Package: recogito (via r-universe)

August 23, 2024

Title Interactive Annotation of Text and Images

Version 0.2.1

Description Annotate text with entities and the relations between them. Annotate areas of interest in images with your labels. Providing 'htmlwidgets' bindings to the 'recogito' <https://github.com/recogito/recogito-js> and 'annotorious' <https://github.com/recogito/annotorious> libraries.

License BSD_3_clause + file LICENSE

Encoding UTF-8

URL https://github.com/DIGI-VUB/recogito

RoxygenNote 7.1.2

Depends R (>= 2.10)

Suggests shiny, magick, opencv, sf

Imports utils, htmlwidgets, htmltools, jsonlite

Repository https://digi-vub.r-universe.dev

RemoteUrl https://github.com/digi-vub/recogito

RemoteRef HEAD

RemoteSha 8ba39d54306fa3ab358ee4238b5751bf37e8a0b5

Contents

annotorious *Annotate images with areas of interest*

Description

Functionality to label areas in images

Usage

```
annotorious(
  inputId = "annotations",
  src,
  tags = c(),
  type = c("annotorious", "openseadragon", "openseadragon-notoolbar"),
  quickselector = TRUE,
  width = NULL,
  height = NULL,
  elementId = NULL,
  dependencies = NULL
)
```

Arguments

inputId	character string with the name to use where annotations will be put into
src	character string with the image/url to annotate
tags	character vector of possible labels you want to use
type	either 'annotorious', 'openseadragon', 'openseadragon-notoolbar' in order to allow zooming with openseadragon or not, with or without a toolbar
quickselector	logical indicating if for type 'openseadragon' the possible tags should be shows as quick buttons to click. Defaults to TRUE.
width	passed on to createWidget
height	passed on to createWidget
elementId	passed on to createWidget
dependencies	passed on to createWidget

Value

An object of class htmlwidget as returned by createWidget

See Also

annotorious-shiny

annotorious-shiny *Shiny bindings for annotorious*

Description

Output and render functions for using annotorious within Shiny applications.

Usage

```
annotoriousOutput(outputId, width = "100%", height = "400px")

renderAnnotorious(expr, env = parent.frame(), quoted = FALSE)

openseadragonOutput(outputId, width = "100%", height = "400px")

renderOpenSeaDragon(expr, env = parent.frame(), quoted = FALSE)

openseadragonOutputNoToolbar(outputId, width = "100%", height = "400px")

renderOpenSeaDragonNoToolbar(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a annotorious
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

Value

An output element for use in a Shiny user interface.
Consisting of a toggle button to switch between rectangle / polygon mode and the html-widget (id: outputId) which contains an image (id: outputId-img)

Examples

```
if(interactive() && require(shiny)){
##
## Annotorious using OpenSeaDragon, allowing to zoom in,
## to select an area, press shift and next select the area
##
library(shiny)
library(recogito)
urls <- paste("https://upload.wikimedia.org/",
```

```
              c("wikipedia/commons/a/a0/Pamphlet_dutch_tulipomania_1637.jpg",
                "wikipedia/commons/6/64/Cat_and_dog_standoff_%283926784260%29.jpg"),
              sep = "")
ui <- fluidPage(actionButton(inputId = "ui_switch", label = "Sample image"),
                openseadragonOutput(outputId = "anno"),
                tags$h3("Results"),
                verbatimTextOutput(outputId = "annotation_result"))
server <- function(input, output) {
  current_image <- reactive({
    input$ui_switch
    list(url = sample(urls, size = 1))
  })
  output$anno <- renderOpenSeaDragon({
    info <- current_image()
   annotorious("annotations", tags = c("IMAGE", "TEXT"), src = info$url, type = "openseadragon")
  })
  output$annotation_result <- renderPrint({
    read_annotorious(input$annotations)
  })
}
shinyApp(ui, server)

##
## Annotorious using OpenSeaDragon, allowing to zoom in, no selection possibilities
## showing how to load a local image
##
library(shiny)
library(recogito)
url <- system.file(package = "recogito", "examples", "Pamphlet_dutch_tulipomania_1637.jpg")
addResourcePath(prefix = "img", directoryPath = dirname(url))
ui <- fluidPage(openseadragonOutputNoToolbar(outputId = "anno", width = "100%", height = "250px"))
server <- function(input, output) {
  output$anno <- renderOpenSeaDragonNoToolbar({
    annotorious("annotations", src = sprintf("img/%s", basename(url)),
                type = "openseadragon-notoolbar")
  })
}
shinyApp(ui, server)

##
## Annotorious without openseadragon
##
library(shiny)
library(recogito)
url <- paste("https://upload.wikimedia.org/",
             "wikipedia/commons/a/a0/Pamphlet_dutch_tulipomania_1637.jpg",
             sep = "")

ui <- fluidPage(annotoriousOutput(outputId = "anno", height = "600px"),
                tags$h3("Results"),
                verbatimTextOutput(outputId = "annotation_result"))
server <- function(input, output) {
  output$anno <- renderAnnotorious({
```

```
   annotorious("annotations", tags = c("IMAGE", "TEXT"), src = url, type = "annotorious")
  })
  output$annotation_result <- renderPrint({
    read_annotorious(input$annotations)
  })
}
shinyApp(ui, server)

##
## Annotorious, without openseadragon changing the url
##
library(shiny)
library(recogito)
urls <- paste("https://upload.wikimedia.org/",
              c("wikipedia/commons/a/a0/Pamphlet_dutch_tulipomania_1637.jpg",
                "wikipedia/commons/6/64/Cat_and_dog_standoff_%283926784260%29.jpg"),
              sep = "")

ui <- fluidPage(actionButton(inputId = "ui_switch", label = "Sample image"),
                annotoriousOutput(outputId = "anno", height = "600px"),
                tags$h3("Results"),
                verbatimTextOutput(outputId = "annotation_result"))
server <- function(input, output) {
  current_image <- reactive({
    input$ui_switch
    list(url = sample(urls, size = 1))
  })
  output$anno <- renderAnnotorious({
    info <- current_image()
   annotorious("annotations", tags = c("IMAGE", "TEXT"), src = info$url, type = "annotorious")
  })
  output$annotation_result <- renderPrint({
    read_annotorious(input$annotations)
  })
}
shinyApp(ui, server)

}
```

`ocv_crop_annotorious` *Crop annotations to a bounding box*

Description

Crop annotations to a bounding box

Usage

```
ocv_crop_annotorious(data, bbox)
```

Arguments

data	an object as returned by read_annotorious
bbox	a vector with elements x, y, xmax, ymax

Value

data where column polygon and the rectangle information in x, y, width, height is limited to the provided bounding box

Examples

```
library(opencv)
data(openseadragon_areas)

url  <- attr(openseadragon_areas, "src")
img  <- ocv_read(url)
bbox <- ocv_info(img)
bbox <- c(xmin = 0, ymin = 0, xmax = bbox$width - 1, ymax = bbox$height - 1)
x    <- ocv_crop_annotorious(data = openseadragon_areas)
x    <- ocv_crop_annotorious(data = openseadragon_areas, bbox = bbox)

img
area <- x[2, ]
ocv_polygon(img, pts = area$polygon[[1]], crop = TRUE)
area <- x[1, ]
ocv_rectangle(img, x = area$x, y = area$y, width = area$width, height = area$height)
area <- x[3, ]
ocv_rectangle(img, x = area$x, y = area$y, width = area$width, height = area$height)
```

---

ocv_read_annotorious *Extract the areas of interests of an image*

Description

Extract the areas of interests of an image

Usage

```
ocv_read_annotorious(data, image)
```

Arguments

data	an object as returned by read_annotorious
image	an ocv image object

Value

a list of ocv images with the extracted areas of interest

Examples

```
library(opencv)
library(magick)
data(openseadragon_areas)

url  <- attr(openseadragon_areas, "src")
img  <- ocv_read(url)

areas <- ocv_read_annotorious(data = openseadragon_areas, image = img)
areas[[1]]
areas[[2]]
img <- lapply(areas, FUN = function(x) image_read(ocv_bitmap(x)))
img <- do.call(c, img)
img <- image_append(img, stack = FALSE)
image_resize(img, "x200")
```

---

openseadragon_areas *A dataset of annotations using openseadragon*

Description

A dataset of annotations using openseadragon

Examples

```
data(openseadragon_areas)
openseadragon_areas
attr(openseadragon_areas, "src")
```

---

read_annotorious *Parse annotorious annotations*

Description

Parse annotorious annotations

Usage

```
read_annotorious(x, src = character())
```

Arguments

x	a character string with json as returned by the htmlwidget
src	a character string with the image src which was used in x

Value

a data.frame with annotations with columns: id, type, label, comment, x, y, width, height, polygon and an attribute `src` with the provided `src`

Examples

```
url <- paste("https://upload.wikimedia.org/",
             "wikipedia/commons/a/a0/Pamphlet_dutch_tulipomania_1637.jpg",
             sep = "")
url <- system.file(package = "recogito", "examples", "Pamphlet_dutch_tulipomania_1637.jpg")
x <- '[
{
"type":"Annotation",
"body":[{"type":"TextualBody","value":"IMAGE","purpose":"tagging"}],
"target":{"selector":{
  "type":"FragmentSelector",
  "conformsTo":"http://www.w3.org/TR/media-frags/",
  "value":"xywh=pixel:41,249.5234375,371,245"}},
"@context":"http://www.w3.org/ns/anno.jsonld",
"id":"#58f0096c-4675-4ea8-9f38-bffce0887ab8"
},
{
"type":"Annotation",
"body":[{"type":"TextualBody","value":"TEXT","purpose":"tagging"}],
"target":{"selector":{
  "type":"FragmentSelector",
  "conformsTo":"http://www.w3.org/TR/media-frags/",
  "value":"xywh=pixel:46,5.523437976837158,371,239.99999952316284"}},
"@context":"http://www.w3.org/ns/anno.jsonld",
"id":"#50035dda-c62b-4f30-bf95-1879d60288a5"}]'
anno <- read_annotorious(x, src = url)
anno


library(magick)
img  <- image_read(url)
area <- head(anno, n = 1)
image_crop(img, geometry_area(x = area$x, y = area$y,
                              width = area$width, height = area$height))
area <- subset(anno, type == "RECTANGLE")
allrectangles <- Map(
  x      = area$x,
  y      = area$y,
  width  = area$width,
  height = area$height,
  f = function(x, y, width, height){
    image_crop(img, geometry_area(x = x, y = y, width = width, height = height))
})
allrectangles <- do.call(c, allrectangles)
allrectangles
```

```
x <- '[
{
  "type":"Annotation",
  "body":[{"type":"TextualBody","value":"IMAGE","purpose":"tagging"}],
  "target":{"selector":{
    "type":"FragmentSelector",
    "conformsTo":"http://www.w3.org/TR/media-frags/",
    "value":"xywh=pixel:43,244.5234375,362,252"
    }},
 "@context":"http://www.w3.org/ns/anno.jsonld",
 "id":"#4eaa8788-0c7e-42d2-b004-4d66b57018a1"},
{
  "type":"Annotation",
  "body":[{"type":"TextualBody","value":"TEXT","purpose":"tagging"}],
  "target":{"selector":{
    "type":"SvgSelector",
    "value":"<svg>
    <polygon points=\\\"75,4 75,58 32,95 32,194 410,195 391,70 373,63 368,3 222,1.5\\\">
    </polygon></svg>"}},
  "@context":"http://www.w3.org/ns/anno.jsonld",
  "id":"#8bf0a557-c847-4a07-91bc-68a98c499615"}]'
x    <- gsub(x, pattern = "\n", replacement = "")
anno <- read_annotorious(x, src = url)
anno
anno$polygon


library(opencv)
img  <- ocv_read(url)
area <- subset(anno, type == "POLYGON")
ocv_polygon(img, pts = area$polygon[[1]])
```

`read_recogito` *Parse recogito annotations*

Description

Parse recogito annotations

Usage

```
read_recogito(x, text = character())
```

Arguments

`x`	a character string with json as returned by the htmlwidget
`text`	a character string with the text which was used in `x`

Value

a data.frame with annotations with columns: id, type, label, chunk_text, chunk_start, chunk_end, relation_from, relation_to, chunk_comment and an attribute `text` with the provided `text`

Examples

```
x <- '[
{
"type":"Annotation",
"body":[
   {"type":"TextualBody","value":"sdfsd","purpose":"commenting"},
   {"type":"TextualBody","value":"Person","purpose":"tagging"}],
"target":{"selector":[
  {"type":"TextQuoteSelector","exact":"ngenious hero"},
  {"type":"TextPositionSelector","start":42,"end":55}]},
"@context":"http://www.w3.org/ns/anno.jsonld",
"id":"#a4ea53d4-69f3-4392-a3dd-cbb7e9ad50cb"
},
{
"type":"Annotation",
"body":[{"type":"TextualBody","value":"Person","purpose":"tagging"},
  {"type":"TextualBody","value":"Location","purpose":"tagging"}],
"target":{"selector":[{"type":"TextQuoteSelector","exact":"far and"},
  {"type":"TextPositionSelector","start":70,"end":77}]},
"@context":"http://www.w3.org/ns/anno.jsonld",
"id":"#d7050196-2537-42bf-9d1b-a3f9e4c9fbc6"
}
]'
read_recogito(x)
```

`recogito` *Annotate text with tags and relations*

Description

Functionality to tag text with entities and relations between these

Usage

```
recogito(
  inputId = "annotations",
  text,
  type = c("relations", "tags"),
  tags = c("Location", "Person", "Place", "Other"),
  mode = c("html", "pre"),
  annotations = "{}",
  width = NULL,
  height = NULL,
  elementId = NULL,
```

```
  dependencies = NULL
)
```

Arguments

inputId	character string with the name to use where annotations will be put into
text	character string with the text to annotate
type	either 'relations' or 'tags' in order to label relations between tags or only plain tags
tags	character vector of possible tags
mode	either 'html' or 'pre'
annotations	character string with a predefined set of annotations
width	passed on to createWidget
height	passed on to createWidget
elementId	passed on to createWidget
dependencies	passed on to createWidget

Value

An object of class htmlwidget as returned by createWidget

See Also

recogito-shiny

Examples

```
txt <- "Josh went to the bakery in Brussels.\nWhat an adventure!"
x   <- recogito(inputId = "annotations", txt)
x
x   <- recogito(inputId = "annotations", txt, type = "tags",
                tags = c("LOCATION", "TIME", "PERSON"))
x
txt <- "Lorem ipsum dolor sit amet consectetur adipiscing elit Quisque tellus urna
  placerat in tortor ac imperdiet sollicitudin mi Integer vel dolor mollis feugiat
  sem eu porttitor elit Sed aliquam urna sed placerat euismod In risus sem ornare
  nec malesuada eu ornare quis dui Nunc finibus fermentum sollicitudin Fusce vel
  imperdiet mi ac faucibus leo Cras massa massa ultricies et justo vitae molestie
  auctor turpis Vestibulum euismod porta risus euismod dapibus Nullam facilisis
  ipsum sed est tempor et aliquam sapien auctor Aliquam velit ligula convallis a
  dui id varius bibendum quam Cras malesuada nec justo sed
  aliquet Fusce urna magna malesuada"
x   <- recogito(inputId = "annotations", txt)
x
x   <- recogito(inputId = "annotations", txt, type = "tags",
                tags = c("LOCATION", "TIME", "PERSON", "OTHER"))
x
```

```
##
## Color the tags by specifying CSS - they should have .tag-{TAGLABEL}
##
library(htmltools)
cat(readLines(system.file(package = "recogito", "examples", "example.css")), sep = "\n")
tagsetcss <- htmlDependency(name = "mytagset", version = "1.0",
                            src = system.file(package = "recogito", "examples"),
                            stylesheet = "example.css")
x   <- recogito(inputId = "annotations", txt,
                tags = c("LOCATION", "TIME", "PERSON", "OTHER"),
                dependencies = tagsetcss)
x
```

`recogito-shiny` *Shiny bindings for recogito*

Description

Output and render functions for using recogito within Shiny applications and interactive Rmd documents.

Usage

```
recogitoOutput(outputId, width = "100%", height = "400px")

renderRecogito(expr, env = parent.frame(), quoted = FALSE)

recogitotagsonlyOutput(outputId, width = "100%", height = "400px")

renderRecogitotagsonly(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`outputId`	output variable to read from
`width, height`	Must be a valid CSS unit (like `'100%'`, `'400px'`, `'auto'`) or a number, which will be coerced to a string and have `'px'` appended.
`expr`	An expression that generates a recogito
`env`	The environment in which to evaluate `expr`.
`quoted`	Is `expr` a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable.

Value

An output element for use in a Shiny user interface.
Consisting of a div of class plaintext which contains an optional toggle button to switch between annotation / relation mode (id: `outputId-toggle`) and the html-widget (id: `outputId`)

Examples

```
if(interactive() && require(shiny)){

##
## Tagging only, no relations
##
library(shiny)
library(recogito)
txt <- "Josh went to the bakery in Brussels.\nWhat an adventure!"
ui <- fluidPage(tags$h3("Provide some text to annotate"),
             textAreaInput(inputId = "ui_text", label = "Provide some text", value = txt),
                tags$h3("Annotation area"),
                recogitotagsonlyOutput(outputId = "annotation_text"),
                tags$hr(),
                tags$h3("Results"),
                verbatimTextOutput(outputId = "annotation_result"))
server <- function(input, output) {
  output$annotation_text <- renderRecogitotagsonly({
    recogito("annotations", text = input$ui_text, tags = c("LOCATION", "TIME", "PERSON"))
  })
  output$annotation_result <- renderPrint({
    read_recogito(input$annotations)
  })
}
shinyApp(ui, server)

##
## Tagging and relations
##
library(shiny)
library(recogito)
txt <- "Josh went to the bakery in Brussels.\nWhat an adventure!"
ui <- fluidPage(tags$h3("Provide some text to annotate"),
             textAreaInput(inputId = "ui_text", label = "Provide some text", value = txt),
                tags$h3("Annotation area"),
                recogitoOutput(outputId = "annotation_text"),
                tags$hr(),
                tags$h3("Results"),
                verbatimTextOutput(outputId = "annotation_result"))
server <- function(input, output) {
  output$annotation_text <- renderRecogito({
    recogito("annotations", text = input$ui_text, tags = c("LOCATION", "TIME", "PERSON"))
  })
  output$annotation_result <- renderPrint({
    read_recogito(input$annotations)
  })
}
shinyApp(ui, server)

}

recogitoOutput(outputId = "annotation_text")
```

```
recogitotagsonlyOutput(outputId = "annotation_text")
```

Index