Harry Potter 1-3. Explorative analysis of movies' transcription

Olga Bystrova Nikita Login

```
#install.packages("tidyverse")
#install.packages("dplyr")
#install.packages("magrittr")
#install.packages("scales")
#install.packages("RColorBrewer")
#install.packages("ggsci")
#install.packages("ggthemes")
#install.packages("lubridate")
#install.packages("viridis")
#install.packages("ggrepel")
#install.packages("reshape")
#install.packages("gridExtra")
#install.packages("tm")
#install.packages("SnowballC")
#install.packages("wordcloud")
#install.packages("NLP")
#install.packages("widyr")
#install.packages("wordcloud2")
#install.packages("tidytext")
#install.packages("janeaustenr")
#install.packages("htmlwidgets")
#install.packages("klaR")
#install.packages("Rtsne")
#install.packages("data.table")
#install.packages("IRdisplay")
```

Introduction and data description

In this project we explore the first three movies related to the Harry Potter novel: Harry Potter and the Philosopher of Stone, Harry Potter and the Chamber of Secrets, Harry Potter and the Prisoner of the Azkaban.

Hypothesis:

Harry Potter should be one of the most common words / bigrams during three movies The proportion of positive words decreases from film to film Speech patterns of positive and negative characters form distinct separate clusters Lines of the main characters (Harry, Ron and Hermione) are easily distinguishable from the rest

The main analysis is done with the use of the R programming language. It is shown to be a very effective language to deal with statistical research. As for the specific libraries, we use tidyverse, tm, nlp and wordcloud2. All these packages allow us not only to manipulate data but also to visualize extracted knowledge points. Let us go to the dataset description. For every movie transcript there are two informative columns:

'Character' and 'Sentence'. The first column shows the name of a character that said the sentence and the second shows the sentence itself. For example:

Character	Sentence
Hagrid	*You are a wizard, Harry*

Table 1: Dataset Example

To analyse data through a concrete movie perspective, we add the third column to the dataset with the title of the film. Overall, we have 1587 sentences with 79 characters. After that we made some preprocessing steps to analyse data: we removed punctuation and stop-words. The final step would be to lowercase all text.

For sentiment evaluation there is an extra database with a dictionary for evaluating the opinion or emotion in text named "Bing". This database contains almost 7,000 popular words with assigned sentiment: positive or negative.

```
options(warn = -23) # ignore all warnings
options(scipen = 10000)
options(repr.plot.width = 14.0, repr.plot.height = 10.0)

suppressMessages(library(tidyverse))
suppressMessages(library(dplyr))
#library(magrittr)
suppressMessages(library(scales)) # visualisation
suppressMessages(library(RColorBrewer)) # color visualisation
suppressMessages(library(ggsci))
suppressMessages(library(ggthemes))
suppressMessages(library(lubridate)) # date ant time management
suppressMessages(library(viridis)) # color maps
suppressMessages(library(ggrepel))
suppressMessages(library(reshape))
suppressMessages(library(gridExtra))
suppressMessages(library(tm)) # text mining
suppressMessages(library(SnowballC)) #snowball stemmer
suppressMessages(library(wordcloud))
suppressMessages(library(NLP))
suppressMessages(library(widyr))
suppressMessages(library(wordcloud2))
suppressMessages(library(tidytext))
suppressMessages(library(janeaustenr))
suppressMessages(library(htmlwidgets))

annotate <- ggplot2::annotate

theme_michau <- theme(legend.position = "bottom", legend.direction = "horizontal", axis.text = element_
plot.caption = element_text(color = "gray65", size = 12.4), legend.text = element_text(size = 16, colou
axis.title = element_text(size = 16.7, face = "bold", color = "gray25"), legend.title = element_text(si
axis.line = element_line(size = 0.4), plot.title = element_text(size = 21.9, face = "bold", colour = "g
panel.grid.major = element_line(colour = "gray80", size = 0.15), plot.subtitle = element_text(size = 16
strip.text = element_text(size = 16.7, face = "bold"), panel.grid.minor = element_line(size = 0))
```

```r
#Input all the data from films 1-3.
Script1 <- read.csv("data/harry-potter-dataset/Harry Potter 1.csv", row.names = NULL, sep = ";", encodi
Script2 <- read.csv("data/harry-potter-dataset/Harry Potter 2.csv", row.names = NULL, sep = ";", encodi
Script3 <- read.csv("data/harry-potter-dataset/Harry Potter 3.csv", row.names = NULL, sep = ";", encodi

names(Script3) <- c("Character", "Sentence")

Script1$Character <- as.character(str_trim(Script1$Character, side = "both"))
Script2$Character <- as.character(str_trim(Script2$Character, side = "both"))
Script3$Character <- as.character(str_trim(Script3$Character, side = "both"))

Script1$Part <- "Sorcerer's Stone"
Script2$Part <- "Chamber of Secrets"
Script3$Part <- "Prisoner of Azkaban"

Script <- rbind(Script1, Script2, Script3)

Script$Part <- factor(Script$Part, levels=c("Prisoner of Azkaban", "Chamber of Secrets", "Sorcerer's St
Script$Character <- str_to_title(Script$Character)

Script <- Script %>%
  mutate(Character = case_when(Character %in% c("Dumbledore") ~ "Dumbledore",
                               Character %in% c("Mcgonagall") ~ "McGonagall",
                               Character %in% c("Hagrid") ~ "Hagrid",
                               Character %in% c("Petunia", "Aunt Petunia") ~ "Aunt Petunia",
                               Character %in% c("Dudley") ~ "Dudley",
                               Character %in% c("Vernon") ~ "Vernon",
                               Character %in% c("Harry") ~ "Harry",
                               Character %in% c("Snake") ~ "Snake",
                               Character %in% c("Someone") ~ "Someone",
                               Character %in% c("Barkeep Tom") ~ "Barkeep Tom",
                               Character %in% c("Man", "Boy", "Boy 1", "Boy 2") ~ "Man/Boy",
                               Character %in% c("Witch") ~ "Witch",
                               Character %in% c("Quirrell") ~ "Quirrell",
                               Character %in% c("Goblin") ~ "Goblin",
                               Character %in% c("Griphook") ~ "Griphook",
                               Character %in% c("Ollivander") ~ "Ollivander",
                               Character %in% c("Trainmaster") ~ "Trainmaster",
                               Character %in% c("Mrs. Weasley") ~ "Mrs. Weasley",
                               Character %in% c("George") ~ "George",
                               Character %in% c("Fred") ~ "Fred",
                               Character %in% c("Ginny") ~ "Ginny",
                               Character %in% c("Ron") ~ "Ron",
                               Character %in% c("Woman", "Girl") ~ "Girl/Woman",
                               Character %in% c("Hermione", "Hermoine") ~ "Hermione",
                               Character %in% c("Neville") ~ "Neville",
                               Character %in% c("Malfoy", "Draco") ~ "Draco Malfoy",
                               Character %in% c("Sorting Hat") ~ "Sorting Hat",
                               Character %in% c("Seamus") ~ "Seamus",
                               Character %in% c("Percy") ~ "Percy",
                               Character %in% c("Sir Nicholas") ~ "Sir Nicholas",
                               Character %in% c("Man In Paint") ~ "Man In Paint",
                               Character %in% c("Fat Lady") ~ "Fat Lady",
```

```
                        Character %in% c("Snape") ~ "Severus Snape",
                        Character %in% c("Dean") ~ "Dean",
                        Character %in% c("Madam Hooch") ~ "Madam Hooch",
                        Character %in% c("Filch") ~ "Filch",
                        Character %in% c("All", "All 3") ~ "Crowd",
                        Character %in% c("Lee Jordan", "Lee  Jordan") ~ "Lee Jordan",
                        Character %in% c("Gryffindors") ~ "Gryffindors",
                        Character %in% c("Flint") ~ "Flint",
                        Character %in% c("Firenze") ~ "Firenze",
                        Character %in% c("Voldemort") ~ "Voldemort",
                        Character %in% c("Students", "Student", "Class") ~ "Student",
                        Character %in% c("Crowd") ~ "Crowd",
                        Character %in% c("Uncle Vernon") ~ "Uncle Vernon",
                        Character %in% c("Dobby") ~ "Dobby",
                        Character %in% c("Aunt Petunia & Dudley") ~ "Aunt Petunia & Dudley",
                        Character %in% c("Mr. Weasley") ~ "Mr. Weasley",
                        Character %in% c("Fred, George, Ron") ~ "Fred, George, Ron",
                        Character %in% c("Fred, George, Ron, Harry") ~ "Fred, George, Ron, Harry"
                        Character %in% c("Lucius Malfoy") ~ "Lucius Malfoy",
                        Character %in% c("Photographer") ~ "Photographer",
                        Character %in% c("Lockhart", "Gilderoy Lockhart") ~ "Gilderoy Lockhart",
                        Character %in% c("Harry And Ron") ~ "Harry And Ron",
                        Character %in% c("Professor Sprout") ~ "Professor Sprout",
                        Character %in% c("Penelope Clearwater") ~ "Penelope Clearwater",
                        Character %in% c("Colin") ~ "Colin",
                        Character %in% c("Cornish Pixies") ~ "Cornish Pixies",
                        Character %in% c("Wood", "Oliver") ~ "Oliver Wood",
                        Character %in% c("Voice") ~ "Voice",
                        Character %in% c("Lupin") ~ "Lupin",
                        Character %in% c("Picture") ~ "Picture",
                        Character %in% c("Slytherins") ~ "Slytherins",
                        Character %in% c("Madam Pomfrey") ~ "Madam Pomfrey",
                        Character %in% c("Moaning Myrtle") ~ "Moaning Myrtle",
                        Character %in% c("Justin Finch-Fletchley") ~ "Justin Finch-Fletchley",
                        Character %in% c("Crabbe") ~ "Crabbe",
                        Character %in% c("Diary") ~ "Diary",
                        Character %in% c("Tom Riddle", "Tom") ~ "Tom Riddle",
                        Character %in% c("Harry-Ron-Hermione") ~ "Harry & Ron and Hermione",
                        Character %in% c("Fudge") ~ "Cornelius Fudge",
                        Character %in% c("Aragog") ~ "Aragog",
                        Character %in% c("Aunt Marge") ~ "Aunt Marge",
                        Character %in% c("Stan Shunpike") ~ "Stan Shunpike",
                        Character %in% c("Vendor") ~ "Vendor",
                        Character %in% c("Housekeeper") ~ "Housekeeper",
                        Character %in% c("Trelawney") ~ "Sybilla Trelawney",
                        Character %in% c("Bem") ~ "Bem",
                        Character %in% c("Pansy Parkinson") ~ "Pansy Parkinson",
                        Character %in% c("Parvati") ~ "Parvati Patil",
                        Character %in% c("Teacher") ~ "Teacher",
                        Character %in% c("Fred & George") ~ "Fred and George",
                        Character %in% c("Madam Rosmerta") ~ "Madam Rosmerta",
                        Character %in% c("Shrunken Head", "Shrunken Head 1", "Shrunken Head 2") ~
                        Character %in% c("Goyle") ~ "Goyle",
```

```
                              Character %in% c("Sirius") ~ "Sirius Black",
                              Character %in% c("Pettigrew") ~ "Peter Pettigrew"))


Bing <- get_sentiments("bing")

firstup <- function(x) {
  substr(x, 1, 1) <- toupper(substr(x, 1, 1))
  x
}

Bing$sentiment <- firstup(Bing$sentiment)
```

Word Frequency and Sentiment analysis

The first obvious thing to do is to check which character said more sentences than others. From Figure 1 you can see that Harry Potter has around 1 000 lines during three movies which is about twice bigger than Ron Weasley, who took the second place. The first three places belong to the three main characters. It is quite surprising that Lucius Malfoy is in the top 15, because he appeared only in the second book and was not a main character at all.

```
Char_Dial <- data.frame(table(Script$Character, Script$Part))
Char_Dial %>%
  arrange(desc(Freq)) %>%
  filter(Var1 %in% c("Harry", "Ron", "Hermione", "Hagrid", "Dumbledore", "Lupin", "McGonagall", "Draco M
                     "Severus Snape", "Lucius Malfoy", "Mrs. Weasley", "Tom Riddle", "Sirius Black", "De
  ggplot(., aes(reorder(Var1, +Freq), Freq, fill = Var2))+
  geom_bar(stat = "identity", width = 0.65)+
  scale_fill_uchicago()+
  coord_flip()+
  guides(fill = guide_legend(title.position = "top", reverse = T))+
  labs(title = "Characters with the most sentences",
       subtitle = "Top 15, by part of a movie series", fill = "Part of a movie series",
       x = "Character", y = "Number of sentence")+
  theme_minimal()+
  theme_michau+
  theme(legend.title.align = 0.5, legend.position = "right", legend.direction = "vertical")
```

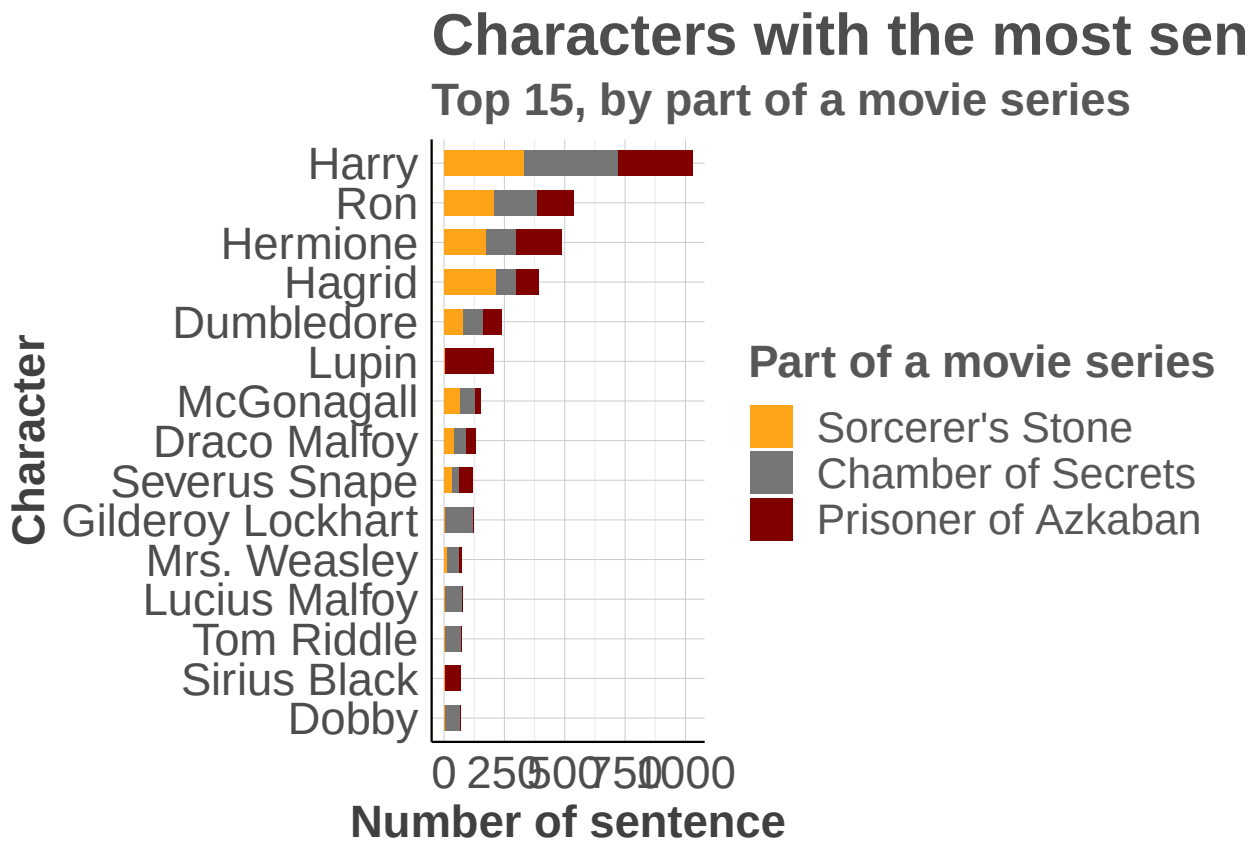

Figure 1: Top 15 characters

There are several characters that appear only in one of the three movies: Lupin, Gilderoy Lockhart, Top Riddle, Sirius Black. Another surprising fact is that Lupin, who appears only in the third movie, has more lines than Professor McGonagall who appears in all three parts of Harry Potter movies.

The next step would be to see what words, bigrams, and trigrams are the most popular in our data. To do so, a Term Document Matrix was made. And by using word cloud library we managed to see the most popular tokens. In both Pictures 2-3 we can see that the name of the main character absolutely dominates with more that 250 appearings (It is quite interesting that for 455 minutes of movie footage it appears 272 times, which is one time per 1.4 minute). Harry's last name is also in the top 15, it is in 10th place. For wordcloud illustration we took only those words that appear more than 10 times. From these two illustrations we may also conclude that the language of the first three books is quite simple and prove that these books are for children.

```
tm <- Corpus(VectorSource(Script$Sentence))
tm <- tm_map(tm, content_transformer(tolower))
tm <- tm_map(tm, removeNumbers)
tm <- tm_map(tm, removeWords, stopwords("english"))
tm <- tm_map(tm, removePunctuation)
tm <- tm_map(tm, stripWhitespace)
tdm <- TermDocumentMatrix(tm)

tdm <- as.matrix(tdm)
tdm <- sort(rowSums(tdm), decreasing = T)
tdm <- data.frame(Word = names(tdm), Number = tdm)
```

```
wc <- tdm %>%
  filter(Number > 8) %>%
  select(Word, Number) %>%
  wordcloud2(., color = alpha("coral3", seq(0.9,0.2,-0.002)), backgroundColor = "white", size = 0.9)

saveWidget(wc,'wordcloud2.html',selfcontained = F)
set.seed(111)
IRdisplay::display_html('<iframe src="wordcloud2.html" width=99% height=500></iframe>')
```

Figure 2: WordCloud of the most frequent unigrams in all three movies

```
tdm %>%
  arrange(desc(Number)) %>%
  slice(1:15) %>%
ggplot(., aes(reorder(Word, +Number), Number))+
  geom_bar(stat = "identity", width = 0.65, fill = "coral3", alpha = 0.85)+
  coord_flip()+
  labs(title = "Most popular words in the first 3 movies",
       subtitle = "Top 15 (without stopwords)",
       x = "Word", y = "Number of sentences")+
  theme_minimal()+
  theme_michau
```

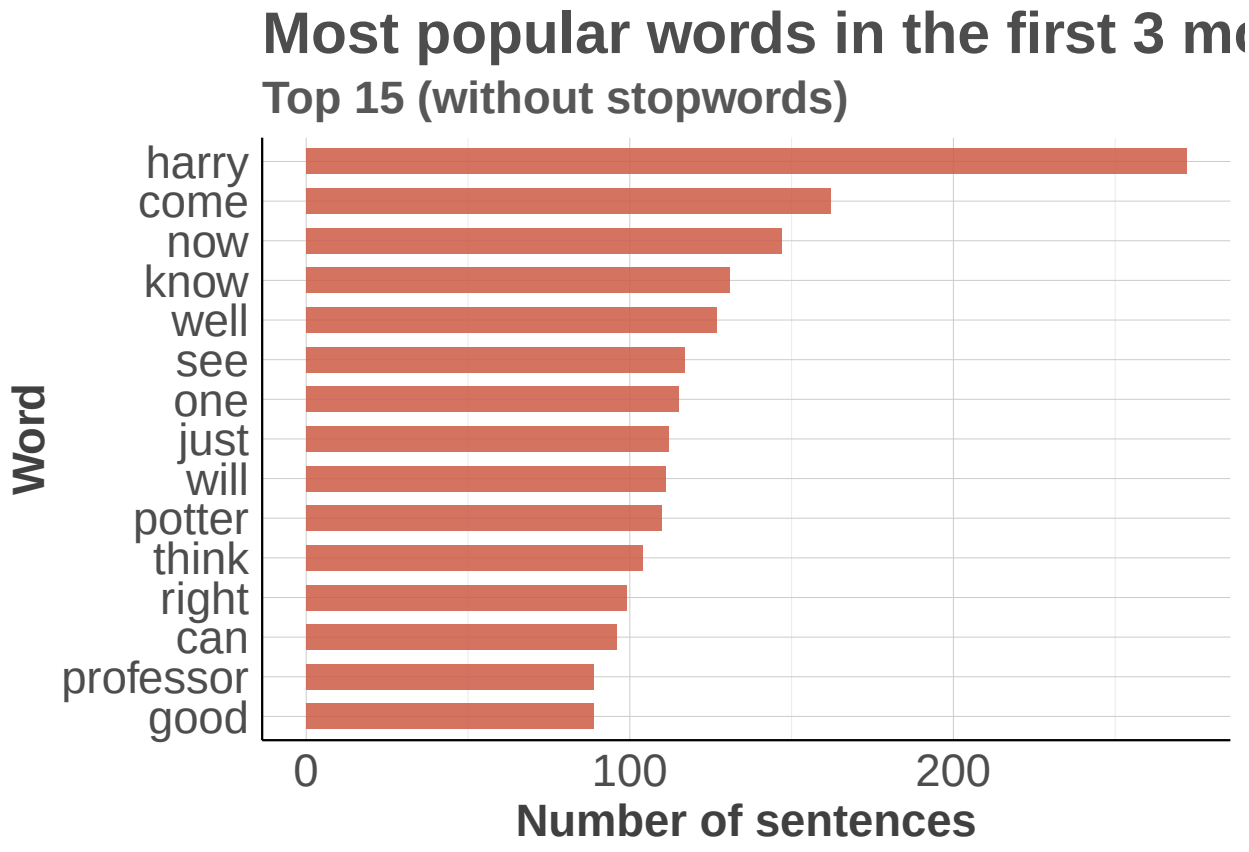

Figure 3: Top 15 words that appear in the first three Harry Potter movies.

```r
tm1 <- Corpus(VectorSource(Script1$Sentence))
tm1 <- tm_map(tm1, content_transformer(tolower))
tm1 <- tm_map(tm1, removeNumbers)
tm1 <- tm_map(tm1, removeWords, stopwords("english"))
tm1 <- tm_map(tm1, removePunctuation)
tm1 <- tm_map(tm1, stripWhitespace)
tdm1 <- TermDocumentMatrix(tm1)

tdm1 <- as.matrix(tdm1)
tdm1 <- sort(rowSums(tdm1), decreasing = T)
tdm1 <- data.frame(Word = names(tdm1), Number = tdm1)
tdm1$Part <- "Sorcerer's Stone"

tm2 <- Corpus(VectorSource(Script2$Sentence))
tm2 <- tm_map(tm2, content_transformer(tolower))
tm2 <- tm_map(tm2, removeNumbers)
tm2 <- tm_map(tm2, removeWords, stopwords("english"))
tm2 <- tm_map(tm2, removePunctuation)
tm2 <- tm_map(tm2, stripWhitespace)
tdm2 <- TermDocumentMatrix(tm2)

tdm2 <- as.matrix(tdm2)
tdm2 <- sort(rowSums(tdm2), decreasing = T)
tdm2 <- data.frame(Word = names(tdm2), Number = tdm2)
tdm2$Part <- "Chamber of Secrets"

tm3 <- Corpus(VectorSource(Script3$Sentence))
tm3 <- tm_map(tm3, content_transformer(tolower))
tm3 <- tm_map(tm3, removeNumbers)
tm3 <- tm_map(tm3, removeWords, stopwords("english"))
tm3 <- tm_map(tm3, removePunctuation)
tm3 <- tm_map(tm3, stripWhitespace)
tdm3 <- TermDocumentMatrix(tm3)

tdm3 <- as.matrix(tdm3)
tdm3 <- sort(rowSums(tdm3), decreasing = T)
tdm3 <- data.frame(Word = names(tdm3), Number = tdm3)
tdm3$Part <- "Prisoner of Azkaban"

tdm_all <- rbind(tdm1, tdm2, tdm3)
tdm_all$Part <- factor(tdm_all$Part, levels=c("Sorcerer's Stone", "Chamber of Secrets", "Prisoner of Azk

tdm_all %>%
  filter(Word %in% c("harry", "come", "now", "know", "well", "see", "one", "just", "will",
                     "potter", "think", "right", "can", "professor", "good")) %>%
ggplot(., aes(Word, Number, fill = Part))+
  facet_wrap(.~Part)+
  geom_bar(stat = "identity", width = 0.65, alpha = 0.85)+
  scale_x_discrete(limits = c("good", "professor", "can",  "right", "think", "potter", "will",
                              "just", "one", "see", "well", "know", "now", "come","harry"))+
  scale_fill_manual(values = c("#ffa319", "#767676", "#800000"))+
  coord_flip()+
  labs(title = "Top 15 most popular words in the first 3 movies",
```

```
      subtitle = "by part of a movie series",
      x = "Word", y = "Number of sentence per movie")+
  theme_minimal()+
  theme_michau+
  theme(legend.position = "none")
```

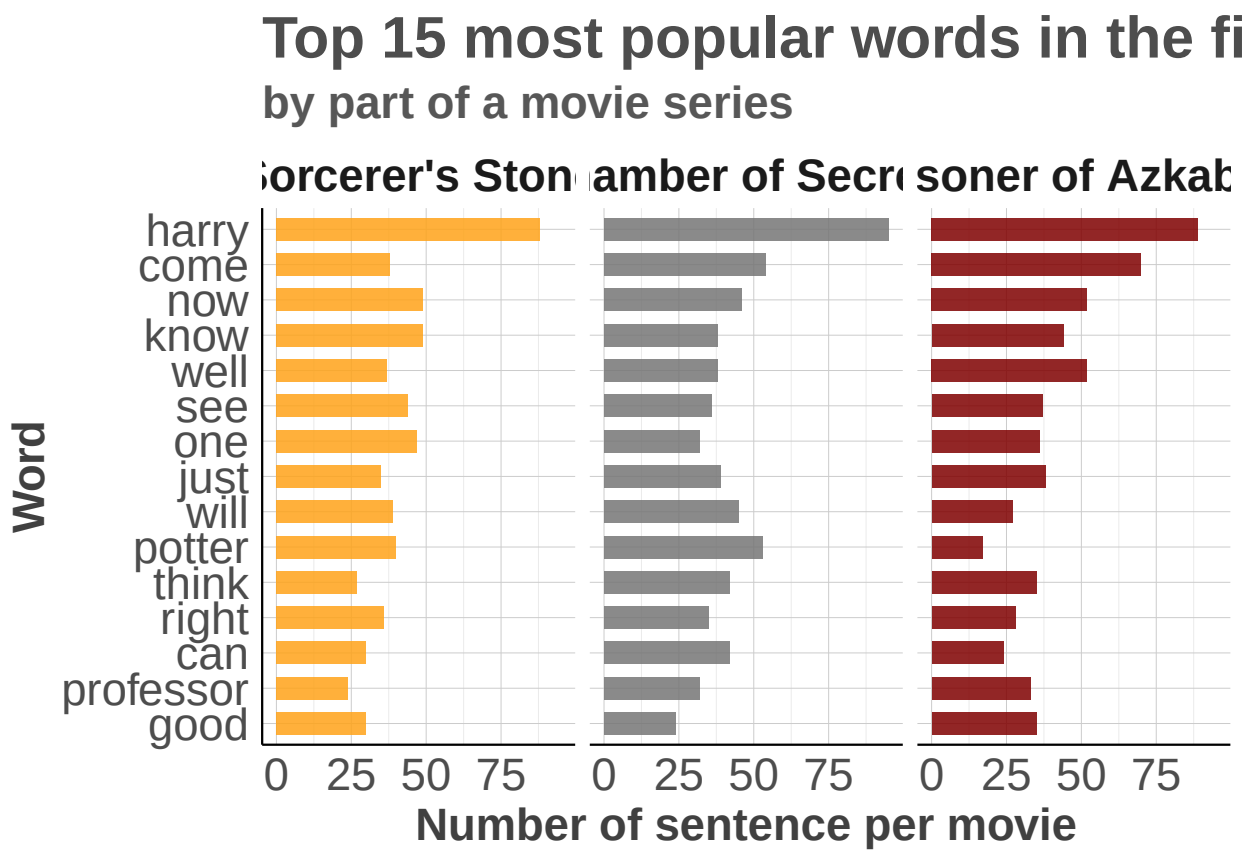

Figure 4: Top 15 words that appear in the first three Harry Potter movies (by part of a movie series).

If we check the proportions of these top 15 words according to the movie part, we would notice an almost balanced distribution. If we exclure Harry's name, we would notice that there is no word that orruces more than 70 times in one movie.

Top 15 bigrams and trigrams are presented in Figure 5. First of all, we have proven our hypothesis number 1 that Harry Potter is the most common bigram and its parts are at the top of unigrams as well. Even its version of 'mr potter' appears on the 7th place). Secondly, there are also only two names in the top list: professor Dumbledore (although his actual name is Albus Percival Wulfric Brian Dumbledore) and Sirius Black

```
Script$Sentence <- as.character(Script$Sentence)

Script %>%
  unnest_tokens(output = word, input = Sentence, token = "ngrams", n = 2) %>%
  filter(is.na(word)==F) %>%
  separate(word, c("word1", "word2"), sep = " ") %>%
  filter(!word1 %in% c("on", "in", "the", "be", "are", "i", "you", "is", "to", "a", "has", "of", "it",
  filter(!word2 %in% c("on", "in", "the", "be", "are", "i", "you", "is", "to", "a", "has", "of", "it",
  unite(word,word1, word2, sep = " ") %>%
```

```
  count(word, sort = T) %>%
  slice(1:15) %>%
ggplot(., aes(reorder(word, +n), n))+
  geom_bar(stat = "identity", width = 0.65, fill = "#a1d76a", alpha = 0.85)+
  coord_flip()+
  labs(title = "Most popular bigrams in the first 3 movies",
       subtitle = "Top 15",
       x = "Bigram", y = "Frequency")+
  theme_minimal()+
  theme_michau
```

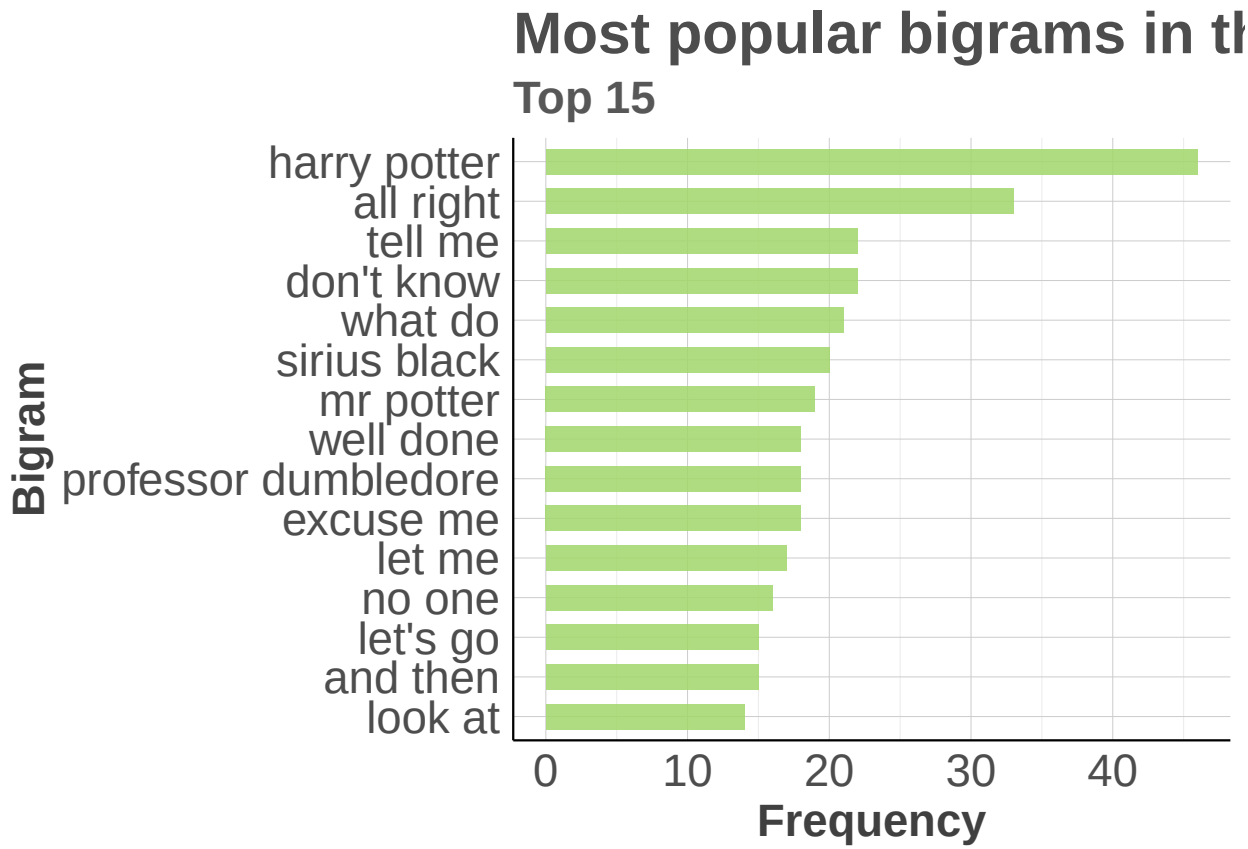

Figure 5: Top 15 bigrams in the first three movies.

As for trigrams, there are several phrases with 'com on' in it. Another interesting fact is that neither Crab nor Goyle appear in the top unigrams but the trigram 'Crab and Goyle' is at the 6th place among trigrams. Based on that we can see how inseparable these characters are. 'Heir of slytherin' and 'chamber of secrets' are at the top of trigrams although the main action related to them happens only in the second movie.

```
Script %>%
  unnest_tokens(output = word, input = Sentence, token = "ngrams", n = 3) %>%
  filter(is.na(word)==F) %>%
  separate(word, c("word1", "word2", "word3"), sep = " ") %>%
  filter(!word1 %in% c("on", "in", "the", "be", "are", "i", "you", "is", "to", "a", "has", "of", "it",
  filter(!word2 %in% c("you", "we", "the")) %>%
  filter(!word3 %in% c("on", "in", "the", "be", "are", "i", "you", "is", "to", "a", "has", "of", "it",
  unite(word,word1, word2, word3, sep = " ") %>%
```

```
  count(word, sort = T) %>%
  slice(1:15) %>%
ggplot(., aes(reorder(word, +n), n))+
  geom_bar(stat = "identity", width = 0.65, fill = "#a1d76a", alpha = 0.85)+
  coord_flip()+
  labs(title = "Most popular trigrams in the first 3 movies",
       subtitle = "Top 15",
       x = "Trigram", y = "Frequency")+
  theme_minimal()+
  theme_michau
```

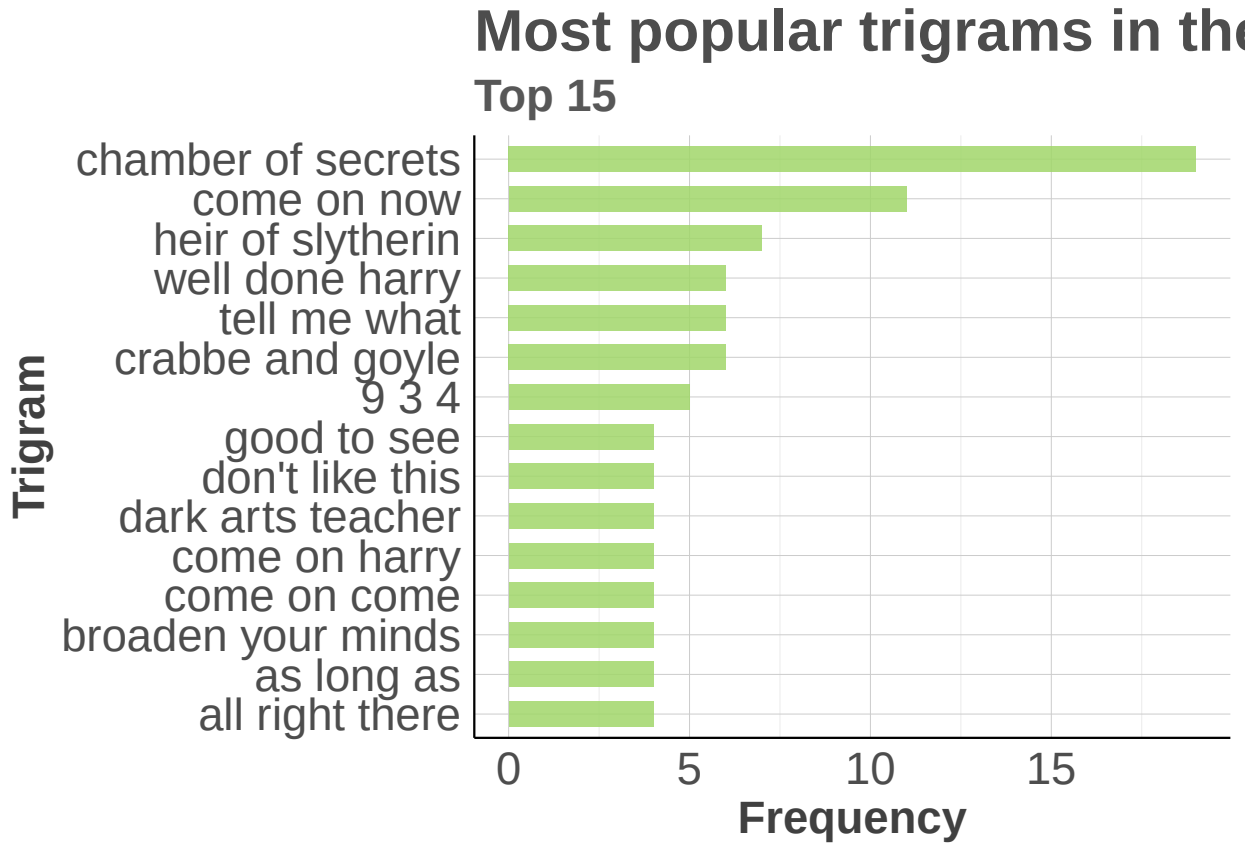

Figure 6: Top 15 trigrams in the first three movies.

The next step in our analysis would be sentiment analysis of given words. With the use of Bing dictionary (it contains around 7 000 words with their sentiment alignment). Our hypothesis was that the proportion of positive words decreases from film to film. If you watch all Harry Potter movies one by one you would notice that the first movies are lighter and more naive than the last one. Let us check this for the first three movies.

From Figure 7 we can see that this hypothesis is not true for all movies. The decrease in the proportion of positive elements is noticeable between the first and the second movie, but in the third movie the situation reverts itself. In Prisoner of Azkaban there are more than 50% words that are positive according to the Bing dictionary. Of course, the full dynamic patterns should be analysed based on all eight movies together.

```
Sentiment <- Script %>%
  unnest_tokens(output = word, input = Sentence) %>%
  left_join(Bing, "word") %>%
```

```
  filter(is.na(sentiment)==F)

Sentiment %>%
  group_by(word, sentiment) %>%
  summarise(count = n(), .groups = 'drop') %>%
  arrange(desc(count)) %>%
  slice(1:20) %>%
ggplot(., aes(reorder(word, +count), count, fill = sentiment))+
  geom_bar(stat = "identity", width = 0.65, alpha = 0.9)+
  scale_fill_brewer(palette = "Set1")+
  coord_flip()+
  labs(title = "Most popular words with assigned sentiment",
       subtitle = "Top 20",
       x = "Word", y = "Frequency", fill = "Sentiment")+
  guides(fill = guide_legend(reverse = T))+
  theme_minimal()+
  theme_michau+
  theme(legend.title.align = 0.5, legend.position = "right", legend.direction = "vertical")
```

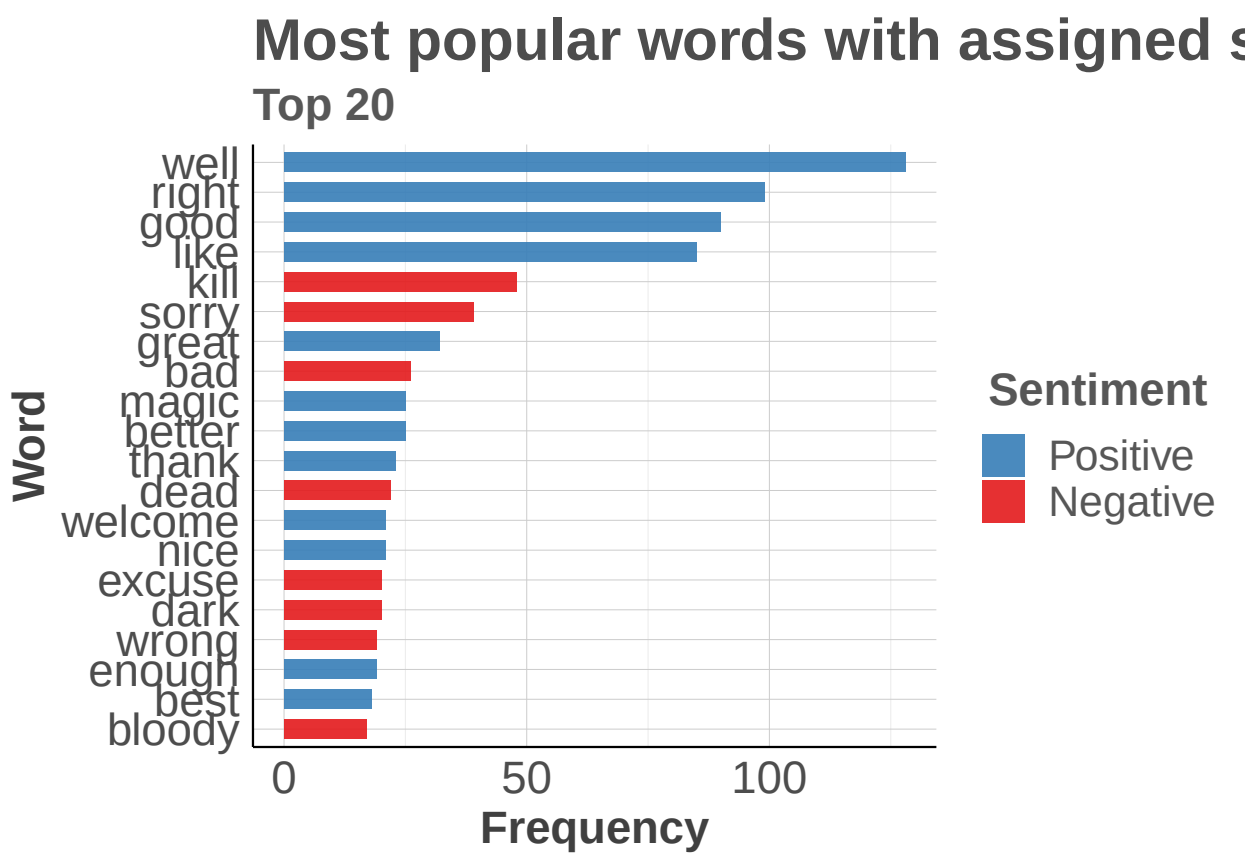

Figure 7: Most popular words with positive and negative sentiment

```
Sentiment %>%
  group_by(Part, sentiment) %>%
  summarise(count = n(), .groups = 'drop') %>%
ggplot(., aes(Part, count, fill = sentiment))+
  geom_bar(stat = "identity", position = "fill", width = 0.7, alpha = 0.9)+
```

```
  scale_fill_brewer(palette = "Set1")+
  scale_y_continuous(labels = scales::percent)+
  coord_flip()+
  labs(title = "Share of words with positive and negative sentiment",
       subtitle = "by part of a movie series", fill = "Sentiment",
       x = "Part of a movie series", y = "Share")+
  guides(fill = guide_legend(reverse = T))+
  theme_minimal()+
  theme_michau+
  theme(legend.title.align = 0.5, legend.position = "right", legend.direction = "vertical")
```

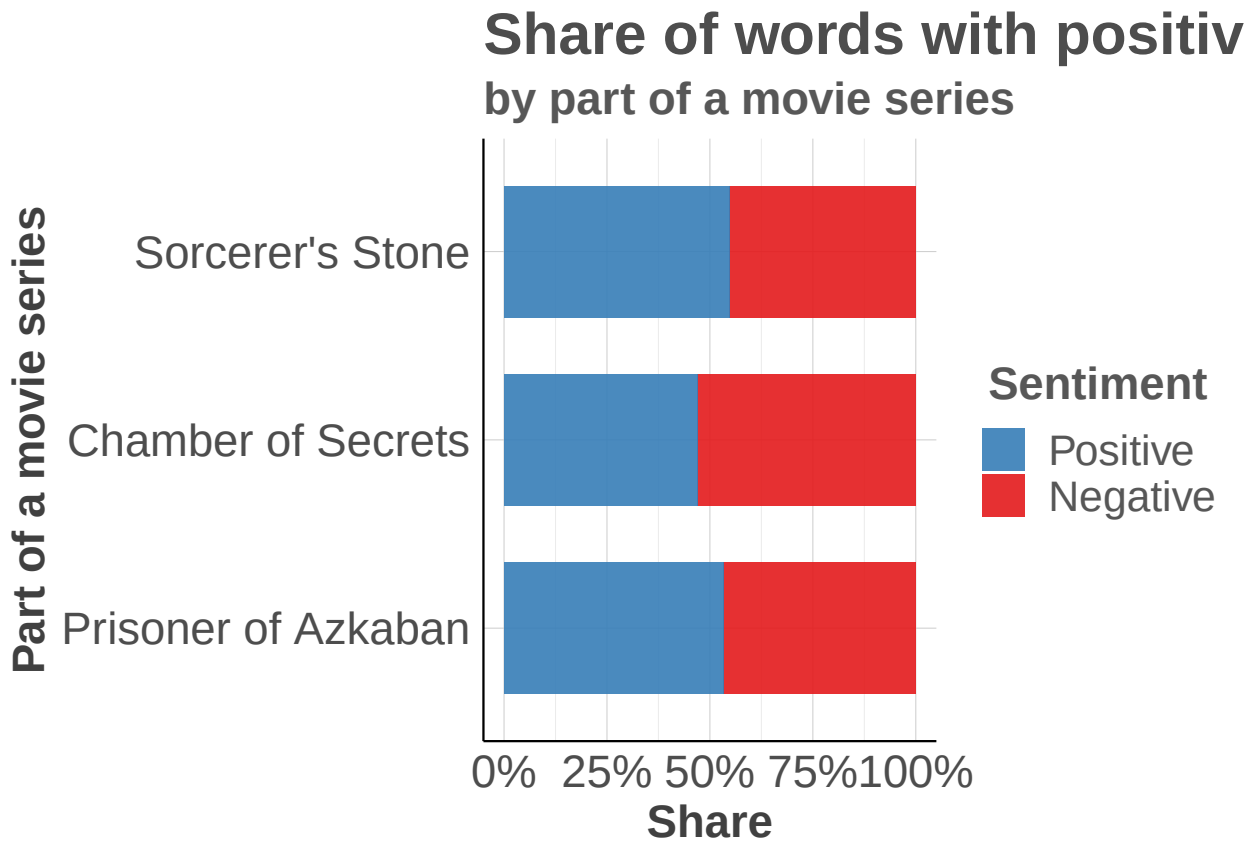

Figure 8: Proportions of words with positive and negative sentiment.

The final step in our analysis is to check the distribution of positive and negative words in characters' perspective.

```
Sentiment %>%
  filter(Character %in% c("Harry", "Ron", "Hermione", "Hagrid", "Dumbledore", "Lupin", "McGonagall", "D
                          "Severus Snape", "Lucius Malfoy", "Mrs. Weasley", "Tom Riddle", "Sirius Black
  group_by(Character, sentiment) %>%
  summarise(count = n(), .groups = 'drop') %>%
ggplot(., aes(Character, count, fill = sentiment))+
  geom_bar(stat = "identity", position = "fill", width = 0.6, alpha = 0.9)+
  scale_x_discrete(limits = c("Dobby", "Sirius Black", "Tom Riddle", "Mrs. Weasley", "Lucius Malfoy", "
                              "McGonagall", "Lupin", "Dumbledore", "Hagrid",  "Hermione", "Ron", "Harry
  scale_fill_brewer(palette = "Set1")+
```

```
  scale_y_continuous(labels = scales::percent)+
  coord_flip()+
  labs(title = "Share of words with positive and negative sentiment",
       subtitle = "by character (top 15 characters with the most sentences)", fill = "Sentiment",
       x = "Character", y = "Share")+
  guides(fill = guide_legend(reverse = T))+
  theme_minimal()+
  theme_michau+
  theme(legend.title.align = 0.5, legend.position = "right", legend.direction = "vertical")
```

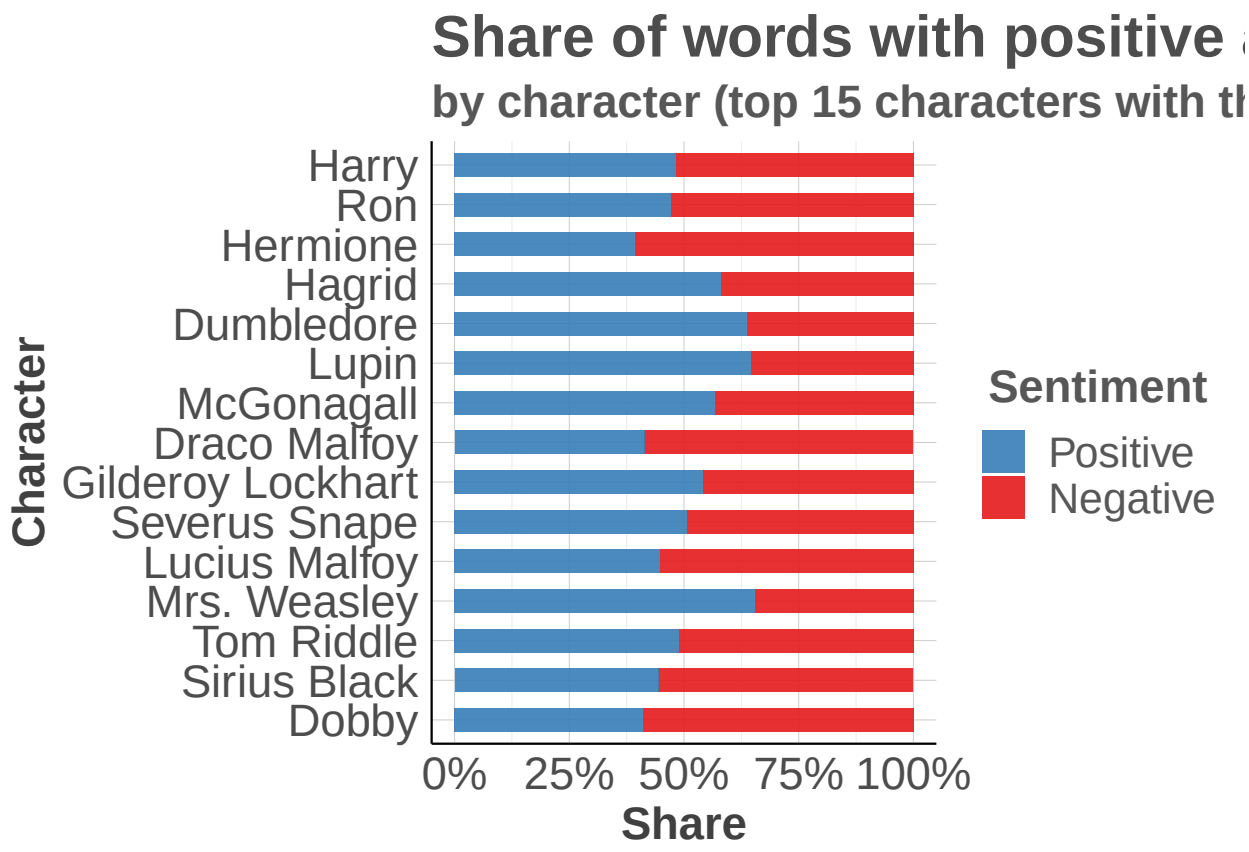

Figure 9: Proportions of positive/negative words according to movie characters.

There are several insights that we can see: 1) Dumbledore, Lupin and Mrs. Weasley (positive characters themselves) have more positive words in their lines than negative, which is not surprising. 2) Draco Malfoy and Lucius Malfoy (negative characters themselves) have more negative words in their vocabulary, which is also not surprising. 3) Hermione and Dobby (positive characters themselves) have more negative words in their sentences than positive. And this is surprising. If for Dobby we could suggest that this happened because he made some mean thing to Harry at first and was a questionable character. But for Hermione, there is no explanation at this point. Finally, Tom Riddle, although he is a negative character (or rather his incarnation in Lord Voldemort) has a sentiment very similar to Harry and to the average, close to an equal share of positive and negative words in his statements.

Character Speech (dis)similarity analysis

In order to analyse (dis)similarities between speech of different characters we created a TF-IDF matrix of all lines from our corpus of three movie scripts. We used implementation of TF-IDF feature extraction algorithm from *tidytext* R package. We've also trained a Nearest Mean Classifier from *klaR* R package to predict which character each line belongs to.

```
suppressMessages(library(klaR))
suppressMessages(library(Rtsne))
```

```
Script$ID <- seq.int(nrow(Script))

sentence_words <- Script %>%
  unnest_tokens(word, Sentence) %>%
  count(ID, word, sort=TRUE) %>%
  ungroup()


sentence_tf_idf <- sentence_words %>%
  bind_tf_idf(word, ID, n)

tfidf_dtm <- sentence_tf_idf %>% cast_dtm(ID, word, tf_idf)

X_tfidf <- as.matrix(tfidf_dtm)
y <- as.factor(Script$Character)

clf <- nm(x=X_tfidf, grouping=y)

y_pred <- predict(clf, newdata=X_tfidf)$class
confusion <- y_pred == y

mean(confusion, na.rm=TRUE)
```

```
## [1] 0.3431433
```

Although the accuracy of Nearest Mean Classifier score beats the majority vote baseline (0.32, the fraction of Harry's lines, see Table 3), it is still fairly low which means that raw TF-IDF scores are far from informative. In order to improve informativity of vector representations of speech lines we used two dimensionality reduction techniques: T-SNE and PCE.

```
set.seed(42)
tsne <- Rtsne(X_tfidf, dims=2, pepplexity=30,
              verbose=TRUE, max_iter=500, check_duplicates=FALSE)
```

```
## Performing PCA
## Consider setting partial_pca=TRUE for large matrices
## Read the 4925 x 50 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 1.87 seconds (sparsity = 0.029767)!
```

```
## Learning embedding...
## Iteration 50: error is 87.723645 (50 iterations in 1.57 seconds)
## Iteration 100: error is 80.198780 (50 iterations in 1.30 seconds)
## Iteration 150: error is 79.123530 (50 iterations in 0.97 seconds)
## Iteration 200: error is 79.094338 (50 iterations in 0.94 seconds)
## Iteration 250: error is 79.054231 (50 iterations in 0.99 seconds)
## Iteration 300: error is 2.104066 (50 iterations in 0.98 seconds)
## Iteration 350: error is 1.636355 (50 iterations in 0.84 seconds)
## Iteration 400: error is 1.410790 (50 iterations in 0.83 seconds)
## Iteration 450: error is 1.279578 (50 iterations in 0.85 seconds)
## Iteration 500: error is 1.195905 (50 iterations in 0.89 seconds)
## Fitting performed in 10.16 seconds.
```

```
colors <- rainbow(length(unique(y)))
names(colors) <- unique(y)
par(mgp=c(2.5,1,0))
plot(tsne$Y, t='n', main="tSNE", xlab="tSNE dimension 1", ylab="tSNE dimension 2", "cex.main"=2, "cex.la
text(tsne$Y, labels=y, col=colors[y])
```

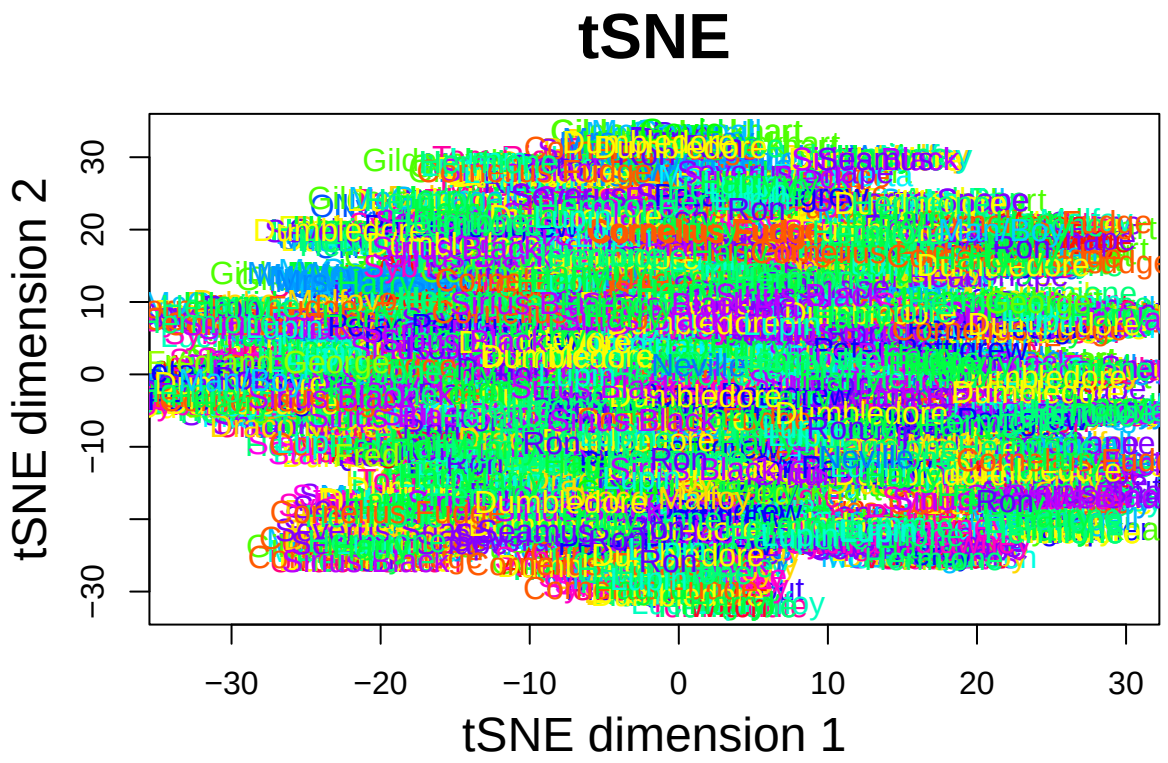

Figure 10: t-SNE plot of lines of all characters

From Figure 10, we can see that t-SNE representation plot of lines of all characters doesn't show anything meaningful. Therefore we decided to keep only characters whose number of lines in concatenation of three movie scripts is no less than of 90% quantile of all character line counts.

```
Char_Dial <- as.tibble(table(Script$Character), .name_repair=~c("Character","n"))
Char_top <- Char_Dial %>%
  filter(n >= 122)
Script_top <- Script %>%
  filter(Character %in% Char_top$Character)
```

```
as.data.frame(quantile(Char_Dial$n, probs=seq(0, 1, 0.1)))
```

```
##      quantile(Char_Dial$n, probs = seq(0, 1, 0.1))
## 0%                                             1.0
## 10%                                            1.0
## 20%                                            1.6
## 30%                                            5.0
## 40%                                            8.4
## 50%                                           19.5
## 60%                                           24.6
## 70%                                           37.1
## 80%                                           52.6
## 90%                                          121.0
## 100%                                        1028.0
```

Table 2: Quantile of character line counts distribution from all three movies

```
Char_top %>%
  mutate(p = n/sum(n))
```

```
## # A tibble: 8 x 3
##   Character        n      p
##   <chr>        <int>  <dbl>
## 1 Draco Malfoy   131 0.0413
## 2 Dumbledore     239 0.0753
## 3 Hagrid         394 0.124
## 4 Harry         1028 0.324
## 5 Hermione       488 0.154
## 6 Lupin          207 0.0652
## 7 McGonagall     152 0.0479
## 8 Ron            536 0.169
```

Table 3: Count and proportion of lines of characters with line count no less than 90% quantile

```
sentence_words <- Script_top %>%
  unnest_tokens(word, Sentence) %>%
  count(ID, word, sort=TRUE) %>%
  ungroup()


sentence_tf_idf <- sentence_words %>%
  bind_tf_idf(word, ID, n)

tfidf_dtm <- sentence_tf_idf %>% cast_dtm(ID, word, tf_idf)
```

```r
X_tfidf <- as.matrix(tfidf_dtm)
char_names <- as.factor(Script_top$Character)

set.seed(42)
tsne <- Rtsne(X_tfidf, dims=2, pepplexity=400,
              verbose=TRUE, max_iter=1000, check_duplicates=FALSE)
```

```
## Performing PCA
## Consider setting partial_pca=TRUE for large matrices
## Read the 3175 x 50 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.94 seconds (sparsity = 0.047611)!
## Learning embedding...
## Iteration 50: error is 82.256149 (50 iterations in 0.59 seconds)
## Iteration 100: error is 78.506561 (50 iterations in 0.57 seconds)
## Iteration 150: error is 77.800373 (50 iterations in 0.47 seconds)
## Iteration 200: error is 77.657386 (50 iterations in 0.49 seconds)
## Iteration 250: error is 77.614670 (50 iterations in 0.45 seconds)
## Iteration 300: error is 1.750629 (50 iterations in 0.40 seconds)
## Iteration 350: error is 1.362648 (50 iterations in 0.42 seconds)
## Iteration 400: error is 1.193181 (50 iterations in 0.43 seconds)
## Iteration 450: error is 1.104410 (50 iterations in 0.43 seconds)
## Iteration 500: error is 1.053303 (50 iterations in 0.44 seconds)
## Iteration 550: error is 1.022765 (50 iterations in 0.44 seconds)
## Iteration 600: error is 1.008311 (50 iterations in 0.44 seconds)
## Iteration 650: error is 0.999049 (50 iterations in 0.45 seconds)
## Iteration 700: error is 0.989833 (50 iterations in 0.44 seconds)
## Iteration 750: error is 0.982409 (50 iterations in 0.45 seconds)
## Iteration 800: error is 0.975778 (50 iterations in 0.46 seconds)
## Iteration 850: error is 0.972261 (50 iterations in 0.47 seconds)
## Iteration 900: error is 0.969250 (50 iterations in 0.50 seconds)
## Iteration 950: error is 0.965273 (50 iterations in 0.49 seconds)
## Iteration 1000: error is 0.960500 (50 iterations in 0.50 seconds)
## Fitting performed in 9.32 seconds.
```

```r
tsne_plot <- data.frame(x1 = tsne$Y[,1], x2=tsne$Y[,2],
                        col=char_names)

ggplot(tsne_plot) + geom_point(aes(x=x1, y=x2, color=col))
```

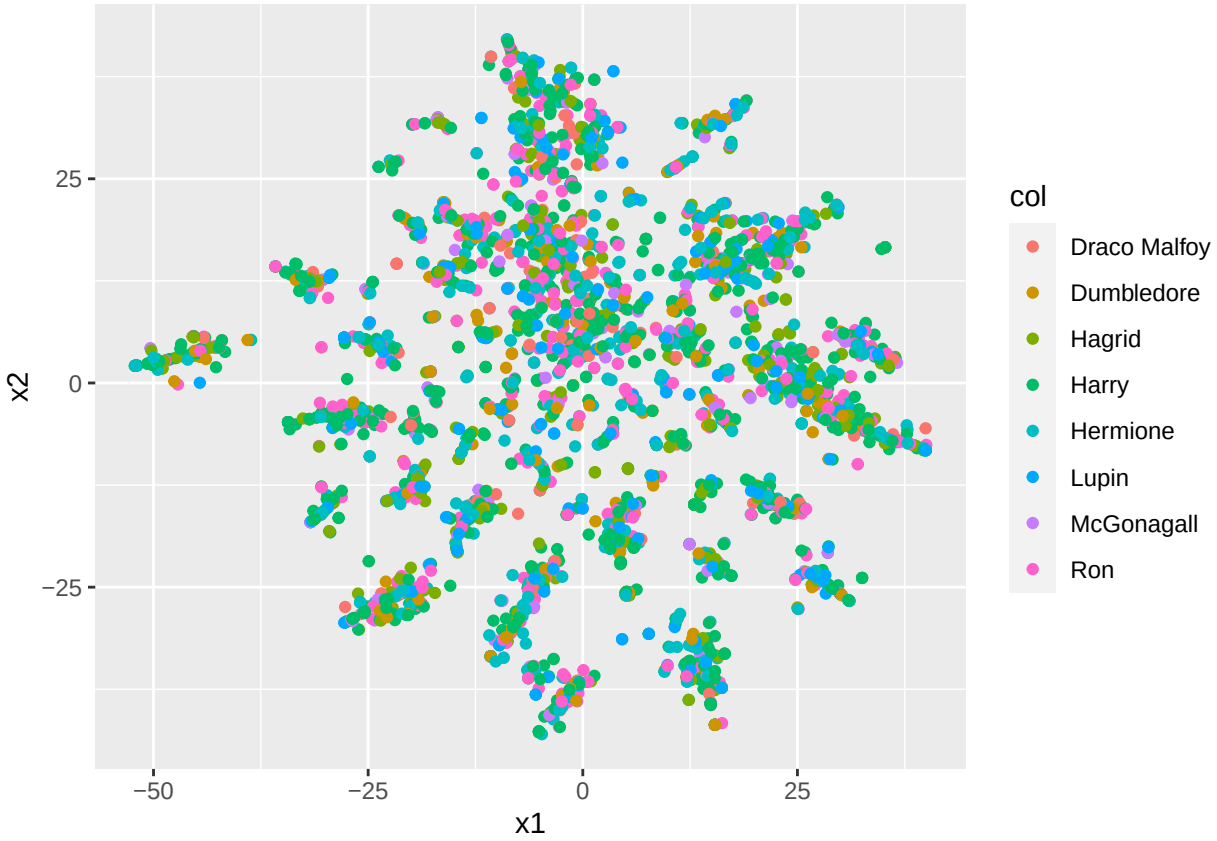

Figure 11: t-SNE plot of lines of characters from 90% quantile

We can see that characters are quite close to each other by TF-IDF if we take into account their single lines. However, this result is in the nature of T-SNE as it tries to preserve local distances between points, sometimes ignoring global ones. In case of TF-IDF scores of single lines (which may: 1.) be very short 2.) highly vary in length) it would be better to use Princinpal Component Analysis for dimensionality reduction and visualization as it is better for preserving global distances (though far worse at preserving local ones).

```
suppressMessages(library('data.table'))
```

```
X_tfidf <- as.data.frame(X_tfidf)
pca <- prcomp(X_tfidf)
var_explained <- pca$sdev^2 / sum(pca$sdev^2)
names(var_explained) <- sprintf("PC%s", seq(1:2545))
var_explained <- bind_rows(var_explained)

# custom function to transpose while preserving names
# taken from https://stackoverflow.com/a/48838799
transpose_df <- function(df) {
  t_df <- data.table::transpose(df)
  colnames(t_df) <- rownames(df)
  rownames(t_df) <- colnames(df)
  t_df <- t_df %>%
    tibble::rownames_to_column(.data = .) %>%
    tibble::as_tibble(.)
  return(t_df)
```

```
}

var_explained <- var_explained %>%
  transpose_df %>%
  dplyr::rename(Component=rowname, var_ratio='1') %>%
  arrange(-var_ratio) %>%
  mutate(cum_sum = cumsum(var_ratio))

var_explained %>% arrange()
```

```
## # A tibble: 2,545 x 3
##    Component var_ratio cum_sum
##    <chr>         <dbl>   <dbl>
##  1 PC1         0.0115   0.0115
##  2 PC2         0.0103   0.0219
##  3 PC3         0.00975  0.0316
##  4 PC4         0.00961  0.0412
##  5 PC5         0.00954  0.0508
##  6 PC6         0.00916  0.0599
##  7 PC7         0.00876  0.0687
##  8 PC8         0.00784  0.0765
##  9 PC9         0.00774  0.0843
## 10 PC10        0.00735  0.0916
## # ... with 2,535 more rows
```

Table 4: Top-10 components of PCA decomposition (by explained variance ratio)

From the PCA represntation of TF-IDF data we selected top components by explained variance ratio, which together explained 95% of variance.

```
X_pca <- as_tibble(predict(pca))
rel_feats <- unique(filter(var_explained, cum_sum<=0.95)$Component)
X_pca_mini <- X_pca[, rel_feats]
```

We also fitted a NM Classifier on this 957 first components of decomposed TF-IDF matrix of lines of characters from 90% quantile of line count distribution

```
clf <- nm(x=X_pca_mini, grouping=char_names)

y_pred <- predict(clf, newdata=X_pca_mini)$class
confusion <- y_pred == char_names

mean(confusion, na.rm=TRUE)
```

```
## [1] 0.4330709
```

We can see that while using PCA and only most 'popular' characters the score is by 10% better than with raw TF-IDF values and all characters (0.34). Compressed data representation derived by PCA also allowed us to use weight-vector-based models. We fitted three logistic regression models to distinguish lines of three main characters (Harry, Hermione and Ron) from the rest of the script lines.

```
y_bin <- as.factor(char_names == "Harry")
xy <- X_pca_mini %>%
  mutate(y=y_bin)
lr_Harry <- glm(y ~ ., data=xy, family="binomial")
#summary(lr_Harry)
```

```
y_bin_pred <- predict(lr_Harry, X_pca_mini) >= 0.5
mean(y_bin_pred == y_bin)
```

```
## [1] 0.7266142
```

```
table(y_bin) / length(y_bin)
```

```
## y_bin
##     FALSE      TRUE
## 0.6762205 0.3237795
```

```
y_bin <- as.factor(char_names == "Hermione")
xy <- X_pca_mini %>%
  mutate(y=y_bin)
lr_Hermione <- glm(y ~ ., data=xy, family="binomial")

y_bin_pred <- predict(lr_Hermione, X_pca_mini) >= 0.5
mean(y_bin_pred == y_bin)
```

```
## [1] 0.8519685
```

```
table(y_bin) / length(y_bin)
```

```
## y_bin
##     FALSE      TRUE
## 0.8462992 0.1537008
```

And with Ron:

```
y_bin <- as.factor(char_names == "Ron")
xy <- X_pca_mini %>%
  mutate(y=y_bin)
lr_Hermione <- glm(y ~ ., data=xy, family="binomial")

y_bin_pred <- predict(lr_Hermione, X_pca_mini) >= 0.5
mean(y_bin_pred == y_bin)
```

```
## [1] 0.8211024
```

```
table(y_bin) / length(y_bin)
```

```
## y_bin
##     FALSE      TRUE
## 0.8311811 0.1688189
```

All three models has shown fairly high accuracy scores (0.72, 0.84 and 0.82 for Harry, Hermione and Ron respectively). Models for Harry and Hermione have beaten the majority vote accuracy baselines (0.67 for Harry and 0.84 for Hermione), while the model for Ron haven't (with the baseline of 0.83).

Conclusion

Overall, we have proven that our first hypothesis is completely true: 'Harry' is the most popular word to appear in the first three movies, 'Harry Potter' is the most common bigram as well.

As for the second hypothesis, it is true only for the first two movies out of three. First and third movies have the advantage of words with a positive character, while the second one with a negative character, but the values in all cases are close to 50%.

Moreover, several interesting insights came up during the analysis: in bigrams and trigrams we can find expressions known from everyday speech, as well as expressions that we only meet in this movie saga; Hermione has more negative words in her lines than positive.

In addition, using TF-IDF representations of character lines and Nearest Mean Classifier we found that it is often quite hard to distinguish characters by their lines in the scripts. The results of T-SNE visualization supported this conclusion, showing that small clusters are formed from lines of mix of different characters and that a line of given character can often be closer to lines from different charcaters than to a given different line from the same character. However, using logistic regression classifier and PCA dimenisonality reduction technique, we found that lines of two main characters - Harry and Hermione - are easily distinguishable from other lines from three movies about "the boy who lived".