Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика, искусственный интеллект и системы управление»

Кафедра ИУ5. Курс «Методы машинного обучения»

Отчет по домашнему заданию

Выполнил:
студент группы ИУ5-23Б
Белоусов Евгений
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2022 г.

Постановка задачи

В качестве домашнего задания по курсу ММО было выбрано проведение эксперимента по выявлению зависимостей времени загрузки и выгрузки метаграфа из метаграфового хранилища.

Теоретическая часть отчета

Цель эксперимента: получить зависимость времени загрузки метаграфа в метаграфовое хранилище и выгрузки оттуда от количества различных элементов метаграфовой модели.

Для этого

1. Производится генерация случайного метаграфа с заданным количеством элементов.
2. Метаграф загружается в хранилище, измеряется время загрузки.
3. Метаграф выгружается из хранилища, измеряется время выгрузки.
4. Производится чистка хранилища.

Таким образом, изменяя параметры генерируемых метаграфов можно получить необходимые зависимости.

Было решено получить зависимости от всех основных элементов метаграфовой модели:

1. Вершин
2. Метавершин
3. Ребер
4. Метаребер
5. Атрибутов

Практическая часть отчета

Описанный выше алгоритм был исполнен на Python с помощью интерактивной среды Jupyter Notebook.

Ввод [1]:
```
import sys
sys.path.append('C:\\Users\\Evgeny\\PycharmProjects\\MetagraphDB\\metagraphdb\\modules\\Module_input_output_xml')
sys.path.append('C:\\Users\\Evgeny\\PycharmProjects\\MetagraphDB\\metagraphdb\\modules\\Module_for_creating_and_modifying_element
```

Ввод [2]:
```
import parcer
import generator
import psycopg2
```

Ввод [3]:
```
gen = generator.Generator()
gen.generateXML('test.xml', 1000, 150, 400, 35)
```

Ввод [4]:
```
test = parcer.IOParcer(dbname='metagraph_db', user='postgres', password='1', host='127.0.0.3')
test.load_xml('test.xml', 'test2')
```

Ввод [5]:
```
test.save_xml(str(test.all_nodeid_id['Main_metanode_id'][0]), 'test2', 'result.xml')
```

Ввод [3]:
```
tables_for_cleaning = ['timetable', 'nodes', 'linkpoints', 'attributes', 'temporal_mutables', 'temporal_immutables']
def delete_all():
    conn = psycopg2.connect(dbname='metagraph_db',
                            user='postgres',
                            password='1',
                            host='127.0.0.2')
    with conn.cursor() as cursor:
        sql = "DELETE FROM metagraph_db.{}"
        for table in tables_for_cleaning:
            cursor.execute(sql.format(table))
    conn.commit()
    conn.close()
```

Ввод [14]:
```
delete_all()
```

Ввод [17]:
```
from tqdm.auto import trange, tqdm
from IPython.display import clear_output
from time import sleep


MAX_VALUE = 2000
MIN_VALUE = 100
STEP = 100

for nodes in range(MIN_VALUE, MAX_VALUE+1, STEP):
    clear_output(wait=False)
    pbar = tqdm(total=(MAX_VALUE)*3, initial=(nodes - MIN_VALUE)*3)
    sleep(0.1)
    pbar.update(STEP)
    print('Hi')
    sleep(0.1)
    pbar.update(STEP)
    print('Hao')
    sleep(0.1)
    pbar.update(STEP)
    print('Hello')
    #pbar.update(1000)
```

```
 95%|#########5| 57000/60000 [00:00<?, ?it/s]

Hi
Hao
Hello
```

подготовка экспериментов

Подготовка экспериментов

Ввод [9]:
```
from tqdm.auto import tqdm
from IPython.display import clear_output
from time import time
import pandas as pd



excel_name = 'nodes.xlsx'
result_xml = 'result.xml'
xml_name = 'DAMDID.xml'
time_stamp = 'DAMDID'
nodes = 1000
metanodes = 500
edges = 1000
metaedges = 100
load_times = []
save_times = []
num = []

MAX_VALUE = 2000
MIN_VALUE = 100
STEP = 100

for nodes in range(MIN_VALUE, MAX_VALUE+1, STEP):
    clear_output(wait=False)
    pbar = tqdm(total=MAX_VALUE*4, initial=(nodes - MIN_VALUE)*4)
    pbar.update(STEP)
    gen = generator.Generator()
    loader = parcer.IOParcer(dbname='metagraph_db', user='postgres', password='1', host='127.0.0.3')
    print('GENERATION')
    gen.generateXML(xml_name, nodes, metanodes, edges, metaedges)
    pbar.update(STEP)
    print('LOADING')
    start_time = time()
    loader.load_xml(xml_name, time_stamp)
    end_time = time()
    load_times.append(end_time - start_time)
    pbar.update(STEP)
    print('SAVING')
    start_time = time()
    loader.save_xml(str(loader.all_nodeid_id['Main_metanode_id'][0]), time_stamp, result_xml)
    end_time = time()
    save_times.append(end_time - start_time)
    pbar.update(STEP)
    print('DELETEING')
    delete_all()
    num.append(nodes)

df = pd.DataFrame({"nodes_num": num,
                   "load_times": load_times,
                   "save_times": save_times})
df.to_excel(excel_name)
```

```
 95%|#########5| 7600/8000 [00:00<?, ?it/s]

GENERATION
LOADING
SAVING
DELETEING
```

Ввод [15]:

```
excel_name = 'metanodes.csv'
result_xml = 'result.xml'
xml_name = 'DAMDID.xml'
time_stamp = 'DAMDID'
nodes = 1000
metanodes = 500
edges = 1000
metaedges = 500
attributes = 5
load_times = []
save_times = []
num = []

MAX_VALUE = 2000
MIN_VALUE = 100
STEP = 100

for metanodes in range(MIN_VALUE, MAX_VALUE+1, STEP):
    clear_output(wait=False)
    pbar = tqdm(total=MAX_VALUE*4, initial=(metanodes - MIN_VALUE)*4)
    pbar.update(STEP)
    gen = generator.Generator()
    loader = parcer.IOParcer(dbname='metagraph_db', user='postgres', password='1', host='127.0.0.3')
    print('GENERATION')
    gen.generateXML(xml_name, nodes, metanodes, edges, metaedges, max_number_of_atributes=attributes)
    pbar.update(STEP)
    print('LOADING')
    start_time = time()
    loader.load_xml(xml_name, time_stamp)
    end_time = time()
    load_times.append(end_time - start_time)
    pbar.update(STEP)
    print('SAVING')
    start_time = time()
    loader.save_xml(str(loader.all_nodeid_id['Main_metanode_id'][0]), time_stamp, result_xml)
    end_time = time()
    save_times.append(end_time - start_time)
    pbar.update(STEP)
    print('DELETEING')
    delete_all()
    num.append(nodes)

df = pd.DataFrame({"metanodes_num": num,
                   "load_times": load_times,
                   "save_times": save_times})
df.to_csv(excel_name)
```

```
 95%|#########5| 7600/8000 [00:00<?, ?it/s]

GENERATION
LOADING
SAVING
DELETEING
```

Ввод [17]:

```
excel_name = 'edges.csv'
result_xml = 'result.xml'
xml_name = 'DAMDID.xml'
time_stamp = 'DAMDID'
nodes = 1000
metanodes = 500
edges = 1000
attributes = 5
metaedges = 500
load_times = []
save_times = []
num = []

MAX_VALUE = 2000
MIN_VALUE = 100
STEP = 100

for edges in range(MIN_VALUE, MAX_VALUE+1, STEP):
    clear_output(wait=False)
    pbar = tqdm(total=MAX_VALUE*4, initial=(edges - MIN_VALUE)*4)
    pbar.update(STEP)
    gen = generator.Generator()
    loader = parcer.IOParcer(dbname='metagraph_db', user='postgres', password='1', host='127.0.0.3')
    print('GENERATION')
    gen.generateXML(xml_name, nodes, metanodes, edges, metaedges, max_number_of_atributes=attributes)
    pbar.update(STEP)
    print('LOADING')
    start_time = time()
    loader.load_xml(xml_name, time_stamp)
    end_time = time()
    load_times.append(end_time - start_time)
    pbar.update(STEP)
    print('SAVING')
    start_time = time()
    loader.save_xml(str(loader.all_nodeid_id['Main_metanode_id'][0]), time_stamp, result_xml)
    end_time = time()
    save_times.append(end_time - start_time)
    pbar.update(STEP)
    print('DELETEING')
    delete_all()
    num.append(nodes)

df = pd.DataFrame({"edges_num": num,
                   "load_times": load_times,
                   "save_times": save_times})
df.to_csv(excel_name)
```

```
 95%|#########5| 7600/8000 [00:00<?, ?it/s]

GENERATION
LOADING
SAVING
DELETEING
```

Ввод [18]:

```
excel_name = 'metaedges.csv'
result_xml = 'result.xml'
xml_name = 'DAMDID.xml'
time_stamp = 'DAMDID'
nodes = 1000
metanodes = 500
attributes = 5
edges = 1000
metaedges = 500
load_times = []
save_times = []
num = []

MAX_VALUE = 2000
MIN_VALUE = 100
STEP = 100

for metaedges in range(MIN_VALUE, MAX_VALUE+1, STEP):
    clear_output(wait=False)
    pbar = tqdm(total=MAX_VALUE*4, initial=(metaedges - MIN_VALUE)*4)
    pbar.update(STEP)
    gen = generator.Generator()
    loader = parcer.IOParcer(dbname='metagraph_db', user='postgres', password='1', host='127.0.0.3')
    print('GENERATION')
    gen.generateXML(xml_name, nodes, metanodes, edges, metaedges, max_number_of_atributes=attributes)
    pbar.update(STEP)
    print('LOADING')
    start_time = time()
    loader.load_xml(xml_name, time_stamp)
    end_time = time()
    load_times.append(end_time - start_time)
    pbar.update(STEP)
    print('SAVING')
    start_time = time()
    loader.save_xml(str(loader.all_nodeid_id['Main_metanode_id'][0]), time_stamp, result_xml)
    end_time = time()
    save_times.append(end_time - start_time)
    pbar.update(STEP)
    print('DELETEING')
    delete_all()
    num.append(nodes)

df = pd.DataFrame({"metaedges_num": num,
                   "load_times": load_times,
                   "save_times": save_times})
df.to_csv(excel_name)
```

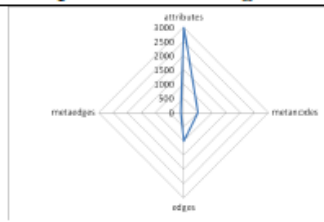
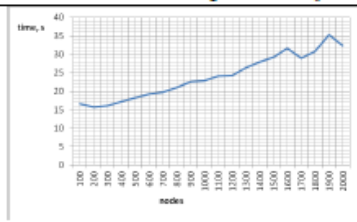
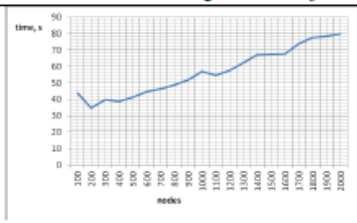
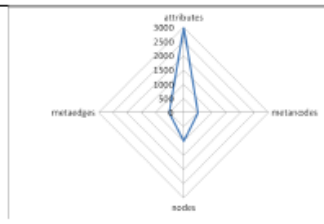
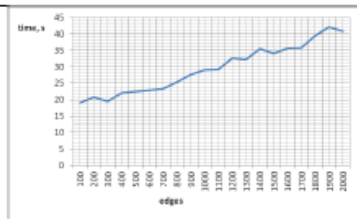
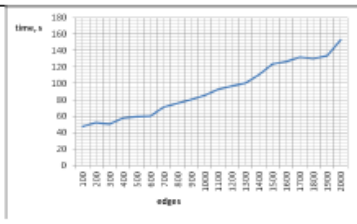
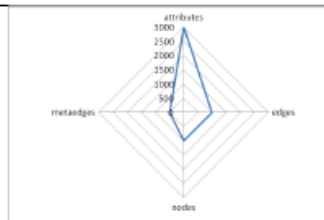
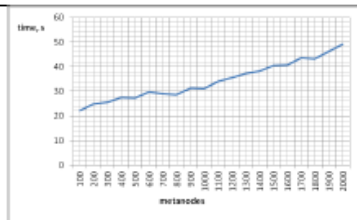
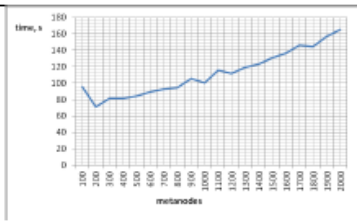
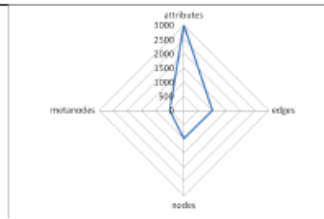
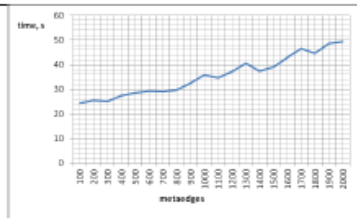
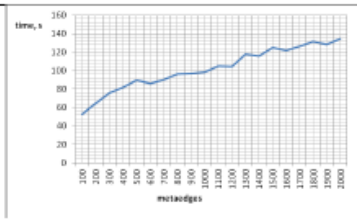
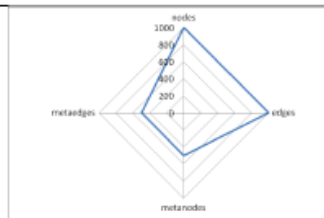
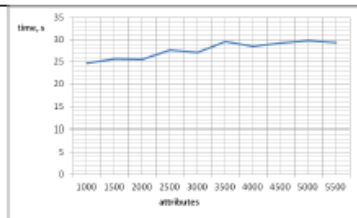
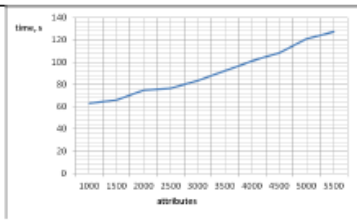
```
 95%|#########5| 7600/8000 [00:00<?, ?it/s]

GENERATION
LOADING
SAVING
DELETEING
```

Ввод [19]:

```
excel_name = 'attributes.csv'
result_xml = 'result.xml'
xml_name = 'DAMDID.xml'
time_stamp = 'DAMDID'
nodes = 1000
metanodes = 500
attributes = 5
edges = 1000
metaedges = 500
load_times = []
save_times = []
num = []

MAX_VALUE = 10
MIN_VALUE = 1
STEP = 1

for attributes in range(MIN_VALUE, MAX_VALUE+1, STEP):
    clear_output(wait=False)
    pbar = tqdm(total=MAX_VALUE*4, initial=(attributes - MIN_VALUE)*4)
    pbar.update(STEP)
    gen = generator.Generator()
    loader = parcer.IOParcer(dbname='metagraph_db', user='postgres', password='1', host='127.0.0.3')
    print('GENERATION')
    gen.generateXML(xml_name, nodes, metanodes, edges, metaedges, max_number_of_atributes=attributes)
    pbar.update(STEP)
    print('LOADING')
    start_time = time()
    loader.load_xml(xml_name, time_stamp)
    end_time = time()
    load_times.append(end_time - start_time)
    pbar.update(STEP)
    print('SAVING')
    start_time = time()
    loader.save_xml(str(loader.all_nodeid_id['Main_metanode_id'][0]), time_stamp, result_xml)
    end_time = time()
    save_times.append(end_time - start_time)
    pbar.update(STEP)
    print('DELETEING')
    delete_all()
    num.append(nodes)

df = pd.DataFrame({"attributes_num": num,
                   "load_times": load_times,
                   "save_times": save_times})
df.to_csv(excel_name)
```

```
 90%|######### | 36/40 [00:00<?, ?it/s]

GENERATION
LOADING
SAVING
DELETEING
```

В результате проведения экспериментов были построены следующие графики:

№	Experiment configuration	Load time dependency	Save time dependency
1			
2			
3			
4			
5			

Выводы по результатам выполнения теоретической и практической частей

В результате проведения эксперимента можно сделать следующие выводы:

1. Наиболее важным результатом является то, что загрузка и выгрузка метаграфа в среднем происходит за линейное время от количества его элементов.
2. Видно, что метаграфовое хранилище может работать с метаграфовыми моделями, состоящими из нескольких тысяч элементов.
3. Неожиданно, время чтения метаграфа из хранилища оказалось дольше, чем время его записи в хранилище. Поэтому в дальнейшем необходимо провести оптимизацию операции чтения для уменьшения этого времени.
4. Временные зависимости от разных элементов метаграфовой модели выглядят одинаково. Это дает хорошие перспективы к взаимодействию с

вложенными структурами.