HITL Visualization Tool (External)

Table of Content

Disclaimer

This tool is not supported by the Google engineering team or product team. It is provided and supported on a best-effort basis by the **DocAI Incubator Team**. No guarantees of performance are implied.

Objective

This tool generates reports based on the information which are present in the HITL labeled JSON file.

The generated report consists of entity names, mentioned text in an easily readable format and along with visual annotations of entities present in each page of the JSON file. The reports are generated in batches of 20 Xlsx Sheets.

Prerequisites

1. Python : Jupyter notebook (Vertex AI) or Google Colab
   Permission to the Google project is needed to access the JSON files.
2. HITL Labeled JSON Files

Tool Installation Procedure

The tool consists of some Python code. It can be loaded and run via:

1. From Google Colab (*Path*) - Before starting, please make a personal copy on your Colab instance, via File/Save a copy in Drive.
   or
2. The code can also be copied into a Google Colab or Vertex Notebook.

Tool Operation Procedure

1. Install and Import the modules

```
Python
pip install pandas Pillow graphviz XlsxWriter
!wget
https://github.com/opensourcedesign/fonts/raw/master/gnu-freefont_freemono/Free
MonoBold.ttf
```

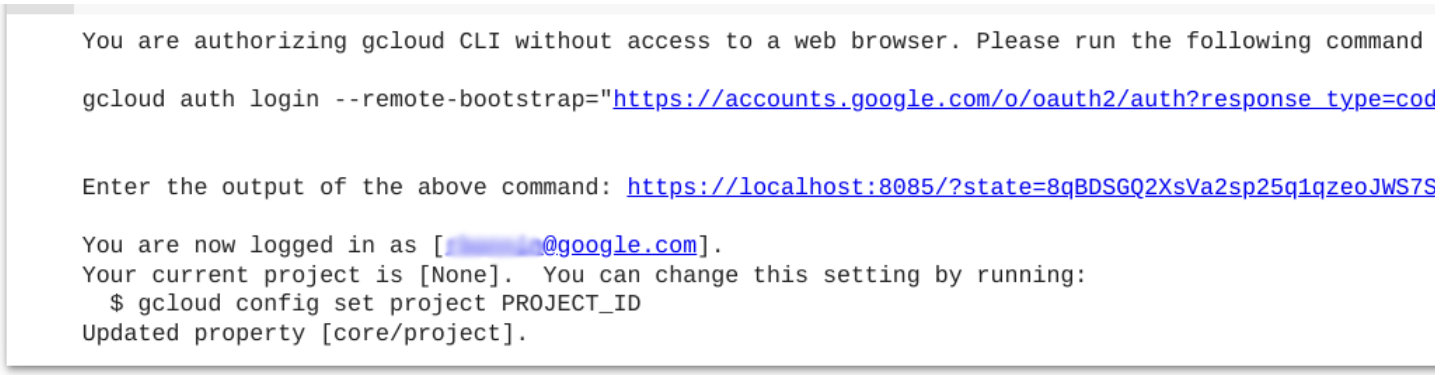
```
Python
from pathlib import Path
from glob import glob
import json
import pandas as pd
from PIL import Image
import os
import time
from pathlib import Path
import itertools
```

```
from PIL import Image, ImageFont, ImageDraw
import base64
import sys
import json
from graphviz import Source
```

2. Gcloud authentication and config

Run the following command to authenticate with your Google account and set the project property

```Python
!gcloud auth login #Not required in Vertex AI Notebook
!gcloud config set project <project_id>
```



```
You are authorizing gcloud CLI without access to a web browser. Please run the following command

gcloud auth login --remote-bootstrap="https://accounts.google.com/o/oauth2/auth?response_type=cod


Enter the output of the above command: https://localhost:8085/?state=8qBDSGQ2XsVa2sp25q1qzeoJWS7S

You are now logged in as [[redacted]@google.com].
Your current project is [None].  You can change this setting by running:
  $ gcloud config set project PROJECT_ID
Updated property [core/project].
```

3. Make a directory to upload files

Creates the folder to upload the files

```Python
!rm -rf upload
!mkdir upload
```

4. Copy the JSON Files

Copies JSON files which need to be visualized from source bucket to the Colab instance folder.

Create a new empty folder in your Google Colab or Vertex AI Notebook and add the path to it as `/<empty-folder-path>/`.

```python
# If your JSON is present inside multiple subfolders please use the first
command or else use the second command.
# Please use the below command to copy single JSON file from a folder
#!gsutil -m cp gs://cbre_files/model_evaluation/9372326859845912644/*.json
/<empty-folder-path>/
# Please use the below command to copy multiple json files from subfolders.
!gsutil -m cp gs://cbre_files/model_evaluation/*/*.json /<empty-folder-path>/
```

5. List of JSON files

Gets the list of JSON files from the *upload* directory and stores in a variable

```python
from glob import glob
filelist=[ i for i in glob( "/<empty-folder-path>/*.json")]
limit = 20 #Number Sheets to be present in an workbook
filelist=list(filelist[i:i+limit] for i in range(0, len(filelist), limit))
```

6. Run the extract_file_Image function

This function will convert the base64 encoded image into PNG and highlight the entities that are present in the JSON.

```python
def extract_file_image(filename, elements, pagenumber):
  document_image = Image.open(filename + ".png")
  draw = ImageDraw.Draw(document_image, 'RGBA')
  fnt = ImageFont.truetype("./FreeMonoBold.ttf", 18)
  for entity in elements:
    # Draw the bounding box around the entities
    vertices = []
    if "pageAnchor" not in entity:
      continue
    if "pageAnchor" in entity and "page" not in
entity["pageAnchor"]["pageRefs"][0] and pagenumber != 0:
      continue
    if "pageAnchor" in entity and (("page" in
entity["pageAnchor"]["pageRefs"][0]) and
(int(entity["pageAnchor"]["pageRefs"][0]["page"]) != pagenumber)):
      continue

    for vertex in
entity["pageAnchor"]["pageRefs"][0]["boundingPoly"]["normalizedVertices"]:
      vertices.append({
        'x': vertex["x"] * document_image.size[0],
        'y': vertex["y"] * document_image.size[1]
      })

    if 'provenance' in entity:
      draw.polygon([vertices[0]['x'], vertices[0]['y'], vertices[1]['x'],
vertices[1]['y'], vertices[2]['x'], vertices[2]['y'], vertices[3]['x'],
vertices[3]['y']], outline='yellow', fill=(255, 254, 0, 100))
      draw.ellipse([vertices[0]['x'], vertices[0]['y']-22,
vertices[0]['x']+22, vertices[0]['y']], outline='black', fill=(255, 254, 0,
100))
      id = entity.get("id", 0)
      textpad = 5 if str(id) == '1' else 0
      draw.text((vertices[0]['x'] + textpad, vertices[0]['y'] - 18), str(id),
font=fnt, fill=(255, 255, 255, 127))
    else:
      draw.polygon([vertices[0]['x'], vertices[0]['y'], vertices[1]['x'],
vertices[1]['y'], vertices[2]['x'], vertices[2]['y'], vertices[3]['x'],
vertices[3]['y']], outline='blue', fill=(0, 0, 255, 70))
      draw.ellipse([vertices[0]['x'], vertices[0]['y']-22,
vertices[0]['x']+22, vertices[0]['y']], outline='black', fill=(0, 0, 255, 70))
      id = entity.get("id", 0)
      textpad = 5 if str(id) == '1' else 0
      draw.text((vertices[0]['x'] + textpad, vertices[0]['y'] - 18), str(id),
font=fnt, fill=(255, 255, 255, 127))

  document_image.save(filename + "_processed.png")
  im = Image.open(filename + "_processed.png")
  rgb_im = im.convert('RGB')
  rgb_im.save(filename + "_processed.jpg")
```

7. Run the create_dot_diagram function

This function will generate a dot graph in the report based on the entities.

Python
```python
def create_dot_diagram(filename, entitylist):
  graph = "forcelabels=true;\n"
  for i in entitylist:
    if type(i[0]) == str:
      s = i[0]
      i[0] = s.replace("-", "")

    graph = graph + str(i[0]) + " [label=\"" + str(i[0]) + ":" + str(i[3]) +
"\n" + i[1][:10] + "\"]\n"

    if i[2] != "":
      graph = graph + str(i[0]) + " -> " + str(i[2]) + ";\n"

  try:
    src = Source("digraph G {\n" + graph + "}")
    src.render(filename + "graph.dot", format='png')
  except Exception as e:
    print(e)
    print("Dot graph skipped")
```

8. Run the serialize_entities function

```python
def serialize_entities(entities, parent=-1):
  for entity in entities:
    newentity = entity.copy()
    newentity['parent'] = parent
    newentity.pop('properties', [])
    elements.append(newentity)

    if "properties" in entity:
      serialize_entities(entity["properties"], entity['id'])
```

9. Run the extract_instances function

This function checks for entities in a JSON and passes the entity list to the *serialize_entities* function.

```python

def extract_instances(jsonfile):
  with open(jsonfile) as json_file:
    data = json.load(json_file)

    if "entities" not in data:
      print("No entities!")
      return data

    serialize_entities(data["entities"])
    data["entities"] = elements

  return data
```

10. Run the page_number function

This function returns only the list of page numbers which have entities.

```
Python
def page_number(json1):
  with open(json1, 'r') as data:
    filename = Path(json1).name
    data_1 = json.load(data)
    entity = data_1["entities"]
    entity_present_page = []

    for entities in entity:
      try:
        if "page" not in entities["pageAnchor"]["pageRefs"][0]:
          page = 0
        else:
          page = entities["pageAnchor"]["pageRefs"][0]["page"]
        entity_present_page.append(int(page))

      except KeyError:
        continue

    entity_present_page = set(entity_present_page)
    len_page = len(data_1["pages"])
    totalpages = set(list(range(len_page)))
    final_pages = list(totalpages.difference(entity_present_page))
    new_pages = list(entity_present_page)

    return new_pages
```

12. Generate the report

```
Python

report_number = 1

for j in filelist:
  writer = pd.ExcelWriter(str(report_number) + '_report.xlsx',
engine='xlsxwriter')
  report_number += 1
  workbook = writer.book
  elements = []

  for jsonfile in j:
    try:
      elements = []
      filename = Path(jsonfile).name
      print("Processing : ", filename)
      globalentities = []
      headerheight = 5

      with open(jsonfile) as json_file:
        data = json.load(json_file)
        elements = extract_instances(jsonfile)
        sheetname = filename[:30]
        for i in elements['entities']:
          provenance = i["provenance"]["type"] if "provenance" in i else ""
          normalizedValue = i["normalizedValue"]["text"] if ("normalizedValue"
in i and "text" in i["normalizedValue"]) else ""
          mentionText = i.get('mentionText', "")
          confidence = i.get('confidence', 0)
          id = i.get('id', -1)
          globalentities.append([id, mentionText, i["parent"], i['type'],
provenance, normalizedValue, round(float(confidence), 2)])

        create_dot_diagram(filename, globalentities)
        entities = pd.DataFrame(globalentities)

        page_list = page_number(jsonfile)
        for pagenumber in page_list:
          with open(filename + "_" + str(pagenumber) + ".png", "wb") as file:

file.write(base64.b64decode(data["pages"][pagenumber]["image"]["content"]))
            extract_file_image(filename + "_" + str(pagenumber),
elements['entities'], pagenumber)
```

```
        entities = entities.rename(columns={0: 'ID', 1: 'Text', 2: 'Parent', 3:
'Type', 4: 'Provenance', 5: 'Norm. Value', 6: 'Confidence'})
        entities.to_excel(writer, sheet_name=sheetname, startrow=headerheight)
        worksheet = writer.sheets[sheetname]
        workbook = writer.book
        bold = workbook.add_format({'bold': True})
        title = workbook.add_format({'bold': True, 'font_size': 16})

        for letter in ["B", "C", "D", "E", "F"]:
          worksheet.set_column(letter + ":" + letter, 25, bold)

        worksheet.write('B1', 'DocAI json parser report', title)
        worksheet.write('B2', 'Filename: ' + filename, title)
        worksheet.write('B5', 'Detected entities', title)

        i = 0
        try:
          for i, pagenumber in enumerate(page_list):
            worksheet.insert_image('K' + str(40 * i + 1), filename + "_" +
str(pagenumber) + "_processed.jpg", {'x_scale': 0.3, 'y_scale': 0.3})
            worksheet.insert_image('Y1', filename + "graph.dot.png",
{'x_scale': 0.6, 'y_scale': 0.9})
        except:
          print("Error saving image file")
          continue

    except Exception as e:
      print(e)
      print("issue with json. Skipped Json")

  writer.save()
  print('Ready!')
!rm -f *.dot *.jpg *.png
```

13. Copy Report to Google Drive

Copies the Generated XLSX Report to Google Drive. The XLSX report can be directly opened from the google drive; it will automatically redirect to the google sheets.

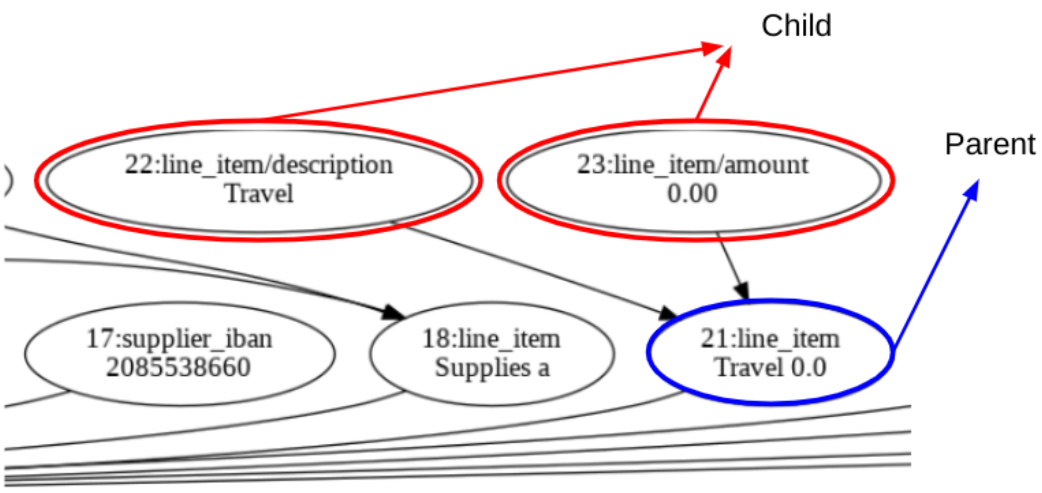
Note: This following code will work only on colab.

```Python
#The below commands will upload the json reports to your personal drive.
#Please replace "Json_report" from the below commands to your required
folder name
!mkdir /content/drive/MyDrive/Json_report
!cp /content/*.xlsx /content/drive/MyDrive/Json_report
```

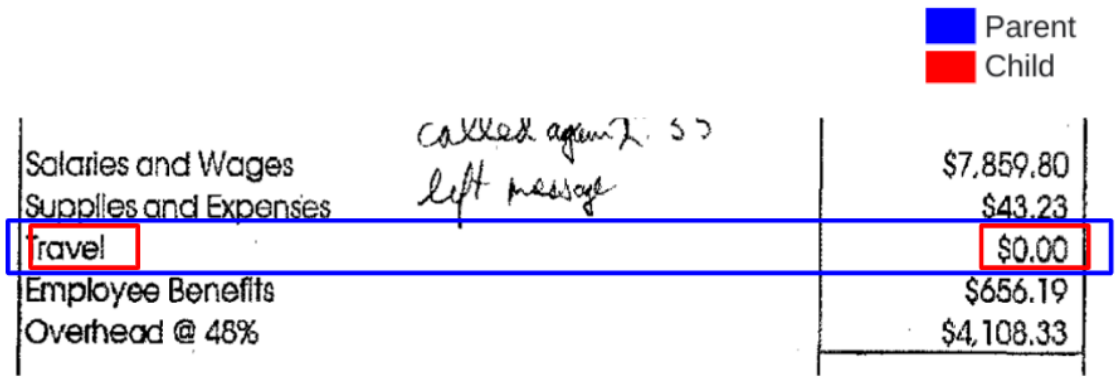
Analyzing the tool output

DOT Graph

Dot graph is used to display the distribution of quantitative data where the graphical representation shows the relations between objects.
Here we are showcasing the parent-child relationship of entities using the dot graph. In the below example, 22 and 23 line_item (Child) belong to the line_item 21 (Parent).



Entities representation in Dot graph



	called again 2:57 left message	
Salaries and Wages		$7,859.80
Supplies and Expenses		$43.23
Travel		$0.00
Employee Benefits		$656.19
Overhead @ 48%		$4,108.33

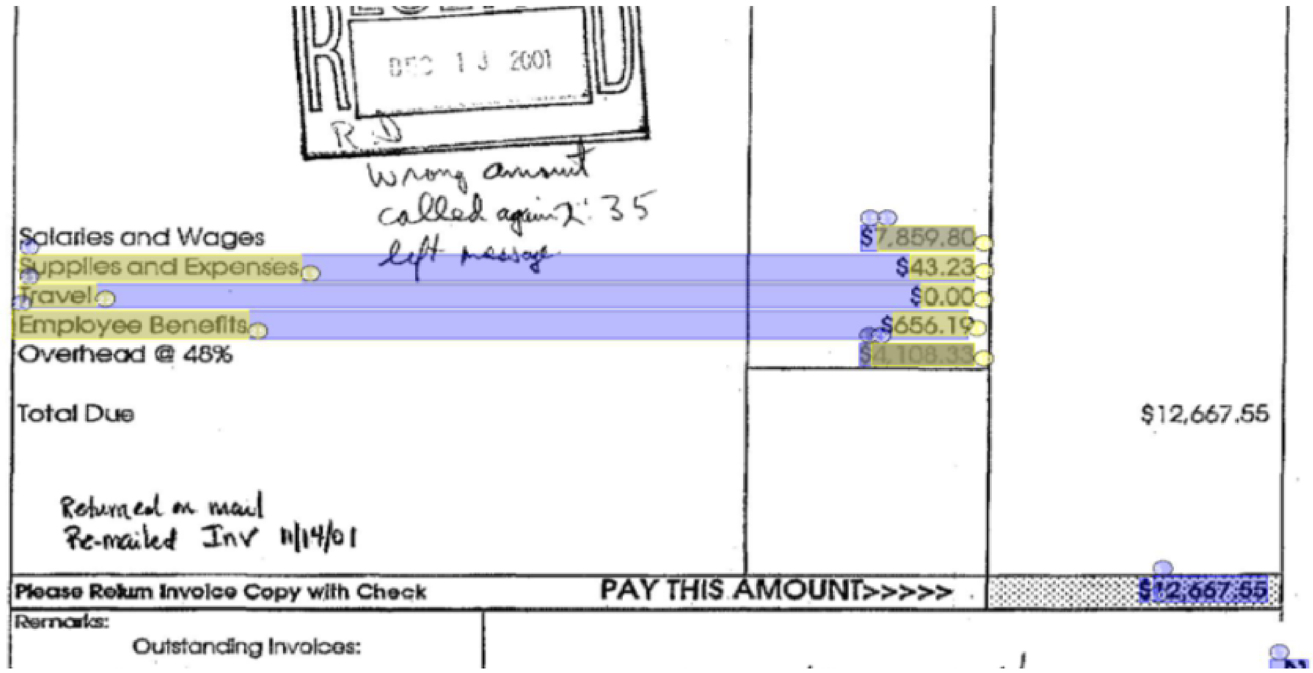
🟦 Parent
🟥 Child

Entities in Document

Visualizing the Annotation

In the below example we can visualize that there are 2 different colored annotation which denotes different use cases

1. 🟪 It denotes the labeled entities through the processor.
2. 🟨 This shows the entities that are changed/relabeled during HITL, where a reviewer makes changes during the process.



DEC 1 3 2001
R.D

Wrong amount
called again 2:35
left message

Solaries and Wages	$7,859.80	
Supplies and Expenses	$43.23	
Travel	$0.00	
Employee Benefits	$656.19	
Overhead @ 48%	$4,108.33	
Total Due		$12,667.55

Returned on mail
Re-mailed Inv 11/14/01

Please Return Invoice Copy with Check PAY THIS AMOUNT>>>>> $12,667.55

Remarks:
Outstanding Invoices: