National College of Ireland

BSc in Computing

2017/2018

Keith Driscoll

X14346081

X14346081@student.ncirl.ie

Trendicator – Fashion Trend Detection

Technical Report

Table of Contents

Executive Summary

The main objective of the Trendicator project is to detect possible current fashion trends via fashion articles, blogs and tweets. The project was undertaken to produce understandable visual representations of emerging fashion-related trends detected online.

Trendicator performs text analysis on gathered data from online fashion sources such as blogs, digital magazines and Twitter. Using this data, produce graphical outputs used to identify and interpret any fashion-related trends.

The Trendicator platform is hosted on the ShinyApps.io web-service, allowing seamless integration between Trendicator's MySQL Database hosted on the Microsoft Azure Cloud.

The steps of this project are discussed in detail through applying the Knowledge Discovery in Databases(KDD) along with suggestions for further work.

1 Introduction

1.1 Background

Fashion has always played a major role in modern cultures, highly impacted by factors including (but not limited to) public figures, politics, economy and religion. Trends appear to be highly impacted by public if masses are influenced by the key factors.

With the creation of the internet, fashion trends have become more unpredictable than ever. Trends were no longer created by a group of individuals; the public had a huge involvement towards emerging fashion trends.

"You no longer need a powerful position at a fashion house to influence style; all you need is your phone." (Fern Seto).

1.2 Aims

During and in between fashion seasons, trends emerge and disappear. Clothing-retail companies such as H&M and Zara are known for their fast-fashion clothing use big data analytics to produce relevant clothing for fashion shoppers.

Trendicator aims to detect emerging trends related to fashion which could be used to influence fashion decisions for individuals or small-businesses. These Individuals would be digital influencers looking for which fashion trend to dress up in. Small fashion businesses managing their social media accounts would be looking to release content which reflects positively on current trends.

Aim 1: Obtain relevant text resources from multiple fashion-relatable sources on the internet.

Aim 2: Pre-processing the data for analysis.

Aim 3: Data reduction and transformation to find useful features.

Aim 4: Apply algorithms for text classification and predictions.

Aim 5: Visualization of data and reporting of findings.

1.3 Technologies

R is the language of choice used within RStudio for data collection and pre-processing. The text will be gathered from uniform resource locators(URLs) and Twitter using a scraping script written in R, allowing the acquisition of text from varying websites and twitter accounts.

The Twitter application programming interface(API) is used to stream tweets with a relevance to the term ‘fashion’.

All data gathered is stored in a comma separated values(csv) file, which is then imported into a relational MySQL database hosted on the Microsoft Azure Cloud.

An RShiny application with an established connection to the Azure database is used for data visualization in a web app.

Microsoft Excel is used to generate statistical tests using the data analytics tool pack.

2 System

2.1 Functional requirements

Gathering of data from multiple relatable sources on the internet. Fashion related text is scraped from multiple websites and tweets using R and stored in a data frame.

Pre-processing collected data and clean text data which is imported into R so the data is prepared for analysis.

Storing the gathered data in a relation MySQL database which will make it easier to perform data transformation techniques. Hosted on the cloud via Microsoft Azure.

Performing text analysis which turns the text data into quantitative data allowing for document term matrices and sentiment analysis.

Creating data models with the transformed text to produce a topic model and a linear regression model. Evaluating the accuracy of the linear regression model using Mean Absolute Error(MAE) and Root Mean Squared Error(RMSE) metrics.

Data is visualized via graphical representations to help better interpret results.

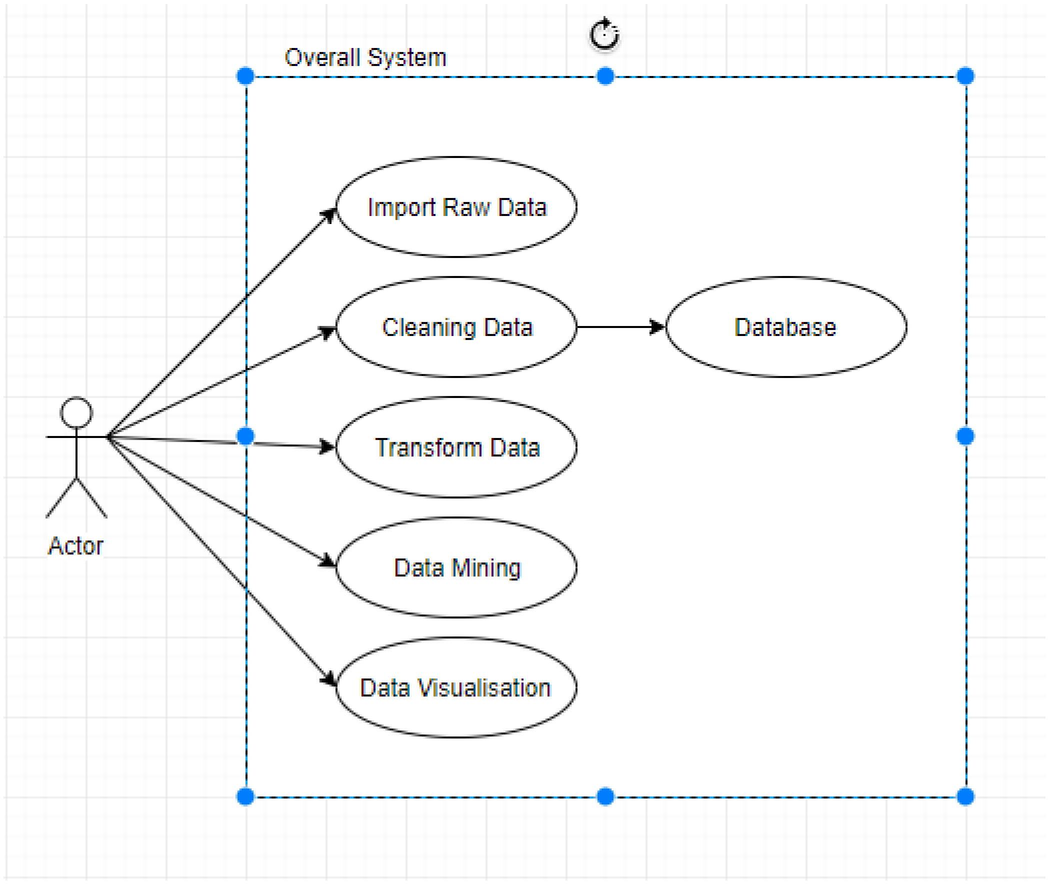
2.1.1 Application Program Interfaces(API)

Twitter API is used to allow for authentication and the ability to collect tweets from Twitter based on specified criteria from Tweepy.

Microsoft Azure is used to deploy the application on the cloud an allow seamless integration between the database and Tableau Online.

2.1.2 Use Case

My Use Case Diagram of an actor(User) interacting with the overall system.



2.2 Non-Functional Requirements

Specifies any other non-functional attributes required by the system. Examples are provided below.

2.2.1 Data requirements

For this project, data scraped from online sources must specifically allow their website data to be collected for personal or educational purposes. This is confirmed through inspection of a website's Term of Service(ToS) or Privacy Policies before scraping is performed. Data collected being passed through R should be in the form of a data frame.

2.2.2 Environmental requirements

To run this project the minimum requirements are:

An intel-compatible platform running Windows 2000, XP/2003/Vista/7/8/10 with administrative privileges. At least 32MB of RAM and 1GB of disk space. Connection to the internet via Ethernet or Wi-Fi. A mouse and keyboard.

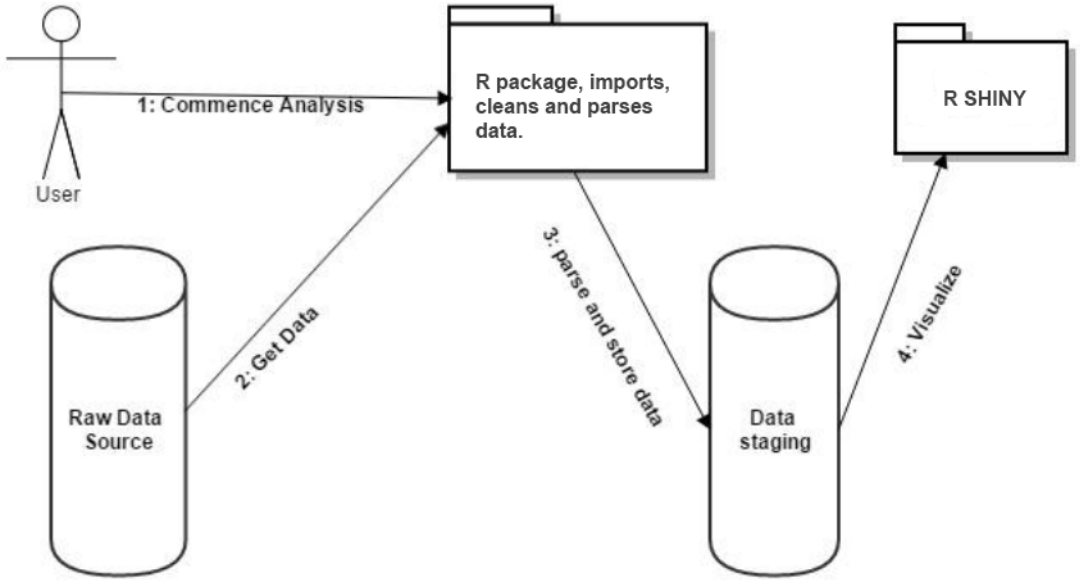
The environment used to run this project are:

A Windows 10 Operating System with 8 CPU cores and 16 Threads; administrative privileges granted. 16GB of DDR4 RAM with a 250GB SSD. Connection to the internet via Ethernet or Wi-Fi. Mouse and keyboard.

2.2.3 Security requirement

Trendicator does not collect personal information from it's data sources therefore no personal information is being stored. All data scraped from online sources is stored in a relational MySQL database on the Microsoft Azure Cloud.

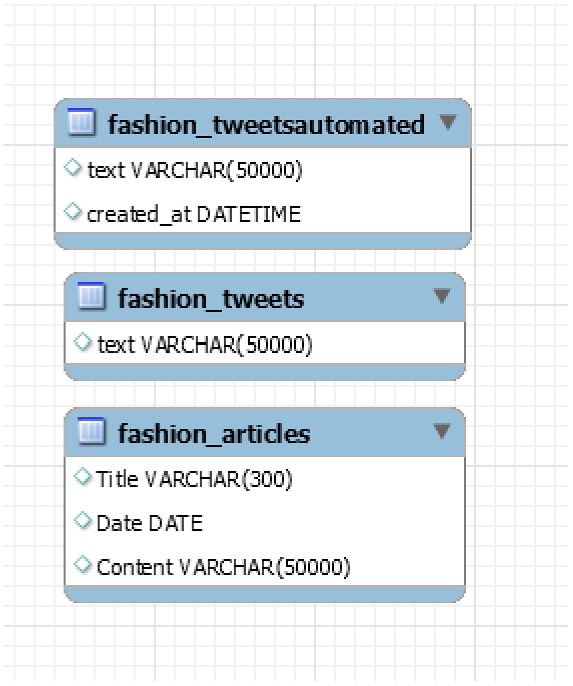
2.3 Design and Architecture



The above is a logical architecture view of the overall system. Firstly, the user commences analysis by loading required R packages. Using R, data is extracted from online sources and parsed into a data frame, then cleaned. Once the data has been meaningfully pre-processed, this data is then inserted into a relational database hosting on the Azure Cloud. Data stored in the Azure cloud is accessed via the R Shiny Application, which performs data mining techniques and visualises results via an interactive web interface.

2.3.1 Database Design

(EER Diagram)



Trendicator's database consists of three tables stored within a MySQL database hosted on the Microsoft Azure Cloud.

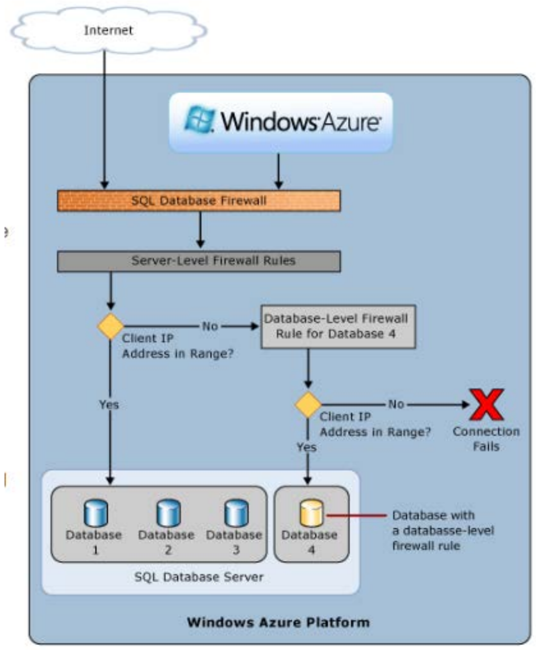
Fashion_tweets stores all the scraped and cleaned twitter data.

Fashion_tweetsautomated stores all the scraped and cleaned twitter data including dates of tweets. This table was primarily used for testing purposes without jeopardizing the data used by the RShiny Application.

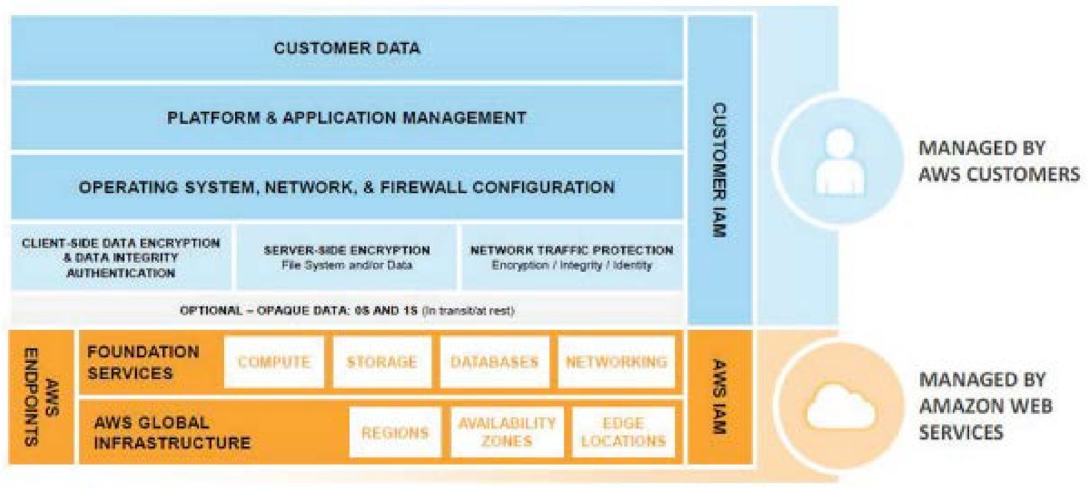
Fashion_articles stores all the scraped and cleaned data from TheFashionSpot, including article titles, dates and text.

2.3.2 Security Architecture

By default, Azure will block all income external connections unless specifically allowed. This is done through whitelisting specific IP addresses in the firewall.

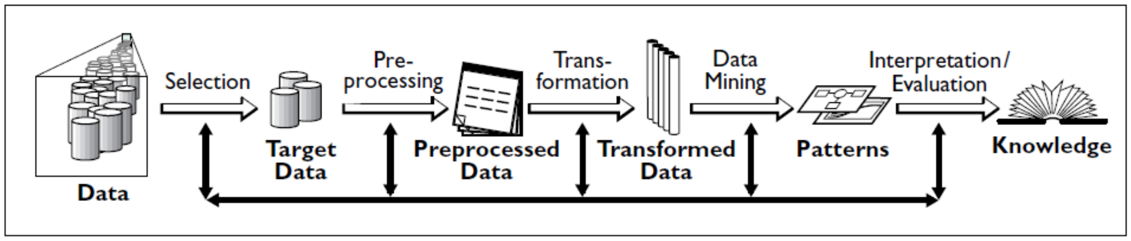

Internet
Windows Azure
SQL Database Firewall
Server-Level Firewall Rules
Client IP Address in Range?
No
Database-Level Firewall Rule for Database 4
Yes
Client IP Address in Range?
No
Connection Fails
Yes
Database 1
Database 2
Database 3
Database 4
Database with a databasse-level firewall rule
SQL Database Server
Windows Azure Platform

The interactive web application is possible via RShinyApps.io which runs in a Amazon Web Services(AWS), inheriting Amazon’s security policies in that respect.


CUSTOMER DATA
PLATFORM & APPLICATION MANAGEMENT
OPERATING SYSTEM, NETWORK, & FIREWALL CONFIGURATION
CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION
SERVER-SIDE ENCRYPTION File System and/or Data
NETWORK TRAFFIC PROTECTION Encryption / Integrity / Identity
OPTIONAL – OPAQUE DATA: 0S AND 1S (in transit/at rest)
CUSTOMER IAM
MANAGED BY AWS CUSTOMERS
AWS ENDPOINTS
FOUNDATION SERVICES
COMPUTE
STORAGE
DATABASES
NETWORKING
AWS GLOBAL INFRASTRUCTURE
REGIONS
AVAILABILITY ZONES
EDGE LOCATIONS
AWS IAM
MANAGED BY AMAZON WEB SERVICES

2.4 *Methodology*

Figure 1. Overview of the steps constituting the KDD process



Knowledge discovery in databases (KDD) is the process of discovering useful knowledge from a collection of data. My objectives and requirements for my project are as follows.

Data Selection - Gathering data from multiple sources on the internet from Europe.

Digital Fashion Magazines – Gathering information collected from online fashion magazines reporting on seasonal designer catwalk releases. Online Fashion Blogs – Gathering information collected from online fashion blogs used to gather fashion information inspired by designer catwalk releases. Online Twitter Conversations – Designers try to create pieces that they believe will trend with the public. Collecting data from fashion related conversations via Twitter will be used to help identify fashion trends.

Data Pre-processing – Data is explored to decide which actions to take for cleaning. This project's Twitter data contained many fields which were disregarded such as account_id, status_id, screen_name. For this project only, text and created_at fields were used. With t data scraped from websites, only data was extracted from specific website elements containing titles, article_text and date. This pre-processed data is then stored in a database.

Data Transformation – The data is used to run statistical procedures such as Document-term-matrices, word stemming and tokenization. This is a vital step for feature extraction, discovering new data from existing data.

Data Mining – Applying topic modelling and term frequency modelling to produce meaningful insights into the data.

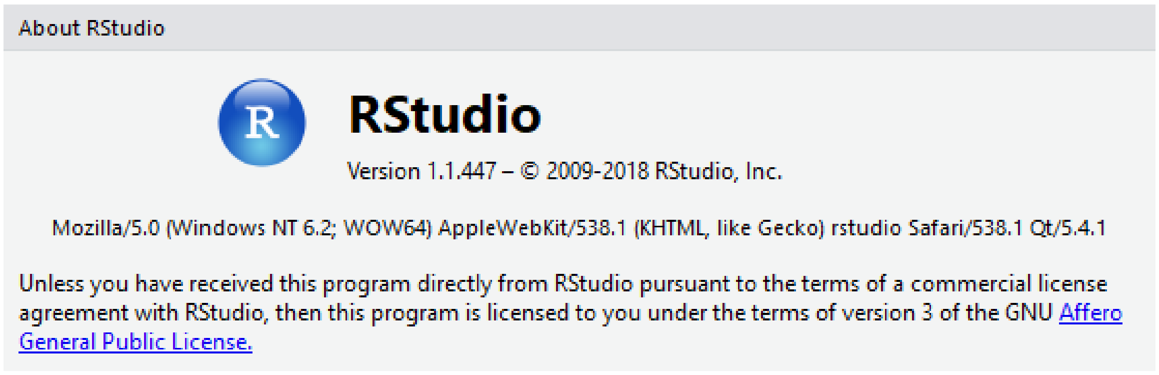
Data Interpretation - It is crucially important for the data to be interpreted and correctly reported, insuring the data is sampled accordingly.

2.5 Implementation

2.5.1 Domain Knowledge

The preliminary step in starting a data analysis project is to know which domain is being analysed and the objective hoping to be achieved in this field. The chosen field of social media is a broad topic but with a key focus on the aspect of fashion. Trendicator aims to analyse current trends in fashion using topic modelling and term frequency, identifying which fashion trends are occurring. The implementation process follows the KDD principles towards discovery of useful knowledge from data.

2.5.2 Data Gathering


About RStudio

RStudio

Version 1.1.447 – © 2009-2018 RStudio, Inc.

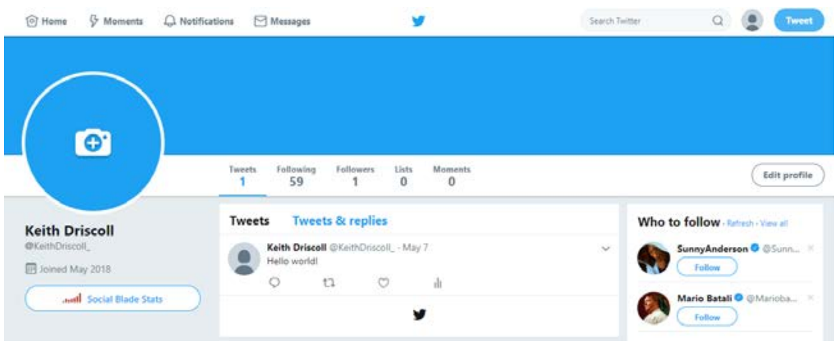
Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/538.1 (KHTML, like Gecko) rstudio Safari/538.1 Qt/5.4.1

Unless you have received this program directly from RStudio pursuant to the terms of a commercial license agreement with RStudio, then this program is licensed to you under the terms of version 3 of the GNU Affero General Public License.

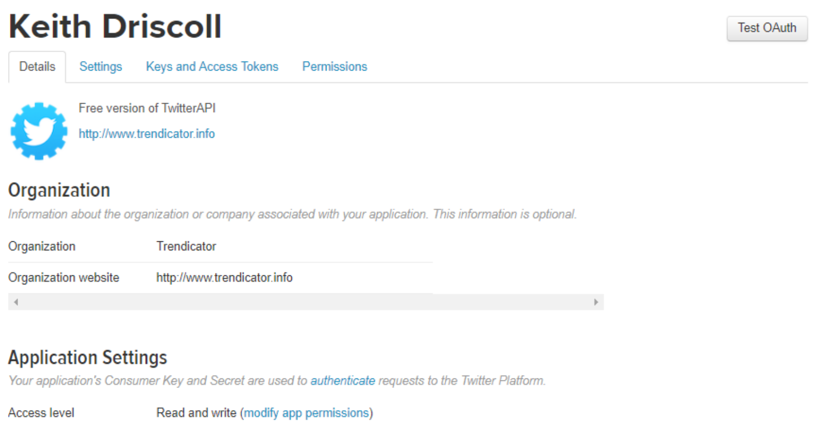
For Trendicator, data was gathered from two sources. A fashion blog called TheFashionSpot and Twitter. Using RStudio to perform the data gathering tasks required.

To collect tweets from Twitter, Twitter and Twitter Apps accounts were needed.

Twitter Account:



Twitter Apps Account


Keith Driscoll

Test OAuth

Details | Settings | Keys and Access Tokens | Permissions

Free version of TwitterAPI
http://www.trendicator.info

Organization

Information about the organization or company associated with your application. This information is optional.

Organization	Trendicator
Organization website	http://www.trendicator.info

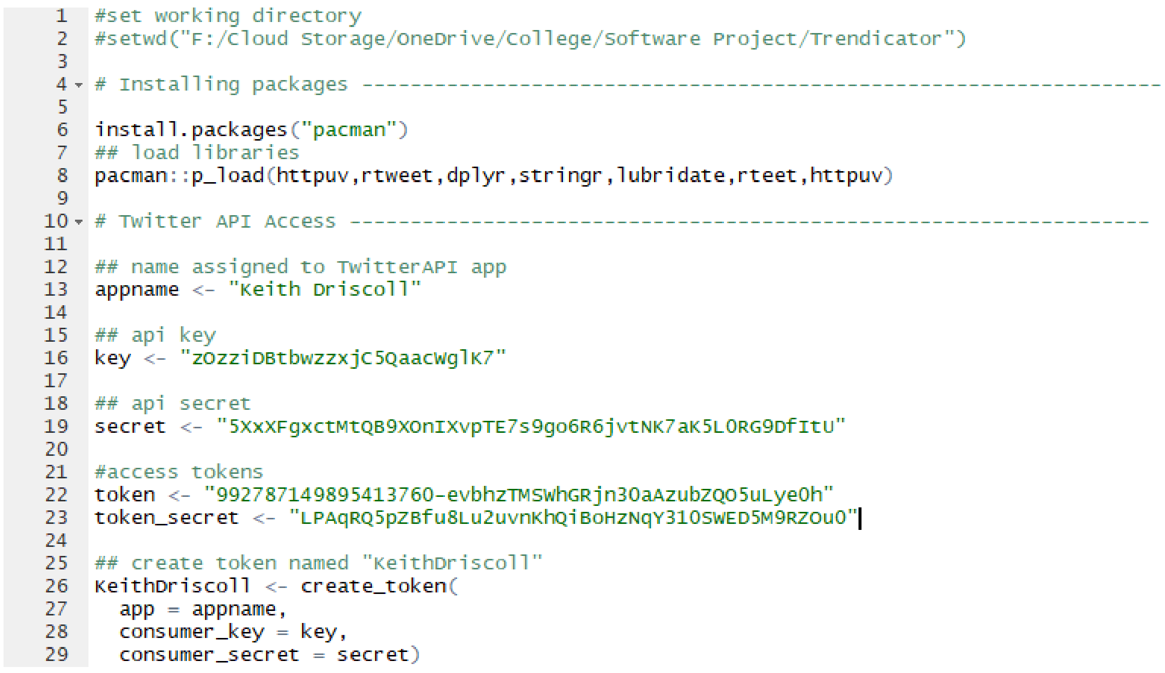
Application Settings

Your application's Consumer Key and Secret are used to authenticate requests to the Twitter Platform.

Access level	Read and write (modify app permissions)

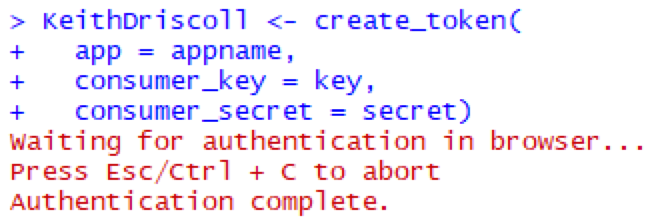
Once these accounts were created and API Keys and Access Tokens were generated, RStudio attempts a connection via TwitterAPI.

Firstly, all required packages must be installed. This project used a CRAN-R package named ‘pacman’ which is a package management tool, allowing for multiple packages to be installed and libraries loaded in just a single line of code.



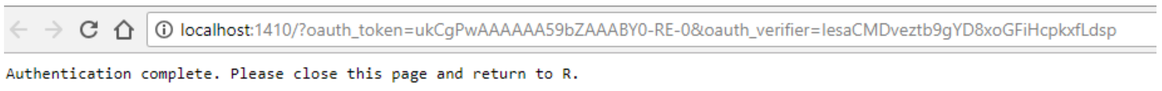
```
#set working directory
#setwd("F:/Cloud Storage/OneDrive/College/Software Project/Trendicator")

# Installing packages -------------------------------------------------------------------

install.packages("pacman")
## load libraries
pacman::p_load(httpuv,rtweet,dplyr,stringr,lubridate,rteet,httpuv)

# Twitter API Access -------------------------------------------------------------------

## name assigned to TwitterAPI app
appname <- "Keith Driscoll"

## api key
key <- "zOzziDBtbwzzxjC5QaacWglK7"

## api secret
secret <- "5XxXFgxctMtQB9XOnIXvpTE7s9go6R6jvtNK7aK5LORG9DfItU"

#access tokens
token <- "992787149895413760-evbhzTMSWhGRjn3OaAzubZQO5uLye0h"
token_secret <- "LPAqRQ5pZBfu8Lu2uvnKhQiBoHzNqY31OSWED5M9RZOuO"

## create token named "KeithDriscoll"
KeithDriscoll <- create_token(
  app = appname,
  consumer_key = key,
  consumer_secret = secret)
```

Twitter API access details were stored within variables in R to more easily create an access token. Once this access was run, authentication is attempted.



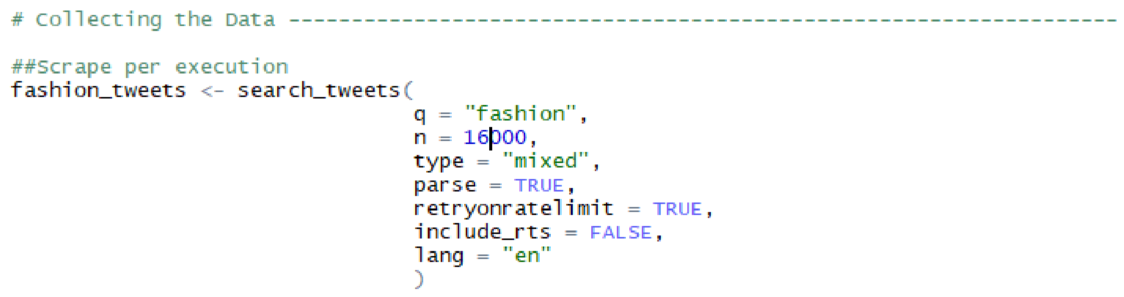
```
> KeithDriscoll <- create_token(
+   app = appname,
+   consumer_key = key,
+   consumer_secret = secret)
Waiting for authentication in browser...
Press Esc/Ctrl + C to abort
Authentication complete.
```

Successful authentication:



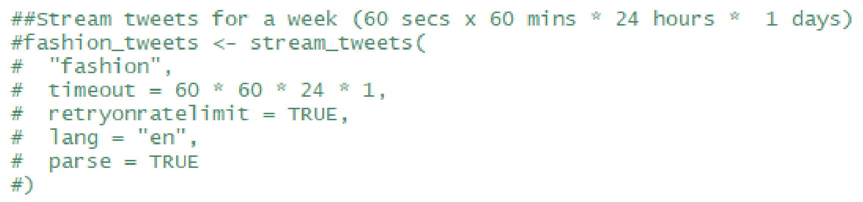
```
localhost:1410/?oauth_token=ukCgPwAAAAAA59bZAAABY0-RE-0&oauth_verifier=IesaCMDveztb9gYD8xoGFiHcpkxfLdsp
Authentication complete. Please close this page and return to R.
```

Two methods of scraping tweets were tried and tested. The first method involves scraping a specified number of tweets. The basic TwitterAPI access allows up to 18000 calls every 15 minutes, if that limit is reached the connection is terminated. Setting the limit of calls to 16000 prevents this from occurring.



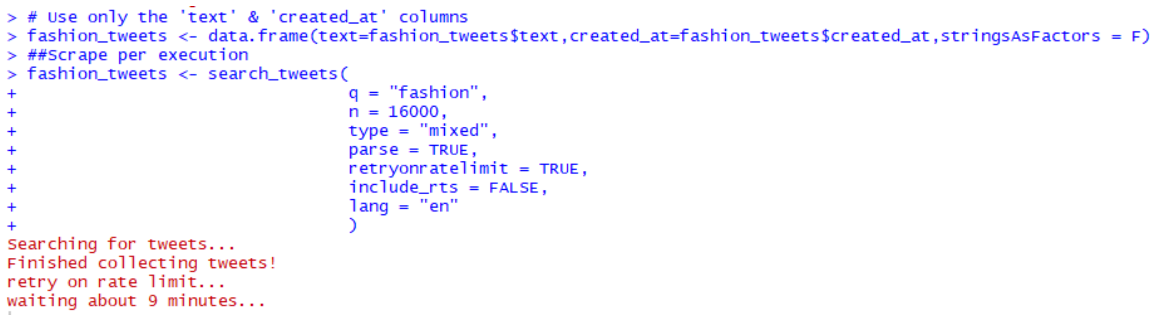
```
# Collecting the Data -----------------------------------------------------------------

##Scrape per execution
fashion_tweets <- search_tweets(
                                q = "fashion",
                                n = 16000,
                                type = "mixed",
                                parse = TRUE,
                                retryonratelimit = TRUE,
                                include_rts = FALSE,
                                lang = "en"
                                )
```

The second method is Tweet streaming which will continuously scrape tweets. This methodology was not mainly used as cleaning live streamed tweets in R is quite difficult to achieve. This was not a functional requirement of Trendicator so it was therefore not achieved.



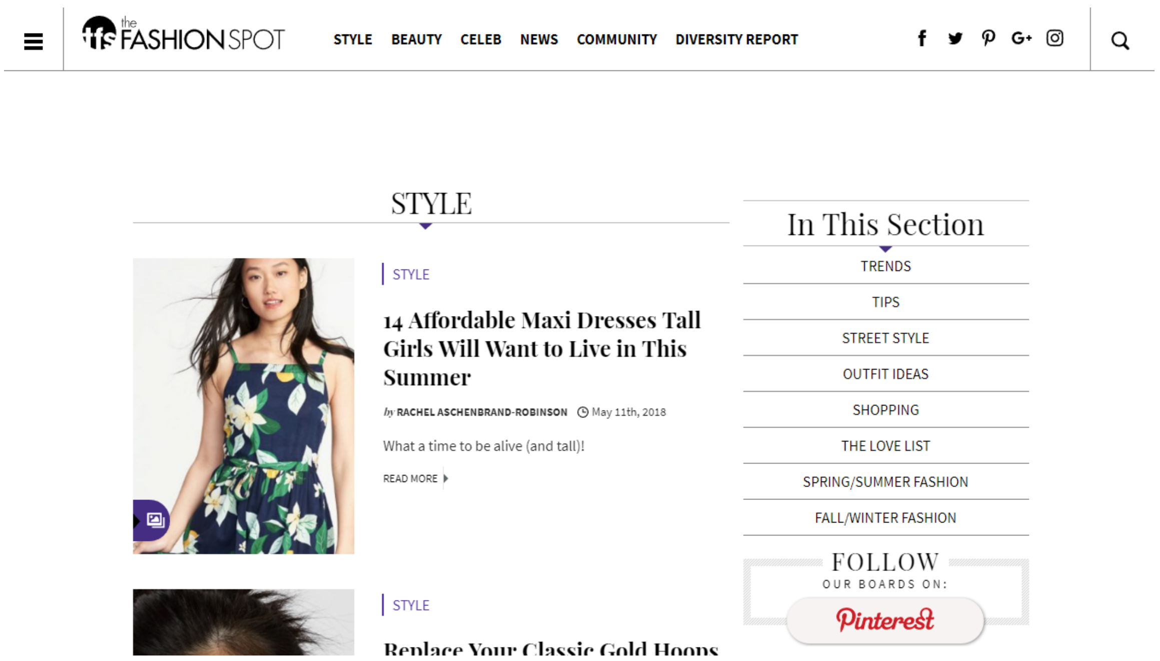
```
##Stream tweets for a week (60 secs x 60 mins * 24 hours *  1 days)
#fashion_tweets <- stream_tweets(
#  "fashion",
#  timeout = 60 * 60 * 24 * 1,
#  retryonratelimit = TRUE,
#  lang = "en",
#  parse = TRUE
#)
```

A termination of connection can then only be possible if the script is run again within 15 minutes, the limit will soon be reached. ‘RetryOnRateLimit = True’ is set in place to attempt a reconnection to collect more tweets until the amount is reached.



```
> # Use only the 'text' & 'created_at' columns
> fashion_tweets <- data.frame(text=fashion_tweets$text,created_at=fashion_tweets$created_at,stringsAsFactors = F)
> ##Scrape per execution
> fashion_tweets <- search_tweets(
+                                 q = "fashion",
+                                 n = 16000,
+                                 type = "mixed",
+                                 parse = TRUE,
+                                 retryonratelimit = TRUE,
+                                 include_rts = FALSE,
+                                 lang = "en"
+                                 )
Searching for tweets...
Finished collecting tweets!
retry on rate limit...
waiting about 9 minutes...
```

Next a Fashion Blog named TheFashionSpot was scraped. Before the script for scraping data from this source was created, the website’s content structure was first inspected.


STYLE BEAUTY CELEB NEWS COMMUNITY DIVERSITY REPORT

STYLE

STYLE

14 Affordable Maxi Dresses Tall Girls Will Want to Live in This Summer

by RACHEL ASCHENBRAND-ROBINSON May 11th, 2018

What a time to be alive (and tall)!

READ MORE

STYLE

Replace Your Classic Gold Hoops

In This Section

TRENDS

TIPS

STREET STYLE

OUTFIT IDEAS

SHOPPING

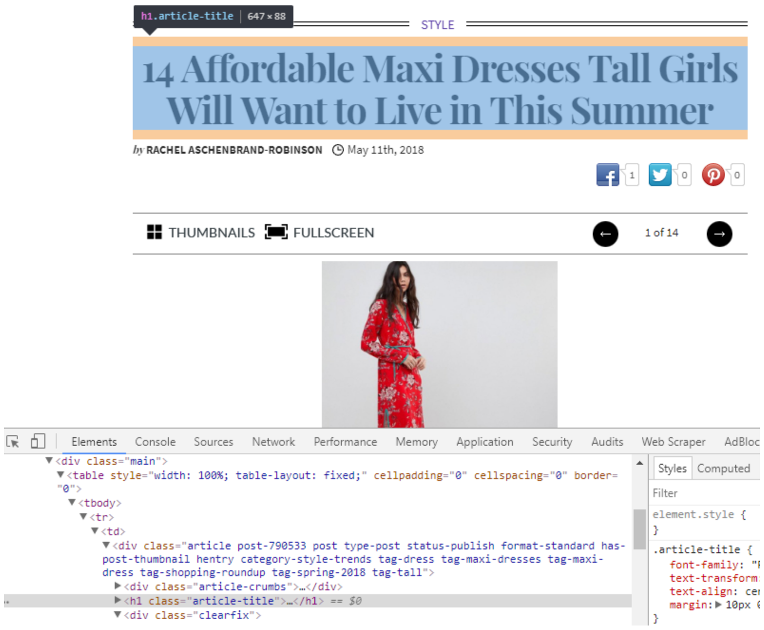
THE LOVE LIST

SPRING/SUMMER FASHION

FALL/WINTER FASHION

FOLLOW OUR BOARDS ON: Pinterest

The HTML tags used for specific sections I wish to scrape data from must be identified. The HTML tag for an article title was found via Google Chrome's developer view.


h1.article-title | 647 × 88

STYLE

14 Affordable Maxi Dresses Tall Girls Will Want to Live in This Summer

by RACHEL ASCHENBRAND-ROBINSON May 11th, 2018

THUMBNAILS FULLSCREEN 1 of 14

Elements Console Sources Network Performance Memory Application Security Audits Web Scraper AdBloc

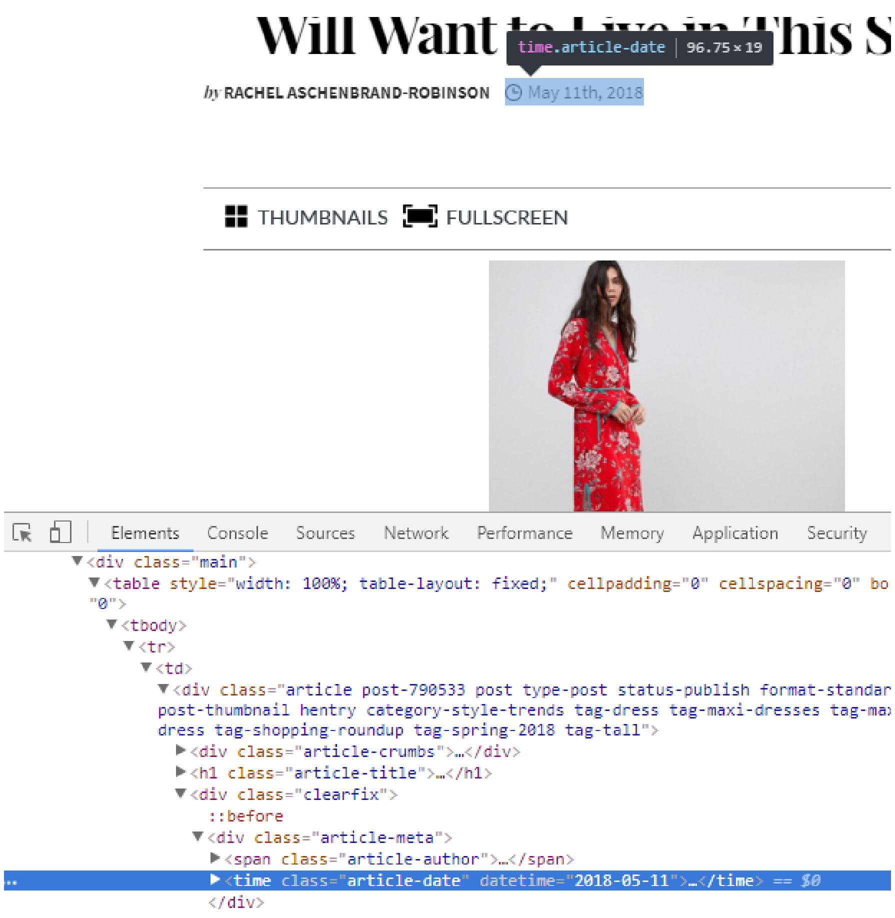
```
▼<div class="main">
  ▼<table style="width: 100%; table-layout: fixed;" cellpadding="0" cellspacing="0" border=
  "0">
    ▼<tbody>
      ▼<tr>
        ▼<td>
          ▼<div class="article post-790533 post type-post status-publish format-standard has-
          post-thumbnail hentry category-style-trends tag-dress tag-maxi-dresses tag-maxi-
          dress tag-shopping-roundup tag-spring-2018 tag-tall">
            ▶<div class="article-crumbs">…</div>
            ▶<h1 class="article-title">…</h1> == $0
            ▼<div class="clearfix">
```

Styles Computed

Filter

```
element.style {
}
.article-title {
  font-family: "
  text-transform
  text-align: ce
  margin:▶ 10px
}
```

Again, using Google Chrome's developer view, the HTML tag for an article's date was found.



Will Want to Live in This S

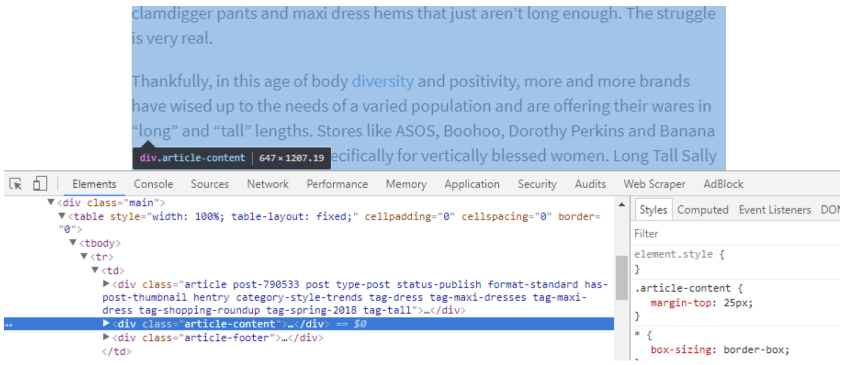
time.article-date | 96.75 × 19

by RACHEL ASCHENBRAND-ROBINSON May 11th, 2018

THUMBNAILS FULLSCREEN

```
▼<div class="main">
  ▼<table style="width: 100%; table-layout: fixed;" cellpadding="0" cellspacing="0" bo
  "0">
    ▼<tbody>
      ▼<tr>
        ▼<td>
          ▼<div class="article post-790533 post type-post status-publish format-standar
          post-thumbnail hentry category-style-trends tag-dress tag-maxi-dresses tag-ma
          dress tag-shopping-roundup tag-spring-2018 tag-tall">
            ▶<div class="article-crumbs">…</div>
            ▶<h1 class="article-title">…</h1>
            ▼<div class="clearfix">
                ::before
              ▼<div class="article-meta">
                ▶<span class="article-author">…</span>
                ▶<time class="article-date" datetime="2018-05-11">…</time> == $0
              </div>
```

And finally, the HTML tag used for the article's main body of text was found.



clamdigger pants and maxi dress hems that just aren't long enough. The struggle is very real.

Thankfully, in this age of body diversity and positivity, more and more brands have wised up to the needs of a varied population and are offering their wares in "long" and "tall" lengths. Stores like ASOS, Boohoo, Dorothy Perkins and Banana

div.article-content | 647 × 1207.19 ecifically for vertically blessed women. Long Tall Sally

```
▼<div class="main">
  ▼<table style="width: 100%; table-layout: fixed;" cellpadding="0" cellspacing="0" border=
  "0">
    ▼<tbody>
      ▼<tr>
        ▼<td>
          ▶<div class="article post-790533 post type-post status-publish format-standard has-
          post-thumbnail hentry category-style-trends tag-dress tag-maxi-dresses tag-maxi-
          dress tag-shopping-roundup tag-spring-2018 tag-tall">…</div>
          ▶<div class="article-content">…</div> == $0
          ▶<div class="article-footer">…</div>
          </td>
```

```
element.style {
}
.article-content {
  margin-top: 25px;
}
* {
  box-sizing: border-box;
```

Since one of Trendicator's requirements is to detect fashion trends, only the style/fashion section of this particular website was scraped. To ensure only articles in this section of the website was scraped, R was used to collect the URLs of articles in that specific section. Storing the list into a data frame. This snippet of code is an example of gathering this list of URLs until 1000 were found.

```
## create data frame to store a list
article_url <- list()

## this loops through 165 style pages to get the url to the articles.
## I used 165 because on the homepage, there are 20 links and on subsequent
## pages, there are 6 links each. (1000 - 20) / 6
for(i in 1:165){
        webpage <- read_html(paste0("http://www.thefashionspot.com/style-trends/page/",i,"/"))
        urls <- webpage %>% html_nodes(".post-title") %>% html_attr("href")
        temp_list <- list(url<-urls)
        article_url <- c(article_url, temp_list)
}
article_url <- unlist(article_url)      ##converts from list to character
```

Vectors were created to store every article title, date and content scraped

```
article_title <- vector()
article_date <- vector()
article_content <- vector()
```

A simple for loop loops through the 1004 articles URLs that were found. Extracting and storing the required information that vectors were created for.

```
## this loops through all 1004 article urls and then extracts and stores the required information
for (i in seq_along(article_url)){
        webpage <- read_html(article_url[i])

        title <- webpage %>% html_nodes(".article-title") %>% html_text(trim <- TRUE)
        article_title <- c(article_title, title[1])

        date <- webpage %>% html_nodes(".article-date") %>% html_text(trim <- TRUE)
        article_date <- c(article_date, date[1])

        content <- webpage %>% html_nodes(".article-content") %>% html_text(trim <- TRUE)
        article_content <- c(article_content, content[1])
}
```

Once the data is collected, everything is stored in a data frame.

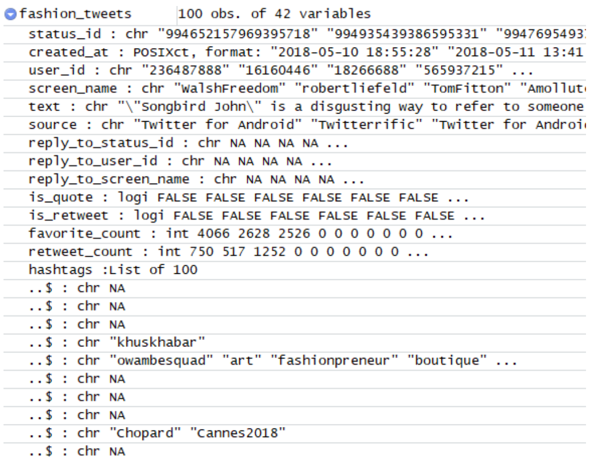
```
## compiling all information into a data frame
articles_df <- data.table::data.table(title = article_title,
                                      date = article_date,
                                      content = article_content)
```

This process can be repeated on multiple fashion blogs as long as they are run on WordPress, which ensures they follow the same webpage structure for articles. Websites developed different will have different structures and would require the scraping script to be modified specifically for them.

2.5.3 Data Pre-Processing

Since the primary goal of Trendicator is to perform text analysis in order to detect fashion trends, only text fields and date fields were selected and cleaned.

Scraped tweets contained 42 variables, many of which were of no interest to this Project.



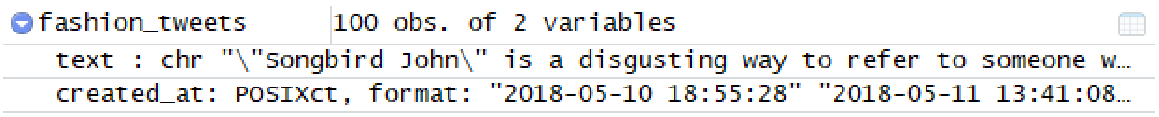
```
fashion_tweets      100 obs. of 42 variables
   status_id : chr "994652157969395718" "994935439386595331" "9947695493
   created_at : POSIXct, format: "2018-05-10 18:55:28" "2018-05-11 13:41
   user_id : chr "236487888" "16160446" "18266688" "565937215" ...
   screen_name : chr "WalshFreedom" "robertliefeld" "TomFitton" "Amollut
   text : chr "\"Songbird John\" is a disgusting way to refer to someone
   source : chr "Twitter for Android" "Twitterrific" "Twitter for Androi
   reply_to_status_id : chr NA NA NA NA ...
   reply_to_user_id : chr NA NA NA NA ...
   reply_to_screen_name : chr NA NA NA NA ...
   is_quote : logi FALSE FALSE FALSE FALSE FALSE FALSE ...
   is_retweet : logi FALSE FALSE FALSE FALSE FALSE FALSE ...
   favorite_count : int 4066 2628 2526 0 0 0 0 0 0 0 ...
   retweet_count : int 750 517 1252 0 0 0 0 0 0 0 ...
   hashtags :List of 100
   ..$ : chr NA
   ..$ : chr NA
   ..$ : chr NA
   ..$ : chr "khuskhabar"
   ..$ : chr "owambesquad" "art" "fashionpreneur" "boutique" ...
   ..$ : chr NA
   ..$ : chr NA
   ..$ : chr NA
   ..$ : chr "Chopard" "Cannes2018"
   ..$ : chr NA
```

First, the tweets had every un-wanted variables removed, using just the 'text' and 'created_at' variables.

Secondly, commas were removed from the tweets which will help with word stemming later in the project. Null values were also removed, if any.

```r
# Cleaning the Data -----------------------------------------------------------------

# Use only the 'text' & 'created_at' columns
fashion_tweets <- data.frame(text=fashion_tweets$text,
                             created_at=fashion_tweets$created_at,
                             stringsAsFactors = F)
fashion_tweets$text <- gsub(",",'',fashion_tweets$text) # Take out commas
na.omit(fashion_tweets) #remove any rows with NULL
```

The result,

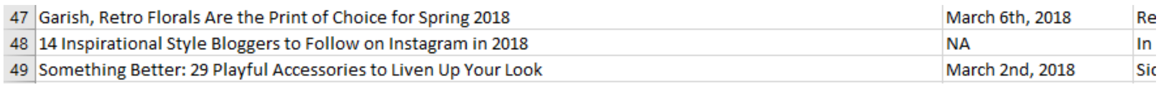


```
fashion_tweets      100 obs. of 2 variables
   text : chr "\"Songbird John\" is a disgusting way to refer to someone w…
   created_at: POSIXct, format: "2018-05-10 18:55:28" "2018-05-11 13:41:08…
```

Which was then saved in .csv format. Scraped tweets were first saved to a 'fashion_tweets.csv' file which overwrites any information previously in the file. This file is used to upload any new data into the database. The second file, 'fashion_tweets_archive' inserts new data into the file without overwriting any existing data. This is used as a precautionary measure.

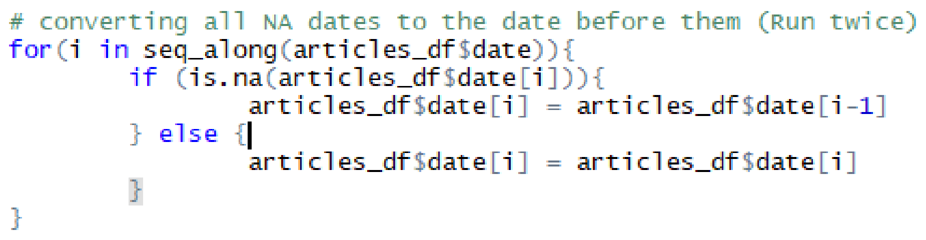
```r
# Export to file -----------------------------------------------------------------

write.csv(fashion_tweets, "fashion_tweets.csv") ## save to csv file
write.table(fashion_tweets, file= "fashion_tweets_archive.csv", row.names=FALSE, append=TRUE) #adds more data to existing .csv
```

Not much cleaning on the data extracted from website articles was needed. Some articles had dates missing.



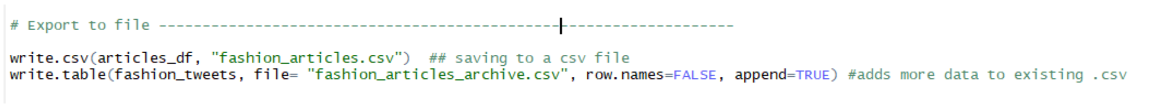
47	Garish, Retro Florals Are the Print of Choice for Spring 2018	March 6th, 2018	Re
48	14 Inspirational Style Bloggers to Follow on Instagram in 2018	NA	In
49	Something Better: 29 Playful Accessories to Liven Up Your Look	March 2nd, 2018	Sic

Due to these articles being collected in chronological order, missing dates were filled in with the date of an article before them.



```
# converting all NA dates to the date before them (Run twice)
for(i in seq_along(articles_df$date)){
        if (is.na(articles_df$date[i])){
                articles_df$date[i] = articles_df$date[i-1]
        } else {
                articles_df$date[i] = articles_df$date[i]
        }
}
```

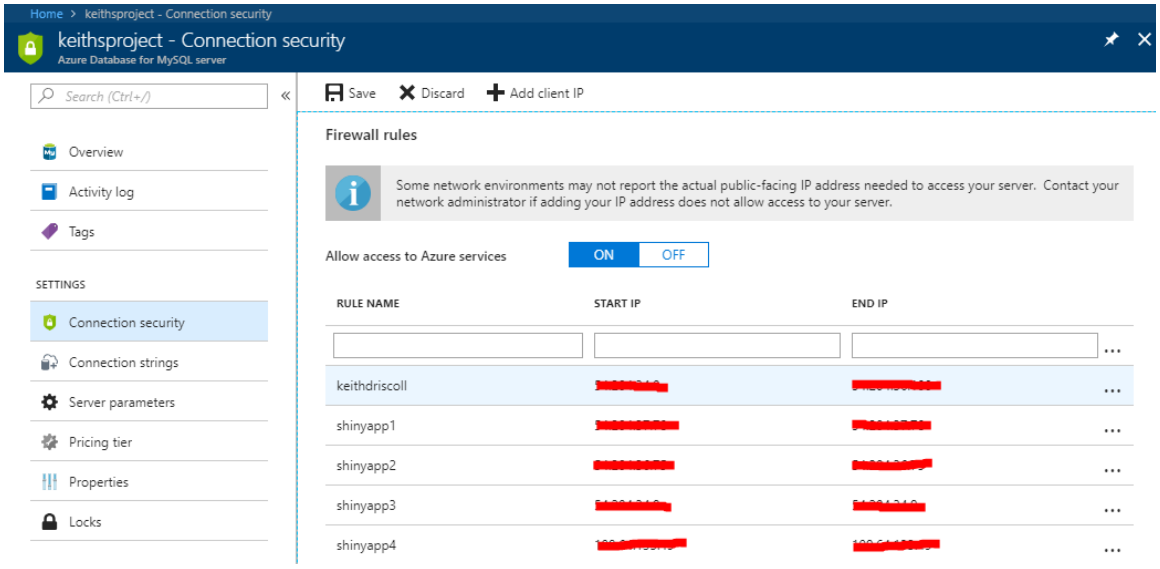
The data was then saved in .csv format. Scraped articles were first saved to a ‘fashion_ articles.csv’ file which overwrites any information previously in the file. This file is used to upload any new data into the database. The second file, ‘fashion_ articles _archive’ inserts new data into the file without overwriting any existing data. This is used as a precautionary measure.



```
# Export to file ---------------------------------------------------------------

write.csv(articles_df, "fashion_articles.csv")  ## saving to a csv file
write.table(fashion_tweets, file= "fashion_articles_archive.csv", row.names=FALSE, append=TRUE) #adds more data to existing .csv
```

2.5.4 Database Connection

A connection was then established between RStudio and the data hosted on the Microsoft Azure cloud. This was first made possible by adding the IP addresses of every external connection trying to connect to the database. This includes the project’s author and the RShiny Application (IP Addresses were censored from this screenshot for security reasons).


Home > keithsproject - Connection security

keithsproject - Connection security

Azure Database for MySQL server

Firewall rules

Some network environments may not report the actual public-facing IP address needed to access your server. Contact your network administrator if adding your IP address does not allow access to your server.

Allow access to Azure services [✓ ON] OFF

RULE NAME	START IP	END IP
keithdriscoll	[redacted]	[redacted]
shinyapp1	[redacted]	[redacted]
shinyapp2	[redacted]	[redacted]
shinyapp3	[redacted]	[redacted]
shinyapp4	[redacted]	[redacted]

Database login credentials are stored and later called to attempt a database connection.

```
if (!require('RMySQL')) install.packages('RMySQL')
library(RMySQL)

## set credentials for mysql
user <- 'keithdriscoll@keithsproject'
password <- '[redacted]
dbname <- 'SoftwareProject'
host <- 'keithsproject.mysql.database.azure.com'
port <-  3306
```

If a user attempts to connect to the database and their IP address is not whitelisted in the firewall, a connection will not be established.

```
> ## create connection to mysql
> MySQLConn <- dbConnect(MySQL(), user=user, password=password,dbname=dbname, host=host ,port=port)
Error in .local(drv, ...) :
  Failed to connect to database: Error: Client with IP address '[redacted] is not allowed to connect to this MySQL server.
```

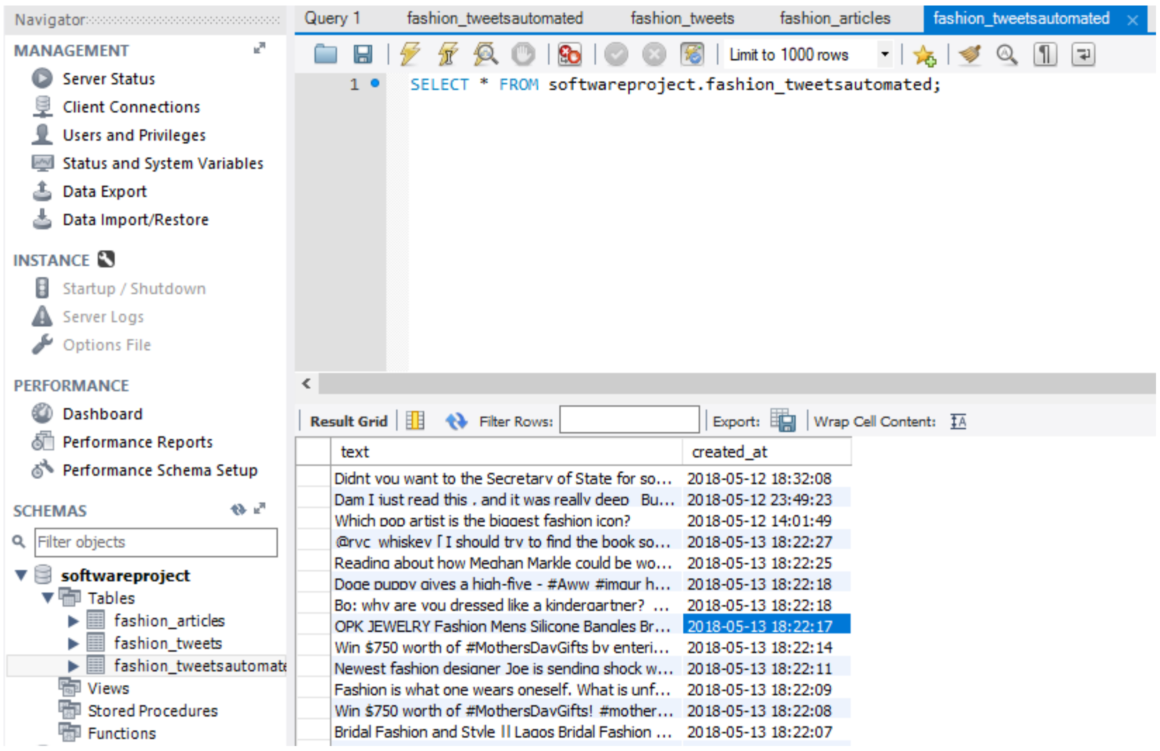
When a successful connection is made, the console does not output a response. To test for a successful connection, a database query was made in R. Below you can see a query to see data from the Fashion_Tweets table within the database named Software Project.

```
> ## create connection to mysql
> MySQLConn <- dbConnect(MySQL(), user=user, password=password,dbname=dbname, host=host ,port=port)
> ## get data
> query <- paste("select * from SoftwareProject.Fashion_Tweets")
> data <- dbGetQuery(MySQLConn, query) ## avg by date,company,building,room
> head(data)


1
    @SpokenPoet50 I so agree @SpokenPoet50, but they thought of it as fashion; when you and I see nothinc
2
fashion public corruption slush fund for the president. AT&amp;T, Novartis, the freaking Russians...\n\nw
3
><U+0093><U+00B7> met gala 2018: heavenly bodies: fashion and the catholic imagination costume institute
4
                      @sqiderman_ me: its just a video game, not real fashion\n\nalso me, looking at that
5
                                                            Cheap Quotes Winona Country Graphic T-Shirt
```

Another SQL query can be made through R to import a csv file into the database.

```
> query <- paste("INSERT IGNORE INTO SoftwareProject.Fashion_TweetsAutomated(text ,created_at) VALUES", inserts)
```

Another approach to checking if this was successful is through inspecting the database via MySQL Workbench, which has established a successful connection to the Azure database; using the same login credentials used within R Studio.



```
SELECT * FROM softwareproject.fashion_tweetsautomated;
```

text	created_at
Didnt you want to the Secretary of State for so…	2018-05-12 18:32:08
Dam I just read this . and it was really deep Bu…	2018-05-12 23:49:23
Which pop artist is the biggest fashion icon?	2018-05-12 14:01:49
@ryc_whiskey [I should try to find the book so…	2018-05-13 18:22:27
Reading about how Meghan Markle could be wo…	2018-05-13 18:22:25
Doge puppy gives a high-five - #Aww #imgur h…	2018-05-13 18:22:18
Bo: why are you dressed like a kindergartner? …	2018-05-13 18:22:18
OPK JEWELRY Fashion Mens Silicone Bangles Br…	2018-05-13 18:22:17
Win $750 worth of #MothersDayGifts by enteri…	2018-05-13 18:22:14
Newest fashion designer Joe is sending shock w…	2018-05-13 18:22:11
Fashion is what one wears oneself. What is unf…	2018-05-13 18:22:09
Win $750 worth of #MothersDayGifts! #mother…	2018-05-13 18:22:08
Bridal Fashion and Style ll Lagos Bridal Fashion …	2018-05-13 18:22:07

2.5.5 Data Transformation

R Shiny establishes a MySQL connection and retrieves the data stored in the database and stores the data into data frames in R. Trendicator's database consists of two tables. One table containing all this project's tweets and another which contains all of this project's articles scraped from online sources.

```r
## create connection to mysql
MySQLConn <- dbConnect(MySQL(), user=user, password=password,dbname=dbname, host=host ,port=port)

## get data
tweets <- dbGetQuery(MySQLConn, "select * from SoftwareProject.Fashion_Tweets")
articles <- dbGetQuery(MySQLConn, "select * from SoftwareProject.Fashion_Articles")
```

One of the most effective ways of transforming data for text analysis is using the Term Document Matrix (TDF). Before applying this transformation technique further cleaning is performed by removing numbers and special characters and more, outlined below.

```r
## function to text
clean_text <- function(data){

    data$text <- gsub("@[a-z,A-Z]*","",data$text) # remove user mentions
    data$text <- gsub("RT @[a-z,A-Z]*","",data$text) # remove user mentions
    data$text <- gsub("#[a-z,A-Z]*","",data$text) # remove hashtags
    data$text <- gsub("(f|ht)(tp)(s?)(://)(\\S*)", "", data$text) # remove URLs (http, https, ftp)
    data$text <- gsub("[^0-9A-Za-z///' ]", "", data$text) # remove all non english / non numeric
    data$text <- gsub("\\s+", " ", data$text) # double spaces
    data$text <- gsub("\\d", " ", data$text) # remove digits
    data$text <- gsub("(f|F)(ashion)", " ", data$text) # remove the word fashion
    data$text <- tolower(data$text) # convert to lower case
    return(data)

}
```

Text cleaning is stored into an R function so it can easily be re-used on other variables in one line of code.

```r
## clean data
tweets <- clean_text(tweets)
articles <- clean_text(articles)
stop_words$word <- gsub("'",'',stop_words$word)
```

Words are tokenized using a tidytext package and stop words being filtered out.

```r
## seperate words in texts into individual words (tokens) and count freuqency of each word
## stop words are filtered
topics <-
    data %>%
    tidytext::unnest_tokens(word,text,drop=F) %>%
    data.frame() %>%
    filter(!word %in% stop_words$word,nchar(word)>2) %>%
    count(text, word, sort = TRUE) %>%
    ungroup()
```

2.5.6 Data Mining

All the data we want to examine is stored within the a single ‘text’ column. The next task is to transform our text column into a matrix of terms.

A Term Document Matrix is defined by (Weber, 2013) as “…a two-dimensional array, with rows and columns. In the TDM, rows represent documents, columns represent terms (in the collection vocabulary). Cell values are term frequency counts, or, more generally, “score” attached to a term for a document”.

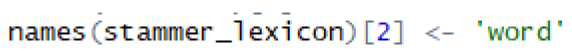
By applying a K-means clustering technique, clustering words that are closely related to each other, this was fitted into a Linear discriminant analysis(LDA) model. An LDA model allows sets of observations to be explained by unobserved groups to produce Trendicator’s Topic Modelling.

```
    ## create document term matrix
    topics_dtm <- topics %>% cast_dtm(text, word, n)

    ## fit LDA model with K clusters
    topics_lda <- topicmodels::LDA(topics_dtm, k = K, control = list(seed = 1234))

    ## convert results into data.frame
    topics_lda <- tidy(topics_lda, matrix = "beta")
```

Ensuring words present in a topic only appear within one cluster, words are aggregated. Arranging most frequent words by the top 15 for each topic.

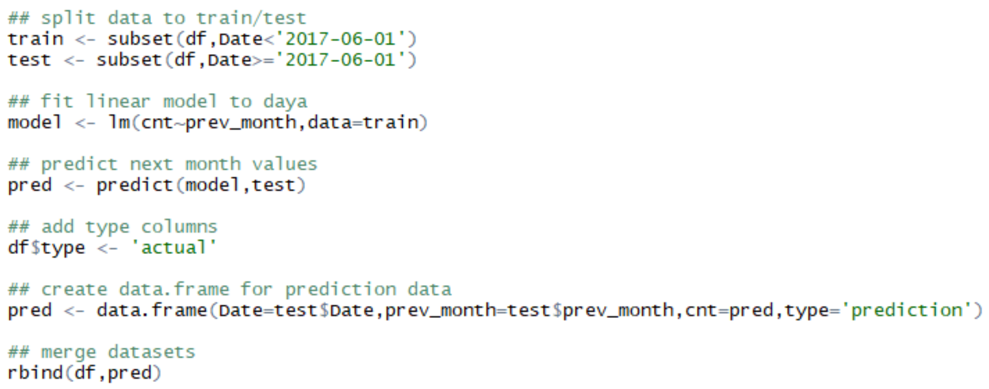
```
    ## aggregate words in topics by max beta score so each terms will appear in only one cluster
    ## and return the top 15 terms for each topic
    top_terms <-
        topics_lda %>%
        group_by(term) %>%
        mutate(maxx=max(beta)) %>%
        filter(maxx==beta) %>%
        ungroup() %>%
        group_by(topic) %>%
        top_n(15, beta) %>%
        ungroup() %>%
        arrange(topic, -beta)

    return(top_terms)

}
```

Instead of generating a very large file containing hundreds of thousands of words stemmed, which also takes a lot of time to generate. An already generated word stem file from GitHub was downloaded via: https://github.com/michmech/lemmatization-lists/blob/master/lemmatization-en.txt and then converted to .csv. The second column header was renamed for the term topic model.



```
names(stammer_lexicon)[2] <- 'word'
```

Later used to produce visual results of Terms Trends, a linear model was implemented to attempt to predict the future projection of trends. The dataset used for the Terms Trends model was split into two sets. A training set and a testing set to the linear regression model for cross validation, then producing it’s predicted results.



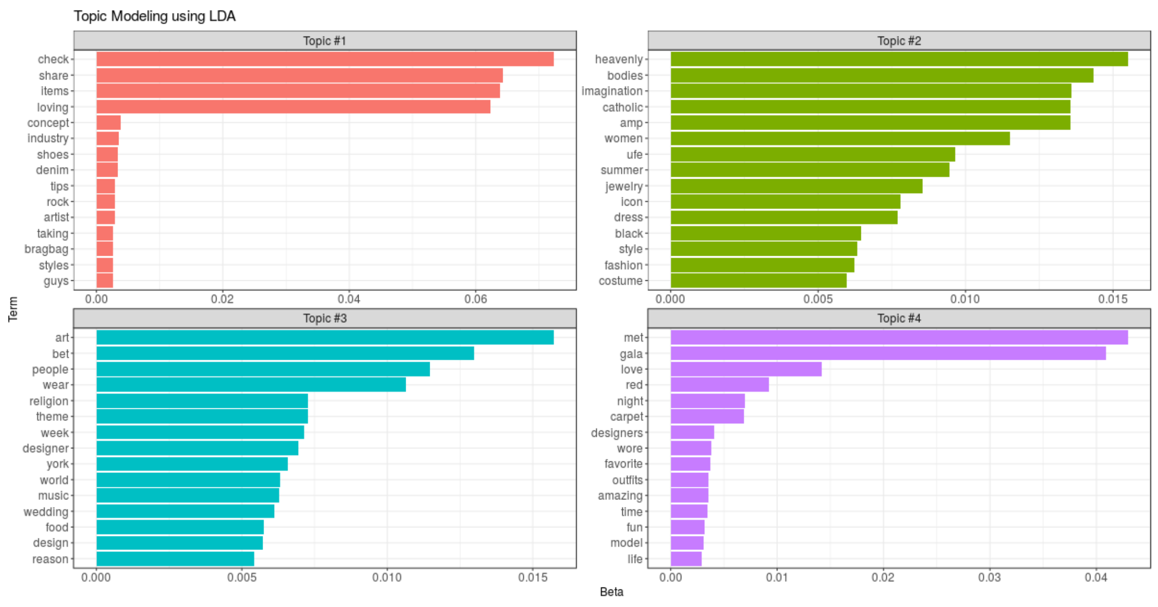
```
## split data to train/test
train <- subset(df,Date<'2017-06-01')
test <- subset(df,Date>='2017-06-01')

## fit linear model to daya
model <- lm(cnt~prev_month,data=train)

## predict next month values
pred <- predict(model,test)

## add type columns
df$type <- 'actual'

## create data.frame for prediction data
pred <- data.frame(Date=test$Date,prev_month=test$prev_month,cnt=pred,type='prediction')

## merge datasets
rbind(df,pred)
```

2.5.7 Data Visualisation

Linear discriminant analysis (LDA) explicitly attempts to model the difference between the classes of data. The topic meaning extracted by interpreting the top N probability words for a given topic. LDA will not output the meaning of topics, rather it will organize words by topic to be interpreted by the user. In simple terms, LDA is the act of distributing things into groups, classes or categories of the same type.

```
####################################################################################
####################################################################################
## topic modeling for tweets


    ## plot
    output$topics_plot <- renderPlot({

        progress <- Progress$new(session, min=1, max=15)
        on.exit(progress$close())

        progress$set(message = 'Updating data\n',
                     detail = 'This can take up to 2 minutes...')

        topics <- topic_modeling(tweets,K=as.numeric(input$topic))
        topics <- topics %>% mutate(term = reorder(term, beta),topic=paste0('Topic #',topic))

        ggplot(topics,aes(x=term, y=beta, fill = factor(topic))) +
        geom_col(show.legend = FALSE) +
        facet_wrap(~ topic, scales = "free") +
        coord_flip()+
        labs(x = 'Term', y = "Beta",title='Topic Modeling using LDA',caption='') +
        theme_bw() +
        theme(legend.position="top"
              ,legend.text=element_text(size=12)
              ,legend.title = element_blank()
              ,strip.text.x = element_text(size=12)
              ,strip.text.y = element_text(size=12)
              ,axis.text.x=element_text(size=12)
              ,axis.text.y=element_text(size=12)
              ,axis.title.x=element_text(size=12)
              ,axis.title.y=element_text(size=12)
              ,plot.title = element_text(size=15))

}, height=900*0.7,width=1500*0.8)


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

Output:

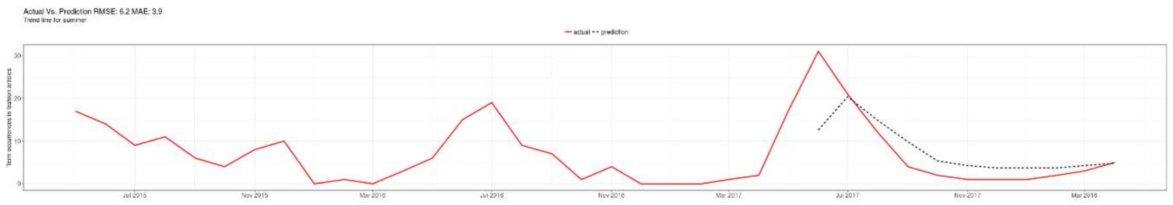


A Term Frequency model counts the frequency of word occurrences over time and plots results onto a ggplot in R. Applying a Linear regression model to compare the future predictions with actual predictions of term frequencies.

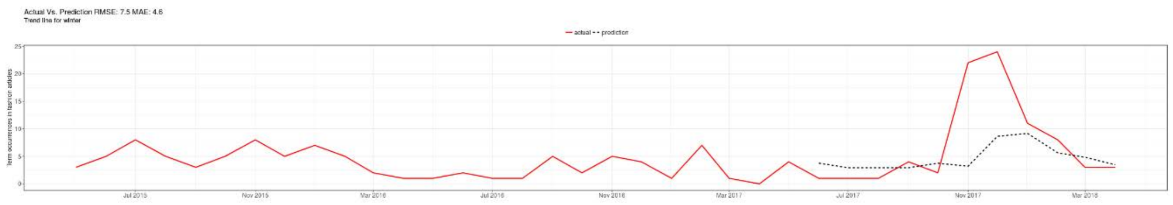
```
## plot
output$trends_plot <- renderPlot({

    # terms <- data.frame(topics=c('seasons','clothes','accessories','footwear','designer','statement','detail','material','color')
    #                     ,terms=c('spring|summer|winter|fall','shirt|trouser|pant|swimsuit|blouse|blazer|dress|skirt|coat|shirt|footwear','earr
    #                     ,stringsAsFactors = F)
    #
    # terms <- subset(terms,topics==input$top_terms)
    # terms$terms <- gsub('\\|',', ',terms$terms)

    ## calc prediction error RMSE (root mean square error),MAE (mean absolute error)
    actual <- trends() %>% filter(type=='actual',Date>='2017-06-01') %>% select(cnt)
    prediction <- trends() %>% filter(type=='prediction',Date>='2017-06-01') %>% select(cnt)
    RMSE <- round(sqrt(mean((actual[[1]]-prediction[[1]])^2)),1)
    MAE <- round(mean(abs(actual[[1]]-prediction[[1]])),1)


    ggplot(trends(),aes(x=Date,y=cnt,color=type,group=type,linetype=type)) +
    geom_line(size=1) +
    scale_color_manual(values = c('red','black'))+
    scale_x_date(date_labels = "%b %Y",date_breaks = "4 month")+
    labs(x = ''
         ,y = "Term occurrences in fashion articles"
         ,title=paste0('Actual Vs. Prediction',' RMSE: ',RMSE,' MAE: ',MAE)
         ,subtitle=paste('Trend line for', input$top_terms)) +
        # ,subtitle=paste(input$top_terms,'include the terms: ',terms$terms)) +
    theme_bw() +
    theme(legend.position="top"
          ,legend.text=element_text(size=12)
          ,legend.title = element_blank()
          ,strip.text.x = element_text(size=12)
          ,strip.text.y = element_text(size=12)
          ,axis.text.x=element_text(size=12)
          ,axis.text.y=element_text(size=12)
          ,axis.title.x=element_text(size=12)
          ,axis.title.y=element_text(size=12)
          ,plot.title = element_text(size=15)
          ,plot.subtitle = element_text(size=13)
          )
})
```

Output of Summer and Winter results:

Summer graph does not spike in the winter months. There are spikes in July for the past three years on the graph. The Linear regression trend somewhat closely resembles the actual trend.



Winter graph does not spike during summer months. There is a spike in the most recent winter months but not in previous years, possibly due to the limited data available. The linear regression trend does not closely resemble the actual trend, again possible due to limited data available.

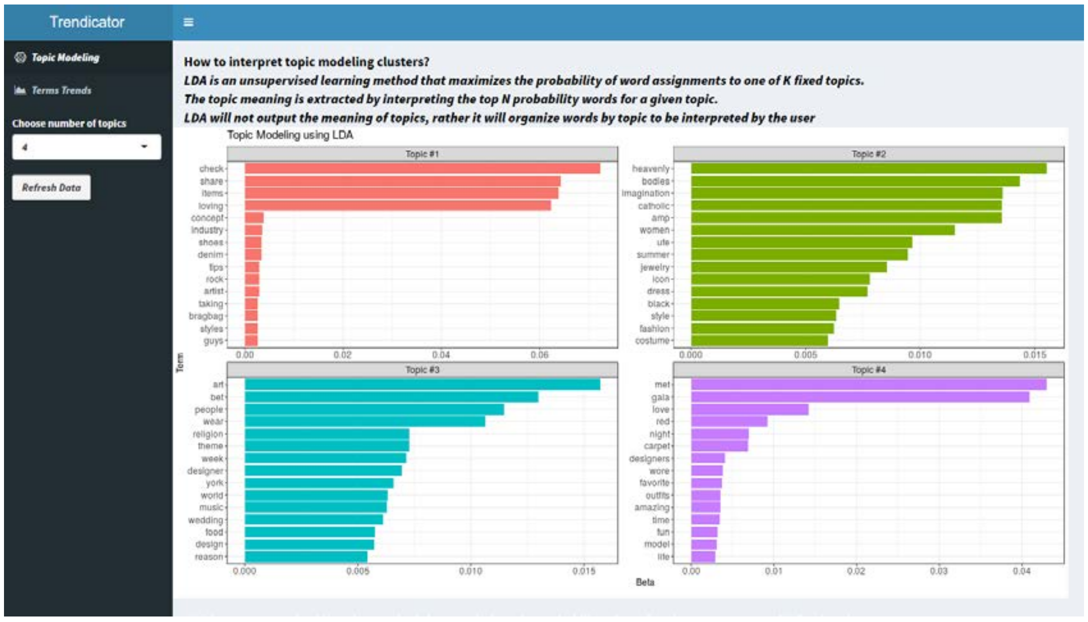


2.6 *Graphical User Interface (GUI) Layout*

Trendicator

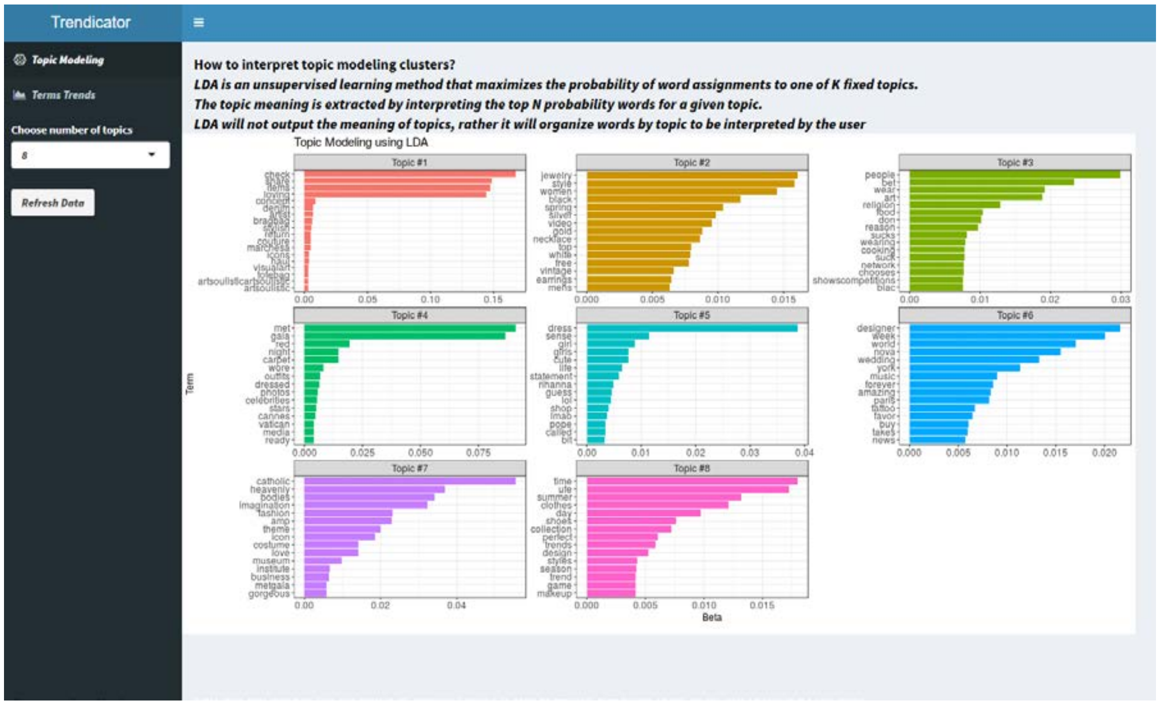
This is the GUI interface of the main system. This is the interactional interface of the web platform in which the user can update the visual representation of data, live, based on their selected criteria available on-screen.

Visual output of the data when grouped into four topics. These four topics are to be interpreted by the user. Topics are more difficult to interpret with less topic groups.



How to interpret topic modeling clusters?
LDA is an unsupervised learning method that maximizes the probability of word assignments to one of K fixed topics.
The topic meaning is extracted by interpreting the top N probability words for a given topic.
LDA will not output the meaning of topics, rather it will organize words by topic to be interpreted by the user

Visual output of the data when grouped into eight topics. The more topics the text is divided into, the easier it may become to interpret. These eight topics are to be interpreted by the user. Topic #4 for example could represent tweets about the recent Met GALA event which was trending on social media.


Trendicator
Topic Modeling
Terms Trends
Choose number of topics
8
Refresh Data

How to interpret topic modeling clusters?

LDA is an unsupervised learning method that maximizes the probability of word assignments to one of K fixed topics.

The topic meaning is extracted by interpreting the top N probability words for a given topic.

LDA will not output the meaning of topics, rather it will organize words by topic to be interpreted by the user

Topic Modeling using LDA
Topic #1
check share items loving concept denim artist bragbag stylish return couture marchesa icons haul visualart totebag artsoulisticartsoulistic artsoulistic
0.00 0.05 0.10 0.15
Topic #2
jewelry style women black spring silver video gold necklace top white free vintage earrings mens
0.000 0.005 0.010 0.015
Topic #3
people bet wear art religion food don reason sucks wearing cooking suck network chooses showscompetitions blac
0.00 0.01 0.02 0.03
Topic #4
met gala red night carpet wore outfits dressed photos celebrities stars cannes vatican media ready
0.000 0.025 0.050 0.075
Topic #5
dress sense girl girls cute life statement rihanna guess lol shop lmao pope called bit
0.00 0.01 0.02 0.03 0.04
Topic #6
designer week world nova wedding york music forever amazing paris tattoo favor buy takes news
0.000 0.005 0.010 0.015 0.020
Topic #7
catholic heavenly bodies imagination fashion amp theme icon costume love museum institute business metgala gorgeous
0.00 0.02 0.04
Topic #8
time ufe summer clothes day shoes collection perfect trends design styles season trend game makeup
0.000 0.005 0.010 0.015
Term
Beta

Will add screenshots once application is live and operational via Tableau.

2.7 Testing

Test ID	**TR_1**
Test Purpose	Engage with Twitter API
Test Environment	Client Hardware: Dell XPS 13 running windows 10 and R Studio latest edition. Test was to store 16000 tweets into .csv file.
Test Steps	Ensuring the RStudio working directory is appropriately set and all required R packages are installed.
Expected Result	16000 tweets to be successfully scraped from Twitter. Then saved into a .csv file
Actual Result	The .csv file created contained 15896 records.
Suggested Action	The dataset was less than expected due to Twitter API limitations.
Resolution	Run the script again with a lower n value of tweets to store.

Test ID	**TR_2**
Test Purpose	Successfully scrape a webpage
Test Environment	Client Hardware: Dell XPS 13 running windows 10 and R Studio latest edition. Test was to store 16000 tweets into .csv file.
Test Steps	Ensuring the RStudio working directory is appropriately set and all required R packages are installed.
Expected Result	1004 articles to be successfully scraped from TheFashionSpot.com. Then saved into a .csv file
Actual Result	The .csv file created contained 1004 records.
Suggested Action	None
Resolution	None, the script run properly

Test ID	**TR_3**
Test Purpose	Reading a .csv file into RStudio.
Test Environment	Client Hardware: Dell XPS 13 running windows 10 and R Studio latest edition. Test was run on fashion_tweets.csv containing over 18000 records.
Test Steps	Ensuring the RStudio working directory is appropriately set, all required R packages are installed and that the file being read is within the working directory.
Expected Result	Fashion_tweets.csv to successful store all its records within a data frame in RStudio.
Actual Result	The .csv file was successful imported into RStudio within a data frame containing every record.
Suggested Action	None.
Resolution	None

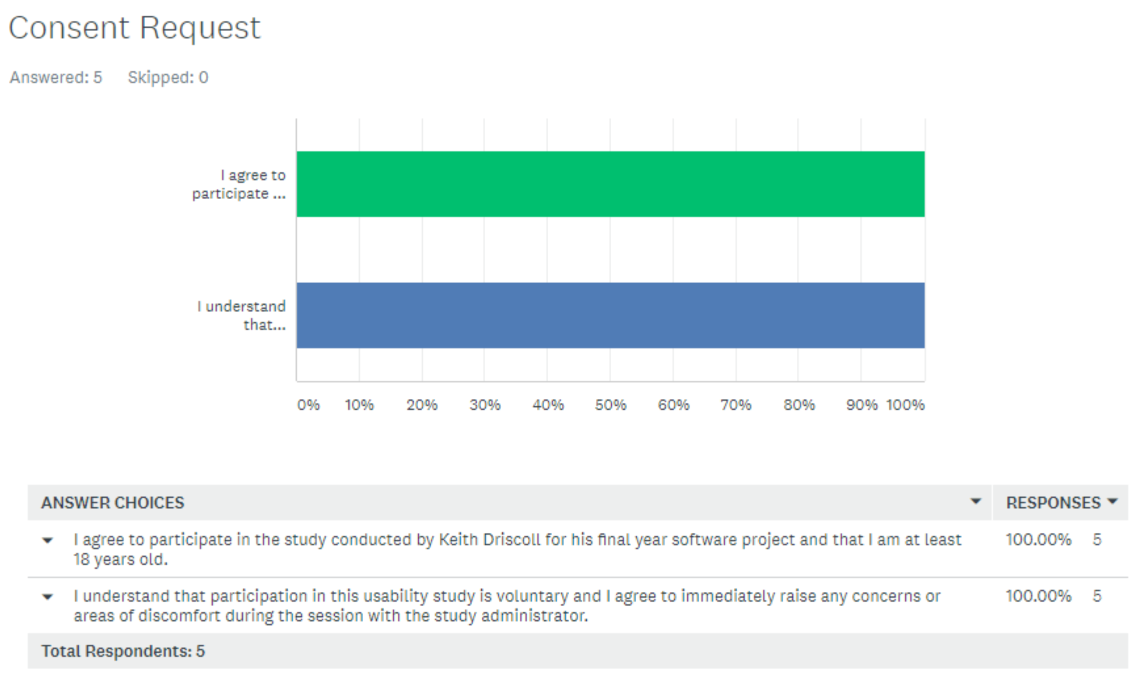
Test ID	**TR_4**
Test Purpose	Importing records from a .csv file into database.
Test Environment	Client Hardware: Dell XPS 13 running windows 10 and R Studio latest edition. Test was run on fashion_tweets.csv containing over 18000 records.
Test Steps	Ensuring the RStudio working directory is appropriately set, all required R packages are installed and that the file being read is within the working directory. Ensure that a connection to the database is established via RStudio.
Expected Result	Fashion_tweets.csv to successful store all its records within an external database.
Actual Result	The data was unsuccessfully imported due to data formatting errors.
Suggested Action	Clean the data before importing records into the database, then try again.
Resolution	Records were successfully imported into the database from a .csv via RStudio.

Test ID	**TR_5**
Test Purpose	Ensuring RShiny is deployed on the RShiny web application
Test Environment	Client Hardware: Dell XPS 13 running windows 10 and R Studio latest edition. Test was run in RStudio.
Test Steps	Ensuring the RStudio working directory is appropriately set, all required R packages are installed and that the file being read is within the working directory. Ensure that a connection to the database is established via RStudio. Ensuring that RShiny account details are correctly inserted.
Expected Result	RShiny Application will open a web application in an internet browser with a working application.
Actual Result	RShiny Application successful opened a new tab in web browser with a working RShiny Web Application.
Suggested Action	None
Resolution	None

Test ID	**TR_6**
Test Purpose	Ensure Data visualization graphs update upon selection of filter criteria.
Test Environment	Client Hardware: Dell XPS 13 running windows 10 and R Studio latest edition. Test was run on www.trendicator.info RShiny web application.
Test Steps	Ensure that the RShiny web application is running and the data has been loaded.
Expected Result	Topic Modelling Graph updates number of graphs for topic based on filter criteria. 4 topics were selected, 4 topic graphs were shown. Term Trends graph updates the plots of a graph based on which filter criteria is selected. Summer graph is selected, summer plots are output. Graph updates when Winter is selected.
Actual Result	Graphs updated based on filter criteria selected by user.
Suggested Action	None
Resolution	None

2.8 *Customer testing*

A survey was performed on users to gain an insight on their thoughts of my application. The survey questions were as follows:

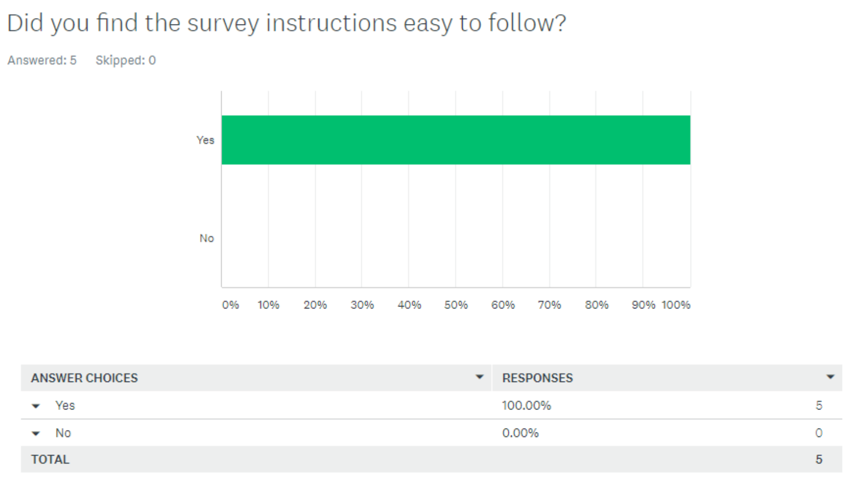


Consent Request

Answered: 5 Skipped: 0

ANSWER CHOICES	RESPONSES	
I agree to participate in the study conducted by Keith Driscoll for his final year software project and that I am at least 18 years old.	100.00%	5
I understand that participation in this usability study is voluntary and I agree to immediately raise any concerns or areas of discomfort during the session with the study administrator.	100.00%	5
Total Respondents: 5		

1. Did you find the instructions easy to follow? (Yes/No)

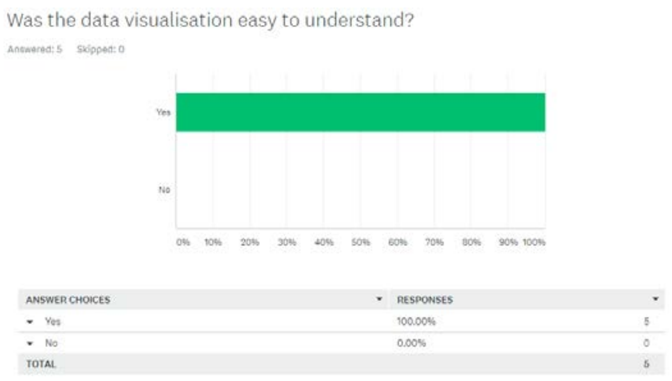


Did you find the survey instructions easy to follow?

Answered: 5 Skipped: 0

ANSWER CHOICES	RESPONSES	
Yes	100.00%	5
No	0.00%	0
TOTAL		5

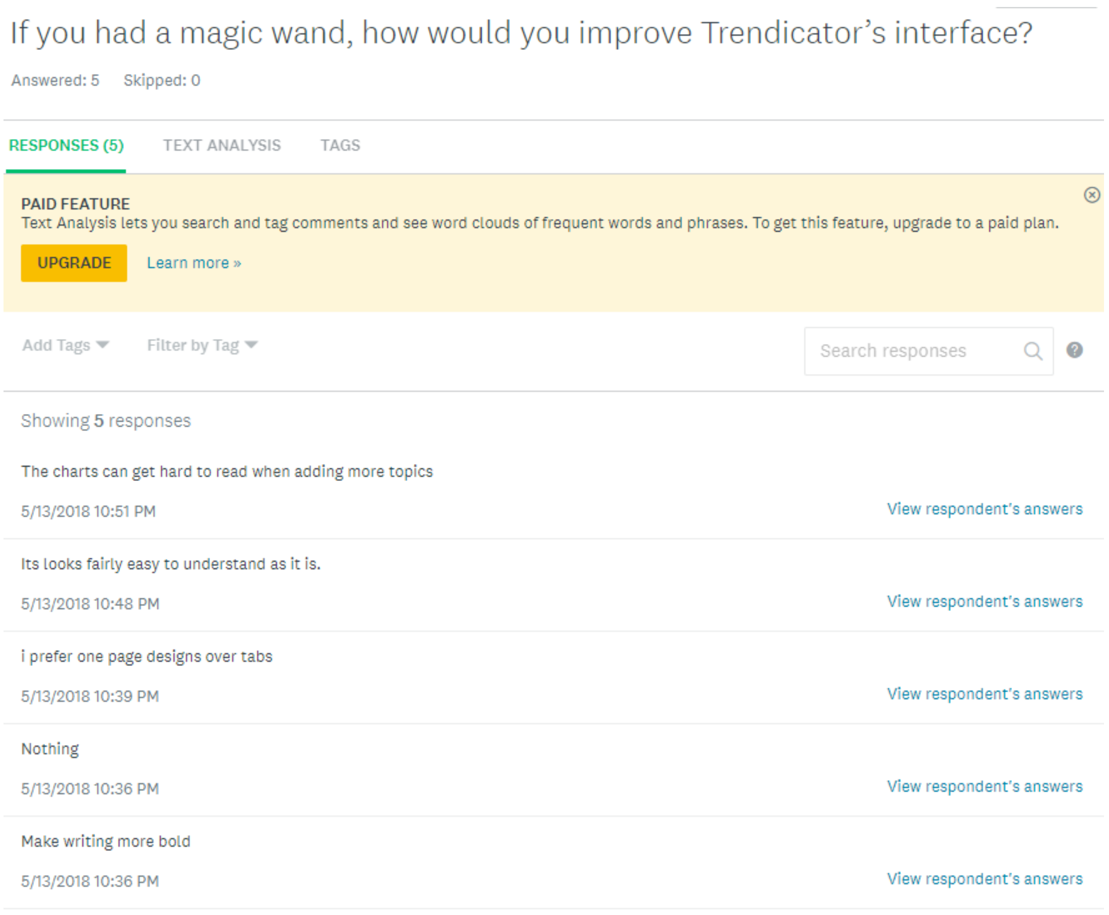
2. If answered No, please explain why.
3. Was the data visualisation easy to understand? (Yes/No)


Was the data visualisation easy to understand?

Answered: 5 Skipped: 0

ANSWER CHOICES	RESPONSES	
Yes	100.00%	5
No	0.00%	0
TOTAL		5

4. If answered No, please explain why.
5. If you had a magic wand, how would you improve Trendicator's interface?


If you had a magic wand, how would you improve Trendicator's interface?

Answered: 5 Skipped: 0

PAID FEATURE
Text Analysis lets you search and tag comments and see word clouds of frequent words and phrases. To get this feature, upgrade to a paid plan.

Showing 5 responses

The charts can get hard to read when adding more topics
5/13/2018 10:51 PM View respondent's answers

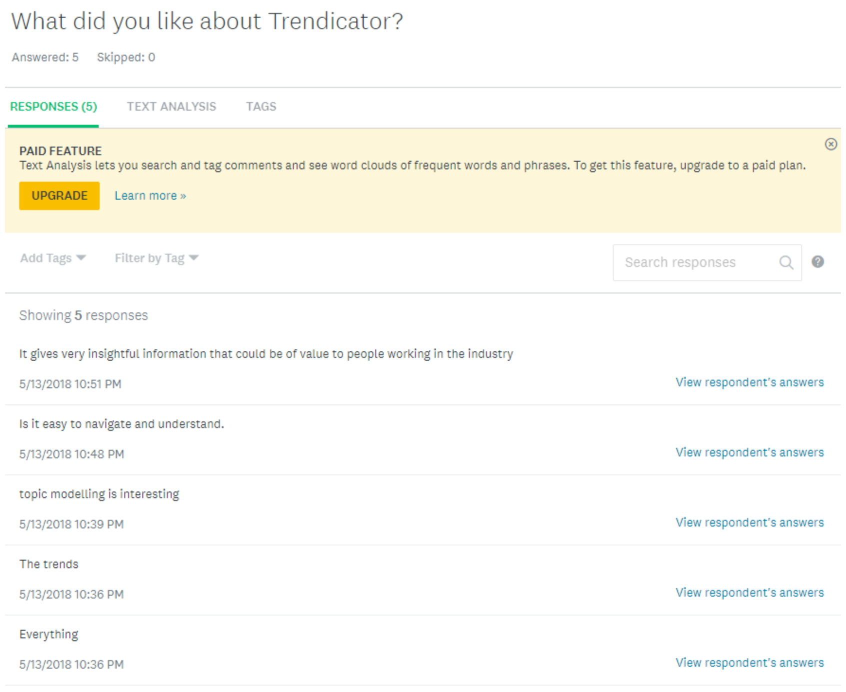
Its looks fairly easy to understand as it is.
5/13/2018 10:48 PM View respondent's answers

i prefer one page designs over tabs
5/13/2018 10:39 PM View respondent's answers

Nothing
5/13/2018 10:36 PM View respondent's answers

Make writing more bold
5/13/2018 10:36 PM View respondent's answers

6. What did you like about Trendicator?



What did you like about Trendicator?

Answered: 5 Skipped: 0

RESPONSES (5) TEXT ANALYSIS TAGS

PAID FEATURE
Text Analysis lets you search and tag comments and see word clouds of frequent words and phrases. To get this feature, upgrade to a paid plan.

UPGRADE Learn more »

Add Tags Filter by Tag Search responses

Showing 5 responses

It gives very insightful information that could be of value to people working in the industry
5/13/2018 10:51 PM View respondent's answers

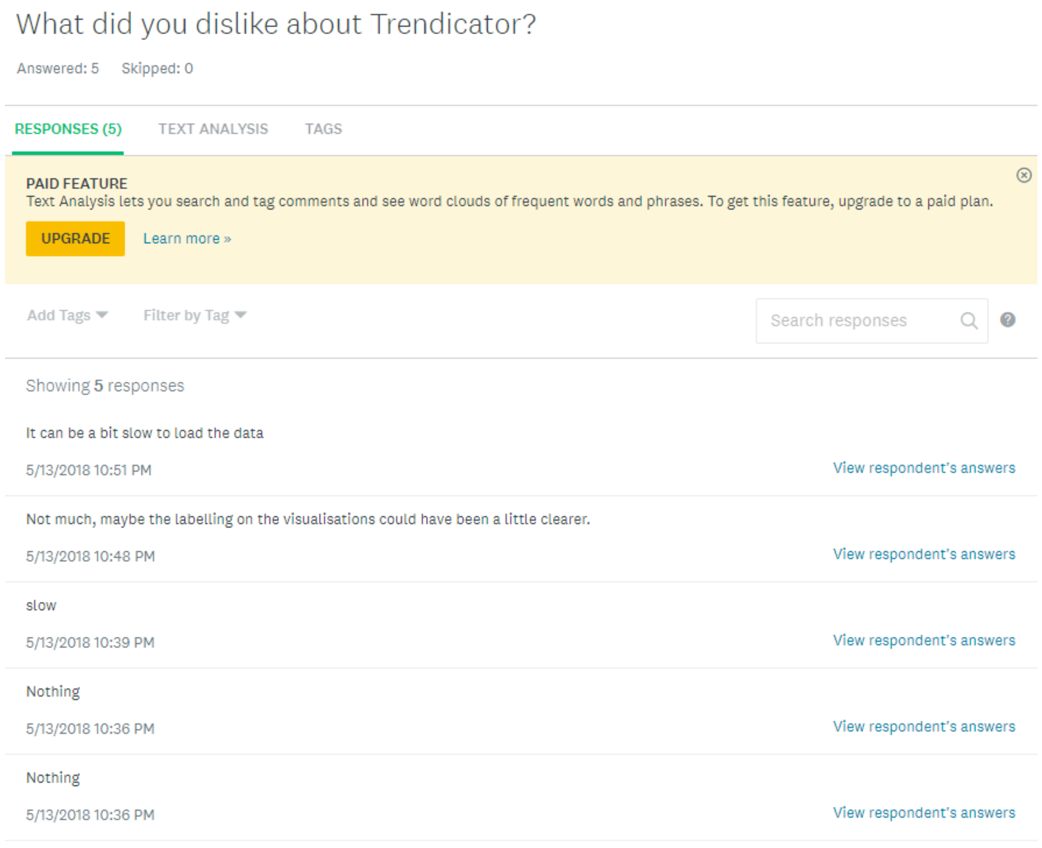
Is it easy to navigate and understand.
5/13/2018 10:48 PM View respondent's answers

topic modelling is interesting
5/13/2018 10:39 PM View respondent's answers

The trends
5/13/2018 10:36 PM View respondent's answers

Everything
5/13/2018 10:36 PM View respondent's answers

7. What did you dislike about Trendicator?



What did you dislike about Trendicator?

Answered: 5 Skipped: 0

RESPONSES (5) TEXT ANALYSIS TAGS

PAID FEATURE
Text Analysis lets you search and tag comments and see word clouds of frequent words and phrases. To get this feature, upgrade to a paid plan.

UPGRADE Learn more »

Add Tags Filter by Tag Search responses

Showing 5 responses

It can be a bit slow to load the data
5/13/2018 10:51 PM View respondent's answers

Not much, maybe the labelling on the visualisations could have been a little clearer.
5/13/2018 10:48 PM View respondent's answers

slow
5/13/2018 10:39 PM View respondent's answers

Nothing
5/13/2018 10:36 PM View respondent's answers

Nothing
5/13/2018 10:36 PM View respondent's answers

2.9 Evaluation

The Linear regression model performed on term frequency distribution of article posts was tested using two metrics.

Mean Absolute Error (MAE): Is a measurement of the average magnitude of the errors in a set of predictions, without considering their direction. It’s the average over the test sample of the absolute differences between predictive and actual results where each individual difference has an equal weight.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

Root Mean Squared Error (RMSE): Is a quadratic scoring rule which measures the average magnitude of the error. It’s the square root of the average of squared differences between the model’s prediction and its actual result.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

These metrics were then tested on Trendicator’s regression models using this code:

```
## calc prediction error RMSE (root mean square error),MAE (mean absolute error)
actual <- trends() %>% filter(type=='actual',Date>='2017-06-01') %>% select(cnt)
prediction <- trends() %>% filter(type=='prediction',Date>='2017-06-01') %>% select(cnt)
RMSE <- round(sqrt(mean((actual[[1]]-prediction[[1]])^2)),1)
MAE <- round(mean(abs(actual[[1]]-prediction[[1]])),1)
```

The lower the results, the more accurate the model is. For this model the RMSE scoring is high as the data set in use does not have good predictiors for the model. Focusing on the upper bound, this means that RMSE has a tendency to be increasingly larger than MAE as the test sample size increases.

3 Conclusions

By conducting this project, Trendicator can identify trends of fashion related words which can be used towards identifying a fashion trend. However, Trendicator is not yet capable of detecting a fashion trend or produce sufficient evaluative measures of its detection accuracy. Largely due to the fact of what little amount of data was used, limiting the models and visual outputs.

The advantages of Trendicator allow for visually representing the analysed data which can help in the future towards detecting actual fashion trends.

Trendicator’s current model can be used to help identify not just fashion related trends but, any trend within data fed into its model. It can also be applied to sources of any language but not sources containing multiple languages without code modification.

Trendicator’s cloud-based platform allows the project to be accessed virtually anywhere and securely.

By adjusting how data is fed into Trendicator’s model, this project is capable of outputting data visualisations of real-time data.

Trendicator is currently limited by its RShinyApps deployment which has very limited computational power resulting in very slow processing of data and page loading times, with limited concurrent user access.

4 Further development or research

General:

Over time this system will evolve by expanding quantity of data collected, allowing more filter parameters to be implemented and to provide more accurate results. In time through learning more advanced machine learning techniques for tech analysis, more features can be extracted from data and produce more insights.

Performance:

This system's algorithm will be enhanced over time, focusing on improving the overall performance and efficiency.

Commercial:

Trendicator could hopefully be used as a tool for marketers assisting with influencing social media marketing decisions, based on current top topic discussions on social media.

If fashion trend forecasting is made possible with this system, commercial avenues will be pursued. Targeting local fashion retailers, boutique stores, bespoke retailing and independent fashion designers; interested in predicting fashion trends for producing new garments for consumers.

Educational:

Students and consumers with an interest in fashion; studying fashion trends of the past and predicting future trends in their designs, could utilize this system to influence their design project(s).

5 References

(Online, Accessed: 24/11/2017) Available at:
https://en.wikipedia.org/wiki/Bag-of-words_model

(Online, Accessed: 24/11/2017) Available at:
http://ieeexplore.ieee.org/document/6428794/

(Online, Accessed: 26/11/2017) Available at:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.475.4569&rep=rep1&type=pdf

(Online, Accessed: 27/11/2017) Available at:
https://arxiv.org/pdf/0807.2569.pdf

(Online, Accessed: 27/11/2017) Available at:
https://deeplearning4j.org/textanalysis

(Online, Accessed: 28/11/2017) Available at:
https://en.wikipedia.org/wiki/Fashion_capital

(Online, Accessed 01/01/2018)
https://blog.exploratory.io/demystifying-text-analytics-preparing-document-and-term-data-for-text-mining-in-r-4f858feb4b77

(Online, Accessed 03/5/2018) Available at:
https://shiny.rstudio.com/

(Online, Accessed 01/5/2018) Available at:
https://shiny.rstudio.com/

(Online, Accessed 05/05/2018) Available at:
https://www.datasciencecentral.com/profiles/blogs/7-important-model-evaluation-error-metrics-everyone-should-know

6 Appendix

6.1 Project Proposal

"To determine the key indicators which directly affected previous geographical fashion trends becoming trends."

Keith Driscoll

x14346081

x14346081@stu
dent.ncirl.ie

6.1.1 Table of Contents

Table of Contents

6.1.2 Objectives

Knowledge discovery in databases (KDD) is the process of discovering useful knowledge from a collection of data. My objectives for my project are as follows.

- Gathering of data from multiple relatable sources on the internet.
  - Fashion Trends will be searched geographically based on particular countries & regions (e.g Europe/Asia/America).
  - Trends are affected by many factors such as colours, fabrics, celebrities, economy and politics, these must be explored for finding key indicators.
- Processing of gathered data to determine the key indicators which directly affected previous geographical fashion trends becoming trends.
  - Machine learning techniques will be used to improve the analysis of gathered data to improve the determination of key indicators.
- Storing the large volumes of gathered data in a data warehouse to prepare for transformation.
- Data mining of stored data to extract useless structured patterns that emerge from data sources.
  - Patterns must be valid, novel, potentially useful and understandable.
- Data interpretation and Evaluation
  - Data will be interpreted and evaluated, any findings will be output on a webpage in graph format.

6.1.3 Background

My inspiration for pursuing this software project idea is due to my keen interest and current involvement within the Dublin fashion industry.

I have plenty of first-hard experience behind commercial fashion shoots, which focus on current fashion trends; from the concept development phase -> to the shoot planning phase -> on set during a shoot -> publishing of a fashion shoot via physical/digital magazines. With fashion styling being the main focus throughout these shoots; influenced by fashion trends at the time. I am determined to explore which factors directly impacted a fashion trend becoming a trend, and how long it stays as a trend.

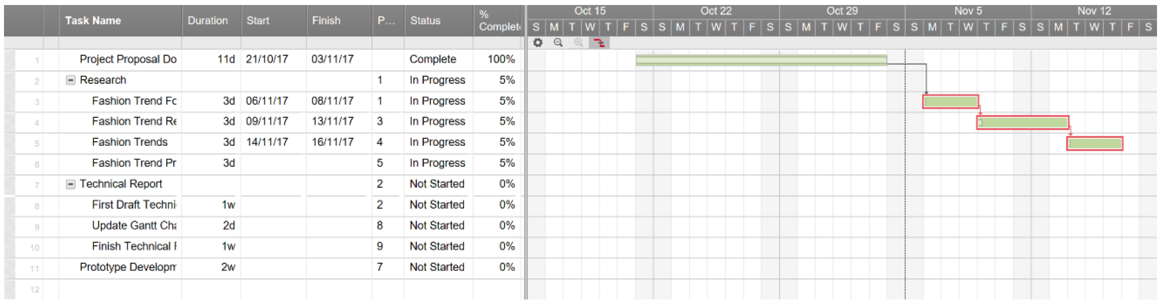
I hope that my project will help me expand my knowledge of the fashion industry worldwide and potentially aid fashion design students/ fashion buyers with my findings.

6.1.4 Technical Approach

Firstly, heavy research and learning of Python will be required. Research is required to better understand how exactly to approach this project, and which route will give the best results. I will also be learning Python so that once I'm finished with research I can soon start on developing a prototype.

6.1.5 Project Plan

Gantt chart using Microsoft Project with details on implementation steps and timelines



	Task Name	Duration	Start	Finish	P…	Status	% Complete
1	Project Proposal Do	11d	21/10/17	03/11/17		Complete	100%
2	Research				1	In Progress	5%
3	Fashion Trend Fc	3d	06/11/17	08/11/17	1	In Progress	5%
4	Fashion Trend Re	3d	09/11/17	13/11/17	3	In Progress	5%
5	Fashion Trends	3d	14/11/17	16/11/17	4	In Progress	5%
6	Fashion Trend Pr	3d			5	In Progress	5%
7	Technical Report				2	Not Started	0%
8	First Draft Techni	1w			2	Not Started	0%
9	Update Gantt Cha	2d			8	Not Started	0%
10	Finish Technical I	1w			9	Not Started	0%
11	Prototype Developm	2w			7	Not Started	0%
12							

6.1.6 Technical Details

Microsoft Azure will be used to setup to run multiple virtual machines to scrape websites for data. Azure will also be used to store and manage my gathered data. The same service will also be used for hosting my webpage which will be used to output my data in graph format.

- Many data analysis, big data and machine learning projects require scraping websites to gather the data which will be worked with. Through internet research, I've chosen **Python** as it's used widely in the data science community, which means there also plenty of online resources I can follow to aid me with my project.
- Large amounts of sources will be scraped for data, **data warehousing** methods will be approached for the storing of data. Data gathered will be complex and data warehousing will allow my data to be more easily maintained, easier to data mine and scalable. Special resources required.
- Relational **SQL** is the desired query language for my database.

Sources of information that will be scraped for usable data in my project will include but not be limited to: digital magazines, fashion blogs, fashion colour & fabric forecast, professional stylist blogs, fashion runway reports, fashion sections of online newspapers.

6.1.7 Evaluation

The prototype will have limited data and perform the basics; which includes manipulating data and outputting it visually via a live web application. I aim to improve this prototype over time by improving the quality of data I collect with machine learning as well as improving my fashion trend indicator definitions.

Signature Date

Signature of student and date

6.2 Mid-Point Technical Report

National College of Ireland

BSc in Computing

2017/2018

Keith Driscoll

X14346081

X14346081@student.ncirl.ie

Identify, using key indicators what directly affected previous geographical fashion trends becoming trends.

Technical Report

National College of Ireland

Table of Contents

Executive Summary

Maximum 300 words. The abstract should mention the problem being addressed, describe the technical solution and briefly report the findings of the evaluation.

Fashion trends are always changing. In the pre-Internet world, the trend forecasting system was clearly defined. Fashion forecasters were to be found at runway shows, where designers and fashion houses would set the agenda with their collections. Forecasters would decide which looks were the most important and would cause the biggest impact in their target markets. These would be outlined in trend reports for their clients – most often chain and department stores – who would design their collections accordingly. Images would trickle down to the public via fashion magazines, and 12 to 18 months after the magazines hit the stands, retail versions of catwalk designs would hit the street. This was how things worked until the Internet happened.

The aim of my project is to identify using key indicators (colour, texture, material), what affected previous fashion trends becoming a trend. This will be achieved applying machine learning techniques aimed at improving (text analysis); data mining for feature extraction and dimensional reduction of text scraped from online webpages.

A user can interact with the system to visually preview, geographically, a previous fashion trend in a select number of years and their seasonal trend. Results are displayed by Year and Season, based on the selected criteria, results are shown for the key indicators: colour, material and texture. The results are then compared with the historical fashion trend from that year & season. This is to compare results from the web application and test for it's accuracy with the fashion trend indicators.

Introduction

Background

Fashion trends are always changing. A fashion trend used to be predictable as it was created by the top influential designers. With the creation of the internet, fashion trends have become more unpredictable than ever. Trends were no longer created by a group of individuals, the public(consumers) had a huge involvement towards fashion trends.

Aims

I want to develop a system that will identify, using key indicators (colour, texture, material) what directly affected previous geographical (select countries within Europe) fashion trends becoming trends. This will be achieved applying machine learning techniques aimed at improving data mining for feature extraction and dimensional reduction of text scraped from online webpages. Results will be visually displayed on an easy to use interactive web application.

Technologies

R Shiny – A open source R package that provides a powerful web framework to develop my data visualisation aspect of my web application.

MySQL – A well-structured relational database model used to store my data used for visual output with R Shiny.

R Shiny Server – A cloud platform which will be used host and manage my database, as well as deploy my web application.

Bootstrap – A CSS framework that provides multiple pre-defined classes to easily and quickly style my web application.

System

Requirements

Knowledge discovery in databases (KDD) is the process of discovering useful knowledge from a collection of data. My objectives and requirements for my project are as follows.

- Gathering of data from multiple relatable sources on the internet.
  - Fashion Trends will be searched geographically in Paris and London, two cities which have a major influence on international fashion trends. *I decided to narrow down the geographical locations from where I gather my data from so I my dataset won't be too large to manage.*
  - Trends are affected by many factors such as colours, fabrics, celebrities, materials, economy and politics. *I decided to narrow down the factors which influence trends to just the following: colour, material and texture. This will allow me to focus further on the machine learning aspect behind my text mining approach for gathering more accurate and reliable data.*
- Processing of gathered data to determine the key indicators which directly affected previous geographical fashion trends becoming trends.
  - Machine learning techniques will be used to improve the analysis of gathered data to improve the determination of key indicators.
    - Natural Language Processing(NLP) – for effective understanding of human-generated text. Making it easier to parse, analyze and extract structured data from humangenerated text. For this project, we'll be applying NLP towards the English vocabulary.
    - *The Bag-of-words model technique will be used to further improve upon the information retrieval. This technique is used for document classification where*

the occurrence(frequency) of each is used as a feature for training a classifier; also aid with spam filtering.

- Data mining of stored data to extract useless structured patterns that emerge from data sources.
    - Patterns must be valid, novel, potentially useful and understandable.
- Storing the large volumes of gathered data in a data warehouse to prepare for transformation. o A well-defined structured MySQL database will be implemented for storing and transforming collected data.

- Data interpretation and Evaluation o Data will be interpreted and evaluated, any findings will be output on a webpage in graph format.

Functional requirements

Backend System Functionality

Data Integration: All the data are collected and integration from scraped multiple online sources.

Data Selection: Selection of data which could potentially be useful for data mining.

Data Cleaning: Data collected may contain errors, missing values or be inconsistent. Techniques are applied to get rid of such anomalies.

Data Transformation: Transforming data into forms appropriate for mining. Techniques applied to the data to accomplish this include smoothing, aggregation and normalization.

Data Mining: Data mining algorithms are implemented to discover interesting patterns that could be utilized. Techniques applied to the data include support vector machines(SVM); which can be used for classification, regression, or other tasks like outlier’s detection, and The Apriori algorithm; find frequent item sets from a transaction Dataset and derive association rules.

Pattern Evaluation & Knowledge Presentation: This step involves visualization, transformation and removing redundant patterns from the patterns generated prior.

Use of Discovered Knowledge: j

User must be able to select from multiple criteria to filter their search of past fashion trends.

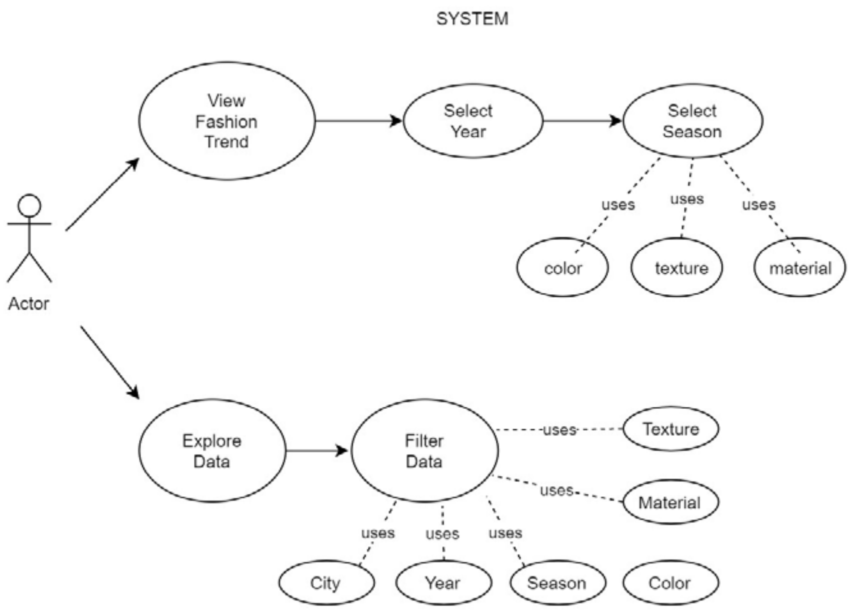
Front-end System Functionality

The system must successfully retrieve requested data from the database, and output the results on a visual graph.

Outputted data from the system must be compared with the historical ‘fashion trend’ of the same criteria(Year, Season) to test for accuracy.

Use Case Diagram

My Use Case Diagram of an actor(User) interacting with the overall system.



Requirement 1: View Fashion Trend User selects a year from a dropdown menu.

Description & Priority

A description of the requirement and its priority. Describes how essential this requirement is to the overall system.

User must select a year to further progress through the trend search algorithm. Upon selection of year, more options become available to the user. A Year must be selected to progress any further.

Use Case: View Fashion Trend

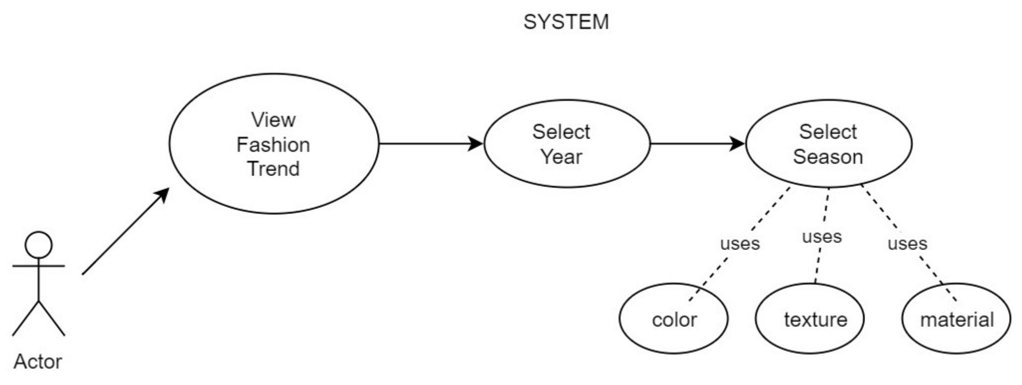
Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

Scope: Allow a user to view specific criteria of a past fashion trend based on year and season.

Description

The user can view visually represented data of a geographical fashion trend, based on year and trend season.

Use Case Diagram: View Fashion Trend



Flow Description

Precondition

The system is in initialisation mode.

This use case starts when the User selects a year from a list of years given.

Main flow

1. The System displays a dropdown list of years the user can choose from.
2. The User selects a year from the dropdown list supplied.
3. The system records the User's selected year and updates the visual data map with a default Season selected; outputting trends of the following indicators: colour, material, texture. <A1>
4. The User selects one of four Seasons; Spring, Summer, Autumn, Winter.
5. The system records the User's selected Season and updates the visual data map with the previously selected Year and current selected Season; outputting trends of the following indicators: colour, material, texture.

Alternate flow

A1. The system outputs the visual data map with default criteria.

Exceptional flow

1. The system outputs the visual data map with default criteria.

Termination

The system presents the visual fashion trend data based on whatever criteria is selected. Default criteria is displayed if not altered.

Post condition

The year selected is saved and an updated view is shown. The system then enters a waiting state.

Requirement 2: Explore Data

Description & Priority

The entire dataset will appear within a table which the user can filter through.

This is not a high priority and not essential to the overall system.

Use Case: Explore Data

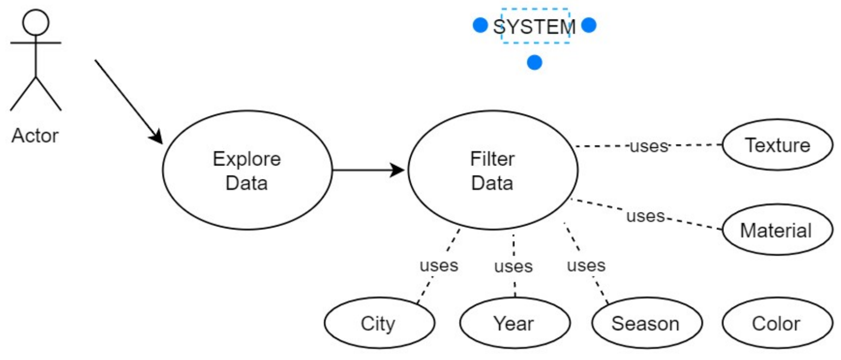
Scope

The scope of this use case is to show all the data being used to visually output the graph and allow the user to filter through the data.

Description

This use case describes the User exploring the data outputted within a table.

Use Case Diagram: Explore Data



Flow Description

Precondition

The system can access the data and output everything within a table.

Activation

This use case starts when the User loads the Data Explorer page of the system.

Main flow

6. The System shows the all the data used in the visual graph in a table.
7. The User selects an attribute from a dropdown box, to filter the columns.
8. The System updates the table to display rows and columns which match the User's selected attribute.

Alternate flow

A1: The system outputs the visual data map with default criteria.

Exceptional flow

A1: The system outputs the table of data unfiltered.

Termination

The User navigations to another page on the web application.

Post condition

The system outputs the table of data with the User's selected criteria to filter it by. System goes into a waiting state.

Data requirements

Dataset must be well-formed, structured and non-redundant. The Dataset must contain relative information aimed towards evaluation of fashion trend key indicators; such as colour, material, texture.

User requirements

A user must be able to access the web application and interact with the visual data filtering criteria.

Upon adjusting filter criteria, the user should receive an almost instant change of data representation, live.

Environmental requirements

The system should be capable of running on any computing device with access to the internet and an internet browser; used to load the web application. If environmental impacts were to be considered, the power consumption of the onlocation server(s) used to host the web application would need to be assessed.

Usability requirements

The user-interface must be interactive and easy-to-use.

The web application must be easy to navigate through.

Any errors that arise must be appropriately identifiable with an easy to understand message corresponsive, navigating the user away from the error that occurs.

Fashion Terminology Dictionary

A dictionary of Fashion Terminology would have to be developed for the machine learning AI to better implement Natural Language Processing when data mining is taking place for the dataset.

Interface requirements

This section describes how the software interfaces with other software products or users for input or output. Examples of such interfaces include APIs, web services, shared memory, data streams, and so forth. Most systems would have a GUI. Add more subsections for other interfaces as reuired.

Interface must allow the User to select one of multiple options, critera that will directly affect the output of the visual data graph. User must not be able to type in their own input, inputs must be predefined; e.g. A user can only select from a list of years provided, they cannot type in the year. This will prevent any errors from occuring from unrecognised inputs by the user of the system.

User must not be allowed to modify the values within the data set; they must only be allowed to view it.

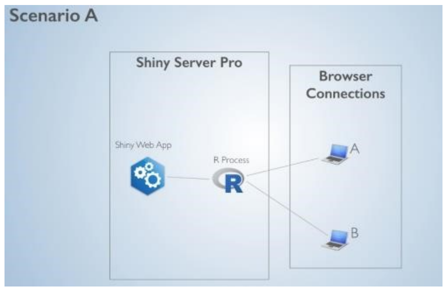
Application Programming Interfaces (API)

Explain which interfaces your system offers or which are used by your system. Examples include Google maps and Weka.

R Shiny will be used to an interactive map which will be used to graphically display data outputted by my algorithm. Based on user's selected criteria, the graph will update live.

Design and Architecture

Describe the design, system architecture and components used. Describe the main algorithms used in the project. (Note use standard mathematical notations if applicable). An architecture diagram may be useful. In case of a distributed system, it may be useful to describe functions and/or data structures in each component separately.



R Shiny Server connects the second user to the same R process. Because R is single threaded, if user A and user B both change an input and trigger R calculations their requests will be handled sequentially. (User B will have to wait for user A’s calculation to complete and their own calculation to complete before they will see an updated output).

R Shiny Server has a few limiters in place:

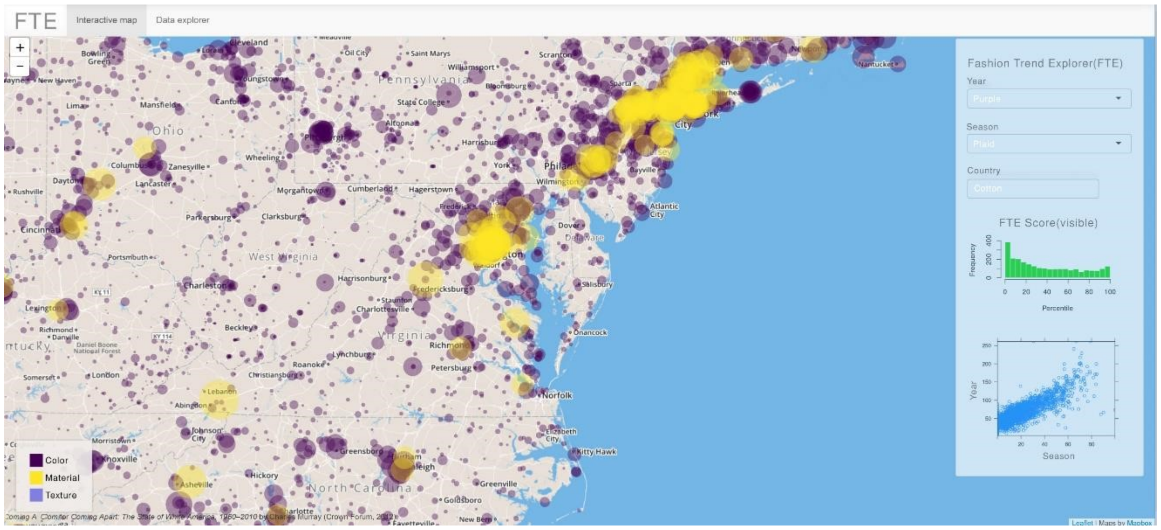
- Max Connections – 20 users can be connected concurrently.
- Max Processes – Maximum number of R processes that can be created per user connected to the R Shiny Server.

These two limiters have a direct impact on the overall performance of the web application.

Graphical User Interface (GUI) Layout

Fashion Trend Explorer

This is the GUI interface of the main system. This is the interactional interface of the web platform in which the user can update the visual representation of data, live, based on their selected criteria available on-screen.



Fashion Trend Explorer(FTE)

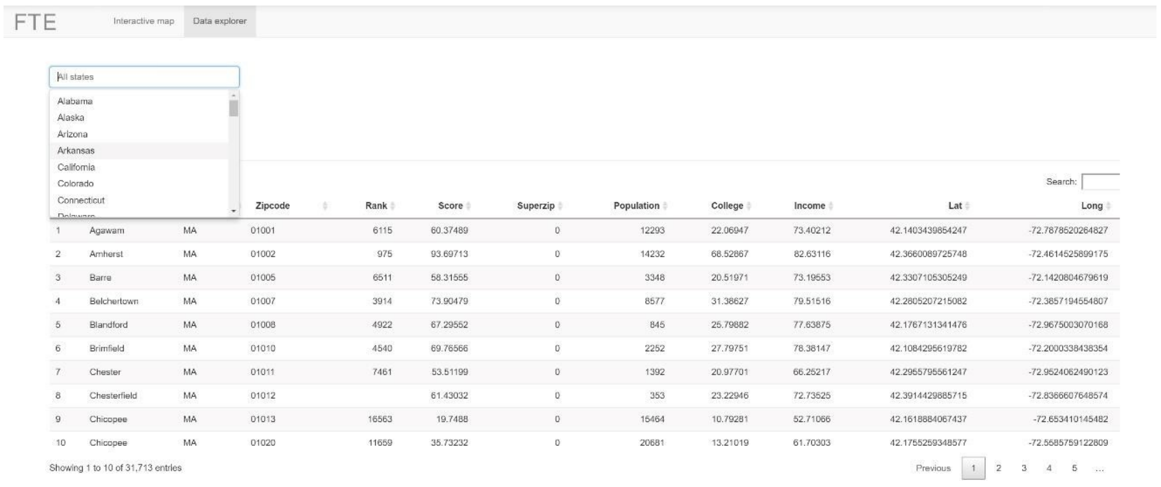
Year: Purple

Season: Plaid

Country: Cotton

Data Explorer

This is the GUI interface of the data explorer tab. This where the user can explore the dataset used to generate the visual data graph on the Fashion Trend Explorer tab. The user can further filter through the data based on column name criteria.



			Zipcode	Rank	Score	Superzip	Population	College	Income	Lat	Long
1	Agawam	MA	01001	6115	60.37489	0	12293	22.06947	73.40212	42.1403439854247	-72.7878520264827
2	Amherst	MA	01002	975	93.69713	0	14232	68.52867	82.63116	42.3660089725748	-72.4614525899175
3	Barre	MA	01005	6511	58.31555	0	3348	20.51971	73.19553	42.3307105305249	-72.1420804679619
4	Belchertown	MA	01007	3914	73.90479	0	8577	31.38627	79.51516	42.2805207215082	-72.3857194554807
5	Blandford	MA	01008	4922	67.29552	0	845	25.79882	77.63875	42.1767131341476	-72.9675003070168
6	Brimfield	MA	01010	4540	69.76566	0	2252	27.79751	78.38147	42.1084295619782	-72.2000338438354
7	Chester	MA	01011	7461	53.51199	0	1392	20.97701	66.25217	42.2955795561247	-72.9524062490123
8	Chesterfield	MA	01012		61.43032	0	353	23.22946	72.73525	42.3914429885715	-72.8366607648574
9	Chicopee	MA	01013	16563	19.7488	0	15464	10.79281	52.71066	42.1618884067437	-72.653410145482
10	Chicopee	MA	01020	11659	35.73232	0	20681	13.21019	61.70303	42.1755259348577	-72.5585759122809

Showing 1 to 10 of 31,713 entries

Non-Functional Requirements

Specifies any other non-functional attributes required by the system. Examples are provided below.

Performance/Response time requirement

Web application is expected to return a response to the User within 3 seconds.

Availability requirement

All the data gathered, and code used will be publicly accessed online via an online GitHub repository.

Recover requirement

System is expected to recover from a server crash with minimal loss/damage to data.

Robustness requirement

Correct output of errors messages is required. The system must be capable of coping with errors during execution and cope with erroneous input.

Security requirement

There is no security requirement applicable for this system. No sensitive data is being processed between the user and the system. All data used within the system is publicly accessed.

Reliability requirement

The system is expected to operate without failure for a specified number of uses over a specified period of time.

Maintainability requirement

The system can be easily modified to correct faults, improve performance and expand the data being accessed; capable of adapting to a changed environment.

Portability requirement

The system is expected to be used across personal devices capable of internet access and a web browser. Resources required by the system's user is minimal.

Extendibility requirement

The system is to be capable of adapting to changes of specification such as more key indicators which affected previous fashion trends.

Reusability requirement

The software behind the system is adaptable to be used for other applications with minimal modifications required. Other types of text analytics

Conclusions

Advantages:

Capability of visually representing the collected data of previous fashion trends via an interactive geographical map.

Data can be filtered according to different criteria specified by the user.

Disadvantages:

A lot of data processing is involved, which can be very time consuming.

Accuracy isn’t guaranteed.

Only works for one language(English).

Opportunities:

Possibility of this project bearing commercial opportunities if fashion trend predictions can be made to a high degree of accuracy.

Application could be translated into multiple languages.

Algorithm could be improved to produce better accuracy of results to a certain degree.

Real-time fashion trend predictions could be implanted.

Limits:

Very resource intensive. The text analysis approach combined with machine learning will require a lot of computer resources for data mining.

Size of dataset is limited.

Further development or research

General:

Over time this system will evolve by expanding the datasets, allowing more filter parameters to be implemented. This would allow other filter parameters such as print/graphics, style-genre, retail, footwear and accessories. As more fashion trend indicators from previous years are identified and expanded on. A possibility of fashion trend patterns which could be identified and implemented in a fashion trend forecasting algorithm.

Performance:

This system's algorithm will be enhanced over time, focusing on improving the overall performance and efficiency.

Commercial:

If fashion trend forecasting is made possible with this system, commercial avenues will be pursued. Targeting local fashion retailers, boutique stores, bespoke retailing and independent fashion designers; interested in predicting fashion trends for producing new garments for consumers.

Educational:

Students and consumers with an interest in fashion; studying fashion trends of the past and predicting future trends in their designs, could utilize this system to influence their design project(s).

Personal:

This system could also be implemented as a tool for creatives such as fashion photographers, videographers and stylists with concept development of editorials, or even commercial work closely related with fashion trends of the past or future.

References

(Online, Accessed: 24/11/2017) Available at: https://en.wikipedia.org/wiki/Bag-of-words_model

(Online, Accessed: 24/11/2017) Available at: http://ieeexplore.ieee.org/document/6428794/

(Online, Accessed: 26/11/2017) Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.475.4569&rep=rep1&type=pdf

(Online, Accessed: 27/11/2017) Available at:

https://arxiv.org/pdf/0807.2569.pdf

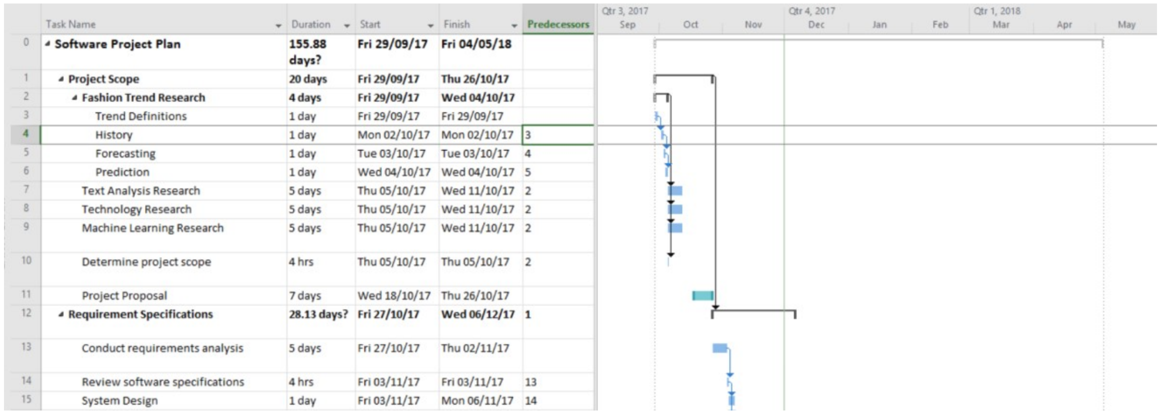
(Online, Accessed: 27/11/2017) Available at:

https://deeplearning4j.org/textanalysis

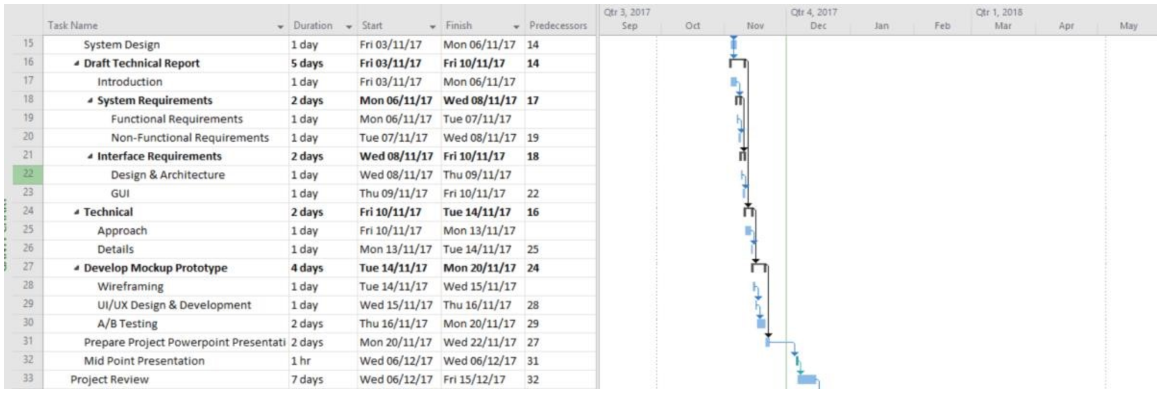
(Online, Accessed: 28/11/2017) Available at: https://en.wikipedia.org/wiki/Fashion_capital

(Online, Accessed 30/12/2017) Available at: https://shiny.rstudio.com/

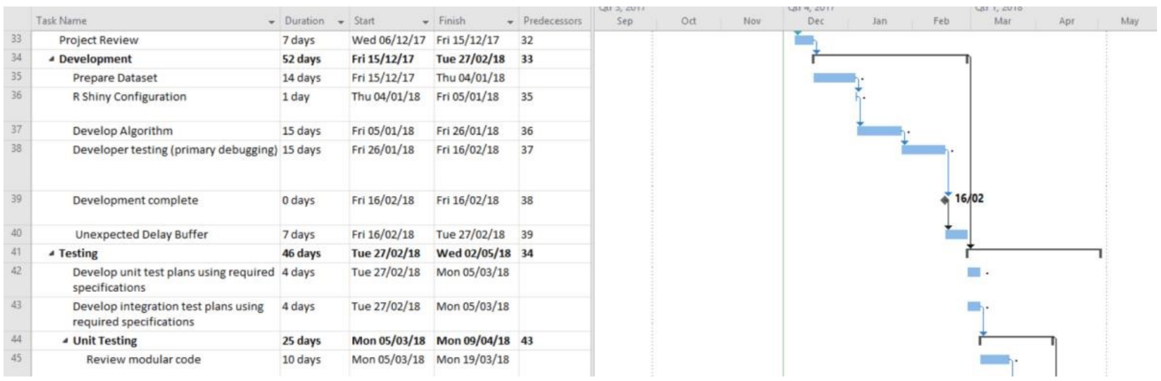
6.3 9.1 Project Plan



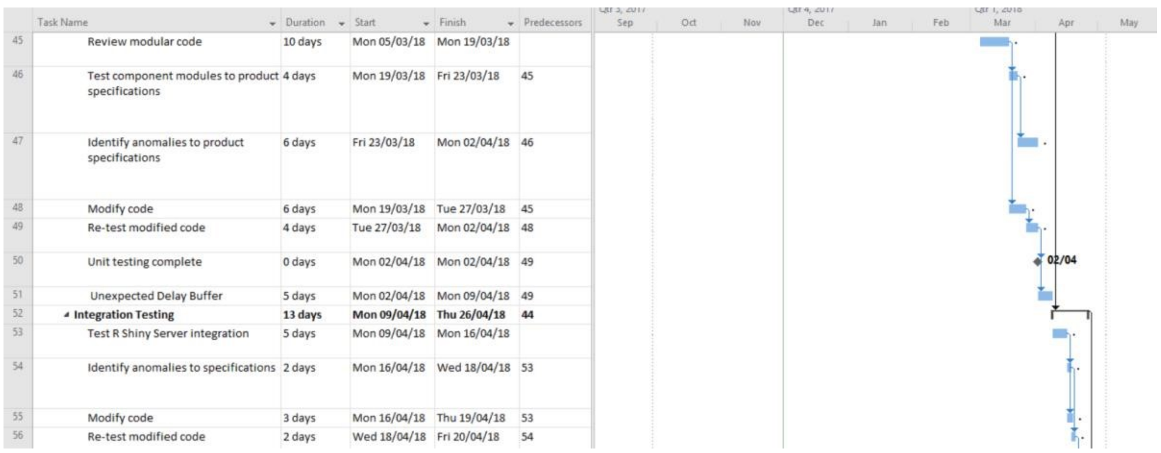
	Task Name	Duration	Start	Finish	Predecessors
0	**Software Project Plan**	**155.88 days?**	**Fri 29/09/17**	**Fri 04/05/18**	
1	**Project Scope**	**20 days**	**Fri 29/09/17**	**Thu 26/10/17**	
2	**Fashion Trend Research**	**4 days**	**Fri 29/09/17**	**Wed 04/10/17**	
3	Trend Definitions	1 day	Fri 29/09/17	Fri 29/09/17	
4	History	1 day	Mon 02/10/17	Mon 02/10/17	3
5	Forecasting	1 day	Tue 03/10/17	Tue 03/10/17	4
6	Prediction	1 day	Wed 04/10/17	Wed 04/10/17	5
7	Text Analysis Research	5 days	Thu 05/10/17	Wed 11/10/17	2
8	Technology Research	5 days	Thu 05/10/17	Wed 11/10/17	2
9	Machine Learning Research	5 days	Thu 05/10/17	Wed 11/10/17	2
10	Determine project scope	4 hrs	Thu 05/10/17	Thu 05/10/17	2
11	Project Proposal	7 days	Wed 18/10/17	Thu 26/10/17	
12	**Requirement Specifications**	**28.13 days?**	**Fri 27/10/17**	**Wed 06/12/17**	**1**
13	Conduct requirements analysis	5 days	Fri 27/10/17	Thu 02/11/17	
14	Review software specifications	4 hrs	Fri 03/11/17	Fri 03/11/17	13
15	System Design	1 day	Fri 03/11/17	Mon 06/11/17	14



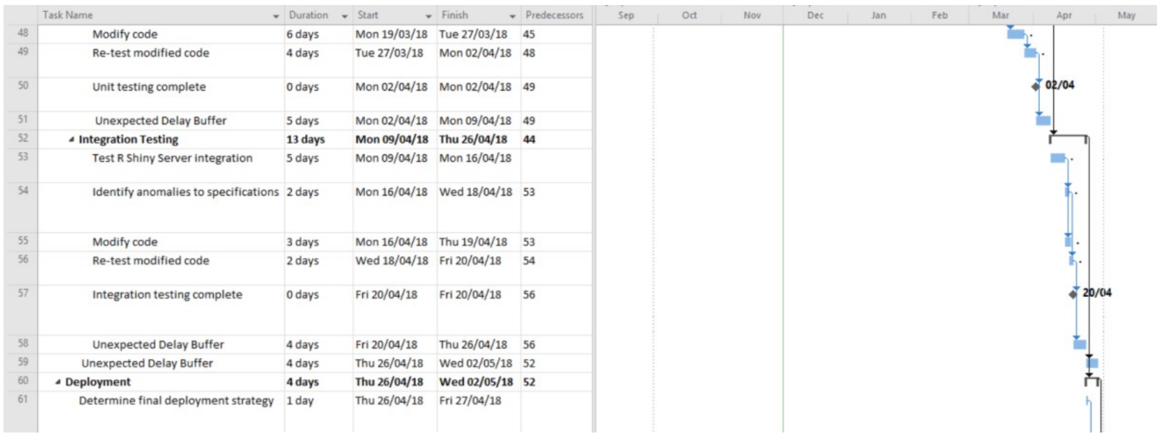
	Task Name	Duration	Start	Finish	Predecessors
15	System Design	1 day	Fri 03/11/17	Mon 06/11/17	14
16	**Draft Technical Report**	**5 days**	**Fri 03/11/17**	**Fri 10/11/17**	**14**
17	Introduction	1 day	Fri 03/11/17	Mon 06/11/17	
18	**System Requirements**	**2 days**	**Mon 06/11/17**	**Wed 08/11/17**	**17**
19	Functional Requirements	1 day	Mon 06/11/17	Tue 07/11/17	
20	Non-Functional Requirements	1 day	Tue 07/11/17	Wed 08/11/17	19
21	**Interface Requirements**	**2 days**	**Wed 08/11/17**	**Fri 10/11/17**	**18**
22	Design & Architecture	1 day	Wed 08/11/17	Thu 09/11/17	
23	GUI	1 day	Thu 09/11/17	Fri 10/11/17	22
24	**Technical**	**2 days**	**Fri 10/11/17**	**Tue 14/11/17**	**16**
25	Approach	1 day	Fri 10/11/17	Mon 13/11/17	
26	Details	1 day	Mon 13/11/17	Tue 14/11/17	25
27	**Develop Mockup Prototype**	**4 days**	**Tue 14/11/17**	**Mon 20/11/17**	**24**
28	Wireframing	1 day	Tue 14/11/17	Wed 15/11/17	
29	UI/UX Design & Development	1 day	Wed 15/11/17	Thu 16/11/17	28
30	A/B Testing	2 days	Thu 16/11/17	Mon 20/11/17	29
31	Prepare Project Powerpoint Presentati	2 days	Mon 20/11/17	Wed 22/11/17	27
32	Mid Point Presentation	1 hr	Wed 06/12/17	Wed 06/12/17	31
33	Project Review	7 days	Wed 06/12/17	Fri 15/12/17	32



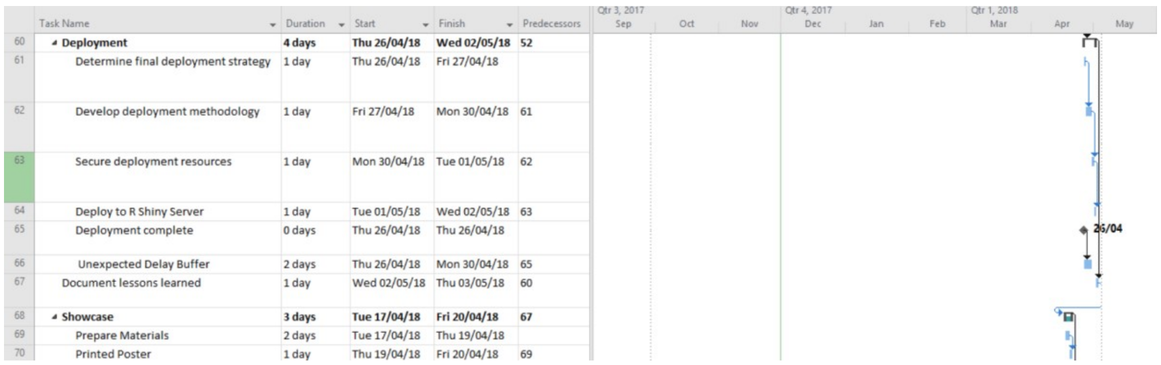
	Task Name	Duration	Start	Finish	Predecessors
33	Project Review	7 days	Wed 06/12/17	Fri 15/12/17	32
34	**Development**	**52 days**	**Fri 15/12/17**	**Tue 27/02/18**	**33**
35	Prepare Dataset	14 days	Fri 15/12/17	Thu 04/01/18	
36	R Shiny Configuration	1 day	Thu 04/01/18	Fri 05/01/18	35
37	Develop Algorithm	15 days	Fri 05/01/18	Fri 26/01/18	36
38	Developer testing (primary debugging)	15 days	Fri 26/01/18	Fri 16/02/18	37
39	Development complete	0 days	Fri 16/02/18	Fri 16/02/18	38
40	Unexpected Delay Buffer	7 days	Fri 16/02/18	Tue 27/02/18	39
41	**Testing**	**46 days**	**Tue 27/02/18**	**Wed 02/05/18**	**34**
42	Develop unit test plans using required specifications	4 days	Tue 27/02/18	Mon 05/03/18	
43	Develop integration test plans using required specifications	4 days	Tue 27/02/18	Mon 05/03/18	
44	**Unit Testing**	**25 days**	**Mon 05/03/18**	**Mon 09/04/18**	**43**
45	Review modular code	10 days	Mon 05/03/18	Mon 19/03/18	



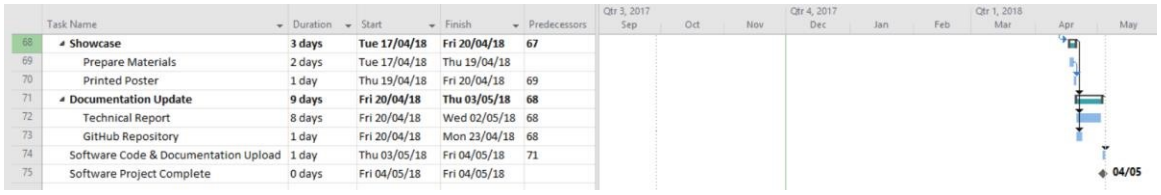
	Task Name	Duration	Start	Finish	Predecessors
45	Review modular code	10 days	Mon 05/03/18	Mon 19/03/18	
46	Test component modules to product specifications	4 days	Mon 19/03/18	Fri 23/03/18	45
47	Identify anomalies to product specifications	6 days	Fri 23/03/18	Mon 02/04/18	46
48	Modify code	6 days	Mon 19/03/18	Tue 27/03/18	45
49	Re-test modified code	4 days	Tue 27/03/18	Mon 02/04/18	48
50	Unit testing complete	0 days	Mon 02/04/18	Mon 02/04/18	49
51	Unexpected Delay Buffer	5 days	Mon 02/04/18	Mon 09/04/18	49
52	**◢ Integration Testing**	**13 days**	**Mon 09/04/18**	**Thu 26/04/18**	**44**
53	Test R Shiny Server integration	5 days	Mon 09/04/18	Mon 16/04/18	
54	Identify anomalies to specifications	2 days	Mon 16/04/18	Wed 18/04/18	53
55	Modify code	3 days	Mon 16/04/18	Thu 19/04/18	53
56	Re-test modified code	2 days	Wed 18/04/18	Fri 20/04/18	54



	Task Name	Duration	Start	Finish	Predecessors
48	Modify code	6 days	Mon 19/03/18	Tue 27/03/18	45
49	Re-test modified code	4 days	Tue 27/03/18	Mon 02/04/18	48
50	Unit testing complete	0 days	Mon 02/04/18	Mon 02/04/18	49
51	Unexpected Delay Buffer	5 days	Mon 02/04/18	Mon 09/04/18	49
52	**◢ Integration Testing**	**13 days**	**Mon 09/04/18**	**Thu 26/04/18**	**44**
53	Test R Shiny Server integration	5 days	Mon 09/04/18	Mon 16/04/18	
54	Identify anomalies to specifications	2 days	Mon 16/04/18	Wed 18/04/18	53
55	Modify code	3 days	Mon 16/04/18	Thu 19/04/18	53
56	Re-test modified code	2 days	Wed 18/04/18	Fri 20/04/18	54
57	Integration testing complete	0 days	Fri 20/04/18	Fri 20/04/18	56
58	Unexpected Delay Buffer	4 days	Fri 20/04/18	Thu 26/04/18	56
59	Unexpected Delay Buffer	4 days	Thu 26/04/18	Wed 02/05/18	52
60	**◢ Deployment**	**4 days**	**Thu 26/04/18**	**Wed 02/05/18**	**52**
61	Determine final deployment strategy	1 day	Thu 26/04/18	Fri 27/04/18	



	Task Name	Duration	Start	Finish	Predecessors
60	**◢ Deployment**	**4 days**	**Thu 26/04/18**	**Wed 02/05/18**	**52**
61	Determine final deployment strategy	1 day	Thu 26/04/18	Fri 27/04/18	
62	Develop deployment methodology	1 day	Fri 27/04/18	Mon 30/04/18	61
63	Secure deployment resources	1 day	Mon 30/04/18	Tue 01/05/18	62
64	Deploy to R Shiny Server	1 day	Tue 01/05/18	Wed 02/05/18	63
65	Deployment complete	0 days	Thu 26/04/18	Thu 26/04/18	
66	Unexpected Delay Buffer	2 days	Thu 26/04/18	Mon 30/04/18	65
67	Document lessons learned	1 day	Wed 02/05/18	Thu 03/05/18	60
68	**◢ Showcase**	**3 days**	**Tue 17/04/18**	**Fri 20/04/18**	**67**
69	Prepare Materials	2 days	Tue 17/04/18	Thu 19/04/18	
70	Printed Poster	1 day	Thu 19/04/18	Fri 20/04/18	69



	Task Name	Duration	Start	Finish	Predecessors
68	**◢ Showcase**	**3 days**	**Tue 17/04/18**	**Fri 20/04/18**	**67**
69	Prepare Materials	2 days	Tue 17/04/18	Thu 19/04/18	
70	Printed Poster	1 day	Thu 19/04/18	Fri 20/04/18	69
71	**◢ Documentation Update**	**9 days**	**Fri 20/04/18**	**Thu 03/05/18**	**68**
72	Technical Report	8 days	Fri 20/04/18	Wed 02/05/18	68
73	GitHub Repository	1 day	Fri 20/04/18	Mon 23/04/18	68
74	Software Code & Documentation Upload	1 day	Thu 03/05/18	Fri 04/05/18	71
75	Software Project Complete	0 days	Fri 04/05/18	Fri 04/05/18	

6.4 Monthly Journals

September

This month I conducted further research into fashion trends as it’s the key area of interest for my project. I also met with my supervisor Thibaut and we agreed to have meetings every two weeks to measure the progress of my project.

October

After meeting with Thibaut trying to find a suitable approach towards my software project, we were still no where close towards achieving that goal. I still had a lot more research to do on fashion trends to figure out how I can identify a fashion trend.

November

Exams are nearing so I have spent less time towards my project, so I could focus on my Continuous Assessments with deadlines nearing as exams are approaching.

This month I mostly watched YouTube videos on have fashion retailers such as Zara and H&M use Big data to make important decision on which clothes they put on sale and how they manage their inventories. Turns out that Big Data analysis is very important for these businesses and plays a major role to their fast-fashion successes.

December

This month I haven’t done any work on my project as I’ve been focusing on my exams. I have been really worried about my Artificial Intelligence exam as I did not fair too well in the CA’s.

January

This month I was informed that my project supervisor will no longer be able to supervise me on my project due to personal reasons happening back at home in Paris. This upset me as my project was heavily relying on machine learning which he specialised in. I also found out that I failed my Artificial Intelligence module with a result of 34.9%. I hope I am still eligible by compensation.

February

My project supervisor was Eamon Nolan who I am happy to have as my supervisor, although he may not be very helpful towards machine learning. I met with Eamon this month and we finished my Project Showcase profile

March

This month has been very busy as the project deadline is nearing! I still don’t have a working version of my project. I hope I can finish this project on time. Eamon also asked for progress on my Technical report which I currently haven’t updated since the mid-point presentations. I need to send it to him soon!

May

This month has started off super crazy as I rush to complete my final year project. In a hope to do so, I scrapped my idea of doing data visualisation on Tableau and modified my prototype that was used for my mid-point presentation. This saved me a lot of time! I also managed to update my Technical report and sent it to my supervisor, he replied very quick and was super helpful!