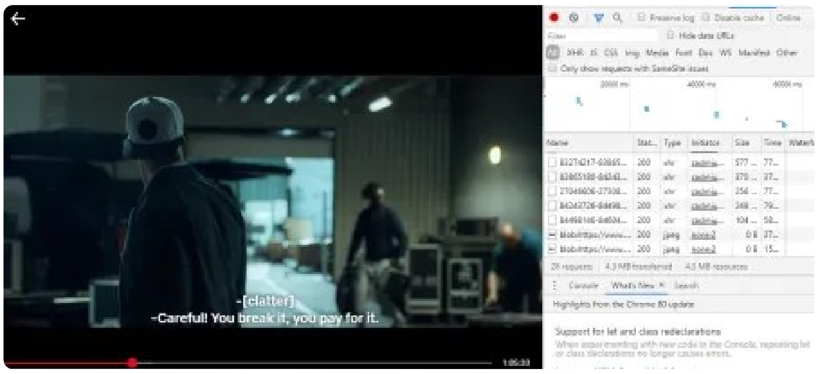
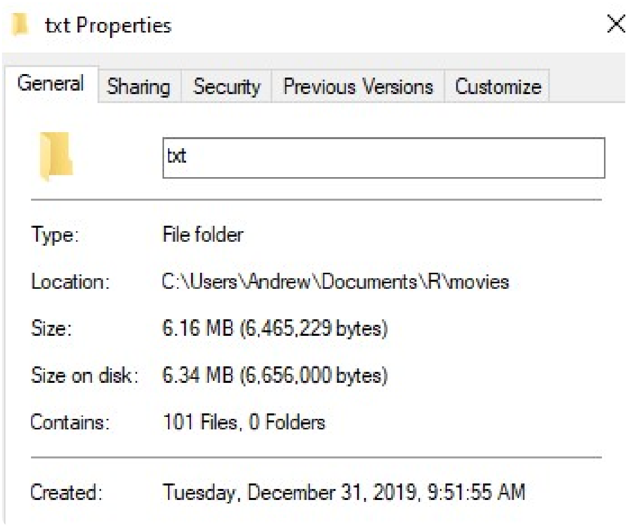
My primary goal was to research popular movies *vocabulary* – words all those heroes actually saying to us and put well known movies onto the scale from the ones with the *widest* lexicon down to below average range despite possible crowning by IMDB rating, Box Office revenue or number / lack of Academy Awards.

This post is the 1st Part of the Movies text analysis and covers **Text loading and cleaning using *readtext* and *textclean* packages.** My first challenge was texts source – I saw text mining projects based on open data *scripts* but I wanted something new, more direct and, as close to what we actually hear AND read from the screen. I decided directly save *subtitles* while streaming using Chrome developer panel.



It requires further transformation through the chain .xml > .srt > .txt (not covered here). Since we have .txt files ready, let`s start from downloading of texts stored in separate folder (1movie = 1 file).


txt Properties

General | Sharing | Security | Previous Versions | Customize

txt

Type:	File folder
Location:	C:\Users\Andrew\Documents\R\movies
Size:	6.16 MB (6,465,229 bytes)
Size on disk:	6.34 MB (6,656,000 bytes)
Contains:	101 Files, 0 Folders
Created:	Tuesday, December 31, 2019, 9:51:55 AM

```
library(readtext)
text_raw <-readtext(paste0(path="C:/Users/Andrew/Documents/R/movies/txt/*"))
text <- gsub("\r?\n|\r", " ", text_raw)
```

To go further it is important to understand what is text and what is noise here. Since subtitles are real text representation of entire movie, its text part contains:

1. Speech (actual text)

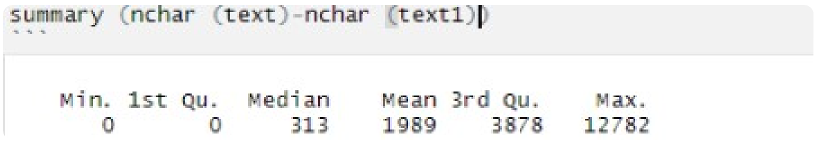
2. Any sound (made by heroes/characters or around)
3. Speakers / characters marks (who is saying)
4. Intro / end themes
5. Other text

All but first are noise, at least for purpose of measuring and compering vocabulary / lexicon. Luckily, there are limited number of patterns to distinct Speech from other sound / noise so using string processing is possible to husk the speech leaving noise apart. Let`s start from the brackets and clean not only brackets but what is inside also using textclean package.

```
library (textclean)

text1 <- str_replace_all (text, "\\[.*?\\]", "")
text1_nchar <- sum (nchar (text1))
```

It is important to check how hard your trim your “garden”, you could do it element-wise if the summary shows any odd losses or, in opposite, very little decrease of characters #.



```
summary (nchar (text)-nchar (text1))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      0       0     313    1989    3878   12782 
```

the same with () +()

```
text2 <- str_replace_all(text1, "\\(.*?\\)", "")
text2_nchar <- sum (nchar (text2))
```

Do not forget any odd characters detected, like, in my case, ‘ï»¿File:’

```
text3 <- str_replace (text2,'ï»¿File:', '')
text3_nchar <- sum (nchar (text3))
```

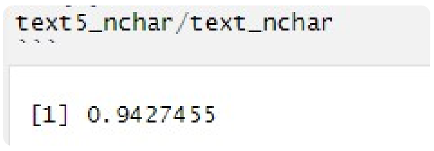
Now is the trickiest part, other brackets – ♪♪ Ideally, it requires you to know the text well to distinct theme songs and background music from the situation where singing is the way the heroes “talk” to us. For instance musicals or a lot of kids movies.

```
text4 <- str_replace_all(text3,"\\âTª.*?\\âTª", "")
text4_nchar <- sum (nchar (text4))
```

Speech markers e.g. JOHN:

```
text5 <- str_replace_all(text4,"[a-zA-Z]+:", "")
text5_nchar <- sum (nchar (text5))
```

Check % of total cut



```
text5_nchar/text_nchar
```

```
[1] 0.9427455
```

Stop cut the weed, let`s replace the rest): @#$%&

```
text6 <- replace_symbol(text5, dollar = TRUE, percent = TRUE, pound = TRUE,
                        at = TRUE, and = TRUE, with = TRUE)
text6_nchar <- sum (nchar (text6))

#time stamps
text7 <- replace_time(text6, pattern = "(2[0-3]|[01]?[0-9]):([0-5][0-9])[.:]?(
text7_nchar <- sum (nchar (text7))

#date to text
replace_date_pattern <- paste0(
  '([01]?[0-9])[/-]([0-2]?[0-9]|3[01])[/-]\\d{4}|\\d{4}[/-]',
  '([01]?[0-9])[/-]([0-2]?[0-9]|3[01])')
text8 <- replace_time(text7, pattern = replace_date_pattern)
text8_nchar <- sum (nchar (text8))

#numbers to text, e.g. 8 = eight
text9 <- replace_number(text8, num.paste = FALSE, remove = FALSE)
text9_nchar <- sum (nchar (text9))
```

Now texts are clean enough and ready for further analysis. That`s it for Movies text analysis. Part 1.