Key/Value Pair Entity Conversion (External)

Table of Contents

Disclaimer **1**
Objective **1**
Step by Step procedure **2**
1. Config file Creation 2
2. Input Details 2
3. Run the Code 3
4. Output 3
Sample Code 4

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 uses Form parser JSON files (Parsed from a processor) from the GCS bucket as input, converts the key/value pair to the entities and stores it to the GCS bucket as JSON files.

Prerequisite

- Vertex AI Notebook
- Form parser Json files in GCS Folders

Step by Step procedure

1. **Config file Creation**
   Run the below code and create a config.ini file for providing input.

```
import configparser
config = configparser.ConfigParser()
config_path= "config.ini" #Enter the path of config file
# Add the structure to the file we will create
config.add_section('Entities_synonyms')
config.set('Entities_synonyms', 'entity1', 'key_synonym1,
key_synonym2, key_synonym3')
config.set('Entities_synonyms', 'entity2', 'key_synonym1,
key_synonym2, key_synonym3')
# Write the new structure to the new file
with open(config_path, 'w') as configfile:
    config.write(configfile)
```

2. **Input Details**

   Once ***config.ini*** file is created with the above step , enter the input in the config file :

   `entity1` = key_synonym1, key_synonym2, key_synonym3

```
[Entities_synonyms]
entity1 = key_synonym1, key_synonym2, key_synonym3
entity2 = key_synonym1, key_synonym2, key_synonym3
```

   Here add the entity name in place of `entity1` and add the synonyms related to the entity in place of key_synonym separated by comma(,). Add multiple entities with their synonyms in the next line.

   Eg :

   Address = AddressName, AddressName1, AddressLine
   InvoiceNumber = Invoice,InvoiceNo
   PaymentDate = SNC, SNCs, SNC1

3. **Run the Code**

a. Copy the code provided in this document, Enter the path of Config file

```
import configparser
config = configparser.ConfigParser()
config_path= "configfile.ini" #Enter the path of config file
# Add the structure to the file we will create
config.add_section('Entities_synonyms')
config.set('Entities_synonyms', 'entity1', 'key_synonym1, key_synonym2, key_synonym3')
config.set('Entities_synonyms', 'entity2', 'key_synonym1, key_synonym2, key_synonym3')
# Write the new structure to the new file
with open(config_path, 'w') as configfile:
    config.write(configfile)
```

Update the config file path

```
#importing necessary modules
import gcsfs
import json
from pathlib import Path
```

b. Update the project id, form parser output path, GCS bucket name and the GCP output for the labeled entities Jsons.

```
#importing necessary modules
import gcsfs
import json
from pathlib import Path
from google.cloud import storage
from io import BytesIO
from google.cloud import documentai_v1beta3 as documentai
import re
import tqdm

fileSystem =gcsfs.GCSFileSystem(project='[redacted]') # your project id
formparser_path = "[redacted]" # path of the form parser output
bucket_name = "[redacted]" # bucket name
output_path = "[redacted]" # [illegible] script

config = configparser.ConfigParser()
config.optionxform = str
config.read(config_path)
```

project ID

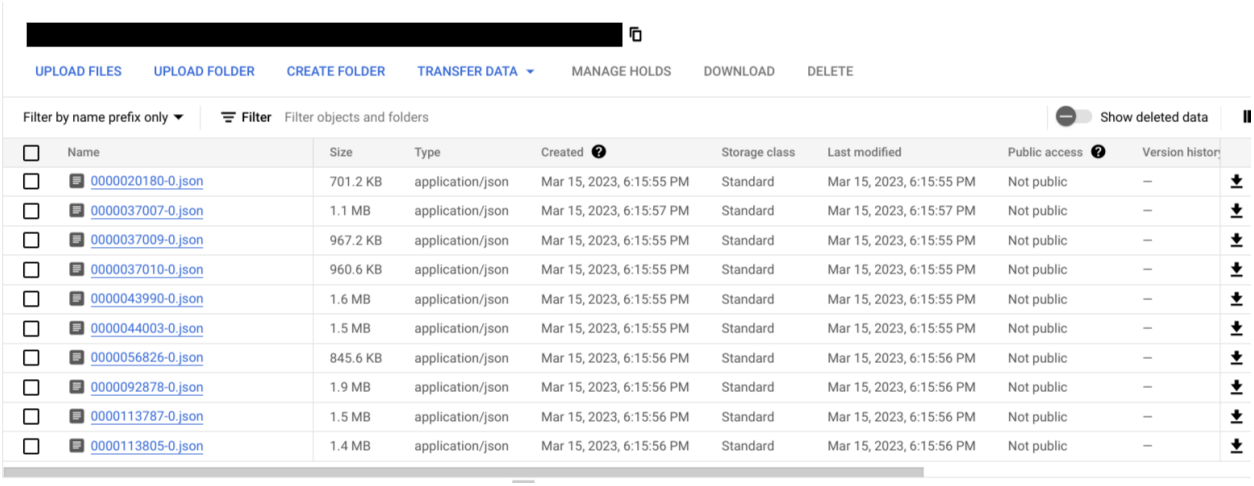
GCP path of form parser output

GCP bucket name

GCS bucket output path for the script

4. Output

We get the converted Json in the GCS path which is provided in the script with the variable name ***output_path .***



Name	Size	Type	Created	Storage class	Last modified	Public access	Version histor
0000020180-0.json	701.2 KB	application/json	Mar 15, 2023, 6:15:55 PM	Standard	Mar 15, 2023, 6:15:55 PM	Not public	–
0000037007-0.json	1.1 MB	application/json	Mar 15, 2023, 6:15:57 PM	Standard	Mar 15, 2023, 6:15:57 PM	Not public	–
0000037009-0.json	967.2 KB	application/json	Mar 15, 2023, 6:15:55 PM	Standard	Mar 15, 2023, 6:15:55 PM	Not public	–
0000037010-0.json	960.6 KB	application/json	Mar 15, 2023, 6:15:55 PM	Standard	Mar 15, 2023, 6:15:55 PM	Not public	–
0000043990-0.json	1.6 MB	application/json	Mar 15, 2023, 6:15:55 PM	Standard	Mar 15, 2023, 6:15:55 PM	Not public	–
0000044003-0.json	1.5 MB	application/json	Mar 15, 2023, 6:15:55 PM	Standard	Mar 15, 2023, 6:15:55 PM	Not public	–
0000056826-0.json	845.6 KB	application/json	Mar 15, 2023, 6:15:56 PM	Standard	Mar 15, 2023, 6:15:56 PM	Not public	–
0000092878-0.json	1.9 MB	application/json	Mar 15, 2023, 6:15:56 PM	Standard	Mar 15, 2023, 6:15:56 PM	Not public	–
0000113787-0.json	1.5 MB	application/json	Mar 15, 2023, 6:15:56 PM	Standard	Mar 15, 2023, 6:15:56 PM	Not public	–
0000113805-0.json	1.4 MB	application/json	Mar 15, 2023, 6:15:56 PM	Standard	Mar 15, 2023, 6:15:56 PM	Not public	–

Sample Code

```
#importing necessary modules
import gcsfs
import json
import google.auth
from pathlib import Path
from google.cloud import storage
from io import BytesIO
from google.cloud import documentai_v1beta3 as documentai
import re
from tqdm import tqdm


PROJECT_ID = “XXXX-XXXX-XXXX” # your project id
bucket_name = "ZZZZ-ZZZZ" # bucket name

credentials, _ = google.auth.default()
fileSystem =gcsfs.GCSFileSystem(project=PROJECT_ID, token=credentials)
formparser_path = "kv_entites_conversion/test_script" # path of the
form parser output
```

```
output_path = "kv_entites_conversion/test_script/output" # output
path for this script
config_path = "/path/to/config.ini
config = configparser.ConfigParser()
config.optionxform = str
config.read(config_path)


def get_file(file_path : str):
    """
    To read files from cloud storage.
    """
    file_object = json.loads(fileSystem.cat(file_path))
    return file_object

def store_blob(document,file_name : str):
    """
    Store files in cloud storage.
    """
    storage_client = storage.Client()
    process_result_bucket = storage_client.get_bucket(bucket_name)
    document_blob = storage.Blob(
        name=str(Path(output_path,file_name)),
        bucket=process_result_bucket)
    document_blob.upload_from_string(json.dumps(document),
content_type="application/json")
    # print(f"File Saved : {file_name}.")

def entity_synonyms(old_entity : str):
    """
    To check for any synonyms for the entites and replace.
    """
    entities_synonyms = config.items('Entities_synonyms')
    for item in entities_synonyms:
        synonym_list = [i.lower().strip() for i in
item[1].split(",")]
        if old_entity.lower() in synonym_list:
            return item[0]
```

```
    #if entity does not match with any synonyms, will return entity
as it is.
    return ""


def entity_data(formField_data : dict,page_number : int):
    """
    Function to create entity objects with some cleaning.
    """
    #Cleaning the entity name
    key_name =
re.sub('[^\w\s]',"",formField_data['fieldName']['textAnchor']['conten
t']).replace(" ","").strip()
    #checking for entity synonyms
    key_name = entity_synonyms(key_name)
    if key_name:
        entity_dict = {
        "confidence" :  formField_data['fieldValue']['confidence'],
        "mentionText" :
formField_data['fieldValue']['textAnchor']['content'],
        "pageAnchor" :{
"pageRefs":[{"boundingPoly":formField_data['fieldValue']['boundingPol
y'], "page":page_number}]},
        "textAnchor" : formField_data['fieldValue']['textAnchor'],
        "type" : key_name
        }

        return entity_dict
    else:
        return {}


def convert_kv_entities(file : str):
    """
    Function to convert form parser key value to entities.
    """
    #get the file object
```

```
    file = get_file(file)
    #initializing entities list
    file['entities'] = []

    for page_number, page_data in enumerate(file["pages"]):
        for formField_number,formField_data in
enumerate(page_data.get('formFields', [])):

            #get the element and push it to the entities array
            entity_obj = entity_data(formField_data, page_number)
            if entity_obj:
                file['entities'].append(entity_obj)

    #removing the form parser data
    for i in range(len(file['pages'])) :
        if "formFields" in file['pages'][i].keys():
            del file['pages'][i]['formFields']

        if "tables" in file['pages'][i].keys():
            del file['pages'][i]['tables']

    return file

def main():
    """
    Main function to call helper functions
    """
    # fetching all the files
    files=[i for i in fileSystem.ls(bucket_name+ '/' +
formparser_path) if i.endswith('.json')]
    for file in tqdm.tqdm(files,desc ="Status : "):

        # converting key value to entites
        entity_json = convert_kv_entities(file)

        #storing the json
        file_name = file.split('/')[-1]
        store_blob(entity_json,file_name)
```

```
#calling main function
main()
```