So rather than arguing over which definition is the best, why not use all definitions simultaneously? This article attempts to define a new definition of marketing, using a data science approach. We can use the R language to scrape the 72 definitions from Heidi’s website and attempt text analysis to extract the essence of marketing from this data set.

I have mentioned that automated text analysis is not always a useful method to extract meaning from a text. I decided to delve a little deeper into automated text analysis to see if we find out anything useful about marketing using the rvest and tidytext packages.

The presentation below shows the slides I use in my introductory into marketing. The code and analyses are shown below the slideshow.

 R Codes to Analyse text and Images

Biorhythm.R

|  |
| --- |
| # Biorhythm |
|  |  |
|  | library(ggplot2) |
|  | library(dplyr) |
|  | library(tidyr) |
|  |  |
|  | biorhythm <- function(dob, target = Sys.Date()) { |
|  | dob <- as.Date(dob) |
|  | target <- as.Date(target) |
|  | t <- round(as.numeric(difftime(target, dob))) |
|  | days <- (t - 14) : (t + 14) |
|  | period <- data.frame(Date = seq.Date(from = target - 14, by = 1, length.out = 29), |
|  | Physical = sin (2 \* pi \* days / 23) \* 100, |
|  | Emotional = sin (2 \* pi \* days / 28) \* 100, |
|  | Intellectual = sin (2 \* pi \* days / 33) \* 100) %>% |
|  | pivot\_longer(-Date, names\_to = "Biorhythm", values\_to = "Percentage") |
|  | ggplot(period, aes(x = Date, y = Percentage, col = Biorhythm)) + geom\_line() + |
|  | ggtitle(paste("DoB:", format(dob, "%d %B %Y"))) + |
|  | geom\_vline(xintercept = as.numeric(target)) |
|  | } |
|  |  |
|  | biorhythm("1969-09-12") |
|  | ggsave("biorhythm.png") |
|  |  |
|  | dob <- as.Date("1969-09-12") |
|  | target <- dob + 23 \*28 \* 33 |
|  | biorhythm(dob, target) |
|  |  |
|  | bio.max <- 0 |
|  | day.max <- NA |
|  | for (days in 0:(365\*80)) { |
|  | target <- dob + days |
|  | biorhythm <- (sin (2 \* pi \* days / 23) \* 100) + (sin (2 \* pi \* days / 28) \* 100) + (sin (2 \* pi \* days / 33) \* 100) |
|  | if (biorhythm > bio.max) { |
|  | bio.max <- biorhythm |
|  | day.max <- target |
|  | } |
|  | } |
|  | bio.max |
|  | day.max-dob |

Enron-corpus.R

|  |
| --- |
| if (!file.exists("enron/enron\_mail\_20150507.tgz")) { |
|  | download.file("https://www.cs.cmu.edu/~./enron/enron\_mail\_20150507.tar.gz", |
|  | destfile = "enron/enron-mail-20150507.tgz") |
|  | } |
|  |  |
|  | if (file.exists("enron/enron-mail-20150507.tgz")) { |
|  | untar("enron/enron-mail-20150507.tgz") |
|  | } |
|  |  |
|  | emails <- list.files("maildir/", full.names = TRUE, recursive = TRUE) |
|  | emails <- emails[grep("/inbox", emails)] |
|  | length(emails) |
|  |  |
|  | library(tibble) |
|  | inboxes <- tibble( |
|  | from = apply(as.data.frame(emails), 1, |
|  | function(x){readLines(x, warn = FALSE)[3]}), |
|  | to = emails, |
|  | stringsAsFactors = FALSE |
|  | ) |
|  |  |
|  | ## Keep only enron.com and strip all but username |
|  | library(stringr) # String manipulation |
|  | inboxes <- inboxes[grepl("@enron.com", inboxes$from),] |
|  | inboxes$from <- str\_remove\_all(inboxes$from, "From: |@enron.com") |
|  | inboxes$to <- str\_remove\_all(inboxes$to, "maildir//|/inbox/[0-9]\*.") |
|  |  |
|  | ## Create list of usernames |
|  | users <- data.frame(user = paste0("maildir/", unique(inboxes$to))) |
|  |  |
|  | ## Remove those without sent mails |
|  | sent <- apply(users, 1, function(x) sum(grepl("sent", dir(x)))) |
|  | users <- subset(users, sent != 0) |
|  |  |
|  | ## Replace username with e-mail name |
|  | users$mailname <- NA |
|  | i < -1 |
|  | for (i in 1:nrow(users)){ |
|  | sentmail <- dir(paste0(users$user[i], "/sent\_items/")) |
|  | name <- readLines(paste0(users$user[i], "/sent\_items/", sentmail[1]), warn = F)[3] |
|  | name <- str\_remove\_all(name, "From: |@enron.com") |
|  | users$mailname[i] <- name |
|  | } |
|  |  |
|  | users$user <- str\_remove(users$user, "maildir/") |
|  | inboxes <- merge(inboxes, by.x = "to", users, by.y = "user") |
|  | inboxes <- tibble(from = inboxes$from, |
|  | to = inboxes$mailname) |
|  |  |
|  | ## Only e-mails between users inside the corpus |
|  | inboxes <- inboxes[inboxes$from %in% inboxes$to,] |
|  |  |
|  | ## Remove no.address |
|  | inboxes <- subset(inboxes, from != "no.address" & to != "no.address") |
|  |  |
|  | ## Remove emails to self |
|  | inboxes <- subset(inboxes, inboxes$from != inboxes$to) |
|  | inboxes |

Literature-review.R

|  |
| --- |
| library(tidyverse) |
|  | library(RQDA) |
|  |  |
|  |  |
|  |  |
|  | ## Visualise codes |
|  | getCodingTable() %>% |
|  | group\_by(codename) %>% |
|  | count() %>% |
|  | arrange(n) %>% |
|  | ungroup() %>% |
|  | mutate(codename = factor(codename, levels = codename)) %>% |
|  | ggplot(aes(codename, n)) + |
|  | geom\_col() + |
|  | coord\_flip() + |
|  | xlab("Code name") + ylab("Occurence") |
|  |  |
|  | set.seed(123) |
|  | comms <- spinglass.community(g, spins = 100) |
|  | par(mar = rep(0, 4)) |
|  | plot(comms, g, |
|  | layout = layout.fruchterman.reingold, |
|  | vertex.label.cex = .7, |
|  | vertex.size = degree(g), |
|  | vertex.label.color = "black", |
|  | vertex.frame.color = NA, |
|  | edge.color = "black", |
|  | vertex.label.family = "sanserif", |
|  | mark.border = NA |
|  | ) |

Marketing-Definition.R

|  |
| --- |
| library(tidyverse) |
|  | library(rvest) |
|  | library(tidytext) |
|  | library(wordcloud) |
|  | library(RColorBrewer) |
|  | lm\_palette <- c("#f7881f", "#55ace1", "#5f6c36") |
|  |  |
|  | ## Scrape definitions from website |
|  | definitions <- read\_html("https://heidicohen.com/marketing-definition/") %>% |
|  | html\_nodes("ol li") %>% |
|  | html\_text() %>% |
|  | as\_tibble() %>% |
|  | mutate(No = 1:nrow(.)) %>% |
|  | select(No, Definition = value) |
|  |  |
|  | ## bag of words |
|  | def\_words <- definitions[1:72, ] %>% |
|  | unnest\_tokens(word, Definition) %>% |
|  | mutate(word = gsub("s$", "", word)) |
|  |  |
|  | word\_freq <- def\_words %>% |
|  | anti\_join(stop\_words) %>% |
|  | count(word) %>% |
|  | filter(!(word %in% |
|  | c("marketing", "vice", "president", "chief", "executive", "’"))) |
|  |  |
|  | pdf("marketingcloud.pdf") |
|  | word\_freq %>% |
|  | with(wordcloud(word, n, max.words = 50, rot.per = .5, |
|  | colors = rev(lm\_palette))) |
|  | dev.off() |
|  |  |
|  | word\_freq %>% |
|  | top\_n(10) %>% |
|  | mutate(word = reorder(word, n)) %>% |
|  | ggplot(aes(word, n)) + geom\_col(fill = lm\_palette[2]) + |
|  | coord\_flip() + |
|  | theme(text = element\_text(size = 15)) |
|  | ggsave("marketing-words.png", width = 6, height = 6) |
|  |  |
|  | ## Topic modeling |
|  | library(topicmodels) |
|  |  |
|  | marketing\_dtm <- word\_freq %>% |
|  | mutate(doc = 1) %>% |
|  | cast\_dtm(doc, word, n) |
|  |  |
|  | marketing\_lda <- LDA(marketing\_dtm, k = 4) %>% |
|  | tidy(matrix = "beta") %>% |
|  | group\_by(topic) %>% |
|  | top\_n(5, beta) %>% |
|  | ungroup() %>% |
|  | arrange(topic, -beta) |
|  |  |
|  | marketing\_lda %>% |
|  | mutate(term = reorder(term, beta)) %>% |
|  | ggplot(aes(term, beta, fill = factor(topic))) + |
|  | geom\_col(show.legend = FALSE) + |
|  | facet\_wrap(~topic, scales = "free") + |
|  | coord\_flip() + |
|  | theme(text = element\_text(size = 20)) |

Org-Roam.R

|  |
| --- |
| ## Visualise Org-Roam databse |
|  |  |
|  | ## Connect to database and extract nodes and links |
|  | library(DBI) |
|  | roam <- dbConnect(RSQLite::SQLite(), "~/Documents/org-roam/data/org-roam.db") |
|  | nodes <- dbGetQuery(roam, "SELECT id, title FROM nodes") |
|  | links <- dbGetQuery(roam, "SELECT source, dest FROM links WHERE type = '\"id\"'") |
|  | dbDisconnect(roam) |
|  |  |
|  | ## Clean node names and create network table |
|  | library(tidyverse) |
|  | nodes <- nodes %>% |
|  | mutate(title = str\_remove\_all(title, "\"")) |
|  |  |
|  | network <- left\_join(links, nodes, by = c("source" = "id")) %>% |
|  | left\_join(nodes, by = c("dest" ="id")) %>% |
|  | select(from = title.x, to = title.y) |
|  |  |
|  | ## Create network diagram |
|  | library(igraph) |
|  | g <- graph.data.frame(network) |
|  | g <- simplify(g, remove.multiple = TRUE, remove.loops = TRUE) |
|  |  |
|  | ## Centrality |
|  | centrality <- tibble(Node = names(degree(g)), |
|  | Degree = degree(g, mode = "total"), |
|  | Links = degree(g, mode = "out"), |
|  | Backlinks = degree(g, mode = "in")) |
|  |  |
|  | centrality %>% |
|  | top\_n(n = 10) %>% |
|  | arrange(desc(Degree)) |
|  |  |
|  | ## Community-detection |
|  | c <- cluster\_spinglass(g) |
|  |  |
|  | communities <- tibble(Node = c$names, |
|  | Community = c$membership) %>% |
|  | left\_join(centrality) %>% |
|  | group\_by(Community) %>% |
|  | summarise(Name = Node[which.max(Backlinks)], |
|  | Nodes = n()) %>% |
|  | arrange(desc(Nodes)) |
|  |  |
|  | ## Visualise graph |
|  | par(mar = c(0, 0, 0, 0)) |
|  | plot(g, |
|  | layout = layout.fruchterman.reingold, |
|  | mark.border = NA, |
|  | vertex.color = c$membership, |
|  | vertex.label = NA, |
|  | vertex.size = sqrt(degree(g, mode = "in") + 1), |
|  | vertex.frame.color = NA, |
|  | edge.arrow.size = 0, |
|  | edge.color = "lightgrey") |
|  |  |
|  | ## Subnetwork |
|  | node <- which(names(V(g)) == "Topology") |
|  | g1 <- make\_ego\_graph(g, order= 2, nodes = node) |
|  | c1 <- cluster\_walktrap(g1[[1]]) |
|  |  |
|  | plot(g1[[1]], layout = layout.kamada.kawai, |
|  | mark.border = NA, |
|  | vertex.size = sqrt(degree(g1[[1]])) + 2, |
|  | vertex.color = c1$membership, |
|  | vertex.frame.color = NA, |
|  | vertex.label.family = "Sanserif", |
|  | vertex.label.color = "black", |
|  | edge.arrow.size = .3) |
|  |  |
|  | ## Interactive visualisation |
|  | library(networkD3) |
|  | library(htmlwidgets) |
|  |  |
|  | ## Convert to object suitable for networkD3 |
|  | d3 <- igraph\_to\_networkD3(g, group = communities$Name[membership(c)]) |
|  | d3$nodes$backlinks <- degree(g, mode = "in") |
|  |  |
|  | ## Create force directed network plot |
|  | p <- forceNetwork(Links = d3$links, Nodes = d3$nodes, |
|  | Source = "source", Target = "target", |
|  | NodeID = "name", Group = "group", |
|  | Nodesize = "backlinks", |
|  | zoom = TRUE, legend = TRUE) |
|  | saveWidget(p, file=paste0( getwd(), "/org-roam.html")) |

Site-Structure.R

|  |
| --- |
| ## Load data (CSV exported from wp\_posts table in MySQL database) |
|  | library(tidyverse) |
|  | library(tidytext) |
|  | library(igraph) |
|  | library(RMySQL) |
|  | library(RColorBrewer) |
|  |  |
|  | ## Download data using RmySQL |
|  | source("dbconnect.R") |
|  | ## lucidmanager <- dbConnect(MySQL(), |
|  | ## user = "username", |
|  | ## password = "Password1", |
|  | ## dbname = "databe name", |
|  | ## host = "URL") |
|  |  |
|  | posts.query <- "SELECT p.post\_name, p.post\_content, t.name |
|  | FROM wp\_posts p |
|  | JOIN wp\_term\_relationships tr ON (tr.object\_id = p.ID) |
|  | JOIN wp\_term\_taxonomy tt ON (tt.term\_taxonomy\_id = tr.term\_taxonomy\_id ) |
|  | JOIN wp\_terms t ON (t.term\_id = tt.term\_id) |
|  | WHERE |
|  | p.post\_type='post' |
|  | AND |
|  | p.post\_status='publish' |
|  | AND |
|  | tt.taxonomy = 'category'" |
|  |  |
|  | pages.query <- "SELECT post\_name, post\_content |
|  | FROM wp\_posts |
|  | WHERE post\_type='page' AND post\_status='publish'" |
|  |  |
|  | posts.data <- dbSendQuery(lucidmanager, posts.query) |
|  | posts <- fetch(posts.data, n=-1) |
|  |  |
|  | pages.data <- dbSendQuery(lucidmanager, pages.query) |
|  | pages <- fetch(pages.data, n=-1) |
|  |  |
|  | dbDisconnect(lucidmanager) |
|  |  |
|  | pages$name = "Page" |
|  | content <- rbind(posts, pages) |
|  | as\_tibble(content) |
|  |  |
|  | ## Extract internal links |
|  | links <- content %>% |
|  | unnest\_tokens(word, post\_content, |
|  | token = stringr::str\_split, pattern = " ") %>% |
|  | filter(grepl(".+lucidmanager.org/", word)) %>% |
|  | mutate(link = gsub(".+lucidmanager.org/", "", word), |
|  | link = gsub("/.+", "", link)) %>% |
|  | filter(link != "tag" & link != "wp-content" & link != "author") %>% |
|  | select(category = name, post\_name, link) |
|  |  |
|  | tibble(links) |
|  |  |
|  | solitary <- c(posts$post\_name, pages$post\_name)[!(c(posts$post\_name, pages$post\_name) %in% |
|  | unique(c(links$post\_name, links$link)))] |
|  |  |
|  | ## Create network |
|  | network <- select(links, post\_name, link) %>% |
|  | graph\_from\_data\_frame(directed = TRUE) + |
|  | vertices(solitary) |
|  |  |
|  | colour <- tibble(post\_name = V(network)$name) %>% |
|  | left\_join(content) %>% |
|  | mutate(name = factor(name), |
|  | colour = brewer.pal(length(unique(content$name)), "Set2")[name]) %>% |
|  | select(-post\_content) |
|  |  |
|  | V(network)$color <- as.vector(colour$colour) |
|  |  |
|  | par(mar = rep(0, 4)) |
|  | plot(network, |
|  | layout = layout.fruchterman.reingold, |
|  | vertex.label.cex = .7, |
|  | vertex.size = degree(network), |
|  | vertex.label.color = "black", |
|  | vertex.frame.color = NA, |
|  | edge.arrow.size = .2, |
|  | edge.color = "darkgray" |
|  | ) |
|  |  |
|  | ## Diagnose network |
|  | which.max(degree(network, mode = "in")) |
|  |  |
|  | E(network)[which\_loop(network)] |
|  |  |
|  | E(network)[which\_multiple(network)] |

Songsim.R

|  |
| --- |
| ## Decoding lyrics |
|  | library(tidyverse) |
|  | library(tidytext) |
|  | library(reshape2) |
|  | library(rvest) |
|  |  |
|  | get\_lyrics <- function(artist, song) { |
|  | artist <- gsub("[^A-Za-z0-9]+", "", tolower(artist)) |
|  | song <- gsub("[^A-Za-z0-9]+", "", tolower(song)) |
|  | artist <- gsub("^the", "", artist) |
|  | url = paste("http://azlyrics.com/lyrics/", |
|  | artist, "/", song, ".html", sep = "") |
|  | print(url) |
|  |  |
|  | azlyrics <- read\_html(url) |
|  | lyrics <- html\_nodes(azlyrics, "div") |
|  | lyrics <- html\_text(lyrics[23]) |
|  | gsub("\r|\n", " ", lyrics) |
|  | } |
|  |  |
|  | plot\_snowflake <- function(artist, song){ |
|  |  |
|  | lyrics <- get\_lyrics(artist, song) |
|  | lyrics <- data\_frame(line = lyrics) %>% |
|  | filter(line != "") |
|  |  |
|  | words <- lyrics %>% |
|  | unnest\_tokens(word, line) |
|  | words\_matrix <- lapply(1:nrow(words), |
|  | function(w){ |
|  | as.character(words[w, 1]) == words |
|  | } |
|  | ) %>% |
|  | do.call(cbind, .) |
|  | rownames(words\_matrix) <- 1:nrow(words) |
|  | colnames(words\_matrix) <- 1:nrow(words) |
|  |  |
|  | melt(words\_matrix, varnames = c("x", "y")) %>% |
|  | ggplot(aes(x, -y, fill = value)) + |
|  | geom\_raster() + |
|  | scale\_fill\_manual(values = c("white", "dodgerblue4"), guide = FALSE) + |
|  | theme\_void() + |
|  | ggtitle(artist, subtitle = song) |
|  | } |
|  |  |
|  |  |
|  | plot\_snowflake("Abba", "Waterloo") |
|  | ggsave("Abba-Waterloo.png") |
|  |  |
|  | artist = "Thomas Edison" |
|  | song <- "Mary Had a Little Lamb" |
|  | lyrics <- "Mary had a little lamb, whose fleece was white as snow. And everywhere that Mary went, the lamb was sure to go." |
|  |  |
|  | library(gridExtra) |
|  | png("DaftPunk-Queen.png", width = 1024, height = 768) |
|  | l1 <- plot\_snowflake("Daft Punk", "Around the world") |
|  | l2 <- plot\_snowflake("Queen", "Bohemian Rhapsody") |
|  | grid.arrange(l1, l2, ncol = 2) |
|  | dev.off() |
|  | getwd() |
|  |  |
|  | artist <- "Frank Zappa" |
|  | song <- "Titties Beer" |

Visualize-wordpress-links.R

|  |
| --- |
| ## Visualise internal link structure |
|  |  |
|  | ## Load data (CSV exported from wp\_posts table in MySQL database) |
|  |  |
|  | library(tidyverse) |
|  | library(tidytext) |
|  | library(igraph) |
|  | library(RMySQL) |
|  |  |
|  | lucidmanager <- dbConnect(MySQL(), |
|  | user = "peterpr5\_rmysql", |
|  | password = "cy7J}O-8IsUQ", |
|  | dbname = "peterpr5\_WPZ1W", |
|  | host = "lucidmanager.org") |
|  |  |
|  | dbListTables(lucidmanager) |
|  |  |
|  | posts <- dbSendQuery(lucidmanager, "SELECT ID, post\_type, post\_name, post\_title, post\_content FROM \_Z1W\_4\_posts") |
|  | on.exit(dbDisconnect(lucidmanager)) |
|  |  |
|  | data = fetch(posts, n=-1) |
|  | data |
|  | as.data.frame(posts) |
|  |  |
|  | dbUnloadDriver(lucidmanager) |
|  |  |
|  | posts <- read\_csv("wp\_posts.csv") %>% |
|  | filter((post\_type == "page" | post\_type == "post") & post\_status == "publish") %>% |
|  | select(ID, post\_date, post\_type, post\_name, post\_title, post\_content) |
|  |  |
|  | internal\_links <- unnest\_tokens(posts, word, post\_content, |
|  | token = stringr::str\_split, pattern = " ") %>% |
|  | filter(grepl(".+lucidmanager.org/", word)) %>% |
|  | mutate(link = gsub(".+lucidmanager.org/", "", word), |
|  | link = gsub("/.+", "", link)) %>% |
|  | filter(link != "tag" & link != "wp-content" & link != "author") %>% |
|  | select(post\_type, post\_title, post\_name, link) |
|  | internal\_links |
|  |  |
|  | link\_network <- internal\_links %>% |
|  | select(post\_name, link) %>% |
|  | graph\_from\_data\_frame() |
|  |  |
|  | E(link\_network)$weight |
|  |  |
|  | pdf("links.pdf") |
|  | plot(link\_network, |
|  | layout = layout.fruchterman.reingold, |
|  | vertex.label.cex = .3, |
|  | vertex.size = degree(link\_network), |
|  | vertex.label.color = "black", |
|  | vertex.frame.color = "white", |
|  | vertex.color = "Dodgerblue", |
|  | edge.arrow.size = .3, |
|  | edge.color = "darkgray" |
|  | ) |
|  | dev.off() |

**Scraping text with Rvest**

Web scraping is a common technique to download data from websites where this data is not available as a clean data source. Web scraping starts with downloading the HTML code from the website and the filtering the wanted text from this file. The rvest package makes this process very easy.

The code for this article uses a pipe (%>%) with three rvest commands. The first step is to download the wanted html code from the web using the read\_html function. The output of this function is piped to the html\_nodes function, which does all the hard work. In this case, the code picks all lines of text that are embedded in an

container, i.e. ordered lists. You can use the [SelectorGadget](http://selectorgadget.com/) to target the text you like to scrape. The last scraping step cleans the text by piping the output of the previous commands to the html\_text function.

The result of the scraping process is converted to a Tibble, which is a type of data frame used in the Tidyverse. The definition number is added to the data, and the Tibble is converted to the format required by the Tidytext package. The resulting data frame is much longer than the 72 definitions because there are other lists on the page. Unfortunately, I could not find a way to filter only the 72 definitions.

library(tidyverse)

library(rvest)

definitions <- read\_html("https://heidicohen.com/marketing-definition/") %>%

html\_nodes("ol li") %>%

html\_text() %>%

as\_data\_frame() %>%

mutate(No = 1:nrow(.)) %>%

select(No, Definition = value)

**Tidying the Text**

The Tidytext package extends the tidy data philosophy to a text. In this approach to text analysis, a corpus consists of a data frame where each word is a separate item. The code snippet below takes the first 72 rows and the unnest\_tokens function extracts each word from the 72 definitions. This function can also extract ngrams and other word groups from the text. The Tidytext package is an extremely versatile piece of software which goes far beyond the scope of this article. Julia Silge and David Robinson have written a book about text mining using this package, which provides a very clear introduction to the craft of analysing text.

The last section of the pipe removes the trailing “s” from each word to convert plurals into single words. The mutate function in the Tidyverse creates or recreates a new variable in a data frame.

library(tidytext)

def\_words <- definitions[1:72, ] %>%

unnest\_tokens(word, Definition) %>%

mutate(word = gsub("s$", "", word))

This section creates a data frame with two variables. The No variable indicates the definition number (1–72) and the word variable is a word within the definition. The order of the words is preserved in the row name. To check the data frame you can run unique(def\_words$No[which(def\_words$word == "marketing")]). This line finds all definition numbers with the word “marketing”, wich results, as expected, in the number 1 to 72.

**Using Rvest and Tidytext to define marketing**

We can now proceed to analyse the definitions scraped from the website with Rvest and cleaned with Tidytext. The first step is to create a word cloud, which is a popular way to visualise word frequencies. This code creates a data frame for each unique word, excluding the word marketing itself, and uses the wordcloud package to visualise the fifty most common words.

library(wordcloud)

library(RColorBrewer)

word\_freq <- def\_words %>%

anti\_join(stop\_words) %>%

count(word) %>%

filter(word != "marketing")

word\_freq %>%

with(wordcloud(word, n, max.words = 50, rot.per = .5,

colors = rev(brewer.pal(5, "Dark2"))))

While a word cloud is certainly a pretty way to visualise the bag of words in a text, it is not the most useful way to get the reader to understand the data. The words are jumbled, and the reader needs to search for meaning. A better way to visualise word frequencies is a bar chart. This code takes the data frame created in the previous snippet, determines the top ten occurring words. The mutate statement reorders to factor levels so that the words are plotted in order.

word\_freq %>%

top\_n(10) %>%

mutate(word = reorder(word, n)) %>%

ggplot(aes(word, n)) +

geom\_col(fill = "dodgerblue4") +

coord\_flip() +

theme(text = element\_text(size=20))

A first look at the word cloud and bar chart suggests that marketing is about customers and products and services. Marketing is a process that includes branding and communication; a simplistic but functional definition.

**Topic Modeling using Tidytext**

Word frequencies are a weak method to analyse text because it interprets each word as a solitary unit. Topic modelling is a more advanced method that analyses the relationships between words, i.e. the distance between them. The first step is to create a [Document-Term Matrix](https://en.wikipedia.org/wiki/Document-term_matrix), which is a matrix that indicates how often a word appears in a text.  As each of the 72 texts are very short, I decided to treat the collection of definitions as one text about marketing. The cast\_dtm function converts the data frame to a Document-Term Matrix.

The following pipe determines the top words in the topics. Just like k-means clustering, the analyst needs to choose the number of topics before analysing the text. In this case, I have opted for 4 topics. The code determines the contribution of each word to the four topics and selects the five most common words in each topic. The faceted bar chart shows each of the words in the four topics.

marketing\_dtm <- word\_freq %>%

mutate(doc = 1) %>%

cast\_dtm(doc, word, n)

marketing\_lda <- LDA(marketing\_dtm, k = 4) %>%

tidy(matrix = "beta") %>%

group\_by(topic) %>%

top\_n(5, beta) %>%

ungroup() %>%

arrange(topic, -beta)

marketing\_lda %>%

mutate(term = reorder(term, beta)) %>%

ggplot(aes(term, beta, fill = factor(topic))) +

geom\_col(show.legend = FALSE) +

facet\_wrap(~topic, scales = "free") +

coord\_flip() +

theme(text = element\_text(size=20))

This example also does not tell me much more about what marketing is, other than giving a slightly more sophisticated version of the word frequency charts. This chart shows me that marketing is about customers that enjoy a service and a product. Perhaps the original definitions are not distinctive enough to be separated from each other. The persistence of the word “president” is interesting as it seems to suggest that marketing is something that occurs at the highest levels in the business.

This article is only a weak summary of the great work by Julia Silge who coauthored the Tidytext package. The video below provides a comprehensive introduction to topic modelling.

**What have we learnt?**

This excursion into text analysis using rvest and Tidytext shows that data science can help us to make some sense out of an unread text. If I did not know what this page was about, then perhaps this analysis would enlighten me. This kind of analysis can assist us in wading through to large amounts of text to select the ones we want to read. I am still not convinced that this type of analysis will provide any knowledge beyond what can be obtained from actually reading and engaging with a text.