Red_LDAvis.R

daitu

Fri Nov 4 00:51:19 2016

```
## 红楼梦文本挖掘之数据预处理####
## 主要用于文本文档的读取和构建
## LDA模型
## 孙玉林；2016年10月31


## 加载所需要的包
library(jiebaR)
```

```
## Loading required package: jiebaRD
```

```
library(tm)
```

```
## Loading required package: NLP
```

```
library(readr)
library(stringr)
library(lda)
library(LDAvis)


## 读取所需要的文件####
## 读取停用词
filename <- "./数据/我的红楼梦停用词.txt"
mystopwords <- readLines(filename)
## 读取红楼梦
filename <-"./数据/红楼梦UTF82.txt"
Red_dream <- readLines(filename,encoding='UTF-8')
```

```
## Warning in readLines(filename, encoding = "UTF-8"): 读'./数据/红楼梦
## UTF82.txt'时最后一行未遂
```

```
## 将读入的文档分章节####
#去除空白行
Red_dream <- Red_dream[!is.na(Red_dream)]
## 删除卷数据
juan <- grep(Red_dream,pattern = "^第+.+卷")
Red_dream <- Red_dream[(-juan)]
## 找出每一章节的头部行数和尾部行数
## 每一章节的名字
Red_dreamname <- data.frame(name = Red_dream[grep(Red_dream,pattern = "^第+.+回")],
                            chapter = 1:120)
## 处理章节名
names <- data.frame(str_split(Red_dreamname$name,pattern = " ",simplify =TRUE))
Red_dreamname$chapter2 <- names$X1
Red_dreamname$Name <- apply(names[,2:3],1,str_c,collapse = ",")
## 每章的开始行数
Red_dreamname$chapbegin<- grep(Red_dream,pattern = "^第+.+回")
## 每章的结束行数
Red_dreamname$chapend <- c((Red_dreamname$chapbegin-1)[-1],length(Red_dream))
## 每章的段落长度
Red_dreamname$chaplen <- Red_dreamname$chapend - Red_dreamname$chapbegin
## 每章的内容
for (ii in 1:nrow(Red_dreamname)) {
  ## 将内容使用句号连接
  chapstrs <- str_c(Red_dream[(Red_dreamname$chapbegin[ii]+1):Red_dreamname$chapend[i
i]],collapse = "")
  ## 剔除不必要的空格
  Red_dreamname$content[ii] <- str_replace_all(chapstrs,pattern = "[[:blank:]]",repla
cement = "")
}
## 每章节的内容
content <- Red_dreamname$content
Red_dreamname$content <- NULL
## 计算每章有多少个字
Red_dreamname$numchars <- nchar(content)
```

```
## 对红楼梦进行分词####
Red_fen <- jiebaR::worker(type = "mix",user = "./数据/红楼梦词典.txt")
Fen_red <- apply_list(as.list(content),Red_fen)
## 去除停用词,使用并行的方法
library(parallel)
cl <- makeCluster(4)
Fen_red <- parLapply(cl = cl,Fen_red, filter_segment,filter_words=mystopwords)
stopCluster(cl)

# compute the table of terms:
## 计算词项的table
term.table <- table(unlist(Fen_red))
term.table <- sort(term.table, decreasing = TRUE)

# 删除词项出现次数较小的词
del <- term.table < 10
term.table <- term.table[!del]
vocab <- names(term.table)

# now put the documents into the format required by the lda package:
## 将文本整理为lda所需要的格式
get.terms <- function(x) {
  index <- match(x, vocab)
  index <- index[!is.na(index)]
  rbind(as.integer(index - 1), as.integer(rep(1, length(index))))
}
documents <- lapply(Fen_red, get.terms)


# 计算相关数据集的统计特征
D <- length(documents)  # number of documents
W <- length(vocab)  # number of terms in the vocab
doc.length <- sapply(documents, function(x) sum(x[2, ]))  # number of tokens per docu
ment
N <- sum(doc.length)  # total number of tokens in the data
term.frequency <- as.integer(term.table)  # frequencies of terms in the corpus
## 使用lda模型
# MCMC and model tuning parameters:
K <- 10
G <- 1000
alpha <- 0.02
eta <- 0.02

# Fit the model:
library(lda)
set.seed(357)
t1 <- Sys.time()
fit <- lda.collapsed.gibbs.sampler(documents = documents, K = K, vocab = vocab,
                                   num.iterations = G, alpha = alpha,
                                   eta = eta, initial = NULL, burnin = 0,
                                   compute.log.likelihood = TRUE)
t2 <- Sys.time()
t2 - t1  # about 24 minutes on laptop
```

```
## Time difference of 28.78411 secs
```

```
## 对模型进行可视化
theta <- t(apply(fit$document_sums + alpha, 2, function(x) x/sum(x)))
phi <- t(apply(t(fit$topics) + eta, 2, function(x) x/sum(x)))


RED_dreamldavis <- list(phi = phi,
                        theta = theta,
                        doc.length = doc.length,
                        vocab = vocab,
                        term.frequency = term.frequency)

# create the JSON object to feed the visualization:
json <- createJSON(phi = RED_dreamldavis$phi,
                   theta = RED_dreamldavis$theta,
                   doc.length = RED_dreamldavis$doc.length,
                   vocab = RED_dreamldavis$vocab,
                   term.frequency = RED_dreamldavis$term.frequency)

serVis(json, out.dir = 'vis')
```

```
## Warning in dir.create(out.dir): 'vis'已存在
```

```
## Loading required namespace: servr
```

```
# MCMC and model tuning parameters:
K <- 20
G <- 1000
alpha <- 0.02
eta <- 0.02

# Fit the model:
library(lda)
set.seed(357)
t1 <- Sys.time()
fit <- lda.collapsed.gibbs.sampler(documents = documents, K = K, vocab = vocab,
                                   num.iterations = G, alpha = alpha,
                                   eta = eta, initial = NULL, burnin = 0,
                                   compute.log.likelihood = TRUE)
t2 <- Sys.time()
t2 - t1  # about 24 minutes on laptop
```

```
## Time difference of 51.0013 secs
```

```
## 对模型进行可视化
theta <- t(apply(fit$document_sums + alpha, 2, function(x) x/sum(x)))
phi <- t(apply(t(fit$topics) + eta, 2, function(x) x/sum(x)))


RED_dreamldavis <- list(phi = phi,
                        theta = theta,
                        doc.length = doc.length,
                        vocab = vocab,
                        term.frequency = term.frequency)

# create the JSON object to feed the visualization:
json <- createJSON(phi = RED_dreamldavis$phi,
                   theta = RED_dreamldavis$theta,
                   doc.length = RED_dreamldavis$doc.length,
                   vocab = RED_dreamldavis$vocab,
                   term.frequency = RED_dreamldavis$term.frequency)

serVis(json, out.dir = 'vis')
```

```
## Warning in dir.create(out.dir): 'vis'已存在
```