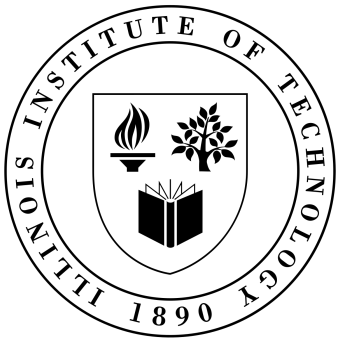


ILLINOIS INSTITUTE OF TECHNOLOGY

CSP- 571 Data Preparation and Analysis

YouTube Comments Sentiment Analysis using R

Susmitha Marripalapu - A20489531 - smarripalapu@hawk.iit.edu

Swetha Radhakrishnan - A20460652 - sradhakrishnan@hawk.iit.edu

Aastha Dhir - A20468022 - adhir2@hawk.iit.edu

December 5, 2022

Abstract

One interesting research area in data science is computational textual analysis. This requires knowledge of a variety of applications and techniques that can be used and applied as they are not readily available in common statistical software packages. Text analysis is well established in R, with a huge collection of dedicated text processing and text analysis packages ranging from low-level string manipulation to advanced text modeling techniques (about 50 packages in total). This project uses three different packages to calculate sentiment scores from YouTube comments and compare the results. These sentiment scores are used to categorize emotions and further visualize the impact the videos have on the masses. These methods can be implemented on any text data to perform sentiment analysis.

Overview

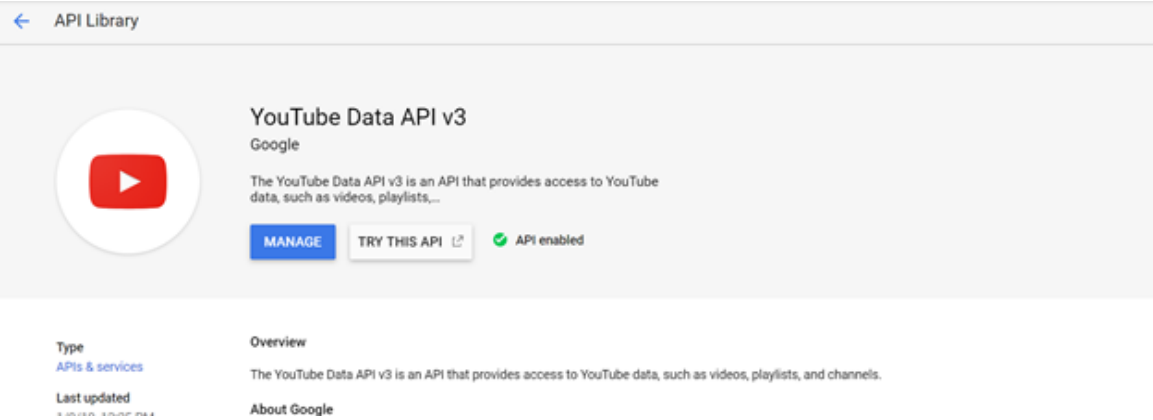
In recent years, social media comments and post types represent the opinions and feedback of the public. Sentiment analysis is performed using varying amounts of data to identify public opinion. This helps in making important business decisions. Some key features extracted from the analysis are polarity - positive or negative, facts - objective statements and opinions - people's point of view. This project uses YouTube comment data to analyze sentiment responses to videos. The main goal of this project is to perform advanced textual analysis on YouTube comment data using the popular open source platform R, which has a wide range of text analysis packages.

This report describes the methodology, implementation, and results of sentiment analysis of comments on eight YouTube videos. Topics such as data preprocessing and challenges in text classification and sentiment analysis are covered in detail. Finally, we compare and analyze the results of different text classification methods in R.

One of the main advantages of doing R text analysis is that it is often possible and relatively easy to switch between different packages and combine them, as you can see in this project. So once the basics of text analysis in R is browsed, access to wider range of advanced text analysis features is gained. For sentiment analysis, we use the 'Sentimentr', 'Syuzhet', and 'Quanteda' (Naive Bayesian Classifier) packages in R. The final performance of the model is measured by plotting the confusion matrix to comprehend the accuracy of the model.

Data Processing

YouTube comment data is imported into the R console using the Google Developer Console's YouTube Data API and the R 'tuber' package. The data includes the video ID, the comments corresponding to each video ID, the author of the comment, the author's channel, the rating, and the timestamp of the published comment.



Sample data of videoId with text comments

	videoId	textDisplay
1	wAZZ-UWGVHI	India is far ahead in this matter.
2	wAZZ-UWGVHI	Lots of credit cards still aren't on there. Everywhere in Canada accepts tap though. Also WSJ is all over apples 🍆 it seems.
3	wAZZ-UWGVHI	Never accept electronic currencies and never allow electronic financial transactions TO ENSURE BUSINESS PRIVACY AND PROTECTION AG
4	wAZZ-UWGVHI	Apple Pay is great until the phone's battery goes flat. Or, there's a widespread power outage and the merchant can only accept cash.
5	wAZZ-UWGVHI	Then there's me, who's been using Samsung Pay literally anywhere that has the original magstripe reader
6	wAZZ-UWGVHI	UPI is a big hit in India (Amazon Pay, Gpay, PhonePe). I guess the amount of Apple Pay transactions in a month is equivalent to UPI payments
7	wAZZ-UWGVHI	Some restaurants refuse to use debit cards or credit cards. Some restaurants are still use cash only.
8	wAZZ-UWGVHI	Wow letting govt spy on you through apps communist
9	wAZZ-UWGVHI	I can't even remember when I used physical card last time
10	wAZZ-UWGVHI	Stop lying, sales propaganda is as bad as any other propaganda.
11	wAZZ-UWGVHI	Payments system in India and china is far ahead. UPI in india is best.
12	wAZZ-UWGVHI	Why TF would I want my wallet to be dependent on battery life? Also with the rate the liberals promoting "social score" it won&#39
13	wAZZ-UWGVHI	I can't stand Apple Pay It would automatically do it on my phone every day I had to take it off!
14	wAZZ-UWGVHI	I use Apple Pay to buy at the school cafeteria.
15	wAZZ-UWGVHI	saying apple pay was adopted slowly? what are they smoking. 8 years is nothing to change from card to digital
16	wAZZ-UWGVHI	Big deal, Japan has been using this kind of payment method for years
17	wAZZ-UWGVHI	While upi killed cards in only few years 😜

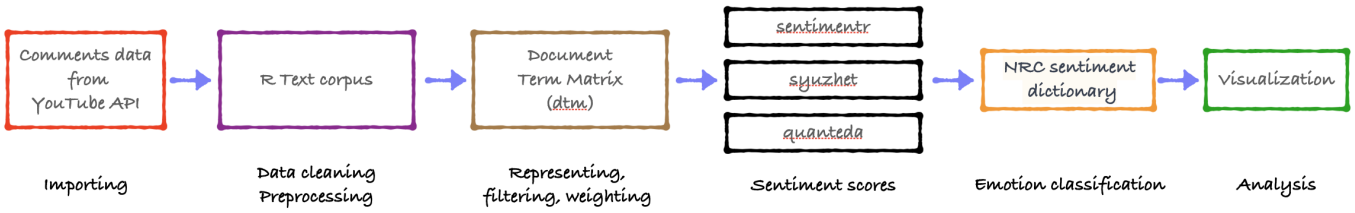
Model testing data:

This project uses a variety of methods to calculate and classify sentiment scores. To test accuracy and compare methods, we use data labeled with Kaggle YouTube statistics. The data is in the form of CSV files that are manually loaded into the R console and used to assess accuracy. For Kaggle data,
https://www.kaggle.com/datasets/advaypatil/youtube-statistics

Data Analysis

The order of operations for sentiment analysis operations can be divided into several categories. The first step, importing data, includes the ability to read data from the YouTube API and save it in the form of a CSV file. Comments in the CSV file are imported into R's raw text corpus. Use string manipulation and preprocessing techniques to manipulate text and process it into textual units such as words and stems. A document term matrix is constructed using tokens, a common technique used to represent corpora. In the next step the sentiment score is calculated using the sentiment, 'syuzhet' and 'quanteda' packages. Emotion classification is done using the NRC sentiment dictionary. All results and comparisons are visualized in R to emotionally categorize the videos and conduct sentiment analysis. Finally, we analyze the differences in the results and outputs of each sentiment analysis package.

Order of sentiment analysis operations



There are many ways to model a text from a mathematical viewpoint. Texts are usually coded as a bag of words (BoW) for text mining. This method disregards both grammatical relations and syntactic categories and simplifies it to unordered collection of words. Consider a group of k texts ,$d_i(i = 1, \ldots, k)$. Text parsing allows us to identify the different tokens used in the collection and create a vocabulary. To reduce the dimensionality of the vocabulary and eliminate the stop-words, a preprocessing step is required. To include more useful combinations of words like the habitual juxtaposition of a particular word with another word and multiwords, "term" is used instead of word. According to the vector space model , each d_i is represented as a vector in the space having b terms belonging to the vocabulary:

$$d_i = t_{i1}, \ldots, t_{im}, \ldots, t_{ib}$$

where t_{im} is the importance of the m^{th} term in di. The document vectors can be arranged in a matrix "X" with k rows (documents) and b columns (terms). The importance of each term in a document is measured by the term frequency which is the number of occurrences of a term m in a document "i".

Three different levels are available for the basic unit of analysis. Polarity-based sentiment analysis typically includes document level and sentence level. Aspect levels are used in topic-based perspectives. At the document level, you can define the placement of each text as a whole and see if it expresses positive, neutral, or negative sentiment. At the sentence level, each document is split into sentences, the polarity of each sentence is obtained, and then merged at the document level to compute the overall alignment.

The positivity/negativeness of a document is evaluated by computing a polarity score of -1, 0, +1 for negative, neutral, and positive terms respectively. The polarity value of each term depends on the lexicon. A lexicon is a list of polarized terms.

Normalization

The process of normalization generally refers to converting words into a more uniform form. A base word can include structural variations such as suffixes with conjugations of verbs or plural forms of nouns. Textual analysis ignores these variations as they are closely and semantically related, reducing the feature space as some features are closely related. Stemming is a common technique used to achieve this uniform shape. It is a rule-based algorithm that converts inflectional forms of words to their base forms (stems). Removing a few common words that aren't very informative reduces the data size and improves computational power. Such words are pre-removed by matching against a predefined list of 'stop words' using 'tm' package found in R.

```
library(tm)
# Data cleaning

df.comments.corpus <- VCorpus(VectorSource(comments$textOriginal))
inspect(df.comments.corpus[1:2])

# Convert to lowercase
tm_map(df.comments.corpus, content_transformer(tolower))

# Remove stop-words
tm_map(df.comments.corpus, removeWords, stopwords("english"))

# specify your custom stopwords as a character vector
tm_map(df.comments.corpus, removeWords, c("can", "india", "get","linus","video"))

#Strip whitespace
```

```
tm_map(df.comments.corpus,content_transformer(stripWhitespace))

# Remove punctuation
tm_map(df.comments.corpus,content_transformer(removePunctuation))

# Text stemming - which reduces words to their root form
tm_map(df.comments.corpus, content_transformer(stemDocument))
```

Model Selection

Polarity and Sentiment classification

For the classification process, it is important to calculate and assign each document with a polarity score. The polarity detection might also be considered as a lexicon based approach to classification. A machine learning approach might me considered if we want to categorize document based on previous knowledge base derived with a training model. Some commonly used classifiers are:

- **Probabilistic classifiers:** Naïve Bayes classifiers or Bayesian networks.
- **Linear classifiers:** Support vector machine, Decision trees.

R packages for Sentiment Analysis

syuzhet

The Syuzhet package attempts to reveal the latent structure of the narrative by means of sentiment analysis. This package reveals the emotional switches that serve as proxies for the narrative movement between conflict and conflict resolution instead of detecting changes in the subject matter of the narrative. This allows for finding the polarity scores of a collection of documents by using different internal dictionaries. Syuzhet incorporates four sentiment lexicons: syuzhet, afinn, bing, and nrc. The main function used to calculate the sentiment scores is get_sentiment(). In this project, we will use all four lexicons and compare the results.

sentimentr

sentimentr is designed to quickly calculate text polarity sentiment in the English language at the sentence level and optionally aggregate by rows or grouping variables. This package attempts to take into account valence shifters while maintaining speed. Valence shifters can be of different types like a negator, amplifier, de-amplifier, or an adversative conjunction. A negator flips the sign of a polarized word like "It is not great.". An amplifier intensifies the impact of a polarized word "I really hate it.". A de-amplifier downtones the impact of a polarized word "She hardly likes it.". An adversative conjunction overrules the previous clause containing a polarized word "I like it but it's not worth it.".

quanteda

quanteda.textmodels is used to scaling models and classifiers for sparse matrix objects representing textual data in the form of a document-feature matrix. In this project, we used textmodels_NB supervised learning model. This is a Naive Bayes classifier for texts which fits a multinomial or Bernoulli Naive Bayes model, given a document-feature matrix and training labels. These training labels are taken from sentimentr results for this project.

Mathematical explaination

The augmented dictionary method of sentimentr can be understood by the equation given in this section. This approach gives better results than a simple lookup dictionary approach that does not consider valence shifters which was discussed in the earlier section. The equation used by the algorithm to assign polarity to each sentence first utilizes a sentiment dictionary to tag polarized words. The paragraph composed of "n" sentences can be given as: $p_i = (\{s_1, s_2, ..., s_n\})$ Each sentence (s_j) is broken into an ordered bag of words. $(s_{i,j} = \{w_1, w_2, ..., w_n\})$ where "w" are the words within each sentence, (s_{ij}) represents the jth sentence in an ith paragraph. Punctuation is removed with the exception of pause punctuations (commas, colons, semicolons) which are considered a word within the sentence. Let us denote pause words as $c**w$. We can represent these words as i,j,k notation as $w_{i,j,k}$. For example $w_{2,1,4}$ would be the fourth word of the first sentence of the second paragraph.

Each word $(w_{i,j,k})$ in the sentence is searched and compared to a syuzhet package dictionary of polarized words and dictionaries in the lexicon package. Positive $(w^+_{i,j,k})$ and negative $(w^-_{i,j,k})$ words are tagged with a +1 and 1 respectively, and 0 is tagged as neutral. Let us denote these polarized words as p**w. This will result in a polar cluster $(c_{i,j,l})$ which is a subset of a sentence $(c_{i,j,l} \subseteq s_{i,j})$.

The polarized context cluster $(c_{i,j,l})$ of words is pulled from around the polarized word (p**w) and defaults to "b" words before and "a" words after p**w to be considered as valence shifters. The n.before and n.after parameters are set by the user as n**b and n**a respectively. The cluster can be represented as $(c_{i,j,l} = \{p**w_{i,j,k} - n**b, .., p**w_{i,j,k}, .., p**w_{i,j,k} - n**a\})$ As the valence shifters are considered, the words in this polarized cluster are tagged as neutral $(w_{i,j,k0})$, negator$(w_{i,j,kn})$, amplifier [intensifier] $(w_{i,j,ka})$, or de-amplifier [downtoner] $(w_{i,j,kd})$. There is no affect of neutral words on the equation but it still is included in the frequency of word (n). According to the weights from polarity_dt argument, each p**w is assigned a weight. This is further weighted based on number of the valence shifters directly surrounding the positive or negative word (p**w). The lower bound of the polar cluster is constrained to $max\{p**w_{i,j,k}n**b, 1, max\{c**w_{i,j,k} < p**w_{i,j,k}\}\}$ and the upper bound is constrained to $min\{p**w_{i,j,k+n**a}, w_{i,j**n}, min\{c**w_{i,j,k} > p**w_{i,j,k}\}\}$ where $w_{i,j**n}$ is the number of words in the sentence.

As per the valence shifters, the polarity of the cluster is changed. The amplifier increases the value. If the cluster has an odd number of negators, the Amplifiers become deamplifiers. De-amplifiers work to reduce polarity. Negation acts on amplifiers/de-amplifiers but also flips the sign of the polarized word. Adversative conjunction reduces the value

placed on the prior clause making the next clause of greater value.

If the weight (z) is to be utilized with amplifiers/de-amplifiers, the weighted context clusters ($c_{i,j,l}$) are summed and divided by the square root of the word count $\sqrt{w_{i,j**n}}$ resulting an unbounded polarity score $\delta_{i,j}$ for each sentence.

By taking the average sentiment score of all sentences, we can find the sentiment of a paragraph. In this project, all these sentiment scores of paragraphs (as a comment) are averaged to get the mean score for a video. $p_{i,\delta_{i,j}} = \frac{1}{n} * \sum \delta_{i,j}$

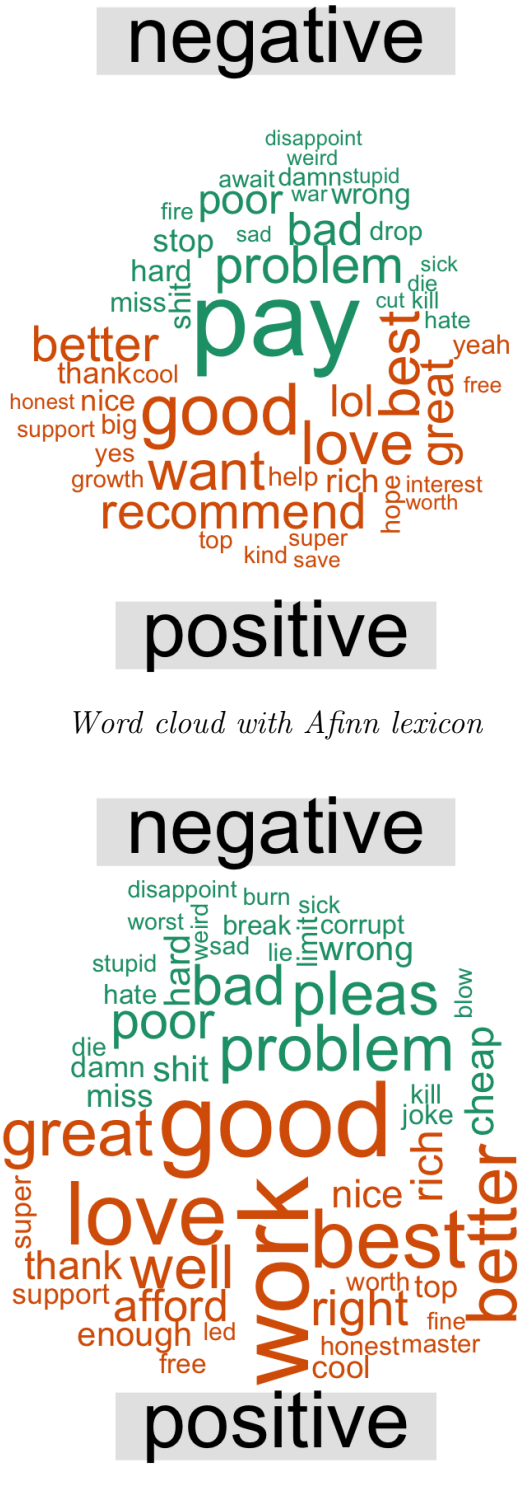
Implementation of Sentiment Analysis

Firstly, we plot the Document term matrix to get the frequency of words. This gives us count of distinct words in all the comments.

```
# Build a term-document matrix
TextDoc_dtm <- TermDocumentMatrix(df.comments.corpus)
dtm_m <- as.matrix(TextDoc_dtm)
# Sort by decreasing value of frequency
dtm_v <- sort(rowSums(dtm_m),decreasing=TRUE)
dtm_d <- data.frame(word = names(dtm_v),freq=dtm_v)
```

	word <chr>	freq <dbl>
peopl	peopl	1282
need	need	1272
student	student	1075
good	good	1054
make	make	1022
use	use	1017
work	work	982
love	love	861
tech	tech	834
much	much	833

We then use word cloud which is one of the most popular ways to visualize and analyze qualitative data. The size of each word indicates the number of occurrences of the word in comments. We generated a word cloud to be able to visualize the negative and positive sentiment words. The orange color represents positive sentiment and the green color represents negative sentiment. The following word cloud sentiment classification is done using bing and afinn lexicons. The figure shows that Good is the most used positive word and Bad, problem and pleas are the most used negative words. We clearly observe that both lexicons have different words classified as positive and negative.



Word cloud with Afinn lexicon

Word cloud with bing lexicon

Video sentiment

The overall sentiment of a video is analyzed by calculating the mean polarity of comments for each video. The below table shows the sentiment scores for each video in dataset.

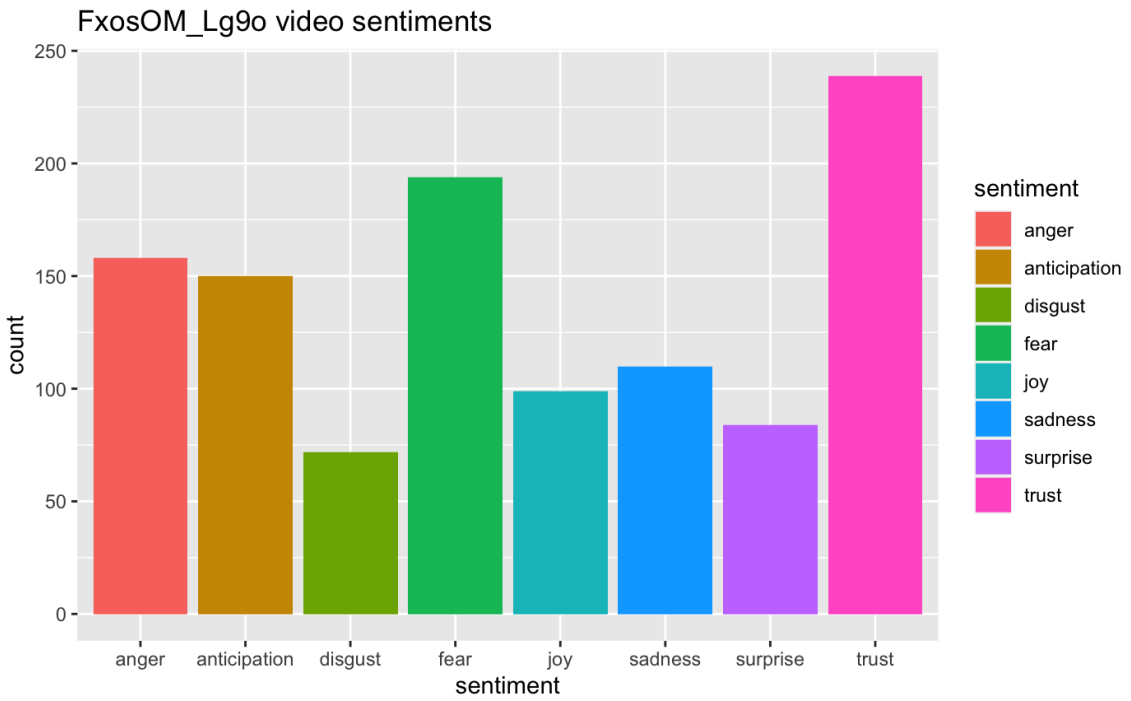
videoId <chr>	videoSentimentr <dbl>	videoSyuzhetSentiment <dbl>	videoBingSentiment <dbl>	videoAfinnSentiment <dbl>
18fwz9Itbvo	0.13421490	0.35465612	0.26475959	1.0596470
4mgePWWCAmA	0.06982809	0.20234662	0.20497597	0.5660164
b3x28s61q3c	0.07618748	0.22649945	0.14525627	0.6538713
ErMwWXQxHp0	0.20409775	0.63102410	0.74763339	1.8836059
FxosOM_Lg9o	-0.01101395	-0.03920530	-0.24105960	-0.2450331
kXiYSI7H2b0	0.01711570	0.03434191	0.01210287	0.1338880
W0QuOku3LRo	0.05838271	0.25737327	-0.08433180	0.1400922
wAZZ-UWGVHI	0.10478785	0.34442289	0.19592629	0.4820563

Polarity of videos

From the above results, we observe that all the methods have produced same sentiments for all videos except for one for which bing produced negative polarity where as others were positive. Let us now examine the emotions the comments have projected for the video with negative polarity score "FxosOM_Lg9o". We used get_nrc_sentiment() function from syuzhet package to get emotion classification.

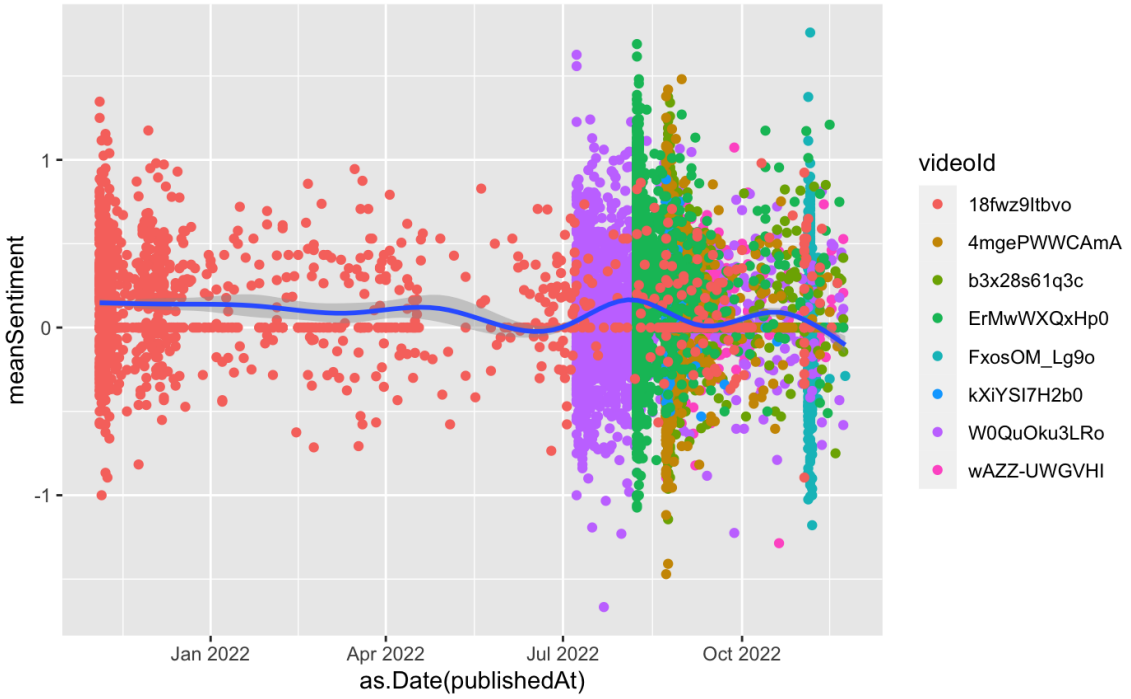
anger <dbl>	anticipation <dbl>	disgust <dbl>	fear <dbl>	joy <dbl>	sadness <dbl>	surprise <dbl>	trust <dbl>	negative <dbl>	positive <dbl>
0	1	0	1	0	0	0	0	1	0
1	1	0	3	1	1	1	1	4	3
4	1	2	2	2	1	1	2	3	3
0	0	0	0	0	0	0	0	0	0
2	1	2	5	1	1	3	1	5	2
1	0	0	1	0	0	0	0	1	0
2	0	0	3	0	0	0	1	3	0
0	3	0	2	0	1	1	2	3	0
0	0	1	0	0	0	0	1	1	1
1	0	1	1	0	0	1	0	1	0

Count of emotions posed by comments of video FxosOM_Lg9o



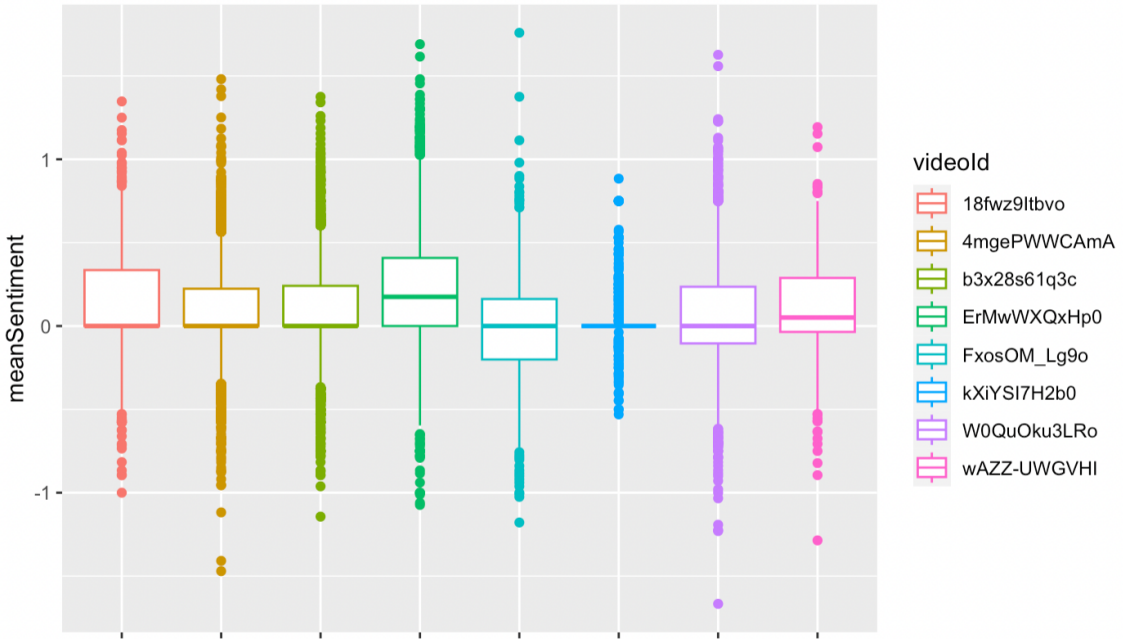
Ploting emotions posed by comments of video FxosOM_Lg9o

From the above graph, it is clear that the sum of negative emotions like anger, disgust, fear, and sadness is higher than the positive emotions. But it is observed that most of the comments showed trust.



Sentiments against comments published date

From above plot, it looks like there haven't been any strong trends over time in comments for specific video. W can observe that there were no many comments between Jan 2022 till July 2022 for the first video. The sentiments are equally spreadout. Let's look at individual videoIds and observe if there is any pattern:



Box plot for each videoId sentiments

From the above box plot, we observe that video "kXiYSI7H2b0" is having neutral sentiment and video "FxosOM_Lg9o" has a negative sentiment. All the remaining videos have a positive sentiment.

Comparision of R packages

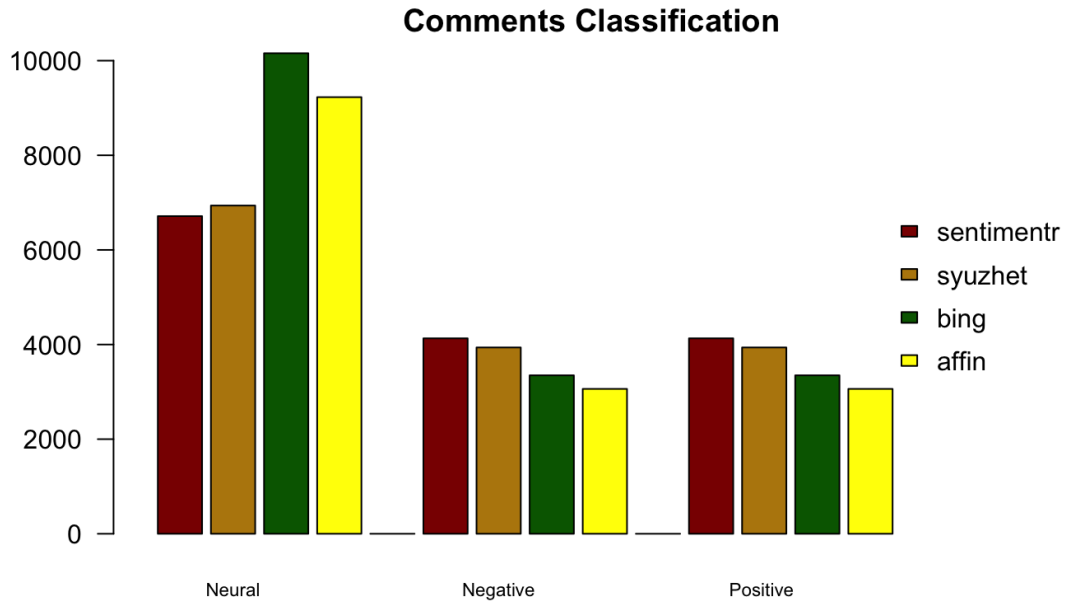
Syuzhet vectors comparision:

Using different methods in Syuzhet package, sentiment scores for all comments are calculated. The scores given by all these methods are in different scales. Below is the sample output for 6 comments.

syuzhet_vector <dbl>	bing_vector <int>	afinn_vector <int>	element_id <int>
0.80	0	1	1
1.40	0	1	2
2.60	1	3	3
1.55	1	3	4
-0.10	0	-1	5
-0.20	0	0	6

Different syuzhet methods

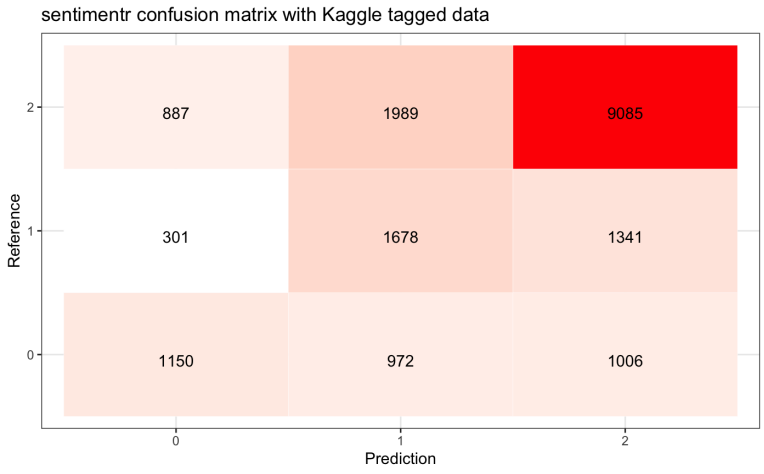
We plotted the side-by-side comparision of a number of comments classified as neutral, positive, and negative by 'sentimentr' and 'syuzhet' packages. The results show that most of the comments are classified as neutral in all methods.

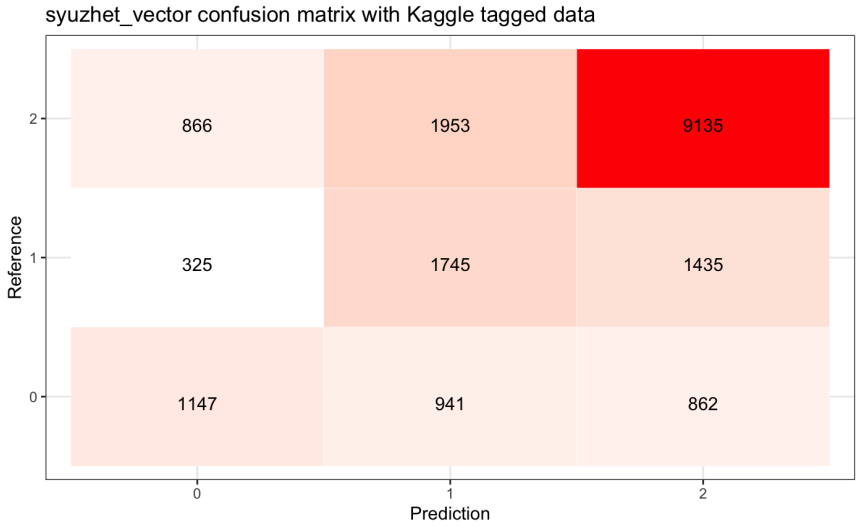


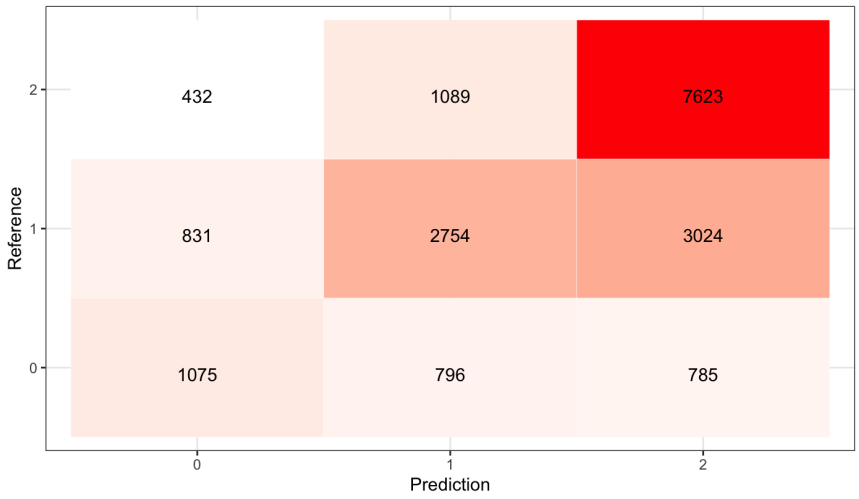
Comments classification for each R package used

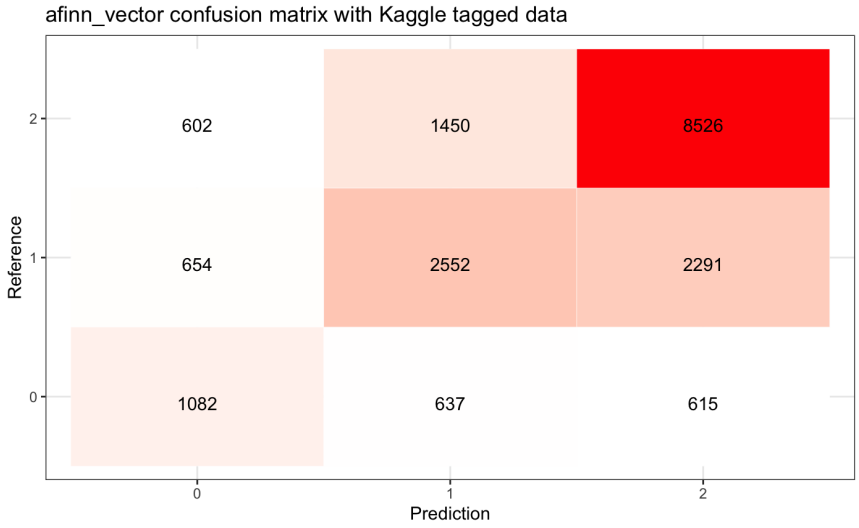
Results and Accuracy

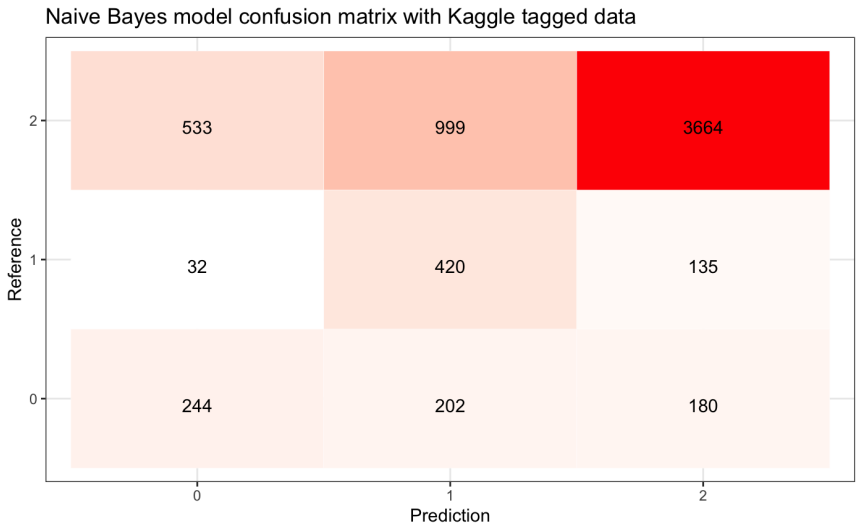
In the below plots, 0 represents negative sentiment, 1 represents neutral sentiment and 2 represents positive sentiment. The highest intensity of the red color is given for the highest matches. Hence we observe that bing and afinn show considerably better accuracy when compared with Kaggle data. 'quanteda' textmodels Naive Bayes model shows the highest accuracy in test data with 67 percent.




syuzhet_vector confusion matrix with Kaggle tagged data
866
1953
9135
325
1745
1435
1147
941
862
Reference
Prediction
0
1
2


bing_vector confusion matrix with Kaggle tagged data
432
1089
7623
831
2754
3024
1075
796
785
Reference
Prediction
0
1
2


afinn_vector confusion matrix with Kaggle tagged data
602
1450
8526
654
2552
2291
1082
637
615
Reference
Prediction
0
1
2



Confusion matrix for all methods.

Conclusion

Text and sentiment analysis with R is a fascinating and new research area having endless scope in data science. There are many other CRAN (Comprehensive R Archive Network) packages like 'meanr' and 'stanford' which are also great for text analysis but are out of scope for this project. In this project, the dimensionality of the data is reduced and text data is normalized to obtain a consistent form. With the help of derived emotion classifications, an analysis of video comments was successfully performed and documented. The maximum accuracy was reached with 'quanteda' textmodels naive Bayes model with 67 percent. All the other models got accuracies between 62 and 66 percent. Reducing noise in data by removing irrelevant words might help increase the model's performance. Since the observed Kaggle dataset is predominantly filled with positive sentiments in more significant numbers compared to negative and neutral, the model training could turn out to be biased. This project gives fundamental implementation steps for text analysis and emotion classification using R and can also be implemented on other texts like Twitter data.

References

- Naldi, Maurizio. "A review of sentiment computation methods with R packages." arXiv preprint arXiv:1901.08319 (2019).

- Wankhade, M., Rao, A.C.S., Kulkarni, C. A survey on sentiment analysis methods, applications, and challenges. Artif Intell Rev 55, 5731–5780 (2022). https://doi.org/10.1007/s10462-022-10144-1

- Brady D. Lund (2020) Assessing library topics using sentiment analysis in R: a discussion and code sample, Public Services Quarterly, 16:2, 112-123, DOI: 10.1080/15228959.2020.1731402

- Text mining with r - https://www.red-gate.com/simple-talk/databases/sql-server/bi-sql-server/text-mining-and-sentiment-analysis-with-r/

- Walaa Medhat, Ahmed Hassan, Hoda Korashy, Sentiment analysis algorithms and applications: A survey, Ain Shams Engineering Journal, Volume 5, Issue 4, 2014, Pages 1093-1113, ISSN 2090-4479, https://doi.org/10.1016/j.asej.2014.04.011.

- mjockers. "GitHub - Mjockers/Syuzhet: An R Package for the Extraction of Sentiment and Sentiment-Based Plot Arcs from Text." GitHub, April 12, 2021. https://github.com/mjockers/sy

- "GitHub - Trinker/Lexicon: A Data Package Containing Lexicons and Dictionaries for Text Analysis." GitHub, October 20, 2018. https://github.com/trinker/lexicon.

- Patwardhan, Soham. "Sentiment Analysis on YouTube Data Using R." Medium, May 30, 2019. https://medium.com/@sohamsp1995/sentiment-analysis-on-youtube-data-using-r-df4021193d1e.

- Youtube Statistics — Kaggle. "Youtube Statistics." /datasets/advaypatil/youtube-statistics.

- Tutorial: Sentiment Analysis in R — Kaggle. "Tutorial: Sentiment Analysis in R." https://www.kaggle.com/code/rtatman/tutorial-sentiment-analysis-in-r/notebook.

- Trinker, "Trinker/SENTIMENTR: Dictionary based sentiment analysis that considers valence shifters," GitHub. [Online]. Available: https://github.com/trinker/sentimentr.

- Google Cloud Platform. "Google Cloud Platform." https://console.cloud.google.com/apis/library?pli pwa-c9eb8.