УДК 004

Сжатие данных с применением алгоритмов LZW на языке программирования python

Батенков Никита Дмитриевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Целью данной работы является написание кода для сжатия текста, используя алгоритм LZW. Для выполнения этой задачи был выбран язык программирования python. Практическая значимость работы заключается в создании скрипта способного сжать текст.
Ключевые слова: информационные технологии, сжатие данных, текст, строки.

Data compression using LZW algorithms in python programming language

Batenkov Nikita Dmitrievich
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of this work is to write a text compression code using the LZW algorithm. The python programming language was chosen to perform this task. The practical significance of the work is to create a script capable of compressing text.
Keywords: information technology, data compression, text, strings.

1 Введение

1.1 Актуальность

В основе любого способа сжатия лежит модель источника данных, или, точнее, модель избыточности. Иными словами, для сжатия данных используются некоторые априорные сведения о том, какого рода данные сжимаются. Не обладая такими сведениями об источнике, невозможно сделать никаких предположений о преобразовании, которое позволило бы уменьшить объём сообщения. Методы, позволяющие на основе входных данных изменять модель избыточности информации, называются адаптивными. Неадаптивными являются обычно узкоспециализированные алгоритмы, применяемые для работы с данными, обладающими хорошо определёнными и неизменными характеристиками.

1.2 Обзор исследований

В.Ю.Мокрый в статье приводит описание последовательности обучения сжатию графической информации на примере алгоритма JPEG [1]. В.Г.Шишмарова рассмотрел проблемы лаконичности речи и способы сжатия текста. [2]. В.В.Исаченко описал новую модель сжатия данных. [3]. Н.Г.Шандрюк в работе представил аппаратную реализацию потокового сжатия данных на ZYNQ xc7z020clg484-1 [4]. О.О.Серый в статье представил метод определения характеристик текстов и их классификации с помощью архивирования [5].

1.3 Цель исследования

Цель исследования - Сжать данные, применив алгоритм LZW

2 Материалы и методы

Для разработки алгоритма был использован язык программирования Python и среда разработки Google Colab.

3 Результаты исследования

В данной работе показана реализация алгоритма LZW.

3.1 Добавления текста для сжатия

В среде программирования Google Colab создаётся переменная с текстом, который в дальнейшем будет сжат (рис. 1), а также создаём словарь, в который запишем символы и их сочетания, которые будем использовать для кодировки.

```
string =input('Please enter the string to be compressed:')
dictionary = {chr(i): i for i in range(1, 123)}
last = 256
p = ""
result1 = []
```

Рисунок 1. Текст для сжатия

3.2 Сжатие текста

Выполняется цикл, который находит новые комбинации символов и записывает их в словарь, присваивая им код. Далее обрабатывается последний символ, считается длинна полученного кода и выводится результат(рис.2).

```
for c in string:
    pc = p + c
    if pc in dictionary:
        p = pc
    else:
        result1.append(dictionary[p])
        dictionary[pc] = last
        last += 1
        p = c

if p != '':
   result1.append(dictionary[p])
x2 = len(result1)
print('The compressed code is:',result1)
```

Рисунок 2. Сжатие текста

3.4 Распаковка текста

Сначала производится обратный импорт кода в новый словарь. После этого с помощью цикла переписываем и декодируем знаки из первого словаря во второй (рис. 3).

```
dictionary2 = {i: chr(i) for i in range(1, 123)}
last2 = 256

result2 = []
p = result1.pop(0)
result2.append(dictionary2[p])

for c in result1:
    if c in dictionary2:
        entry = dictionary2[c]
    result2.append(entry)
    dictionary2[last2] = dictionary2[p] + entry[0]
    last2 += 1
    p = c

print('The decoding result is:')
print(''.join(result2))
```

Рисунок 3. Распаковка текста

Для наглядности результата добавляется расчёт и вывод длины строки до кодировки, после кодировки и коэффициент сжатия (рис.4).

```
x1 = len(string)
x3 = (x2*9)/(x1*8)
print('String length:',x1)
print('Length after encoding:',x2)
print('LZW compression ratio:',x3)
```

Рисунок 4. Расчёт и вывод информации об результатах кодирования

3.4 Работа программы

После запуска программы получается результат кодировки и декодировки также выводятся длины строки до выполнения программы, после и коэффициент сжатия (рис.5).

```
Please enter the string to be compressed:Study, study and study again
The compressed code is: [83, 116, 117, 100, 121, 44, 32, 115, 257, 259, 32, 97, 110, 100, 262, 264, 121, 266, 103, 97, 105, 110]
The decoding result is:
Study, study and study again
String length: 28
Length after encoding: 22
LZW compression ratio: 0.8839285714285714
```

Рисунок 5. Выполнение программы

4 Выводы

В ходе выполнения работ был реализован алгоритм LZW, который сжимает текст. Также программа показала, что коэффициент сжатия составил менее единицы, что указывает на успешность сжатия текста.

Библиографический список

1. Шишмарова В.Г. Способы информационного сжатия текста без потери информации: замещение, опущение, совмещение // Сборник статей Международного научно-исследовательского конкурса. 2019. С. 91-102.
2. Исаченко В.В. Сжатие данных на основе сборки слов// Сборник материалов XII Международной школы-конференции студентов, аспирантов и молодых ученых. 2016. С. 332-336.
3. Шандрюк Н.Г. Реализация на плис алгоритма сжатия данных в потоке // Фундаментальные проблемы радиоэлектронного приборостроения. 2018. Т. 18. № 4. С. 1007-1010.
4. Серый О.О. Метод кластеризации сообщений с помощью Архивирующего преобразования//ScienceRise. 2015. Т. 6. № 2 (11). С. 76-79.
5. Мокрый В.Ю. Последовательность обучения алгоритмам сжатия графической информации на примере алгоритма JPEG // Вестник Волжского университета им. В.Н. Татищева. 2011. № 18. С. 135-139.