Amazon reviews sentiment analysis

Sofia Gambirasio

In the following I analyse the Amazon reviews of the **Sony WH-CH520 Wireless Bluetooth Headphones**.

Data Scraping

Firstly I scrape the following information: title of the review, content of the review and stars, and I save them in a tibble. Beware that non-uk reviews need to be scraped separately

```
library(tidyverse)
library(xml2)
library(rvest)
```

```
amazon_reviews <- function(id, page) {
  url <- paste0("https://www.amazon.co.uk/product-reviews/",id, "/?pageNumber=", page)
  html <- read_html(url)

  # scrape review TITLE UK
  title = html %>%
    html_elements("[class='a-size-base a-link-normal review-title a-color-base review-title-content a-te
    html_text2()

  # scrape review TITLE not UK
  title = title %>%
    c(html %>%
        html_elements("[class='a-size-base review-title a-color-base review-title-content a-text-bold']
        html_text2())

  # scrape review TEXT (UK and not-UK)
  text = html %>%
    html_elements("[class='a-size-base review-text review-text-content']") %>%
    html_text2()

  # scrape review STARS (UK and not-UK)
  star = html %>%
    html_elements("[data-hook='review-star-rating']") %>%
    html_text2()

  star = star %>%
    c(html %>%
        html_elements("[data-hook='cmps-review-star-rating']") %>%
        html_text2())

  # Return a tibble with all the scraped data
  tibble(title, text, star, page = page) %>%
    return()
}
```

```
# We use the map_df function from the purrr package in order to iterate the task over multiple pages.
id = "B0BTJ8ZXG5"
page = 1:39
data_scraped = map_df(page, ~amazon_reviews(id = "B0BTJ8ZXG5", page = .))

# add a doc_id
data_scraped$doc_id = 1:nrow(data_scraped)
```

Actually we will use the following dataset which contains all reviews up until the 6/6/2023 for reproducibility

```
data = readRDS(file = 'C:\\Users\\sofia\\OneDrive\\Desktop\\Lavoro\\Portfolio\\text mining\\scraped_revi

glimpse(data)
```

```
## Rows: 359
## Columns: 5
## $ title  <chr> "I love these", "Sony have done a cracking job on these Heaphon~
## $ text   <chr> "I only brought these for travel as in-ear buds only last a cer~
## $ star   <chr> "4.0 out of 5 stars", "5.0 out of 5 stars", "4.0 out of 5 stars~
## $ page   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, ~
## $ doc_id <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, ~
```

Preprocessing

Language detection

Since we will be using methods that are based on the english language, we remove all reviews that are not in english

```
library(cld2)
```

```
# apply the language detector to both title and text
data$title_lang = detect_language(data$title)
data$text_lang = detect_language(data$text)
# combination of languages between title and text
table(Text = data$text_lang, Title = data$title_lang, useNA = "always")
```

```
##       Title
## Text    ca de en es fr it pt <NA>
##   de     0 48  0  0  0  0  0   13
##   en     0  0 75  0  0  0  0   39
##   es     1  0  1 38  0  0  2   20
##   fr     0  0  2  0 24  0  0    8
##   it     0  0  4  0  0 46  0   22
##   pt     0  0  0  0  0  0  3    1
##   sv     0  0  0  0  0  0  0    1
##   <NA>   0  1  1  1  0  1  1    6
```

```
# filter out reviews which have a text not in english
data = data %>%
  filter(text_lang == "en")
```

Stars of the reviews

Let's create a new numeric variable with the number of stars saved as text in the variable `star`

```r
data = data %>%
  mutate(score = as.numeric(substring(star, 1, 1)))
summary(data$score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   4.000   5.000   4.377   5.000   5.000
```

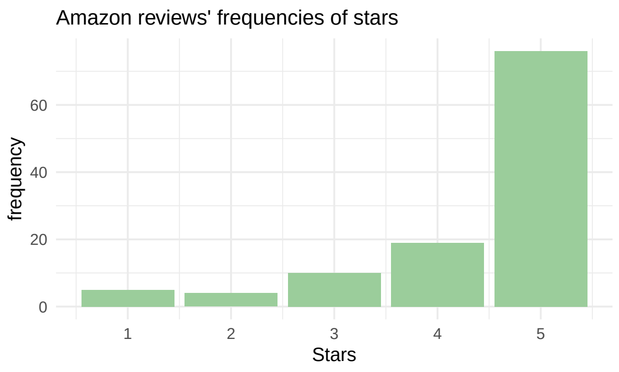
```r
library(knitr)
```

In the following table and plot we can see clearer that these headphones have in general really high ratings: the 67 percent of reviews have given 5 stars to the product.

```r
# table to see the number and percentage of ratings with a certain score
tab_stars = data %>%
  count(score) %>%
  mutate(perc = round(n/sum(n), 2)) %>%
  rename(count = n)
kable(tab_stars, align = 'l')
```

score	count	perc
1	5	0.04
2	4	0.04
3	10	0.09
4	19	0.17
5	76	0.67

```r
# bar plot with stars frquencies
data %>%
  ggplot(aes(x = score)) +
  geom_bar(aes(y = (..count..)), fill = "darkseagreen3") +
  labs(title = "Amazon reviews' frequencies of stars",
       x = "Stars", y = 'frequency') +
  theme_minimal()+
  theme(plot.title = element_text(color = "black", size = 12),
        plot.subtitle = element_text(color = "black"))
```



Preprocessing for content and sentiment analysis

```r
library(tidytext)
```

Tokenization

```r
data = data %>%
  mutate(id=seq_along(text))
head(data)
```

```
## # A tibble: 6 x 9
##   title                       text  star   page doc_id title~1 text_~2 score    id
##   <chr>                       <chr> <chr> <int>  <int> <chr>   <chr>   <dbl> <int>
## 1 "I love these"              "I o~ 4.0 ~     1      1 <NA>    en          4     1
## 2 "Sony have done a cracki~   "Fir~ 5.0 ~     1      2 en      en          5     2
## 3 "SONY BLUETOOTH \U0001f3~   "Fri~ 4.0 ~     1      3 en      en          4     3
## 4 "Uncomfortable and signi~   "I b~ 3.0 ~     1      4 en      en          3     4
## 5 "very good"                 "i g~ 4.0 ~     1      5 <NA>    en          4     5
## 6 "Satisfied"                 "I w~ 5.0 ~     1      6 <NA>    en          5     6
## # ... with abbreviated variable names 1: title_lang, 2: text_lang
```

```r
tidy.data = data %>%
  unnest_tokens(word,text)
```

We want to make sure that we are **not removing stop words that are relevant for our analysis**. In particular, there could be words in stop words that are associated with a sentiment, removing them could alter the final results about the sentiment of our reviews. Therefore we perform an *inner join between the tokens, the stop words and also the nrc* (the lexicon that we will use in the analysis) to identify the meaningful words that would be deleted with the stopwords.

```r
# inner join with stop words to asses if we want to remove all of them
nrc = get_sentiments ('nrc') %>%
  filter(sentiment %in% c('positive','negative'))
not_stop_words = tidy.data %>%
  inner_join(stop_words) %>%
  inner_join(nrc) %>%
  select(word) %>%
  count(word,sort = T)
```

```
## Joining with `by = join_by(word)`
## Joining with `by = join_by(word)`
```

We get as an output 10 words that we proceed to remove from the stop words list so that they are not cancelled out. Now we can finally remove the remaining stop words from the tokens and further clean them.

```r
# remove the just identified not stop words from the stop words
stop_words_1 = stop_words %>%
  filter(!word %in% not_stop_words$word)

# further clean from meaningless tokens
tidy.data = tidy.data %>%
  anti_join(stop_words_1) %>%
  filter(!str_detect(word,'[[:digit:]]')) %>%
  filter(!str_detect(word, '.\\..')) %>%
  filter(!str_detect(word, '^[:alpha:]{2}$')) %>%
  filter(!str_detect(word, '.:.'))
```

We now produce a frequency count of the most frequent words in the comments, to assess that we have removed all the meaningless words.

```r
freq.df = tidy.data %>%
  count(word, sort = T)
freq.df %>% slice(1:20)
```

```
## # A tibble: 20 x 2
##    word            n
##    <chr>       <int>
##  1 headphones    127
##  2 sound          86
##  3 good           70
##  4 quality        64
##  5 sony           49
##  6 battery        41
##  7 ear            38
##  8 bluetooth      37
##  9 price          36
## 10 life           34
## 11 comfortable    28
## 12 app            25
## 13 ears           25
## 14 noise          25
## 15 easy           24
## 16 music          24
## 17 hours          23
## 18 wear           23
## 19 wireless       22
## 20 time           21
```

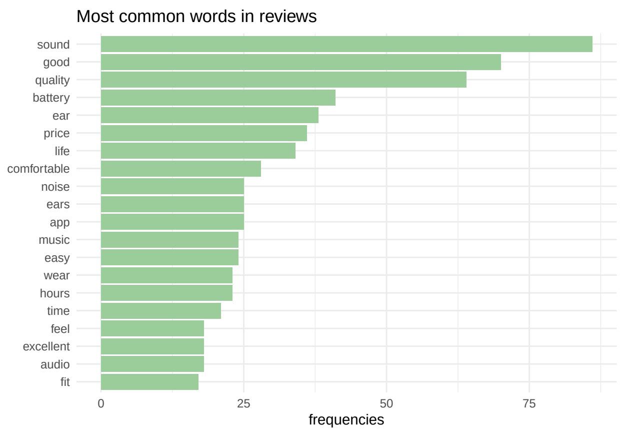
Better to remove words that are included in the full name of the product, as they are not conveying any additional information : "headphones", "headphone","sony", "bluetooth","wireless"

```
# remove additional words
new_stop_words = c("headphones", "headphone","sony", "bluetooth","wireless")
tidy.data.1 = tidy.data %>%
  filter(!word %in% new_stop_words)
```

Analysis of the content

Let's look at the frequency plot with the most common words in the reviews

```
freq.df.1 = tidy.data.1 %>%
  count(word, sort = T)

freq.df.1 %>%
  slice(1:20) %>%
  mutate(word = reorder(word,n)) %>%
  ggplot(aes(word,n))+
  geom_col(show.legend=F, fill = 'darkseagreen3') +
  xlab(NULL)+
  ylab('frequencies')+
  ggtitle('Most common words in reviews')+
  coord_flip()+
  theme_minimal()
```



Sentiment analysis

Choice of the lexicon

Fistly we must choose the lexicon for the dictionary based method. To do so, we perform inner joins between the words that we have selected and the words in the three lexicons to see which one has the highest number of matches

```
# bing
bing = get_sentiments('bing')
tidy.data.1 %>%
  select(word) %>%
  unique() %>%
  inner_join(bing) %>%
  count() %>%
  print()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   225
```

```
# nrc
nrc = get_sentiments ('nrc') %>%
  filter(sentiment %in% c('positive','negative'))
tidy.data.1 %>%
  select(word) %>%
  unique() %>%
  inner_join(nrc) %>%
  count() %>%
  print()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   233
```

```
# afinn
afinn = get_sentiments('afinn')
tidy.data.1 %>%
  select(word) %>%
  unique() %>%
  inner_join(afinn) %>%
  count() %>%
  print()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   160
```

nrc is the lexicon with the highest number of matching words, therefore we will use it

Nrc tidytext

```
# compute the sentiments for each comment
sent.nrc.tidy = tidy.data.1 %>%
  inner_join(nrc) %>%
  count(id,sentiment) %>%
  pivot_wider(names_from = sentiment,values_from = n,values_fill = 0) %>%
  mutate(nrc_tidy_sent = positive-negative) %>%
  select(id,nrc_tidy_sent)
head(sent.nrc.tidy)
```

```
## # A tibble: 6 x 2
##      id nrc_tidy_sent
##   <int>         <int>
## 1     1             0
## 2     2             9
## 3     3            10
## 4     4            -5
## 5     5             4
## 6     6            -4
```

```
# join the sentiments and the comments
sentiment_all_wide = sent.nrc.tidy %>%
  full_join(data) %>%
  select(id,title,text,score,nrc_tidy_sent)

tail(sentiment_all_wide)
```

```
## # A tibble: 6 x 5
##      id title                        text                         score nrc_t~1
##   <int> <chr>                        <chr>                        <dbl>   <int>
## 1    55 Bluetooth connection breaking "tends to break the Bluetoo~     3      NA
## 2    64 Can't connect                "The product does not conne~     2      NA
## 3   110 It is awesome headphone      "I like it. Thank you a lot"     5      NA
## 4   111 Sony WH-CH520                "Ok.excellent thank you"         5      NA
## 5   112 Ottimo                       "The media could not be loa~     5      NA
## 6   113 Bad Bluetooth connectivity   "The bluetooth kept disconn~     3      NA
## # ... with abbreviated variable name 1: nrc_tidy_sent
```

Some comments were not assigned a polarity! We can take a closer look at them

```
sentiment_all_wide %>%
  filter(is.na(nrc_tidy_sent)) %>%
  select(text)
```

```
## # A tibble: 10 x 1
##    text
##    <chr>
##  1 "i haven't charged them for 3 days and they are still under 70% and i have b~
##  2 "Great overall.Thank you"
##  3 "Great headphones for the price and the sound is quality."
##  4 "The colour and feel of the headphones is great. Really comfy, easily foldab~
##  5 "tends to break the Bluetooth connection"
##  6 "The product does not connect when using Facebook video calling. It is extre~
##  7 "I like it. Thank you a lot"
##  8 "Ok.excellent thank you"
##  9 "The media could not be loaded.\n Ottimo prodotto"
## 10 "The bluetooth kept disconnecting after 2-3 times only"
```

We notice that they are 10 and the reason is that their words were not contained in the nrc lexicon. However, according to human interpretation, they do have a polarity (ex. 'I like it, thank you a lot') that this method is not able to grasp.

Nrc udpipe

Use udpipe to account for negators, amplifiers and deamplifiers up to 2 words before the one that it is assessing

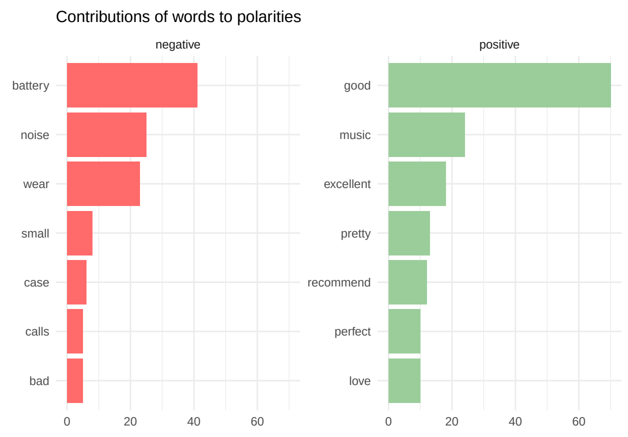
```r
library(udpipe)
library(textdata)
library(SnowballC)
```

```r
# prepare the data for udpipe
data$text=iconv(data$text, to= 'UTF-8')
output=udpipe(data, "english-gum")

# prepare the nrc lexicon for udpipe
nrc = get_sentiments("nrc") %>%
  mutate(sentiment=ifelse(sentiment=="negative", -1, 1)) %>%
  rename(term="word", polarity="sentiment")

# create stems
output1=output %>%
  mutate(stem=wordStem(token))

# compute polarity
sent.nrc.udpipe=txt_sentiment(x=output, term="lemma",
                              polarity_terms=nrc,
                              polarity_negators = c("not","no","didn't","without","neither"),
                              polarity_amplifiers = c("really","very","definitely","super"),
                              polarity_deamplifiers = c("barely","hardly"),
                              amplifier_weight=0.8,
                              n_before=2,
                              n_after=0,
                              constrain=F)

# add to sentiment_all_wide the polarity computed with udpipe
sentiment_all_wide$nrc_udipipe_sent=sent.nrc.udpipe$overall$sentiment_polarity
# still create two tibbles with the individual polarities of the two methods
udpipe_polarity= tibble(polarity=sentiment_all_wide$nrc_udipipe_sent,method="udpipe")
tidy_polarity= tibble(polarity=sentiment_all_wide$nrc_tidy_sent,method="tidy")
```

Polarities in the documents

Most frequent words by sentiment

```r
# frequency table of the sentiments by word
nrc = get_sentiments('nrc') %>%
  filter(sentiment %in% c('positive','negative'))

word_pol_freq = tidy.data.1 %>%
  inner_join(nrc) %>%
  count(word,sentiment) %>%
  group_by(sentiment) %>%
  arrange(desc(n))

# plot in two histograms by polarity (negative vs positive)
word_pol_freq %>%
  filter(!sentiment == 'neutral') %>%
  group_by(sentiment) %>%
  slice_max(n, n = 7, with_ties = F) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot()+
  geom_col(aes(n, word, fill= sentiment), show.legend = FALSE)+
  facet_wrap(~sentiment, scales = "free_y")+
  labs(x = NULL, y = NULL, title = 'Contributions of words to polarities') +
  theme_minimal() +
  theme(plot.title = element_text(color = "black", size = 12)) +
  scale_fill_manual(values = c('indianred1','darkseagreen3'))
```



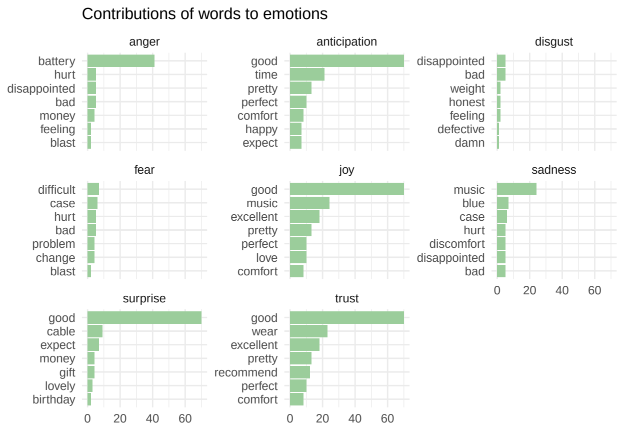
Contribution of words to emotions

```r
# look at the contribution of words to emotions
nrc11 = get_sentiments ('nrc') %>%
  filter(!sentiment %in% c('positive','negative'))

word_pol_freq = tidy.data.1 %>%
  inner_join(nrc11) %>%
  count(word,sentiment) %>%
  group_by(sentiment) %>%
  arrange(desc(n))
```

```r
head(word_pol_freq)
```

```
## # A tibble: 6 x 3
## # Groups:   sentiment [5]
##   word    sentiment        n
##   <chr>   <chr>        <int>
## 1 good    anticipation    70
## 2 good    joy             70
## 3 good    surprise        70
## 4 good    trust           70
## 5 battery anger           41
## 6 music   joy             24
```

```r
word_pol_freq %>%
  group_by(sentiment) %>%
  slice_max(n, n = 7, with_ties = F) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot()+
  geom_col(aes(n, word), fill= 'darkseagreen3', show.legend = FALSE)+
  facet_wrap(~sentiment, scales = "free_y")+
  labs(x = NULL, y = NULL, title = 'Contributions of words to emotions') +
  theme_minimal() +
  theme(plot.title = element_text(color = "black", size = 12))
```



Model comparison

Now compare the nrc tidytext with the nrc udpipe sentiments

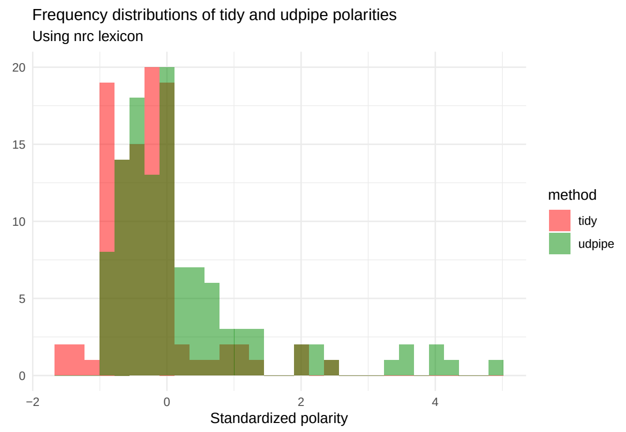
```r
sentiment_all=bind_rows(udpipe_polarity,tidy_polarity) %>%
  mutate(polarity_std = scale(polarity))#add standardize column

# summary statistics udpipe vs tidytext
sentiment_all %>%
  group_by(method) %>%
  summarise(mean=mean(polarity,na.rm=TRUE),
            sd=sd(polarity,na.rm=TRUE),
            n=n(),
            min=min(polarity,na.rm=TRUE),
            max=max(polarity,na.rm=TRUE))
```

```
## # A tibble: 2 x 6
##   method  mean    sd     n   min   max
##   <chr>  <dbl> <dbl> <int> <dbl> <dbl>
## 1 tidy    1.85  3.73   114  -6      17
## 2 udpipe  5.08  6.72   114  -1.8    31
```

```
# the same but with the standardized variables
sentiment_all %>%
  group_by(method) %>%
  summarise(mean=mean(polarity_std,na.rm=TRUE),
            sd=sd(polarity_std,na.rm=TRUE),
            n=n(),
            min=min(polarity_std,na.rm=TRUE),
            max=max(polarity_std,na.rm=TRUE))
```

```
## # A tibble: 2 x 6
##   method   mean    sd     n    min   max
##   <chr>   <dbl> <dbl> <int>  <dbl> <dbl>
## 1 tidy   -0.295 0.652   114 -1.67   2.35
## 2 udpipe  0.269 1.17    114 -0.933  4.80
```

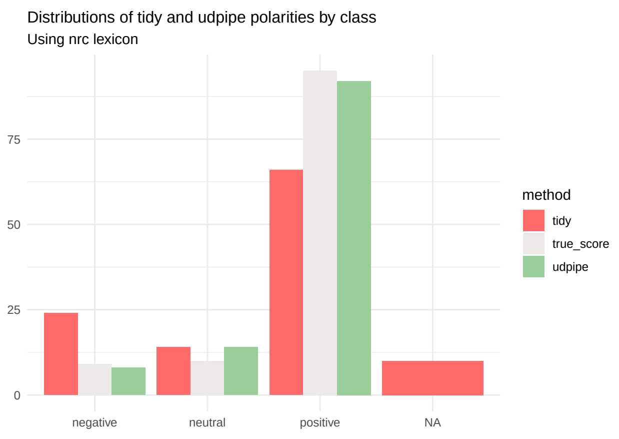
```
# overlapping histograms of the standardized polarity
sentiment_all %>%
  ggplot()+
  geom_histogram(aes(polarity_std, fill = method), alpha =0.5,position = 'identity')+
  scale_fill_manual(values = c('red','green4')) +
  theme_minimal()+
  labs(x = "Standardized polarity", y = NULL,
       title = 'Frequency distributions of tidy and udpipe polarities',
       subtitle = 'Using nrc lexicon') +
  theme(plot.title = element_text(color = "black", size = 12))
```



To assess the models, we assume that the scores given by the stars are the true polarity. Therefore we exploit this to see which model predicts a sentiment closer to the true polarity.

```
# bar chart pos-neg_neutral

# create tibble with true scores (the scores of the stars)
scores = tibble(method = 'true_score', polarity_classes = data$score) %>%
  arrange(polarity_classes) %>%
  mutate(polarity_classes = ifelse(polarity_classes<3,'negative',
                                   ifelse(polarity_classes>3,'positive','neutral')))
```

```r
# mutate the scores into polarity and bind them to sentiment all
sentiment_all = sentiment_all %>%
  arrange(polarity) %>%
  mutate(polarity_classes = ifelse(polarity<0,'negative',
                                   ifelse(polarity>0,'positive','neutral'))) %>%
  select(method,polarity_classes) %>%
  bind_rows(scores)

# plot the methods in dodged histogram
sentiment_all %>%
  ggplot()+
  geom_bar(aes(polarity_classes, fill = method), position = 'dodge')+
  scale_fill_manual(values = c('indianred1','snow2','darkseagreen3')) +
  theme_minimal()+
  labs(x = NULL, y = NULL,
       title = 'Distributions of tidy and udpipe polarities by class', subtitle = 'Using nrc lexicon')
  theme_minimal() +
  theme(plot.title = element_text(color = "black", size = 12))
```



What appears clearly from the plot is that the udpipe method provides polarities that match definitely more the true ones of the scores, while the tidy one gives less positives and more negatives than the true ones. However this doesn't necessarily imply that udpipe gave the correct polarity to the documents, therefore we compute correlations between the two polarities of the two methods and the true scores

```r
# correlations

# correlations between the two polarities and the true scores
sentiment_all_wide %>%
  filter(complete.cases(.)) %>%
  select(score,nrc_tidy_sent,nrc_udipipe_sent) %>%
  cor() %>%
  round(digits = 2)
```

```
##                  score nrc_tidy_sent nrc_udipipe_sent
## score             1.00          0.10             0.11
## nrc_tidy_sent     0.10          1.00             0.22
## nrc_udipipe_sent  0.11          0.22             1.00
```

The correlations are quite low in order to draw conclusions. However it is interesting to notice that they are more similar to each other, rather than to the true score. We can then build confusion matrices to see specifically the mismatches.

```r
# confusion matrices

# make sentiment_all_wide with qualitative polarities
sentiment_all_wide = sentiment_all_wide %>%
  mutate(score_class = ifelse(score<3,'negative',
                              ifelse(score>3,'positive','neutral')),
         tidy_class = ifelse(nrc_tidy_sent<0,'negative',
                             ifelse(nrc_tidy_sent>0,'positive','neutral')),
         udpipe_class = ifelse(nrc_udipipe_sent<0,'negative',
                               ifelse(nrc_udipipe_sent>0,'positive','neutral')))
glimpse(sentiment_all_wide)
```

```
## Rows: 114
## Columns: 9
## $ id               <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16~
## $ title            <chr> "I love these", "Sony have done a cracking job on the~
## $ text             <chr> "I only brought these for travel as in-ear buds only ~
## $ score            <dbl> 4, 5, 4, 3, 4, 5, 5, 5, 5, 4, 4, 5, 5, 5, 4, 5, 5, 5,~
## $ nrc_tidy_sent    <int> 0, 9, 10, -5, 4, -4, 0, -2, 4, 4, 3, 2, -2, 5, 2, 1, ~
## $ nrc_udipipe_sent <dbl> 6.0, 22.8, 23.4, 15.6, 10.8, 7.8, 4.0, 14.6, 9.8, 4.0~
## $ score_class      <chr> "positive", "positive", "positive", "neutral", "posit~
## $ tidy_class       <chr> "neutral", "positive", "positive", "negative", "posit~
## $ udpipe_class     <chr> "positive", "positive", "positive", "positive", "posi~
```

```r
#CM tidytext
CM_tidy = table(sentiment_all_wide$tidy_class,sentiment_all_wide$score_class)
CM_tidy
```

```
##
##            negative neutral positive
##   negative        4       3       17
##   neutral         1       0       13
##   positive        3       4       59
```

```r
#CM udpipe
CM_udpipe = table(sentiment_all_wide$udpipe_class,sentiment_all_wide$score_class)
CM_udpipe
```

```
##
##            negative neutral positive
##   negative        0       1        7
##   neutral         2       1       11
##   positive        7       8       77
```

```r
# create function to compute performance indexes
perf_indexes = function(cm){
  correct_neg = cm[1,1] / (cm[1,1] + cm[2,1] + cm[3,1])
  correct_neu = cm[2,2] / (cm[1,2] + cm[2,2] + cm[3,2])
  correct_pos = cm[3,3] / (cm[1,3] + cm[2,3] + cm[3,3])
  accuracy = sum(diag(cm)) / sum(cm)
  return(c(correct_neg=correct_neg,
           correct_neu=correct_neu,
           correct_pos=correct_pos,
           accuracy=accuracy))
}
```

```
perf_index_tidy = round(perf_indexes(CM_tidy),digits = 2)
perf_index_tidy
```

```
## correct_neg correct_neu correct_pos    accuracy
##        0.50        0.00        0.66        0.61
```

```
perf_index_udpipe = round(perf_indexes(CM_udpipe),digits = 2)
perf_index_udpipe
```

```
## correct_neg correct_neu correct_pos    accuracy
##        0.00        0.10        0.81        0.68
```

The udpipe method is better at identifying positive and neutral comments, while the tidy works better for negative ones (however we must point out that the true neutral value of udpipe is 0.10 and the true negative of tidytext 0.50, both being at maximum as good as random). The overall accuracy though, is higher for udpipe, so we decide that **udpipe is overall the best performer** for this context.

```
# percentages of polarities in documents (udpipe)
sentiment_all_wide %>%
  count(udpipe_class) %>%
  mutate(perc_polarity = n/nrow(sentiment_all_wide)*100)
```

```
## # A tibble: 3 x 3
##   udpipe_class     n perc_polarity
##   <chr>        <int>         <dbl>
## 1 negative         8          7.02
## 2 neutral         14         12.3
## 3 positive        92         80.7
```

Therefore, based on the udpipe classification of the documents, we can conclude that *]7% of documents are negative, 12.3% are neutral and 80.7% are positive*].

```
# contribution of words to the sentiment
nrc = get_sentiments ('nrc') %>%
  filter(sentiment %in% c('positive','negative'))
```