HITL REJECTED DOCUMENTS TRACKING [External]

Table of Contents

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 is used to track the HITL rejected documents. List of Long Running Operation(LRO) ids are used as an input to show the list of rejected files in HITL for those LROs and also copying the processed json to the GCS folder provided with entity HITL_Status added in the json.

Prerequisite

- Vertex AI Notebook
- List of LROs

Step by Step procedure

Install the latest version of document ai using below command

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

1. **Input Details**

```python
LRO_list=['1945491532426716613','1945491532426716613'] #should be a list
gcs_hitl_rejected_path='gs://XXX/XXX/XX/' # path should end with '/'
project_id='XXXX-XXXX-XXXX'
location='us' # Processor location
```

- `LRO_list:` provide the list of LROs (operation id after batch processing)
- `gcs_hitl_rejected_path:` provide the gcs path to save the json file
- `project_id:` provide the name of the project
- `Location:` provide the location of processor like ‘us’ or ‘eu’

2. **Run the Code**

Run the rest of the code without any edit for CSV file and the modified json to save in the gcs path.

3. **Output**

The output after execution is in 2 formats, the first one is a CSV file (HITL_Status_Update.csv) with File names, HITL status and Reason for the rejection in HITL if rejected.

		File_Name	HITL_Status	Reason
1	0	1.jpg	SUCCEEDED	
2	1	2.png	REJECTED	Too small
3	2	mllb0074_merged.pdf	REJECTED	Too blurry

The second output after execution is the JSON file with the entity HITL_Status added in the DocumentProto json file and saved in the GCS path provided.

Ex: {'type': 'HITL_Status', 'mentionText': 'REJECTED_Too blurry'}

Sample Code

Python

```
# INPUT DETAILS TO BE PROVIDED
LRO_list=['1945491532426716613','1945211532426712334'] #should be a list
gcs_hitl_rejected_path='gs://xxx/xxxx/xxx/' # path should end with '/'
project_id='xxxx-xxxxx-xxxxx'
location='us' # Processor location



# Run the code
from google.cloud import documentai_v1beta3 as documentai
from google.cloud import storage
import pandas as pd
import json
import gcsfs
from google.cloud import storage
fs=gcsfs.GCSFileSystem(project=project_id)
def load_json_files_from_folder_uri(folder_uri,hitl_dict,gcs_output_path):
    """Loads all the JSON files in a GCP folder URI as JSON objects."""
    client = storage.Client()
    uri_parts = folder_uri.split('/')
    bucket_name = uri_parts[2]
    folder_name = '/'.join(uri_parts[3:])
    bucket = client.get_bucket(bucket_name)
```

```
    blobs = bucket.list_blobs(prefix=folder_name)
    for blob in blobs:
        if blob.content_type == 'application/json':
            blob_content = blob.download_as_bytes()
            json_data = json.loads(blob_content.decode('utf-8'))
        if 'entities' in json_data.keys():
            json_data['entities'].append(hitl_dict)
            #print(json_data['entities'])
            file_name=blob.name.split('/')[-1]
            #comment the below line if you dont want the JSON file to be
saved in GCS path

fs.pipe(gcs_output_path+file_name,bytes(json.dumps(json_data,ensure_ascii=Fa
lse),'utf-8'),content_type='application/json')


def dataframe_HITL_status(metadata_op):
    df = pd.DataFrame(columns = ['File_Name', 'HITL_Status', 'Reason'])
    for i in range(len(metadata_op.individual_process_statuses)):
        if 'human_review_operation' in
metadata_op.individual_process_statuses[i].human_review_status:

File_Name=metadata_op.individual_process_statuses[i].input_gcs_source.split(
'/')[-1]

hitl_op_id_1=(metadata_op.individual_process_statuses[i].human_review_status
.human_review_operation).split('/')[-1]

x_hitl_1=client.get_operation({"name":f"projects/{project_id}/locations/{loc
ation}/operations/{hitl_op_id_1}"})

hitl_status_1=documentai.ReviewDocumentResponse.deserialize(x_hitl_1.respons
e.value)
            hitl_status_1_state=""
            Reason_1=""
            if hitl_status_1.state.name=='REJECTED':
                hitl_status_1_state=hitl_status_1.state.name
                Reason_1=hitl_status_1.rejection_reason
            elif hitl_status_1.state.name=='SUCCEEDED':
```

```
                hitl_status_1_state=hitl_status_1.state.name
                Reason_1=" "
            df=pd.concat([df, pd.DataFrame({'File_Name' : File_Name,
'HITL_Status' : hitl_status_1_state, 'Reason' : Reason_1}, index=[0])],
ignore_index = True)
    return df

df_merge = pd.DataFrame()
for i in LRO_list:
    opts = {}
    if location == "eu":
        opts = {"api_endpoint": "eu-documentai.googleapis.com"}
    elif location == "us":
        opts = {"api_endpoint": "us-documentai.googleapis.com"}
        #opts = {"api_endpoint":
"us-autopush-documentai.sandbox.googleapis.com"}
    client = documentai.DocumentProcessorServiceClient(client_options=opts)

x=client.get_operation({"name":f"projects/{project_id}/locations/{location}/
operations/{i}"})

metadata_op=documentai.BatchProcessMetadata.deserialize(x.metadata.value)
    hitl_file_dict={}
    for i in range(len(metadata_op.individual_process_statuses)):
        if 'human_review_operation' in
metadata_op.individual_process_statuses[i].human_review_status:
            pre_hitl_file=
metadata_op.individual_process_statuses[i].output_gcs_destination
            post_hitl_file=
(metadata_op.individual_process_statuses[i].human_review_status.human_review
_operation).split('/')[-1]
            hitl_file_dict[pre_hitl_file]=post_hitl_file
        else:
            continue

    for processed_path,hitl_path in hitl_file_dict.items():
        hitl_op_id=hitl_path.split('/')[-1]
        #print(hitl_op_id)
```

```
x_hitl=client.get_operation({"name":f"projects/{project_id}/locations/{locat
ion}/operations/{hitl_op_id}"})

hitl_status=documentai.ReviewDocumentResponse.deserialize(x_hitl.response.va
lue)
        HITL_dict={}
        if hitl_status.state.name=='REJECTED':
            HITL_dict['type']='HITL_Status'
            HITL_dict['mentionText']=hitl_status.state.name +'_'+
hitl_status.rejection_reason
            #print(processed_path)
            #print(HITL_dict)

load_json_files_from_folder_uri(processed_path,HITL_dict,gcs_hitl_rejected_p
ath)
        else:
            pass


    df=dataframe_HITL_status(metadata_op)
    df_merge=pd.concat([df_merge, df], ignore_index=True)

df_merge.to_csv("HITL_Status_Update.csv")
```

HITL Rejected Document Tracking for Single Doc / Sync

def get_hitl_lro_status(project_id, location, operation_id):

```python
def get_hitl_lro_status(project_id, location, operation_id):
    '''
    Checks the status of the operation_id, if the status is done and the
HITL state is REJECTED,
    then it returns the status and an entity with type="HITL_Status",
mention_text="REJECTED_rejected_reason"
    '''
    if location == "eu":
        opts = {"api_endpoint": "eu-documentai.googleapis.com"}
    elif location == "us":
        opts = {"api_endpoint": "us-documentai.googleapis.com"}
    # DocumentAI Client Instance
    client = documentai.DocumentProcessorServiceClient(client_options=opts)
    # An empty Entity
    HITL_entity=documentai.Document.Entity();

    # Get the status of the operation_id

hitl_status_response=client.get_operation({"name":f"projects/{project_id}/lo
cations/{location}/operations/{operation_id}"})
    if(hitl_status_response.done):
        # get the response from HITL console

hitl_response=documentai.ReviewDocumentResponse.deserialize(hitl_status_resp
onse.response.value)
        hitl_status=hitl_response.state.name
        if hitl_status=='REJECTED':
            HITL_entity.type='HITL_Status'
            HITL_entity.mention_text=hitl_response.state.name +'_'+
hitl_response.rejection_reason
            return hitl_status,HITL_entity # returns the hitl_status and a
dictionary with 'type':'HITL_status',
'mentionText':'REJECTED_rejected_reason'
        else:
            return hitl_status,HITL_entity # returns the hitl_status and an
empty dict
```

```
    else:
        # Returns and an empty dictionary if HITL is in progress
        return
documentai.ReviewDocumentOperationMetadata.deserialize(hitl_status_response.
metadata.value).common_metadata.state.name,HITL_entity
```