



Jupyter Notebook



Докладчик: Килинкаров Д.

Что такое jupyter-ноутбук



The screenshot shows a Jupyter Notebook interface with a file explorer on the left, a code editor in the center, and an output area at the bottom. The code in the editor is for character-based text generation using an LSTM. The output area shows the result of a code cell, which is a list of characters: [' ', 'o', 'и', 'е', 'а', 'н', 'т', 'с', 'р', 'в', 'л', 'м', 'п', 'к', 'д', 'я', 'у', 'ы', 'а', 'б', ''].

File Explorer (left):

- sample_data
- child.txt

Code Editor (center):

```
[1] from collections import Counter

import torch
import torch.nn as nn
import torch.nn.functional as F
import numpy as np

[4] TRAIN_TEXT_FILE_PATH = 'child.txt'

with open(TRAIN_TEXT_FILE_PATH) as text_file:
    text_sample = text_file.readlines()
    text_sample = ' '.join(text_sample)

def text_to_seq(text_sample):
    char_counts = Counter(text_sample)
    char_counts = sorted(char_counts.items(), key = lambda x: x[1], reverse=True)

    sorted_chars = [char for char, _ in char_counts]
    print(sorted_chars)
    char_to_idx = {char: index for index, char in enumerate(sorted_chars)}
    idx_to_char = {v: k for k, v in char_to_idx.items()}
    sequence = np.array([char_to_idx[char] for char in text_sample])

    return sequence, char_to_idx, idx_to_char

sequence, char_to_idx, idx_to_char = text_to_seq(text_sample)

[' ', 'o', 'и', 'е', 'а', 'н', 'т', 'с', 'р', 'в', 'л', 'м', 'п', 'к', 'д', 'я', 'у', 'ы', 'а', 'б', '']

[5] SEQ_LEN = 256
    BATCH_SIZE = 16

def get_batch(sequence):
    trains = []
    targets = []
    for _ in range(BATCH_SIZE):
        batch_start = np.random.randint(0, len(sequence) - SEQ_LEN)
        chunk = sequence[batch_start: batch_start + SEQ_LEN]
        train = torch.LongTensor(chunk[:-1]).view(-1, 1)
        target = torch.LongTensor(chunk[1:]).view(-1, 1)
        trains.append(train)
        targets.append(target)
    return torch.stack(trains, dim=0), torch.stack(targets, dim=0)
```

Нагрузка на систему

Файл, с которым работает ноутбук

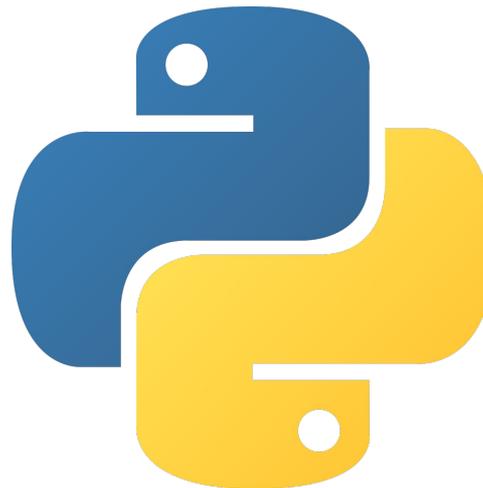
Фрагменты кода

Результат работы фрагмента

Jupyter Notebook — это командная оболочка для интерактивных вычислений. Отличие от традиционной среды разработки в том, что код можно разбить на куски и выполнять их в произвольном порядке.

Какие языки поддерживаются

- Python
- Ruby
- Perl
- R
- bash-скрипты



BASH
THE BOURNE-AGAIN SHELL

Ноутбук в облаке

The screenshot shows a Jupyter Notebook with the following code and output:

```
# крутим рулетку, на которой 18 ч...
# Станем, как и в прошлом случае,
ball = random.randint(1,35)
# пусть первые 18 будут черными -
# если наша ставка сыграла - мы и
if ball in range(1,19):
    # получаем назад нашу ставку
    money = bet * 2
    # увеличиваем количество побед
    win += 1
else:
    # иначе - увеличиваем количес
    loose += 1

# выводим результат игры по третьей стратегии
print("Выиграно ставок: " + str(win))

# строим графики
fig = go.Figure()
# для первой стратегии
fig.add_trace(go.Scatter(x=games1, y=balance1, name = "Отрицательное матожидание"))
# для второй
fig.add_trace(go.Scatter(x=games2, y=balance2, name = "Нулевое матожидание"))
# и для третьей
fig.add_trace(go.Scatter(x=games3, y=balance3, name = "Положительное матожидание"))
# выводим графики в браузер
fig.show()
```

Output:

```
Выиграно ставок: 2099 (48.8371552165)
Выиграно ставок: 2143 (49.8465044885)
Выиграно ставок: 2374 (51.1047025443)
```

The graph displays three lines representing different betting strategies over 40,000 games. The y-axis represents balance, ranging from 0 to 2M. The x-axis represents the number of games, ranging from 0 to 40k. The legend indicates: blue line for 'Отрицательное матожидание' (Negative expectation), red line for 'Нулевое матожидание' (Zero expectation), and green line for 'Положительное матожидание' (Positive expectation). The blue line shows a steady decline, the red line remains relatively flat, and the green line shows a steady increase.

kaggle™

colab

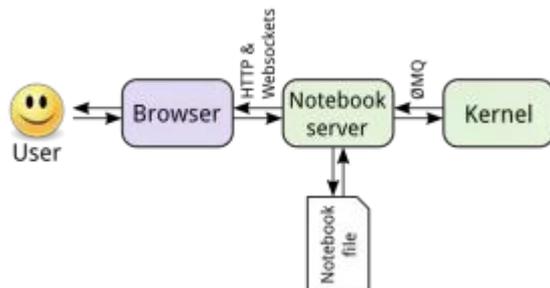
FLOYDHUB



Amazon SageMaker

Где применяются jupyter-ноутбуки

- Машинное обучение
- Нейросети
- Визуализация данных
- Анализ данных
- Сфера образования



IP[y]: Notebook spectrogram Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Help

Cell Toolbar: None

Simple spectral analysis

An illustration of the [Discrete Fourier Transform](#) using windowing, to reveal the frequency content of a sound signal.

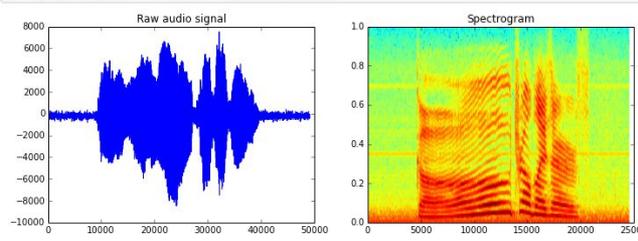
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi}{N}kn} \quad k = 0, \dots, N-1$$

We begin by loading a datfile using SciPy's audio file support:

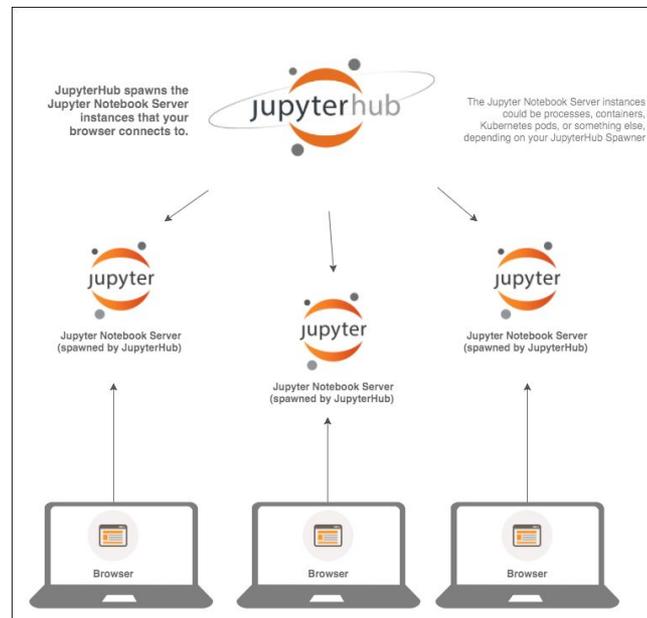
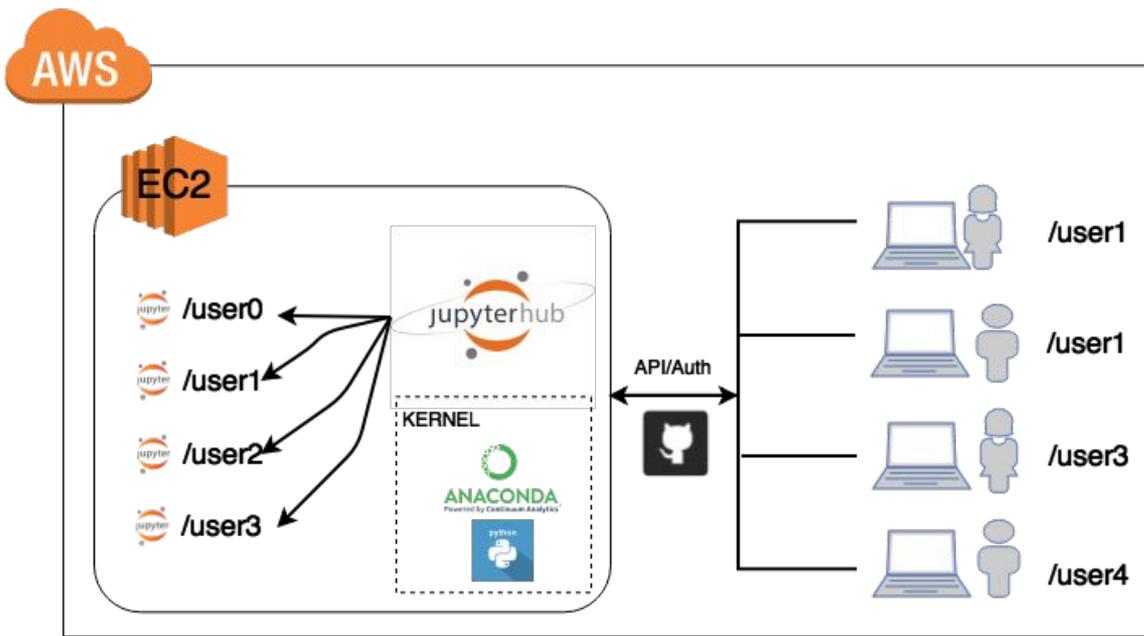
```
In [1]: from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin specgram routine:

```
In [2]: %matplotlib inline
from matplotlib import pyplot as plt
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
```



Проект Jupyter. JupyterHub.



Проекты Jupyter. JupyterLab.



File Edit View Run Kernel Tabs Settings Help

Python 3

In this Notebook we explore the Lorenz system of differential equations:

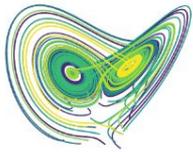
$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= -\beta z + xy\end{aligned}$$

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two called attractors.

```
In [4]: from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

Output View

sigma 10.00
beta 2.67
rho 28.00



lorezn.py

```
9 def solve_lorenz(N=10, max_time=4.0, sigma=10.0, beta=8./3, rho=28.0):
10     """Plot a solution to the Lorenz differential equations."""
11     fig = plt.figure()
12     ax = fig.add_axes([0, 0, 1, 1], projection='3d')
13     ax.axis('off')
14
15     # prepare the axes limits
16     ax.set_xlim((-25, 25))
17     ax.set_ylim((-35, 35))
18     ax.set_zlim((5, 55))
19
20     def lorenz_deriv(x,y,z, t0, sigma=sigma, beta=beta, rho=rho):
21         """Compute the time-derivative of a Lorenz system."""
22         x, y, z = x,y,z
23         return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]
24
25     # Choose random starting points, uniformly distributed from -15
26     np.random.seed(1)
27     x0 = -15 + 30 * np.random.random((N, 3))
28
```

Clusters Account: user@qubole.com Help

Menu Bar

File Edit View Run Kernel Tabs Spark Settings Help

Users / @qubole.c

new.ipynb

Left Sidebar

PySpark Spark SparkR Python

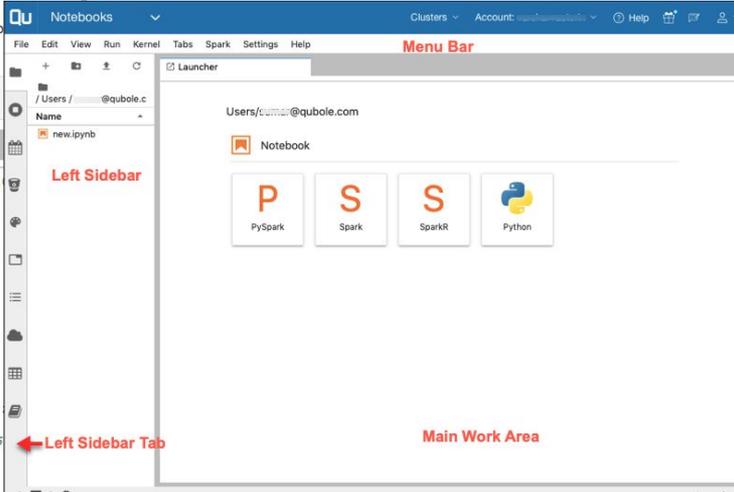
← Left Sidebar Tab

Main Work Area

Launcher

Users/cuman@qubole.com

Notebook

The image shows the JupyterLab Launcher interface. It features a top menu bar with options like File, Edit, View, Run, Kernel, Tabs, Spark, Settings, and Help. Below the menu bar is a sidebar containing a file browser and a list of users. The main area displays a grid of application launchers for PySpark, Spark, SparkR, and Python. A red arrow points to the sidebar, and another red arrow points to the main work area.