Київський національний університет імені
Тараса Шевченка

Звіт до лабораторної роботи №1

Застосування N-грам

Пядика Любомира, MI-4

26 листопада 2018 р.

1 Постановка задачі

Необхідно розв'язати задачу, яка складається із наступних етапів:

- Розбити корпус на речення, а речення - на слова.
- Побудувати N-грами та N-1-грами у межах кожного речення корпусу.
- Укласти частотні словники N-грам та N-1-грам.
- Оцінити параметри згладжування для алгоритму Віттена-Белла: для кожної (N-1)-грами знайти кількість типів N-грам, які можна утворити з даної (N-1)-грами у даному корпусі.
- Оцінити параметри згладжування для алгоритму Гуда-Тюрінга: 1) Побудувати частотний словник частот N-грам (тобто частота і скільки є типів N-грам, що мають цю частоту); 2) Оцінити згладжену частоту N-грам за формулою 6.29 в книжці Журавського на сторінці 239.[2] 3) Побудувати графік частот частот N-грам.

2 Теоретичні відомості

N-грама [1] — послідовність з n елементів. З семантичної точки зору, це може бути послідовність звуків, складів, слів або букв. На практиці частіше зустрічається N-грами як ряд слів, стійкі словосполучення називають колокацію. Послідовність з двох послідовних елементів часто називають біграм, послідовність з трьох елементів називається триграма. Не менш чотирьох і вище елементів позначаються як N-грами, N замінюється на кількість послідовних елементів.

Загальне використання N-грам N-грами в цілому знаходять своє застосування в широкій області наук. Вони можуть застосовуватися, наприклад, в галузі теоретичної математики, біології, картографії, а також в музиці. Найбільш часто використання N-грам включає наступні області:

- вилучення даних для кластеризації серії супутникових знімків Землі з космосу, щоб потім вирішити, які конкретні частини Землі на зображенні,
- пошук генетичних послідовностей,
- в області генетики використовуються для визначення того, з яких конкретних видів тварин зібрані зразки ДНК,
- в комп'ютерному стисненні,
- для індексування даних в пошукових системах; з використанням N-грам, як правило, індексовані дані, пов'язані зі звуком.

Використання N-грам для потреб обробки природної мови В області обробки природної мови N-грами використовується в основному для передбачення на основі імовірнісних моделей. N-програмних модель розраховує ймовірність останнього слова N-грами, якщо відомі всі попередні. При використанні цього підходу для моделювання мови передбачається, що поява кожного слова залежить тільки від попередніх слів.

Іншим застосуванням N-грам є виявлення плагіату. Якщо розділити текст на кілька невеликих фрагментів, представлених N-грамами, їх легко порівняти один з одним і таким чином отримати ступінь подібності аналізованих документів. N-грами часто успішно використовуються для категоризації тексту і мови. Крім того, їх можна використовувати для

створення функцій, які дозволяють отримувати знання з текстових даних. Використовуючи N-грами, можна ефективно знайти кандидатів, щоб замінити слова з помилками.

Методи для вилучення N-грам У зв'язку з частим використанням N-грам для вирішення різних завдань необхідний надійний і швидкий алгоритм для вилучення їх з тексту. Відповідний інструмент для вилучення N-грам повинен бути в змозі працювати з необмеженим розміром тексту, працювати швидко і ефективно використовувати наявні ресурси. Є кілька методів вилучення N-грам з тексту. Ці методи засновані на різних принципах: Алгоритм Nagao 94 для текстів на японському, Алгоритм Лемпеля-Зива-Велч, Суффіксний масив, суфіксне дерево, інвертований індекс.

3 Програмна реалізація

Лабораторну роботу було написано на мові програмування Python. Програма складається із 2 файлів (main.py - головний файл, який виконує аналіз тексту і будує графік; count_dict.py - реалізовує клас, який читає текст, і будує необхідні структури із нього, як словники N-1, N-грам, і уніграм тощо).

Для тестування було використано різні тексти, у тому числі "Bible" і "Alice in the Wonderland", які доступні у публічному доступі в інтернеті.

3.1 Приклад виконання програми

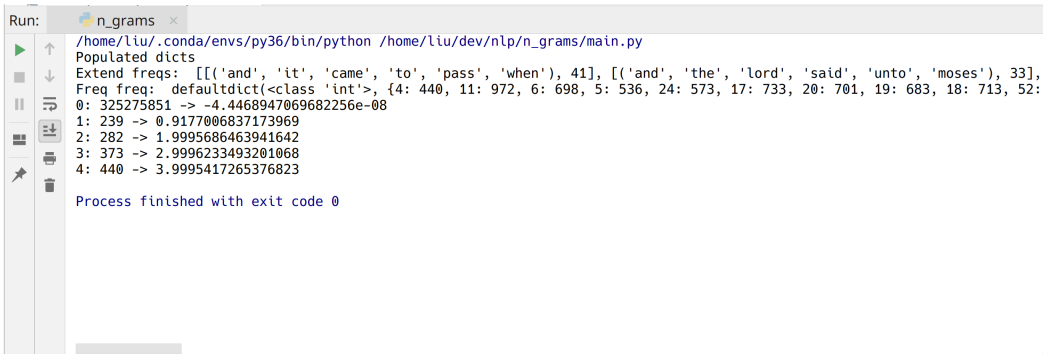

Рис. 1: Запущений main.py файл у PyCharm

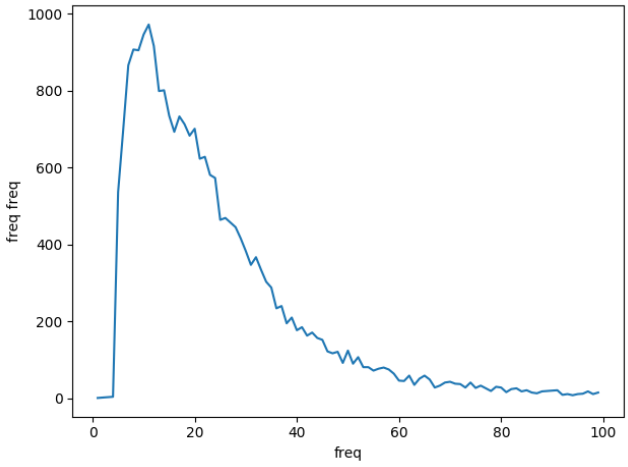

Рис. 2: Графік частоти частот N-грам при N=7 на тексті Біблії

3.2 Лістинг програми

main.py

```
from collections import defaultdict

import numpy
from matplotlib import pyplot

from n_grams.count_dict import CountDict


def main():
    train_path = "../data/bible.txt"
    # train_path = "../data/alice_in_wonderland.txt"
    smoothing = "laplace"

    # читаємо навчальний текст і
    # обчислюємо для нього N-1 та N-грами
    counts = CountDict(train_path, n=7)
    counts.populate()
    print("Populated dicts")

    # оцінюємо параметри згладжування для алгоритму Віттена-Белла:
    # знаходимо для кожної N-1-грами кількість типів N-грам,
    # які можна утворити з даної N-1-грами у даному корпусі
    extend_freqs = defaultdict(int)
    for n1_gram in counts.n1_grams.keys():
        extend_freqs[n1_gram] = 0
    for n2_gram in counts.n2_grams.keys():
        n1_gram = n2_gram[:-1]
        if n1_gram in counts.n1_grams:
            extend_freqs[n1_gram] += 1

    extend_freqs = [[key, count] for key, count in extend_freqs.items()]
    extend_freqs = sorted(extend_freqs, key=lambda x: x[1], reverse=True)
    print("Extend freqs: ", extend_freqs[:50])

    # оцінюємо параметри згладжування для алгоритму Гуда-Тюрінга
    # будуємо частотний словник частот N-грам (тобто частота і скільки є типів N-грам, щ
    freq_freqs = defaultdict(int)
    for freq in counts.n2_grams.values():
        freq_freqs[freq] += 1
    freq_freqs[0] = counts.unique_n1_grams * counts.unique_unigrams - counts.unique_n2_g
    print("Freq freq: ", freq_freqs)

    # поріг Катца (Katz threshold)
    k = 5
    for i in range(k):
        old_freq = freq_freqs[i]
        freq_freqs[i] = (((i + 1) * (freq_freqs[i + 1]) / (freq_freqs[i])) -
                         ((freq_freqs[k + 1] * i * (k + 1)) / freq_freqs[1])) /\
```

```
                        (1 - (freq_freqs[k + 1] * (k + 1)) / (freq_freqs[1]))
            print('{}: {} -> {}'.format(i, old_freq, freq_freqs[i]))

    bounds = [1, 100]
    x = numpy.arange(*bounds, 1)
    y = [freq_freqs[x] for x in x]

    pyplot.plot(x, y)
    pyplot.xlabel('freq')
    pyplot.ylabel('freq freq')
    pyplot.show()


if __name__ == '__main__':
    main()
```

count_dict.py

```
from collections import defaultdict
import re

# допоміжний клас для зчитування навчального тексту і
# обчислення для нього N-1 та N-грам
class CountDict(object):
    def __init__(self, filename, n):
        self.filename = filename
        self.n1 = n-1
        self.n2 = n

        # ініціалізуємо словники N-1 та N-грам
        self.unigrams = defaultdict(int)
        self.n1_grams = defaultdict(int)
        self.n2_grams = defaultdict(int)

    def populate(self):
        # регулярний вираз символів, які будуть вилучаютися
        to_remove = re.compile("(,)|(;)|(:)|(!)|(\?)|(\")|(\()|(\))|(\[)|(\])|(\s-)")

        # зчитуємо навчальний файл порядково
        with open(self.filename) as f:
            for line in f:
                line = line.lower()
                sentences = line.split('.')

                # виділяємо окремо речення, і структурні елементи речень,
                # щоб уникнути помилкових рекомендацій
                for sentence in sentences:
                    # переводимо рядок до нижнього регістру
                    # і вилучаємо символи за допомогою попередньої регулярки
                    words = to_remove.sub("", sentence)

                    # розділяємо рядок на токени (слова)
```

```
                tokens = words.split()

                # заповнюємо словник уніграм,
                # підраховуючи кільки кожної
                for word in tokens:
                    self.unigrams[(word,)] += 1

                # заповнюємо словник N-1-грам,
                # підраховуючи кільки кожної
                for i in range(len(tokens) - self.n1 + 1):
                    n1_gram = tuple()
                    for j in range(self.n1):
                        n1_gram = n1_gram + (tokens[j], )
                    self.n1_grams[n1_gram] += 1

                # заповнюємо словник N-грам,
                # підраховуючи кільки кожної
                for i in range(len(tokens) - self.n2 + 1):
                    n2_gram = tuple()
                    for j in range(self.n2):
                        n2_gram = n2_gram + (tokens[j], )
                    self.n2_grams[n2_gram] += 1

        # к-ть унікальних уніграм та біграм
        self.unique_unigrams = len(self.unigrams.keys())
        self.unique_n1_grams = len(self.n1_grams.keys())
        self.unique_n2_grams = len(self.n2_grams.keys())
```

Література

[1] N-gram. https://en.wikipedia.org/wiki/N-gram.

[2] Speach and language processing. http://www.deepsky.com/~merovech/voynich/voynich_manchu_reference_materials/PDFs/jurafsky_martin.pdf.