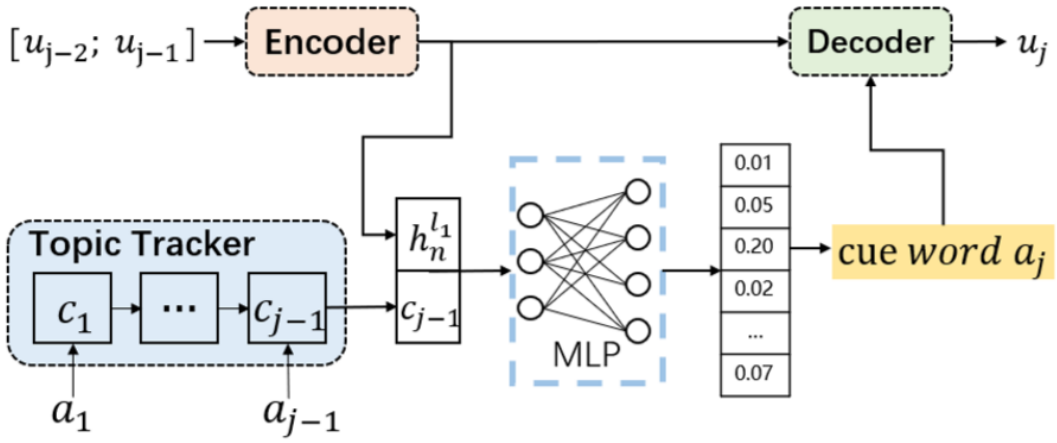
Dynamic Ce Words Planning to Flow Longer Conversations 复现报告

文章简介

这篇文章主要提出了一种通过CueWord引导生成对话的模型，文章使用了一个seq2seq模型来进行回答生成，对于这个网络他的一个输入时上下文的一组对话(query, answer)，文章模型如图



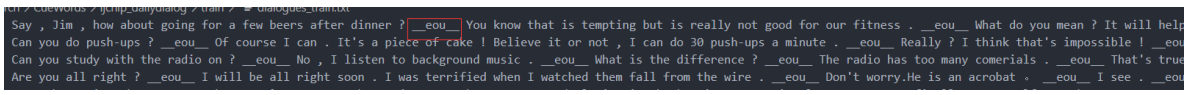
首先我们将一组对话输入到Encoder模型中，Encoder模型是一个两层的LSTM模型，他的第一层输入为t时刻的词向量和t-1时刻的隐藏层状态，第二层输入为第一层t时刻的隐藏层状态和该层t-1时刻的隐藏层状态，同时做了一个padding，由全0补充，这地方推测是为了保持encoder模型和decoder模型的输入转态相同，方便decoder模型使用encoder模型的隐藏层状态作为初始状态。

经过ecoder模型以后，我们提取第一层n时刻的的隐藏层转态和上一句话的cueword作为一个输入，经过MLP模型，然后通过Softmax得到CueWord字典中每个单词作为CueWord的概率，然后获取回答的预测CueWord、

最后通过获取的预测CueWord经过一个Decoder模型得到我们输出的句子，这个Decoder模型也是一个两层的LSTM模型，第一层t时刻输入为经过线性变换后的CueWord，包含了CueWord的信息和t-1时刻第一层的隐藏层状态，第二层t时刻的输入为第一层t时刻的隐藏层状态和t-1时刻第二层的隐藏层状态以及t-1时刻的输出。最后我们将第二层的输出经过一个softmax用来预测dict字典中每个单词出现的概率选取概率最高的单词作为这次输出。

实现过程

数据处理



如图训练数据中以**eou**作为换行符 每一行是一轮对话，所以第一步是根据**eou**进行分割分割将对话转化成多行格式

```
en.txt
Say , Jim , how about going for a few beers after dinner ?
You know that is tempting but is really not good for our fitness .
What do you mean ? It will help us to relax .
Do you really think so ? I don't . It will just make us fat and act silly . Remember last time ?
I guess you are right.But what shall we do ? I don't feel like sitting at home .
I suggest a walk over to the gym where we can play singsong and meet some of our friends .
That's a good idea . I hear Mary and Sally often go there to play pingpong.Perhaps we can make a foursome with them .
Sounds great to me ! If they are willing , we could ask them to go dancing with us.That is excellent exercise and fun , too .
Good.Let ' s go now .
All right .
```

如图，然后我们需要删除所有标点符号这里使用正则表达式删除，同时删除少于2轮的对话，保证所有对话大于等于三轮，这里使用了data_process.py中的remove函数进行处理

根据论文中所提，我们认为CueWord是出现次数最多的1000个名词，动词和形容词，所以我们需要通过nlt库对英文词性进行处理，我们分析出词性以后将单词还原成原型，这样对于同一种词根的单词就不会多次统计，然后我们统计出出现次数最多的999个单词作为潜在CueWord，对于一句话我们统计其中的每个单词,在CueWord列表中出现的单词中词书出现最多的我们作为这句话的CueWord，如果没有这样的单词，那么我们定义EPT作为他的CueWord，但是这样的EPT不能超过1000个，这里的EPT理解为，上文可能是一种结束话题的语句，所以下文可以随意说话。如图 将最后一个单词作为这句话的CueWord

```
Say Jim how about go for a few beer after dinner few
You know that be tempt but be really not good for our fitness know
What do you mean It will help us to relax help
Do you really think so I don t It will just make us fat and act silly Remember last time think
I guess you be right But what shall we do I don t feel like sit at home don
I suggest a walk over to the gym where we can play singsong and meet some of our friend meet
That s a good idea I hear Mary and Sally often go there to play UNK Perhaps we can make a UNK with them good
Sounds great to me If they be willing we could ask them to go dance with us That be excellent exercise and fun too great
Good Let s go now Let
All right right
```

这一部分的实现主要使用了NLT库的词性分析和词性还原，使用data_process.py中read_en_train_data进行处理

最后我们还需要使用gensim模型进行训练,使用word2vec_train.py即可，这里设置min_count=5,词向量大小为600

数据加载

最后我们需要将数据对其转为为可以训练的数据，对于Single-turn Superised Learnings 我们首先依次处理每一行，对于有上下文的我们将两句话放到一组，先对query处理如果不足44个单词，在句首padding 0补齐并转化成词向量，再对answer处理，如果不足22个单词，在句尾padding 0不起 （实际验证中这样效果比较好), target使用ID表示即可，load.py中load_cue_word_data进行处理。

另一方面对于对话部分训练，我们将所有对话先只取padding后的前三轮保证他对其，然后传入训练模型，使用load.py中load_data进行处理

训练部分

首先时模型部分，因为我们对decoder模型需要做多次Sampling所以将Decoder模型和其他部分分割，使用了网络一个是CueWrodSelect网络包括了Encoder模型和MLP选择CueWord模型，另一部分是Decoder网络包括了Decoder模型

CueWordSelcetNetbaokuol两个LSTMCell层作为Encoder模型，2个线性层作为MLP模型

```
    def __init__(self, input_size = 600, hidden_size = 1000):
        '''
        :param input_size: 词向量长度
        :param hidden_size: 隐藏层大小
        '''
        self.hidden_size = hidden_size
        self.input_size = input_size
        super(CueWordSelectNet, self).__init__()
        self.encoder1 = nn.LSTMCell(input_size + hidden_size, hidden_size, bias=True)
        self.encoder2 = nn.LSTMCell(input_size + hidden_size + hidden_size,
hidden_size, bias = True)
        self.layer1 = nn.Linear(hidden_size + 1000,4000)
        self.layer2 = nn.Linear(4000,1000)
```

DecoderNet中包括了两个LSTMCell层作为Decoder模型，2个线性层，一个作为对CueWord的线性变化函数，另一个作为输出层将隐藏层转台隐射到和字典大小相同的网络层中去方便softmax选取答案

```
    def __init__(self, input_size = 600, hidden_size = 1000):
        '''
        :param input_size: 词向量长度
        :param hidden_size: 隐藏层节点数
        '''
        super(Decoder, self).__init__()
        self.hidden_size = hidden_size
        self.input_size = input_size
        self.layer = nn.Linear(input_size, hidden_size, bias=True)
        self.decoder1 = nn.LSTMCell(hidden_size + hidden_size, hidden_size, bias=True)
        self.decoder2 = nn.LSTMCell(input_size + hidden_size + hidden_size,
hidden_size, bias=True)
        self.linear = nn.Linear(hidden_size, dict_size)
```

需要注意的是，我们再向decoder模型中decoder2LSTMCell层输入之前需要将输出的[dict_size]大小的输出层状态提取出选择的单词，然后用word2vec将其转化成对应的词向量进行输入，否则效果会大幅度下滑。

具体训练部分，我们首先读入所有数据，然后按照8:1:1的大小分割为训练集，验证集和测试集，使用Adam优化器，对于第一轮自监督学习我们使用交叉熵损失函数，第二轮学习我们使用自定义的损失函数和梯度更新方式，如图其主要是通过SMN得到对话评分作为一个系数，然后通过CueWord向量的相似度得到主题的评分最后加权用来评估一组对话效果的好坏，最后用于更新CueWord的选取部分，并没有更新对话训练部分，所以我理解这部分其实重点在于优化CueWord选取部分的正确率，如果CueWord选取的正确率已经很高的话这部分的意义不是很大

$$\nabla_\theta(\mathcal{J}) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{1}{L_i}\sum_{j=1}^{L_i}\nabla_\theta \log p(a_{i,j}|s_{i,j}) \cdot \left(\sum_{k=j}^{j+T-1} r_{i,k}\cdot\gamma^{k-j} - b_{i,j}\right)$$

一些训练过程中处理的细节：

- 单轮自监督学习中，设置第一个网络的学习率为0.0001，第二个网络的学习率为0.002，此时效果比较好
- CueWordSelcet部分特别容易训练，一般1~2轮以后就可以达到90%的准确率，此时如果继续训练会导致过拟合现象，但是DecoderNet比较难以训练往往需要多轮训练，所以我设置第3轮以后不进行CueWordSelect部分的梯度更新，只更新DecoderNet部分，这样就避免了CueWordSelect准确率的下降
- 尝试没经过一段时间后下调学习率，避免出现学习率过大导致无法收敛的原因，但是实际效果不佳，可能是处理方式不得当
- 在单轮自监督学习中，为了方便交叉熵函数计算，输出的时候将一个batch的所有输出压缩到一个一维向量和target进行交叉熵计算，可以减少代码复杂度
- 两个网络学习，要避免梯度重复更新，使用detach函数隔离梯度关系
- 一开始有过训练结束后，测试结果对于所有对话输出同一个单词的现象，查阅相关资料后发现，训练的时候使用的是batch_size = 64，而单独测试的时候使用了batch_size = 1,这可能他使用了之前训练的参数的一个均值来进行计算，导致计算结果趋向平均，改为测试集batch_size = 64恢复正常。
- RL训练部分一直效果不佳，一方面可能是因为SMN模型因为确实数据无法预训练，导致评分结果只是用CueWord相似度导致评分结果不佳，另一方面可能因为CueWordSelect的准确率已经较高（95%）导致合理参数难以调整

训练结果

```
epoch 99 , valicorredt 0.9303
find
Where be you go to find one find
['Where', 'Four', 'go', 'go', 'find', 'find', 'find', 'find', 'work', 'work', 'work', 'work', 'work', 'work', 'work', 'work', 'work', 'work', 'work', 'work', 'work']
have
I have no idea have
['no', 'have', 'no', 'no', 'no', 'no', 'no', 'size', 'size', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no', 'no']
know
Do you want to know where I buy mine know
['Do', 'you', 'entitle', 'where', 'where', 'where', 'where', 'where', 'where', 'where', 'where', 'where', 'tell', 'tell', 'tell', 'tell', 'tell', 'tell', 'tell', 'tell', 'tell']
get
Where do you get it from get
['it', 'from', 'Mr', 'get', 'it', 'from', 'IKEA', 'IKEA', 'IKEA', 'Help', 'Help', 'per', 'Help', 'car', 'old', 'old', 'old', 'old', 'old', 'old', 'taxi', 'taxi']
get
I get it from IKEA get
['it', 'get', 'it', 'from', 'IKEA', 'IKEA', 'IKEA', 'IKEA', 'IKEA', 'IKEA', 'IKEA', 'IKEA', 'IKEA', 'IKEA', 'IKEA', 'IKEA', 'IKEA', 'IKEA', 'IKEA', 'IKEA', 'IKEA']
much
How much do it cost you much
['How', 'much', 'do', 'cost', 'cost', 'cost', 'cost', 'cost', 'cost', 'tell', 'tell', 'tell', 'him', '50', '50', '50', '50', '50', '50', '50', '50', '50']
```

如图第一行是预测的CueWord，第二行是标准输出，第三行是我的输出，可以看到已经能够对上一部分，目前训练100轮，效果是每句话大致能又一半多的字符对应，但是尚不能单独成句，推测需要继续训练或者调整参数

```
['Help', 'will', 'll', 'just', 'just', 'minute', 'minute', 'use', 'use', 'use', 'minute', 'minute', 'minute', 'minute', 'minute
test correct 0.9587
```

目前最好的CueWordSelect模型准确率能达到95.87%。

可能改进点

1. 一方面我认为文章生成对话的瓶颈可能是CueWord的选取，论文中默认使用出现次数最多的名词、动词或者形容词从直观上感觉到是不对的，我们有太多的常用语，他可能只是一个修饰作用，比如Do作为一个常用动词可能出现词书比较多，但是往往不是重要的CueWord，所以可能CueWord选取策略本身标注的部分就不是很准确，
2. 比如一句话中表现强烈情感的单词可以作为CueWord，所以我们可以借用情感判断的模型找到对一句话情感影响最大的那个单词，认为这个单词作为CueWord
3. 另一方面，注意到中国人说话重点放在后面，而外国人说话重点放在前面所以我们可以限制CueWord出现的范围比如前10个词，这样也可以进一步提升CueWord的实际准确率
4. 除此以外我认为可以抓取形容词和副词出现的前后的名词，以及上下位重复提到的名词或者动词作为CueWord
5. 另一方面文章使用简单的两个LSTM实现encoder和decoder，可以更换方式比如使用Transformer进行对话训练
6. 最后对于SMN部分，因为没有数据无法训练这一部分，用来评估对话相似度，我认为可能可以通过一个LSTM模型生成一个句子特征向量，然后认为属于相邻对话的特征向量相似度会比较高，把他们放到一起去作为输入期望能够将他们分到以累，然后使用相量之间的cos相似度作为评分