# R包展示

## 文本挖掘系列—tm包

数据分析工具实践

祁弦 1600022701
梁傅淇 1600022754


tm

# tm简介

- tm(text mining)包是R语言中为**文本挖掘**提供综合性处理的package。

- tm包可以实现：数据输入、文集处理、元数据管理，创建单词-文本矩阵。

# tm安装及调用



- tm包的安装方式是 install.packages(“tm”) 【直接下载、安装】
- tm包依赖于其他的packages，如：NLP\slam\BH
- 安装完成后，可通过 library(“tm”) 载入tm包，即可调用tm包函数
- vignette(“tm”) 函数会打开tm.pdf的英文文件，其中包含tm包的函数与使用方法

# 使用tm包进行文本挖掘流程：载入数据

——以对苹果公司评价的tweet为例

## // 载入数据

```
> tweets = read.csv("tweets.csv", stringsAsFactors=FALSE)
```

**read.csv()**

**当使用R做文本分析的时候，需要加上stringAsFactors这个参数，这样文本才会被妥善地读入。**

### 数据集(tweets.csv)介绍

tweets.csv包含了1181条数据，其中，Tweet列是真实的推特文本，Avg列是推特文本的情感评分（-2到2，代表强烈负面到强烈正面的情绪）。此数据集由MITx: 15.071x The Analytics Edge 课程组提供。

# 使用tm包进行文本挖掘流程：查看数据

——以对苹果公司评价的tweet为例

## // 查看数据情况

```
> str(tweets)
'data.frame':   1181 obs. of  2 variables:
 $ Tweet: chr  "I have to say, Apple has by far the best customer care service I have ev
 $ Avg  : num  2 2 1.8 1.8 1.8 1.8 1.8 1.6 1.6 1.6 ...
```

**str()**

# 使用tm包进行文本挖掘流程：处理数据

——以对苹果公司评价的tweet为例

## // 新增Negative列，对于Avg小于-1的，Negative列设为TRUE

```
> tweets$Negative = as.factor(tweets$Avg <= -1)
> tweets$Negative
   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  [20] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  [39] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  [58] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  [77] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
  [96] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [115] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [134] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [153] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [172] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [191] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

**在这里，我们研究负面情绪较强烈的tweets。**

# 使用tm包进行文本挖掘流程：处理数据

——以对苹果公司评价的tweet为例

## // 创建*Corpus*

```
> corpus = Corpus(VectorSource(tweets$Tweet))
> corpus
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 1181
```

**Corpus()**
**VectorSource()**

### Corpus介绍

Corpus是tm包引入的一个概念，意思就是**文集**。这里我们将我们所有tweet集合成一个文集。

# 使用tm包进行文本挖掘流程：处理数据

——以对苹果公司评价的tweet为例

**// 数据预处理：小写化/移除标点符号**

```
> corpus =  tm_map(corpus, tolower)
> corpus =  tm_map(corpus, removePunctuation)
```

**// 数据预处理：移除公司名字/移除停止词**

```
> corpus = tm_map(corpus, removeWords, c("apple", stopwords("english")))
```

**tm_map()**

**Stopwords**

停止词，a，the，or等使用频率很多，但又无实际意义的字或词，常为冠词、介词、副词或连词等。

# 使用tm包进行文本挖掘流程：处理数据

——以对苹果公司评价的tweet为例

**// 数据预处理：去除词尾（如-er，e等）**

```
> corpus = tm_map(corpus, stemDocument)
```

**tm_map()**

**词尾**

在文本挖掘的过程中，类似-ly，-er，-e结尾的词语的词尾会被去除。有专门的算法识别词根，并去除词尾。

# 使用tm包进行文本挖掘流程：处理数据

——以对苹果公司评价的tweet为例

## // 整合成矩阵，探索词频

```
> frequencies = DocumentTermMatrix(corpus)
> frequencies
<<DocumentTermMatrix (documents: 1181, terms: 3289)>>
Non-/sparse entries: 8980/3875329
Sparsity           : 100%
Maximal term length: 115
Weighting          : term frequency (tf)
> inspect(frequencies[1000:1005,505:515])
<<DocumentTermMatrix (documents: 6, terms: 11)>>
Non-/sparse entries: 1/65
Sparsity           : 98%
Maximal term length: 23
Weighting          : term frequency (tf)
Sample             :
      Terms
```

**DocumentTermMatrix()**
**inspect()**

# 使用tm包进行文本挖掘流程：处理数据

——以对苹果公司评价的tweet为例

**// 将在0.05%以上tweet文本出现过的词语保留下来**

```
> sparse = removeSparseTerms(frequencies, 0.995)
> tweetsSparse = as.data.frame(as.matrix(sparse))
> tweetsSparse$Negative = tweets$Negative
```

**removeSparseTerms()**

**去除词语的用意**

我们将只出现过一次或者很少次数的词语去掉，能够减少变量，集中精力在影响更大的词语上。

# 使用tm包进行文本挖掘流程：模型预测

——以对苹果公司评价的tweet为例

**此部分不使用tm包的功能**
**需要使用到CaTools包**

## // 分割数据集：训练集和测试集

```
> set.seed(123)
> split = sample.split(tweetsSparse$Negative, SplitRatio = 0.7)
> trainSparses = subset(tweetsSparse, split==TRUE)
> testSparses = subset(tweetsSparse, split==FALSE)
```

## // 模型预测

**此部分不使用tm包的功能**
**需要使用到rpart包**

```
> tweetCART = rpart(Negative~., data=trainSparses, method="class")
> predictCART = predict(tweetCART, newdata=testSparses, type="class")
> table(testSparses$Negative, predictCART)
       predictCART
        FALSE TRUE
  FALSE   294    6
  TRUE     37   18
> (294+18)/(294+6+37+18)
[1] 0.8788732
>
```

# 其他示例

## 数据处理示例

```
> reuters <- tm_map(DATANAME,  as.PlainTextDocument)    //去除标签
> reuters <- tm_map(DATANAME,  stripWhitespace)         //去除空格
```

注：在这里需要注意的是，如果使用中文分词法，由于词之间无有像英文一样的空隔，好在有Java已经解决了这样的问题，我们只需要在R-console里加载rJava与rmmseg4j两个工具包即可。如：

```
>mmseg4j("中国人民从此站起来了")
 [1] 中国  人民  从此  站  起来
```

# 其他示例

## 矩阵处理示例

> **findFreqTerms(dtm, 5)** //寻找**dtm**中出现5次以上的词语

显示示例如下：

```
[1] "15.8"  "accord"  "agency"  "ali"
[5] "analysts"  "arab"  "arabia"  "barrel."
[9] "barrels"  "bpd"  "commitment"  "crude"
[13] "daily"  "dlrs"  "economic"  "emergency"
[17] "energy"  "exchange"  "exports"  "feb"
[21] "futures"  "government"  "gulf"  "help"
[25] "hold"  "international"  "january"  "kuwait"
[29] "march"  "market"
```

# 其他示例

如果需要考察多个文档中特有词汇的出现频率，可以手工生成字典，并将它作为生成矩阵的参数：

```
> d <- Dictionary(c("prices", "crude", "oil")))
> inspect(DocumentTermMatrix(reuters, list(dictionary = d)))
```

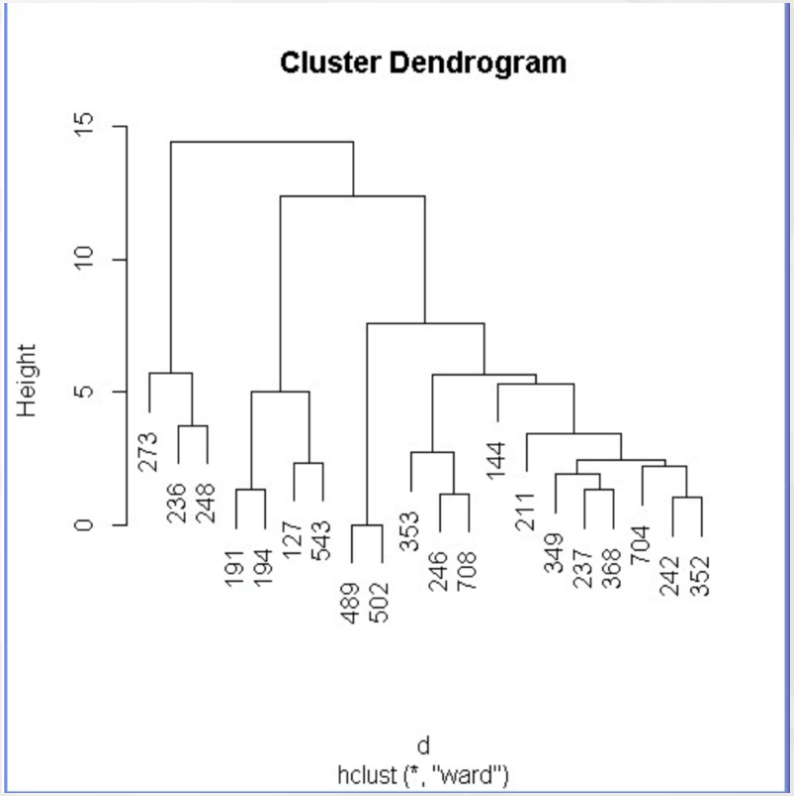
因为生成的term-document矩阵是一个稀疏矩阵，再进行降维处理，之后转为标准数据框格式：

```
//parse值越少，最后保留的term数量就越少
> dtm2 <- removeSparseTerms(dtm, sparse=0.95)
//最后将term-document矩阵生成数据框就可以进行聚类等操作了。
> data <- as.data.frame(inspect(dtm2))
```

# 其他示例

## 聚类示例

```
> data.scale <- scale(data)
> d <- dist(data.scale, method = "euclidean")
> fit <- hclust(d, method="ward")
>plot(fit)    //图形见右
```



# 作业

**以下哪个函数是用来去除停止词的？**

**A. tm_map(DATA, tolower)**

**B. tm_map(DATA, removePunctuation)**

**C. tm_map(DATA, removewords, stopwords(“english”))**

**D. tm_map(DATA, stripWhitespace)**


谢
谢
观
看