成　　绩 _____

北京航空航天大學
BEIHANG UNIVERSITY

深度学习与自然语言处理
第五次大作业

Seq2Seq

院（系）名称	自动化科学与电气工程学院
专业名称	电子信息
学生学号	ZY2103812
学生姓名	朱远哲
指导教师	秦曾昌

2022 年 6 月

目　　录

1 问题描述

基于Seq2seq模型来实现文本生成的模型，输入可以为一段已知的金庸小说段落，来生成新的段落并做分析。

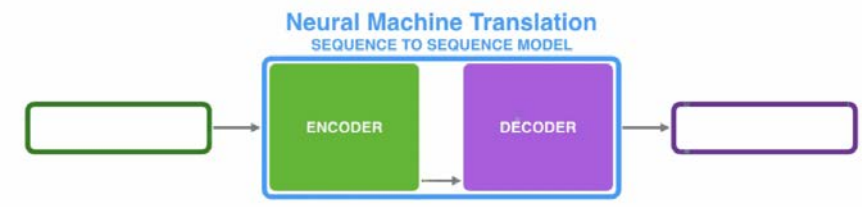
2 问题表达

2.1 seq2seq 介绍

seq2seq 是一个Encoder–Decoder 结构的网络，它的输入是一个序列，输出也是一个序列。Encoder 中将一个可变长度的信号序列变为固定长度的向量表达，Decoder 将这个固定长度的向量变成可变长度的目标的信号序列。

很多自然语言处理任务，比如聊天机器人，机器翻译，自动文摘，智能问答等，传统的解决方案都是检索式(从候选集中选出答案)，这对素材的完善程度要求很高。seq2seq模型突破了传统的固定大小输入问题框架。采用序列到序列的模型，在NLP中是文本到文本的映射。其在各主流语言之间的相互翻译以及语音助手中人机短问快答的应用中有着非常好的表现。

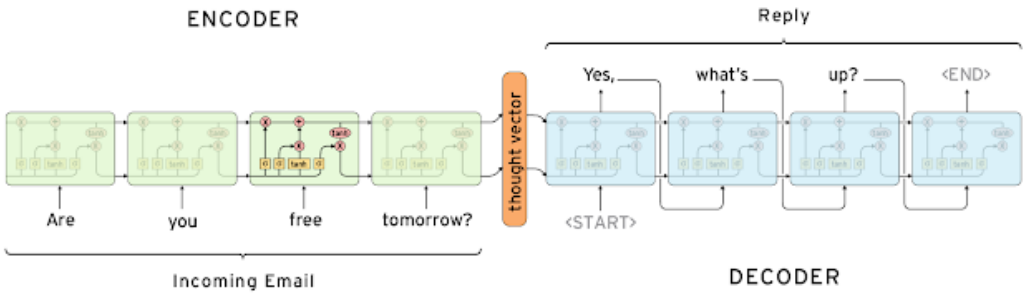
2.2 编解码器结构



Seq2Seq解决问题的主要思路是通过深度神经网络模型将一个作为输入的序列映射为一个作为输出的序列，这一过程由编码输入（encoder）与解码输出（decoder）两个环节组成。

输入的数据(文本序列)中的每个元素(词)通常会被编码成一个稠密的向量，这个过程叫做word embedding。经过循环神经网络（RNN），将最后一层的隐层输出作为上下文向量。encoder和decoder都会借助于循环神经网络(RNN)这类特殊的神经网络完成，循环神经网络会接受每个位置(时间点)上的输入，同时经过处理进行信息融合，并可能会在某些位置(时间点)上输出。

最初的Encoder-Decoder模型由两个RNN组成，后多用LSTM模型。编码器处理输入序列中的每个元素，将捕获的信息编译成向量。在处理整个输入序列之后，编码器将上下文发送到解码器，解码器逐项开始产生输出序列。



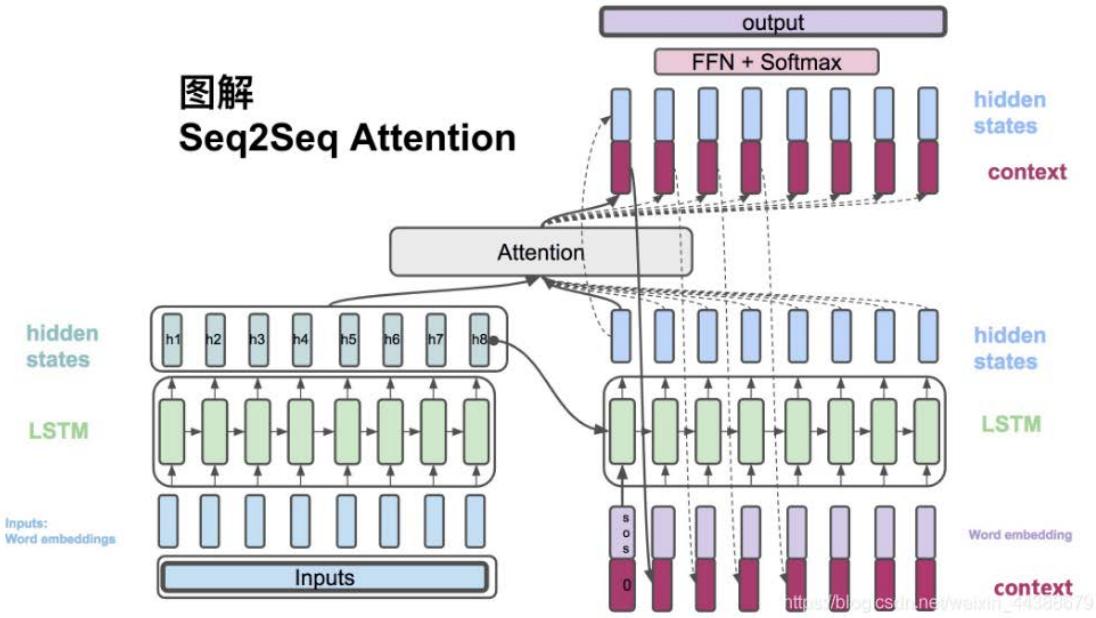
图中展示的是一个邮件对话的应用场景，图中的 Encoder 和 Decoder 都只展示了一层的普通的 LSTMCell。从上面的结构中，我们可以看到，整个模型结构还是非常简单的。 EncoderCell 最后一个时刻的状态 [cXT,hXT]就是上面说的中间语义向量 cc ，它将作为 DecoderCell 的初始状态。然后在 DecoderCell 中，每个时刻的输出将会作为下一个时刻的输入。以此类推，直到 DecoderCell 某个时刻预测输出特殊符号 <END>结束。

2.3 attention 机制

RNN模型虽然具有记忆性，但是当Encoder阶段输入序列过长时，解码阶段的RNN也无法很好地针对最早的输入序列解码。Attention注意力分配的机制被提出，就是为了解决这个问题。在Decoder阶段每一步解码，都能够有一个输入，对输入序列所有隐藏层的信息h_1,h_2,…h_Tx进行加权求和。打个比方就

是每次在预测下一个词时都会把所有输入序列的隐藏层信息都看一遍，决定预测当前词时和输入序列的那些词最相关。

所谓的注意力机制，可以粗略地理解为是一种对于输入的信息，根据重要程度进行不同权重的加权处理(通常加权的权重来源于softmax后的结果)的机制，是一个在解码阶段，简单地对编码器中的hidden states进行不同权重的加权处理的过程。



3 具体算法实现

3.1 数据处理

首先选取了相对了解较多的射雕英雄传作为输入进行分析，需要对语料库进行断句分词，使用jieba分词工具，同时对语料进行预处理，去除广告和无意义的字符，得到分词后的文件。在这一步需要注意，在一般的NLP处理中，会需要去停用词。由于word2vec的算法依赖于上下文，而上下文有可能就是停词。因此对于word2vec，我们可以不用去停词，同时加上了人名使jieba分词能够更好地将人名分出来。

```
jieba.suggest_freq('郭靖', True)
jieba.suggest_freq('黄蓉', True)
jieba.suggest_freq('杨康', True)
jieba.suggest_freq('穆念慈', True)
jieba.suggest_freq('黄药师', True)
jieba.suggest_freq('欧阳锋', True)
jieba.suggest_freq('洪七公', True)
jieba.suggest_freq('周伯通', True)
jieba.suggest_freq('柯镇恶', True)
jieba.suggest_freq('梅超风', True)
```

```
def read_data():
    with open('射雕英雄传.txt', "r", encoding='gb18030') as f:
        file_read = f.readlines()
        all_text = ""
        for line in file_read:
            line = re.sub('\s', '', line)
            line = re.sub('！', '。', line)
            line = re.sub('？', '。', line)# 保留句号
            line = re.sub('[\u0000-\u3001]', '', line)
            line = re.sub('[\u3003-\u4DFF]', '', line)
            line = re.sub('[\u9FA6-\uFFFF]', '', line)
            all_text += line
        f.close()
    return all_text
```

3.2 模型训练

在利用Jieba分词库对输入语料进行分词后，仿照第四次作业一样，利用Word2Vec模型对输入语料进行词嵌入，模型训练采用gensim中的Word2vec模型进行训练，并设置相应参数，之后利用torch框架进行LSTM神经网络训练，代码如下：

```python
def train():
    embed_size = 1024
    epochs = 20
    end_num = 10

    print("读取数据开始")
    all_text = read_data()
    text_terms = list()
    for text_line in all_text.split('。'):
        seg_list = list(jieba.cut(text_line, cut_all=False))  # 使用精确模式
        if len(seg_list) < 5:
            continue
        seg_list.append("END")
        text_terms.append(seg_list)
    print("读取数据结束")
    # 获得word2vec模型
    print("开始计算向量")
    if not os.path.exists('model.model'):
        print("开始构建模型")
        model = Word2Vec(sentences=text_terms, sg=0, vector_size=embed_size, min_count=1, window=10, epochs=10)
        print("模型构建完成")
        model.save('model.model')
    print("模型已保存")
    print("开始训练")
    sequences = text_terms
    vec_model = Word2Vec.load('model.model')
    model = Net(embed_size)
    optimizer = torch.optim.AdamW(params=model.parameters(), lr=0.0001)
    for epoch_id in range(epochs):
        for idx in trange(0, len(sequences) // end_num - 1):
            seq = []
```

```python
            seq = []
            for k in range(end_num):
                seq += sequences[idx + k]
            target = []
            for k in range(end_num):
                target += sequences[idx + end_num + k]
            input_seq = torch.zeros(len(seq), embed_size)
            for k in range(len(seq)):
                input_seq[k] = torch.tensor(vec_model.wv[seq[k]])
                target_seq = torch.zeros(len(target), embed_size)
            for k in range(len(target)):
                target_seq[k] = torch.tensor(vec_model.wv[target[k]])
            all_seq = torch.cat((input_seq, target_seq), dim=0)
            optimizer.zero_grad()
            out_res = model(all_seq[:-1])
            f1 = ((out_res[-target_seq.shape[0]:] ** 2).sum(dim=1)) ** 0.5
            f2 = ((target_seq ** 2).sum(dim=1)) ** 0.5
            loss = (1 - (out_res[-target_seq.shape[0]:] * target_seq).sum(dim=1) / f1 / f2).mean()
            loss.backward()
            optimizer.step()
            if idx % 50 == 0:
                print("loss: ", loss.item(), " in epoch ", epoch_id, " res: ", out_res[-target_seq.shape[0]:].max(dim=1).
        state = {"models": model.state_dict()}
        torch.save(state, "model/" + str(epoch_id) + ".pth")
```

3.3 模型测试预处理

在模型训练完毕后，读取测试语料，进行简单的预处理，然后对模型进行测试。

```
def read_test_data():
    name = 'test.txt'
    with open(name, "r", encoding='utf-8') as f:
        file_read = f.readlines()
        all_text = ""
        for line in file_read:
            line = re.sub('\s', '', line)
            line = re.sub('！', '。', line)
            line = re.sub('？', '。', line)
            line = re.sub('，', '。', line)# 保留句号
            line = re.sub('[\u0000-\u3001]', '', line)
            line = re.sub('[\u3003-\u4DFF]', '', line)
            line = re.sub('[\u9FA6-\uFFFF]', '', line)
            all_text += line
    return all_text
```

3.4 模型测试输出

代码输出代码及结果如下

```
def test():
    embed_size = 1024
    print("start read test data")
    text = read_test_data()
    text_terms = list()
    for text_line in text.split('。'):
        seg_list = list(jieba.cut(text_line,cut_all=False))  # 使用精确模式
        if len(seg_list) < 5:
            continue
            seg_list.append("END")
            text_terms.append(seg_list)
    print("end read data")
    checkpoint = torch.load("model/" + str(49) + ".pth")

    model = Net(embed_size).eval().cuda()
    model.load_state_dict(checkpoint["models"])
    vec_model = Word2Vec.load('model.model')

    seqs = []
    for sequence in text_terms:
        seqs += sequence

    input_seq = torch.zeros(len(seqs), embed_size).cuda()
    result = ""
    with torch.no_grad():
        for k in range(len(seqs)):
            input_seq[k] = torch.tensor(vec_model.wv[seqs[k]])
        end_num = 0
        length = 0
        while end_num < 10 and length < 2000:
            print("length: ", length)
```

```
            out_res = model(input_seq.cuda())[-2:-1]
            key_value = vec_model.wv.most_similar(positive=np.array(out_res.cpu()), topn=20)
            key = key_value[np.random.randint(20)][0]
            if key == "END":
                result += "。"
                end_num += 1
            else:
                result += key
            length += 1
            input_seq = torch.cat((input_seq, out_res), dim=0)
    print(result)
```

4 运行结果

4.1 运行结果

1. 输入段落如下：

郭靖这才醒悟，回身前跃，到了一根柱子边上。梅超风五指抓来，郭靖立即缩身柱后，秃的一声，梅超风五指已插入了柱中。她全凭敌人拳风脚步之声而辨知对方所在，柱子固定在地，决无声息，郭靖在酣战时斗然间躲到柱后，她哪里知道？待得惊觉，郭靖呼的一掌，从柱后打了出来，当下只得硬接，左掌照准来势猛推出去。两人各自震开数步，她五指才从柱间拔出。

2. 输出段落如下：

郭靖相舞得飞刀，心神大火火烧天色，桌子冲入，满以为桌子竹签羊腿海水涌摇动深入眼皮，用劲一抖，摇动用劲，转瞬间桌子山间，山间海水一抖，尖跌入衣服，涌深入跌入抬石弓，满以为涌弓，弓卫士眼皮，满以为抬石羊腿海水衣服，海水深入用劲。

4.2 结果讨论与分析

从结果可以看出，输出的文本逻辑性并不是特别好，并不能完全和输入文本对应上，但也可以看出一些相关性。其中有多方面的原因，由于代码运行速度限制，训练的迭代次数只有20次，导致模型效果不是很好。模型网络本身相对简单，本身效果也没有很理想。

5 个人总结与体会

本次大作业实现了利用seq2seq模型进行段落文本的分析，通过本次程序编写，我对python编程变得更加熟悉，同时也学习了解了LSTM和seq2seq模型，本次大作业相对最为困难，由于黑箱模型训练的不确定性和不可见性，算法效果的调整有难度，通过算法的学习程序的编写，又让我对自然语言处理有了更加深刻的理解，也通过大作业有了很大的收获。