МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования  
«Белорусский государственный технологический университет»  
  
  
  
  
  
  
Отчёт по лабораторной работе №3

ЭЛЕМЕНТЫ ТЕОРИИ ИНФОРМАЦИИ. ИНФОРМАТИВНОСТЬ ДАННЫХ

В РАЗЛИЧНЫХ КОДИРОВКАХ

Выполнил: студент 3 курса специальности ИСиТ Калоша И.В.  
Проверила: Ржеутская Н. В.

Минск 2020

**Элементы теории информации. Информативность данных в различных кодировках**

**Цель:** приобретение практических навыков трансформации данных и сопоставление энтропийных свойств используемых при этом алфавитов.

**Теоретические сведения**

Из энтропийных оценок (алфавитов и сообщений), полученных в ходе выполнения лабораторной работы № 2, мы выяснили, что энтропия зависит от статических характеристик самих алфавитов и сообщений (вспомним энтропию по Шеннону и по Хартли). Энтропия максимальна при равномерном появлении букв на любом месте сообщения. Для характеристики источника сообщений с различным алфавитом представляет интерес сравнение фактической энтропии источника с максимально возможной. В этом смысле введено понятие избыточности источника сообщений, или избыточности алфавита. Избыточностью алфавита называют уменьшение информационной нагрузки на один символ вследствие разной вероятности и взаимозависимости появления его символов в сообщениях.

В наиболее общем виде избыточность алфавита R можно оценить отношением энтропии по Хартли и по Шеннону; при этом первая рассчитывается по выражению (2.2), вторая – по формуле (2.1) При выполнении предыдущей работы мы убедились, что формально одно и то же сообщение, но представленное на основе алфавита русского (белорусского, английского или иного) языка – с одной стороны, и представленное в кодах ASCII – с другой, будут характеризоваться различным количеством содержащейся в них информации. Эта дополнительная избыточность обусловлена переносом сообщения из одной среды в другую или, иначе говоря, кодированием символов исходного алфавита. Утверждение восьмибитных кодировок (ASCII) как стандарта принесло некоторые проблемы. К этому моменту уже существовала определенная инфраструктура, использующая семибитные кодировки. Известны проблемы с «обрезанием восьмого бита» в системе электронной почты. Утверждение восьмибитного символа дало 256 различных значений, что позволило уместить в одной кодовой таблице и общепринятые символы (цифры, знаки препинания, латиницу), и символы кириллицы. Уже созданное к тому времени и работающее программное обеспечение зачастую было приспособлено для семибитных кодировок, что приводило, например, к тому, что почтовый сервер при передаче письма обнулял старшие биты в каждом байте сообщения. Одним из решений проблемы стала кодировка (а точнее – алгоритм) base64. В PGP алгоритм base64 используется для кодирования бинарных данных. Кодирование base64 разработано для представления произвольных последовательностей октетов в форме, позволяющей использовать строчные и прописные буквы. Используется 65-символьное подмножество набора символов US-ASCII, обеспечивающее представление одним печатным символом 6 битов данных (дополнительный 65-й символ используется для обозначения функции специальной обработки). Процесс кодирования представляет группу из 24 последовательных битов в форме строки из 4 символов. Обработка выполняется слева направо, а 24-битная исходная группа образуется конкатенацией трех 8-битных групп (байтов). Данные 24 бита после этого трактуются как 4 сцепленных группы по 6 битов, каждая из которых транслируется в один символ алфавита base64. Каждая 6-битная группа используется в качестве индекса массива из 64 печатных символов. Символы алфавита, соответствующие индексу, помещаются в выходную строку [6]. Кодирование base64 с безопасным алфавитом используется для представления URL и имен файлов. В табл. 3.1 перечислен алфавит, используемый для base64- кодировки. Значения представлены в различных системах счисления: десятичной (10), двоичной (2), восьмеричной (8) и шестнадцатеричной (16 или hex). Еще раз обратимся к процессу кодировки. Как было выше установлено, каждые 6 битов буфера, начиная с самых старших, используются как индексы строки «ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/», и ее символы, на которые указывают индексы, помещаются в выходную строку. Если кодируются только один или два байта, в результате получаются только первые два или три символа строки, а выходная строка дополняется двумя или одним символами «=». Это предотвращает добавление дополнительных битов к восстановленным данным.

Процесс повторяется над оставшимися входными данными. Такая обработка выполняется в тех случаях, когда последняя группа входных данных содержит меньше 24 битов. Кодируемое значение всегда завершается полным квантом кодирования. Если на входе доступно менее 24 битов, входная группа дополняется (справа) нулями до формирования целого числа 6-битных групп. Заполнение в конце данных осуществляется как раз с использованием символа «=». Поскольку входная информация base64 всегда включает целое число октетов, возможны лишь перечисленные ниже случаи:

• размер финального блока кодирования на входе кратен 24 битам, кодированный результат будет содержать целое число 4-символьных групп без заполнения символами «=»;

• размер финального блока кодирования на входе составляет 8 битов, выходной блок будет представлять 2 символа, дополненные последовательностью из двух символов заполнения «==»;

• размер финального блока кодирования на входе составляет 16 битов, выходной блок будет представлять 3 символа, дополненные символом заполнения «=».

**Листинг кода**

import base64

import numpy as np

import matplotlib.pyplot as plt

def ShenonAndHartly(path):

with open(path) as f:

a=np.array(list(f.read()))

b,cnt = np.unique(a, return\_counts=True)

p=cnt/np.sum(cnt)

zip\_iterator = zip(b,cnt)

value\_dict = dict(zip\_iterator)

plt.bar(list(value\_dict.keys()),value\_dict.values(),color='g') plt.show()

First\_H=-np.sum(p\*np.log2(p))#shenon

Second\_H=np.log2(len(b))#hartly

overfill=(Second\_H - First\_H/Second\_H)\*100

print(b) print(cnt)

print("Энтропия Шеннона: ",First\_H) print("Энтропия Хартли: ",Second\_H) print("Избыточность: ", overfill)

def XOR(a,b):

a=list(a)

b=list(b)

if(len(a)>len(b)):

for i in range(len(a)-len(b)):

b.append(0)

if(len(b)>len(a)):

for i in range(len(b)-len(a)):

a.append(0)

#result=np.logical\_xor(a,b)

result=set(b)^set(a)

return result

**Ход выполнения**

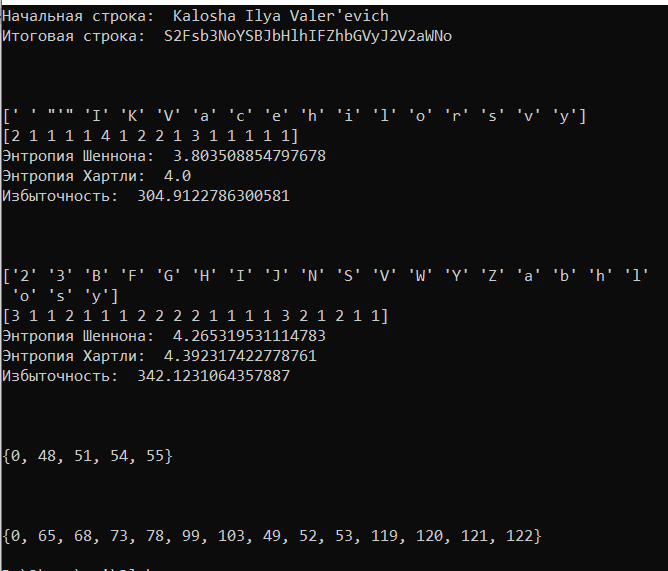
****

Рисунок 1 – Вывод консоли

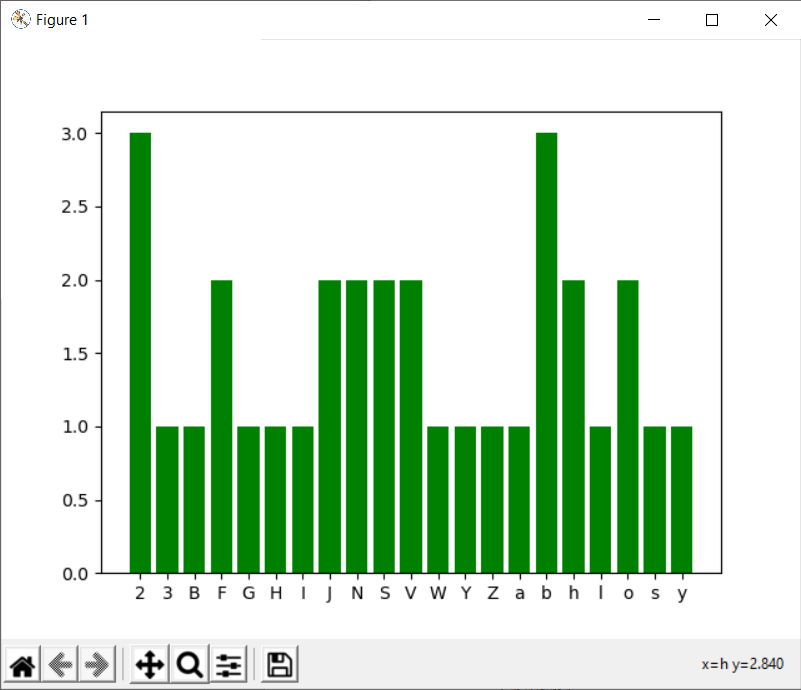
****

Рисунок 2 – График вероятностей

**Вывод**: в данной работе были проверены энтропийные свойства алфавитов. Были рассмотрены энтропии Шеннона и Хартли. Опробованы и реализованы их алгоритмы. С их помощью были оценены энтропийные показатели алфавитов. Было выяснено, что алфавиты с большим количеством символов являются избыточными и информацию, которую несет каждый символ алфавита, уменьшается.