**TEXT MINING AND NATURAL LANGUAGE PROCESSING**

**USING NAIVE BAYES AND SVM**

**A PROJECT REPORT**

CSA1672-Data warehousing and Data Mining for Web Data Mining

**Submitted to**

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

**In partial fulfilment of the award of the degree of**

BACHELOR OF ENGINEERING IN COMPUTERSCIENCE

**By**

P.S.S.RAMCHAND [192225113]

**Supervisor**

Dr.PORKODI



**SAVEETHA SCHOOL OF ENGINEERING**

**SIMATS CHENNAI- 602105**

**JUNE-2024**

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **Title** | **Page. No.** |
| 1 | ABSTRACT | 3 |
| 2 | INTRODUCTION | 3-4 |
| 3 | MATERIALS AND METHODS | 4-6 |
| 4 | RESULTS AND  DISCUSSIONS | 6-7 |
| 5 | CONCLUSIONS | 7 |
| 6 | REFERENCES | 7 |

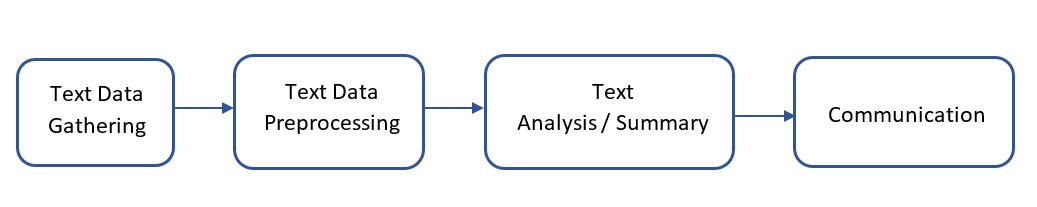
## **Abstract:Text Mining and Text Analysis**

Depending on how it is organized, data can be grouped into two categories: ****structured data**** and ****unstructured data****. ****Structured data**** is data that has been predefined and formatted to a tabular format with rows and columns, such as data stored in a relational database, or membership information housed in an Excel spreadsheet. While ****unstructured data**** does not have a predefined data format. It comes in various formats, for example, email, presentation, images, etc. Another category is a blend between structured and unstructured data formats, which is called ****semi-structured data****. It refers to what would normally be considered unstructured data, but that also has metadata that identifies certain characteristics. Some common examples of semi-structured data are XML, JSON, and HTML files.

****Text mining****, or ****text analysis****, is the ****process of exploring and analyzing unstructured or semi-structured text data to identify key concepts, pattens, relationships, or other attributions of the data****. Text mining began with the computational and information management areas, whereas text analysis originated in the humanities with the manual analysis of text such as newspaper indexes and Bible concordances. Now these two terms are exchangeable, and generally refer to ****the use of computational methods to explore and analyze unstructured text data****.

## **Introduction::Simplified Process of Text Mining Projects**

A simplified process of a typical text mining study can include four steps: data gather, text preprocessing or cleaning, text analysis, and communication.



Simplified Process of Typical Text Mining Projects

In terms of data gathering, we may create a dataset or select existing datasets.

After a dataset is generated, usually, we need to preprocess or clean the text to get it ready for analysis. Common techniques used for preparing a dataset include converting text to lower case, removing punctuations and non-alphanumeric characters, remove stopwords, tokenization, tagging parts of speech, word replacement, stemming and lemmatization, etc.

Next step will be text mining or analysis. Some common text mining methods are topic modelling, sentiment analysis, term frequency and TF-IDF, and collocation analysis.

We will then communicate the findings from text mining through publications, conferences, or other channels.

Various text preprocessing techniques and text mining methods serve different research purposes. This lesson is to demo how to use the R package tidytext to preprocess text data from an existing dataset to perform a sentiment analysis.

## **Preprocess and Clean Text**

### Tidy Data and Tidy Text

R is powerful processing structured data, or tabular data, where data display in columns or tables. R can also handle unstructured and semi-structured data such as text. Julia Silge and David Robinson followed the [tidy data principles](https://doi.org/10.18637/jss.v059.i10) branded by [Hadley Wickham](https://hadley.nz/) and developed the package tidytext to preprocess and analyze textual data.

Tidy data sets allow manipulation with a standard set of “tidy” tools, including popular packages such as dplyr ([Wickham and Francois 2016](https://www.tidytextmining.com/references.html" \l "ref-R-dplyr)), tidyr ([Wickham 2016](https://www.tidytextmining.com/references.html" \l "ref-R-tidyr)), ggplot2 ([Wickham 2009](https://www.tidytextmining.com/references.html" \l "ref-R-ggplot2)), and broom ([Robinson 2017](https://www.tidytextmining.com/references.html" \l "ref-R-broom)). These packages extend the capacities of tidytext of exploring and visualizing textual data. Users can transit fluidly between these packages by keeping the input and output in tidy formats.

### Token and Tokenization

Many text mining or analysis methods are based on counting words, phrases, sentences, or other meanful segments. Spliting textual data into segments enables the computer to count and perform other calculations. These segments are called ****tokens**** and the process of splitting textual data is called ****tokenization****.

In the package tidytext, ****tidy text**** is defined as a one-token-per-row data frame, where a ****token**** can be a character, a word, a n-gram, a sentence, a paragraph, a tweet, etc.

Here is a simple example to explain how to use tidytext to tokenize textual data. In R, textual data can be stored as character vectors. For example:

### R

lyrics <- c("How many roads must a man walk down", "Before you call him a man?", "How many seas must a white dove sail", "Before she sleeps in the sand?", "Yes, and how many times must the cannonballs fly", "Before they're forever banned?")lyrics

### OUTPUT

[1] "How many roads must a man walk down"

[2] "Before you call him a man?"

[3] "How many seas must a white dove sail"

[4] "Before she sleeps in the sand?"

[5] "Yes, and how many times must the cannonballs fly"

[6] "Before they're forever banned?"

To tokenize this character vector, we first need to put it into a data frame. We use the function [tibble](https://tibble.tidyverse.org/reference/tibble.html) from the package tidyverse to convert a character vector into a tibble.

### R

install.package("tidyverse")library(tidyverse)lyrics\_df <- tibble(line = 1:6, lyrics)lyrics\_df

### OUTPUT

# A tibble: 6 × 2

line lyrics

<int> <chr>

1 1 How many roads must a man walk down

2 2 Before you call him a man?

3 3 How many seas must a white dove sail

4 4 Before she sleeps in the sand?

5 5 Yes, and how many times must the cannonballs fly

6 6 Before they're forever banned?

Next, we will use the function [unnest\_tokens](https://rdrr.io/pkg/tidytext/man/unnest_tokens.html) to break the lyrics into words. In the meanwhile, punctuations are stripped.

The function unnest\_tokens has three primary arguments:

1. tbl: the data frame to be tokenized.
2. output: the column to be created as string or symbol.
3. input: the column that gets split as string or symbol.

### R

install.package("tidytext")library(tidytext)unnest\_tokens(tbl = lyrics\_df, output = word, input = lyrics)

### OUTPUT

# A tibble: 41 × 2

line word

<int> <chr>

1 1 how

2 1 many

3 1 roads

4 1 must

5 1 a

6 1 man

7 1 walk

8 1 down

9 2 before

10 2 you

# … with 31 more rows

The result of unnest\_tokens is a tibble. In our case, the lyrics is split into 41 words with each word takes a row. The input column lyrics is removed; the new column, or the output column word, is added; and the column line is kept unchanged.

Beyond these three primary arguments, the function [unnest\_tokens](https://rdrr.io/pkg/tidytext/man/unnest_tokens.html) also has several optional arguments. The default token is “words”. It can be set as “characters”, “sentences”, “n-grams”, “lines”, “paragraphs”, etc. unnest\_tokens automatically converts tokens to lowercase and drops the input column if not specified. Punctuations are stripped during the tokenization. Converting text to lower case and removing punctuations are also common text preprocess or cleaning techniques. Since the function [unnest\_tokens](https://rdrr.io/pkg/tidytext/man/unnest_tokens.html) can fulfill these tasks, we do not need to perform them separately.

Since the first argument of unnest\_tokens is a data frame, we can also use pipes to send a data frame to it and obtain the same results:

### R

lyrics\_df %>% unnest\_tokens(word, lyrics)

### Stop Words

When analyzing text, usually, some extremely common words such as “the”, “have”, “is”, “are” are of little value in serving the research purposes. We want to exclude them from the textual data entirely. These words are called ****stop words****. Removing ****stop word**** is one of the common text preprocessing techniques, which allows researchers to focus on the important words in the textual data instead. There is no single universal list of stop words used by all text analysis tools, nor any agreed upon rules for identifying stop words, and indeed not all tools even use such a list. Therefore, any group of words can be chosen as the stop words for a given purpose.

R package stopwords provides stop word lists for multiple languages and sources. It is easily extended. The package tidytext also offers a data frame, stop\_words, to host English stop words from three lexicons - onix, SMART, and snowball, with non-ASCII characters removed. The data frame stop\_words includes 1,149 stop words. We use it in this lesson when excluding stop words from our data sets.

We can use the function [anti\_join](https://www.rdocumentation.org/packages/dplyr/versions/0.7.8/topics/join) to exclude stop words from the textural data set. For example:

### R

lyrics\_df %>% unnest\_tokens(word, lyrics) %>% anti\_join(stop\_words)

### OUTPUT

# A tibble: 14 × 2

line word

<int> <chr>

1 1 roads

2 1 walk

3 2 call

4 3 seas

5 3 white

6 3 dove

7 3 sail

8 4 sleeps

9 4 sand

10 5 times

11 5 cannonballs

12 5 fly

13 6 forever

14 6 banned

After removing the stop words, only 14 words left in the lyrics.

## **Apply Data Preprocessing to a Text Mining Project**

### Project Gutenberg Collection

In terms of data gathering, we can create our own data sets or use existing textual datasets. In this lesson, we will use the [Project Gutenberg](https://www.gutenberg.org/) as the source of our data sets. The [Project Gutenberg](https://www.gutenberg.org/) is a collection of free electronic books, or eBooks, available online. The R package [gutenbergr](https://cran.r-project.org/web/packages/gutenbergr/vignettes/intro.html), developed by [David Robinson](https://en.wikipedia.org/wiki/David_G._Robinson_(data_scientist)), allows users to download public domain works from the Project Gutenberg collection as well as search and filter works by author, title, language, subjects, and other metadata. Project Gutenberg ID is one of the most important metadata, which we can use to download the text for each novel.

Let’s use [The Time Machine](https://www.gutenberg.org/ebooks/35) as an example to see how to find the Gutenberg ID and download the text.

### R

install.packages("gutenbergr")library(gutenbergr)gutenberg\_metadata %>% filter(title == "The Time Machine")

### OUTPUT

# A tibble: 3 × 8

gutenberg\_id title author gutenberg\_author\_id language gutenberg\_bookshelf rights has\_t…¹

<int> <chr> <chr> <int> <chr> <chr> <chr> <lgl>

1 35 The Time Machine Wells, H. G. (Herbert George) 30 en Science Fiction/Movie Books Public domain in the USA. TRUE

2 6620 The Time Machine Wells, H. G. (Herbert George) 30 en Movie Books/Science Fiction Copyrighted. Read the copyright no… FALSE

3 17401 The Time Machine Wells, H. G. (Herbert George) 30 en Movie Books Copyrighted. Read the copyright no… FALSE

# … with abbreviated variable name ¹ has\_text

### R

time\_machine <- gutenberg\_download(35)time\_machine

### OUTPUT

# A tibble: 3,174 × 2

gutenberg\_id text

<int> <chr>

1 35 "The Time Machine"

2 35 ""

3 35 "An Invention"

4 35 ""

5 35 "by H. G. Wells"

6 35 ""

7 35 ""

8 35 "CONTENTS"

9 35 ""

10 35 " I Introduction"

# … with 3,164 more rows

# ℹ Use `print(n = ...)` to see more rows

We can preprocess the text by tokenizing it words, removing punctuations, converting it to lower case, and removing stop words. The clean data has 11,268 rows and each row contains one word.

### R

tidy\_time\_machine <- time\_machine %>% unnest\_tokens(word,text) %>% anti\_join(stop\_words) tidy\_time\_machine

### OUTPUT

# A tibble: 11,268 × 2

gutenberg\_id word

<int> <chr>

1 35 time

2 35 machine

3 35 invention

4 35 contents

5 35 introduction

6 35 ii

7 35 machine

8 35 iii

9 35 time

10 35 traveller

# … with 11,258 more rows

# ℹ Use `print(n = ...)` to see more rows

## **Word Frequencies**

Since many text analysis methods are based on word counts in the textual data, we can first calculate word counts or word frequency. Word frequency looks at how often words are repeated in texts. To count the words, we can use the function [count](https://dplyr.tidyverse.org/reference/count.html) from the package [dplyr](https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html).

### R

tidy\_time\_machine %>% count(word, sort = TRUE)

### OUTPUT

# A tibble: 4,172 × 2

word n

<chr> <int>

1 time 207

2 machine 88

3 white 61

4 traveller 57

5 hand 49

6 morlocks 48

7 people 46

8 weena 46

9 found 44

10 light 43

# … with 4,162 more rows

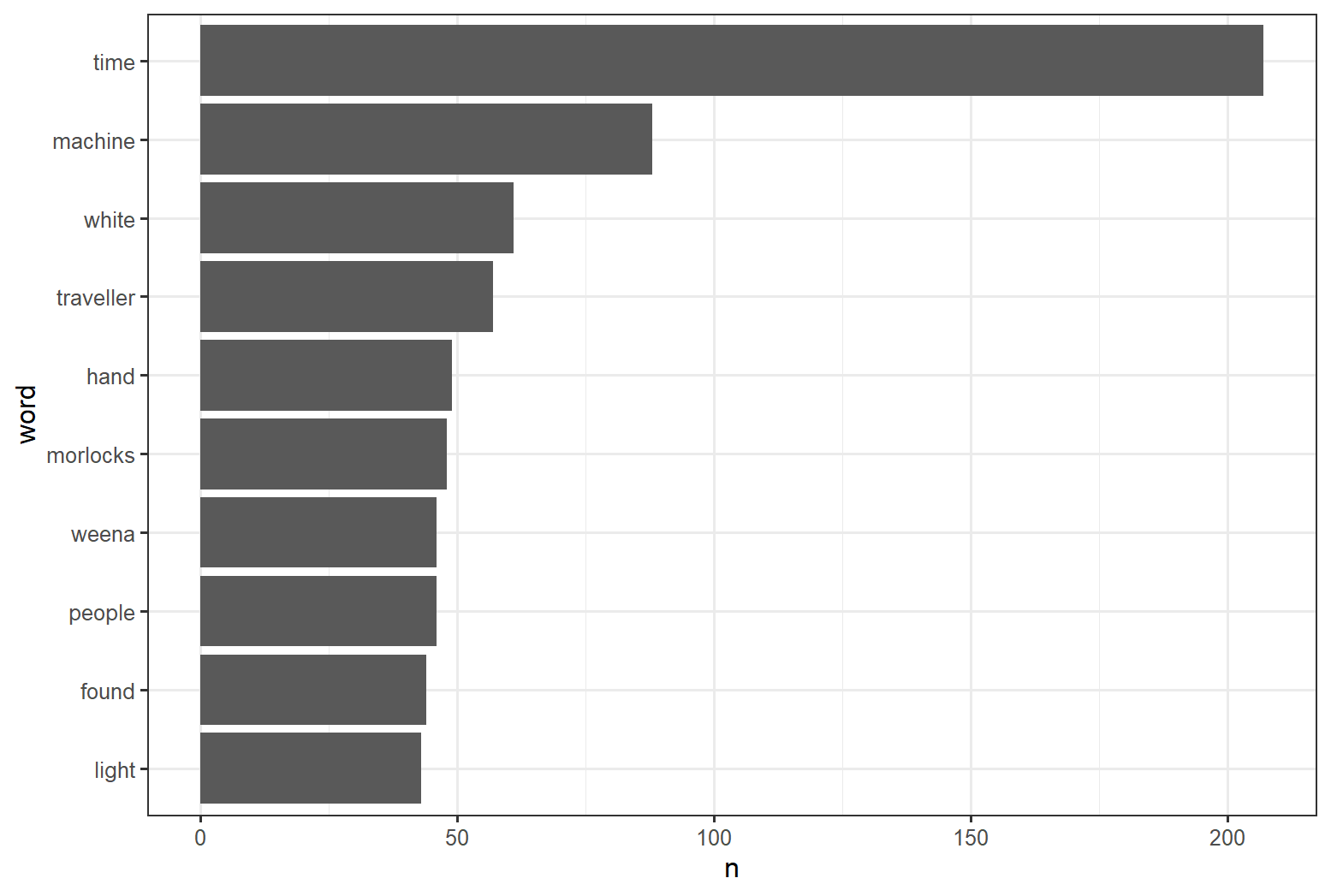
# ℹ Use `print(n = ...)` to see more rows

After removing stop words, the novel The Time Machine contains 11,268 words, where 4,172 are unique. The word time is most used word and it appears 207 times in the novel.

Beyond displaying the word frequencies in a table, we can also visualize it using the package [ggplot2](https://cran.r-project.org/web/packages/ggplot2/index.html) or the packages [wordcloud](https://cran.r-project.org/web/packages/wordcloud/wordcloud.pdf).

### R

tidy\_time\_machine %>% count(word, sort = TRUE) %>% filter(n > 40) %>% mutate(word = reorder(word, n)) %>% ggplot(aes(n, word))+ geom\_col()+ theme\_bw()

The output is a column chart: 

**REFERENCES:**

1. Gupta R, Gupta H, Mohania M. Cloud computing and big data analytics: what is new from databases perspective? In: *1st International Conference on Big Data Analytics* (*BDA*), New Delhi, India, 2012, 42–61.
2. Agrawal D, Das S, Abbadi AE. Big data and cloud computing: current state and future opportunities. In: *14th International Conference on Extending Database Technology* (*EDBT*), Uppsala, Sweden, 2011, 530–533.
3. Madden S. From databases to big data. *IEEE Internet Comput* 2012, **16**: 4–6.
4. Kraska T. Finding the needle in the big data systems haystack. *IEEE Internet Comput* 2013, **17**: 84–86.
5. Alpaydin E. *Introduction to Machine Learning*. 2nd ed. Cambridge, MA: MIT Press; 2010.
6. Witten IH, Frank E, Hall MA. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Amsterdam: Morgan Kaufmann; 2011.
7. March ST, Hevner AR. Integrated decision support systems: a data warehousing perspective. *Decis Support Syst* 2007, **43**: 1031–1043.
8. Watson HJ, Wixom BH. The current state of business intelligence. *Computer* 2007, **40**: 96–99.
9. Provost F, Fawcett T. *Data Science for Business. What You Need to Know about Data Mining and Data-Analytic Thinking*. 1st ed. Sebastopol, CA: O'Reilly; 2013.
10. Mell P, Grance T. The NIST definition of cloud computing (draft) recommendations of the national institute of standards and technology. *NIST Spec Publ* 2011, **145**.