P6이 마지막 과제
중간 이후 다 베이스가 되어있고 시작한다

TEXT MINING

Lecture 10

SENTIMENT ANALYSIS

KEUNGOUI KIM

awekim@handong.edu

Handong Global University, 2024-2.

Sentiment Analysis

감성

Sentiment in Text

- Sentiment
  - a view of or attitude toward a situation or event; an opinion [Oxford Languages]

- Sentiment in text
  - In (unstructured) text data, there is also a sentiment or tone.
  - The tone of the text can be captured either by the term or the context.

  - Ex) “This match was tragic but Lorris was extraordinary.”
  - Ex) “Dier was shocking.”
  - Ex) “Conte and Perisic's Dangerous Cohabitation.”

- Also known as opinion mining
  - Use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information [Wikipedia]
  - Contextual mining of text which identifies and extracts subjective information in source material, and helping a business to understand the social sentiment of their brand, product or service while monitoring online conversations [towarddatascience.com]

  - Sentiment analysis allows us to capture “the emotional tone behind a body of text”

- How to label "sentiment" in such cases?
  - Classifying with emoticons whether a tweet is "positive" or "negative" (Read, 2005; Park & Paroubek, 2010)

    ex. (positive) ^^, ^-^, (negative) -_-, --;;;
  - Classifying with the number of review points whether the review is "positive " or "negative" (Pang et al., 2002)

    ex. (positive) ★ ★ ★ ★, (negative) ★ ★ ★
  - Classifying with a politician's behavior whether he is "positive" or "negative" on certain legitimate (Thomas et al., 2006)

Applications of Sentiment Analysis

- Subjectivity detection
  - Under the assumption that any assumptions or hypotheses are based on the subjective opinion, sentiment analysis can be applied to determine the subjective opinion.

- Opinion classification
  - Sentiment analysis can be used to determine one’s opinion

- Targeted sentiment analysis
  - Allows us to distinguish the writer’s sentiment on the target object

Sentiment Analysis

- Two approaches
  - Lexicon-based approach: determining sentiment using existing knowledge
  - ML-based (supervised) approach: determining sentiment using contextual knowledge

Sentiment Lexicon Resource

dictionary

- AFINN
  - Developed by Finn Årup Nielsen
  - Sentiment values with a range of -5 to 5.

  Finn Årup Nielsen, "A new ANEW: evaluation of a word list for sentiment analysis in microblogs", Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages. Volume 718 in CEUR Workshop Proceedings: 93-98. 2011 May. Matthew Rowe, Milan Stankovic, Aba-Sah Dadzie, Mariann Hardey (editors)

	word	value
	<chr>	<dbl>
1	breathtaking	5
2	hurrah	5
3	outstanding	5
4	superb	5
5	thrilled	5

	word	value
	<chr>	<dbl>
1	some kind	0

	word	value
	<chr>	<dbl>
1	bastard	-5
2	bastards	-5
3	bitch	-5
4	bitches	-5
5	cock	-5

Sentiment Lexicon Resource

- (Bing Liu's) BING
  - Positive or negative.
  - Opinion word expansion and target extraction through double propagation
  - One of the well-known opinion lexicon resources

G Qiu, B Liu, J Bu, C Chen. 2011. Computational linguistics, 37(1): 9-27.

	word	sentiment		word	sentiment
	<chr>	<chr>		<chr>	<chr>
1	2-faces	negative	1	abound	positive
2	abnormal	negative	2	abounds	positive
3	abolish	negative	3	abundance	positive
4	abominable	negative	4	abundant	positive
5	abominably	negative	5	accessable	positive
6	abominate	negative	6	accessible	positive
7	abomination	negative	7	acclaim	positive
8	abort	negative	8	acclaimed	positive
9	aborted	negative	9	acclamation	positive
10	aborts	negative	10	accolade	positive

Sentiment Lexicon Resource

- EmoLex
  - NRC Word-Emotion Association Lexicon
  - Sentiment words (ex. Trust, fear, negativity , sadness, anger, surprise, positivity, disgust, joy, anticipation)
  - Extracted sentiment information from crowd-sourcing

Mohammad, S. M. & Turney, P. D. 2013. Crowdsourcing a word-emotion association lexicon, Computational Intelligence, 29(3): 436-465.

	word	sentiment
	<chr>	<chr>
1	abacus	trust
2	abbot	trust
3	absolution	trust
4	abundance	trust
5	academic	trust
6	accolade	trust
7	accompaniment	trust
8	accord	trust
9	account	trust
10	accountability	trust

	word	sentiment
	<chr>	<chr>
1	abandon	fear
2	abandoned	fear
3	abandonment	fear
4	abduction	fear
5	abhor	fear
6	abhorrent	fear
7	abominable	fear
8	abomination	fear
9	abortion	fear
10	absence	fear

	word	sentiment
	<chr>	<chr>
1	abandon	sadness
2	abandoned	sadness
3	abandonment	sadness
4	abduction	sadness
5	abortion	sadness
6	abortive	sadness
7	abscess	sadness
8	absence	sadness
9	absent	sadness
10	absentee	sadness

Sentiment Lexicon Resource

- Laughran-McDonald sentiment lexicon
  - Specialized in the financial field
  - Positive, negative, litigious, uncertain, constraining, superfluous
  - Not recommended to use for text from other fields

	word	sentiment
	<chr>	<chr>
1	able	positive
2	abundance	positive
3	abundant	positive
4	acclaimed	positive
5	accomplish	positive
6	accomplished	positive
7	accomplishes	positive
8	accomplishing	positive
9	accomplishment	positive
10	accomplishments	positive

	word	sentiment
	<chr>	<chr>
1	abandon	negative
2	abandoned	negative
3	abandoning	negative
4	abandonment	negative
5	abandonments	negative
6	abandons	negative
7	abdicated	negative
8	abdicates	negative
9	abdicating	negative
10	abdication	negative

	word	sentiment
	<chr>	<chr>
1	abovementioned	litigious
2	abrogate	litigious
3	abrogated	litigious
4	abrogates	litigious
5	abrogating	litigious
6	abrogation	litigious
7	abrogations	litigious
8	absolve	litigious
9	absolved	litigious
10	absolves	litigious

	word	sentiment
	<chr>	<chr>
1	abeyance	uncertainty
2	abeyances	uncertainty
3	almost	uncertainty
4	alteration	uncertainty
5	alterations	uncertainty
6	ambiguities	uncertainty
7	ambiguity	uncertainty
8	ambiguous	uncertainty
9	anomalies	uncertainty
10	anomalous	uncertainty

	word	sentiment
	<chr>	<chr>
1	aegis	superfluous
2	amorphous	superfluous
3	anticipatory	superfluous
4	appertaining	superfluous
5	assimilate	superfluous
6	assimilating	superfluous
7	assimilation	superfluous
8	bifurcated	superfluous
9	bifurcation	superfluous
10	cessions	superfluous

- textdata::get_sentiments('*name*')
  - Imports sentiments list
  - afinn / bing / nrc

install.packages("tidytext")

```
> install.packages('textdata')
trying URL 'https://cran.rstudio.com/bin/macosx/contrib/
Content type 'application/x-gzip' length 499072 bytes (4
==================================================
downloaded 487 KB


The downloaded binary packages are in
        /var/folders/_l/bsz42vx93p185vfk77pkvksr0000gn/T
s
```

library(tidytext)

```
> get_sentiments('afinn')
Do you want to download:
 Name: AFINN-111
 URL: http://www2.imm.dtu.dk/pubdb/views/publication_det
 License: Open Database License (ODbL) v1.0
 Size: 78 KB (cleaned 59 KB)
 Download mechanism: https

1: Yes
2: No

Selection: 1
```

```
> get_sentiments('afinn')
# A tibble: 2,477 × 2
```

	word <chr>	value <dbl>
1	abandon	-2
2	abandoned	-2
3	abandons	-2
4	abducted	-2
5	abduction	-2
6	abductions	-2
7	abhor	-3
8	abhorred	-3
9	abhorrent	-3
10	abhors	-3

```
# … with 2,467 more rows
```

```
> get_sentiments('bing')
# A tibble: 6,786 × 2
```

	word <chr>	sentiment <chr>
1	2-faces	negative
2	abnormal	negative
3	abolish	negative
4	abominable	negative
5	abominably	negative
6	abominate	negative
7	abomination	negative
8	abort	negative
9	aborted	negative
10	aborts	negative

```
# … with 6,776 more rows
```

```
> get_sentiments('nrc')
# A tibble: 13,872 × 2
```

	word <chr>	sentiment <chr>
1	abacus	trust
2	abandon	fear
3	abandon	negative
4	abandon	sadness
5	abandoned	anger
6	abandoned	fear
7	abandoned	negative
8	abandoned	sadness
9	abandonment	anger
10	abandonment	fear

```
# … with 13,862 more rows
```

Since all these lists are a set of "words," we need to create a word data set.

Sentiment Sets

- Since sentiment sets are case-sensitive, any sentiment sets can be used.
- If possible, using a specific type of sentiment list that matches the context of the text data sample is recommended

```
> get_sentiments('afinn') %>% filter(word=='abandon')
# A tibble: 1 × 2
  word    value
  <chr>   <dbl>
1 abandon    -2
> get_sentiments('bing') %>% filter(word=='abandon')
# A tibble: 0 × 2
# … with 2 variables: word <chr>, sentiment <chr>
> get_sentiments('nrc') %>% filter(word=='abandon')
# A tibble: 3 × 2
  word    sentiment
  <chr>   <chr>
1 abandon fear
2 abandon negative
3 abandon sadness
```

Lexicon-based Sentiment Analysis Process I

Lexicon-based Sentiment Analysis

- Lexicon-based sentiment analysis
  - Analyzing the “tone” of the document based on the list of words and their frequency
  - Simply speaking, lexicon-based sentiment analysis is conducting a text frequency analysis with the sentiment lexicon.
  - To do so, a sentiment lexicon resource is needed.

- Analyze Jane Austen's famous novel "Pride & Prejudice"

- Import the data
  - janeaustenr::austen_books() returns Jane Austen's six major novels
  - Jane Austen's famous novels are presented in a data.frame format → It will make our life much easier!!

```
> library(janeaustenr)
> austen_books() %>% dplyr::select(book) %>% unique
# A tibble: 6 × 1
  book
  <fct>
1 Sense & Sensibility
2 Pride & Prejudice
3 Mansfield Park
4 Emma
5 Northanger Abbey
6 Persuasion
> austen_books() %>% dplyr::select(text) %>% unique %>% data.frame %>%
+   head(30)
                                                                      text
1                                                    SENSE AND SENSIBILITY
2
3                                                           by Jane Austen
4                                                                   (1811)
5                                                                CHAPTER 1
6   The family of Dashwood had long been settled in Sussex.  Their estate
7    was large, and their residence was at Norland Park, in the centre of
8       their property, where, for many generations, they had lived in so
9     respectable a manner as to engage the general good opinion of their
10  surrounding acquaintance.  The late owner of this estate was a single
11   man, who lived to a very advanced age, and who for many years of his
```

Step 2) Text Pre-processing

- Create a variable called *pp* that only contains “Pride & Prejudice”
  - It should not contain any row with missing values
  - Remove the first two rows (Title and author name)

```
> pp <- austen_books() %>%
+   filter(book == "Pride & Prejudice" & text !="")
> pp %>% head
# A tibble: 6 × 2
  text                                                                   book
  <chr>                                                                  <fct>
1 PRIDE AND PREJUDICE                                                    Pride & Preju…
2 By Jane Austen                                                         Pride & Preju…
3 Chapter 1                                                              Pride & Preju…
4 It is a truth universally acknowledged, that a single man in possession Pride & Preju…
5 of a good fortune, must be in want of a wife.                          Pride & Preju…
6 However little known the feelings or views of such a man may be on his  Pride & Preju…
> pp %<>% filter(text != "PRIDE AND PREJUDICE") %>%
+   filter(text != "By Jane Austen")
> pp %>% head
# A tibble: 6 × 2
  text                                                                    book
  <chr>                                                                   <fct>
1 Chapter 1                                                               Pride & Prej…
2 It is a truth universally acknowledged, that a single man in possession Pride & Prej…
3 of a good fortune, must be in want of a wife.                           Pride & Prej…
4 However little known the feelings or views of such a man may be on his  Pride & Prej…
5 first entering a neighbourhood, this truth is so well fixed in the minds Pride & Prej…
6 of the surrounding families, that he is considered the rightful property Pride & Prej…
```

Step 2) Text Pre-processing

- Create a new column called “ch”
  - Try

```
> pp %>% head
# A tibble: 6 x 3
  text                                                                  book                  ch
  <chr>                                                                 <fct>              <dbl>
1 It is a truth universally acknowledged, that a single man in possession Pride & Prejudice     1
2 of a good fortune, must be in want of a wife.                         Pride & Prejudice     1
3 However little known the feelings or views of such a man may be on his Pride & Prejudice     1
4 first entering a neighbourhood, this truth is so well fixed in the minds Pride & Prejudice    1
5 of the surrounding families, that he is considered the rightful property Pride & Prejudice    1
6 of some one or other of their daughters.                              Pride & Prejudice     1
```

Step 2) Text Pre-processing

- Create a new column called “ch”
  - “ch” represents the chapter number
  - Check the data and write down codes for generating this new column

```
> pp %>% head
# A tibble: 6 × 2
  text                                                          book
  <chr>                                                         <fct>
1 Chapter 1                                                     Pride & Prej…
2 It is a truth universally acknowledged, that a single man in possession  Pride & Prej…
3 of a good fortune, must be in want of a wife.                 Pride & Prej…
4 However little known the feelings or views of such a man may be on his   Pride & Prej…
5 first entering a neighbourhood, this truth is so well fixed in the minds Pride & Prej…
6 of the surrounding families, that he is considered the rightful property Pride & Prej…
```

Chapter numbers are included in the text column

Find the row index of the case including “Chapter”

```
> pp.ch.index <- which(substr(pp$text,1,7)=="Chapter")
> pp.ch.index
 [1]     1    80   157   303   399   485   694   876  1060  1211  1420  1568  1627  1772
[15]  1871  2017  2315  2429  2875  3039  3182  3357  3508  3654  3823  3958  4151  4266
[29]  4394  4612  4718  4854  4990  5156  5342  5591  5766  5887  5980  6114  6260  6464
[43]  6624  7050  7247  7399  7651  8001  8195  8393  8587  8763  9027  9282  9427  9637
[57]  9888 10035 10253 10473 10612
```

```
> pp$text[pp.ch.index]
 [1] "Chapter 1"  "Chapter 2"  "Chapter 3"
 [7] "Chapter 7"  "Chapter 8"  "Chapter 9"
[13] "Chapter 13" "Chapter 14" "Chapter 15"
[19] "Chapter 19" "Chapter 20" "Chapter 21"
[25] "Chapter 25" "Chapter 26" "Chapter 27"
[31] "Chapter 31" "Chapter 32" "Chapter 33"
```

- Create a new column called “ch”

```
> pp$ch <- 0
> for(i in 1:length(pp.ch.index)){
+   pp$ch[pp.ch.index[i]:(pp.ch.index[(i+1)]-1)] <- i
+ }
Error in pp.ch.index[i]:(pp.ch.index[(i + 1)] - 1) : NA/NaN argument
> pp$ch <-
+   ifelse(pp$ch==0, max(pp$ch)+1, pp$ch)
> pp %<>%
+   filter(substr(text,1,7)!="Chapter")
> pp %>%
+   filter(ch==max(ch))
# A tibble: 107 x 3
   text                                                         book      ch
   <chr>                                                        <fct>  <dbl>
 1 Happy for all her maternal feelings was the day on which Mrs. Bennet got Pride…    61
 2 rid of her two most deserving daughters. With what delighted pride      Pride…    61
 3 she afterwards visited Mrs. Bingley, and talked of Mrs. Darcy, may      Pride…    61
 4 be guessed. I wish I could say, for the sake of her family, that the    Pride…    61
 5 accomplishment of her earnest desire in the establishment of so many    Pride…    61
 6 of her children produced so happy an effect as to make her a sensible,  Pride…    61
 7 amiable, well-informed woman for the rest of her life; though perhaps it Pride…    61
 8 was lucky for her husband, who might not have relished domestic felicity Pride…    61
 9 in so unusual a form, that she still was occasionally nervous and       Pride…    61
10 invariably silly.                                                       Pride…    61
```

Error occurs because it does not cover the last chapter, but it’s ok.

과정이 필요함
이렇게 Jan책으로 시작해서 만들수있어 이렇게 만들잖아 21p로 이상태에서 감성분석하기 위해서 어떻게 하냐 챕터정보 없어서 챕터 정보를 추가해야해 (책단위와 책으로 보는건 다르니간) 문장이니깐 감정은 단어에다가 붙이니깐 22p로 고쳐 챕터별 단어 챕터별 단어 만약 너가 감성분석.... 21->22->23 과정을 알아야함

Step 2) Text Pre-processing

- Create chapter – token table called *pp.clean*
  - Space tokenization
  - Convert to lower cases
  - Remove stopwords ('smart' source)
  - Remove punctuation
  - Lemmatize

```
> '%ni%' <- Negate('%in%')
> library(textstem)
> pp.clean <- 0
> for (i in 1:length(unique(pp$ch))){
+   temp <- pp %>% filter(ch==i)
+   temp.text <- temp$text %>%
+     str_split(" ") %>% unlist %>% tolower
+   temp.text %<>%
+     str_remove_all("[:punct:]") %>%
+     lemmatize_words
+   temp.text <- temp.text[
+     temp.text %ni% stopwords::stopwords('en', source='smart')]
+   pp.clean <- rbind(pp.clean,
+                     data.frame(ch=i, token=temp.text))
+ }
> pp.clean <- pp.clean[2:nrow(pp.clean),]
> pp.clean %<>%
+   filter(token!="") %>%
+   filter(token!="mr") %>%
+   filter(token!="mrs")
> pp.clean %>% head
  ch        token
1  1        truth
2  1  universally
3  1  acknowledge
4  1       single
5  1          man
6  1   possession
```

Remove extra stopwords

- As a final step, create a summarized table called *pp.clean.sum*
  - *ch – token – n*
  - Try

ch	token	n
<dbl>	<chr>	<int>
1	dear	8
1	bennet	7
1	good	6
1	visit	6
1	bingley	5
1	man	5
1	daughter	4
1	girl	4
1	marry	4
1	single	4

Step 2) Text Pre-processing

- As a final step, create a summarized table called *pp.clean.sum*
  - *ch – token - n*

```
> pp.clean.sum <-
+   pp.clean %>% group_by(ch) %>%
+   count(token) %>%
+   arrange(ch,desc(n))
> pp.clean.sum %>% head(10)
# A tibble: 10 × 3
# Groups:   ch [1]
      ch token        n
   <dbl> <chr>    <int>
 1     1 dear         8
 2     1 bennet       6
 3     1 visit        6
 4     1 man          5
 5     1 bingley      4
 6     1 daughter     4
 7     1 girl         4
 8     1 marry        4
 9     1 single       4
10     1 wife         4
```

Step 2) Text Pre-processing – tidytext::unnest_token()

- tidytext::unnest_tokens() 더 강력한..
  - Split a column into tokens, flattening the table into one-token-per-row.
  - Uses *hunspell_parse tokenizer*: takes a character vector with text (plain, latex, man, html or xml format), parses out the words and returns a list with incorrect words for each line
  - Easy to implement, but not recommended for advanced work

```
> austen_books() %>%
+   filter(book == "Pride & Prejudice" & text !="")
# A tibble: 10,721 x 2
   text
   <chr>
 1 "PRIDE AND PREJUDICE"
 2 "By Jane Austen"
 3 "Chapter 1"
 4 "It is a truth universally acknowledged, that a single man in possession"
 5 "of a good fortune, must be in want of a wife."
 6 "However little known the feelings or views of such a man may be on his"
 7 "first entering a neighbourhood, this truth is so well fixed in the minds"
 8 "of the surrounding families, that he is considered the rightful property"
 9 "of some one or other of their daughters."
10 "\"My dear Mr. Bennet,\" said his lady to him one day, \"have you heard that"
# ... with 10,711 more rows
```

```
> austen_books() %>%
+   filter(book == "Pride & Prejudice" & text !="") %>%
+   unnest_tokens(word, text)
# A tibble: 122,204 x 2
   book              word
   <fct>             <chr>
 1 Pride & Prejudice pride
 2 Pride & Prejudice and
 3 Pride & Prejudice prejudice
 4 Pride & Prejudice by
 5 Pride & Prejudice jane
 6 Pride & Prejudice austen
 7 Pride & Prejudice chapter
 8 Pride & Prejudice 1
 9 Pride & Prejudice it
10 Pride & Prejudice is
# ... with 122,194 more rows
```

Lexicon-based Sentiment Analysis Process II

Step 3) Sentiment Analysis with ‘afinn’

- Meaning of sentiment value of ‘afinn’
  - High and positive sentiment value indicates a “positive” term
  - Low and negative sentiment value indicates a “negative” term

```
> pp.word.affin <- austen_books() %>%
+   filter(book == "Pride & Prejudice") %>%
+   unnest_tokens(word, text) %>%
+   inner_join(get_sentiments('afinn'))
Joining, by = "word"
> pp.word.affin %>%
+   left_join(pp.word.affin %>% count(word))
Joining, by = "word"
# A tibble: 7,783 x 4
   book              word      value     n
   <fct>             <chr>     <dbl> <int>
 1 Pride & Prejudice good          3   200
 2 Pride & Prejudice want          1    44
 3 Pride & Prejudice dear          2   158
 4 Pride & Prejudice no           -1   490
 5 Pride & Prejudice want          1    44
 6 Pride & Prejudice cried        -2    91
 7 Pride & Prejudice want          1    44
 8 Pride & Prejudice no           -1   490
 9 Pride & Prejudice dear          2   158
10 Pride & Prejudice delighted     3    23
# … with 7,773 more rows
```

- Meaning of sentiment value of 'afinn'
  - High and positive sentiment value indicates a "positive" term
  - Low and negative sentiment value indicates a "negative" term

```
> pp.word.affin %>% group_by(book, word) %>%
+   dplyr::summarise(value=sum(value)) %>% arrange(desc(value))
`summarise()` has grouped output by 'book'. You can override us
argument.
# A tibble: 846 × 3
# Groups:   book [1]
   book             word      value
   <fct>            <chr>     <dbl>
 1 Pride & Prejudice good        600
 2 Pride & Prejudice great       426
 3 Pride & Prejudice dear        316
 4 Pride & Prejudice love        276
 5 Pride & Prejudice pleasure    276
 6 Pride & Prejudice happy       249
 7 Pride & Prejudice hope        242
 8 Pride & Prejudice happiness   216
 9 Pride & Prejudice better      184
10 Pride & Prejudice affection   174
# … with 836 more rows
```

```
> pp.word.affin %>% group_by(book, word) %>%
+   dplyr::summarise(value=sum(value)) %>% arrange(value)
`summarise()` has grouped output by 'book'. You can overri
argument.
# A tibble: 846 × 3
# Groups:   book [1]
   book             word   value
   <fct>            <chr>  <dbl>
 1 Pride & Prejudice miss    -566
 2 Pride & Prejudice no      -490
 3 Pride & Prejudice cried   -182
 4 Pride & Prejudice ill     -150
 5 Pride & Prejudice lost     -87
 6 Pride & Prejudice poor     -76
 7 Pride & Prejudice afraid   -74
 8 Pride & Prejudice leave    -62
 9 Pride & Prejudice alone    -60
10 Pride & Prejudice pain     -56
# … with 836 more rows
```

Step 3) Sentiment Analysis with 'bing'

- 'bing' tells us whether a word is "positive" or "negative"
  - Using this classification of "positive" and "negative," we can simply distinguish the tone of the word.

```
> pp.word.bing <- austen_books() %>%
+   filter(book == "Pride & Prejudice") %>%
+   unnest_tokens(word, text) %>%
+   inner_join(get_sentiments('bing'))
Joining, by = "word"
> pp.word.bing %>%
+   left_join(pp.word.bing %>% count(word))
Joining, by = "word"
# A tibble: 8,704 × 4
   book              word        sentiment     n
   <fct>             <chr>       <chr>     <int>
 1 Pride & Prejudice pride       positive     48
 2 Pride & Prejudice prejudice   negative      6
 3 Pride & Prejudice good        positive    200
 4 Pride & Prejudice fortune     positive     39
 5 Pride & Prejudice well        positive    224
 6 Pride & Prejudice rightful    positive      1
 7 Pride & Prejudice impatiently negative      5
 8 Pride & Prejudice objection   negative     15
 9 Pride & Prejudice enough      positive    106
10 Pride & Prejudice fortune     positive     39
# … with 8,694 more rows
```

Step 3) Sentiment Analysis with 'bing'

- 'bing' tells us whether a word is "positive" or "negative"
  - Comparing the frequency of positive and negative words
  - Finding the top 5 positive and negative words

```
> pp.word.bing %>% group_by(book, sentiment) %>%
+   dplyr::summarise(count=length(word))
`summarise()` has grouped output by 'book'. You car
argument.
# A tibble: 2 × 3
# Groups:   book [1]
  book             sentiment count
  <fct>            <chr>     <int>
1 Pride & Prejudice negative   3652
2 Pride & Prejudice positive   5052
> pp.word.bing %>% group_by(book, sentiment) %>%
+   dplyr::summarise(count=length(unique(word)))
`summarise()` has grouped output by 'book'. You car
argument.
# A tibble: 2 × 3
# Groups:   book [1]
  book             sentiment count
  <fct>            <chr>     <int>
1 Pride & Prejudice negative    838
2 Pride & Prejudice positive    592
```

```
> pp.word.bing %>% group_by(sentiment, word) %>%
+   dplyr::summarise(count=length(word)) %>% ungroup %>%
+   group_by(sentiment) %>% arrange(desc(count)) %>% slice(1:5)
`summarise()` has grouped output by 'sentiment'. You can overrid
`.groups` argument.
# A tibble: 10 × 3
# Groups:   sentiment [2]
   sentiment word       count
   <chr>     <chr>      <int>
 1 negative  miss         283
 2 negative  object        48
 3 negative  scarcely      45
 4 negative  impossible    44
 5 negative  poor          38
 6 positive  well         224
 7 positive  good         200
 8 positive  great        142
 9 positive  enough       106
10 positive  better        92
```

Step 3) Sentiment Analysis with 'nrc'

- 'nrc' provides the list of sentiments related to the word.
  - Using the list of sentiments, we can find the most representative sentiment that is being used in the text

```
> pp.word.nrc <- austen_books() %>%
+   filter(book == "Pride & Prejudice") %>%
+   unnest_tokens(word, text) %>%
+   inner_join(get_sentiments('nrc'))
Joining, by = "word"
> pp.word.nrc %>%
+   left_join(pp.word.nrc %>% count(word))
Joining, by = "word"
# A tibble: 29,064 × 4
   book              word       sentiment     n
   <fct>             <chr>      <chr>     <int>
 1 Pride & Prejudice pride      joy          96
 2 Pride & Prejudice pride      positive     96
 3 Pride & Prejudice prejudice  anger        12
 4 Pride & Prejudice prejudice  negative     12
 5 Pride & Prejudice truth      positive     54
 6 Pride & Prejudice truth      trust        54
 7 Pride & Prejudice possession anger        36
 8 Pride & Prejudice possession disgust      36
 9 Pride & Prejudice possession fear         36
10 Pride & Prejudice possession negative     36
# … with 29,054 more rows
```

Step 3) Sentiment Analysis with ‘nrc’

- ‘nrc’ provides the list of sentiments related to the word.

```
> pp.word.nrc %>%
+   left_join(pp.word.nrc %>% count(word)) %>%
+   group_by(sentiment) %>% dplyr::summarise(n=sum(n)) %>%
+   arrange(desc(n))
Joining, by = "word"
# A tibble: 10 x 2
   sentiment         n
   <chr>         <int>
 1 positive     860378
 2 trust        704070
 3 joy          680636
 4 anticipation 668688
 5 surprise     404757
 6 negative     244577
 7 sadness      189333
 8 fear         125082
 9 anger         85166
10 disgust       72395
```

```
> pp.word.nrc %>%
+   left_join(pp.word.nrc %>% count(word)) %>%
+   group_by(sentiment) %>% dplyr::summarise(n=sum(n)) %>%
+   arrange(n)
Joining, by = "word"
# A tibble: 10 x 2
   sentiment         n
   <chr>         <int>
 1 disgust       72395
 2 anger         85166
 3 fear         125082
 4 sadness      189333
 5 negative     244577
 6 surprise     404757
 7 anticipation 668688
 8 joy          680636
 9 trust        704070
10 positive     860378
```

Limitation of Lexicon-based Approach

- In general, a lexicon-based approach is recommended when analyzing a document with a large number of texts.

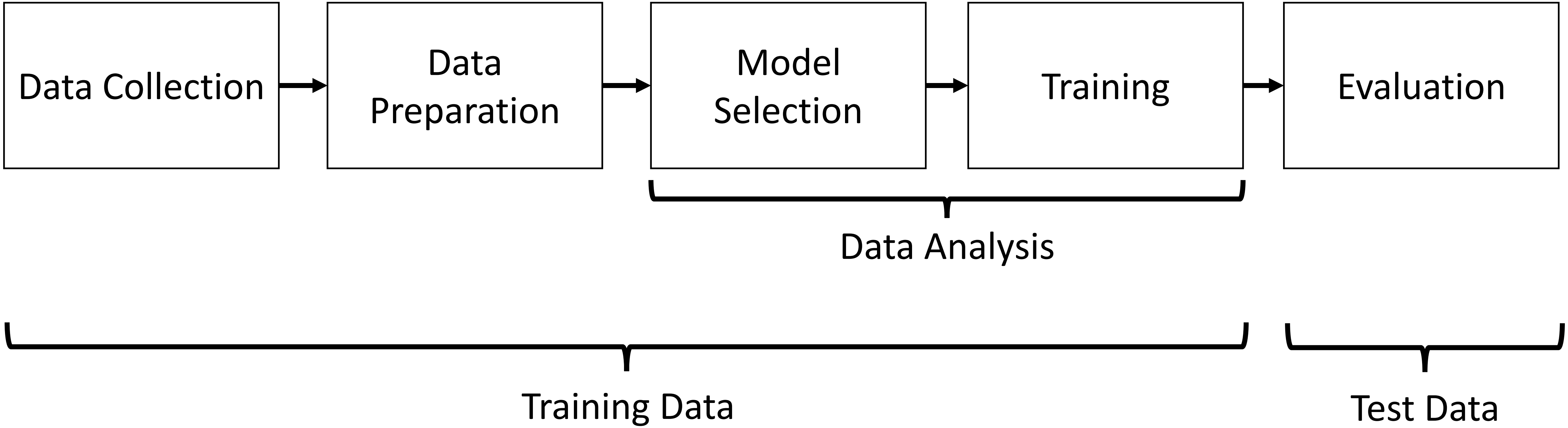
- If lexicon-based sentiment analysis is used for a short document, it may create an unrealistic result.
  - It’s because word frequency matters in a lexicon-based approach
  - An alternative solution is to use an n-gram word.

ML-based Sentiment Analysis

- Purpose of machine learning
  - Prediction
  - Classification or regression
- Key concept of machine learning
  - Training data
  - Test data
  - Sampling
  - Label
  - Performance check
  - Validation

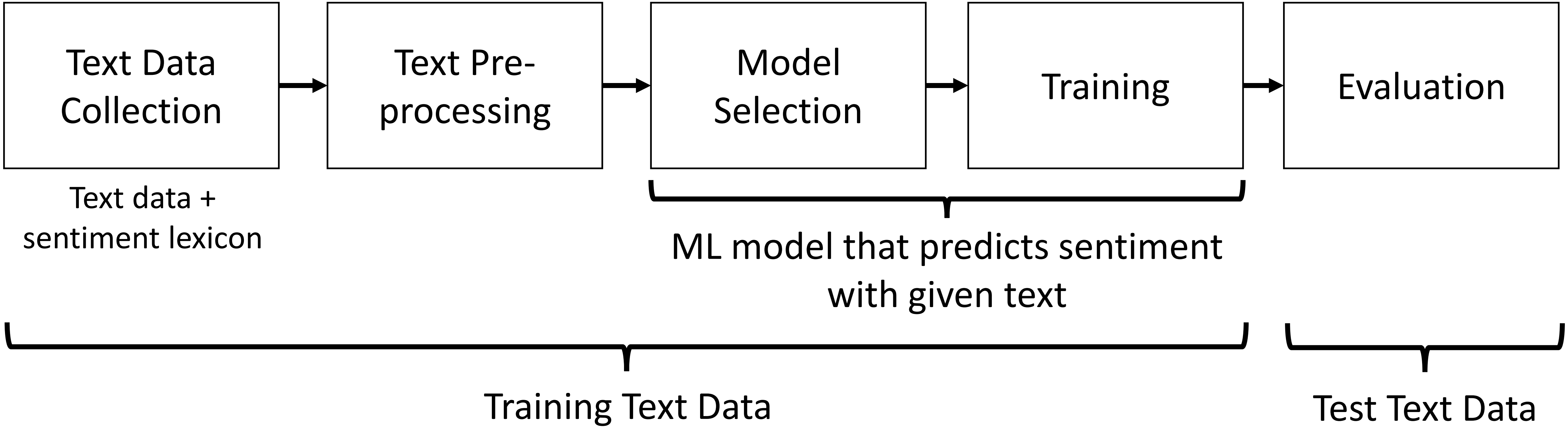
Machine Learning Overview

- Machine learning process



Machine Learning Overview

- ML-based sentiment analysis



Machine Learning in R

- caret (classification and regression training) package
  - R package that allows us to train different types of algorithms using a simple caret::train() function

- caret::train(x, y, method = ‘*name of ML*’)
  - method = ‘rpart’ ➔ Decision Tree
  - method = ‘ranger’ ➔ Random Forest
  - method = ‘xgbTree’ ➔ XGBoost
  - method = ‘knn’ ➔ K-Nearest Neighbor
  - method = ‘nnet’ ➔ Neural Network

ML-based
Sentiment Analysis Process

Step 1) Import data

- Create a ML model that predicts the sentiment of text
- Import the data

```
> load(file="R file/R file_LEC10/hs.df.RData")
> hs.df %>% head(1)

                                          text
1 I am LOVIN my Life right about now! I'm loving the people God is placing in my life. #Happy&amp;Focu
sed! Striving to be the BEST WOMAN I can be!
  label  type      ID
1 Happy train Happy_1
> hs.df %>% nrow
[1] 180
> hs.df %>% filter(type=="train") %>%
+   filter(label=="Happy") %>% nrow
[1] 80
> hs.df %>% filter(type=="train") %>%
+   filter(label=="Sad") %>% nrow
[1] 80
> hs.df %>% filter(type=="test") %>%
+   filter(label=="Happy") %>% nrow
[1] 10
> hs.df %>% filter(type=="test") %>%
+   filter(label=="Sad") %>% nrow
[1] 10
```

```
> table(hs.df$label, hs.df$type)

        test train
  Happy   10    80
  Sad     10    80
```

Train set

Test set

Step 2) Text Pre-processing

- Create a variable called *hs.df.clean* that only contains cleaned and tokenized text
  - Remember that this is a quick and simple text pre-processing…
  - Label – type – ID - word

```
> library(dplyr)
> library(stringr)
> library(magrittr)
> library(textstem)
> hs.df.clean <-
+   hs.df %>%
+   unnest_tokens(word, text, "words", to_lower=TRUE) %>%
+   mutate(word = lemmatize_words(word)) %>%
+   anti_join(stop_words, by="word")
> hs.df.clean %>% head
  label  type      ID   word
1 Happy train Happy_1  lovin
2 Happy train Happy_1   life
3 Happy train Happy_1   love
4 Happy train Happy_1 people
5 Happy train Happy_1    god
6 Happy train Happy_1   life
```

Each document's tokenized words listed in data.frame

Remove meaningless numbers, punctuations, etc.

```
> hs.df.clean$word.ed <-
+   hs.df.clean$word %>%
+   str_remove_all("[:digit:]{1,}") %>%
+   str_remove_all("[:lower:]{1,}\\.[:lower:]{1,}") %>%
+   str_remove_all("[:punct:]{1,}") %>%
+   str_remove_all("[^a-z0-9]")
> hs.df.clean %<>%
+   filter(word.ed!="")
> dim(hs.df.clean)
[1] 1028    5
```

Step 2) Text Pre-processing

- Create a label vector
  - Train label & Test label

```
> hs.df.train.label <-
+   hs.df$label[hs.df$type=="train"] %>%
+   as.factor
> hs.df.test.label <-
+   hs.df$label[hs.df$type=="test"] %>%
+   as.factor
```

- Create a matrix – train set
  - Document-Term Matrix

Any problems?

```
> hs.df.train.dtm <- hs.df.clean %>%
+   filter(type=="train") %>%
+   select(ID, word.ed) %>%
+   count(ID, word.ed) %>%
+   cast_dtm(document=ID, term=word.ed, value=n)
> hs.df.train.dtm
<<DocumentTermMatrix (documents: 160, terms: 481)>>
Non-/sparse entries: 865/76095
Sparsity           : 99%
Maximal term length: 21
Weighting          : term frequency (tf)
> hs.df.train.dtmatrix <-
+   hs.df.train.dtm %>% as.matrix
```

```
> hs.df.test.dtm <- hs.df.clean %>%
+   dplyr::filter(type=="test") %>%
+   select(ID, word.ed) %>%
+   count(ID, word.ed) %>%
+   cast_dtm(document=ID, term=word.ed, value=n)
> hs.df.test.dtm
<<DocumentTermMatrix (documents: 20, terms: 80)>>
Non-/sparse entries: 110/1490
Sparsity           : 93%
Maximal term length: 12
Weighting          : term frequency (tf)
> hs.df.test.dtmatrix <-
+   hs.df.test.dtm %>% as.matrix
```

- Create a matrix – test set
  - Document-Term Matrix

```
> hs.df.train.dtmatrix <-
+   hs.df.train.dtm %>% as.matrix
> train.vector <- hs.df.clean %>% dplyr::filter(type=="train") %>%
+   select(word.ed) %>% unique
> test.vector <- hs.df.clean %>% dplyr::filter(type=="test") %>%
+   select(word.ed) %>% unique
> train.test.missing <-
+   train.vector$word.ed[train.vector$word.ed %ni% test.vector$word.ed]
> train.test.missing.df <- data.frame(ID=NA,
+                                     word.ed=train.test.missing)
```

Find list of train-set words that are not included in the test-set

- Create a matrix – test set
  - Document-Term Matrix

```
> hs.df.test.dtm <- hs.df.clean %>%
+   dplyr::filter(type=="test") %>%
+   select(ID, word.ed) %>%
+   rbind(train.test.missing.df) %>%
+   count(ID, word.ed) %>%
+   cast_dtm(document=ID, term=word.ed, value=n)
> row.names(hs.df.test.dtm)
 [1] "Happy_81" "Happy_82" "Happy_83" "Happy_84" "Happy_85" "Happy_86" "Happy_87" "Happy_88"
 [9] "Happy_89" "Happy_90" "Sad_81"   "Sad_82"   "Sad_83"   "Sad_84"   "Sad_85"   "Sad_86"
[17] "Sad_87"   "Sad_88"   "Sad_89"   "Sad_90"   NA
> hs.df.test.dtmatrix <-
+   hs.df.test.dtm %>% as.matrix
> hs.df.test.dtmatrix <-
+   hs.df.test.dtmatrix[1:20,]
> row.names(hs.df.test.dtmatrix)
 [1] "Happy_81" "Happy_82" "Happy_83" "Happy_84" "Happy_85" "Happy_86" "Happy_87" "Happy_88"
 [9] "Happy_89" "Happy_90" "Sad_81"   "Sad_82"   "Sad_83"   "Sad_84"   "Sad_85"   "Sad_86"
[17] "Sad_87"   "Sad_88"   "Sad_89"   "Sad_90"
```

Test set should contain all words that are included in the train set
➔ Words that appear in the train set but not in the test set will get value of 0.

Meaningless one added after rbind

- Random Forest

```
> library(caret)
> set.seed(1009)
> dim(hs.df.train.dtmatrix)
[1] 160 481
> length(hs.df.train.label)
[1] 160
> dim(hs.df.test.dtmatrix)
[1]  20 517
> length(hs.df.test.label)
[1] 20
> rf.train <- train(x = hs.df.train.dtmatrix,
+                   y = hs.df.train.label,
+                   method = "ranger")
> rf.train
Random Forest

160 samples
481 predictors
  2 classes: 'Happy', 'Sad'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 160, 160, 160, 160, 160, 160, ...
Resampling results across tuning parameters:

  mtry  splitrule   Accuracy   Kappa
    2   gini        0.4703794  0.003731980
    2   extratrees  0.4690001  0.001476015
  241   gini        0.9748948  0.949440247
  241   extratrees  0.9728379  0.945273378
  481   gini        0.9740783  0.947757591
  481   extratrees  0.9741312  0.947858234

Tuning parameter 'min.node.size' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were mtry = 241, splitrule = gini and min.node.size = 1.
```

Check the dimension of data

Create a Random Forest model with train set

Step 4 - RF) Performance Check

- Check the performance
  - Accuracy
  - Kappa
  - Sensitivity
  - Specificity

```
> rf.train.pred <- predict(rf.train)
> table(rf.train.pred, hs.df.train.label)
             hs.df.train.label
rf.train.pred Happy Sad
        Happy    80   1
        Sad       0  79
```

Check the performance with the train set

```
> rf.test.pred <-
+   predict(rf.train,
+           newdata = hs.df.test.dtmatrix)
> table(rf.test.pred, hs.df.test.label)
            hs.df.test.label
rf.test.pred Happy Sad
       Happy    10   1
       Sad       0   9
```

Check the performance of model with the test set

```
> table(rf.test.pred, hs.df.test.label) %>%
+   confusionMatrix
Confusion Matrix and Statistics

            hs.df.test.label
rf.test.pred Happy Sad
       Happy    10   1
       Sad       0   9

               Accuracy : 0.95
                 95% CI : (0.7513, 0.9987)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 2.003e-05

                  Kappa : 0.9

 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9000
         Pos Pred Value : 0.9091
         Neg Pred Value : 1.0000
             Prevalence : 0.5000
         Detection Rate : 0.5000
   Detection Prevalence : 0.5500
      Balanced Accuracy : 0.9500

       'Positive' Class : Happy
```

Step 3 - NN) ML-based Sentiment Analysis

- Neural Network

```
> nn.train <- train(x = hs.df.train.dtmatrix,
+                   y = hs.df.train.label,
+                   method = "nnet")
# weights:  484
initial  value 110.230278
iter  10 value 0.005996
iter  20 value 0.000122
iter  20 value 0.000047
iter  20 value 0.000047
final  value 0.000047
converged
```

Create a neural network model

```
> nn.train
Neural Network

160 samples
481 predictors
  2 classes: 'Happy', 'Sad'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 160, 160, 160, 160, 160, 160, ...
Resampling results across tuning parameters:

  size  decay  Accuracy   Kappa
  1     0e+00  0.9656416  0.9305773
```

Step 4 - NN) Performance Check

- Check the performance
  - Accuracy
  - Kappa
  - Sensitivity
  - Specificity

```
> nn.train.pred <- predict(nn.train)
> table(nn.train.pred, hs.df.train.label)
             hs.df.train.label
nn.train.pred Happy Sad
        Happy    80   1
        Sad       0  79
```

Check the performance with the train set

```
> nn.test.pred <-
+   predict(nn.train,
+           newdata = hs.df.test.dtmatrix)
> table(nn.test.pred, hs.df.test.label)
            hs.df.test.label
nn.test.pred Happy Sad
       Happy    10   1
       Sad       0   9
```

Check the performance of model with the test set

```
> table(nn.test.pred, hs.df.test.label) %>%
+   confusionMatrix
Confusion Matrix and Statistics

            hs.df.test.label
nn.test.pred Happy Sad
       Happy    10   1
       Sad       0   9

               Accuracy : 0.95
                 95% CI : (0.7513, 0.9987)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 2.003e-05

                  Kappa : 0.9

 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9000
         Pos Pred Value : 0.9091
         Neg Pred Value : 1.0000
             Prevalence : 0.5000
         Detection Rate : 0.5000
   Detection Prevalence : 0.5500
      Balanced Accuracy : 0.9500

       'Positive' Class : Happy
```