Об одной реализации арифметического кода*

©1993, 1998, М. В. Захаров
Россия, 354057, г. Сочи, а/я 490.
e-mail: maxime@tnet.sochi.ru

Здесь описывается построение арифметического кода, алгоритм которого не содержит операций умножения и деления. Т.к. эти операции в современных персональных ЭВМ выполняются на порядок дольше, чем операции сложения и сдвига, то данный алгоритм будет обладать примерно во столько же раз большим быстродействием, чем классическая схема реализации арифметического кода. Однако, как будет показано ниже, мы будем проигрывать в эффективности[1] кодирования по сравнению с той же классической схемой.

1 Классическая схема

Идея классической схемы состоит в следующем [1]: обрабатываемый текст представляется как интервал вещественных чисел, входящий в интервал $(0\ldots1)$. Каждый поступающий на обработку символ данных уменьшает этот интервал пропорционально вероятности своего появления. По мере поступления очередного символа сообщения интервал, необходимый для его представления, становится все меньше и меньше, а число бит для записи этого интервала - увеличивается. Заметим, что изначально длина интервала равна 1. Известно, что сумма всех вероятностей появления символов алфавита также равна 1. Пусть N - количество символов алфавита и пусть p_i - вероятность появления в тексте символа с номером i. Тогда i-му символу поставим в соответсвие полуинтервал:

$$\left(\sum_{k=0}^{i-1} p_k, \sum_{k=0}^{i} p_k\right], \quad i = 1\ldots N;\;\; p_0 = 0 \tag{1}$$

Если первым на вход поступает символ p_k, то вместо исходного мы берем k-ый интервал и делим его на N частей в тех же пропорциях, что и исходный. Теперь мы готовы к приему второго символа и т.д.

*Версия от 25 ноября 1998 г.

[1]Под эффективностью здесь и далее подразумевается отношение размера сжатого текста к размеру исходного.

Если априорно вероятности появления символов в тексте не известны, то можно воспользоваться какой-нибудь их оценкой, например, почти всегда используется частота. Тогда суммам в (1) будут соответствовать так называемые интегральные частоты. Пусть υ_i - частота появления i-го символа в тексте, тогда его интегральная частота будет равна

$$Q_i = \sum_{k=0}^{i-1} \upsilon_i, \quad \upsilon_0 = 0 \tag{2}$$

Обозначив $High_i$ правую границу интервала, соответствующего уже закодированному фрагменту текста к моменту поступления на вход i-го символа текста, а Low_i - левую, можно записать описанный выше алгоритм кодирования:

$$\begin{aligned} Range_{i+1} &= High_i - Low_i \\ High_{i+1} &= Low_i + Range_{i+1} \cdot Q_{c_i+1} \\ Low_{i+1} &= Low_i + Range_{i+1} \cdot Q_{c_i} \\ High_0 &= 1 \\ Low_0 &= 0 \end{aligned} \tag{3}$$

c_i-номер буквы алфавита, соответствующей i-му символу сообщения, $c_i \in [1, N]$, N - число букв алфавита.

Когда мы просмотрим таким образом весь исходный текст, то заменим два числа $High_m$ и Low_m (m - число символов исходного текста) одним числом, например:

$$Average = \frac{High_m + Low_m}{2} \tag{4}$$

Из соотношений (3) легко видеть, что полученное число (4) будет находиться в интервале, соответствующем первому закодированному символу (не говоря уже о том, что оно будет из интервала $[0; 1)$).

Пусть $Value$ - число, полученное на входе алгоритмом декодирования, тогда каждый символ можно раскодировать применяя следующую последовательность действий: найдем символ c такой, что

$$Q_c \leq \frac{Value - Low_j}{High_j - Low_j} < Q_{c+1} \tag{5}$$

Это и будет очередной символ закодированного текста. После этого необходимо выполнить действия (3). И можно переходить к раскодированию очередного символа.

Классическая схема арифметического кодирования обладает наименьшей погрешностью кодирования среди всех известных на сегодняшний момент статистических методов сжатия. Но одновременно является и самым медленным методом.

Погрешность конкретной реализации арифметического кодирования зависит от точности представления чисел $High_i$ и Low_i. Поэтому, если мы хотим увеличить точность кодирования, мы должны позаботиться о более точном представлении этих чисел.

2 Асимметричная схема

Асимметричная - так как вместо двух счетчиков классической схемы используется только один. Данная схема описана в [2].

Арифметический кодировщик этой схемы представляет кодируемую строку как интегральную вероятность появления строк, предшествующих данной в лексикографическом порядке и отличающихся от нее всего лишь последним символом. Для подсчета интегральной вероятности текста используется некоторое приближение вероятности появления контекста, определяемое конкретной реализацией и обозначаемое в дальнейшем $A(s)$. Таким образом, если обозначить $C(s)$ кодовое слово, т.е. интегральную вероятность строки s, то кодовое слово для строки, полученной из данной добавлением одного символа можно получить следующим образом:

$$\begin{aligned} C(s1) &= C(s) \\ C(si) &= C(s) + A(s1) + \ldots + A(si-1) \end{aligned}$$

где $si-1$ обозначает строку s, дополненную справа символом алфавита с номером $i-1$. Для пустого текста полагают $A(\lambda)=1$, $C(\lambda)=0$; λ - пустой текст.

Для однозначного декодирования необходимо соблюдение еще одного условия [2]:

$$A(s) = A(s1) + \ldots + A(sN) \tag{6}$$

3 Описание реализации

Перейдем от дробных чисел асимметричной схемы к целым числам: вместо частот символов будем использовать их количество в данном контексте. Для целых чисел, представленных в двоичной системе, деление и умножение на числа, кратные 2, эквивалентно операциям сдвига двоичного представления этих чисел на соответсвующее количество разрядов. А операции сдвига в современных ЭВМ выполняются значительно быстрее, чем операции умножения и деления.

Обозначим A_i - количество появлений в контексте символа алфавита сообщения с номером i, Q_i - интегральное количество появлений этого символа, т.е.

$$Q_i = \sum_{k=0}^{i-1} A_i \tag{7}$$

Кодом сжимаемого текста является интегральная сумма количеств появления составляющих его символов. Обозначим его $C(s)$. Изначально $A_i(\lambda) = 1$, $C(\lambda) = 0$. Каждый следующий символ кодируется следующим образом:

$$C(si) = [C(s) + Q_i] \cdot 2^{k_i} \tag{8}$$

Значение k_i определяется из условия однозначности декодирования:

$$[C(s)+Q_i]\cdot 2^{k_i}+Q_j<[C(s)+Q_{i+1}]\cdot 2^{k_i},\quad \forall j\in \mathbf{Z}\cap[1,N] \tag{9}$$

4 Оценка погрешности

Под погрешностью кодирования понимается разность между средним числом бит, затрачиваемым на кодирование одного символа текста, и шенноновской энтропией этого текста.

Оценим эффективность нового метода кодирования. Из (7) видно, что в (9) $\max_j Q_j = Q_N$, где N - число символов алфавита.

$$[C(s)+Q_i]\cdot 2^{k_i}+Q_N<[C(s)+Q_{i+1}]\cdot 2^{k_i}\Rightarrow$$
$$\Rightarrow[Q_{i+1}-Q_i]\cdot 2^{k_i}>Q_N\Rightarrow\{(7)\}\Rightarrow A_i\cdot 2^{k_i}>Q_N$$

Т.к. $A_i\geq 1$, $\forall i$, то

$$2^{k_i}>\frac{Q_N}{A_i} \tag{10}$$

Т.к. k_i может быть только целым, то

$$k_i=\left\lceil\log_2\frac{Q_N}{A_i}\right\rceil, \tag{11}$$

Таким образом, k_i является количеством бит, которым кодируется текущий символ текста. Рассмотрим логарифм в правой части равенства (11):

$$\log_2\frac{Q_N}{A_i}=\log_2\frac{Q_N\cdot M}{A_i\cdot M}=\log_2(\frac{M}{A_i}\cdot\frac{M-A_N}{M})=\log_2\frac{1-\upsilon_N}{\upsilon_i},$$

где M - количество символов в тексте, υ_i - частота символа i. Для эргодических источников $\upsilon_i\to p_i$ при $M\to\infty$. Тогда количество бит, затрачиваемых на кодирование i-го символа текста можно записать так:

$$k_i=\left\lceil\log_2\frac{1-p_N}{p_i}\right\rceil. \tag{12}$$

По Шеннону каждый символ текста можно кодировать используя $log_2\frac{1}{p_i}$ бит. Следовательно, избыточность кодирования одного символа для нового метода будет равна

$$R_i(p_N,p_i)=\left\lceil log_2\frac{1-p_N}{p_i}\right\rceil-log_2\frac{1}{p_i}$$

Теперь можно записать сумму, аналогичную шенноновской энтропии для нахождения среднего количества бит, затрачиваемых на кодирование одного символа:

$$S=\sum_{i=1}^{N}p_i\cdot\left\lceil\log_2\frac{1-p_N}{p_i}\right\rceil$$

Итак, погрешность нового метода равна

$$E = S - H, \tag{13}$$

где H - шенноновская энтропия символа сообщения.

5 Результаты тестирования

Тестирование алгоритма проходило на стандартных тестах "Calgary Text Compression Corpus"[2].

Тестирование алгоритма на тестах Calgary Text Compression Corpus

тест	размер	энтропия символа, бит	среднее число бит на символе	rangecode[3] Михаэля Шиндлера
bib	111261	5.201	5.830	5.279
book1	768771	4.527	5.059	4.598
book2	610856	4.793	5.294	4.854
geo	102400	5.646	6.056	5.709
news	377109	5.190	5.671	5.270
obj1	21504	5.948	6.539	6.136
obj2	246814	6.260	6.576	6.137
paper1	53161	4.983	5.562	5.029
paper2	82199	4.601	5.153	4.674
pic	513216	1.210	1.924	1.238
progc	39611	5.199	5.710	5.291
progl	71646	4.770	5.266	4.825
progp	49379	4.869	5.422	4.970
trans	93695	5.533	6.024	5.600

Ссылки

[1] **Witten Ian H., Neal Radford M., Cleary John G.** Arithmetic coding for data compression. *Communication of the ACM, June 1987, vol. 30, Number 6.*

[2] **Rissanen Jorma, Mohiuddin K.M.** A Multiplication-Free Multialphabet Arithmetic Code. *IEEE Transactions on communications, vol. 37, №2 (1989)*

[2] ftp://ftp.cpsc.ucalgary.ca/pub/projects/text.compression.corpus/

[3] http://www.compressconsult.com/rangecode/