

Data Science Capstone - Tweet Classssification System - Catastrophe or Not?

Carlos Yáñez Santibáñez

April 11, 2020

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 2 |
| 2 | Analysis | 3 |
| 2.1 | Processing and cleaning up the data | 4 |
| 2.2 | Scoring each tweet. | 9 |
| 2.3 | Finalising the model and data quality. | 13 |
| 2.4 | Tuning | 18 |
| 3 | Results | 26 |
| 4 | Conclusion | 31 |

1 Introduction

2 Analysis

As mentioned in the Introduction, this report presents the attempt to build a machine learning model to classify Tweets in a binary model. In this particular case, the question to answer is whether those tweets correspond to a catastrophic event or not. The idea for this was taken of a Kaggle Competition for Starters ([Real or Not? NLP with Disaster Tweets](#)). Instead of using the data provided in Kaggle, in this report the original version in [Data For Everyone](#) is used.

To start, let's have look at sample data:

Table 1: Sample Data

| id | target | text | keyword | location |
|-------|--------|--|-------------------|---------------------------|
| 1 | 1 | Just happened a terrible car crash | NA | NA |
| 6 | 1 | All residents asked to 'shelter in place' are being notified by officers. No other evacuation or shelter in place orders are expected | NA | NA |
| 33 | 0 | London is cool ;) | NA | NA |
| 49 | 1 | @bbcmtd Wholesale Markets ablaze http://t.co/lHYXEOHY6C | ablaze | Birmingham |
| 160 | 0 | 'The harder the conflict the more glorious the triumph.' Thomas Paine | aftershock | 304 |
| 914 | 1 | Uganda Seen as a Front Line in the Bioterrorism Fight | bioterrorism | Alaska |
| 3 999 | 0 | I forgot to bring chocolate with me. Major disaster. | disaster | Los Angeles, London, Kent |
| 7 336 | 1 | Finnish ministers: Fennovoima nuclear reactor will go ahead http://t.co/vB3VFm76ke #worldnews #news #breakingnews | nuclear%20reactor | World |

As seen above, the dataset contains the following observations (to match Kaggle's format):

- **id** : unique sequential identifier.
- **target** : indicator of whether the tweet corresponds to a catastrophic event (**1**) or not (**0**).
- **text** : tweet, as extracted from Twitter.
- **keyword** : search term used in Twitter's search bar to retrieve the tweets.
- **location** : Name of the place where the tweet in question originated.

For this report, **keyword** and **location** will be ignored, choosing to focus on the text (tweet) only. After a brief observation of its content, in this analysis the text will be divided into four components:

- The text proper. i.e. the *natural language* sentence in the text. In the code, this will be called **words**. All the tweets in this file are written in English language, but they may contain non-English characters, emojis and emoticons.

- The **hashtags**, which are in user created categorisation. This is a meaningful component of the tweet and may help clarify whether the message is a real disaster. For example, in below tweet, the hashtag helps us to elucidate this is cricket commentary rather than a fire disaster:

Table 2: Tweet about cricket

| id | text |
|-------|---|
| 1 674 | Australia's Ashes disaster - how the collapse unfolded at Trent Bridge cricket |

- The **links** contained in the tweet - perhaps the respective Internet domain could help distinguish “quality” disaster sources (e.g. official announcement from relevant authority, The Red Cross/Crescent) from non-disastarr ones (e.g. gossip magazine using hyperbolic language). To obtain meaningful content though, the links need to be “unshortened” from the t.co addresses provided by Twitter.
- Who is mentioned in the tweet (by their Twitter **handles**). Similarly to the previous points, it may be useful to distinguish between people reaching out to government/emergency service from *shock jocks* using florid language.

2.1 Processing and cleaning up the data

In order to obtain the data in the desired format for analysis, the function *prepare_data* has been created. Its code performs the below activities:

1. Extract hashtags, links and Twitter handles from the tweet.
2. Clean up the remaining text, removing links and handles, and performing transformation operations on emojis, emoticons and other special characters.
3. (Optional) Replace swear words and profane language with unique strings, in case seeing those wants to be avoided, yet retaining the words for analysis. Please note this report has been generated with the uncensored version, thus results may vary.
4. “Unshorten” link URLs to obtain domain names of linked sources.

The first step is similar for the three components - below is the code use to remove the hashtags:

```
source_data$hashtag <- str_extract_all(source_data$text, "#\\S+")
source_data <- source_data %>%
  mutate(hashtag = gsub(x=hashtag,
    pattern = "character\\(0)", replacement = "")) %>%
  mutate(hashtag = gsub(x=hashtag, pattern = "c\\(", replacement = "")) %>%
  mutate(hashtag = gsub(x=hashtag, pattern = "\\\"", replacement = "")) %>%
  mutate(hashtag = gsub(x=hashtag, pattern = "\"", replacement = "")) %>%
  mutate(hashtag = gsub(x=hashtag, pattern = ",", replacement = ""))
```

In the following step, both links and mentions are removed, keeping the hashtags since they may also be part of the sentence in question. Then, leveraging the functions in the **textclean** package, the remaining data is further normalised, by replacing :

- contractions with full words,
- emojis and emoticos with equivalent word (e.g. :) with 'smile'),
- all non-ASCII characters with equivalent,
- internet slang with full words,
- html code,
- money symbols,
- numbers with full words,
- ordinals with full words `replace_ordinal`,
- timestamps.

The code that achieves this is shown below:

```
#remove mentions and links from text, save original text in different observation

source_data$original_text <- source_data$text
source_data <- source_data %>%
  mutate( text = gsub(x = text, pattern = "#", replacement = "")) %>%
  mutate( text = gsub(x = text, pattern = "@\\S+", replacement = "")) %>%
  mutate(text = gsub(x=text,
                    pattern = "(s?)(f|ht)tp(s?)://\\S+\\b",
                    replacement = ""))

#further clean text with textclean package

source_data$text<-replace_contraction(source_data$text)
source_data$text<- replace_emoji(source_data$text)
source_data$text<- replace_emoticon(source_data$text)
source_data$text<-replace_non_ascii(source_data$text,impart.meaning=TRUE)
source_data$text<-replace_internet_slang(source_data$text)
source_data$text<-replace_html(source_data$text)
source_data$text<-replace_money(source_data$text)
source_data$text<-replace_number(source_data$text)
source_data$text<-replace_ordinal(source_data$text)
source_data$text<-replace_time(source_data$text)
```

An optional step is to remove swear and profane vocabulary from the text, in case that is desired. This has been done through the below code:

```
#profanity removal

if(profanity_clean==1){

  #download profane words and prep for matching
  data(profanity_zac_anger)
  data(profanity_alvarez)
  data(profanity_arr_bad)
  data(profanity_banned)
```

```

data(profanity_racist)
Sys.sleep(100)

special_chars <- as_tibble(c("\\!", "\\@", "\\#", "\\$", "\\&", "\\(", "\\)",
                             "\\-", "\\'", "\\.", "\\/", "\\+", "\\'", '\\\"'))
special_chars$replacement <- paste0("st",
                                     stri_rand_strings(nrow(special_chars),
                                                         3, '[a-zA-Z0-9]'))

profanity<-as_tibble(c(profanity_zac_anger,profanity_alvarez,
                      profanity_arr_bad,profanity_banned,
                      profanity_racist))
profanity<-unique(profanity)
profanity$replacement <- paste0("pr",
                                stri_rand_strings(nrow(profanity),
                                                    7, '[a-zA-Z0-9]'))

for(i in 1:nrow(special_chars)){
  profanity <- profanity %>%
    mutate(value=gsub(special_chars[i,]$value, special_chars[i,]$replacement,
                      value))
}
profanity <- profanity%>% mutate(value=paste0('\\b', value, '\\b'))

rm(profanity_zac_anger,profanity_alvarez,profanity_arr_bad,
   profanity_banned,profanity_racist)

#clean up profane words, replace by random string

for(i in 1:nrow(special_chars)){
  twitter_df <- twitter_df %>%
    mutate(text=gsub(special_chars[i,]$value,
                     special_chars[i,]$replacement,
                     text))
}

for(i in 1:nrow(profanity)){
  twitter_df <- twitter_df %>%
    mutate(text=gsub(profanity[i,]$value,
                     profanity[i,]$replacement,
                     text,ignore.case = TRUE))
}

twitter_df <- twitter_df %>%
  mutate(text=gsub("st[a-zA-Z0-9][a-zA-Z0-9][a-zA-Z0-9]",
                  "",text,ignore.case = TRUE))

```

```
rm(profanity,special_chars)
}
```

As a last step, the function *expand_urls* from the *longurl* package has been leveraged to unshorten the t.co URLs, to obtain the domains of the links. This is done through the below code:

```
split<-200
value <- round(nrow(processed_source)/split,0)
segments <- tibble(start=integer(),end=integer())
segments <- add_row(segments, start=1,end=value)

for(i in 2:(split-1)){
  segments <- add_row(segments, start=value*(i-1)+1,end=i*value)
}

segments <- add_row(segments, start=value*(split-1)+1,
                    end=nrow(processed_source))

# first segment
segment <- processed_source[segments[1,]$start:segments[1,]$end,]
domains<-get_domains(segment,anonimised = 0)
iteration <- 1

for(i in 179:split){
  segment <- processed_source[segments[i,]$start:segments[i,]$end,]
  domains_i<-get_domains(segment,anonimised = 0)
  domains <- rbind(domains,domains_i)
  iteration <-segments[i,]$start
}

domains <- domains %>%
  mutate(domain = gsub(x=expanded_url,
                      pattern = "(http|ftp|https)://",replacement = "")) %>%
  mutate(domain = gsub(x=domain,
                      pattern = "www.", replacement = "")) %>%
  mutate(domain = gsub(x=domain,
                      pattern = "ww[0-9].", replacement = "")) %>%
  mutate(domain = gsub(x = domain,
                      pattern = "/S+", replacement = ""))

domains$domain <- domain(domains$expanded_url)
domains <- domains %>%
  mutate(domain = gsub(x=domain, pattern = "www.", replacement = "")) %>%
  mutate(domain = gsub(x=domain, pattern = "ww[0-9].", replacement = "")) %>%
  mutate(domain = gsub(x = domain, pattern = "m.", replacement = ""))

domains_list <- domains %>% select(domain) %>% unique(.)
```

```
domains_list$domain_key <- paste0("domain_",seq.int(nrow(domains_list)))
domains <- domains %>% left_join(domains_list,by="domain")
```

As result of this processing, we have two data frames to use for further analysis: one with the processed tweets and another with a list of t.co URLs and domains. A sample of boths is shown below:

Table 3: Sample Data

| id | target | text | hashtag | link | mention |
|-------|--------|--|--------------------------------|------------------------|---------|
| 1 | 1 | Just happened a terrible car crash | NA | NA | NA |
| 6 | 1 | All residents asked to 'shelter in place' are being notified by officers. No other evacuation or shelter in place orders are excuse me tongue sticking out ected | NA | NA | NA |
| 33 | 0 | London is cool wink | NA | NA | NA |
| 49 | 1 | Wholesale Markets ablaze | NA | http://t.co/lHYXEOHY6C | @bbcmtd |
| 160 | 0 | 'The harder the conflict the more glorious the triumph.' Thomas Paine | NA | NA | NA |
| 914 | 1 | Uganda Seen as a Front Line in the Bioterrorism Fight | NA | NA | NA |
| 3 999 | 0 | I forgot to bring chocolate with me. Major disaster. | NA | NA | NA |
| 7 336 | 1 | Finnish ministers: Fennovoima nuclear reactor will go ahead worldnews news breakingnews | #worldnews #news #breakingnews | http://t.co/vB3VFm76ke | NA |

Table 4: Sample Data

| link | domain | domain_key |
|-------------------------|-------------------|------------|
| https://t.co/irWqCEZWEU | bbc.co.uk | domain_1 |
| https://t.co/lHYXEOHY6C | twitter.com | domain_2 |
| https://t.co/pmlOhZuRWR | sigalert.com | domain_17 |
| https://t.co/GKYe6gjTk5 | lawsociety.org.uk | domain_18 |
| https://t.co/FaXDzI90dY | change.org | domain_61 |
| https://t.co/SB5R7ShcCJ | change.org | domain_61 |
| https://t.co/ZNTg2wndmJ | twitter.com | domain_2 |
| https://t.co/JLRw0Oi9ee | twitter.com | domain_2 |

Both datasets in their “uncensored” and “sanitised” versions have been made available on [GitHub](#).

Before proceeding, we will split this data into three datasets:

- a **training** dataset, to be used for further analysis, modelling and tuning.
- a **testing** dataset, to be used in tuning.
- a **validation** dataset, which will put aside and only use at the end to generate the last results.

Unless said otherwise all the below analysis has been done with the **training** dataset only!

2.2 Scoring each tweet.

In order to use a machine learning method that allows us to determine whether any relationship between the content of each tweet and its target status, we need to be able to generate a function that rates each piece of content. A way of do this is to generate a numeric value for each tweet, that then can be used for estimate whether it is “catastrophic” or not.

In the case with text, a simple way to achieve this is to assign each constituent word a score and then use those to calculate a sentence value (with a simple way of calculating such value being adding to individual scores). The following lines explain the way this was done in this particular exercise - please note that as a starting point, we are making them assumption this is a good method - how to determine the right formula is a problem by itself.

Taking the each tweets processed text, the first step is to “tokenise” this dataset, i.e. split it in its component words - for this the function `unnest_tokens` from the *tidytext* package has been used. Since we would like to know how this words are split between catastrophic (“positive”) and non-catastrophic (“negative”) entries and perhaps use this a scoring base, we will also remove “stop words” (common words such as “and” “or”, basic verbs,etc), we will filter them using the `stop_words` dataset available on *tidytext*.

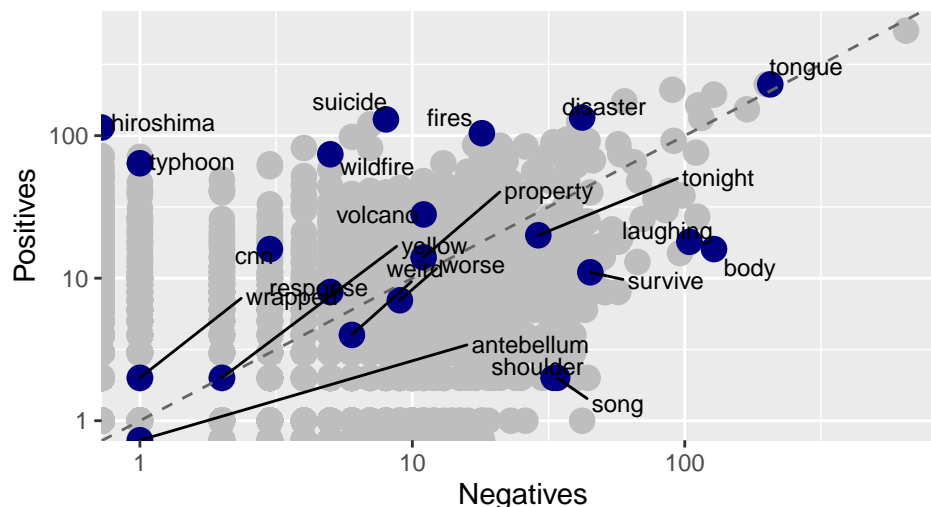
The code to achieve this goes as follows:

```
data("stop_words")
extra_stop_words$lexicon <- "EXTRA"
stop_words<-rbind(stop_words,extra_stop_words)
rm(extra_stop_words)

tokenised_words <- training_dataset %>% unnest_tokens(word,text)
tokenised_words <- output$tokenised_words %>%
  anti_join(stop_words, by="word")
```

With the data, we can calculate how frequent each word is, in general and in both positive and negative tweets. The chart below shows this, highlighting some selected words for analysis:

Pos. and neg. ccurrences for each word (log scale)

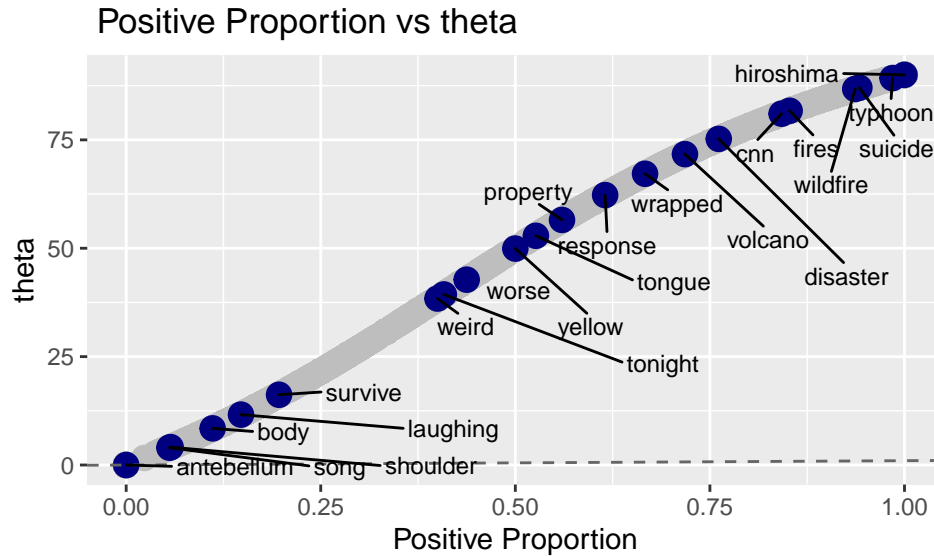


Since we are looking at the impact of the word for a tweet being positive, the above chart can be reduced to one indicator, the proportion each words appears positive tweets against its total number of appearances.

From here, the question is how to take this into a score for each word that:

- Properly weighs “disaster” related words.
- Penalises “non disaster” words but without reducing the impact of more ambivalent terms.
- Allows to establish a clear division, so when compoundend it clearly helps to distinguish positives from negatives.

After thinking how to achieve this objective, a score was defined taken angle of each point in the avoid chart, being this an effective measure of how “positive” or “negative” a word is. If we call this angle θ , we can generate an initial score that looks like the below chart:



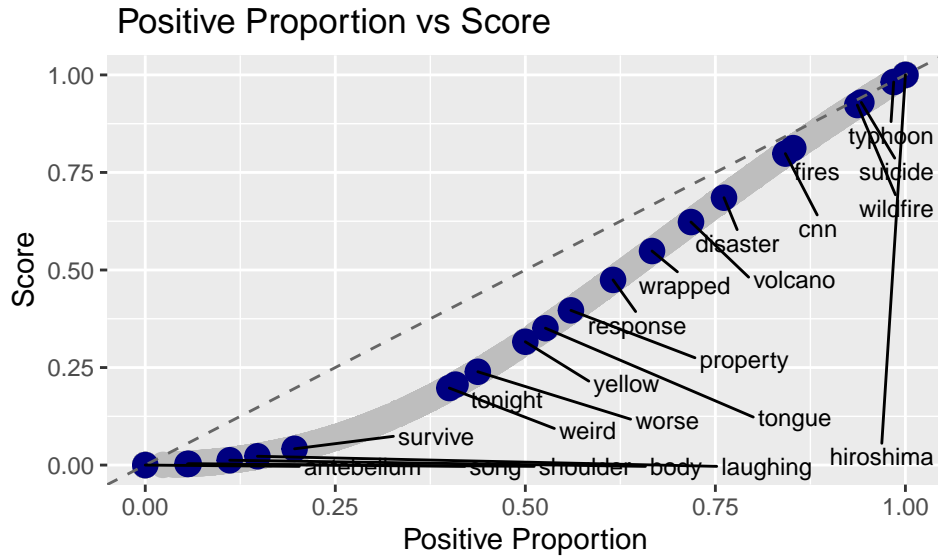
Although this is a good approach, after several iterations the following formula has been chosen instead:

$$pre_score = \sin(pos_proportion)^{\lambda_1} * \frac{\theta}{90^\circ}$$

$$score = \frac{pre_score}{\max(pre_score)} - \lambda_2$$

where * $pos_proportion$: Proportion of times where word was found in a “positive” tweet. This sinus of this parameter is used to help penalise less relevant words. * λ_1 : Parametrisation parameter, used to modulate the effect of less relevant words. * θ : angle of vector $(positive_proportion, negative_proportion)$, as discussed previously. * λ_2 : Offset to potentially give a negative value to less relevant words, in order to provide compensation - e.g. in case a “disaster” word is used in figurative context amongst many non relevant words. This will be treated as a tuning parameter.

Setting λ_1 to 1 and λ_2 , we have a sample distribution, as follows:

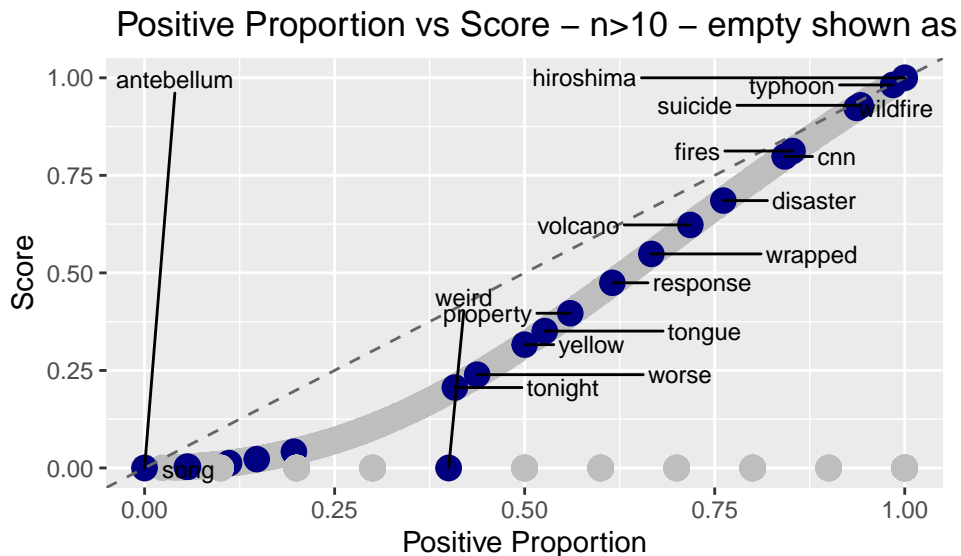


Looking at the chart above, we have achieved the objectives of giving distinct scores depending on the apparent “catastrophe” value of each word. This however, does not reflect the amount of times a word appears - it may be the case that a particular term appears just one and gets an extreme score which is not really a good indicator of its real "catastrophe value". Thus, a last modification of the formula is required:

$$score = \begin{cases} \emptyset & n \leq \lambda_3 \\ \frac{pre_score}{\max(pre_score)} - \lambda_2 & n > \lambda_3 \end{cases}$$

where * n : number of times the word is appears in the training set, * λ_3 : arbitrary threshold to filter word with a low number of occurrences (and possibly biased). This is a tuning parameter.

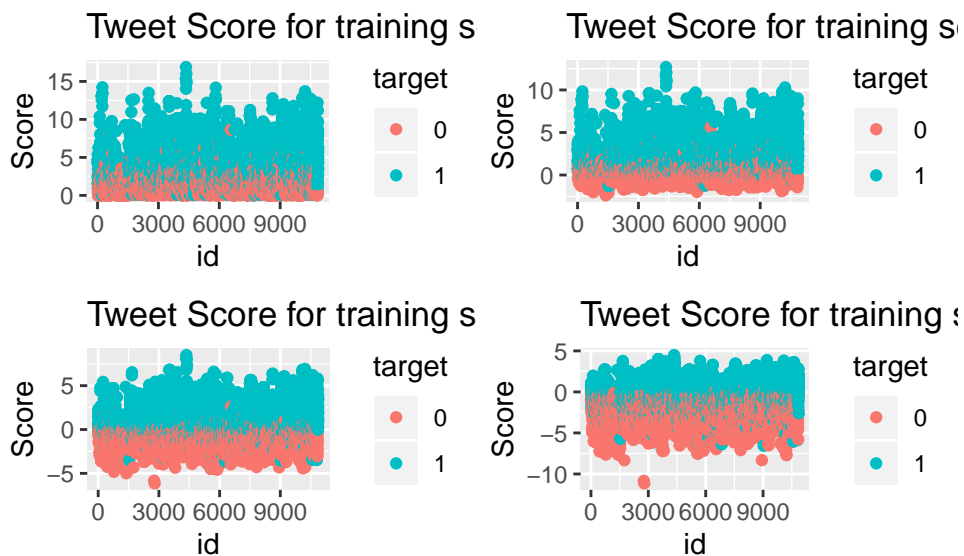
Using $\lambda_3 = 10$ for illustration purposes, we have the below chart:



With a set of scores, we would like to know if this produce a set of tweet scores that are a potential good input for a regression function. Due its simplicity, if we using a sum to calculate a score per tweet we can obtain those values with the below code:

```
tweet_score <- tokenised_words %>%
  left_join(word_scores,by="word") %>% filter(!is.na(score)) %>%
  group_by(id) %>% summarise(word_score=sum(score)) %>%
  ungroup()
```

which, will produce results like the below graphs (for different offset values):



From the above, we can see that there is a diffuse division between positive and negative tweets. It is worth noticing this could potentially be improved with better modelling - however this could be a problem but itself and the above - with tuning - is deemed acceptable for a first approximation.

A similar process has been done to create tweet scores for hashtags, links (or rather the domains of the unshortened URLs) and mentions. After try and error, a slightly different formula has been chosen for these three observations, represented below for hashtags:

$$pre_score = \left(\frac{pos_proportion}{\lambda_1} \right)^{\lambda_2} \cdot \frac{\theta}{90}$$

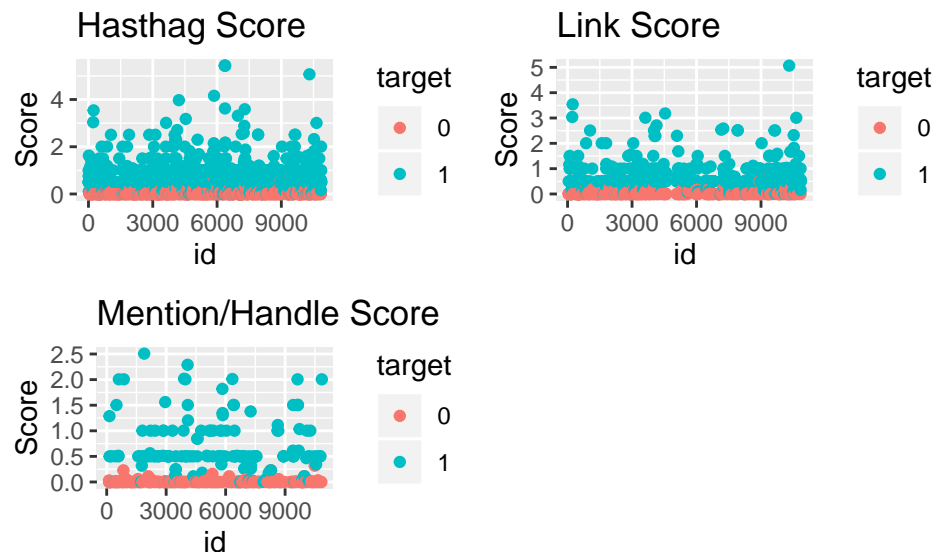
$$score = \begin{cases} 0 & n \leq \lambda_4 \\ \frac{pre_score}{max(pre_score)} - \lambda_3 & n > \lambda_4 \end{cases}$$

where

- *pos_proportion* is the proportion of the term in the entire corpus of positive hashtags/domains/mentions,
- λ_1 tuning parameter that re-scales - everything above the threshold is relevant, the rest not so much,
- λ_2 tuning parameter to weigh down irrelevant terms and weigh up relevant ones,

- θ angle of vector (*negatives, positives*),
- λ_3 , Offset to potentially give a negative value to less relevant term, in order to provide compensation - e.g. in case a “disaster” word is used in figurative context amongst many non relevant term This will be treated as a tuning parameter.
- λ_4 , threshold of number of occurrences necessary to consider the score as valid.

The below charts show an example of score generated using this method:



In order to facilitate the next steps, the above process has been written into three functions, namely:

- **tokenise_data**, which takes the data in the pre-processed format and generates a dataframe with the token words, hashtags, domains and Twitter handles.
- **calculate_scores**, which takes the output of the previous function (*tokenised* data) and calculate the scores for each word, hashtag, link and handle. All the previously discussed tuning parameter have been defined as inputs for this function.
- **score_tweets**, which will generate a score for each tweet for each of categories previously mentioned.

All these functions are written in the accompanying .R file, with inline commentary explaining all the steps involved in the process.

2.3 Finalising the model and data quality.

From here, we can move to the next step and to generate the last part of the model - the one that will render a prediction.

Considering the work done so far, there are potentially many ways to use the scores to create a model for prediction. After weighing the options, the following two-step approach will be used:

1. First, a regression algorithm will be used to train a model for each attribute - i.e. 4 different predictions will be generated using each of the 4 scores previously created independently. The result of this process will be vector for each tweet, with the results of each prediction (1,0 or potentially N/A if there is no such attribute for a given tweet.)
2. Then this vector will be used again as a input of a second machine learning method , which will attempt to predict the target result based on the individual responses.

For this process, two functions have been written in this report .R file:

- **fit_model**, which is the training function. Its inputs are the scores plus relevant tuning parameters. It will output 5 machine learning models (for each attribute plus the aggregate one).
- **predict_values**, which will take the training score, the test data and the models previously generated. it will output the predictions and measure accuracy if the target value is provided.

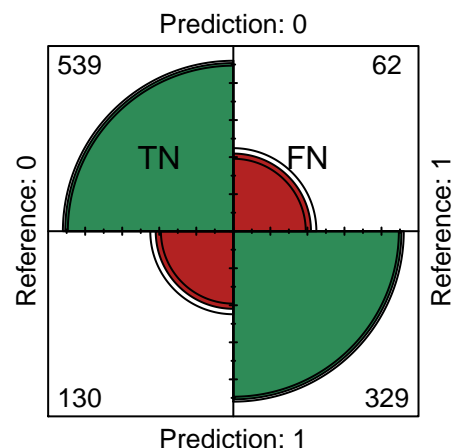
These functions leverage the caret package and they have been written to provide some flexibility - for example **fit_model** allows to be run using different machine learning models. For further details, please refer to the detailed commentary in the .R file.

For the initial run - that will help us establish and initial benchmark - the general linear model (glm) in caret is used for the individual predictions. For the aggregate prediction, we will use [xgbBoost](#). For this the below code is used:

The accuracy for table and confusion matrix for this example are:

Table 5: Initial results

| Method | Accuracy |
|-----------|-----------|
| word | 0.8183538 |
| hashtag | 0.8148148 |
| link | 0.8108108 |
| handle | 0.8095238 |
| aggregate | 0.8188679 |



integer(0)

As observed, there are large number of false positives and false negatives. A sample of those is provided below:

Table 6: Initial results - False Positives

| id | target | original_text |
|--------|--------|--|
| 269 | 0 | EMS1: NY EMTs petition for \$17 per hour â€˜minimum wageâ€™™ http://t.co/4oa6SWlxmR #ems #paramedics #ambulance |
| 1 026 | 0 | Come and join us Tomorrow! August 7 2015 at Transcend:Blazing the Trail to the Diversified World of Marketing... http://t.co/Fd5od5PfCF |
| 3 581 | 0 | @mtnredneck It does when you are searching on a desolate part of the beach with no cell service and a mile away from your car. |
| 9 779 | 0 | #entertainment Hollywood movie about trapped miners released in Chile: SANTIAGO Chile (AP) â€˜” The Hollyw... http://t.co/C22ecVl4Hw #news |
| 10 380 | 0 | @Glosblue66 no idea what this means. Look at our violent crime rate without weapons. Ban guns we become like Mexico not Australia |
| 10 402 | 0 | NEW YORK: A whirlwind day of activities in New York. Breakfast at the Millennium Hotel United Nations Plaza. Lunch... http://t.co/laYZBA9y8h |

Table 7: Initial results - False Negatives

| id | target | original_text |
|-------|--------|--|
| 3 | 1 | Heard about #earthquake is different cities, stay safe everyone. |
| 253 | 1 | Anyone travelling Aberystwyth-Shrewsbury right now there's been an incident. Services at a halt just outside Shrews. Ambulance on scene. |
| 1 172 | 1 | #Detroit has made progress against blight but too many burned out shells of houses remain. |
| 1 968 | 1 | You picture buildings burning to the ground from the basement to the streetlight. I'm not your drinking problem a hole is in the sky. |
| 6 571 | 1 | @Sport_EN Just being linked to Arsenal causes injury. |

A more detailed review of both sets, seems to indicated that the source file curation does seem to contain some errors : tweets that are seemingly not a catastrophe are tagged as such and vicerversa. One particular example of this are tweets allusive to the Hiroshima bombing in World

War 2. Although those tweets a tragedy remembrance and do not carry the connotation of a current catastrophe, there are marked as “disaster” in the dataset.

Table 8: Initial results - Hiroshima Tweets

| id | target | original_text |
|-------|--------|---|
| 1 535 | 1 | Hiroshima prepares to remember the day the bomb dropped http://t.co/oJHCGZXLSt |
| 1 626 | 1 | 70 years ago today the United States of America bombed Hiroshima in Japan. |
| 1 642 | 1 | Just Happened in Asia: Japan marks 70th anniversary of Hiroshima atomic bombing http://t.co/E5dEmdScEh |
| 1 652 | 1 | 70th Anniversary of Hiroshima bombing https://t.co/juwvomhGPd |
| 3 995 | 1 | I visited Hiroshima in 2006. It is an incredible place. This model shows devastation of the bomb. http://t.co/Gid6jqN8UG |
| 5 032 | 1 | How 'Little Boy' Affected the People In Hiroshima - Eyewitness Testimonials! - https://t.co/5x5hSV5sKO |
| 5 041 | 1 | On anniversary of Hiroshima bombing illustrated timeline of bombings. Eyewitness account particularly horrifying http://t.co/GZIb0mAwmn |
| 5 523 | 1 | 70 years ago the first atomic attack flattened #Hiroshima 3 days later it was #Nagasaki both war crimes to put Moscow in its place |
| 7 129 | 1 | Courageous and honest analysis of need to use Atomic Bomb in 1945. #Hiroshima70 Japanese military refused surrender. https://t.co/VhmtYtptGR |
| 7 441 | 1 | The day Hiroshima through the eyes of the bomber crew and survivors http://t.co/AYEWJ8marn via @MailOnline |
| 7 490 | 1 | Here we are 70 years after the nuclear obliteration of Hiroshima and Nagasaki and Iâ€™m wondering iâ€™_ http://t.co/fvkekft4Rs |
| 7 520 | 1 | Hannah: 'Hiroshima sounds like it could be a place in China. Isn't that where the oil spill was?' |
| 9 397 | 1 | Hiroshima survivors fight nuclear industry in Brazil Ã¢?? video http://t.co/GLZmGBM7w0 |
| 9 442 | 1 | Remembrance http://t.co/ii4EwE1QIr #Hiroshima http://t.co/H3vUsqzyQo |

The above reflects two important problems with machine learning and natural language:

1. This re-enforces that quality training data is essential. Bad data will probably result in bad results - even with a very sophisticated algorithm (It's called *Data Science* after all!).
2. Human language is complicated. Meaning of each word is highly contextual and changes depending on the register, dialect and over time. This can be difficult for a machine learning model to **grasp**.

To illustrate the second point, the above false positives and false negatives have been manually review and their new positive/negative result has been modified accordingly and added into the

original data set. After feeding them back into the training dataset, selecting a new testing sample and running the machine learning model again, the below results are obtained:

Table 9: Initial results - corrected dataset

| Component | Accuracy |
|-----------|-----------|
| word | 0.7691580 |
| hashtag | 0.7968750 |
| link | 0.7101449 |
| handle | 0.7500000 |
| aggregate | 0.7688679 |

This illustrates that better training data can lead to better results. It is also worth noticing this may well reflect a real-life scenario, where a social media monitoring centre is constantly reviewing false negatives and false positives with the purpose of keep re-training a model to adapt to changes language, emerging events, etc.

Of course, the other components in the model can also benefit of this *curated* approach. In this report, and for illustration process - the list of hashtags in the training set has been reviewed and then detected a couple of hashtags that most probably mean catastrophe (e.g. like #earthquake most like means one) and other most probably won't (like #ashes2017, which is unmistakably cricket, not matter how colourful and hiperbolic is language).

Using a sample set of curated hashtag, it is possible to observe a small increment in accuracy. Handles and links will be left as a action to improve for the time being.

Table 10: Initial results - Curated Hashtags

| Component | Accuracy |
|-----------|-----------|
| word | 0.7738884 |
| hashtag | 0.8181818 |
| link | 0.7101449 |
| handle | 0.7500000 |
| aggregate | 0.7735849 |

Looking at the work done so far, we have the below table showing the improvements obtained by improving and supplementing the data:

Table 11: Initial Comparison

| attempt | accuracy |
|------------------------------------|-----------|
| Initial training | 0.7571429 |
| Initial training - Data Correction | 0.7688679 |
| Curated hashtags | 0.7735849 |

2.4 Tuning

With this new, *augmented* dataset the next step of the process is to tune the model. Looking at how the model it's constructed, there are *many* aspects to tune, namely:

- The parameters used to calculate the tweet, hasthag, link and mention scores.
- The selection of classification algorithms used.
- The tuning parameters of each selection algorithm.

Obviously, tuning every single parameter will result in quite a lengthy exercise. Instead, this report will focus on some of the key components in this model and discuss what works and what else could be done.

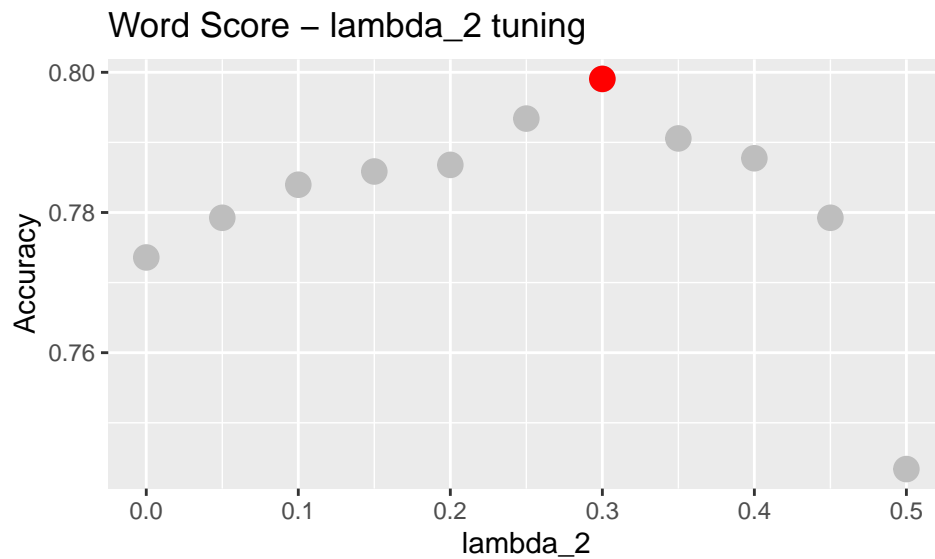
The first aspect to tune is the each word in the training corpus gets a individual score. This is perhaps the key component of the model. From the previous section, we have that this score is calculated through the below formulas:

$$pre_score = \sin(pos_proportion)^{\lambda_1} * \frac{\theta}{90^\circ}$$

$$score = \begin{cases} \emptyset & n \leq \lambda_3 \\ \frac{pre_score}{max(pre_score)} - \lambda_2 & n > \lambda_3 \end{cases}$$

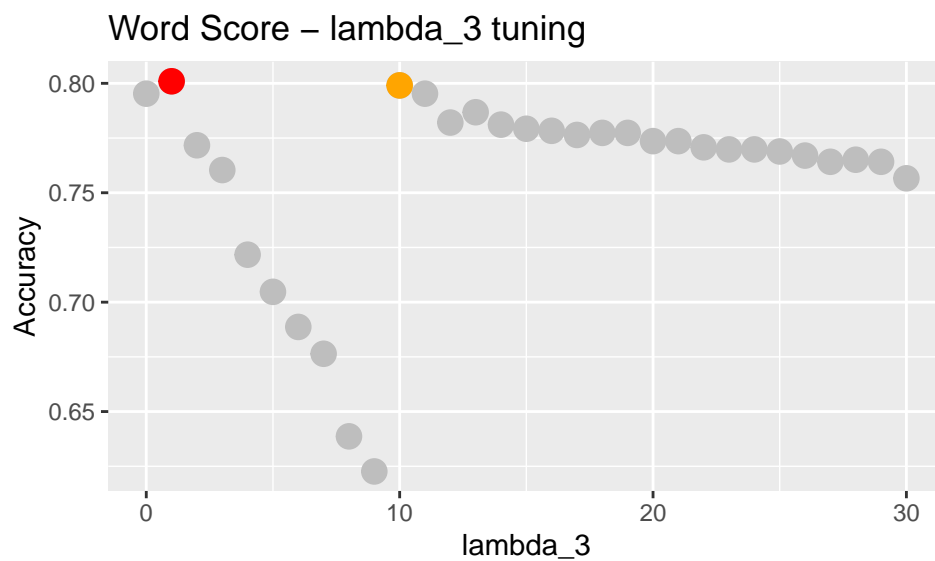
where * *pos_proportion*: Proportion of times where word was found in a “positive” tweet. This sinus of this parameter is used to help penalise less relevant words. * λ_1 : Parametrisation parameter, used to modulate the effect of less relevant words. * θ : angle of vector (*positive_proportion, negative_proportion*), as discussed previously. * λ_2 : Offset to potentially give a negative value to less relevant words, in order to provide compensation - e.g. in case a “disaster” word is used in figurative context amongst many non relevant words. This will be treated as a tuning parameter. * *n* : number of times the word is appears in the training set, * λ_3 : arbitrary threshold to filter word with a low number of occurrences (and possibly biased). This is a tuning parameter.

In this report, λ_2 and λ_3 will be tuned, to see how the penalisation of irrelevant words and minimum number of occurrences affect the model. The results of tuning `lambda_2` are shown below.



As shown above, the best result is obtained with $\lambda_2=0.3$. It is also worth noticing that increasing this value won't necessarily result in better results, since it will then penalise “neutral” terms too.

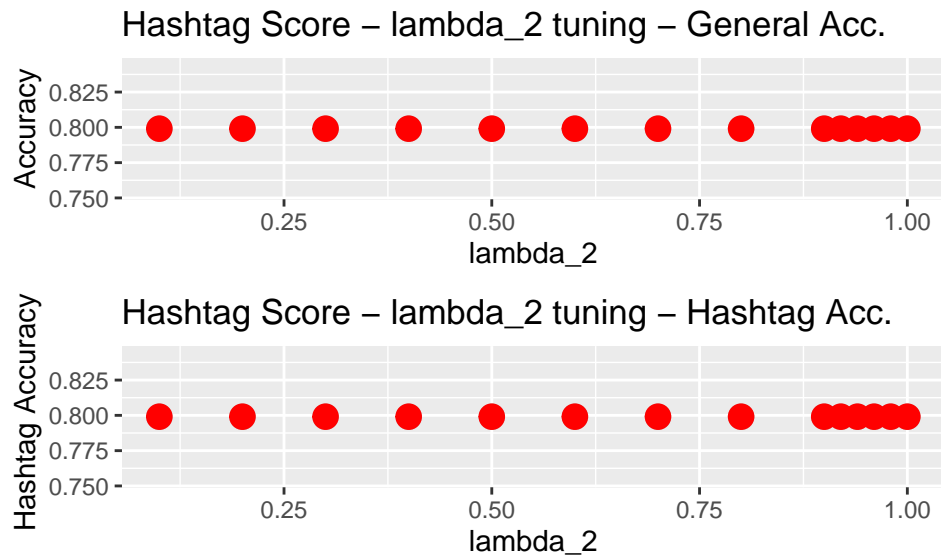
With this value, we will then also find if there is an optimal λ_3 , i.e. to see if removing the words that appear seldomly have an negative impact in the model. The results are shown below:



This result is a bit more interesting. As observed above, the first optimal λ_3 is quite a low value, however there is a relative bigger decrease in accuracy around its neighbours when compared against the second best λ_3 . It is important to notice that the accuracy depends on the corpus of words in the test set and how they are combined into sentences. This is of course random and it will change from set to set. Even though we can do cross validation, perhaps it is not a bad idea (as an initial step), choose a perhaps sub-optimal value for the sake of consistency - at least until more thorough tuning is possible. Therefore, to continue with this exercise, a value of $\lambda_3=10$ has been chosen.

After the word scoring, perhaps the most relevant component in the model is the scoring function for hashtags (given that hashtag intend to be **meaningful** descriptors). Similarly to the word scoring, we will tune the equivalent parameters hashtag's λ_2 and λ_3 .

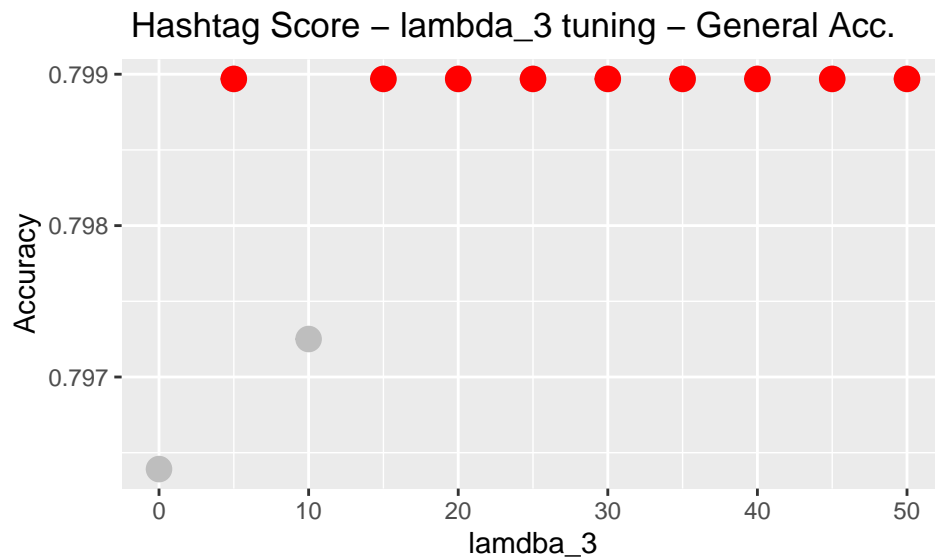
The results for the tuning of λ_2 are shown below:



The first graph shows the overall accuracy results (comparable to the word tuning), while the second graph shows the results where there is a hashtag. From them, the following points can be made:

- Smaller variations in the overall score can be explained to the fact that not all tweets have hashtags, thus the effect of tuning this part of the model will obviously have a limited effect on the general result.
- Taking the above point, it looks like its contribution to general score it is mostly when hashtags do indicate a catastrophe (and therefore have a bigger score). It is worth noticing that many of the high-score hashtag were not ranked by the algorithm but manually added - which is perhaps an indication that manual curation is not necessarily a bad idea - especially when data is poor quality or insufficient.

With this mind, the tuning of λ_3 will help to determine the overall effect of hashtag occurrence. The results are shown in the below chart.



This result seems similar in nature to the tuning for quantity cut-off for the word scoring.

Looking at the results so far (table below), this tuning process has only resulted in small gains when compared with the initial data corrections. Taking that the effect of hashtag tuning is small, in this report the tuning of the links and handle scoring functions will not be conducted. This doesn't mean they may bring an improved accuracy but perhaps are not worth the effort given the existing training dataset.

Table 12: Initial Comparison

| attempt | accuracy |
|------------------------------------|-----------|
| Initial training | 0.7571429 |
| Initial training - Data Correction | 0.7688679 |
| Curated hashtags | 0.7735849 |
| Optimised word score - lambda_2 | 0.7990566 |
| Optimised word score - lambda_3 | 0.7990566 |
| Optimised hashtag score - lambda_3 | 0.7989691 |

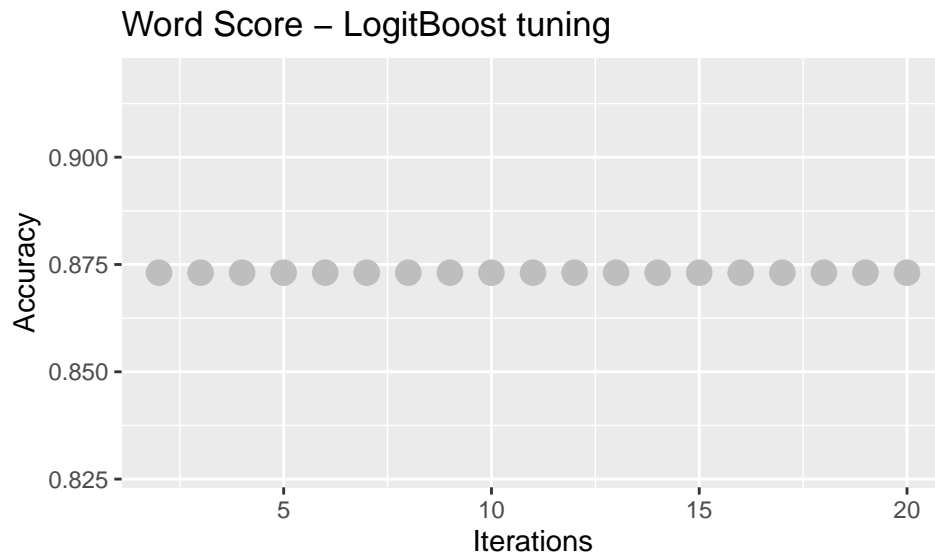
Instead, the decision is to focus on the tuning of the classification functions. At this point, the model uses the caret package to leverage to known machine learning algorithms, namely:

- *glm* (as in generalised linear model) in its logisitic regression mode to turn word, hashtag, link and mention to make a prediction on those components independently.
- *xgbTree* (extreme gradient boosting) to predict an “overall” positive or negative based on the previous predictions.

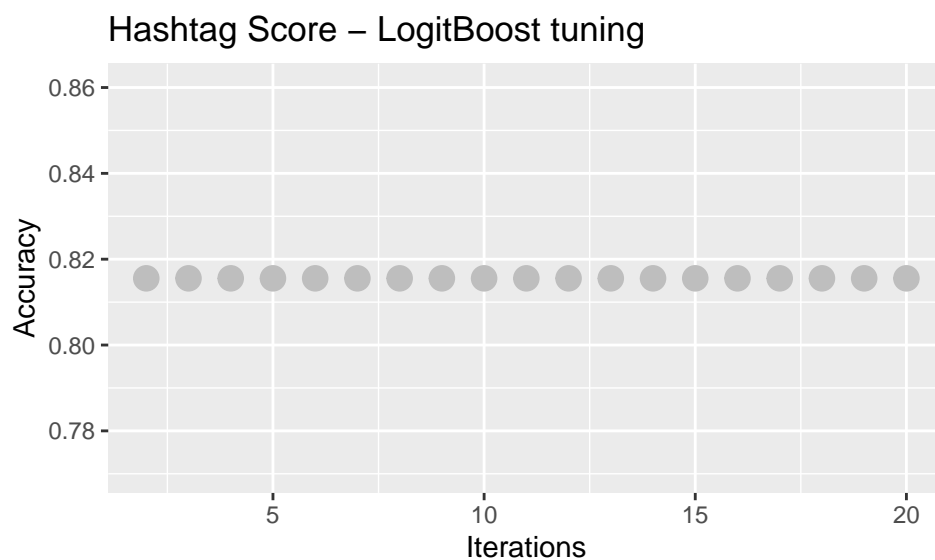
Considering that the assignment instructions state **“For this project, you will be applying machine learning techniques that go beyond standard linear regression”**, the next stage will be to asses if there is any model that produces better results than glm. (Although so far this is not a **standard** linear regression model anymore.)

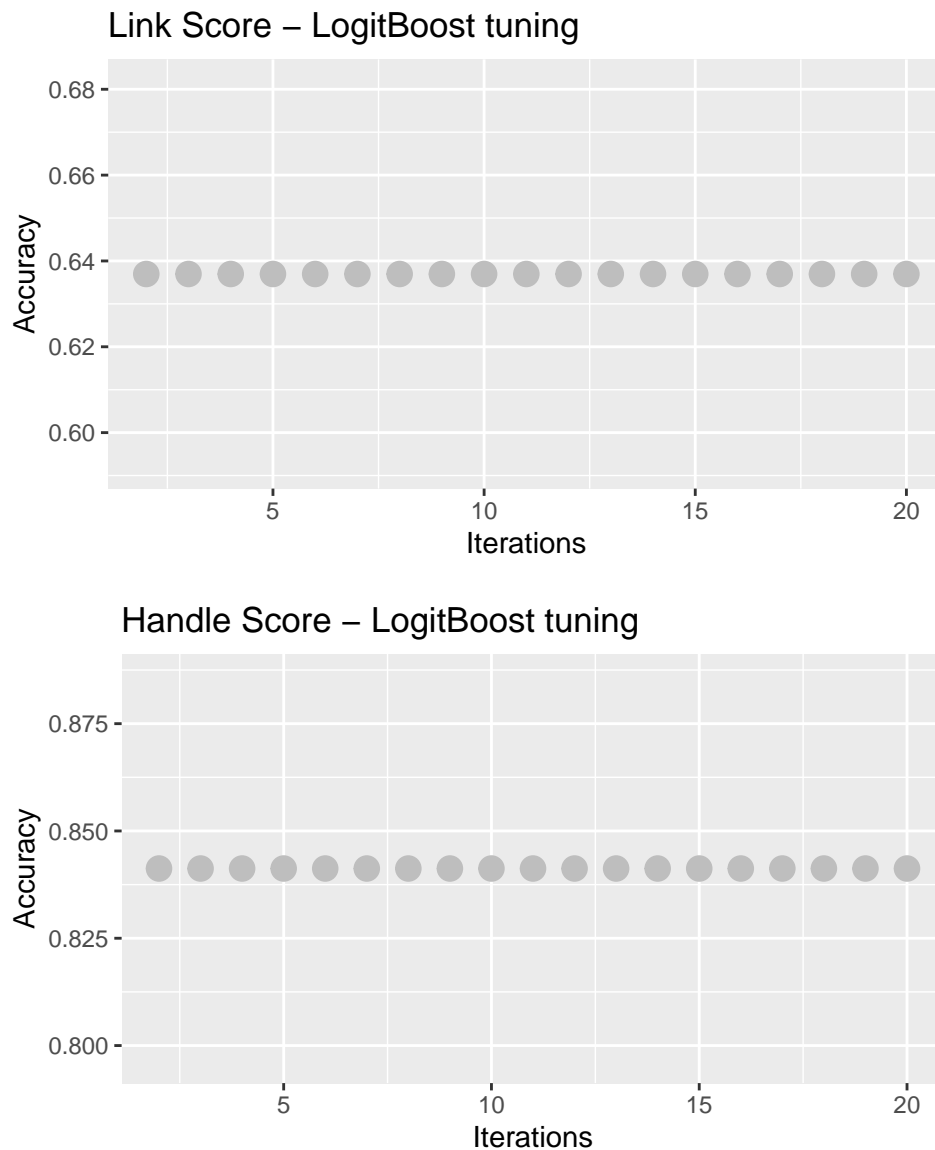
After investigating a number of possible algorithms, the **LogitBoost** will use as comparison. This algorithm is available in R through the **caTools** package.

As a starting point, we will try this algorithm for the word score classification function. The below results show the best result after tuning its only parameter, which is the number of iterations.



Please note graph above shows the accuracy results for the LogitBoost regression only, not for the entire model. However, it shows that in this case the tuning has not effect. In this case, the overall accuracy equals *0.795*. These result is no better than the one using the a simple logistic regression. Similar results are obtained when the same comparison is repeated for the other scoring components.





In this case, the accuracy value equals **0.795**, which is no better than the classification using a logistic regression.

This results shouldn't be confusing - a more "sophisticated" method does not necessarily mean a better result the important thing is to find the model that better represents the problem at hand (when possible). Simpler models may also have an advantage in terms of processing time - which may be a key factor to consider even if other models perform marginally better.

Completed the above steps, there is one aspect left to tune : the XGBoost algorithm. This method has several parameters, for which a good explanation of them can be found [here](#).

Given the number of parameters, a full sweep of all them could take very, very long. However, the caret packages already does some tuning by default. We will use the results from the previous iteration as a starting point.

Table 13: XGBoost - Best Tune

| nrounds | max_depth | eta | gamma | colsample_bytree | min_child_weight | subsample | |
|---------|-----------|-----|-------|------------------|------------------|-----------|------|
| 13 | 50 | 1 | 0.3 | 0 | 0.8 | 1 | 0.75 |

Below is the code used for tuning:

```

training_tokenised <- tokenise_data(training,extra_stop_words,
                                   domains_data,training_flag=TRUE)
scores<- calculate_scores(training_tokenised,
                          word_parameters=c("sin",1,0,
                                             word_lambda_3,word_lambda_2),
                          hashtag_parameters=c("exp",4,
                                                hashtag_lambda_2,hashtag_lambda_3),
                          handle_parameters=c("exp",4,0.8,4),
                          link_parameters=c("exp",4,0.9,4),
                          manual_scores = manual_scores)
training_vector <- score_tweets(training,training_tokenised,scores)

aggregate_tuning_grid <- expand.grid(nrounds = seq(45,55,1),
                                   max_depth = 1:3,
                                   eta = seq(0.2,0.4,0.1),
                                   gamma =seq(0,1,0.1),
                                   colsample_bytree = seq(0.7,0.9,0.1),
                                   min_child_weight = 1:3,
                                   subsample = 0.75)

fitting_model <- fit_model(training_vector,method_word="glm",
                           method_hashtag="glm",
                           method_link="glm",
                           method_handle="glm",
                           method_aggregate="xgbTree",
                           tuneGrid_aggregate = aggregate_tuning_grid)

results<-predict_values(test,fitting_model,scores,extra_stop_words,domains_data)

aggregate_model_3 <- fitting_model$aggregate
stats_7 <- results$eval

```

The optimised solution results in an accuracy of **0.799**. The optimised XGBoost parameters are:

Table 14: XGBoost - Best Tune

| | nrounds | max_depth | eta | gamma | colsample_bytree | min_child_weight | subsample |
|-----|---------|-----------|-----|-------|------------------|------------------|-----------|
| 254 | 45 | 1 | 0.2 | 0.2 | 0.8 | 3 | 0.75 |

When compared with all the previous results, we have the below chart

Table 15: Initial Comparison

| attempt | accuracy |
|------------------------------------|-----------|
| Initial training | 0.7571429 |
| Initial training - Data Correction | 0.7688679 |
| Curated hashtags | 0.7735849 |
| Optimised word score - lambda_2 | 0.7990566 |
| Optimised word score - lambda_3 | 0.7990566 |
| Optimised hashtag score - lambda_3 | 0.7989691 |
| Word score - LogitBoost | 0.7952830 |
| Other scores - LogitBoost | 0.7989691 |
| Optimised XGBoost | 0.7990566 |

As shown above, the greatest gains in accuracy are due to the data correction and the tuning of the word scoring formula. Further tuning only produces worse or marginally better results (from hashtag optimisation to optimised XGBoost, for example). Furthermore, the first impression is that perhaps there is a ceiling that has been reached, perhaps due to the limitation of the model or quality of the training data (bad inputs -> bad results).

Nevertheless, up to this point we have try touched all the relevant aspects of the model. Therefore (for the purpose of the this assignment), the only step left is to assess against the validation data and then comment based on those results.

3 Results

As the last step of this report, the model has been assessed against the **validation** dataset that was set aside at the beginning of this exercise. In order to accomplish this, the below steps have been followed:

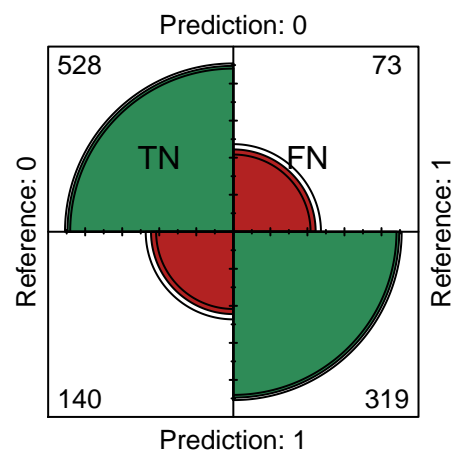
1. **Training** and **test** set have been merged into one dataset.
2. The model has been trained against this new dataset.
3. The newly-trained model has been used to generate prediction for the **validation** dataset

The result of this is displayed in the below accuracy summary

Table 16: XGBoost - Best Tune

| Component | Accuracy |
|-----------|-----------|
| word | 0.7994324 |
| hashtag | 0.7979798 |
| link | 0.7536232 |
| handle | 0.8750000 |
| aggregate | 0.7990566 |

The confusion matrix for this result is shown below:



```
## integer(0)
```

Looking at those results, it looks like we have reached some sort of ceiling for the model. As mentioned before, possible culprits are the model or the data. Considering that at the start of this report data issues were flagged, it wouldn't be unreasonable to suspect that is a main contributor. To confirm this, both false positives and false negatives were inspected - below are a sample of both groups:

Table 17: Validation Results - False Positives

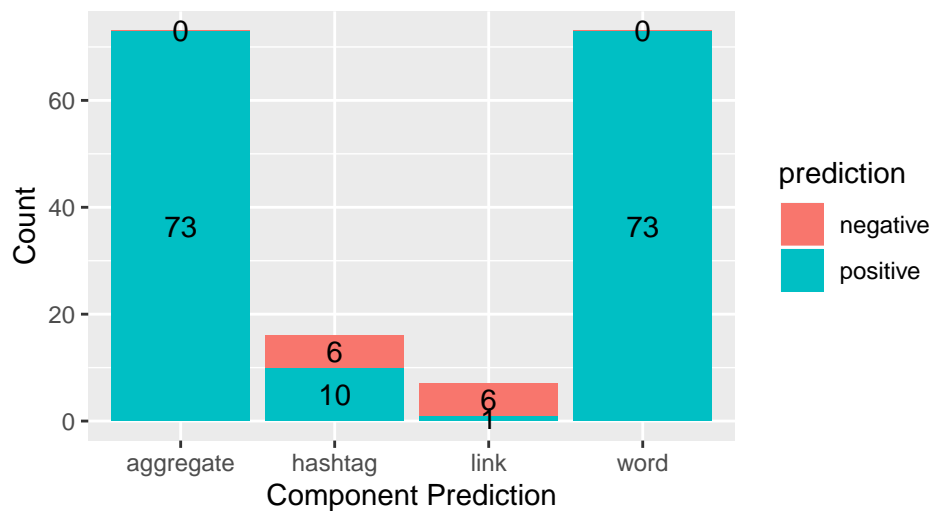
| id | target | original_text |
|--------|--------|--|
| 490 | 0 | 9 Charts Prove Financial Crisis Part 2 Has BEGUN!: The Financial Armageddon Economic Collapse Blog tracks tren... http://t.co/vHCXTvCINr |
| 650 | 0 | Credit to @pfannebeckers for inspiring me to rediscover this fantabulous #tbt http://t.co/wMHy47xkiL |
| 2 153 | 0 | @BarackObama hello Mr. President there is a really big problem in Puerto Rico regarding the water situation no more like a catastrophe... |
| 5 434 | 0 | Attention Service Members Veterans Educators First Responders in Jacksonville FL http://t.co/4UrtBEAcE5 |
| 9 600 | 0 | Raise your words not your voice. It is rain that grows flowers not thunder. |
| 10 402 | 0 | NEW YORK: A whirlwind day of activities in New York. Breakfast at the Millennium Hotel United Nations Plaza. Lunch... http://t.co/laYZBA9y8h |
| 10 492 | 0 | 11:57am Wildfire by The Mynabirds from Lovers Know |

Table 18: Validation Results - False Negatives

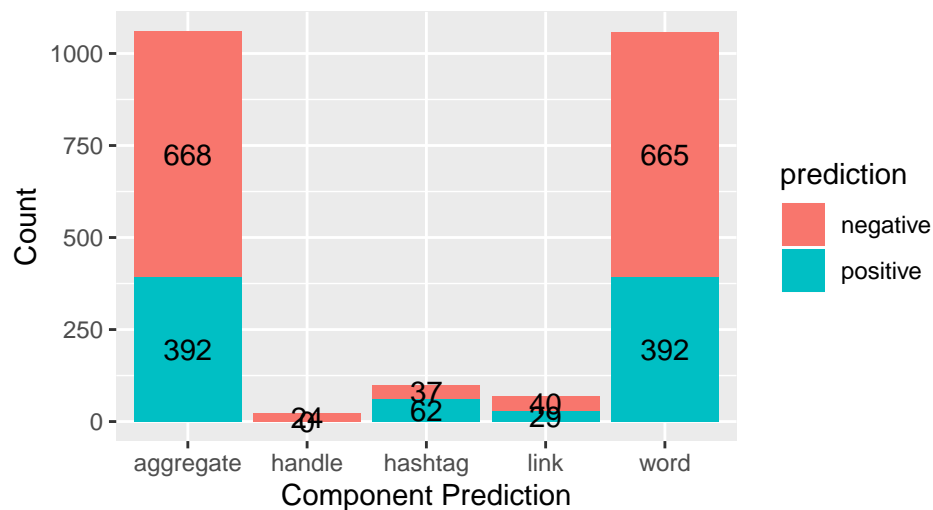
| id | target | original_text |
|-------|--------|---|
| 283 | 1 | New Nanotech Device Will Be Able To Target And Destroy Blood Clots http://t.co/MnmyJXQ9go #science |
| 391 | 1 | World Annihilation vs Self Transformation http://t.co/pyehwodWun Aliens Attack to Exterminate Humans http://t.co/8jxqL8Cv8Z |
| 1 172 | 1 | #Detroit has made progress against blight but too many burned out shells of houses remain. |
| 1 772 | 1 | Fire hazard associated with installation of non-compliant external cladding on http://t.co/bTPQdehl3p - By @www.cbplawyers |
| 2 341 | 1 | Why did I come to work today.. Literally wanna collapse of exhaustion |
| 3 259 | 1 | Salvation Army bid to demolish cottages http://t.co/3kuXonOchl #Southend http://t.co/enQaSCGFyw |
| 4 310 | 1 | Dust Storm 'en route' from Alice Springs to Uluru http://t.co/4ilt6FXU45 |
| 9 781 | 1 | Literally trapped in my room Cuz my bathroom being remodeled. The only exit is through a window |
| 9 795 | 1 | Hollywood Movie About Trapped Miners Released in Chile: 'The 33' Hollywood movie about trapped miners starring... http://t.co/tyyfG4qQvM |

After inspecting for set, it looks like there is a problem of misclassification with the false negatives. In this case the model *correctly* predicts most of them as not a disaster tweet. However, upon inspection it seems the model is clearly *incorrectly* classifying most of all false positives. It is worth noticing though that all of them contain “catastrophe” words. A possible reason of this that the model is mostly driven by the words rather than the other attributes and its limited lexicon results in wrong scores for those tweets. Fortunately, it is possible to see for those tweets whether a prediction for each component was generated or not, which is shown in the below charts, for false positives and all validation tweets.

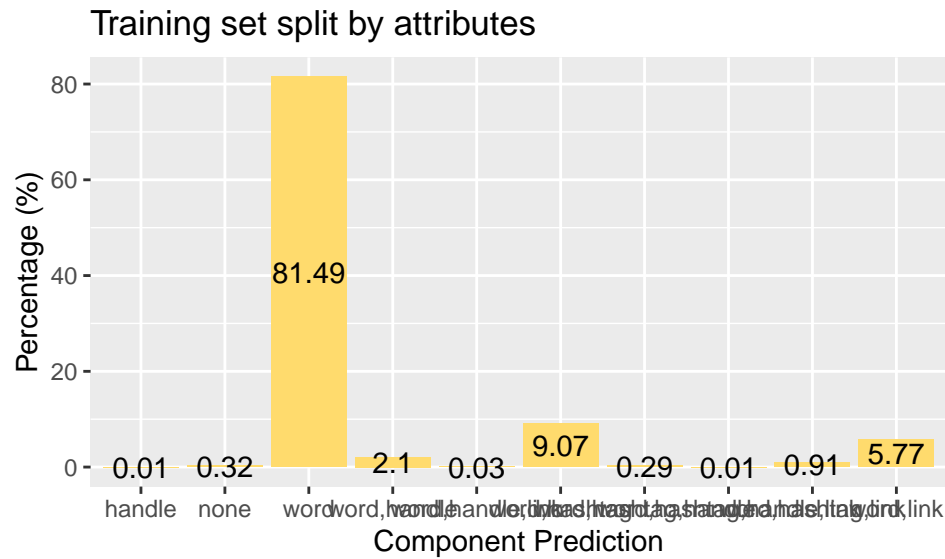
Count of Predictions per component – False Positives



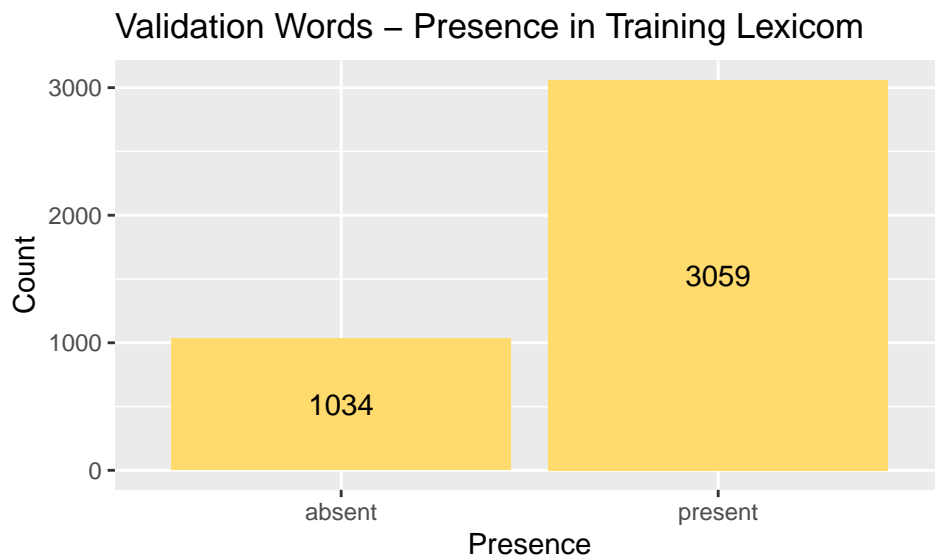
Count of Predictions per component – All Validation Re



Both results show - interestingly that the overall results is almost entirely driven by the word scoring model. In fact, this model is basically ignoring the other attributes. This can be explained by the attribute distribution in the training data. As shown in below graph, the tweets that only have words exceed by far any other combination, which probably results in the XGBoost algorithm ignoring all other components. Although in theory hashtags, links and handles make sense, the existing data is making them irrelevant.



This probably only compounds another problem previously mentioned: the absence of several words in the lexicon created by the training data. This is summarised by the below chart.



The consequence of this will be incorrect or incomplete scoring - this is one probable reason for the high number of false negatives. Again, the conclusion seems to be that good quality training data, in reasonable quantities is critical for a good machine learning model.

4 Conclusion

- Good quality data matters. Bad input means bad results.
- Sometimes manual curation will improve the results.
- There is so much to tune.