

## 一、复现原因：

结合硕士论文工作，本次机器学习理论方法实践应用做一个关于事件抽取的代码复现。事件抽取是信息抽取领域一个重要的研究方向。事件抽取把含有事件信息的非结构化文本以结构化的形式呈现出来。非结构化文本较为复杂多样，中文信息抽取的研究起步较晚，事件抽取可以减少人工干预，为用户提供更加方便的信息服务，甚至可以发展为信息主动推送服务。由于此类少有论文给予相关代码，复现了 2020 年科大讯飞事件抽取挑战赛冠军组的程序。

比赛官网：<http://challenge.xfyun.cn/topic/info?type=hotspot>

## 二、主要思路：

这是一个基于深度学习的事件抽取。将任务分割为触发词抽取，论元抽取，属性抽取。具体而言是论元和属性的抽取结果依赖于触发词，因此只有一步误差传播。因 time loc 并非每个句子中都存在，并且分布较为稀疏，因此将 time & loc 与 sub & obj 的抽取分开（role1 提取 sub & obj；role2 提取 time & loc）

模型先进行触发词提取，由于数据集的特殊性，模型限制抽取的事件仅有一个，如果抽取多个触发词，选择 logits 最大的 trigger 作为该句子的触发词，如果没有抽取触发词，筛选整个句子的 logits，取 argmax 来获取触发词；

然后根据触发词抽取模型抽取的触发词，分别输入到 role1 & role2 & attribution 模型中，进行后序的论元提取和属性分类；四种模型都是基于 Roberta-wwm 进行实验，加入了不同的特征。

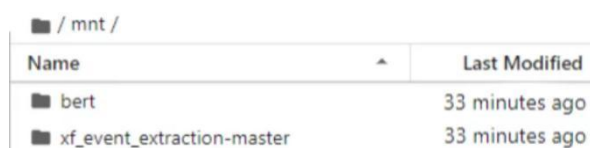
最后将识别的结果进行整合，得到提交文件。

## 三、项目运行：

### 模型训练

1. 矩池云选择：“预装：Python 3.7， CUDA 10.2， cuDNN 7.6， Pytorch 1.6.0， Ubuntu 18.04， VNC”，其他配置按照 requirements.txt 中的要求安装依赖包，需要一个一个装。（遇到问题，有一个包无法安装，但对后面没有影响）

2. 下载 bert，BERT 采用的是哈工大的全词覆盖 wwm 模型，下载地址：<https://github.com/ymcui/Chinese-BERT-wwm>



Name	Last Modified
bert	33 minutes ago
xf_event_extraction-master	33 minutes ago

3. 数据转换：数据转换部分只提供代码和已经转换好的数据，具体操作在 src\_final/preprocess 中的 convert\_raw\_data 中，包含对初赛/复赛数据的清洗和转换。

4. 配置好环境后，在 JupyterLab 的 Terminal 中输入：bash ./script/final/train.sh 命令执行训练模型（遇到问题，解决方案：将 train 文件里的目录改为绝对路径），

依次更改模型名称。分别训练 trigger 提取模型，

```
14 export TASK_TYPE="trigger"
```

Role1/role2 提取模型训练,

```
14 export TASK_TYPE="role1"
```

```
14 export TASK TYPE="role2"
```

和 attribution 分类模型训练,

```
14 export TASK_TYPE="attribution"
```

运行成功后会在 `./out/final` 文件夹下面：生成 4 个文件，保存着模型。

```

/ ... / out / final /

```

Name	Last Modified
attribution	a day ago
role1	a day ago
role2	a day ago
trigger	a day ago

训练数据 2087 条： `▼ root: [] 2087 items`，线下评估结果在每个模型文件目录下 `eval_metric.txt` 下，保留最优线下结果作为训练结果

```
105 eval_save_path = os.path.join(opt.dev_dir, 'eval_metric.txt')
```

## 模型验证

1. 输入：模型训练好以后在 JupyterLab 的 Terminal 中输入 `bash mnt/xf_event_extraction2020Top1-master/script/final/dev.sh` 进行验证
2. 测试集的 `dev.json` 如图：

```

▼ 1:
  sentence: "之后日本代表提出能否重新签订一份投降书，结果麦克阿瑟不同意了"
  ▼ events: [] 1 item
    ▼ 0:
      ▼ trigger:
        text: "提出"
        length: 2
        offset: 6
        tense: "过去"
        polarity: "肯定"
      ▼ arguments: [] 2 items
        ▼ 0:
          role: "subject"
          text: "日本代表"
          offset: 2
          length: 4
        ▼ 1:
          role: "object"
          text: "能否重新签订一份投降书"
          offset: 8
          length: 11
      ▼ distant_triggers: [] 2 items
        0: "提出"
        1: "签订"

```

Trigger 模型: precision=0.9377, recall=0.9227, f1=0.9301, 10000 是 max f1 step

```
06/02/2022 08:38:53 - INFO - src_final.utils.functions_utils - Load ckpt from mnt/xf_event_extraction2020Topl-master/out/final/trigger/roberta_www_pg_d_enhanced/checkpoint-100000/model.pt
06/02/2022 08:39:00 - INFO - src_final.utils.functions_utils - Use single gpu in: ['0']
Get role task predict logits: 100% ██████████ 2/2 [00:27<00:00, 13.97s/it]
06/02/2022 08:39:28 - INFO - __main__ - In step 100000:
In start threshold: 0.5; end threshold: 0.5
[MIRCO] precision: 0.9377, recall: 0.9227, f1: 0.9301
Zero pred nums: 0
06/02/2022 08:39:28 - INFO - main - Max fl is: 0.9301075268817204, in step 100000
```

**Attribution 模型：**（遇到问题，找不到路径文件，解决方法：更改的 dev.sh

中 dev\_dir 路径文件名称) Attribution 测试时, 用百度 ERNIE 模型对 attribution 十折交叉验证, polarity=0.9893, tense=0.9867, f1=0.988

```
06/02/2022 08:59:28 - INFO - transformers.modeling_utils - loading weights file mnt/bert/torch_roberta_wm/pytorch_model.bin
06/02/2022 08:59:34 - INFO - src_final.utils.functions_utils - Load ckpt from mnt/xf_event_extraction2020Top1-master/out/final
attribution/roberta_wm_pgd_enhanced/checkpoint-100000/model.pt
06/02/2022 08:59:43 - INFO - src_final.utils.functions_utils - Use single gpu in: ['0']
Get attribution task predict logits: 100% ██████████ 2/2 [00:28<00:00, 14.20s/it]
06/02/2022 09:00:11 - INFO - __main__ - In step 100000:
[ACC] polarity: 0.9893, tense: 0.9867
06/02/2022 09:00:11 - INFO - __main__ - Max fl is: 0.988, in step 100000
```

**Role1 模型：**分别计算 object 和 subject 的 precision 和 recall，f1，分别计算两类的 F1 之后再使用 micro 方式得到整体的 f1，f1=0.717，precision=0.8133，recall=0.7861

```
06/02/2022 09:03:22 - INFO - transformers.modeling_utils - loading weights file mnt/bert/torch_roberta_wwm/pytorch_model.bin
06/02/2022 09:03:28 - INFO - src_final.utils.functions_utils - Load ckpt from mnt/xf_event_extraction2020Top1-master/out/final/roberta_wwm_distance_pgd_enhanced/checkpoint-1392/model.pt
06/02/2022 09:03:35 - INFO - src_final.utils.functions_utils - Use single gpu in: ['0']
Get role task predict logits: 100% ██████████ 2/2 [00:28<00:00, 14.19s/it]
06/02/2022 09:04:04 - INFO - __main__ - In step 1392:
In start threshold: 0.5; end threshold: 0.5
[object] precision: 0.8133, recall: 0.7861, f1: 0.7995.
[subject] precision: 0.8647, recall: 0.8402, f1: 0.8523.
[MIRCO] precision: 0.7291, recall: 0.7066, f1: 0.7177
06/02/2022 09:04:04 - INFO - __main__ - Max f1 is: 0.7176754469179996, in step 1392
```

Role2 模型：分别计算 time 和 loc 的 precision 和 recall, f1, 分别计算两类的 F1 之后再使用 micro 方式得到整体的 f1, f1=0.1056

```
06/02/2022 09:05:38 - INFO - transformers.modeling_utils - loading weights file mnt/bert/torch_roberta_wwm/pytorch_model.bin
06/02/2022 09:05:44 - INFO - src_final.utils.functions_utils - Load ckpt from mnt/xf_event_extraction2020Top1-master/out/fin
al/role2/roberta_wwm_distance_pg_d_enhanced/checkpoint-1092/model.pt
06/02/2022 09:05:51 - INFO - src_final.utils.functions_utils - Use single gpu in: ['0']
Get role task predict logits: 100% ██████████ 2/2 [00:29<00:00, 14.58s/it]
06/02/2022 09:06:20 - INFO - __main__ - In step 1092:
[time] precision: 0.7381, recall: 0.9490, f1: 0.8304.
[loc] precision: 0.6364, recall: 0.7368, f1: 0.6829.
[MIRCO] precision: 0.0945, recall: 0.1198, f1: 0.1057
06/02/2022 09:06:20 - INFO - __main__ - Max f1 is: 0.10565577800234889, in step 1092
```

## 模型测试

1. 输入和结果保存：在 JupyterLab 的 Terminal 中输入 `bash mnt/xf_event_extraction2020Top1-master/script/final/test.sh` 进行数据测试，结果存在 submit 文件夹下的 `submit_v1.json` 文件中。
2. 数据处理：

因为比赛分为两个阶段，初赛复赛数据不一样，需要把初赛数据转化为复赛数据，对./data/preliminary/raw\_data/中的 stack.json 数据集进行处理，生成复赛样式的初赛数据 raw\_preliminary.json，同时对初赛数据 raw\_stack.json 进行清洗。

- 2.测试数据集位置: `./data/final/raw_data/sentences.json`, 如下图所示, 只有 `sentence` 和 `words`:

```
▼ root: [] 879 items
▼ 0:
  sentence: "青年化学家侯印国告诉记者，中国文化的高峰出现在宋朝，国学大师陈寅恪说“华夏民族之文化，历数千载之演进，而造极于赵宋之世”"
  ► words: [] 60 items
  ► 1:
```

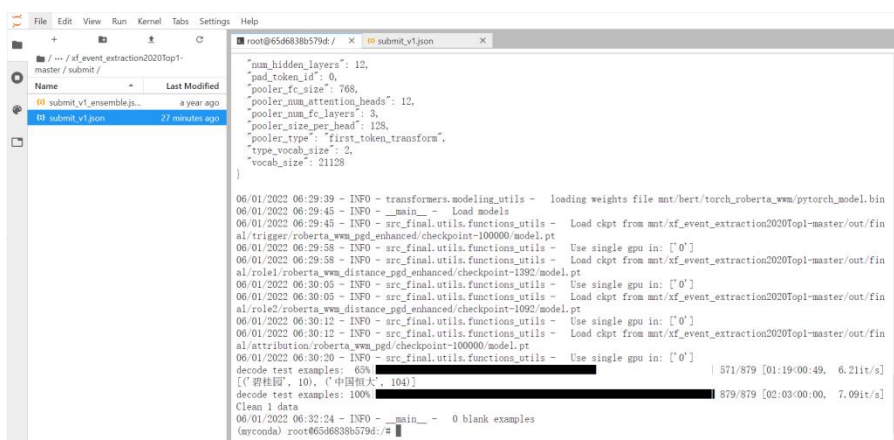
取一个短句子展示一下 `words` 的形式，如图：

```

▼ 5:
  sentence: "此后，郭盛又参加了高唐州之战，但并无表现"
  ▼ words: [] 20 items
    0: "此"
    1: "后"
    2: "，"
    3: "郭"
    4: "盛"
    5: "又"
    6: "参"
    7: "加"
    8: "了"
    9: "高"
    10: "唐"
    11: "州"
    12: "之"
    13: "战"
    14: "，"
    15: "但"
    16: "并"
    17: "无"
    18: "表"
    19: "现"

```

3.bash ./script/final/test.sh 测试运行成功截图：



4.submit\_v1.json 结果图：



可以识别出事件的触发词“告诉”，支持该事件论元（主体、客体、时间、地点）。同时也可以抽取表达事件发生的状态，包括极性、时态。

测试效果：

```

|/      classification      |      score      |
| :-----: | :-----: |
|      submit_v1.json      |      0.73684      |
| submit_v1_ensemble.json | **0.73859** |

```

## 抽取思想：

### 1、触发词提取器模型框架为 BERT + Feature

```

# trigger 提取器
class TriggerExtractor(BaseModel):
    def __init__(self,
                  bert_dir,
                  dropout_prob=0.1,
                  use_distant_trigger=False,
                  **kwargs):
        super(TriggerExtractor, self).__init__(bert_dir=bert_dir,
                                                dropout_prob=dropout_prob)

        self.use_distant_trigger = use_distant_trigger

        out_dims = self.bert_config.hidden_size

```

将标注数据中的所有 Trigger 作为知识库，用 10 折交叉构造 distant trigger，选取长度为 2 的 distant trigger，一句话可能标注多个 trigger，但结果只有一个 trigger，则把这个 trigger 放在前面，trigger 词典是有顺序的。

```

181 triggers = dict(sorted(triggers.items(), key=lambda x: x[1], reverse=True))

```

最后得到一个含有 224 个 trigger 的词典。

```

219 triggers_dict = {key: idx + 1 for idx, key in enumerate(triggers.keys())}
220 print(triggers_dict)

```

在测试集抽取 trigger 时，如果解码出多个 trigger，选取 logits 最大的那个 trigger 作为候选。

选取排在前五位的 trigger 展示：



```

root@65d6838b579d: /  X submit_v1.json  X sentences.json  X triggers_dict.json  X
▼ root:
  发布: 1
  推出: 2
  采用: 3
  支持: 4
  接受: 5

```

也有可能知识库里面没有需要抽取的 trigger，通过比较句子中匹配知识库的所有 distant trigger 的 start + end logits，选取最大的一个作为解码出的 trigger

### 2、time & loc 模型



```

# time & loc 提取器
class Role2Extractor(BaseModel):
    def __init__(self,
                  bert_dir,
                  dropout_prob=0.1,
                  use_trigger_distance=False,
                  **kwargs):
        super(Role2Extractor, self).__init__(bert_dir=bert_dir,
                                              dropout_prob=dropout_prob)

        out_dims = self.bert_config.hidden_size

        self.use_trigger_distance = use_trigger_distance

        self.conditional_layer_norm = ConditionalLayerNorm(out_dims, eps=self.bert_config.layer_norm_eps)

```

### 3、论元提取器模型框架为 BERT-ConditionalLayerNorm

```

# sub & obj 提取器
class Role1Extractor(BaseModel):
    def __init__(self,
                  bert_dir,
                  dropout_prob=0.1,
                  use_trigger_distance=False,
                  **kwargs):
        super(Role1Extractor, self).__init__(bert_dir=bert_dir,
                                              dropout_prob=dropout_prob)

        out_dims = self.bert_config.hidden_size

        self.use_trigger_distance = use_trigger_distance

        self.conditional_layer_norm = ConditionalLayerNorm(out_dims, eps=self.bert_config.layer_norm_eps)

```

模型输入：原始文本+Trigger 在文本中的位置

特征：文本中所有词到 Trigger 的相对距离，Trigger 本身的相对距离为 0。

属性分类器模型框架为 BERT-DynamicPooling

```

# tense & polarity 分类器
class AttributionClassifier(BaseModel):
    def __init__(self,
                  bert_dir,
                  dropout_prob=0.1):
        super(AttributionClassifier, self).__init__(bert_dir=bert_dir,
                                                    dropout_prob=dropout_prob)

        out_dims = self.bert_config.hidden_size

        self.pooling_layer = nn.AdaptiveMaxPool1d(output_size=1)

        self.tense_classifier = nn.Linear(out_dims * 3, 4)
        self.polarity_classifier = nn.Linear(out_dims * 3, 3)

        self.criterion = nn.CrossEntropyLoss()

        init_blocks = [self.tense_classifier, self.polarity_classifier]

        self._init_weights(init_blocks)

```

模型输入：原始文本+Trigger 在文本中的位置。

### 四、比较分析

调整参数，重新找了事件数据进行验证，命名为的 dev1.json 验证集数据有 329 个，下图分别是 trigger、attribution、role1、role2 的 f1

```
06/07/2022 12:31:43 - INFO - src_final.utils.functions_utils - Load ckpt from mnt/xf_event_extraction2020Top1-master/out/final/trigger/roberta_wwm_pg_d_enhanced/checkpoint-100000/model.pt
06/07/2022 12:31:52 - INFO - src_final.utils.functions_utils - Use single gpu in: ['0']
Get role task predict logits: 100% [redacted] 2/2 [00:24:00:00, 12.40s/it]
06/07/2022 12:32:17 - INFO - __main__ - In step 100000:
In start threshold: 0.5; end threshold: 0.5
[MIRCO] precision: 0.9392, recall: 0.9224, f1: 0.9307
Zero pred nums: 0
06/07/2022 12:32:17 - INFO - __main__ - Max f1 is: 0.930722891566265, in step 100000
```

trigger 从 f1=0.9301 到 f1=0.9307，有所提升

```
06/07/2022 12:54:50 - INFO - transformers.modeling_utils - loading weights file mnt/bert/torch_roberta_wwm/pytorch_model.b
in
06/07/2022 12:54:57 - INFO - src_final.utils.functions_utils - Load ckpt from mnt/xf_event_extraction2020Top1-master/out/final/attribution/roberta_wwm_pg_d_enhanced/checkpoint-100000/model.pt
06/07/2022 12:55:07 - INFO - src_final.utils.functions_utils - Use single gpu in: ['0']
Get attribution task predict logits: 100% [redacted] 2/2 [00:25:00:00, 12.61s/it]
06/07/2022 12:55:32 - INFO - __main__ - In step 100000:
[ACC] polarity: 0.9881, tense: 0.9881
06/07/2022 12:55:32 - INFO - __main__ - Max f1 is: 0.9880597014925373, in step 100000
```

Attribution: 有所提升

```
06/07/2022 12:51:25 - INFO - src_final.utils.functions_utils - Load ckpt from mnt/xf_event_extraction2020Top1-master/out/final/role1/roberta_wwm_distance_pg_d_enhanced/checkpoint-1392/model.pt
06/07/2022 12:51:33 - INFO - src_final.utils.functions_utils - Use single gpu in: ['0']
Get role task predict logits: 100% [redacted] 2/2 [00:25:00:00, 12.62s/it]
06/07/2022 12:51:58 - INFO - __main__ - In step 1392:
In start threshold: 0.5; end threshold: 0.5
[object] precision: 0.8060, recall: 0.7826, f1: 0.7941.
[subject] precision: 0.8605, recall: 0.8406, f1: 0.8504.
[MIRCO] precision: 0.7241, recall: 0.7053, f1: 0.7146
06/07/2022 12:51:58 - INFO - __main__ - Max f1 is: 0.7145747452973374, in step 1392
```

Role1 有所提升

```
06/07/2022 12:52:36 - INFO - transformers.modeling_utils - loading weights file mnt/bert/torch_roberta_wwm/pytorch_model.b
in
06/07/2022 12:52:43 - INFO - src_final.utils.functions_utils - Load ckpt from mnt/xf_event_extraction2020Top1-master/out/final/role2/roberta_wwm_distance_pg_d_enhanced/checkpoint-1092/model.pt
06/07/2022 12:52:50 - INFO - src_final.utils.functions_utils - Use single gpu in: ['0']
Get role task predict logits: 100% [redacted] 2/2 [00:25:00:00, 12.97s/it]
06/07/2022 12:53:16 - INFO - __main__ - In step 1092:
[time] precision: 0.7203, recall: 0.9551, f1: 0.8213.
[loc] precision: 0.5556, recall: 0.6667, f1: 0.6061.
[MIRCO] precision: 0.0912, recall: 0.1196, f1: 0.1035
06/07/2022 12:53:16 - INFO - __main__ - Max f1 is: 0.1035046555800033, in step 1092
```

Role2: 略微下降

## 五、项目优缺点

### 优点:

- 1、舍弃 CRF 结构，采用指针式解码的方案，并利用 trigger 字典。如果未解码出 trigger，则比较句子中匹配知识库的所有 distant trigger 的 start + end logits，选取最大的一个作为解码出的 trigger
- 2、具体而言是论元和属性的抽取结果依赖于触发词，因此只有一步误差传播。
- 3、根据数据的特征，限制解码输出一个 trigger，如果解码出多个 trigger，选取 logits 最大的那个 trigger 作为候选 trigger
- 4、根据数据的特点，发现绝大数都有触发词，故采用 trigger 左右两端动态池化特征作为全局特征；
- 5、因 time loc 并非每个句子中都存在，并且分布较为稀疏，将 time & loc 和 subject & object 的提取分开，采用两个独立的模型进行提取。
- 6、由于样本类别不均极其严重，采用 10 折交叉验证的方法来提升模型的泛化性能。

### 缺点:

- 1、因为测试集中的数据不是每一个都带时间和地点，所以时间和地点抽取效果不是很好，如一个句子里存在两个时间，只能抽取前面的时间，但比赛数据的测试集只有一个时间效果很好，只有两条数据集有地点；
- 2、由于测试集部分数据的 trigger 词在后面，sub 和 obj 识别顺序会颠倒；
- 3、特殊测试集 trigger 与数据集 trigger 库不一致情况下，无法抽取 trigger，但可以在 trigger 字典库中手动添加

## 六、未来展望

在技术层面，由于事件复杂的内部结构，需要依靠专家系统设计事件框架，目前仍然没有形成通用的事件框架体系。另外，依靠人工标注语料数据不仅耗时费力而且成本高昂，导致现有的事件语料数据规模不大、类型较少。现阶段各个类型的事件抽取任务性能较低，不能满足产业应用的需要。

语言层面，中文语言表述的灵活多样给事件抽取任务带来很大的挑战，依赖的底层自然语言处理技术，如分词、命名实体识别、句法分析等在中文语言上的性能都会影响到事件抽取的结果。基于神经网络的事件抽取方法自动提取特征，避免了人工设计特征的繁琐工作。事件抽取任务被构建成端到端的系统，使用包含了丰富语言特征的词向量作为输入，减少了底层自然语言处理工具带来的误差。