

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

09.03.04 - Программная инженерия
Профиль направления подготовки бакалавриата
«Системное и прикладное программное обеспечение»

ОТЧЁТ О ПРОХОЖДЕНИИ ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ

Выполнил:

студент 2 курса группы 22207

Афанасьев Артём Игоревич

Место прохождения практики:

Кафедра информатики и математического
обеспечения

Сроки прохождения практики:

30.05-09.06

Руководитель практики:

к.т.н., доцент

Богоявленская Ольга Юрьевна

Оценка _____

Дата _____

Содержание

Введение	3
1 Модуль статистики суффиксов	4
1.1 Методы класса SuffixStat	4
2 Тестирование	7
Заключение	8
Список литературы	9

Введение

В наши дни идея генерации текста по математическим моделям набирает всё большую популярность в повседневной жизни. Современные решения, как правило, основываются на колоссальных нейронных сетях, включающих в себя триллионы нейронов. Но, если ограничиться генерацией текста по статистическим моделям, то данная задача решается с помощью алгоритма на основе Марковских цепей [1].

Цель практики - реализовать алгоритм построения статистики суффиксов и префиксов по заданным текстам.

Задачи производственной практики:

1. Ознакомление с теорией и литературой по генерации текстов с помощью Марковских цепей;
2. Создание собственной структуры данных для хранения префиксов текста на C++ и её последующая интеграция в Python;
3. Создание программного модуля по подсчёту и анализу префиксов и суффиксов в тексте;
4. Тестирование разработанного модуля и структуры данных.

Организация и кооперация с другими разработчиками данной задачи (Кирилловым Иваном и Павлов Максим), а также контроль версий программного кода и распределение подзадач осуществлялось с помощью GitHub.

1 Модуль статистики суффиксов

Для подсчёта статистик и анализа суффиксов в текстах был разработан python-класс `SuffixStat`. При инициализации достаточно указать строку с текстом (можно и с многострочным) в первом аргументе, и кол-во слов в префиксе во втором, чтобы найти соответствующие суффиксы после них.

Листинг 1: Пример использования `SuffixStat`

```
from python_code.suffix_statistics import SuffixStat

example = SuffixStat("Test_text_1", 2)
example.add("Test_text_2", 3)
print(example.most_common_in_text(0, 10))
```

1.1 Методы класса `SuffixStat`

1. Добавление текста в структуру для анализа

`add(text, k) -> None`

`text` - строка исходного текста, `k` - кол-во слов в префиксе

Листинг 2: Пример использования `SuffixStat`

```
def add(self, text, k) -> None:
    assert (type(text) == str)
    self.stat.append(StatisticM.statistic\_counter())
    self.text.append(list(filter(lambda word: word != ' ', \
                                text.translate(str.maketrans(' ', ' ', \
                                string.punctuation)) \
                                .replace('\t', ' ') \
                                .replace('\n', ' ') \
                                .split(' '))))
    index = len(self.stat)-1
    for cur in range(len(self.text[index]) - k):
        suffix = self.text[index][cur+k]
        self.stat[index].add(suffix.lower())
```

2. Топ n по частоте суффиксов в тексте index

most_common_in_text(self, index, n) -> List[List]

Листинг 3: Пример использования SuffixStat

```
def most\_common\_in\_text(self, index, n) -> List:
    if (type(index) != int or type(n) != int or n < 1 or
        index < 0 or index > len(self.stat)-1):
        return []
    self.stat[index].set\_pointer(0)
    arr = []
    s = self.stat[index].get\_next()
    prev = -1
    count = -1
    while s != "":
        if int(s.split('_')[-1]) != prev:
            arr.append([])
            prev = int(s.split('_')[-1])
            count += 1
        if count == n:
            break
        arr[count].append(s.split('_')[:-1][0])
        s = self.stat[index].get\_next()
    self.stat[index].set\_pointer(0)
    return list(filter(lambda x: x != [], arr))
```

3. Средняя частота встречаемости заданного суффикса в текстах

mean_frequency_of_suffix_occurrence(self, suffix) -> float

Листинг 4: Пример использования SuffixStat

```
def max\_frequency\_of\_suffix\_occurrence(self, suffix) -> int:
    if type(suffix) != str:
        return 0
    max\_n: int = 0
    for text in self.stat:
        if text.get\_by\_pref(suffix) > max\_n:
```

```

        max\_n = text.get\_by\_pref(suffix)
    return max\_n

```

4. Масимальная частота употребления заданного суффикса в текстах

```

max_frequency_of_suffix_occurrence(self, suffix) -> int

```

Листинг 5: Пример использования SuffixStat

```

def mean\_frequency\_of\_suffix\_occurrence(self, suffix) -> float:
    if type(suffix) != str:
        return 0.0
    arr = []
    for text in self.stat:
        arr.append(text.get\_by\_pref(suffix))
    return sum(arr) / len(arr)

```

2 Тестирование

Тестирование является важной и неотъемлемой частью разработки любого программного кода. Для проверки работы разработанного модуля статистик суффиксов были написаны 16 тестов.

```
===== test session starts =====  
collecting ... collected 3 items  
  
test.py::test_statistic_class PASSED [ 33%]  
test.py::test_prefix_statistic PASSED [ 66%]  
test.py::test_suffix_statistic PASSED [100%]  
  
===== 3 passed in 0.64s =====  
  
Process finished with exit code 0
```

Рис. 1: Тестирование разработанного программного кода и структуры данных

Заключение

В результате производственной практики была поставлена и достигнута задача. Разработанный модуль статистики суффиксов для последующей генерации текстов на его основе. Созданный в процессе производственной практики программный код, его тесты и документация доступны на GitHub для использования в дальнейших задачах.

Список литературы

1. Керниган, Брайан У., Пайк, Роб. Практика программирования. : Пер. с англ. - М. : ООО "И.Д. Вильямс -288 с.