有毒无毒二分类检测模型-----说明文档

1、验收交付文件说明

（1）有毒无毒二分类检测模型

使用的中文预训练模型型号为chinese-electra-180g-small-discriminator，接下游分类任务后将模型训练好的权重信息保存为PyTorch 的默认模型文件格式.pth。

（2）有毒文本数据（10W敏感数据.csv）

收集的有毒文本数据，共计100220条，覆盖5类31种风险。数据来源如下：

1. 违反社会主义核心价值观的这一类数据主要来自《九评共产党》、《共产主义的终极目的》、《解体党文化》、《新中国联邦宣言》、《关于中国的一百个常识》、微信敏感词2019、网易前端过滤敏感词库 、小红书审查百科：习近平敏感词库、Twitter等。
2. 歧视性内容这一类数据主要来自于中文冒犯性语言检测数据集COLD、新浪微博性别歧视评论数据集SWSR、 英文仇恨言论数据集译文、“美国之音”、“西藏之音”、Twitter、YouTube等
3. 其他包含中国互联网联合辟谣平台的常见谣言、自编关于领土争端等争议性问题等。

（3）关键词库（关键词库.csv）

使用联想不良关键词列表，经百度智能云平台过滤后共含9298个敏感词，该关键词库均为高度敏感词语，一旦在句子中出现，即可判别句子为敏感。

（4）测试脚本（script_for_two_classification.py）

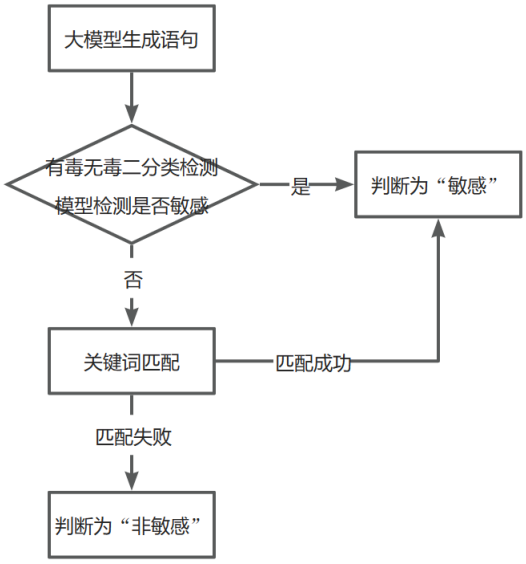
2、算法基本原理

（1）算法前处理——过滤非中文

添加模型的前处理，过滤了非中文字符，只保留输入的中文字符以及阿拉伯数字。

（2）算法后处理——关键词匹配

为进一步增强安全性，添加了“关键词匹配”模块。**当输入语句经有毒无毒检测模型判断为“非敏感”后，为进一步过滤敏感语句，该语句将与关键词库（关键词库.csv——含9300个敏感词）中的敏感词进行匹配，若匹配成功，则该语句输出为“敏感”，反之则为“非敏感”。**经有毒无毒检测模型直接判断为“敏感”的语句无需与关键词库中的敏感词进行匹配。具体流程如下：



3、脚本使用说明

修改模型路径model_dir（解压后模型文件夹的绝对路径）、修改模型保存路径.pth文件（解压后模型文件夹中.pth文件绝对路径）、关键词库绝对路径word_bank。输入选择1.gpu/2.cpu进行测试，随后可选择自定义的测试集或者单条输入进行预测。

```python
# 修改模型文件夹绝对路径
    parser.add_argument('--model_dir', type=str, default='')
# 修改模型保存路径.pth文件
    parser.add_argument('--model_save_path', type=str, default='', help="Trained model save path")
# 修改关键词库路径
    parser.add_argument('--word_bank', type=str, default='', help="word_bank_path")
```

4、脚本详解

（1）算法前处理

过滤非中文字符，只保留中文字符以及阿拉伯数字。

```python
# 只保留中文和阿拉伯数字
def keep_chinese(text):
    pattern = re.compile(r'[^\u4e00-\u9fa50-9, ,。.:： ;; ? ?!! ]')  # 匹配非中文和数字的正则表达式
    chinese_and_numbers_text = re.sub(pattern, '', text)  # 使用正则表达式替换非中文和数字为空字符串
    return chinese_and_numbers_text
```

（2）算法后处理——关键词匹配

为进一步确保检测模型的安全性，添加了“关键词匹配”模块。经二分类模型判断为“非敏感”的语句与敏感词库中的敏感词进行匹配，若一旦含有任一敏感词，则重新判断为“敏感”，代码部分如下。后续我们也将继续寻找其他算法加强安全保障。

```python
#敏感词库匹配
def word_bank_match(text):
    # 从CSV文件中读取敏感词库
    word_bank = args.word_bank
    df = pd.read_csv(word_bank)
    flag = False
    # 将其转换为列表
    keyword_list = df['text'].tolist()
    # 预编译正则表达式
    pattern_list = [re.compile(re.escape(keyword)) for keyword in keyword_list]

    start_time = time.time()  # 记录匹配开始时间
    # 在匹配过程中使用预编译的正则表达式
    for pattern, keyword in zip(pattern_list, keyword_list):
        if pattern.search(text):
            print(f"发现敏感词：{keyword}")
            end_time = time.time()  # 记录匹配结束时间
            matching_time = end_time - start_time  # 计算匹配所用时间
            print(f"匹配共用时：{matching_time * 1000}ms")
            flag = True
    return flag
```

（3）脚本主体框架

数据预处理GenDateSet()、自定义模型结构MinirbtForPairwiseCLS()、模型加载load_model()、分词器选择tokenizer_choose()、敏感词库匹配word_bank_match()、功能选择function_choose()。