

Structure of json file guide

The data in json file was structured having python dictionaries in mind. When parsing the json file in python, the result will be a dictionary of dictionaries.

There are 2 reasons for this, the first is that dictionaries are easily parsed into JSON (the data was scraped using python) and second, the tables in the dataset were not consistent with their lines. Consider the two tables below:

ISIC_0000037	
Info	
Unique ID	5436e3afbae478396759f119
Dataset	ISIC_UA-1_1
Created	October 9, 2014 at 14:36:15
License	CC-0
Clinical Metadata	
age_approx	70
benign_malignant	benign
diagnosis	nevus
diagnosis_confirm_type	
melanocytic	true
sex	male
Acquisition Metadata	
Dimensions (pixels)	1504 × 1129
Unstructured Metadata	
diagnosis	Nevus
id1	38
localization	Back
site	bar

ISIC_0012076	
Info	
Unique ID	559437d39fc3c1315514665d
Dataset	ISIC_MSK-1_1
Created	July 1, 2015 at 13:56:19
License	CC-0
Clinical Metadata	
age_approx	45
benign_malignant	benign
diagnosis	lentigo simplex
diagnosis_confirm_type	histopathology
melanocytic	false
sex	male
Acquisition Metadata	
Dimensions (pixels)	6681 × 4420
Unstructured Metadata	
Breslow	
Clark	
biopsy done	Y
clinical diagnosis/impression	simple lentigo
dermoscopy	NPD
lesion id	1222
localization	I parietal scalp
pathology diagnosis	simple lentigo
pathology diagnosis subtype	
pathology report available	Y
patient	559
previous biopsy	N

It is clear that image tables may have different amounts of information. It seems this happens for sections Clinical Metadata and Acquisition Metadata. As such, the raw scraped data was parsed into a dictionary-like structure to account for the changing nature of the table.

Note: Since the parsing into json was done using unordered dictionaries, the data in the file is unordered. Indexes however, are consecutive and unique so ordering it (programmatically) should not be a problem.

The json structure is as follows:

```
{index:image characteristics:table headlines:table data}}}
```

```
{ "752": { "page": "751 - 800", "img_url": "https://isic-
archive.com/api/v1/image/54d90e40bae4780ab00c328c/download?contentDisposition=inline",
"img_path": "ima/ISIC_0000752.jpg", "img": "ISIC_0000752.jpg (JPEG Image,
3024&nbsp;\u00d7&nbsp;2016 pixels) - Scaled (31%)", "parsed_data": {"Unstructured Metadata":
{"race": "Hispanic", "diagnosis": "nevus"}, "Info": {"Unique ID": "54d90e40bae4780ab00c328c",
"Dataset": "ISIC_SONIC_1", "Created": "February 9, 2015 at 13:45:04", "License": "<a
target=\"_blank\" href=\"https://creativecommons.org/publicdomain/zero/1.0/\">CC-0</a>"},
"Acquisition Metadata": {"Dimensions (pixels)": "3024 \u00d7 2016"}, "Clinical Metadata":
{"benign_malignant": "benign", "diagnosis_confirm_type": "single image expert consensus",
"diagnosis": "nevus", "sex": "female", "melanocytic": "true", "age_approx": "10"}}, "data":
"<tbody><tr><td colspan=\"2\" class=\"isic-image-details-table-section-
header\">Info</td></tr><tr><td>Unique
ID</td><td><code>54d90e40bae4780ab00c328c</code></td></tr><tr><td>Dataset</td><td>ISIC_
SONIC_1</td></tr><tr><td>Created</td><td>February 9, 2015 at
13:45:04</td></tr><tr><td>License</td><td><a target=\"_blank\"
href=\"https://creativecommons.org/publicdomain/zero/1.0/\">CC-0</a></td></tr><tr><td
colspan=\"2\" class=\"isic-image-details-table-section-header\">Clinical
Metadata</td></tr><tr><td>age_approx</td><td>10</td></tr><tr><td>benign_malignant</td><td>be
nign</td></tr><tr><td>diagnosis</td><td>nevus</td></tr><tr><td>diagnosis_confirm_type</td><td>
single image expert
consensus</td></tr><tr><td>melanocytic</td><td>true</td></tr><tr><td>sex</td><td>female</td></tr><tr><td
colspan=\"2\" class=\"isic-image-details-table-section-header\">Acquisition
Metadata</td></tr><tr><td>Dimensions (pixels)</td><td>3024 \u00d7 2016</td></tr><tr><td
colspan=\"2\" class=\"isic-image-details-table-section-header\">Unstructured
Metadata</td></tr><tr><td>diagnosis</td><td>nevus</td></tr><tr><td>race</td><td>Hispanic</td>
</tr></tbody>"}
```

For illustration purposes, let's assume that the processing of this file is done in python.

```
elements=json.load(file)
```

The above makes elements a dictionary of dictionaries.

Index

Indexes were assigned sequentially as the images were scraped, meaning that if the first image in the dataset is ISIC_000000 then its index is 0.

This example shows image 752. So let's assume we want to work with that image.

```
element=elements[752]
```

Image characteristics

Some image characteristics (not related to the table in the website) were added to ease the manipulation of the dataset. Assume we want to load the jpg file of image 752. First, we would have to retrieve it from its path. Since all images were saved in a folder called "ima" then the path can easily be constructed because it is saved as an image characteristic in the json file

```
incomplete_image_path=elements[752]["img_path"]
```

This will return a string with the value "ima/ISIC_0000752.jpg"

```
complete_image_path="the/path/leading/to/the/location/of/"+incomplete_image_path
```

This allows for a hassle-free programmatic retrieval of images.
What follows is a list of all the keys and the value they return.

Key	Retrieval method (as per example)	Returns
"data"	<code>elements[752]["data"]</code>	Raw version of the scraped data. It's the html code that contains the table
"img_path"	<code>elements[752]["img_path"]</code>	a string with the value the last part in the path to the image (folder and file name).
"img_url"	<code>elements[752]["img_url"]</code>	string with the url to the hi-res image.
"img"	<code>elements[752]["img"]</code>	string with the name of the image
"page"	<code>elements[752]["page"]</code>	string with the value of the website's page in which that particular image is located.
"parsed_data"	<code>elements[752]["parsed_data"]</code>	dictionary containing table information. For more information see table headlines section

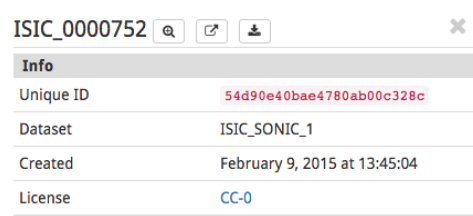
Table headlines

Keys related to this color refer to the data scraped from the table that appears next to the image in the website.

The keys in this section refer to the headlines of each table. More specifically: Info, Clinical Metadata, Acquisition Metadata and Unstructured Metadata
To access such data we follow the same logic as before. Lets assume we want to access the image's information from the table

```
info=elements[752][["page"]]["parsed_data"]["Info"]
```

The above will return a dictionary with containing the lines of headline info.



Info	
Unique ID	54d90e40bae4780ab00c328c
Dataset	ISIC_SONIC_1
Created	February 9, 2015 at 13:45:04
License	CC-0

```
info={"Unique ID":54d90e40bae4780ab00c328c, "Dataset":"ISIC_SONIC_1...."}
```

It would be wise to check first which keys are available by exploring the dictionary of each table because as shown before not all images contain the same information.

Table Data

These keys will deliver the actual value of each line as a string. Following the example from the previous section, lets assume we want to extract the date in which the image was created

```
date= info=elements[752][["page"]]["parsed_data"]["Info"]["Created"]
```

The same logic applies to any value in the data assuming it exists in the table.