2019

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

Table of Contents

List of Figures

ABSTRACT

In the era of modern Data science and Big Data, it is no longer a wonder to enable machine learning to understand human language and know what people are feeling and thinking with their surroundings. The term is called Sentiment Analysis or Opinion Mining which combines the power of natural language processing, text analysis and computational linguistics to classify subjective information or the emotional state of the writer/subject/topic. Instead of just identifying a positive/negative/neutral sentiment, we can also extract keywords that intensifies different emotions such as joy, excitement, frustration, fear etc. from the content.

In this project, Google Daily Search Trends are used to choose the topic on which various text analytics application will be applied. Google now processes over 40,000 search queries every second on average and over 3.5 billion searches per day [1]. To see what the world is looking for, there is a Trending Searches page addition to Google Trends that publish the most frequently searched terms along with their search volume and related news stories of the past 24-hour across various countries. The URL of daily search trend list is available at https://trends.google.com/trends/trendingsearches/daily?geo=US. For this project, the first trending search topic is selected and analyzed to create a text analytics visualization report. To get the most recent data about the search topic, various news articles related to the search topic and Twitter data source is used. All these data sources are publicly available content. This project represents a system that assigns sentiment scores and extracts key emotion associated with the opinion expressed in these news stories and Twitter posts on a certain trending search topic. Although full comprehension of natural language text remains well beyond the power of machines, the implemented statistical analysis of moderately simple sentiment indications can provide a meaningful quantitative summary of these large amount of qualitative information. The project is primarily implemented on Microsoft Power BI, Python and R programming platform. Power BI is a data visualization tool that supports a large range of data sources (virtually any data source) to load, transform and clean the data into a data model. A great feature of Power BI is we can connect to a web page and import its data into a dataset. In Power BI, the dataset is usually referred to as a Query or Table. In Power BI, first the raw text data about the most searched topic is captured from various news articles and Twitter feeds. The text data is used to apply various text analytics application by identifying the text polarity, tallying positive and negative words used in the text, extracting emotion category of the words etc. After conducting necessary analysis, a text analytics summary dashboard is created in Power BI Desktop. To generate the report with latest data, a scheduled refresh is configured on the dataset. Finally, the report is shared with power BI users associated with a UNCW or Office 365 email account.

Chapter 1

INTRODUCTION

1.1: Objectives

Regardless of the industry three words *data*, *insights* and *conversion*s are thrown around quite a bit in digital world. The *insights* derived from proper *data* analysis steer us towards ways of achieving increased *conversions*. For an organization, the quantitative data (e.g. metrics such as net promoter score, customer effort and satisfaction score on a range of 0-10 etc.) can be easily measured and displayed in charts or an Excel sheet to reach these conversations. But still little is known about the qualitative side (e.g. open answers) or the unstructured data. While quantitative data can be measured, qualitative data is not quite as straightforward to analyze it.

To analyze and understand large number of unstructured data like customer opinion, user feedback, product reviews, Text Analytics is used to derive meaning out of text and written communications. There are several different techniques used to analyze text and unstructured data. To quickly identify common topics (e.g. pricing, service, account, etc.) and issues that arise among customers, frequency of these topics can be counted. Sometimes a group of words can provide more insight than just one word alone. For example, when words like “costs”, “expensive” and “monthly” are grouped together, a business can conclude that some customers might think the monthly costs are too expensive for a certain product or services. Another form of applying text analytics techniques is Sentiment Analysis to gauge the severity of the text data based on positive, negative and neutral word usage as well as the sentiment associated with commonly used words.

Text analytics offers a viable service for managers at companies of all sizes and across all industries. Managers can retrieve data related to a business, its products and its services from blog posts, social networks, web polls and surveys and other online sources quickly and effortlessly. Although text analytics provides an excellent data collection service, this can be challenging for a manager who is conducting data mining for the first time. Working with text analytics is a cumbersome process and usually requires many man-hours to identify patterns and trends. The process can seem complicated at first and the software quite specialized, but it is no longer in the exclusive preview of data scientists. To conduct unstructured text analytics, it is critical to choose the right software with a user-friendly graphical interface and the ability to seamlessly interact with large data sets. you will need data scientists just to run the technical part of the software. Using Power BI and its ability to create stunning reports for text analytics from different sources, this task can be achieved in no time. The project is aimed to produce a fully automated graphical representation of text analytics application in an integrated environment with Power BI, Python and R programming languages. The focus is to assign sentiment scores to social media and news data, extract the intensities of different emotions (excitement, frustration,

happiness, sad, anger etc.) and represents most used words using a wordcloud. The result is published to power BI service to share with other users within the organization. Datasets is scheduled daily to generate the text analytics summary with the most recent social media and news text data.

1.2: Business Case of the Application

Text analytics offer huge opportunities for companies regardless of industry. Companies and individuals want to settle on better informed business decisions based off identifiable and quantifiable insight. With progressions in Text Analysis, companies can now be able to mine text to get insights and improve their service to prosper in the competitive market. There are countless number of industries that can be benefitted by applying text analytics to collate and act on company's data. Perhaps, the application is most suited on market research to figure out the media space that a company lives in and how it is received by its audience.

Text feedback is the closest a company got to a one to one conversation with every customer, every citizen, and every employee. In free text, customers can express what they really care about a product or service and why, unconstrained by the questions company decided to ask them. Many companies receive thousands of rows of feedback per day. Even with just tens or hundreds of feedback items per day, manually uncovering themes out of these feedbacks can be daunting and difficult. This situation is where automated text analytics can help in sorting out the key topics talked about and reveal the general sentiment per topic. This project incorporates an end to end text analytics application from data collection to publishing and scheduling the result in a regular basis.

Any e-commerce company can collect their customer experience data on its forums and websites by scraping all these reviews and feedback. Data collector module in this system integrates web scraping from multiple online news sources using Python programming language. The web scraping module can be easily modified by any organization according to their specific website's HTML structure. Furthermore, necessary data or feedbacks can also be collected efficiently. These extracted text content of customer feedbacks can be passed to the text analytics module to understand how favorable users found a product or service and what customers talked most about it on the Internet.

In recent years the explosion of social media has made available an unprecedented amount of real-time data to measure public opinion. According to the New York Times, more than one billion election-related tweets were posted on Twitter during the last presidential election, from the first presidential debate until election day [36]. As a social media platform, Twitter has emerged as a popular communication channel between leaders of contesting parties and voters. During election campaigns, the contesting parties and voters express their opinions on Twitter generating huge amount of unstructured data. Interested parties can use these data to monitor

election campaigns, gauge political polarization and negative campaigning, and even forecast election results combined with traditional off-line election polling at any point in time. The Text Analytics module of this system can help answer some critical questions like which issues are getting more attention from the public, what candidates are garnering more positive / negative sentiment, people's reaction to events during the election campaign in real time etc.

The following sections are structured as follows: Chapter 2 will discuss the Power BI platform and Text Analytics application that are used in building the system. Chapter 3 will go into how the system was designed initially before the implementation. Chapter 4 will provide a demonstration of data collection, integration of Text Analytics module and creating visualizations with the results obtained from these text analytics applications, as well as publishing these visualizations to Power BI service to share and schedule the system in a regular basis. Chapter 5 will discuss the benefits the system unlashes on users to perform daily text analysis operation with Power BI platform. The challenges and adjustments have been discussed in chapter 6. Finally, chapter 7 describes some of the limitations the system includes and how the project can be extended in future.

Chapter 2

SYSTEM ANALYSIS

2.1: Introduction to Power BI

Power BI is a Data Visualization and Business Intelligence (BI) tool provided by Microsoft. It supports a wide variety of data sources to load, transform and clean the data into a data model. The aggregated data model is later used to create charts or graphs to provide visuals of the data and discover what is important for an organization. All these interactive dashboards and BI reports can be shared with other Power BI users within the organization.

2.1.1: Common Uses of Power BI

Data is useful only if people can access it, understand it and act from it. With the advancement of cloud computing, APIs and various open source data initiatives, the bottleneck has moved from accessing data to understanding it. Power BI is used to present and visualize data so better decisions can be made from it. Below is a summary of Power BI report creation process:

Step 1: Find the data.
Step 2: Load the data in Power BI and process it.
Step 3: Model the data and perform necessary transformations.
Step 4: Perform visualizations, branding and design.

The data models created from Power BI can be used in several ways for organizations, including telling stories through charts and data visualizations within the data. Power BI's Q&A (question and answer) features can also answer questions in real time and help with forecasting to make sure departments meet business objectives. Company administrators can get more insight into how departments are doing from evaluating a Power BI report.

2.1.2: Architecture of Power BI

The Power BI ecosystem comprises of seven key components [3].

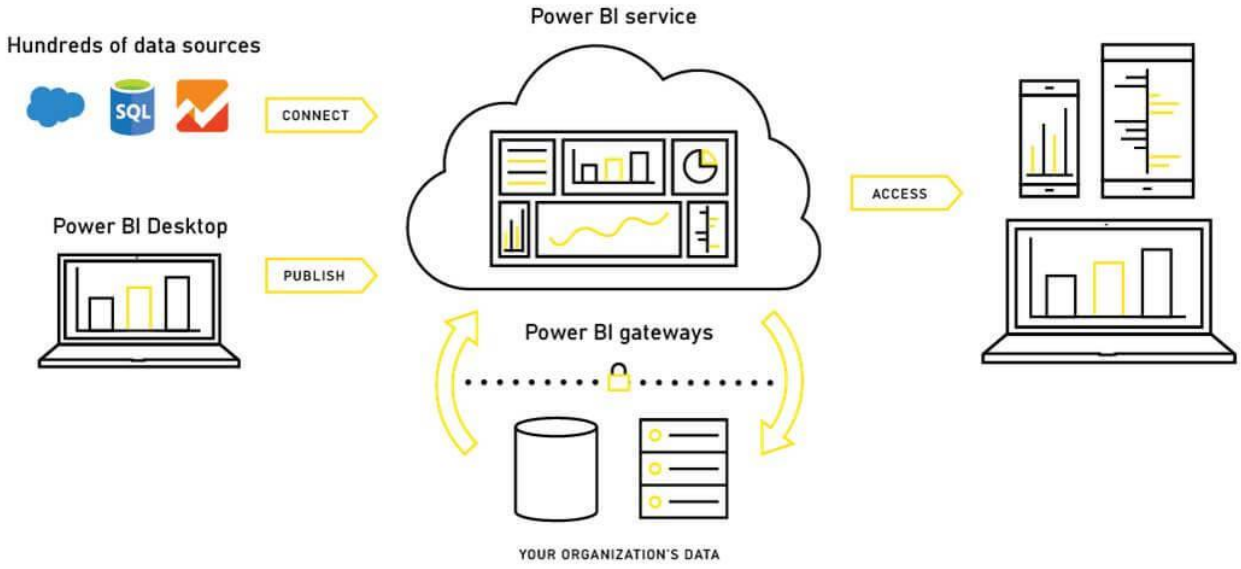


Figure 1: Overview of a Power BI architectural Diagram. [4]

i. **Power BI Desktop:** The Windows-desktop-based application for PCs and desktops, is used to create and publish reports to the Service.
ii. **Power BI Service:** The SaaS (software as a service) based online service is used to publish the visualization reports.
iii. **Power BI Mobile:** With Power BI mobile apps, users can stay connected to their data from anywhere. Currently the apps are available for Windows, iOS, and Android platform.
iv. **Power BI Gateway:** Power BI on-premises Gateways are used to refresh data by connecting on-premises data sources without the need to move the data. It allows to query large datasets and benefit from the existing investments.
v. **Power BI Embedded:** Power BI REST (Representational State Transfer) API is used to build dashboards and reports into custom applications that serve Power BI users, as well as non-Power BI users.
vi. **Power BI Report Server:** An On-Premises Power BI Reporting solution for companies that won't or can't store data in the cloud-based Power BI Service.

2.1.3: Comparison of Power BI with Other Business Intelligence Tools

Today, every organizations are being flooded with new data from all directions. Business Intelligence (BI) covers all the activities necessary for a company to turn this raw information into actionable knowledge. In search for a BI or data visualization tool, the choice come across the two front-runners in the category: Power BI and Tableau.

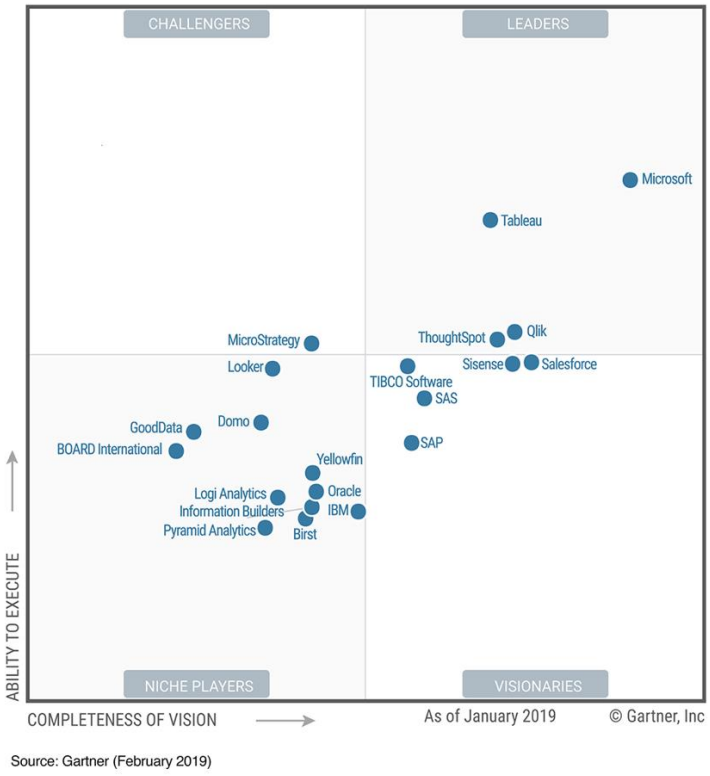


Figure 2: Magic Quadrants for Analytics & BI Platforms [5]

Feature to feature they are very similar products; both BI tools have their own strengths and weaknesses, and each will suit businesses based on requirements. In this section, Tableau and Power BI is compared using some key parameters including usability, setup, price, support, maintenance, self-service feature and support of different data types. Below are some of the important key Differences Between Power BI vs Tableau.

i. Visualization: Tableau built their product on the philosophy of "seeing and exploring" data for over a decade. When visualization is the prime focus of the analytics, Tableau is the right tool for the companies as it requires no coding language. Users can create charts, scatter plots using drag and drop method, and Tableau also does not restrict the number of data points. On contrary, Power BI has around 3500 data points for drilling down across dataset and conduct an analysis [35]. Power BI also let users to create their own custom visualization that allows to explore and display data in captivating ways.

ii. ETL Data Discovery: Power BI has a robust set of tools for ETL (Extract, Transform and Load) and data discovery when compared to Tableau. Data prep is 80-90% of any reporting requirement [6]. Visualization becomes the easy part when there is a super powerful data model. For data shaping, Power BI offers a self-documenting query editor with many options in the ribbon itself. Power query uses M language and DAX (Data Analysis Expression). Tableau does not have anything quite like that on this side.

iii. Adaption to the Ecosystem: Because Power BI was originally a mostly Excel-driven product, it has a higher adoption rate among newbies. Besides, it is tightly integrated with the Microsoft ecosystem (O365, Azure, Power APP, Power Flow, Share Point) which makes it more compelling to share the end results with everyone within the organization.

iv. Q&A capability: Sometimes the fastest way to get an answer from data is to ask a question using natural language. Power BI is the only product that embraces a Q&A-based natural language interface and Cortana-driven speech capability. We can simply type a question for example, "what were total sales last year" and appropriate visualization will be explored about the data in Power BI [23].

v. Price: Power BI's biggest strength is its rock-bottom cost and value. Personal version of Tableau cost $420 and the Professional version will cost $840 annually [7]. For a product that is totally comparable to the category leader, the Pro version of Power BI will cost only $120. It also comes with a free Power BI desktop version which itself covers a lot more than Tableau.

2.1.4: Users of Power BI

Though Power BI is a self-service BI tool that brings data analytics to employees, it is mostly used by the greater audience of data analysts, data scientists, decision makers, and business intelligence professionals who create the data models before disseminating reports throughout the organization. Power BI's user interface is intuitive for users familiar with Excel and its deep integration with other Microsoft products makes it a very versatile self-service tool that requires little upfront training [2]. Therefore, users without an analytical background can still navigate Power BI and create reports.

2.2: Introduction to Text Analytics

The unifying theme behind Text analytics is the need to "turn text into numbers". It refers to the process of transforming unstructured text into a structured, numerical format so powerful analytical algorithms can be applied to on these text ranging from individual words to entire document database. Text analysis works by breaking apart sentences and phrases into their components, and then evaluating each part's role and meaning using complex software rules and machine learning algorithms. Text analytics forms the foundation of numerous natural language processing (NLP) features, including named entity recognition, categorization, and sentiment analysis. This project is primarily focused on applying sentiment analysis with news and social media data.

2.2.1: Sentiment Analysis

The term Sentiment Analysis within NLP features aim to answer four questions [8]:

- Who is talking?
- What are they talking about?
- What are they saying about those subjects?
- How do they feel?

So, in broad term, Sentiment Analysis (SA) or Opinion Mining is the process to determine whether a piece of writing is positive, negative or neutral as well as the emotional tone behind these words. With the rapid advancement into the era of widespread internet access, publicly available information over Internet is constantly growing. It's estimated that 80% of the world's data is unstructured [11] and comes from mostly text data, like emails, support tickets, chats, social media, surveys, articles, and documents. Data analysts and other professionals use sentiment analysis tools to derive useful information and context-rich insights from this sea of unstructured text in an efficient and cost-effective way than manually sorting through thousands of tweets, customer support conversations, or customer reviews and online news content.

2.2.2: Sentiment Analysis Scope

Sentiment analysis can be applied at different levels of scope [9]:

- Document level sentiment analysis obtains the sentiment of a complete document or paragraph.
- Sentence level sentiment analysis obtains the sentiment of a single sentence.
- Sub-sentence level sentiment analysis obtains the sentiment of sub-expressions within a sentence.

2.2.3: Types of Sentiment Analysis

Different types of sentiment analysis use different strategies and techniques to identify the sentiments of input text data. These SA tools range from systems that focus on polarity (positive, negative, neutral) to systems that detect feelings and emotions (angry, happy, sad, etc.) or identify intentions (interested v. not interested).

1. Fine-Grained Sentiment Analysis

It is based on the idea that an opinion consists of a sentiment (positive or negative) and a target (of opinion) [22]. It gives more precise results about the level of polarity of the opinion, so instead of just talking about positive, neutral, or negative opinions it provides the following scoring categories:

- Very positive
- Positive
- Neutral
- Negative
- Very negative

For example, mapped onto a 5-star rating in a review, e.g.: Very Positive = 5 stars and Very Negative = 1 star. Fine -grained sentiment analysis also provide different flavors of polarity by identifying if the sentiment is associated with a feeling, such as, anger, sadness, or worries (i.e. negative feelings) or happiness, love, or enthusiasm (i.e. positive feelings).

2. Emotion Detection

Emotion detection aims at detecting emotions like, happiness, frustration, anger, sadness, etc. from a given text data [22]. Many emotion detection systems resort to lexicons (the lists of most popular positive and negative words databases and the emotions they convey) or complex machine learning algorithms. One of the downsides of lexicon-based sentiment analysis is that the way people express their emotions varies and so do the lexical items they use. Some words that would typically express anger (like 'kill' e.g. your customer support is killing me) might also express happiness in some texts (e.g. you are killing it).

3. Aspect-Based Sentiment Analysis

Aspect-Based Sentiment Analysis identify and determine the sentiment towards specific aspects in text [22]. For example, when analyzing a review about a hotel we might be interested in not only whether the review is talking about a positive, neutral, or negative polarity but also which aspects or features (staff, beds, location etc.) of the hotel people talk about.

4. Intent Analysis:

Intent analysis detects the user's intention behind a message rather than what people say with that message [22]. It identifies whether a text relates an opinion, news, marketing, complaint, suggestion, appreciation or query. For examples:

"Your customer support is a disaster. I've been on hold for 20 minutes".
"I would like to know how to replace the cartridge".
"Can you help me fill out this form?"

In this project, the first two types of Fine-Grained and Emotion Detection sentiment analysis is used on the input text data collected from social media platform and news article sources.

Chapter 3

Project System Design (as Proposed)

The project involves the usage of Microsoft Power BI, Python and R programming languages to collect and analyze text data from various News Articles and Twitter application platform. The objective of the project is to find the polarity of the words retrieved from these sources about a search topic.

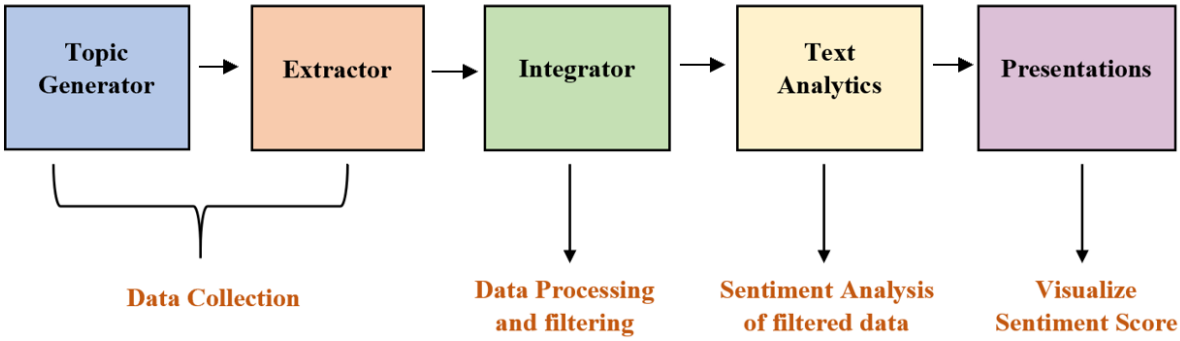


Figure 3: The Component Architecture of Proposed Framework

Each step in the framework involves several sub-tasks.

1. Topic Generator:

To select the Topic for sentiment analysis, the proposed experimental framework uses Google Daily Search Trend data. With Power BI Desktop, we will connect the google daily search trend web page (https://trends.google.com/trends/trendingsearches/daily?geo=US) to power BI and import the trending search topics. A dataset of search topics is created in Power BI and can be refreshed either on demand or scheduled. Upon refreshing data each time, the dataset is updated from the source. The Refreshed dataset is used as a source for further steps.

Figure 4: Topic Generator Details

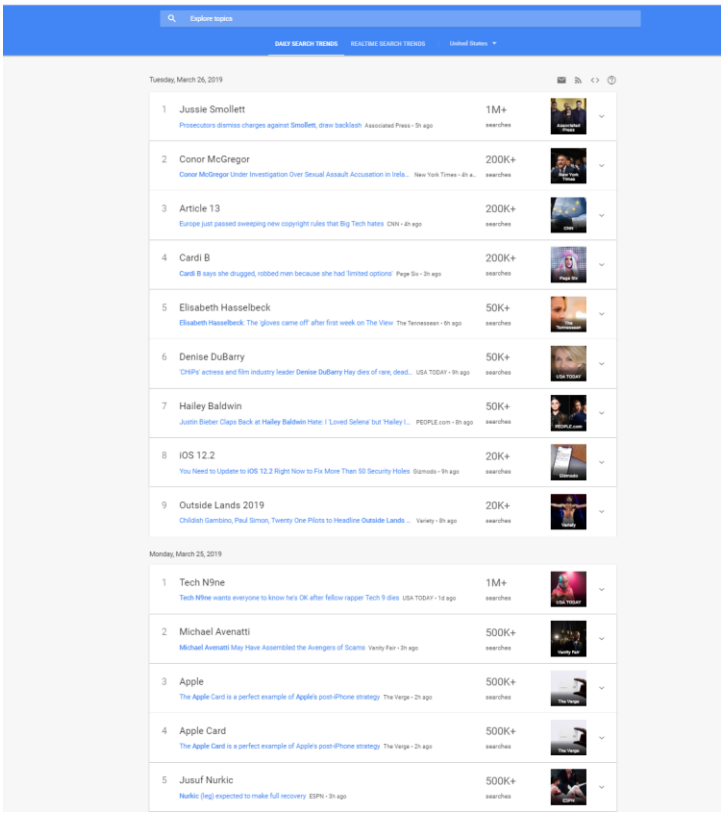


Figure 4: Screenshot of Google Daily Search Trend Web Page at March 26, 2019

	Title	No_of_Search	News Source	Publication_Date	Publication_Time
1	Jussie Smollett	1000000	Associated Press	3/26/2019	9:00:00 AM
2	Conor McGregor	200000	New York Times	3/26/2019	12:00:00 AM
3	Article 13	200000	CNN	3/26/2019	7:00:00 AM
4	Cardi B	200000	Page Six	3/26/2019	1:00:00 PM
5	Elisabeth Hasselbeck	50000	The Tennessean	3/26/2019	8:00:00 AM
6	Denise DuBarry	50000	USA TODAY	3/26/2019	9:00:00 AM
7	Hailey Baldwin	50000	PEOPLE.com	3/26/2019	3:00:00 PM
8	iOS 12.2	20000	Gizmodo	3/26/2019	7:00:00 AM
9	Outside Lands 2019	20000	Variety	3/26/2019	4:00:00 PM

Figure 5: Extracted Topic in Power BI

2. Data Extractor:

The Extractor Module scrape the unstructured text data from multiple data sources. The data sources are:

i. Google News Article: To get the most recent information about a search query, the project will be using the news articles that appears under NEWS tab of a Google search. For scraping the news articles, Python programming language is used. Because of its rich ecosystem, a Python script can be easily integrated to Power BI to import data and use powerful Python visualization inside Power BI.

Python is an open source programming language, there are many libraries available to perform web scraping function. For scraping, the following two Python modules are used:

- Urllib2**:** A Python module used for fetching URLs from a web page.
- BeautifulSoup: A Python library for pulling data out of HTML and XML files. We can extract tables, lists, paragraph and put filters to import information from web pages. It works with HTML parser (parser takes input in the form of a sequence of tokens or program instructions and builds a data structure in the form of a parse tree or an abstract syntax tree) included in Python's standard library to provide idiomatic ways of navigating, searching, and modifying the parse tree. It also supports several third-party Python parsers like LXML parser.

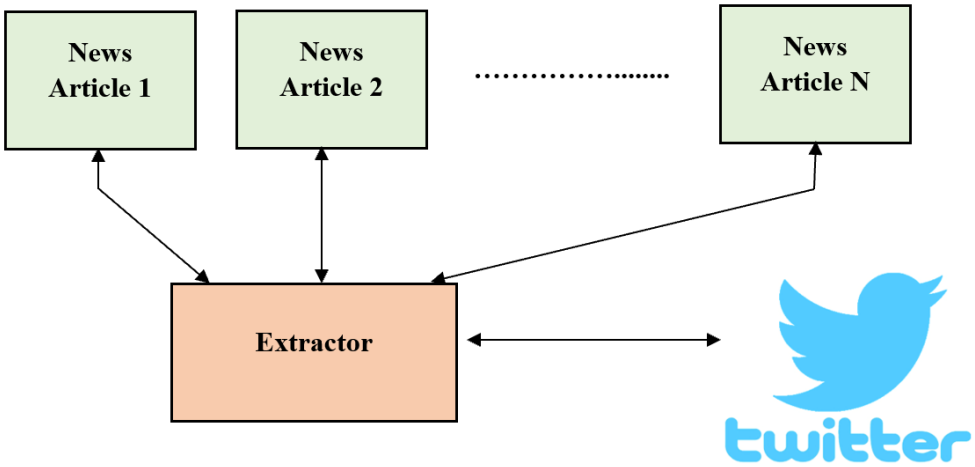


Figure 6: Block Diagram of Data Extractor Segment

ii. Twitter: Twitter is an online news and social networking service on which users post and interact with messages known as "tweets". For this project, this is more suitable than any other social media platform to pull-out real-time data of public opinion on the search topic. The amount of publicly available relevant data is much larger for Twitter as compared Facebook or Instagram. Raw tweets is retrieved using R scripts with Twitter streaming API.

3. Integrator:

Data integration phase involves combining data from multiple news articles and Twitter platforms to provide a unified view of the data collected for a search trend. In power BI desktop, a Query is created with a dataset for every new data Source connection. To meet the visualization requirement of the data, query can be edited in the Query Editor, where step-by-step instructions are provided to transform and shape the data. The original data source is not

affected, only this view of the data is adjusted, or shaped. In Query Editor data can be also combined from two or more data sources to consolidate them into one useful query. After getting the text data related to the selected keyword, the next step will be cleaning and preprocessing of the text to avoid feeding unnecessary data into the analysis. To clean the data, one of the most common process is to "tokenize" it which refer splitting the raw input text into individual words. Data is further filtered by removing any stop words (am, is, are, the, etc.) numbers, punctuations, special characters, Unicode characters, URL links, extra spaces, etc.

4. Text Analytics:

The text Analytics module of the proposed framework performs the actual sentiment analysis on the data collected from multiple news articles and Twitter post. Below are some of the possible ways to do the text analysis are described.

Text Analytics API:

The Text Analytics API is a cloud-based service which is a part of Microsoft Azure Cognitive Services (a collection of machine learning and AI algorithms in the cloud). The API provides advanced natural language processing over raw unstructured text. Currently it includes four main functions: sentiment analysis, key phrase extraction, language detection, and entity linking [12]. For the project deliverable, the following two function can be used.

i) Sentiment analysis: The API analyze raw text data and returns a numeric score between 0 and 1. Scores close to 1 indicate positive sentiment, and scores close to 0 indicate negative sentiment.

ii) Key Phrase Extraction: The API returns a list of strings denoting the key talking points in the input text. The list of strings will be later used to generate a word cloud.

Data Limitation for Text Analytics API Service:

Limit	Value
Maximum size of a single document	5,120 characters
Maximum size of entire request	1 MB
Maximum number of documents in a request	1,000 documents

The rate limit is 100 calls per minute. If the API is request more than 100 times within a minute, it will return an empty document. But we can submit a large quantity of documents in a single call (up to 1000 documents) [13].

Sentiment Analysis in R:

When it comes to Text Analysis, R has a wide variety of useful packages in understanding and extracting insights from the unstructured qualitative text data. After applying sentiment analysis strategy, the quantitative feedback can be treated as structured one.

Package **RSentiment**:

RSentiment is a dictionary-based text analysis library that analyze sentiment and assigns score to the following categories of sentiments: Positive, Negative, Very Positive, Very Negative, Neutral (0) and Sarcasm (99). For a vector of sentences, it counts the number of sentences in each category of sentiment. In calculating the score, negation and various degrees of adjectives are taken into consideration. It deals only with English sentences [14].

Package **sentimentr:**

sentimentr is an augmented dictionary lookup because sometimes a simple dictionary lookup may not be modeling the sentiment appropriately**.** Unlike other packages, sentimentr considers valence shifters which affect the polarity of the text data [15]. The common valence shifters are:

- Negators (“I do **not** like it.")
- Amplifiers / Intensifiers ("I **really** like it.")
- De-Amplifiers (I **hardly** like it."),
- Adversative conjunctions (I like it **but** it's not worth it)

In the case of negators and adversative conjunctions the entire sentiment of the clause may be reversed or overruled. It calculates the sentiment score on a range of 0 -1(negative) to 1(positive) where 0 represents neutral.

Package **syuzhet:**

The package comes with four sentiment dictionaries include "syuzhet" (default), "afinn", "bing" and "nrc" [17]. The `get_nrc_sentiment` function of this package implements NRC Emotion lexicon developed by, Saif Mohammad [16]. The NRC emotion lexicon is a list of words and their associations with eight emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive). The `get_nrc_sentiment` function returns a data frame in which each row represents a sentence from the original file.

5. Presentation:

Presentation is the fifth and last phase of the proposed framework. In this step, a dashboard is created with a sentiment meter of polarity score, wordcloud of most common words and a bargraph of different emotion categories associated with news and social media data in Power BI. Following three visualizations were proposed to include in final report.

Figure 7: Sample visualization of Sentiment Meter, WordCloud and Emotion Bargraph

A sentiment meter to graphically indicate the general interpretation (positive or negative) about a trending search topic. A Word Cloud visually interprets the most prominent phrases from a given text data. The importance of each word is visualized by its size and color. The more a specific word appears in the textual data source, the bigger and bolder it appears in the word cloud.
Finally, A graph of emotion classes represents the emotional tone behind the sentences from a viewpoint of anger, anticipation, disgust, joy, sadness, surprise. This kind of visualization is mostly applicable for social media data which is tweet post in this project.

Chapter 4

System Architecture (as Implemented)

Even though a good number of studies have been done to analyze Twitter data, I do not see any work to evaluate text analytics by relying on Twitter and News data in together. This chapter represents a more detailed work breakdown throughout the project. The chapter describes the process of capturing text data from various news articles and Twitter feeds, identifying the polarity, tallying positive and negative words in the data, and conducting further analysis to create a text analysis summary dashboard. The dashboard is later published to Power Bi service to share with other users.

4.1: Extracting Topic list from Google Daily Search Trend URL

To select a topic on which various text analytics application will be performed, a primary dataset of google trending search topics is created in Power BI. A great feature of Power BI is we can connect to a web page and import its data into Power BI Desktop. In Power BI Desktop, select Get Data > Web from the Home ribbon.

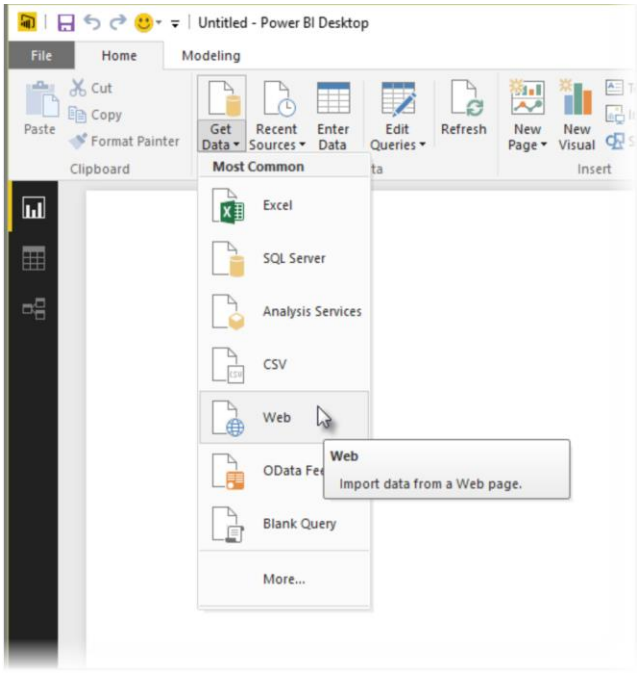


Figure 8: Importing Data from Web source in Power BI

A dialog appears, asking for the URL of the web page from which to import data. In this step, Google daily search trend web page is connected to import the trending search topics.

Figure 9: Connecting a webpage URL to Power BI

Once Power BI desktop connects to the URL, the navigator window splits into two parts: a list of objects on the left and data preview on the right. In the list of objects, the first object is Document; the rest are tables that Power BI found on the page. After clicking OK, the navigator window brings the data to Power Query Editor. From here, drill down transformation is applied to **Table** objects till our desired data table is reached.

To find out the desired information, the XML source of the URL is reviewed in Google Chrome.

Figure 10 : XML Structure of Trending Search RSS Feed

From the above XML structure, information about a trending search topic is nested under *item* tag. In Power Query Editor, when the ***Table*** object is clicked, it expands new rows with *channel, title, description, link, atom:link* and *item* columns.

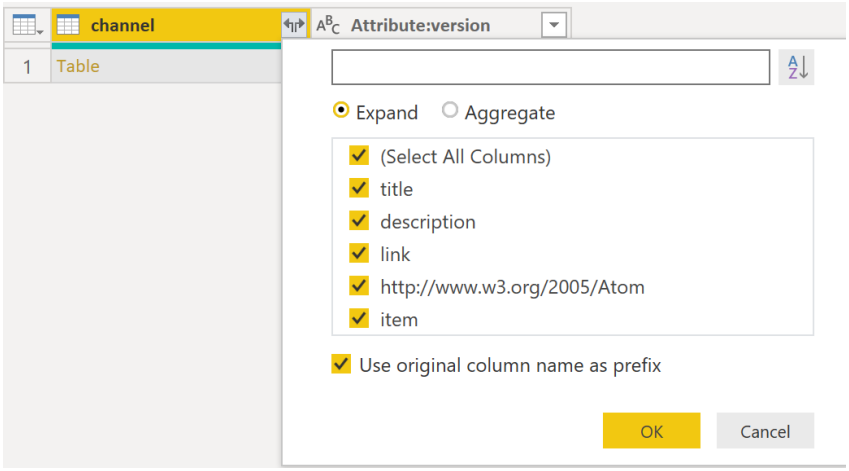


Figure 11 : Expanding Table Object in Power Query Editor

As the desired information is nested in item tag, only item column is selected when expanding the table object. As, we are only interested in title, publication date and approximate search traffic about a search topic the following column are expanded in next step.

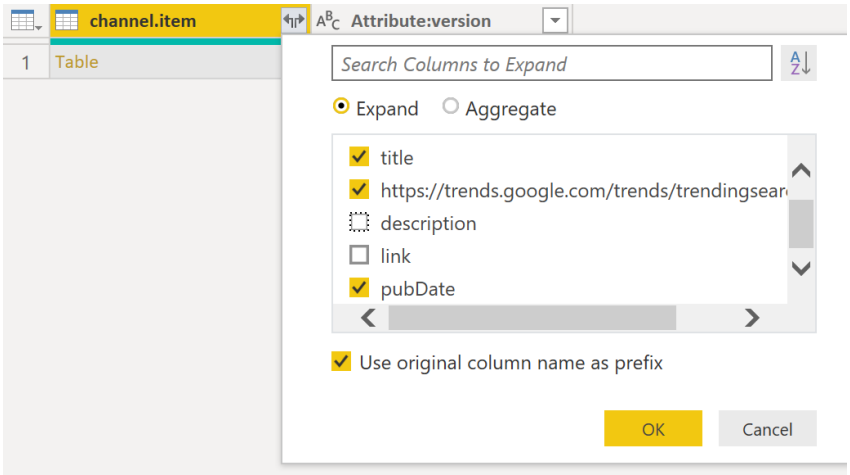


Figure 12 : Expanding Table Object in Power Query Editor - Step 2

After expanding the required columns, Publication Date column is splitted by date and time using following M (programing language used in Power BI) formula

```
= Table.SplitColumn(#"Removed Columns2", "Publication Date", Splitter.SplitTextByDelimiter
    ("2019", QuoteStyle.Csv), {"Publication Date.1", "Publication Date.2"})
```

	Title	approx_traffic(# of Search)	Publication Date
1	Bomb cyclone	20,000+	Wed, 27 Nov 2019 00:00:00 -...
2	Bomb cyclone	20,000+	Wed, 27 Nov 2019 00:00:00 -...
3	Dolly Parton	500,000+	Tue, 26 Nov 2019 18:00:00 -0...
4	Dolly Parton	500,000+	Tue, 26 Nov 2019 18:00:00 -0...
5	Real Madrid	200,000+	Tue, 26 Nov 2019 12:00:00 -0...
6	Real Madrid	200,000+	Tue, 26 Nov 2019 12:00:00 -0...
7	Knives Out	200,000+	Tue, 26 Nov 2019 17:00:00 -0...
8	Knives Out	200,000+	Tue, 26 Nov 2019 17:00:00 -0...
9	Duke basketball	200,000+	Tue, 26 Nov 2019 20:00:00 -0...
10	Duke basketball	200,000+	Tue, 26 Nov 2019 20:00:00 -0...
11	Clippers vs Mavericks	200,000+	Tue, 26 Nov 2019 19:00:00 -0...
12	Clippers vs Mavericks	200,000+	Tue, 26 Nov 2019 19:00:00 -0...
13	Albania	100,000+	Tue, 26 Nov 2019 07:00:00 -0...
14	Albania	100,000+	Tue, 26 Nov 2019 07:00:00 -0...
15	Juventus	100,000+	Tue, 26 Nov 2019 13:00:00 -0...
16	Juventus	100,000+	Tue, 26 Nov 2019 13:00:00 -0...
17	Weather radar	100,000+	Tue, 26 Nov 2019 06:00:00 -0...
18	Godfrey Gao	100,000+	Tue, 26 Nov 2019 22:00:00 -0...
19	Godfrey Gao	100,000+	Tue, 26 Nov 2019 22:00:00 -0...
20	Tulsi Gabbard	100,000+	Tue, 26 Nov 2019 23:00:00 -0...
21	Tulsi Gabbard	100,000+	Tue, 26 Nov 2019 23:00:00 -0...
22	Tina Turner	50,000+	Tue, 26 Nov 2019 12:00:00 -0...
23	Tina Turner	50,000+	Tue, 26 Nov 2019 12:00:00 -0...
24	The Weeknd	50,000+	Tue, 26 Nov 2019 13:00:00 -0...
25	The Weeknd	50,000+	Tue, 26 Nov 2019 13:00:00 -0...

All the applied steps are listed in the query settings pane of Power Query Editor. All the additional shaping steps are applied to a query, are captured in the **Applied Steps** section. These steps can be renamed, deleted by right-clicking the step in the **Applied Steps** section, and choosing from the menu that appears. All query steps are carried out in the order they appear in the **Applied Steps** pane.

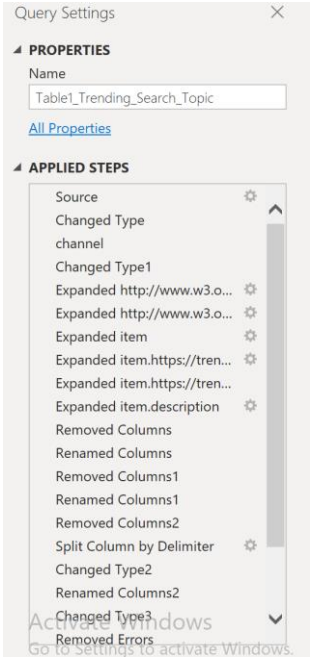


Figure 13: Applied Steps in Power Query Editor

Following is a snapshot of trending search topics after applied the above transformations.

	Title	Approx_Traffic(# of Se...	Publication Date	Publication Time
1	Bomb cyclone	20,000+	11/27/2019	3:00:00 AM
2	Dolly Parton	500,000+	11/26/2019	9:00:00 PM
3	Real Madrid	200,000+	11/26/2019	3:00:00 PM
4	Knives Out	200,000+	11/26/2019	8:00:00 PM
5	Duke basketball	200,000+	11/26/2019	11:00:00 PM
6	Clippers vs Mavericks	200,000+	11/26/2019	10:00:00 PM
7	Albania	100,000+	11/26/2019	10:00:00 AM
8	Juventus	100,000+	11/26/2019	4:00:00 PM
9	Weather radar	100,000+	11/26/2019	9:00:00 AM
10	Tina Turner	50,000+	11/26/2019	3:00:00 PM
11	The Weeknd	50,000+	11/26/2019	4:00:00 PM
12	Sadie Robertson	50,000+	11/26/2019	3:00:00 PM
13	Sam Darnold	50,000+	11/26/2019	3:00:00 PM
14	Omarion	50,000+	11/26/2019	9:00:00 PM
15	Charlie Brown Thanksgiving	50,000+	11/26/2019	9:00:00 AM
16	Ilhan Omar	50,000+	11/26/2019	10:00:00 AM
17	Michigan State basketball	50,000+	11/26/2019	1:00:00 PM
18	Daisy Ridley	50,000+	11/26/2019	11:00:00 AM

Figure 14 : Table of Trending Search Topic Lists

For applying various text analytics application to one topic at a time, only the first row with the most searched items is selected in the dataset. This dataset is used as a source table for further analysis.

	Title	Approx_Traffic(# of Se...	Publication Date	Publication Time
1	Bomb cyclone	20,000+	11/27/2019	3:00:00 AM

Figure 15: Trending Search Topic

4.2: Capturing Twitter Feeds into Power BI

In order to create a Twitter REST API to capture tweets, I needed to take following steps.

4.2.1: Getting Twitter API Keys

Twitter REST API allows the retrieval of tweets and related information filtered by keywords. To obtain a twitter REST API, it requires to register a developer account. Once, a developer account is created, a new application is created with some application details filling in the following form.

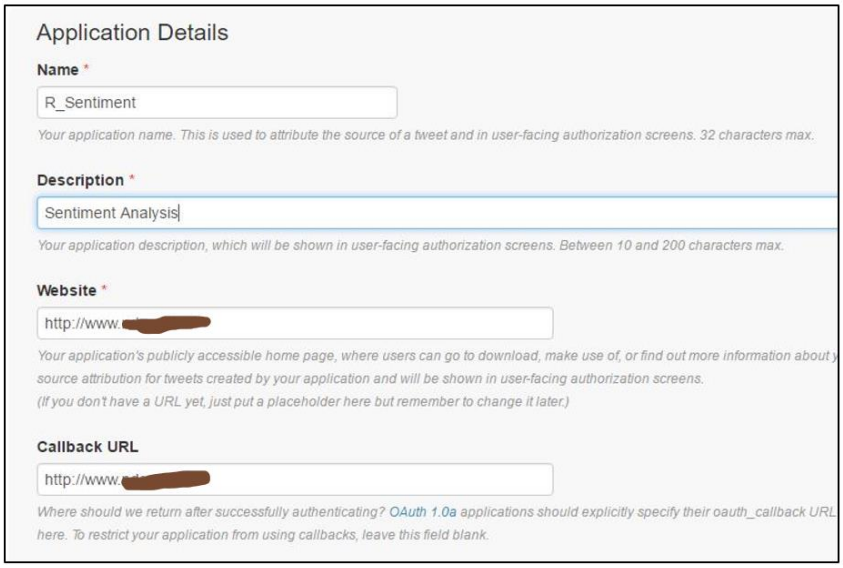


Figure 16: Creating an application for Twitter Developer Account

Once the application is complete, it returns user credentials with four keys: API key, API Secret Key, Access Token, and Access Token Secret that allow the newly created Twitter application to read Twitter information.

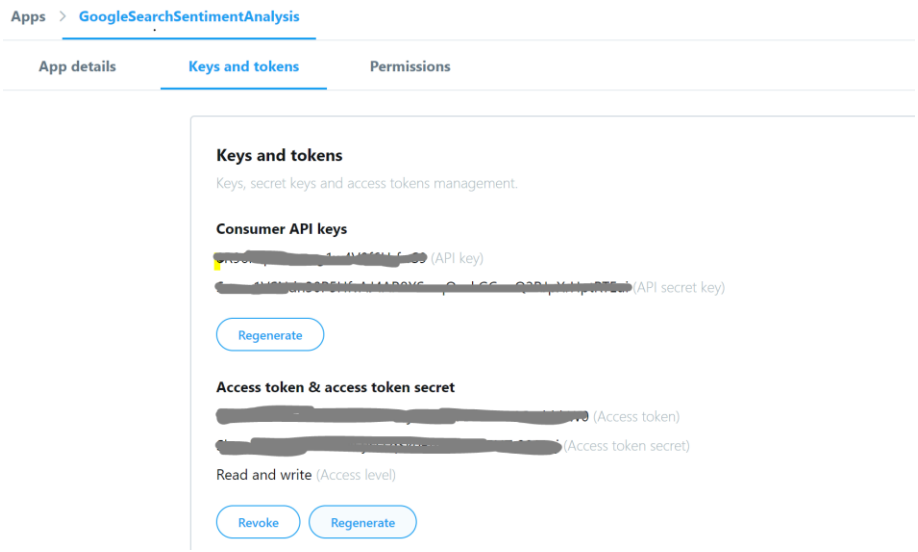


Figure 17: API Credentials for Twitter Developer Account

4.2.2: Creating a Dataset of Tweets in Power BI using R Library 'rtweet'

After getting the API keys, the next step is to create a table for Tweeter dataset filtered by a trending search term. The query/table created in listing 3.1 is used as the source table here. All the rows except the first are removed to concise the table into 1x1 cell with only the top searched topic title. The topic is then used as a keyword to retrieve tweets using a R script.

With just a few steps in Power BI Desktop, R scripts can be added, and the resulting datasets is imported into an existing query. To run R scripts, R and all the necessary libraries used in the script need to be installed on the local machine [10]. Once the Twitter application is set up with necessary API keys, the next step is accessing tweets in R using 'rtweet' package available at https://cran.r-project.org/web/packages/rtweet/rtweet.pdf. The package implements necessary calls designed to collect and organize data via Twitter's REST and Streaming API. In R script, to authenticates REST API access to Twitter, a token is created with App Name, API key, API secret, Access token, and Access token secret information.

```
appname <- [redacted]
key <- "[redacted]
secret <- "[redacted]"
access_token <- "[redacted]"
access_secret <- "[redacted]"

twitter_token <- create_token (
          app = appname,
          consumer_key = key,
          consumer_secret = secret,
          access_token = access_token,
          access_secret = access_secret  )
```

After a successful authentication, using search_tweets() function from 'rtweet' library, I searched all tweets using the topic in figure 1 as a keyword.

```
df_tweets <- search_tweets( search_term, n = 1000, lang = "en",
                            include_rts = FALSE, `-filter` = "replies")
```

search_tweets() function returns a data frame of Twitter statuses matching a user provided keyword from the past 6 - 9 days with following arguments. As, the tweets are captured using R script in Power BI, to avoid time out events in requesting Tweeter API call, the resulted dataframe of search_tweet() function includes 1000 most recent Tweets related to the search term.

- search query: Query to be searched, used to filter and select tweets to return from Twitter's REST API
- n: the total number of tweets to return.
- lang = "en", to return only English language tweets.
- Include_rts = FALSE, to not include retweets in search results.
- -filter = 'replies', to exclude replies.

After getting the Twitter feeds, the next step is cleaning and preprocessing of these tweet text to avoid feeding unnecessary data into analysis. The following R function perform a data cleaning operation by removing URL links, screen names (staring with @), punctuation, numbers, extra spaces, hashtag and other regular expression, Unicode characters etc.

```
clean_tweet_text <- function(raw_tweets)
{
  require(stringi)
  c(
    url_pattern = "http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|
                   (?:%[0-9a-fA-F][0-9a-fA-F]))+",
    handle_pattern = "(^|[^@[:word:]])@([:word:]{1,15})\b",
    entity_pattern = "&[^[:space:]]*;",
    other_pattern = "[[:punct:][:cntrl:][:digit:]]",
    twitter_hashtag_regex <- "(^|[^&\\p{L}\\p{M}\\p{Nd}_\u200c\u200d\ua67e\u05be\
                        u05f3\u05f4\u309b\u309c\u30a0\u30fb\u3003\u0f0b\u0f0c\u00b7])
                        (#|\uFF03)(?!\uFE0F|\u20E3)([\\p{L}\\p{M}\\p{Nd}_\u200c\u200d\
                        ua67e\u05be\u05f3\u05f4\u309b\u309c\u30a0\u30fb\u3003\u0f0b\
                        u0f0c\u00b7]*[\\p{L}\\p{M}][\\p{L}\\p{M}\\p{Nd}_\u200c\u200d\
              ua67e\u05be\u05f3\u05f4\u309b\u309c\u30a0\u30fb\u3003\u0f0b\u0f0c\u00b7]*)"
  ) -> patterns

  patterns <- sprintf("(%s)", paste0(patterns, collapse="|"))

  cleaned_tweets <- stri_replace_all_regex(raw_tweets, patterns, "")
  cleaned_tweets <- stri_trim_both(cleaned_tweets)
  cleaned_tweets
}
```

Finally, an output dataset df-final_tweets is created by including only the text column of data frame df_tweets originally created from search_tweet function. This output dataset is returned to Power BI query to perform text analysis within it.

```
df_final_tweets <- data.frame (
  Text = clean_tweet_text(df_tweets$text),
  stringsAsFactors = FALSE
)
```

Below is a snapshot of dataset of cleaned tweets for keyword "Eddie Murphy"

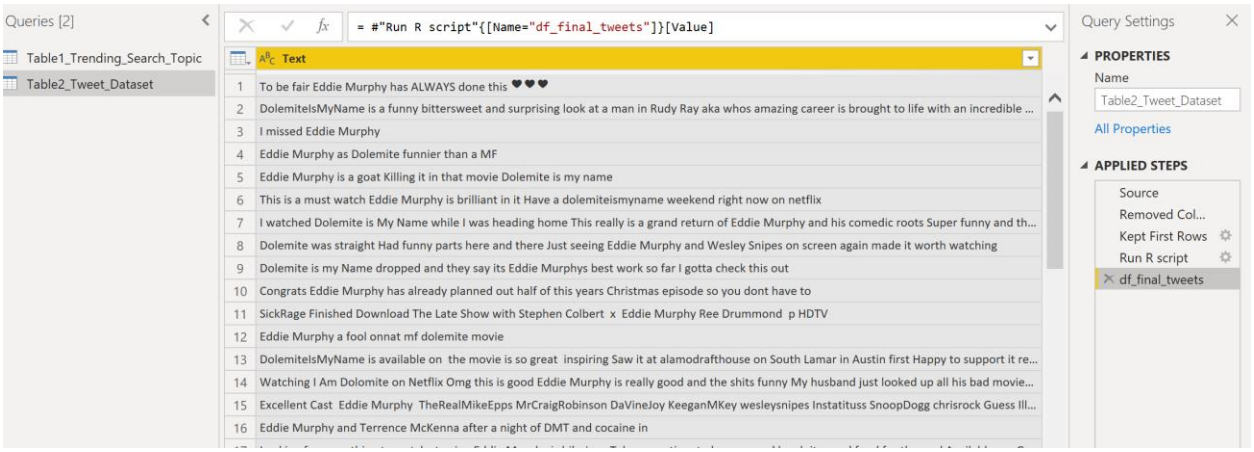


Figure 18: Dataset of Tweets after Text Cleaning and Preprocessing

4.2.3: Creating a Dataset of Emotion Types associated with Tweets in Power BI using R Library 'syuzhet'

After importing Twitter data frame in Power BI, a dataset of different emotion categories of words associated with these tweets is created using R package 'syuzhet'. The package is a combination of three most commonly used sentiment lexicons: AFINN, BING and NRC. The package is available at https://cran.r-project.org/web/packages/syuzhet/syuzhet.pdf. Among these lexicons, the NRC emotion lexicon is a list of words and their associations with eight emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive) [11] over other 2 lexicons that only categorize words into two sentiments (positive and negative). Following is an example using a sample text:

```
mytext <- c( '3Hours Late Flight -and now we need to wait TWENTY MORE MINUTES for a gate!
     I have Patience but none for incompetence')
```

For the above text, syuzhet library's get_nrc_sentiment() function Calls the NRC sentiment dictionary to calculate the presence of eight different emotion words. The get_nrc_sentiment() function returns a data frame in which each row represents a sentence from the input dataset.

```
get_nrc_sentiment(mytext)
anger anticipation disgust fear joy sadness surprise trust negative positive
    0            2       0    0   0       1        0     2        3        1
```

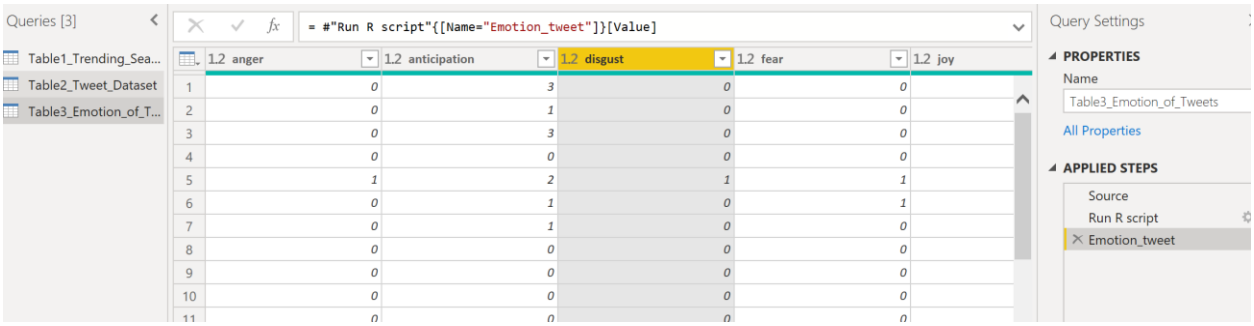


Figure 19: Dataset of Words Count Associated with Different Emotion Categories

4.3: Extracting News Headlines into Power BI

In this step a dataset of News Headline is created) from a Google search result of the trending topic (created in listing 3.1) using Python programming language. To limit the number of articles, the first 30 most recent news articles are selected from the search result. For extracting news data from each page, a web function is implemented using Python programming languages. Besides Python, R is also used for web scrapping using package like RCurl. However, R is a bit too focused on data processing and less to user interfacing. Python is more user-friendly in retrieving data and more straightforward to do non-statistical tasks in Python. Besides, Power BI limits some R libraries to run in its environment. Considering these factors, with well-maintained libraries like BeautifulSoup and requests, web scraping in Python is far convenient than in R.

When adding a Python script, power BI create a data frame of the selected input columns in a data frame called 'dataset'. This 'dataset' could be manipulated and analyzed by Python's 'PANDAS' library. The following python script is used to create a search topic (query) from the input 'dataset'.

```
# 'dataset' holds the input data for this script
import pandas as pd

Topic = pd.DataFrame(dataset)
q = Topic.loc[0,'Title']
query = str(q)
```

For pulling data out of news article's HTML and XML file, python 'request_html' and 'BeautifulSoup' module is used. Using 'request_html' library in python, a session object is created to send an HTTP request by making a simple GET request to the URL **https://www.google.com/search?q=" + query+"&num=30&source=lnms&tbm=nws**. The GET request is stored in a response variable, *response*. If the connection is successful, print out of *response* returns a Response 200 message, otherwise returns a different message [18].

```
from requests_html import HTMLSession
session = HTMLSession()
response = session.get("https://www.google.com/search?q=" + query+"&num=30&source=lnms&tbm=nws")
```

Next, I created a BeautifulSoup object soup using the response content. The soup object allows print out, search through, and perform several functions on the webpage's HTML structure. To scrape information from a web page, we need to find the HTML tag under which the information appears. As, we are only interested in headlines of an article, in following figure we can see that it appears under tag. For a google search result's news tab, the div class id is always consistent for any search query.

Figure 20: HTML Structure of a Google Search Page

Once the specific tag is isolated, the find_all() method is used to find all the instances of the tag on a page. To access the text inside an HTML element, the .text method is used. As the URL returns top 30 search result, so find_all() returns a list of headlines, to loop through, each tag a List 'Headline' is created.

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')
headline_results = soup.find_all('div', class_='phYMDf nDgy9d')
Headlines = []
for h in headline_results:
    Headlines.append(h.text)
```

Finally, the list of headlines is binded as a column to the output datframe.

```
output_Headline = pd.DataFrame(Headlines)
```

If the script runs successfully, a Navigator dialog box appears with all the data frames resulted in python script.

	Name	Value
1	dataset	Table
2	output_Headline	Table
3	Topic	Table

To load the data and use it in power BI, only the final output dataset with Headlines is selected as shown in following image.

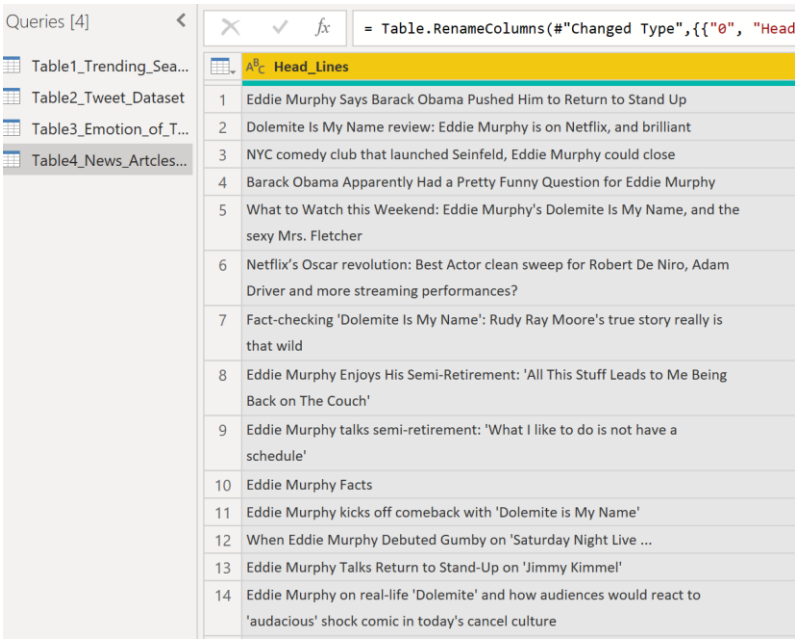


Figure 21: Dataset of News Headlines

4.3.1: Creating a Dataset of Emotion Types Associated with Headlines in Power BI using R Library 'syuzhet'

After extracting the news headlines in Power BI, same steps of listing 3.2.3 are repeated to create a dataset (similar in figure 15) of different emotion categories of associated words is created using R package 'syuzhet'.

4.4: Visualization

A dashboard is created by applying various text analytics application to the extracted datasets from Twitter and News article in Power BI desktop. A great advantage of using Power BI is, users can create their own custom visualization to explore and display data in captivating ways. As most of the text analysis work in this project is carried out using various R library, I used R custom visualization. Regular R Powered Visual in Power BI only creates and displays a static image where in-visual interactivity is not possible. Besides, If the report is shared with other users, they can also see the R code used to create the visual. But custom visual makes the visual more dynamic and interactive with zooming or hovering options. Also, the custom R visualizations can be easily transferable and deployable to other users without sharing the underneath R code.

Following visualizations are included in the final dashboard.

4.4.1: A Bar Chart of Different Emotion Types

With the explosion of digital and social media, people are expressing personal feelings or emotions in textual forms. The dataset of different emotion word created in listing 3.2.3 is used as input dataset ('Values' in following R script) for this chart. The dataframe Values has eight columns each of different emotion category. The following R script calculates the total number of words and percentage for each emotion category and save the results in another dataframe named 'emotion_sum'.

```
emotion_bar = colSums(Values)
emotion_sum = data.frame(Words_Count=emotion_bar, emotion=names(emotion_bar))

emotion_sum$emotion = factor(emotion_sum$emotion,levels=emotion_sum$emotion[order(emotion_sum$Words_Count, decreasing = TRUE)])

emotion_sum$Word_Percentage = round( (emotion_sum$Words_Count / sum(emotion_sum$Words_Count))*100 ,2)
```

Following is a preview of the dataframe 'emotion_sum' with number of words for each emotion category and percentage across the 1000 tweets dataset.

Filter

	Words_Count	emotion	Word_Percentage
anger	210	anger	7.40
anticipation	242	anticipation	8.52
disgust	110	disgust	3.87
fear	514	fear	18.10
joy	121	joy	4.26
sadness	116	sadness	4.09
surprise	126	surprise	4.44
trust	295	trust	10.39
negative	651	negative	22.93
positive	454	positive	15.99

A bar chart is created from dataframe 'emotion_sum' where each bar represents the total number of words associated with corresponding emotion category.

```
plot_ly(data = emotion_sum,
            x = ~emotion,
            y = ~Words_Count,
            type = "bar",color=~emotion,
            text = ~paste(emotion,Word_Percentage,"%"),
            textposition = "auto",
            hoverinfo = "text", showlegend = F) %>%
            layout(xaxis=list(title=""), showlegend=FALSE,
           title="")
```

The steps are repeated using news data.

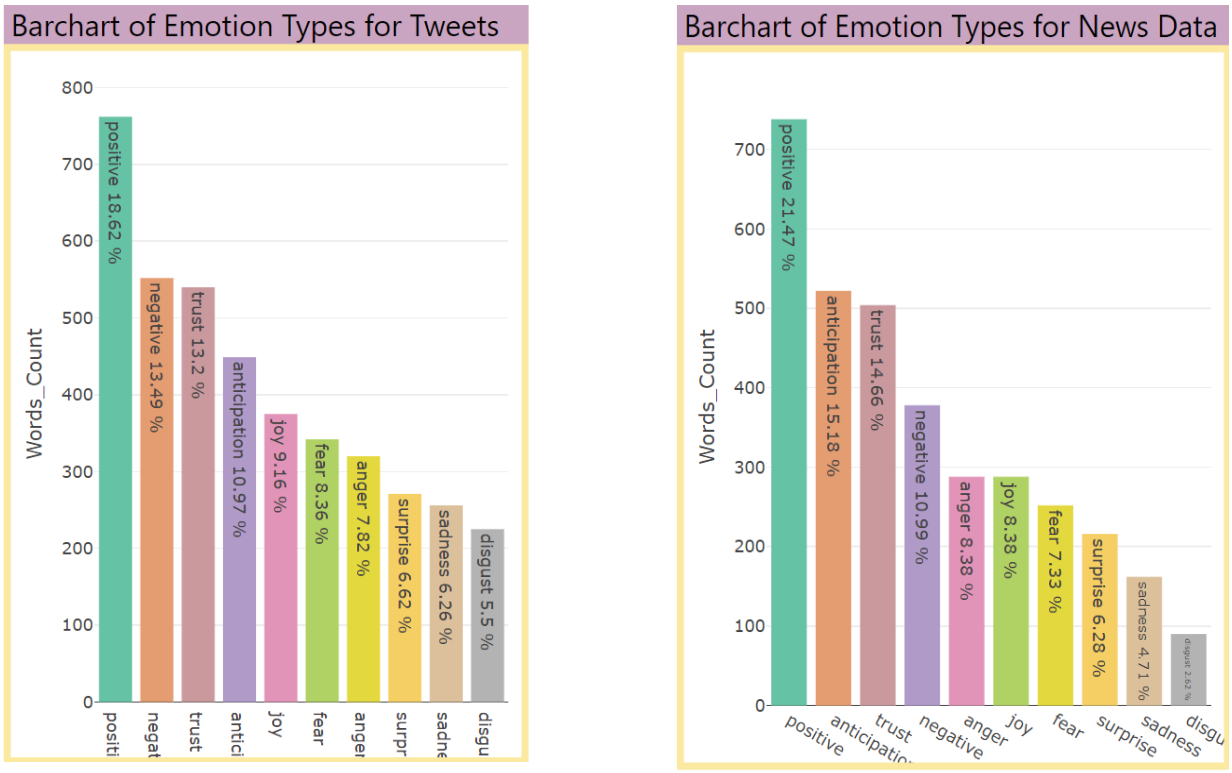


Figure 22: Bar Chart of Emotion Types Associated with the words in Tweet and News Data.

In the above charts, x-axis represent types of emotion and y-axis is for words count for that emotion type. We can see that, for both twitter and news data most words represent positive emotion.

4.4.2: Doughnut Chart of Sentiment Score Associated with Tweeter and News Data

Although the emotion bar graph in figure [18] gives an idea about different words used in a text, but it does not represent the overall polarity (positive, negative, neutral etc.) of a sentence. To

calculate the sentiment score of each tweet and news articles text, I used R 'sentimentr' library. sentimentr is designed to quickly calculate text polarity sentiment at the sentence level and optionally aggregate by rows or grouping variable(s). Following steps describe the process of using this library to create a Doughnut chart of overall sentiment score for Tweeter and news data set.

Step -1:

Creating a new data frame by converting the input dataset of text (dataset created in listing 3.2.2 and 3.3) into a character vector.

```
library(sentimentr)
input_dataset <- data.frame(text = as.character(Values$Text), stringsAsFactors=FALSE)
```

Step -2:

The get_sentences() function of 'sentimentr' package extract sentences from an input character vector. After applying sentiment_by() function to get_sentences() object, a sentiment/polarity score is returned on a range of -1(negative) to 1(positive) where 0 represents neutral. Here is an example using a sample text:

```
library(sentimentr)
mytext <- c('3Hours Late Flight-and now we have to wait TWENTY mORE MINUTES for a gate!',
            ' I have patience but none for incompetence')
sentiment_by(mytext)
```

sentiment_by function calculates a polarity score for each sentence of 'mytext'. We can see that the first sentence is negative and the second one is very positive.

```
sentiment_by(mytext)
 element_id word_count sd ave_sentiment
          1         15 NA    -0.1807392
          2          7 NA     0.8502311
```

The following R code calculates the polarity score of each tweet in an input data frame and assigns the score in another column of same data frame.

```
mytext <- get_sentences(input_dataset$text)
Sentiment = sentiment_by(mytext,by=NULL)

#adding a new column to the data frame by rounding average sentiment score upto 2 decimal point
input_dataset <- dplyr::mutate(input_dataset, Senti_Score = round(Sentiment$ave_sentiment , 2 ))
```

Below is a screenshot of the resulted data frame with sentiment score calculated from sentiment_by function.

	Cleaned_Text	Senti_Score
1	In all of my years I have never seen anything like this at all and I pr...	-0.44
2	President Trump said hes not sure that hes ever even heard of a Ca...	-0.27
3	Thatâ€™s my kitchen window that water is hitting and that has to ...	-0.14
4	Total darkness screaming wind We are running out of high ground...	-0.54
5	This video captures devastation in the Bahamas from Hurricane Do...	-0.47
6	Thatâ€™s my kitchen window that water is hitting and that has to ...	-0.14
7	This video captures devastation in the Bahamas from Hurricane Do...	-0.47
8	US President Trump said hes not sure that hes ever even heard of ...	-0.26
9	In all of my years I have never seen anything like this at all and I pr...	-0.44
10	Total darkness screaming wind We are running out of high ground...	-0.54
11	Made no call for casualties at Maralago But if my house was being ...	0.16
12	Hurricane Dorian Tropical Cyclone Update Atlantic	-0.43
13	TAKE A LOOK AT THIS Heres a look at the flooding and damage H...	-0.20
14	CHECK THIS OUT A CBS viewer captured the NOAA WP flying over...	-0.04

Figure 23: Dataset of Tweets with Sentiment Score

Step-3:

Created a polarity variable for each sentiment by setting up the threshold score for positive, negative and neutral sentiments category. For my analysis, I classified a score greater than 0.1 represents positive sentiment and score less than -0.1 represent negative sentiment. All the score in between represents neutral sentiment. Then, I subset the data frame into individual datasets for each sentiment category.

```
sentiment.positive = subset(input_dataset, Senti_Score < -0.1 )
sentiment.negative = subset(input_dataset, Senti_Score > 0.1)
sentiment.neutral = subset(input_dataset, Senti_Score > - 0.1 & Senti_Score < 0.1)
```

Step-4:

summarize the overall positive, negative and neutral scores by adding positive, negative and neutral sum of the previous subset data set. Then created a temporary dataframe with the summarized value.

```
N= nrow(input_dataset)

Npositive = nrow(sentiment.positive)
NNegative = nrow(sentiment.negative)
NNeutral =  nrow(sentiment.neutral)

dataframe_doughnat=data.frame(topic=c("Positive", "Negative","Neutral"),
                              number=c(Npositive,NNegative,NNeutral))
```

The same steps are repeated with news data to assign sentiment score with news article's text data.

Step-5:

Here is the code to create an overall sentiment comparison doughnut chart for these three-sentiment categories.

```
plot_ly(data=dataframe_doughnat, labels = ~topic, values = ~number,
        type = 'pie', textinfo = 'label+percent',hole = 0.4) %>%
  layout(title = '',
   xaxis = list(showgrid = TRUE, zeroline = TRUE, showticklabels = TRUE),
   yaxis = list(showgrid = TRUE, zeroline = TRUE, showticklabels = TRUE),
   legend = list(orientation = 'h',xanchor ='center', x =0.5))
```

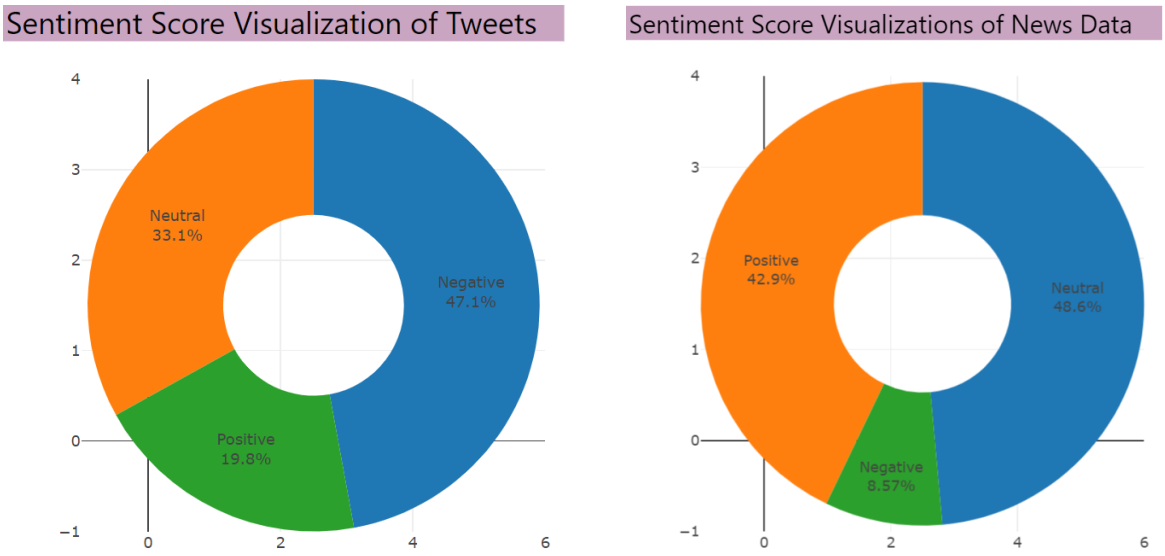


Figure 24: Doughnut Chart of Overall Sentiment Score for Tweeter and News Data

4.4.3: Most common positive and negative words using R 'tidytext'

The tidytext package available at https://cran.rproject.org/web/packages/tidytext/vignettes/tidytext.html provides functions and supporting data sets to allow conversion of text to and from tidy formats. Tidy datasets are easy to manipulate, model and visualize, and have a specific structure where each variable is a column, each observation is a row, and each type of observational unit is a table [19]. To transform the input text to a tidy data structure, tidytext's unnest_tokens function creates a data frame of one word in each row by splitting each row of input text.

```
library(tidytext)
tidy_dataset <- input_dataset %>% unnest_tokens("word", TEXT)
```

R library dplyr's anti_join() is used to remove stop words that are not useful for an analysis such as "the", "of", "to", and so forth in English (kept in the tidytext dataset stop_words).

```
data("stop_words")
library(dplyr)
tidy_dataset %>% anti_join(stop_words)
```

The tidytext package contains several sentiment lexicons. Three general-purpose lexicons are afinn, bing and nrc. The bing lexicon from Bing Liu and collaborators, categorizes English words in a binary fashion into positive and negative categories.

```
get_sentiments("bing")
```

```
## # A tibble: 6,786 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 2-faces     negative
##  2 abnormal    negative
##  3 abolish     negative
##  4 abominable  negative
##  5 abominably  negative
##  6 abominate   negative
##  7 abomination negative
##  8 abort       negative
##  9 aborted     negative
## 10 aborts      negative
## # … with 6,776 more rows
```

A subset of 'bing' lexicon's word and their sentiment.

tidytext provides a function get_sentiments() to get specific sentiment lexicon and then use inner_join() to perform the sentiment analysis. One advantage of having the data frame with both sentiment and word is to analyze word counts that contribute to each sentiment. To find out how much each word contributed to each sentiment R library dplyr's count() is used.

```
bing_word_counts <- tidy_dataset %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

Following is a preview of the dataframe 'bing_word_counts' with number of words for each sentiment category and how many times (n) the word appeared in the input text data.

	word	sentiment	n
1	grand	positive	23
2	like	positive	19
3	trump	positive	13
4	damage	negative	11
5	slowly	negative	11
6	relief	positive	10
7	powerful	positive	9
8	right	positive	9
9	disaster	negative	8
10	weaken	negative	8
11	calm	positive	7
12	dangerous	negative	7
13	dead	negative	7
14	deadly	negative	7
15	safe	positive	7
16	well	positive	7
17	devastation	negative	6

This can be shown visually, using a stacked bar chart is created from the above data frame to visually present top 10 most common words that contribute to positive and negative sentiment.

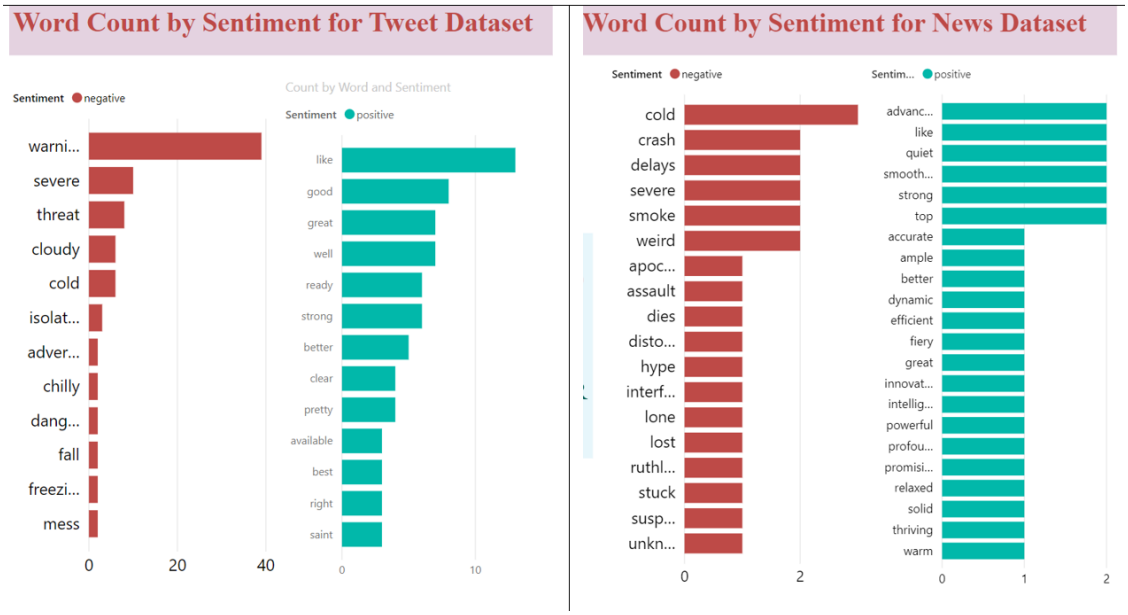


Figure 25: Words Contribution to Positive and Negative Sentiment in Tweet and News Dataset

4.4.4: WordCloud of Most Common Phrases

Word cloud is a text mining technique that highlights the most frequently used keywords in a paragraph of text. Ian Fellows R 'wordcloud' package available at https://cran.r-project.org/web/packages/wordcloud/wordcloud.pdf is used to analyze texts and quickly visualize the keywords as a word cloud. The dataset consists with two columns. A 'text' column of tweet / news text data and a 'Title' column of the trending search topic (created in figure 15). After loading the required library in R, a data frame is created converting the input dataset of text (dataset created in listing 3.2.2 and 3.3) into a character vector. Since each row is made up of multiple combined words, to filter out words or count which occur most frequently, the data frame is tokenized with one token (word) in each row. To tokenize the text into individual words and transform it to a tidy data structure, I used tidytext's unnest_tokens() function [20].

```
library(tidytext)
library(stringr)
library(wordcloud)

dataset2 <- data.frame(lapply(dataset, as.character), stringsAsFactors=FALSE)

#extract the search term from original dataset
term <- dataset$Title[1]

#creating a tibble of customed undesirable words of the topic to remove from wordcloud.
# by splitting the term into individual words using 'stringr' strsplit() function.
# strsplit() returns a list of word which is converted to a tiblle using unlist() function

undesirable_words <- tibble( word = unlist(strsplit(term, " ")))

tidy_dataset <- dataset2 %>%
  unnest_tokens("word", Text)

data("stop_words")

tidy_dataset %>%
  anti_join(stop_words) %>%
  anti_join(undesirable_words, by = "word") %>%  #removing custom words
  count(word) %>%
  with(wordcloud(word, n, max.words = 100, random.order=FALSE,
                 rot.per =0.3 ,scale = c(2,.8),colors=brewer.pal(8,"Dark2")))
```

Once the data is in one-word-per-row format, the English stop words (kept in the tidytext dataset stop_words) that are not useful for an analysis such as "the", "of", "to", and so forth is removed with an anti_join(). After the necessary data clean up the importance of words is illustrated as a word cloud with wordcloud() function. Arguments of the word cloud generator functions are:

- word: the words in a tidy datatset
- freq: the frequencies of the word appear in the text
- max.words: Plots the specified number of words and discard least frequent terms .
- random.order: By setting this to false, the words with the highest frequency are plotted first.
- scale: Indicates the range of sizes of the words.

- rot.per: Determines the fraction of words that are plotted vertically.

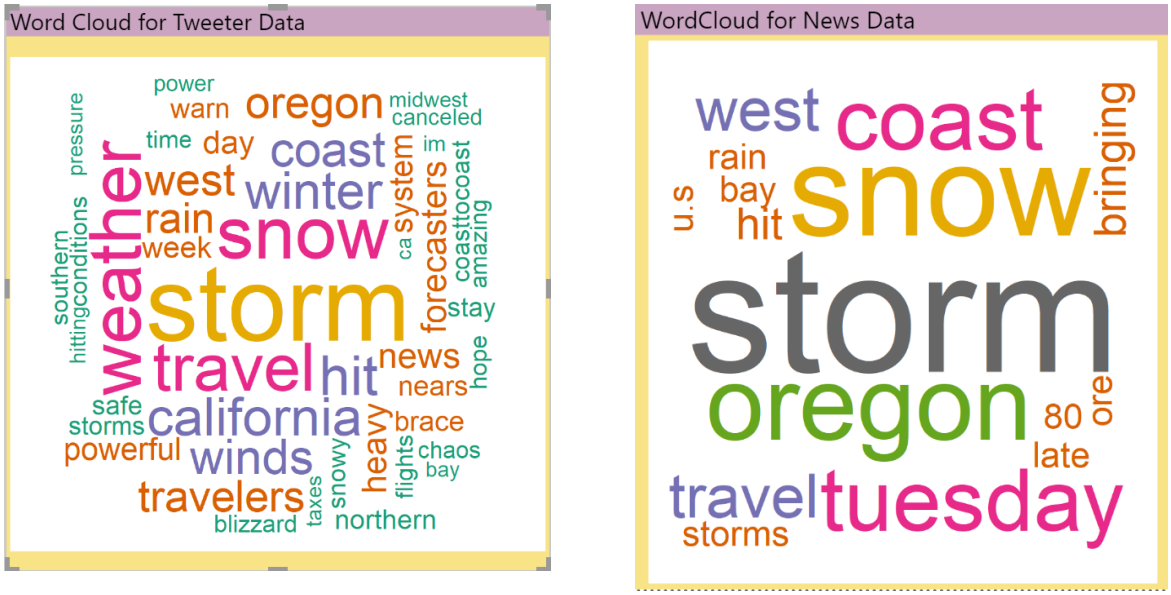


Figure 26: WordClouds of most common words in Tweeter and News Data

The final report contains 5 pages each with different visualization for Tweet and News Data.

Figure 27: Final Report – Page 1

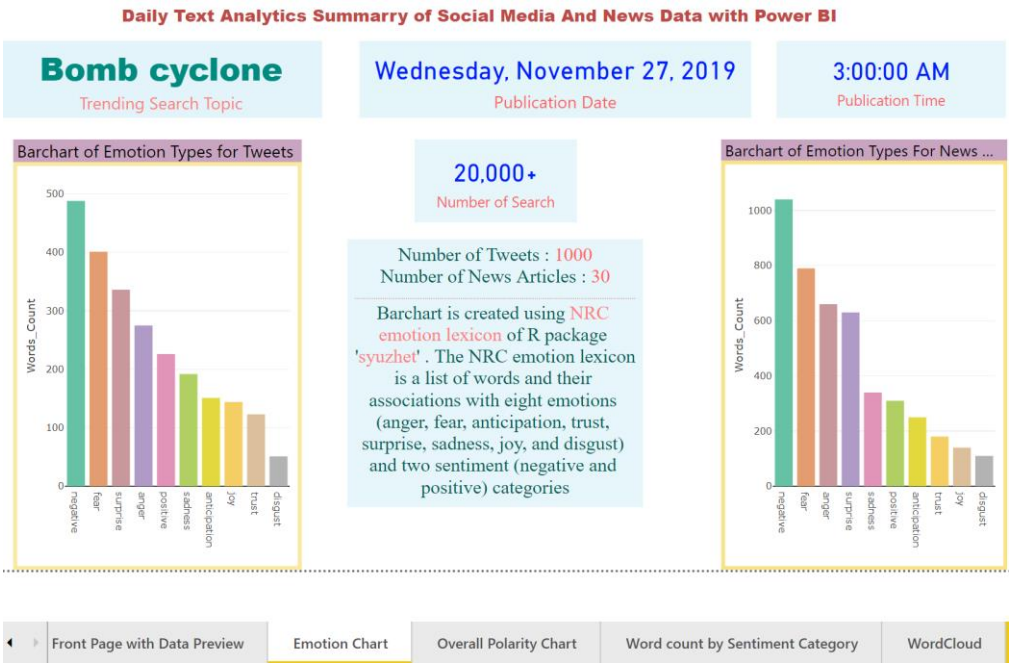


Figure 28: Final Report – Page 2

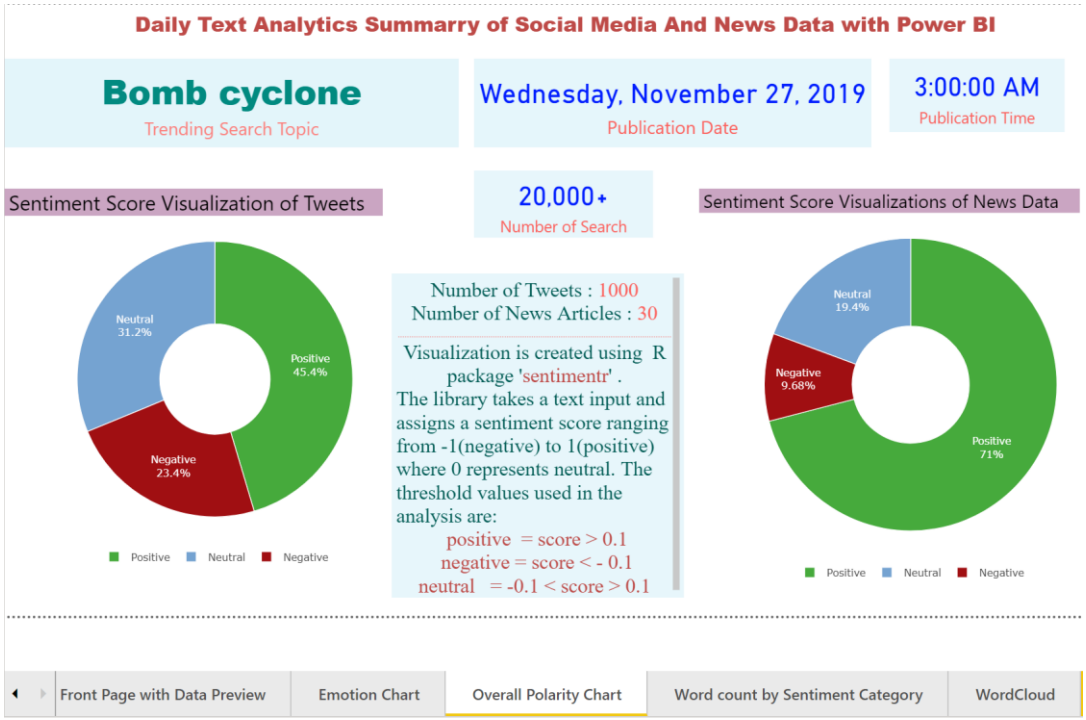


Figure 29: Final Report – Page 3

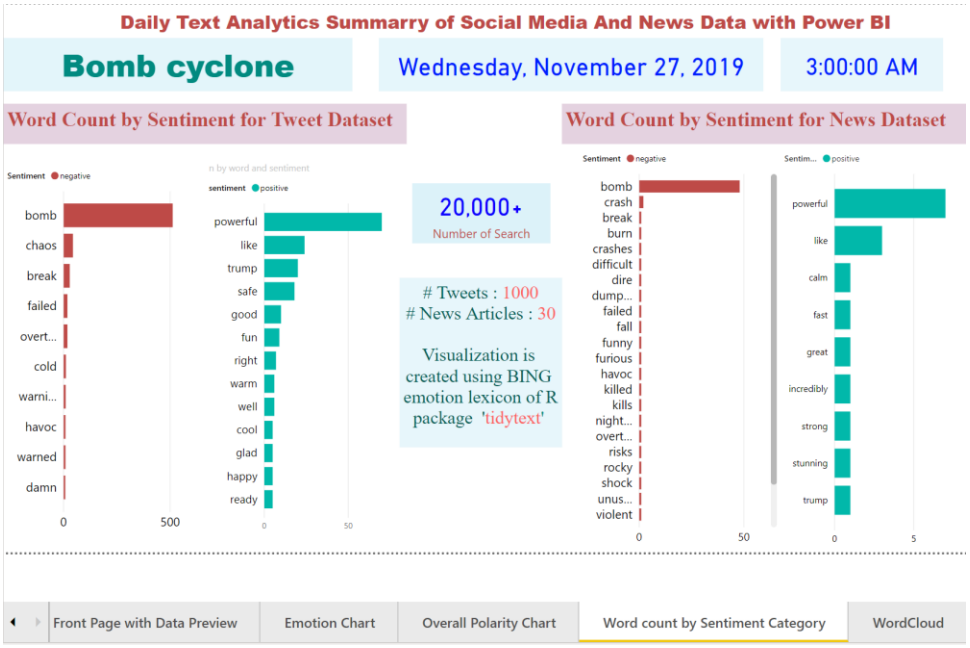


Figure 30: Final Report – Page 4

Figure 31: Final Report – Page 5

4.5: Publishing the Report to Power BI Service

After creating report in Power BI Desktop, the report and dataset is uploaded to the Power BI service. To generate the report with most updated data, a scheduled refresh is configured on the dataset so that Power BI can refresh the dataset for the reports. The scheduled refresh defines the frequency and time slots to refresh the dataset. Having configured a refresh schedule, the dataset settings page informs about the next refresh time. As Google daily search trends data will be the primary data source for this project, automatic data refresh is scheduled three times a day. The following screenshot shows a refresh schedule on an eight-hour interval.

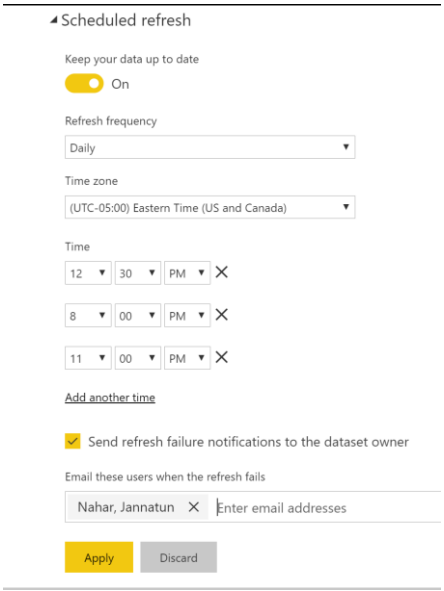


Figure 32: Scheduling Automatic Dataset Refresh in Power BI Service

The following figure shows a completed refresh cycle for a sample dataset.

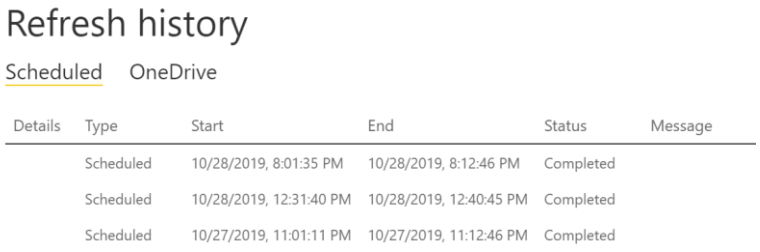


Figure 33: Automatic Dataset Refresh Log History in Power BI Service

4.5.1: Setting up Refresh Failure Notifications

Power BI sends refresh failure notifications indicating the error message through email to the dataset owner so that the owner can act in a timely manner should refresh issues occur. Following screenshot for an example of such refresh failure notification.

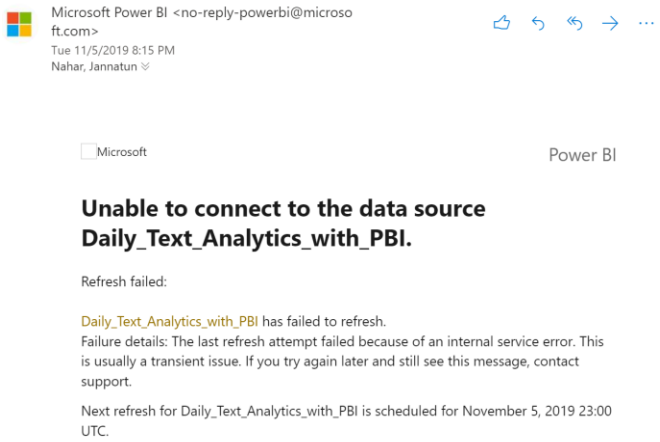


Figure 34: Dataset Refresh Failure Notification

Power BI also offers sharing and collaboration of dashboards, reports, and data within an organization. For this project, the report will be shared with all power BI users of UNCW domain. A recipient with a xxxxx@uncw.edu power BI account can view it and interact with it but can't edit it.

Chapter 5

Benefits

In an age where data is constantly being collected, the exact amount of unstructured data is unknown, but it is undeniably huge. International Data Corporation (IDC) estimates that unstructured data is growing at a rate of 62 percent per year and predicts that by 2020, 93 percent of all data in the digital universe will be unstructured [21]. Today, game-changing technologies like text analytics are transforming this unstructured information into structured data that is being further analyzed using traditional business intelligence software like Power BI. This chapter is focused on the benefits Power BI unleashes on users who are performing daily text analysis operation.

5.1: Reference project for professionals

Recently, most Enterprise data includes free form text that comes from various social media posts, product reviews and survey results. To unlock the value of these text data, an advanced analytics application equipped with Tools like R, Python, text mining & sentiment lexicon might be a good solution. With Power BI, all these tools can be integrated into a single platform to analyze text data and create effective visual dashboard. This project will serve as a reference document for professionals, developer or analyst to leverage the power of text analytics in conjunction with Power BI.

5.2: Integration of multiple data sources into a single data flow

In the modern BI world, data preparation is considered the most time-consuming task, estimated by experts as taking 60% - 80% of the time and cost of a typical analytics project [24]. A strong side that makes Power BI stand out of its competitor is Power BI does not limit data connectivity even though it is coming from a lot of different data sources. In this project, I added data sources from the web, Twitter application and various news article into a single data flow.

5.3: Result is Repeatable and Scheduled

After publishing the report in Power BI service, the dataset is automatically uploaded along with the report. To keep the most recent trending search topic and tweets, news data about the searched topic, all the data sources gets refreshed three times a day. So, the user can have an updated report after a regular interval of time automatically if there are any changes in the data source.

5.4: Sharing Report with other Users

As a popular open-source platform, R has an extensive user community that develops and maintains a wide range of text analysis packages [25]. In summary, we can do amazing things in R. But, sometimes creating a report is not enough. It can sometimes be challenging to present the output, share and explain it with other users with a up to date result. With cloud-based Power BI tool, the report is publishable to other user to share the latest data and information. The advantage to sharing a power BI report versus mass distribution is that the target audience could be tailored for a specific individual, group, team, department, or entire enterprise as necessary. people it is shared with can view it and interact with it but can't edit unless they have given owner access to the dataset.

5.5: Integration of R from Power BI

Even though Power BI has a great set of tools for ETL and data discovery, sometimes data shape doesn't quite match what is needed to produce the visualization for example Stacked Clustered Column Charts. In situation like this, power BI allows us to easily integrate R, a programming language widely used by statisticians, data scientists, and data analysts. Using R in Power BI's Query Editor, we can prepare data model and create reports of powerful custom R visualizations. This project can be used as an end-to-end reference document for professionals on how to integrate R script and use its visual along with Power BI's in home visualization.

5.6: Integration of Python from Power BI:

Considering this, a user wants to quickly grab some interesting data while surfing on the web and do some analysis. One of my favorite features of Power BI is the ability to extract data from websites using Power Query. Data in a tabular format in any webpages can be easily imported in power BI. But handling web page object that are not tables (as example raw text) is less straightforward. For this we need to navigate through HTML tags in a programmable fashion. Python does a really great job for scraping and wrangling data from multiple sources. The Python part of the project comes handy and is kind of acting like the "glue" to get multiple things integrated together in power BI platform.

GitHub repository of the project is available at https://github.com/JannatNahar/Text-Analytics.

Chapter 6

Challenges and Lesson Learned

With most data analytics application, getting insights out of the data is a complex process that has its ups and downs. As a data integration platform Power BI is no different, although it's user-friendly and intuitive interface does help smooth out some of these issues and make the process easier to handle. In this module the challenges and adjustments are discussed in detail.

6.1: Web Scrapping from Multiple News Articles

Power BI's "Get Data" from web feature in makes it a lot easier to import data in a tabular format (way structured) from webpages. For this project, I had to collect large unstructured text data from multiple news article. I implemented a web scraping function in python. Web scraping is an automated method used to extract large amounts of unstructured data from websites and store it in a structured form. When we run the code for web scraping, a request is sent to the mentioned URL. As a response to the request, the server sends the data and allows the request to read the HTML or XML page. The code then, parses the HTML or XML page, finds the data and extracts it [26]. The data is usually nested in HTML tags. So, we need to inspect the page to see, under which tag the data we want to scrape is nested.

For a trending search topic, a Google search result returns various news articles. The challenge was to create a universal solution for web scraping news text data from these articles. After doing a lot of research I could not found a solution of this problem because the way data is stored on each website is usually specific to that site. My initial plan was to use the RSS feed for popular newspapers like New York times, the Washington post etc. But, a lot of time there was no recent feed for a search result on these newspapers which created an empty dataset. So, I decided to use the news article lists returned by a Google search result (under the news tab). One thing that is similar to every article is, the block of text data exists in in the HTML paragraph **<p>** element.

Although, not all the <p> tag's text data is useful, using this tag I had the raw text news data from every URLs in Python. As Power BI is great with data cleansing and transformation, after importing the data set from Python I calculated the character length of each paragraph and filtered out the rows with length smaller than 100 characters.

```
Table.AddColumn(#"Trimmed Text", "LENGTH", each Text.Length([TEXT]))

Table.SelectRows(#"Added Custom", each [LENGTH] >= 100)
```

Now a dataset is created from news article's text content. Although it is not the most accurate but reasonable enough to apply text analytics application to it.

6.2: Assigning Polarity/Sentiment Score to Tweets and News Data

The objective of the project was to represents a system that assigns polarity scores and extract key phrases from the opinion expressed in news stories and twitter posts on a search topic. To assign polarity score, I experimented with following text analysis modules:

6.2.1: Integrating Azure Text Analytics API

As Microsoft Power BI is part of Microsoft suite business products, it has a tight integration with its other business tools such as Azure services. Text Analytics API is a cloud based azure service that does language detection, key phrase extraction, topic extraction and sentiment analysis to a given set of text. According to text analytics API pricing [27], 5000 API transactions are allowed in the free tier each month. For instance, if a dataset of 1,000 tweets are sent for sentiment analysis in a single API call, that counts for 1,000 transactions. Also, If an API supports more than one annotation operation, that will also be considered. If an API call performs both sentiment analysis and key-phrase extraction on 1,000 tweets, that will count for 2,000 transactions (2 annotations * 1,000 tweets).

So, on using Text analytics API with the tweets (1000) and news dataset, number of transactions of my free tier will reach transaction limit in just 1 day. For the subsequent API request it results a Forbidden Access errors. Also, with the closest paid tier the transaction limits are up to 25,000 per month for $74.71 /month pricing rate [27]. It was difficult to build a system based on this Azure service.

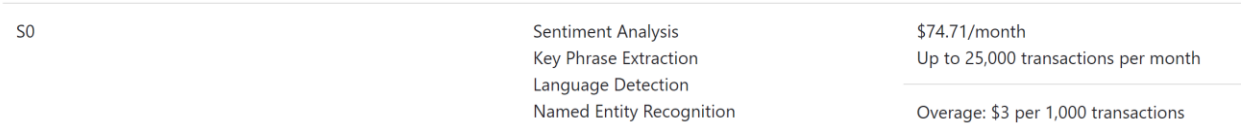


Figure 35: Pricing for Azure Text Analytics API

6.2.2: Integrating R 'sentimentr' Package

R has a wide variety of useful text analytics packages to apply sentiment analysis strategy and assign polarity score to text data. For example "RSentiment" package loads text and calculates score of each sentence based on presence of positive and negative words into five categories very negative (smaller than -1), negative(-1), neutral(score 0), positive(1), very positive(greater than 1) and Sarcasm(score 99) [28] . Another R package "sentimentr" calculates sentiment score on a range of -1 (negative) to 1(positive) where 0 represents neutral.

One of the most valued features in Power BI Desktop is its integration with R scripts to import or transform data, as well as create visualizations. With Azure free tier API limit limitations, these R packages were the possible alternatives to assign polarity score to social media and news data. While many R packages are supported in the Power BI service and some packages are currently not [29]. A table of supported R packages in Power BI is available at https://docs.microsoft.com/en-us/power-bi/service-r-packages-support . According to this list, 'RSentiment' package is not supported in power BI currently but 'sentimentr' does. So, a possible option was to use this package to calculate polarity score and import the resulting data frame with score back in power BI. Although, the package is listed in supported R package documents in Microsoft official documentation, using the package results to a Data format conversion error [30].

Another way to use R in power BI is to use custom visuals created with R scripts. Regular R powered visual in power BI captures the output plot and displays it as a static image. In this Custom R visual Plotly and htmlwidgets packages are used to make the visual more dynamic and interactive. Plotly is an R package for creating interactive web-based graphs via the open source JavaScript graphing library available at https://plot.ly/r/getting-started/.The following steps describe the process of creating a custom R HTML visual of sentiment score using "sentimentr" package and importing it into power BI desktop. Following steps describe the process of creating a custom R visual in Power BI.

Step -1: After installing Node.js, the first step is to install Power BI visual tools package by writing following in command prompt.

```
1 npm install -g powerbi-visuals-tools
```

To check the package is installed correctly, following command should return the information about Power BI Custom Visual Tools.

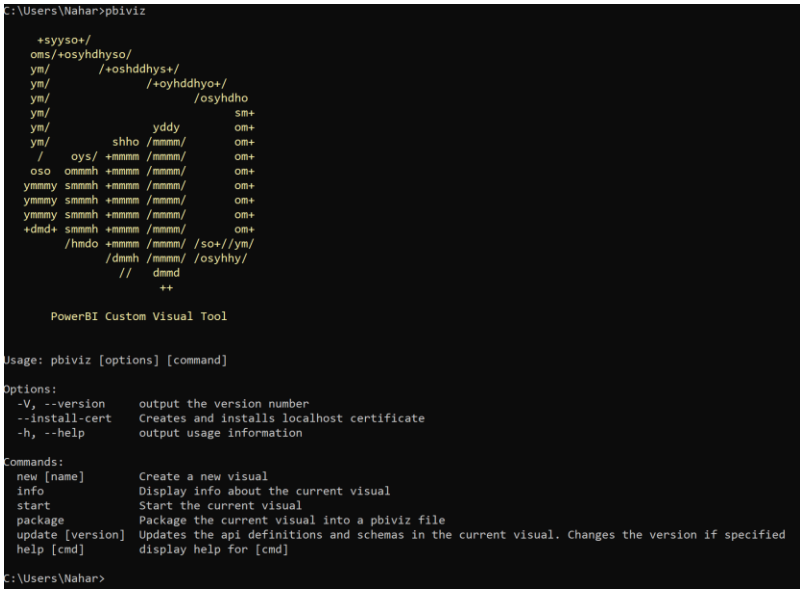


Figure 36: Installing PBIVIZ package in Power BI

Step-2: The following command creates a new HTML visual using "rhtml" template

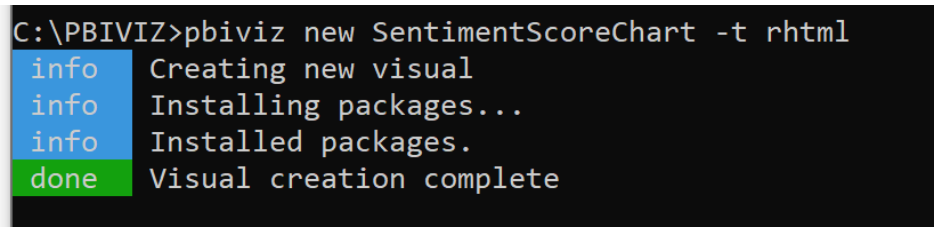
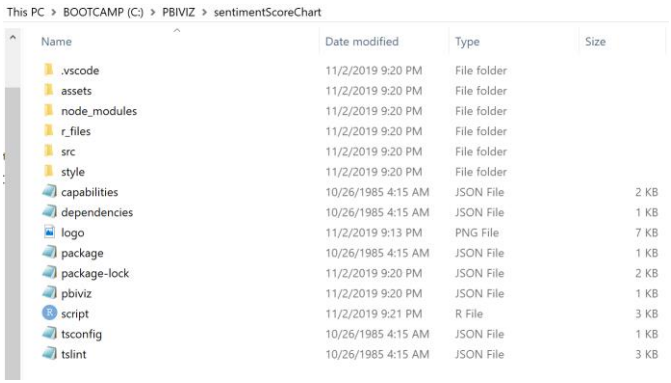


Figure 37: Creating a New R Custom Visualization in Power BI – Part1

After installing successfully, a new package named SentimentScoreChart will be created in the navigated folder. This directory contains all the files associated with the custom R visual. Inside the folder PBIVIZ package provides a default template with a 'script.r' file.



Step - 3: The script.r file is now customized to create the required R visual. There are three parts of this r script. The first part declares all the necessary R libraries to create the visual. The 'sentimentr' package is successfully worked here. To create a HTML output in R htmlwidgets package is used with ggplot2 and plotly package.

```
source('./r_files/flatten_HTML.r')

############### Library Declarations ###############
libraryRequireInstall("ggplot2");
libraryRequireInstall("plotly")
library(htmlwidgets)
library(dplyr)
library(sentimentr)
####################################################
```

The second part of script.r contains actual r code that transforms the data and plot the result. The dataset imported from power BI is stored inside the variable data frame "Values" with the columns in the original dataset.

```
################## Actual code ##################

input_dataset <- data.frame(text = as.character(Values$Text), stringsAsFactors=FALSE)
mytext <- get_sentences(input_dataset$text)
Sentiment = sentiment_by(mytext,by=NULL)
input_dataset <- dplyr::mutate(input_dataset, Senti_Score = round(Sentiment$ave_sentiment , 2 ))

sentiment.positive = subset(input_dataset, Senti_Score < -0.1 )
sentiment.negative = subset(input_dataset, Senti_Score > 0.1)
sentiment.neutral = subset(input_dataset, Senti_Score > - 0.1 & Senti_Score < 0.1)

N= nrow(input_dataset)
Npositive = nrow(sentiment.positive)
NNegative = nrow(sentiment.negative)
NNeutral =  nrow(sentiment.neutral)

dftemp=data.frame(topic=c("Positive", "Negative","Neutral"), number=c(Npositive,NNegative,NNeutral))

p <- plot_ly(data=dftemp, labels = ~topic, values = ~number, type = 'pie', textinfo = 'label+percent',hole = 0.4) %>%
  layout(title = '',
         xaxis = list(showgrid = TRUE, zeroline = TRUE, showticklabels = TRUE),
         yaxis = list(showgrid = TRUE, zeroline = TRUE, showticklabels = TRUE),
         legend = list(orientation = 'h',xanchor ='center', x =0.5))

################################################
```

The third and final part of script.r file calls a function from ggplot2 package, which plots the dataset using the Plotly package for converting it into HTML with zoom in/ out, auto scale etc. capabilities [31]. The function internalSaveWidget saves the widget and merge all the external files into one.

```
############# Create and save widget ###############

g = ggplotly(p);
internalSaveWidget(g, 'out.html');

####################################################
```

Step-4: After editing the script.r the next step is to edit the "pbiviz.json" to provide some basic information like description, author name etc. about the visual.

pbiviz - Notepad

File Edit Format View Help

```
{"visual":{
            "name":"sentimentScoreChart",
            "displayName":"SentimentScoreChart",
            "guid":"sentimentScoreChartDA344AD3EECA4E63ABB81D4251E651AC",
            "visualClassName":"Visual",
            "version":"1.0.0",
            "description":"doughnut chart using sentimentr package",
            "supportUrl":" your url goes here ",
            "gitHubUrl":""
          },
  "apiVersion":"2.6.0",
  "author":{
            "name":" author's name",
            "email":" author's email "
           },
  "assets":{
            "icon":"assets/logo.png"},
            "externalJS":null,
            "style":"style/visual.less",
            "capabilities":"capabilities.json",
            "dependencies":"dependencies.json",
            "stringResources":[]
           }
```

To change the icon of the visual, navigate to the assets folder and place the logo there (the default logo is icon.png). Also edit the visual.ts file in source folder to replace NodeListOf with HTMLCollectionOf in visual.ts.

visual - Notepad

File Edit Format View Help

```
        } catch (err) {
            return;
        }

        // if 'updateHTMLHead == false', then the co
        // this option allows loading and parsing of
        if (updateHTMLHead || this.headNodes.length
            while (this.headNodes.length > 0) {
                let tempNode: Node = this.headNodes.pop();
                document.head.removeChild(tempNode);
            }
            let headList: NodeListOf<HTMLHeadElement> = el.getElementsByTagName("head");
            if (headList && headList.length > 0) {
                let head: HTMLHeadElement = headList[0];
                this.headNodes = ParseElement(head, document.head);
            }
        }

        // update 'body' nodes, under the rootElement
        while (this.bodyNodes.length > 0) {
            let tempNode: Node = this.bodyNodes.pop();
            this.rootElement.removeChild(tempNode);
        }
        let bodyList: NodeListOf<HTMLBodyElement> = el.getElementsByTagName("body");
        if (bodyList && bodyList.length > 0) {
            let body: HTMLBodyElement = bodyList[0];
            this.bodyNodes = ParseElement(body, this.rootElement);
        }
```

Find

Find what: NodeListOf

Find Next

Cancel

Direction

Up Down

Match case

Wrap around

Step-5: Finally to create a .pbiviz file execute the 'pbiviz package' command from the root directory. Every time a file is updated, this command needs to be executed to update the chart in power BI.

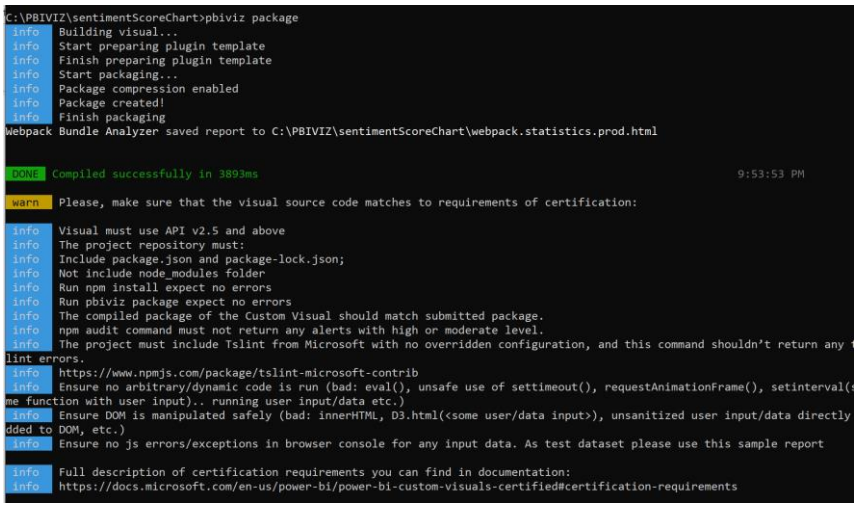
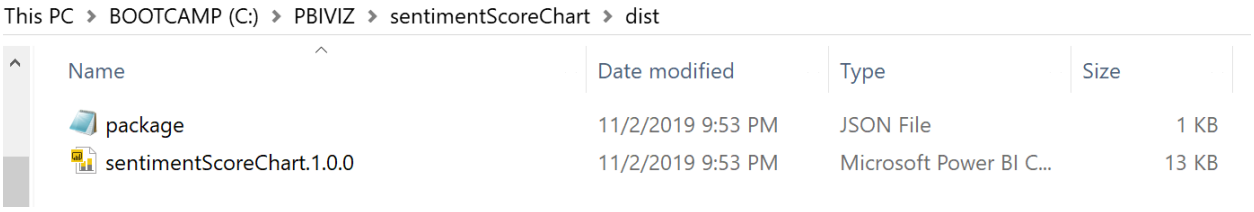


Figure 38: Creating a New R Custom Visualization in Power BI – Part 2

A new custom visual is now created in dist folder.



The chart is now can be added to power BI by import from file option.

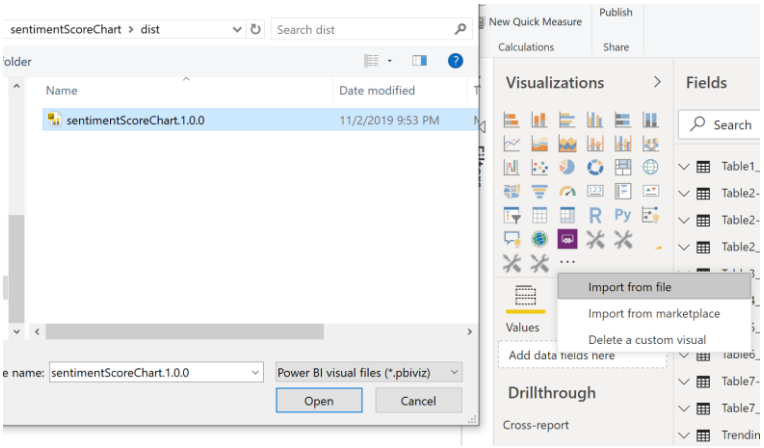


Figure 39: Importing a R Custom Visualization in Power BI – Part 1

After opening the .pbiviz file in power BI , select the visual and dataset from the right panel and the chart will be presented in power BI report panel.

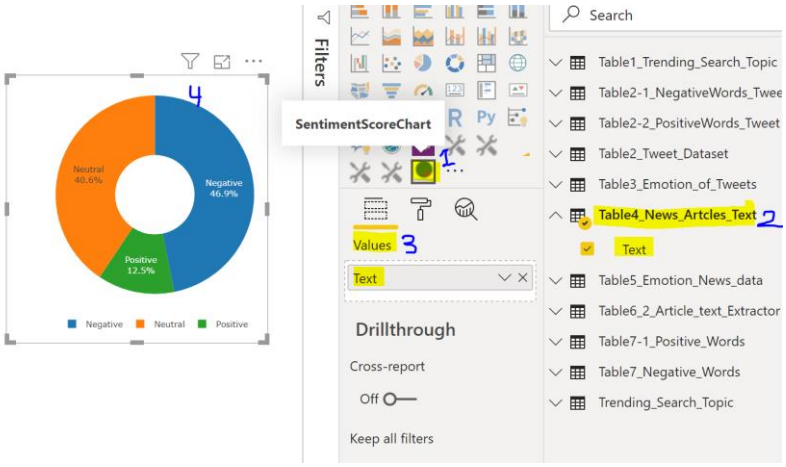


Figure 40: Importing a R Custom Visualization in Power BI – Part 2

6.3: Project Execution Timeline

Process	As Planned	As Implemented
Data collection	May, 2019	August, 2019
Integration of News and Social Media Data	June, 2019	September, 2019
Text Analysis	July, 2019	September, 2019
Initial Deployment of Dashboard to Power BI Service	August, 2019	October, 2019
Testing the Result	September, 2019	October, 2019
Final Report Writing	September, 2019	End of October, 2019
Capstone Defense	October, 2019	November 26th, 2019

Chapter 7

Limitations and Future Work

The text analytics application developed in this project provides a detailed implementation of most common text analysis techniques in an integrated platform of R, Python and Microsoft Power BI. The application developed is still crude and failed a few scenarios but is enough to test the usability of the concept. Many advancements can be further done in this project to make the outcomes refined. This chapter describes some of the limitations it includes and how the project can be extended in future.

7.1: Performance Issues with large datasets.

Following table represents the number of tweets and news articles used in the project:

No. of Tweets	No. of news article (full_text)	#of news article (headlines)
1000	10	30

For a importing a larger data set with more than 1000 tweets and 30 news articles, sometimes Power BI gives slow performance issues as it has been observed during implementation. As all of this data are imported in real-time, Power BI might face trouble in connecting to the web pages that leads to timeout during the request processing.

To opt-out of facing this issue in future, advanced data pipeline can be implemented to store the tweets and news data in a database table and use this table as a source to import data in power BI.

7.2: Limited Sharing of Data

Power BI Reports can be shared only with those users who have the same email domains or the ones who have their email domains listed in the organization's Office 365 tenant.

7.3: Data Quality Issue for News Articles

As, no universal solution is found to scrape only actual text data from multiple news articles, the dataset includes some unnecessary data with no evocative contribution to the analysis. Future research can be done to efficiently scrape only the meaningful text from these news URLs.

7.4: Data Refreshing in Power BI service with Personal Gateway

After publishing report to power BI service, data can be refreshed as long as required data accessibility credential is set up correctly. There are three main data access scenarios [32]:

- A dataset uses data sources that reside on-premises
- A dataset uses data sources in the cloud
- A dataset uses data from both, on-premises and cloud sources

If a Power BI dataset uses a data source that cannot be accessed over a direct network connection, a gateway connection is required for the dataset to enable a refresh schedule. Gateway is an application that can be installed on any local machine or servers for creating the connection and passing data through.

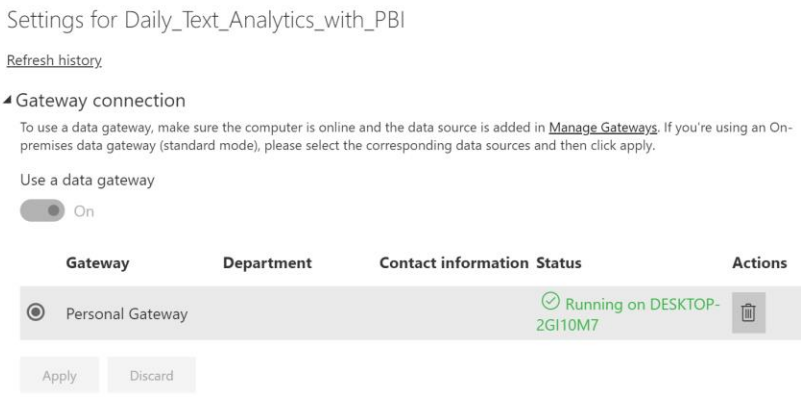
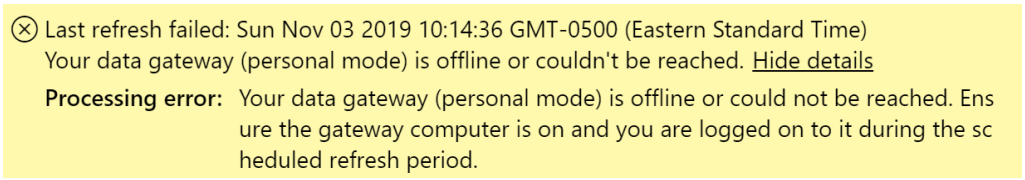


Figure 41: Adding a Personal Gateway to the Dataset in Power BI Service

Although the project uses web-based Twitter and news data, but R and python script is used as data source to transform and import data in power BI. Currently, R and Python scripts are supported for data gateway personal mode only [33]. That means, to perform a successful scheduled data refresh the gateway status must be online (a machine that runs 24/7 hrs. a day), otherwise the refresh will be failed. The following is a screenshot of a failed data refresh scenario.



Future studies can be done here by going serverless to build a successful schedule data refresh. As a part of Microsoft business suit, one possible option to go serverless is to use Azure Function and Storage with power BI. Azure Functions is a serverless compute service that

runs a timer-triggered code on a periodic basis and add a connection (or binding) to an Azure storage account. The storage account could be used to import data sources to power BI. Azure Function supports a variety of languages, but currently R is not supported [34]. As, R has a wide variety of available text analytics packages compared to Python, I could not implement the whole system using Azure Function. In future it is expected that the supported language list in Azure Function will grow, and R script can also be executed in a serverless fashion. A Personal Gateway can also be configured in the storage server that is always up and running, as well as with a more powerful machine with more CPU and RAM to cater for data refresh queries.

Conclusion

The objective of this capstone work was to implement a fully automatic and repeatable text analytics application using Power BI platform. Therefore, I designed the experiments and followed necessary steps. First, data about the most frequently searched term in Google over the past 24 hours across US geographic location is collected in Power BI. Then, Twitter feeds related to the trending search topic is captured using Twitter Streaming API. Using Python scripts in power BI, text data from multiple news articles about the trending search term is also extracted. The next step was to remove punctuation, numbers, HTML links, extra spaces, etc. from the captured text data to avoid unnecessary data into the analysis module. Compatibility analysis of currently existing text analytics module with power BI is performed by evaluating various advantages and disadvantages of these modules. After rigorous comparison, R libraries `sentimentr`, `syuzhet` and `tidytext` is found to be most suitable because of the superiority over other library's integration capability in power BI. Besides, these libraries are completely free to use, widely referenced and with no limitation of data size. By using `sentimentr`, `syuzhet` and `tidytext` libraries, I was able to create a report and share it with other power BI users. As Google trending search term is very dynamic and changes throughout the day, the dataset for the report is refreshed three times a day with an interval of eight hours. Therefore, the report remains compatible with the turbulent trends of modern time.

Most text analytics works have been performed using R programming languages. Very few works have integrated text analytics application with power BI, as it is a more recent tool than other Business Intelligence tools in market. Power BI is constantly being updated and have many advantages over past BI tools. My decision to use power BI is primarily due to two factors. Firstly, Power BI can easily integrate R and Python programming languages the two most popular data analysis platform. Secondly, the final reports can be easily shared with other users via Power BI service. My work is unique to the fact that it can work with various source data like Twitter and online news articles simultaneously.

Even though a good number of studies have been done to analyze Twitter data, I do not see any work to evaluate text analytics by relying on Twitter and news data together. I have developed a unique system to provide a clearer view on both public and media perception about a certain topic. This project work can serve as a reference document for professionals, developer or analyst to leverage the power of text analytics in conjunction with power BI. The project work can be further modified to include more advanced text analytics module e.x., Topic modeling, text categorizations, thematic analysis etc. In an age where data is constantly being collected, the project can serve as a prototype for many future developers to build an end-to-end text analytics application through power BI to gain valuable insights from organization's data and act upon them to improve overall operation.

References

1. http://www.internetlivestats.com/google-search-statistics/
2. https://searchcontentmanagement.techtarget.com/definition/Microsoft-Power-BI
3. https://en.wikipedia.org/wiki/Power_BI
4. https://mindmajix.com/blogs/images/powerbi-architecture.jpg
5. https://info.microsoft.com/ww-landing-gartner-mq-bi-analytics-2019.html?LCID=EN-US
6. https://softcrylic.com/blogs/top-8-advantages-power-bi-tableau/
7. https://www.geekwire.com/2017/tableau-software-continues-move-to-subscription-model-releases-new-prices-for-data-visualization-products/
8. https://www.lexalytics.com/technology/text-analytics
9. https://www.cs.uic.edu/~liub/FBS/SentimentAnalysis-and-OpinionMining.pdf
10. https://docs.microsoft.com/en-us/power-bi/desktop-r-scripts
11. https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm
12. https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/
13. https://docs.microsoft.com/en-us/azure/cognitive-services/text-analytics/overview
14. https://cran.r-project.org/web/packages/RSentiment/index.html
15. https://cran.r-project.org/web/packages/sentimentr/index.html
16. http://www.purl.org/net/NRCemotionlexicon
17. https://cran.r-project.org/web/packages/syuzhet/index.html
18. http://theautomatic.net/2019/01/19/scraping-data-from-javascript-webpage-python/
19. Wickham Hadley, " Tidy Data," *Journal of Statistical Software,* August 2014, Volume 59, Issue 10.
20. https://www.tidytextmining.com/tidytext.html#the-unnest_tokens-function
21. https://go.medallia.com/rs/669-VLQ-276/images/Medallia-Why-Text-Analytics.pdf
22. https://monkeylearn.com/sentiment-analysis/
23. https://docs.microsoft.com/en-us/power-bi/power-bi-tutorial-q-and-a
24. https://powerbi.microsoft.com/fr-fr/blog/introducing-power-bi-data-prep-wtih-dataflows/
25. K. Welbersa, W.V. Atteveldtb, and K. Benoit, " Text Analysis in R *", Communications Methods and Measures* 2017, VOL. 11, NO. 4, 245–265
26. https://www.edureka.co/blog/web-scraping-with-python/
27. https://azure.microsoft.com/en-us/pricing/details/cognitive-services/text-analytics/

28. M. Naldi, " A review of sentiment computation methods with R packages ", January 2019
29. https://docs.microsoft.com/en-us/power-bi/service-r-packages-support
30. https://community.powerbi.com/t5/Desktop/Error-in-running-an-R-script-using-sentimentr-library-in-PowerBI/m-p/539046
31. https://github.com/Microsoft/PowerBIvisuals/blob/master/RVisualTutorial/CreateRHTML.md
32. https://docs.microsoft.com/en-us/power-bi/refresh-data
33. https://docs.microsoft.com/en-us/power-bi/service-gateway-personal-mode
34. https://docs.microsoft.com/en-us/azure/azure-functions/supported-languages
35. https://www.educba.com/power-bi-vs-tableau/
36. https://www.netowl.com/2018/06/29/calling-the-election-sentiment-analysis-for-election-campaign-monitoring