

# Package ‘googleCloudVisionR’

February 7, 2020

**Title** Access to the 'Google Cloud Vision' API for Image Recognition,  
OCR and Labeling

**Description** Interact with the 'Google Cloud Vision' <<https://cloud.google.com/vision/>>  
API in R. Part of the 'cloudyr' <<https://cloudyr.github.io/>> project.

**Version** 0.2.0

**BugReports** <https://github.com/cloudyr/googleCloudVisionR/issues>

**Imports** googleAuthR, jsonlite, purrr, data.table, glue

**License** MIT + file LICENSE

**LazyData** true

**RoxygenNote** 6.1.1

**Encoding** UTF-8

**Suggests** knitr, rmarkdown, testthat, mockery, covr

**NeedsCompilation** no

**Author** Jenő Pal [cre],  
Tamas Koncz [aut],  
Balazs Varkoly [aut],  
Peter Lukacs [aut],  
Eszter Kocsis [aut],  
Florian Teschner [ctb]

**Maintainer** Jenő Pal <[paljency@gmail.com](mailto:paljency@gmail.com)>

**Repository** CRAN

**Date/Publication** 2020-02-07 14:00:02 UTC

## R topics documented:

call_vision_api . . . . .	2
create_request_body . . . . .	3
create_single_image_request . . . . .	3
encode_image . . . . .	4
extractor . . . . .	4
extract_annotations . . . . .	5

extract_error . . . . .	5
extract_response . . . . .	6
face_detection_extractor . . . . .	6
gcv_get_available_feature_types . . . . .	7
gcv_get_image_annotations . . . . .	7
gcv_get_raw_response . . . . .	8
gcv_get_response . . . . .	9
get_bounding_boxes . . . . .	10
get_invalid_image_paths . . . . .	10
label_detection_extractor . . . . .	11
landmark_detection_extractor . . . . .	11
logo_detection_extractor . . . . .	12
ocr_extractor . . . . .	12
split_to_chunks . . . . .	13

<b>Index</b>	<b>14</b>
--------------	-----------

---

call_vision_api	<i>helper function to send POST request to the Google Vision API</i>
-----------------	--

---

## Description

sends the request defined in ‘body‘ to the API

## Usage

```
call_vision_api(body, apiEndpoint = "images:annotate",
  httpRequestType = "POST")
```

## Arguments

body,	output of create_request_body()
apiEndpoint	character, api endpoint
httpRequestType	character, type of the http request

## Value

API response in raw format

---

create\_request\_body    *helper function to create json for response request*

---

**Description**

creates a json output from the inputs

**Usage**

```
create_request_body(imagePaths, feature, maxNumResults)
```

**Arguments**

imagePaths	character, file paths, URLs or Cloud Storage URIs of the images, can be a combination of all three
feature	character, one out of: "LABEL_DETECTION", "FACE_DETECTION", "TEXT_DETECTION", "DOCUMENT_TEXT_DETECTION", "LOGO_DETECTION", "LANDMARK_DETECTION"
maxNumResults	integer, the maximum number of results (per image) to be returned.

**Value**

request body (payload), encoded as json

---

create\_single\_image\_request  
*helper function to create a list of details of one image annotation request*

---

**Description**

creates a list output from the inputs

**Usage**

```
create_single_image_request(imagePath, feature, maxNumResults)
```

**Arguments**

imagePath	character, file path, URL or Cloud Storage URI of the image
feature	character, one out of: "LABEL_DETECTION", "FACE_DETECTION", "TEXT_DETECTION", "DOCUMENT_TEXT_DETECTION", "LOGO_DETECTION", "LANDMARK_DETECTION"
maxNumResults	integer, the maximum number of results (per image) to be returned.

**Value**

list of request details for one image

encode\_image      *helper function to base64 encode the image file*

---

**Description**

base64 encodes an image file

**Usage**

```
encode_image(imagePath)
```

**Arguments**

imagePath      character, path to the image

**Value**

get the image back as encoded file

---

extractor      *helper function code to provide an extractor function for different feature types*

---

**Description**

a utility to provide functions to extract features from the API response

**Usage**

```
extractor(featureType)
```

**Arguments**

featureType      the type of annotation as called in the response object

**Value**

a function

---

extract\_annotations     *helper function code to extract the annotations*

---

**Description**

a utility to extract features from the API response

**Usage**

```
extract_annotations(responses, imagePath, featureType)
```

**Arguments**

responses	an API response object
imagePaths	character, file paths, URLs or Cloud Storage URIs of the images, can be a combination of all three
featureType	the type of annotation as called in the response object

**Value**

a data.table

---

extract\_error     *helper function code to extract error from API response into a data.table*

---

**Description**

helper function code to extract error from API response into a data.table

**Usage**

```
extract_error(responses, imagePath)
```

**Arguments**

responses	an API response object
imagePaths	character, file paths, URLs or Cloud Storage URIs of the images, can be a combination of all three

**Value**

a data.table

---

extract_response	<i>helper function code to extract the response data.frame</i>
------------------	--

---

**Description**

a utility to extract features from the API response

**Usage**

```
extract_response(responses, imagePaths, feature)
```

**Arguments**

responses	an API response object
imagePaths	character, file paths, URLs or Cloud Storage URIs of the images, can be a combination of all three
feature	character, one out of: "LABEL_DETECTION", "FACE_DETECTION", "TEXT_DETECTION", "DOCUMENT_TEXT_DETECTION", "LOGO_DETECTION", "LANDMARK_DETECTION"

**Value**

a data.table

---

face_detection_extractor	<i>helper function code to extract API response into a data.table for given feature type</i>
--------------------------	--

---

**Description**

helper function code to extract API response into a data.table for given feature type

**Usage**

```
face_detection_extractor(response)
```

**Arguments**

response	an element of the API response object
----------	---------------------------------------

**Value**

a data.table

---

`gcv_get_available_feature_types`  
*helper function code to record available feature types*

---

**Description**

helper function code to record available feature types

**Usage**

```
gcv_get_available_feature_types()
```

**Value**

a list of available features names and their types (as returned by the API)

**Examples**

```
gcv_get_available_feature_types()
```

---

`gcv_get_image_annotatons`  
*Get parsed image annotations from the Google Cloud Vision API*

---

**Description**

Given a list of images, a feature type and the maximum number of responses, this functions calls the Google Cloud Vision API, and returns the image annotations in a data.table format.

**Usage**

```
gcv_get_image_annotatons(imagePaths, feature = "LABEL_DETECTION",
  numResults = NULL, batchSize = 64L, savePath = NULL)
```

**Arguments**

<code>imagePaths</code>	character, file paths, URLs or Cloud Storage URIs of the images, can be a combination of all three
<code>feature</code>	character, one out of: "LABEL_DETECTION", "FACE_DETECTION", "TEXT_DETECTION", "DOCUMENT_TEXT_DETECTION", "LOGO_DETECTION", "LANDMARK_DETECTION"
<code>numResults</code>	integer, the maximum number of results (per image) to be returned.
<code>batchSize</code>	integer, the chunk size for batch processing
<code>savePath</code>	character, if specified, results will be saved to this path (as .csv)

**Value**

a data frame with image annotation results

**Examples**

```
## Not run:
# Label Detection (default), with maximum 7 results returned per image
imagePath <- system.file(
  "extdata", "golden_retriever_puppies.jpg", package = "googleCloudVisionR"
)
gcv_get_image_annotiations(imagePaths = imagePath, maxNumResults = 7)

# Face detection
imagePath <- system.file(
  "extdata", "arnold_wife.jpg", package = "googleCloudVisionR"
)
gcv_get_image_annotiations(imagePaths = imagePath, feature = "FACE_DETECTION")

# Google Cloud Storage URI as input
gcv_get_image_annotiations("gs://vision-api-handwriting-ocr-bucket/handwriting_image.png")

## End(Not run)
```

---

`gcv_get_raw_response` *Get raw API response from the Google Cloud Vision API*

---

**Description**

Given a list of images, a feature type and the maximum number of responses, this functions calls the Google Cloud Vision API, and returns the raw response from the API. For a friendlier response, refer to the ‘`gcv_get_image_annotiations`’ function, which returns results in a `data.table` format (however, the information returned is limited compared to the raw response).

**Usage**

```
gcv_get_raw_response(imagePaths, feature = "LABEL_DETECTION",
  maxNumResults = NULL)
```

**Arguments**

<code>imagePaths</code>	character, file paths, URLs or Cloud Storage URIs of the images, can be a combination of all three
<code>feature</code>	character, one out of: "LABEL_DETECTION", "FACE_DETECTION", "TEXT_DETECTION", "DOCUMENT_TEXT_DETECTION", "LOGO_DETECTION", "LANDMARK_DETECTION"
<code>maxNumResults</code>	integer, the maximum number of results (per image) to be returned.



**Value**

a response object returned by the API. To get the image annotations, take the "content" element from the object

**Examples**

```
## Not run:
  imagePath <- system.file(
    "extdata", "golden_retriever_puppies.jpg", package = "googleCloudVisionR"
  )
  raw_response <- gcv_get_raw_response(imagePaths = imagePath, maxNumResults = 7)

  str(raw_response)
  raw_response[["content"]]

## End(Not run)
```

---

`gcv_get_response`      *helper function to call the API for one batch of images*

---

**Description**

helper function to call the API for one batch of images

**Usage**

```
gcv_get_response(imagePaths, feature, maxNumResults)
```

**Arguments**

<code>imagePaths</code>	character, file paths, URLs or Cloud Storage URIs of the images, can be a combination of all three
<code>feature</code>	character, one out of: "LABEL_DETECTION", "FACE_DETECTION", "TEXT_DETECTION", "DOCUMENT_TEXT_DETECTION", "LOGO_DETECTION", "LANDMARK_DETECTION"
<code>maxNumResults</code>	integer, the maximum number of results (per image) to be returned.

**Value**

a data frame with image annotation results

---

`get_bounding_boxes`      *helper function code to extract Bounding Box x,y coordinates for an API response element*

---

**Description**

helper function code to extract Bounding Box x,y coordinates for an API response element

**Usage**

```
get_bounding_boxes(response)
```

**Arguments**

`response`              an element of the API response object

**Value**

a data.table

---

`get_invalid_image_paths`  
*helper function to validate input image paths*

---

**Description**

helper function to validate input image paths

**Usage**

```
get_invalid_image_paths(vec)
```

**Arguments**

`vec`                      a vector of paths

**Value**

vector of invalid paths from @vec

---

label\_detection\_extractor

*helper function code to extract API response into a data.table for given feature type*

---

**Description**

helper function code to extract API response into a data.table for given feature type

**Usage**

```
label_detection_extractor(response)
```

**Arguments**

response            an element of the API response object

**Value**

a data.table

---

landmark\_detection\_extractor

*helper function code to extract API response into a data.table for given feature type*

---

**Description**

helper function code to extract API response into a data.table for given feature type

**Usage**

```
landmark_detection_extractor(response)
```

**Arguments**

response            an element of the API response object

**Value**

a data.table

---

logo\_detection\_extractor

*helper function code to extract API response into a data.table for given feature type*

---

**Description**

helper function code to extract API response into a data.table for given feature type

**Usage**

```
logo_detection_extractor(response)
```

**Arguments**

response            an element of the API response object

**Value**

a data.table

---

ocr\_extractor

*helper function code to extract API response into a data.table for given feature type*

---

**Description**

helper function code to extract API response into a data.table for given feature type

**Usage**

```
ocr_extractor(response)
```

**Arguments**

response            an element of the API response object

**Value**

a data.table

---

split\_to\_chunks      *helper function to split a vector to approximately equally sized chunks*

---

**Description**

helper function to split a vector to approximately equally sized chunks

**Usage**

```
split_to_chunks(vec, chunkSize)
```

**Arguments**

vec                    a vector  
chunkSize            integer, how long should the chunks be?

**Value**

a list of chunks

# Index

[call\\_vision\\_api](#), [2](#)  
[create\\_request\\_body](#), [3](#)  
[create\\_single\\_image\\_request](#), [3](#)

[encode\\_image](#), [4](#)  
[extract\\_annotations](#), [5](#)  
[extract\\_error](#), [5](#)  
[extract\\_response](#), [6](#)  
[extractor](#), [4](#)

[face\\_detection\\_extractor](#), [6](#)

[gcv\\_get\\_available\\_feature\\_types](#), [7](#)  
[gcv\\_get\\_image\\_annotations](#), [7](#)  
[gcv\\_get\\_raw\\_response](#), [8](#)  
[gcv\\_get\\_response](#), [9](#)  
[get\\_bounding\\_boxes](#), [10](#)  
[get\\_invalid\\_image\\_paths](#), [10](#)

[label\\_detection\\_extractor](#), [11](#)  
[landmark\\_detection\\_extractor](#), [11](#)  
[logo\\_detection\\_extractor](#), [12](#)

[ocr\\_extractor](#), [12](#)

[split\\_to\\_chunks](#), [13](#)