classecol: vignette

Thomas Frederick Johnson Github: GitTFJ

January 2021

classecol

classecol is a package to perform nature-related text classifications of public opinion data (trained using twitter data). classecol provides a series of functions which can clean data, pull together outputs from multiple sentiment analysis approaches into one function, and has three models to classify text: bio - to assess who the user is e.g. a person, expert, nature organisation, or something else; nature - to assess if the tweet is relevant to nature (or the natural world), and if so, describe whether the tweet is showing concern, interest or fear; hunting - to assess if the tweet is relevant to hunting, and if so, describe whether the tweet is pro- or against-hunting.

Loading classecol

The most recent and throughly tested version of classecol is only available as a github repository so needs to be installed through github. Before installing classecol, we highly recommend installing the latest version of Java, which can be found at (https://www.java.com/en/download/)

```
library(devtools)
install_github("GitTFJ/classecol")
library(classecol)
library(reticulate)
```

classecol has three core themes: text cleaning and processing, sentiment analysis, and supervised text classification

Text cleaning and processing

clean()

Cleans social media text converting up to 1000 multi-word nature-related hashtags and over 1500 abbreviations into readable text. This function also converts nearly 150 emoticons and 175 slang-words into readable text using terms sourced from the lexicon R package (Rinker, 2019a). Can select a 'simple' or 'full' clean, where a 'simple' clean includes: hashtag conversion, removal of twitter specific syntax (e.g. RTs), removal of urls, conversion of emoticons, removal of special characters. 'full' includes all of the above as well as converting abbreviations, slang, grades and ratings. The cleaning process can also be completely customised, selecting or removing any paricular claning element. For example, one could do a 'full' clean, but specify that hashtags should not be converted (see below).

```
text = "BTW tomorrow will be the best #TrophyHunting :)"
text
```

```
## [1] "BTW tomorrow will be the best #TrophyHunting :)"
```

```
clean(text, level = "simple")
```

```
## [1] "BTW tomorrow will be the best trophy hunting smiley"
```

```
clean(text, level = "full")
```

```
## [1] "by the way tomorrow will be the best trophy hunting smiley"
```

```
clean(text, level = "full", hashtag = F)
```

```
## [1] "by the way tomorrow will be the best TrophyHunting smiley"
```

valence()

Checks for the presence of negator (flips the meaning of the text e.g. I am NOT sad), amplifier (adds intensity to the text e.g. I am VERY sad), de-amplifier (softens the text e.g. I am KIND OF sad), and adversative-conjunction terms (overrules the previous sentiment e.g. I am sad, BUT proud), which could alter the meaning of the text. Terms and examples sourced from the lexicon (Rinker, 2019a) and sentiment (Rinker, 2019b) R packages.

```
rbind(
  valence("I am not sad"),
  valence("I am very sad"),
  valence("I am kind of sad"),
  valence("I am sad, but proud"))
```

```
##   negator amplifier deamplifier ad_conjunction
## 1       1         0           0              0
## 2       0         1           0              0
## 3       0         0           1              0
## 4       0         0           0              1
```

contract()

Performs stemming (trims word to their simplest form e.g. cars becomes car) and lemmatisation (identifies the core theme of a term, bringing synonymous terms into one word e.g. automobile becomes car) within R. Function is a wrapper for textstem (Rinker, 2018), qdap (Rinker, 2020), and lexicon (Rinker, 2019a) R packages. This function is not necessary when running any of the hun_class, nat_class, and bio_class models, as the stemming and lemmatisation are conducted within the substantially faster Python program.

```
contract("consulting")
```

```
## [1] " consult "
```

lang_eng()

Checks if the language is English or not, the hun_class, nat_class, and bio_class models are designed for English text. Function is a wrapper to the cld2 (Ooms & Sites, 2018) R package.

```
lang_eng("hallo und willkommen bei classecol")#German
```

```
## [1] 0
```

```
lang_eng("hello and welcome to classecol")#English
```

```
## [1] 1
```

Sentiment analysis

senti_matrix()

Conducts sentiment analysis (assesses polarity of text) using 11 approaches and pulls all approaches into a matrix of sentiment. Approaches are drawn from the sentimentr (Rinker, 2019b), lexicon (Rinker, 2019a), syuzhet (Jockers, 2017), and meanr (Schmidt, 2019) R packages.

```
rbind(
  senti_matrix("I love wildlife so much, This is the best day ever"),
  senti_matrix("I hate wildlife so much, This is the worst day ever"))
```

```
##   jockers_rinker jockers huliu loughran_mcdonald senticnet sentiword
## 1           0.56    0.56  0.84              0.3      0.37      0.42
## 2          -0.56   -0.56 -0.84             -0.3     -0.53     -0.33
##   socal_google nrc afinn bing meanr
## 1         1.49   1     6    2     2
## 2         0.00  -1    -6   -2    -2
```

supervised text classification

The classecol package's supervised text classification is run through R but is reliant on a python back-end (which dramitcally improves the speed of the classification). So before running any code, you will need to complete the following the steps: 1) Download and install python - which is used to run the text classification models. We recommend downloading python 3.6.x which is what the package has been extensively tested on, which is available at https://www.python.org/downloads/release/python-369/ ; 2) Download classecol python models - classecol has an associated python repository that needs to be linked to the classecol package; 3) Download and install python packages - the text classification models rely on keras, tensorflow, pandas, nltk, bs4, sklearn, and scikit-learn (version 0.19.1). Packages can be installed manually following https://packaging.python.org/tutorials/installing-packages/. 4) Link python and R - to run pyhton through R, we need to tell R were the downloaded python program can be found.

supervised text classification setup for Windows operating systems

Step 1 of this process can be automated using the addeR py:download() function, which downloads the recommended version of python. This function will download python and prompt its install. In the first step

of the install process, if prompted, we highly recommend selecting 'Add python to the PATH'. Then select the basic install and once complete, close the install window. At this point, you will be prompted to quit R. For the python install to be recognised by R, its important to quit and re-open R. This can be done at any convenient point in time, but must completed before any more text clasification functions are run.

```
#If python is not downloaded, use the addeR package
#library(addeR)
#addeR::py_download() #Automatic python download
```

load_classecol()

For steps 2 to 4, we provide a set of parameters to simplify the process through the load_classecol() function. This function prepares the python aspects of classecol automatically. The first time classecol is run, its important to set the parameters so download_models = T and download_modules = T. Once these steps have succesfully downloaded, their data will be permanently stored and they can be set as FALSE in future cases. The link_py parameter must be set as TRUE run every time a new R environment is opened.

```
#On the first run, set download_models = T and download_modules = T
load_classecol(download_models = F, download_modules = F, link_py = T)
```

At this point we are ready to prepare and classify the data.

Hunting classifier

The hunting classifier 'hun_class()' works best with twitter data after a simple clean. The 'type' parameter within the 'hun_class()' function specifies the type of text classification model, options include: relevance - use this model when the objective is to identify if text is relevant (or not) to hunting; stance - use this model on text relevant to hunting, where the objective is to identify the stance of the text (pro- or against-hunting); full - use this model when the objective is to identify text that is relevant to hunting, and when relevant, describe the stance; combining the relevance and stance steps into one model.

```
df = data.frame(
  text = c(
    "I hate hunting. Ban it now!",
    "Cant wait to go camping this weekend #hunting #fishing",
    "Hunting for my car keys"),
  stringsAsFactors = F)
#sys <- import("sys", convert = TRUE)
#sys$path
#hun_class(
#  text_vector = classecol::clean(df$text, level = "simple"),
#  type = "full")
```