Spring 2017

Aspect Based Sentiment Analysis of Walmart Customer Reviews using Hybrid Pattern Recognition

TEAM RVESTERS

ANITA, ANJANA, PRASHANTH, SRIRAMAN, SURIYAA

University of Connecticut

Table of Contents

1. Executive Summary

Abstract

Customer reviews on e-commerce websites play a vital role in driving the customer purchasing behavior. Walmart currently follows an overall five-star rating system and it is not reflective of the actual sentiments expressed by the buyers and the verbal descriptive reviews do not distinctively enlighten prospective customers about the specific aspects in products or services. It is essential for Walmart to mine user reviews to extract sentiments by different aspects specific to every product (like quality, durability, warranty service, pricing, reliability etc.) and service (like shipping, packaging, return/ replacement policies etc.). This process can help draw specific insights about the product and service which will help both Walmart and potential customers to take more informed decisions.

Approach

The entire project was divided into multiple modules:

- The user reviews were scraped from Walmart.com for the product: “Straight Talk Apple iPhone 5S”
- The reviews were segregated into sentences, cleaned and pre-processed
- Product and service dictionaries were created by collating an exhaustive list of terms relevant to each category
- Hybrid patterns of Parts of Speech in English that best describes the opinion about aspects were identified
- The polarities for the extracted opinionated phrases were calculated
- Based on the range of resulting values, the polarities were classified into: Positive, Negative and Neutral
- A user interface was designed to provide a 360^{o} view of user opinions with respect to the different aspects of the product and service. Additionally, a granular level user opinions were also captured through visually appealing charts and word clouds in RShiny.
- In this process, we explored and utilized several R packages - rvest, tm, openNLP, qdap, Shiny and WordCloud

Results

Opinionated phrases were successfully extracted and sentiment polarity was assigned to each feature corresponding to the aspects. Since there is no existing system to compare the results, manual testing was carried out on a sample of 100 reviews. The manual interpretation of sentiments was compared with the predictions of the algorithm and a classification accuracy metric: ***F1 score*** was computed. The algorithm has classified the sentiments with an overall accuracy of 67%.

Challenges Faced

The major challenges faced in the project were iterating through multiple pages to scrape the user reviews, opinion mining, enhancing the stop word list and function overriding between packages.

Future Scope

In terms of the improving the algorithm, modules like slang correction, spell checks can be incorporated. In terms of scalability, real time analysis with optimized time complexity can be achieved. The scope of the project can be extended by aggregating reviews from other e-commerce websites which will facilitate a ***“One-platform that fits all”*** experience.

2. Modular Approach

2.1. Module 1: Web scraping

- Initialize a dataframe "*allreviews*" for storing all the scraped reviews
- HTML source code of the review page is captured

```
linkid=paste("https://www.walmart.com/reviews/product/50285046?limit=20&page=1&sort=relevancy",sep ="")
link=read_html(linkid) #Read the source code of the HTML page
```

- The CSS element of the HTML page that contains the "no. of reviews" data is extracted
- Based on the number of reviews, the deployed code traverses through multiple pages to obtain all the reviews and not the ones just from the first page

```
no_of_reviews=link %>%
  html_node(".heading-e") %>%
  html_text()
review_count=as.integer(first.word(no_of_reviews))

pages=ifelse(review_count%%20==0,review_count%/%20,review_count%/%20+1)
```

- Each review in Walmart had a unique ID captured within the "*.js-review-list*" class

```
<div class="js-review-list">

    <div class="Grid customer-review js-customer-review" data-content-id="113800930">
      <div class="Grid-col u-size-8-12-m js-customer-review-body customer-review-body"
        <div class="Grid">
```

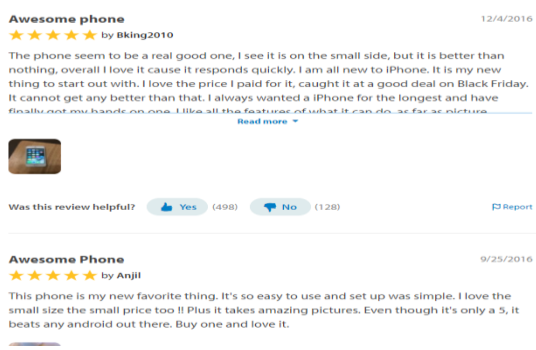
```
review_tag=link %>%
  html_node(".js-review-list") %>%
  html_children()
length(review_tag)
```

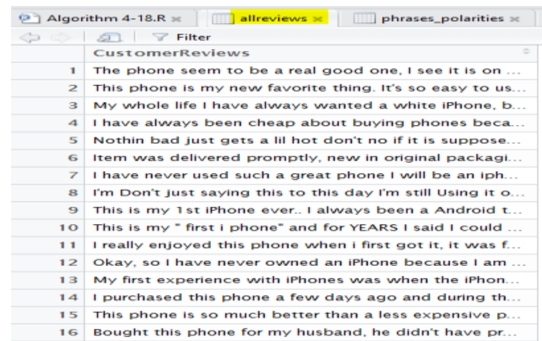
- Based on the length of the review tag, for each child node (n), using SelectorGadget (an open source tool that makes CSS selector generation and discovery on complicated sites a breeze), the appropriate CSS node and child class hierarchies (.js-customer-review:nth-child(n).js-customer-review-text") that contain the review text were identified and extracted.

```
<div class="customer-review-text">
  <p class="js-customer-review-text" data-max-height="110">The phone seem t
with. I love the price I paid for it, caught it at a good deal on Black Fri
ality, they are so clear when I take a picture, love the versatility of it.
 all as I did it all by myself on straight talk website. The price was the
at what you get once it get in your hands and you start operating it. GO fo
</div>
<div class="review-media-img-container review-media-img-container-lg js-rev
```

```
selector=paste(".js-customer-review:nth-child(",j,") .js-customer-review-text")
review=link %>%
  html_node(css=selector) %>%
  html_text()
```

- The extracted text is transferred to the dataframe "allreviews"





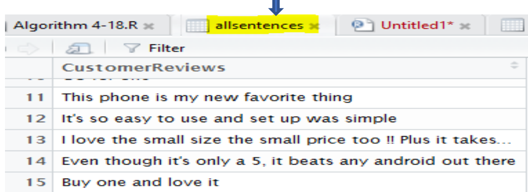
- **Packages & Functions used:**

Package(s)	Function(s)	Functionality
Rvest	Read_html()	Read the HTML source code of the webpage containing reviews
	html_node()	Access the CSS element/class tags
	html_text()	Access the textual content within the CSS element/class tags
	html_children()	Access the child elemnts within a specific CSS element/class tag
Hmisc	first.word()	Identify the first word in the string
Splitstackshape	cSplit()	Split the paragraph of reviews into sentences

2.2. Module 2: Data Pre-processing

- Split the sentences in each review into different rows of a new dataframe "allsentences"

```
> allreviews[2,1]
[1] "This phone is my new favorite thing. It's so easy to use and set up was simple. I love the small size the small price too !! Plus
it takes amazing pictures. Even though it's only a 5, it beats any android out there. Buy one and love it."
```



- Converted the dataframe into a corpus to utilize tm() package functions that removed punctuations and converted the text into lower case

```
corp_sent=Corpus(VectorSource(allsentences$CustomerReviews))
corp_sent=tm_map(corp_sent,FUN =tolower)
corp_sent=tm_map(corp_sent,removePunctuation)
```

- Prepared a custom stop words list to be used for removing them from the sentences (Appendix-Screenshot 1.1)
- Converted the Corpus back to a dataframe "sentences_cleaned". Dataframe was transposed to list the sentences in each row of the data frame instead of column

```
sentences_cleaned=data.frame(text=get("content",corp_sent),stringsAsFactors = F)
sentences_cleaned=t.data.frame(sentences_cleaned)
```

- Trimmed trailing and leading whitespaces using trim() and clean() to trim multiple spaces between words

```
> allsentences[13,1]
[1] I love the small size the small price too !! Plus it takes amazing pictures
```

```
> sentences_cleaned[13,1]
[1] love small size small price takes amazing pictures
```

- ✓ Stopwords “the”, “too” etc. have been removed
- ✓ “Plus” in now “plus”
- ✓ !! removed
- ✓ Single space between all words

- **Packages & Functions used:**

Package(s)	Function(s)	Functionality
Tm	Corpus()	Converting the dataframe into a Corpus object
	VectorSource()	Extends the class Source representing a vector where each entry is interpreted as a document
	tm_map	apply transformation functions by taking a text document as input and returning a text document.
	clean()	Trim additional spaces between words
Gdata	Trim()	Trim leading and trailing spaces in a string
Data.table	t.data.frame	Transpose a dataframe

- Created two exhaustive and distinct lists that hold features related to service and product aspects

```
product_nouns=c("price","screen", "android",  "featuress",  "money",  "apps",   "battery",  "carrier",    "quality",  "camera",
"brand",  "life", "cell", "size", "speaker",  "memory",   "upgrade",       "model",    "storage",  "fingerprint",  "touch",
"button", "charger",  "device", "speed",  "version",  "sound",  "space",  "port", "provider", "unlock", "charge",   "cover",
"performance",  "security", "technology", "voice",  "itunes", "software", "video", "music", "bluetooth", "pixel", "volume",
"settings", "videos", "budget", "cord", "games",  "keyboard", "protector",    "capabilities", "browser", "display", "feature",
"resolution", "siri", "scanner",  "standby",  "thumbprint",      "adapter",  "audio",  "backup", "durability", "functionality",
"feature",  "headphone",  "interface",  "ios",  "jack", "power",  "processor",  "specs",  "usb",  "weight", "width",
"windows",  "wireless", "antivirus","sim","picture",  "product",  "iphone", "phone",  "ipad", "smartphone", "cellphone",
"gadget", "iphone5",  "mac",  "macbook")
service_nouns=c("exchange","contract","coverage","signal","store","activation","network", "discount", "deal", "order",
"shipping", "account",  "sale", "delivery", "package",  "refund", "walmart",  "walmart.com",  "company",  "att",  "gsm","cdma",
"tmobile",  "return", "packaging","cancellation","refund", "delivery",  "replacement",  "pickup", "sales",  "services",
"warranty", "defect", "damage", "insurance",  "offer",  "payment",  "rollback", "shipment", "ship", "tracking","verizon"
,"voucher","discount","coupons", "service")
```

2.3. Module 3: Hybrid Pattern Recognition

- Identified the parts of speech combinations (Appendix-Screenshot 2) using tagPOS() function and such phrases were subset separately using phrases() function
- tagPOS() function returns the different tags for different POS (Parts Of Speech) as in Appendix-Table 1.1
- For the following example, each word is split into different elements of a vector using str_split() and tagPOS() function is applied to each of them

```
posText<- "phone really good amazing pictures clicked"
```

```
s <- unlist(lapply(posText, function(x) { str_split(x, "\n") }))
result <- lapply(s,tagPOS)
result <- as.data.frame(do.call(rbind,result))
result["POStagged"]
result["POStags"]
tags = result["POStags"]
```

```
> result["POStagged"]
                                                        POStagged
1 phone/NN seem/VBP really/RB good/JJ amazing/JJ pictures/NNS clicked/VBN
> result["POStags"]
                         POStags
1 NN, VBP, RB, JJ, JJ, NNS, VBN
```

- Each word is suffixed with "/{tag}" in the POStagged column of the results vector
- POStags column contains only the tags of the words
- Each word in the sentence is unlisted and stored as vector in s1
- Each tag in the POStags column of the results vector is unlisted and stored in tags vector

```
> s1
[1] "phone"    "really"   "good"      "amazing"  "pictures" "clicked"
> tags <- unlist(lapply(text, function(x) { str_split(x, " ") }))
> tags
text1 text2 text3 text4 text5 text6
 "NN"  "RB"  "JJ"  "JJ" "NNS" "VBN"
```

Package(s)	Function(s)	Functionality
NLP and openNLP	Maxent_Sent_Token_Annotator()	Establishing the distribution of parts-of-speech tags for word tokens
	Maxent_Word_Token_Annotator()	Computing the word token annotations
	Maxent_POS_Tag_Annotator()	Computing POS tag annotations
Base R	Grepl()	For matching patterns
Stringr	Str_count()	Counting the number of string matches
	Str_split()	Splitting the string into parts

- grepl() returns TRUE if a string contains the pattern of a tag, otherwise FALSE; if the parameter is a string vector, returns a logical vector (match or not for each element of the vector)

```
grepl(pattern, x, ignore.case = FALSE, perl = FALSE,
      fixed = FALSE, useBytes = FALSE)
```

- For example, nouns are tagged as NN, NNS, NNP, NNPS. Based on the following code, if the tag stored in next.next.pos variable has any of the above strings, grepl() will return TRUE, else FALSE. Boolean conditions written in every loop within the pairs() function uses this logic

```
J <- c("JJ")   #Adj
N <- c("^N[A-Z]*")
pattern matching in
R <- c("^R[A-Z]*")
pattern matching in
V <- c("^V[A-Z]*")
```

- Representation of symbols in the pattern parameter of the grepl() are as follows:
  - ^ starts with
  - [] between
  - * anything after that
- If the sentence has a noun, noun flag is set to 1, else it is set to 0
- The tags obtained from tagPOS() are passed as input to the phrases() function
- The output phrases are appended to a dataframe phrases
- Inside the pairs(), if there is no noun in the sentence, the verbs, adjectives, adverbs are attached to the noun "product" using paste() and the entire phrase is returned

- If the sentence contains a noun (i.e.) tags contain "NN"→ then for each tag → current, previous, next, next next tags are stored in the following variables:

```
prev.pos = tags[i-1] #iden
this.pos = tags[i] #assign
next.pos = tags[i+1] #iden
next.next.pos = tags[i+2]
```

- Hybrid patterns considered in Module 4 are matched/recognized using the grepl() boolean conditions in multiple IF-ELSE loops
- Nearest noun is searched for regular expression patterns that do not contain nouns (E.g. RR, VB)
- The noun is then prefixed with the pattern and stored as a phrase

```
posText<- "phone really good amazing pictures clicked"
```

```
           Phrases
1 phone really good
2  amazing pictures
```

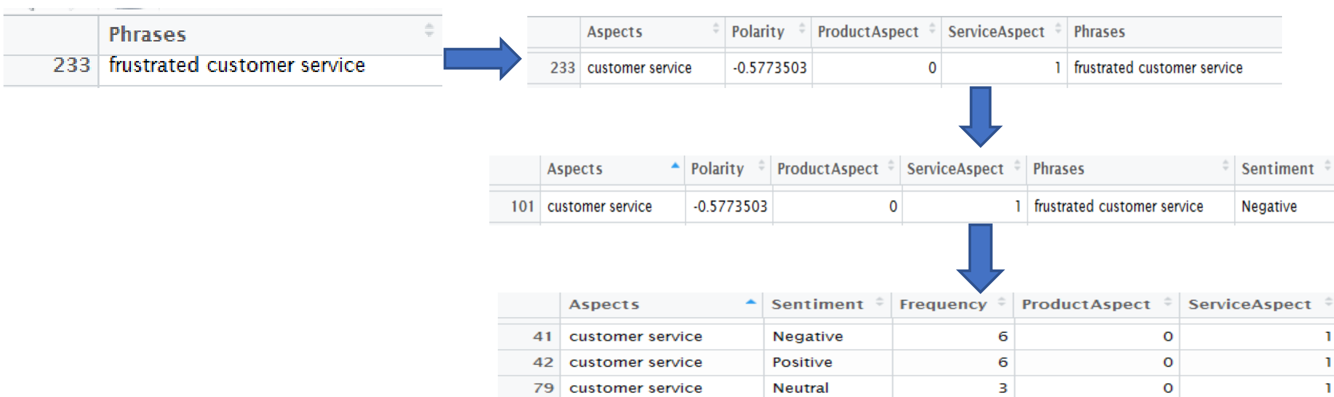
2.4. Module 4: Polarities of phrases

- All the phrases returned by the pairs() function are stored in phrases dataframe
- The output of the module is a dataframe with the following attributes:
  - Aspect/noun from the phrase
  - The entire phrase
  - Polarity value (between -1 and 1)
  - Product aspect flag (1 if the aspect is in the custom product aspects list)
  - Service aspect flag (1 if the aspect is in the custom service aspects list)
- Each phrase is passed to the extractPOS() to extract the noun from the phrase
  - The noun is looked up on the product aspects' list using the match() function
  - If match is found, the position number in the custom list is returned. Product aspect flag is set to 1. Else NULL is returned
  - If NULL is returned, the noun is looked up on the service aspects' list using match() function
  - If there is a match, the position number in the custom list is returned, Service aspect flag is set to 1. Else NULL is returned
- In special cases, there might be multiple nouns in the phrases. In such cases, the nouns are unlisted and for each noun, the above operation is performed
- Each phrase is passed to the function polarity() from the qdap package. It returns a dataframe with multiple attributes. The all.polarity field contains the polarity value between -1 and +1 based on the approximation of the sentiment (polarity) within the text. This value is stored in the output dataframe
- For the aspects that are not in the custom-made product/service dictionaries will have their respective flags set to 0 in the dataframe phrases_polarities. Such records are deleted from the dataframe

- Based on a polarity cutoff, a new column is mutated onto a dataframe indicating whether the sentiment is "Positive" (>0), "Negative" (<0) or "Neutral" (=0)
- A frequency table is constructed based on the frequency of each aspect grouped by the sentiment

Package(s)	Function(s)	Functionality
Qdap	polarity	Approximate the sentiment (polarity) of customer review text
Dplyr	mutate	Add a new column to the dataframe by applying a formula
Base R	match()	Lookup a specific noun onto a custom made aspects' list
	unlist()	Unlist/split a multi-word string into a vector of strings
	wc()	Identify the word count of a string
sqldf	Group By	SQL to identify frequency of aspects grouped by sentiment

- **Example:**



	Phrases
233	frustrated customer service

	Aspects	Polarity	ProductAspect	ServiceAspect	Phrases
233	customer service	-0.5773503	0	1	frustrated customer service

	Aspects	Polarity	ProductAspect	ServiceAspect	Phrases	Sentiment
101	customer service	-0.5773503	0	1	frustrated customer service	Negative

	Aspects	Sentiment	Frequency	ProductAspect	ServiceAspect
41	customer service	Negative	6	0	1
42	customer service	Positive	6	0	1
79	customer service	Neutral	3	0	1

2.5. Module 5: Visualization

Objective

To obtain the list of product and service-related terms corresponding to the product and to create 'Positive', 'Negative' and 'Neutral' word clouds for the same. Subsequently, sentiment-based word clouds also to be created for each of the product and service related aspects along with a bar plot of the percentage of positive, negative and neutral sentiments for the selected aspect.

Packages & functions used

Package(s)	Function(s)	Functionality
WordCloud	wordcloud()	To generate word cloud
Stringi	toString()	To convert an R object into a string format
Stringr	strsplit()	Splits a string of words to create a list of character vectors
Shiny	ShinyApp()	To create a shiny application with a user interface UI
	renderPlot()	To create a plot in the RShiny user interface
Shinythemes	shinytheme()	To add themes to the RShiny user interface

Output obtained from Module 4

	X	Aspects	Polarity	ProductAspect	ServiceAspect	Phrases
1	1	phone	0.0000000	1	0	phone seem real
2	2	phone	1.0392305	1	0	phone real good
3	3	side	0.0000000	0	0	small side
4	4	nothing	0.7071068	0	0	better nothing
5	5	love cause	0.5773503	0	0	overall love cause
6	6	iphone	0.0000000	1	0	new iphone
7	7	thing start	0.0000000	0	0	new thing start
8	8	deal	0.7071068	0	1	good deal
9	9	friday	0.0000000	0	0	black friday
10	10	product	0.7071068	1	0	product better

	X	Aspects	Polarity	ProductAspect	ServiceAspect	Phrases	StrAspects	StrPhrases	Extracted
1	1	phone	0.0000000	1	0	phone seem real	phone	phone seem real	seem real
2	2	phone	1.0392305	1	0	phone real good	phone	phone real good	real good
3	3	side	0.0000000	0	0	small side	side	small side	small
4	4	nothing	0.7071068	0	0	better nothing	nothing	better nothing	better
5	5	love cause	0.5773503	0	0	overall love cause	love cause	overall love cause	overall
6	6	iphone	0.0000000	1	0	new iphone	iphone	new iphone	new
7	7	thing start	0.0000000	0	0	new thing start	thing start	new thing start	new
8	8	deal	0.7071068	0	1	good deal	deal	good deal	good
9	9	friday	0.0000000	0	0	black friday	friday	black friday	black
10	10	product	0.7071068	1	0	product better	product	product better	better
11	11	handss	0.0000000	0	0	longest hands	handss	longest hands	longest hands
12	12	picture quality	0.0000000	1	0	picture quality	picture quality	picture quality	
13	13	picture	0.0000000	1	0	take picture	picture	take picture	take
14	14	time	0.0000000	0	0	take hardly time	time	take hardly time	take hardly
15	15	activation	0.0000000	1	0	activate activation	activation	activate activation	activate

The Aspects and Phrases columns are converted as string values using toString() function and assigned to two columns StrAspects and StrPhrases in a new dataframe – Polarities. Here, Aspect/Noun is separated from the StrPhrases to obtain the respective sentiment/adjective. Now, the bigram strings in the 'Extracted' column are split into their unigram formats and mapped to their respective aspects. Additionally, a SentimentTag is also added to the data frame depending on the Polarity. The below rules are followed while assigning the SentimentTag.

- If Polarity > 0, the Positive tag is set
- If Polarity = 0, the Neutral tag is set
- If Polarity < 0, the Negative tag is set

	Aspect	Adjective	AspAdj	ProductAspect	ServiceAspect	Polarity	SentimentTag	Merged
2	phone	seem	phone seem	1	0	0.0000000	Neutral	phone seem Neutral
3	phone	real	phone real	1	0	0.0000000	Neutral	phone real Neutral
5	phone	real	phone real	1	0	1.0392305	Positive	phone real Positive
6	phone	good	phone good	1	0	1.0392305	Positive	phone good Positive
8	side	small	side small	0	0	0.0000000	Neutral	side small Neutral
10	nothing	better	nothing better	0	0	0.7071068	Positive	nothing better Positive
12	love cause	overall	love cause overall	0	0	0.5773503	Positive	love cause overall Positive
14	iphone	new	iphone new	1	0	0.0000000	Neutral	iphone new Neutral
16	thing start	new	thing start new	0	0	0.0000000	Neutral	thing start new Neutral
18	deal	good	deal good	0	1	0.7071068	Positive	deal good Positive
20	friday	black	friday black	0	0	0.0000000	Neutral	friday black Neutral
22	product	better	product better	1	0	0.7071068	Positive	product better Positive
24	handss	longest	handss longest	0	0	0.0000000	Neutral	handss longest Neutral
27	picture	take	picture take	1	0	0.0000000	Neutral	picture take Neutral
29	time	take	time take	0	0	0.0000000	Neutral	time take Neutral
30	time	hardly	time hardly	0	0	0.0000000	Neutral	time hardly Neutral
32	activation	activate	activation activate	1	0	0.0000000	Neutral	activation activate Neutral

A new column AspAdj is created which merges the Aspect and Adjective columns. This is done to calculate the number of occurrence (frequency) of the Aspect-Adjective bigram as displayed in the Freq column in the below screenshot.

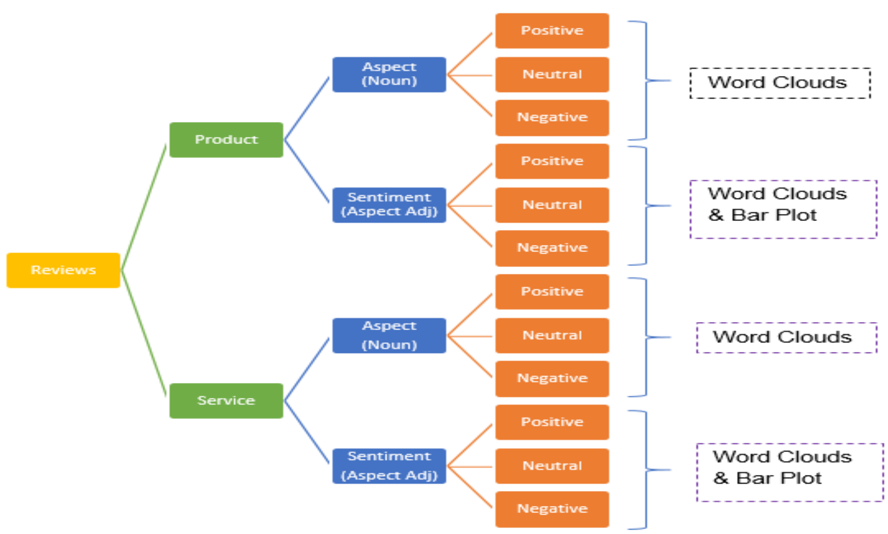
	Merged	Aspect	Adjective	AspAdj	SentimentTag	Freq
3	access datas access Negative	access datas	access	access datas access	Negative	2
4	access datas unable Negative	access datas	unable	access datas unable	Negative	2
54	customer service terrible Negative	customer service	terrible	customer service terrible	Negative	2
124	incoming audio poor Negative	incoming audio	poor	incoming audio poor	Negative	2
129	internet slow Negative	internet	slow	internet slow	Negative	2
153	issuess great Negative	issuess	great	issuess great	Negative	2
154	issuess havent Negative	issuess	havent	issuess havent	Negative	2
186	music louder Negative	music	louder	music louder	Negative	4

Now, to display the sentiment bar plots, we need to calculate the percentage of positive, negative and neutral reviews at the Aspect – Adjective level. For this, the number of reviews are aggregated and the positive, negative and neutral percentages (pp, nup, nep) are obtained.

	fp.Aspects	V1.Negative	V1.Neutral	V1.Positive	total	pp	nup	nep
1	access datas	5	0	0	5	0.000000	0.000000	100.0000000
2	customer service	3	3	3	9	33.333333	33.333333	33.3333333
3	incoming audio	3	0	0	3	0.000000	0.000000	100.0000000
4	internet	3	3	0	6	0.000000	50.000000	50.0000000
5	issuess	5	0	0	5	0.000000	0.000000	100.0000000
6	music	5	0	0	5	0.000000	0.000000	100.0000000
7	phone	43	404	686	1133	60.547220	35.657546	3.7952339
8	phone issuess	7	0	0	7	0.000000	0.000000	100.0000000
9	phoness	3	18	15	36	41.666667	50.000000	8.3333333
10	picturess	3	0	3	6	50.000000	0.000000	50.0000000
11	price	3	14	289	306	94.444444	4.575163	0.9803922
12	problemss	6	3	0	9	0.000000	33.333333	66.6666667
13	product	17	266	1455	1738	83.716916	15.304948	0.9781358
14	sim card	3	5	0	8	0.000000	62.500000	37.5000000
15	time	29	17	0	46	0.000000	36.956522	63.0434783

Word Clouds

To display product and service-related word clouds, on basis of the following product nouns selected, the following list of the service and product subset data frames were formed.

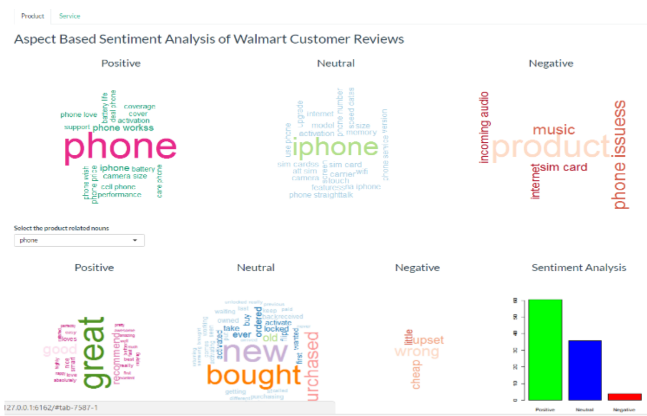


The final output user interface is rendered using RShiny for which the Shiny package is used. A fluid page is created as the Shiny UI and the theme is set as 'flatly'. Two tabs are set – one is the 'Product' tab and the other is the 'Service' tab.

Visualization Guide

- The three word clouds in the first part of the user interface indicate the Positive, Neutral and Negative nouns / aspects for the product (in the Product tab) and service (in the Service tab)

- The three word clouds in the second part of the user interface indicate the Positive, Neutral and Negative sentiment words (adjectives) for each of the aspects (nouns) that are available in the dropdown that is related to the product (in the Product tab) and service (in the Service tab)
- The size of each word in the word cloud denotes the frequency (number of occurrences) of the word in the reviews
- The Sentiment Analysis graph is a translation of the total number of positive, negative and neutral sentiments associated with each of the product and service aspects (nouns)



3. Challenges Faced

- Since only 20 reviews can be viewed per page on Walmart, iterating through multiple webpages to scrape all the reviews of the product was cumbersome
- Manually creating product & service dictionaries was time consuming
- Enhancing the stop word list based on manual skimming of reviews was tedious
- Since multiple packages were used, there was function overriding between packages leading to syntax and logical errors. Debugging the same had some challenges

4. Testing Methodology

Manual intervention was required to test the accuracy of the algorithm since we did not have any metric to validate on.

Steps followed:

- In order to carry out an in-sample testing, we took 100 user reviews from Walmart.com
- Identified the product and service aspects from each of these reviews
- Human interpretation of the sentiments pertaining to each aspect was recorded
- Product and Service flags were captured accordingly
- Compared the actual sentiment vs the predicted results from the algorithm
- Based on the confusion matrix, F-score was calculated

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \qquad \text{Precision} = \frac{tp}{tp + fp} \qquad \text{Recall} = \frac{tp}{tp + fn}$$

5. Results

On comparing the algorithm output with the manually interpreted sentiments, we achieved the following results. Based on the confusion matrix, we computed precision, recall and F1 score for each of the classes.

Actual

Predicted	Positive	Neutral	Negative
Positive	34	3	1
Neutral	17	61	22
Negative	4	2	6

	Precision	Recall	F1-Score
Positive	89%	62%	73%
Neutral	61%	92%	73%
Negative	50%	21%	30%

The results suggest that the algorithm has predicted the positive and neutral sentiments towards product and service aspects with high precision and recall measures. However, there is a bias towards the Neutral sentiments. The algorithm has classified the sentiments towards aspects with an overall accuracy of 67%

6. Future Scope

- **Slang correction -** Internet language is highly prevalent when it comes to writing reviews. It is essential to incorporate the internet slang transformation to achieve better accuracy. Eg. users tend to use “luv” instead of “love” which gets ignored in our polarity calculation algorithm.
- **Spell check functions** - Similarly, users tend to misspell while recording reviews. Eg. “phon” instead of “phone” which will be ignored by the algorithm. Spelling correction module needs to be incorporated to get better results.
- **Scalability -** It is essential to enhance dictionaries that would be relevant to variety of products on the e-commerce website.
- **Real - time analysis -** The current system requires a trigger to populate the visualizations. Real time analysis systems can be developed to facilitate better user experience.
- **One-platform fits all -** The scope of the project can be extended to aggregating the user reviews of the same products from other e-commerce websites which will facilitate the user to get a “One-platform that fits all” experience. This will also prevent customers from digressing to other websites for product comparison and ensure a streamlined shopping experience.

7. References

1. Extracting Product Features and Opinion Words Using Pattern Knowledge in Customer Reviews by Su Su Htay and Khin Thidar Lynn, Article ID 394758 published for The Scientific World Journal Volume 2013 (2013)
2. qdap: Bridging the Gap Between Qualitative Data and Quantitative Analysis, extracted from https://cran.r-project.org/web/packages/qdap/index.html
3. Package 'NLP' extracted from https://cran.r-project.org/web/packages/NLP/NLP.pdf
4. Package 'Shiny' extracted from https://cran.r-project.org/web/packages/shiny/shiny.pdf
5. Package 'rvest' extracted from https://cran.r-project.org/web/packages/rvest/rvest.pdf
6. Package 'ShinyThemes' extracted from https://cran.r-project.org/web/packages/shinythemes/shinythemes.pdf
7. Package 'stringr' extracted from https://cran.r-project.org/web/packages/stringr/stringr.pdf

8. Appendix

```
custom_stopwords=c("see", "get ", "went", "youll", "go", "plus", "able", "saw", "
g",
                    "got", "came", "even,always", "said", "swear", "can", "will",
                    "actual", "stuff", "find", "even", "christmas", "com", "also
"testing",
                    "im", "research", "lots", "mom", "found", "go", "decide", "si
"11272015",
                    "option", " pick anytime", "christmas present", "cdma", "so
                    "16mb", "requirement", "anyone", "far", "hiccup", "son,christ
                    "several", "still", "people", "please", "website", "etc", "il
"finally",
                    "simply", "reached", "everything", "house", "know", "front ",
"must",
                    "surprisingly", "youve", "latest", "understand", "took", "lets
                    "canada", "told", "ahold", "gas", "thinking", "associate", "
                    "add", "asked", "always", "sendingoverallreceiving", "decided"
                    "figured", "didnt", "looking", "within", "pursepocket", "youd
                    "alreadydrive", "shaely")
corp_sent=tm_map(corp_sent,removeWords,c(stopwords("en"),custom_stopwords))
```

Screenshot 1.1

(Adjective, noun)	(low battery), (good memories), (awesome camera), and so forth
(Adjective, noun, noun)	(high quality pictures)
(Adverb, adjective)	(extremely pleased), (very easy), (really annoying), (absolutely amazing), and so forth
(Adverb, adjective, noun)	(very compact camera), (very good pictures), and so forth
(Adverb, verb)	(personally recommend)
(Adverb, adverb, adjective)	(not so bad), and so forth
(Verb, noun)	(recommend camera), (appreciate picture), and so forth
(Verb, adverb)	(perform well)

Screenshot 1.2

JJ	Adjective	Yellow	NN	Noun, sing. Or mass	Cat
JJR	Adj., Comparative	Biggest	NNS	Noun, Plural	Cats
JJS	Adj., Superlative	Busiest	NNP	Proper Noun, singular	iPhone
RB	Adverb	Never	NNPS	Proper Noun, Plural	Carolinas
RBR	Adverb, Comparative	Faster	VB	Verb, base form	Eat
RBS	Adverb, Superlative	Quickest	VBD	Verb, past tense	Ate
VBN	Verb, Past participle	Eaten	VBP	Verb, non-3sg pres	Eat
VBG	Verb, Gerund	Eating	VBZ	Verb 3sg Pres	Eats

Table 1.1