Introduction to NLP with R

Nicolas Attalides

Workshop Setup:

▶ Wi-Fi

Network Name:

Password:

▶ Resources

R (recommended: version 3.6.1)

RStudio (recommended: version 1.2.5001)

💡 ▶ Data

Download the file from:

https://github.com/nattalides/BarcelonaR_workshop_Introduction_to_NLP_with_R/tree/master/data

What is NLP?

Natural language processing (NLP) is a field within linguistics and artificial intelligence that enables computers to understand and interact with human language.

Some examples where NLP can be found are virtual assistants like Siri or Alexa, automatic spell checking, word autocompletion and machine translation such as Google Translate.
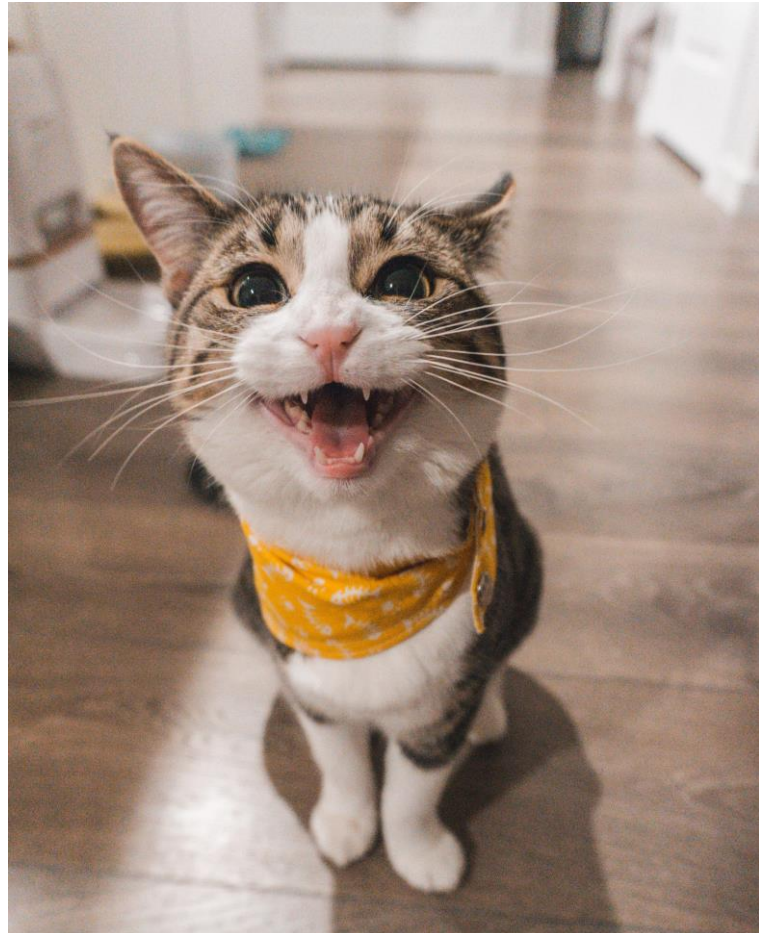
Topics

▶ Workshop aim:

Learn the basics of how to do text mining and sentiment analysis as well as some background theory on NLP.

▶ Topics:

- Tokenization and stop words

- Sentiment analysis

- Term Frequency - Inverse Document Frequency (TF-IDF)

Live Coding Example 1

1. Load the star_wars_scripts.rds dataset

column	description	column	description
line	The line of the script	length	Number of characters
movie	Which star wars episode	ncap	Number of capitalised words
title	Movie title	nexcl	Number of exclamation points
character	Character's name	nquest	Number of question marks
dialogue	Character's text	nword	Number of words

2. Which movie has the most lines?

3. Which movie has the most characters?

4. Summarise the lines, exclamations, questions, words per character per movie and sort by words (descending).

5

Live Coding Example 1

```r
library(dplyr)
library(magrittr)

# Load the star_wars_scripts.rds dataset
df <- readRDS("data/star_wars_scripts.rds")

# Which movie has the most lines?
df %>%
  group_by(movie) %>%
  summarise(line_count = n())
```

Live Coding Example 1

```r
# Which movie has the most characters?
df %>%
  group_by(movie) %>%
  summarise(character_count = n_distinct(character))

# Summarise the lines, exclamations, questions, words per character per movie
res <- df %>%
  group_by(movie, character) %>%
  summarise(line_count = n(),
            total_excl = sum(nexcl),
            total_quest = sum(nquest),
            total_words = sum(nword)) %>%
  arrange(desc(total_words))
```

Tokenization

Tokenization is the process of breaking up a text into individual tokens. A token is a unit of text that we use for text analysis. Most commonly tokens are single words. We will follow Hadley Wickham's tidy data structure and use the {tidytext} package to process our data into a table with one-token-per-row format – where a token is a single word.

A token can be more complex such as a sentence, a paragraph or a successive sequence of words, called an n-gram. A special case of an n-gram is when we tokenize by pairs of 2 consecutive words (n = 2) which we call "bigrams".

Stop words

When analysing text we will come across words that are not very meaningful. For example, in English, particularly common words such as "the", "a", "of", "to" etc. are not useful for analysis. We can remove them by using a list of words called "stop words". The {tidytext} package contains a dataset of stop words from three lexicons.

```
> stop_words
# A tibble: 1,149 x 2
   word         lexicon
   <chr>        <chr>
 1 a            SMART
 2 a's          SMART
 3 able         SMART
 4 about        SMART
 5 above        SMART
 6 according    SMART
 7 accordingly  SMART
 8 across       SMART
 9 actually     SMART
10 after        SMART
# ... with 1,139 more rows
```

Live Coding Example 2

1. Use {tidytext} to tokenize the star wars scripts, where a token is a single word to create a one-token-per-row data frame (also remove the summary columns).
2. Remove the stop words from the data frame and create "tidy_script" object.
3. Find the top 5 words for all movies and create a bar chart visualisation.
4. Find the most common word used for all the characters. What do you think is Yoda's?
5. Create an awesome word cloud!

Images in workshop GitHub repo.

Live Coding Example 2

```r
library(tidytext)

# Load the star_wars_scripts.rds dataset
df <- readRDS("data/star_wars_scripts.rds")

# Use {tidytext} to tokenize the star wars scripts, where a token is a single
# word to create a one-token-per-row data frame (also remove summary columns)
tidy_script <- df %>%
   select(-length, -ncap, -nexcl, -nquest, -nword) %>%
   unnest_tokens(output = word, input = dialogue)

# Remove the stop words from the data frame and create "tidy_script" object
tidy_script <- tidy_script %>%
   anti_join(stop_words, by = "word")
```

Live Coding Example 2

```r
# Find the top 5 words for all movies and create a bar chart visualisation.
library(ggplot2)

tidy_script %>%
  count(word, movie) %>%
  ungroup() %>%
  group_by(movie) %>%
  top_n(5) %>%
  ungroup() %>%
  ggplot(aes(word, n, fill = movie)) +
  geom_col(show.legend = FALSE) +
  labs(y = NULL, x = NULL) +
  facet_wrap(~movie, ncol = 3, scales = "free_y") +
  coord_flip()
```
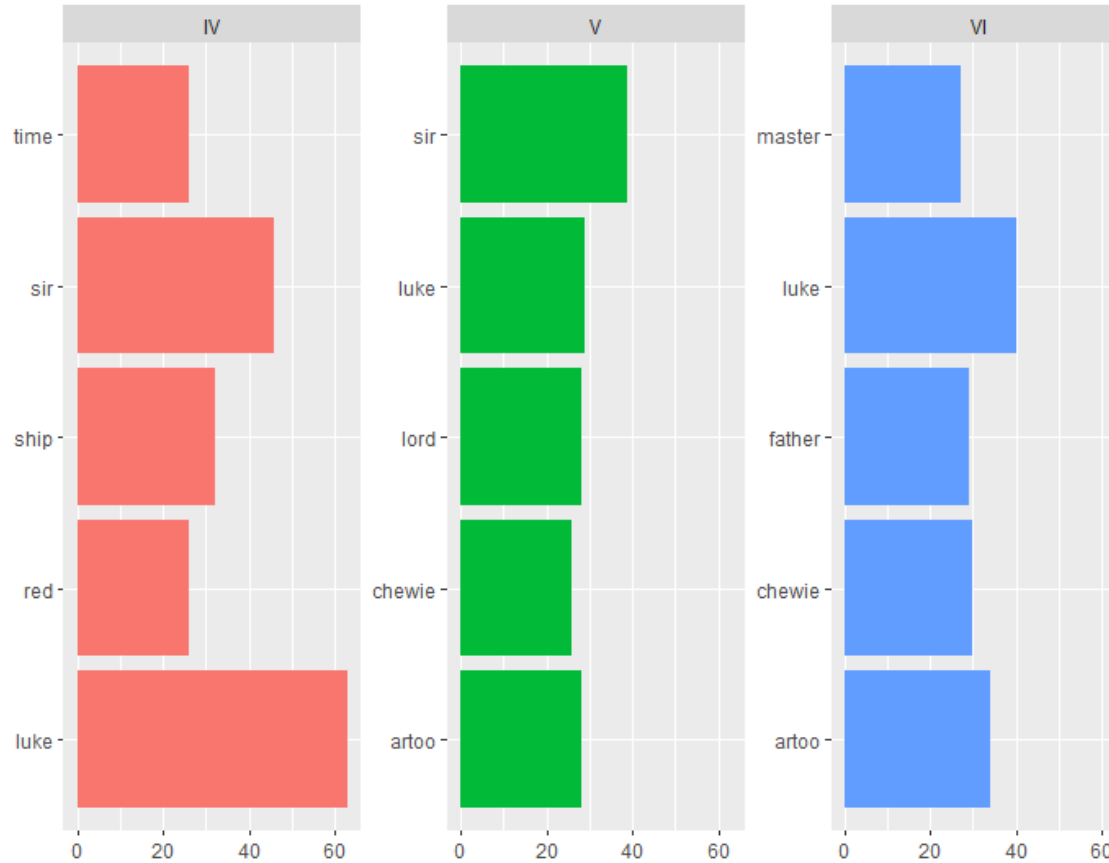
Live Coding Example 2

Live Coding Example 2

```
# Find the most common word used for all the characters.
# What do you think is Yoda's?
res <- tidy_script %>%
    count(word, character) %>%
    ungroup() %>%
    group_by(character) %>%
    top_n(1) %>%
    ungroup() %>%
    arrange(desc(n))
```

	word	character	n
1	artoo	THREEPIO	57
2	sir	THREEPIO	57
3	chewie	HAN	46
4	artoo	LUKE	24
5	ben	LUKE	24
6	luke	BEN	22
7	luke	LEIA	16
8	lord	PIETT	15
9	force	YODA	14
10	luke	BIGGS	13
11	master	VADER	12
12	friends	EMPEROR	7

Live Coding Example 2

```r
# Create an awesome word cloud!
# devtools::install_github("lchiffon/wordcloud2")
# Might require some package installation steps
library(wordcloud2)

plot_data <- tidy_script %>%
  count(word) %>%
  ungroup() %>%
  mutate(word = factor(word),
         freq = as.numeric(n)) %>%
  arrange(desc(freq))

wordcloud2(plot_data, size = 1, figPath="data/vader.png")

wordcloud2(plot_data, size = 1, figPath="data/yoda.png")
```

Live Coding Example 2

Sentiment analysis

It is easy for humans to understand the emotional content of a piece of text and interpret it as something positive or negative. We can even describe some text as expressing anger, disgust or surprise.

Sentiment analysis, which is also known as opinion mining, is the process where we aim to attach emotional content to a piece of text in a programmatic way.

The most common method to analyse the sentiment of a piece of text is to add up the individual sentiment content of each of the words that make up the text.

We will use one of the three general purpose sentiment lexicons of the {tidytext} package, namely the AFINN lexicon.

17

Live Coding Example 3 </>
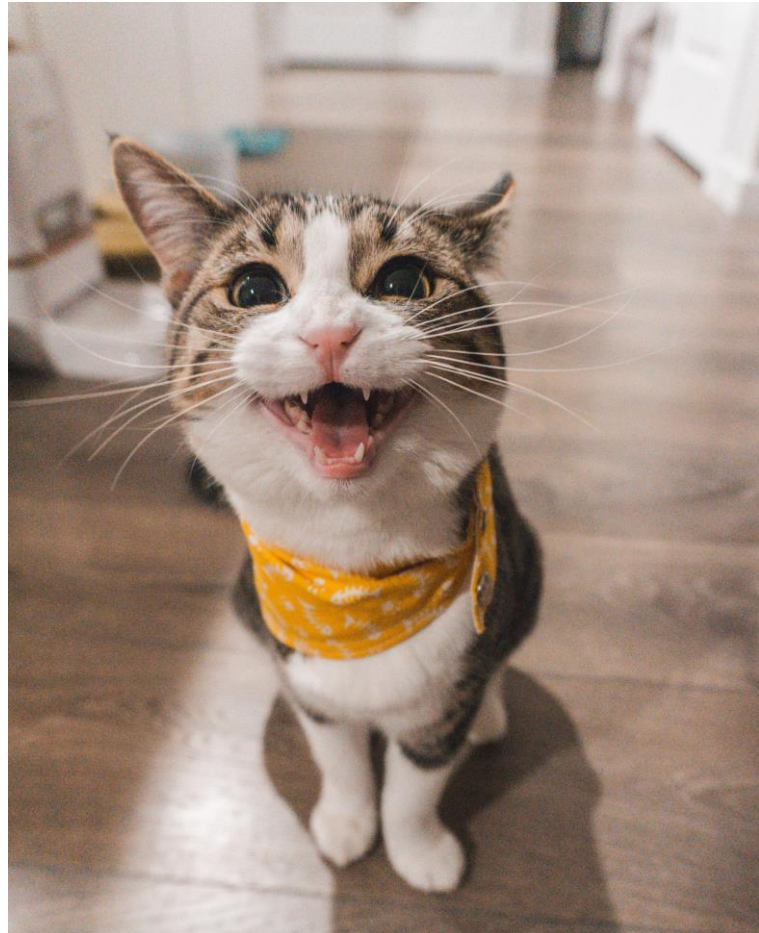
1. Use {tidytext} and create the data frame "afinn" of the AFINN sentiment lexicon and inspect it.

2. Inner join the AFINN sentiment lexicon to "tidy_script" from Example 2 and calculate the total sentiment score per movie per line.

3. Attach the sentiment scores to the original star wars script dataset. What do you think is the most negative script line from all movies?

4. Who is the most negative character of all movies?

5. Visualise the sentiment score changes line by line for each movie.

Live Coding Example 3

```r
# Use {tidytext} and create the data frame "afinn" of the AFINN sentiment
# lexicon and inspect it.
afinn <- get_sentiments("afinn")

# Inner join the AFINN sentiment lexicon to tidy_script from Example 2
# and calculate the total sentiment per movie per line
sentiment_script <- tidy_script %>%
  inner_join(afinn) %>%
  group_by(movie, line) %>%
  mutate(sentiment = sum(value)) %>%
  ungroup() %>%
  select(-word, -value) %>%
  distinct()
```

Live Coding Example 3

Some sample extracts from AFINN

word	value
breathtaking	5
hurrah	5
outstanding	5
superb	5
thrilled	5
amazing	4
awesome	4
brilliant	4
ecstatic	4
euphoric	4
exuberant	4
fabulous	4
fantastic	4
fun	4
funnier	4
funny	4
godsend	4
heavenly	4

word	value
unsure	-1
urgent	-1
verdict	-1
verdicts	-1
vociferous	-1
waste	-1
wavering	-1
widowed	-1
worn	-1
some kind	0
aboard	1
absorbed	1
accept	1
accepted	1
accepting	1
accepts	1
achievable	1
active	1

Total sentiment per movie per line

line	movie	title	character	sentiment
1	IV	A New Hope	THREEPIO	-6
2	IV	A New Hope	THREEPIO	-2
3	IV	A New Hope	THREEPIO	-1
5	IV	A New Hope	THREEPIO	1
11	IV	A New Hope	IMPERIAL OFFICER	-2
15	IV	A New Hope	VADER	1
17	IV	A New Hope	TROOPER	-2
18	IV	A New Hope	THREEPIO	-2
19	IV	A New Hope	THREEPIO	-3
21	IV	A New Hope	THREEPIO	-2
23	IV	A New Hope	CAPTAIN	-2
24	IV	A New Hope	THREEPIO	-2
25	IV	A New Hope	THREEPIO	1
33	IV	A New Hope	BIGGS	1
34	IV	A New Hope	LUKE	-1
35	IV	A New Hope	DEAK	-1

Live Coding Example 3

```r
# Attach the sentiment scores to the original star wars script dataset.
# What do you think is the most negative script line from all movies?
df <- df %>%
  inner_join(sentiment_script)


# Who is the most negative character of all movies?
res <- df %>%
  group_by(character) %>%
  summarise(total_sentiment = sum(sentiment)) %>%
  ungroup() %>%
  arrange(total_sentiment)
```
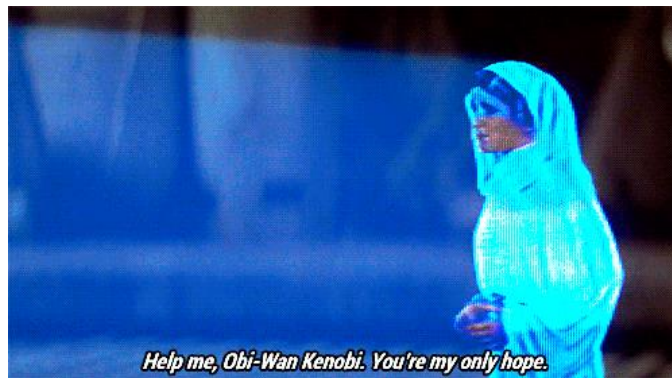
Live Coding Example 3

Sentiment score per script line per movie

	line	movie	title	character	dialogue	length	ncap	nexcl	nquest	nword	sentiment
1	1	IV	A New Hope	THREEPIO	Did you hear that? They've shut down the main reactor. W...	103	4	1	1	20	-6
2	2	IV	A New Hope	THREEPIO	We're doomed!	13	1	1	0	3	-2
3	3	IV	A New Hope	THREEPIO	There'll be no escape for the Princess this time.	49	2	0	0	10	-1
4	5	IV	A New Hope	THREEPIO	I should have known better than to trust the logic of a half-s...	104	1	0	0	17	-1

And the award for the most negative script line from all movies goes to…

And the award for the most negative character goes to…

Live Coding Example 3

```r
# Visualise the sentiment score changes line by line for each movie
df %>%
  ggplot(aes(line, sentiment, fill = movie)) +
  geom_col(show.legend = FALSE) +
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank()) +
  facet_wrap(~movie, ncol = 1)
```

Live Coding Example 3
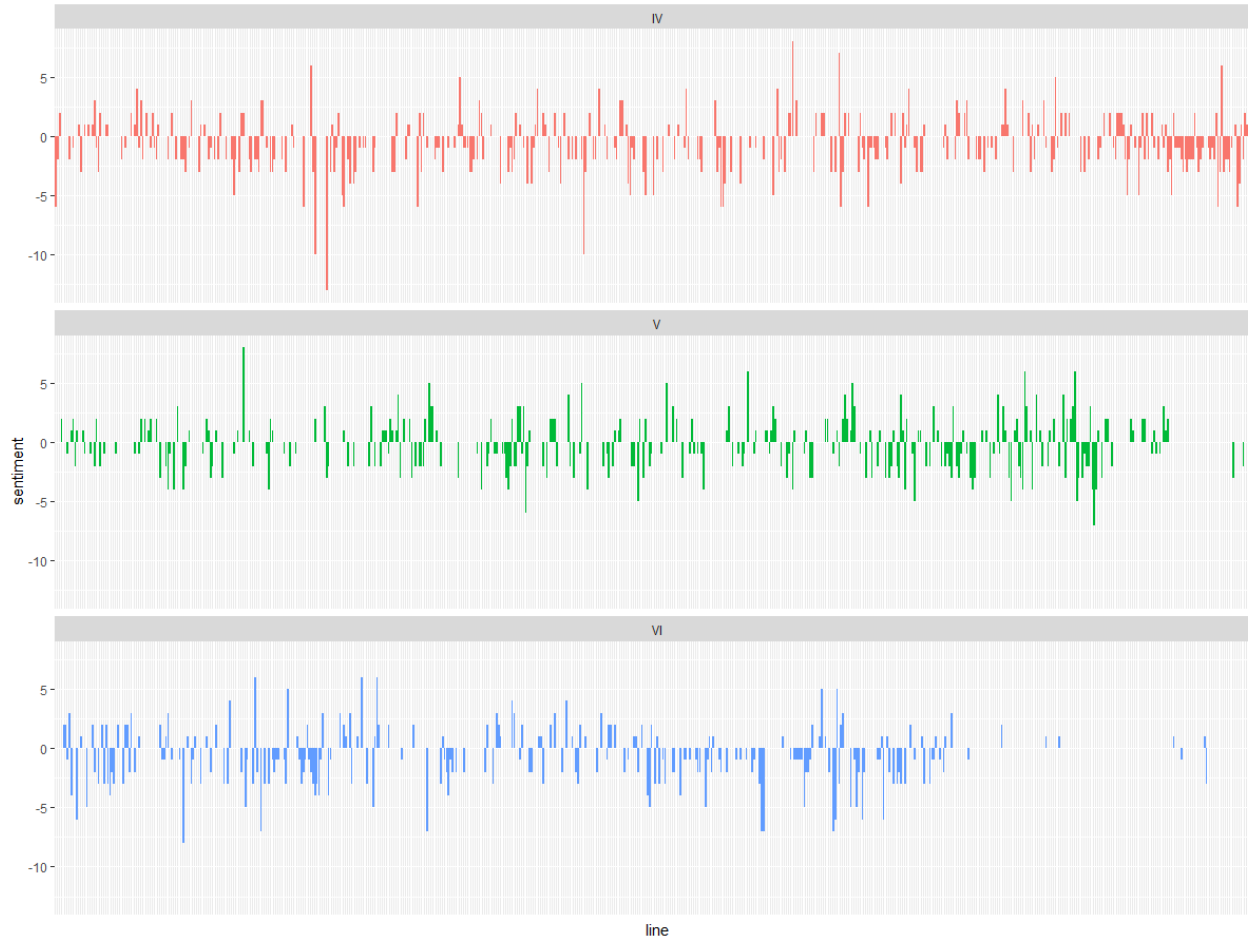
Term Frequency - Inverse Document Frequency

Q: Lets say we have a document, how could we quantify what it is about?

A: Look at and analyse the words that make up the document!

Term Frequency (TF) how frequently does a term appear in the document

Inverse Document Frequency (IDF) adjust the weight of commonly used terms (such as "the") by decreasing the word's importance

Combining the two measures by multiplying them results in a TF-IDF score which reflects the frequency of a term adjusted for how rarely it is used.

Live Coding Example 4
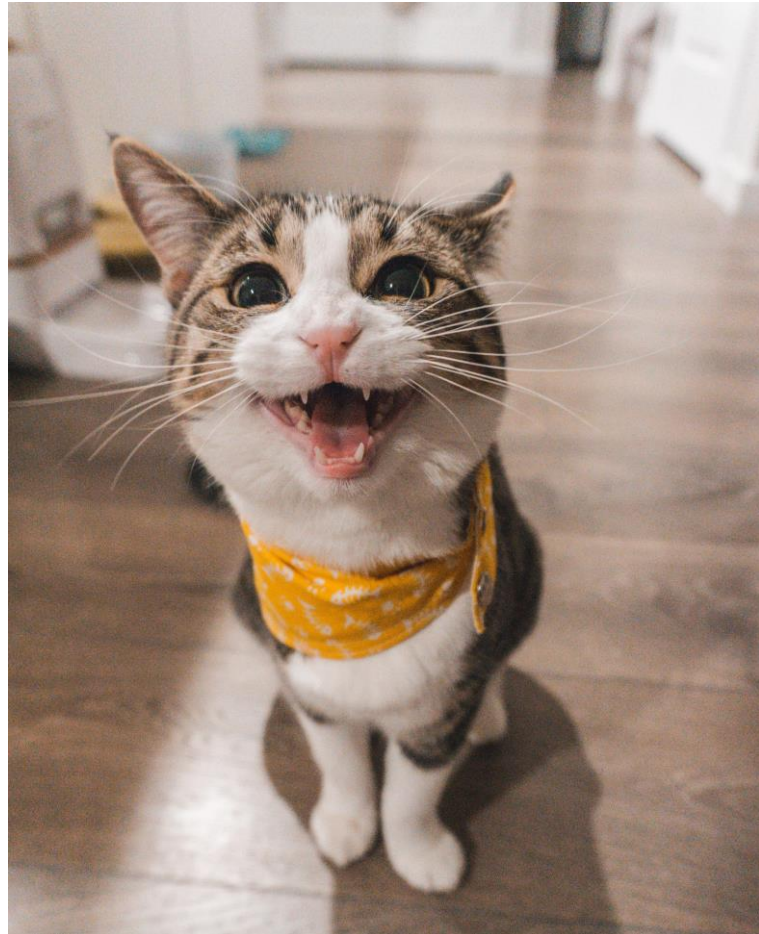
1. Use {tidytext} to tokenize the star wars scripts, where a token is a single word to create a one-token-per-row data frame. Also remove summary columns. Then count the frequency of each word for each move and apply the TF-IDF function of {tidytext} and extract the top 10 words per movie.

Live Coding Example 4

```r
# Use {tidytext} to tokenize the star wars scripts, where a token is a single
# word to create a one-token-per-row data frame. Also remove summary columns.
# Then count the frequency of each word for each move and apply the
# TF-IDF function of {tidytext} and extract the top 10 words per movie
tf_idf_script <- df %>%
  select(-length, -ncap, -nexcl, -nquest, -nword) %>%
  unnest_tokens(output = word, input = dialogue) %>%
  count(movie, word, sort = TRUE) %>%
  bind_tf_idf(word, movie, n) %>%
  ungroup() %>%
  group_by(movie) %>%
  top_n(10) %>%
  arrange(movie, desc(tf_idf))
```

Live Coding Example 4

movie	word	n	tf	idf	tf_idf
IV	kenobi	19	0.0016775561	1.0986123	0.0018429837
IV	uncle	15	0.0013243864	1.0986123	0.0014549871
IV	biggs	10	0.0008829242	1.0986123	0.0009699914
IV	red	26	0.0022956030	0.4054651	0.0009307869
IV	plans	9	0.0007946318	1.0986123	0.0008729923
IV	alderaan	20	0.0017658485	0.4054651	0.0007159899
IV	aboard	7	0.0006180470	1.0986123	0.0006789940
IV	detention	7	0.0006180470	1.0986123	0.0006789940
IV	minutes	7	0.0006180470	1.0986123	0.0006789940
IV	academy	6	0.0005297545	1.0986123	0.0005819949
IV	level	6	0.0005297545	1.0986123	0.0005819949
IV	season	6	0.0005297545	1.0986123	0.0005819949
IV	senate	6	0.0005297545	1.0986123	0.0005819949
IV	year	6	0.0005297545	1.0986123	0.0005819949

Other resources – Text mining with R

Get the book online at: https://www.tidytextmining.com/

Thank you to our sponsors and partners!