DocAI - Script for Removing Empty Bounding Boxes (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.

Purpose of the Script

The purpose of this document is to provide instructions and a Python script for removing empty bounding boxes from a labeled JSON file. The script identifies and removes any bounding boxes (entities) in the JSON file that do not contain any mentionText or textAnchors, streamlining the labeling process and improving the accuracy of the labeling data.

Prerequisites

1. Python : Jupyter notebook (Vertex AI)
2. Service account permissions in projects.

Installation Procedure

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

1. Upload the IPYNB file or copy the code to the Vertex Notebook and follow the operation procedure.

NOTE: Don't Execute the Script with Processor Dataset Path. Export the dataset to json and then use that bucket as an input.

Operation Procedure

1. Import the modules

***Note** : external modules are used so they need to be installed. To install run these commands :*

```python
!pip install gcsfs
!pip install google-cloud
```

```python
import json
import google.auth
from tqdm import tqdm
```

```
import gcsfs
import pandas as pd
from google.cloud import storage
from pathlib import Path
```

2. Setup the required inputs

- **PROJECT_ID** - Your Google project id or name
- **BUCKET_NAME** - Name of the bucket
- **INPUT_FOLDER_PATH** - The path of the folder containing the JSON files to be processed, without the bucket name.
- **OUTPUT_FOLDER_PATH** - The path of the folder where the JSON files need to be stored after process, without the bucket name.

Note *: Both Input and output paths should be in the same bucket.*

Python
```
PROJECT_ID = "xxxxxx-xxxxxx-xxxxx"
BUCKET_NAME = "xxxxxxx"
INPUT_FOLDER_PATH = "xxxxxxxx/xxxxxxxxx/xxxxxx" #Path without
bucket name
OUTPUT_FOLDER_PATH = "xxxxxxxx/xxxxxxx/xxxxxxx/xxx" #Path without
bucket name
credentials, _ = google.auth.default()
fs=gcsfs.GCSFileSystem(project=PROJECT_ID, token=credentials)
```

3. Execute the code

Python
```
def get_file(file_path : str):
    """
    To read files from cloud storage.
```

```
    """
    file_object = json.loads(fs.cat(file_path))
    return file_object

def store_blob(document,file: str):
    """
    Store files in cloud storage.
    """
    storage_client = storage.Client()
    result_bucket = storage_client.get_bucket(BUCKET_NAME)
    document_blob = storage.Blob(
        name=str(file),
        bucket=result_bucket)
    document_blob.upload_from_string(json.dumps(document),
content_type="application/json")


def main():
    logs = pd.DataFrame(columns=['FileName'])

    files=[i for i in
fs.find(f"{BUCKET_NAME}/{INPUT_FOLDER_PATH}") if
i.endswith(".json")]
    json_files_list = [get_file(i) for i in files]
    print("No. of files : ",len(files))

    for index in tqdm(range(len(files))):
        file_name = files[index].split('/',1)[-1]
        output_file_name =
file_name.replace(INPUT_FOLDER_PATH,OUTPUT_FOLDER_PATH)
        is_updated = False
        json_content = json_files_list[index]
        sub_log = pd.DataFrame(columns=[file_name])
        if "mime_type" in json_content.keys():
            mention_text_key = "mention_text"
            text_anchor_key = "text_anchor"
```

```
            text_segment_key = "text_segments"
        else:
            mention_text_key = "mentionText"
            text_anchor_key = "textAnchor"
            text_segment_key = "textSegments"
        if "entities" in json_content.keys():
            for i in
reversed(range(len(json_content['entities']))):
                entity = json_content['entities'][i]
                if mention_text_key not in entity.keys():
                    sub_log =
sub_log.append({file_name:entity['type']},ignore_index = True)
                    del json_content['entities'][i]
                    is_updated = True
                    continue
                else:
                    if 'properties' in
json_content['entities'][i].keys() and
entity[mention_text_key].strip():
                        for j in
range(len(json_content['entities'][i]['properties'])-1,-1,-1):
                            if mention_text_key in
json_content['entities'][i]['properties'][j].keys():
                                if
json_content['entities'][i]['properties'][j][mention_text_key].st
rip() == "":
                                    sub_log =
sub_log.append({file_name:json_content['entities'][i]['properties
']['type']},ignore_index = True)
                                    del
json_content['entities'][i]['properties'][j]
                                    is_updated = True
                                    continue
                            elif mention_text_key not in
json_content['entities'][i]['properties'][j].keys():
```

```
                                sub_log =
sub_log.append({file_name:json_content['entities'][i]['properties
'][j]['type']},ignore_index = True)
                                del
json_content['entities'][i]['properties'][j]
                                is_updated = True
                                continue
                            if text_anchor_key not in
json_content['entities'][i]['properties'][j].keys():
                                sub_log =
sub_log.append({file_name:json_content['entities'][i]['properties
'][j]['type']},ignore_index = True)
                                del json_content['entities'][i]
                                is_updated = True
                                continue
                            elif text_anchor_key in
json_content['entities'][i]['properties'][j].keys():
                                if text_segment_key not in
json_content['entities'][i]['properties'][j][text_anchor_key].key
s():
                                    sub_log =
sub_log.append({file_name:json_content['entities'][i]['properties
'][j]['type']},ignore_index = True)
                                    del
json_content['entities'][i]['properties'][j]
                                    is_updated = True
                                    continue
                                elif
len(json_content['entities'][i]['properties'][j][text_anchor_key]
[text_segment_key]) < 1 :
                                    sub_log =
sub_log.append({file_name:json_content['entities'][i]['properties
'][j]['type']},ignore_index = True)
                                    del
json_content['entities'][i]['properties'][j]
                                    is_updated = True
```

```
                                continue

                    elif not entity[mention_text_key].strip():
                        sub_log =
sub_log.append({file_name:entity['type']},ignore_index = True)
                        del json_content['entities'][i]
                        is_updated = True
                        continue

                if text_anchor_key not in entity.keys():
                    sub_log =
sub_log.append({file_name:entity['type']},ignore_index = True)
                    del json_content['entities'][i]
                    is_updated = True
                    continue
                elif text_anchor_key in entity.keys():
                    if text_segment_key not in
entity[text_anchor_key].keys():
                        sub_log =
sub_log.append({file_name:entity['type']},ignore_index = True)
                        del json_content['entities'][i]
                        is_updated = True
                        continue
                    elif
len(entity[text_anchor_key][text_segment_key]) < 1 :
                        sub_log =
sub_log.append({file_name:entity['type']},ignore_index = True)
                        del json_content['entities'][i]
                        is_updated = True
                        continue
        else:
            print('Entities missing : ',files[index])
        # if is_updated:
        store_blob(json_content,output_file_name)
        if not sub_log.empty:
            logs = pd.concat([logs,sub_log],axis = 1)
```

```
    logs.drop('FileName',axis=1,inplace=True)
    logs.to_csv('output.csv',index = False)
main()
```

Output File

The script deletes all bounding boxes (entities) in the JSON file that do not contain any mentionText or textAnchors, and overwrites the file. The script will also create a CSV file containing a list of deleted entities.

Reference Links

Drive Link to IPYNB File : empty_bounding_box_removal_script.ipynb

Sample CSV output File : empty_entity_output.csv