МИНОБРНАУКИ РОССИИ
ФГБОУ ВО «СГУ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

ОТЧЁТ ПО ПРАКТИЧЕСКИМ ЗАДАНИЯМ ПО КУРСУ «НЕИРОННЫЕ СЕТИ»

ЛАБОРАТОРНАЯ РАБОТА

студента 4 курса 431 группы
специальности 10.05.01 — Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Гущина Андрея Юрьевича

Проверил
доцент _____ И. И. Слеповичев

Саратов 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задания выполнены на языке программирования Rust с использованием библиотек

- `clap`
- `serde`
- `serde_json`
- `burn`
- `xml`

Для сборки задания N необходимо перейти в директорию `nntaskN` и выполнить команду

```
cargo build --release
```

В результате сборки загрузятся соответствующие библиотеки и появится директория `target` в текущей директории. В ней будет находиться исполняемый файл `nntaskN` (или `nntaskN.exe` на Windows). Примеры запуска каждого из заданий приведены ниже.

1 Создание ориентированного графа

Вход

Текстовый файл с описанием графа в виде списка дуг:

$$(a_1, b_1, n_1), (a_2, b_2, n_2), \ldots, (a_k, b_k, n_k),$$

где a_i — начальная вершина дуги i, b_i — конечная вершина дуги i, n_i — порядковый номер дуги в списке всех заходящих в вершину b_i дуг.

Выход

— Ориентированный граф с именованными вершинами и линейно упорядоченными дугами (в соответствии с порядком из текстового файла), либо
— Сообщение об ошибке в формате файла, если ошибка присутствует.

Пример исполнения программы

Пусть задан файл `tests/t1_input.txt` со следующим содержимым:

```
(f, b, 1), (b, a, 2), (d, c, 4), (e, c, 5), (c, a, 3)
```

При запуске программы командой

```
./nntask1 --input1 ./tests/t1_input.txt --output1 ./tests/t1_output.xml
```

в результате получим файл `tests/t1_output.xml`

```
<graph>
  <vertex>a</vertex>
  <vertex>b</vertex>
  <vertex>c</vertex>
  <vertex>d</vertex>
  <vertex>e</vertex>
  <vertex>f</vertex>
  <arc>
    <from>f</from>
    <to>b</to>
    <order>1</order>
  </arc>
  <arc>
    <from>b</from>
    <to>a</to>
    <order>2</order>
```

```
17    </arc>
18    <arc>
19      <from>c</from>
20      <to>a</to>
21      <order>3</order>
22    </arc>
23    <arc>
24      <from>d</from>
25      <to>c</to>
26      <order>4</order>
27    </arc>
28    <arc>
29      <from>e</from>
30      <to>c</to>
31      <order>5</order>
32    </arc>
33  </graph>
```

2 Создание функции по графу

Вход

Ориентированный граф с именованными вершинами как описано в задании 1.

Выход

— Линейное представление функции, реализуемой графом в префиксной скобочной записи, либо
— Сообщение об ошибке в формате файла, если ошибка присутствует, либо
— Сообщение о наличии в графе циклов.

Пример исполнения программы

Пусть задан файл `tests/t1_output.xml` со следующим содержимым:

```
<graph>
  <vertex>a</vertex>
  <vertex>b</vertex>
  <vertex>c</vertex>
  <vertex>d</vertex>
  <vertex>e</vertex>
  <vertex>f</vertex>
  <arc>
    <from>f</from>
    <to>b</to>
    <order>1</order>
  </arc>
  <arc>
    <from>b</from>
    <to>a</to>
    <order>2</order>
  </arc>
  <arc>
    <from>c</from>
    <to>a</to>
    <order>3</order>
  </arc>
  <arc>
    <from>d</from>
    <to>c</to>
    <order>4</order>
```

```
27    </arc>
28    <arc>
29      <from>e</from>
30      <to>c</to>
31      <order>5</order>
32    </arc>
33  </graph>
```

При запуске программы командой

```
1  ./nntask2 --input1 ./tests/t1_output.xml --output1 ./tests/t2_output.txt
```

в результате получим файл `tests/t2_output.txt`

```
1  a(b(), c(d(), e()))
```

Если в качестве ввода задан граф с циклом (файл `tests/t2_input_cycle.xml`):

```
1  <graph>
2    <vertex>a</vertex>
3    <vertex>b</vertex>
4    <vertex>c</vertex>
5    <vertex>d</vertex>
6    <vertex>e</vertex>
7    <arc>
8      <from>a</from>
9      <to>b</to>
10     <order>1</order>
11   </arc>
12   <arc>
13     <from>a</from>
14     <to>c</to>
15     <order>2</order>
16   </arc>
17   <arc>
18     <from>c</from>
19     <to>d</to>
20     <order>3</order>
21   </arc>
22   <arc>
23     <from>c</from>
24     <to>e</to>
25     <order>4</order>
```

```
26    </arc>
27    <arc>
28      <from>e</from>
29      <to>c</to>
30      <order>5</order>
31    </arc>
32  </graph>
```

То при запуске программа выведет сообщение:

```
1  Некорректный ввод - в графе есть циклы
```

3 Вычисление значения функции на графе

Вход

1. Текстовый файл с описанием графа в виде списка дуг (задание 1).
2. Текстовый файл соответствий арифметических операций именам вершин:

```
{
    "a_1": "операция_1",
    "a_2": "операция_2",
    ...,
    "a_n": "операция_n",
}
```

где a$_i$ $- - - i -$, $_i$ $- -$, a_i.

Допустимы следующие символы операций:

— «+» — сумма значений,

— «*» — произведение значений,

— «exp» — экспонирование входного значения,

— «число» — любая числовая константа.

Выход

Значение функции, построенной по графу (1) и файлу (2).

Пример исполнения программы

Пусть задан файл `tests/t1_output.xml` с некоторым графом:

```
<graph>
  <vertex>a</vertex>
  <vertex>b</vertex>
  <vertex>c</vertex>
  <vertex>d</vertex>
  <vertex>e</vertex>
  <vertex>f</vertex>
  <arc>
    <from>f</from>
    <to>b</to>
    <order>1</order>
  </arc>
  <arc>
    <from>b</from>
    <to>a</to>
```

```
16      <order>2</order>
17    </arc>
18    <arc>
19      <from>c</from>
20      <to>a</to>
21      <order>3</order>
22    </arc>
23    <arc>
24      <from>d</from>
25      <to>c</to>
26      <order>4</order>
27    </arc>
28    <arc>
29      <from>e</from>
30      <to>c</to>
31      <order>5</order>
32    </arc>
33  </graph>
```

Также задан файл `tests/t3_ops.json` с соответствием операций вершинам:

```
1  {
2    "a": "+",
3    "b": "exp",
4    "c": "*",
5    "d": 5.0,
6    "e": 9.0,
7    "f": 2.0
8  }
```

При запуске программы командой

```
1  ./nntask3 --input1 ./tests/t1_output.xml --input2 ./tests/t3_ops.json
   ↪  --output1 ./tests/t3_output.txt
```

в результате получим файл `tests/t3_output.txt`

```
1  52.38905609893065
```

4 Построение многослойной нейронной сети

Вход

1. Текстовый файл с набором матриц весов межнейронных связей в формате:

```
{
    "weights": [
        [
            [M1_11, M1_12, ..., M1_1n],
            ...
            [M1_m1, M1_m2, ..., M1_mn]
        ],
        ...,
        [
            [Mp_11, Mp_12, ..., Mp_1n],
            ...
            [Mp_m1, Mp_m2, ..., Mp_mn]
        ]
    ]
}
```

2. Текстовый файл с входным вектором в формате:

```
x_1, x_2, ..., x_n
```

Выход

1. Сериализованная многослойная нейронная сеть (в формате XML или JSON) с полносвязной межслойной структурой.
2. Файл с выходным вектором – результатом вычислений НС в формате:

```
y_1, y_2, ..., y_n
```

3. Сообщение об ошибке, если в формате входного вектора или файла описания НС допущена ошибка.

Пример исполнения программы

Пусть задан файл `tests/t4_w.json` с весами НС:

```
{
    "weights": [
        [
            [0.47519493033675375, 0.015705490366171526, 0.9433818257724572],
            [0.48092032736144574, 0.13929695479782134, 0.6869903232566065],
            [0.436988975888717, 0.20037642195993755, 0.17561406275527947]
```

```
        ],
        [
            [0.042224071742743785, 0.15331022315027187, 0.464635658411239],
            [0.6000159964796773, 0.22606113281552231, 0.5301212736820182],
            [0.19651133783303198, 0.7498835958139106, 0.28721556978456597]
        ],
        [
            [0.11837615025116721, 0.00927217999098906, 0.7504596929897048],
            [0.5675946231090779, 0.9748635791740536, 0.30501309542663524],
            [0.8574872089946126, 0.3047120321509168, 0.3376899733092712]
        ]
    ]
}
```

Также задан файл `tests/t4_x.txt` с входным вектором:

```
1, 2, 3
```

Для использования программы необходимо сконвертировать заданный файл с весами в модель, которую может использовать библиотека burn.

Для этого необходимо запустить подкоманду `convert`:

```
./nntask4 --weights ./tests/t4_w.json --output ./tests/t4_model.json
```

После этого, модель можно использовать для вычислений НС:

```
./nntask4 run --model ./tests/t4_model.json --input ./tests/t4_x.txt --output
↪  ./tests/t4_output.txt
```

В результате работы программы получим файл `tests/t4_output.txt`

```
[0.6565478, 0.69785804, 0.7188464]
```

а также серилизованную модель `tests/t4_model.json`

```
{
  "metadata": {
    "float": "f32",
    "int": "f32",
    "format":
    ↪  "burn_core::record::file::PrettyJsonFileRecorder<burn_core::record::settings:
    "version": "0.11.1",
    "settings": "FullPrecisionSettings"
  },
```

```
  "item": {
    "layers": [
      {
        "weight": {
          "id": "1a77f782-9729-48f4-ae6c-feb4b098efa4",
          "param": {
            "value": [
              0.47519493,
              0.4809203,
              0.43698898,
              0.01570549,
              0.13929695,
              0.20037642,
              0.94338185,
              0.6869903,
              0.17561406
            ],
            "shape": [
              3,
              3
            ]
          }
        },
        "bias": null
      },
      {
        "weight": {
          "id": "149e9eae-09c6-4ded-9ec4-dd2dbbb13942",
          "param": {
            "value": [
              0.042224072,
              0.15331022,
              0.46463567,
              0.600016,
              0.22606114,
              0.53012127,
              0.19651134,
              0.7498836,
              0.28721556
            ],
            "shape": [
```

```
              3,
              3
            ]
          }
        },
        "bias": null
      },
      {
        "weight": {
          "id": "b85dec71-9c34-4aca-94c5-23cce7439c1b",
          "param": {
            "value": [
              0.11837615,
              0.56759465,
              0.8574872,
              0.00927218,
              0.9748636,
              0.30471203,
              0.7504597,
              0.3050131,
              0.33768997
            ],
            "shape": [
              3,
              3
            ]
          }
        },
        "bias": null
      }
    ]
  }
}
```