01_data_collection_and_preprocess

March 16, 2025

1 Clinical Trials

1.0.1 Data Collection

[ ]:
```python
def remove_description_module(data):
    # Count the number of outer lists
    num_outer_lists = len(data)
    # print(f"Number of outer lists: {num_outer_lists}")

    # Traverse the outer list
    for i, outer_item in enumerate(data):
        # print(f"Processing outer list {i+1}/{num_outer_lists}")

        # Ensure it's a list before iterating
        if isinstance(outer_item, list):
            for j, inner_item in enumerate(outer_item):
                # print(f"Processing inner list {j+1}/{len(outer_item)} in␣
↪outer list {i+1}")

                # Ensure the structure matches expected format
                if isinstance(inner_item, dict) and "protocolSection" in␣
↪inner_item:
                    if "descriptionModule" in inner_item["protocolSection"]:
                        del inner_item["protocolSection"]["descriptionModule"]
                        # print(f"Removed 'descriptionModule' from inner list␣
↪{j+1} in outer list {i+1}")

    return data
```

[ ]:
```python
import requests
import pandas as pd
import json
import time

diseases = [
    "pancreatic cancer",
    "chagas disease",
    "endometriosis",
```

```
    "drug resistant tuberculosis",
    "duchenne muscular dystrophy",
    "alzheimer's",
    "influenza",
    "hepatitis",
    "malaria",
    "breast-cancer",
]


# disease_to_consider = diseases[1]

for disease_to_consider in diseases:

    # Initial URL for the first API call
    base_url = "https://clinicaltrials.gov/api/v2/studies"
    params = {
        "query.titles": disease_to_consider,
        "pageSize": 100
    }

    # Initialize an empty list to store the data
    data_list = []
    all_data = []


    # Loop until there is no nextPageToken
    x=1
    while True:
        if x>50:
            break

        x = x+1
        # Print the current URL (for debugging purposes)
        print("Fetching data from:", base_url + '?' + '&'.join([f"{k}={v}" for␣
 ↪k, v in params.items()]))

        # Send a GET request to the API
        response = requests.get(base_url, params=params)


        # Check if the request was successful
        if response.status_code == 200:
            data = response.json()  # Parse JSON response
            studies = data.get('studies', [])  # Extract the list of studies
            all_data.append(studies)
            # print(data)
```

```
            # Check for nextPageToken and update the params or break the loop
            nextPageToken = data.get('nextPageToken')
            if nextPageToken:
                params['pageToken'] = nextPageToken  # Set the pageToken for␣
 ↪the next request
            else:
                break  # Exit the loop if no nextPageToken is present
        else:
            print("Failed to fetch data. Status code:", response.status_code)
            break
        print(len(all_data))

        time.sleep(10)

    # Cleaning the collected data
    all_data = remove_description_module(all_data)

    print(all_data)

    with open("data/CT_"+disease_to_consider+".json", "w") as json_file:
        json.dump(all_data, json_file)
```

1.0.2 Data Pre-processing for RARE Diseases

[1]:
```
import json
import pandas as pd
import os

# Function to read a JSON file
def read_json_file(file_path):
    with open(file_path, "r", encoding="utf-8") as file:
        data = json.load(file)
    return data

diseases = [
    "CT_pancreatic cancer",
    "CT_chagas disease",
    "CT_endometriosis",
    "CT_drug resistant tuberculosis",
    "CT_duchenne muscular dystrophy",
]


master_df = df = pd.DataFrame()

for disease_to_consider in diseases:
```

```
    # Example usage
    file_path = "data/"+disease_to_consider+".json"
    studies = read_json_file(file_path)
    data_list = []



    for batch_study in studies:
        for study in batch_study:
                # Safely access nested keys
                nctId = study['protocolSection']['identificationModule'].
↪get('nctId', 'Unknown')
                overallStatus = study['protocolSection']['statusModule'].
↪get('overallStatus', 'Unknown')
                startDate = study['protocolSection']['statusModule'].
↪get('startDateStruct', {}).get('date', 'Unknown Date')
                try:
                    conditions = ', '.
↪join(study['protocolSection']['conditionsModule'].get('conditions', ['No␣
↪conditions listed']))
                except:
                    conditions = "No conditions listed"

                acronym = study['protocolSection']['identificationModule'].
↪get('acronym', 'Unknown')

                # Extract interventions safely
                interventions_list = study['protocolSection'].
↪get('armsInterventionsModule', {}).get('interventions', [])
                interventions = ', '.join([intervention.get('name', 'No␣
↪intervention name listed') for intervention in interventions_list]) if␣
↪interventions_list else "No interventions listed"

                # Extract locations safely
                locations_list = study['protocolSection'].
↪get('contactsLocationsModule', {}).get('locations', [])
                locations = ', '.join([f"{location.get('city', 'No City')} -␣
↪{location.get('country', 'No Country')}" for location in locations_list]) if␣
↪locations_list else "No locations listed"

                # Extract dates and phases
                primaryCompletionDate =␣
↪study['protocolSection']['statusModule'].get('primaryCompletionDateStruct',␣
↪{}).get('date', 'Unknown Date')
                studyFirstPostDate = study['protocolSection']['statusModule'].
↪get('studyFirstPostDateStruct', {}).get('date', 'Unknown Date')
```

```
                lastUpdatePostDate = study['protocolSection']['statusModule'].
↪get('lastUpdatePostDateStruct', {}).get('date', 'Unknown Date')
                studyType = study['protocolSection']['designModule'].
↪get('studyType', 'Unknown')
                phases = ', '.join(study['protocolSection']['designModule'].
↪get('phases', ['Not Available']))
                phases = phases.split(",")[-1].strip()
                if(phases == "EARLY_PHASE1"):
                     phases = "PHASE1"

                # phases = get_highest_phase(study)
                lead_sponsor_name =␣
↪study["protocolSection"]["sponsorCollaboratorsModule"]["leadSponsor"].
↪get('name', 'Unknown Sponsor')
                lead_sponsor_type =␣
↪study["protocolSection"]["sponsorCollaboratorsModule"]["leadSponsor"].
↪get('class', 'Unknown Sponsor Type')

                disease_to_consider = disease_to_consider.
↪replace("_clinical_trials", "")

                # Append the data to the list as a dictionary
                data_list.append({
                    "NCT ID": nctId,
                    "Acronym": acronym,
                    "Overall Status": overallStatus,
                    "Start Date": startDate,
                    "Conditions": conditions,
                    "Interventions": interventions,
                    "Locations": locations,
                    "Primary Completion Date": primaryCompletionDate,
                    "Study First Post Date": studyFirstPostDate,
                    "Last Update Post Date": lastUpdatePostDate,
                    "Study Type": studyType,
                    "Phases": phases,
                    "Sponsor": lead_sponsor_name,
                    "Sponsor Type": lead_sponsor_type,
                    "Disease": disease_to_consider.replace(" ", "_")
                })

    # print(disease_to_consider)
    print(f"Found {len(data_list)} records for {disease_to_consider}")
    df = pd.DataFrame(data_list)

    # print(df)
    # print(len(df))
```

```
    os.makedirs("data/", exist_ok=True)
    # df.to_csv("data/"+disease_to_consider.replace(" ", "_")+"_procesed.csv",␣
 ↪index=False)
    master_df = pd.concat([master_df, df], ignore_index=True)
master_df.to_csv("data/CT_all_rare_disease_procesed.csv", index=False)
print("Total records processed for RARE Disease", len(master_df))
```

```
Found 2580 records for CT_pancreatic cancer
Found 55 records for CT_chagas disease
Found 589 records for CT_endometriosis
Found 60 records for CT_drug resistant tuberculosis
Found 364 records for CT_duchenne muscular dystrophy
Total records processed for RARE Disease 3648
```

1.0.3 Data Pre-processing for COMMON Diseases

[2]:
```
import json
import pandas as pd
import os

# Function to read a JSON file
def read_json_file(file_path):
    with open(file_path, "r", encoding="utf-8") as file:
        data = json.load(file)
    return data


diseases = [
    "CT_alzheimer's",
    "CT_influenza",
    "CT_breast-cancer",
    "CT_hepatitis",
    "CT_malaria",
]

master_df = df = pd.DataFrame()

for disease_to_consider in diseases:
    # Example usage
    file_path = "data/"+disease_to_consider+".json"
    studies = read_json_file(file_path)
    data_list = []



    for batch_study in studies:
        for study in batch_study:
```

```
                # Safely access nested keys
                nctId = study['protocolSection']['identificationModule'].
↪get('nctId', 'Unknown')
                overallStatus = study['protocolSection']['statusModule'].
↪get('overallStatus', 'Unknown')
                startDate = study['protocolSection']['statusModule'].
↪get('startDateStruct', {}).get('date', 'Unknown Date')
                try:
                    conditions = ', '.
↪join(study['protocolSection']['conditionsModule'].get('conditions', ['No␣
↪conditions listed']))
                except:
                    conditions = "No conditions listed"

                acronym = study['protocolSection']['identificationModule'].
↪get('acronym', 'Unknown')

                # Extract interventions safely
                interventions_list = study['protocolSection'].
↪get('armsInterventionsModule', {}).get('interventions', [])
                interventions = ', '.join([intervention.get('name', 'No␣
↪intervention name listed') for intervention in interventions_list]) if␣
↪interventions_list else "No interventions listed"

                # Extract locations safely
                locations_list = study['protocolSection'].
↪get('contactsLocationsModule', {}).get('locations', [])
                locations = ', '.join([f"{location.get('city', 'No City')} -␣
↪{location.get('country', 'No Country')}" for location in locations_list]) if␣
↪locations_list else "No locations listed"

                # Extract dates and phases
                primaryCompletionDate =␣
↪study['protocolSection']['statusModule'].get('primaryCompletionDateStruct',␣
↪{}).get('date', 'Unknown Date')
                studyFirstPostDate = study['protocolSection']['statusModule'].
↪get('studyFirstPostDateStruct', {}).get('date', 'Unknown Date')
                lastUpdatePostDate = study['protocolSection']['statusModule'].
↪get('lastUpdatePostDateStruct', {}).get('date', 'Unknown Date')
                studyType = study['protocolSection']['designModule'].
↪get('studyType', 'Unknown')
                phases = ', '.join(study['protocolSection']['designModule'].
↪get('phases', ['Not Available']))
                phases = phases.split(",")[-1].strip()
                if(phases == "EARLY_PHASE1"):
                    phases = "PHASE1"
```

```
                # phases = get_highest_phase(study)
                lead_sponsor_name =␣
↪study["protocolSection"]["sponsorCollaboratorsModule"]["leadSponsor"].
↪get('name', 'Unknown Sponsor')
                lead_sponsor_type =␣
↪study["protocolSection"]["sponsorCollaboratorsModule"]["leadSponsor"].
↪get('class', 'Unknown Sponsor Type')

                disease_to_consider = disease_to_consider.
↪replace("_clinical_trials", "")

                # Append the data to the list as a dictionary
                data_list.append({
                    "NCT ID": nctId,
                    "Acronym": acronym,
                    "Overall Status": overallStatus,
                    "Start Date": startDate,
                    "Conditions": conditions,
                    "Interventions": interventions,
                    "Locations": locations,
                    "Primary Completion Date": primaryCompletionDate,
                    "Study First Post Date": studyFirstPostDate,
                    "Last Update Post Date": lastUpdatePostDate,
                    "Study Type": studyType,
                    "Phases": phases,
                    "Sponsor": lead_sponsor_name,
                    "Sponsor Type": lead_sponsor_type,
                    "Disease": disease_to_consider.replace(" ", "_")
                })

    # print(disease_to_consider)
    print(f"Found {len(data_list)} records for {disease_to_consider}")
    df = pd.DataFrame(data_list)

    # print(df)
    # print(len(df))
    os.makedirs("data/", exist_ok=True)
    # df.to_csv("data/"+disease_to_consider.replace(" ", "_")+"_procesed.csv",␣
↪index=False)
    master_df = pd.concat([master_df, df], ignore_index=True)
master_df.to_csv("data/CT_all_common_disease_processed.csv", index=False)
print("Total records processed for COMMON Disease", len(master_df))
```

```
Found 2169 records for CT_alzheimer's
Found 2303 records for CT_influenza
Found 5000 records for CT_breast-cancer
```

```
Found 3517 records for CT_hepatitis
Found 1129 records for CT_malaria
Total records processed for COMMON Disease 14118
```

2 Pubmed

2.0.1 Data Collection

[ ]:
```
from pymed import PubMed
import pandas as pd

diseases = [
    "pancreatic cancer",
    "chagas disease",
    "endometriosis",
    "drug resistant tuberculosis",
    "duchenne muscular dystrophy",
    "influenza",
    "breast-cancer",
    "hepatitis",
    "malaria",
    "alzheimer"
]
email = 'xxxxxxxx@gmail.com'
pubmed = PubMed(tool="PubMedSearcher", email=email)
print("\nPubMed Clinical Trials Articles:")

for disease in diseases:
    disease_to_consider = disease
    search_term = "alzheimer's+clinical+trials"
    results = pubmed.query(search_term, max_results=10000)

    articles_data = []

    for article in results:
        article_dict = article.toDict()
        raw_authors = article_dict.get("authors", [])
        authors = ", ".join([a["name"] if isinstance(a, dict) and "name" in a␣
 ↪else str(a) for a in raw_authors]) if raw_authors else None

        articles_data.append({
            "pubmed_id": article_dict.get("pubmed_id", "").partition('\n')[0],
            "title": article_dict.get("title", None),
            "keywords": ", ".join(article_dict.get("keywords", [])) if␣
 ↪article_dict.get("keywords") else None,
            "journal": article_dict.get("journal", None),
            "abstract": article_dict.get("abstract", None),
```

```
            # "methods": article_dict.get("methods", None),
            # "results": article_dict.get("results", None),
            # "conclusions": article_dict.get("conclusions", None),
            # "copyrights": article_dict.get("copyrights", None),
            # "doi": article_dict.get("doi", None),
            "publication_date": article_dict.get("publication_date", None),
            # "authors": authors
        })

    df_articles = pd.DataFrame(articles_data)
    # print(df_articles.head())
    df_articles.to_csv("data/Pubmed_"+disease_to_consider+".csv", index=False)
```

2.0.2 Data Pre-processing

[1]:
```
import os
import pandas as pd

# Define the directory containing the CSV files
directory = "data/"  # Change this to your actual directory

# Ensure the directory exists
if not os.path.exists(directory):
    print(f"Directory '{directory}' does not exist.")
    exit()

master_df = pd.DataFrame()
# Process all CSV files in the directory
for filename in os.listdir(directory):
    if filename.startswith("Pubmed_"):
        if filename.endswith(".csv"):  # Only process CSV files
            filepath = os.path.join(directory, filename)

            # Load CSV
            df = pd.read_csv(filepath)

            # Remove specified columns if they exist
            columns_to_remove = ["copyrights", "doi", "authors"]
            try:
                df.drop(columns=[col for col in columns_to_remove if col in df.
 ↪columns], inplace=True)
            except:
                print("Error in removing columns")

            # Save back to the same file
            df['category']= filename
            df.to_csv(filepath, index=False)
```

```
            print("Number of instances in "+filename, len(df))
            master_df = pd.concat([master_df, df], ignore_index=True)

print("Pubmed Data:")
master_df.to_csv('data/all_pubmed.csv', index=False)
print("all_pubmed",len(master_df))
```

```
Number of instances in Pubmed_Pancreatic_Cancer.csv 9831
Number of instances in Pubmed_influenza.csv 8905
Number of instances in Pubmed_pancreatic cancer.csv 60
Number of instances in Pubmed_Chagas_Disease.csv 680
Number of instances in Pubmed_malaria.csv 6855
Number of instances in Pubmed_Endometriosis.csv 2839
Number of instances in Pubmed_breast-cancer.csv 171
Number of instances in Pubmed_chagas disease.csv 60
Number of instances in Pubmed_Duchenne_Muscular_Dystrophy.csv 1423
Number of instances in Pubmed_hepatitis.csv 7087
Number of instances in Pubmed_alzheimer.csv 60
Number of instances in Pubmed_Drug_Resistant_Tuberculosis.csv 1274
Pubmed Data:
all_pubmed 39245
```