

NLP Assignment: Kindle Store Reviews Analysis

Introduction





The Kindle Store is an online e-book e-commerce store operated by Amazon as part of its retail website and can be accessed from any Amazon Kindle, Fire tablet or Kindle mobile app. At the launch of the Kindle in November 2007, the store had more than 88,000 digital titles available in the U.S. store. This number increased to more than 275,000 by late 2008, and exceeded 765,000 by August 2011. In July 2014, there were over 2.7 million titles available. As of March 2018 there are over six million titles available in the U.S.




One of the store's novelties was one-click buying, which allowed consumers to rapidly purchase an e-book. The Kindle Store employs a recommendation system that examines the user's purchase history, browsing history, and reading behavior before recommending stuff it believes the user would enjoy. As a result, it is critical to analyze user reviews in order to promote books and enhance sales.

Data Description

This 5-core dataset of product reviews from Amazon Kindle Store category from May 1996 - July 2014. Contains total of 982619 entries. Each reviewer has at least 5 reviews and each product has at least 5 reviews in this dataset. Each observation contains 10 attributes that include information such as reviewer ID, product ID etc. The table below provides brief descriptions of each attribute and its type.

Kindle Review

 Variable	 Type	 Description
	Long	Index
<u>reviewerID</u>	String	ID of the reviewer, e.g. A2SUAM1J3GNN3B
<u>reviewerName</u>	String	name of the reviewer
<u>asin</u>	String	ID of the product, e.g. 0000013714

 Variable	 Type	 Description
<u>helpful</u>	Array	helpfulness rating of the review, e.g. 2/3
<u>reviewText</u>	String	text of the review
<u>overall</u>	Integer	rating of the product
<u>summary</u>	String	summary of the review
<u>unixReviewTime</u>	Timestamp	time of the review (unix time)
<u>reviewTime</u>	Date	time of the review (raw)

This dataset is taken from Amazon product data, Julian McAuley, UCSD website.
<http://jmcauley.ucsd.edu/data/amazon/>

Experiments

0. Load Packages

```
library("tm") # for text mining
library(tidyverse) # for data processing
library(plyr)
library("SnowballC") # for text stemming, reduces words to their root form
library("syuzhet") # for sentiment analysis
library(udpipe) # tokenization, Parts of Speech Tagging, Lemmatization and Dependency
library(lattice) # for bar plot etc.
```

1. Data Pre-processing

a. Get samples from raw data

Since there are nearly one million data, limited by local computing power and resources, it was selected a sample of one percent of the total data base on the **overall** rating.

```
setwd("/Users/vinci/git/Intelligent-Systems/Unit5-Assignment")
getwd()
```

```
rawData <- read.csv('data/kindle_reviews.csv') %>%
  group_by(overall) %>%
  sample_frac(.01)
```

```
rawData<-data.frame(rawData)
rawData$index <- row.names(rawData)
```

b. Check the most frequent, identical review texts

```
rawData %>%
  group_by(reviewText) %>%
  dplyr::summarize(n_reviews = n()) %>%
  mutate(pct = n_reviews / sum(n_reviews)) %>%
  arrange(-n_reviews) %>%
  top_n(10, n_reviews)
```

```
## # A tibble: 9,826 x 3
##   reviewText                                n_reviews    pct
##   <chr>                                <int>    <dbl>
## 1 ""                                    1 1.02e-4
## 2 "-SPOILER FREE REVIEWsteamy, dangerous, passionate,one-in-- 1 1.02e-4
## 3 ",The Basic Recipes Every Kitchen Should Have title caught~ 1 1.02e-4
## 4 "! Unusual, and different, easy and quick, anyone can mak~ 1 1.02e-4
## 5 "...because \"Them\", though definitely 50s, was much bet~ 1 1.02e-4
## 6 "...Mz. Geary gave me an escape from reality, a tender, l~ 1 1.02e-4
## 7 "...about an uptight doctor and a hot boy but it turns out~ 1 1.02e-4
## 8 "...Ever since reading The Fire, I never know how John wil~ 1 1.02e-4
## 9 "...from laughter. Brownie is Bubba's nephew, and a more ~ 1 1.02e-4
## 10 "...geez, as much as I really, really, hated Nathan..I don'~ 1 1.02e-4
## # ... with 9,816 more rows
```

From the output, there are no duplicate reviews, or empty reviews, which means that the raw data has been processed. So those reviews just need to be normalized in the next step.

2. Text Normalization

We normalize text to lessen its unpredictability and bring it closer to a preset "standard." This reduces the quantity of diverse data that the computer has to cope with, resulting in increased efficiency. Normalization procedures such as stemming and lemmatization aim to reduce a word's inflectional and occasionally derivationally related forms to a single base form. We process the text according to the following steps:

a. Load the data as a corpus:

```
ReviewCorpus <- Corpus(VectorSource(rawData$reviewText))
```

b. Replacing `/`, `@` and `|` with space

```
toSpace <- content_transformer(function (x , pattern) gsub(pattern, " ", x))
ReviewCorpus <- tm_map(ReviewCorpus, toSpace, "/" )
ReviewCorpus <- tm_map(ReviewCorpus, toSpace, "@")
ReviewCorpus <- tm_map(ReviewCorpus, toSpace, "\\|\\|")
```

c. Convert the text to lower case

```
ReviewCorpus <- tm_map(ReviewCorpus, content_transformer(tolower))
```

d. Remove punctuation

```
ReviewCorpus <- tm_map(ReviewCorpus, removePunctuation)
```

e. Remove numbers

```
ReviewCorpus <- tm_map(ReviewCorpus, removeNumbers)
```

f. Remove extra white spaces

```
ReviewCorpus <- tm_map(ReviewCorpus, stripWhitespace)
```

g. Remove English common stop words

```
ReviewCorpus <- tm_map(ReviewCorpus, removeWords, stopwords("english"))
```

h. Text stemming - reduces words to their root form

```
ReviewCorpus <- tm_map(ReviewCorpus, stemDocument)
```

i. Take a look at the normalized result

Before printing the output, the corpus needs to be parsed to generate a `Dataframe`.
The result shows below:

```
normalWords <- ldply (ReviewCorpus, data.frame)
names(normalWords)[1] <- 'cleanReviewText'
normalWords$index <- row.names(normalWords)
head(normalWords$cleanReviewText)
```

j. Merge cleaned reviews in raw data

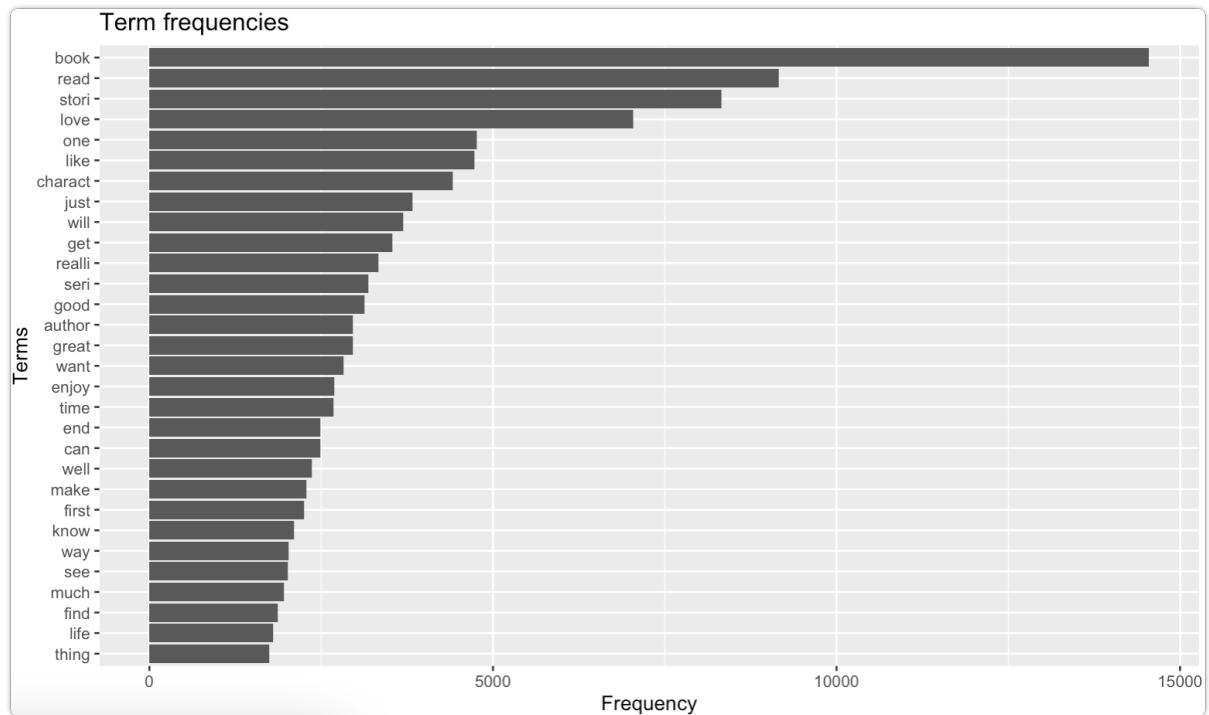
```
preprocessedData <- merge(rawData, normalWords, by = "index", all = TRUE)
```

Reviews Analysis

1. The top 30 words

We build a Term Document Matrix(TDM) to represents documents vectors in matrix form in which the rows correspond to the terms in the document, columns correspond to the documents in the corpus and cells correspond to the weights of the terms. So we can easily count out the top 30 most frequently used words in reviews.

```
tdm <- TermDocumentMatrix(ReviewCorpus)
tdm <- as.matrix(tdm)
freq = rowSums(tdm)
freq.high = tail(sort(freq), n = 30)
freq.df = as.data.frame(freq.high)
freq.df$names <- rownames(freq.df)
ggplot(freq.df, aes(reorder(names, freq.high), freq.high)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  xlab("Terms") +
  ylab("Frequency") +
  ggtitle("Term frequencies")
```



2. Sentiment analysis

The Syuzhet Package comes with four sentiment dictionaries and provides a method for accessing the robust, but computationally expensive, sentiment extraction tool developed in the NLP group at Stanford.

We can run nrc sentiment analysis to return data frame with each row classified as one of the following emotions, rather than a score: anger, anticipation, disgust, fear, joy, sadness, surprise, trust

It also counts the number of positive and negative emotions found in each row.

- To see top 10 line sentiments

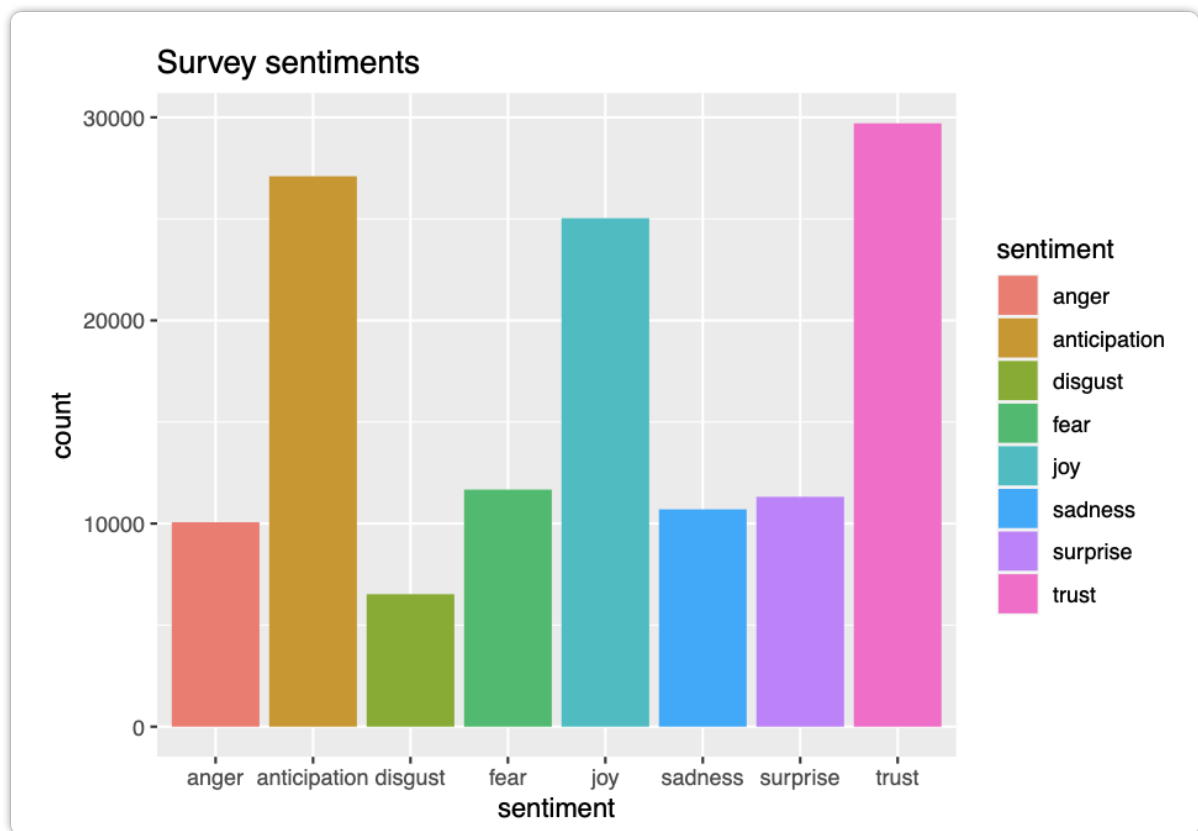
```
d <- get_nrc_sentiment(preprocessedData$cleanReviewText)
head(d, 10)
```

##	anger	anticipation	disgust	fear	joy	sadness	surprise	trust	negative	positive	
## 1	0		2	1	1	2	1	0	3	2	5
## 2	0		1	0	0	1	0	0	0	1	1
## 3	0		2	0	0	0	0	0	0	0	2
## 4	1		1	1	1	0	2	0	0	3	1
## 5	0		0	0	0	1	0	0	2	0	3
## 6	0		1	0	0	1	0	0	1	0	2
## 7	0		3	0	0	1	0	0	1	2	2
## 8	0		1	0	0	1	0	1	1	0	1
## 9	0		4	0	0	4	0	1	5	3	7
## 10	0		1	0	0	1	0	0	1	0	3

- Count of words associated with each sentiment

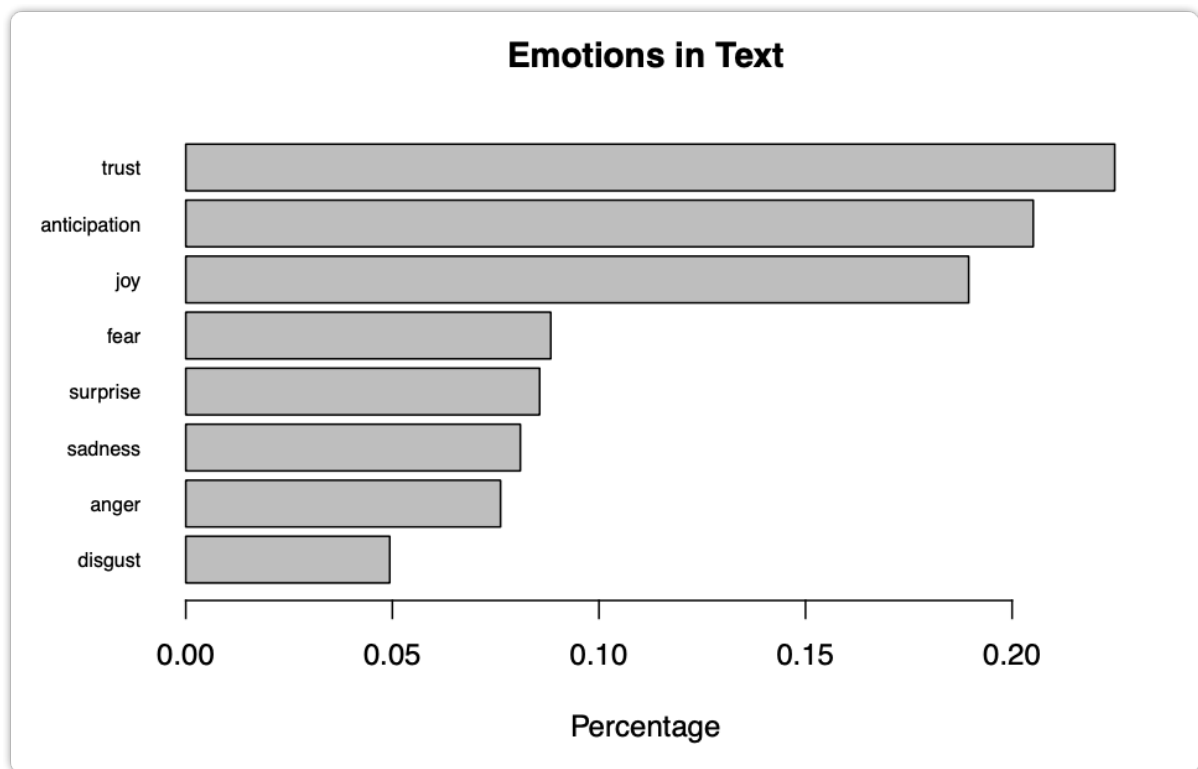
```
#transpose
td <- data.frame(t(d))
td_new <- data.frame(rowSums(td[2:9826]))
#Transformation and cleaning
names(td_new)[1] <- "count"
td_new <- cbind("sentiment" = rownames(td_new), td_new)
rownames(td_new) <- NULL
td_new2 <- td_new[1:8,]

quickplot(
  sentiment,
  data = td_new2,
  weight = count,
  geom = "bar",
  fill = sentiment,
  ylab = "count"
) + ggtitle("Survey sentiments")
```



- Count of words associated with each sentiment, expressed as a percentage

```
barplot(  
  sort(colSums(prop.table(d[, 1:8]))),  
  horiz = TRUE,  
  cex.names = 0.7,  
  las = 1,  
  main = "Emotions in Text",  
  xlab = "Percentage"  
)
```

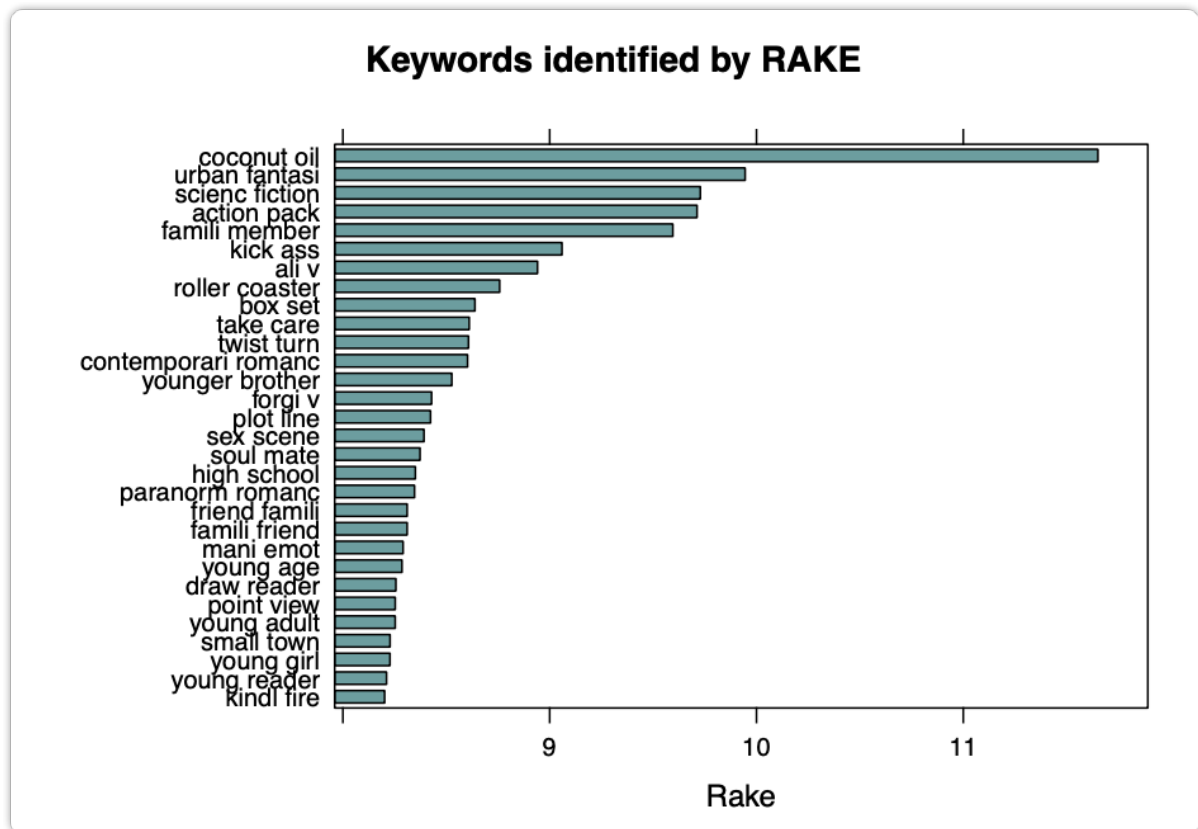



3. Top 20 keywords

In addition, we can analyze the keywords of the reviews and then make relevant book recommendations. Here we use the Rapid Automatic Keyword Extraction Algorithm(RAKE), which is contained in the `Udpipe` package. This natural language processing toolkit provides language-agnostic 'tokenization', 'parts of speech tagging', 'lemmatization' and 'dependency parsing' of raw text.

```
ud_model <- udpipe_download_model(language = "english")
ud_model <- udpipe_load_model(ud_model$file_model)
x <- udpipe_annotate(ud_model,
                     x = preprocessedData$cleanReviewText,
                     doc_id = preprocessedData$index)
x <- as.data.frame(x)
stats <- keywords_rake(
  x = x,
  term = "lemma",
  group = "doc_id",
  relevant = x$upos %in% c("NOUN", "ADJ")
)
stats$key <- factor(stats$keyword, levels = rev(stats$keyword))
barchart(
  key ~ rake,
  data = head(subset(stats, freq > 3), 20),
  col = "cadetblue",
  main = "Keywords identified by RAKE",
```

```
xlab = "Rake"
)
```



Conclusion

By analyzing the reviews as described above, we were able to have a good understanding of basic data processing, as well as text normalization (including the removal of numbers, punctuation, stop words, etc.), and then calculate a matrix to count word frequencies, in addition to analyzing the sentiment of reviews, as well as keywords, by using existing NLP analysis tools (`tm` , `SnowballC` , `syuzhet` and `udpipe` etc.). Among the such reviews, "trust", "anticipation", "joy" are the most common sentiments, and " family members", "science fiction", and " small town" are the most common keywords, which can be easily seen in the plot.

Project Repository is on [Github Unit5-Assignment](#)

Reference

1. [Text Mining and Sentiment Analysis in with R](#)

2. Natural Language Processing for predictive purposes with R
 3. UDPipe Natural Language Processing - Basic Analytical Use Cases
-