An Application Example of Sentiment Analysis Using R

Jollanda Shara

University "Eqrem Cabej"

Gjirokaster, Albania.

e-mail: jokrisha@yahoo.com

Abstract

Sentiment analysis is a machine learning tool that analyzes texts for polarity, from positive to negative. Machines automatically learn how to detect sentiment without human input, only by training machine learning tools with examples of emotions in text To put it simple, sentiment analysis is the process of analyzing a piece of text and to determine if the writer's attitude towards the subject matter is positive, negative, or neutral. Sentiment analysis models can be trained to read far off trivial definitions, to understand things like, context, sarcasm, and misapplied words. [4] In this paper we have given an application example of sentiment analysis using R, that of the Tolstoy's book, "Anna Karenina". For our analysis, we have used the AFINN, BING and NRC lexicons. Some text analysis, such as bigrams, trigrams etc., is done, as well.

1. Introduction

Sentiment analysis is a research branch placed at the heart of natural language processing (NLP), computational linguistics and text mining. It refers to any measures by which subjective information is extracted from textual documents. In other words, it extracts the polarity of the expressed opinion in a range spanning from positive to negative. As a result, one may also refer to sentiment analysis as opinion mining (Pang and Lee 2008). Sentiment analysis is used for many applications, especially in business intelligence. Some examples of applications for sentiment analysis include:

- Analyzing the social media discussion around a certain topic;
- Evaluating survey responses;
- Determining whether product reviews are positive or negative.

Sentiment analysis is applied to a numerous domains and textual sources. Research has worked out various approaches to measuring sentiment. We mention here an overview by Pang and Lee, 2008 which is a comprehensive, domain-independent study on the subject.

On the one hand, machine learning viewpoints are preferred when one attempts for high prediction performance. However, machine learning usually works as a black-box, thereby making interpretations difficult. On the other hand, dictionary-based approaches generate lists of positive and negative words. The respective occurrences of these words are then combined into a

single sentiment score. Therefore, the underlying decisions become traceable and researchers can understand the factors that result in a specific sentiment.

In addition, sentiment analysis allows us to generate tailored dictionaries. These are customized to a specific domain, improve prediction performance compared to pure dictionaries and allow full interpretability. Details of this methodology can be found in (Pröllochs, Feuerriegel, and Neumann 2018).

In the process of performing sentiment analysis, one must convert the running text into a machine-readable format. This is achieved by executing a series of preprocessing operations. First, the text is tokenized into single words, followed by some preprocessing steps which will be described in the other section. [1]

2. Sentiment Analysis

Sentiment Analysis is a process of extracting opinions that have different polarities. By polarities, we mean positive, negative or neutral. It is also known as opinion mining and polarity detection. With the help of sentiment analysis, we can find out the nature of opinion that is reflected in documents, websites, social media feed, etc. Sentiment analysis is a type of classification where the data is classified into different classes. These classes can be binary in nature (positive or negative) or, they can have multiple classes (happy, sad, angry, etc.). Sentiment analysis is not perfect, and as with any automatic analysis of language, many of us will have errors in our results. It also cannot tell us why a writer is feeling a certain way. However, it can be useful to quickly summarize some qualities of text, especially if we have so much text that a human reader cannot analyze all of it. As we mentioned above, the process of performing sentiment analysis involves converting the text into a machine-readable format. This is done using a number of preprocessing steps:

- tokenize the text into single words,
- remove stop-words and punctuation,
- stem the text,
- convert it to lowercase.[1, 6]

Originally, sentiment analysis was tested on product reviews on e-commerce platforms such as Amazon.com, where these issues play a relatively minor role. In the case of press texts or discourse in social media, on the other hand, it is often more difficult to evaluate what a sentiment rating refers to or what sentiment level should be regarded as “normal”. For example, press texts generally express little emotion, and negative terms often predominate, without this necessarily being due to a bad state of the world. Finally, one should bear in mind that sentiment analysis is a heuristic procedure that always produces incorrect individual classifications, which ideally does not carry too much weight when examining some changes in the sentiment course over time. There are many ways to do sentiment analysis. Many approaches use the same general idea, however:

- Create or find a list of words associated with strongly positive or negative sentiment;
- Count the number of positive and negative words in the text;
- Analyze the mix of positive to negative words.

Many positive words and few negative words indicates positive sentiment, while many negative words and few positive words indicates negative sentiment. The first step, creating or finding a word list (also called a lexicon), is generally the most time-consuming. While we can often use a lexicon that already exists, if our text is discussing a specific topic we may need to add to or modify it.[2] Computational text analysis has become an exciting research field with many applications in communication research. It can be a difficult method to apply, however, because it requires knowledge of various techniques, and the software required to perform most of these techniques is not readily available in common statistical software packages. As a popular open-source platform, R has an extensive user community that develops and maintains a wide range of text analysis packages. R is a free, open-source, cross-platform programming environment. In contrast to most programming languages, R was specifically designed for statistical analysis, which makes it highly suitable for data science applications. Although the learning curve for programming with R can be sharp, especially for people without prior programming experience,

the tools now available for carrying out text analysis in R make it easy to perform powerful, cutting-edge text analytics using only a few simple commands. One of the keys to R's explosive growth (Fox & Leanage, 2016; Tiobe, 2017) has been its densely populated collection of extension software libraries, known in R terminology as packages, supplied and maintained by R's extensive user community. Each package extends the functionality of the base R language and core packages, and in addition to functions and data must include documentation and examples, often in the form of vignettes demonstrating the use of the package. The best-known package repository, the Comprehensive R Archive Network (CRAN), currently has over 10,000 packages that are published, and which have gone through an extensive screening for procedural conformity and cross-platform compatibility before being accepted by the archive. R thus features a wide range of inter-compatible packages, maintained and continuously updated by scholars, practitioners, and projects such as RStudio and rOpenSci. Furthermore, these packages may be installed easily and safely from within the R environment using a single command. R thus provides a solid bridge for developers and users of new analysis tools to meet, making it a very suitable programming environment for scientific collaboration. Text analysis in particular has become well established in R. There is a great collection of dedicated text processing and text analysis packages, from low-level string operations (Gagolewski, 2017) to advanced text modeling techniques such as fitting Latent Dirichlet Allocation models (Blei, Ng, & Jordan, 2003; Roberts et al., 2014)-about 50 packages in total. Furthermore, there is an increasing effort among developers to cooperate and coordinate, such as the rOpenSci special interest group. One of the main advantages of performing text analysis in R is that it is often possible, and relatively easy, to switch between different packages or to combine them. Recent efforts among the R text analysis developers'community are designed to promote this interoperability to maximize flexibility and choice among users. As a result, learning the basics for text analysis in R provides access to a wide range of advanced text analysis features.

3. Lexicons

Sentiment analysis is done based on lexicons. A lexicon in simpler terms is a vocabulary, say the English lexicon. In this context, a lexicon is a selection of words with the two polarities that can be used as a metric in sentiment analysis. There are many different types of lexicons that can be used depending on the context of the data we are working with. There is also a possibility of creating a custom lexicon depending on how much customisation we would like to make with our data.

The AFINN lexicon is perhaps one of the simplest and most popular lexicons that can be used extensively for sentiment analysis. It is a list of English terms manually rated for valence with an integer between -5 (negative) and +5 (positive) by Finn Årup Nielsen between 2009 and 2011. The AFINN lexicon was developed in 2009 primarily to analyze Twitter sentiment (Nielsen 2011). It consists of 2,477 words with 878 positive and 1,598 negative on a scale of -5 to +5, as we mentioned above, with a mean of -0.59. The Bing lexicon uses a binary categorization model that sorts words into positive or negative positions. It has been developed by Minqing Hu and Bing Liu starting with their 2004 paper where their goal was opinion mining of customer reviews (Hu and Liu 2004). It contains a total of 6,786 word - 2,005 positive and 4,781 negative. Classification is binary, a word is either positive or negative. Word-Emotion Association (NRC Emotion Lexicon) is another lexicon. It is a list of 5,636 English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive). The annotations were manually done by crowdsourcing on Amazon Mechanical Turk and detailed in a paper by Mohammad and Turney (2013). ([7, 8] The tidytext package provides access to several sentiment lexicons. The function get_sentiments() allows us to get specific sentiment lexicons with the appropriate measures for each one.

```
library(tidytext)
```

```
head(get_sentiments("afinn"))
```

```
## # A tibble: 6 x 2
##   word       value
##   <chr>      <dbl>
## 1 abandon       -2
## 2 abandoned     -2
## 3 abandons      -2
## 4 abducted      -2
## 5 abduction     -2
## 6 abductions    -2
```

```
head(get_sentiments("bing"))
```

```
## # A tibble: 6 x 2
##   word       sentiment
##   <chr>      <chr>
## 1 2-faces    negative
## 2 abnormal   negative
## 3 abolish    negative
## 4 abominable negative
## 5 abominably negative
## 6 abominate  negative
```

```
head(get_sentiments("nrc"))
```

```
## # A tibble: 6 x 2
##   word      sentiment
##   <chr>     <chr>
## 1 abacus    trust
```

```
## 2 abandon   fear
## 3 abandon   negative
## 4 abandon   sadness
## 5 abandoned anger
## 6 abandoned fear
```

nrc seems to have large number of words and their sentiments compared to other two.

```
nrc_sentiment <- get_sentiments("nrc")
unique(nrc_sentiment$sentiment)
```

```
##  [1] "trust"        "fear"         "negative"     "sadness"      "anger"
##  [6] "surprise"     "positive"     "disgust"      "joy"
"anticipation"
```

4. The example of "Anna Karenina"

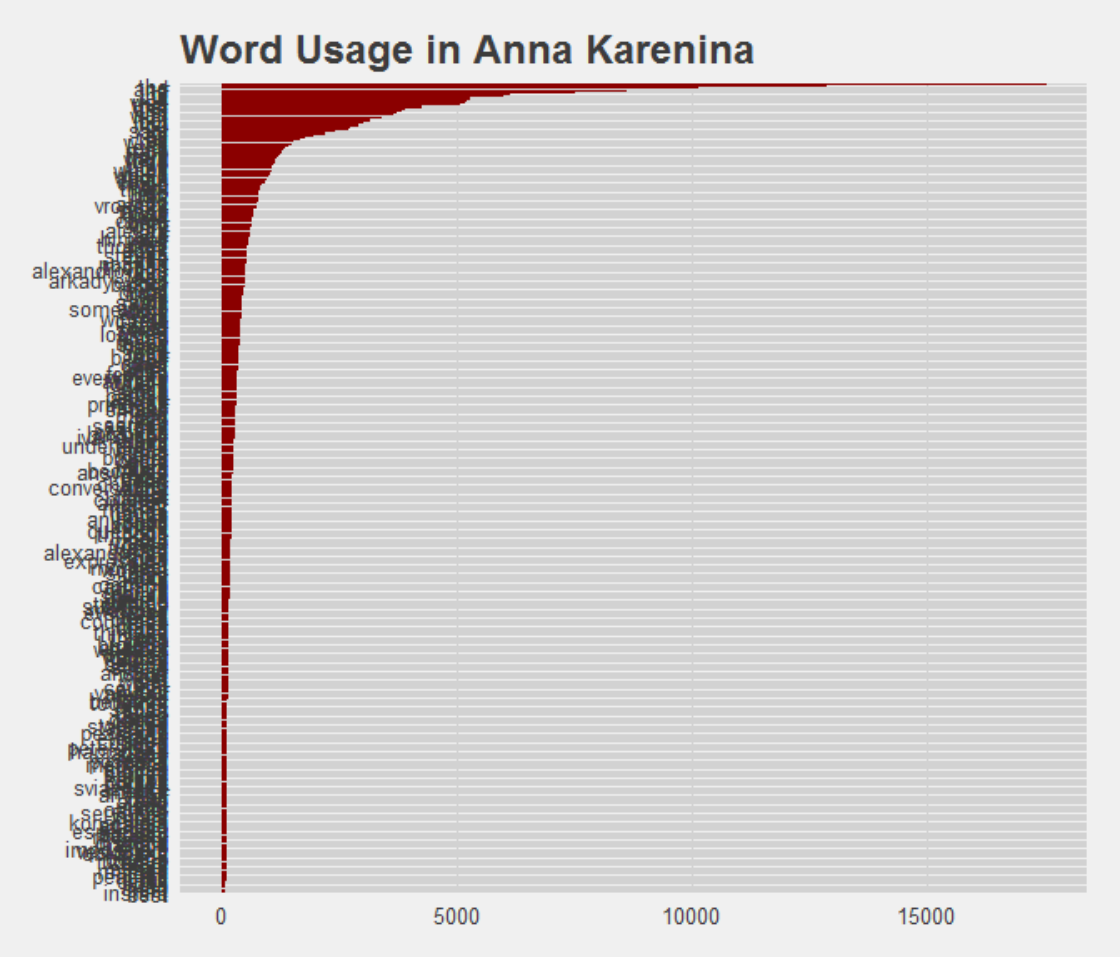
Load libraries and data.

The gutenbergr package.

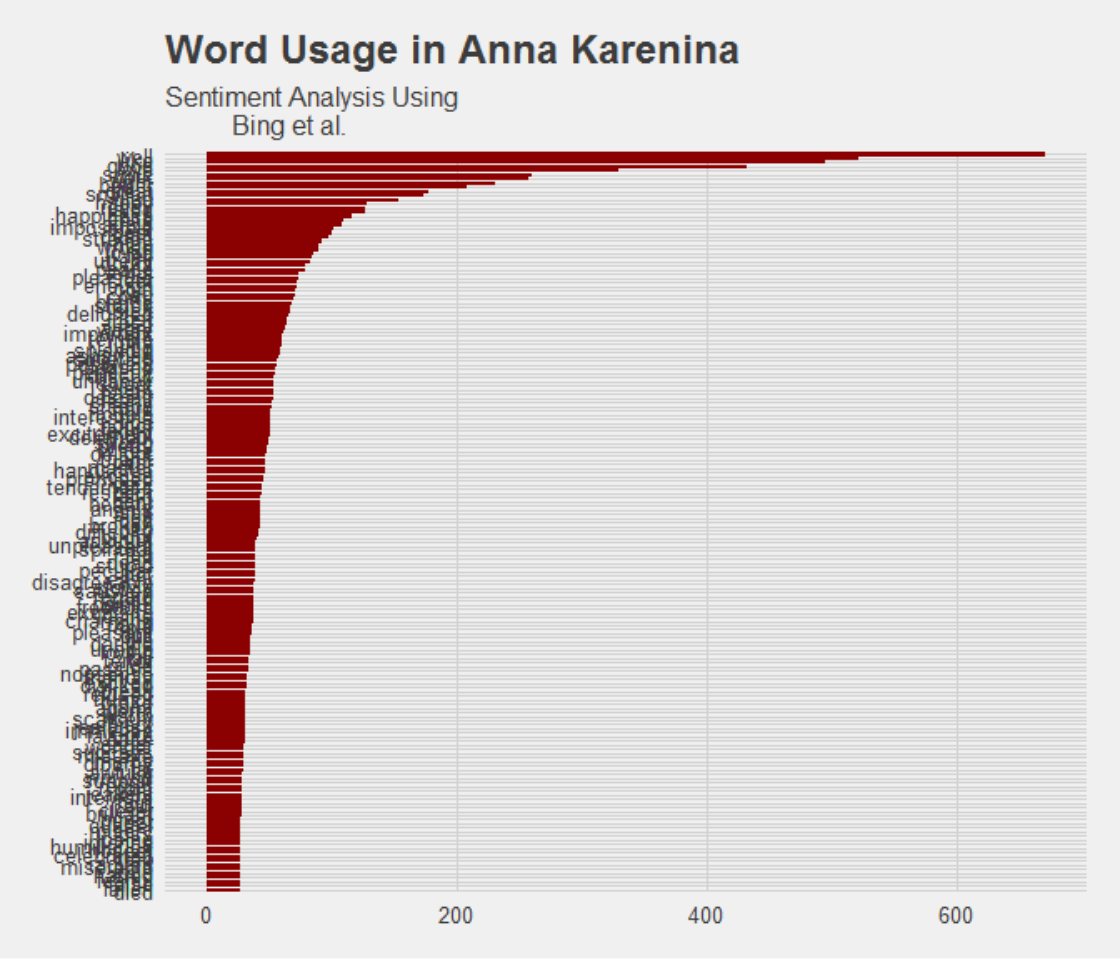
Project Gutenberg is a digital archive of public domain books. The R package **gutenbergr** (Robinson 2016) facilitates the importation of these texts into R. We will combine this with the **tidyverse** and **tidytext** libraries to practice text mining. The package includes tools both for downloading books (stripping out the unhelpful header/footer information), and a complete dataset of Project Gutenberg metadata, gutenberg_metadata, that can be used to find works of interest. In this paper, we use the function gutenberg_download() that downloads one or more works from Project Gutenberg by ID, but other functions can be used for exploring metadata, pair Gutenberg ID with title, author, language, etc., or gather information about authors, as well.[3]

```
library(tidyverse)
library(gutenbergr)
library(tidytext)
library(textdata)
library(stringr)
library(tidyr)
library(dplyr)
library(reshape2)
library(wordcloud)
library(scales)
library(ggthemes)
library(ggraph)
library(igraph)
library(widyr)
```

```
library(SentimentAnalysis)
library(tm)

options(digits = 3)

gutenberg_metadata
```



Now, let's use ggplot2 to produce a horizontal bar chart showing positive and negative word usage in Anna Karenina using the Bing et al. sentiment lexicon.

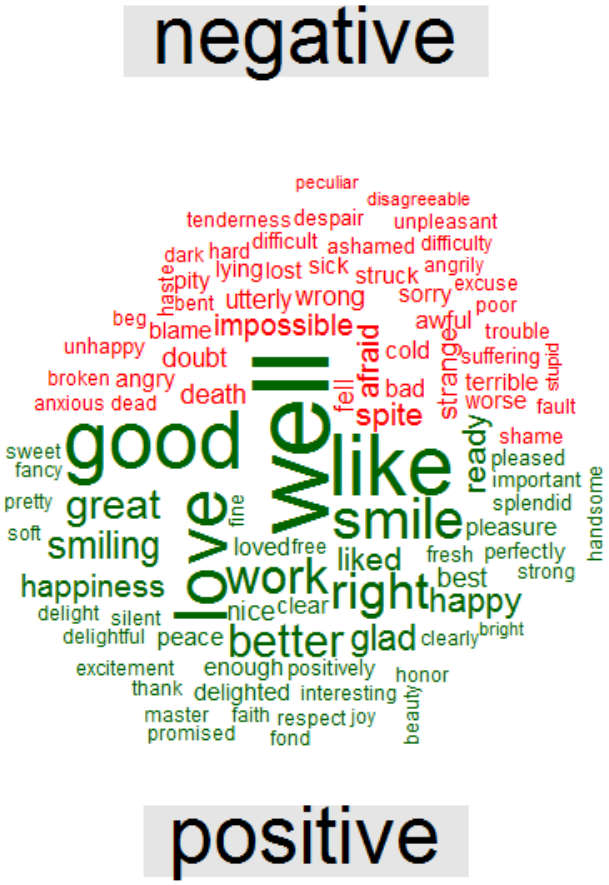


We can also use the NRC sentiment lexicon to get a better insight into how specific emotions play a role in a given text by filtering the data and joining it to our tokenized text. To see, for example, how fear appears in Anna Karenina, we can run the following code chunks:

```
nrc_fear <- get_sentiments("nrc") %>%

  filter(sentiment == "fear")

nrc_fear

tidy_tolstoy %>%

  filter(gutenberg_id == 1399) %>%

  inner_join(nrc_fear)
```

```
## # A tibble: 6,106 x 5
##    gutenberg_id chapter linenumber word         sentiment
##           <int>   <int>      <int> <chr>        <chr>
##  1         1399      47      17293 confusion    fear
##  2         1399      47      17294 intrigue     fear
##  3         1399      47      17308 warning      fear
##  4         1399      47      17316 buried       fear
##  5         1399      47      17338 vanished     fear
##  6         1399      47      17343 hopelessness fear
##  7         1399      47      17347 awful        fear
##  8         1399      47      17349 despair      fear
##  9         1399      47      17350 painful      fear
## 10         1399      47      17354 surprise     fear
## # ... with 6,096 more rows

## Joining, by = "word"
```

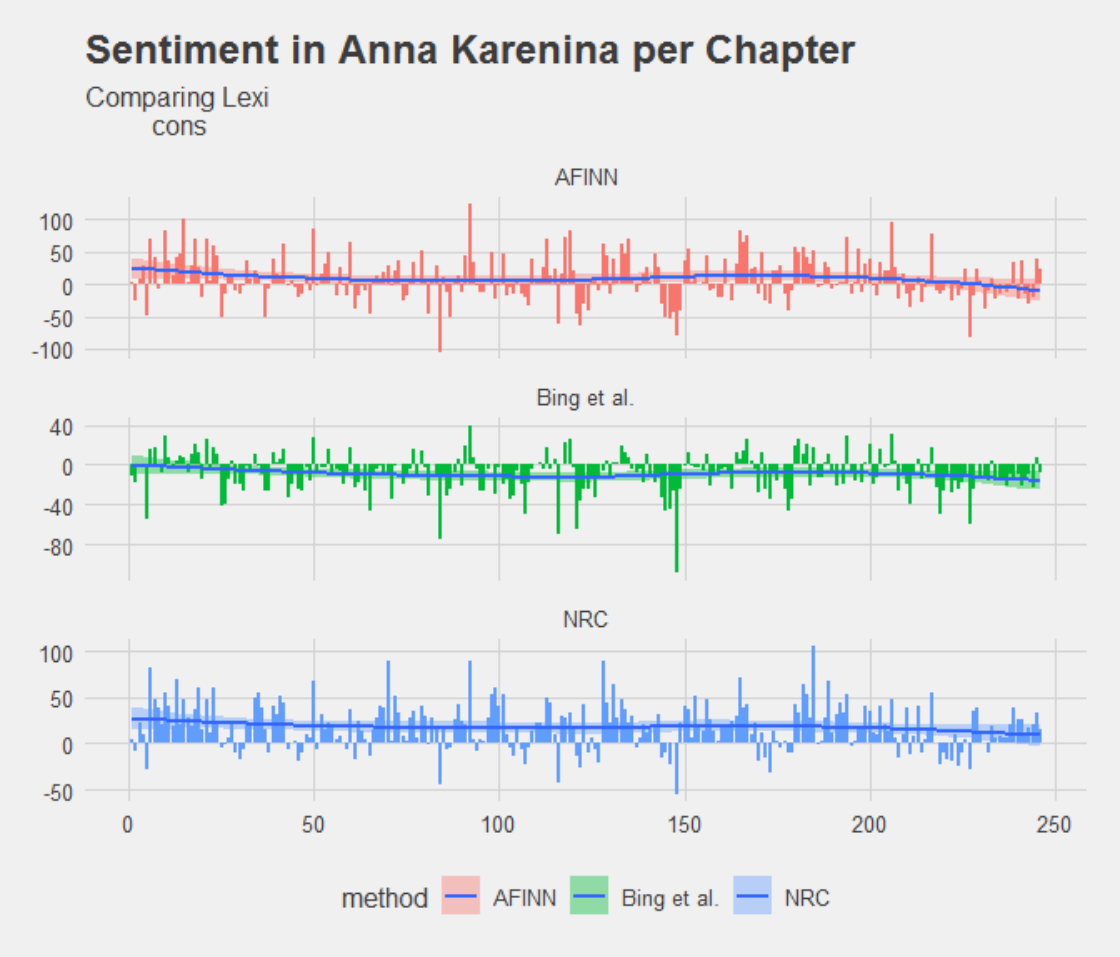
The next word cloud will enable us to efficiently visualize the negative as well as positive groups of data. Therefore, we are now able to see the different groups of data based on their corresponding sentiments.



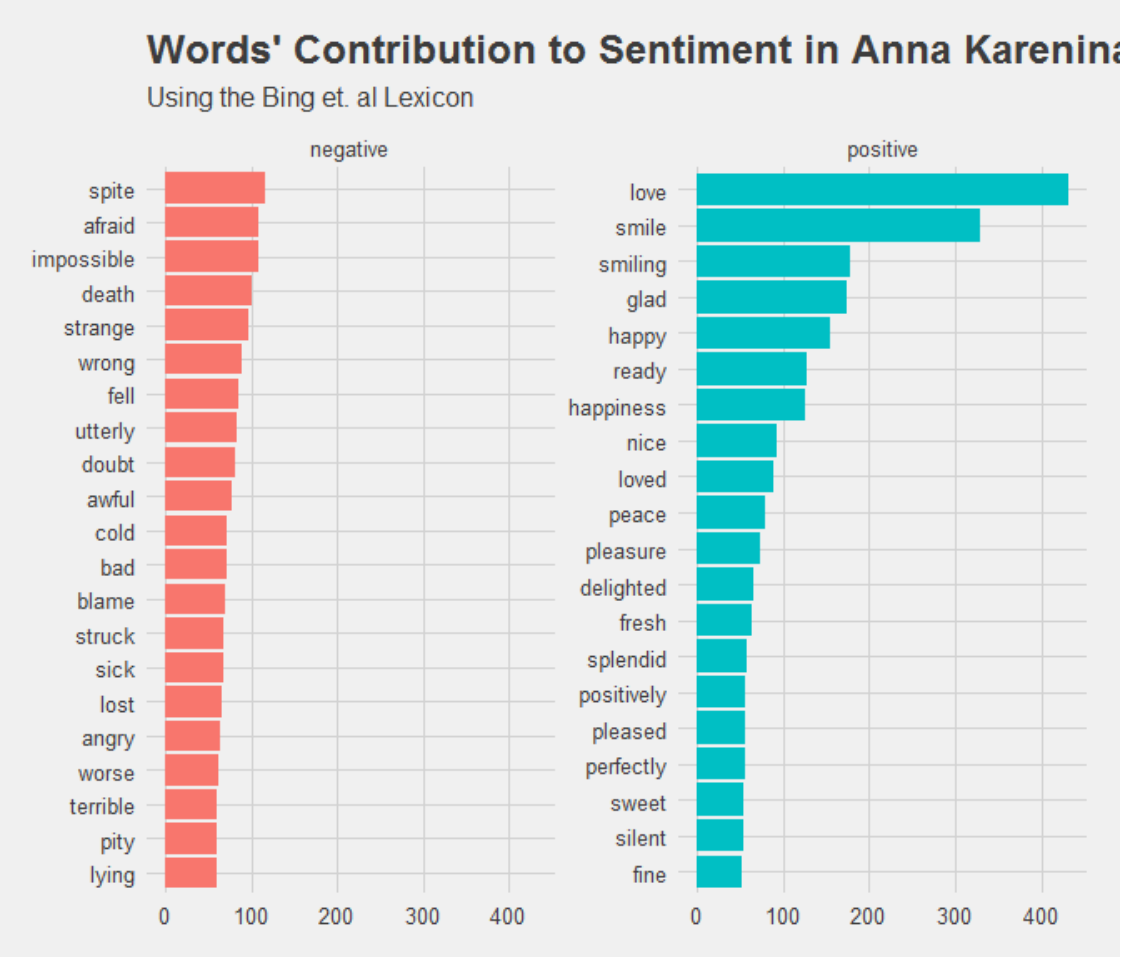
The following tibbles show the sentiment values with different lexicons in Anna Karenina per chapter. We have plotted the sentiment in Anna Karenina per chapter, comparing lexicons, as well:

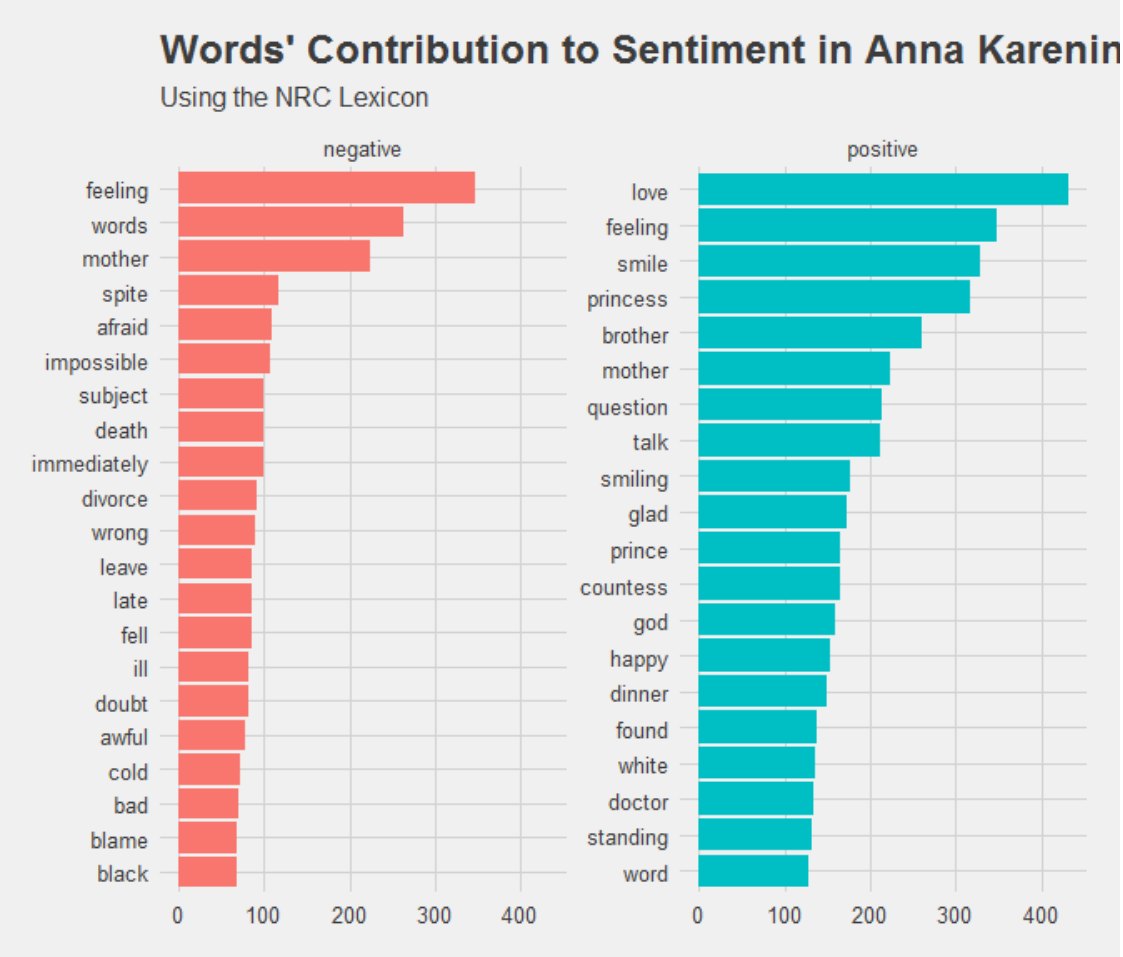
```
## # A tibble: 246 x 3
##    index sentiment method
##    <int>     <dbl> <chr>
##  1     1         2 AFINN
##  2     2       -24 AFINN
##  3     3         8 AFINN
##  4     4        30 AFINN
##  5     5       -48 AFINN
##  6     6        70 AFINN
##  7     7        42 AFINN
##  8     8        -8 AFINN
##  9     9         0 AFINN
## 10    10        84 AFINN
## # ... with 236 more rows
```

```
## # A tibble: 492 x 5
##    method      index negative positive sentiment
##    <chr>       <int>    <dbl>    <dbl>     <dbl>
##  1 Bing et al.     1       24       13       -11
##  2 Bing et al.     2       23        6       -17
##  3 Bing et al.     3       31       30        -1
##  4 Bing et al.     4       53       50        -3
##  5 Bing et al.     5       89       35       -54
##  6 Bing et al.     6       73       89        16
##  7 Bing et al.     7       23       40        17
##  8 Bing et al.     8       14       13        -1
##  9 Bing et al.     9       22       14        -8
## 10 Bing et al.    10       65       94        29
## # ... with 482 more rows

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The following plots show words' contribution to sentiment in Anna Karenina, using the Bing et. al, the first and the nrc, the second.

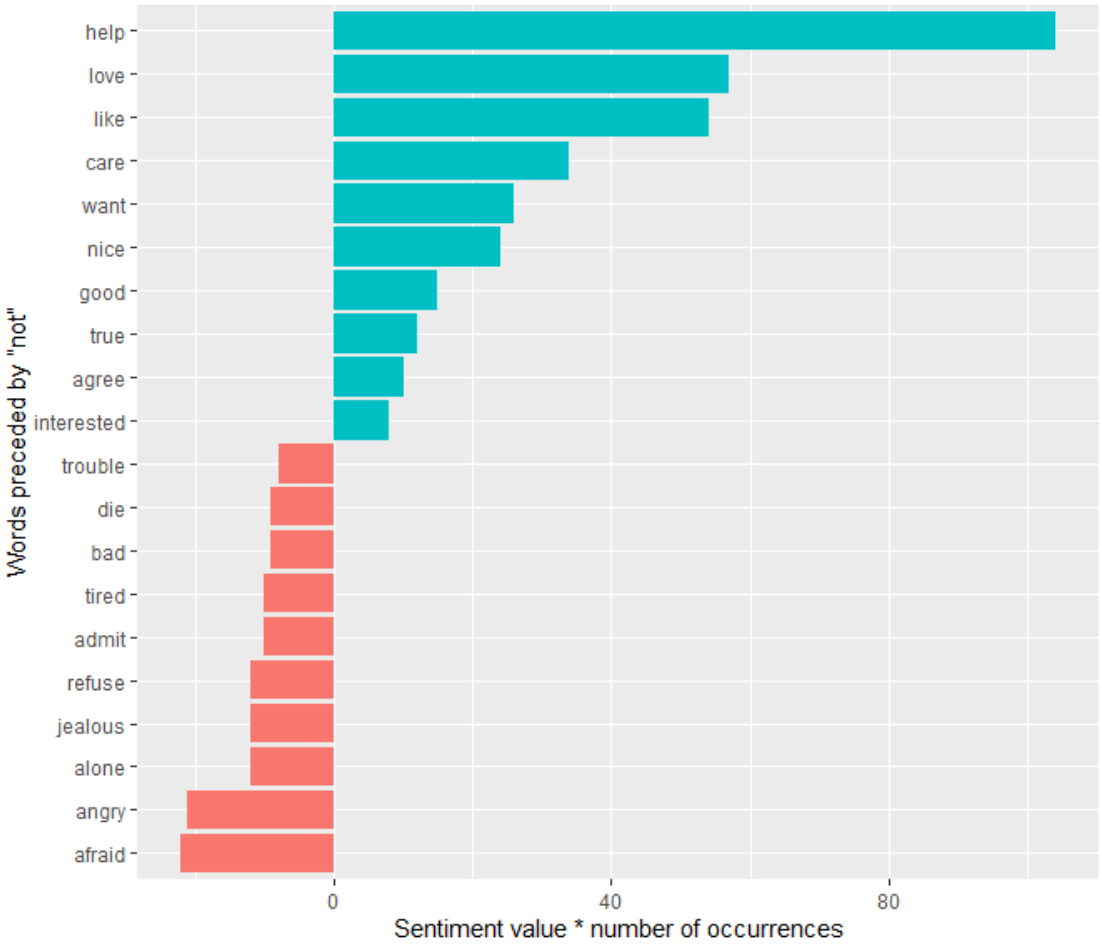




We can perform some text analysis counting, for example, the bigrams, negated words, etc. and then plotting them:

```
## # A tibble: 117,106 x 2
##    bigram      n
##    <chr>   <int>
##  1 <NA>     8786
##  2 of the   1801
##  3 in the   1533
##  4 he had    943
##  5 to the    939
##  6 he was    852
##  7 at the    836
##  8 and the   687
##  9 it was    639
## 10 on the    622
## # ... with 117,096 more rows
```

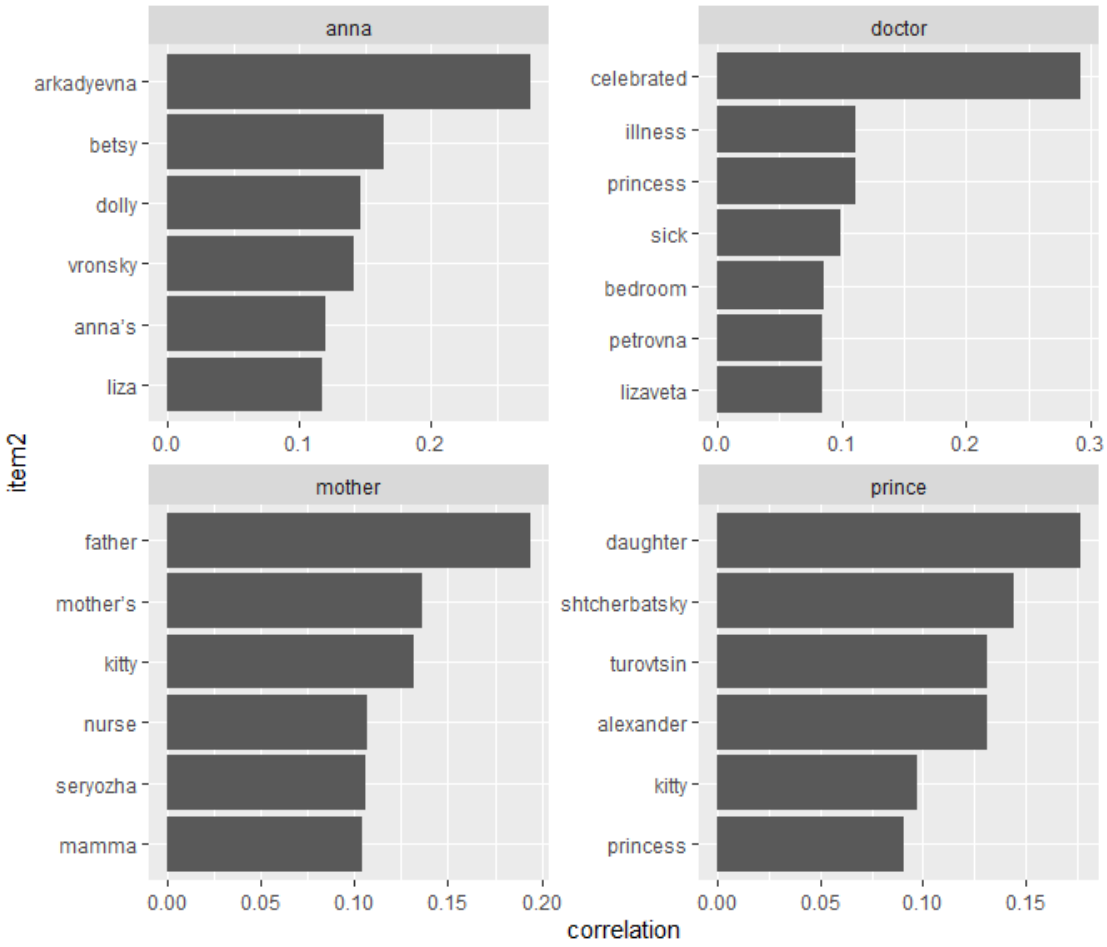
```
## # A tibble: 728 x 3
##    word1 word2      n
##    <chr> <chr>  <int>
##  1 not   to       201
##  2 not   a        110
##  3 not   be        95
##  4 not   know      78
##  5 not   the       67
##  6 not   in        66
##  7 not   help      52
##  8 not   been      50
##  9 not   have      46
## 10 not   merely    45
## # ... with 718 more rows
```

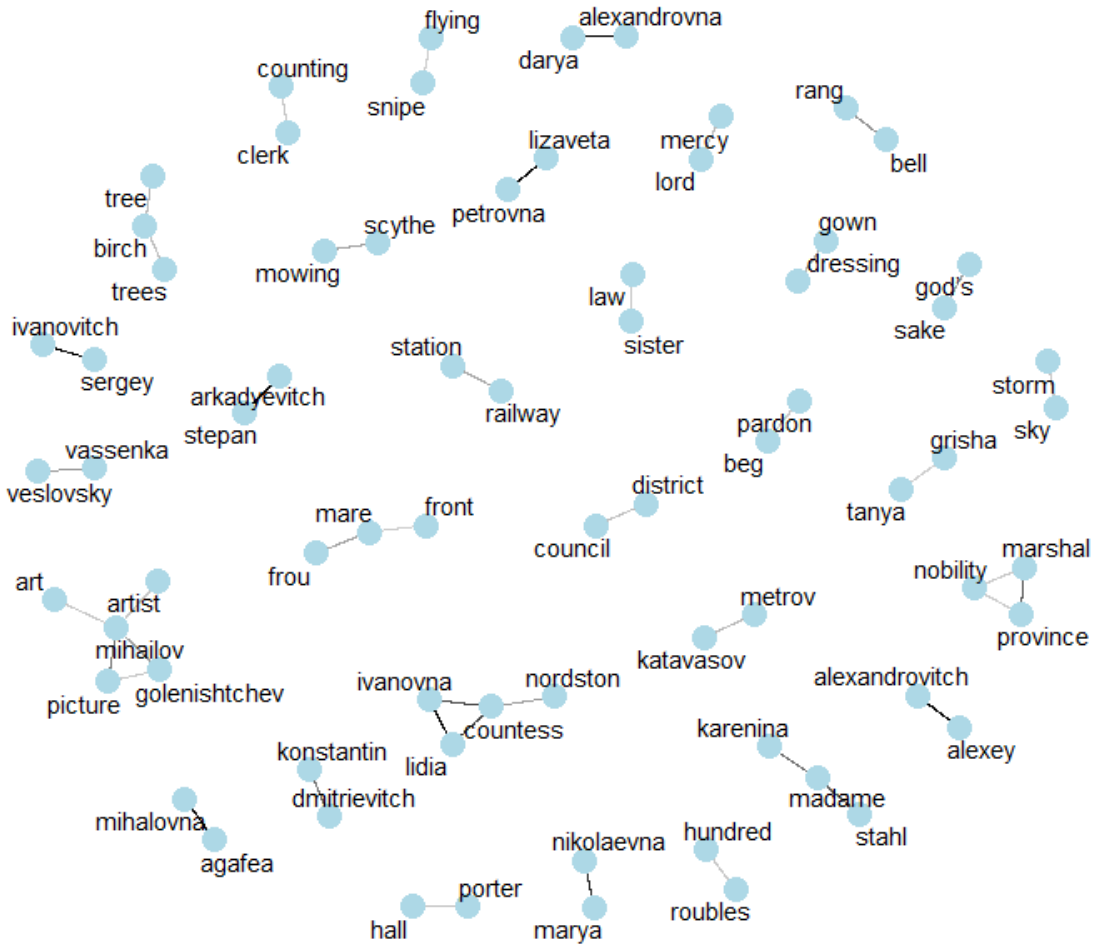


These tibbles show the number of word-pairs, one of them being “lidia”-the first one . In the second we need to filter for at least relatively common words first. The third one describes the word’s correlation when we filter with the word :”countess”. And, finally, the first plot lets us pick particular interesting words and find the other words most associated with them and in the

last one we use ggraph to visualize the correlations and clusters of words that were found by the widyr package.

```
## # A tibble: 1,428 x 3
##    item1 item2              n
##    <chr> <chr>          <dbl>
##  1 lidia ivanovna          73
##  2 lidia countess          65
##  3 lidia alexey            51
##  4 lidia alexandrovitch    48
##  5 lidia anna              16
##  6 lidia eyes              15
##  7 lidia smile             15
##  8 lidia time              14
##  9 lidia stepan            14
## 10 lidia arkadyevitch      14
## # ... with 1,418 more rows

## # A tibble: 1,429,220 x 3
##    item1        item2        correlation
##    <chr>        <chr>              <dbl>
##  1 petrovna     lizaveta           1
##  2 lizaveta     petrovna           1
##  3 arkadyevitch stepan             0.959
##  4 stepan       arkadyevitch       0.959
##  5 mihalovna    agafea             0.939
##  6 agafea       mihalovna          0.939
##  7 ivanovitch   sergey             0.937
##  8 sergey       ivanovitch         0.937
##  9 alexandrovna darya              0.923
## 10 darya        alexandrovna       0.923
## # ... with 1,429,210 more rows

## # A tibble: 1,195 x 3
##    item1    item2          correlation
##    <chr>    <chr>                <dbl>
##  1 countess lidia               0.599
##  2 countess ivanovna            0.572
##  3 countess nordston            0.390
##  4 countess landau              0.230
##  5 countess alexey              0.125
##  6 countess alexandrovitch      0.119
##  7 countess karenina            0.0952
##  8 countess son                 0.0876
##  9 countess madame              0.0850
## 10 countess letter              0.0791
## # ... with 1,185 more rows
```


anna
arkadyevna
betsy
dolly
vronsky
anna's
liza
0.0
0.1
0.2
doctor
celebrated
illness
princess
sick
bedroom
petrovna
lizaveta
0.0
0.1
0.2
0.3
mother
father
mother's
kitty
nurse
seryozha
mamma
0.00
0.05
0.10
0.15
0.20
prince
daughter
shtcherbatsky
turovtsin
alexander
kitty
princess
0.00
0.05
0.10
0.15
item2
correlation



5. Conclusion.

In today's environment where we're enduring from data overload, companies might have mountains of customer feedback collected. Yet for mere humans, it's still impossible to analyze it manually without any sort of error or bias. Sentiment analysis provides answers into what the most important issues are. Because sentiment analysis can be automated, decisions can be made based on a significant amount of data rather than plain intuition that isn't always right. Sentiment analysis is useful for quickly gaining insights using large volumes of text data.[5] In this paper we have tried to give a feeling of it, through a simple application example using R. But, as we mentioned above, there is a huge number of its applications described in numerous papers, books, etc.

References:

[1] Stefan Feuerriegel & Nicolas Pröllochs, SentimentAnalysis**,** MIT License Copyright (c) 2021.

[2] Welcome to Text Mining with R by O'Reilly.

[3] Julia Silge and David Robinson **,** Text Mining with R: A Tidy Approach, 2021-03-16.

[4] https://monkeylearn.com/blog/sentiment-analysis-machine-learning/

[5] https://getthematic.com/insights/sentiment-analysis/

[6] https://opendatascience.com/sentiment-analysis-in-r-made-simple/

[7] https://bookdown.org/psonkin18/berkshire/sentiment.html

[8] http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm