classecol: vignette

Thomas Frederick Johnson Github: GitTFJ

6 July 2020

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.

Instructions to prepare package

This package is runs 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 install a version of python - we recommend Python 3.6.9 which is what the package has been tested on; available here. You will also need to install a selection of packages on python: numpy, os, pandas, re, nltk, bs4, string, joblib, pickle, sklearn, keras, tensorflow, and time. Packages can be installed following these instructions. If at anypoint the error 'Module not found' appears, install the listed module/package following the above instructions.

The most recent and throughly tested version of classecol is only available as a github repository so needs to be installed through github

```r
library(devtools)
install_github("GitTFJ/classecol", upgrade = "never")
library(classecol)
library(reticulate)
```

The python text classification models which classecol rely's on are not autoatically downloaded when the package is installed. Instead, its necassary to download and save an additional github repository, which is automated through the 'download_models' function. To encuorage a seamless use of classecol, we recommend storing these models alongside the classecol package, which can found locally using the 'find.package' function.

```r
direc = paste(find.package("classecol"),"/models", sep = "")
download_models(direc)
```

'reticulate' offers a function to link the R classecol package to the python backend which contains the classification models. However, this automated function can perform inconsistantly, so recommend manuallly specifying pythons absolute filepath location. The file to search for is 'python.exe'.

You will also need to specify the location you have downloaded the models to and then send this location to python with the function 'r_to_py'

```r
#Specify your own path!!!
reticulate::use_python("C:User/Python/python.exe")
direc = paste(direc, "/classecol-models-master/", sep = "")
model_directory = reticulate::r_to_py(direc)
```

At this point we are ready to prepare and classify the data. It is very important that all naming conventions are matched, otherwise the classification will fail e.g. the text needs to be in a column called 'text', in a dataframe called 'df', and assigned as 'data' using the 'reticulate::r_to_py' function. See below.

Hunting classifier

The hunting classifier 'hun_class()' works best with twitter data after a simple clean.

```
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"))
df$text = classecol::clean(df$text, level = "simple")
data = reticulate::r_to_py(df)
hun_class(
  type = "Full",
  directory = direc)
```

```
## [1] "Relevant (against-hunting)" "Relevant (pro-hunting)"
## [3] "Irrelevant"
```

Nature classifier

The nature classifier 'nat_class()' works best with twitter data after a full clean and also requires sentiment analysis on the text.

```
df = data.frame(
  text = c(
    "I love walking in nature - so serene",
    "Why are the government not stopping the destruction of the rainforest?!",
    "Tiger wins the PGA tour again!"))
df$text = classecol::clean(df$text, level = "full")
sm = as.matrix(cbind(
  valence(df$text),
  lang_eng(as.character(df$text)),
  senti_matrix(as.character(contract(df$text)))))
data = reticulate::r_to_py(df)
sent_mat = reticulate::r_to_py(sm)
nat_class(
  type = "Trimmed",
  directory = direc)
```

```
## [1] "Pro-nature (positive phrasing)" "Pro-nature (negative phrasing)"
## [3] "Irrelevant"
```

Bio classifier

The biographical classifier 'bio_class()' works best with twitter data in its raw form, so none of the text should be cleaned. However, it is neccasary to join the twitter name and description into one column named 'text' split with a space.

```r
df = data.frame(
  name = c(
    "Jane Doe ",
    "Thomas Frederick Johnson",
    "Fictional University"),
  description = c(
  "Business leader, banker, parent, and cyclist",
  "Ecology and conservation researcher",
  "Campus life and study at the Fictional University. Follow for news and updates"))
df$text = paste(df$name, df$description)
data = reticulate::r_to_py(df)
bio_class(
  type = "Full",
  directory = direc)
```

```
## [1] "Person" "Expert" "Other"
```