Is the Customer Always Right?

An exploration of customer feedback through Natural Language Processing.

By @josiahjdavis

Within the U.S., the slogan is often attributed to Marshall Fields.

"Assume that the customer is right until it is plain beyond all question he is not."

Marshall Field's pioneered many retail practices now considered standard:

- One-price price tag

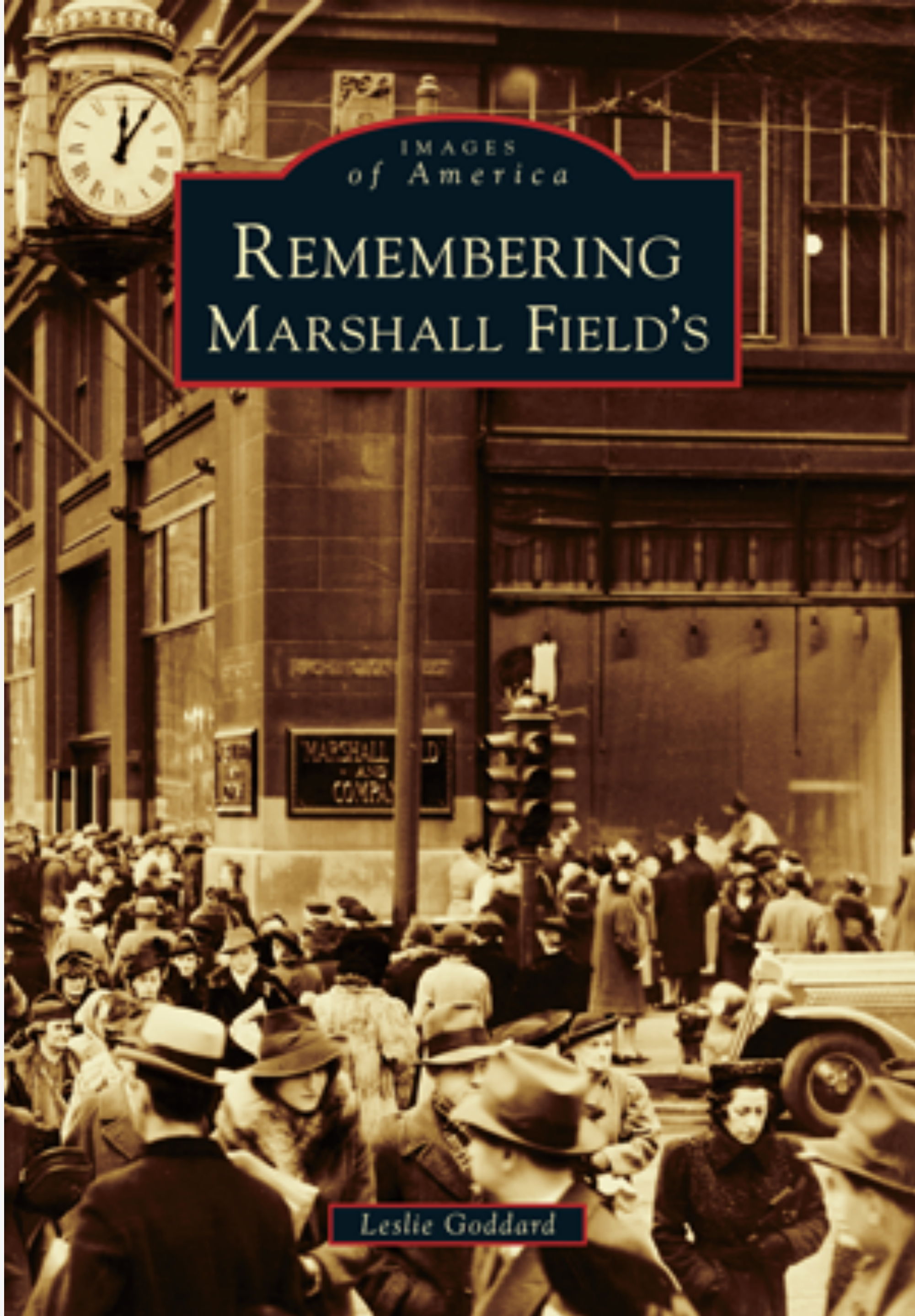
- Any product can be returned

- No aggressive salespeople

- Bridal registry

- Personal Shopper

- Escalators



The customer *is* always right, but all slogans deserve qualification.

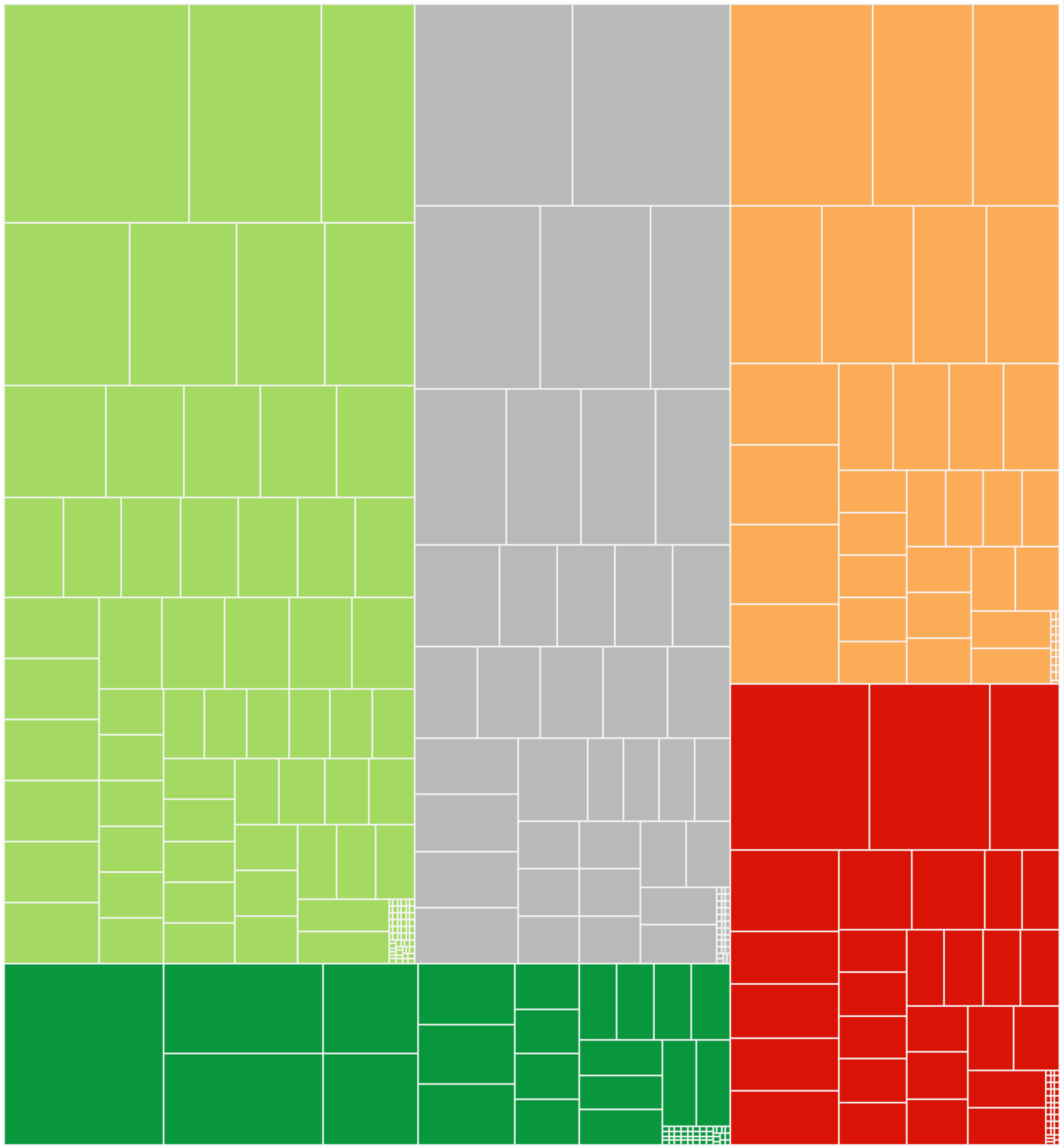
Some customer feedback is more **useful** than others.

Challenge

The volume and unstructured nature of textual information makes text mining a formidable task.

Insight

Structured data can be used to frame the analysis of unstructured data.



Usefulness

Each node in the treemap corresponds to a review. The size is proportional to the number of usefulness votes received.

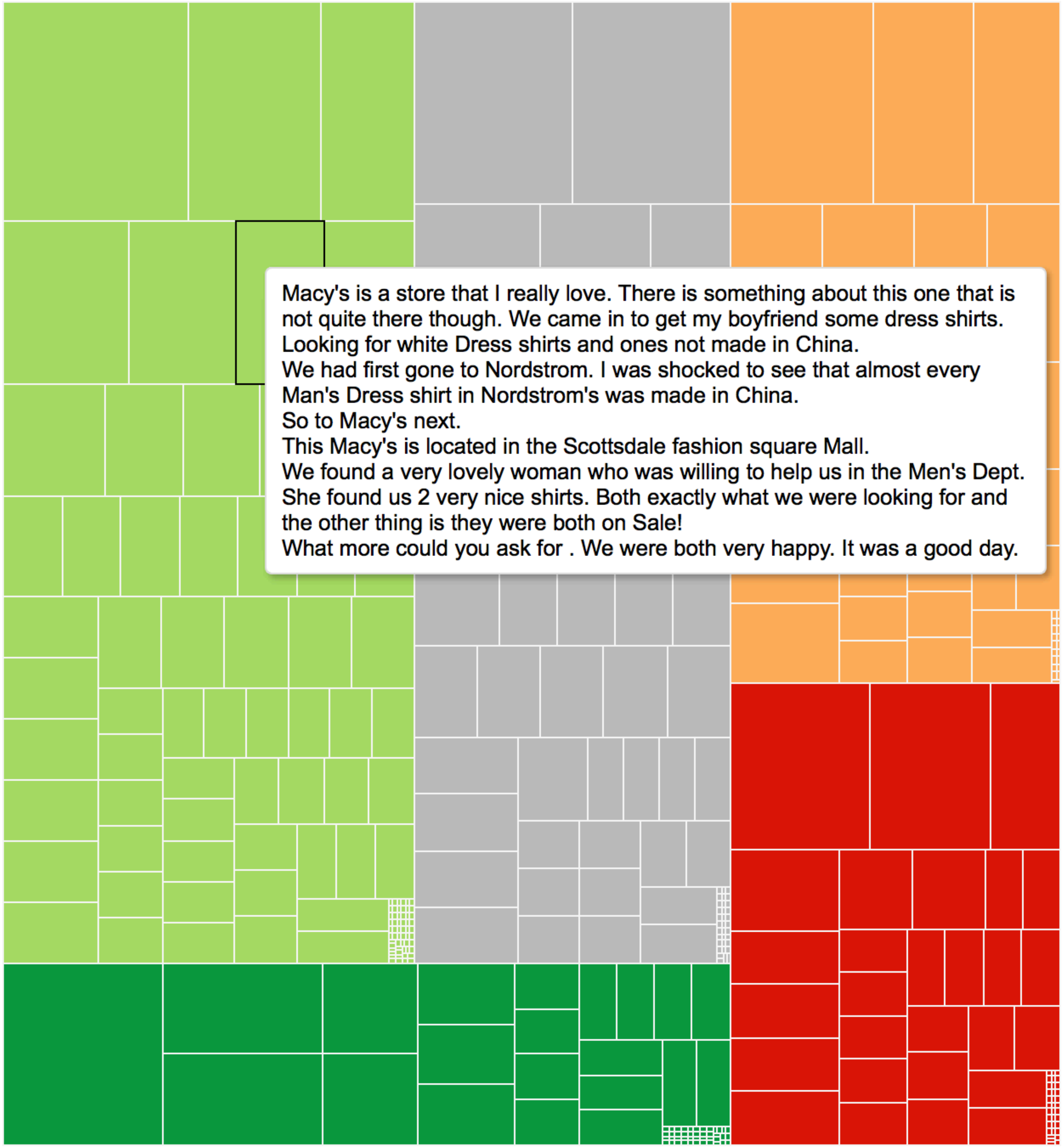
🟩 5 Stars

🟩 4 Stars

⬜ 3 Stars

🟧 2 Stars

🟥 1 Star



Macy's is a store that I really love. There is something about this one that is not quite there though. We came in to get my boyfriend some dress shirts.
Looking for white Dress shirts and ones not made in China.
We had first gone to Nordstrom. I was shocked to see that almost every Man's Dress shirt in Nordstrom's was made in China.
So to Macy's next.
This Macy's is located in the Scottsdale fashion square Mall.
We found a very lovely woman who was willing to help us in the Men's Dept. She found us 2 very nice shirts. Both exactly what we were looking for and the other thing is they were both on Sale!
What more could you ask for . We were both very happy. It was a good day.

Usefulness

Each node in the treemap corresponds to a review. The size is proportional to the number of usefulness votes received.

🟩 5 Stars
🟩 4 Stars
⬜ 3 Stars
🟧 2 Stars
🟥 1 Star

Some customer complaints are more **frequent** than others

Challenge

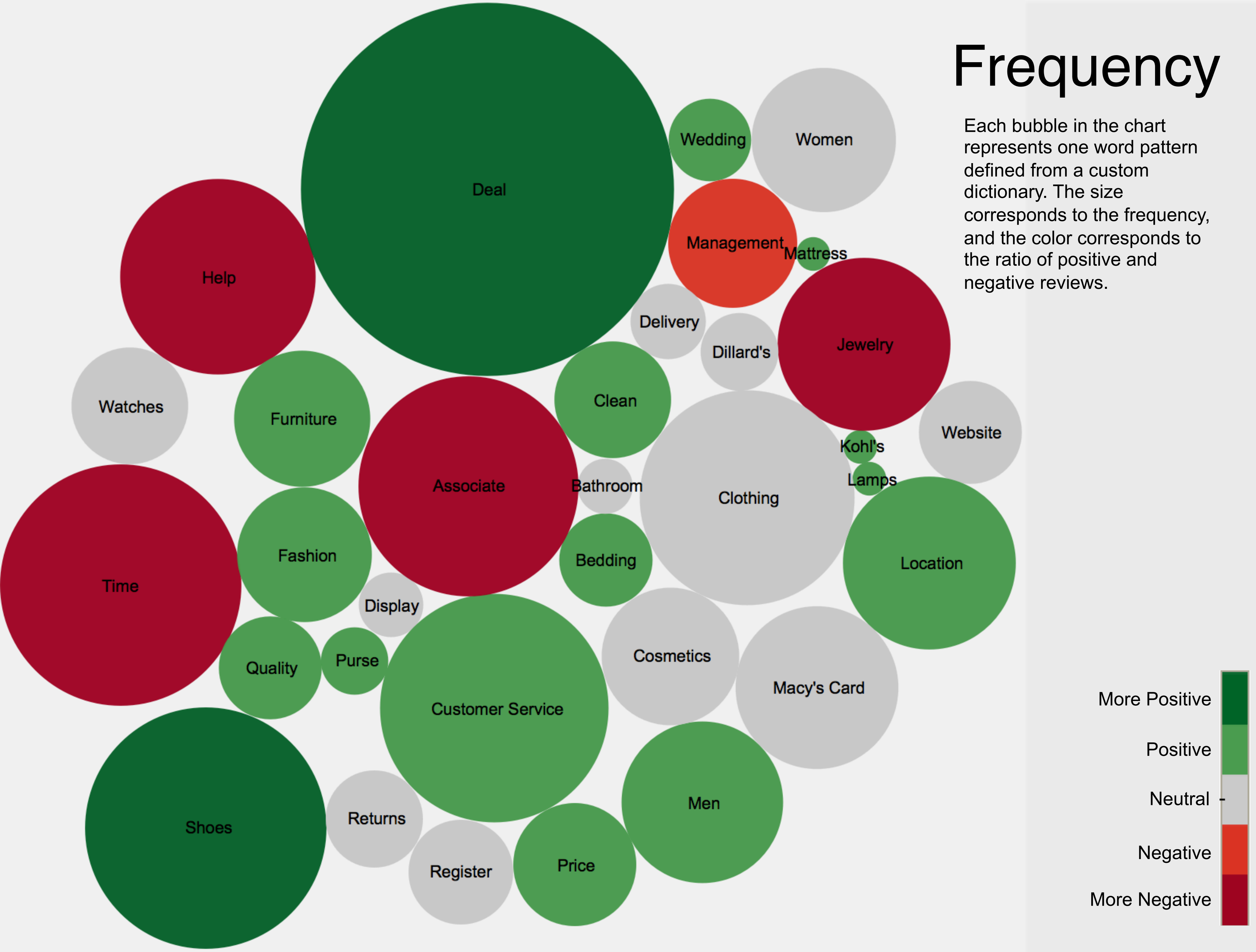
Written text is fraught with misspellings, and full of synonyms (e.g., employee, associate).

Insight

Using a dictionary of word patterns to define terms provides greater flexibility than mere tokenization.

Frequency

Each bubble in the chart represents one word pattern defined from a custom dictionary. The size corresponds to the frequency, and the color corresponds to the ratio of positive and negative reviews.



Some customers are more **nuanced** than others.

Challenge

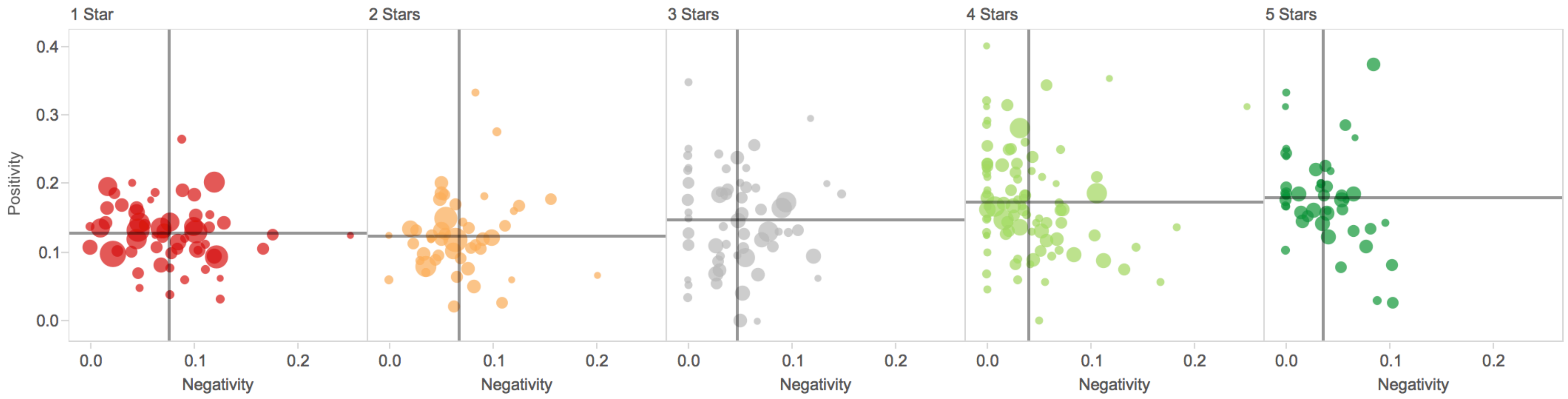
Human beings are easily biased by un-related factors (e.g., Halo/Horns Effect).

Insight

Evaluating sentiment at both the document and token level can be more insightful than either one in isolation.

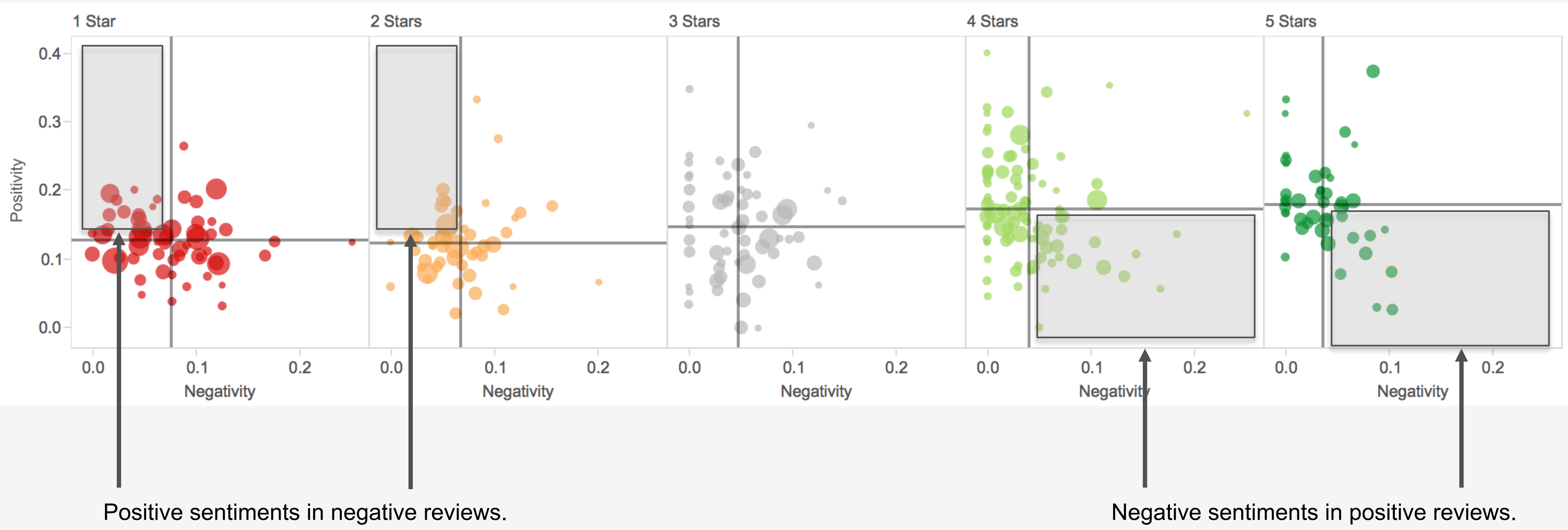
Nuance

Each point in the scatter chart below corresponds to a review. The y- and x-axes are defined as the number of positively and negatively scored words divided by the number of total words. Words are scored using an off-the-shelf sentiment library.



Nuance

Each point in the scatter chart below corresponds to a review. The y- and x-axes are defined as the number of positively and negatively scored words divided by the number of total words. Words are scored using an off-the-shelf sentiment library.



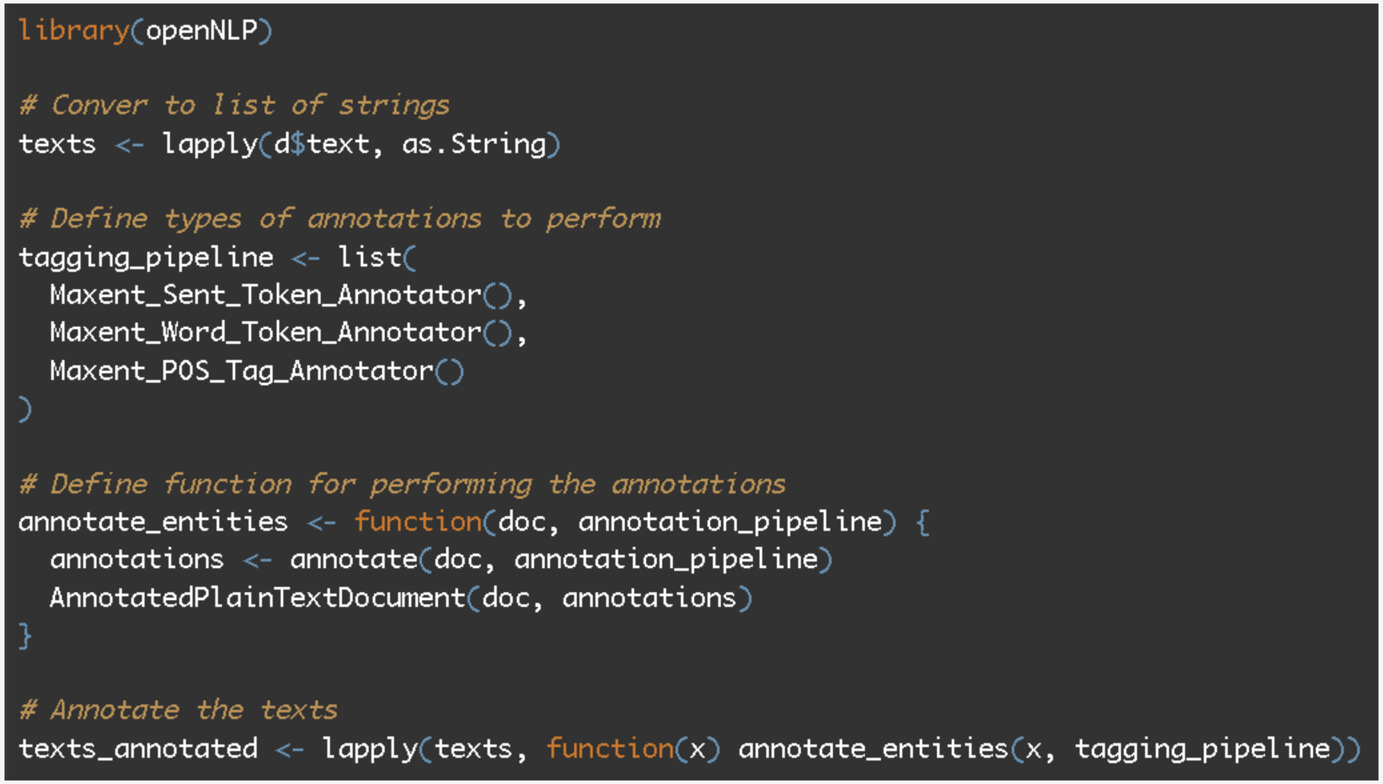
Is the customer always right?

Yes, but some customers are more right than others.

Appendix: Code

Thank you.

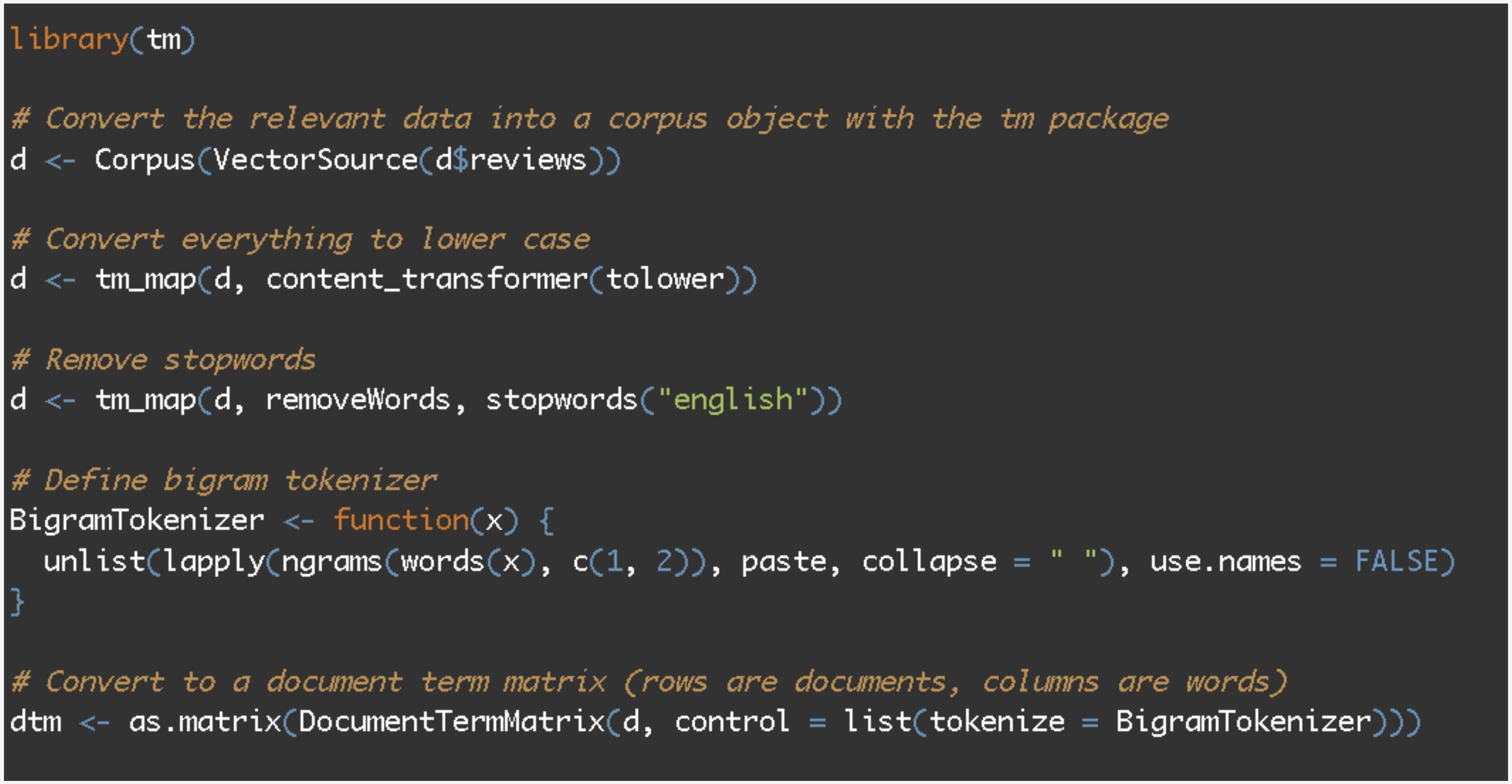
Parts of Speech



```r
library(openNLP)

# Conver to list of strings
texts <- lapply(d$text, as.String)

# Define types of annotations to perform
tagging_pipeline <- list(
  Maxent_Sent_Token_Annotator(),
  Maxent_Word_Token_Annotator(),
  Maxent_POS_Tag_Annotator()
)

# Define function for performing the annotations
annotate_entities <- function(doc, annotation_pipeline) {
  annotations <- annotate(doc, annotation_pipeline)
  AnnotatedPlainTextDocument(doc, annotations)
}

# Annotate the texts
texts_annotated <- lapply(texts, function(x) annotate_entities(x, tagging_pipeline))
```

https://github.com/josiahdavis/earl/blob/master/posTagger.R

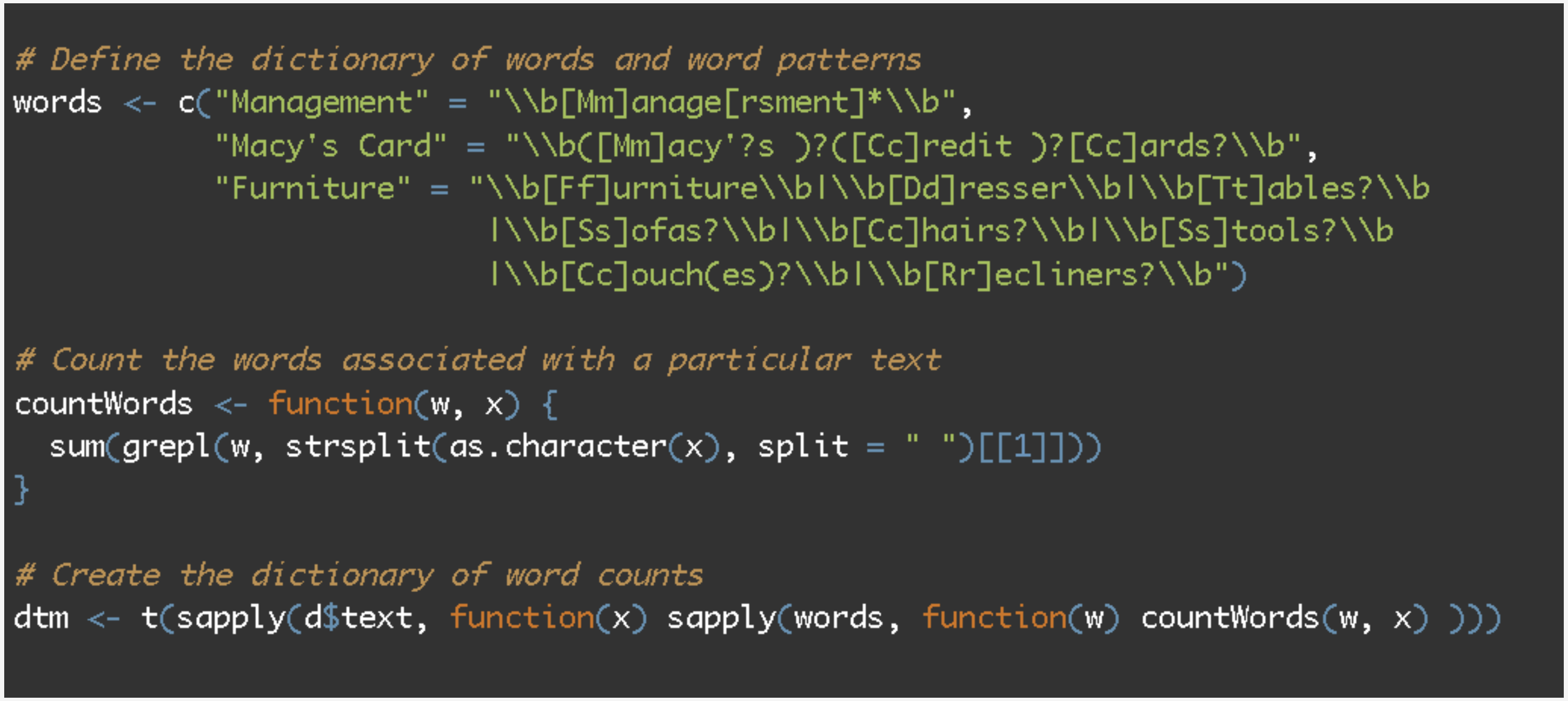
Tokenization



```r
library(tm)

# Convert the relevant data into a corpus object with the tm package
d <- Corpus(VectorSource(d$reviews))

# Convert everything to lower case
d <- tm_map(d, content_transformer(tolower))

# Remove stopwords
d <- tm_map(d, removeWords, stopwords("english"))

# Define bigram tokenizer
BigramTokenizer <- function(x) {
  unlist(lapply(ngrams(words(x), c(1, 2)), paste, collapse = " "), use.names = FALSE)
}

# Convert to a document term matrix (rows are documents, columns are words)
dtm <- as.matrix(DocumentTermMatrix(d, control = list(tokenize = BigramTokenizer)))
```

https://github.com/josiahdavis/earl/blob/master/dtm.R

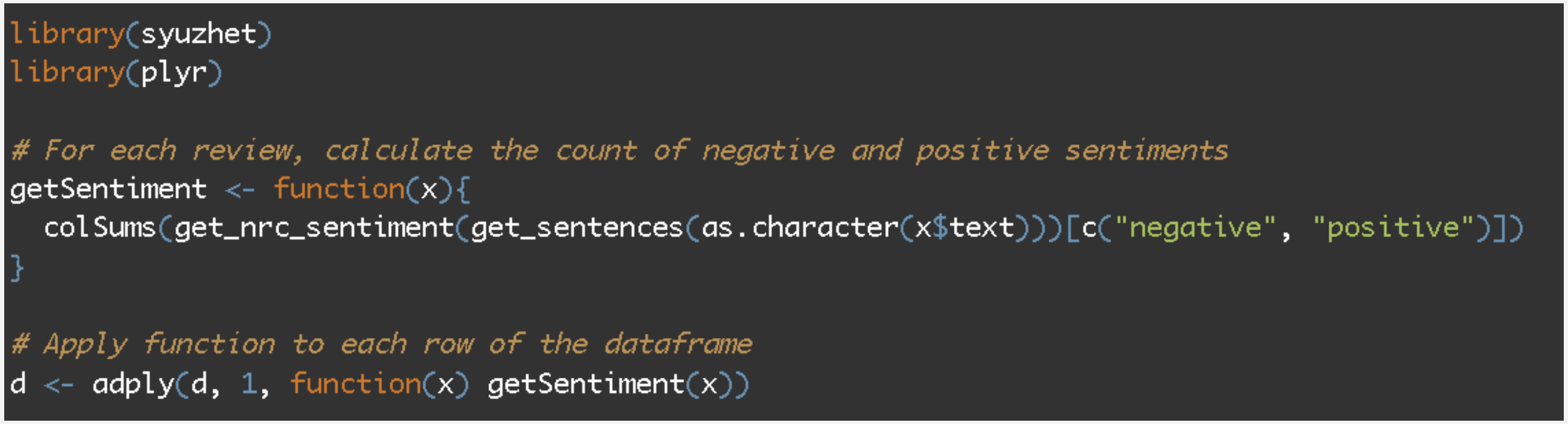
Using a Custom Dictionary



```r
# Define the dictionary of words and word patterns
words <- c("Management" = "\\b[Mm]anage[rsment]*\\b",
           "Macy's Card" = "\\b([Mm]acy'?s )?([Cc]redit )?[Cc]ards?\\b",
           "Furniture" = "\\b[Ff]urniture\\b|\\b[Dd]resser\\b|\\b[Tt]ables?\\b
                         |\\b[Ss]ofas?\\b|\\b[Cc]hairs?\\b|\\b[Ss]tools?\\b
                         |\\b[Cc]ouch(es)?\\b|\\b[Rr]ecliners?\\b")

# Count the words associated with a particular text
countWords <- function(w, x) {
  sum(grepl(w, strsplit(as.character(x), split = " ")[[1]]))
}

# Create the dictionary of word counts
dtm <- t(sapply(d$text, function(x) sapply(words, function(w) countWords(w, x) )))
```

https://github.com/josiahdavis/earl/blob/master/dictionary.R

Sentiment Analysis



```r
library(syuzhet)
library(plyr)

# For each review, calculate the count of negative and positive sentiments
getSentiment <- function(x){
  colSums(get_nrc_sentiment(get_sentences(as.character(x$text))))[c("negative", "positive")])
}

# Apply function to each row of the dataframe
d <- adply(d, 1, function(x) getSentiment(x))
```

https://github.com/josiahdavis/earl/blob/master/reviewSentiment.R