

知识工程-作业 8 中文事件抽取

2024214500 叶璨铭

代码与文档格式说明

本文档使用 Jupyter Notebook 编写，遵循 Diátaxis 系统 Notebook 实践

https://nbdev.fast.ai/tutorials/best_practices.html，所以同时包括了实验文档和实验代码。

本文档理论上支持多个格式，包括 ipynb, docx, pdf 等。您在阅读本文档时，可以选择您喜欢的格式来进行阅读，建议您使用 Visual Studio Code（或者其他支持 jupyter notebook 的 IDE，但是 VSCode 阅读体验最佳）打开 ipynb 格式的文档来进行阅读。

为了记录我们自己修改了哪些地方，使用 git 进行版本控制，这样可以清晰地看出我们基于助教的代码在哪些位置进行了修改，有些修改是实现了要求的作业功能，而有些代码是对原本代码进行了重构和优化。我将我在知识工程课程的代码，在作业截止 DDL 之后，开源到

<https://github.com/2catycm/THU-Coursework-Knowledge-Engineering.git>，方便各位同学一起学习讨论。

代码规范说明

在我们实现函数过程中，函数的 docstring 应当遵循 fastai 规范而不是 numpy 规范，这样简洁清晰，不会 Repeat yourself。相应的哲学和具体区别可以看

https://nbdev.fast.ai/tutorials/best_practices.html#keep-docstrings-short-elaborate-in-separate-cells

为了让代码清晰规范，在作业开始前，使用 ruff format 格式化助教老师给的代码；

```
> ruff format
5 files reformatted
```

很好，这次代码格式化没有报错。

注意 pylance 报错

```
om torch.utils.data.dis “AdamW”是未知的导入符号 PyLance(reportAttributeAccessIssue)
om tqdm import tqdm, tr (import) AdamW: Unknown
om utils import get_lab
查看问题 (F2) 快速修复... (Alt+Enter) 使用 Copilot 修复 (Ctrl+I)
om transformers import AdamW, get_linear_schedule_with_warmup
om transformers import (
```

参考

<https://github.com/huggingface/transformers/issues/15497>

```
> pip show transformers
Name: transformers
Version: 4.50.3
Summary: State-of-the-art Machine Learning for JAX, PyTorch and TensorFlow
```

我们的更新。

没区别，实际上就用 torch 的好了

```
from transformers import get_linear_schedule_with_warmup
from torch.optim import AdamW
```

实验环境准备

采用上次的作业专属环境，为了跑通最新方法，使用 3.12 和 torch 2.6

```
conda create -n assignments python=3.12
conda activate assignments
pip install -r ../requirements.txt
pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu126
pip install -U git+https://github.com/TorchRWKV/flash-linear-attention
```

注意到

```
from sequeval.metrics import precision_score, recall_score, f1_score
```

参考 <https://github.com/chakki-works/sequeval>

```
pip install sequeval
```

原理回顾和课件复习

课上详细介绍了事件抽取任务的一些基本特点和难点，事件肯定和时间有关系，是动态的知识，“清华大学在北京”就是相对静态的。“五月份清华举行校庆”那才有事件。实体图谱就没有事件图谱厉害。

事件会有**触发词 trigger**，决定事件的类别；论元 **argument**，实体、事件、属性，应该有时间，没有事件就不是论元。

数据准备

download.sh 的清华网盘链接过期了，还好非常好的学长给的压缩包已经处理好了数据 raw。

现在我们写 preprocess

首先观察数据格式

```
{ "id": 5997, "text": "后来靠着李烈钧、蒋伯诚等人的说情，又恰逢老蒋新娶宋美龄的好心情，才被释放出来，并且恢复了监狱科长的职务", "labels": [{"trigger": "释放", 12}],  
{ "id": 5777, "text": "在这前一天，上年岁末，毛泽东在书房里会见两位美国客人——美国前总统尼克松的女儿朱莉·尼克松·艾森豪威尔和女婿戴维·艾森豪威尔", "labels": [{"trigger": "会见", 12}],  
{ "id": 6691, "text": "花旗港员工获派八千英资银行取消派息，累食息一族见财化水，但美资银行花旗集团则派钱支持港区同事应对疫境", "labels": [{"trigger": "派钱", 12}],  
{ "id": 3773, "text": "黄山风景区管委会在5日晚间作出回应，小长假里黄山每天限流2万人，比平时的5万人限量已压缩了六成", "labels": [{"trigger": "限流", 12}],  
{ "id": 3472, "text": "领导小组构架内蒙古自治区成立了煤炭资源领域违法违规问题专项整治工作领导小组", "labels": [{"trigger": "成立", 12}],  
{ "id": 2211, "text": "为案民警介绍，为了让这个案件办理过程更加公正，尽量为受害人挽回损失，乐清警方首次聘请了第三方对小珍等人资金往来进行审计", "labels": [{"trigger": "聘请", 12}],  
{ "id": 5864, "text": "1961年底，台湾特务机关派上校级特务张露芝到南越西贡，在美国和南越特务机关的配合下，向设立在西贡的特务机构——南越第三工作指", "labels": [{"trigger": "派遣", 12}],  
{ "id": 4484, "text": "原标题：商务部：今年年末我国汽车保有量有望超越美国今天下午，国务院联防联控机制召开新闻发布会，介绍应对疫情影响、稳定和扩大消费", "labels": [{"trigger": "发布", 12}],  
{ "id": 6985, "text": "食堂门口还贴着一张海报：沈阳金杯公司发行股票", "labels": [{"trigger": "发行", 18}, {"object": "沈阳金杯公司", 12}],
```

这个实际上是 jsonl，一行一个 json

每一行数据，有 id, text, labels, distant_trigger 四个字段 id 就是数据的 id, text 就是句子, distant_trigger 是句子里面 trigger 的列表。labels 则是多个对象，一个对象是有 trigger, object, subject, time 和 location。

```
{  
  "id": 5009,  
  "text": "晚年的毛岸青关心国家大事，关注祖国统一，拥护改革开放，热心支持老少边穷地区建设，多次和夫人邵华、儿子毛新宇重走长征路，到革命老区、到工厂、到农村调研，并以多种形式帮助失学儿童，支持创办了多个青少年爱国主义教育基地",  
  "labels": [  
    {  
      "trigger": ["支持创办", 88],  
      "object": ["儿子毛新宇", 48],  
      "subject": ["多个青少年爱国主义教育基地", 93],  
      "time": "",  
      "location": "",  
    }  
  ],  
  "distant_trigger": ["帮助", "支持", "拥护", "建设", "开放", "创办", "改革", "关注"],  
}
```

为什么 labels 里面的比 distant_trigger 的少呢？

远距离触发词（Distant Trigger）是一组与事件相关的关键词或短语，它们可能不是直接的触发词，但可以提供上下文信息，帮助模型更全面地理解事件的语义。这些词通常是通过某种方法（如规则、统计分析或知识库）从文本中提取出来的，用于扩展触发词的语义范围。在标注数据不足的情况下，远距离触发词可以作为一种辅助信息。

现在我们来写要求的函数，首先了解 IOB2 格式

[https://en.wikipedia.org/wiki/Inside%E2%80%93outside%E2%80%93beginning_\(tagging\)](https://en.wikipedia.org/wiki/Inside%E2%80%93outside%E2%80%93beginning_(tagging))。

IOB 是 inside, outside, beginning 的缩写，也叫作 BIO。

IOB2 比较好理解，

Alex B-PER

is O

going O

to O

Los B-LOC

Angeles I-LOC

in O

California B-LOC

```
import json
```

```
import os
```

```
from tqdm import tqdm
```

```
def process_trigger_data(file):
```

```
    """
```

```
        convert raw data into sequence-labeling format data for trigger ide  
ntification, and save to `./data/processed/trigger`
```

```
        each line in the converted contains `token[space]label`
```

```
        you can use IOB2 format tagging https://en.wikipedia.org/wiki/Insid  
e%E2%80%93outside%E2%80%93beginning\_\(tagging\), or other tagging schema  
you think fit
```

```
        use empty line to indicate end of one sentence
```

```
        if using IOB2 format, the example output would be like
```

```
        ``
```

```
        本 O
```

```
        平 O
```

```
        台 O
```

```
        S O
```

```
        S O
```

```
        R O
```

```
        N O
```

```
        发 B-EVENT
```

```
        表 I-EVENT
```

```
        了 O
```

```
        题 O
```

```
        为 O
```

```
        《 O
```

```
        夏 O
```

```
        季 O
```

```
        冠 O
```

```
        状 O
```

病 0
毒 0
流 0
行 0
会 0
减 0
少 0
吗 0

对 0
于 0
该 0
举 0
动 0
```\n""

```
with open(f"./data/raw/{file}.json", encoding="utf-8") as f:
 lines = f.readlines()
 outlines = []
 #####
 bar = tqdm(lines, desc="Processing trigger data")
 for line in bar:
 sample_object = json.loads(line.strip())
 text = sample_object["text"]
 tokens = list(text) # 单字分词
 labels = sample_object["labels"]
 labels_seq = ["0"] * len(tokens) # 默认情况
 # 依据 labels 进行标注
 for label in labels:
 if label["trigger"]:
 start = label["trigger"][1] # 起始位置
 end = start + len(label["trigger"][0]) # 结束位置
 # 触发词标注
 labels_seq[start] = "B-EVENT"
 for i in range(start + 1, end):
 labels_seq[i] = "I-EVENT"
 # 生成输出
 for i in range(len(tokens)):
 outlines.append(f"{tokens[i]} {labels_seq[i]}")
 outlines.append("") # 句子结束标志, 用空行分割

 #####

if not os.path.exists("./data/processed/trigger"):
 os.makedirs("./data/processed/trigger", exist_ok=True)
```

```
with open(f"./data/processed/trigger/{file}.txt", "w") as f:
 f.writelines("\n".join(outlines))
```

注意我们忽略了 `distant_trigger` 。

同样地来写 `process_argument_data`

```
def process_argument_data(file):
 """
```

```
 convert raw data into sequence-labeling format data for argument id
 entification, and save to `./data/processed/argument`
 event triggers are surrounded by `` `` markers
 each line in the converted contains `token[space]label`
 you can use BIO format tagging https://en.wikipedia.org/wiki/Inside-Outside_beginning_\(tagging\), or other tagging schema
 you think fit
```

```
 use empty line to indicate end of one sentence
 if using IOB2 format, the example output would be like
    ```
```

```
3 B-object
0 I-object
年 I-object
代 I-object
<event> 0
参 0
加 0
<event/> 0
中 B-subject
共 I-subject
中 I-subject
央 I-subject
的 I-subject
特 I-subject
种 I-subject
领 I-subject
导 I-subject
工 I-subject
作 I-subject
```

```
他 0
告 0
诉 0
新 0
京 0
报 0
记 0
者 0
```

```

"""
with open(f"./data/raw/{file}.json", encoding="utf-8") as f:
    lines = f.readlines()
    outlines = []
    #####
    bar = tqdm(lines, desc="Processing argument data")
    for line in bar:
        sample_object = json.loads(line.strip())
        text = sample_object["text"]
        tokens = list(text)
        labels = sample_object["labels"]
        labels_seq = ["0"] * len(tokens)
        # 依据 labels 进行标注
        for label in labels:
            # 先处理触发词， 如果强行插入新词的话，不太好，我决定后面输出的时
            候特别操作。
            if label["trigger"]:
                start = label["trigger"][1] # 起始位置
                end = start + len(label["trigger"][0]) # 结束位置
                # 触发词标注
                labels_seq[start] = "B-EVENT"
                for i in range(start + 1, end-1):
                    labels_seq[i] = "0"
                labels_seq[end-1] = "E-EVENT" # 触发词最后一个字标注为 E-
EVENT

            # 处理 object subject
            if label["object"]:
                start = label["object"][1]
                end = start + len(label["object"][0])
                labels_seq[start] = "B-object"
                for i in range(start + 1, end):
                    labels_seq[i] = "I-object"

            if label["subject"]:
                start = label["subject"][1]
                end = start + len(label["subject"][0])
                labels_seq[start] = "B-subject"
                for i in range(start + 1, end):
                    labels_seq[i] = "I-subject"

            # 生成输出
            for i in range(len(tokens)):
                if labels_seq[i] == "B-EVENT":
                    outlines.append("<event> 0")
                    outlines.append(f"{tokens[i]} 0")
                elif labels_seq[i] == "E-EVENT":

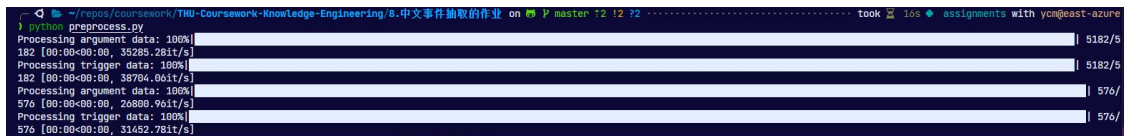
```

```

        outlines.append(f"{tokens[i]} 0")
        outlines.append("<event/> 0")
    else:
        # 普通情况
        outlines.append(f"{tokens[i]} {labels_seq[i]}")
    outlines.append("") # 句子结束标志, 用空行分割

```

难点是 `event` 标签比较特殊。




```

python preprocess.py
Processing argument data: 100% | 5182/5
182 [00:00<00:00, 35289.28it/s]
Processing trigger data: 100% | 5182/5
182 [00:00<00:00, 38704.00it/s]
Processing argument data: 100% | 576/
576 [00:00<00:00, 26800.90it/s]
Processing trigger data: 100% | 576/
576 [00:00<00:00, 31492.78it/s]

```

这就成功处理。



```

大 I-object
龄 I-object
的 I-object
好 I-object
心 I-object
情 I-object
, 0
才 0
被 0
<event> 0
释 0
放 0
<event/> 0
出 B-subject
来 I-subject
0

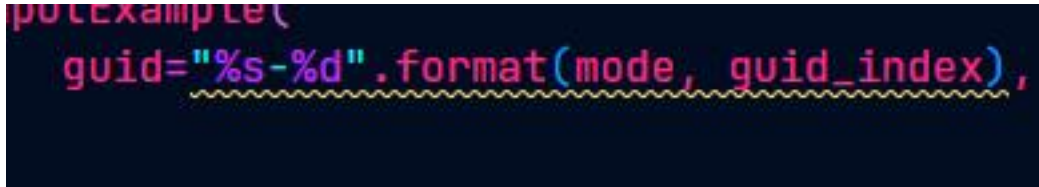
```

“由于一个事件触发词或论元可能跨越多个 `token`，因此标注格式采用 `I0B2`”
 跨越多个 `token` 的意思是多个 `token` 组成一个实体，不是说可以跳过中间的 `token` 联合后面的 `token`。

数据加载

我们看下 `utils.py` 文件，我们需要完成 `read_examples_from_file` 和 `convert_examples_to_features`

首先修复语法错误



改成 `f"{mode:s}-{guid_index:d}"`

```
def read_examples_from_file(data_dir, mode):
    """
    read file and convert to a list of `InputExample`s
    """
    file_path = os.path.join(data_dir, "{}.txt".format(mode))
    guid_index = 1
    examples = []
    with open(file_path, encoding="utf-8") as f:
        file = f.read()
        samples = file.split("\n\n")
        for guid_index, sample in enumerate(samples):
            lines = sample.split("\n")
            words = []
            labels = []
            for line in lines:
                if line.strip() == "":
                    continue
                line = line.split(" ")
                if len(line) == 2:
                    words.append(line[0])
                    labels.append(line[1])
                else:
                    raise ValueError(
                        "Error in line format: {} in file {}".format(line,
file_path)
                    )

            examples.append(
                InputExample(guid=f"{mode:s}-{guid_index:d}", words=words,
labels=labels)
            )
    return examples
```

原本的逻辑不太方便，我直接重构了。

```

def convert_examples_to_features(
    examples: List[InputExample],
    label_list, # a list of all unique labels
    max_seq_length: int, # all sequence should be padded or truncated
    to `max_seq_length`
    tokenizer, # PretrainedTokenizer
    pad_token_label_id: int, # label id for pad token
) -> List[InputFeatures]: # features
    """Loads a list of `InputExample`s into a list of `InputFeatures`s"""

    cls_token = tokenizer.cls_token
    sep_token = tokenizer.sep_token
    pad_token_id = tokenizer.pad_token_id # padded token id
    # tokenizer.convert_tokens_to_ids([tokenizer.pad_token])[0]
    label_map = {label: i for i, label in enumerate(label_list)}

    features = []
    for ex_index, example in enumerate(examples):
        # Hint: remember to add `[CLS]` and `[SEP]` tokens for BERT model
        # e.g. [CLS] the dog is hairy . [SEP]
        tokens = []
        label_ids = []
        for word, label in zip(example.words, example.labels):
            word_tokens = tokenizer.tokenize(word)

            # Bert 模型中，一个单词可能会被切分成多个子词，我们需要将标签分配
            # 给这些子词
            if len(word_tokens) > 0:
                tokens.extend(word_tokens)
                # 只有第一个子词保留原始标签，其余子词使用特殊标签 "x"
                label_ids.extend([label_map[label]] + [pad_token_label_id] * (len(word_tokens) - 1))

            # 添加[CLS]和[SEP]标记
            special_tokens_count = 2
            if len(tokens) > max_seq_length - special_tokens_count:
                tokens = tokens[: (max_seq_length - special_tokens_count)]
                label_ids = label_ids[: (max_seq_length - special_tokens_count)]

            tokens += [sep_token]
            label_ids += [pad_token_label_id]
            tokens = [cls_token] + tokens
            label_ids = [pad_token_label_id] + label_ids

        input_ids = tokenizer.convert_tokens_to_ids(tokens)

```

注意mask 的处理, 只有真实的token 对应的mask 值为1, padding 的token 对应的mask 值为0

```
input_mask = [1] * len(input_ids)
padding_length = max_seq_length - len(input_ids)
input_ids = input_ids + ([pad_token_id] * padding_length)
input_mask = input_mask + ([0] * padding_length)
label_ids = label_ids + ([pad_token_label_id] * padding_length)

assert len(input_ids) == max_seq_length
assert len(input_mask) == max_seq_length
assert len(label_ids) == max_seq_length

features.append(
    InputFeatures(
        input_ids=input_ids, input_mask=input_mask, label_ids=label_ids
    )
)
return features
```

运行效果

CUDA_VISIBLE_DEVICES=7 python -u main.py --mode trigger

```
CUDA_VISIBLE_DEVICES=7 python -u main.py --mode trigger
5 ['0', '8-EVENT', 'I-EVENT']
Some weights of BertForTokenClassification were not initialized from the model checkpoint at hf/chinese-bert-wwm-ext and are newly initialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Iteration: 100% | 648/648 [01:08<00:00, 9.51it/s]
Evaluating: 100% | 72/72 [00:02<00:00, 30.83it/s]
{'precision': np.float64(0.7395973154362416), 'recall': np.float64(0.7118863849895097), 'f1': np.float64(0.7254772876892693)}
Epoch: 5% | 1/20 [01:13<23:23, 73.85s/it]
Iteration: 11% | 74/648 [00:07<01:00, 9.54it/s]
```

首先下载了 tokenizer

然后开始训练。

```
72 [00:02<00:00, 30.38it/s]
Iteration: 100% | 648/648 [01:08<00:00, 9.51it/s]
48 [01:06<00:00, 9.74it/s]
Evaluating: 100% | 72/72 [00:02<00:00, 30.83it/s]
72 [00:02<00:00, 32.04it/s]
{'precision': np.float64(0.7329192546583851), 'recall': np.float64(0.7622739018087855), 'f1': np.float64(0.7473084230525648)}
72 [00:02<00:00, 32.12it/s]
Epoch: 100% | 20/20 [24:34<00:00, 73.84s/it]
20 [24:34<00:00, 73.84s/it]
Evaluating: 100% | 72/72 [00:02<00:00, 30.83it/s]
72 [00:02<00:00, 27.99it/s]
{'precision': np.float64(0.7604166666666666), 'recall': np.float64(0.7545219638242894), 'f1': np.float64(0.7574578469520183)}
```

然后我们运行

python -u transform.py

```
> python -u transform.py
Processing: 576it [00:00, 19603.05it/s]
```

python -u main.py --mode argument

注意学长 README 忘记 --mode 了

运行到最后我们才发现，test.txt 里面有 B-time 和 B-location

```
” 0
4 B-time
月 I-time
1 I-time
8 I-time
日 I-time
上 I-time
午 I-time
1 I-time
0 I-time
点 I-time
, 0
记 B-object
者 I-object
辗 0
转 0
1 B-time
个 I-time
多 I-time
小 I-time
时 I-time
<event> 0
```

```

Maximum sequence length exceeded: No prediction for '运'.
Maximum sequence length exceeded: No prediction for '司'.
Maximum sequence length exceeded: No prediction for '机'.
Maximum sequence length exceeded: No prediction for '月'.
Maximum sequence length exceeded: No prediction for '2'.
Maximum sequence length exceeded: No prediction for '6'.
Maximum sequence length exceeded: No prediction for '日'.
Maximum sequence length exceeded: No prediction for ', '.
Maximum sequence length exceeded: No prediction for '6'.
Maximum sequence length exceeded: No prediction for '支'.
Maximum sequence length exceeded: No prediction for '国'.
Maximum sequence length exceeded: No prediction for '家'.

```

还遇到了 Max seq length 的警告。这是正常的，代码中确实模型设置了 max length 而数据集中确实有“货运司机”的数据，是在句子的末尾，所以无法预测，相应地评估也不评估就行了。

```

26     ["放", "派出"]}]
27     "trigger": ["发现"]}]
28     ["返于黑龙江省和滨海边疆区乌苏里斯克的货运司机", "L
29     "distant_trigger": ["指挥", "防控", "参加"]}]
30     "time": "", "location": ""}], "distant_tri
31
32     ["报告", 62], "object": ["北京市", 57], "su

```

紧急修复代码

```

if label['time']:
    start = label['time'][1]
    end = start + len(label['time'][0])
    labels_seq[start] = "B-time"
    for i in range(start + 1, end):
        labels_seq[i] = "I-time"

if label['location']:
    start = label['location'][1]
    end = start + len(label['location'][0])
    labels_seq[start] = "B-location"
    for i in range(start + 1, end):
        labels_seq[i] = "I-location"

```



```

> python preprocess.py
Processing argument data: 100%|          | 5182/5182 [00:00:00:00, 34621.32it/s]
Processing trigger data: 100%|          | 5182/5182 [00:00:00:00, 45248.86it/s]
Processing argument data: 100%|          | 576/576 [00:00:00:00, 29582.72it/s]
Processing trigger data: 100%|          | 576/576 [00:00:00:00, 43310.01it/s]
> CUDA_VISIBLE_DEVICES=7 python -u main.py --mode trigger

```

重新运行。

```

Evaluating: 100%|          | 72/72 [00:02:00:00, 25.76it/s]
{'precision': np.float64(0.7285812285812284), 'recall': np.float64(0.7661498788010336), 'f1': np.float64(0.7468513853904282)}
Epoch: 100%|          | 70/72 [00:02:00:00, 26.23it/s]
Evaluating: 100%|          | 20/20 [28:12:00:00, 84.61it/s]
{'precision': np.float64(0.7393483789273183), 'recall': np.float64(0.7622739818887855), 'f1': np.float64(0.7506361323155216)}

```

一阶段的结果是一样的。

再次运行

这一次我们运行成功，可以查看结果

```

P_homework8.ipynb M  eval_results.txt U X
8.中文事件抽取的作业 > checkpoint > argument > eval_results.txt
1  f1 = 0.6716077537058153
2  precision = 0.6678004535147393
3  recall = 0.6754587155963303
4

```

注意 tokenizer 里面自动增加了两个新的 token

```

事件抽取的作业 > checkpoint > argument > checkpoint-best > {} added_to
1  {
2  "<event/>": 21129,
3  "<event>": 21128
4  }
5

```

包含许多不明确的 unicode 字符

禁用不明确的突出显示



```
1  ” 0 I-time
2  4 B-time
3  月 I-time
4  1 I-time
5  8 I-time
6  日 I-time
7  上 I-time
8  午 I-time
9  1 I-time
0  0 I-time
1  点 I-time
2  , 0
3  记 B-object
4  者 I-object I-...t
5  辗 0
6  转 0
7  1 0
8  个 0
9  多 0 I-...t
0  小 0
1  时 0
2  <event> 0 I-...t
3  找 0
4  到 0
5  <event/> 0
6  了 0
7  邓 B-subject
8  凡 I-subject
9  全 I-subject
0  的 I-subject
1  家 I-subject
2  , 0 I-...t
3  不 0 B-...t
```

预测的输出看起来非常合理。