复旦大学计算机科学技术学院

2018-2019 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： 蒋艳琪 **学号：** 16307130316 **专业：** 计算机

论文名称： 对联的生成与应对——基于 HMM 模型与 Seq2Seq 模型

成绩：

《自然语言处理》课程项目

对联的生成与应对——基于 HMM 模型与 Seq2Seq 模型

编程语言：Python3.6
所需 Python 库：

BeautifulSoup：用于 html 网页的解析
pypinyin：获取汉字的读音和音调
jieba：用于中文分词和词性标注
nltk：用于获取 bigram 与频率分布
tensorflow：用于神经网络的搭建与训练
wordcloud/harvesttext/networkx/pyecharts：用于数据分析，网络关系图的绘制

程序清单：
auto-couplet/couplet analysis.py：对联文本分析
auto-couplet/HMM_couplet.py：HMM 模型
auto-couplet/images：文本分析生成的图片
seq2seq_couplet/seq2seq_couplet.py：Seq2Seq 模型
seq2seq_couplet/data：训练得到的模型
couplet：收集的对联语料

关键词： 生成上联 联语应对 文本分析 序列模型

思路来源

春节将至，对联作为我国的传统艺术形式，寓意吉祥，寄托了人们对未来的美好祝愿。对联要求对仗工整、朗朗上口，这也造成了古往今来很多绝对的产生。所以我想设计一个自动生成与应对对联的系统，以辅助对联创作，尽量使得对联有意义有内涵，又充满新意。或许人们能从人工智能的奇妙组合中得到新的创作灵感。

语料获取与数据清洗

要能做到自动生成对联和应对对联，必须要有大量的语料。为此，我先在网上搜索了对联，找到了一些对联大全的网站，比如中国对联网[1]，对联欣赏网[2]，360 个人图书馆[3]。这些网站虽然有大量对联数据，而且是静态加载的 html 页面，爬虫比较方便，但是爬取到的网页格式不一致，需要对得到的数据进行清洗以得到标准的对联数据集。

[1] 中国对联网 http://www.duiduilian.com/
[2] 对联欣赏网 http://www.86duilian.com/index.html
[3] 360 个人图书馆 http://www.360doc.com/content/17/1208/19/1003261_711342387.shtml

数据清洗的流程大致如下：

1. 查看原始 html 网页，确定其中对联文字所在的位置。
2. 利用 BeautifulSoup 进行网页的解析，比如，提取出对联文字所在的类，提取标签中的文字内容
3. 将对联按照上下联分别存入单独的 txt 文件

在处理 html 文件的过程中，由于格式不一，遇到了很多麻烦。比如有的网站不仅包含上下联，还包含横批；有的网站还用不同的颜色显示文字，需要对<p>标签中的内容进行过滤。我自己写的爬虫程序在 spider_couplet.py 中。比较成功地爬取了中国对联网的四字春联，和 360 个人图书馆博客的春联数据。但是后来我在 github 上发现了已经整理好的对联数据[4]，于是就直接拿来使用了。

对联文本分析

为了生成符合人们认知的对联，我首先对得到的对联数据进行了文本分析。分析方式主要包括：生成词云图，分析对联中出现的高频词；判断对联中每个字的平仄，得到对联的音韵特点；提取对联中的常见意象，分析这些意象的共现关系。

提取高频词

【方法】首先，将得到的上下联每一行拼接起来，组成一个纯内容的文本。利用 jieba 分词工具进行分词，利用 nltk 的 FreqDist 得到词汇频率表。提取其中最高频的 100 个词，生成词云图：

由于数据集中春联较多，可以明显地看到一些与春天、祖国、盛世相关的，带有美好祝愿的词汇，还有红梅、玉兔、梅花等富有新年特点的意象。背景图片是一个灯笼的形状。

对联的音韵特点

分析对联的音韵特点，主要是每个字的平仄。平仄相谐是一种听觉感知形态的语音层面的对

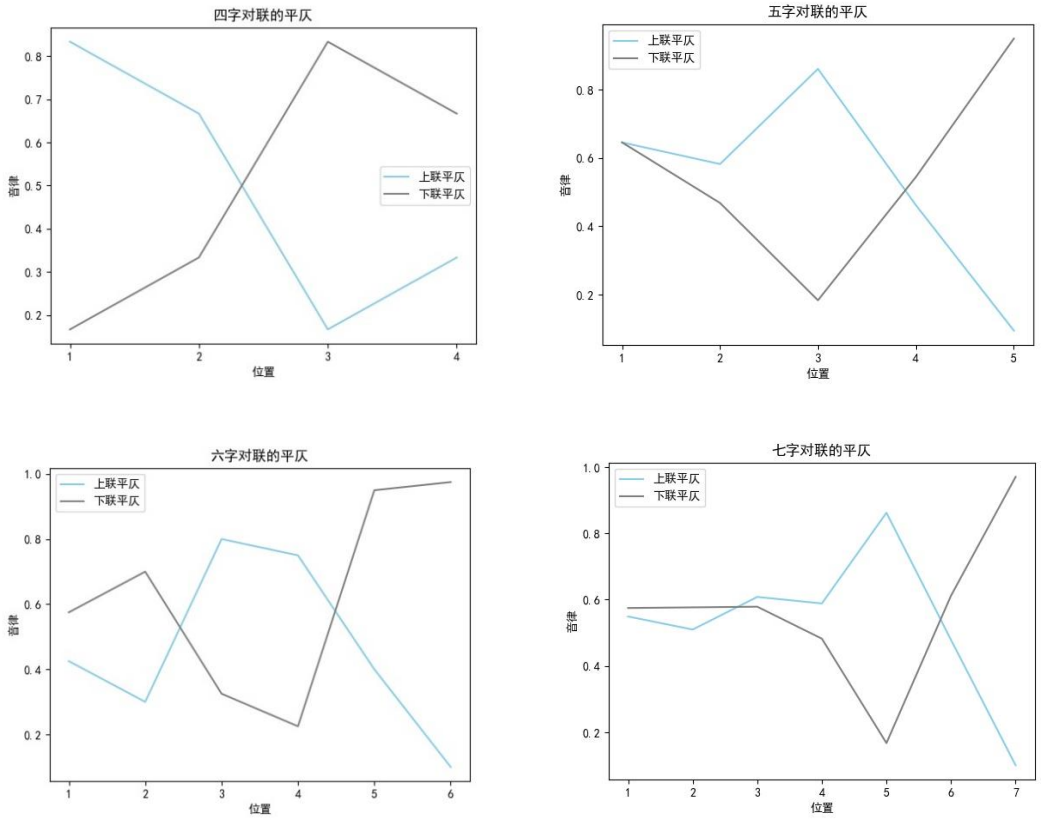
[4] https://github.com/htw2012/seq2seq_couplet

仗，一副对联要符合“平仄”的要求，必须要遵守“字节要平仄相间”、“对字要平仄相反”、“上联末字要收仄”、“下联末字要收平”的基本原则，人们耳熟能详的对联音律歌诀就有“平平仄仄平平仄，仄仄平平仄仄平”。但是由于对联的字数不定，所以不能简单地和格律诗进行类比；而“对字平仄要相反”的原则在实际运用中要求也不是很绝对，为了表意往往很难做到平仄相反。但是一般还是应该有一些平仄的变换，这样才能产生错落回旋的音乐感，给人轻重缓急的美感享受。[5]

【方法】为了获取每个字的平仄，调用了 pypinyin 库得到了每个汉字的拼音。利用 pypinyin.pinyin(sentence, style=pypinyin.TONE2)语句为对联中每个字标注拼音和音标。编写了程序 tone()获取每个字的平仄。

```
def tone(c):  # 平为True  仄为False
    res = pypinyin.pinyin(c, style=pypinyin.TONE2)
    res = res[0][0]
    for i in res:
        if i.isdigit():
            if i == '1' or i == '2':
                return True
            else:
                return False
    return False
```

由于四字、五字、六字、七字对联的平仄特征比较明显，故对其分别进行音律分析。以四字对联为例，首先提取所有的四字的上联和下联，对每个句子中的每个字判断其平仄，统计相同位置的字的平仄，得到所有四字上联、四字下联的平均平仄程度。其中音律取值从 0 到 1，取值越高表示这个位置音律为平的概率越大。绘图如下：



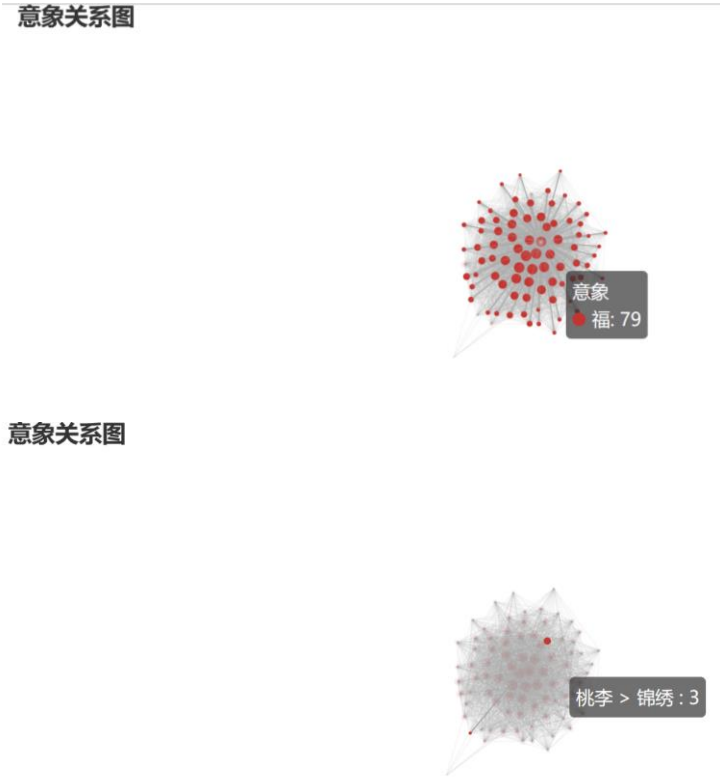
由折线图分析，四字对联能见到明显的“平平仄仄，仄仄平平”的特征，七字对联也能见到比较明显的“平平仄仄平平仄，仄仄平平仄仄平”的特征。上联收仄、下联收平更是所有对联基

[5] 对联例话（5）：对联的平仄（上）http://www.duiduilian.com/zhishi/gjlh/3313.html

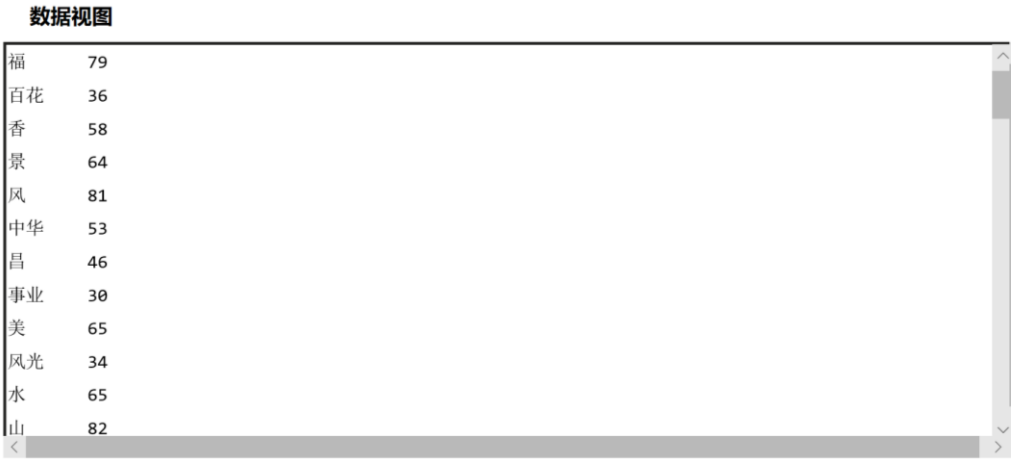
本都要遵循的规律。

意象共现关系

为了挖掘意象之间的网络关系，可以利用邻近共现关系。主要流程为实体识别-实体链接-建立连接。由于一般意象都是名词，利用 jieba 的分词和词性标注工具识别出所有的名词实体，利用 nltk.FreqDist 构建频率表，选取其中最高频的 100 个名词意象。每当一对意象在上下联内同时出现，就给它们加一条边。利用 HarvestText 提供的函数可直接完成这一过程，得到一张 networkx 类型的 graph，用 pyecharts 进行可视化（为动态交互式图表，html 格式，在 auto-couplet/images 中），这里给出截图：



数据视图中可以查看各个意象的中心度（与其他意象的关联程度）：



有趣的意象共现关系比如：猪与财，猪与锦绣，猪与小康。由此可以挖掘出一些意象的隐含意蕴。

HMM 模型

对联的特点在于字数较少，有较多的常见意象，并且上下联相同位置的词之间有一定的对应性。于是我考虑从分词得到的所有词汇中提取候选词汇，使用词汇堆砌的方式创作上联；同时利用上下联之间的对应性，根据生成的上联对出相应的下联。

模型准备

HMM 模型常用于文本生成。为了实现 HMM 模型，首先我计算出了状态转移矩阵和生成矩阵的值。主要包括如下几种：

1) 词性转移概率。要计算 pos1→pos2 的概率，需要得到(pos1, pos2)的数量，以及 pos1 的数量，通过相对频率计算转移概率。举例如下(其中 START 表示句子开始，END 表示句子结束，ns、n、c、v 等为 jieba 词性标注所用的标记[6])：

词性转移对	概率
(START, ns)	0.077
(c, v)	0.183
(n, END)	0.244

2) 每个词由每个词性产生的概率。要计算 pos→word 的概率，需要得到恰好标记为 pos 的 word 数量，以及 pos 的总数量，通过相对频率计算产生概率。举例如下：

(词性，词)	概率
(ns，北国)	0.0018
(a，茂盛)	0.0015
(v，把酒)	0.0037

3) 上下联之间词对应的概率。对上下联分词后处于相同位置且字数相等的词（由于 jieba 分词不一定能精确地将上下联划分为恰好对应的语言单位），计算给定上联的 word1，恰好对应下联的 word2 的概率。同时为了防止上联训练语料库中未出现过的词找不到对应的词，把每个对联都切成单字，计算上下联单字之间对应的概率。用于对对联。举例如下：

上下联对应	概率
(江山，岁月)	0.085
(江山，华夏)	0.021
(昌盛，吉祥)	0.125
(昌盛，兴隆)	0.125
(红梅，喜鹊)	0.091
(红梅，金穗)	0.030

4) 词转移概率。将切分出来的词组成 bigram，计算 word1→word2 的概率。用于生成上联。举例如下：

词转移对	概率
(中华，大地)	0.025
(中华，江山)	0.008

[6] Jieba 分词词性标注以及词性说明 https://blog.csdn.net/enter89/article/details/80619805

(春光，似锦)	0.004
(春光，送暖)	0.004
(红梅，吐蕊)	0.006
(红梅，映雪)	0.027

【方法】对每个联句，首先使用 jieba 分词和词性标注工具得到每个词和其词性，再利用 nltk 的 bigrams 方法和 FreqDist、ConditionalFreqDist 进行词频统计，最后计算出上述概率。为了避免每次重新计算，我将得到的概率利用 pickle 以模型文件的形式保存了起来，每次初始化类对象的时候先加载预训练的模型，若模型不存在再进行重新计算（所有的 pkl 文件）。

生成上联

用户可以指定上联的开始，以及上联的长度。生成上联的过程用到了一个递归函数：

```
# 指定上联长度为length，初始的start用户指定。用于生成对联。
def gen_shanglian(self, start, p, length):
        if start not in self.cfd_trans_word.keys():
            return False
        nxt = list(self.cfd_trans_word[start].keys())
        random.shuffle(nxt)
        for choice in nxt:
            if len(choice) + p > length:
                return False
            if len(choice) + p == length:  # 刚好生成成功
                self.shanglian.append(choice)
                return True
            self.shanglian.append(choice)
            q = len(choice) + p
            if self.gen_shanglian(choice, q, length):
                return True
            self.shanglian.pop()
        return False
```

按照这种方式，生成十句上联，然后再对其进行筛选。由于对联的字数不定，所以每个字的平仄规律不像宋词那样明确，于是我只规定了最后一个字的平仄。上联的最后一个字若是平声，则不合要求，删去。然后比较每个句子的概率，主要是考虑相继词转移的概率和相继词性转移的概率，这样就能选出语义上比较连贯，语法上也比较通顺的句子。按照马尔可夫链，将词转移概率和词性转移概率分别相乘，得到每个句子的概率。生成上联所用函数为：

```
def Gen_shanglian(self, start, length):  # 封装了递归函数
    prob_dict = defaultdict(float)
    for i in range(10):  # 生成十句比较好的句子
        if self.gen_shanglian(start, len(start), length):
            sent = self.shanglian[:]
            sent.insert(0, start)
            last_word = sent[-1][-1]  # 获取最后一个字，判断平仄
            if not tone(last_word):  # 若是平声，不合要求，不往下看
                sent_prob = 1
                for j in range(1, len(sent)):
                    pos1 = list(jieba.posseg.cut(sent[j - 1]))[0].flag
                    pos2 = list(jieba.posseg.cut(sent[j]))[0].flag
                    sent_prob = sent_prob * max(0.000001, self.trans_prob[(sent[j - 1], sent[j])]) *
max(0.000001, self.pos_prob[(pos1, pos2)])
                prob_dict[''.join(sent)] = sent_prob
            self.shanglian.clear()

    prob_dict = sorted(prob_dict.items(), key=lambda item: item[1], reverse=True)
    for item in prob_dict:
        print(item[0], ' ', item[1])
    self.shanglian = prob_dict[0][0]
    return self.shanglian
```

生成结果举例如下：

春风争颂中华兴　4.120950483596436e-09
春风送暖万户明　2.575807884843403e-10
春风浩浩歌丽日　1.4802810376962125e-10
春风明示我需你　4.640979853374872e-12
春风福倍添新韵　2.595428765188698e-13

可以看到还是能够生成比较优美通顺的句子的，但是也会出现“春风明示我需你”这样不知所云的句子。由此可见，纯粹基于马尔可夫链的文本生成还是有一定的局限性。

应对下联

将上联的分词结果传入下联。但是要考虑到，用户输入的上联分词后的结果可能并不出现在训练语料之中。为了能起码对出下联，将找不到对应词的词进行切分，直接找和它对应的单字。这也就不可避免地造成了一些句子的表意不通。我考虑的是按照概率取出下联对应的字，再按照下联词之间转移的概率和上下联对应的概率挑选最大概率的下联。

```
def gen_xialian(self, raw):  # 传入上联分词结果
    dic = defaultdict(list)
    shanglian = []
    for (i, word) in enumerate(raw):
        if word not in self.cfd_refer.keys():  # 没有在训练数据中找到这个词，不得不将其拆成单字寻找对应
            chars = list(word)
            for char in chars:
                for j in self.cfd_single_refer[char].keys():
                    dic[char].append(j)
                shanglian.append(char)
        else:
            shanglian.append(word)
            for(base, refer) in self.refer_prob.keys():
                if base == word:
                    dic[base].append(refer)

    temp = []
    temp_prob = []
    for i in range(10):  # 形成十个下联，选择概率较高的
        res = []
        for word in shanglian:
            index = min(int(random.random() * len(dic[word])), len(dic[word])-1)
            res.append(dic[word][index])
        temp.append(res)
    prob_dict = defaultdict(float)
    for i in range(len(temp)):
        sent = temp[i]
        last_word = sent[-1][-1]
        if not tone(last_word):  # 下联要求是平声
            continue
        sent_prob = max(self.refer_prob[(shanglian[0], sent[0])], 0.000001)  # 对拆分的单字加以惩罚
        for j in range(1, len(shanglian)):
            sent_prob = sent_prob * max(0.000001, self.trans_prob[(sent[j-1], sent[j])]) * max(self.r
efer_prob[(shanglian[j], sent[j])], 0.000001)
        prob_dict[''.join(sent)] = sent_prob
    prob_dict = sorted(prob_dict.items(), key=lambda item:item[1], reverse=True)
    for item in prob_dict:
        print(item[0], ' ', item[1])
    return prob_dict[0][0]
```

比较好的应对效果如下：
春风轻拂千山绿 ，江山彩润万里和
春风轻拂千山绿 ，瑞气细润万事丰

春风款款温雨润 ，旭日冉冉富花香
雪里云山似 ，花间风景如
九州大地松尤绿 ，四海神州柏更苍
九州激荡舒清景 ，四海顿生展宏图
九州曙色含绿水 ，四海春光有青山
九州大展登云志 ，四海新开奋月心
九州俊彦振中兴 ，四海忠人迎富裕
梅报春百花齐放 ，鹊开道万马奔腾
飞雪片片醉 ，画图声声歌
江山双归合民意 ，社稷一震腾春风

但是发现随机挑选的话，会生成一些非常奇怪、完全不对的句子。比如：
春风轻拂千山绿的下联：
福地再润万里留　5.8791747284850106e-36
正气曙润万家丰　1.1758349456970024e-36
爱国喜暖增辉欢　1.1758349456970024e-36
喜报再祚万里歌　1.9102011753200403e-37
华夏微蒸万户流　6.349508706763813e-41
战鼓结开万事春　5.820382981200162e-41
景色喜捧万户荣　1.5873771766909532e-41
马扬厚满扬波欢　5.291257255636511e-42
阳历微耀万家浓　2.6456286278182554e-42
灵鼠东祚万事青　2.6456286278182554e-42

可见基于 HMM 模型的生成效果并不稳定，仍然有待改进。

Seq2Seq 模型

Seq2Seq 在机器翻译、文章摘要领域有着广泛的应用，主要是一个 Encoder-Decoder 框架。对对联与机器翻译有着天然的相似性，虽然对对联不像机器翻译那样有几乎唯一的正确答案，但是考虑到古汉语言简意赅、单字表意精炼的特点，可以直接构造上联到下联字与字之间的映射，完成对对联的过程。我参考了知乎专栏“基于 TensorFlow 框架的 Seq2Seq 英法机器翻译模型”[7]和“简单的 Seq2Seq 实现作对联”[8]，将其应用于对联的生成。

数据预处理

由于神经网络需要大量数据，我在 github 上找到了一个有 77 万对联的语料库[9]。由于时间和硬件条件的限制，最终选择了前 10 万条对联作为训练集，200 条对联作为验证集。

首先，将对联分为上联和下联，上联用于 Encoder 的输入，下联用于 Decoder 的输入和损失的计算。将上联文本视为 source，下联文本视为 target。

对数据的预处理主要包括：

- 根据语料构造对联文本的词典

[7] https://zhuanlan.zhihu.com/p/37148308
[8] https://www.jianshu.com/p/83443b2baf27
[9] https://github.com/wb14123/couplet-dataset

- 建立中文到词典中 index 的双向映射表
- 添加一些特殊符号：由于对联语料库中句子的长度是不确定的，而在 RNN 处理时，需要保证每个 batch 中句子的长度一致，因此需要通过<PAD>对长度不足的句子进行补全。<GO>和<EOS>用在 Decoder 端的序列，告诉 Decoder 句子的起始与结束，<UNK>则用来替代一些未出现的词或者低频词。

第一副对联处理效果如下：
['晚风摇树树还挺']
[8546, 784, 5283, 4594, 4594, 8587, 3602]
['晨露润花花更红']
[5260, 1993, 5677, 2249, 2249, 9070, 3962, 3]
对下联末尾增加了<EOS>标记（index 为 3）

模型构建

在基本的 Seq2Seq 模型中，主要包括三个部分 Encoder、Decoder 以及连接两者的 Context vector。输入一个序列，Encoder 端将序列编码到语义空间，转换为一个固定长度的向量，这个向量包含了输入序列的信息，再将这个向量作为输入传递给 Decoder 端生成新的序列，将语义向量进行解码，获得所需输出。这种机制的优点是灵活，Encoder、Decoder 端并不限制使用何种神经网络，而且是一个端到端的过程，将语义理解和语言生成合在一起。

构建 Encoder

在 Encoder 层，首先需要将输入的词进行 embedding 嵌入，转化为稠密向量，再输入给网络层进行学习。这里设置的网络层为两层 LSTM，因为 LSTM 相比 RNN 能够更有效地处理对联中单字间的长距离影响。项目中 embedding 主要用到了 tensorflow 提供的 embed_sequence，构建多层 LSTM 用到了 LSTMCell，MultiRNNCell 和 dynamic_rnn。返回的 LSTM 序列状态就是 Context vector，下一步会传递给 Decoder 端进行序列生成。

构建 Decoder

Decoder 端的输入需要经过特殊处理。在 Decoder 端，利用 Context vector 和上一轮的输出结果来生成当前阶段的输出。由于最后一个输出词并不参与到下一轮的输入中，所以应当去掉每个句子的最后一个字符<EOS>,再添加一个起始符<GO>.

同样对 Decoder 端的输入数据进行 Embedding，传给两层 LSTM，再构造一个全连接层，最后交给 Decoder 解码。

在 Decoder 端，还要注意把训练和预测所用的 Decoder 分开。在 Training Decoder 中，强制使用标准下联 target 序列作为输入传给 Decoder 端 LSTM 的每个阶段，而不是使用前一阶段的预测输出，这样就可以避免上一轮有误差的输出影响到后续所有的预测结果。这种方式被称为 Teacher Forcing 机制。而在 Predicting Decoder 中，由于无法得知真实的 target，只能利用上一阶段的输出和隐层状态来进行预测。而这两个 Decoder 是可以共享参数的。

构建 Seq2Seq

将 Encoder 和 Decoder 拼接起来，构建 Seq2Seq 模型。

模型训练

（1）数据划分：如前所述，训练集(test set)共含标准对联 10 万条，验证集（Validation Set）共含标准对联 200 条。

（2）超参设置：

```
# Number of Epochs
epochs = 500
# Batch Size
batch_size = 200
# num_units in LSTM
rnn_size = 50
# Number of Layers
num_layers = 2
# Embedding Size
encoding_embedding_size = 15
decoding_embedding_size = 15
# Learning Rate
learning_rate = 0.01
```

(3)模型的保存与恢复：由于 RNN 类的神经网络训练时间都比较长，故我每训练十个 epoch 就将模型保存一次，若训练中断，下一次也可以直接读入断点处的模型继续进行训练。

模型测试与使用

在测试阶段，首先将保存的模型进行恢复，再进行预测。对输入的上联，首先将其转换成词典中的 index 序列，并且进行必要的序列补全操作。加载好训练的图之后，喂给网络进行运算，再将输出的序列转化为汉字，即可得到下联的应对结果。

实验结果

由于硬件条件和时间限制，也不是很会调网络参数，期间几次发现 training loss 和 validation loss 不再下降调整学习率，得到了多代模型。最终选用的模型训练了 180 个 epoch，但是 loss 已经稳定在 4.014 不再下降，我猜想是因为对联本身具有局限性，语言翻译几乎是有标准答案的，对联则不然，考虑每个单字都可能有多个字可以与之相对，况且训练选用的数据集也不一定是非常工整的对仗，所以就会导致 loss 比较高。而且对联的组合千变万化，必然会出现大量训练集中从未出现的组合。最终实验结果举例如下：

上联
词典 Index: [3287, 784, 3583, 1027, 986, 3836, 2613]
上联: 春 风 轻 拂 千 山 绿

下联
词典 Index: [8310, 8867, 5233, 4923, 1974, 8595, 3287]
预测下联: 旭 日 长 歌 满 院 春

上联
词典 Index: [8088, 3446, 7375, 2688, 2447, 456, 2427]
上联: 九 州 大 展 登 云 志

下联
词典 Index: [7968, 526, 5233, 4923, 2688, 6220, 9115]
预测下联: 万 里 长 歌 展 国 猷

上联
词典 Index: [3962, 622, 3973, 1188, 2397]
上联: 红 梅 开 岁 首

下联
词典 Index: [4677, 2178, 2688, 8430, 4017]
预测下联: 紫 气 展 人 家

其实我个人感觉对的还不错，挺有创意的，虽然不是非常工整。

总结

此次课程项目由对联的文本分析出发，首先对对联的高频词汇、音韵特点和意象共现关系进行了分析，然后用 HMM 模型实现了上联的生成和下联的应对，最后利用 Seq2Seq 机器翻译框架实现了下联的应对。

但是目前的模型还很不完善，主要在于：

1. 生成上联时只是简单考虑了相邻词和相邻词性的转移概率，造成句子不够连贯，有时会生成一些不知所云的句子；HMM 模型应对下联时，无法做到围绕着一个主题去生成，总体有词汇堆砌之感。未来可以考虑采用 RNN 生成上联文本。
2. 由于对神经网络接触不多，此次我只是仿照已有框架实现了一个初步的 Seq2Seq 框架，对设置网络结构和调整网络超参数等都不熟悉，造成了训练结果 loss 较高，对部分上联的应答还不够合适。未来可以考虑在基础 Seq2Seq 模型的基础上进行模型优化，主要是采用双向 LSTM 编码层以捕获更多的特征信息，加入 Attention 机制以赋予当前字更高的权重（更加符合对仗要求），采用 BLEU 对模型应对效果加以评估。

总体说来，此次课程项目让我将课堂上学到的一些自然语言处理的技巧付诸实践，也学习到了很多文本分析、数据可视化的方法，还初步接触到了网络爬虫和神经网络的搭建训练方法。在完成整个 PJ 的过程当中，虽然我遇到了很多难点，但是最终都成功地通过查阅资料或改变方法解决了。同时我也认识到了自然语言处理的局限性，比如 jieba 对偏古文的分词并不是很支持，还有很多特殊对联可能是机器永远也无法学习到的（比如拆字联、谐音联、巧意联）。要想得到真正句意通顺、有内涵的对联是一个庞大的课题。但是仍然可以将这种生成手法作为一种辅助。

论文评语（教师填写）：

任课教师签名：

日　　期：