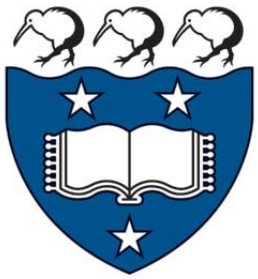
Text Analytics

Jason Peter Cairns

Supervised by Chris Wild



Bachelor of Science (Honours)
Department of Statistics
The University of Auckland
New Zealand

Todo list

should this be an abstract? 9

Acknowledgements

Contents

Listings **5**

Tables **6**

Figures **7**

1 Introduction **9**
1.1 Intention 9
1.2 Background: Text Analytics (incl. examples) 9
1.2.1 common functions: sentiment, summarisation, scoring 9
1.2.2 Existing Systems 10
1.2.3 current issues 11
1.3 Background: inZight 11
1.3.1 What iNZight is - capabilities, popularity, etc. 11
1.3.2 how our program fits in - shiny, inzight lite etc. 11
1.4 Literature Review (existing packages in R) 11
1.4.1 Copy over from notes, flesh out a bit 11
1.4.2 Praise tidytext book, complain about the package . . 11
1.5 Scope of work 11
1.5.1 types of text that we can work with: novels, free response data etc. 11
1.5.2 discuss limitations placed: not going into linguistic territory etc. 11

2 Text Analytics Prolusion **12**
2.1 overview 12
2.1.1 Explain broadness of term 12
2.1.2 compile glossary from terms here 12
2.1.3 Areas of text analytics in a data science framework . . 12
2.1.4 what we have done 12
2.1.5 what we haven't done 12
2.2 terms 12
2.2.1 terms and their centrality 14

2.2.2 generalisation: n-grams, sentences etc. 14
2.3 Historical Background 14
2.3.1 computer science vs statistics - reflection in data science 14
2.4 Processing 14
2.4.1 why process 14
2.4.2 stopwords, lemmatisation etc. 14
2.4.3 modelling vs db joins - more info in notes 14
2.5 scores & statistics 14
2.5.1 why compute scores & statistics 14
2.5.2 scoring - tf-idf, word count 14
2.5.3 Suggestions for further research - more on the statistics of words 14
2.5.4 recount the book of John text analysis 14
2.6 Sentiment 14
2.6.1 why sentiment 14
2.6.2 Process of sentiment 14
2.6.3 sentiment modelling vs db joins 14
2.6.4 our implementation and why 14
2.6.5 reviews 14
2.6.6 issues 14
2.7 Summarisation 14
2.7.1 why compute summarisations 14
2.7.2 lexrank, textrank - include notes on lexrank 14
2.7.3 other methods 14
2.7.4 reddit bot example 14
2.8 what we didn't do (yet) 14
2.8.1 topic modelling 14
2.8.2 Term correlation 14
2.8.3 modelling based on linguistic features 14
2.9 Visualisation 14
2.9.1 talk about score vs structure 14
2.9.2 complain about tag clouds 14
2.9.3 talk about ggpage 14
2.9.4 discuss our experimentations with some alternative visualisations 14

3 Program Structure & Development **15**
3.0.1 why R 16
3.0.2 Why Shiny 16
3.0.3 why tidyverse 16
3.0.4 Git 16
3.0.5 possible future: datatables, futures, etc. 16
3.0.6 why functional 16
3.0.7 Why lossless data 16

3.1 Program Architecture 16
3.1.1 Why structure it like it has been 16
3.1.2 make graph of architecture 16
3.1.3 Describe package and package creation 16
3.1.4 following three sections copy and paste from the notes - buffing up as necessary 16
3.1.5 include screenshots 16
3.2 Import 16
3.3 Insight 16
3.4 Visualisation 16
3.5 User Interface 16

4 Conclusion **17**
4.1 Summary 17
4.1.1 summarise successes 17
4.1.2 summarise failures 17
4.1.3 general thoughts on the topic 17
4.2 Recommendations 17
4.2.1 educational potential of text analytics 17
4.2.2 what else remains 17

5 Appendix **18**

Glossary **35**

Index **35**

Bibliography **36**

List of source codes

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Intention

Text Analytics serves to glean insight from a body of text. Within the broad category of text analytics, we seek to answer questions about what the text is communicating, what is felt about it, and how this information is structured. In this dissertation, we demonstrate the creation of a user-friendly program to perform text analytics functions using modern R with the Shiny web application framework. In a literate style, we illustrate top-down the structure of such a program, as well as the data structures and computational processes that have established their value for such a program.

should this be an abstract?

1.2 Background: Text Analytics (incl. examples)

1.2.1 common functions: sentiment, summarisation, scoring

Text Analytics is comprised of a variety of processes and techniques to extract information from text. The text almost always requires some initial processing. Some of the following functions have proven utility, and are expanded upon in chapter 2;

- Sentiment: In order to answer what emotions are conveyed in a text, sentiment analysis is commonly performed. The technique yields some measure of what is represented in an emotional sense by the text, with a range different methods and their associated outputs allowing for different forms of the analysis. Sentiment analysis won't pick up the subtle nuances that a human reader would, but generally gives reasonable output over the extent of a text.
- Associated Words: The meaning of a text is dependent on the structure between and within words. Looking at how words are associated,

through correlation, common sequences, visualisation of sections, etc., allow for a clear high-level assessment of the associations between words. The higher level not only saves individual efforts, but will demonstrate any emergent properties inherent to a text, in a way that a direct reading won't necessarily reveal.

- Summarisation: Automation of an executive summary, or a list of key words, typically falls under the purview of summarisation. The primary aim is to rank and select the most "representative" words or sentences from a text. A few major techniques dominate, being somewhat complex in nature. The results are generally surprisingly well representative of a text.
- Feature Counts: The simplest quantitative measure is very often the most informative; from simple word counts, to selective counts of sentences within groups, counting features can reveal how much written weighting is given to various elements, aiding insight into both structure and sentiment simultaneously.

1.2.2 Existing Systems

There are several existing systems in the field of Text Analytics. The field was initially nurtured as a sub-field of Computer Science, being computationally-dependent in nature. More recently, there has been increasing statistical interest. The existing systems reflect this; most older text analytics programs were Artificial Intelligence focussed, being experimental in nature, typically composed in lisp. More recently, major statistical programs hae been incorporating text analytic features, with a few smaller text analytics specific programs appearing. SAS, SPSS, and R are all examples of major statistical processing systems, with recent additions of text analytics capabilities. An overview of R packages aiding in text analytics will be given in section 1.4

1.2.3 current issues

1.3 Background: inZight

1.3.1 What iNZight is - capabilities, popularity, etc.

1.3.2 how our program fits in - shiny, inzight lite etc.

1.4 Literature Review (existing packages in R)

1.4.1 Copy over from notes, flesh out a bit

1.4.2 Praise tidytext book, complain about the package

1.5 Scope of work

1.5.1 types of text that we can work with: novels, free response data etc.

1.5.2 discuss limitations placed: not going into linguistic territory etc.

Chapter 2

Text Analytics Prolusion

2.1 overview

Most importantly, words must be extracted, serving as the basic unit of analysis, from which more complex items may be derived.

2.1.1 Explain broadness of term

2.1.2 compile glossary from terms here

2.1.3 Areas of text analytics in a data science framework

2.1.4 what we have done

2.1.5 what we haven't done

2.2 terms

term

2.2.1 terms and their centrality

2.2.2 generalisation: n-grams, sentences etc.

2.3 Historical Background

2.3.1 computer science vs statistics - reflection in data science

2.4 Processing

2.4.1 why process

2.4.2 stopwords, lemmatisation etc.

2.4.3 modelling vs db joins - more info in notes

2.5 scores & statistics

2.5.1 why compute scores & statistics

2.5.2 scoring - tf-idf, word count

2.5.3 Suggestions for further research - more on the statistics of words

2.5.4 recount the book of John text analysis

2.6 Sentiment

2.6.1 why sentiment

2.6.2 Process of sentiment

2.6.3 sentiment modelling vs db joins

2.6.4 our implementation and why

2.6.5 reviews

2.6.6 issues

2.7 Summarisation

2.7.1 why compute summarisations

2.7.2 lexrank, textrank - include notes on lexrank

2.7.3 other methods

2.7.4 reddit bot example

2.8 what we didn't do (yet)

2.8.1 topic modelling

2.8.2 Term correlation

2.8.3 modelling based on linguistic features

2.9 Visualisation

Chapter 3

Program Structure & Development

3.0.1 why R

3.0.2 Why Shiny

3.0.3 why tidyverse

3.0.4 Git

3.0.5 possible future: datatables, futures, etc.

3.0.6 why functional

3.0.7 Why lossless data

3.1 Program Architecture

3.1.1 Why structure it like it has been

3.1.2 make graph of architecture

3.1.3 Describe package and package creation

3.1.4 following three sections copy and paste from the notes - buffing up as necessary

3.1.5 include screenshots

3.2 Import

3.3 Insight

3.4 Visualisation

3.5 User Interface

Chapter 4

Conclusion

4.1 Summary

4.1.1 summarise successes

4.1.2 summarise failures

4.1.3 general thoughts on the topic

4.2 Recommendations

4.2.1 educational potential of text analytics

4.2.2 what else remains

Chapter 5

Appendix

The following pages are a copy of the documentation for the R package created as a part of this dissertation. They were automatically generated through the Roxygen2 system.

Package 'inzightta'

August 16, 2019

Title iNZight Text Analytics

Version 0.0.0.9000

Description Provides text analytics functions for the importation, analysis, and visualisation of text. This package is designed specifically for output in shiny, with the analytical functions all working well with dplyr tools.

License GPL-3

Encoding UTF-8

LazyData true

Imports readr, tibble, stringr, dplyr, readxl, purrr, tidytext, textstem, magrittr, stats, textrank, lexRankr

RoxygenNote 6.1.1

R topics documented:

aggregate_sentiment 2
determine_stopwords 3
format_data 3
get_bigram 4
get_chapters 4
get_filetype 5
get_parts 5
get_search 6

get_sections 6
get_sw 7
get_valid_input 7
ifexp 8
import_base_file 8
import_csv 9
import_excel 9
import_files 10
import_txt 10
index_bigram 11
keywords_tr 11
key_sentences 12
table_textcol 12
term_count 13
term_freq 13
text_prep 14
ungroup_by 14
word_sentiment 15

Index **16**

`aggregate_sentiment` *Get statistics for sentiment over some group, such as sentence.*

Description

Get statistics for sentiment over some group, such as sentence.

Usage

```
aggregate_sentiment(.data, aggregate_on, statistic)
```

Arguments

`.data`	character vector of words
`aggregate_on`	vector to aggregate .data over; ideally, sentence_id, but could be chapter, document, etc.
`statistic`	function that accepts na.rm argument; e.g. mean, median, sd.

determine_stopwords *determine stopword status*

Description

determine stopword status

Usage

```
determine_stopwords(.data, ...)
```

Arguments

.data	vector of words
...	arguments of get_sw

Value

a [tibble][tibble::tibble-package] equivalent to the input dataframe, with an additional stopword column

format_data *formats imported data into an analysis-ready format*

Description

formats imported data into an analysis-ready format

Usage

```
format_data(data)
```

Arguments

data	a tibble formatted with a text and (optional) group column

Value

a [tibble][tibble::tibble-package] formatted such that columns correspond to identifiers of group, line, sentence, word (groups ignored)

get_bigram *Determine bigrams*

Description

Determine bigrams

Usage

```
get_bigram(.data)
```

Arguments

.data	character vector of words

Value

character vector of bigrams

get_chapters *sections text based on chapters*

Description

sections text based on chapters

Usage

```
get_chapters(.data)
```

Arguments

.data	vector to section

Value

vector of same length as .data with chapter numbers

get_filetype *Get filetype*

Description

Get filetype

Usage

```
get_filetype(filepath)
```

Arguments

filepath	string filepath of document

Value

filetype (string) - NA if no extension

get_parts *sections text based on parts*

Description

sections text based on parts

Usage

```
get_parts(.data)
```

Arguments

.data	vector to section

Value

vector of same length as .data with part numbers

get_search *creates a search closure to section text*

Description

creates a search closure to section text

Usage

```
get_search(search)
```

Arguments

search	a string regexp for the term to seperate on, e.g. "Chapter"

Value

closure over search expression

get_sections *sections text based on sections*

Description

sections text based on sections

Usage

```
get_sections(.data)
```

Arguments

.data	vector to section

Value

vector of same length as .data with section numbers

get_sw *Gets stopwords from a default list and user-provided list*

Description

Gets stopwords from a default list and user-provided list

Usage

```
get_sw(lexicon = "snowball", addl = NA)
```

Arguments

lexicon	a string name of a stopword list, one of "smart", "snowball", or "onix"
addl	user defined character vector of additional stopwords, each element being a stopword

Value

a [tibble][tibble::tibble-package] with one column named "word"

get_valid_input *helper function to get valid input (recursively)*

Description

helper function to get valid input (recursively)

Usage

```
get_valid_input(options, init = TRUE)
```

Arguments

options	vector of options that valid input should be drawn from
init	whether this is the initial attempt, used only as recursive information

Value

readline output that exists in the vector of options

ifexp *scheme-like if expression, without restriction of returning same-size table of .test, as ifelse() does*

Description

scheme-like if expression, without restriction of returning same-size table of .test, as ifelse() does

Usage

```
ifexp(.test, true, false)
```

Arguments

.test	predicate to test
true	expression to return if .test evals to TRUE
false	expression to return if .test evals to TRUE

Value

either true or false

import_base_file *Base case for file import*

Description

Base case for file import

Usage

```
import_base_file(filepath)
```

Arguments

filepath	string filepath of file for import

Value

imported file with document id

import_csv *Import csv file*

Description

Import csv file

Usage

```
import_csv(filepath)
```

Arguments

filepath	a string indicating the relative or absolute filepath of the file to import

Value

a [tibble][tibble::tibble-package] of each row corrresponding to a line of the text file, with the column named "text"

import_excel *Import excel file*

Description

Import excel file

Usage

```
import_excel(filepath)
```

Arguments

filepath	a string indicating the relative or absolute filepath of the file to import

Value

a [tibble][tibble::tibble-package] of each row corrresponding to a line of the text file, with the column named "text"

import_files *Import any number of files*

Description

Import any number of files

Usage

```
import_files(filepaths)
```

Arguments

filepaths	char vector of filepaths

Value

a [tibble][tibble::tibble-package] imported files with document id

import_txt *Import text file*

Description

Import text file

Usage

```
import_txt(filepath)
```

Arguments

filepath	a string indicating the relative or absolute filepath of the file to import

Value

a [tibble][tibble::tibble-package] of each row corrresponding to a line of the text file, with the column named "text"

`index_bigram` *get bigram at index i of list1 & 2*

Description

get bigram at index i of list1 & 2

Usage

```
index_bigram(i, list1, list2)
```

Arguments

`i`	numeric index to attain bigram at
`list1`	list or vector for first bigram token
`list2`	list or vector for second bigram token

Value

bigram of list1 and list2 at index i, skipping NA's

`keywords_tr` *Determine textrank score for vector of words*

Description

Determine textrank score for vector of words

Usage

```
keywords_tr(.data)
```

Arguments

`.data`	character vector of words

Value

vector of scores for each word

key_sentences *get score for key sentences as per Lexrank*

Description

get score for key sentences as per Lexrank

Usage

```
key_sentences(.data, aggregate_on)
```

Arguments

.data	character vector of words
aggregate_on	vector to aggregate .data over; ideally, sentence_id

table_textcol *Interactively determine and automatically mark the text column of a table*

Description

Interactively determine and automatically mark the text column of a table

Usage

```
table_textcol(data)
```

Arguments

data	dataframe with column requiring marking

Value

same dataframe with text column renamed to "text"

term_count *Determine the number of terms at each aggregate level*

Description

Determine the number of terms at each aggregate level

Usage

```
term_count(.data, aggregate_on)
```

Arguments

.data	character vector of terms
aggregate_on	vector to split .data on for insight

Value

vector of number of terms for each aggregate level, same length as .data

term_freq *Determine term frequency*

Description

Determine term frequency

Usage

```
term_freq(.data)
```

Arguments

.data	character vector of terms

Value

numeric vector of term frequencies

`text_prep` *takes imported one-line-per-row data and prepares it for later analysis*

Description

takes imported one-line-per-row data and prepares it for later analysis

Usage

```
text_prep(.data, lemmatize = TRUE, stopwords = TRUE,
  sw_lexicon = "snowball", addl_stopwords = NA)
```

Arguments

`.data`	tibble with one line of text per row
`lemmatize`	boolean, whether to lemmatize or not
`stopwords`	boolean, whether to remove stopwords or not
`sw_lexicon`	string, lexicon with which to remove stopwords
`addl_stopwords`	char vector of user-supplied stopwords

Value

a [tibble][tibble::tibble-package] with one token per line, stopwords removed leaving NA values, column for analysis named "text"

`ungroup_by` *helper function to ungroup for dplyr. functions equivalently to group_by() but with standard (string) evaluation*

Description

helper function to ungroup for dplyr. functions equivalently to group_by() but with standard (string) evaluation

Usage

```
ungroup_by(x, ...)
```

Arguments

`x`	tibble to perform function on
`...`	string of groups to ungroup on

Value

x with ... no longer grouped upon

`word_sentiment` *Determine sentiment of words*

Description

Determine sentiment of words

Usage

```
word_sentiment(.data, lexicon = "afinn")
```

Arguments

`.data`	vector of words
`lexicon`	sentiment lexicon to use, based on the corpus provided by tidytext

Value

vector with sentiment score of each word in the vector

Index

aggregate_sentiment, 2

determine_stopwords, 3

format_data, 3

get_bigram, 4
get_chapters, 4
get_filetype, 5
get_parts, 5
get_search, 6
get_sections, 6
get_sw, 7
get_valid_input, 7

ifexp, 8
import_base_file, 8
import_csv, 9
import_excel, 9
import_files, 10
import_txt, 10
index_bigram, 11

key_sentences, 12
keywords_tr, 11

table_textcol, 12
term_count, 13
term_freq, 13
text_prep, 14

ungroup_by, 14

word_sentiment, 15

Glossary

term "a word or expression that has a precise meaning in some uses or is peculiar to a science, art, profession, or subject'[1] — here text analysts have capitalised on the generalisation of "term'to include subcomponents or aggregations of words. 9

Bibliography

[1] Merriam-Webster Dictionary, ed. *Term — Definition of Term*. 17th Aug. 2019. URL: https://www.merriam-webster.com/dictionary/term (cit. on p. 35).