TASK 3 - Development History

Regarding to divide our workload, each group member will incharge each task in this assignment, and if there any challenge happens, one will support another. Here is the information about student main incharge:

2. Task 2: Wei Yu Su

1. Task 1: Anh Huy Phung

data.

Task 1

Today, we went through the assignment specification, and started exploring task 1

Date: 05/04/2023

Contribution:

Group member 1:

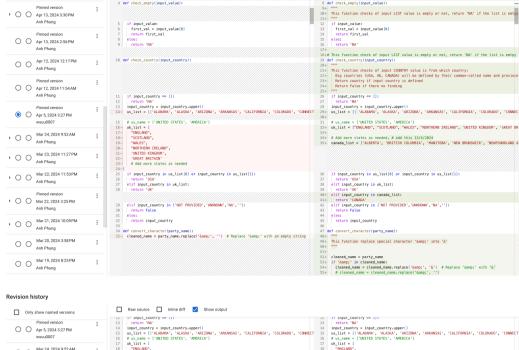
formation under such tagnames.

Group member 2: • understood the data structure and figure it out duplicate in property tagname

• understood the data structure under tagename and develop functions extract

proof Revision history

Apr 13, 2024 4:03 PM



```
Mar 23, 2024 11:27 PM
Anh Phung
Pinned version

Mar 22, 2024 3:25 PM

And Physics
Mar 21, 2024 10:09 PM
Anh Phung
  O O Mar 20, 2024 3:58 PM Anh Phung
 Mar 18, 2024 2:24 PM
Anh Phung
 Mar 14, 2024 11:45 PM
Anh Phung
  Pinned version

Mar 14, 2024 11:45AM
Anh Phung
       properties>
                   roperty>
                               <serial-no>75418582</serial-no>
```

```
roperty>
             <serial-no>75418582</serial-no>
             <registration-no>2232041</registration-no>
        </property>
Date 13/04/2024
Today, we started reading the data by python and extracted the key fields we
needed. We developed some functions to identify party's name and country.
Contribution:
```

<registration-no>2232041</registration-no>

· Define how to classify party name belong to which entity (individual or organization)

Group member 1:

Group member 2: Define how to classify country of assignee and assignor, base on related

attributes ('nationality', 'country-name', 'state', 'city')

Regular expression pattern to match individual-related terms
indi_pattern = r'\b(?:individual)\b'
title_pattern = r'\b(?:Mr\.|Mrs\.|Ms\.|Dr\.|Prof\.|Rev\.|Sir|Madam)\b' indi_pattern = r'\b(?:Indixasue...

title_pattern = r'\b(?:Mr\.|Mrs\.|Ms\.|Dr\.|Prof\.|Rev\.|party...

party_name, entity_value = check_empty(party_name), check_empty(entity_value)

party_name, entity_value = check_empty(party_name), check_empty(entity_value)

if re.search(indi_pattern, entity_value, re.IONORECASE):
indi_info = re.search(indi_pattern, party_name, re.IGNORECASE)[0]
party_name = remove_person_itiles(party_name)
return party_name, entity_value

of country classification.

Proof

Remove titles as 2 forms, take Mr as example and this applies for other tit 76+ This function replaces title at beginning and end of name pattern_titles_comma = r',\s=(?:Mr\, |Mrs\, |Miss\, |Ms\, |Mx\, |Sir\, |Dame\, |Dr\, |Cllr\, |Llady\, attern_titles = r'(?:Mr\, |Miss\, |Ms\, |Mx\, |Sir\, |Dame\, |Dr\, |Cllr\, |Llady\, attern_titles = r'(?:Mr\, |Miss\, |Ms\, |M

First picture is the development of classify personal's title, second is development

78+ - Check if any organization pattern exists in the name.
71+ - If yes, keep the name as it is.
72+ - Otherwise, consider it an individual and remove titles.
73 | ""
74 def remove_person_titles(text_input):
75+ | ""

cleaned_text_titles = re.sub(pattern_titles_comma, '', text_input, flags=re.IGNORI
cleaned_text = re.sub(pattern_titles, '', cleaned_text_titles, flags=re.IGNORECASI

91
92+# Regular expression pattern to match organization-related terms
93+org_pattern = r'\b(?:ltd|llp|llc|BV|B\.V\.|company|corporation|cor|bank|banking|inc 95+# Clean party_name and entity_value 96 party_name, entity_value = convert_character(party_name), convert_character(entity_value)

88 # Regular expression pattern to match individual-related terms
89 indi_pattern = r'\b{?:individual\\b'
90 title_pattern = r'\b{?:Mr\.|Mrs\.|Mrs\.|Dr\.|Prof\.|Rev\.|Sir|Madam\\b'
91

```
| 102 | 103+# If entity_value cannot be defined entity is individual or organization | 104+# # search for organization pattern(characters or dba attribute) to classify organization | 104+# # search for organization pattern(characters or dba attribute) to classify organization | 105+# # if these signals can't be found then treat party,name as individual | 105+# if these signals can't be found then treat party,name as individual | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit entity_value.upper() in ['NOT PROVIDED', 'UNRONOM', 'NA', '']; | 106-bit
                                                                                                                                                                                               7+# This function check of input LIST value is empty or not, return 'NA' if the list :
8 def check_empty(input_value):
       4 def check_empty(input_value):
       5  if input_value:
6   first_val = input_value[0]
7   return first_val
8   else:
9   return 'NA'
                                                                                                                                                                                              18+# This function check of input LIST value is empty or not, return 'NA' if the list : 19 def check_country(input_country):
     10 def check_country(input_country):
              if input_country == []:
                                                                                                                                                                                                       input_country = input_country.upper()
us_list = [[ 'ALABANA', 'ALASKA', 'ARIZONA', 'ARKANSAS', 'CALIFORNIA', 'COLORADO',
               input_country = input_country.upper()
us_list = [[ 'ALABAMA', 'ALASKA', 'ARIZONA', 'ARKANSAS', 'CALIFORNIA', 'COLORADO'
                                                                                                                                                                                                     # Add more states as needed, # add this 13/4/2024
canada_list = ['ALBERTA', 'BRITISH COLUMBIA', 'MANITOBA', 'NEW BRUNSWICK', 'NEMFOL
if input_country in us_list[0] or input_country in us_list[1]:
    return 'Ust
elif input_country in uk_list:
    return 'Ust

            19 # Add more states as nee
canada_List = 'NABERTA
21 if input_country in us_
22 return 'USA'
23 elif input_country in ul
24 return 'UK'
25 elif input_country in c
26 return 'CAMADA'
27 elif input_country in [
28 return False
29 else:
30 return input_country
31
             return 'USA'
elif input_country in uk_list:
    return 'UK'
elif input_country in canada_list:
                                                                                                                                                                                                      return 'UK'
elif input_country in canada_list:
                                                                                                                                                                                                     return 'CANADA'
elif input_country in ['NOT PROVIDED','UNKNOWN','NA','']:
return False
            elif input_country in ['NOT PROVIDED','UNKNOWN','NA','']:
return False
     31
32 def convert_character(party_name):
                                                                                                                                                                                             47 def convert_character(party_name):
    33 cleaned_name = party_name
                                                                                                                                                                                                      cleaned_name = party_name
Date 18/04/2024
Today, we started review all of the functions and generated file submission file.
Contribution:
Group member 1:
                  Review all of the generated functions, give example in each function for easier
                    documentation
Group member 2:

    Documentation

Proof
These are pictures to show our explaination to some of funtions and examples.
```

cleaned_name = party_name if '6mmg;' in cleaned_name: cleaned_name = cleaned_name.replace('6amp;', '6') cleaned_name = cleaned_name.replace('6amp;', '6') def fagname_info(tag_name, text_info, get_detail_info = True):

<assignment>\n

def property_converted(text_info):

print('rf_id: ', assignment_info)

pattern = tagname_pattern(tag_name)
matches = re.findalt[pattern, text_info, re.DOTALL)
if get_detal_info:
apply_functions = lambda x: convert_character(check_empty(x))
return apply_functions(matches).strip()

<reel-no>4656</reel-no>\n

This code extract all raw information under assignment—entry tagname without transformation assignment_info = rf_id_converted('reel-no', 'frame-no', raw_info)

try:
data=json.loads(content)
except:
raise TypeError("Invalid json: unable to load data!")

studentcols.append(each)

Please input your group number:21 Task 1 json file passed!

except valueerror:
print('Invalid date input or input not exist, try again')

Returns the pattern for input tagna return rf'<{tag_name}>(.*?)</{tag_name}> def check_empty(input_value):

Finally we create our final submission file.

Checks of input list value is empty or not, return 'NA' if the list is emtpy

This function gets all information within the tagname from a xml-formated text, decided by get_detail_info parameter:
- get_detail_info is Frue then get all the within detail and transform it via check_empty function
- get_detail_info is Faits then just return all the pattern-matches in text_info is Faits then just return all the pattern-matches in text_info

[] # This code extract all rew information under assignment-entry taggame without transformation assignment_infor = tagmame_info('assignment_info,' raw_info, False) printiassignment_info ('assignment_info,' raw_info, False) # This code extract all raw information under assignment-entry tagname with transformation (check empty, convert special letters and strip) real_info = tagname_info('real-no', raw_info, True) print(real_info)

This function extracts all serial under property tag name, eliminates the duplicated ones and returns total properties in the assignment split_data = tagname_info('property', text_info, False)
for element in split_data:
 patterm_serial = r'<serial-noo(.*?)</pre>
serial = re.findallipatterm_serial, element, re.DOTALL)
if check_empty(serial) = 'M':

<last-update-date>20111108</last-update-date>\n

<frame-no>0341</frame-no>\n

This code extract all raw information under assignment-entry tagname with transformation (check empty, convert special letters and strip) last_update_date = date_converted('last_update_date', raw_info) print('last_update_date', last_update_date') # This code also extract all raw information under conveyance-text tagname with transformation (check empty, convert special letters and strip) conveyance_text = tagname_info'(conveyance-text', raw_info, True) print('conveyanc_text') conveyance_text'. # We can extract correspondent_party information under correspondent_party tagname with transformation (check empty, convert special letters and strip)
Although we have many person-or-organization-name tagnames but the tagname_info function catch the first one, which is the attribute of correspondent_party
correspondent_party = tagname_info('person-or-organization-name', raw_info, True)
print('correspondent_party'; correspondent_party) # This code extract all raw information under assignment-entry tagname without transformation property_information = property_converted(raw_info) print(property-count: , property_information) rf_id: 46560341
last_update_date: 2011-11-08
conveyance_text: ASSIGNS THE ENTIRE INTEREST
correspondent_party: JENNIFER E. LACROIX
property-count: 2 5.2. Generate final file We will create our file json file by extracting our group xml text's file (number 21) from google drive [] group_2l_input_path = '/content/drive/Shareddrives/FIT5196_51_2824/A1/Students data/Task 1/Group@2l.txt' group_2l_output_path = '/content/drive/Mydrive/Taskl_2l.json' json_file(group_2l.input_path, group_2l.output_path, group_2l.output_pat JSON file saved successfully at /content/drive/MyDrive/task1_21.json [] import json import ands as pd import ands as pd with open("/content/drive/MyOrive/taskl_21.json".format('21'.zfill(3)),"r") as file: content=file.read()

Google Colab Workbook Link

Task 1