自然语言处理大作业三

519030910383、*519030910379*、*519030910370*、*519030910393* 张哲昊、王子龙、赵峻图、宋思晓

1. 摘要

语音是口语对话系统中最主要的输入，语音识别模块可以将音频输入转化为对应的文字信息，而接下来就是对话系统中最重要的操作之一，即语义理解，也就是获得用户的意图，为后续处理提供信息。在本次任务中，我们在老师和助教提供的基于序列标注方法的 baseline 模型的基础上，提出了自己的改进方法来解决语义理解的问题。

2. 方法

在这里我们使用的是 BIO 模型的序列标注算法，通过为每个中文字符打上标签来区分各个字符的所属片段。为了简化模型，这里并没有选择 BIEO 的方式进行标注，也就是少了片段结尾“end”这一项。

2.1. 预训练模型

在对算法进行改进时，我们首先替换了实验所使用的 word2vec 词嵌入，而是使用了多种 huggingface 中的预训练模型进行词嵌入。

```
self.pre_model = AutoModelForMaskedLM.
    from_pretrained(self.config.pre_path)
out_1 = self.pre_model(input_ids,
    output_hidden_states=True)
hidden = out_1.hidden_states
ebedding = torch.cat(hidden[-i:], dim=2).detach()
```

我们使用了 Bert-base-Chinese、Chinese-bert-wwm-ext、Chinese-electra-180g-base-discriminator 以及 gpt2-base-Chinese 四种与训练模型进行实验比较结果。（以下分别简称 BBC、CBWE、CEBD、GBC）

2.2. 噪声处理

我们知道在中文系统中，发音偏差问题、口音问题（在这里我们将其统一称为噪音问题）是非常普遍的，比如有些方言会将”北京“二字读成”背景“。此时如果需要机器能够正确识别用户的意图，最关键的一项步骤就是要设计相应的去噪算法来处理这些噪声问题。对此我们使用的去噪算法如下所示：

```
def projection(self, slot, val):
    if val in self.entities[slot] or not val:
        return val
    minind = None
    mindis = 1000000, 200
    inv_dict = self.invert_docs[slot]
    for v in set.union(*(inv_dict[ch] for ch in
    val)):
        tmp = lev(v, val), abs(len(v) - len(val))
        if tmp < mindis:
            mindis = tmp
            minind = v
    if mindis >= (0.45*len(val), 20):
        return None
    return minind

def lev(a,b):
    n, m = len(a), len(b)
    if n > m:
     a, b = b, a
     n, m = m, n
    current = range(n + 1)
    for i in range(1, m + 1):
     previous, current = current, [i] + [0] * n
     for j in range(1, n + 1):
      add, delete = previous[j] + 1, current[j -
    1] + 1
      change = previous[j - 1]
      if a[j - 1] != b[i - 1]:
       change = change + 1
      current[j] = min(add, delete, change)
    return current[n]
```

这里我们引入了字符编辑距离的概念，对于那些带有噪音的输入，由于我们在给定的词表中找不到对应的字符，所以我们要根据我们设定好

的算法来计算目标词与词表中各词之间的“距离”，找到词表中与目标词距离最近的词进行替换之后，再进行后续的操作。

2.3. 尝试与思考

2.3.1. Transformer 替换 RNN

尝试用 Transformer 的 encoder 层替换 RNN 示例代码如下：

```
encoder_layer=nn.TransformerEncoderLayer(d_model=512,nhead=8)
transformer_encoder=nn.TransformerEncoder(encoder_layer,num_layers=6)
```

但是在实际实现过程中由于使用 rnn_utils.pack_padde_sequence 对数据进行整理，该函数的输出格式不符合上述 nn.TransformerEncoderLayer 的输入格式。由于时间关系，没有进一步实现。

2.3.2. 对话历史的使用

尝试利用 RNN 在对话历史训练的隐层作为辅助特征，对当前序列的预测进行辅助，但实验效果不佳。

3. 实验结果

首先我们展示的是没有进行改进的，使用不同的 encoder cell 的实验结果，如表1所示：

表 1: 原算法的结果

Encoder cell	Dev acc	Fscore
LSTM	74.92	79.27
GRU	75.24	79.52
RNN	73.83	77.71

接着我们使用比较常用的 LSTM 作为我们的 Encoder cell 来运行我们的改进算法，我们分别实验了只加入去噪算法和同时加入去噪算法并更换预训练模型的情况，实验结果如表2所示（见下一页）：

表 2: 改进算法实验结果对比

	Dev acc	Fscore
原模型	74.92	79.27
去噪	83.55	88.41
BBC	**84.69**	**89.51**
CBWE	83.50	87.91
CEBD	81.43	86.33
GBC	81.98	84.38

4. 分析与总结

在本次大作业中，我们查阅各种资料，通过替换预训练模型以及加入我们的去噪算法，对老师和助教提供的 baseline 模型进行了改进，从实验结果中我们可以看到，我们的去噪算法是比较成功的，将验证集准确率提高了接近 9 个百分点。而在预训练模型方面，我们发现使用 Bert-base-Chinese 预训练模型的效果还是非常不错的，而其他几个预训练模型效果不尽如意，效果不如原来的 word2vec 模型。从整体上看我们的改进算法还是相对成功的。

5. 参考与致谢

参考博客 https://www.jianshu.com/p/e4aca5e9e650 加载预训练模型。

感谢时若曦同学的讨论与帮助。

6. 小组成员分工

张哲昊：实验设计与代码实现

王子龙：实验设计与代码实现

赵峻图：问题调研与报告撰写

宋思晓：问题调研与报告撰写