Пшенко Артем, ФИТ 3-4

Информационная безопасность

Отчет по лабораторной работе № 2.

В данной лабораторной работе нужно было разработать свой конвертер или воспользоваться онлайн base64-кодировщиком для представления текста в формате base64. Исходный код функции для перевода текста в base64-формат представлен в листинге 1.

Листинг 1. Исходный код функции для перевода текста в base64-формат.

|  |
| --- |
| def convert\_to\_base64(file):  with open(file, 'rb') as f:  text = f.read()  encoded\_text = b64encode(text)  with open('./base64.txt', 'wb') as f:  f.write(encoded\_text)  return encoded\_text |

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.

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.

После реализации данной функции необходимо было посчитать энтропию Шеннона и Хартли для файлов на латинице и в формате base64, а также получить распределение частотных свойств алфавита и избыточности алфавитов. Для решения данной задачи воспользуемся функциями, реализованными в 1-й лабораторной работе.

Листинг 2. Исходный код функций, взятых из первой лабораторной работы.

|  |
| --- |
| def draw\_hist(probabilities):  plt.bar(probabilities.keys(), probabilities.values())  plt.xlabel("Символ")  plt.ylabel("Вероятность")  plt.title("Вероятности символов")  plt.show()  histograms = {} |

Продолжение листинга 2.

|  |
| --- |
| def calculate\_Shannon\_entropy(file\_name):  with open(file\_name, 'r') as f:  text = f.read()  #словарь вида "символ - количество"  symbol\_counts = collections.Counter(text)  #print(symbol\_counts, '\n')  total\_symbols = len(text)  #словарь вида "символ - его вероятность"  probabilities = {symbol: count / total\_symbols for symbol, count in symbol\_counts.items()}  #print(probabilities)  if file\_name not in histograms:  draw\_hist(probabilities)  histograms[file\_name] = probabilities  entropy = -sum(p \* math.log2(p) for p in probabilities.values())  return entropy |

Частным случаем энтропии Шеннона считается энтропия Хартли. Дополнительным условием при этом является то, что все вероятности одинаковы и постоянны для всех символов алфавита:



Здесь N – мощность алфавита. Для нахождения энтропии Хартли была реализована соответствующая функция. Ее исходный код приведен в листинге 2.

Листинг 3. Исходный код функции для вычисления энтропии Хартли

|  |
| --- |
| def calculate\_Hartley\_entropy(file\_name):  with open(file\_name, 'r') as f:  text = f.read()  N = len(set(text))  return log2(N) |

Диаграмма распределения вероятностей символов для файла на латинице приведена на рисунке 1.

|  |
| --- |
|  |

Рисунок 1. Диаграмма распределения вероятностей символов для файла на латинице

Диаграмма распределения вероятностей символов для файла в формате base64 приведена на рисунке 2.

|  |
| --- |
|  |

Рисунок 2. Диаграмма распределения вероятностей символов для файла в формате base64

Результаты вычисления энтропии Хартли и Шеннона для файла на латинице и в формате base64 приведены на рисунке 3.

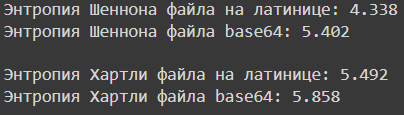


Рисунок 3. Результаты вычисления энтропии Хартли и Шеннона для файла на латинице и в формате base64

Для нахождения избыточности алфавитов воспользуемся формулой:



Здесь *HCh(A)* – энтропия Хартли, а *HS(A)* – энтропия Шеннона. Результат вычисления избыточности для файла на латинице и в формате base64 приведен на рисунке 4.



Рисунок 4. Результаты вычисления избыточности алфавитов для файла на латинице и в формате base64

В последнем задании нужно было реализовать функцию для вычисления XOR. Исходный код приведен в листинге 4.

Листинг 4. Исходный код для вычисления XOR

|  |
| --- |
| def xor(a, b, is\_base64=False):  if is\_base64:  a = b64decode(a)  b = b64decode(b)  if len(a) != len(b):  if len(a) < len(b):  a += b'0' \* (len(b) - len(a))  print(f'Заполнение нулями, чтобы параметры были одинаковы по длине: {a}')  else:  b += b'0' \* (len(a) - len(b))  print(f'Заполнение нулями, чтобы параметры были одинаковы по длине: {b}')  result = [(a ^ b) for a, b in zip(a, b)]  return result |

Результаты вычисления XOR для моей фамилии (Pshenko) и имени (Artyom) на латинице и в формате base64 приведен на рисунке 5.

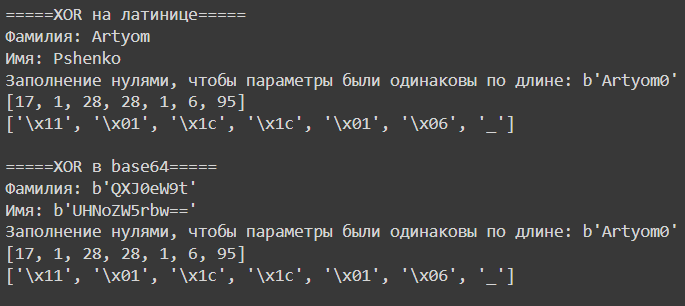


Рисунок 5. Результаты вычисления XOR для фамилии и имени на латинице и в формате base64.