Intelligent Systems - NLP Deliverable

Racil Kacem

January 12, 2021

The goal of this document is to analyze the book *Harry Potter and the Order of the Phoenix* which was published on June 21, 2003.

Indeed, it is the Harry Potter book with the highest number of words. We will try to look to the words which are used (total number of words, total number of unique words, characters, word frequency etc.) and find patterns in the text.

Thanks to this analysis, we will see that is possible to know who are the most important characters of the book and answer some interesting questions such as:

- Who is Harry's closest friend ?
- Is this book a negative or positive book ?

Preparation

Check working directory

In this project, we don't have to check the working directory. Indeed, the corpus is hosted online and we'll only need to import it using its link.

Load libraries

```
library(reshape2)
library(NLP)
library(stringr)
library(tm)
library(SnowballC)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(RColorBrewer)
library(syuzhet)
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:NLP':
##
##     annotate
```

```
library(rJava)
.jinit(parameters="-Xmx4g")
library(openNLP)
library(openNLPmodels.en)
```

Load corpus

The text file is hosted online on my GithHub repository.

```
file = "https://github.com/racilk/Harry-Potter-NLP/blob/main/hp.txt?raw=true"
text = readLines(file)
corpus = Corpus(VectorSource(text))
```

We can check the length of the corpus:

```
length(corpus)
```

```
## [1] 9296
```

Create a default term document matrix

```
tdm = TermDocumentMatrix(corpus)
```

```
tdm
```

```
## <<TermDocumentMatrix (terms: 24207, documents: 9296)>>
## Non-/sparse entries: 193250/224835022
## Sparsity           : 100%
## Maximal term length: 95
## Weighting          : term frequency (tf)
```

```
length(dimnames(tdm)$Terms)
```

```
## [1] 24207
```

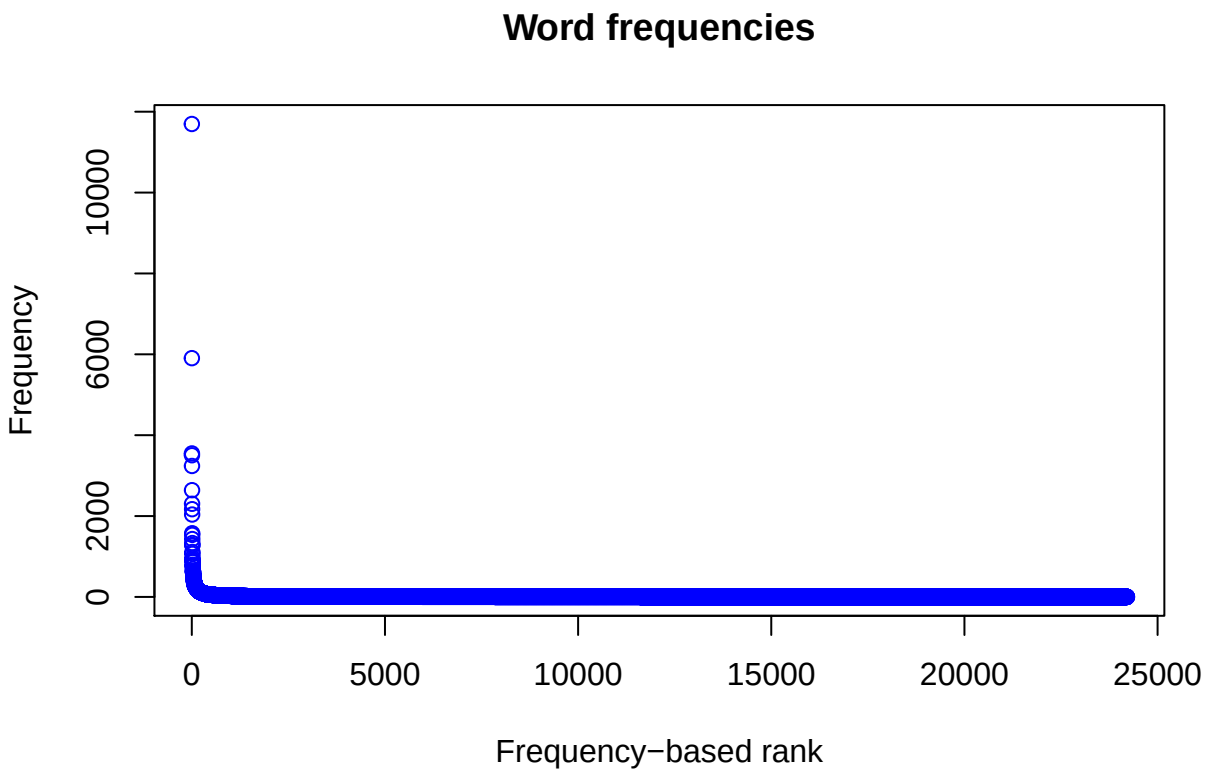
There are 24207 words in this book. Let's sum the content of all terms (i.e., rows) and see the frequency of the terms just shown.

```
freq=rowSums(as.matrix(tdm))
head(freq,10)
```

```
##      and    harry    order  phoenix   potter      the  rowling  chapter
##     5905     2640       65       21       62    11696        1       43
##      one demented
##      412        1
```

If we plot those frequencies ordered, we can see how the corpus behaves following Zipf's law.

```
plot(sort(freq, decreasing = T),col="blue",main="Word frequencies", xlab="Frequency-based rank", ylab =
```



We can now print the ten most frequent terms. Given that we haven't done any transformation on the text file yet, it will mostly be stopwords.

```
tail(sort(freq),n=10)
```

```
##  with  that   you   had harry   his  said   was   and   the
##  1575  2038  2170  2304  2640  3239  3504  3549  5905 11696
```

An other measure of the vocabulary richness of the book will be the number of words which appear once:

```
sum(freq == 1)
```

```
## [1] 12600
```

We can check that 12600 terms out of 24207 only appear once in our corpus.

Create a TDM after applying transformations to the corpus

We will now apply transformations to the text in order to delete the stopwords, the numbers, the punctuation etc. First, we can print the different transformations and the stopwords.

```
getTransformations()
```

```
## [1] "removeNumbers"     "removePunctuation" "removeWords"
## [4] "stemDocument"      "stripWhitespace"
```

```
stopwords()
```

```
##   [1] "i"          "me"         "my"         "myself"     "we"
##   [6] "our"        "ours"       "ourselves"  "you"        "your"
##  [11] "yours"      "yourself"   "yourselves" "he"         "him"
##  [16] "his"        "himself"    "she"        "her"        "hers"
##  [21] "herself"    "it"         "its"        "itself"     "they"
##  [26] "them"       "their"      "theirs"     "themselves" "what"
##  [31] "which"      "who"        "whom"       "this"       "that"
##  [36] "these"      "those"      "am"         "is"         "are"
##  [41] "was"        "were"       "be"         "been"       "being"
##  [46] "have"       "has"        "had"        "having"     "do"
##  [51] "does"       "did"        "doing"      "would"      "should"
##  [56] "could"      "ought"      "i'm"        "you're"     "he's"
##  [61] "she's"      "it's"       "we're"      "they're"    "i've"
##  [66] "you've"     "we've"      "they've"    "i'd"        "you'd"
##  [71] "he'd"       "she'd"      "we'd"       "they'd"     "i'll"
##  [76] "you'll"     "he'll"      "she'll"     "we'll"      "they'll"
##  [81] "isn't"      "aren't"     "wasn't"     "weren't"    "hasn't"
##  [86] "haven't"    "hadn't"     "doesn't"    "don't"      "didn't"
##  [91] "won't"      "wouldn't"   "shan't"     "shouldn't"  "can't"
##  [96] "cannot"     "couldn't"   "mustn't"    "let's"      "that's"
## [101] "who's"      "what's"     "here's"     "there's"    "when's"
## [106] "where's"    "why's"      "how's"      "a"          "an"
## [111] "the"        "and"        "but"        "if"         "or"
## [116] "because"    "as"         "until"      "while"      "of"
## [121] "at"         "by"         "for"        "with"       "about"
## [126] "against"    "between"    "into"       "through"    "during"
## [131] "before"     "after"      "above"      "below"      "to"
## [136] "from"       "up"         "down"       "in"         "out"
## [141] "on"         "off"        "over"       "under"      "again"
## [146] "further"    "then"       "once"       "here"       "there"
## [151] "when"       "where"      "why"        "how"        "all"
## [156] "any"        "both"       "each"       "few"        "more"
## [161] "most"       "other"      "some"       "such"       "no"
## [166] "nor"        "not"        "only"       "own"        "same"
## [171] "so"         "than"       "too"        "very"
```

Let's apply the different transformations:

```
corpus = tm_map(corpus, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(corpus, content_transformer(tolower)):
## transformation drops documents
```

```
corpus = tm_map(corpus,removeWords,stopwords())
```

```
## Warning in tm_map.SimpleCorpus(corpus, removeWords, stopwords()): transformation
## drops documents
```

```
corpus = tm_map(corpus,removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(corpus, removePunctuation): transformation drops
## documents
```

```
corpus = tm_map(corpus,removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(corpus, removeNumbers): transformation drops
## documents
```

```
corpus = tm_map(corpus,stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(corpus, stripWhitespace): transformation drops
## documents
```

Let's inspect the beginning of the book to be sure that the transformations have correctly been applied.

```
inspect(corpus[1:50])
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 50
##
##  [1] harry potter order phoenix
##  [2] j k rowling
##  [3]
##  [4]  chapter one
##  [5] dudley demented
##  [6]  hottest day summer far drawing close drowsy silence lay large square houses privet drive cars
##  [7]  skinny blackhaired bespectacled boy pinched slightly unhealthy look someone grown lot short spa
##  [8]  whole harry thought congratulated idea hiding perhaps comfortable lying hot hard earth hand nob
##  [9]  almost though thought fluttered open window vernon dursley harrys uncle suddenly spoke
## [10]  glad see boys stopped trying butt anyway
## [11]  know said aunt petunia unconcerned house
## [12]  uncle vernon grunted
## [13]  watching news said scathingly like know really normal boy cares news dudley got clue going doub
## [14]  vernon shh said aunt petunia windows open
## [15]  oh yes sorry dear
## [16]  dursleys fell silent harry listened jingle fruit n bran breakfast cereal watched mrs figg batty
## [17]  dudders tea
## [18]  polkisses said aunt petunia fondly got many little friends popular
## [19]  harry suppressed snort difficulty dursleys really astonishingly stupid son dudley swallowed dir
## [20]  opening notes music heralded seven oclock news reached harrys ears stomach turned perhaps tonig
## [21]  record numbers stranded holidaymakers fill airports spanish baggagehandlers strike reaches seco
## [22]  give em lifelong siesta snarled uncle vernon end newsreaders sentence matter outside flowerbed
```

```
## [23]  let long slow breath stared brilliant blue sky every day summer tension expectation temporary
## [24]  kept listening just case small clue recognised really muggles unexplained disappearance perhaps
## [25]  harry closed eyes now blazing evening sky newsreader said finally bungy budgie found novel way
## [26]  harry opened eyes reached waterskiing budgerigars nothing else worth hearing rolled cautiously
## [27]  moved two inches several things happened quick succession
## [28]  loud echoing crack broke sleepy silence like gunshot cat streaked parked car flew sight shriek
## [29]  harry felt though head split two eyes streaming swayed trying focus street spot source noise ba
## [30]  put away uncle vernon snarled harrys ear now anyone sees
## [31]  get harry gasped seconds struggled harry pulling uncles sausagelike fingers left hand right ma
## [32]  panting harry fell forwards hydrangea bush straightened stared around sign caused loud cracking
## [33]  lovely evening shouted uncle vernon waving mrs number seven opposite glaring behind net curtai
## [34]  continued grin horrible manic way curious neighbours disappeared various windows grin became gr
## [35]  harry moved steps closer taking care stop just short point uncle vernons outstretched hands res
## [36]  devil mean boy asked uncle vernon croaky voice trembled fury
## [37]  mean said harry coldly kept looking left right street still hoping see person made cracking noi
## [38]  making racket like starting pistol right outside
## [39]  make noise said harry firmly
## [40]  aunt petunias thin horsy face now appeared beside uncle vernons wide purple one looked livid
## [41]  lurking window
## [42]  yes yes good point petunia window boy
## [43]  listening news said harry resigned voice
## [44]  aunt uncle exchanged looks outrage
## [45]  listening news
## [46]  well changes every day see said harry
## [47]  clever boy want know really give listening news tosh know perfectly well lot
## [48]  careful vernon breathed aunt petunia uncle vernon lowered voice harry barely hear lot get news
## [49]  know said harry
## [50]  dursleys goggled seconds aunt petunia said nasty little liar lowered voice harry lipread next w
```

Then, we build the TDM of the transformed corpus in order to display the ten most frequent words (it shouldn't be stopwords now).

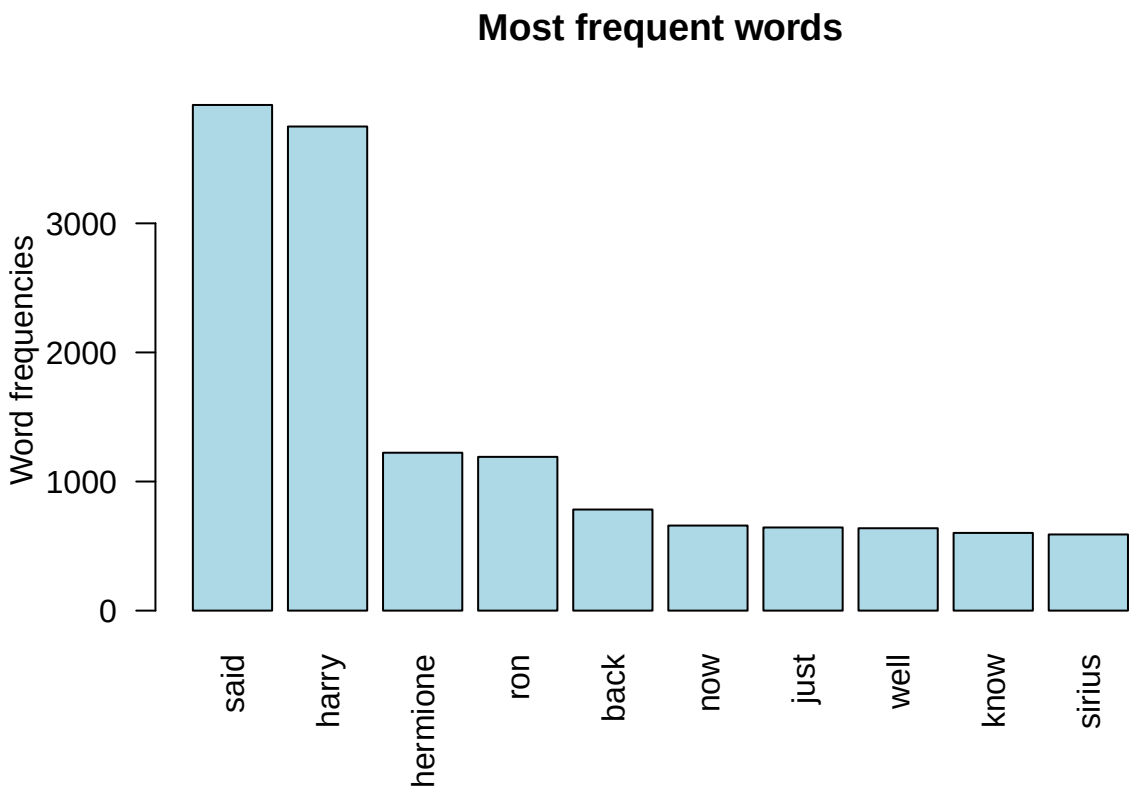
```
corpus_dtm = TermDocumentMatrix(corpus)
dtm_m = as.matrix(corpus_dtm)
dtm_v = sort(rowSums(dtm_m),decreasing=TRUE)
dtm_d = data.frame(word = names(dtm_v),freq=dtm_v)
head(dtm_d, 10)
```

```
##              word freq
## said         said 3917
## harry       harry 3750
## hermione hermione 1223
## ron           ron 1191
## back         back  783
## now           now  659
## just         just  644
## well         well  638
## know         know  602
## sirius     sirius  590
```

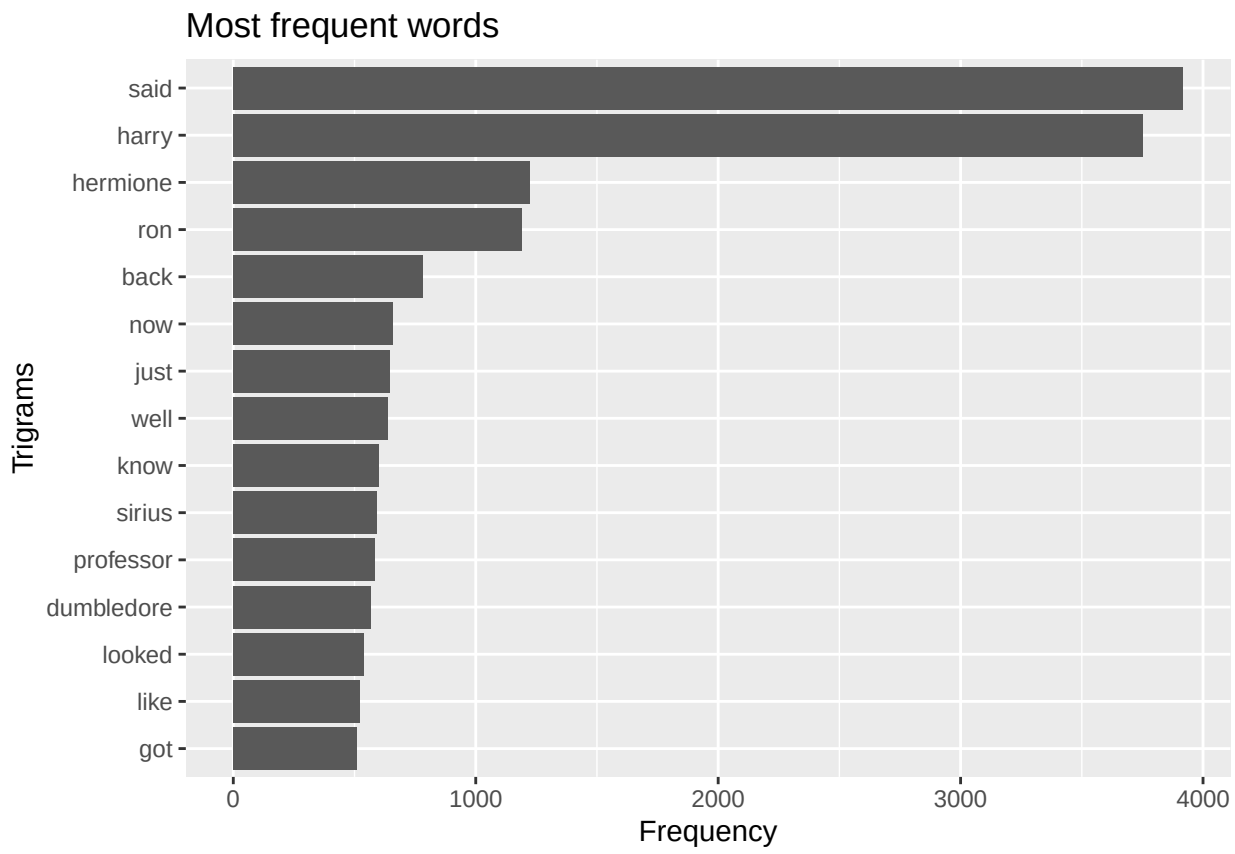
Given that there is a lot of dialogues in this book, the most frequent word is the verb "said".

It is also possible to plot the most frequent words in another way:

```
barplot(dtm_d[1:10,]$freq, las = 2, names.arg = dtm_d[1:10,]$word,
        col ="lightblue", main ="Most frequent words",
        ylab = "Word frequencies")
```

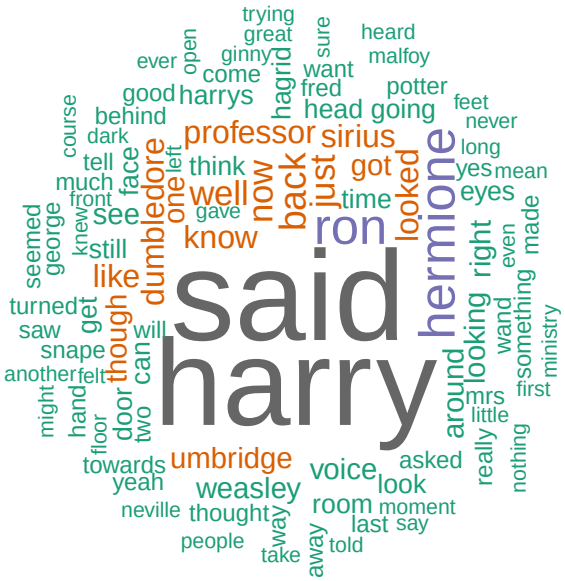



```
ggplot(head(dtm_d,15), aes(reorder(word,freq), freq)) +
  geom_bar(stat="identity") + coord_flip() +
  xlab("Trigrams") + ylab("Frequency") +
  ggtitle("Most frequent words")
```



We can also generate a word cloud of the book:

```
set.seed(1234)
wordcloud(words = dtm_d$word, freq = dtm_d$freq, min.freq = 5,
          max.words=100, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"))
```



We can see that Ron and Hermione are Harry's best friends. Finally, it is also clear that Sirius and Dumbledore are also close to Harry.

Words with a frequency >= 5

```
corpus_bag = findFreqTerms(corpus_dtm, lowfreq = 5)
class(corpus_bag)
```

```
## [1] "character"
```

```
#We can print some of the words that are in our bag of words
corpus_bag[1:50]
```

```
##  [1] "harry"      "order"      "phoenix"    "potter"     "chapter"
##  [6] "one"        "dudley"     "back"       "banned"     "boy"
## [11] "breeze"     "cars"       "close"      "cool"       "day"
## [16] "drawing"    "drive"      "due"        "dusty"      "emerald"
## [21] "far"        "flat"       "flowerbed"  "four"       "gleaming"
## [26] "green"      "hope"       "houses"     "large"      "lawns"
## [31] "lay"        "left"       "lying"      "number"     "outside"
## [36] "person"     "privet"     "retreated"  "shade"      "silence"
## [41] "square"     "stood"      "summer"     "thrown"     "use"
## [46] "usual"      "usually"    "wide"       "windows"    "appearance"
```

Sentiment Analysis

In this part, we are going to discover if the overall sentiment of this book is positive or negative.

First, we'll need a list of positive and negative words.

```
pos_file = "http://ptrckprry.com/course/ssd/data/positive-words.txt"
poswords = scan(pos_file, what='character', comment.char=';')
poswords[1:50]
```

```
##  [1] "a+"              "abound"          "abounds"         "abundance"
##  [5] "abundant"        "accessable"      "accessible"      "acclaim"
##  [9] "acclaimed"       "acclamation"     "accolade"        "accolades"
## [13] "accommodative"   "accomodative"    "accomplish"      "accomplished"
## [17] "accomplishment"  "accomplishments" "accurate"        "accurately"
## [21] "achievable"      "achievement"     "achievements"    "achievible"
## [25] "acumen"          "adaptable"       "adaptive"        "adequate"
## [29] "adjustable"      "admirable"       "admirably"       "admiration"
## [33] "admire"          "admirer"         "admiring"        "admiringly"
## [37] "adorable"        "adore"           "adored"          "adorer"
## [41] "adoring"         "adoringly"       "adroit"          "adroitly"
## [45] "adulate"         "adulation"       "adulatory"       "advanced"
## [49] "advantage"       "advantageous"
```

```
neg_file = "http://ptrckprry.com/course/ssd/data/negative-words.txt"
negwords = scan(neg_file, what='character', comment.char=';')
negwords[1:50]
```

```
##  [1] "2-faced"       "2-faces"       "abnormal"      "abolish"
##  [5] "abominable"    "abominably"    "abominate"     "abomination"
##  [9] "abort"         "aborted"       "aborts"        "abrade"
## [13] "abrasive"      "abrupt"        "abruptly"      "abscond"
## [17] "absence"       "absent-minded" "absentee"      "absurd"
## [21] "absurdity"     "absurdly"      "absurdness"    "abuse"
## [25] "abused"        "abuses"        "abusive"       "abysmal"
## [29] "abysmally"     "abyss"         "accidental"    "accost"
## [33] "accursed"      "accusation"    "accusations"   "accuse"
## [37] "accuses"       "accusing"      "accusingly"    "acerbate"
## [41] "acerbic"       "acerbically"   "ache"          "ached"
## [45] "aches"         "achey"         "aching"        "acrid"
## [49] "acridly"       "acridness"
```

After that, based on these lists, we can find how many positive and negative words there are in our bag.

```
match_pos = match(corpus_bag, poswords)
match_pos[1:200]
```

```
##   [1]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA 234  NA  NA 360  NA
##  [16]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
##  [31]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
##  [46]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
##  [61]  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
```

```
##  [76]   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA  309   NA   NA
##  [91]   NA   NA   NA   NA   NA   NA  927   NA   NA   NA   NA   NA   NA   NA   NA
## [106]   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
## [121]   NA   NA   NA  805   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
## [136]   NA   NA   NA   NA 1088   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
## [151]   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
## [166] 1300   NA   NA   NA 1604   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
## [181]   NA   NA  751   NA   NA   NA 1319   NA   NA   NA   NA   NA   NA   NA   NA
## [196]   NA   NA   NA   NA   NA
```

```
match_pos_bool = !is.na(match(corpus_bag, poswords))
match_pos_bool[1:200]
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
##  [13] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [157] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## [169] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [181] FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [193] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
sum(!is.na(match(corpus_bag, poswords)))
```

```
## [1] 220
```

There are 220 positive words in our bag (positive words which appear at list five times in the book).

```
sum(!is.na(match(corpus_bag, negwords)))
```

```
## [1] 417
```

There are 417 negative words in our bag (negative words which appear at list five times in the book).

We can calculate the overall score of the book Harry Potter and the Order of the Phoenix:

```
score = sum(!is.na(match(corpus_bag, poswords))) - sum(!is.na(match(corpus_bag, negwords)))
score
```

```
## [1] -197
```

We can conclude that this book is a negative book which is the case because there many wars in this story.

Conclusions

Let's sum up all the facts that we found by analyzing the text of this book:

- Ron and Hermione are Harry's best friends
- Sirius and Dumbledore are also close to Harry
- There are many dialogues in the book (said is the most frequent word of the book)
- There are about twice as many negative words as positive words in the book: this is a negative book, bad things happen in this story

---