实战案例7：基于标题对短视频进行分类

作者： Robin
日期： 2018/10

1. 案例描述

面对大规模的短文本形式的数据，如何快速而准确地从中获取所需的关键信息，进行文本挖掘或商业挖掘，短文本分类技术发挥着非常重要的作用。基于标题对短视频进行分类项目旨在结合中文分词及机器学习的相关内容完成短文本分类任务，包括文本的预处理、特征表示及构建分类器等知识点。该项目的实践有助于强化学员对文本相似度、文本特征及机器学习相关技术内容的理解和应用。

2. 数据集描述

- 数据文件
  - **iqiyi_title_info.csv**: 爱奇艺短视频标题数据集，其中每行一则短视频标题信息

3. 任务描述

- 使用文本处理，特征提取及机器学习方法进行短视频标题分类

4. 主要代码解释

- 代码结构

```
lect07_proj
├── dataset
    ├── iqiyi_title_info.csv   # 短视频标题数据集
    ├── stop_words         # 停用词存放的目录
        ├── 中文停用词库.txt
        ├── 哈工大停用词表.txt
        ├── 四川大学机器智能实验室停用词库.txt
    ├── output         # 停用词存放的目录
        ├── proc_title.csv   # 处理后的分词结果
├── main.py      # 主程序
├── utils.py      # 工具文件，包含文本预处理、特征工程等
├── config.y     # 配置文件
├── lect07_proj_readme.pdf   # 案例讲解文档
```

- **utils.py**

对每个文本进行预处理操作，包括：

1. 去除非中文字符
2. 结巴分词+词性标注

3. 去除停用词

预处理后的文本为“空格”隔开单词的单词

```
def prepare_data():
    ...
    # 中文文本分词
    title_df['words'] = title_df['title'].apply(preprocess_text, args=(stopwords,))
    ...
```

- **utils.py**

在该案例中使用了两种文本特征：

1. TF-IDF
2. 词袋模型

最终的特征为两种特征的合并。注意，为节省空间，sklearn中的文本特征提取结果为稀疏矩阵，如果需要对特征进行操作，需要使用to_array()将其转换为普通ndarray

```
def do_feature_engineering(train_data, test_data):
    ...
    # TF-IDF特征提取
    tfidf_vectorizer = TfidfVectorizer(max_features=config.n_common_words)
    train_tfidf_feat = tfidf_vectorizer.fit_transform(train_proc_text).toarray()
    test_tfidf_feat = tfidf_vectorizer.transform(test_proc_text).toarray()

    # 词袋模型
    count_vectorizer = CountVectorizer(max_features=config.n_common_words)
    train_count_feat = count_vectorizer.fit_transform(train_proc_text).toarray()
    testcount_feat = count_vectorizer.transform(test_proc_text).toarray()
    ...
```

5. 案例总结

- 该项目通过使用常用的文本预处理及特征提取操作，实现了根据短视频标题对其进行分类，包含了如下内容:
    - 文本预处理：去除特殊字符，分词，去除停用词
    - 文本特征提取：TF-IDF，词袋模型（词频统计）

6. 课后练习

- 修改代码，对比使用一种特征和多种特征对模型性能的影响

参考资料

1. sklearn文本处理
2. sklearn文本特征
3. NLTK

4. 结巴分词