Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Отчёт
по практической работе №1 «Вычисление энтропии Шеннона»

Выполнил:
студент группы ИП-014
Бессонов А.О.

Работу проверил:
старший преподаватель
Дементьева К.И.

Новосибирск 2024 г.

Постановка задачи

Цель работы: Экспериментальное вычисление оценок энтропии Шеннона текстов. Изучение свойств энтропии Шеннона.

Задание:

1. Для выполнения работы потребуются три текстовых файла с различными свойствами. Объем файлов больше 10 Кб, формат txt. В первом файле содержится последовательность символов, количество различных символов больше 2 (3,4 или 5). Символы **последовательно и независимо** с равными вероятностями генерируются с помощью датчика псевдослучайных чисел и записываются в файл.

Для генерации второго файла необходимо сначала задать набор вероятностей символов (количество символов такое же, как и в первом файле), а затем **последовательно и независимо** генерировать символы с соответствующей вероятностью и записывать их в файл, вероятности в процессе записи файла не меняются.

В качестве третьего файла необходимо выбрать художественный текст на русском (английском) языке. Для алфавита текста предполагается, что строчные и заглавные символы не отличаются, знаки препинания опущены, к алфавиту добавлен пробел, для русских текстов буквы «е» и «ё», «ь» и «ъ» совпадают.

2. Составить программу, определяющую несколько оценок энтропии созданных текстовых файлов. Вычисление значения по формуле Шеннона **настоятельно рекомендуется** оформить в виде отдельной функции, на вход которой подается массив (список) вероятностей символов, выходной параметр – значение, вычисленное по формуле Шеннона.

Вычислить три оценки энтропии Шеннона для каждого из файлов. Рекомендуется вычисление оценки оформить в виде отдельной функции с параметром имя файла:

Первая оценка H_1. Сначала определить частоты отдельных символов файла, т.е. отношения количества отдельного символа к общему количеству символов в файле. Далее используя полученные частоты как оценки вероятностей, рассчитать оценку энтропии по формуле Шеннона.

Вторая оценка H_2. Определить частоты всех последовательных пар символов в файле. Для того правильной оценки энтропии H_2 пары символов нужно рассматривать с перехлестом.

Пример. Пусть имеется такая последовательность ФЫВАФПРО Под парами понимаются пары соседних символов, т.е.

ФЫ ЫВ ВА АФ ФП ПР РО

Для подсчета оценки энтропии H_2 необходимо подсчитать частоту каждой пары символов и подставить в формулу Шеннона. Полученное значение оценки энтропии следует разделить на 2.

Третья оценка H_3. Определить частоты всех последовательных троек символов в файле. Для того правильной оценки энтропии H_3 **тройки** символов нужно рассматривать с перехлестом.

Для подсчета оценки энтропии H_3 необходимо подсчитать частоту каждой тройки символов и подставить в формулу Шеннона. Полученное значение оценки энтропии следует разделить на 3.

По желанию можно продолжить процесс вычисления оценок с использованием частот четверок, пятерок символов и т.д.

3. После тестирования программы необходимо заполнить таблицу для отчета и в отчете **проанализировать** полученные результаты, объяснить

замеченные эффекты. Для получения теоретических значений энтропии использовать наборы вероятностей, которые использовались при генерации файлов, для файла с текстом на естественном языке не заполнять.

4. Оформить отчет, должен содержать заполненную таблицу и анализ полученных результатов, по желанию в отчет можно включить описание программной реализации. **В отчет не нужно включать содержимое этого файла.** Загрузить отчет в электронную среду.

5. Анализ полученных результатов можно оформить в виде ответов на вопросы

Ход работы

Для первого файла был выбран алфавит:

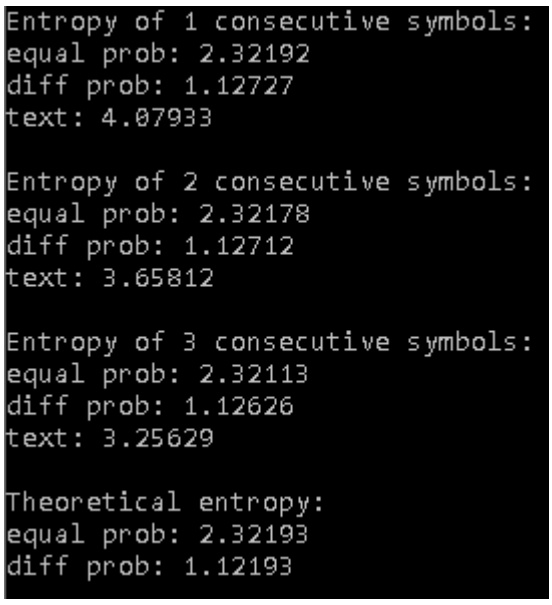
Символы	a	b	c	d	e

Для второго файла был выбран следующий набор вероятностей символов:

Символы	a	b	c	d	e
Вероятности	0.80	0.05	0.05	0.05	0.05

Для третьего файла был выбран художественный текст на английском языке «Красавица и Чудовище».

Результаты работы программы:



Название файла	H_1	H_2	H_3	Максимально возможное значение энтропии	Теоретическое значение энтропии
Файл 1	2.32192	2.32178	2.32113	2.32192	2.32193
Файл 2	1.12727	1.12712	1.12626	1.12727	1.12193
Файл 3	4.07933	3.65812	3.25629	4.07933	-

Выводы

Первый файл состоит из символов алфавита, которые сгенерированы равновероятно. Второй файл состоит из символов алфавита, которые сгенерированы с определенными вероятностями. Третий файл — это художественный текст, в котором символы имеют осмысленную структуру.

Первый и второй файлы имеют одинаковый алфавит, поэтому имеет смысл сравнения оценок энтропии этих файлов. Поскольку в первом файле символы равновероятны, то энтропия этого файла больше, чем у второго, где символы не равновероятны и один из них имеет огромную вероятность по сравнению с другими. Это связано с тем, что энтропия отражает меру неопределенности. Так же можно заметить, что фактические значения энтропии максимально похожи на теоретические значения.

По полученным результатам видно, что чем более длинную последовательность подряд идущих символов мы берем для вычисления энтропии, тем меньше энтропия становится. Для первых двух файлов значения хоть и уменьшаются, но не значительно по сравнению с третьим файлом, который является художественным текстом. Это связано с тем, что последовательности символов в тексте осмыслены и представляют собой слова и предложения.

Код программы

```
#include <cmath>
#include <fstream>
#include <iostream>
#include <list>
#include <map>
#include <random>
#include <vector>

using namespace std;

char check_char(char c) {
    if (!isalpha(c) && !isdigit(c) && c != ' ')
        return -1;

    if (isalpha(c))
        return tolower(c);

    return c;
}

void check_probabilities(vector<double> probabilities) {
    double remains = 1;
    int index_of_max = 0;

    for (int i = 0; i < probabilities.size(); i++) {
        remains -= probabilities[i];
        if (probabilities[i] > probabilities[index_of_max])
            index_of_max = i;
    }

    probabilities[index_of_max] += remains;
}

void generate_file_equal_prob(vector<char> alphabet) {
    ofstream file("equal_prob.txt");

    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<int> distribution(0, alphabet.size() - 1);

    for (long i = 0; i < 1 << 15; i++) {
        file << alphabet[distribution(gen)];
    }

    file.close();
}

void generate_file_diff_prob(vector<char> alphabet, vector<double>
alphabet_probabilities) {
    ofstream file("diff_prob.txt");

    random_device rd;
    mt19937 gen(rd());
    discrete_distribution<> distribution(alphabet_probabilities.begin(),
alphabet_probabilities.end());

    for (long i = 0; i < 1 << 15; i++) {
        file << alphabet[distribution(gen)];
    }
```

```
    file.close();
}

double calculate_entropy(vector<double> probabilities) {
    double entropy = 0;
    for (double probability : probabilities)
        entropy += probability * log2(probability);

    return -entropy;
}

double get_evaluation(string file_name, int limit) {
    if (limit < 1 || limit > 100) return -1;

    map<string, long> alphabet;
    list<char> buffer;
    long counter = 0;

    ifstream file(file_name);

    char c;
    while (file.get(c)) {
        if (check_char(c) != -1) {
            if (buffer.size() == limit) {
                buffer.pop_front();
            }

            buffer.push_back(check_char(c));

            if (buffer.size() == limit) {
                string str = "";
                for (char symbol : buffer)
                    str += symbol;

                alphabet[str]++;
                counter++;
            }
        }
    }

    file.close();

    vector<double> probabilities;

    for (const auto& symbol : alphabet) {
        probabilities.push_back((double)symbol.second / counter);
        //cout << symbol.first << " " << symbol.second << "\n";
    }

    check_probabilities(probabilities);

    return calculate_entropy(probabilities) / limit;
}

int main() {
    vector<char> alphabet = { 'a', 'b', 'c', 'd', 'e' };
    vector<double> alphabet_probabilities = { 0.80, 0.05, 0.05, 0.05, 0.05 };

    if (alphabet.size() != alphabet_probabilities.size()) {
        cout << "ERROR";
        return -1;
    }
```

```
    check_probabilities(alphabet_probabilities);

    generate_file_equal_prob(alphabet);
    generate_file_diff_prob(alphabet, alphabet_probabilities);

    for (int i = 1; i <= 3; i++) {
        cout << "Entropy of " << i << " consecutive symbols:\n";
        cout << "equal prob: " << get_evaluation("equal_prob.txt", i) << "\n";
        cout << "diff prob: " << get_evaluation("diff_prob.txt", i) << "\n";
        cout << "text: " << get_evaluation("Beauty_and_the_Beast.txt", i) <<
"\n\n";
    }

    vector<double> alphabet_prob_equal;
    for (char c : alphabet)
        alphabet_prob_equal.push_back((double)1 / alphabet.size());
    check_probabilities(alphabet_prob_equal);

    cout << "Theoretical entropy:\n";
    cout << "equal prob: " << calculate_entropy(alphabet_prob_equal) << "\n";
    cout << "diff prob: " << calculate_entropy(alphabet_probabilities) << "\n";

    return 0;
}
```