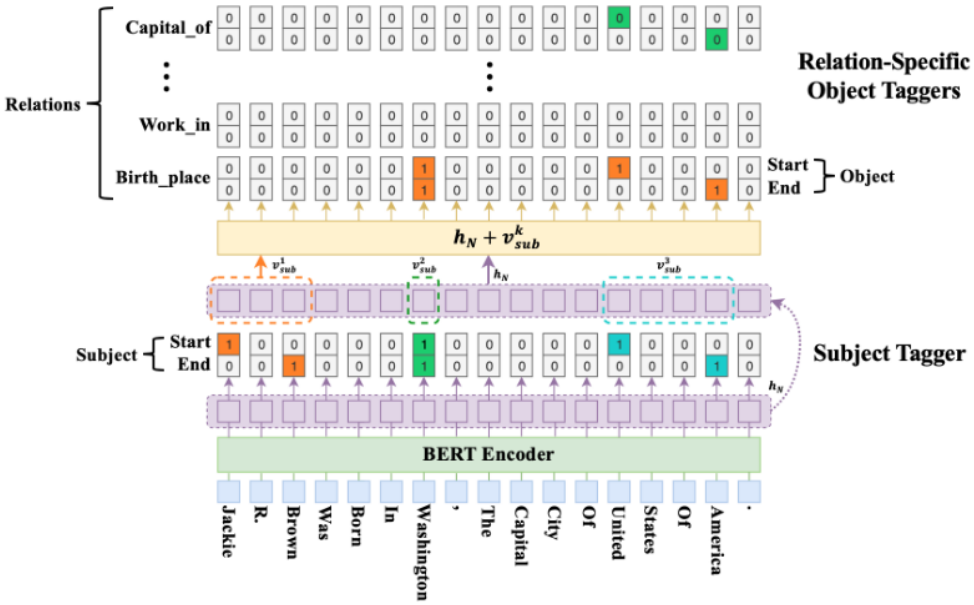
关于CasRel模型subject与object匹配问题的勘误

这两天我在跑根据CasRel Pytorch Reimplement改写的适用于CMeIE的数据集, 但是我发现怎么训练都无法达到原作者给出的性能标准, 于是我又仔细地研究了一下原代码, 然后发现有关subject和object的匹配问题我之前理解的有偏差(因为他有两个长得很像的标识符Orz, dbq555), 我之前一直以为训练和测试时的过程是一样的, 实际上并不是, 为了方便验证我的理解, 以下引用上面别人写的代码进行解释.

模型



训练

抽取subject的过程是没有问题的, 利用subject抽取object的过程本身也是没有问题的, 问题出在sub-mapping到底是什么, 下面是已知subject的情况下抽取object的函数.

```
    def get_objs_for_specific_sub(self, sub_head_mapping, sub_tail_mapping,
encoded_text):
        # [batch_size, 1, bert_dim]
        sub_head = torch.matmul(sub_head_mapping, encoded_text)
        # [batch_size, 1, bert_dim]
        sub_tail = torch.matmul(sub_tail_mapping, encoded_text)
        # [batch_size, 1, bert_dim]
        sub = (sub_head + sub_tail) / 2
        # [batch_size, seq_len, bert_dim]
        encoded_text = encoded_text + sub
        # [batch_size, seq_len, rel_num]
        pred_obj_heads = self.obj_heads_linear(encoded_text)
        pred_obj_heads = torch.sigmoid(pred_obj_heads)
        # [batch_size, seq_len, rel_num]
        pred_obj_tails = self.obj_tails_linear(encoded_text)
        pred_obj_tails = torch.sigmoid(pred_obj_tails)
        return pred_obj_heads, pred_obj_tails
```

不考虑batch的情况下, 这里sub_head_mapping是一个记录subject的head信息的向量(长度为seq_len), 它在subject的头的索引处为1, 其他位置为0, 同理object亦然. 按照我之前的讲解, sub_head_mapping似乎应该是一个序列的所有subject信息都包含的一个向量(如果一个序列subject不止一个, 那么应该把所有的head处都标1), 但事实不是这样的, sub_head_mapping仅包含一个subject的信息, 这个subject是从subject集合随机选取的.

```
sub_head_idx, sub_tail_idx = choice(list(s2ro_map.keys()))
sub_head, sub_tail = np.zeros(text_len), np.zeros(text_len)
sub_head[sub_head_idx] = 1
sub_tail[sub_tail_idx] = 1
obj_heads, obj_tails = np.zeros((text_len, self.config.rel_num)),
np.zeros((text_len, self.config.rel_num))
for ro in s2ro_map.get((sub_head_idx, sub_tail_idx), []):
    obj_heads[ro[0]][ro[2]] = 1
    obj_tails[ro[1]][ro[2]] = 1
```

s2ro_map是记录了一个序列的所有subject->relation&object映射信息的map, 我们在训练时使用的sub_head_mapping是使用choice函数随机从subject集合中选取的, 也就是说sub_head_mapping**最多只会有1个位置为1**, sub_tail_mapping亦然, 我的理解这样做是为了使训练过程和测试过程尽量一致, 因为在测试时我们使用的subject_head_mapping和sub_test_mapping也只包含一个subject的信息.

于是obj_head和obj_tail也仅仅包含所选中的subject所对应的全部objec信息(一个subject可能对应多个object).

测试

```
if subjects:
    triple_list = []
    # [subject_num, seq_len, bert_dim]
    repeated_encoded_text = encoded_text.repeat(len(subjects), 1, 1)
    # [subject_num, 1, seq_len]
    sub_head_mapping = torch.Tensor(len(subjects), 1,
encoded_text.size(1)).zero_()
    sub_tail_mapping = torch.Tensor(len(subjects), 1,
encoded_text.size(1)).zero_()
    for subject_idx, subject in enumerate(subjects):
        sub_head_mapping[subject_idx][0][subject[1]] = 1
        sub_tail_mapping[subject_idx][0][subject[2]] = 1
        sub_tail_mapping = sub_tail_mapping.to(repeated_encoded_text)
        sub_head_mapping = sub_head_mapping.to(repeated_encoded_text)
        pred_obj_heads, pred_obj_tails =
model.get_objs_for_specific_sub(sub_head_mapping, sub_tail_mapping,
repeated_encoded_text)
```

我记得之前我说的是一下把所有的subject和所有的object识别出来然后进行两两完全匹配(dbq555), 实际上, 模型确实会对每个subject单独抽取对应的object, 模型的过程是这样的: 我们已经对subject做了预测得到一个sub_list, 对sub_list的每一个subject我们都得到一个sub_head_mapping和sub_tail_mapping, 然后我们利用仅包含唯一subject的mapping信息抽取object和relation, 这样一个句子可能有多个subject, 每个subject对应的**object和relation的抽取都是仅利用当前subject且独立的**, 这样就能比较好的解决关系嵌套问题.

这样做的一个问题在于不能一个batch一个batch测试, 但是因为是测试, 也没有太多影响.

总结

训练时sub_mapping的信息仅包含一个subject(从subject_list中随机选取), 测试时对每个可能的subject都得到对应的sub_mapping进行object和relation预测.

训练时仅选取一个subject的理由我感觉有: 1.因为预测时一次仅选择一个subject, 保证一致 2.在epoch比较大的情况下其实每个subject都比较均匀的参与了训练(大概)

值得注意的是, 作为标准标注计算损失的sub_head和sub_tail依然是包含全部subject信息的01标注序列, 但是作为标准标注用于计算损失的object_head和obj_tail仅仅包含随机选中的subject对应的object.

结果

说实话导致我之前训练效果差的原因并不是这个, 而是因为我的bert没有微调Orz, 我把bert参数的梯度打开之后就只能以很小的batch_size训练, 但是依然基本达到了上面那个github中声称的性能(原作者F1为47.59)

CMeIE-P: 59.274
CMeIE-R: 38.419
CMeIE-F1: 46.621

代码已经更新CasRel-Sherlocoder, 发现啥问题或者有啥我没说明白的可以私信XD