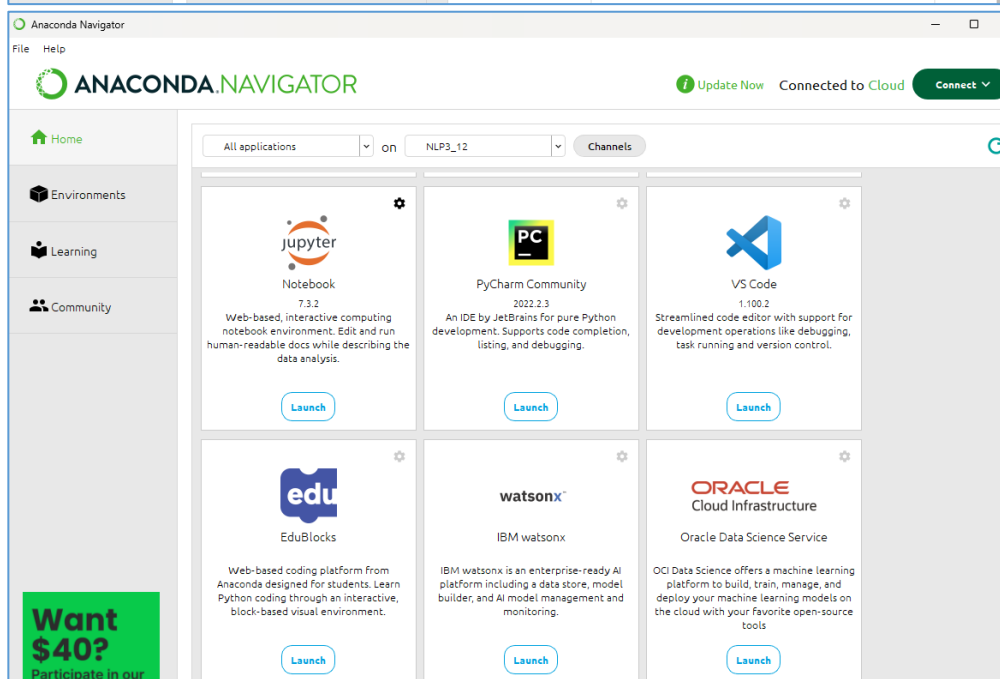
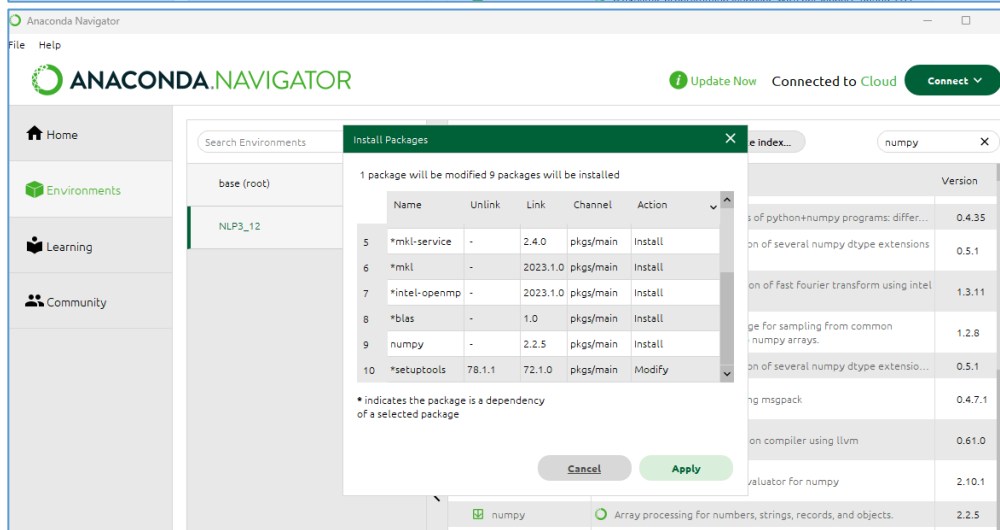
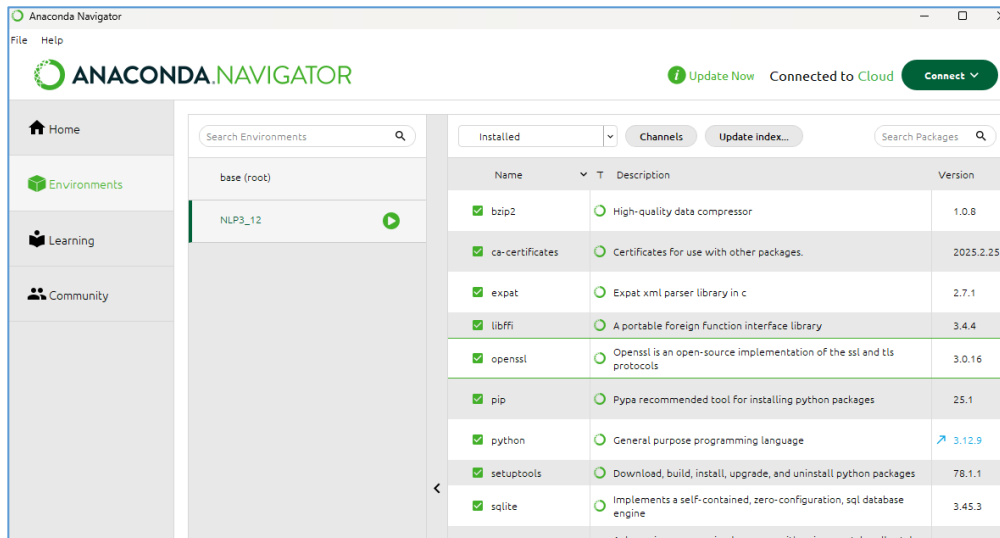

Вводный курс по Big Data. Обучение в записи

Урок 2. Семинар: Введение в Большие Данные

Оглавление

Семинар. Установка и настройка окружения Anaconda.....	2
Семинар. Выполнение практических задач.....	4
Домашняя работа.....	7

Семинар. Установка и настройка окружения Anaconda



Jupyter

LMV_1_BIG_DATA

Last Checkpoint: 21 минуту назад

File

Edit

View

Run

Kernel

Settings

Help

Not Trusted

+

✂

▶

■

↺

▶▶

Markdown

▼

JupyterLab

Python 3 (ipykernel)

Подготовка окружения

[7]: !pip install opendatasets

Requirement already satisfied: opendatasets in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (0.1.22)
Requirement already satisfied: tqdm in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from opendatasets) (4.67.1)
Requirement already satisfied: kaggle in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from opendatasets) (1.7.4.5)
Requirement already satisfied: click in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from opendatasets) (8.2.0)
Requirement already satisfied: colorama in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from click->opendatasets) (0.4.6)
Requirement already satisfied: bleach in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from kaggle->opendatasets) (6.2.0)
Requirement already satisfied: certifi>=14.05.14 in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from kaggle->opendatasets) (2025.4.26)
Requirement already satisfied: charset-normalizer in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from kaggle->opendatasets) (3.3.2)
Requirement already satisfied: idna in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from kaggle->opendatasets) (3.7)
Requirement already satisfied: protobuf in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from kaggle->opendatasets) (6.31.0)
Requirement already satisfied: python-dateutil>=2.5.3 in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from kaggle->opendatasets) (2.9.0.post0)
Requirement already satisfied: python-slugify in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from kaggle->opendatasets) (8.0.4)
Requirement already satisfied: requests in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from kaggle->opendatasets) (2.32.3)
Requirement already satisfied: setuptools>=21.0.0 in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from kaggle->opendatasets) (72.1.0)
Requirement already satisfied: six>=1.10 in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from kaggle->opendatasets) (1.17.0)

Jupyter

LMV_1_BIG_DATA

Last Checkpoint: 21 минуту назад

File

Edit

View

Run

Kernel

Settings

Help

Not Trusted

+

✂

▶

■

↺

▶▶

Code

▼

JupyterLab

Python 3 (ipykernel)

Подготовка окружения

[12]: !pip install pandas

Collecting pandas
 Downloading pandas-2.2.3-cp312-cp312-win_amd64.whl.metadata (19 kB)
 Requirement already satisfied: numpy>=1.26.0 in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from pandas) (2.2.5)
 Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from pandas) (2.9.0.post0)
 Collecting pytz>=2020.1 (from pandas)
 Downloading pytz-2025.2-py2.py3-none-any.whl.metadata (22 kB)
 Collecting tzdata>=2022.7 (from pandas)
 Downloading tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
 Requirement already satisfied: six>=1.5 in c:\users\user\anaconda3\envs\nlp3_12\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
 Downloading pandas-2.2.3-cp312-cp312-win_amd64.whl (11.5 MB)
----- 0.0/11.5 MB ? eta -:-:-
----- 0.3/11.5 MB ? eta -:-:-
----- 0.8/11.5 MB 2.6 MB/s eta 0:00:05
----- 1.3/11.5 MB 2.2 MB/s eta 0:00:05
----- 1.8/11.5 MB 2.2 MB/s eta 0:00:05

[11]: import opendatasets as od
import pandas as pd
import time
from pathlib import Path

ModuleNotFoundError Traceback (most recent call last)
Cell In[11], line 2
 1 import opendatasets as od
----> 2 import pandas as pd
 3 import time
 4 from pathlib import Path

ModuleNotFoundError: No module named 'pandas'

Установка Pandas и загрузка ключа (токена) Kaggle ...

Семинар. Выполнение практических задач

The screenshot shows two instances of the JupyterLab environment. The top instance displays the output of several Python commands:

```
[21]: large_list_of_strings = list_of_strings * 1_000_000
```

```
[22]: len(large_list_of_strings)
```

```
[22]: 3000000
```

```
[ ]: large_list_of_strings[:10]
```

```
[50]: ['abc123',  
      'питон',  
      'sasha2000',  
      'abc123',  
      'питон',  
      'sasha2000',  
      'abc123',  
      'питон',  
      'sasha2000',  
      'abc123']
```

```
[23]: %time print(find_longest_string(large_list_of_strings))
```

(python', 6)
CPU times: total: 2.44 s
Wall time: 2.46 s

Как мы уже знаем, для того, чтобы код выполнялся быстрее, можно взять более мощный компьютер/процессор (vertical scaling), либо считать на нескольких компьютерах/процессорах/ядрах параллельно (horizontal scaling). Пример выше является идеально параллелизуемой задачей. Подготовим данные для распараллеливания:

The bottom instance shows the preparation of parallelizable data:

```
[24]: from functools import reduce
```

```
def chunkify(list_of_strings, number_of_chunks=30):  
    step = len(list_of_strings) // number_of_chunks  
    if step != 0:  
        for i in range(0, len(list_of_strings), step):  
            yield list_of_strings[i : i + step]  
    else:  
        yield list_of_strings
```

```
def chunks_mapper(chunk):  
    return find_longest_string(chunk)
```

```
def reducer(p, c):  
    if p[1] > c[1]:  
        return p  
    return c
```

```
[25]: %%timeit  
data_chunks = chunkify(large_list_of_strings, number_of_chunks=1)  
#step 1:  
mapped = map(chunks_mapper, data_chunks)  
#step 2:  
reduced = reduce(reducer, mapped)  
print(reduced)
```

(python', 6)
(python', 6)
(python', 6)
(python', 6)
(python', 6)
(python', 6)
(python', 6)
(python', 6)
(python', 6)
2.45 ± 28.2 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

Jupyter LMV_1_BIG_DATA Last Checkpoint: 1 час назад

File Edit View Run Kernel Settings Help Не доверены

Уценка

```
[26]: %%timeit
data_chunks = chunkify(large_list_of_strings, number_of_chunks=2)
#step 1:
mapped = map(chunks_mapper, data_chunks)
#step 2:
reduced = reduce(reducer, mapped)
print(reduced)

('python', 6)
('python', 6)
('python', 6)
('python', 6)
('python', 6)
('python', 6)
('python', 6)
('python', 6)
2.44 s ± 16.6 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Наши данные, в среднем, обчитываются за 2.44 с.

Jupyter LMV_1_BIG_DATA Last Checkpoint: 1 час назад

File Edit View Run Kernel Settings Help Не доверены

Код

Наши данные, в среднем, обчитываются за 2.44 с.

```
[27]: from joblib import Parallel, delayed
```

```
ModuleNotFoundError                                Traceback (most recent call last)
Cell In[27], line 1
----> 1 from joblib import Parallel, delayed

ModuleNotFoundError: No module named 'joblib'
```

Jupyter LMV_1_BIG_DATA Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help Not Trusted

Markdown

Подготовка окружения

```
[29]: !pip install joblib
```

```
Collecting joblib
  Downloading joblib-1.5.0-py3-none-any.whl.metadata (5.6 kB)
  Downloading joblib-1.5.0-py3-none-any.whl (307 kB)
Installing collected packages: joblib
Successfully installed joblib-1.5.0
```

Jupyter LMV_1_BIG_DATA Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help Not Trusted

Markdown

```
[30]: from joblib import Parallel, delayed
```

```
[31]: %%timeit
N = 2
data_chunks = chunkify(large_list_of_strings, number_of_chunks=N)
#step 1:
mapped = Parallel(n_jobs=N)(delayed(chunks_mapper)(chunk) for chunk in data_chunks)
#step 2:
reduced = reduce(reducer, mapped)

877 ms ± 6.58 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

В два потока стало считать быстрее. Но из-за ограничений colab мы не можем полноценно воспользоваться мультиточечностью, попробуйте на личном компьютере какой у вас будет прирост (у меня 1.29 против 1.74, а в 4 потока считается за 0.7 секунды)?

```
[32]: %%timeit
N = 4
data_chunks = chunkify(large_list_of_strings, number_of_chunks=N)
#step 1:
mapped = Parallel(n_jobs=N)(delayed(chunks_mapper)(chunk) for chunk in data_chunks)
#step 2:
reduced = reduce(reducer, mapped)

482 ms ± 10.3 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
[33]: %%timeit
N = 8
data_chunks = chunkify(large_list_of_strings, number_of_chunks=N)
#step 1:
mapped = Parallel(n_jobs=N)(delayed(chunks_mapper)(chunk) for chunk in data_chunks)
#step 2:
reduced = reduce(reducer, mapped)

299 ms ± 14.1 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
[36]: %%timeit
N = 12
data_chunks = chunkify(large_list_of_strings, number_of_chunks=N)
#step 1:
mapped = Parallel(n_jobs=N)(delayed(chunks_mapper)(chunk) for chunk in data_chunks)
#step 2:
reduced = reduce(reducer, mapped)

262 ms ± 8.95 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
[35]: %%timeit
N = 10
data_chunks = chunkify(large_list_of_strings, number_of_chunks=N)
#step 1:
mapped = Parallel(n_jobs=N)(delayed(chunks_mapper)(chunk) for chunk in data_chunks)
#step 2:
reduced = reduce(reducer, mapped)

272 ms ± 11.6 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Домашняя работа

Ознакомьтесь с библиотеками для распараллеливания задач в Python:

1. multiprocessing
2. joblib
3. dask

Условия и критерии выполнения домашнего задания внутри файла

1. MULTIPROCESSING

Модуль **multiprocessing** в Python позволяет запускать несколько процессов одновременно для параллельного выполнения задач. Каждый процесс имеет собственную память и независимое выполнение, в отличие от потоков, которые работают в рамках одного процесса.

Это позволяет обходить ограничение Global Interpreter Lock (GIL), что даёт возможность эффективно использовать многоядерные процессоры.

Основные понятия

Некоторые ключевые элементы модуля multiprocessing:

Класс Process — для создания и управления процессами. Позволяет запускать функции в новом процессе.

Класс Pool — предоставляет пул процессов для выполнения задач параллельно. Позволяет распределять задачи между процессами.

Класс Queue — для обмена данными между процессами через очередь.

Классы Lock, Event, Condition, Semaphore — для синхронизации процессов и предотвращения состязаний при доступе к общим ресурсам.

Преимущества и недостатки

Преимущества:

- истинный параллелизм для CPU-задач;
- изоляция процессов (сбой в одном не влияет на другие).

Недостатки:

- высокие накладные расходы на создание процессов;
- сложности с обменом данными (требуется сериализация через pickle);
- большее потребление памяти.

Примеры использования

Модуль multiprocessing подходит для работы с CPU-зависимыми задачами, такими как сложные вычисления, обработка изображений, научные расчёты. Также его можно использовать для распараллеливания большого объёма данных или ускорения итераций тяжёлых циклов.

Если основная нагрузка связана с вводом-выводом (например, сетевые запросы или работа с файлами), лучше использовать асинхронное программирование или многопоточность.

2. JOBLIB

Joblib — библиотека для распараллеливания задач в Python, которая предоставляет инструменты для параллельных вычислений, кэширования и эффективной обработки данных.

Основные функции и возможности

Распараллеливание вычислений с помощью класса `Parallel`. Он позволяет распределить выполнение итераций цикла или вызовы функций на несколько ядер процессора.

Кэширование результатов с помощью класса `Memory`. При повторном вызове функции с одинаковыми аргументами результат не вычисляется заново, а возвращается из кэша.

Сохранение и загрузка объектов Python с помощью функций `joblib.dump()` и `joblib.load()`. Поддерживаются различные методы сжатия данных, например `zlib`, `gzip`, `bz2`, `xz`.

Примеры использования

Распараллеливание цикла. Например, вычисление квадратного корня для каждого элемента списка чисел. Без параллельных вычислений это работает последовательно, а с помощью `Joblib` можно распараллелить цикл.

Параллельная загрузка файлов. Например, импорт нескольких файлов CSV или Excel. `Joblib` позволяет распределить задачи между несколькими ядрами или потоками, сократив общее время загрузки.

3. DASK

`Dask` — библиотека Python для параллельных вычислений и масштабируемой обработки данных. Она позволяет работать с данными, которые не помещаются в оперативную память, и исполнять задачи распределённо на многопроцессорных машинах или кластерах.

Основные возможности

Разбиение данных на части (чанки) для параллельной обработки.

Поддержка ленивых вычислений — операции выполняются только при необходимости.

Интеграция с другими библиотеками Python, такими как `Pandas`, `NumPy` и `Scikit-learn`.

Масштабируемость — работа от одного ядра до кластера.

Визуализация вычислений и мониторинг в браузере.

Установка и подключение

Установка можно использовать пакетный менеджер `pip` или `conda`. Например, в дистрибутиве `Anaconda` `Dask` установлен по умолчанию.

Подключение осуществляется через импорт модулей, например `import dask.dataframe as dd` или `import dask.array as da`.

Примеры использования

Обработка большого CSV-файла. Вместо загрузки всего файла в память `Dask` загружает его частями, а затем выполняет параллельные операции, такие как фильтрация и группировка.

Анализ логов веб-сервера. Например, подсчёт количества запросов к каждой странице.

Обработка финансовых данных для вычисления скользящего среднего.

PYTHON

```
import json
from pathlib import Path
from functools import reduce

# Функция mapper: извлекает оценку и вычисляет промежуточные значения
def mapper(path):
    n, mean, M2 = 0, 0.0, 0.0
    if path.is_file() and path.suffix == '.json':
        with open(path, 'r') as f:
            info = json.load(f)
            score = float(info['movieIMDbRating'])
            n += 1
            delta = score - mean
            mean += delta / n
            M2 += delta * (score - mean)
    return n, mean, M2

# Функция reducer: объединяет промежуточные результаты
def reducer(score_data1, score_data2):
    n1, mean1, M21 = score_data1
    n2, mean2, M22 = score_data2

    n = n1 + n2
    if n == 0:
        return 0, 0.0, 0.0

    delta = mean2 - mean1
    mean = mean1 + delta * n2 / n
    M2 = M21 + M22 + delta ** 2 * n1 * n2 / n

    return n, mean, M2

# Измерение времени выполнения всего кода
%%time

# Применяем mapper и объединяем результаты с помощью reducer
n, mean, M2 = reduce(reducer, map(mapper, Path('imdb-user-reviews').glob('*/*')))

# Выводим результат
if n > 0:
    print(f'Среднее: {mean}, Стандартное отклонение: {(M2 / n) ** (1/2)}')
else:
    print("Нет данных для обработки")
```

Ссылка на репозиторий:

<https://github.com/olgashenkel/GeekBrains-specialization-ELECTIVES/tree/main/12.%20Introductory%20course%20on%20Big%20Data>