给出了每 200 迭代次数记录的平均交叉熵，和随着迭代次数变化的交叉熵。

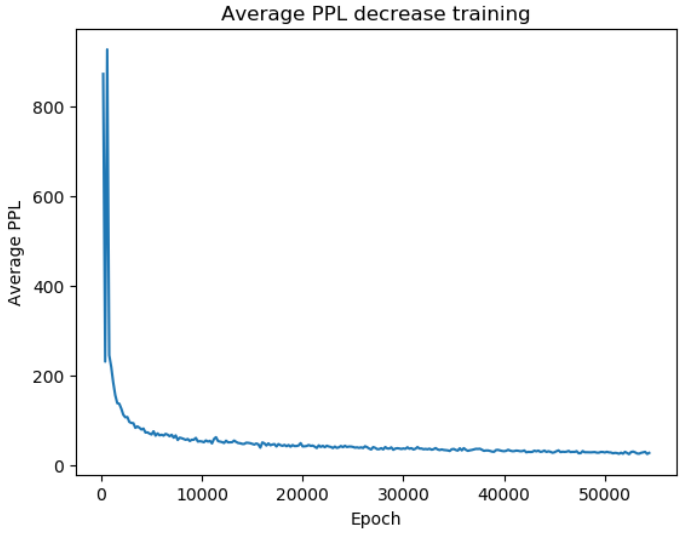

图 4.1

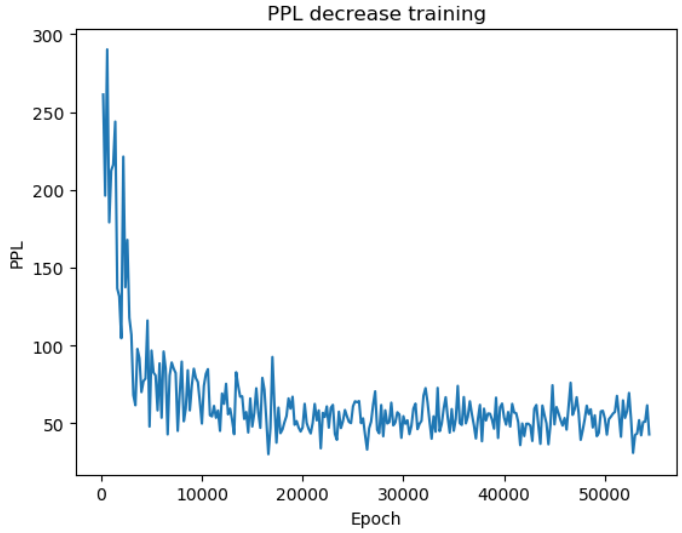

图 4.2

4.1 实验评估

交叉熵从一开始的 261.1714 降到了 42.9374，而且从图像上看，最后是趋于收敛的。但是在 10000 迭代次数的时候，再到 50000 多迭代次数的学习过程看：首先每 200 迭代次数呈现出来的结果就是跌宕，即使大趋势是在下降。而且下降的趋势特别缓慢，一开始的 10000 下降的特别快，测试结果基本都是”I”,”the”,”.” 的短词汇短句子，固然误差很大。

说明一下几点：

- 学习到的短句子回答更多。
- 长句子中无关词汇很多。
- 数据集没有处理很好。

数据集上我个人感觉没有做的很好，电影对白作为训练语料库不是特别符合对话机器人的使用方法。我本来是想使用电影的对话作为对话的基准，但是电影的对话过于：1. 情景化。2. 冲突化。3. 台词个性化和口语化。如果我的能力更高，我会选择书籍中的对话系统，因为书面语更适合训练，而且选择文学大家写的文本训练效果应该较好，不过数据集可能需要大量的人工时间从书籍中抽取对白。

4.2 测试结果

总体训练了近 60000 迭代次数，最终的呈现效果，个人感觉机器人学习到的回答句子有部分能理解是可能在某些情景会发生的回答，但是很多时候回答的句子有些不搭边（当然人回答语句脱离语境本身就有些不搭边）。初心是想将机器人训练的像电影对白那么高质量，呈现出来的效果感觉它是一个没有受过教育的低龄儿童的回答。基本问题都能过关，且语法错误基本没有。

```
me: Hi
Ning: hi .
me: Hello!
Ning: hello .
me: How are you?
Ning: fine .
me: How do you do?
Ning: i 'm a little good .
me: Are you happy today?
Ning: no .
me: What's your name?
Ning: i 'm not .
me: Where are you from?
Ning: south .
```

```
me: thank you
Ning: you 're a little late .
me: I'm sorry.
Ning: why ?
me: I'm late.
Ning: what ?
me: ok
Ning: i 'm not sure i 'm not .
me: How old are you?
Ning: i 'm a little .
me: How old are you?
Ning: thirty-five .
```

图 4.3　基本问题

测试比较长的句子的时候，机器人就基本不会回答正确的语句了。我推测原因在于：1. 本身训练集不太大，没有很多情景或者语言组织的方式。2. 没有对白语境，人类也很难回答。3. 长句子更难学习。

```
me: Actually you are a robot. I'm not kidding.
Ning: i 'm not sure .
me: You can't just leave me like this.
Ning: why not ?
me: I don't know what you are talking about.
Ning: i do n't know .
me: Is there anything you haven't told me?
Ning: what ?
```

图 4.4

第 5 章　实验感想

本次实验研究了 seq2seq 的模型，结合了上课所学的词向量、分词、RNN 的知识，实际应用了 python 写神经网络模型解决语言问题。本身我对电影的台词就很感兴趣，本来是想做一个编剧的台词生成系统，利用各种电影的剧本台词生成一段精彩的对话。但是发现，生成模型的训练集要求肯定很大，个人笔记本确实很难办到较好的训练过程出结果。并且台词的数据集需要大量的爬虫，而我确实没有爬虫的经验，所以无奈就放弃。

后来想到的就是 QQ 聊天群里经常会有聊天机器人，而且科幻电影中的机器人确实都是会说话会聆听会回答的机器人，所以就确定做一个聊天机器人。我在网上也测试了很多别人写好的聊天机器人，特别是有一个博主用三千万中文字幕语料库写出的聊天机器人，在线测试了一下，发现基本的问题能回答，但是基于文本语义语境的对话，有质量的对话基本没有。而且训练出来的机器人不知怎么特别喜欢骂人……所以我也打消了使用中文电影字幕语料库。最后采用了查资料查到的康奈尔电影学院的语料库，在 nlp 的论文中也时常引用。

我参考了 github 上面一位大神写的 seq2seq 模型[3]，里面建立好了 encoder 和 decoder，且使用了 GRU 和 attention 的机制。最后测试出来的结果还行，比较符合我一开始的认知了。GRU 和 Attention 的机制我看懂了大概。

这是第一次使用神经网络 python 代码跑模型，一开始安装 anaconda 和 pytorch 和 cuda 的经历就让我痛不欲生…花费很多时间搭建好环境后，使用 gpu 跑代码，个人笔记本实在是跑不动，最后训练了 12 小时跑出了 60000 迭代次数已经很欣慰了。可能也是 batchsize 没有选择好吧。不过应该是数据集的处理问题上是本次实验结果不够理想的最大原因。神经网络构建部分代码我尽力的去看了，中间矩阵乘法的过程有点难以理解，还有 pytorch 的语法也让我琢磨了半天。

总体而言，这次实验对我 python 代码的书写和阅读能力有了提升，并且领我进入了神经网络的门。实验过程让我加深了对神经网络的理解。

参考文献

[1] [EB/OL]. https://www.guru99.com/seq2seq-model.html.

[2] Danescu-Niculescu-Mizil C, Lee L. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs.[C]//Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011. 2011.

[3] Li Xin. dumb-chatbot[EB/OL]. 2019. https://github.com/Great-Li-Xin/dumb-chatbot.