**Situation Entity Type Classification组件设计文档\_中文数据**

1. 任务定义
   1. 组件研发目标

判断句子的situation entity type，输入一个句子，使用深度学习模型对其做situation entity type classification。在统一的框架下对基于深度学习的分类模型进行实现和封装，对外提供多种类型的接口：

1. **编程调用接口**：待开发
2. **模型训练接口**：待开发
3. **服务接口**：包括离线服务和在线服务：离线服务适用于大批量的句子分类，提供python可运行代码；在线服务适用于单个句子和小批量句子分类，提供 HTTP POST 调用接口。
   1. 输入输出
      1. 模型训练接口

**输入**：训练数据文件、模型名及超参数、运行参数

**输出**：模型实例

* + 1. 服务接口（离线python调用）

**Python 调用示例**

**import os**

**import json**

**import torch**

**from transformers import BertTokenizer**

**from model.sentence\_level\_BERT import MyModel**

**pre\_train\_path = 'pre\_train\_cn'**

**def do\_classification(data):**

**seType\_dict = ["no\_event", "event"]**

**if type(data) is str:**

**data = {'sentence': data}**

**if type(data) is dict:**

**data = [data]**

**model\_for\_prediction = MyModel(dropout=0.5, num\_labels=2, pre\_train\_path=pre\_train\_path).load(os.getcwd() + '/BERT\_1\_cn.pt')**

**tokenizer = BertTokenizer.from\_pretrained(pre\_train\_path)**

**for i in range(len(data)):**

**sentence = data[i]['sentence']**

**inputs = tokenizer(sentence, return\_tensors="pt").input\_ids.cuda()**

**print("inputs.shape: ", inputs.shape)**

**label = torch.tensor(0).unsqueeze(0).cuda()**

**pre\_label, loss = model\_for\_prediction(inputs, label)**

**label = seType\_dict[pre\_label]**

**data[i]['situation entity type'] = label**

**return data**

**if \_\_name\_\_ == '\_\_main\_\_':**

**data0 = {'sentence': "马龙赢了"}**

**data1 = [**

**{'sentence': "马龙赢了"},**

**{'sentence': "地球有多大"}]**

**data2 = "地球有多大"**

**data3 = [{'sentence': "地球有多大"}]**

**data = data2**

**print("input: ", json.dumps(data, sort\_keys=False, indent=4, separators=(',', ':'), ensure\_ascii=False))**

**result = do\_classification(data) # from str to [dict, dict, dict, ...]**

**print("type(result): ", type(result))**

**print(result)**

**print(json.dumps(result, sort\_keys=False, indent=4, separators=(',', ':'), ensure\_ascii=False))**

* + 1. 服务接口（离线batch调用）

代码与1.2.2相同

* + 1. 服务接口（在线）

HTTP 方法: POST

⚫ 请求 URL: http://127.0.0.1:8000/user

⚫ URL 参数：无

⚫ 请求格式说明：

POST 方式调用

使用 JSON 格式的结构体或一个字符串来描述一个请求的具体内容

body 整体文本内容只支持 UTF-8 编码格式。

⚫ 请求参数

|  |  |  |
| --- | --- | --- |
| 字段名 | 值类型 | 说明 |
| sentence | 字符串 | 单个句子token数目不超过510 |

⚫ body 请求示例

示例1

[

{

"sentence":"地球有多大"

},

{

"sentence":"马龙赢了"

}

]

示例2

{

"sentence":"冬奥开幕"

}

示例3

"冬奥开幕"

**返回说明**

⚫ 返回格式：

JSON 格式

返回内容为 UTF-8 编码

⚫ 返回参数

|  |  |  |
| --- | --- | --- |
| 字段名 | 值类型 | 说明 |
| situation entity type | 字符串 | 句子的situation entity type |

⚫ 返回实例

[

{

"sentence":"冬奥开幕",

"situation entity type":"event"

},

{

"sentence":"地球有多大",

"situation entity type":"no\_event"

}

]

**Python 调用示例**

**import json**

**import time**

**import requests**

**if \_\_name\_\_ == '\_\_main\_\_':**

**data0 = {'sentence': "马龙赢了"}**

**data1 = [**

**{'sentence': "马龙赢了"},**

**{'sentence': "地球有多大"}]**

**data2 = "地球有多大"**

**data3 = [{'sentence': "地球有多大"}]**

**data = json.dumps(data1) # data become str**

**t = time.time()**

**try:**

**r = requests.get("http://127.0.0.1:8000/user/" + data)**

**result = json.loads(r.text) # from str to [dict, dict, dict, ...]**

**print("type(result): ", type(result))**

**print(result)**

**print(json.dumps(result, sort\_keys=False, indent=4, separators=(',', ':'), ensure\_ascii=False))**

**except:**

**print('network instability, try again')**

**print(time.time()-t)**

**网页调用示例**

**[http://192.168.14.14:8000/user/”地球有多大”](http://192.168.14.14:8000/user/%22地球有多大%22)**

* 1. 数据交换与模型持久化

1) 算法完善：随着研究领域的发展，本组件将持续添加新的实体消解算法。

2) 模型实例扩展：后续将提供特定领域的模型实例，以满足实验室的研发和科研需求。

1. 架构设计
2. 规约设计

描述常用模型在公开数据集上的抽取性能（质量）和效率（速度）。

各实体模型在基准数据集上达到相应论文所描述的性能指标。

**模型性能**

质量：Precision、F1-Score

模型性能说明

|  |  |  |  |
| --- | --- | --- | --- |
| 模型名称 | 数据集名称 | Precision | F1-Score |
| sentence-level BERT | Weibo\_event（中文） | 0.88 | 0.87 |

**运行速度**

速度：每秒钟处理的文本句子数量。Sent/Sec

模型效率说明

|  |  |  |  |
| --- | --- | --- | --- |
| 模型名称 | 调用方式 | 是否使用 GPU | Sent/Sec |
| sentence-level BERT | 在线单句 | 是 | 0.5 |
| sentence-level BERT | 离线单句 | 是 | 0.6 |
| sentence-level BERT | 在线batch | 是 | 1.3 |
| sentence-level BERT | 离线batch | 是 | 1.4 |