第八天

2019年6月17日 9:32

seq2seq的优化

1. teacher forcing机制：
    a. 在seq2seq中避免一步错，步步错的情况
    b. 把真实值作为下一步的输入，能够加快模型的训练速度
    c. 在输出确定的情况下：可以在循环中使用teacher forcing
    d. 如果输出不确定，应该在循环外使用

2. Attention
    a. 初始化decoder的隐藏状态Z_i
    b. 和encoder每个时间步进行计算，得到的结果进行softmax，转化为概率
    c. 概率和encoder每个时间步记性相乘相加，得到C_i
    d. C_i作为decoder的输入，得到输出
3. attention的计算：
    a. 计算attention weight
    b. context vector
    c. attention result
4. badhanau 和Luong attention的区别
    a. 计算attention结果的位置不同
        i. baha 在decoder的每个时间步之前计算attention的结果，bahd使用的双向GRU计算attention的结果
        ii. Luong 在decoder每个时间步之后计算attention的二级果，使用单向多层GRU
    b. attention weight的区别
        i. bahd： V* tanh（Wzi-I + Uh_j）
        ii. luong:
            1）general :进行矩阵变换后进行矩阵乘法
            2）dot：对应位置相乘
            3）concat：
5. beam search
    a. decoder中的用来优化预测结果的
    b. decoder中
        i. 输入sos，得到多个个输出
        ii. 选择其中的概率最大的beam width个保留，分别作为下一次的输入

iii. 从所有的输出中选择概率最大的beam width个保留，分别作为下一次的输入
iv. 重复2-3步骤，知道到达max_len，获取到达EOS

6. 梯度裁剪
   a. 限制梯度的大小，最终抑制梯度爆炸
   b. nn.utils.clip_grad_norm_(model.paramters(),5)
7. 模型的优化方案：
   a. 参数的初始化
   b. 优化现有的数据，语料
      i. 数据进行清洗
         1）标点、表情、外文的处理
         2）把时间、人名、地点等名词替换成对应的各自符号
      ii. 从不同角度，不同复杂程度去准备语料
         1）角度：天气，吃饭，性别
         2）复杂度：简单、一般、复杂
   c. 工程的角度出发优化
      i. 使用模板，对常见的问题进行匹配，返回预设的答案
      ii. 使用分类模型，进行对问题的分类，返回预设的答案
      iii. 使用搜索模型，从现有的语料库中，返回相似问题对应的答案

问答机器人：

1. 实现逻辑
   a. 问题处理
   b. 相似问题的召回（可能相似的前K个问题）
   c. 相似问题的排序（把具体的相似度进行排序）
2. 问题的处理
   a. 基础的处理（清理）
   b. 实体识别，判断用户的问题中主语（用主语来过滤结果）
   c. 获取句子向量（计算相似度）
3. 相似问题的召回
   a. 海选，可能相似的结果
   b. 海选之后进行排序，速度更快
   c. 相似度计算的方法
      i. 在对应的主题中进行召回
      ii. 对数据进行聚类，在某一个类别中进行相似度的计算
   d. 思考：没有考虑词语的顺序?
4. 相似问题的排序
   a. 使用深度学习建立模型，输入用户的问题，和召回的问题，返回相似度

b. 思考：
    i. 数据的来源
        1) 抓取的百度知道
        2) 手动构造
    ii. 模型如何构建
        1) 孪生神经网络
        2) embedding+lstm

召回的流程

1. 准备数据
    a. 存到本地
    b. 存到数据库
2. 问题和语料转化为向量
    a. 使用tfidf
3. 计算相似度
    a. Pysparnn

```
# 1、 原始数据构造索引
cp = ci.MultiClusterIndex(features_vec, data)

#2. 索引中传入带搜索数据，返回结果
cp.search(search_features_vec, k=1, k_clusters=2, return_distance=False)
```

pysparnn的原理

1. 簇修建，简单的聚类后从个类别中进行数据的搜索，提高效率
2. 步骤
    a. 数据预处理：
        i. 随机选择根号N个样本作为learder
        ii. 剩下的数据每个找到和她最相近的leader，作为一簇
    b. 查询
        i. 计算leader和问题q的相似度，找到最相近的leader
        ii. 再计算q和leader中最相似的K个结果，返回
    c. 优化：
        i. 每个follower属于多个leader
        ii. 每次查询，查询最相似的多个leader
3. BM25的原理
    a. bm25 是一种最佳匹配方法
    b. 让词语的重要程度随着数量的增加而衰减
    c. 对句子的长度进行归一化，计算的结果受到句子长度的影响会变弱

d. bm25(i) = tf*中间项*idf
e. 中间项= (k+1)tf/tf + k(1-b+b d/avdl)