# 文本分类任务

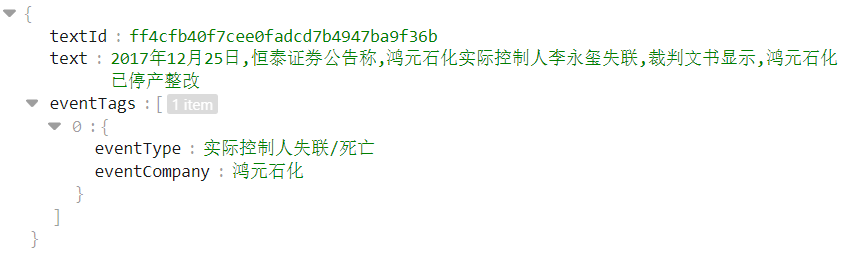
1. 实验目的：熟悉Pretrain的基本流程。
2. 任务说明：比赛方案是pipeline形式，
   1. 第一步NER任务（抽取文本中的公司名称）已经完成。
   2. 第二步是文本分类，拿到公司名称和文本，分类得到最终的公司事件类型。
3. 最终需要给张裕欣博士提交一份代码和结果。
4. 评测指标：精确率（Precision, P）、召回率（Recall, R）、F1值（F1-measure, F1）
   1. 采用**微平均计算F值**即所有样本一起计算P和R。最后，整体的F1值将按照不包含M-N中事件类型的样本F1值乘权重0.3，以及包含M-N中事件类型的样本F1值乘权重0.7进行累加。

## 数据集

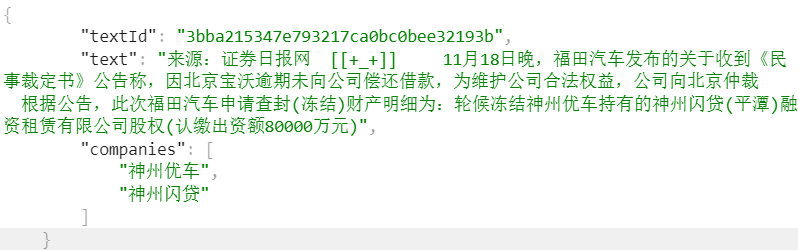
1. 给定100+事件类型及其事件主题公司，训练数据中对其中16个事件类型（见下方备注）只保留10条左右的训练样本。测试集中包含这16个类型的大量待抽取样本。该任务旨在通过部分标注事件对未标注或者欠标注的事件类型和事件主体进行预测。
2. 备注：欠标注的事件类型包括”关闭分支机构”, “澄清辟谣”, “公司涉及传销”, “产品质量问题”, “重大安全事故”, “非法集资”, “第一大股东变化”, “财务报表更正”, “授信额度上升”, “产品虚假宣传”, “实际控制人违规”, “停产停业”, “引入新股东”, “资金紧张”, “债务融资失败”, “频繁动产融资”。**测试数据集中针对以上事件类型将进行加权评估**。
3. 数据示例：

| 输入：一段文本T  输出：文本T中所有的类型以及每个事件类型的事件主体公司  示例：  输入：{“textId”:”123456”, “text”:“众巢股份正式启动上市”}  输出：{“textId”:”123456”,” eventTags”:[{“eventType”: “上市”, “eventCompany”: “众巢股份”}] } |
| --- |

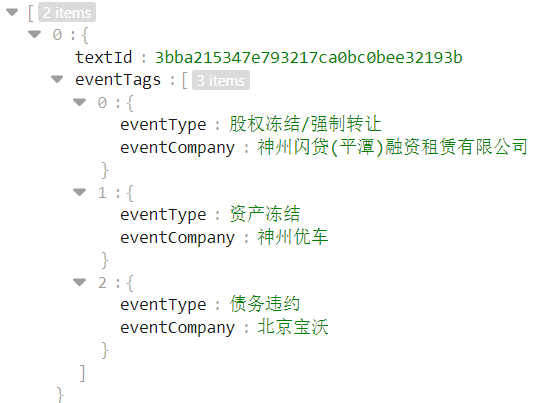
1. 训练集



1. 测试集



1. 提交结果示例



## 实验过程

1. 在test\_nered.json中“text”抽取出“eventType”和“eventCompany”，其中"companies"是候选的公司名称。
2. 一个text中可能包含多个事件，两个公司名称可以对应多个事件。

### 方法：基于BERT的文本分类

将事件类型识别看作一个173分类任务，其中输入是text+company，输出是eventType。

#### 数据预处理

将 train.jsonl 按照7:2:1的比例划分成规整的 train.jsonl, val.jsonl, test.jsonl.

格式为：text,eventCompany\_!\_eventTypeId

| 2017年12月25日,恒泰证券公告称,鸿元石化实际控制人李永玺失联,裁判文书显示,鸿元石化已停产整改,鸿元石化\_!\_0 |
| --- |

对于文本分类这个场景来说，只需要构造原始文本对应的Token序列，并在首位分别再加上一个[CLS]符和[SEP]符作为输入即可。

数据集一共包含有61967条数据，173个类别。示例数据：

| 2017年12月25日,恒泰证券公告称,鸿元石化实际控制人李永玺失联,裁判文书显示,鸿元石化已停产整改,鸿元石化\_!\_0 |
| --- |

其中\_!\_左边为事件描述，也就是后面需要用到的分类文本，右边为类别标签。

#### 数据集构建

##### 第1步：定义tokenize

第1步需要完成的就是将输入进来的文本序列tokenize到字符级别。对于中文语料来说就是将每个字和标点符号都给切分开。在这里，我们可以借用transformers包中的BertTokenizer方法来完成。

##### 第2步：建立词表

由于BERT预训练模型中已经有了一个给定的词表（vocab.txt），因此我们并不需要根据自己的语料来建立一个词表。当然，也不能够根据自己的语料来建立词表，因为相同的字在我们自己构建的词表中和vocab.txt中的索引顺序肯定会不一样，而这就会导致后面根据token id 取出来的向量是错误的。

进一步，我们只需要将vocab.txt中的内容读取进来形成一个词表即可

##### 第3步：转换为Token序列

在得到构建的字典后，我们需要将训练集、验证集和测试集转换成Token序列。

##### 第4步：padding处理与mask

从第3步的输出结果看出，在对原始文本序列tokenize转换为Token ID后还需要对其进行padding处理。

##### 第5步：构造DataLoader与使用示例

经过前面4步的操作，整个数据集的构建就算是已经基本完成了，只需要再构造一个DataLoader迭代器即可。

#### 加载预训练模型

1. 查看模型参数：参数pytorch\_model.bin被载入后变成了一个有序的字典OrderedDict，并且其中一共有207个参数，其名字分别就是列表中的各个元素。
2. 载入模型配置：通常情况下我们都会定义一个配置类，然将这些参数从config.json载入后来实例化这个配置类，接下来便可以以访问类成员的方式来使用这些参数。
3. 载入并初始化：我们只需要通过如下方式便可以返回一个通过bert-base-chinese初始化的BERT模型：
   1. bert = BertModel.from\_pretrained(config, bert\_pretrained\_model\_dir)

#### 文本分类前向传播

1. 在BERT网络模型的基础之上来添加一个分类层，来完成最后的文本分类任务
2. 模型预测结果输出到一个文件中。

## 在线结果

