Yelp Challenge Project - Appendix

David Álvarez Pons

Downloading and Extracting data

Downlaoding data.

```
downloadFolderName <- "downloads"
fileName <- "dataset.zip"
filePath <- paste(downloadFolderName, fileName, sep="/")
dataUrl <- "https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/yelp_dataset_challenge_academic_da

if (!file.exists("downloads")){
    dir.create(downloadFolderName)
    download.file(dataUrl, destfile = filePath, method="curl");
}
dir(downloadFolderName)
```

Extracting data.

```
if (length(dir(downloadFolderName)) < 2){
    unzip(filePath, exdir = downloadFolderName)
}
dir(downloadFolderName)
```

JSON data structures.

Source: http://www.yelp.com/dataset_challenge

```
# business.json
{
    'type': 'business',
    'business_id': (encrypted business id),
    'name': (business name),
    'neighborhoods': [(hood names)],
    'full_address': (localized address),
    'city': (city),
    'state': (state),
    'latitude': latitude,
    'longitude': longitude,
    'stars': (star rating, rounded to half-stars),
    'review_count': review count,
    'categories': [(localized category names)]
    'open': True / False (corresponds to closed, not business hours),
    'hours': {
        (day_of_week): {
            'open': (HH:MM),
            'close': (HH:MM)
        },
        ...
```

```
    },
    'attributes': {
        (attribute_name): (attribute_value),
        ...
    },
}

# review.json
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}

# user.json
{
    'type': 'user',
    'user_id': (encrypted user id),
    'name': (first name),
    'review_count': (review count),
    'average_stars': (floating point average, like 4.31),
    'votes': {(vote type): (count)},
    'friends': [(friend user_ids)],
    'elite': [(years_elite)],
    'yelping_since': (date, formatted like '2012-03'),
    'compliments': {
        (compliment_type): (num_compliments_of_this_type),
        ...
    },
    'fans': (num_fans),
}

# checkin.json
{
    'type': 'checkin',
    'business_id': (encrypted business id),
    'checkin_info': {
        '0-0': (number of checkins from 00:00 to 01:00 on all Sundays),
        '1-0': (number of checkins from 01:00 to 02:00 on all Sundays),
        ...
        '14-4': (number of checkins from 14:00 to 15:00 on all Thursdays),
        ...
        '23-6': (number of checkins from 23:00 to 00:00 on all Saturdays)
    }, # if there was no checkin for a hour-day block it will not be in the dict
}

# tip.json
{
    'type': 'tip',
```

```
    'text': (tip text),
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'date': (date, formatted like '2012-03-14'),
    'likes': (count),
}
```

Data preprocessing

Reading JSON files

```
library(jsonlite)
```

```
jsonFiles <- lapply(dir("downloads/yelp_dataset_challenge_academic_dataset/", "*.json", full.names = TRU
    readLines(file)
})
```

Transformation JSON –> lists

```
listFiles <- lapply(jsonFiles, function(fileType){
    t(sapply(fileType, function(jsonFile){
        fromJSON(jsonFile, flatten = TRUE)
    }, USE.NAMES = FALSE))
})
```

Saving lists into RDS

```
dataFolderName <- "data"
if (!file.exists(dataFolderName)){
    dir.create(dataFolderName)
    sapply(listFiles, function(listFile){
        fileName <- paste0(listFile[1,]$type, ".rds")
        filePath <- paste(dataFolderName, fileName, sep="/")
        saveRDS(listFile, file = filePath)
    })
}
```

Exploratory Data Analysis

Counting reviews vs `review_count` file.

```
##       business_id              review_count comp_count
##  [1,] "DP7Ov9gay6NeKbFEdHAgKA" "4"          "4"
##  [2,] "ZIUGkjX2C7a39Ntbm1kUSQ" "12"         "11"
##  [3,] "4_6hH6CJaHwuetBabnnkiQ" "4"          "2"
##  [4,] "vNOLPRovcpO0B2Iq1oSe6g" "4"          "8"
##  [5,] "-c82ZQHqSLPKC1llWHAjGw" "7"          "6"
##  [6,] "y8VphkZ7kojHS0Oyj3JUsQ" "5"          "4"
##  [7,] "ZqJfiK_Vz85FBiTfDuNFzQ" "9"          "8"
##  [8,] "K1cqP1GQ9fLd67SkJHx5FA" "4"          "4"
##  [9,] "lNr0Vb_7BGpV6TVmZgujOA" "3"          "2"
## [10,] "z59lJzU0BoA7dz_itojXAQ" "5"          "5"
```

Cleaning data

Feature selection functions.

```
getBusiness <- function(business_id){
    indexes <- unlist(business[,"business_id"]) == business_id;
    business[indexes,]
}

buildDataReviewBusiness <- function(business_ids){
    sapply(business_ids, function(bi){
        biz <- getBusiness(bi)
        c(biz$name, biz$city, biz$state, biz$open, biz$latitude, biz$longitude)
    }, USE.NAMES = FALSE)
}

buildDataReview <- function(reviews){
    biz.data <- buildDataReviewBusiness(unlist(reviews[,"business_id"]))

    data.frame(
        review_id = unlist(reviews[,"review_id"]),
        business_id = unlist(reviews[,"business_id"]),
        business_name = biz.data[1,],
        business_latitude = biz.data[5,],
        business_longitude = biz.data[6,],
        business_city = biz.data[2,],
        business_state = biz.data[3,],
        business_open = biz.data[4,],
        user_id = unlist(reviews[,"user_id"]),
        date = unlist(reviews[,"date"]),
        votes_funny = sapply(reviews[,"votes"], function(v) v$funny),
        votes_useful = sapply(reviews[,"votes"], function(v) v$useful),
        votes_cool = sapply(reviews[,"votes"], function(v) v$cool),
        text = unlist(reviews[,"text"]),
        stars = unlist(reviews[,"stars"]),
        stringsAsFactors = FALSE
    )
}
```

Subsetting features data set.

```
nBiz <- length(unlist(business[,"business_id"]))
set.seed(1234);
subsetBusiness <- sample(unlist(business[,"business_id"]), nBiz * 0.02)
subsetReviews <- dsReview[dsReview$business_id %in% subsetBusiness,]
saveRDS(subsetReviews, file="data/subset_features.rds")
nrow(subsetReviews)
```

```
dsReview <- buildDataReview(review)
saveRDS(dsReview, file = "data/features.rds")
```

Configuring WordNet database

We will download the database from the corresponding web page https://wordnet.princeton.edu/wordnet/download/

```
wordnetUrl <- "http://wordnetcode.princeton.edu/wn3.1.dict.tar.gz"
dbWordnetFile <- "downloads/wordnet.dict.tar.gz"

if (!file.exists(dbWordnetFile)){
    download.file(wordnetUrl, destfile = dbWordnetFile, method="curl");
}
dir("downloads")
```

Now we extract the downloaded `tar.gz` into the `data` folder.

```
wordnetDir <- "data/Wordnet-3.1"
untar(dbWordnetFile, exdir = wordnetDir)
```

Finally, we need to set up an environment variable so the library knows where to find the data.

```
Sys.setenv(WNHOME = wordnetDir);
```

Data transformations

```
library(tm)
library(wordnet)
library(openNLP)
library(SnowballC)
```

```
sent_token_annotator <- Maxent_Sent_Token_Annotator()
word_token_annotator <- Maxent_Word_Token_Annotator()
pos_tag_annotator <- Maxent_POS_Tag_Annotator()
```

```
convert_text_to_sentences <- function(text, lang = "en") {
    # Convert text to class String from package NLP
    text <- as.String(text)
    # Sentence boundaries in text
    sentence.boundaries <- annotate(text, sent_token_annotator)
    # Extract sentences
    sentences <- text[sentence.boundaries]
    # return sentences
    return(as.character(sentences))
}

splitReviewsIntoSentences <- function(reviews){
    result <- data.frame();
    apply(reviews, 1, function(review){
        sentences <- convert_text_to_sentences(review["text"])
        nSent <- length(sentences)
        result <<- rbind(
```

```
            result,
            data.frame(
                review_id = rep(review["review_id"], nSent),
                business_id = rep(review["business_id"], nSent),
                business_name = rep(review["business_name"], nSent),
                business_latitude = rep(review["business_latitude"], nSent),
                business_longitude = rep(review["business_longitude"], nSent),
                business_city = rep(review["business_city"], nSent),
                business_state = rep(review["business_state"], nSent),
                business_open = rep(review["business_open"], nSent),
                user_id = rep(review["user_id"], nSent),
                date = rep(review["date"], nSent),
                votes_funny = rep(review["votes_funny"], nSent),
                votes_useful = rep(review["votes_useful"], nSent),
                votes_cool = rep(review["votes_cool"], nSent),
                text = sentences,
                stars = rep(review["stars"], nSent),
                stringsAsFactors = FALSE
            )
        )
        remove(sentences)
    })
    rownames(result) <- NULL
    result
}
```

```
reviewSentences <- splitReviewsIntoSentences(subsetReviews)
saveRDS(reviewSentences, file = "data/sentences.rds")
```

```
tagPOS <- function(x) {
    if (nchar(x) == 0) return("")
    y1 <- annotate(x, list(sent_token_annotator, word_token_annotator))
    y2 <- annotate(x, pos_tag_annotator, y1)
    y2w <- subset(y2, type == "word")
    tags <- sapply(y2w$features, '[[', "POS")
    r1 <- sprintf("%s/%s", unlist(strsplit(x, " ")), tags)
    r2 <- paste(r1, collapse = " ")
    return(r2)
}

# List of openNLP tags: http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
tagOpenNlpToWordnet <- function(tag) {
    t <- strtrim(tag, 1)
    res <- tag
    if (t == "N") res <- "NOUN"
    if (t == "J") res <- "ADJECTIVE"
    if (t == "R") res <- "ADVERB"
    if (t == "V") res <- "VERB"
    res
}

findSynonyms <- function(x){
    paste(sapply(unlist(strsplit(x, " ")), function(token){
```

```
        tVec <- unlist(strsplit(token, "/"))
        word <- tVec[1]
        pos <- tagOpenNlpToWordnet(tVec[2])
        synonym <- tryCatch(
            {
                syns <- c(synonyms(wordStem(word), pos))
                if (length(syns) == 0) synonym <- word
                else tolower(syns[1])
            },
            error = function(e){word}
        )
        remove(tVec, word, pos);
    }), collapse = " ")
}

trim <- function (x) gsub("^\\s+|\\s+$", "", x)

getOrderedUniqueWordsInSentence <- function(sentence){
    paste(sort(unique(unlist(strsplit(sentence, " ")))), collapse = " ")
}

normalizeSentences <- function(sentences){
    txt <- sentences
    txt <- tolower(txt)
    txt <- removeNumbers(txt)
    txt <- removeWords(txt, stopwords("english"))
    txt <- removePunctuation(txt)
    txt <- stemDocument(txt)
    txt <- trim(stripWhitespace(txt))
    #txt <- sapply(txt, tagPOS, USE.NAMES = FALSE)
    #txt <- sapply(txt, findSynonyms, USE.NAMES = FALSE)
    txt <- sapply(txt, getOrderedUniqueWordsInSentence, USE.NAMES = FALSE)
    txt
}
```

```
reviewNormalized <- reviewSentences
reviewNormalized$text <- normalizeSentences(reviewSentences$text)
saveRDS(reviewNormalized, file = "data/normalized.rds")
```

Prediction modelling

```
countWordsStars <- function(data, groupInterval){
    words <- data.frame();
    count <- 0;
    apply(data, 1, function(row){
        count <<- count + 1;
        ws <- unlist(strsplit(row["text"], " "));
        words <<- rbind(
            words,
            data.frame(
                word = ws,
```

```
                count = rep(1, length(ws)),
                stars = rep(as.numeric(row["stars"]), length(ws))
            )
        )
        remove(ws)
        if (count %% groupInterval == 0){
            d0 <- dim(words);
            words <<- ddply(words, ~word, summarize, count=sum(count), stars=sum(stars))
            d1 <- dim(words)
            print(paste(count, "rows", "reduced from", d0[1], "to", d1[1]));
        }
    })
    rownames(words) <- NULL
    words
}
```

```
words <- countWordsStars(training, 1000)
words <- ddply(words, ~word, summarize, count=sum(count), stars=sum(stars))
words <- mutate(words, avg.stars = stars / count)
saveRDS(words, "data/training_words.rds")
```

```
words <- readRDS("data/training_words.rds")
words$word <- as.character(words$word)
toRemove <- words$avg.stars == 5.0 & nchar(words$word) > 20 & words$count == 1;
words <- words[!toRemove,]
saveRDS(words, "data/training_words.rds")
```

```
predictSentence <- function(sentence){
    trainWordsData <- words[words$word %in% unlist(strsplit(sentence, " ")),]
    totalCount <- sum(trainWordsData$count);
    sum((data$count / totalCount) * data$avg.stars)
}

predictDataset <- function(dataset){
    sapply(dataset$text, function(sentence){
        predictSentence(as.character(sentence))
    })
}
```