T.C.
MARMARA UNIVERSITY
INSTITUTE OF SOCIAL SCIENCE
DEPARTMENT OF BUSINESS ADMINISTRATION
DISCIPLINE OF QUANTITATIVE METHODS

SOCIAL MEDIA MINING WITH R

Master of Science Dissertation

SAHBAN TARIQ MALIK

ISTANBUL, 2018

T.C.
MARMARA UNIVERSITY
INSTITUTE OF SOCIAL SCIENCE
DEPARTMENT OF BUSINESS ADMINISTRATION
DISCIPLINE OF QUANTITATIVE METHODS

SOCIAL MEDIA MINING WITH R

Master of Science Dissertation

SAHBAN TARIQ MALIK

SUPERVISOR: Dr. Öğr.Üyesi Ayşe Çınar

ISTANBUL, 2018

T.C.
MARMARA ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜ

TEZ ONAY BELGESİ

İŞLETME (İNGİLİZCE) Anabilim Dalı SAYISAL YÖNTEMLER (İNGİLİZCE) Bilim Dalı TEZLİ YÜKSEK LİSANS öğrencisi SAHBAN TARIQ MALIK'nın SOCIAL MEDIA MINING WITH R adlı tez çalışması, Enstitümüz Yönetim Kurulunun 7.06.2018 tarih ve 2018-16/34 sayılı kararıyla oluşturulan jüri tarafından oy birliği / oy çokluğu ile Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Savunma Tarihi 13/06/2018

Öğretim Üyesi Adı Soyadı			**İmzası**
1.	Tez Danışmanı	Dr. Öğr. Üyesi AYŞE ÇINAR	
2.	Jüri Üyesi	Prof. Dr. GÖKHAN SİLAHTAROĞLU	
3.	Jüri Üyesi	Prof. Dr. SERRA YURTKORU	

GENEL BİLGİLER

İsim ve Soyadı: Sahban Tariq Malik
Anabilim Dalı: İşletme (İng.)
Programı: Sayısal Yöntemler (İng.)
Tez Danışmanı: Dr. Ayşe Çınar

Tez Türü ve Tarihi: Yüksek Lisans – Mayıs 2018
Anahtar Kelimeler: Veri Madenciliği, Metin Madenciliği, Duygu Analizi, Kelime bulutu, Tokenization

ÖZET

R İLE SOSYAL MEDYA MADENCİLİĞİ

Bu çalışmada, sosyal media web uygulamalarından veri çıkartma, veri hazırlama veya düzeltme, tokenizasyon, kelime sıklığı tahminleme, kelime yığınının duygu analizi ve görselleştirilmesi gibi birçok sosyal media madencilik tekniği R ortamında uygulanmıştır. Bu teknikler için R fonksiyonları oluşturularak, bu fonksiyonlar Türk Hava Yolları vaka çalışmasında uygulanmıştır. Sosyal media web uygulamalarından R programlama dili kullanılarak, söz konusu şirkete ait Facebook ve TripAdvisor web sayfasında yer alan yorum ve görüşleri kapsayan sosyal media verisi alınmıştır. Facebook ve TripAdvisor'dan veri temini için R paketleri kullanılmıştır. Bir sonraki aşamada, kompleks veri yapısı ve gereksiz sütunlar içeren dağınık ham veri, biri Facebook diğeri TripAdvisor için, `dataframe` yapısında iki farklı düzenli veri setine dönüştürülmüştür. Çalışmanın devamında yorum ve görüşlerden oluşan veri seti tokenize tekniği ile cümlelere ve ardından sözcüklere indirgenmiştir. Bunu yanısıra, veri seti içinde yer alan geçmiş zaman ve şimdiki zaman fiilleri de kök fiil haline dönüştürülmüştür. Çalışmanın son aşamasında, tokenize edilmiş olan Facebook yorumları ve TripAdvisor görüşleri üzerinde; kelime sıklık sayımı, duygu analizi ve kelime yığını görselleştirmesi gibi çeşitli metin analiz teknikleri uygulanmıştır.

GENERAL INFORMATION

Name and Surname:	Sahban Tariq Malik
Field:	Business Administration
Programme:	Quantitative Methods
Supervisor:	Dr. Ayşe Çınar
Degree Awarded and Date:	Master – May 2018

Keywords:	Data Mining, Text Mining, Tokenization Social Media Mining, Tidy Data Natural Language Processing, Word Cloud, Sentiment Analysis, R Programming Language.

ABSTRACT

SOCIAL MEDIA MINING WITH R

In this research study, many social media mining techniques, such as data extraction, data wrangling or tidying, tokenization, estimation of word frequency, sentiment analysis and visualization of word cloud, have been applied in R environment. The study builds R functions for these techniques. Later, these functions are used in the case study on Turkish Airlines. TurkishAirlines' social media data, i.e. comments posted by TurkishAirlines' Facebook followers and reviews posted by the customers on TripAdvisor Website, are scraped from the social media web applications using R programming language. R packages, built for web scraping, are used to retrieve data from Facebook and TripAdvisor. Afterward, the messy extracted data, with the complex data structure and unnecessary columns, are converted into two different tidy datasets, one for Facebook and other for TripAdvisor. Subsequently, the responses, either comments or reviews, are tokenized into sentence and words. The tokenized data have been cleaned by extracting NA values and stop words. Moreover, the verbs in different forms, such as present simple, present participle, past simple and past participle, are converted into the base form of verbs. Lastly, text analysis techniques such as word frequency count, sentiment analysis, and word cloud visualization are applied to tokenized Facebook comments and TripAdvisor reviews.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

With the rise in social media popularity, the trend of writing reviews and feedback on social media websites has increased dramatically. These reviews posted by the existing customers strongly influence the buying behavior of potential customers. Consequently, the consumer brands have started investing in social media branding and advertising campaigns. In contrast to other mediums of advertisements, social media advertisement not only results in huge response by fans and followers, but also benefits the companies by generating leads. A powerful social media marketing campaign with an effective strategy would develop positive brand image among consumers, whereas a campaign with an ineffectual strategy would result in negative responses by the potential or existing consumers.

An effective social media marketing strategy cannot be devised without having an up-to-the-minute information about consumers' perception on the brand and its customer services. A deep understanding of consumers' and potential customers' perception can only be obtained from their feedback, reviews, criticisms or comments on the social media marketing campaigns. By using the latest data mining and particularly text mining techniques, the big data of customers' reviews and comments can be successfully used to get a strong insight into consumers' perception towards customer services of any company.

There are number of programming frameworks that can be used to extract consumers' reviews and comments from different social media web applications, such as Python, PHP and Perl. The study used statistical software environment R and R programming language to extract and analyze data from social media web applications, like Facebook and TripAdvisor. The open source R software, developed for efficient data handling, data analysis, is based on the S language. Ross Ihaka & Robert Gentleman wrote an

experimental R to be used in their teaching laboratory. Upon receiving positive feedback, they release R source code as “free software” in June of 1995. With the rise in interest in R, the developers introduce the mailing list and people started porting applications to it (Ihaka 1998).

Today, despite being outside the conventional programming language, R is the most famous statistical software used by a large community of programmers. One of the reasons for R’s popularity is its package system. R packages encapsulate the user-contributed code, data, functions, tests and documentation together. More than 6,000 packages are available on the CRAN (Comprehensive R Archive Network) system. It is as a public clearing house for packages built in R (Wickham, R packages: organize, test, document, and share your code 2015). The huge variety of easily accessible and useful packages is one of the reasons why R is so successful. The study used famous R packages such as `Rfacebook`, `rvest`, `dplyr`, `ggplot2`, `tidyr` and `tidytext` for different data mining techniques. These packages when loaded in an R environment provide a number of beneficial functions that save time for programmers while writing complex codes. This study thoroughly explains how to use these functions effectively for social media mining. Furthermore, to avoid code redundancy, the study provides new functions that effectively deal with Facebook and TripAdvisors data at a same time. These functions are used to clean and analyze Turkish Airlines’ social media data.

2. Text Mining

One of the rapidly growing, exciting and new areas of computer science research is text mining. It aims to deal with information overload by using multiple techniques such as data mining, information retrieval, natural language processing and machine learning. In this section, the study provides an overview of different text mining technique along with their real-world applications for businesses in various fields, such as social media analytics, marketing and brand management.

In general, we can define text mining as an information-intensive process in which a user interacts with a collection of documents through various analysis tools. As similar to data mining process, text mining aims to retrieve beneficial information from text data sources through identification and exploration of words used in the text documents. Furthermore, this mining methodology seeks to identify and discover interesting patterns in text documents to extract valuable knowledge efficiently. Indeed, text mining derives its way and motivation from influential research on data mining. Therefore, architectural similarities can be observed in text mining and data mining system, such as preprocessing techniques, prediction and pattern discovery algorithms and visualization techniques (Feldman and Sanger 2007).

In contrast to data mining system where data is stored in structured format, the document based text data are initially stored in weekly or relatively less structured format. Therefore, the text data are first transformed from weekly structured format into a more structured format using some preprocessing techniques, which are not related to data mining system (Feldman and Sanger 2007). In addition to the methodologies and techniques derived from data mining system, text mining derives many techniques from different areas, such as natural language processing, information retrieval, and corpus-based computational linguistics.

The study below explains some of the famous text mining techniques. Although the research study does not implement all of the techniques mentioned in this section, the study aims to provide the summary of text mining techniques and their applicability in the real world.

2.1 Collection of Document

The key component of text mining is a collection of documents. In simple words, a collection of documents is a grouping of similar text-based documents (Feldman and Sanger 2007). The total number of documents in a collection can reach from many thousands to many millions. Most text mining techniques aim to extract patterns from these collections and analyze these patterns to get valuable insights. For instance, a collection of research papers in the field of marketing can be considered as a document collection. Similarly, the reviews of a tourism brand altogether become a collection of documents, when a single review is considered as a document.

The text mining techniques are not implemented on unprepared and unstructured documents. A considerate amount of time and emphasis is devoted in preprocessing procedures. These preprocessing methods include data extraction, data wrangling, tokenization and linguistic research methodologies.

2.2 Document

Document is another basic component in text mining. Document can be defined as that unit of document collection that correlates, not necessarily, with some other real-life published document, such as an article, e-mail, news, reviews, research paper and social media posts. In other words, document is a collection of words that represents a similar type of units within a document collection. It cannot be inferred that a document can only exists within one specific collection, however, document can be a member of many different types of collections (Feldman and Sanger 2007). For instance, considering this research study as a document, the study can be a member of more than one document collections, such as text mining, information retrieval, and natural language processing.

2.2.1 Document Structure

In contrary to the misleading tag that document is an unstructured data, it can be seen as a somewhat structured object. According to a linguist's point of view, even a simple document has a fair amount of syntactic and semantic structure. With the help of typographical elements, such as capitalization, punctuation marks, spacing, underlining and tables, we can identify document's components and subcomponents such as titles, paragraphs, table records and author names (Feldman and Sanger 2007). Furthermore, a sequence of words can also be considered as a meaningful dimension of document structure. Documents, including business reports, news articles or stories, legal communications, are considered to be weekly structured documents as they do not have strong typographical or layout marks to define structure. Whereas, HTML pages, emails and PDF document having style-sheet, HTML tags, and heavy typographical elements are usually considered as semi structured documents.

2.3 Document Features

To explore the implicit structure of documents and convert the implicit representation of document structure to an explicit representation, it is important to identify the document features that represent documents as a whole. The design, approach and even performance of text mining preprocessing techniques are based on the document's features. Some of the commonly used document features discussed in this section are characters – the building block- words, terms and concepts. The aim of text mining is to identify the document features; moreover, normalize and validate these features against dictionaries or other vocabulary sources, such as lexicon or thesauri.

2.3.1 Characters

As mentioned above, characters are the building block of other higher-level semantic document features, i.e. words and terms. The characters include basic-level letters, numbers, special characters and even spaces. The character is the most basic document feature and that is why it offers very limited benefit in the application of text mining techniques. The main advantage of

characters, especially special characters and spacing, is to identify the positions of a word, a sentence, or a set of different words (bigrams or trigrams).

2.3.2 Words

Word-level feature is one of the basic level features of a document. This feature has more semantic richness than character and that is why it is more meaningful for text mining analysis. The focus of this research study is on word-level text mining analytics. A word in a document is also known as a linguistic token. The word level representation of a document or a document collection usually contains hundreds and thousands of distinct words. The process of representing a document in words is also known as tokenization. Some level of optimization is required before optimal word-level representation of document, such as extraction of stop-words, numbers and characters.

2.3.3 Terms

Single words (unigram) and multiword (n-gram) phrases are the terms that are extracted from the collection of documents using term-extraction techniques. Term-level representation of a document contains a subset of all the terms existing in the document. For example, a documents contains the sentence “They have a nice inflight entertainment system and the high speed Wi-Fi was free for Business Class and very reasonable for economy class”. Some of single word form terms that will be included to represent the document will include, “nice”, “inflight”, and “Wi-Fi”. Moreover, the terms will also include n-gram words such as “Business Class” and “economy class” and “entertainment system”. Term extraction techniques are more sophisticated than tokenization (or word extraction) techniques. The raw text document is represented in normalized terms, i.e. sequence of tokens connected with different parts of speech. These term extraction methodologies, like term normalization are not explained and employed in this research study.

2.3.4 Concepts

In comparison with the three above mentioned document features, concepts feature is more sophisticated feature, and therefore requires highly advanced text mining techniques to be extracted from the documents. The concepts in a document are usually extracted manually; however, using statistical, hybrid categorization, rule-based and other complex preprocessing techniques, they can be identified from documents (Feldman and Sanger 2007). For instance, a review of airline industry may not necessarily include “entertainment” or “aircraft” but these concepts might be found among the list of concepts required to represent the document collection.

As compare to other features explained above, terms and concepts are the most expressive features of the document with higher semantic value. However, the study emphasizes on the word level and character level features, since; other features require more advanced techniques.

In this section, the study explains some of the text mining techniques that are not implemented in this study. The application of these analysis techniques is high when there are more than one documents and the aim is to explore the association of words between different documents. These text mining techniques include vector models, in which a word is represented as a vector. These vector models are most commonly used for semantic similarity, which is the similarity between different words, sentences and documents.

2.4 Term Document Matrix

The term documents matrix is a two dimensional table in which each row represents a specific word in the vocabulary while each column signifies a document. Furthermore, each cell of a term document matrix indicates the frequency of words in a specific document – represented by a column. In the field of information retrieval, the term document matrix was defined as a vector space model (Salton 1972). A vector is an array of number, while a vector space is a collection of vectors that have particular dimensions. The position of each vector element represents a particular dimension, therefore; the sequence of elements in a vector is not arbitrary.

2.5 Semantic Similarity

As mentioned, the vector space models help to find the semantic similarity between different documents. The documents are represented by column vectors in vector space model or term documents matrix (Jurafsky and Martin, Speech and Language Processing 2017). The spatial visualization of these vectors can help to explore the semantic similarity between documents. In a spatial visualization, the vectors nearer to each other are more similar than the vectors distant from each other. Generally, the vectors have tens of thousands of elements and therefore it is impossible to visualize that highly dimensional vectors and visualize the similarity. The problem of finding the similarity between high dimensional vectors is addressed in the subsection below.

2.5.1 Similarity Measurement

One of the widely used semantic similarity metric is the cosine, similar to most of the vector similarity measures used in the field of natural language processing. The cosine methodology is based on the linear algebra operation, i.e. inner product or dot product.

$$dot - product(\bar{v}, \bar{w}) = \bar{v} \,.\, \bar{w} = \sum_{i=1}^{N} v_i \, w_i = v_1 w_1 + v_2 w_2 + \cdots + v_N w_N$$

The cosine method or inner product serves as the similarity metric because it will give a large value when the two vectors are close to each other or there is high similarity between each other. Alternatively, a low value of dot product indicates that there is high level of dissimilarity between vectors. If the dot product results in value zero, it means that the vectors are orthogonal vectors and completely dissimilar to each other.

There are number of other similarity metrics, such as Jaccard (Jaccard 1912), Dice (Curran 2003) Kullback-Leibler divergence or relative entropy (Kullback and Leibler 1951) and Janson-Shannon divergence (Lee 1999).

In the next two sub sections, the two major types of classification techniques, supervised and unsupervised, are discussed briefly.

2.6 Supervised Classification

In this section, supervised classification techniques and their application in text mining are discussed briefly. These techniques are used to assign labels to tokens, terms, and documents. The categorization labels, like positive, negative, and spam detection labels, are drawn from a set of labeled lexicon or training dataset. The supervised classification techniques use these lexicon and training dataset and classify test data into different labels. When the analysis is conducted at word-level document feature, the tokens or words can be classified by comparing them with different lexicons as conducted in this study. However, in a term-level document feature analysis, when the aim is to label a sentence or a whole paragraph, more advanced supervised classification machine learning techniques can be applied, such as naive Bayes, logistic regression, support vector machine (SVM), random forests and neural networks. The goal of supervised classification of terms, i.e. n-gram tokens, is to classify the term into one of a set of different classes. These machine-learning algorithms build classifier model of each class using training dataset. The training dataset is made using human intelligence and each word in a training dataset is categorized and labeled. Subsequently, when classifier models are applied to the observations, the algorithms return the label or a class that has most likely generated each observation. For instance, in the case of Bayesian classification, the classifier model for each class is made using Bayes' rule. To apply the Bayes classifier, word position in a term is considered by simply indexing every word position in a document. Subsequently, the occurrence probability of a term in each class is estimated (Jurafsky and Martin, Speech and Language Processing 2017). Finally, the term (n-gram) with the highest probability of occurring in a specified class is categorized in that class.

2.7 Unsupervised Classification

In the case of huge textual datasets, when it is difficult and expensive to label each word for its topic or sense, an unsupervised approach is used to induce the sense of each word without human interaction. This unsupervised text mining approach is also known as word sense induction (WSI) (Jurafsky and Martin, Speech and Language Processing 2017). Most of the

techniques of unsupervised classification required clustering algorithm. In word sense induction, first the data is trained in four steps: first, document is tokenized into words; second, a context vector – defines the presence of word in a particular dependency - is computed for each word; third, the tokens are classified into different clusters using clustering technique; last, the vector centroid of each cluster is computed that represent the sense or topic of that cluster. Furthermore, using the clustering algorithm and the distance metric between context vectors, the tokens are classified into right clusters (Schutze 1992).

2.7.1 Agglomerative Clustering

Agglomerative clustering is one of the major unsupervised classification techniques. In this clustering technique, each training case is initially assigned to its own cluster. Subsequently, new clusters are made in a bottom-up manner by continuous merging of two different but most similar clusters. The clustering process continues until either an optimal value of a global measure is achieved or some specified number of clusters is made. In the case of large training dataset that is making the agglomerative technique very expensive, random sampling technique can be implemented on the training dataset to obtain same results (Jurafsky and Martin, Speech and Language Processing 2017).

2.8 Social Media Mining

In past ten years, social media has been widely used as a major channel of communication that allows users and businesses to connect and interact on a worldwide level (Xu 2016)**.** The most popular social media website, Facebook, has over one billion active members and more than 890 million active users everyday (Carlsson et al. 2015). Twitter is another dominant micro-blogging social media network that has more than 330 million monthly active users (Statista, Facebook - Statistics & Facts 2018). According to Statista, TripAdvisor has become top ranked tourism website with 7 million listings for hotels, airlines, restaurants and attractions, and over 460 million user opinions and reviews on the listings Statis (Statista, TripAdvisor - Statistics & Facts

2018). With the advancement of technology, the social media websites are enabled to manage, store and analyze a significant amount of user-generated data (Kleindienst et al. 2015). The rapidly growing usage social media and the user generated data, in the form of texts, images, video and geographical locations, provides an opportunity for business enterprises to extract business and customer insights (Schreck and Keim 2013). The enormous social media data can play major role in modern business strategy making processes, since they provide a unique opportunity to gain insight on customer perception and maintain leverage over the competitors (Chen et al. 2012). A survey carried out in November 2016 - asking individuals about the affect of customers' reviews on their opinion about businesses - found that online customer reviews play a vital role in shaping people's opinion about brands, such as a positive review make them trust a business more (Statista, How do online customer reviews affect your opinion of a local business? 2017).

Web scraping is a data mining technique to extract data from Internet through various means (Pereira and T 2015) With the help of web scraping services, the unstructured data is converted into structured data and stored into a data bank. Renita and Vanitha in their paper considers web scraping as the primary step to transform useful user generated data into business asset (Pereira and T 2015)

It is said that 80 percent of data analysis effort is spend in the process of preparing and tidying data (Dasu and Johnson 2003)**.** Hadley Wickham (2014) in his paper "Tidy Data" provides an efficient process of data tidying, i.e. structuring data sets to facilitate analysis. The paper provides a standard way to organize data values in a data table. The core principles of tidy data are based on Codd's relational algebra (Codd 1990) and relational database. However, the principles are framed in a language known to statisticians (Wickham, Tidy Data 2014)**.**

In text mining, tidying dataset includes tokenization of document. Therefore, a tidy text format is defined as being a dataset with one-token-per-row (Silge and Robinson, Text Mining with R - A Tidy Approach 2018). Tokenization is the process of identification of tokens, a meaningful unit of a text, i.e. word or a sentence. A foremost step for information retrieval is tokenization and token indexing on the basis of some parameters (Dong, Husain and E. Chang 2008). The main advantage of tokenization is that it allows us to use storage place effectively (Wong et al. 1985). Furthermore, in addition to the identification of tokens, tokenization includes the estimation of token or word count (Singh and Saini 2014).

Sentiment analysis (also known as opinion mining) is the method of analyzing people's emotions, opinions, and sentiments toward an object (Ahmadi 2017). According to most of the resources, sentiment analysis and opinion mining have exactly the same meaning; however, some consider opinion mining to be different from sentiment analysis (Ahmadi 2017). Opinion mining is the way of extracting people's opinion from the document, whereas sentiment analysis explores the sentiments of the given text document (Ahmadi 2017).

3. METHODOLOGY

This section describes a complete methodology to retrieve data from social media websites, including Facebook, Twitter and TripAdvisor. Furthermore, the process of extracting useful information from raw data has also been explained in this section. The section is divided into five major sub sections as shown in Figure 1 below:

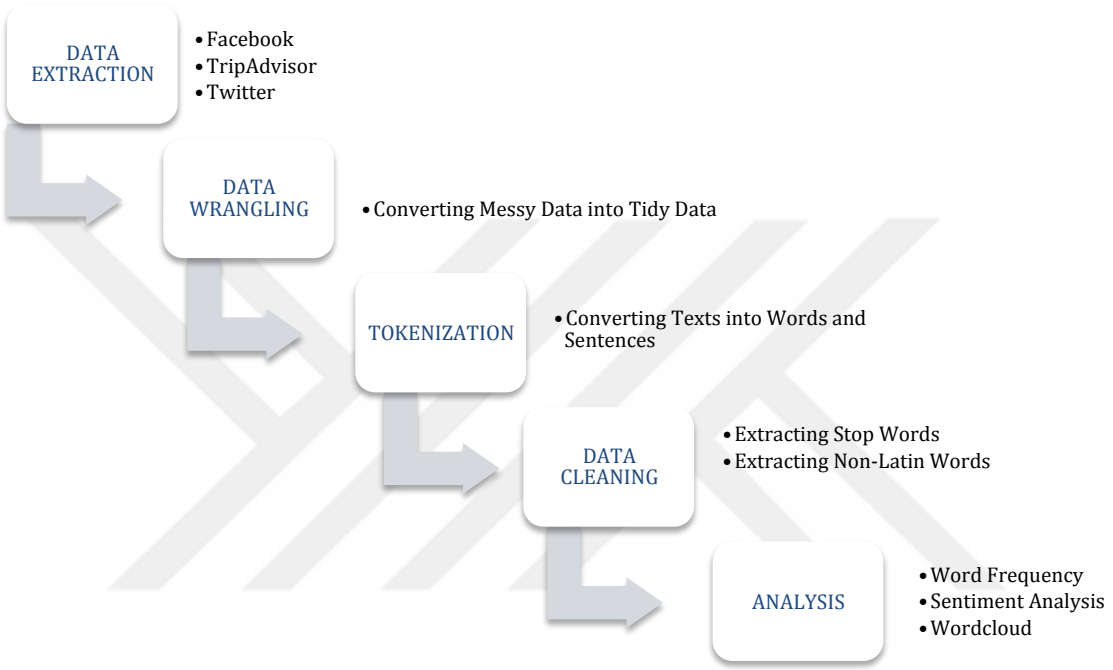


Figure 1: Methodology Flowchart

3.1 Data Extraction

The first and foremost step in social media mining is data extraction from social media web applications. Social media web applications are the websites that people use to interact with other people and to build social relationships or social networks with other people who share similar career or personal interests, experiences, mutual connections and activities.

The data is extracted using a technique called "Web Scraping" (also termed as Screen Scraping and Web Harvesting). Using this technique, a large amount of data is extracted from multiple pages of websites that can easily be saved into a local file in a computer or a database in a table format (WebHarvy 2017).

Among many other web scraping software, the study used statistical software environment R and R programming language to extract data from social media web applications, like Facebook, TripAdvisor and Twitter. Three different R packages are employed to extract data from above mentioned three different social media websites. R packages are the fundamental units of reproducible R code. These packages include reusable R functions, sample data and the documentation that explains how to use the built-in functions (Wickham, R Packages: Organize, Test, Document, and Share Your Code 2015).

R Package `Rfacebook` provides an interphase to the Facebook API through which Facebook data is extracted and stored as a `dataframe` or `lists` in R software environment (Barbera, et al. 2017). Before extracting data from Facebook, an App is created on the Facebook platform by logging into www.developers.facebook.com. The App is used to connect to the Facebook API. The Facebook App has its own App ID and App Secret that is used to connect to the R session. `Rfacebook` provides an easy function, `fbOAuth()`, to build a connection. The function requires two parameters, i.e. `fb0Auth(app_id, app_secret)`. After the authentication of the connection between Facebook and R session is successful, the reusable `auth_fb` object is saved for next time use. The `auth_fb` object contains the app details, like ID and secret, and authentication details, which are the only requirements when the connection is made next time.

```
library(Rfacebook)
# Fb Authorization
auth_fb <- Rfacebook::fbOAuth(
  app_id="1380496555352781",
  app_secret="eb3abc42d1e00536e6f4e37e58fc0b5d",
  extended_permissions = TRUE)
# Saving variable auth_fb in a file and loading it
save(auth_fb, file="auth_fb")
load("auth_fb")
```

Subsequently, Rfacebook function, `getPage()` is used to extract the data related to posts, such as Facebook post content, number of likes, shares and comments. The function requires three parameters, `page`, `token` and `n`. The page refers to the pageID

```
# Extract posts from turkish airlines page
turkishairlines <- Rfacebook::getPage(page = "turkishairlines",
                                      token = auth_fb, n = 100)
```

Furthermore, the function `getPost()` is used to extract the user comments and their reactions along with their display names. Earlier, Facebook allows all Apps to extract all kind of data from public pages, however; due to recent changes in Facebook policies, the App who has been given access to the public page by page's admin can extract posts and comments.

Another R package, `rvest`, helps to extract data from html webpages. The study used the package to scrape the data from TripAdvisor web application. TripAdvisor is a travel and restaurant website company providing hotel, airlines and restaurant reviews and other travel specific content. The author of `rvest`, Hadley Wickham, provides complete code to scrape review data and reviews' ids from TripAdvisor (Wickham, hadley/rvest 2015). The study mainly used Hadley's code to extract data from TripAdvisor. As the reviews are posted on more than one webpage, `for loop` was added to the code to generate multiple URLs and to extract data from multiple pages at a time. The only input required for data extraction is the URL of the desired webpage. Three main functions of `rvest` that are employed during the extraction are `readhtml()`, `html_node()` and `html_attr()` (Wickham, rvest: Easily Harvest (Scrape) Web Pages 2016).

```
library("rvest")
df_total = data.frame()
for (i in seq(0, 6940, 10))
{
  if (i == 0) {
    url <- "https://www.tripadvisor.com/Airline_Review-d8729174-Review
```

```
s-Turkish-Airlines"
  }
  else  {
    url <- paste(
      "https://www.tripadvisor.com/Airline_Review-d8729069-Reviews-or"
,i,"-Turkish-Airlines#REVIEWS",
      sep = "")
  }
  reviews <- url %>%
    read_html() %>%
    html_nodes("#REVIEWS .innerBubble")
  id <- reviews %>%
    html_node(".quote a") %>%
    html_attr("id")
  quote <- reviews %>%
    html_node(".quote span") %>%
    html_text()
  rating <- reviews %>%
    html_node(".rating .rating_s_fill") %>%
    html_attr("alt") %>%
    gsub(" of 5 stars", "", .) %>%
    as.integer()
  date <- reviews %>%
    html_node(".rating .ratingDate") %>%
    html_attr("title") %>%
    strptime("%b %d, %Y") %>%
    as.POSIXct()
  review <- reviews %>%
    html_node(".entry .partial_entry") %>%
    html_text()
  df <- data.frame(id, quote, rating, date, review, stringsAsFactors =
 FALSE)
  df_total <- rbind(df_total, df)
}
# Save an object to a file
saveRDS(df_total, file = "tripadvisor_turkishairlines6846.rds")
```

R package `twitteR` is used to scrape the tweets, retweets and favorites of any Twitter user. In similar to Facebook web scraping, a twitter App is created after logging into www.apps.twitter.com. The function of `twitteR` package, `setup_twitter_oauth()`, with four parameters `consumer_key`, `consumer_secret`, `access_token`, and `access_secret`, builds connection between R session and twitter App. The values of the above mentioned parameters are obtained after the twitter App is created. Subsequently, `userTimeline()` and `retweets()` functions are employed to scrape the tweets and retweets of any specific tweet respectively (Gentry 2015).

```
# Load Requried Package
library("twitteR")
# Authonitical keys
consumer_key <- 'tAyR9LyhATfD90aA7Ft1Zfj3I'
consumer_secret <- 'vX1RHqHHDpnmNOqrGPMVnmnQjQvG98X3xlB7T7zv4hKcvj7tVv
access_token <- '2572842085-vExbB4HNvN57zmQhoQdbmutC16a4kdMdh1xVta5'
access_secret <- 'HtTHSeAOz1WPcUX8nfW5ddwZ1TbXZGFB4pSHU0IZ3agvA'

twitteR::setup_twitter_oauth(consumer_key, consumer_secret,
                             access_token, access_secret)
tweets <- userTimeline("turkishairlines", n=200)
```

3.2 Data Wrangling

The extracted social media data is in raw format and huge amount of effort is required to clean before getting it ready for analysis. Through effective data wrangling, the data is converted from raw format to another format that is easy to clean, manipulate, model and visualize. Hadley Wickham (2014) in his paper “Tidy Data” provides an effective way of data wrangling data through which messy datasets are converted into tidy datasets by using only small set of tools. The tidy datasets have a specific data structure: each variable is a column, each observation is a row, and each type of observational unit is a table (Wickham, Tidy Data 2014). Tidy data sets are obtained and manipulated

through a set of tidy tools, including famous packages `dplyr` (Wickham et al. 2017), `ggplot2` (Wickham, ggplot2: Elegant Graphics for Data Analysis 2009), `tidyr` (Wickham and Henry, tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions 2018), and `broom` (Robinson 2018). Hence, the study converts the raw social media data into tidy data so that transition between the packages for manipulation, interpretation, and visualization purposes becomes smooth.

3.3 Tokenization

The tokenization is a process of converting a text into meaning unit of text, such as a word, n words (or n-gram), sentence or paragraph. The social media data is in the raw text format. In order to convert the untidy text data into tidy data, tokenization process is employed that split text into tokens. The table with a tidy text structure contains a one token in each row. Usually, the token in tidy dataset is a single word, but it can be an n-gram (collection of words), sentence, or paragraph.

The R package `tidytext` (Silge and Robinson, tidytext: Text Mining and Analysis Using Tidy Data Principles in R 2016) is used to tokenize the social media texts, like posts, comments, reviews and tweets, and convert them into one-token-each-row format. The study used tidytext's `unnest_token()` function. The function requires two basic functions, one is the output column that is created when the text is tokenized and other is the input column that the input text for tokenization. An additional benefit of `unnest_tokens()` is that it converts the tokens to lowercase, which makes them more comparable with other datasets.

```
tokenize <- function(file, data_type) {
  data_tibble <- readRDS(file = file) %>%
    tibble::as_tibble()
  data_vector <- data_tibble %>%
    dplyr::pull(data_type) %>%
    iconv(from = "UTF-8", to = "Latin1")
```

```
  tokens <- tibble::as_tibble(data_vector) %>%
    dplyr::filter(!is.na(value)) %>%
    dplyr::mutate(response_number = rownames(.)) %>%
    dplyr::select(response_number, value) %>%
    tidytext::unnest_tokens(word, value)
  tokens
}
```

The `tokenize` function converts text into tokens into following steps: first, the data is stored into a dataframe. Second, the untokenized text data is saved into a vector with `character` format, where each element of a vector is a sentence or paragraph. Third, the encodings of the character vector is converted to `Latin1`. Fourth, the vector is again transformed into a dataframe and `NA` values are filtered out. Fifth, the numbers are assigned to each sentence or paragraph using:

```
mutate(line = rownames(.))
```

Subsequently, the desired column in tokenized into words using `unnest_tokens` function. The `value` column is the input column argument and `word` is the output column argument of `unnest_tokens`. The column is named `value` automatically in R when the vector is converted into a dataframe. These steps are shows in Figure 2 as a flowchart.

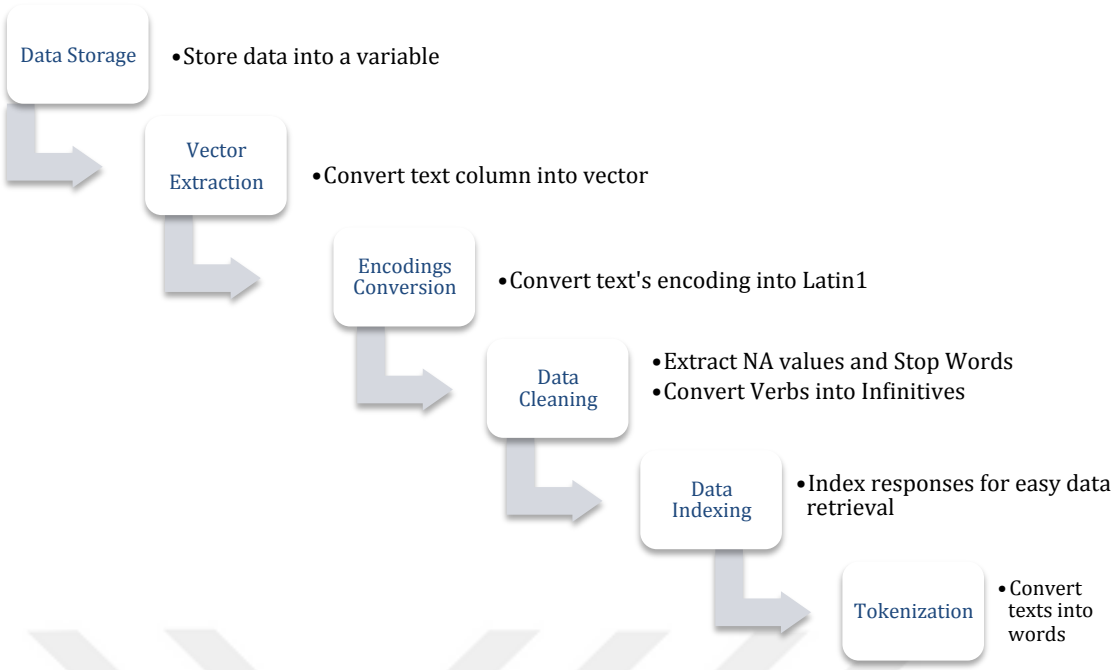


Figure 2: Tokenization Process Flowchart

3.3.1 N-gram Tokenization

The `unnest_token()` function provides an option to specify the type of token, i.e. by word, or n-gram – succeeding sequence of words. Sometimes, tokenization by n-gram, especially bigram (two words) is useful for sentiment and frequency analysis. For instance, when examining pairs of two succeeding words, an additional parameter, `token = ngrams` and n = 2, is added to `unnest_tokens()`, as it examines two consecutive words, also known as `bigrams` (Jurafsky and Martin 2017). Words, such as "not good", "not bad", and "along with" are the examples of bigrams as they are usually used consecutively in a text document.

3.3.2 Sentence Tokens

Until now, the study has used unigram or word tokens for frequency visualization and sentiment analysis. However, in addition to word tokens, there is another form of token, i.e. sentence tokens (Silge and Robinson, Text Mining with R - A Tidy Approach 2018). The sentence token in a tidy data set considers a sentence in a text as a token. In the function `tidytext::unnest_tokens()`, there is an argument `token` which requires

either a "words" or "sentences" as an input. In order to use "sentences" as a token, the data must be have a tidy format and the character vector must be encoded in latin1 encoding. Following are the steps in the construction of sentence_tokens function as given below. First, the review or comment data is tokenized into words. Second, using dplyr::filter(), the words in English dictionary are selected or filtered out. The English dictionary is obtained from the package qdapDictionaries. Third, with the help of comments and reviews id, the tokens are combined into sentences, using paste function, and saved into column named sentence.

```
paste(word, collapse = " ")
```

Fourth, the encodings of the characters inside the sentence column is converted from UTF8 to latin1. Lastly, after grouping the dataframe by id, the sentence column is tokenized into sentences.

```
sentence_tokens <- function(data = untidy_response_facebook,
                            response_column = "comments",
                            group_by = "comment_number") {
  # English Dictionary
  qdapDictionaries::DICTIONARY[,1]

  en_word_comments <- data %>%
    dplyr::ungroup() %>%
    tidytext::unnest_tokens_("word", response_column) %>%
    dplyr::filter(word %in% qdapDictionaries::DICTIONARY[,1])

  en_word_sentence_comments <- en_word_comments %>%
    dplyr::group_by_("id", group_by) %>%
    dplyr::mutate(sentence = paste(word, collapse = " ")) %>%
    dplyr::distinct(sentence, .keep_all = TRUE) %>%
    dplyr::as_data_frame() %>%
    dplyr::mutate(sentence = iconv(sentence, to = 'latin1')) %>%
    dplyr::ungroup()
```

```
  # Sentence as tokens with post number and comment number
  en_word_sentence_comments %>%
    dplyr::select_("id", group_by, "sentence") %>%
    dplyr::ungroup() %>%
    tidytext::unnest_tokens(sentences, sentence, token="sentences")
}
```

3.4 Stop Words

The tokens in the tidy data structure often contains frequent but less informative words. These words do not add value in the analytics therefore they are removed from the data and thus known as stop words (Jurafsky and Martin 2017). For example, “I”, “he”, “they’ll” are some of the English stop words. The study used a list of stop words from the “tm” package that contains 174 stop words. Furthermore, new words can be added to the stop words lists using `c()` function. For instance, the study added “miss”, “airlines”, “flight”, and “turkish” to the stop words by writing this code: `c(“miss”, “airlines”, “flight”, “turkish”, stopwords(“en”))`. These stop words are eliminated from the tidytext dataset with a dplyr function, i.e. `anti_join()`.

```
data(stop_words)

# Custom stop words
custom_stop_words <- data.frame(word = c("miss", "flight", "tukish",
                                         "airlines", "flights",
                                         "airline", "turkish"),
                                lexicon = c("custom")) %>%
  rbind(stop_words)
```

3.5 Multiple Verb Forms

The tokens include multiple forms of verbs that are considered different words during analysis. The study aims to convert verbs in different forms, including present simple, past simple, present participle and past participle into base form of verbs. According to the research, there is not any specific package in R that deals with the issue of

converting verbs from different forms into basic forms. The study develops its own lexicon of verbs with its other forms of verbs. The lexicon is named as `sahban_base_lexicon` and saved along with other datasets in the `.rda` format. The lexicon contains a total of 2,154 different verbs. The lexicon data have two columns, `base` column, refers to the base form of verbs, and `non_base` column, refers to all forms of verbs other than base form. Furthermore, the study provides a function `extract_non_base` that converts the other verb forms tokens into base form verbs. The function takes `data` as an argument. The dataset is the output of the function `unnest_tokens` that contains a column named `word`.

```
extract_non_base <- function(data) {
  data %>%
    dplyr::rename(non_base = word) %>%
    dplyr::left_join(readRDS("sahban_base_lexicon"), by = "non_base")
%>%
    dplyr::mutate(base = ifelse(is.na(base), non_base, base)) %>%
    dplyr::rename(word = base) %>%
    dplyr::select(-one_of("non_base"))
}
```

The study also provides a way to augment `sahban_base_lexicon` by adding new verbs and their forms. The new verbs are added using the function `dplyr::bind_rows`. As seen in an example below, the verb travel is added to the list with its past simple and present participle. Although, the verb was included in the list, many users have used double "l" in the word travelled and travelling instead of single "l". In order to increase accuracy of analysis, the study included misspelled verbs in the lexicon.

```
# Adding words manually to the lexicon
sahban_base_lexicon <- readRDS("sahban_base_lexicon") %>%
  dplyr::bind_rows(data.frame(base = c("travel", "travel"),
                              non_base = c("travelled", "travelling")))
saveRDS(sahban_base_lexicon, "sahban_base_lexicon")
```

3.6 Plural Nouns

The tokens include plural nouns that are considered different words than their respective singular nouns during analysis. The study aims to convert plural nouns into singular form of nouns. According to the research, there is not any specific package in R that deals with the issue of converting nouns from plural to singular form. The study develops its own lexicon of nouns with its plural forms. The lexicon is named as `sahban_noun_lexicon` and saved along with other datasets in the `.rda` format. The lexicon contains a total of 4,489 different nouns. The lexicon data have two columns, `noun` column, refers to the singular form of nouns, and `plural` column, which refers to plural form of nouns. Furthermore, the study provides a function `extract_plural` that converts the plural noun tokens into singular form. Similar to the `extract_non_base` function, the function takes `data` as an argument. The dataset is the output of the function `unnest_tokens` that contains a column named `word`.

```
# Convert Plural Nouns to Singular Nouns

extract_plural <- function(data) {
  data %>%
    dplyr::rename(plural = word) %>%
    dplyr::left_join(readRDS("sahban_noun_lexicon"), by = "plural") %>
%
    dplyr::mutate(noun = ifelse(is.na(noun), plural, noun)) %>%
    dplyr::rename(word = noun) %>%
    dplyr::select(-one_of("plural"))
}
```

3.7 Word Frequency

The most basic and common task in social media mining is to find word frequencies. Although computing word frequencies is a simple analytical technique, reasonable and intuitive word frequencies can lead to deep insights and logical findings from the data, especially while comparing word frequencies among different texts. The study

employed tidy tools to compute frequencies smoothly and intuitively. In this study, the `dplyr` function `count()` is used to find the frequency of each token in the text. Afterwards, highly frequent words, having frequency of more than specific limit, and their frequencies are selected from the tidy datasets.

```
word_count <- function(data) {
  readRDS(data) %>%
    dplyr::anti_join(custom_stop_words, by = "word") %>%
    dplyr::count(word, sort = TRUE) %>%
    tibble::as_tibble()
}
```

These frequent words and their frequencies have been plotted using a package `ggplot2()` and its function `ggplot()`. In addition to the data as an input, minimum number of words to be included in a bar chart plot is provided as an input to a newly built function `word_count_plot`. The function is based on the code provided by Julia Silge and David Robinson in their book "Text Mining with R" (Silge and Robinson, Text Mining with R - A Tidy Approach 2018).

```
word_count_plot <- function(data, min_count) {
  tokens_count <- readRDS(data)
  tokens_count %>%
    dplyr::filter(n > min_count) %>%
    dplyr::mutate(word = reorder(word, n)) %>%
    ggplot2::ggplot(mapping = ggplot2::aes(word, n)) +
    ggplot2::geom_col() +
    ggplot2::xlab("Most Frequent Words") +
    ggplot2::coord_flip()
}
```

3.8 Sentiment Analysis

In the previous sections, the study explained the method of tidying the raw data and explained how can tidy tools be used to tokenize the texts. This tokenization helped to compute and compare word frequencies. In this section, the study aims to further

analyze the tokens, extracted from texts, by using sentiment analysis or opinion mining technique. When humans read a text, they infer whether a token of text is positive, negative or characterized by some other emotion like anger or joy. The study employed text-mining tools to find out the emotions in the text programmatically.

The study considers text as a combination of its individual tokens or words and the overall sentiment score of the text is basically the sum of the sentiment score of individual tokens. This sentiment analysis approach is easy to implement while using tidy data principles as tidy datasets has one token in each row.

In R programming, in contrast to the method of removing stop words where `anti_join()` function is used, the sentiment scores of unigrams in a tidy dataset are evaluated using dplyr's function `inner_join()`. After the tokenization of the texts using `unnest_tokens()`, in order to keep track of which posts and comment of the posts each token comes from, the study used `group_by` and `mutate` functions in dplyr package.

As mentioned above, the `unnest_tokens()` requires to parameters, input column and output column. The study chose the name word for the output column as the lexicon datasets and stop words datasets have columns with the name `word`; thus, making inner joins and anti joins simpler.

The next step after removing stop words and scoring the tokens, the study counts how many negative and positive tokens are there in each comment, review or tweet. The study then uses `dplyr::spread()` to have positive and negative sentiment scores in separate columns. Finally, the net sentiment score is calculated by subtracting negative score from the positive for each comment, review and tweet. The overall sentiment score of a review or a comment ease the way of finding the undesirable comments and negative reviews.

3.8.1 Sentiment Lexicons

There are a variety of datasets or sentiment lexicons that specify the sentiment content of tokens in a text. The study used three popular general-purpose sentiment lexicons to evaluate the emotion in text, i.e. AFINN, Opinion Lexicon and Emolex. All of these lexicons are based on single word tokens, i.e. unigrams.

AFINN

AFINN is one of the three sentiment datasets used in this study. AFINN is a list of English words valued for opinions or emotions with an integer between minus five (-5) to plus five (+5), where negative score indicates negative emotions and positive score indicates positive emotions. Finn Årup Nielsen in 2009-2011 manually labeled the list of 2477 words and phrases (Nielsen 2011).

A general function `sentiments_afinn` is made that takes tidy data as an argument and provides a sentiment score of a post, a sentence or a paragraph. First, the function uses `inner_join()` function to assign a score to each word that is in the data and that also exits in the afinn lexicon. After grouping by `post_number`, the score of each word in a post is added to get a total sentiment of a post.

```
#afinn
sentiments_afinn <- function(data) {
  data %>%
    dplyr::inner_join(tidytext::get_sentiments("afinn"), by = "word")
%>%
    dplyr::group_by(id, response_number) %>%
    dplyr::summarise(score = sum(score))
}
```

Opinion Lexicon

Another general-purpose sentiment lexicon used in this research study is “Opinion Lexicon”, developed by Bing Liu and collaborators (Hu and Liu 2004). The lexicon is a

list English positive or negative opinions or sentiment words. The total number of words in this opinion lexicon is six thousand eight hundred words. Bing's lexicon is obtained from the R package `tidytext` where it is named as `bing`. It categorizes the tokens in a binary fashion, either positive or negative category (Hu and Liu 2004).

Emolex

NRC Word Emotion Association Lexicon, also known as "Emolex", is the third emotion lexicon used in this study (Mohammad and Turney 2013). The lexicon is the list of English words and their association with two sentiments, positive and negative, and eight main emotions, i.e. surprise, anger, sadness, fear, trust, anticipation, joy, and disgust. In similar to other two above-mentioned lexicons, Emolex is also based on single words, i.e. unigrams. The lexicon was constructed through crowdsourcing on Amazon Mechanical Turk, a marketplace for work where developers hire humans for the tasks requiring human intelligence (Mohammad and Turney 2013).

These lexicons are accessed through tidytext by using its function `get_sentimens()`. The names of these AFINN, Opinion Lexicon and Emolex in tidytext sentiment datasets are `AFINN`, `bing` and `nrc` respectively. The lexicons are validated either through crowdsourcing or through social media data, such as Twitter data or restaurant reviews. They assign a score to each word in a text and subsequently the scores of are added up to find the sentiment score of a whole text. As most of the English words are neutral, the lexicons contain only those words that indicate some opinion or emotion.

The function, `response_sentiments`, is valid for two sentiment lexicons, Opinion Lexicon (bing) and Emlox (NRC). As these two lexicon treats positive and negative sentiments separately, the function is able to subtract negative sentiment score from positive sentiment score to give an overall score to a sentence or paragraph. The body of the function `response_sentiments` is given below:

```
response_sentiments <- function(data, lexicon, group_by = sentiment) {
  data %>%
    dplyr::inner_join(get_sentiments(lexicon), by = "word") %>%
    dplyr::count(response_number, sentiment) %>%
    tidyr::spread(sentiment, n, fill = 0) %>%
    dplyr::mutate(sentiment = positive - negative) %>%
    dplyr::ungroup()
}
```

3.8.2 Most Frequent Sentiments

Using the above mentioned three lexicons, tokens are classified into sentiments. The next step of analysis is to find out the most frequent sentiments in a document. The most common sentiment helps businesses to have an insight about overall sentiment of consumers. For instance, if positive sentiment count outweighs the negative sentiment count, it can be supposed that the consumers have overall positive attitude toward the company. A simple R function to estimate most frequent sentiments in a tidy dataset is given below:

```
frequent_sentiments <- function(data) {
  data %>%
    dplyr::inner_join(tidytext::get_sentiments("bing")) %>%
    dplyr::ungroup() %>%
    dplyr::count(word, sentiment, sort = TRUE)
}
```

3.8.3 Plot Frequent Sentiments Counts

In subsequent to the estimation of frequent sentiments in a text, the study plots the frequency of sentiments using flipped bar charts. A function `plot_sentiment_count` is made that takes a tidy dataset as its only input and gives two bar charts with most frequent positive and negative sentiments respectively. The function is based on the code provided in the book Text Mining with R by Julia Silge and David Robinson (Silge

and Robinson, tidytext: Text Mining and Analysis Using Tidy Data Principles in R 2016).

```
plot_sentiment_count <- function(data) {
  data %>%
    dplyr::group_by(sentiment) %>%
    dplyr::top_n(10) %>%
    dplyr::ungroup() %>%
    dplyr::mutate(word = reorder(word, n)) %>%
    ggplot2::ggplot(ggplot2::aes(word, n, fill = sentiment)) +
    ggplot2::geom_col(show.legend = FALSE) +
    ggplot2::facet_wrap(~sentiment, scales = "free_y") +
    ggplot2::labs(y = "Sentiments",
                  x = "Frequency”) +
    ggplot2::coord_flip()
}
```

3.8.4 Sentiments to All Words Ratio

Although the total number of positive or negative sentiments depicts the overall sentiment content in a sentence or a paragraph, the numerical value is lacking; since, the total number of words in the sentence (or paragraph) is not taken into account. For instance, while comparing the negative sentiment content of two different sentences, only the total number of negative words in a sentence would not give us a complete picture, instead a ratio of negative word count to total word count provide a better comparison. The study provides an R function that takes a tokenized tidy dataset and `sentiment_type`, positive or negative, as an input and returns a dataframe with top 10 most negative or positive user generated responses.

```r
sentiment_token_ratio <- function(data, sentiment_type = "negative") {
  negative_sentiment <- get_sentiments("bing") %>%
    dplyr::filter(sentiment == sentiment_type)

  wordcounts <- data %>%
    dplyr::group_by(response_number) %>%
    dplyr::summarize(word = n())

  data %>%
    dplyr::semi_join(negative_sentiment) %>%
    dplyr::group_by(id, response_number) %>%
    dplyr::summarize(negativewords = n()) %>%
    dplyr::left_join(wordcounts, by = c("response_number")) %>%
    dplyr::mutate(ratio = negativewords/word) %>%
    dplyr::top_n(10) %>%
    dplyr::ungroup() %>%
    dplyr::arrange(desc(ratio))
}
```

3.9 Word Cloud

WordCloud is a visualization technique that gives word cloud, an image made-up of words used in a document or text. In a word cloud, the size of each word specifies the importance and frequency of the word. Word cloud of emotions or opinions provides insights about user perception about the target subject, such as a brand or a product.

In R programming, the `wordcloud` package is used to build a word cloud that indicates the most frequent sentiments in the comments and reviews. The `word_cloud` function, as given below, requires two arguments: first, a dataset with a column containing tokens and another column with tokens' frequency; second, a numerical value indicating the maximum number of words to be visualized in a word cloud.

```
library(wordcloud)

word_cloud <- function(data, max_words) {
  data %>%
    dplyr::count(word) %>%
    with(wordcloud::wordcloud(word, n, max.words = 50))
}
```

3.9.1 Sentiment Cloud

Furthermore, an interesting wordcloud can be obtained after labeling the tokens into positive and negative sentiments. First, sentiment analysis is done and tokens are labeled as positive or negative by using inner_join(). Afterwards, the data structure of tidy dataset is converted from dataframe to matrix using acast() function of reshape2 package. Finally, comparison.cloud() function is used to get a word cloud that compares the most frequent positive and negative sentiments. The visualization helps us to figure out the most important negative and positive sentiments in a text, however, the size of the words cannot be compared across sentiments. A general `sentiment_cloud` function is made to avoid code redundancy. The function gives a cloud of sentiments when a tidy data, having a tokenized column with a name `word`, is passed through its `dataset` argument. The function is based on the code provided by Julia Silge and David Robinson in their book "Text Mining with R" (Silge and Robinson, Text Mining with R - A Tidy Approach 2018).

```
library(reshape2)

sentiment_cloud <- function(dataset) {
  dataset %>%
    dplyr::inner_join(tidytext::get_sentiments("bing")) %>%
    dplyr::count(word, sentiment, sort = TRUE) %>%
    reshape2::acast(word ~ sentiment, value.var = "n", fill = 0) %>%
    wordcloud::comparison.cloud(colors = c("#F8766D", "#00BFC4"),
                                max.words = 100)
}
```

4. DATA

The study aims to conduct social media mining of Turkish international company that use English language as a medium of communication on Social Media web applications. In the research, the study finds out that Turkish Airlines "Türk Hava Yolları" is one of the most famous Turkish brands worldwide. The company is actively manages its social media accounts and interacts with its fans and followers on social media websites in English language. Furthermore, it has large number of followers in all over the world and thousands of people who have travelled through Turkish Airlines have posted their reviews on social media websites.

The Turkish Airlines social media branding campaign is highly effective on Facebook. Its fan page has more than 10 million likes and followers. For research purposes, the study scraped data from company's Facebook page as mentioned in the methodology section. Firstly, Facebook posts and total number of comments, likes and shares are extracted for six months, i.e. from 15 March 2017 to 15 October 2017. The data is stored in a dataframe, which is named as "posts data". Furthermore, a post-specific ID is used to scrape content of each comment in every Facebook post. The comment dataset has a complicated "list" data structure. The list of comment dataset contains three hundred (300) lists for each post. Each post-specific list contains three lists, each containing a dataframe that provides details about posts, comments and reactions respectively. The list structure is converted into a dataframe that contains the content of three hundred (300) posts and their comments. The study named this dataframe as "comments data". Although the conversion from list to dataframe resulted in post-specific data redundancy, it eases our way in data cleaning and tokenization through tidy tools. The comment dataset contains six thousand five hundred and sixty four rows (6.564), which means on average there are 22 comments on each Facebook post in last six months. After tokenization, the number of rows in a dataset is extended to more than thirty two

thousand rows (32,000). There are total thirty two thousand one hundred and fifty seven (32,157) words in six thousand five hundred and sixty four (6,564) comments of total 300 posts.

The Facebook data were retrieved in the month of October 2017 and the permission from the admin of Turkish Airlines' page was not required. However, after Facebook took initiatives to reform its privacy policy on 28th March 2018, the data extraction from Facebook page requires access to the page's data from the admin (Jenkins 2018).

In addition to the Facebook comment data, the study also scrapes reviews from another social media web application, TripAdvisor. There are more than six thousand eight hundred reviews of Turkish airlines on TripAdvisor. As TripAdvisor is a review based website, the website provides a user-friendly interface with tips for writing a great review. Therefore, most of the genuine customers' reviews are written in proper English language. The study aims to take advantage of these well-written reviews and explore the customers' perception about brands through sentiment analysis. The study extracted reviews of last two and a half years, i.e. from January 2016 to April 2018. There are total reviews of 6,846 extracted from 686 pages. The review dataset includes id for each review, quotes and reviews.

5. Case Study – Turkish Airlines

This research thesis conducts a case study on Turkish Airlines social media data posted by its followers or customers in different social media forums. As explained under the data extraction section in methodology, the data is extracted from the two main web applications, Facebook and TripAdvisor. In this section, the study explains the structure of data extracted and the use of R packages and function that are required to clean the messy data and make it useful for analysis. Furthermore, the after converting the data in an appropriate tidy format, the sentiment analysis is conducted along with sentiment word clouds for the visualization of customers' perception about Turkish Airlines.

5.1 Extracting Facebook Posts

The Facebook ID of the Turkish Airlines' page is "turkishairlines". Using the page id, Facebook authentication, and `getPage()` function, the data of recent 300 posts is extracted.

```
# Extract posts from turkish airlines page
data_turkishairlines <- Rfacebook::getPage(page = "turkishairlines",
                                           token = auth_fb, n = 300)
```

The scraped data has multiple columns, such as `id`, `from_id`, `from_name`, `message`, `type`, `likes_count`, `comments_count`, and `shares_count` etc. These data is viewed in the form of `tibble`, a data structure type that prints data in r console in an easily readable format. As seen in the table below, the `from_id` column is the id of Turkish Airlines pages, the type column denotes the type of the posts, either photo or video. Furthermore, the `message` column specifies the message posted by Turkish Airliness along with the photo or video. Each row in the table below provides information about one specific Facebook post.

```
# Viewing limited variables and rows
data_turkishairlines %>%
```

```
  dplyr::select(from_id, likes_count, type, comments_count,
                shares_count, message, id) %>%
  tibble::as_tibble() %>%
  head(5)
```

After saving the data into `data_turkishairlines` variable, the data can be reused for printing. The function `dplyr::select()` is used to select and sequence selected columns of a tidy dataframe. Using `head()` function, only first five rows are printed as shown below:

```
## # A tibble: 5 x 7
##       from_id likes_count  type comments_count shares_count
##         <chr>       <dbl> <chr>          <dbl>        <dbl>
## 1 90430042759         254 photo             12           18
## 2 90430042759         300 photo            132           92
## 3 90430042759         869 photo             39           43
## 4 90430042759         261 photo              9            7
## 5 90430042759         227 photo             13           11
## # ... with 2 more variables: message <chr>, id <chr>
```

As the space is limited for all of the columns to be printed, the `tibble()` function doesn't show all of the columns' data. Instead, the name of the columns and their data type is printed at the end, like `message` and `id`. Furthermore, user can view a specific column's content using `dplyr::select()` as shown in the code below:

```
# Viewing one recent post's message

data_turkishairlines %>%
  dplyr::select(message) %>%
  tibble::as_tibble() %>%
  head(2)
```

The code gives first two values of a message column, corresponds to the latest two posts on Turkish Airlines Facebook page.

```
## # A tibble: 2 x 1
##    message
```

```
##     <chr>
## 1  Our flights to Samarkand start on March 16th, 2018!
## 2 "We are looking for new pilots! Join us at Turkish Airlines
##    Pilot Roadshow
```

5.2 Extracting Facebook Comments

In converse to the Facebook post extraction, the comment data extraction is quite complicated. The `Rfacebook::getPost()` is used to extract the comments for specified posts. As comment extraction requires the related posts id, this process becomes quite complex. As shown below, the `getPost` function requires post's id, number of comments, and Facebook authentication.

```
get_comments <- getPost(post = post_id, n = 50000, token=auth_fb,
                        comments = TRUE, reactions = TRUE)
```

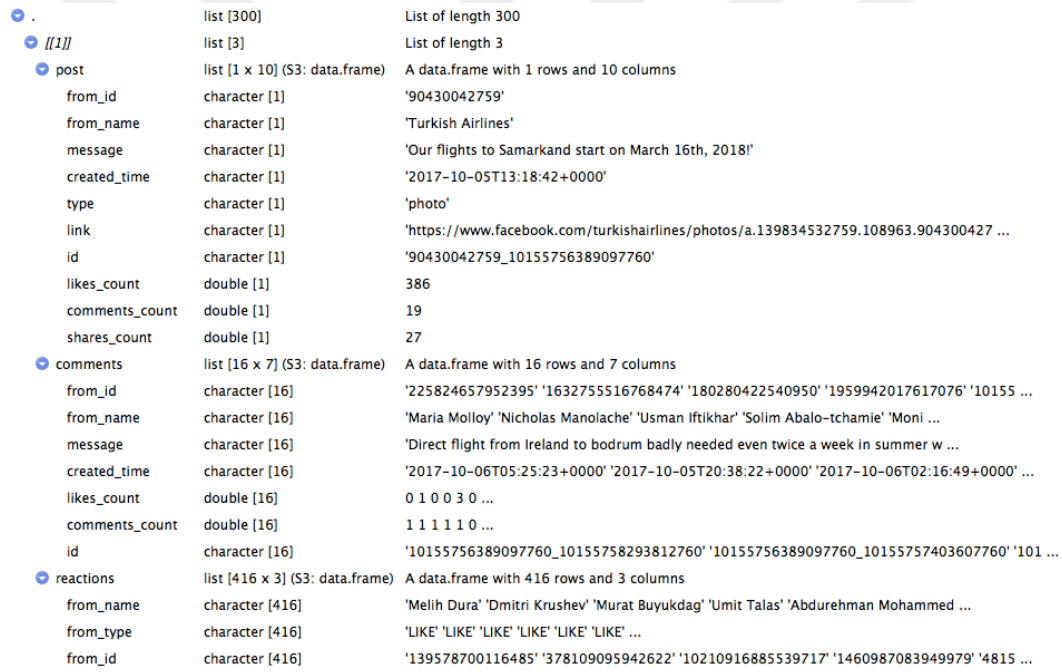
The `getPost()` only extracts comments of one post at a time, however, the study requires comments of 300 different posts in one go. In order to scrape all of the comments, the study used the function `lapply()` that runs the same code three hundred times with 300 different post ids. The post ids are store in a separate R vector using `$` sign as shown. The `$` sign is used to index the content in the `id` column that exists in `data_turkishairlines` variable.

```
data_turkishairlines_id <- data_turkishairlines$id
```

The vector is then passed as the first argument of `lapply` function and following the `getPost` function name and the values of all the arguments of `getPost` function except first. The `lapply` function is applying the vector of ids to the `getPost` function in a loop.

```
threehundred_posts <- data_turkishairlines_id[1:300] %>%
                  lapply(getPost, n = 50000, token=auth_fb,
                  comments = TRUE, reactions = TRUE)
```

All the comments data of 300 posts is stored inside a list which contains 300 sub lists. Inside each sub list, there are three sub-sub-lists containing data related to posts, comments and reactions in a dataframe respectively. The data structure of the lists is shown in the image given below. The complex list structure contains numerous kinds of data that is out of the scope for this research study, like names of the people who shared the posts and reacted to the posts in anyway. In this study, the comments posted by the followers are only required for this study. It can be seen in the image below, the comments content is saved inside a `message` column under the list with the name `comments`.



.	list [300]	List of length 300
[[1]]	list [3]	List of length 3
post	list [1 x 10] (S3: data.frame)	A data.frame with 1 rows and 10 columns
from_id	character [1]	'90430042759'
from_name	character [1]	'Turkish Airlines'
message	character [1]	'Our flights to Samarkand start on March 16th, 2018!'
created_time	character [1]	'2017-10-05T13:18:42+0000'
type	character [1]	'photo'
link	character [1]	'https://www.facebook.com/turkishairlines/photos/a.139834532759.108963.904300427 ...
id	character [1]	'90430042759_10155756389097760'
likes_count	double [1]	386
comments_count	double [1]	19
shares_count	double [1]	27
comments	list [16 x 7] (S3: data.frame)	A data.frame with 16 rows and 7 columns
from_id	character [16]	'225824657952395' '1632755516768474' '180280422540950' '1959942017617076' '10155 ...
from_name	character [16]	'Maria Molloy' 'Nicholas Manolache' 'Usman Iftikhar' 'Solim Abalo-tchamie' 'Moni ...
message	character [16]	'Direct flight from Ireland to bodrum badly needed even twice a week in summer w ...
created_time	character [16]	'2017-10-06T05:25:23+0000' '2017-10-05T20:38:22+0000' '2017-10-06T02:16:49+0000' ...
likes_count	double [16]	0 1 0 0 3 0 ...
comments_count	double [16]	1 1 1 1 1 0 ...
id	character [16]	'10155756389097760_10155758293812760' '10155756389097760_10155757403607760' '101 ...
reactions	list [416 x 3] (S3: data.frame)	A data.frame with 416 rows and 3 columns
from_name	character [416]	'Melih Dura' 'Dmitri Krushev' 'Murat Buyukdag' 'Umit Talas' 'Abdurehman Mohammed ...
from_type	character [416]	'LIKE' 'LIKE' 'LIKE' 'LIKE' 'LIKE' 'LIKE' ...
from_id	character [416]	'139578700116485' '378109095942622' '10210916885539717' '1460987083949979' '4815 ...

Figure 3: Data Structure of Facebook Comments

The comments are extracted from the list through three level indexing. First, each post-specific list is indexed; second, comments list is indexed using `$` sign; last, the `message` column is indexed as given in the code.

```
# Saving comments for first post in new variable
full_comments <- threehundred_posts[[1]]$comments$message %>%
  as.data.frame() %>%
  setNames("comments") %>%
  dplyr::mutate(id = data_turkishairlines_id[1])
```

The above code extract and saves the comments of first post in a variable named `full_comments`. However, to extract all posts data, a loop is required to index each post's list one by one. In each loop, the comments of one post is extracted and saved first in a variable `comment` and later the data from `comment` is saved into a variable `full_comments` using `rbind()` function, which binds rows of two dataframes.

```
# Saving comments of all 300 posts.
for (i in 1:299) {
  comment <- threehundred_posts[[i+1]]$comments$message %>%
    as.data.frame() %>%
    setNames("comments") %>%
    dplyr::mutate(id = data_turkishairlines_id[i+1])
  full_comments = rbind(full_comments, comment)
}
```

In the above loop, along with the comments' message, the post's ids are also added in the dataset in order to keep track of the columns as to which post does the comment belong. Later, the post ids are used to merge the posts data and comment data, i.e. `data_turkishairlines` and `full_comments` respectively.

```
# Assigning the post ids to its comments.
# The posts ids are repeated when there are more than one comment.
full_comment_post <- full_comments %>%
  dplyr::left_join(data_turkishairlines, by = "id")
```

As it can be seen in the dataframe printed below using `tibble()`, there are total 11 columns that include all the columns of posts data and one new column of comments. As there are more than one comments in a single post, there are more than one row for a single post due to numerous comments, therefore the post-specific data is redundant.

```
full_comment_post %>%
  tibble::as_tibble() %>%
  head(3)
```

```
## # A tibble: 3 x 11
##       from_id
##         <chr>
## 1 90430042759
## 2 90430042759
## 3 90430042759
## # ... with 10 more variables: comments <fctr>, id <chr>, from_name
<chr>, message <chr>, created_time <chr>, type <chr>, link <chr>, stor
y <chr>, likes_count <dbl>, comments_count <dbl>
```

For analysis purposes, a subset dataset can be made by selecting the required columns `dplyr::select()`. For example, the post's id and comments can only be selected in a subset dataset. Later, the rest of the columns can be added to the subset dataset using `dplyr::leftjoin()`. In an example below, two of the top comments of a post are printed. The `head()` function, with an argument valued two, selects the top two rows of a dataframe which has only `comments` as a single column.

```
full_comment_post %>%
  dplyr::select(comments) %>%
  tibble::as_tibble() %>%
  head(2)
```

```
## # A tibble: 2 x 1
##   comments
##   <fctr>
## 1 Direct flight from Ireland to bodrum badly needed even twice a
##   week
## 2 Just curious you do flights to Tbilisi?
```

5.3 Extracting TripAdvisor Reviews

TripAdvisor is the best web application that allows the companies in tourism and transport industry to explore their consumers' perception about their services using the big data of reviews and ratings. The study leverages reviews posted by Turkish Airlines' customers on TripAdvisor website. The method for extracting reviews data is mentioned in the section methodology. The data is stored in a `.rda` file format so that it can be accessed whenever needed, without scarping it from TripAdvisor repeatedly. Firstly, using `readRDS` function, the dataset is read and stored in a new variable `trip_turkishairlines`.

```
trip_turkishairlines <- readRDS(file = "tripadvisor_turkishairlines6846.rds")
```

The review dataset contains five variables, `id`, `data`, `quote`, `rating`, and `review`. The two main variables, `id` and `review` are used in the study for analysis. In contrast to the Facebook's extracted data, the data is already clean as the TripAdvsior's data structure is not as complex as Facebook's data structure.

```
trip_turkishairlines %>%
  dplyr::select(id, date, quote, rating, review) %>%
  tibble::as_tibble() %>%
  head(3)

## # A tibble: 3 x 5
##            id       date
##         <chr>     <dttm>
## 1 rn575914601 2018-04-26
## 2 rn575863809 2018-04-26
## 3 rn575859234 2018-04-26
## # ... with 3 more variables: quote <chr>, rating <int>,
## # review <chr>
```

As seen above, the printed table above doesn't show the content of `quote`, `rating` and `review`. The `tibble` format prints the dataframe in a tidy way by minimizing the

column with outsized data; furthermore, all of the columns after that minimized column are also suppressed. In order to view a review, the column `review` is selected first using `dplyr::select` and then `head` function is used to print the first review only. It can be seen below that the review is negative and the customer is complaining about customer service.

```
trip_turkishairlines %>%
  dplyr::select(review) %>%
  tibble::as_tibble() %>%
  dplyr::sample_n(1)
```

```
## # A tibble: 1 x 1
##  review
##  <chr>
## 2 Turkish airlines is the best airline I have had the pleasure of
 flying with. We flew from Dublin to Istanbul & Istanbul to Sharm el
 sheikh and returned with them. The cabin crew are extremely pleasant
 and always have a smile on there faces.we had meals included no
 charge and very nice too TV on all flights and very enjoyable...
```

5.4 Tokenization

The next step after extracting the data and cleaning it is tokenization of texts, i.e. comments and reviews. As mentioned in the methodology section, tokenization is the conversion of texts in to single words (unigram) or multiple words (n-grams). In this case, the texts tokenized into single words and saved into a dataframe. Each row of a dataframe contains a single word. The process of tokenization is briefly explained in the methodology section. The study created a function `tokenize` that takes `dataframe` and the `column`, which needs to be tokenized, as an arguments.

```
facebook_tokens <- tokenize("300_posts_comments", "comments")

facebook_tokens %>%
  tibble::as_tibble()
```

```
## # A tibble: 32,157 x 2
##    response_number    word
##              <chr>   <chr>
##  1               1  direct
##  2               1  flight
##  3               1    from
##  4               1 ireland
##  5               1      to
##  6               1  bodrum
##  7               1   badly
##  8               1  needed
##  9               1    even
## 10               1   twice
## # ... with 32,147 more rows
```

It can be seen in the table above, one of the comments is parsed into words and all of the words are included as tokens. The comment, “Direct flight from Ireland to bodrum badly needed even twice a week”, is tokenized into words. Furthermore, capital letters of all the word are converted into small letters to simplify the analysis.

```
tripadvisor_tokens <- tokenize("tripadvisor_turkishairlines6846.rds",
"review")

tripadvisor_tokens %>%
  tibble::as_tibble()
```

The response_number in the table refers to the comment number in case of Facebook data and review number in the case of TripAdvisor data. The total number of words in TripAdvisor data is ten times the number of words in Facebook data. The tokenized sample TripAdvisor data is given below:

```
## # A tibble: 325,179 x 2
##    response_number       word
##              <chr>      <chr>
##  1               1         to
##  2               1      start
```

```
##  3               1       with
##  4               1         me
##  5               1        and
##  6               1         my
##  7               1     sister
##  8               1        had
##  9               1          a
## 10               1 connection
## # ... with 325,169 more rows
```

5.4.1 Indexed Response Function

The `tokenize` function helps to convert text into sentences; however, the function does not keep track of the post and the comments from which the word is extracted. If the company aims to track a specific word, i.e. tries to locate the whole comment or the particular post under which the specific word is used, only `tokenize` function will not be helpful. To solve this issue, the study develops a new function, `numbered_response_tokens` that is used to get a resultant dataset with `post_number` and `comment_number` columns to track tokens. It requires two arguments, one is non-tokenized dataset and the other is the response column, either `review` or `comments`. First, the function converts the encoding of characters in a response column from `UTF-8` to `Latin1`. Second, a new column is made with the name `post_number`, assigining numbers to each review of Facebook Post.

```
numbered_response_tokens <- function(file, response_type) {

  dataset <-  readRDS(file = file)

  dataset %>%
    tibble::as_tibble() %>%
    dplyr::mutate(response_type =
                    iconv(pull(., response_type),
                          from = "UTF-8", to = "Latin1")) %>%
    dplyr::mutate(post_number = as.numeric(factor(id))) %>%
    dplyr::group_by(id) %>%
    dplyr::mutate_if(is.factor, as.character)
}
```

Facebook tokens require tracking on multiple levels, i.e. at post level and comment level, tracked by `post_number` and `comment_number` respectively. However, the TripAdvisor tokens are tracked using review numbers only. The column that refers to review number is named as `post_number` so that a function that deals with both Facebook and TripAdvisor data can be made. The tokenization of Facebook comments is done after a new column `response_number` is added to the dataset. Consequently, the `custom_stop_words` are taken out from the data using `dplyr::anti_join()`.

```
tidy_response_facebook <-
  numbered_response_tokens("300_posts_comments", "comments") %>%
  dplyr::mutate(response_number = row_number()) %>%
  tidytext::unnest_tokens(word, comments) %>%
  dplyr::anti_join(custom_stop_words)

tidy_response_facebook %>%
  tibble::as_tibble() %>%
  dplyr::select(post_number, response_number, word, type)
```

```
## # A tibble: 24,007 x 4
##    post_number response_number             word  type
##          <dbl>           <int>            <chr> <chr>
##  1           2               2 seniyorumtürkiye video
##  2           3               2             waaw  link
##  3           4              29          kettani photo
##  4           4              27             daba photo
##  5           4              27             yaba photo
## # ... with 24,002 more rows
```

An extra step of adding `comment_number` is not taken while tokenizing reviews. As mentioned earlier, it requires single level indexing and the post numbers have already been assigned in the function `numbered_response_tokens`.

```
tidy_response_tripadvisor <-
  numbered_response_tokens(
            "tripadvisor_turkishairlines6846.rds", "review") %>%
  tidytext::unnest_tokens(word, review) %>%
  dplyr::anti_join(custom_stop_words)
```

The table below shows different tokens and their review numbers. The data is sorted with respect to `response_number` column in the table. It can be seen that two different words “promised” and “the” are considered to be a single word as the review writer did not place any space before and after full stop. While comparing the words with English dictionary, the token “promised.the” will be considered as a noise and therefore it will be eliminated from the data.

```
tidy_response_tripadvisor %>%
  tibble::as_tibble() %>%
  dplyr::select(id, response_number, word, quote)
  dplyr::arrange(response_number)
```

```
## # A tibble: 124,757 x 3
## # Groups:   id [6,846]
##    id          response_number word
##    <chr>                 <dbl> <chr>
##  1 rn342674723               1 ticket
##  2 rn342674723               1 receive
##  3 rn342674723               1 promised.the
##  4 rn342674723               1 connection
##  5 rn342674723               1 istanbul
##  6 rn342674723               1 leave
##  7 rn342674723               1 immediately
##  8 rn342674723               1 dubai
##  9 rn342674723               1 arrive
## 10 rn342674723               1 grind
## # ... with 124,747 more rows
```

5.4.2 Sentence Tokens

The comments and review texts are tokenized using the function `sentence_tokens` as explained in the methodology section. The function requires three arguments, an untidy dataset - `data`, the column that needs to be tokenized - `response_column`, and the column which is used for grouping – `group_by`. In Facebook comments dataset the `response_column` is comments and the data is grouped by `comment_number`. It means that each comment is considered as a sentence in this study.

```
facebook_sentence_tokens <-
  sentence_tokens(data = untidy_response_facebook,
                  response_column = "comments",
                  group_by = "comment_number")
```

```
facebook_sentence_tokens %>%
  head(3)

## # A tibble: 3 x 3
##                              id comment_number
##                           <chr>          <int>
## 1 90430042759_10155125620582760              1
## 2 90430042759_10155125620582760              2
## 3 90430042759_10155125620582760              3
## # ... with 1 more variables: sentences <chr>
```

In the case of TripAdvisor's sentence tokenization, each review is considered to be a sentence. The `post_number` seen in the code given below is the review number and the `reponse_column` is `review` that contains review content. Since the function `sentence_tokens` is constructed to take data from both Facebook and TripAdvisor as an argument, a more general word "response" is used instead of comments or reviews. First three rows of the sentence tokenized data set is shown below. As the `sentences` variable has too many characters, it is minimized by R.

```
tripadvisor_sentence_tokens <-
  sentence_tokens(data = untidy_response_tripadvisor,
                  response_column = "review",
                  group_by = "post_number")

tripadvisor_sentence_tokens %>%
  head(3)

## # A tibble: 3 x 3
##            id post_number
##         <chr>       <dbl>
## 1 rn342674723           1
## 2 rn342740773           2
## 3 rn342772345           3
## # ... with 1 more variables: sentences <chr>
```

As previously mentioned, `select` and `head` functions are used to select the tokenized column and print the first row of the column in R console. One of the reviews from TripAdvisor can be seen in the sentence tokenized form. It can be seen that not all of the words in the review are included in the token due to spelling errors or punctuation marks next to the word.

```
tripadvisor_sentence_tokens %>%
  dplyr::select(sentences) %>%
  head(1)

## # A tibble: 1 x 1
##   sentences
##   <chr>
## 1 what it on the the ticket is not what you are going to receive
##   as flight from leaves when the flight from the ground staff off
##   er over night stay as a matter of course just another routine
##   day
```

5.5 Word Count

After tokenization of customers' feedback, the simplest analysis that can be conducted on tokens is counting the frequency of each distinct word used in the text. As explained in methodology section, the function `word_count` is used to find out the most frequent words in Facebook comments and TripAdvisor reviews.

```
facebook_word_count <- word_count("facebook_tokens")

facebook_word_count

## # A tibble: 7,622 x 2
##    word         n
##    <chr>    <int>
##  1 fly        236
##  2 love       224
##  3 istanbul   185
##  4 nice       141
##  5 service     96
##  6 travel      89
##  7 day         84
##  8 3           82
##  9 time        79
## 10 world       70
## # ... with 7,612 more rows
```

In Facebook comments, the most frequent word is “fly”. In 7,947 English words, 236 times word “fly” is used. The most frequent use of word fly and love shows that most of the followers have positive sentiments about Turkish Airlines. Furthermore, the use of word “istanbul” shows that most of the followers commenting on Facebook posts are Istanbul’s fans as well. Eithers these followers have used Turkish Airlines to travel Istanbul or they are willing to travel Istanbul in future. One can also say that one of the words that come into people’s mind when they hear about Turkish Airlines is “Istanbul”. In the light of this analysis, Turkish Airlines can introduce an offer in which it provides one or two day visits to connecting-flight Turkish Airlines’ passengers having stay in Istanbul. Therefore, passengers wishing to visit Istanbul will prefer Turkish Airlines to other Airlines even at higher fare.

```
tripadvisor_word_count <- word_count("tripadvisor_tokens")

tripadvisor_word_count

## # A tibble: 8,819 x 2
##    word         n
##    <chr>    <int>
##  1 service   3067
##  2 food      3042
##  3 istanbul  2531
##  4 fly       2431
##  5 time      2310
##  6 seat      1991
##  7 staff     1419
##  8 travel    1223
##  9 class     1167
## 10 plane     1025
## # ... with 8,809 more rows
```

The word frequency of TripAdvisor’s reviews depicts a different picture than the word frequency of Facebook comments. As seen in the table above, out of 10,463 distinct

tokens in 6,846 reviews, the customers most frequently use the word “service” and “food”. The usage shows that the major concern of the passengers travelling in Turkish Airlines is food. Although it cannot be claimed whether customers have positive sentiments about food or negative, the passengers are very particular regarding food served in the Airlines. The second major concern of the customers is customer service. It can be said that customers are more particular about food than the time (delays), staff attitude and seats, as food is at the top of the list followed by time staff and seats.

In the list of top ten most frequent words, most of the words are common in Facebook and TripAdvisor data, i.e. “love”, “istanbul”, “travel”, “fly” and “service”. It means that a large number of users either commenting on Facebook posts or writing reviews on TripAdvisor data have similar concerns and identical perception about Turkish Airlines.

Although the above word frequency tables specifies the list of most frequent words, a visualization technique, horizontal bar chart, displays relative word frequencies. This technique helps to figure out the frequency of one word relative to other related word. For instance, it can be seen that positive sentiments, such as “wow”, “beautiful”, “happy” and “amaze” are very near to each other and their bar charts have almost equal lengths, showing similar frequencies of these sentiments. The bar charts in the Figure 2 and Figure 3 are made using `word_count_plot` function, i.e. built for the purpose of this study. The two arguments required by the functions are the tokenized dataset and the minimum number of tokens to be shown in the bar chart.

```
word_count_plot("tokens_count_300", 25)
```

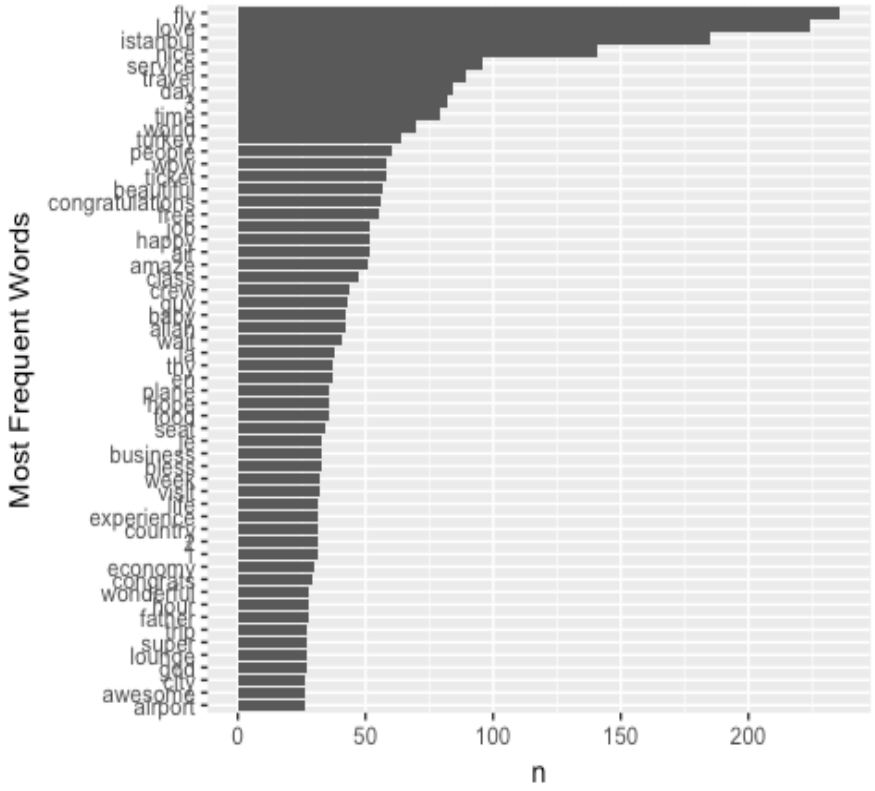


Figure 4: Facebook Most Frequent Word Count Plot

```
word_count_plot("TA_tokens_count_6846", 250)
```

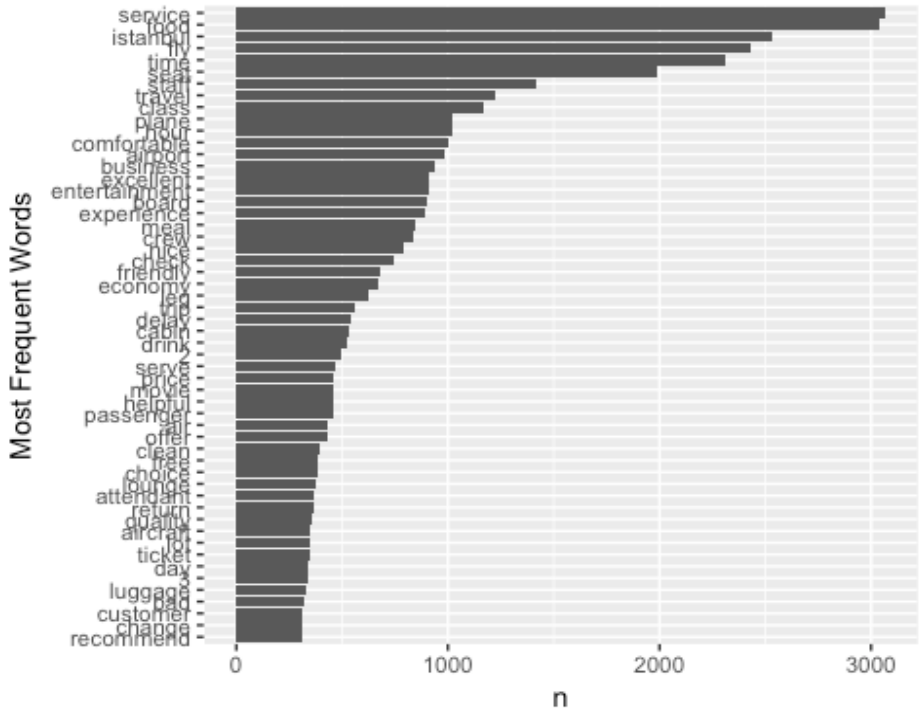


Figure 5: TripAdvisor Most Frequent Word Count Plot

The function `word_count_plot` requires two arguments, i.e. dataset and minimum frequency of the word. The word with frequency less than the minimum frequency will not be plotted in the bar chart. As the total number of words in reviews is comparatively more than total words in comments, the minimum frequency value for TripAdvisor tokens is 10 times of the minimum frequency for comments, i.e. 25. A huge difference in minimum frequency level is due to the fact that the noise, grammatical and spelling errors, in Facebook comments is higher than the noise in TripAdvisor reviews. Furthermore, it can also be assumed that the comments data has more variance than the review data, as the frequency level of most frequent words in reviews is 10 times higher than the frequency level of most frequent words in comments. In the most frequent word frequency plot above, it can be seen that words like “entertainment,” “movie”, and “meal” are frequently used by the customers. As entertainment is a great deal for customers, Turkish Airlines should focus on its customers’ entertainment to make their travelling experience more pleasant.

5.6 Sentiment Analysis

The simple analysis technique such as word count and word count visualization explore the most common words in a text; however, they include a number of neutral words that do not add value to the analysis. To make analysis more meaningful, another analysis technique, sentiment analysis, is employed that deals with sentiment, opinion or emotions, tokens or words. As mentioned in the methodology section, out of all the words, the words with either positive or negative sentiments are selected for analysis. Three different sentiment lexicons are used in this study: bing, nrc and AFINN.

Bing

Using `response_sentiments` function, the total number of positive and negative words in a response, a comment or a review, is figured out. Sentiment score of a response is calculated by subtracting the negative words count from the count of positive words. A negative score shows that negative words are more than positive; hence, the response is a negative response. On the other hand, positive response has positive sentiment score.

```
# bing
facebook_sentiments_bing <-
          response_sentiments(tidy_response_facebook, "bing")

facebook_sentiments_bing %>%
  dplyr::select(response_number, negative, positive, sentiment)
```

```
## # A tibble: 1,572 x 4
##    response_number negative positive sentiment
##              <int>    <dbl>    <dbl>     <dbl>
##  1               1        0        1         1
##  2               2        0        1         1
##  3               3        0        1         1
##  4               3        1        1         0
##  5               4        0        1         1
## # ... with 1,567 more rows
```

The dataset and the sentiment - `bing` - are two arguments of the function. The table above specifies the sentiment scores of ten comments on Facebook. The comments that do not have any sentiments are discarded from the analysis. There are total of 1,562 comments with sentiments out of 6,564 total comments in 300 posts, i.e. 23.7% of the total comments. The table shown above has more than one comments with similar response number. For instance, the response number 3 is repeated twice in the above dataset. The dataset is grouped by post ids. The two responses with a response number "3" correspond to two different posts. As Facebook ids are very lengthy, the id column is minimized using `select` function. Furthermore, all of the comments in the table are positive except the response listed in 7^{th} row. There is a majority of positive comments

in Facebook that shows the followers of Turkish Airlines have positive perception about the brand.

```
tripadvisor_sentiments_bing <-
          response_sentiments(tidy_response_tripadvisor, "bing")

tripadvisor_sentiments_bing

## # A tibble: 6,393 x 5
##             id response_number negative positive sentiment
##          <chr>           <dbl>    <dbl>    <dbl>     <dbl>
##  1 rn342740773               2        1        3         2
##  2 rn342772345               3        4        1        -3
##  3 rn342937701               4        0        3         3
##  4 rn343011985               5        1        1         0
##  5 rn343117108               6        0        1         1
## # ... with 6,388 more rows
```

In similar to the sentiment analysis of Facebook comments, the sentiment analysis on TripAdvisor reviews is also conducted. There are total of 6,393 reviews with sentiments out of 6,836 total reviews, i.e. 93.5% of total reviews. The percentage of sentiment content in TripAdvisor is roughly four times higher than the percentage of sentiment content in Facebook. It shows that the TripAdvisor review data is more valuable for analysis than the comments' data. This analysis technique helps to figure out the negative and positive reviews without effort. The positive reviews can be separated from the negative reviews and further analysis can be conducted on two different datasets. Furthermore, the sentiment score helps to compare the sentiment content between reviews, i.e. which review is more positive or negative than the other review.

NRC

NRC (Emolex) lexicon is more comprehensive lexicon that indicates the type of positive and negative sentiments in a response, such as anger, anticipation, disgust, fear, joy, sadness, surprise and trust. The sentiment analysis is conducted on Facebook comments using `response_sentiments` function as given below:

```
facebook_sentiments_nrc <-
            response_sentiments(tidy_response_facebook, "nrc")
```

The `tidy_response_facebook` is a tokenized Facebook comment dataframe that has `post_number` and `response_number` and `word` as three main columns. These dataset is compared with `nrc` lexicon as mentioned in the methodology to get the table as printed below:

```
facebook_sentiments_nrc
```

```
## # A tibble: 1,807 x 13
##                                id response_number anger anticipation
##                             <chr>           <int> <dbl>        <dbl>
##  1 90430042759_10155125620582760               1     0            0
##  2 90430042759_10155125620582760               2     0            0
##  3 90430042759_10155125620582760               3     0            0
##  4 90430042759_10155125620582760               4     0            0
##  5 90430042759_10155126945012760               1     0            1
## # ... with 1,802 more rows, and 9 more variables: disgust <dbl>,
## #   fear <dbl>, joy <dbl>, negative <dbl>, positive <dbl>,
## #   sadness <dbl>, surprise <dbl>, trust <dbl>, sentiment <dbl>
```

The number of comments having `nrc` sentiment words, i.e. 1,572, is more than the number of comments having `bing` sentiment words, i.e. 1807. It shows that `nrc` lexicon has power to analyze more words than `bing` lexicon. It is because of the fact that `nrc` use many different emotions rather than just two, positive or negative. Furthermore, sentiment analysis through `nrc` lexicon is more suitable to analyze consumers' perception as it specifies the types of negative emotions, like disgust, fear, anger and

sadness, as well as the types of positive emotions, like surprise, trust and anticipation. It can be seen in the table below that customers have commonly used the words of anger, disgust and fear in their reviews in TripAdvisor. Furthermore, there is a `sentiment` column that gives an overall sentiment score of the review.

```
tripadvisor_sentiments_nrc <-
            response_sentiments(tidy_response_tripadvisor, "nrc")

tripadvisor_sentiments_nrc
```

```
## # A tibble: 6,765 x 13
##             id response_number anger anticipation disgust  fear
##          <chr>           <dbl> <dbl>        <dbl>   <dbl> <dbl>
##  1 rn342674723               1     0            1       0     0
##  2 rn342740773               2     1            1       1     2
##  3 rn342772345               3     2            0       2     2
##  4 rn342937701               4     0            1       0     1
##  5 rn343011985               5     1            2       1     1
## # ... with 6,760 more rows, and 6 more variables: joy <dbl>,
## #   negative <dbl>, positive <dbl>, sadness <dbl>, surprise <dbl>,
## #   trust <dbl>, sentiment <dbl>
```

AFINN

As mentioned in the methodology, the `sentiments_afinn` function while using the `afinn` lexicon assigns a sentiment score to each word in a response. These scores are added to get the sentiment score of a response.

```
facebook_sentiments_afinn <-
               sentiments_afinn(tidy_response_facebook)

facebook_sentiments_afinn
```

```
## # A tibble: 1,634 x 3
##                                id response_number score
##                             <chr>           <int> <int>
##  1 90430042759_10155125620582760               1     5
##  2 90430042759_10155125620582760               2     3
##  3 90430042759_10155125620582760               3     3
##  4 90430042759_10155125620582760               7     2
##  5 90430042759_10155126945012760               3     2
## # ... with 1,629 more rows
```

A negative score indicates a negative comment or review while positive score shows that the response have over positive sentiment content in it. For instance, in the table given below, the third, fifth and eighth review of TripAdvisor are negative and comparatively 8th review is more negative than 5th, while 5th is more negative than 3rd review.

```
tripadvisor_sentiments_afinn <-
               sentiments_afinn(tidy_response_tripadvisor)

tripadvisor_sentiments_afinn

## # A tibble: 6,095 x 2
##    response_number score
##              <dbl> <int>
##  1               1     1
##  2               2     7
##  3               3    -2
##  4               4     3
##  5               5    -3
##  6               6     2
##  7               7     3
##  8               8    -5
##  9               9     4
## 10              10     7
## # ... with 6,085 more rows
```

5.6.1 Most Frequent Sentiments

Frequency of sentiments helps determine the overall sentiment score of a text. By looking at the most frequent sentiments, useful insight can be drawn that can be used to evaluate consumers' view of the company. Most frequent sentiments from the comments data of Turkish Airline's Facebook are reported below. It can be seen that the top 10 most frequent sentiments are positive which shows that most frequent user response to the company's Facebook posts is positive.

```
facebook_frequent_sentiments <- frequent_sentiments(tidy_response_face
book)

facebook_frequent_sentiments

## # A tibble: 384 x 3
##    word            sentiment     n
##    <chr>           <chr>     <int>
##  1 love            positive    328
##  2 nice            positive    155
##  3 beautiful       positive     87
##  4 congratulations positive     83
##  5 wow             positive     80
##  6 amaze           positive     73
##  7 free            positive     72
##  8 happy           positive     69
##  9 bless           positive     45
## 10 super           positive     37
## # ... with 374 more rows
```

Most frequent sentiments from the reviews data of TripAdvisor are presented below. It can be seen that most of the words are positive. However, the word "bad" appears in the top 10 most frequent sentiment words. The most frequent words can help the company draw useful insight about the areas of improvement as well as the areas that are important for the consumer experience. For example, the words "comfortable,"

"friendly" and "helpful" indicate the importance of the behavior of the airline staff with the passengers. However, most frequent words do not represent the overall sentiment score.

```
tripadvisor_frequent_sentiments <- frequent_sentiments(tidy_response_tripadvisor)

tripadvisor_frequent_sentiments

## # A tibble: 1,317 x 3
##    word        sentiment     n
##    <chr>       <chr>     <int>
##  1 comfortable positive   1026
##  2 excellent   positive    927
##  3 nice        positive    816
##  4 friendly    positive    700
##  5 delay       negative    552
##  6 helpful     positive    472
##  7 clean       positive    406
##  8 free        positive    399
##  9 bad         negative    324
## 10 recommend   positive    318
## # ... with 1,307 more rows
```

5.6.2 Plot Frequent Sentiments Counts

The Figure 4 shows a horizontal bar chart of frequent words side by side which enables one to compare the frequency of most frequent negative words and that of most frequent positive words. The frequency of top positive words is much higher than that of negative words, indicating that the overall positive sentiment score of the Facebook comments is higher than the negative sentiment score.

```
plot_sentiment_count(facebook_frequent_sentiments)
```

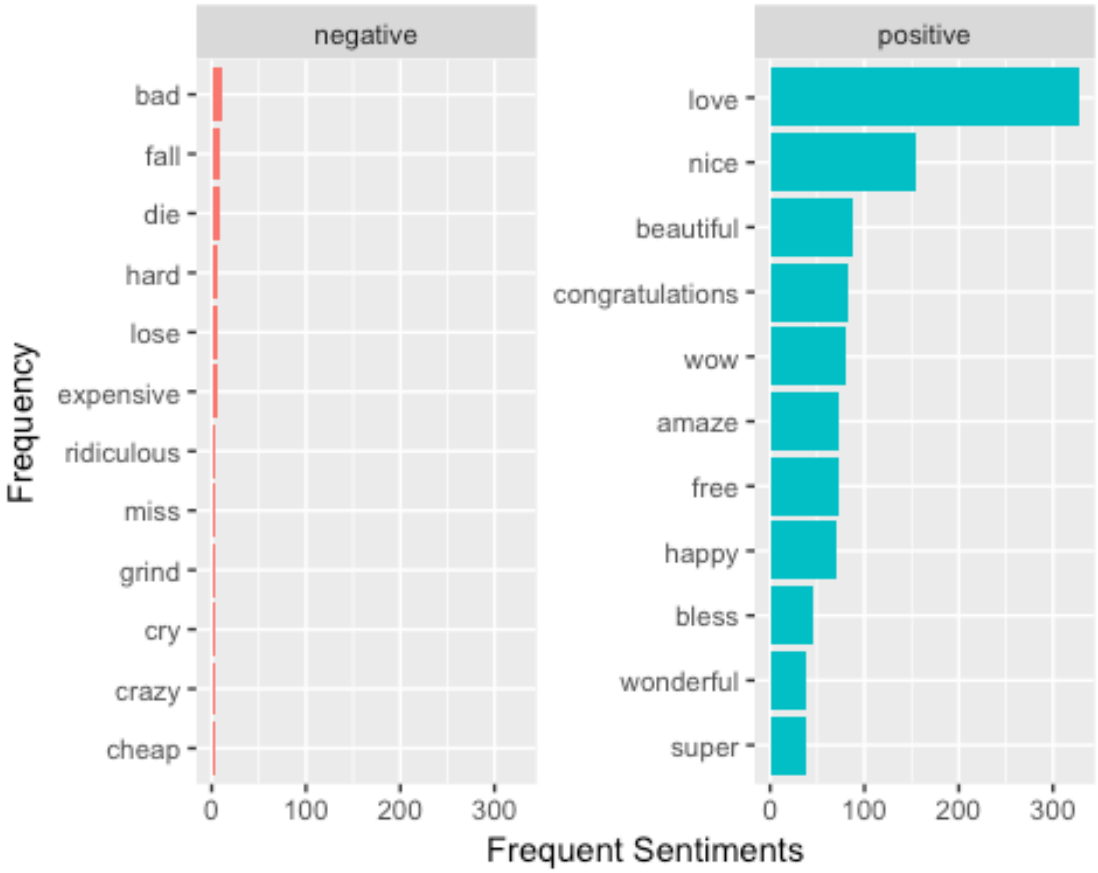


Figure 6: Facebook Most Frequent Sentiment Count Plot

Similarly, the frequency plot of most frequent positive and negative words in TripAdvisor review data is presented below. Unlike in the Facebook comments, the use of negative words is more frequent in the TripAdvisor reviews. Nevertheless, the use of positive words is much more common as compared to the use of negative words which indicates the overall sentiment score. Moreover, the nature of negative words indicates the concerns of consumers. For instance, “delay,” “miss”, and “lose” indicate that most of the negative comments are related to a bad experience of the consumer and the company can focus on these areas to reduce future negative reviews.

```
plot_sentiment_count(tripadvisor_frequent_sentiments)
```

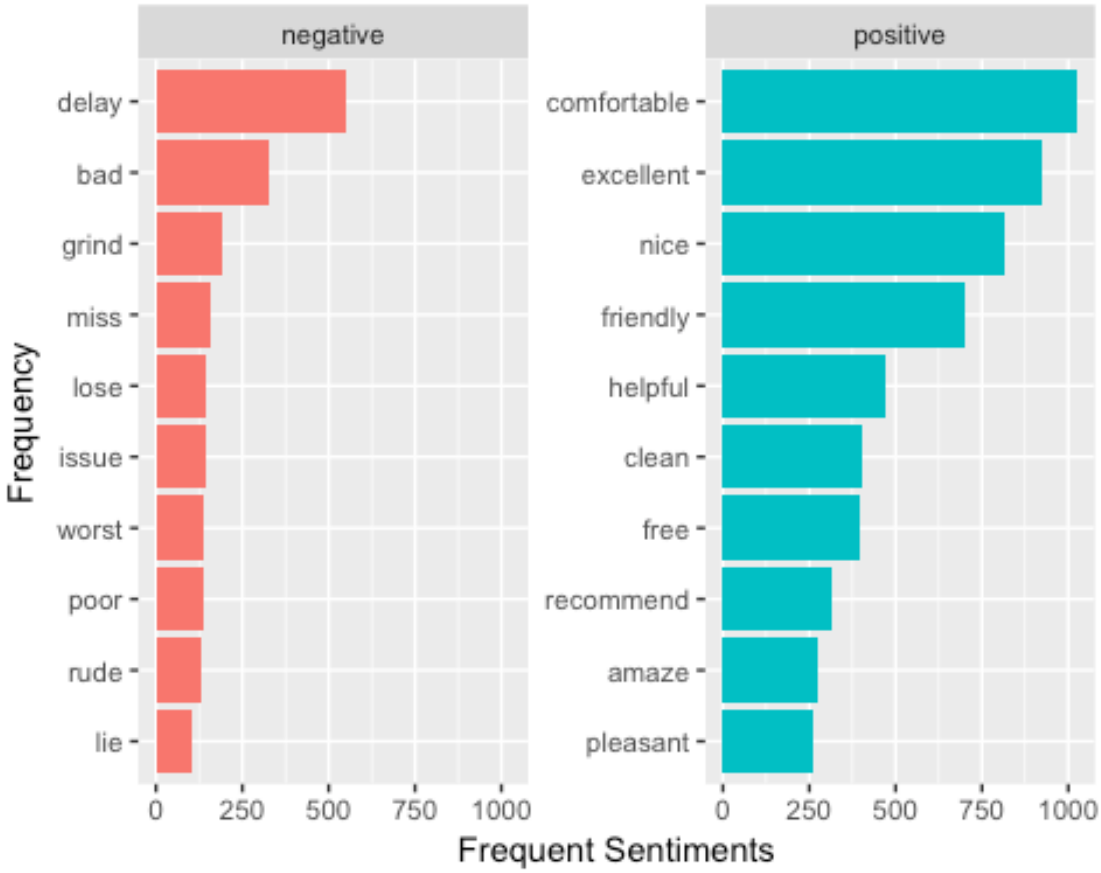


Figure 7: TripAdvisor Most Frequent Sentiment Count Plot

5.6.3 Sentiments to All Words Ratio

The ratio of sentiment count to all words count gives us an insight about how important is that sentiment in a comment or review. If the negative words are excessively used in a comment, it means that the response is highly negative. The study calculates the ratio by using `sentiment_token_ratio` function as explained in the methodology section.

```
facebook_negative_sentiment_ratio <-
  sentiment_token_ratio(tidy_response_facebook, "negative")
```

```
facebook_negative_sentiment_ratio
```

```
## # A tibble: 10 x 4
##    response_number negativewords  word      ratio
##              <int>         <int> <int>      <dbl>
##  1             281             1     2 0.50000000
##  2             805             1     2 0.50000000
##  3             661             1     3 0.33333333
##  4             687             1     3 0.33333333
##  5             358             1     6 0.16666667
##  6             314             1     7 0.14285714
##  7             157             1     8 0.12500000
##  8             475             1    10 0.10000000
##  9             503             1    11 0.09090909
## 10             102             1    18 0.05555556
```

In Facebook comments, the most negative comments contains only two or three words that result in higher sentiment to all words ratio. It shows that some of the followers have just posted two words with at least one negative word. Furthermore, the last row of the table shows that in response “102” of a certain post has one negative word in total 18 words. Please note that in the above table the post ids are minimized for better visualization of table.

In the case of TripAdvisor reviews, the frequency of negative words as compare to the total words count in top 10 negative words is higher than frequency of negative word in Facebook comments. The high frequency of negative words shows the level of negative sentiment content in negative comments. The sentiment-to-all-word-ratio also helps to rank the reviews based on sentiment content. Thus, analysts can figure out the comments with highest negative sentiment content. Consequently, the analysts can find common concerns and complaints of consumers.

```
tripadvisor_negative_sentiment_ratio <-
  sentiment_token_ratio(tidy_response_tripadvisor, “negative")
```

```
tripadvisor_negative_sentiment_ratio

## # A tibble: 13 x 4
##    response_number negativewords  word     ratio
##              <dbl>         <int> <int>     <dbl>
##  1            4976             5     7 0.7142857
##  2            3008             4     8 0.5000000
##  3            4705             4     8 0.5000000
##  4            4794             5    10 0.5000000
##  5            4384             5    11 0.4545455
##  6            3993             4     9 0.4444444
##  7            2284             9    21 0.4285714
##  8            5965             3     7 0.4285714
##  9            6477             3     7 0.4285714
## 10            3296             3     8 0.3750000
```

In the example above, the TripAdvisor review with the highest ratio is "4976". The study finds out the most negative review using the code given below. The required review is selected using `filter` function from `dplyr` package. Subsequently, the `review` column is selected using `select`. The review shows that the word "bad" is used five times in the comments, showing extreme consumer dissatisfaction. Furthermore, it can be seen that the word "food" is used twice in the comment that demonstrates consumer's biggest concern.

The importance of indexed tokenization is evident from the example discussed. Without proper indexing, it would have been impossible to find a specific review with a certain level of sentiment content.

```
numbered_response_tokens(
  "tripadvisor_turkishairlines6846.rds", "review") %>%
  dplyr::select(id, response_number = post_number, review,
                quote, rating, date) %>%
  dplyr::filter(response_number == 4976) %>%
  dplyr::select(review, id)
```

```
## # A tibble: 1 x 2
      review
       <chr>
## 1 the flight was very bad and the food very bad it so bad and i wou
ld not flight with now on so bad the food so bad i not flight with not
from soon
```

5.7 Word Cloud

Word cloud is a very interesting technique to visually consolidate information in a large amount of text. The following figure represents a word cloud constructed from the data of Turkish Airline's Facebook page. The word cloud represents fifty words that appear most frequently in response, i.e. comments, to official posts by the company. The size of the words depicts the importance and frequency of its usage. For example, it can be seen that the word "fly" and "love" are the first and second most frequent words respectively. As already discussed in the section "Word Count," these words appear most frequently in the comments of Facebook page.

```
word_cloud(tidy_response_facebook, max_words = 50)
```

Figure 8 Facebook Word Cloud

Moreover, as the word cloud shows most frequent words, it gives an idea of the user's perception about the company's services and social media campaign. The words "congratulations," "beautiful," "super," "wonderful," "wow" and "happy" show that most of the users have a positive brand image of the company. It also indicates that users tend to like the Facebook posts of the company. In other words, the word cloud gives an insight into the brand image of the company and the success of the social media campaign.

```
word_cloud(tidy_response_tripadvisor, max_words = 75)
```

Figure 9: TripAdvisor Word Cloud

Similarly, the above word cloud is constructed using the data of reviews about Turkish Airlines on TripAdvisor. Again, it shows the most frequent seventy-five words that were used in the reviews. Unlike the Facebook word cloud, the word cloud of TripAdvisor data shows that most of the users talk about the services provided by the company. This can give useful insight to the company when it comes to improving their services. As

already mentioned before, TripAdvisor data provides an insight into the mindset of Turkish Airline's existing or potential customers. The words such as "crew," "cabin," "staff," "seat," "attendant," "movie," "plane," "lounge," etc. show that the reviews provide feedback to the company which can help them to improve their services. It can be deduced that users value the quality of food and service much more than they value movies or seats which shows that the company can increase retention rate or repeat customers through good quality service during the flight. Moreover, the words "friendly," "excellent," "helpful" and "nice" show that users' feedback about the company's services is generally positive.

5.7.1 Sentiment Cloud

A sentiment cloud is a word cloud that is obtained after labeling the tokens (i.e. words) as positive and negative. As already mentioned, the size and number of the positive words cannot be compared with that of negative words. Nevertheless, it provides another interesting visual technique to get an overview of the user-response. The following sentiment cloud is constructed from the comment data of Turkish Airlines Facebook page and reviews data of TripAdvisor.

```
sentiment_cloud(tidy_response_facebook)
```

Figure 10: Facebook Sentiment Cloud

While the word cloud showed that the dominating sentiments in Facebook comments and TripAdvisor reviews were largely positive, the sentiment cloud also enables the company to see the prevalent negative comments and reviews so that they can deduce constructive criticism from them. This will enable the company to work towards improving their services and modifying their social media campaign to satisfy the users who had a negative experience with the company. For instance, words like “delay,” “lose,” “miss,” “rude,” “fault,” “broken,” “error” and “awful” clearly show negative experiences of the users that can tarnish the brand image of the company in eyes of potential consumers. This highlights areas of focus for the company when evaluating its services.

```
sentiment_cloud(tidy_response_tripadvisor)
```

Figure 11: TripAdvisor Sentiment Cloud

It is important to notice here that the positive sentiments in fact dominate the user response for Turkish Airlines in our analysis. Nonetheless, the negative responses are still valuable for a complete analysis to avoid a faulty and incomplete conclusion. Therefore, sentiment cloud proves to be very useful here as it provides an all-inclusive overview of the response by showing both negative and positive words.

6. CONCLUSION, IMPLICATIONS AND LIMITATIONS

6.1 Conclusion

The study employs various text mining techniques to retrieve useful information from Facebook and TripAdvisor, including web scraping, data cleaning, data wrangling, indexed tokenization, word frequency count, sentiment analysis, visualization of sentiment count and world cloud. Using several packages in R, comments on the Facebook page of Turkish Airlines and reviews about Turkish Airlines on TripAdvisor have been retrieved. The text in comments and reviews has been tokenized, i.e. converted into independent words, and the data has been cleaned for noise. The tokenized text has then been used for various analyses as an example of potential application of the data.

A number of interesting findings have been obtained during the information retrieval process. The customers' reviews on TripAdvisor website are less noisy, i.e. they have less spelling and grammatical errors, than Facebook comments. Facebook and TripAdvisor contain user generated data, providing insight into consumers' perception about brands and their customer service. Airlines, such as Turkish Airlines, can get valuable information and feedback from the text posted by their consumers.

For instance, the case study on Turkish Airlines social media data provides various handy findings. First, the brand perception of Turkish Airlines on consumers and potential consumers' minds is generally positive. Second, the major concerns of the consumers travelling through Turkish Airlines are food, timeliness and entertainment. Furthermore, the techniques of sentiment analysis rank the consumer responses from most negative to less negative response through indexed tokenization. Lastly, word cloud and sentiment cloud provides a complete overview of the users' perception, opinion, emotions and sentiments regarding a brand.

6.2 Implications for Future Research

The research study has a number of implications for future research in the field of social media analysis to find consumers' perception about brands. The study provides a complete process to scrape data from three different social media web applications, Facebook, Twitter and TripAdvisor. Especially in the case of TripAdvisor, despite of the fact that multiple URLs are required to retrieve all of the reviews, the study gives a code to extract all reviews at once. The reviews are a great source of information for research analysts and modern businesses who want to assess the brands' worth in the eyes of existing and potential consumers. Furthermore, researchers can conduct competitive analysis and make a perceptual map that specifies the position of a company relative to other companies in a same industry. Moreover, pairwise correlation analysis can be conducted on the indexed and tokenized responses of different companies to find out the most frequent co-occurring words.

The study provides a complete guideline for converting raw textual data, in weak structured form, into more structured dataset. The analysis of structured form of textual data is less expensive as compare to the unstructured form. Although the study emphasized on word-level feature or unigram tokens, it paves the way to extract more advanced level features, like terms or concepts, by providing more structured format of documents. Furthermore, the indexed tokenization of documents would help in evaluating the semantic similarity and applying supervised or unsupervised clustering techniques.

6.3 Limitations

It is to be noted that the sentiment analysis technique does not take qualifiers into account, for instance, negated texts like “not good” or “not comfortable”. The lexicon-based technique is based on single words only; therefore, negated texts are not being considered appropriately for analysis.

Another drawback of lexicon-based sentiment analysis methodology is that the size of the text has an impact on the analysis. While adding up the tokens’ sentiment score in a larger text, positive and negative sentiment score can be averaged out to be zero. The sentiment score obtained through this technique is more accurate when texts are paragraph-sized or sentence-sized (Silge and Robinson, tidytext: Text Mining and Analysis Using Tidy Data Principles in R 2016). However, the research study considers each Facebook’s comment or post, Twitter’s tweet and TripAdvisor’s review as a text and finds their sentiment score. As the size of comments, tweets and reviews is not large, there is minimal effect of text size on sentiment analysis.

7. REFERENCES

WebHarvy. *What is Web Scraping?* 2017. https://www.webharvy.com/articles/what-is-web-scraping.html.

Wickham, Hadley. *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York, 2009.

—. "hadley/rvest." *Github.* 2015. https://github.com/hadley/rvest/blob/master/demo/tripadvisor.R.

—. *R packages: organize, test, document, and share your code.* O'Reilly Media, Inc., 2015.

—. *R Packages: Organize, Test, Document, and Share Your Code.* O'Reilly Media, Inc., 2015.

—. "rvest: Easily Harvest (Scrape) Web Pages." *CRAN.* 2016. https://CRAN.R-project.org/package=rvest.

Wickham, Hadley. "Tidy Data." *Journal of Statistical Software* 59, no. 10 (2014): 1-23.

Wickham, Hadley, and Lionel Henry. "tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions." Vers. 0.8.0. *CRAN.* 2018. https://CRAN.R-project.org/package=tidyr.

Wickham, Hadley, Romain Francois, Lionel Henry, and Kirill Müller. "dplyr: A Grammar of Data Manipulation." Vers. 0.7.2. *CRAN.* 2017. https://CRAN.R-project.org/package=dplyr.

Wong, S. K. M., W. Ziarko, and P. C. N. Wong. "Generalized Vector Space Model in Information Retrieval." *8th Annual International ACM SIGIR Conference on Research and Development and Information Retrieval.* NewYork, 1985. 18-25.

Xu, H. "Benefits and Concerns of Using Social Media - Users ' Perspective Benefits and Concerns of Using Social Media ." *MWAIS.* 2016.

Ahmadi, Mojtaba. "A Sentiment Analysis Application for Twitter Data." Tampere University of Applied Sciences, 2017.

Barbera, Pablo, Michael Piccirilli, Andrew Geisler, and Wouter. "Rfacebook: Access to Facebook API via R." *CRAN.* 2017. https://CRAN.R-project.org/package=Rfacebook (accessed 2017).

Curran, J. R. "From Distributional to Semantic Similarity." *Ph.D Thesis.* University of Edinburgh, 2003.

Carlsson, S., S. Sarker, and S. Zafeiropoulou. "What's Trending in Social Media Analytics Area? A Retrospective." 2015.

Chen, H., R.H.L Chiang, and V.C. Storey. "Business Intelligence and Analytics: From Big Data To Big Impact ." *MIS Quarterly*, 2012: 1165-1188.

Codd, Edgar F. *The Relational Model for Database Management: Version 2.* Addison-Wesley Longman Publishing Co. Inc., 1990.

Dasu, T., and T. Johnson. *Exploratory Data Mining and Data Cleaning.* Vol. 479. John Wiley & Sons, 2003.

Dong, H., F.K. Husain, and E. Chang. "A Survey in Traditional Information Retrieval Models." *IEEE International Conference on Digital Ecosystems and Technologies*, 2008: 397-402.

Feldman, Ronen, and James Sanger. *The Text Mining Handbook.* Cambridge University Press, 2007.

Gentry, Jeff. "twitteR: R Based Twitter Client." *CRAN.* 2015. https://CRAN.R-project.org/package=twitteR.

Ihaka, Ross. "R : Past and Future History." *Computing Science and Statistics*, 1998.

Hu, Minqing, and Bing Liu. "Mining and Summarizing Customer Reviews." *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* Seattle, 2004. 22-25.

Jurafsky, Daniel, and James H. Martin. *Speech and Language Processing.* Pearson International, 2017.

—. *Speech and Language Processing.* Third Edition Draft. 2017.

Jaccard, P. "The Distribution of the Flora of the Alpine Zone." *New Phytologist* 11 (1912): 37-50.

Jenkins, Aric. *Facebook Just Revealed 3 Major Changes to Its Privacy Settings.* March 28, 2018. http://time.com/5218395/facebook-privacy-settings-changes-cambridge-analytica/ (accessed April 25, 2018).

Kullback, S., and R. A. Leibler. "On Information and Sufficiency." *Annals of Mathematical Statistics* 22 (1951): 79-86.

Kleindienst, D., R. Pfleger, and M. Schoch. "The Business Alignment of Social Media Analytics." *European Conference on Information Systems* . 2015.

Lee, L. "Measures of Distributional Similarity." *In ACL-99*, 1999: 25-32.

Nielsen, Finn Årup. "A new ANEW: Evaluation of a word list for sentiment analysis in microblogs." *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages 718 in CEUR Workshop Proceedings.* 2011. 93-98.

Mohammad, Saif M., and and Peter D. Turney. "Crowdsourcing A Word–Emotion Association Lexicon." *Computational Intelligence* 29, no. 3 (2013): 436-465.

Pereira, Renita Crystal, and Vanitha T. "Web Scraping of Social Networks." *International Journal of Innovative Research in Computer and Communication Engineering* 3, no. 7 (10 2015).

Salton, Gerard. "The SMART Retrieval System: Experiments in Automatic Document Processing." *EEE Transactions on Professional Communication* (IEEE) 15, no. 1 (1972): 17-17.

Schutze, H. "Dimensions of Meaning." *Supercomputing.* IEEE Press, 1992. 787-796.

Schreck, T., and D. Keim. "Visual Analysis of Social Media Data." *Computer*, 2013.

Silge, Julia, and David Robinson. *Text Mining with R - A Tidy Approach.* O'REILLY, 2018.

Silge, Julia, and David Robinson. "tidytext: Text Mining and Analysis Using Tidy Data Principles in R." *JOSS* (The Open Journal) 1, no. 3 (2016).

Singh, Vikram, and Balwinder Saini. "An Effective Tokenization Algorithm For Information Retrieval Systems." *Department of Computer Engineering* (National Institute of Technology, Kurukshetra, Haryana, India), 2014.

Statista. *Facebook - Statistics & Facts.* January 10th, 2018. https://www.statista.com/topics/751/facebook/.

—. *How do online customer reviews affect your opinion of a local business?* 11 2017. https://www.statista.com/statistics/315751/online-review-customer-opinion/.

—. *TripAdvisor - Statistics & Facts.* 2018. https://www.statista.com/topics/3443/tripadvisor/.

Robinson, David. "broom: Convert Statistical Analysis Objects into Tidy Data Frames." *CRAN.* 2018. https://CRAN.R-project.org/package=broom.

8. APPENDIX

Facebook Data Extraction

```
library(Rfacebook)

# Fb Authorization
fb_oauth <- Rfacebook::fbOAuth(
  app_id="1380496555352781",
  app_secret="eb3abc42d1e00536e6f4e37e58fc0b5d",
  extended_permissions = TRUE)

# Saving variable fb_oauth in a file and loading it
save(fb_oauth, file="fb_oauth")
load("fb_oauth")

# Extract posts from turkish airlines page
turkishairlines <- Rfacebook::getPage(page = "turkishairlines",
                                      token = fb_oauth, n = 2000)

# Save the posts in R data file
saveRDS(turkishairlines, "turkishairlines_2000")

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

# Read R data file and store in new variable
data_turkishairlines <- readRDS("turkishairlines_2000")

# Storing the post ids in a new variable
data_turkishairlines_id <- data_turkishairlines$id

# Viewing limited variables and rows
data_turkishairlines %>%
  dplyr::select(from_id, likes_count, type, comments_count,
                shares_count, message) %>%
  tibble::as_tibble() %>%
  head(5)

## # A tibble: 5 x 6
##   from_id     likes_count type  comments_count shares_count message
##   <chr>             <dbl> <chr>          <dbl>        <dbl> <chr>
## 1 90430042759         254 photo             12           18 Our flights t…
## 2 90430042759         300 photo            132           92 "We are looki…
## 3 90430042759         869 photo             39           43 The most plea…
```

```
## 4 90430042759         261 photo              9            7 Famous actor …
## 5 90430042759         227 photo             13           11 Within the sc…
```

```
data_turkishairlines %>%
  dplyr::select(message) %>%
  tibble::as_tibble() %>%
  head(2)
```

```
## # A tibble: 2 x 1
##   message
##   <chr>
## 1 Our flights to Samarkand start on March 16th, 2018!
## 2 "We are looking for new pilots! Join us at Turkish Airlines Pilot Roads…
```

```
# Extracting comments and reactions from the extracted posts
# The comments are extracted using post ids. Therefore, id variable
# is used.
all_posts <- data_turkishairlines_id[1:300] %>%
  lapply(getPost, n = 50000, token=fb_oauth,
         comments = TRUE, reactions = TRUE)

# Save first 300 posts comments in a R data file.
saveRDS(all_posts, "300_posts")
```

Converting Facebook lists data into dataframe

```
# Reading the saved RDS file instead of applying getPost function repeatedly.
# The resultant variable is a list which includes a post specific list.
# Within each post specific list there are post comments and reactions.
threehundred_posts <- readRDS("300_posts")

# Saving comments for first post in new variable
full_comments <- threehundred_posts[[1]]$comments$message %>%
  as.data.frame() %>%
  setNames("comments") %>%
  dplyr::mutate(id = data_turkishairlines_id[1])

# Expanding full_comments by adding comments of remaining posts.
for (i in 1:299) {
  comment <- threehundred_posts[[i+1]]$comments$message %>%
    as.data.frame() %>%
    setNames("comments") %>%
    dplyr::mutate(id = data_turkishairlines_id[i+1])
  full_comments = rbind(full_comments, comment)
}

# Assigning the post ids to its comments.
# The posts ids are repeated when there are more than one comment.
full_comment_post <- full_comments %>%
  dplyr::left_join(data_turkishairlines, by = "id")

# Saving the comments of posts in R data file.
saveRDS(full_comment_post, "300_posts_comments")

readRDS("300_posts_comments") %>%
  subset(select = c(3,1,2,4,5,6,7,8,9,10,11)) %>%
```

```
  tibble::as_tibble() %>%
  head(5)
```

```
## # A tibble: 5 x 11
##   from_id  comments id    from_name message created_time type  link  story
##   <chr>    <fct>    <chr> <chr>     <chr>   <chr>        <chr> <chr> <chr>
## 1 9043004… Direct … 9043… Turkish … Our fl… 2017-10-05T… photo http… <NA>
## 2 9043004… Just cu… 9043… Turkish … Our fl… 2017-10-05T… photo http… <NA>
## 3 9043004… What ab… 9043… Turkish … Our fl… 2017-10-05T… photo http… <NA>
## 4 9043004… What ab… 9043… Turkish … Our fl… 2017-10-05T… photo http… <NA>
## 5 9043004… Very in… 9043… Turkish … Our fl… 2017-10-05T… photo http… <NA>
## # ... with 2 more variables: likes_count <dbl>, comments_count <dbl>
```

```
readRDS("300_posts_comments") %>%
  dplyr::select(comments) %>%
  tibble::as_tibble() %>%
  head(2)
```

```
## # A tibble: 2 x 1
##   comments
##   <fct>
## 1 Direct flight from Ireland to bodrum badly needed even twice a week in …
## 2 Just curious you do flights to Tbilisi?
```

Twitter Data Extraction

```
# Load Requried Packages
library("SnowballC")
library("tm")
library("twitteR")
library("syuzhet")

# Authonitical keys
consumer_key <- 'tAyR9LyhATfD90aA7Ft1Zfj3I'
consumer_secret <- 'vX1RHqHHDpnmNOqrGPMVnmnQjQvG98X3xlB7T7zv4hKcvj7tVv'
access_token <- '2572842085-vExbB4HNvN57zmQhoQdbmutC16a4kdMdh1xVta5'
access_secret <- 'HtTHSeAOz1WPcUX8nfW5ddwZ1TbXZGFB4pSHU0IZ3agvA'

twitteR::setup_twitter_oauth(consumer_key, consumer_secret,
                             access_token, access_secret)

tweets <- userTimeline("turkishairlines", n=200)
```

Trip Advisor Data Extraction

```
library("rvest")

url <- "https://www.tripadvisor.com/Airline_Review-d8729174-Reviews-Turkish-Air
lines"
url <- "https://www.tripadvisor.com/Airline_Review-d8729174-Reviews-or20-Turkis
h-Airlines#REVIEWS"

df_total = data.frame()

for (i in seq(0, 20050, 10))
{
  if (i == 0) {
```

```
    url <- "https://www.tripadvisor.com/Airline_Review-d8729069-Reviews-Emirate
s"
  }

  else  {
    url <- paste(
      "https://www.tripadvisor.com/Airline_Review-d8729069-Reviews-or",i,"-Emir
ates#REVIEWS",
      sep = "")
  }

  reviews <- url %>%
    read_html() %>%
    html_nodes("#REVIEWS .innerBubble")

  id <- reviews %>%
    html_node(".quote a") %>%
    html_attr("id")

  quote <- reviews %>%
    html_node(".quote span") %>%
    html_text()

  rating <- reviews %>%
    html_node(".rating .rating_s_fill") %>%
    html_attr("alt") %>%
    gsub(" of 5 stars", "", .) %>%
    as.integer()

  date <- reviews %>%
    html_node(".rating .ratingDate") %>%
    html_attr("title") %>%
    strptime("%b %d, %Y") %>%
    as.POSIXct()

  review <- reviews %>%
    html_node(".entry .partial_entry") %>%
    html_text()

  df <- data.frame(id, quote, rating, date, review, stringsAsFactors = FALSE)
  df_total <- rbind(df_total, df)
}

# Save an object to a file
saveRDS(df_total, file = "tripadvisor_turkishairlines6846.rds")

trip_turkishairlines <- readRDS(file = "tripadvisor_turkishairlines6846.rds")

trip_turkishairlines %>%
  dplyr::select(id, date, quote, review) %>%
  tibble::as_tibble() %>%
  dplyr::sample_n(4)

## # A tibble: 4 x 5
##   id          date                quote       review
##   <chr>       <dttm>              <chr>       <chr>
## 1 rn575863809 2018-04-26 00:00:00 Istanbul…  Turkish airlines is…
```

```
## 2 rn575859234 2018-04-26 00:00:00 Broke my…  Travelled from Atat…
## 3 rn575854758 2018-04-26 00:00:00 comforta…  it is amazing trave…
## 4 rn575817214 2018-04-26 00:00:00 Despite …  Always like travell…

trip_turkishairlines %>%
  dplyr::select(review) %>%
  tibble::as_tibble() %>%
  head(2)

## # A tibble: 2 x 1
##   review
##   <chr>
## 1 "To start with me and my sister had a connection in Istanbul where th…
## 2 "Turkish airlines is the best airline I have had the pleasure of flyi…
```

Tokenization

```
library(tidytext)
data(stop_words)

# Custom stop words
custom_stop_words <- data.frame(word = c("miss", "flight", "tukish",
                                         "airlines", "flights",
                                         "airline", "turkish", "de"),
                                lexicon = c("custom")) %>%
  rbind(stop_words)

tokenize <- function(file, data_type) {
  data_tibble <- readRDS(file = file) %>%
    tibble::as_tibble()

  data_vector <- data_tibble %>%
    dplyr::pull(data_type) %>%
    iconv(from = "UTF-8", to = "Latin1")

  tokens <- tibble::as_tibble(data_vector) %>%
    dplyr::filter(!is.na(value)) %>%
    dplyr::mutate(response_number = rownames(.)) %>%
    dplyr::select(response_number, value) %>%
    tidytext::unnest_tokens(word, value)

  tokens
}

# Convert non_base verbs into base verbs

extract_non_base <- function(data) {
  data %>%
    dplyr::rename(non_base = word) %>%
    dplyr::left_join(readRDS("sahban_base_lexicon"), by = "non_base") %>%
    dplyr::mutate(base = ifelse(is.na(base), non_base, base)) %>%
    dplyr::rename(word = base) %>%
    dplyr::select(-one_of("non_base"))
}
```

```r
# Convert Plurals to Singular Noun

extract_plural <- function(data) {
  data %>%
    dplyr::rename(plural = word) %>%
    dplyr::left_join(readRDS("sahban_noun_lexicon"), by = "plural") %>%
    dplyr::mutate(noun = ifelse(is.na(noun), plural, noun)) %>%
    dplyr::rename(word = noun) %>%
    dplyr::select(-one_of("plural"))
}

facebook_tokens <- tokenize("300_posts_comments", "comments") %>%
  extract_non_base() %>%
  extract_plural()

tripadvisor_tokens <- tokenize("tripadvisor_turkishairlines6846.rds", "review")
 %>%
  extract_non_base() %>%
  extract_plural()

saveRDS(tokens_count, "tokens_count_300")
saveRDS(tokens_count, "TA_tokens_count_6846")

facebook_tokens %>%
  tibble::as_tibble()

## # A tibble: 32,200 x 2
##    response_number word
##    <chr>           <chr>
##  1 1               direct
##  2 1               flight
##  3 1               from
##  4 1               ireland
##  5 1               to
##  6 1               bodrum
##  7 1               badly
##  8 1               need
##  9 1               even
## 10 1               twice
## # ... with 32,190 more rows

tripadvisor_tokens %>%
  tibble::as_tibble()

## # A tibble: 326,163 x 2
##    response_number word
##    <chr>           <chr>
##  1 1               to
##  2 1               start
##  3 1               with
##  4 1               me
##  5 1               and
##  6 1               my
##  7 1               sister
##  8 1               had
##  9 1               a
```

```
## 10 1               connection
## # ... with 326,153 more rows
```

Word Count

```
word_count <- function(data) {
  readRDS(data) %>%
    dplyr::anti_join(custom_stop_words, by = "word") %>%
    dplyr::count(word, sort = TRUE) %>%
    tibble::as_tibble()
}

facebook_word_count <- word_count("facebook_tokens")

## Warning: Column `word` joining character vector and factor, coercing into
## character vector

tripadvisor_word_count <- word_count("tripadvisor_tokens")

## Warning: Column `word` joining character vector and factor, coercing into
## character vector

facebook_word_count

## # A tibble: 7,622 x 2
##    word         n
##    <chr>    <int>
##  1 fly        236
##  2 love       224
##  3 istanbul   185
##  4 nice       141
##  5 service     96
##  6 travel      89
##  7 day         84
##  8 3           82
##  9 time        79
## 10 world       70
## # ... with 7,612 more rows

tripadvisor_word_count

## # A tibble: 8,819 x 2
##    word         n
##    <chr>    <int>
##  1 service   3067
##  2 food      3042
##  3 istanbul  2531
##  4 fly       2431
##  5 time      2310
##  6 seat      1991
##  7 staff     1419
##  8 travel    1223
##  9 class     1167
## 10 plane     1025
## # ... with 8,809 more rows
```

Word Count Plot

```
word_count_plot <- function(data, min_count) {
  tokens_count <- readRDS(data)

  tokens_count %>%
    dplyr::anti_join(custom_stop_words) %>%
    dplyr::filter(n > min_count) %>%
    dplyr::mutate(word = reorder(word, n)) %>%
    ggplot2::ggplot(mapping = ggplot2::aes(word, n)) +
    ggplot2::geom_col() +
    ggplot2::xlab("Most Frequent Words") +
    ggplot2::coord_flip()
}

word_count_plot("tokens_count_300", 25)

## Joining, by = "word"

## Warning: Column `word` joining character vector and factor, coercing into
## character vector
```

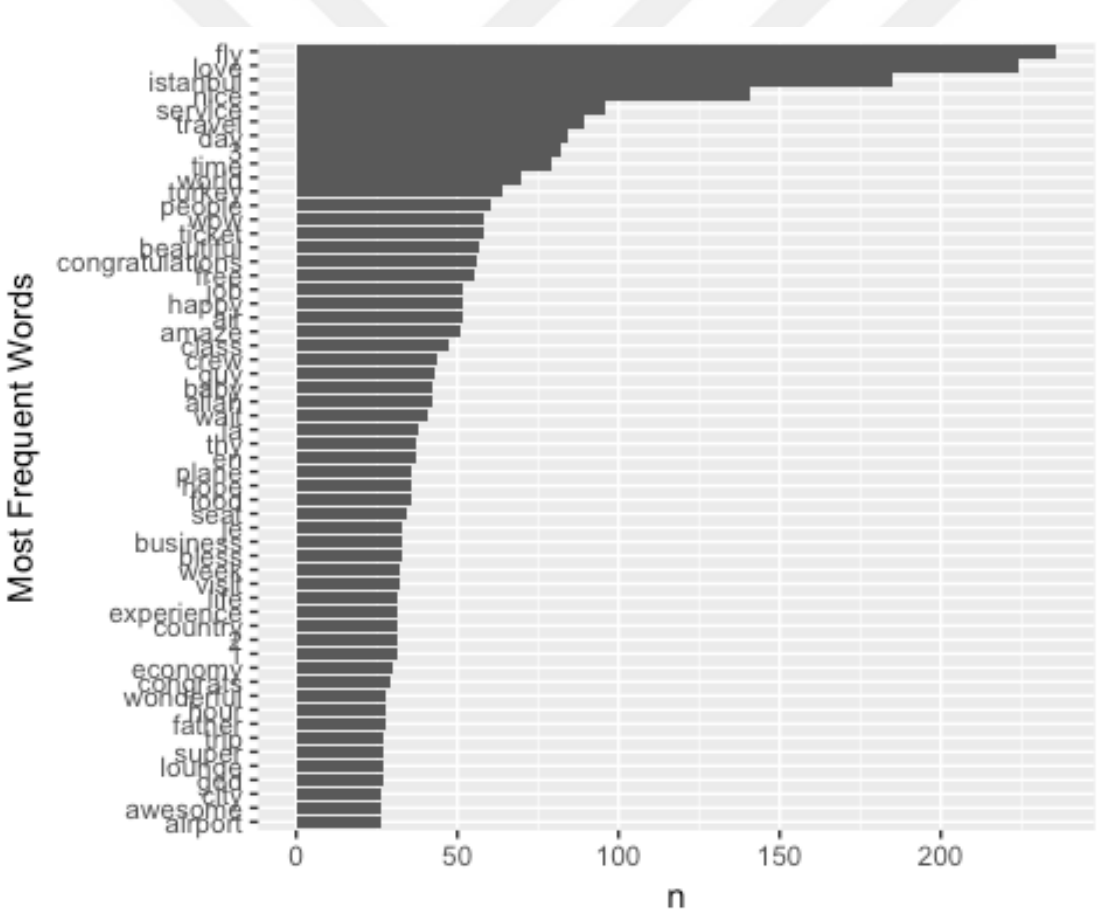


Figure 12: Facebook Word Count Plot - Appendix

```
word_count_plot("TA_tokens_count_6846", 300)

## Joining, by = "word"
```

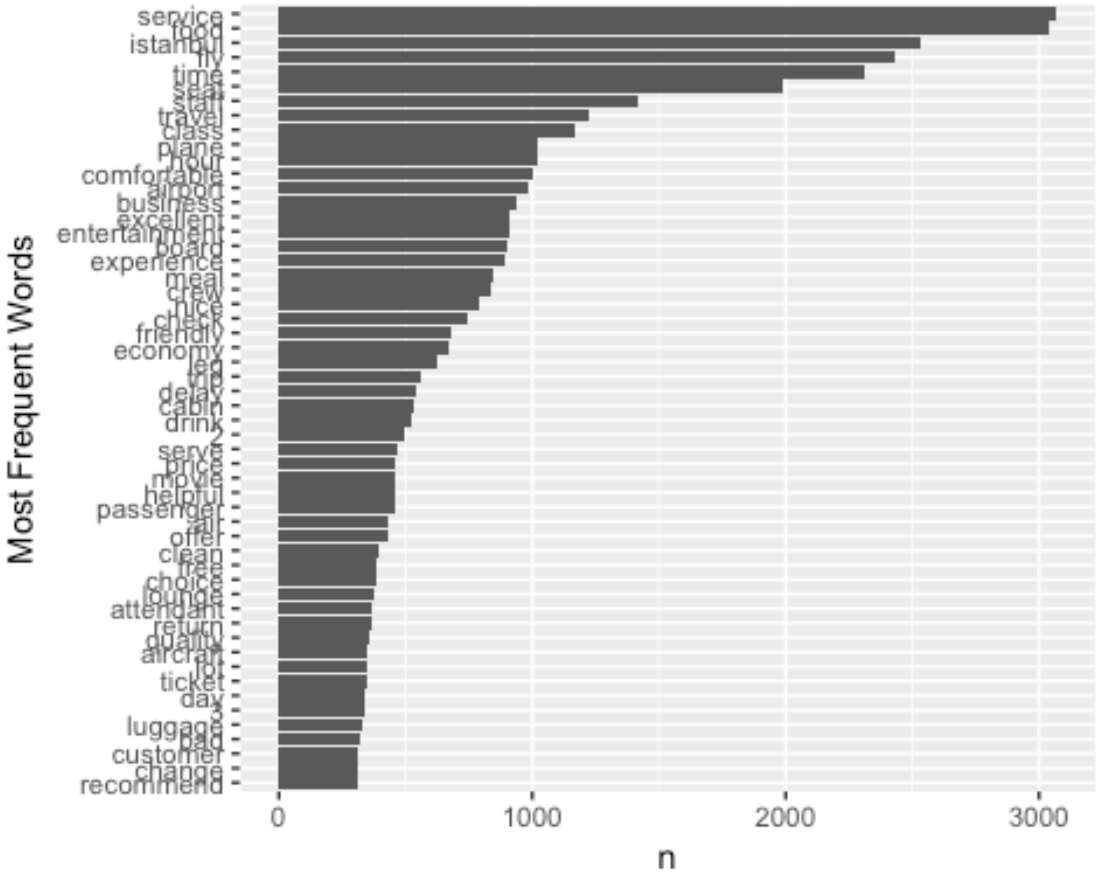


Figure 13: TripAdvisor Word Count Plot - Appendix

```
# Adding words manually to the lexicon
sahban_base_lexicon <- readRDS("sahban_base_lexicon") %>%
  dplyr::bind_rows(data.frame(base = c("travel", "travel"),
                              non_base = c("travelled", "travelling")))
saveRDS(sahban_base_lexicon, "sahban_base_lexicon")
```

Indexed Response Function

```
numbered_response_tokens <- function(file, response_type) {

  dataset <-  readRDS(file = file)

  dataset %>%
    tibble::as_tibble() %>%
    dplyr::mutate(response_type =
                    iconv(pull(., response_type),
                          from = "UTF-8", to = "Latin1")) %>%
    dplyr::mutate(post_number = as.numeric(factor(id))) %>%
    dplyr::group_by(id) %>%
    dplyr::mutate_if(is.factor, as.character)
}



# Convert non_base verbs into base verbs

extract_non_base <- function(data) {
  data %>%
```

```
    dplyr::rename(non_base = word) %>%
    dplyr::left_join(readRDS("sahban_base_lexicon"), by = "non_base") %>%
    dplyr::mutate(base = ifelse(is.na(base), non_base, base)) %>%
    dplyr::rename(word = base) %>%
    dplyr::select(-one_of("non_base"))
}
```

Indexed Tokenization

The `numbererd_response_function()` is used to find out numbered tokenization.

```
tidy_response_facebook <-
  numbered_response_tokens("300_posts_comments", "comments") %>%
  dplyr::mutate(comment_number = row_number()) %>%
  dplyr::select(id, post_number, response_number = comment_number,
                comments, created_time, type, likes_count, comments_count,
                shares_count) %>%
  tidytext::unnest_tokens(word, comments) %>%
  dplyr::anti_join(custom_stop_words) %>%
  extract_non_base() %>%
  extract_plural()

## Joining, by = "word"

## Warning: Column `word` joining character vector and factor, coercing into
## character vector

tidy_response_tripadvisor <-
  numbered_response_tokens("tripadvisor_turkishairlines6846.rds", "review") %>%

  dplyr::select(id, response_number = post_number, review, quote, rating, date)
 %>%
  tidytext::unnest_tokens(word, review) %>%
  dplyr::anti_join(custom_stop_words) %>%
  extract_non_base() %>%
  extract_plural()

## Joining, by = "word"

tidy_response_facebook %>%
  tibble::as_tibble() %>%
  dplyr::select(post_number, response_number, word, type) %>%
  subset(select = c(2,3,4,5))

## Adding missing grouping variables: `id`

## # A tibble: 23,953 x 4
##    post_number response_number word      type
##          <dbl>           <int> <chr>     <chr>
##  1         293               1 direct    photo
##  2         293               1 ireland   photo
##  3         293               1 bodrum    photo
##  4         293               1 badly     photo
##  5         293               1 week      photo
##  6         293               1 summer    photo
##  7         293               1 fantastic photo
##  8         293               2 curious   photo
```

```
##  9         293               2 tbilisi   photo
## 10         293               3 fligths   photo
## # ... with 23,943 more rows

tidy_response_tripadvisor %>%
  tibble::as_tibble() %>%
  dplyr::select(id, response_number, word) %>%
  dplyr::arrange(response_number)

## # A tibble: 125,065 x 3
## # Groups:   id [6,846]
##    id          response_number word
##    <chr>                 <dbl> <chr>
##  1 rn342674723               1 ticket
##  2 rn342674723               1 receive
##  3 rn342674723               1 promised.the
##  4 rn342674723               1 connection
##  5 rn342674723               1 istanbul
##  6 rn342674723               1 leave
##  7 rn342674723               1 immediately
##  8 rn342674723               1 dubai
##  9 rn342674723               1 arrive
## 10 rn342674723               1 grind
## # ... with 125,055 more rows
```

Sentence Tokenization

```
untidy_response_facebook <-
  numbered_response_tokens("300_posts_comments", "comments") %>%
  dplyr::mutate(comment_number = row_number()) %>%
  dplyr::select(id, post_number, comment_number, comments,
                created_time, type, likes_count, comments_count,
                shares_count)

untidy_response_tripadvisor <-
  numbered_response_tokens("tripadvisor_turkishairlines6846.rds", "review") %>%

  dplyr::select(id, post_number, review, quote, rating, date)


sentence_tokens <- function(data = untidy_response_facebook,
                            response_column = "comments",
                            group_by = "comment_number") {
  # English Dictionary
  qdapDictionaries::DICTIONARY[,1]

  en_word_comments <- data %>%
    dplyr::ungroup() %>%
    tidytext::unnest_tokens_("word", response_column) %>%
    dplyr::filter(word %in% qdapDictionaries::DICTIONARY[,1])

  en_word_sentence_comments <- en_word_comments %>%
    dplyr::group_by_("id", group_by) %>%
    dplyr::mutate(sentence = paste(word, collapse = " ")) %>%
    dplyr::distinct(sentence, .keep_all = TRUE) %>%
    dplyr::as_data_frame() %>%
    dplyr::mutate(sentence = iconv(sentence, to = 'latin1')) %>%
```

```
    dplyr::ungroup()

  # Sentence as tokens with post number and comment number

  en_word_sentence_comments %>%
    dplyr::select_("id", group_by, "sentence") %>%
    dplyr::ungroup() %>%
    tidytext::unnest_tokens(sentences, sentence, token = "sentences")
}

facebook_sentence_tokens <-
  sentence_tokens(data = untidy_response_facebook,
                  response_column = "comments",
                  group_by = "comment_number")

tripadvisor_sentence_tokens <-
  sentence_tokens(data = untidy_response_tripadvisor,
                  response_column = "review",
                  group_by = "post_number")

facebook_sentence_tokens

tripadvisor_sentence_tokens

tripadvisor_sentence_tokens
  dplyr::select(sentences)
```

Response sentiments

```
response_sentiments <- function(data, lexicon, group_by = sentiment) {
  data %>%
    dplyr::inner_join(get_sentiments(lexicon), by = "word") %>%
    dplyr::count(response_number, sentiment) %>%
    tidyr::spread(sentiment, n, fill = 0) %>%
    dplyr::mutate(sentiment = positive - negative) %>%
    dplyr::ungroup()
}

# bing
facebook_sentiments_bing <- response_sentiments(tidy_response_facebook, "bing")
tripadvisor_sentiments_bing <- response_sentiments(tidy_response_tripadvisor, "
bing")

# nrc
facebook_sentiments_nrc <- response_sentiments(tidy_response_facebook, "nrc")
tripadvisor_sentiments_nrc <- response_sentiments(tidy_response_tripadvisor, "n
rc")

#afinn
sentiments_afinn <- function(data) {
  data %>%
    dplyr::ungroup() %>%
    dplyr::inner_join(tidytext::get_sentiments("afinn"), by = "word") %>%
    dplyr::group_by(id, response_number) %>%
    dplyr::summarise(score = sum(score))
}
```

```
facebook_sentiments_afinn <- sentiments_afinn(tidy_response_facebook)
tripadvisor_sentiments_afinn <- sentiments_afinn(tidy_response_tripadvisor)

facebook_sentiments_bing %>%
  dplyr::select(response_number, negative, positive, sentiment)
```

```
## # A tibble: 1,588 x 4
##    response_number negative positive sentiment
##              <int>    <dbl>    <dbl>     <dbl>
##  1               1        0        1         1
##  2               2        0        1         1
##  3               3        0        1         1
##  4               3        1        0        -1
##  5               4        0        1         1
##  6               2        0        1         1
##  7               3        3        1        -2
##  8              18        0        1         1
##  9              21        0        2         2
## 10              23        0        1         1
## # ... with 1,578 more rows
```

```
tripadvisor_sentiments_bing
```

```
## # A tibble: 6,426 x 5
##    id          response_number negative positive sentiment
##    <chr>                 <dbl>    <dbl>    <dbl>     <dbl>
##  1 rn342674723               1        1        0        -1
##  2 rn342740773               2        1        3         2
##  3 rn342772345               3        4        1        -3
##  4 rn342937701               4        0        3         3
##  5 rn343011985               5        1        1         0
##  6 rn343117108               6        0        1         1
##  7 rn343117743               7        0        2         2
##  8 rn343141565               8        4        2        -2
##  9 rn343168849               9        0        4         4
## 10 rn343189478              10        0        3         3
## # ... with 6,416 more rows
```

```
facebook_sentiments_nrc
```

```
## # A tibble: 1,895 x 13
##    id      response_number anger anticipation disgust  fear   joy negative
##    <chr>             <int> <dbl>        <dbl>   <dbl> <dbl> <dbl>    <dbl>
##  1 904300…               1     0            0       0     0     2        0
##  2 904300…               2     0            0       0     0     1        0
##  3 904300…               3     0            0       0     0     1        0
##  4 904300…               4     0            0       0     0     0        0
##  5 904300…               1     0            1       0     0     0        1
##  6 904300…               4     0            0       0     0     0        0
##  7 904300…               2     1            2       0     1     1        0
##  8 904300…               3     0            0       0     0     0        3
##  9 904300…               4     0            0       0     1     0        0
## 10 904300…               5     0            1       0     0     0        0
## # ... with 1,885 more rows, and 5 more variables: positive <dbl>,
## #   sadness <dbl>, surprise <dbl>, trust <dbl>, sentiment <dbl>
```

```
tripadvisor_sentiments_nrc
```

```
## # A tibble: 6,796 x 13
##    id      response_number anger anticipation disgust  fear   joy negative
##    <chr>             <dbl> <dbl>        <dbl>   <dbl> <dbl> <dbl>    <dbl>
##  1 rn3426…               1     0            2       0     0     0        2
##  2 rn3427…               2     1            2       1     2     3        0
##  3 rn3427…               3     2            0       2     1     0        3
##  4 rn3429…               4     0            2       0     0     3        0
##  5 rn3430…               5     1            2       1     1     1        1
##  6 rn3431…               6     0            2       0     0     1        0
##  7 rn3431…               7     0            3       0     0     4        0
##  8 rn3431…               8     1            1       1     2     2        4
##  9 rn3431…               9     0            3       0     0     3        0
## 10 rn3431…              10     1            2       0     1     2        0
## # ... with 6,786 more rows, and 5 more variables: positive <dbl>,
## #   sadness <dbl>, surprise <dbl>, trust <dbl>, sentiment <dbl>
```

```
facebook_sentiments_afinn
```

```
## # A tibble: 1,647 x 3
## # Groups:   id [?]
##    id                              response_number score
##    <chr>                                     <int> <int>
##  1 90430042759_10155125620582760                 1     5
##  2 90430042759_10155125620582760                 2     3
##  3 90430042759_10155125620582760                 3     3
##  4 90430042759_10155125620582760                 7     2
##  5 90430042759_10155126945012760                 3     2
##  6 90430042759_10155126945012760                 4     2
##  7 90430042759_10155129495572760                 2     3
##  8 90430042759_10155129495572760                 3    -3
##  9 90430042759_10155129495572760                11    -1
## 10 90430042759_10155129495572760                18     1
## # ... with 1,637 more rows
```

```
tripadvisor_sentiments_afinn
```

```
## # A tibble: 6,139 x 3
## # Groups:   id [?]
##    id          response_number score
##    <chr>                 <dbl> <int>
##  1 rn342674723               1     0
##  2 rn342740773               2     7
##  3 rn342772345               3    -2
##  4 rn342937701               4     5
##  5 rn343011985               5    -3
##  6 rn343117108               6     2
##  7 rn343117743               7     5
##  8 rn343141565               8    -5
##  9 rn343168849               9     6
## 10 rn343189478              10     7
## # ... with 6,129 more rows
```

Sentiments to All Words Ratio

```
sentiment_token_ratio <- function(data, sentiment_type = "negative") {

  negative_sentiment <- get_sentiments("bing") %>%
    dplyr::filter(sentiment == sentiment_type)

  wordcounts <- data %>%
    dplyr::group_by(response_number) %>%
    dplyr::summarize(word = n())

  data %>%
    dplyr::semi_join(negative_sentiment) %>%
    dplyr::group_by(id, response_number) %>%
    dplyr::summarize(negativewords = n()) %>%
    dplyr::left_join(wordcounts, by = c("response_number")) %>%
    dplyr::mutate(ratio = negativewords/word) %>%
    dplyr::top_n(10) %>%
    dplyr::ungroup() %>%
    dplyr::arrange(desc(ratio))

}

facebook_negative_sentiment_ratio <-
  sentiment_token_ratio(tidy_response_facebook, "negative")

## Joining, by = "word"

## Selecting by ratio

tripadvisor_negative_sentiment_ratio <-
  sentiment_token_ratio(tidy_response_tripadvisor, "negative")

## Joining, by = "word"
## Selecting by ratio

facebook_positive_sentiment_ratio <-
  sentiment_token_ratio(tidy_response_facebook, "positive")

## Joining, by = "word"
## Selecting by ratio

tripadvisor_positive_sentiment_ratio <-
  sentiment_token_ratio(tidy_response_tripadvisor, "positive")

## Joining, by = "word"
## Selecting by ratio

facebook_negative_sentiment_ratio

## # A tibble: 192 x 5
##    id                            response_number negativewords  word  ratio
##    <chr>                                   <int>         <int> <int>  <dbl>
##  1 90430042759_101551815520227…              281             1     2 0.5
##  2 90430042759_101551815520227…              805             1     2 0.5
##  3 90430042759_101551815520227…              661             1     3 0.333
##  4 90430042759_101551815520227…              687             1     3 0.333
##  5 90430042759_101551815520227…              503             2    11 0.182
```

```
##  6 90430042759_101551815520227…              358             1     6 0.167
##  7 90430042759_101551815520227…              314             1     7 0.143
##  8 90430042759_101551815520227…              157             1     8 0.125
##  9 90430042759_101551815520227…              475             1    10 0.1
## 10 90430042759_101551815520227…               69             2    30 0.0667
## # ... with 182 more rows
```

```
tripadvisor_negative_sentiment_ratio
```

```
## # A tibble: 3,389 x 5
##    id          response_number negativewords  word ratio
##    <chr>                 <dbl>         <int> <int> <dbl>
##  1 rn541387435            4976             5     7 0.714
##  2 rn465760178            3008             4     8 0.5
##  3 rn525881971            4705             4     8 0.5
##  4 rn518927507            4384             5    11 0.455
##  5 rn513737468            3993             4     9 0.444
##  6 rn425622278            2284             9    21 0.429
##  7 rn557983424            5965             3     7 0.429
##  8 rn569818446            6477             3     7 0.429
##  9 rn530267095            4794             4    10 0.4
## 10 rn547414313            5547             4    10 0.4
## # ... with 3,379 more rows
```

```
numbered_response_tokens(
  "tripadvisor_turkishairlines6846.rds", "review") %>%
  dplyr::select(id, response_number = post_number, review,
                quote, rating, date) %>%
  dplyr::filter(response_number == 4976) %>%
  dplyr::select(review, id)
```

```
## # A tibble: 1 x 2
## # Groups:   id [1]
##   review                                                          id
##   <chr>                                                           <chr>
## 1 "the flight was very bad and the food very bad it so bad and … rn5413…
```

Most Frequent Sentiments

```
frequent_sentiments <- function(data) {
  data %>%
    dplyr::inner_join(tidytext::get_sentiments("bing")) %>%
    dplyr::ungroup() %>%
    dplyr::count(word, sentiment, sort = TRUE)
}

facebook_frequent_sentiments <-
          frequent_sentiments(tidy_response_facebook)

## Joining, by = "word"

tripadvisor_frequent_sentiments <-
          frequent_sentiments(tidy_response_tripadvisor)
```

```
## Joining, by = "word"
```

```
facebook_frequent_sentiments
```

```
## # A tibble: 383 x 3
##    word            sentiment     n
##    <chr>           <chr>     <int>
##  1 love            positive    328
##  2 nice            positive    155
##  3 beautiful       positive     87
##  4 congratulations positive     83
##  5 wow             positive     80
##  6 amaze           positive     73
##  7 free            positive     72
##  8 happy           positive     69
##  9 bless           positive     45
## 10 super           positive     37
## # ... with 373 more rows
```

```
tripadvisor_frequent_sentiments
```

```
## # A tibble: 1,310 x 3
##    word        sentiment     n
##    <chr>       <chr>     <int>
##  1 comfortable positive   1026
##  2 excellent   positive    927
##  3 nice        positive    816
##  4 friendly    positive    700
##  5 delay       negative    552
##  6 helpful     positive    472
##  7 clean       positive    406
##  8 free        positive    399
##  9 bad         negative    324
## 10 recommend   positive    318
## # ... with 1,300 more rows
```

Plot Frequent Sentiments Counts

```
plot_sentiment_count <- function(data) {
  data %>%
    dplyr::group_by(sentiment) %>%
    dplyr::top_n(10) %>%
    dplyr::ungroup() %>%
    dplyr::mutate(word = reorder(word, n)) %>%
    ggplot2::ggplot(ggplot2::aes(word, n, fill = sentiment)) +
    ggplot2::geom_col(show.legend = FALSE) +
    ggplot2::facet_wrap(~sentiment, scales = "free_y") +
    ggplot2::labs(y = "Frequent Sentiments",
                  x = "Frequency") +
    ggplot2::coord_flip()
}

plot_sentiment_count(facebook_frequent_sentiments)
```

```
## Selecting by n
```

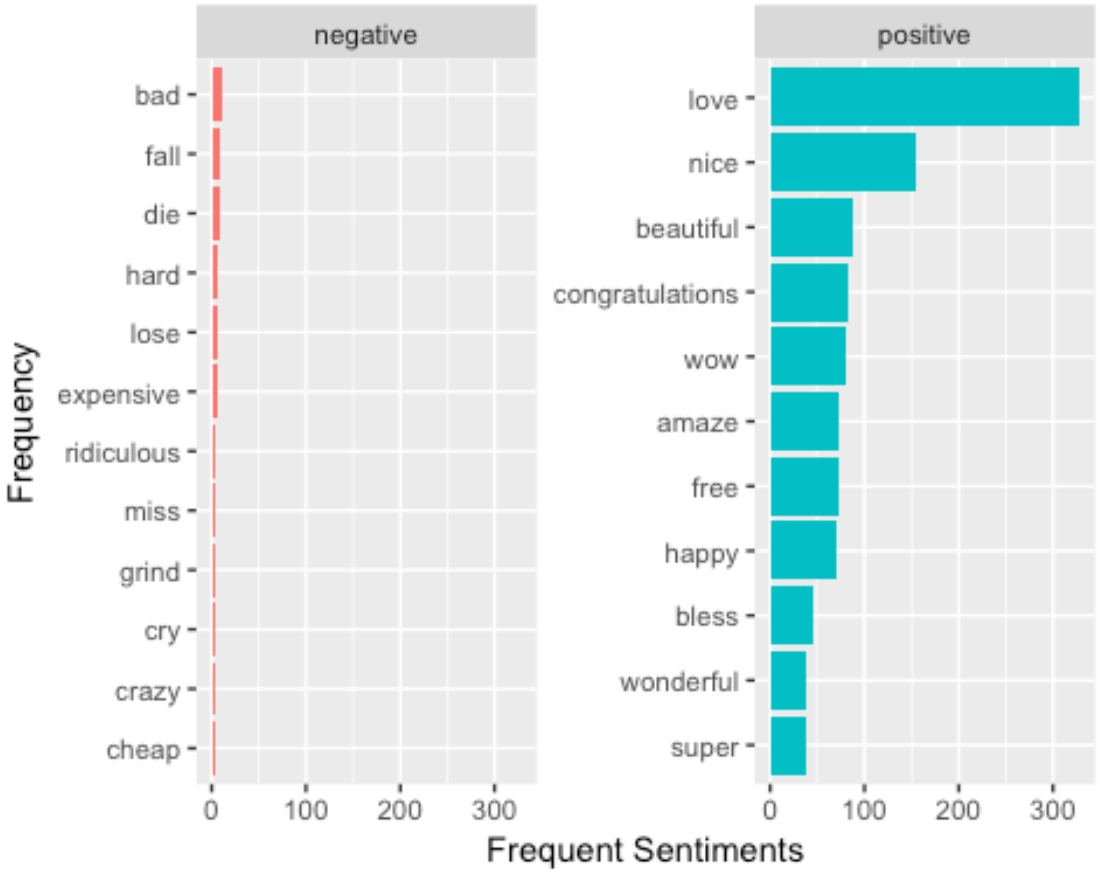


Figure 14: Facebook Sentiment Count Plot - Appendix

```
plot_sentiment_count(tripadvisor_frequent_sentiments)

## Selecting by n
```

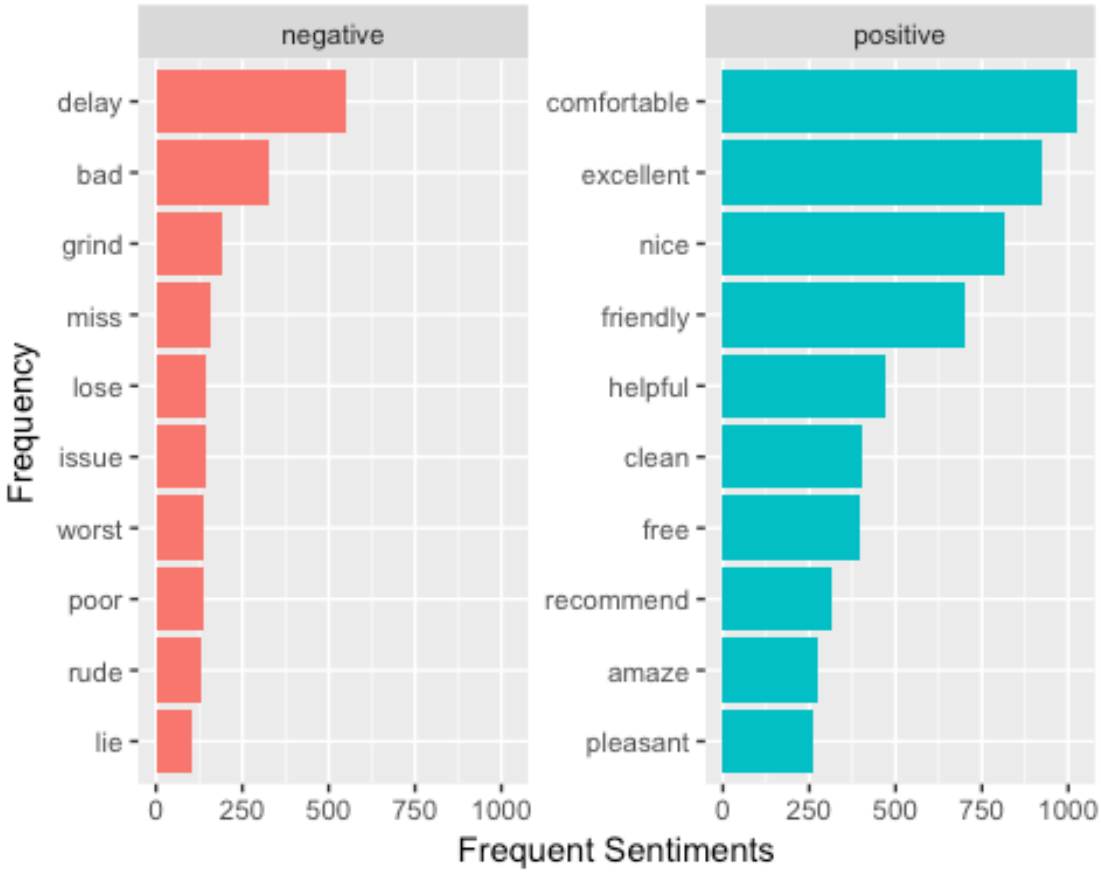


Figure 15: TripAdvisor Sentiment Count Plot - Appendix

Word Cloud

```
library(wordcloud)

## Loading required package: RColorBrewer

word_cloud <- function(data, max_words) {
  data %>%
    dplyr::ungroup() %>%
    dplyr::count(word) %>%
    with(wordcloud::wordcloud(word, n, max.words = max_words))
}

word_cloud(tidy_response_facebook, max_words = 50)
```

Figure 16: Facebook Wordcloud - Appendix

```
word_cloud(tidy_response_tripadvisor, max_words = 75)
```

Figure 17: TripAdvisor Wordcloud - Appendix

Sentiment Cloud

```
library(reshape2)

sentiment_cloud <- function(dataset) {
  dataset %>%
    dplyr::inner_join(tidytext::get_sentiments("bing")) %>%
    dplyr::count(word, sentiment, sort = TRUE) %>%
    reshape2::acast(word ~ sentiment, value.var = "n", fill = 0) %>%
    wordcloud::comparison.cloud(colors = c("#F8766D", "#00BFC4"),
                                max.words = 100)
}

sentiment_cloud(tidy_response_facebook)
```

```
## Joining, by = "word"
```

Figure 18: Facebook Sentiment Cloud - Appendix

```
sentiment_cloud(tidy_response_tripadvisor)
```

```
## Joining, by = "word"
```

Figure 19: TripAdvisor Sentiment Cloud - Appendix