Метод скользящего словаря LZ-77

5 января 2023 г. 1:25

Кодер хранит в памяти скользящий словарь объемом W (последние W букв)
Чем больше W, тем лучше сжатие, но больше будет требоваться памяти для хранения скользящего словаря и больше вычислений будет производиться в процессе поиска.

Считаем, что символы $x_1, \ldots, x_{n-1}$ уже закодированы. Кодер смотрит на следующие символы: начиная с x_n и дальше, пытается найти как можно более длинную последовательность, которая уже встречалась раньше.

- Если ничего не нашлось (т.е. встретился какой-то новый символ), то декодеру передается флаг 0 (1 бит, который говорит о том, что ничего не нашлось) и равномерным кодом бинарное представление этой новой буквы (8 бит в нашем случае, если 256 букв).
- Если нашлось, то декодеру передается флаг 1, а также значения d и l

d - расстояние от текущей позиции до начала найденного в словаре слова, кодируется равномерным кодом длины $\lceil log(W) \rceil$
l - длина найденного слова, кодируется неравномерным побуквенным кодом, например, унарным (или Хаффманом, или чем-то еще... в теории, Шенноном можно... но не нужно)

Пример, когда $vlc(x) = mon(x)$

Шаг	Флаг	$(x_1, \ldots, x_n)$	W	d	l	Код	Биты
0	0	I	0	-	0	0 bin(I)	9
1	0	F	1	-	0	0 bin(F)	9
2	0	_	2	-	0	0 bin(_)	9
3	0	W	3	-	0	0 bin(W)	9
4	0	E	4	-	0	0 bin(E)	9
5	1	_	5	2	1	1 010 0	5
6	0	C	6	-	0	0 bin(C)	9
7	0	A	7	-	0	0 bin(A)	9
8	0	N	8	-	0	0 bin(N)	9
9	1	N	9	0	1	1 0000 0	6
10	0	O	10	-	0	0 bin(O)	9
11	0	T	11	-	0	0 bin(T)	9
12	1	_	12	6	1	1 0110 0	6
13	0	D	13	-	0	0 bin(D)	9
14	1	O	14	3	1	1 0011 0	6
15	1	_	15	2	1	1 0010 0	6

Пример, когда $vlc(x) = mon(x)$

Шаг	Флаг	$(x_1, \ldots, x_n)$	W	d	l	Код	Биты
16	1	A	16	8	1	1 01000 0	7
17	0	S	17	-	0	0 bin(S)	9
18	1	_WE_	18	15	4	1 01111 11000	11
19	1	W	22	2	1	1 00010 0	7
20	1	O	23	8	1	1 01000 0	7
21	0	U	24	-	0	0 bin(U)	9
22	0	L	25	-	0	0 bin(L)	9
23	1	D	26	12	1	1 01100 0	7
24	1	_WE_	27	8	4	1 01000 11000	11
25	1	S	31	13	1	1 01101 0	7
26	0	H	32	-	0	0 bin(H)	9
27	1	OULD_	33	9	5	1 001001 11001	12
28	1	DO_AS_WE_	38	24	9	1 011000 1110001	14
29	1	CAN	47	40	3	1 101000 101	10
Всего бит:							257

Возможные модификации LZ-77

- Использовать код Хаффмана или арифметическое кодирование для $vlc(x)$.
- Ограничить максимальное l, $W = 2^w$ для более удобной организации памяти[4].
- Хранение словаря в виде древовидной структуры[5].
- Избыточность LZ-77 с увеличением W определяется как:

чем больше размер окна, тем меньше избыточность

$$\bar{R} \approx H_\infty(X) + \frac{\log \log W}{\log W}$$