Сжатие информации с учётом контекста

Александра Игоревна Кононова

МИЭТ

21 декабря 2020 г. — актуальную версию можно найти на `https://gitlab.com/illinc/otik`

Источник с памятью

Вероятность порождения элемента определяется предысторией источника $\rightarrow$ при сжатии необходимо учитывать контекст символа.

источник Маркова (N-го порядка) — состояние на i-м шаге зависит от состояний на N предыдущих шагах:
$i-1, i-2, \ldots, i-N \rightarrow$ сжимаются не отдельные символы, а устойчивые сочетания — слова:
коды Зива—Лемпеля (LZ77, LZ78);

аналоговый сигнал — источник количественных данных Маркова 1-го порядка $\rightarrow$ кодирование длин повторений (Run Length Encoding, RLE).

Концепция RLE

Вместо кодирования данных кодируются длины участков, на которых данные сохраняют неизменное значение.

$$\mathrm{AAAAAAAABCCCCC} \rightarrow 8 \times \mathrm{A}, 1 \times \mathrm{B}, 4 \times \mathrm{C}$$

Повторение символа c подряд L раз $(L \times c)$, $L > 0$ будем записывать как пару (L, c).

Если c и L одного размера — это имеет смысл только при $L > 2$.

Реализации RLE

1. «Сжимать» любую цепочку: $L \times c \to (L, c)$ для любого L ($L \geqslant 1$):
   0123456789ABCDEF → 10111213141516171819 1A1B1C1D1E1F
   увеличение объёма в наихудшем случае — в 2 раза;
2. Сжатые и несжатые цепочки: $\begin{cases} L \times c, L \geqslant 3 \to (1 + L, c), \\ c_1 \ldots c_L, L \geqslant 1 \to (0 + L, c_1 \ldots c_L) \end{cases}$

байт-октет — 7 бит на L ($0 \ldots 127$ без коррекции), 4-битный — 3 ($0 \ldots 7$):

0123456789AAAAAAAABCDEF → 0123456, 789, AAAAAAA, ABCDEF →
→701234563789FA6ABCDEF (21) (для октетов ув. в нх. сл. на $\frac{1}{127}$)

Коррекция L: запись $L - 1$ для несжатой ($1 \ldots 8$), $L - 3$ для сжатой ($3 \ldots$ A):

0123456789AAAAAAAABCDEF → 01234567, 89, AAAAAAAA, BCDEF →
→7012345671 89DA4BCDEF (20) (для октетов ув. в нх. сл. на $\frac{1}{128}$)

3. Префикс сжатой цепочки в несжатом тексте: $L \times c \to (p, L, c), L \geqslant 4$:
   $p =$ C: 0123456789AAAAAAAABCDEF (23) → 0123456789C8ABC1CDEF (20)
   или 0123456789C8ABC0DEF (19) (для октетов ув. в нх. сл. на $\frac{2}{256}$ или $\frac{1}{256}$)

Коррекция: для C0 $(p, L - 3, c)$; для C1C $\begin{cases} L \times c, c \neq p \to (p, c, L - 4) \\ L \times p \to (p, p, L - 1), \end{cases}$

Код Зива–Лемпеля, LZ77

1977 г., Якоб Зив (Jacob Ziv) и Абрахам Лемпель (Abraham Lempel): последующие вхождения цепочки заменяются ссылками на первое: (S, L)

Скользящее окно: область перед текущей позицией кодирования, в которой можем искать и адресовать ($w \leqslant \max(S)$) ссылки

Если S записывается одним байтом (n различных значений), то максимальный размер окна $w = \max(S) = n - 1$.

Наихудший случай (для 4-битного байта):
`0123456789ABCDEF0123456789ABCDEF`...
в пределах окна нет ни одного совпадения.

Увеличить размер окна w можно, записывая S двумя байтами.

Пример №1

Alternative vexillum codicis inf. interpretatio (AVCII): ,абвдеиклмнорты
0123456789ABCDEF

0123456789012345678901234567890123456789012345678901234567890123
там корабли лавировали, лавировали, да не вылавировали
$\begin{cases} S_1 = 23 - 11 = 12 = \texttt{C}, \\ L_1 = 13 = \texttt{D} \end{cases}$ — кодирование 4-битными байтами возможно

0123456789012345678901234567890123456789012345678901234567890123
там корабли лавировали, лавировали, да не вылавировали
$S_2 = 20 > \texttt{F}$ — кодирование 4-битным байтом невозможно

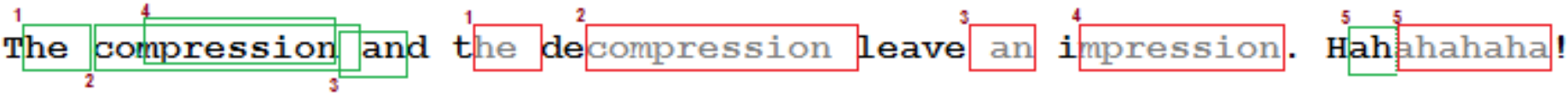
При программной реализации (поиск только в окне) при $w = 15$ будет найдено только одно совпадение (S_1, L_1).

там корабли лавировали,(S_1, L_1)да не вылавировали
$(S_1, L_1) = \texttt{CD} = \texttt{op}$

Пример №2, Михаил Быстрянцев

The compression and the decompression leave an impression. Hahahahaha!

The compression and t[he]de[compression]leave[an] i[mpression]. Hah[ahahaha]!



Абсолютные ссылки (P, L), где P — позиция первого вхождения цепочки, L — её длина:

The compression and t$(2,3)$de$(5,12)$leave$(16,3)$ i$(8,7)$. Hah$(61,7)$!

Замена абсолютных позиций P на относительные смещения S:

The compression and t$(20,3)$de$(22,12)$leave$(28,3)$ i$(42,7)$. Hah$(2,7)$!

Как отделить несжатый текст от ссылок?

Реализации LZ77

1. Несжатый символ c как «ссылка» $(0, c)$ — увеличение объёма в нх. сл. вдвое
2. Цепочка, за которой следует c — тройка (S, L, c), несжатый символ c — тройка $(0, 0, c)$ — увеличение объёма в наихудшем случае втрое
3. Быстрянцев: символы и цепочки группируются по 8, тип — байт-флаги:

   `The␣c o mp` 0b 0000 0000 The comp
   `ressi o n␣` 0b 0000 0000 ression
   `and␣t(20,3)de` 0b 0000 0100 and t 20 03 de

   — увеличение объёма в наихудшем случае на $\frac{1}{8}$
4. Префикс p как маркер ссылки: $\begin{cases} \text{ссылка } (p, S, L): \ 0 < S \leqslant w, L \geqslant 4, \\ \text{символ } p \to (p, 0) \end{cases}$

   — увеличение объёма в наихудшем случае на $\frac{1}{256}$

   $p = \texttt{F} = $ ы; $S_1 = 12 = \texttt{C}$, $L_1 = 13 = \texttt{D}$

   `там корабли лавировали,ыCDда не вы0лавировали` (L без коррекции)

   `там корабли лавировали,ыC9да не вы0лавировали` ($L \to L - 4$)

Сравнение с RLE

Префикс $p = \texttt{C}$:

Повторение одинаковых символов:
RLE:
0123456789AAAAAAAABCDEF → 0123456789C8ABC0DEF $23 \to 19$
LZ77:
0123456789AAAAAAAABCDEF → 0123456789AC17BC0DEF $23 \to 20$

Повторение подстрок:
RLE: $28 \to 24$
01234560123478 9AAAAAAAABCDEF → 01234560123478 9C8ABC0DEF
LZ77: $28 \to 23$
0123456012347 89AAAAAAAABCDEF → 0123456C75789AC17BC0DEF

Задача №1

1. Рассчитать количество информации в сообщении $C = \text{«}\underbrace{\text{ля-ля-ля-...-ля}}_{\text{80 раз}}\text{»}$ (кодировка koi8-r)
2. Закодировать сообщение C алгоритмом LZ77

Информация и модели источника

Заменяя вероятности символов на их оценки по модели X, получаем **оценку** количества информации:

1. Без памяти, $A_1 = \{\text{л}, \text{я}, \text{-}\}$:
$$I_1 = 2 \cdot 80 \cdot \left(-\log_2 \left(\tfrac{80}{239} \right) \right) + 79 \cdot \left(-\log_2 \left(\tfrac{79}{239} \right) \right) = 254{,}2 \text{ бит} = 31{,}8 \text{ байт}$$

2. Без памяти, $A_1 = \{\text{ля-}, \text{ля}\}$:
$$I_2 = 79 \cdot \left(-\log_2 \left(\tfrac{79}{80} \right) \right) + 1 \cdot \left(-\log_2 \left(\tfrac{1}{80} \right) \right) = 7{,}6 \text{ бит} = 0{,}9 \text{ байт}$$

3. Маркова, $A_1 = \text{koi8-r}, N = 3$ со следующими оценками вероятностей:
$p(c_1 = \text{л}) = p(c_2 = \text{я}) = p(c_3 = \text{-}) = \frac{1}{256}$, $p(\text{-}|\text{-ля}) = \frac{79}{80}$,
$p(\text{л}|\text{ля-}) = p(\text{я}|\text{я-л}) = 1$, $p(eof|\text{-ля}) = \frac{1}{80}$:

$$I_3 = 3 \cdot \log_2 256 + 79 \cdot \left(-\log_2 \left(\tfrac{79}{80} \right) \right) + 1 \cdot \left(-\log_2 \left(\tfrac{1}{80} \right) \right) = 31{,}6 \text{ бит} = 3{,}9 \text{ байт}$$

Задача №2

Используя 4-битный байт, закодируйте:

1. Сообщение (64 = 0x40 символов) — LZ77:

	0123456789ABCDEF	0123456789ABCDEF	0123456789ABCDEF	0123456789ABCDEF
00	косил косой коса	рь за косарь кос	ой косой космеи	на косе при осе.

AVCII:	код i	0123456789ABCDEF
	a_i	.аезийклмнопрсь
	$\nu(a_i)$	B1431338111C1392

а) S занимает 1 байт (4 бита); б) S занимает 5 бит; в) S занимает 2 байта.

2. 16-цветный рисунок 8×8 — RLE, LZ77.

Код Зива–Лемпеля, LZ78 (концепция)

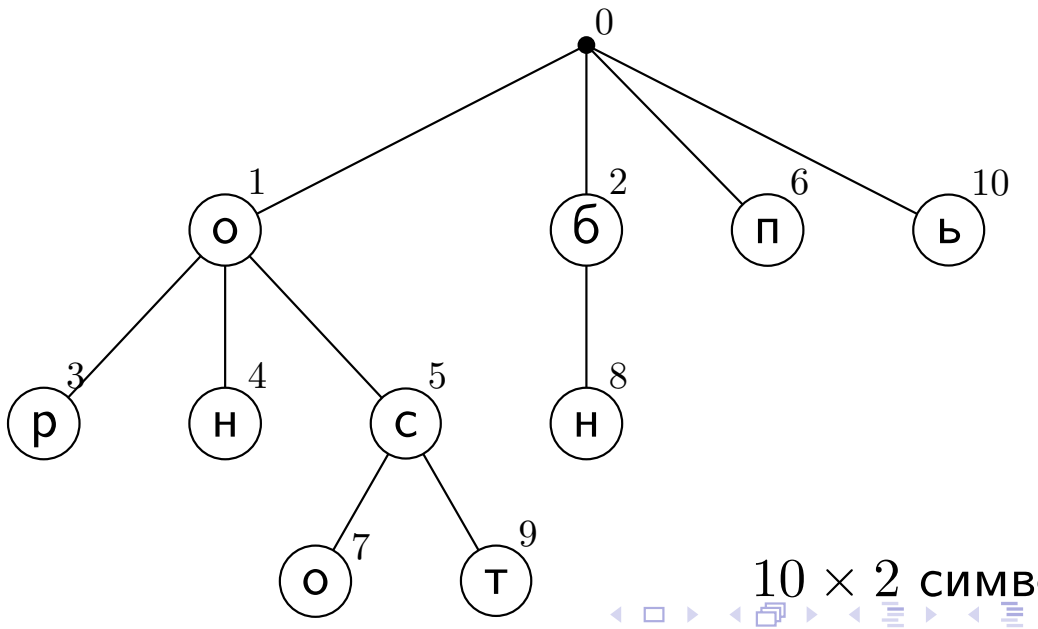
1978 г., Якоб Зив (Jacob Ziv) и Абрахам Лемпель (Abraham Lempel):

1. Скользящее окно не используем — кодируем в один проход вперёд $\Longrightarrow$ высокая скорость кодирования-декодирования.
2. Словарь = дерево, узел — номер и символ (n, c),
   корень — $(0,$ пустая строка$)$, слово читается от корня.
3. Вначале словарь пуст (только корень).
4. На каждом шаге
   - к словарю добавляется узел (лист);
   - в выходной поток — номер родителя и символ нового листа (P, c).
5. Когда кончается ёмкость номера листа, дерево:
   - либо уничтожается и растится заново;
   - либо ветви уничтожаются выборочно (сложно);
   - либо фиксируется и не растёт (нет прикорневого узла $\Longrightarrow$ сбои);
   - либо увеличивается разрядность номера.
6. При необходимости вх-й поток дополняется (либо конец обр-ся особо).

«Обороноспособность» (18, 8 разных)

1. Вначале словарь = корень (пустая строка), $n = 0$, $i = 0$.
2. $P = 0$ (текущий узел — корень), c_i (текущий символ входного потока);
3. Если c_i — дочерний P, $P = c_i$ и читаем c_{i+1} $(++i)$
4. Если c_i нет в дочерних узлах P:
   - добавляем P дочерний узел (n, c_i), $++n$ и читаем c_{i+1} $(++i)$;
   - в выходной поток пишем (P, c_i).

1	(0,о)	о
2	(0,б)	б
3	(1,р)	ор
4	(1,н)	он
5	(1,с)	ос
6	(0,п)	п
7	(5,о)	осо
8	(2,н)	бн
9	(5,т)	ост
10	(0,ь)	ь



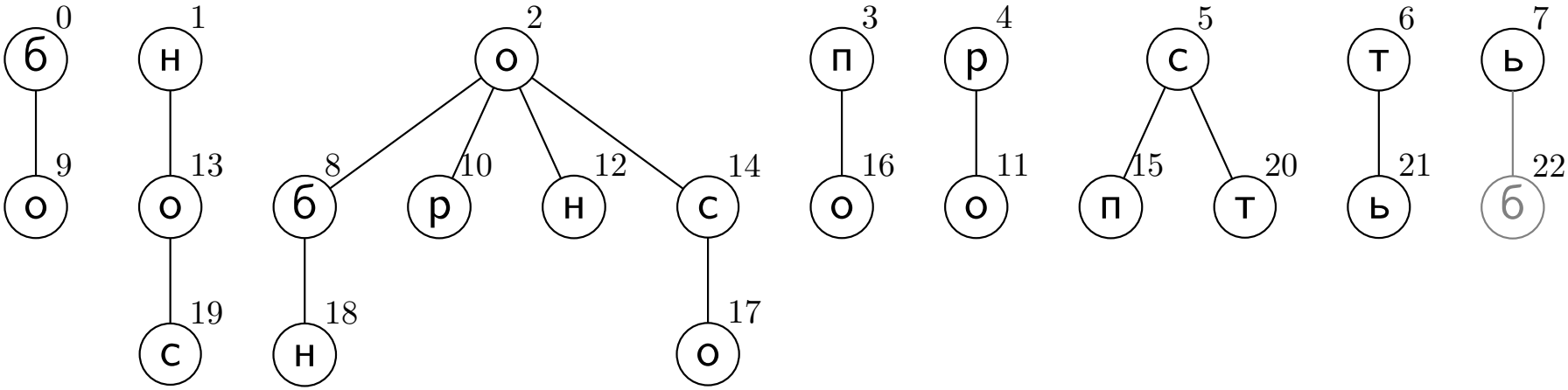
10×2 символов

Код Зива–Лемпеля–Велча, LZW

1984 г., Терри Велч (Terry Welch) по концепции LZ78:

1. Вначале словарь = первый уровень (все одиночные символы, N штук). Тогда корень можно не нумеровать (прикорневые нумеруем с нуля).
2. При добавлении P дочернего узла (n, c_i):
   - оставляем c_i во входном потоке;
   - в выходной поток пишем (P).
3. При декодировании узла n:
   - в выходной поток пишем всю ветвь;
   - в дерево добавляем только прикорневой узел.
4. $|n| >> |c|$, во многих реализациях увеличивается по битам.
5. Дерево часто разворачивается в таблицу.
6. Вх-й поток всегда дополняется как минимум одним незначащим символом.

«Обороноспособность» (18, алфавит из 8)



8	(2, б)	об	2
9	(0, о)	бо	0
10	(2, р)	ор	2
11	(4, о)	ро	4
12	(2, н)	он	2
13	(1, о)	но	1
14	(2, с)	ос	2
15	(5, п)	сп	5

16	(3, о)	по	3
17	(14, о)	осо	14
18	(8, н)	обн	8
19	(13, с)	нос	13
20	(5, т)	ст	5
21	(6, ь)	ть	6
22	(7, б)	ьб	7
15	символов		

Декодирование (замечания)

1. При декодировании на i-м шаге (входной номер P_i) в выходной поток добавляется строка C_i, соответствующая узлу P_i (то есть на один символ короче, чем была на i-м шаге при кодировании),

   а в дереве словаря от самого P_i должен отрасти дочерний узел с номером i и неизвестным символом c_i.

2. Символ c_i узла i становится известным только на шаге $i+1$ (c_i — это первый символ подстроки C_{i+1} шага $i+1$),

   поэтому на практике узел i добавляется в словарь на шаге $i+1$.

3. Если на $i+1$ шаге получаем ссылку на ещё не добавленный узел $P_{i+1} = i$, то всё равно c_i — это первый символ подстроки C_{i+1},

   а первый символ C_{i+1} мы знаем даже при $P_{i+1} = i$ (как и все до предпоследнего включительно)!

   При $P_{i+1} = i$ последний символ строки C_{i+1} совпадает с первым: $C_{i+1} = axx\ldots xa$.

Декодирование

0	1	2	3	4	5	6	7
б	н	о	п	р	с	т	ь

2	о		
0	б	8	(2, б)
2	о	9	(0, о)
4	р	10	(2, р)
2	о	11	(4, о)
1	н	12	(2, н)
2	о	13	(0, о)
5	с	14	(2, с)

3	п	15	(5, с)
14	ос	16	(3, о)
8	об	17	(14, о)
13	но	18	(8, н)
5	с	19	(13, с)
6	т	20	(5, т)
7	ь	21	(6, ь)

Спасибо за внимание!

МИЭТ
http://miet.ru/

Александра Игоревна Кононова
illinc@mail.ru