Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Отчет о прохождении производственной практики

Генерация текста без использования статистики префиксов

Выполнил студент 2 курса группы 22207:
Гагарин Валерий Владимирович

Направление подготовки бакалавриата:
09.03.04 Программная инженерия

Место прохождения практики:
Институт математики и информационных технологий

Период прохождения практики:
29.05.23 - 09.06.23

Руководитель:
Богоявленская Ольга Юрьевна, к.т.н, доцент

Оценка:

_____
оценка

Дата:

_____
дата

Петрозаводск — 2023

Содержание

Введение

Теория

Цепи Маркова — это математический инструмент, который помогает моделировать случайные процессы, где будущее зависит только от текущего состояния, а не от того, как оно сюда попало. В генерации текстов это означает, что мы используем тексты, чтобы определить, как вероятнее всего будет продолжаться предложение или текст.
В качестве основы для генерации текста была выбрана матрица переходов. Этот метод позволяет строить текст, используя вероятности переходов между словами или символами, без необходимости учитывать статистику префиксов.

Цель практики

Реализовать генерацию текста без использования статистики префиксов.

Задачи практики

1. Чтение документации по генерации текстов с помощью цепей Маркова.
2. Реализация генерации текста без использования статистики префиксов.
3. Тестирование разработонного алгоритма.

Модуль построения матрицы перехода и генерации текста

Листинг

Листинг 1: модуль TransitionMatrix, строящий матрицу перехода и генерирующий текст на ее основе

```
import numpy

class TransitionMatrix:
    """класс
    , реализующийгенерациютекстаспомощьюцепеймаркова
    """
    def __init__(self, corpus, length_prefix):
        """ инициализацияклассa  """
        self.corpus = corpus
        self.length_prefix = length_prefix
        self.prefixes = None
        self.matrix = None

    def create_prefixes(self):
        """формированиепарпрефикс
           - суффиксизразбитогонасловаисходноготекста
        """
        self.first_prefix = None
        self.prefixes = {}
        for index in range(0, len(self.corpus) - self.length_prefix + 1):
            prefix = tuple(self.corpus[index:index + self.length_prefix])
            if len(self.corpus) <= index + self.length_prefix:
                suffix = None
            else:
                suffix = self.corpus[index + self.length_prefix]
            self.add_prefix(prefix, suffix)
            if self.first_prefix == None:
                self.first_prefix = prefix
        return self.prefixes

    def add_prefix(self, prefix, suffix):
        """сохранениеколичествасуффиксовсопределеннымпрефиксомвсловарь
```

```
        """
        if prefix not in self.prefixes:
            self.prefixes[prefix] = {}
        if suffix not in self.prefixes[prefix]:
            self.prefixes[prefix][suffix] = 0
        self.prefixes[prefix][suffix] += 1

    def make_matrix(self):
        """построениематрицыперехода

        """
        if self.prefixes == None:
            self.create()
        self.matrix = {}
        for prefix, suffixes in self.prefixes.items():
            total_count = sum(suffixes.values())
            self.matrix[prefix] = {suffix: count / total_count for suffix, count i
        return self.matrix

    def generate_text_by_matrix(self):
        """генерациятекстаспомощьюматрицыперехода

        """
        if self.matrix == None:
            self.make_matrix()
        current_prefix = self.first_prefix
        result = list(current_prefix)
        while(True):
            suffixes = list(self.matrix[current_prefix].keys())
            probabilities = list(self.matrix[current_prefix].values())
            suffix = numpy.random.choice(suffixes, p=probabilities)
            if suffix == None:
                break
            else:
                result.append(str(suffix))
            current_prefix = tuple(list(current_prefix[1:]) + [suffix])
        return result
```

Методы

Инициализация

На вход принимается разбитый текст на массив слов и длина префикса.

create_prefixes

Метод, состовляющий пару - префикс: суффикс. Первый префикс, с которого начинается исходный текст и начнется сгенерированный, записываем в переменную. Концом текста является суффикс с значением None.

add_prefix

Сохраняем префикс и суффикс в словарь. Увеличиваем число появлений суффикса на 1.

make_matrix

Составление матрицы переходов. Проходим по каждому префиксу, получаем суффиксы и количество их появлений, считаем вероятность появления соответствующего суффикса после префикса.

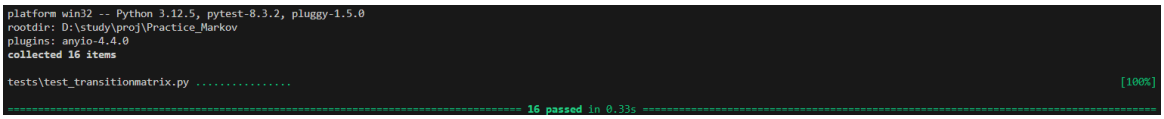
generate_text_by_matrix

Генерируем текст на основе матрицы перехода. Начинаем с первого префикса, получаем суффиксы и вероятности, с помощью взвешанного случайного выбора получаем суффикс и строим новый перфикс. Генерируем текст пока суффикс не окажется указателем конца текста.

Тестирование

Для проверки работоспособности программы были написаны 16 тестов: 13 позитивных и 3 негативных. Были протестрированы следующие случаи:

1. Наличие всех префиксов и суффиксов в матрице перехода.
2. Корректное распределение вероятностей в матрице переходов.
3. Проверка, что сумма вероятностей перехода префикса в суффикс равняется 1 для всех суффиксов.
4. Длина сгенерированного текста больше 0.
5. Ошибка генерация текста для пустого текстового корпуса.
6. Длина префикса должна быть числом.
7. Длина префикса не может быть больше длины текстового корпуса - 1. Все тесты были успешно пройдены. c0.9



Результат тестирования

Заключение

В ходе практики был реализован алгоритм генерации текста с помощью цепей Маркова с построением матрицы перехода. Код загружен на github и доступен по ссылке Transition matrix.

Список литературы

1. Керниган Б., Пайк Р. *Практика программирования*. 8-е изд. М., СПб., Киев: Издательский дом "Вильямс 2015. 288 с.

2. Краткое введение в цепи Маркова // Habr URL: `https://habr.com/ru/articles/455762/`

3. Генерация текста с помощью цепей Маркова // Блог Саши Беспоясова URL: `https://bespoyasov.ru/blog/text-generation-with-markov-chains/`