

SUSTINEO EXTRACTOR

A tool to extract data from materiality matrices

Author: Michele Ferro

Institute: Dipartimento di Matematica e Informatica | Università di Catania

June 2023



Materiality Matrices

SustiNEO PDF2Plot

SustiNEO Extractor

Materiality Matrices

What is it?

The **materiality matrix** is the final output of a process know as **materiality analysis**.

Materiality analysis: in short

It is a process that makes able to detect everything has an impact on the **company's business** or on which the business could have an impact.

The term **materiality** highlights the importance of all the elements that make clear the company's commitment on being **sustainable**.

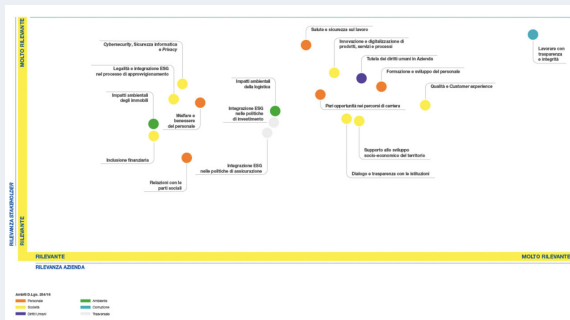
To make the analysis effective, the principal needed element is the involvement of all the **stakeholders** who influence and/or are influenced by the organization activities, know as **stakeholder engagement**.

Some of these are:

- ⊙ employees;
- ⊙ providers;
- ⊙ customers;
- ⊙ medias;
- ⊙ population;
- ⊙ and so on...

How does the materiality matrix work?

The **materiality matrix** is (typically) a 2D plot having on the y-axis the values that are relevant for the **company** and on the x-axis those that are relevant to the **stakeholders**.



By the positioning of the themes, the viewer could have a clear visual about the company **sustainability objectives** and how relevant are.

Why it is important?

After viewing the materiality matrix, the next step is a further comparison in which all the functions involved in the analysis process and the top management try to understand the polarization values and the classification reasons.

Without this passage, the materiality matrix becomes **a mere reporting exercise**.

A problem: the various forms of the materiality matrix

Anyway, there are various forms of this plots, often poorly understandable and full of articulated iconographies that makes them practically not very very useful due to the lack of real quantitative data.

This makes all the plots a sort of **patinated graphics**.

In order to obtain more understandable data, some python utilities have been developed.

SustiNEO PDF2Plot

SustiNEO PDF2Plot is a Python software build using the PyMuPDF and the OpenCV libraries; substantially, it allows to extract materiality matrices from PDF-type files.

It's behavior can be summarized into four steps:

1. interested string search;
2. interested page conversion;
3. plot detection;
4. user adjustment (optional).

1. Interested string search

In (almost) every sustainability report, the plot regarding materiality analysis is portrayed under the heading **materiality matrix**; obviously, this string depends on the PDF native language, so in order to automatize the search, the user can pass a three characters argument that identifies the text language (i.e. `ita`, `eng`, etc.).

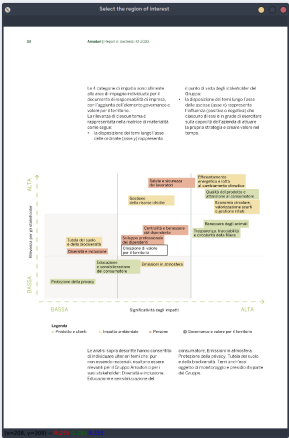
During this first step, with the help of the PyMuPDF libraries, the script simply **searches** through the document file every page where this string is present, and ignores all the other ones.

2. Interested page conversion

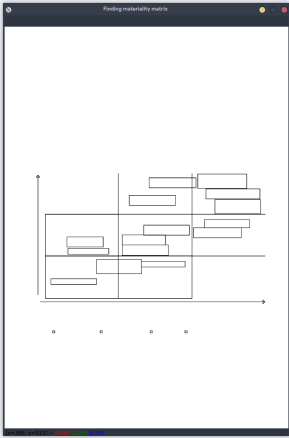
Following, of every interest page are made two different kind of **conversions**.

- ⊙ **RAW**: The first conversion is a plain page in PNG-format, which will be later used to make the cropping operations.
- ⊙ **Vector**: the second one is another PNG-format image, which instead contains only the vectorial data (it's a textless page containing only the plot shapes); this data will be later used to detect the actual plot.

2. Interested page conversion >_ EXAMPLES



(a) RAW image

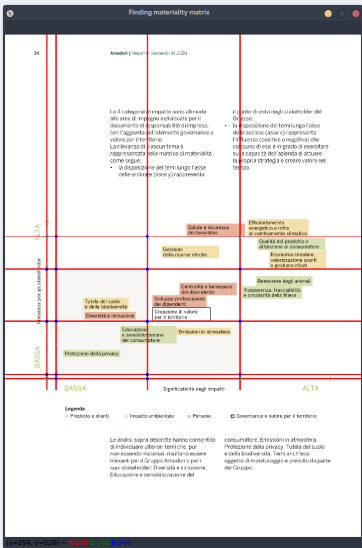


(b) Vector-data image

3. Plot detection

The plot is detected using the **Hough transform** on the image containing the **vector data**. The limiter vertices are given by the Hough lines **intersections**; their (x, y) coordinates are then memorized and used as delimiter points to crop the RAW image, in order to obtain the final plot. Then the plot is showed to the user.

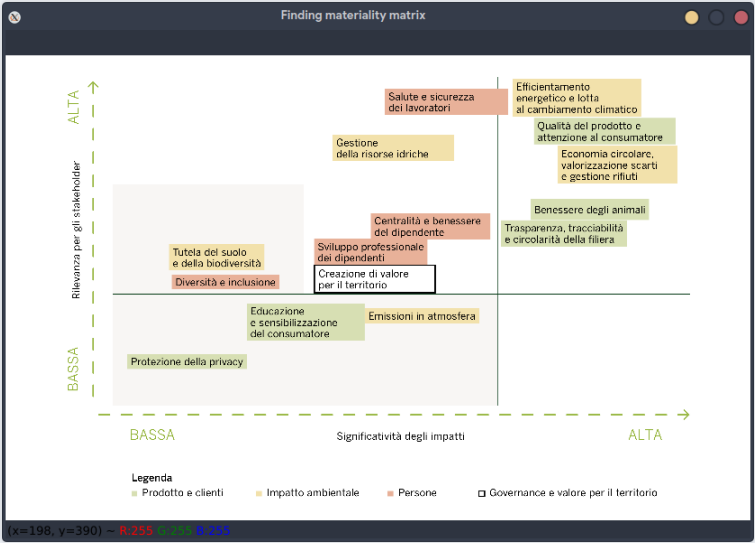
3. Plot detection >_ EXAMPLE



4. Final user adjustment (optional)

The prompt asks the **user** if the cropped area actually contains the plot; if affirmative, the image is saved to the disk, otherwise the user can adjust the crop, making this operation manually.

At the end of this step, the plot is finally saved on a local folder as a PNG-format image.



SustiNEO Extractor

SustiNEO Extractor is another Python software built using the PyTesseract (a Python wrapper for Google's *Tesseract-OCR*) and the OpenCV libraries; substantially, it allows to extract data from **box-type** and **blob-type** materiality matrices.

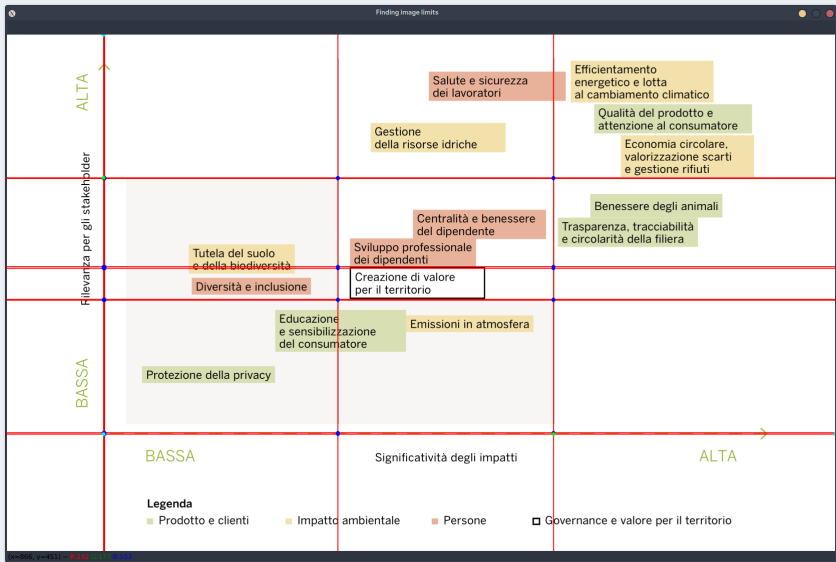
It's behavior can be summarized into four steps:

1. plot/legend subdivision;
2. plot interpretation;
3. legend interpretation;
4. final export.

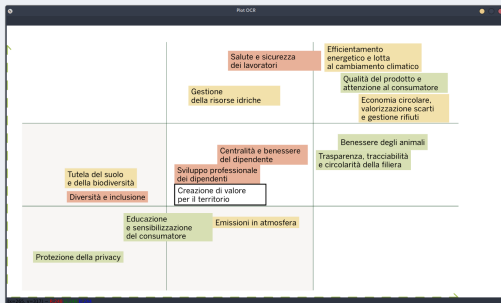
1. Plot/legend subdivision >_ Box-type

- ⊙ Using a **bounding-box** approach, after **binarizing** the input plot image, all the lines that form it are detected through the **Hough transform**.
- ⊙ Next, the **intersections** are computed and through them, the software detects the minimum-area box of the plot, the one on the bottom-left.
- ⊙ Using this rectangle, is then used a 1:3 aspect ratio to find the entire area of the plot: in particular, the subdivision is done using the three limiting vertices of the rectangle including the plot (top-left, bottom-left, top-right).
- ⊙ Finally, the cropped plot and legend sections of the image are passed as input to the data-extraction module of the software.

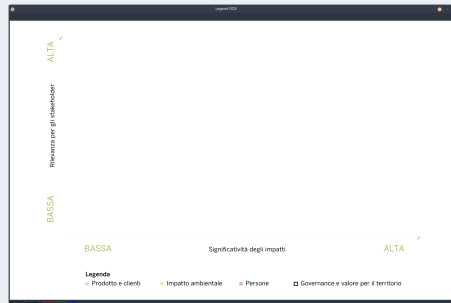
1. Plot/legend subdivision >_ Box-type [EXAMPLE]



1. Plot/legend subdivision >_ Box-type [EXAMPLES]



(a) Plot section



(b) Legend section

In this case, a different approach - based on a **blob detector** - has been chosen.

- ⊙ Using the detector, the software finds all the **blobs** (circular forms) that have the same coordinates, disposed both in horizontal and in vertical.
- ⊙ To distinguish the **legend**'s blob from the actual **plot**'s blob (representing the values in (x, y) coordinates), the software counts the number of blobs that share the same coordinates:
 - if more than two, then we are in presence of a legend;
 - else there is no legend on the plot.
- ⊙ Finally, the software detects the **first** and the **last** blob composing the legend, and using them as delimiters, it "crops" the legend away from the input image.

1. Plot/legend subdivision >_ Blob-type [EXAMPLE]



1. Plot/legend subdivision >_ Blob-type [EXAMPLES]



(a) Plot section



(b) Legend section

2. Plot interpretation

This is obviously the main passage of the plot processing; it is divided into the following subsequent steps:

1. shapes detection;
2. text detection;
3. legend link.

2.1. Shapes Detection >_ Box-type

The data-extraction `PlotOCR_Box` class object does a shape detection operation onto the plot file data.

In particular, after **binarizing** the input plot (again, but using a **different threshold** in order to obtain the shapes of the rectangles in the image) and using a 2×2 -**kernel dilatation** (in order to **remove the thin lines**), all the corners are extracted.

The detected shapes, called `LabelBoxes`, are then stored in a data structure in order to be used for further operations.

2.1. Shapes Detection >_ Blob-type

In this case, is the data-extraction PlotOCR_Blob class object to do the shape detection operation onto the plot file data.

More specifically, after **binarizing** the input plot and using a 2×2 -kernel dilatation, all the blobs are detected using a blob detector; however, in this passage a different radius threshold is set in order to discard all **keypoints** detected from the letters of the labels.

2.1. Shapes detection >_ Blob-type [EXAMPLE]



2.2. Text detection

Another **threshold** operation is done, but this time in manner to leave only the text and ignore the shape.

Also during this pass, two different cases can be distinguished:

- ⊙ **black text on colored images**: the binarized image has exclusively black text, without any background (as it is discarded during the thresholding);
- ⊙ **white text on colored images**: the binarized image has white text on black boxes.

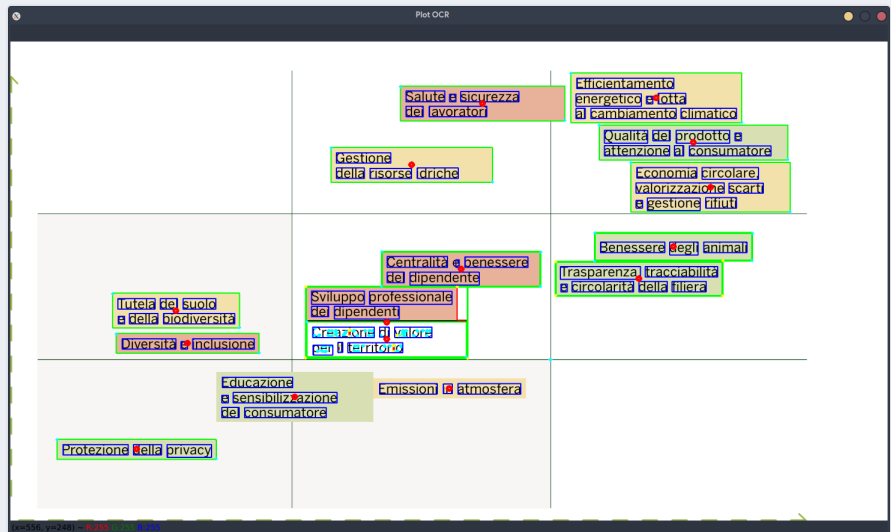
As the OCR module works better with black-texted images, in the 2nd case the image is **converted to negative** in order to proceed like in the 1st one.

Next, calling the pyTesseract module we are able to obtain a dictionary containing different informations about the words on the box, their respective pixel-coordinates, and bounding boxes informations.

All these informations are used to instantiate TextBox class objects for every detected word.

The **bounding boxes** and the respective coordinates are used to detect which word is contained in which colored box, and thus in which LabelBox, forming then the label.

2.2. Text detection >_ Box-type [EXAMPLE]

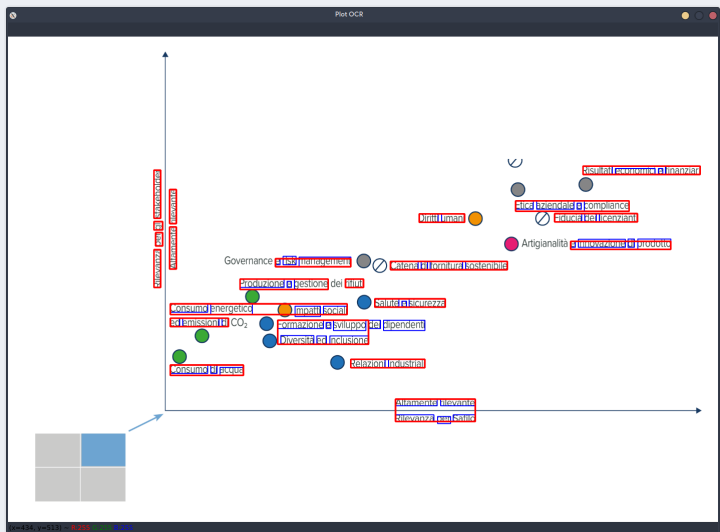


Every TextBox is put into a LabelBoxColorless according to the following **heuristic**:

- ⊙ if the current word has a distance $d_x < t_x$ from next one, then they are on the same row;
- ⊙ else, the next word is in the following row if the distance between the first word of the first row has a distance $d_y < t_y$.

Finally, each BlobBox is joined to the nearest LabelBoxColorless (using the **euclidean distance** between the **center of the blob** and the **center of the LabelBoxColorless**).

2.2. Text detection >_ Blob-type [EXAMPLE]



2.3. Legend link

Meanwhile, for every **rectangle** or **blob** in the plot, the **color data** is saved (both in RGB and HSV format) and then stored in the `color_rgb` and `color_hsv` attributes of every single `TextBox`-class objects.

This information will be used in the next part.

All the `LabelBoxes` or `BlobBoxes` data are finally given as output.

3. Legend interpretation

If actually present, the **legend image-data** is passed as input to the LegendOCR class object.

At first, calling the pyTesseract module, we obtain the same kind of dictionary obtained in the previous pass. In the legend, the data appears in the following manner:



■ Persone

So, converting the legend image in HSV color-space format, and then taking the **non-zero point** for every color in colors_hsv obtained during the plot extraction pass, we are able to detect the colored shape's position.

Then, we can process the legend: knowing that the first TextBox of every single legend-string is next near to the color shape, the **euclidean distance** between them is **minimum**; to avoid errors, we can here use a threshold. Given this **minimum distance**, we know that the first word of the label is of that label of the legend: so, we can create a LegendBox object identified by the **position** of the colored shape.

4. Final export

Here, the `LabelBoxes` and the `LegendBoxes` data are passed to the `Exporter` class object.

If there is no Legend, the legend-data is set as `None` and then discarded.

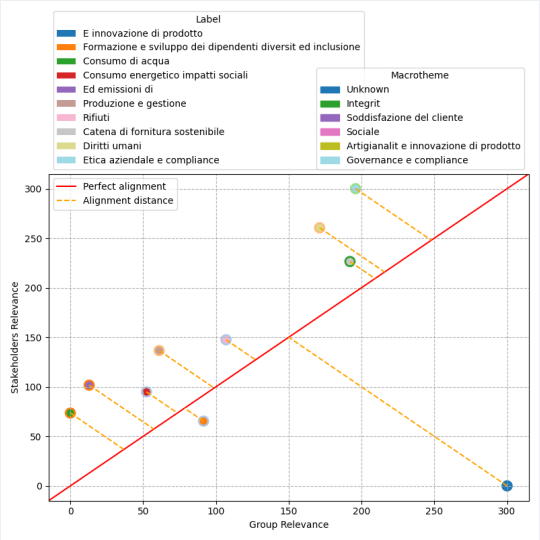
The following operations are done:

1. **normalization**: the coordinates of the `LabelBoxes`' middle points are normalized in the range `[0, 300]`, in order to have 100 units for each block (vertically and horizontally).
2. **CSV export**: the collected data are gathered in a pandas dataframe and then exported in a csv file on the disk.
3. **PNG export**: the pandas dataframe data are used to generate a plot using the `matplotlib` module, and finally saved on a local folder.


4. Final export >_ Open CSV-format


	▲▼	Label ▼	Macrotheme ▼	Group Rel ▼	Stake Rel ▼	Rank Group ▼	Rank Stake ▼	Rank Absolute ▼	Alignment ▼
	0	E innovazione di prodotto	Unknown	300	0	10	0	10	100
	1	Formazione e sviluppo dei dipendenti diversit ed inclusione	Integrit	91.51	65.36	4.44	1.11	4.44	0
	2	Consumo di acqua	Soddisfazione del cliente	0	73.52	0	2.22	0	17.3
	3	Consumo energetico impatti sociali	Integrit	52.32	94.86	2.22	3.33	2.22	5.98
	4	Ed emissioni di	Soddisfazione del cliente	12.93	101.66	1.11	4.44	1.11	22.85
	5	Produzione e gestione	Sociale	61	136.61	3.33	5.56	3.33	18.06
	6	Rifiuti	Integrit	106.95	147.5	5.56	6.67	5.56	5.26
	7	Catena di fornitura sostenibile	Artigianalit e innovazione di prodotto	192.08	226.48	7.78	7.78	7.78	3.01
	8	Diritti umani	Sociale	171.24	260.51	6.67	8.89	6.67	23.05
	9	Etica aziendale e compliance	Governance e compliance	195.95	300	8.89	10	8.89	28.45

4. Final export >_ Open PNG-format



These softwares and all the documentation are hosted on GitHub

 https://github.com/nebuchadneZZar01/sustineo_extractor

 <mailto:michele.ferro1998@protonmail.com>