知识表示与推理课程

命名实体识别实验报告

姓名：徐柯炎

学号：2021110683

日期：2024.5.28

一、 基于 ltp 的命名实体识别

（1. 简述命名实体识别相关概念。）

命名实体识别(Named Entity Recognition，简称 NER)是自然语言处理(NLP)中的一项重要任务，旨在从文本中识别并分类具有特定意义的实体，如人名、地名、组织机构、日期、时间、百分比、货币等。NER 的目标是识别文本中与特定类别相关的命名实体，并为这些实体分配预定义的类别标签。

NER 系统通常采用监督学习方法，使用带有标签的训练数据进行模型训练。训练数据包含了文本句子和对应的实体标签，模型通过学习这些示例来预测新的文本中的实体和它们的类别。

（2. 更改例句，使用 ltp 库完成命名实体识别，分析对应识别结果。）

这里采用的例句为：“裴友生，男，汉族，湖北蕲春人，1957 年 12 月出生，大专学历。”

结果如下：

```
Loading weights from local directory
[[('Nh', '裴友生'), ('Ns', '湖北')]]
```

分析识别结果：“裴友生”是人名，“湖北”是地名，分析结果基本正确。

二、 基于 bert 的命名实体识别

（1. 补全各部分代码，完成手册中三部分实验内容，在报告中以截图方式展示补全代码。）

Process.py

```
        ## todo 定义相关文件初始地址
        self.data_path = "./duie_data/"
        self.train_file = os.path.join(self.data_path, "train.json")
        self.dev_file = os.path.join(self.data_path, "dev.json")
        self.test_file = os.path.join(self.data_path, "test.json")
        self.schema_file = os.path.join(self.data_path, "duie_schema.json")
```

```
                ## todo 基于模版文件 获取subject_type
                subject_type = data.get("subject_type")
                ## todo 基于模版文件 获取object_type
                object_type = data.get("object_type").get("@value")
                if "人物" in subject_type:
                    subject_type = "人物"
                if "人物" in object_type:
                    object_type = "人物"
```

```
                    try:
                        # todo 找到subject在文本中的位置
                        subject_re_res = re.finditer(re.escape(spo['subject']), line['text'])
                        subject_type = spo["subject_type"]
                        if "人物" in subject_type:
                            subject_type = "人物"
```

```
                    try:
                        # todo 找到object在文本中的位置
                        object_re_res = re.finditer(re.escape(spo['object']['@value']), line['text'])
                        object_type = spo['object_type']['@value']
                        if "人物" in object_type:
                            object_type = "人物"
```

```
## todo 补全训练集和验证集的处理结果保存地址
processDuieData.get_ner_data(processDuieData.train_file,
                             os.path.join(processDuieData.data_path, train_path))
processDuieData.get_ner_data(processDuieData.dev_file, os.path.join(processDuieData.data_path, dev_path))
```

model.py

```
## todo 获取批量大小
batch_size = seq_out.size(0)
seq_out, _ = self.bilstm(seq_out)
seq_out = seq_out.contiguous().view(-1, self.lstm_hiden * 2)
seq_out = seq_out.contiguous().view(batch_size, self.max_seq_len, -1)
## todo 使用线性层进行最后的预测
seq_out = self.linear(seq_out)
logits = self.crf.decode(seq_out, mask=attention_mask.bool())
```

main.py

```
            # todo 梯度清零
            self.optimizer.zero_grad()
            # todo 反向传播计算梯度
            loss.backward()
            # todo 更新参数
            self.optimizer.step()
            # todo 更新学习率
            self.schedule.step()
            print(f"【train】{epoch}/{self.epochs} {global_step}/{self.total_step} loss:{loss.item()}")
            global_step += 1
            if global_step % self.save_step == 0:
                torch.save(self.model.state_dict(), os.path.join(self.output_dir, "pytorch_model_ner.bin"))
```

```
# todo 基于NerDataset 加载数据集
train_dataset = NerDataset(train_data, args, tokenizer)
dev_dataset =  NerDataset(dev_data, args, tokenizer)
train_loader = DataLoader(train_dataset, shuffle=True, batch_size=args.train_batch_size, num_workers=2)
dev_loader = DataLoader(dev_dataset, shuffle=False, batch_size=args.dev_batch_size, num_workers=2)
```

data_loader.py

```
    ## todo 补全input_ids至最大长度
    input_ids = tmp_input_ids + [self.tokenizer.pad_token_id] * (self.max_seq_len - len(tmp_input_ids))
    ## todo 补全attention mask至最大长度
    attention_mask += [0] * (self.max_seq_len - len(attention_mask))
    labels = [self.label2id[label] for label in labels]
    labels = [0] + labels + [0] + [0] * (self.max_seq_len - len(tmp_input_ids))
```

（2.　对于训练部分给出训练成功截图，并分析模型训练代码逻辑。）

训练代码逻辑如下：

① 导入预先设置的超参数，并将超参数写入 ner_args.json 中，方便后续调用；

② 导入 tokenizer 并且指定使用的模型，设置使用的 device（不能用 cuda 就使用 cpu 运算）；

③ 准备使用的数据集：读入 train.json 和 dev.json，分别用于训练和测试，然后使用 NerDataset 类加载数据集，并通过 DataLoader 导入；

④ 准备 optimizer 和训练策略（在训练的过程中调整学习率）；

⑤ 然后通过 Trainer 类来加载训练模型；

⑥ 开始训练：计算模型输出，然后反向传播并更新参数和学习率，计算损失函数；

⑦ 最后测试模型。

训练成功截图：

```
【train】1/1 821/834 loss:8.726783752441406
【train】1/1 822/834 loss:14.25103759765625
【train】1/1 823/834 loss:8.69035816192627
【train】1/1 824/834 loss:11.895206451416016
【train】1/1 825/834 loss:6.9027099609375
【train】1/1 826/834 loss:18.70758819580078
【train】1/1 827/834 loss:8.050189018249512
【train】1/1 828/834 loss:5.962691307067871
【train】1/1 829/834 loss:7.127409934997559
【train】1/1 830/834 loss:10.782281875610352
【train】1/1 831/834 loss:7.0029296875
【train】1/1 832/834 loss:9.351893424987793
【train】1/1 833/834 loss:9.072681427001953
【train】1/1 834/834 loss:15.364395141601562
```

	precision	recall	f1-score	support
Date	0.80	0.87	0.83	128
Number	0.70	0.83	0.76	23
Text	0.58	0.66	0.62	61
人物	0.79	0.92	0.85	1589
企业	0.71	0.50	0.58	143
企业/品牌	0.00	0.00	0.00	4
作品	0.00	0.00	0.00	12
国家	0.83	0.65	0.73	68
图书作品	0.70	0.80	0.75	179
地点	0.58	0.37	0.45	30
城市	0.00	0.00	0.00	5
奖项	0.50	0.07	0.12	14
学校	0.67	0.79	0.73	67
影视作品	0.75	0.81	0.78	281
文学作品	0.00	0.00	0.00	5
景点	0.00	0.00	0.00	2
机构	0.52	0.69	0.59	112
歌曲	0.85	0.80	0.82	172
气候	0.67	0.67	0.67	3
电视综艺	0.70	0.96	0.81	27
行政区	0.55	0.75	0.63	8
语言	0.00	0.00	0.00	1
音乐专辑	0.71	0.91	0.79	32
micro avg	0.76	0.83	0.79	2966
macro avg	0.50	0.52	0.50	2966
weighted avg	0.75	0.83	0.78	2966

（3. 提供预测部分代码，并给出手册中例句的命名实体识别结果。）

部分预测代码如下：

```
test_data = []
str_list = ['《民航客运服务会话》是1995年中国民航出版社出版的图书，作者是周石田',
            '再有之后的《半生缘》，蒋勤勤饰演的顾曼璐完全把林心如的曼桢衬得像是涉世未深的小姑娘，毫无半点风情',
            '裴友生，男，汉族，湖北蕲春人，1957年12月出生，大专学历',
            '吴君如演的周吉是电影《花田喜事》，在周吉大婚之夜，其夫林嘉声逃走失踪，后来其夫新科状元高中回来，周吉急往城楼
for i in range(len(str_list)):
    data = {}
    data['text'] = [char for char in str_list[i]]
    data['labels'] = ['O' for _ in data['text']]
    data['id'] = 0
    test_data.append(data)

dev_dataset =  NerDataset(test_data, args, tokenizer)
print(dev_dataset[0])
dev_loader = DataLoader(dev_dataset, shuffle=False, batch_size=args.dev_batch_size, num_workers=2)

model = BertNer(args)
model.to(device)

pred = test_model(model, dev_loader, args.output_dir, args.id2label, device)
for i in range(len(pred)):
    print(test_data[i]['text'])
    print(pred[i])
```

手册中例句的命名实体识别结果如下：

```
['《', '民', '航', '客', '运', '服', '务', '会', '话', '》', '是', '1', '9', '9', '5', '年', '中', '国', '民', '航', '出', '版', '社', '出', '版',
'的', '图', '书', '，', '作', '者', '是', '周', '石', '田']
['O', 'B-图书作品', 'I-图书作品', 'I-图书作品', 'I-图书作品', 'I-图书作品', 'I-图书作品', 'I-图书作品', 'I-图书作品', 'O', 'O', 'O', 'O', 'O', 'O',
 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-人物', 'I-人物', 'I-人物']
['再', '有', '之', '后', '的', '《', '半', '生', '缘', '》', '，', '蒋', '勤', '勤', '饰', '演', '的', '顾', '曼', '璐', '完', '全', '把', '林', '
心', '如', '的', '曼', '桢', '衬', '得', '像', '是', '涉', '世', '未', '深', '的', '小', '姑', '娘', '，', '毫', '无', '半', '点', '风', '情']
['O', 'O', 'O', 'O', 'O', 'O', 'B-影视作品', 'I-影视作品', 'I-影视作品', 'O', 'O', 'B-人物', 'I-人物', 'I-人物', 'O', 'O', 'O', 'B-人物', 'I-人物',
 'I-人物', 'O', 'O', 'O', 'B-人物', 'I-人物', 'I-人物', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', '
O', 'O', 'O', 'O']
['裴', '友', '生', '，', '男', '，', '汉', '族', '，', '湖', '北', '蕲', '春', '人', '，', '1', '9', '5', '7', '年', '1', '2', '月', '出', '生', '
，', '大', '专', '学', '历']
['B-人物', 'I-人物', 'I-人物', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', '
O', 'O', 'O', 'O']
['吴', '君', '如', '演', '的', '周', '吉', '是', '电', '影', '《', '花', '田', '喜', '事', '》', '，', '在', '周', '吉', '大', '婚', '之', '夜', '
，', '其', '夫', '林', '嘉', '声', '逃', '走', '失', '踪', '，', '后', '来', '其', '夫', '新', '科', '状', '元', '高', '中', '回', '来', '，', '周
', '吉', '急', '往', '城', '楼', '相', '识', '，', '但', '林', '嘉', '声', '却', '言', '夫', '妻', '情', '断', '，', '覆', '水', '难', '收']
['B-人物', 'I-人物', 'I-人物', 'O', 'O', 'B-人物', 'I-人物', 'O', 'O', 'O', 'O', 'B-影视作品', 'I-影视作品', 'I-影视作品', 'I-影视作品', 'O', 'O',
'O', 'B-人物', 'I-人物', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-人物', 'I-人物', 'I-人物', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O
', 'O', 'O', 'O', 'O', 'O', 'O', 'B-人物', 'I-人物', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-人物', 'I-人物', 'I-人物', 'O', 'O', 'O', 'O', 'O',
 'O', 'O', 'O', 'O', 'O', 'O']
```

```
民航客运服务会话：图书作品      周石田：人物
半生缘：影视作品        蒋勤勤：人物    顾曼璐：人物    林心如：人物
裴友生：人物
吴君如：人物    周吉：人物      花田喜事：影视作品      周吉：人物      林嘉声：人物    周吉：人物      林嘉声：人物
```

三、 基于大模型的命名实体识别

（1. 至少采用两种大模型完成例句的命名实体识别，并按照 json 格式展示实验结果）

以下是我采用两种大模型完成例句的命名实体识别的 prompt：

下面将给你几段文本数据，请你对此文本进行命名实体识别，并按照 json 格式输出结果。数据如下："['《民航客运服务会话》是 1995 年中国民航出版社出版的图书，作者是周石田', '再有之后的《半生缘》，蒋勤勤饰演的顾曼璐完全把林心如的曼桢衬得像是涉世未深的小姑娘，毫无半点风情', '裴友生，男，汉族，湖北蕲春人，1957 年 12 月出生，大专学历', '吴君如演的周吉是电影《花田喜事》，在周吉大婚之夜，其夫林嘉声逃走失踪，后来其夫新科状元高中回来，周吉急往城楼相识，但林嘉声却言夫妻情断，覆水难收']"

展示实验结果：

例句 1："《民航客运服务会话》是 1995 年中国民航出版社出版的图书，作者是周石田"

文心一言：

```
{
    "text": "《民航客运服务会话》是1995年中国民航出版社出版的图书，作者是周石田",
    "entities": [
        {
            "name": "民航客运服务会话",
            "type": "书名"
        },
        {
            "name": "1995年",
            "type": "时间"
        },
        {
            "name": "中国民航出版社",
            "type": "出版机构"
        },
        {
            "name": "周石田",
            "type": "作者"
        }
    ]
},
```

通义千问：


```
{
  "text": "《民航客运服务会话》",
  "type": "Book",
  "author": "周石田",
  "publication_year": "1995",
  "publisher": "中国民航出版社"
},
```


例句 2：“再有之后的《半生缘》，蒋勤勤饰演的顾曼璐完全把林心如的曼桢衬得像是涉世未深的小姑娘，毫无半点风情”

文心一言：


```
{
    "text": "再有之后的《半生缘》，蒋勤勤饰演的顾曼璐完全把林心如的曼桢衬得像是涉世未深
    "entities": [
        {
            "name": "半生缘",
            "type": "剧名"
        },
        {
            "name": "蒋勤勤",
            "type": "演员"
        },
        {
            "name": "顾曼璐",
            "type": "角色"
        },
        {
            "name": "林心如",
            "type": "演员"
        },
        {
            "name": "曼桢",
            "type": "角色"
        }
    ]
},
```


通义千问：


```
{
  "text": "《半生缘》",
  "type": "TVSeries/Movie",
  "actor": "蒋勤勤",
  "character": "顾曼璐",
  "comparison_actor": "林心如",
  "comparison_character": "曼桢"
},
```


例句 3：“裴友生，男，汉族，湖北蕲春人，1957 年 12 月出生，大专学历”

文心一言：

```
    {
        "text": "裴友生，男，汉族，湖北蕲春人，1957年12月出生，大专学历",
        "entities": [
            {
                "name": "裴友生",
                "type": "人名"
            },
            {
                "name": "男",
                "type": "性别"
            },
            {
                "name": "汉族",
                "type": "民族"
            },
            {
                "name": "湖北蕲春",
                "type": "地点"
            },
            {
                "name": "1957年12月",
                "type": "时间"
            },
            {
                "name": "大专",
                "type": "学历"
            }
        ]
    },
```


通义千问：


```
{
  "text": "裴友生",
  "type": "Person",
  "gender": "男",
  "ethnicity": "汉族",
  "birthplace": "湖北蕲春",
  "birth_year": "1957",
  "birth_month": "12",
  "education": "大专学历"
},
```


例句 4：“吴君如演的周吉是电影《花田喜事》，在周吉大婚之夜，其夫林嘉声逃走失踪，后来其夫新科状元高中回来，周吉急往城楼相识，但林嘉声却言夫妻情断，覆水难收”

文心一言：

```
{
    "text": "吴君如演的周吉是电影《花田喜事》，在周吉大婚之夜，其夫林嘉声逃走失踪，后来
    "entities": [
        {
            "name": "吴君如",
            "type": "演员"
        },
        {
            "name": "周吉",
            "type": "角色"
        },
        {
            "name": "花田喜事",
            "type": "电影名"
        },
        {
            "name": "林嘉声",
            "type": "角色"
        },
        {
            "name": "新科状元",
            "type": "职业"
        }
    ]
}
```


通义千问：


```
{
  "text": "《花田喜事》",
  "type": "Movie",
  "actor": "吴君如",
  "character": "周吉",
  "event": "大婚之夜",
  "related_person": "林嘉声",
  "related_event": "逃走失踪",
  "later_event": "新科状元高中回来",
  "location": "城楼",
  "quote": "夫妻情断，覆水难收"
}
```