邓娜代码分析

事件分析

feature_extraction.py

利用卡方检验来判断某个词是不是情感词——即该词与正负情感有没有关系。

https://www.jianshu.com/p/807b2c2bfd9b

ChiSquare类

- `__init__()`：利用文档标签和文档单词构造总"单词-频数”字典、正"单词-频数”字典、负"单词-频数”字典。
- `__calculate()`：进行卡方检验，计算得到的卡方值作为特征得分pos_score。计算公式为：

$$X^2 = \sum \frac{(\text{observed - expected})^2}{\text{expected}}$$

- `best_words`：按特征得分对单词从大到小排序，取出num个特征词。

Corpus.py

构建语料库。

Corpus类

- `__init__()`：读入包含 *带正负标签的文档* 的文件构造正负文档数组。
- `get_corpus()`：从正负文档数组中选取数量相同的数据（从start到end），返回数据和数据标签(1或0)。
- `get_train_corpus()`：调用 `get_corpus()` 函数，返回0到num这么多的数据及标签。
- `get_corpus()`：调用 `get_corpus()` 函数，返回train_num到train_num + num这么多的数据及标签。
- `get_all_corpus()`：返回所有数据和数据标签(1或0)。

派生类

以下类均是继承自Corpus类，初始化时指定了语料库文件位置：

MoiveCorpus类

Moive2Corpus类

WaimaiCorpus类

Waimai2Corpus类

HotelCorpus类

准备语料库相关函数

准备语料库txt：从本地的数据源得到数据，并在f_corpus中生成对应的xx_corpus.txt文件，便于后续使用。

`get_movie_corpus()`

`get_movie2_corpus()`

`get_hotel_corpus()`

`get_waimai_corpus()`

测试函数

`test_corpus()`：测试以上派生类。

classifiers.py

分类器。

DictClassifier类

- `__init__()`：准备用户词典、情感词典、其他词典。
- `classify()`：调用`analyse_sentence()`函数进行分类。
- `analysis_file()`：对整个文件分析。循环调用`analyse_sentence()`函数对单个句子分析。
- `analyse_sentence()`：将评论分句，对每个子句进行情感分析；根据最后得分的正负返回1或0.
- `__analyse_clause()`：判断各种句式，然后逐个分析分词判断类型，然后综合连词的情感值和标点符号的情感值。返回子句分析的数据结构。

```
sub_clause =
{
  "score": 0,
  "positive": [],
  "negative": [],
  "conjunction": [],
  "punctuation": [],
  "pattern": []
}
```

判断句式函数

- `__is_clause_pattern1()`
- `__is_clause_pattern2()`
- `__is_clause_pattern3()`

分析单词及判断词类型

- `__analyse_word()`：调用以下函数判断词的类型。

查找对应词典，看词是不是在词典中来判断类型。

- __is_word_conjunction
- __is_word_punctuation
- __is_word_positive
- __is_word_negative

情感词分析

`__emotional_word_analysis()`：在情感词典内，构建一个以情感词为中心的字典数据结构：

```
orientation =
{
  "key": core_word,
  "adverb": [],
  "denial": [],
  "value": value
}
```

在三个前视窗内，分别判断是否有否定词、副词、情感词，更新数据结构，添加情感分析值，最后返回该数据结构。

分句相关函数

- `__divide_sentence_into_clauses()`：先调用 `__split_sentence()` 根据标点符号分割句子，然后识别句式来分句。
- `__split_sentence()`：根据标点符号分割句子。

其他函数

- `__get_phrase_dict()`：读入短语词典。
- `__get_dict()`：读入词典。
- `__write_runout_file()`：写入文件操作。
- `__output_analysis()`：输出分析的数据结构结果。

KNNClassifier类

所谓K最近邻，就是k个最近的邻居的意思，说的是每个样本都可以用它最接近的k个邻居来代表。

kNN算法的核心思想是如果一个样本在特征空间中的k个最相邻的样本中的大多数属于某一个类别，则该样本也属于这个类别，并具有这个类别上样本的特性。

- `__doc2vector()`：将文档中的单词转换为向量，向量维数为单词总数，每一维是对应单词的出现频率。

略。

BayesClassifier类

- `__train()`：输入训练数据，获得每个单词的频数和概率信息。
- `classify()`：用正负情感词的概率之和作为情感得分（分别计算pos_score和neg_score），如果pos_score大于neg_score则输出正向，否则输出反向。

MaxEntClassifier类

略。

SVMClassifier类

- `words2vector()`：词转化为词向量，向量值根据特征词的频数得到。
- `__train()`：输入训练向量，调用SVC的fit函数进行训练。
- `classify()`：词转化为词向量，调用SVC的predict函数进行预测。
- `predict_proba()`：调用SVC的predict_proba函数进行预测。
  - predict是训练后返回预测结果，是标签值。
  - predict_proba返回的是一个 n 行 k 列的数组， 第 i 行 第 j 列上的数值是模型预测 第 i 个预测样本为某个标签的概率，并且每一行的概率和为1。

tools.py

Write2File类

- `append(), write()`：将结果写入文件（普通文本）中
- `write_contents()`：将准确率写入xls表格中
- `write_results()`：将结果写入xls表格中

`get_accuracy()`：计算分析后标签的准确率（利用F-measure方法对比实际标签和预测标签）

test.py

测试：分别测试各个方法，各个语料库。

过程：

1. 获取训练、测试语料集
2. 特征提取
3. 基于不同方法进行分类
4. 计算准确率结果