Text-Mining-Mark-Twain–Novels.R

YASHADA

2023-07-01

Title: Text Mining Mark Twain' Novels

Author: Yashada Nikam

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(rvest)
```

```
## Warning: package 'rvest' was built under R version 4.2.3
```

```
library(purrr)
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 4.2.3
```

```
library(gutenbergr)
```

```
## Warning: package 'gutenbergr' was built under R version 4.2.3
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
# Download the 5 novels

twain<- gutenberg_download(c(76, 74, 1837, 3176, 86))
```

```
## Determining mirror for Project Gutenberg from https://www.gutenberg.org/robot/harvest
```

```
## Using mirror http://aleph.gutenberg.org
```

```
tidy_twain <- twain %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words)
```

```
## Joining with `by = join_by(word)`
```

Remove common unimportant words like “the” and “and” using stop words.

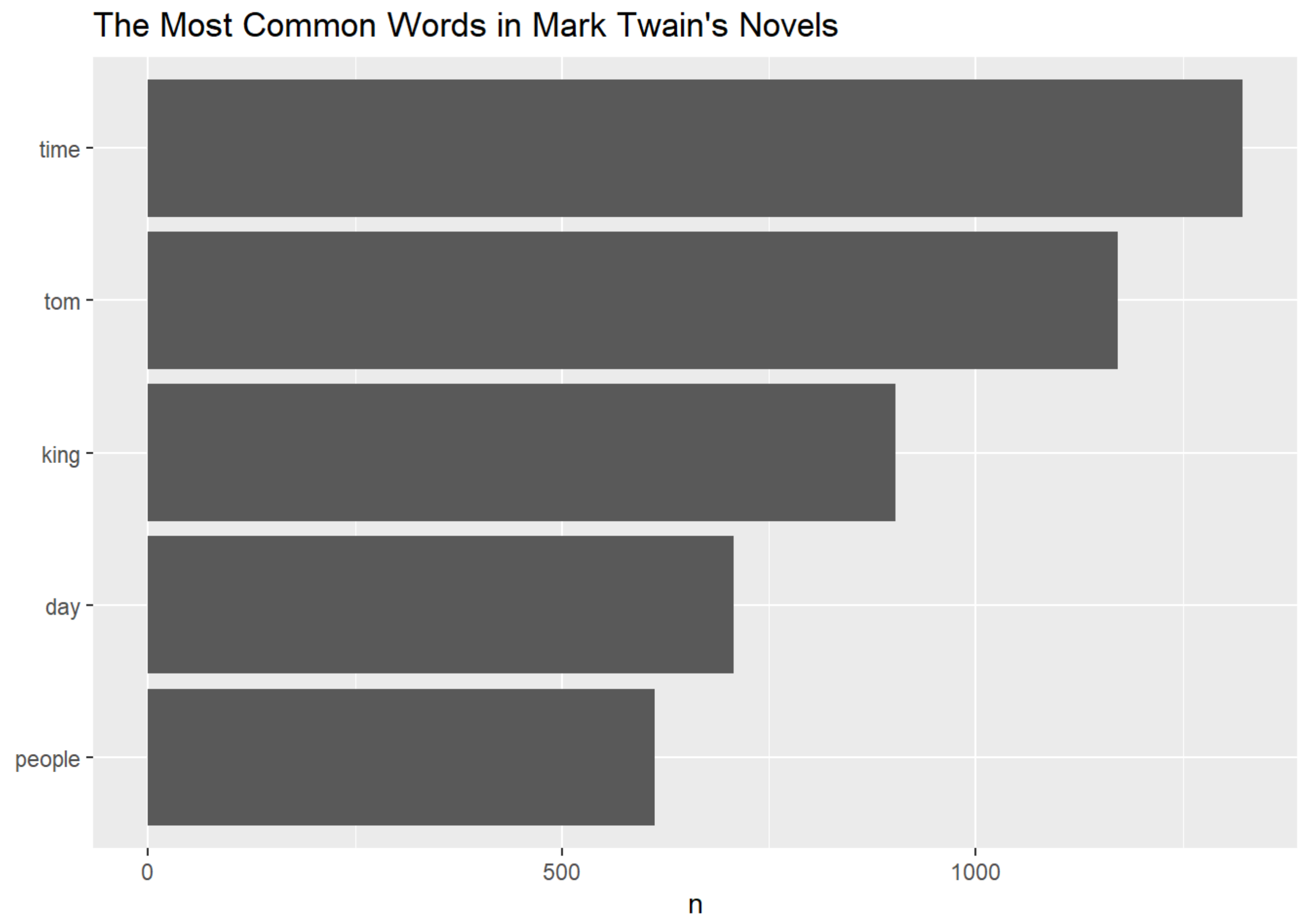
The result, tidy_twain, contains only the important words, making it easier to analyze

After removing the stop words, here is a list of words starts from the most frequent.

```
tidy_twain %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 23,666 × 2
##    word        n
##    <chr>   <int>
##  1 time     1322
##  2 tom      1171
##  3 king      903
##  4 day       708
##  5 people    612
##  6 don’t     554
##  7 sir       512
##  8 night     505
##  9 head      482
## 10 hundred   473
## # ℹ 23,656 more rows
```

```
tidy_twain %>%
  count(word, sort = TRUE) %>%
  filter(n > 600) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip() + ggtitle("The Most Common Words in Mark Twain's Novels")
```



“time” appears 1322 times, “tom” appears 1171 times, “king” appears 903 times, and so on.

These word frequencies provide valuable insights into the themes or topics that are most prevalent

in Mark Twain’s writings.

Sentiment Analysis

get_sentiments(“bing”): This function retrieves the Bing lexicon, a pre-defined list of words

labeled as positive or negative based on sentiment analysis.

```
bing_word_counts <- tidy_twain %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  ungroup()
```

```
## Joining with `by = join_by(word)`
```

```
## Warning in inner_join(., get_sentiments("bing")): Detected an unexpected many-to-many
relationship between `x` and `y`.
## ℹ Row 80396 of `x` matches multiple rows in `y`.
## ℹ Row 5581 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

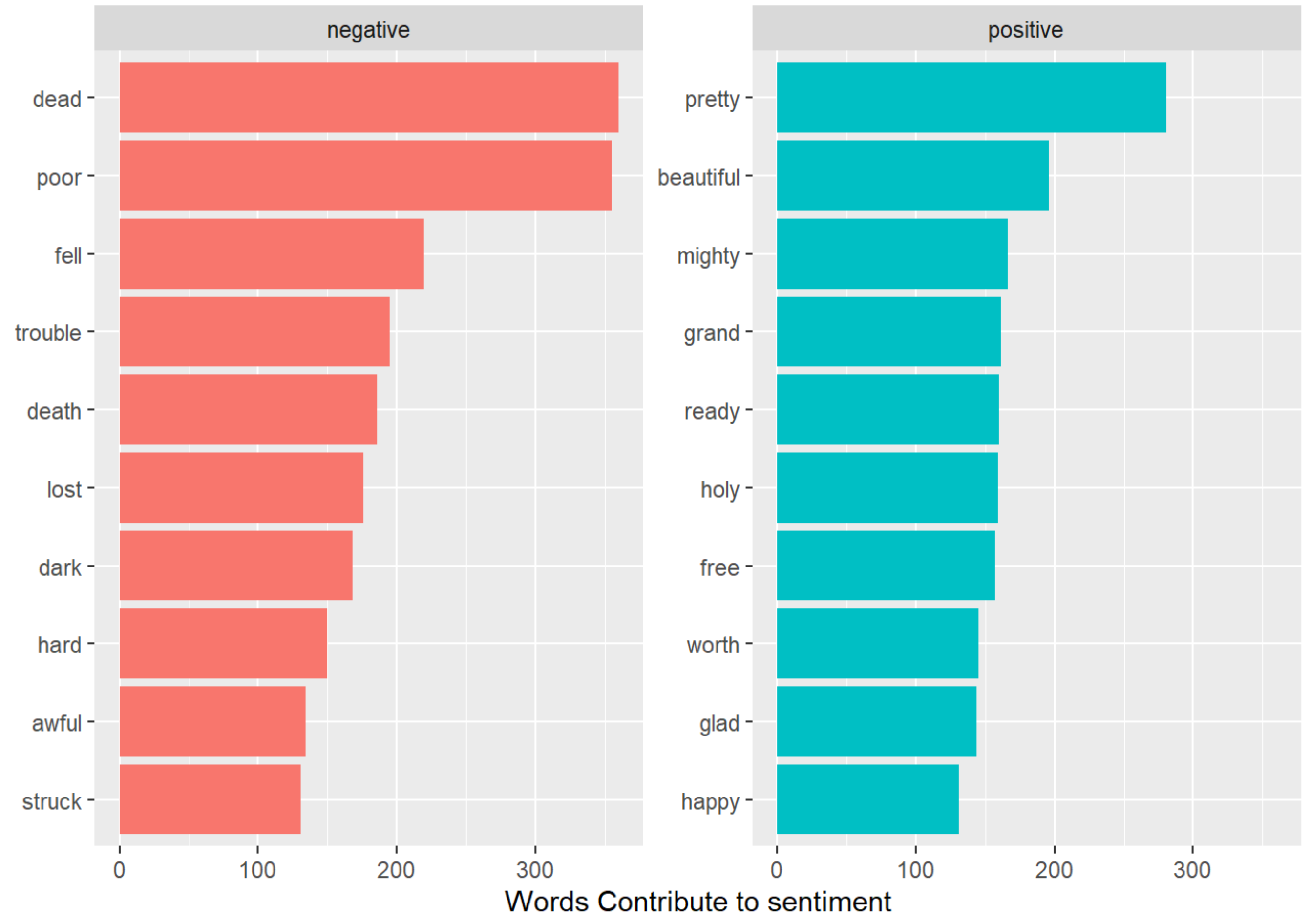
```
bing_word_counts
```

```
## # A tibble: 3,169 × 3
##    word      sentiment     n
##    <chr>     <chr>     <int>
##  1 dead      negative    360
##  2 poor      negative    355
##  3 pretty    positive    281
##  4 fell      negative    220
##  5 beautiful positive    196
##  6 trouble   negative    195
##  7 death     negative    186
##  8 lost      negative    176
##  9 dark      negative    168
## 10 mighty    positive    166
## # ℹ 3,159 more rows
```

Sentiment Categories

```
bing_word_counts %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Words Contribute to sentiment",
       x = NULL) +
  coord_flip()
```

```
## Selecting by n
```



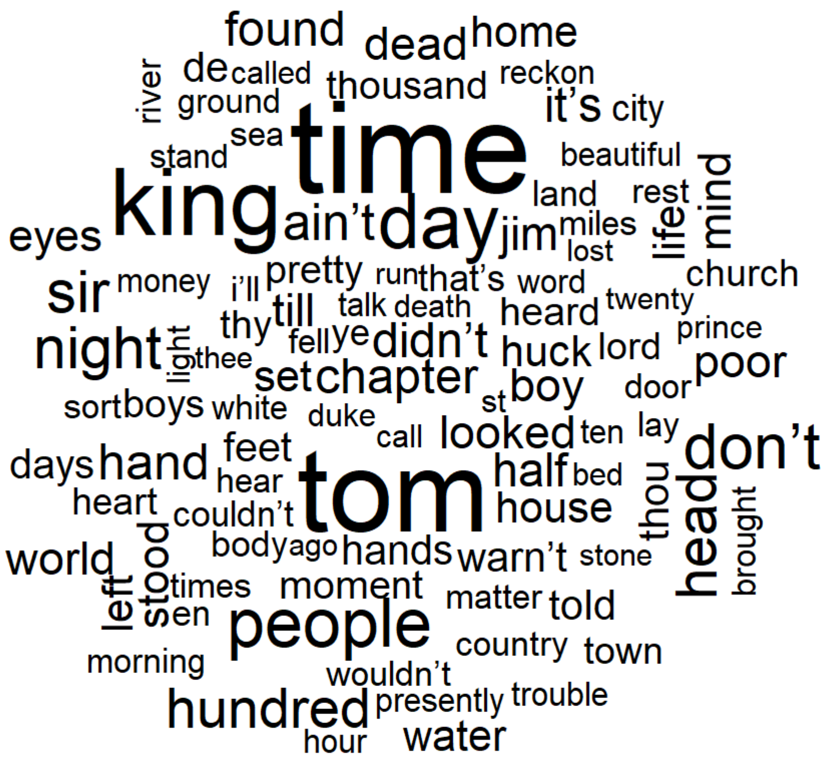
We can observe which words are most frequently associated with positive sentiments (like “pretty” and “beautiful”) and which words are more commonly connected to negative sentiments (like “dead” and “poor”).

```
library(RColorBrewer)
library(wordcloud)
```

```
## Warning: package 'wordcloud' was built under R version 4.2.3
```

```
tidy_twain%>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100))
```

```
## Joining with `by = join_by(word)`
```



```
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.2.3
```

```
tidy_twain %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("#F8766D", "#00BFC4"),
                   max.words = 100)
```

```
## Joining with `by = join_by(word)`
```

```
## Warning in inner_join(., get_sentiments("bing")): Detected an unexpected many-to-many
relationship between `x` and `y`.
## ℹ Row 80396 of `x` matches multiple rows in `y`.
## ℹ Row 5581 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): comfortable could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): splendid could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): heaven could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): perfectly could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): honest could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): fresh could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): miracle could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): gentle could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): perfect could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): stately could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): proud could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): clean could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): comfort could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): wise could not be fit on page. It will not be plotted.
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): proper could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): sweet could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): bright could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): handsome could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): glory could not be fit on page. It will not be plotted.
```

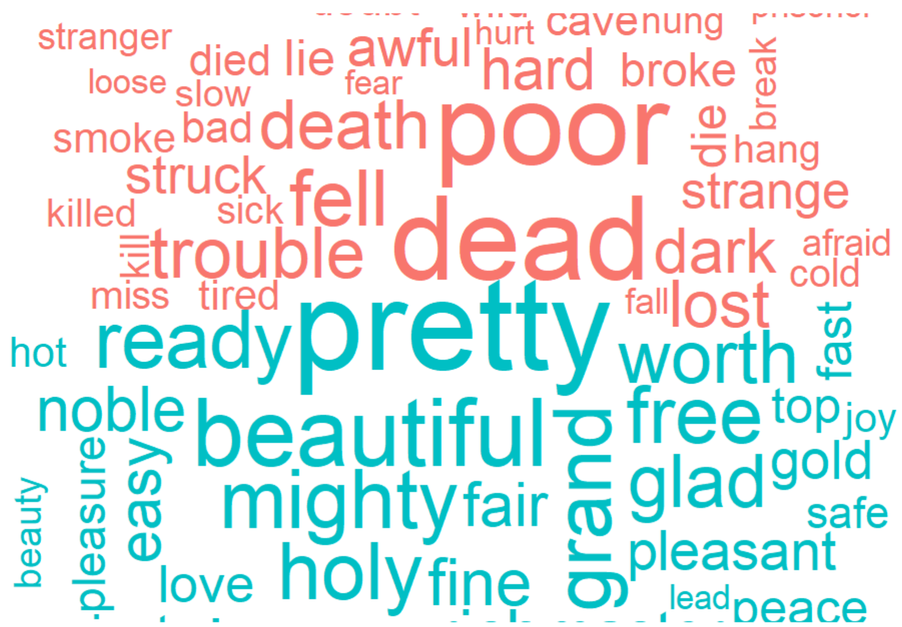
```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): smile could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): celebrated could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): pride could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): famous could not be fit on page. It will not be plotted.
```

```
## Warning in comparison.cloud(., colors = c("#F8766D", "#00BFC4"), max.words =
## 100): silent could not be fit on page. It will not be plotted.
```



Relationships between words

```
twain_bigrams <- twain %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

twain_bigrams
```

```
## # A tibble: 534,736 × 2
##    gutenberg_id bigram
##           <int> <chr>
##  1           74 the adventures
##  2           74 adventures of
##  3           74 of tom
##  4           74 tom sawyer
##  5           74 <NA>
##  6           74 <NA>
##  7           74 by mark
##  8           74 mark twain
##  9           74 <NA>
## 10           74 samuel langhorne
## # ℹ 534,726 more rows
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.2.3
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:reshape2':
##
##     smiths
```

```
bigrams_separated <- twain_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

bigram_counts <- bigrams_filtered %>%
  count(word1, word2, sort = TRUE)

bigram_counts
```

```
## # A tibble: 41,150 × 3
##    word1 word2         n
##    <chr> <chr>     <int>
##  1 <NA>  <NA>      11653
##  2 tom   sawyer       65
##  3 aunt  polly        57
##  4 tom   canty        48
##  5 injun joe          47
##  6 sir   launcelot    47
##  7 mary  jane         42
##  8 aunt  sally        38
##  9 thou  shalt        35
## 10 holy  land         33
## # ℹ 41,140 more rows
```

Now, each token in the data represents a bigram, meaning two words paired together.

If either of the words in a bigram is a stop word, that word will be removed from the bigram.

After filtering out these stop words, we are interested in identifying the most frequent bigrams

that remain in the data.

```
bigrams_united <- bigrams_filtered %>%
  unite(bigram, word1, word2, sep = " ")

bigrams_united
```

```
## # A tibble: 60,872 × 2
##    gutenberg_id bigram
##           <int> <chr>
##  1           74 tom sawyer
##  2           74 NA NA
##  3           74 NA NA
##  4           74 mark twain
##  5           74 NA NA
##  6           74 samuel langhorne
##  7           74 langhorne clemens
##  8           74 NA NA
##  9           74 NA NA
## 10           74 NA NA
## # ℹ 60,862 more rows
```

```
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.2.3
```

```
##
## Attaching package: 'igraph'
```

```
## The following object is masked from 'package:tidyr':
##
##     crossing
```

```
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
bigram_graph <- bigram_counts %>%
  filter(n > 20) %>%
  graph_from_data_frame()
```

```
## Warning in graph_from_data_frame(.): In `d' `NA' elements were replaced with
## string "NA"
```

```
bigram_graph
```

```
## IGRAPH 6b74219 DN-- 38 24 --
## + attr: name (v/c), n (e/n)
## + edges from 6b74219 (vertex names):
##  [1] NA       ->NA        tom      ->sawyer    aunt     ->polly
##  [4] tom      ->canty     injun    ->joe       sir      ->launcelot
##  [7] mary     ->jane      aunt     ->sally     thou     ->shalt
## [10] holy     ->land      hundred  ->yards     thou     ->art
## [13] hundred  ->feet      sir      ->marhaus   sunday   ->school
## [16] miles    ->hendon    centuries->ago       st       ->john
## [19] thou     ->hast      st       ->peter's   sir      ->sagramor
## [22] ten      ->minutes   miss     ->watson    sir      ->kay
```

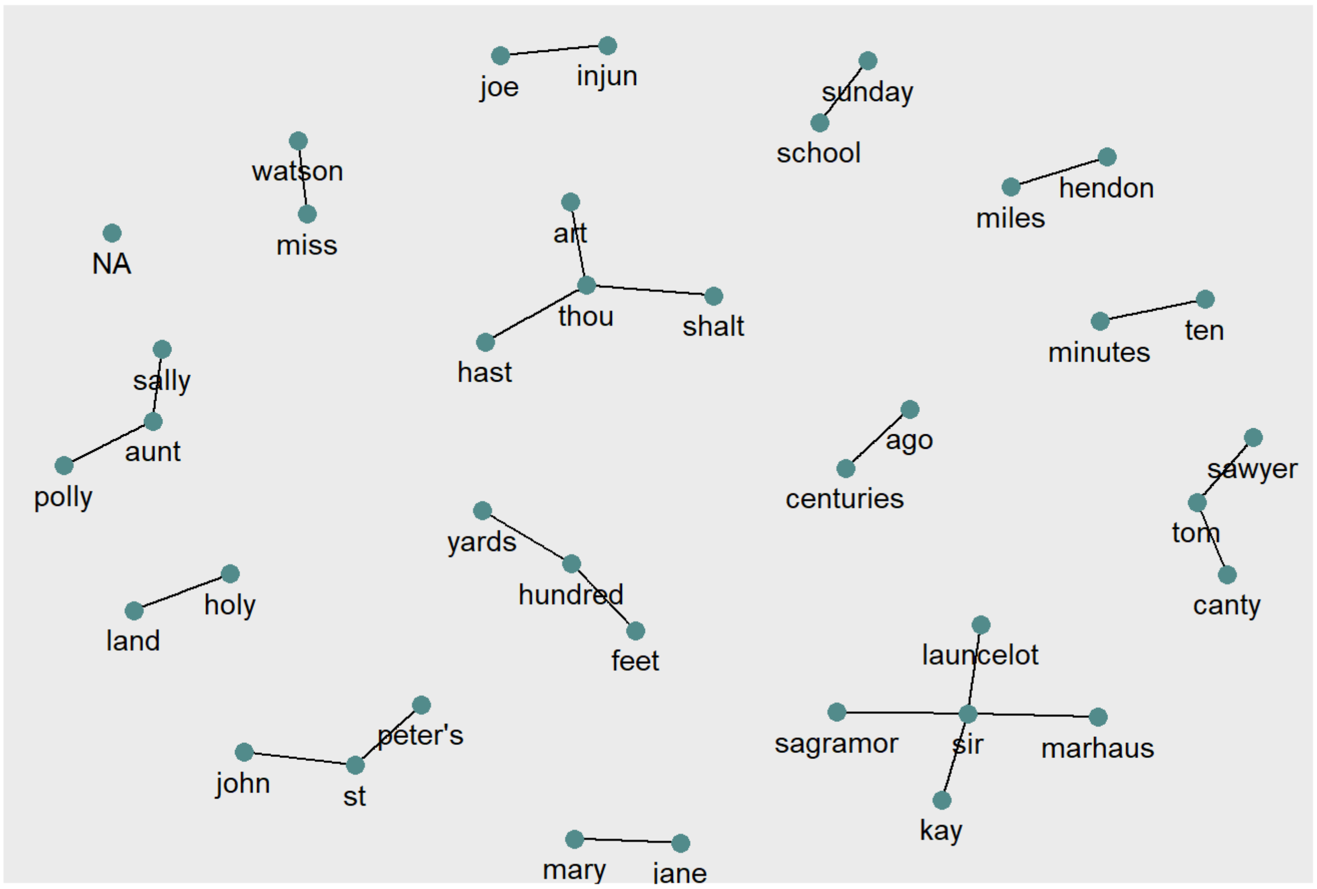
```
library(ggraph)
```

```
## Warning: package 'ggraph' was built under R version 4.2.3
```

```
set.seed(2017)
ggraph(bigram_graph, layout = "fr") +
  geom_edge_link() +
  geom_node_point(color = "darkslategray4", size = 3) +
  geom_node_text(aes(label = name), vjust = 1.8) + ggtitle("Common Bigrams in Twain's No
vels")
```

```
## Warning: Using the `size` aesthetic in this geom was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` in the `default_aes` field and elsewhere instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

Common Bigrams in Twain's Novels



Through text mining and sentiment analysis of Mark Twain's novels using R,

we gained valuable insights into his writing style, frequent word usage, and emotional tone.

This project sheds light on the themes and sentiments prevalent in Twain's literary works,

providing a deeper understanding and appreciation of his contributions to literature.