Yelp Dataset Review

John Antony

March 4, 2017

Reading review data from JSON file

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
reviews_file_path = "/Users/JohnAntony/Desktop/Main/Applications/R/MachineLearning/Yelp/yelp_dataset_challenge_ro
und9/yelp_academic_dataset_review.json"

review_lines <- read_lines(reviews_file_path, n_max = 100000, progress = FALSE)

library(stringr)
library(jsonlite)

reviews_combined <- str_c("[", str_c(review_lines, collapse = ", "), "]")

reviews <- fromJSON(reviews_combined) %>%
  flatten() %>%
  tbl_df()
#head(reviews)
remove(review_lines,reviews_combined)
```

Reading Business data from JSON File

```
business_file_path = "/Users/JohnAntony/Desktop/Main/Applications/R/MachineLearning/Yelp/yelp_dataset_challenge_r
ound9/yelp_academic_dataset_business.json"

business_lines <- read_lines(business_file_path, n_max = 100000, progress = FALSE)

business_combined <- str_c("[", str_c(business_lines, collapse = ", "), "]")

business <- fromJSON(business_combined) %>%
  flatten() %>%
  tbl_df()

#head(business)
business[order(business$business_id),]
```

```
## # A tibble: 100,000 × 16
##               business_id                        name
##                     <chr>                       <chr>
## 1  __1uG7MLxWGFIv2fCGPiQQ   SpinalWorks Chiropractic
## 2  __8j8yhsmE98wNWHJNyAgw                Urawa Sushi
## 3  __blIPRrsfEoaioSPj1olQ        Property Frameworks
## 4  __bqGGnOjtY9eEhrZAUsgA       Galangal Thai Fusion
## 5  __CQ2SE4NXFFjYfrB_TJ6w St. Gabriel Medical Clinic
## 6  __D6AVR_hLpW_bott0-upA           Skinapeel Beauty
## 7  __FFoyg0XmJluBBNE0QP0w    Better Health Solutions
## 8  __fMLrmv9M1_W4kBvR2VnQ                Dairy Queen
## 9  __G0Ug3CK2yCDdQLYpd0ww                     LV spa
## 10 __H_61gpm7eViPMbWxPZSg                     Subway
## # ... with 99,990 more rows, and 14 more variables: neighborhood <chr>,
## #   address <chr>, city <chr>, state <chr>, postal_code <chr>,
## #   latitude <dbl>, longitude <dbl>, stars <dbl>, review_count <int>,
## #   is_open <int>, attributes <list>, categories <list>, hours <list>,
## #   type <chr>
```

```
remove(business_combined,business_lines)
```

Cleanup

```
rest_reviews = aggregate(text ~ business_id, data = reviews, paste, collapse = ",")
rest_reviews$text = tolower(rest_reviews$text)
#head(rest_reviews)
```

Defining functions for Sentiment Scoring and Pulling positive and negative words

```r
score.sentiment = function(sentences, pos.words, neg.words, .progress='none')
{
    require(plyr)
    require(stringr)

    # we got a vector of sentences. plyr will handle a list or a vector as an "l" for us
    # we want a simple array of scores back, so we use "l" + "a" + "ply" = laply:
    scores = laply(sentences, function(sentence, pos.words, neg.words) {

        # clean up sentences with R's regex-driven global substitute, gsub():
        sentence = gsub('[[:punct:]]', '', sentence)
        sentence = gsub('[[:cntrl:]]', '', sentence)
        sentence = gsub('\\d+', '', sentence)
        # and convert to lower case:
        sentence = tolower(sentence)

        # split into words. str_split is in the stringr package
        word.list = str_split(sentence, '\\s+')
        # sometimes a list() is one level of hierarchy too much
        words = unlist(word.list)

        # compare our words to the dictionaries of positive & negative terms
        pos.matches = match(words, pos.words)
        neg.matches = match(words, neg.words)

        # match() returns the position of the matched term or NA
        # we just want a TRUE/FALSE:
        pos.matches = !is.na(pos.matches)
        neg.matches = !is.na(neg.matches)

        # and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():
        score = sum(pos.matches) - sum(neg.matches)

        return(score)
    }, pos.words, neg.words, .progress=.progress )

    scores.df = data.frame(score=scores, text=sentences)
    return(scores.df)
}
```

```
HIDict = readLines("/Users/JohnAntony/Desktop/Main/Applications/R/MachineLearning/data_files/inqdict.txt")
dict_pos = HIDict[grep("Pos",HIDict)]
poswords = NULL
for (s in dict_pos) {
    s = strsplit(s,"#")[[1]][1]
    poswords = c(poswords,strsplit(s," ")[[1]][1])
}
dict_neg = HIDict[grep("Neg",HIDict)]
negwords = NULL
for (s in dict_neg) {
    s = strsplit(s,"#")[[1]][1]
    negwords = c(negwords,strsplit(s," ")[[1]][1])
}
poswords = tolower(poswords)
negwords = tolower(negwords)
pos.words = unique(poswords)
neg.words = unique(negwords)
```

Sentiment Score for Review Texts

```
score = score.sentiment(rest_reviews$text, pos.words, neg.words)
```

```
## Loading required package: plyr
```

```
## -------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## -------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
rest_reviews_score = cbind(rest_reviews$business_id, score$score)
colnames(rest_reviews_score) <- c("business_id", "SentimentScore")
rest_reviews_score = as.data.frame(rest_reviews_score)

#rest_reviews_score
```

Creating Dataframe for regression

```
suppressMessages(library(dplyr))
#business_score_rating = merge(x = rest_reviews_score[ ,c("SentimentScore")], y = business[ ,c("stars")],  by.res
t_reviews_score='business_id', by.business='business_id', all.x = TRUE)

business_score_rating = dplyr::left_join(rest_reviews_score, business, by = "business_id")
```

```
## Warning in left_join_impl(x, y, by$x, by$y, suffix$x, suffix$y): joining
## character vector and factor, coercing into character vector
```

```
business_score_rating = dplyr::select(business_score_rating, SentimentScore, stars)
business_score_rating$SentimentScore <- as.numeric(as.character(business_score_rating$SentimentScore))
#head(business_score_rating)
```

Regression of Sentiment Score with Business Score

```
res = lm(business_score_rating$SentimentScore ~ business_score_rating$stars)
print(summary(res))
```

```
##
## Call:
## lm(formula = business_score_rating$SentimentScore ~ business_score_rating$stars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -109.29  -21.40  -10.18    8.82  605.77
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  11.6247     1.4733    7.89 3.32e-15 ***
## business_score_rating$stars   5.8879     0.3938   14.95  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 40.28 on 10275 degrees of freedom
##   (4399 observations deleted due to missingness)
## Multiple R-squared:  0.02129,    Adjusted R-squared:  0.02119
## F-statistic: 223.5 on 1 and 10275 DF,  p-value: < 2.2e-16
```

Creating a Network Cloud on Positive and Negative Sentiments

```
reviews_rest <- subset(reviews, business_id=="GdCIMZ9BTT4ywETWcByfJA")
#tail(names(sort(table(reviews_rest$business_id))), 1)
#head(reviews_rest)
```

Getting the bigrams

```
library(dplyr)
library(tidytext)
library(tidyr)

reviews_text = subset(reviews_rest, select = c(text) )

reviews_bigrams = reviews_text %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)
#reviews_bigrams
```

Using dplyr Count for counting the occurance of bigrams

```
#reviews_bigrams %>%
#  dplyr::count(bigram, sort = TRUE)
```

Bigram Cleaning for Text analysis

```
bigrams_separated <- reviews_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  filter(word1 %in% neg.words)


bigram_counts <- bigrams_filtered %>%
  dplyr::count(word1, word2, sort = TRUE)
remove(reviews_bigrams)
```

Using igraph to discover the network graph

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:tidyr':
##
##     %>%, crossing
```

```
## The following object is masked from 'package:stringr':
##
##     %>%
```

```
## The following objects are masked from 'package:dplyr':
##
##     %>%, as_data_frame, groups, union
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
bigram_graph <- bigram_counts %>%
  graph_from_data_frame()

bigram_graph
```

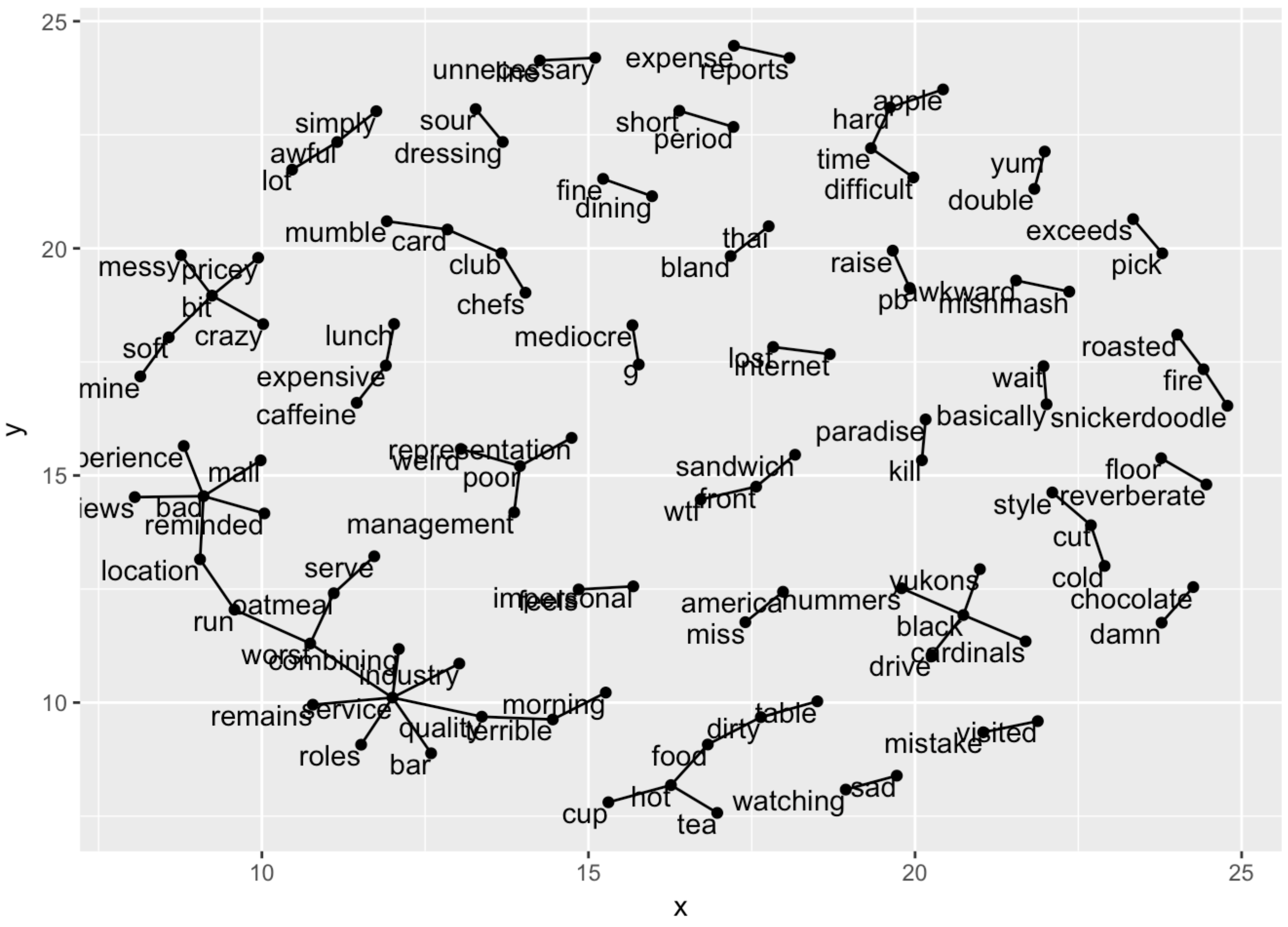
```
## IGRAPH DN-- 103 71 --
## + attr: name (v/c), n (e/n)
## + edges (vertex names):
##  [1] club     ->card       fire     ->roasted    service  ->industry
##  [4] awful    ->lot        awful    ->simply     awkward  ->mishmash
##  [7] bad      ->experience bad      ->location   bad      ->mall
## [10] bad      ->reminded   bad      ->reviews    bit      ->crazy
## [13] bit      ->messy      bit      ->pricey     bit      ->soft
## [16] black    ->cardinals  black    ->hummers    black    ->yukons
## [19] bland    ->thai       club     ->chefs      cold     ->cut
## [22] cut      ->style      damn     ->chocolate  difficult->time
## + ... omitted several edges
```

```
library(ggraph)
```

```
## Loading required package: ggplot2
```

```
set.seed(2017)

ggraph(bigram_graph, layout = "fr") +
  geom_edge_link() +
  geom_node_point() +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```


unnecessary
line
expense
reports
simply
awful
lot
sour
dressing
short
period
hard
apple
time
difficult
yum
double
fine
dining
mumble
card
club
chefs
thai
bland
exceeds
pick
messy
pricey
bit
crazy
soft
mine
lunch
expensive
caffeine
mediocre
g
raise
pb
awkward
mishmash
lost
internet
roasted
fire
wait
snickerdoodle
basically
paradise
kill
perience
mall
iews
bad
reminded
location
representation
weird
poor
management
sandwich
front
wtf
floor
reverberate
style
cut
cold
serve
oatmeal
run
worst
combining
industry
remains
service
quality
roles
bar
morning
terrible
impersonal
feels
america
hummers
miss
yukons
black
cardinals
drive
chocolate
damn
dirty
table
food
hot
cup
tea
watching
sad
mistake
visited
x
y
10
15
20
25