**Прикладные задачи**

1. **Работа с PubMed**

Используем Entrez для парсинга публикация pubMed

from Bio import Entrez  
  
Entrez.email = **"e.p@d\_health.pro"** # Говорю NCBI кто я есть  
handle = Entrez.einfo()  
result = handle.read()  
handle.close()  
print(**"Список доступных баз NCBI"**)  
print(result)  
  
**'Считаем количество публикаций в PubMed'**handle = Entrez.einfo(db=**"pubmed"**)  
record = Entrez.read(handle)  
print(record[**"DbInfo"**][**"Description"**], **"Публикаций"**, record[**"DbInfo"**][**"Count"**], **"Последнее обновление"**,  
 record[**"DbInfo"**][**"LastUpdate"**])  
handle.close()  
print(**'**\n**'**)  
  
**'Структура и поля публикаций'**for field in record[**"DbInfo"**][**"FieldList"**]:  
 print(**'%(Name)s, %(FullName)s, %(Description)s'** % field)  
print(**'**\n**'**)  
  
**'отбираем erectile dysfunction'**handle = Entrez.esearch(db=**"pubmed"**, term=**"erectile dysfunction"**)  
record = Entrez.read(handle)  
print(record[**"Count"**])  
print(record)  
handle.close()  
hh = Entrez.efetch (db = **"pubmed"**, id = **"31639279"**)  
print(hh.read())  
hh.close()

1. **Изучение проблемных областей исследований посредством представления фактов и знаний на графах**

**Дано:**

Имеется база данных публикаций биомедицинской литературы PubMed (> 30 млн. article).

Имеются программные интерфейсы доступа к ней.

**Требуется:**

1. Идентифицировать всех авторов и их принадлежность к учреждениям проводящим исследования (по какому-либо произвольно формируемому запросу – напр. “Decision Support Systems”)
2. Провести сортировку публикаций по значимости (квантиль) издания
3. Построить граф связности авторов – (узлы-авторы, ноды-количество совместных публикаций)

Пример решения:

import csv  
import itertools  
  
from pymed import PubMed  
  
pubmed = PubMed(tool="For MyGraphAnalysis", email="e.p@d-health.pro")

# запрос СППР

query = "decision support system[Title]"  
print("Найдено", pubmed.getTotalResultsCount(query))  
  
# Execute the query against the API  
results = list(pubmed.query(query, max\_results=1000))  
  
# В узлы пишем авторов   
nodes = {  
 author: index  
 for index, author in enumerate(  
 set(  
 itertools.chain.from\_iterable(  
 [  
 [  
 f'{author["lastname"]} {author["firstname"]}'  
 for author in article.authors  
 ]  
 for article in results  
 ]  
 )  
 )  
 )  
}  
  
# В связи - + для каждой комбинации авторов (co-authorship)  
edges = list(  
 itertools.chain.from\_iterable(  
 [  
 [combination for combination in itertools.combinations(co\_author\_list, 2)]  
 for co\_author\_list in [  
 [  
 nodes[f'{author["lastname"]} {author["firstname"]}']  
 for author in article.authors  
 ]  
 for article in results  
 ]  
 ]  
 )  
)  
  
# De-duplicate the list of edges by adding a weight  
edges = set([(edge[0], edge[1], edges.count(edge)) for edge in edges])  
  
  
# Создаем файл с узлами графа   
with open("./nodes.csv", "w", encoding="utf8", newline="") as nodes\_file:  
  
 # Create a CSV writer  
 writer = csv.writer(nodes\_file, delimiter=",")  
  
 # Write the header  
 writer.writerow(["id", "label"])  
  
 # Loop over the authors and create rows  
 for name, index in nodes.items():  
 writer.writerow([index, name])  
  
# Создаем файл со связями  
with open("./edges.csv", "w", encoding="utf8", newline="") as edge\_file:  
  
 # Create a CSV writer  
 writer = csv.writer(edge\_file, delimiter=",")  
  
 # Write the header  
 writer.writerow(["source", "target", "weight"])  
  
 # Loop over the edges and put them in the file  
 for edge in edges:  
 writer.writerow(edge)

Для построения графа использовали Gephi.

Нужно найти красивую Python-граф библиотеку и перенести в интерфейс создаваемого решения.

**3. Примеры работы с данными запроса  
  
from pymed import PubMed  
  
  
# Create a PubMed object that GraphQL can use to query  
# Note that the parameters are not required but kindly requested by PubMed Central  
# https://www.ncbi.nlm.nih.gov/pmc/tools/developers/  
pubmed = PubMed(tool="MyTool", email="my@email.address")  
  
# Create a GraphQL query in plain text  
query = "occupational health[Title]"  
  
  
# Execute the query against the API  
results = pubmed.query(query, max\_results=500)  
  
# Loop over the retrieved articles  
for article in results:  
  
 # Print the type of object we've found (can be either PubMedBookArticle or PubMedArticle)  
 print(type(article))  
  
 # Print a JSON representation of the object  
 print(article.toJSON())  
""  
  
# Третий пример работы с данными  
# Формирование сложного поискового запроса  
from pymed import PubMed  
  
# Create a PubMed object that GraphQL can use to query  
# Note that the parameters are not required but kindly requested by PubMed Central  
# https://www.ncbi.nlm.nih.gov/pmc/tools/developers/  
pubmed = PubMed(tool="MyTool", email="e.p@d-health.pro")  
  
# Create a GraphQL query in plain text  
query = '(("2018/05/01"[Date - Create] : "3000"[Date - Create])) AND (erectyile dysfunction[Title] AND diabetes)'  
#print("Найдено", pubmed.getTotalResultsCount(query))  
  
# Execute the query against the API  
results = pubmed.query(query, max\_results=500)  
  
# Loop over the retrieved articles  
for article in results:  
  
 # Extract and format information from the article  
 article\_id = article.pubmed\_id  
 title = article.title  
 if article.keywords:  
 if None in article.keywords:  
 article.keywords.remove(None)  
 keywords = '", "'.join(article.keywords)  
 publication\_date = article.publication\_date  
 abstract = article.abstract  
  
 # Show information about the article  
 print(  
 f'{article\_id} - {publication\_date} - {title}**\n**Keywords: "{keywords}"**\n**{abstract}**\n**'  
 )  
  
  
from pymed import PubMed  
  
# Create a PubMed object that GraphQL can use to query  
# Note that the parameters are not required but kindly requested by PubMed Central  
# https://www.ncbi.nlm.nih.gov/pmc/tools/developers/  
pubmed = PubMed(tool="MyTool", email="my@email.address")  
  
# Create a GraphQL query in plain text  
query = '(("2018/05/01"[Date - Create] : "3000"[Date - Create])) AND (Xiaoying Xian[Author] OR diabetes)'  
#print("Найдено", pubmed.getTotalResultsCount(query))  
  
# Execute the query against the API  
results = pubmed.query(query, max\_results=500)  
  
# Loop over the retrieved articles  
for article in results:  
  
 # Extract and format information from the article  
 article\_id = article.pubmed\_id  
 title = article.title  
 if article.keywords:  
 if None in article.keywords:  
 article.keywords.remove(None)  
 keywords = '", "'.join(article.keywords)  
 publication\_date = article.publication\_date  
 abstract = article.abstract  
  
 # Show information about the article  
 print(  
 f'{article\_id} - {publication\_date} - {title}**\n**Keywords: "{keywords}"**\n**{abstract}**\n**'  
 )  
"""**