Нижегородский государственный университет им. Н.И. Лобачевского

Федеральное агентство по образованию
Национальный проект «Образование»
Инновационная образовательная программа ННГУ. Образовательно-научный центр
«Информационно-телекоммуникационные системы: физические основы и математическое обеспечение»

Л.П. Жильцова

Современные проблемы теории кодирования

Учебно-методические материалы по программе повышения квалификации «Информационные технологии и компьютерное моделирование в прикладной математике»

Нижний Новгород
2007

Учебно-методические материалы подготовлены в рамках инновационной образовательной программы ННГУ: Образовательно-научный центр «Информационно-телекоммуникационные системы: физические основы и математическое обеспечение»

Жильцова Л.П. Современные проблемы теории кодирования. Учебно-методические материалы по программе повышения квалификации «Информационные технологии и компьютерное моделирование в прикладной математике». Нижний Новгород, 2007, 80 с.

В учебно-методических материалах освещаются вопросы экономного кодирования информации. Для различных классов языков сообщений излагаются теоретические результаты, характеризующие возможности сжатия информации при учете ее вероятностных и структурных свойств. Приводится краткий обзор методов кодирования, применяемых в приложениях.
Для преподавателей, научных работников, аспирантов и студентов, специализирующихся в области дискретной математики и теории кодирования, или интересующихся проблемами теории кодирования.

Оглавление

ВВЕДЕНИЕ

В работе излагаются результаты, относящиеся к одному из направлений теории кодирования, в котором изучаются вопросы сжатия информации. Такое кодирование называется экономным.

Вопросы сжатия информации играют важную роль в информатике. Это связано с развитием вычислительной техники и средств связи и, как следствие, с необходимостью хранения и передачи больших объемов информации.

Возможности сжатия информации определяются ее вероятностными и структурными (синтаксическими) свойствами.

С помощью вероятностей моделируются частоты появления различных букв в сообщениях, частоты фрагментов сообщений, частоты появления самих сообщений.

Дополнительные возможности для сжатия появляются, когда в качестве сообщений рассматриваются не любые последовательности символов, а только некоторые из них. Например, в текстах русского языка не могут встречаться фрагменты "ггг", "аь" и т.д., и это можно учитывать при кодировании.

Начало математическому исследованию вопросов экономного кодирования, учитывающего вероятностные и структурные свойства информации, было положено работой К. Шеннона «Математическая теория связи», опубликованной в 1948 году. При построении алгоритмов кодирования Шенноном учитывались главным образом вероятностные свойства сообщений, порождаемых источником с конечным числом состояний. Однако эти вероятностные свойства определялись как вероятностными свойствами состояний источника, так и синтаксическими (структурными) свойствами последовательностей символов, генерируемых источником.

Вопросы экономного кодирования с учетом структурных свойств информации рассматривались в работах Ал. А. Маркова. Он изучал возможности сжатия сообщений, также генерируемых источниками с конечным числом состояний, при простом с алгоритмической точки зрения способе кодирования – алфавитном, или побуквенном кодировании. Ал. А. Марков показал, что во многих случаях учет структурных свойств при кодировании позволяет более эффективно сжимать информацию.

Продолжением этого направления являлись исследования автора настоящей работы. В них рассматривались вопросы алфавитного кодирования контекстно-свободных языков (КС-языков), являющихся существенным расширением языков сообщений, генерируемых источниками с конечным числом состояний. Оказалось, что в классе контекстно-

свободных языков даже для такого простого способа кодирования, как алфавитное, многие проблемы алгоритмически неразрешимы. К ним относятся проблема распознавания однозначности декодирования и проблема оптимального кодирования. Поэтому в настоящее время при изучении вопросов экономного кодирования для таких сложных классов языков сообщений, как класс КС-языков, большое внимание уделяется построению асимптотически оптимальных алгоритмов экономного кодирования.

Необходимость в исследовании математических вопросов кодирования КС-языков объясняется тем, что КС-языки – одно из ближайших расширений языков, описываемых источниками с конечным числом состояний. КС-языки широко применяются на практике для описания синтаксических свойств естественных языков и языков программирования высокого уровня. Известны также примеры их использования для описания классов геометрических и физических объектов. В настоящее время достаточно большое количество публикаций посвящено использованию контекстно-свободных грамматик для моделирования структурных свойств различных видов информации: звуковой, графической, и др.

Кроме теоретических результатов, относящихся к вопросам кодирования сообщений с учетом их вероятностных и структурных свойств, в работе приводится краткий обзор методов кодирования, которые появились в последние годы и применяются на практике в системах сжатия данных.

ГЛАВА 1. АЛФАВИТНОЕ КОДИРОВАНИЕ

В настоящей главе излагаются математические результаты, связанные с наиболее простым методом кодирования – побуквенным кодированием.

1.1. Основные определения

Пусть $B=\{b_1,b_2,\mathbf{K},b_m\}$ – алфавит. Конечную последовательность символов $b=b_{i_1}b_{i_2}\,\mathbf{K}\,b_{i_n}$ будем называть словом, а число n - длиной слова (длину слова β будем обозначать через $|\beta|$). Через B^* обозначается множество всех слов в алфавите B. Слова из B^* будем называть также сообщениями. Языком сообщений назовем произвольное подмножество $L\subseteq B^*$.

Пусть L – язык. Под двоичным кодированием языка L понимается инъективное отображение $f:L\to\{0,1\}^+$, где $\{0,1\}^+$ - множество всех непустых двоичных последовательностей. Далее под кодированием будет пониматься двоичное кодирование.

Отображение f ставит в соответствие слову $b\in L$ слово $a\in\{0,1\}^+$, a будем называть кодом сообщения b .

Требование инъективности отображения f означает, что различные сообщения должны кодироваться разными двоичными последовательностями. При выполнении этого требования обеспечивается однозначность декодирования сообщений. Такое кодирование называется взаимно-однозначным.

В теории кодирования рассматриваются не любые инъективные отображения f, а те из них, которые могут быть реализованы алгоритмами.

Наиболее изученными являются вопросы экономного кодирования для случая, когда $L=B^*$, свойства информации описываются вероятностями появления букв в сообщениях из B^*, а в качестве способа кодирования применяется побуквенное, или алфавитное, кодирование.

Алфавитное кодирование задается схемой, в которой каждой букве алфавита ставится в соответствие двоичная последовательность символов:

$$f:\begin{cases} b_1\to v_1\\ b_2\to v_2\\ \quad\mathbf{L}\\ b_m\to v_m, \end{cases}\qquad v_i\in\{0,1\}^*.$$

Коды v_i называются элементарными, а их набор $V=(v_1,v_2,\mathbf{K}\,v_m)$ - кодом.

При построении схемы кодирования используется дополнительная информация о вероятностях появления букв – вероятностное распределение $P = (p_1, p_2, \mathbf{K}\, p_m)$.

При построении схем алфавитного кодирования возникают две основные задачи:

1) Проблема распознавания взаимной однозначности алфавитного кодирования;

2) Задача построения схемы кодирования, обеспечивающего наибольшее сжатие.

1.2. Проблема распознавания взаимной однозначности алфавитного кодирования

Пример 1.1. Рассмотрим схему

$$f_1 : \begin{cases} b_1 \to 0 \\ b_2 \to 01. \end{cases}$$

f_1 задает взаимно-однозначное кодирование. Достаточно заметить, что перед каждым вхождением символа 1 стоит символ 0, поэтому каждое вхождение 1 вместе с предшествующим нулем кодирует букву b_2. Символ 0, за которым не следует 1, кодирует букву b_1. Например, последовательность $b = 001010001$ имеет единственную расшифровку $a = b_1 b_2 b_2 b_1 b_1 b_2$.

Пример 1.2. Рассмотрим схему

$$f_2 : \begin{cases} b_1 \to 0 \\ b_2 \to 01 \\ b_3 \to 001. \end{cases}$$

Кодирование, задаваемое схемой f_2, не является взаимно-однозначным. Последовательность $b = 001$ допускает две расшифровки $a_1 = b_1 b_2$ и $a_2 = b_3$.

Для непосредственной проверки взаимной однозначности необходимо в общем случае проверить бесконечное множество пар слов.

Определение 1.1. *Пусть слово* a *имеет вид* $a_1 a_2$. *Тогда* a_1 *называется префиксом слова* a, *а* a_2 - *суффиксом* a. *Если* $0 < |a_1| < |a|$, *то* a_1 *называется собственным префиксом* a, *если* $0 < |a_2| < |a|$, *то* a_2 *– собственный суффикс* a.

Определение 1.2. *Схема алфавитного кодирования* f *обладает свойством префикса, если для любых* i *и* j $(1 \le i, j \le m, i \ne j)$ *элементарный код* v_i *не является префиксом элементарного кода* v_j.

Определение 1.3. *Алфавитное кодирование, схема которого обладает свойством префикса, называется префиксным.*

Теорема 1.1. *Если схема обладает свойством префикса, то алфавитное кодирование является взаимно-однозначным.*

Таким образом, свойство префикса является достаточным условием взаимной однозначности.

Теорема 1.2. *Если алфавитное кодирование со схемой f обладает свойством взаимной однозначности, то длины элементарных кодов* $l_i = |v_i| (i = 1, \mathbf{K}, m)$ *удовлетворяют неравенству Мак-Миллана:*

$$\sum_{i=1}^{m} 2^{-l_i} \le 1.$$

Неравенство Мак-Миллана является необходимым условием взаимной однозначности кода со схемой *f*, но не достаточным.

Рассмотрим схему f_2, приведенную ранее:

$$f_2 : \begin{cases} b_1 \to 0 \\ b_2 \to 01 \\ b_3 \to 001. \end{cases}$$

Для нее $l_1 = 1,$ $l_2 = 2,$ и $l_3 = 3,$ и неравенство Мак-Миллана выполняется: $2^{-1} + 2^{-2} + 2^{-3} = 7/8 < 1.$ Однако задаваемое схемой f_1 кодирование не является взаимно-однозначным.

Теорема 1.3. *Если набор натуральных чисел* $l_1, l_2, \mathbf{K} l_m$ *удовлетворяет неравенству Мак-Миллана, то существует префиксное кодирование, удовлетворяющее равенствам:*

$$|v_1| = l_1, |v_2| = l_2, \mathbf{K}, |v_m| = l_m.$$

Следствие. *Если существует взаимно-однозначное алфавитное кодирование с заданными длинами элементарных кодов, то существует также префиксное кодирование с теми же длинами элементарных кодов.*

Неравенство Мак-Миллана в силу справедливости теоремы 1.3 можно рассматривать и как достаточное условие взаимной однозначности, в том смысле, что если неравенство для некоторой схемы кодирования выполняется, можно заменить эту схему схемой со свойством префикса с теми же длинами элементарных кодов.

Пусть задан код $V = (v_1, v_2, \mathbf{K} v_m)$. Элементарные коды определяют бинарное кодовое дерево. Из каждой вершины дерева выходит не более двух ребер в вершины следующего яруса, левое из которых помечается символом 0, а правое – символом 1. Элементарным

кодам соответствуют вершины дерева, определяемые путем, идущим от корня. Если код префиксный, элементарные коды расположены в листьях дерева.

Дерево называется насыщенным, если из каждой вершины, не являющейся листом, в следующий ярус выходит ровно два ребра.

На рис. 1.1 изображено дерево для схемы префиксного кодирования f_3.

$$f_3: \begin{cases} b_1 \to 00 \\ b_2 \to 111 \\ b_3 \to 100 \\ b_4 \to 01 \\ b_5 \to 101 \\ b_6 \to 1100 \\ b_7 \to 1101. \end{cases}$$

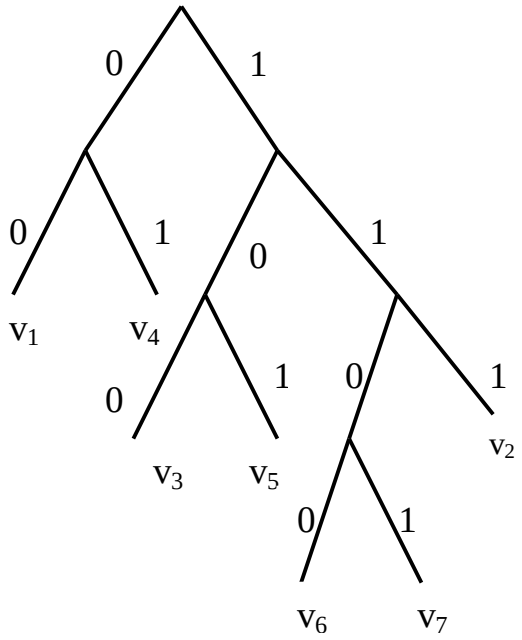


Рис.1.1. Кодовое дерево для схемы кодирования f_3.

Проблема распознавания взаимной однозначности алфавитного кодирования решена Ал. А. Марковым (1963 г.). Алгоритм распознавания состоит в следующем.

Для кода $V = (v_1, v_2, \mathbf{K}, v_m)$ пусть S_1 - множество слов, обладающих следующим свойством. Слово β является собственным префиксом некоторого элементарного кода v_i и одновременно собственным суффиксом некоторого v_j. Положим $S = S_1 \cup \{l\}$ (l - пустое слово).

Сопоставим коду V ориентированный граф G, вершинами которого являются элементы множества S. Вершины α и β соединяем ориентированным ребром (α, β), если существует элементарный код v_j и последовательность элементарных кодов $P = v_{i_1} v_{i_2} \mathbf{K} v_{i_k}$, такие, что $v_j = a v_{i_1} v_{i_2} \mathbf{K} v_{i_k} b$. При этом P может быть пустой, если α и β оба непустые.

Ребру (α, β) припишем последовательность $v_{i_1} v_{i_2} \mathbf{K} v_{i_k}$. Ребро (λ, λ) присутствует в графе тогда и только тогда, когда существует v_j и последовательность $P = v_{i_1} v_{i_2} \mathbf{K} v_{i_k}$ $(k \geq 2)$, такие, что $v_j = v_{i_1} v_{i_2} \mathbf{K} v_{i_k}$.

Теорема 1.4. *Алфавитный код V является взаимно-однозначным тогда и только тогда, когда в графе G отсутствуют ориентированные циклы, проходящие через вершину λ.*

Пример 1.3. Пусть $B = (b_1, b_2, b_3)$, $V = \{1, 010, 101\}$. Построим множества S_1 и S: $S_1 = \{0, 1, 01, 10\}$, $S = \{0, 1, 01, 10, \lambda\}$. Для этого выпишем все нетривиальные разложения элементарных кодов v_i.

Для v_1 нет нетривиальных разложений.

$v_2 = 010 = 0\ v_1\ 0 = 01\ l\ 0 = 0\ l\ 10$

$v_3 = 101 = l\ v_1\ 01 = 10\ v_1\ l = 1\ l\ 01 = 10\ l\ 1.$

Соответствующий коду граф изображен на рис.1.2.

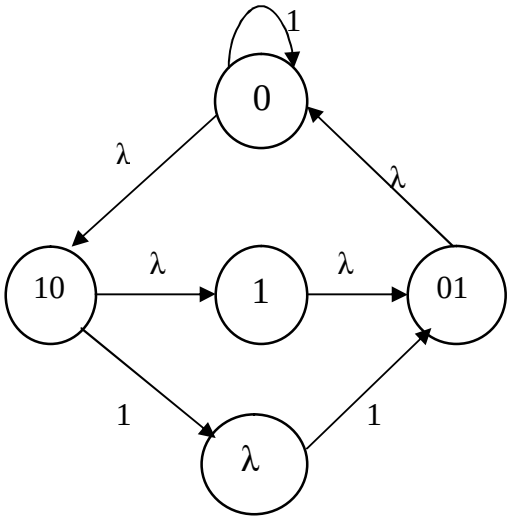


Рис.1.2. Граф для примера 1.3

Граф содержит ориентированный цикл, проходящий через вершину λ, следовательно, код V не является взаимно-однозначным.

По графу нетрудно построить двоичную последовательность, допускающую две расшифровки. Для этого достаточно, начиная с вершины λ, приписать друг к другу двоичные последовательности, соответствующие вершинам и ребрам графа, вдоль найденного цикла.

Слово γ=1010101, соответствующее циклу, допускает две расшифровки: $b_1b_2b_3$ и $b_3b_2b_1$.

Пример 1.4. Пусть $B=(b_1,b_2,b_3,b_4b_5)$, $V=\{1,01,100,0100,0000\}$. Построим множества S_1 и S: $S_1=\{0,00,000,1\}$, $S=\{0,00,000,1,\lambda\}$.

Выпишем все нетривиальные разложения для элементарных кодов.

$v_2 = 01 = 0\ v_1\ \lambda = 0\ \lambda\ 1$

$v_3 = 100 = \lambda\ v_1\ 00 = 1\ \lambda\ 00$

$v_4 = 0100 = 0\ v_1\ 00 = \lambda\ v_2\ 00 = 0\ v_3\ \lambda$

$v_5 = 0000 = 0\ \lambda\ 000 = 00\ \lambda\ 00 = 000\ \lambda\ 0.$

Соответствующий коду граф изображен на рис.1.2.

Граф не содержит ориентированный цикл, проходящий через вершину λ, следовательно, код V является взаимно-однозначным.

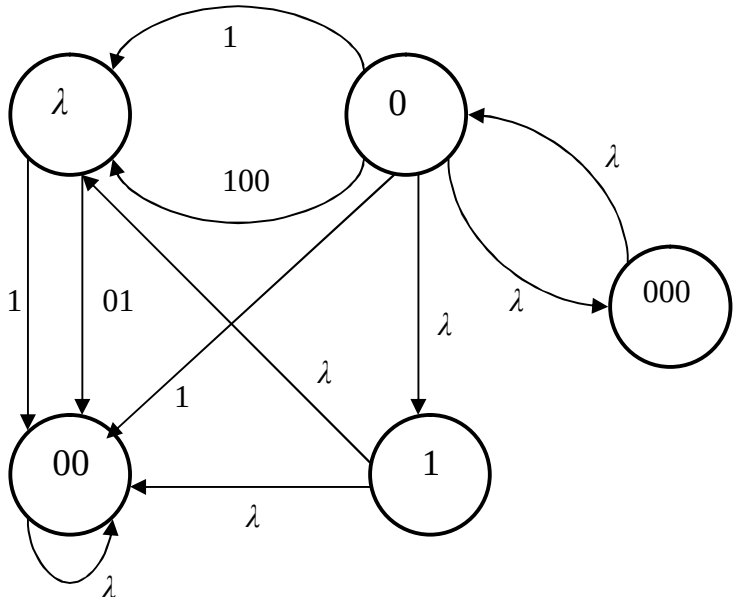


Рис.1.3. Граф для примера 1.4

1.3. Алгоритм построения префиксного кода по набору длин элементарных кодов

Пусть задан набор чисел $l_1, l_2, \mathbf{K} l_m$, удовлетворяющих неравенству Мак-Миллана: $\sum_{i=1}^{m} 2^{-l_i} \le 1$. В силу теоремы 1.3 существует префиксный код с набором длин $(l_1, l_2, \mathbf{K} l_m)$ элементарных кодов. Приведем алгоритм К. Шеннона построения префиксного кода по набору длин.

Будем полагать $l_1 \le l_2 \le \mathbf{K} \le l_m$. Построим последовательность чисел $q_1, q_2, \mathbf{K}, q_m$ по следующим правилам:

$q_1 = 0,$

$q_{i+1} = q_i + 2^{-l_i} (i = 1,2, \mathbf{K}, m-1).$

Очевидно, $0 \le q_i < 1$ и q_i имеет единственное представление в виде двоичной дроби с l_i знаками после запятой:

$q_i = \sum_{j=1}^{l_i} c_j^{(i)} \cdot 2^{-j}$, где $c_j^{(i)}$=0 или $c_j^{(i)}$=1.

Рассмотрим код $V = (v_1, v_2, \mathbf{K}, v_m)$, где $v_i = c_1^{(i)} c_2^{(i)} \mathbf{K} c_{l_i}^{(i)}$.

Так как наборы длин упорядочены по неубыванию, при $h > i$ выполняются неравенства $l_h \ge l_i$ и $q_h \ge q_i + 2^{-l_i}$. Поэтому элементарный код v_h отличается от элементарного кода v_i в l_i первых разрядах. Следовательно, построенный код является префиксным.

Пример 1.5. Рассмотрим набор чисел L=(2,3,3,3,4,4,4). Так как

$2^{-2} + 2^{-3} + 2^{-3} + 2^{-3} + 2^{-4} + 2^{-4} + 2^{-4} = \frac{13}{16} < 1$, неравенство Мак-Миллана выполняется.

Построим последовательность чисел $q_1, q_2, q_3, q_4, q_5, q_6, q_7$, записывая их в двоичной системе счисления.

$$q_1 = 0{,}00$$
$$q_2 = 0 + 2^{-2} = 0{,}010$$
$$q_3 = 0{,}01 + 2^{-3} = 0{,}01 + 0{,}001 = 0{,}011$$
$$q_4 = 0{,}011 + 2^{-3} = 0{,}011 + 0{,}001 = 0{,}100$$
$$q_5 = 0{,}1 + 2^{-3} = 0{,}1 + 0{,}001 = 0{,}1010$$
$$q_6 = 0{,}101 + 2^{-4} = 0{,}101 + 0{,}0001 = 0{,}1011$$
$$q_7 = 0{,}1011 + 2^{-4} = 0{,}1011 + 0{,}0001 = 0{,}1100.$$

Построим схему f алфавитного кодирования, выбирая в качестве элементарного кода v_i последовательность из 0 и 1 длины l_i, образующую дробную часть числа q_i:

$$f:\begin{cases} v_1 = 00 \\ v_2 = 010 \\ v_3 = 011 \\ v_4 = 100 \\ v_5 = 1010 \\ v_6 = 1011 \\ v_7 = 1100. \end{cases}$$

Нетрудно убедиться в том, что построенный код является префиксным.

1.4. Алгоритмы экономного алфавитного кодирования

При построении экономных кодов используется дополнительная информация о вероятностях появления букв в сообщениях.

Пусть на буквах алфавита $B = \{b_1, b_2, \mathbf{K}, b_m\}$ задано распределение вероятностей $P = \{p_1, p_2, \mathbf{K}, p_m\}$, $p_i \geq 0$, $\sum_{i=1}^{m} p_i = 1$.

Под стоимостью кодирования f понимается величина

$$C_f(P) = \sum_{i=1}^{m} p_i \mid v_i \mid$$

(Здесь $\mid v_i \mid$ - длина элементарного кода буквы b_i).

Стоимость кодирования определяет число двоичных разрядов, которые тратятся в среднем на кодирование одной буквы.

$C_f(P)$ - это средняя длина элементарного кода, которая показывает, во сколько раз увеличивается средняя длина слова при кодировании f.

Пример 1.6. Пусть $B = \{b_1, b_2, b_3, b_4\}$, $P = \{0{,}40; 0{,}25; 0{,}20; 0{,}15\}$.

Рассмотрим две схемы алфавитного кодирования и определим для них стоимости кодирования.

$$f_1:\begin{cases} b_1 \to 00 \\ b_1 \to 01 \\ b_1 \to 10 \\ b_1 \to 11, \end{cases} \qquad f_2:\begin{cases} b_1 \to 1 \\ b_1 \to 01 \\ b_1 \to 000 \\ b_1 \to 001. \end{cases}$$

Для f_1 стоимость кодирования $C_{f_1}(P)=2,$ для f_2 стоимость кодирования $C_{f_2}(P)=1{,}95.$ Таким образом, стоимость кодирования может изменяться при переходе от одной схемы кодирования к другой.

Положим $C^*(P)=\inf_f C_f(P).$ Код V^* со схемой f^* такой, что $C_{f^*}(P)=C^*(P),$ называется оптимальным для набора вероятностей P. Можно показать, что величина $C^*(P)$ достигается при некоторой схеме f^* и может быть определена как $\min_f C_f(P).$ Оптимальные коды дают в среднем минимальное увеличение длин слов при соответствующем кодировании. В силу теоремы 1.3 при построении оптимальных кодов можно ограничиться рассмотрением префиксных кодов.

Рассмотрим алгоритмы построения оптимальных и близких к оптимальным кодов.

1.4.1. Алгоритм Хаффмана (1952 г.)

Алгоритм Хаффмана строит оптимальный префиксный код. При рассмотрении алгоритмов кодирования наряду с элементарными кодами вершинам кодового дерева будем приписывать вероятности соответствующих букв. Алгоритм Хаффмана основан на следующих свойствах оптимальных кодов.

Лемма 1.1. *Если код* $V=(v_1,v_2,\mathbf{K},v_m)$ - *оптимальный для* $P=(p_1,p_2,\mathbf{K},p_m),$ *то* $|v_i|\le|v_j|$ *при* $p_i>p_j$.

Из леммы следует, что в оптимальном дереве вероятности букв, приписанные вершинам k-го яруса, не меньше вероятностей, приписанных вершинам $(k+1)$-го яруса.

Лемма 1.2. *Оптимальному префиксному коду соответствует насыщенное кодовое дерево.*

Лемма 1.3. *Две самые маленькие вероятности в оптимальном кодовом дереве находятся на нижнем ярусе. Перестановкой элементарных кодов нижнего яруса их можно поставить в вершины, для которых инцидентные им ребра выходят из одной вершины.*

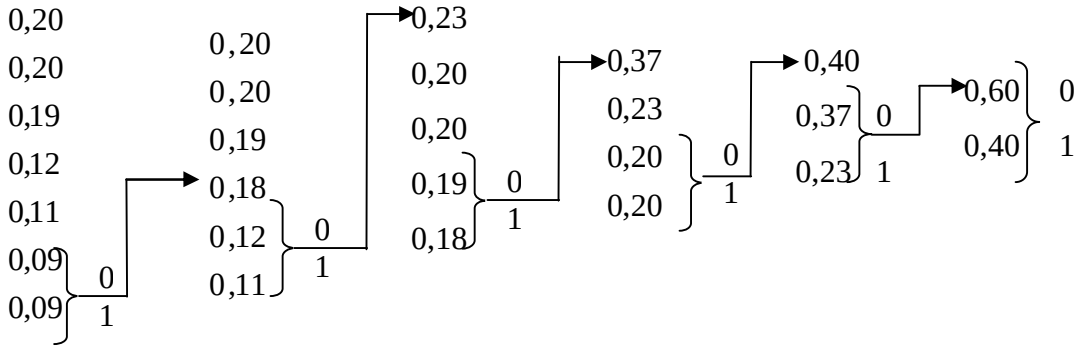
Теорема 1.5 (теорема редукции). *Если код с длинами* $(l_1,l_2,\mathbf{K},l_{m-1},l_m)$ *является оптимальным для распределения вероятностей* $P=(p_1,p_2,\mathbf{K},p_{m-1},p_m),$ *то код с длинами* $(l_1,l_2,\mathbf{K},l_{m-1}-1)$ *также будет оптимальным для распределения вероятностей* $P'=(p_1,p_2,\mathbf{K},p_{m-1}+p_m).$

Теорема редукции позволяет свести задачу построения оптимального кода мощности m к задаче построения оптимального кода мощности m-1. На ней основан алгоритм

Хаффмана, который заключается в следующем. Пусть вероятности в распределении $P=(p_1, p_2, \mathbf{K}, p_m)$ расположены в порядке невозрастания. На каждом шаге объединяются две буквы, имеющие наименьшие вероятности. Вместо этих двух букв вводится новая буква с вероятностью $p = p_{m-1} + p_m$. Вероятность p вставляется в оставшийся набор вероятностей так, чтобы в получившемся новом наборе вероятности остались расположенными в порядке невозрастания. Продолжаем процесс объединения вероятностей до тех пор, пока не останутся две буквы алфавита. Одной из них приписывается символ 0, другой – символ 1 (оптимальный код для двух букв при произвольном распределении вероятностей). Затем из оптимального кода для двух букв строится оптимальный код для трех букв, и т.д. Продолжая этот процесс, придем к искомому оптимальному коду для m букв.

Пример 1.7. Пусть $B=\{b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$, P=(0,20;0,20;0,19;0,12;0,11;0,09;0,09).

Процесс построения оптимального кода можно представить следующим образом:



Фигурными скобками отмечены объединяемые вероятности. Для каждой скобки верхнему члену приписываем символ 0, нижнему – символ 1. Затем осуществляем движение в обратном направлении к $p_1, p_2, \mathbf{K}, p_7$ и, проходя скобки, выписываем соответствующие элементарные коды.

Например, путь 0,60 – 0,23 – 0,11 дает элементарный код 011 для буквы b_5. Таким образом, мы получаем следующую схему f для оптимального кода:

$$f:\begin{cases} b_1 \to 10 \\ b_2 \to 11 \\ b_3 \to 000 \\ b_4 \to 010 \\ b_5 \to 011 \\ b_6 \to 0010 \\ b_7 \to 0011. \end{cases}$$

Стоимость кодирования $C^*(P) = 2{,}78$.

1.4.2. Алгоритм Фано (1961 г.)

Упорядоченный в порядке невозрастания вероятностей список букв делится на две последовательные части так, чтобы суммы вероятностей входящих в них букв как можно меньше отличались друг от друга. Буквам из первой части приписываем символ 0, а буквам из второй части – символ 1. Далее точно так же поступаем с каждой из полученных частей, если она содержит хотя бы две буквы. Построенный код является префиксным, и ему соответствует насыщенное кодовое дерево.

В алгоритме Фано кодовое дерево строится от корня, а в алгоритме Хаффмана – начиная с листьев. Это отличие позволяет в алгоритме Хаффмана полнее использовать специфику данного распределения вероятностей и строить оптимальный код. Алгоритм Фано строит код, близкий к оптимальному.

Пример 1.8. Применим алгоритм Фано к тому же распределению вероятностей.

Пусть $B = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$, P=(0,20;0,20;0,19;0,12;0,11;0,09;0,09).

0,59	0,20	0	0,39	0,20	0	0,20	0
	0,20			0,20	1	0,20	0
	0,19			0,19		0,19	1
0,41	0,12	1	0,23	0,12	0	0,12	0
	0,11			0,11		0,11	1
	0,09		0,18	0,09	1	0,09	0
	0,09			0,09		0,09	1

Получаем следующую схему алфавитного кодирования:

$$f:\begin{cases} b_1 \to 00 \\ b_2 \to 010 \\ b_3 \to 011 \\ b_4 \to 100 \\ b_5 \to 101 \\ b_6 \to 110 \\ b_7 \to 111. \end{cases}$$

Стоимость кодирования $C_{\Phi}(P) = 2{,}80$.

1.4.3. Алгоритм Шеннона (1948 г.)

Алгоритм Шеннона применим в случае, когда все вероятности $p_i > 0$. Букве b_i ставится в соответствие последовательность из $l_i = \left\lceil \log \frac{1}{p_i} \right\rceil$ двоичных символов (здесь $\lceil x \rceil$ - ближайшее целое сверху числа x и $\log$ здесь и везде далее берется по основанию 2). Алгоритм Шеннона основан на том, что выбранные длины l_i (i=1,2, …, m) удовлетворяют неравенству Мак-Миллана. После выбора длин применяется алгоритм Шеннона построения схемы кодирования по заданному набору длин элементарных кодов, описанный ранее.

Пример 1.9.

Пусть $B = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$, P=(0,20;0,20;0,19;0,12;0,11;0,09;0,09).

Вычислим набор длин для P.

$$l_1 = l_2 = \left\lceil \frac{1}{0{,}20} \right\rceil = 3,$$

$$l_3 = \left\lceil \frac{1}{0{,}19} \right\rceil = 3,$$

$$l_4 = \left\lceil \frac{1}{0{,}12} \right\rceil = 4,$$

$$l_5 = \left\lceil \frac{1}{0{,}11} \right\rceil = 4,$$

$$l_6 = l_7 = \left\lceil \frac{1}{0{,}09} \right\rceil = 4.$$

Построим префиксный код по алгоритму Шеннона с вычисленными длинами элементарных кодов.

$$f:\begin{cases} b_1 \to 000 \\ b_2 \to 001 \\ b_3 \to 011 \\ b_4 \to 1001 \\ b_5 \to 1011 \\ b_6 \to 1101 \\ b_7 \to 1110. \end{cases}$$

Стоимость кодирования $C_{Ш}(P) = 3{,}41.$

1.4.4. Энтропия и ее связь со стоимостью оптимального алфавитного кодирования

Важную роль для оценки эффективности кодирования играет энтропия вероятностного распределения:

$$H(P) = -\sum_{i=1}^{m} p_i \log p_i.$$

Пусть $C^*(P)$ - стоимость оптимального алфавитного кодирования.

Теорема 1.6. $C^*(P) \geq H(P).$

Доказательство. Будем использовать неравенство $\log x \leq (x-1)\cdot \log e.$ Так как для длин элементарных кодов выполняется неравенство Мак-Миллана, $\sum_{i=1}^{m} 2^{-l_i} \leq 1.$

Рассмотрим разность *H(P)-C*(P):*

$$H(P) - C^*(P) = -\sum_{i=1}^{m} p_i \log p_i - \sum_{i=1}^{m} p_i l_i = \sum_{i=1}^{m} p_i \left(\log \frac{1}{p_i} - l_i \right) =$$

$$= \sum_{i=1}^{m} p_i \left(\log \frac{1}{p_i} + \log 2^{-l_i} \right) = \sum_{i=1}^{m} p_i \left(\log \frac{2^{-l_i}}{p_i} \right) \leq \log e \sum_{i=1}^{m} p_i \left(\frac{2^{-l_i}}{p_i} - 1 \right) =$$

$$= \log e \left(\sum_{i=1}^{m} 2^{-l_i} - \sum_{i=1}^{m} p_i \right) = \log e \left(\sum_{i=1}^{m} 2^{-l_i} - 1 \right) \leq 0.$$

Отсюда $C^*(P) \geq H(P).$

Теорема доказана.

При некоторых распределениях стоимость $C^*(P)$ может достигать нижней границы.

Пример 1.10. Рассмотрим распределение вероятностей $P_m = \left(\frac{1}{2}, \frac{1}{2^2}, \mathbf{K}, \frac{1}{2^{m-2}}, \frac{1}{2^{m-1}}, \frac{1}{2^{m-1}} \right)$ Вычислим $C(m) = C(P_m).$

Положим $l_i = \left\lceil \log \frac{1}{p_i} \right\rceil$. Получим

$$l_1 = 1,\ l_2 = 2, \mathbf{K},\ l_i = i, \mathbf{K},\ l_{m-2} = m-2,\ l_{m-1} = l_m = m-1.$$

Величины l_i удовлетворяют неравенству Мак-Миллана, следовательно, существует префиксный код с таким набором длин элементарных кодов. Так как p_i являются степенью двойки, $l_i = -\log p_i$, поэтому $C(P_m) = -\sum_{i=1}^{m} p_i \log p_i = H(P_m)$.

В общем же случае $C^*(P) = H(P) + e$, где $0 \le e < 1$, как показывает следующая теорема.

Теорема 1.7. $C^*(P) < H(P) + 1$.

Доказательство. Возьмем $l_i = -\log p_i$, $(i = 1, \mathbf{K}, m)$. Тогда $2^{-l_i} \le p_i < 2^{-(l_i - 1)}$, откуда $\sum_{i=1}^{m} 2^{-l_i} \le \sum_{i=1}^{m} p_i = 1$. Т.е. набор длин $(l_1, \mathbf{K}, l_m)$ реализуем.

Но $l_i < -\log p_i + 1$, откуда $p_i l_i < -p_i \log p_i + p_i$, и суммируя по i, получаем:

$$C^*(P) \le \sum_{i=1}^{m} p_i l_i < -\sum_{i=1}^{m} p_i \log p_i + \sum_{i=1}^{m} p_i = H(P) + 1.$$

Теорема доказана.

Стоимость оптимального кодирования может быть как угодно близка и к верхней оценке.

Дополнительные возможности для сжатия могут возникнуть при конечно-автоматном кодировании. Вместо того, чтобы кодировать каждую букву, разобьем сообщение на блоки длины *N*, которые и будем кодировать как буквы нового алфавита B_N.

Пусть P_N - распределение вероятностей на B_N, которое индуцируется распределением *P* на *B*:

$$p_{i_1 i_2 \mathbf{K} i_N} = p(b_{i_1} b_{i_2} \mathbf{K}\ b_{i_N}) = p_{i_1} p_{i_2} \mathbf{K}\ p_{i_N}.$$

Теорема 1.8. $H(P_N) = N \cdot H(P)$.

Доказательство проведем индукцией по *N*. При *N*=1 утверждение теоремы тривиально. Пусть теорема верна при $N = 2, \mathbf{K}, k-1$. Тогда

$$H(P_k) = -\sum_{\langle i_1 \mathbf{K} i_k \rangle} p_{i_1} p_{i_2} \mathbf{K}\, p_{i_k} \log(p_{i_1} p_{i_2} \mathbf{K}\, p_{i_k}) = -p_1 \sum_{\langle i_1 \mathbf{K} i_{k-1} \rangle} p_{i_1} p_{i_2} \mathbf{K}\, p_{i_{k-1}} \left(\log p_{i_1} p_{i_2} \mathbf{K}\, p_{i_{k-1}} + \log p_1\right) -$$

$$- \mathbf{K} - p_m \sum_{\langle i_1 \mathbf{K} i_{k-1} \rangle} p_{i_1} p_{i_2} \mathbf{K}\, p_{i_{k-1}} \left(\log p_{i_1} p_{i_2} \mathbf{K}\, p_{i_{k-1}} + \log p_m\right) =$$

$$= p_1 \cdot H(P_{k-1}) - p_1 \cdot \log p_1 + \mathbf{K} + p_m \cdot H(P_{k-1}) - p_m \cdot \log p_m = \sum_{i=1}^{m} p_i \cdot H(P_{k-1}) + H(P) =$$

$$= H(P_{k-1}) + H(P) = (k-1) \cdot H(P) + H(P) = k \cdot H(P).$$

Теорема доказана.

Покажем, что, выбирая длину блока N достаточно большой, можно сделать стоимость кодирования на одну букву сообщения $C_N(P)$ сколь угодно близкой к $H(P)$.

Теорема 1.9. $H(P) \le C_N(P) < H(P) + \frac{1}{N}$.

Доказательство. Имеем:

$$N \cdot H(P) = H(P_N) \le C(P_N) = N \cdot C_N(P) < H(P_N) + 1 = N \cdot H(P) + 1.$$

Отсюда получаем:

$$H(P) \le C_N(P) < H(P) + \frac{1}{N}.$$

При $N \to \infty$ $C_N(P) \to H(P)$. Теорема доказана.

Таким образом, увеличивая длину блока, мы приближаемся сколь угодно близко к нижней границе.

Пример 1.11. Пусть $B = \{b_1, b_2, b_3\}$, P={0,5;0,4;0,1}.

Применяя алгоритм Хаффмана к распределению P, получаем следующую схему кодирования:

$$f_1 : \begin{cases} b_1 \to 1 \\ b_1 \to 00 \\ b_1 \to 01. \end{cases}$$

Вычислим стоимость оптимального кодирования: $C_1 = 1{,}5$.

Положим N = 2 и рассмотрим всевозможные блоки длины 2. Определим произведение вероятностей каждого блока как произведение вероятностей входящих в него букв:

$$p(b_1b_1) = 0{,}25; \quad p(b_2b_1) = 0{,}20; \quad p(b_3b_1) = 0{,}05;$$
$$p(b_1b_2) = 0{,}20; \quad p(b_2b_2) = 0{,}16; \quad p(b_3b_2) = 0{,}04;$$
$$p(b_1b_3) = 0{,}05; \quad p(b_2b_3) = 0{,}04; \quad p(b_3b_3) = 0{,}01.$$

Применяя алгоритм Хаффмана к построенному вероятностному распределению, получим следующую схему кодирования для блоков длины N=2:

$$f_2 : \begin{cases} b_1b_1 \to 01 \\ b_1b_2 \to 10 \\ b_1b_3 \to 00000 \\ b_2b_1 \to 11 \\ b_2b_2 \to 001 \\ b_2b_3 \to 00011 \\ b_3b_1 \to 00001 \\ b_3b_2 \to 000100 \\ b_3b_3 \to 000101. \end{cases}$$

Построенная схема имеет стоимость кодирования одного блока $C_2 = 2{,}78,$ и стоимость кодирования одной буквы $\frac{C_2}{2} = 1{,}39.$ Найдем энтропию $H(P)$:

$$H(P) = -(0{,}5 \cdot \log 0{,}5 + 0{,}4 \cdot \log 0{,}4 + 0{,}1 \cdot \log 0{,}1) \approx 1{,}36.$$

Получаем $H(P) \approx 1{,}36 < \frac{C_2}{2} = 1{,}39 < C_1 = 1{,}5.$

1.4.5. Возможности сжатия при алфавитном кодировании, учитывающем синтаксис языка сообщений

Дополнительные возможности для сжатия появляются при $L \subset B^*$, когда в качестве сообщений могут быть не любые последовательности символов, а только некоторые из них.

Пример 1.12 алфавитного кодирования, учитывающего синтаксис языка сообщений:

В качестве языка L рассмотрим множество слов в алфавите $B = \{b_1, b_2, b_3\}$, не содержащих диграмму b_1b_2. Синтаксис этого языка описывается источником с двумя состояниями, изображенным на рис.1.4.

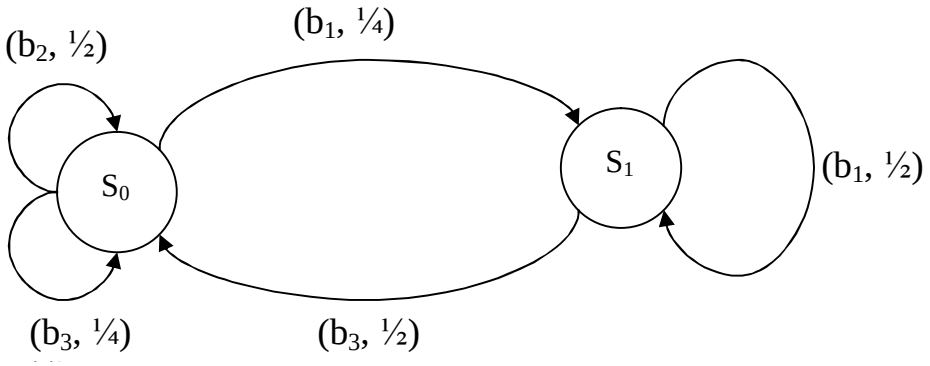


Рис. 1.4. Источник, генерирующий сообщения, не содержащие диграмму b_1b_2.

Приведем для этого языка две схемы кодирования. Первая схема построена без учета синтаксиса языка, вторая учитывает запрещенную диграмму b_1b_2.

Так как диграмма b_1b_2 не может встречаться в словах языка L, буква b_3 может быть закодирована последовательностью 10.

Насколько эффективно можно использовать те или иные свойства языка для сжатия информации при алфавитном кодировании?

В языке сообщений может присутствовать алфавитная избыточность: некоторые буквы алфавита могут быть фиктивными или контекстно-различимыми. Поясним эти понятия.

Определение 1.4. *Пусть* $B=\{b_1,\mathbf{K},b_m\}$ *и* $L\subseteq B^*$. *Буква* b_i *называется фиктивной в* L, *если отображение* $a\to a'$, *состоящее в замене* b_i *пустым словом* l *во всех вхождениях* b_i *в* a, *таково, что из* $a,b\in L$ *и* $a\neq b$ *следует* $a'\neq b'$. *В противном случае буква* b_i *называется существенной в* L.

Буквы b_i *и* b_j $(i\neq j)$ *называются контекстно различимыми в* L, *если отображение* $a\to a'$, *состоящее в замене* b_i *буквой* b_j *во всех вхождениях* b_i *в* a, *таково, что из* $a,b\in L$ *и* $a\neq b$ *следует* $a'\neq b'$.

Определение 1.5. *Язык* L *называется неприводимым, если все его буквы существенные и попарно контекстно неразличимы. В противном случае говорят, что* L *допускает алфавитную редукцию.*

За счет алфавитной избыточности можно сжимать сообщения языка в любое число

$f_1: B^*\to\{0,1\}^*$	$f_2: L\to\{0,1\}^*$
Алгоритм Хаффмана	*С учетом структуры*
$b_1\to 1$	$b_1\to 1$
$b_2\to 00$	$b_2\to 0$
$b_3\to 01$	$b_3\to 10$

раз.

Пример 1.13. Пусть $L_1=\{(abc)^n : n=1,2,\mathbf{K}\}$.

Рассмотрим схему кодирования

$$f_1 : \begin{cases} a \to 0 \\ b \to l \\ c \to l. \end{cases}$$

Схема f_1 задает взаимно-однозначное кодирование языка L_1, так как фрагмент abc может быть закодирован одним символом 0, для взаимной однозначности кодирования достаточно каким-то образом кодировать число вхождений фрагмента abc в слово языка.

Пример 1.14. Пусть $L_2 = \{a^+ b\} \cup \{c^*\}$. (Здесь a^+ означает произвольную непустую последовательность букв a).

Буквы b и c контекстно различимы, их можно кодировать одинаково. Поэтому следующая схема кодирования f_2 задает взаимно-однозначное кодирование языка L_2:

$$f_1 : \begin{cases} a \to 0 \\ b \to 1 \\ c \to 1. \end{cases}$$

Пусть L неприводимый язык и $F(L)$ – множество всех взаимно-однозначных кодирований языка L, задаваемых схемами алфавитного кодирования.

Пусть на множестве букв алфавита B языка L задано распределение вероятностей P. Обозначим через $C^*(L,P)$ стоимость оптимального алфавитного кодирования, учитывающего синтаксис языка L, и через $C^*(B^*,P)$ стоимость кодирования по алгоритму Хаффмана (обозначавшуюся ранее как $C^*(P)$). Для $a \in L$ через $C^*(L,a)$ обозначим $|f^*(a)|$, где f^* - оптимальное алфавитное кодирование, учитывающее синтаксис языка L, и через $C^*(B^*,a)$ - длину кода слова $a \in L$, построенного по алгоритму Хаффмана.

В качестве меры эффективности кодирования, учитывающего синтаксис языка L, рассмотрим коэффициент сжатия

$$Z(L) = \inf \left\{ \frac{C^*(L,a)}{C^*(B^*,a)} : a \in L \right\}.$$

Теорема 1.10. *Пусть L – неприводимый язык. Тогда*

$$1/2 \le Z(L) \le 1.$$

Из теоремы следует, что неприводимый язык может быть сжат с помощью алфавитного кодирования не более чем в два раза.

Теорема 1.11. *Для почти всех неприводимых языков*

$$Z(L) = 1.$$

Таким образом, почти все неприводимые языки несжимаемы с помощью алфавитного кодирования.

При рассмотрении алфавитного кодирования, учитывающего синтаксические свойства языка, возникают две задачи:

1. Проблема распознавания взаимной однозначности алфавитного кодирования.
2. Задача оптимального кодирования.

Проблема распознавания взаимной однозначности алфавитного кодирования для языков, описываемых источниками с конечным числом состояний, решена Ал.А.Марковым (1961 г.)

Задача оптимального алфавитного кодирования для конечных источников сложная – она остается открытой. Задача решена для отдельных подклассов источников.

ГЛАВА 2. КОДИРОВАНИЕ ВЕРОЯТНОСТНЫХ ИСТОЧНИКОВ С КОНЕЧНЫМ ЧИСЛОМ СОСТОЯНИЙ

Впервые задачу экономного кодирования, учитывающего синтаксические свойства сообщений наряду с вероятностными свойствами, рассмотрел К. Шеннон в работе «Математическая теория связи» (1948 г.) Для моделирования синтаксических свойств он использовал эргодический источник с конечным числом состояний. Дискретный источник представляется некоторым марковским процессом. Для каждого состояния задано свое распределение вероятностей на множестве букв алфавита, т.е. вероятность появления буквы зависит от того, в каком состоянии находится в данный момент источник. Свойство эргодичности источника означает, что ориентированный граф, с вершинами – состояниями и дугами, определяющими переходы от одного состояния к другому, является сильно связным, т.е. для любой пары состояний (S_1, S_2) существует ориентированный путь из S_1 в S_2. Кроме того, источник должен быть непериодическим, но это требование не является принципиальным.

Для задания источника I необходимо для каждого состояния S_i и каждой буквы b_j определить p_{ij} - вероятность появления буквы b_j в состоянии S_i, где $\sum_{j=1}^{m} p_{ij} = 1$ для всех i=1, …, n , причем $p_{ij} > 0$ в том и только том случае, когда в состоянии S_i может появляться буква b_j, т.е. существует дуга, выходящая из состояния S_i, помеченная буквой b_j.

Через $\overline{p}_{ij}$ обозначим сумму вероятностей букв, переводящих источник из состояния S_i в состояние S_j.

Для эргодического источника через $P = (p_1, p_2, \mathbf{K}, p_n)$ обозначим распределение вероятностей на множестве состояний, p_i вероятность попадания источника в состояние S_i.

Вероятности p_i удовлетворяют системе уравнений:

$$\begin{cases} p_j = \sum_{j=1}^{n} p_i \cdot \overline{p}_{ij}, \\ \sum_{i=1}^{n} p_i = 1. \end{cases}$$

Соотношениями $p_i = \sum_{j=1}^{m} p_j \cdot p_{ji}, (i = 1, \mathbf{K}, m)$ определяются финальные вероятности букв алфавита *B*. Роль этих финальных вероятностей при анализе достаточно длинных сообщений языка вытекает из усиленного закона больших чисел:

Если $m_i(N)$ - число вхождений буквы b_i в сообщение длины *N*, то для любого i и любого $e > 0$ имеет место

$$\lim_{N \to \infty} P\left(\left| \frac{m_i(N)}{N} - p_i \right| < e \right) = 1,$$

т.е. почти все сообщения длины *N*, порождаемые источником, имеют частотную характеристику, близкую к $p(I) = (p_1, p_2, \mathbf{K}, p_m)$. Поэтому $p(I)$ может рассматриваться как «типичная» частотная характеристика достаточно длинных сообщений. Это также означает, что частотная характеристика сообщений является статистически устойчивой, и с хорошей степенью приближения может быть определена выборочными исследованиями экспериментально.

Другая характеристика вероятностного источника *I* – энтропия *H(I)*:

$$H(I) = -\sum_i p_i \sum_j p_{ij} \log p_{ij} .$$

Теорема 2.1 (теорема Шеннона):

1) *Если P(N) – вероятность порождения источником I сообщения длины N, то для любого* $e > 0$ *имеет место*

$$\lim_{N \to \infty} P\left(\left| \frac{\log 1/p(N)}{N} - H(I) \right| < e \right) = 1.$$

Все сообщения достаточно большой длины N разбиваются на две группы:

маловероятные и высоковероятные (вероятность каждого сообщения приблизительно равна $2^{-NH(I)}$;

2) Стоимость любого кодирования, имеющего однозначное декодирование, ограничена снизу величиной H(I) (в качестве стоимости кодирования рассматривается число двоичных разрядов, приходящееся на кодирование одной буквы сообщения);

3) Для любого ε>0 существует равномерное (блочное) кодирование f такое, что его стоимость $C_f < H(I) + \varepsilon$.

Таким образом, алгоритм блочного кодирования является асимптотически оптимальным и для эргодических источников.

При его построении Шеннон главным образом учитывал вероятностные свойства сообщений. Алгоритм блочного кодирования при реализации требует больших вычислительных ресурсов.

Пример 2.1 кодирования для эргодического источника с двумя состояниями.

$B = \{b_1, b_2, b_3\}$, источник порождает язык сообщений L – множество слов, не содержащих диграмму $b_1 b_2$ (см. рис.1.4).

Составим систему уравнений для $P = (p_1, p_2)$:

$$\begin{cases} p_1 = p_1 \cdot \dfrac{3}{4} + p_2 \cdot \dfrac{1}{2} \\ p_2 = p_1 \cdot \dfrac{1}{4} + p_2 \cdot \dfrac{1}{2} \\ p_1 + p_2 = 1. \end{cases}$$

Решая систему, получаем: $p_1 = \dfrac{2}{3}, p_2 = \dfrac{1}{3}.$

Используя найденное решение, вычислим вероятности появления букв:

$$p = (p_1, p_2, p_3) = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$$

Найдем энтропию источника: $H(I) = \dfrac{4}{3}.$ Следовательно, нижней оценкой любого кодирования является значение $\dfrac{4}{3}$.

Поскольку буквы алфавита B равновероятны в сообщениях большой длины, применяя к $p = = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ алгоритм Хаффмана, получаем стоимость кодирования $C^*(p) = \dfrac{5}{3}.$ Видно, что стоимость алфавитного кодирования отличается от нижней границы на 1/3.

Применяя блочное равномерное кодирование и увеличивая длину блока, можно приблизиться к $\dfrac{4}{3}$ с любой степенью точности.

ГЛАВА 3. ВОПРОСЫ КОДИРОВАНИЯ СТОХАСТИЧЕСКИХ ЯЗЫКОВ. СООТНОШЕНИЕ МЕЖДУ СТОИМОСТЬЮ ОПТИМАЛЬНОГО КОДИРОВАНИЯ И ЭНТРОПИЕЙ СТОХАСТИЧЕСКОГО ЯЗЫКА

В этой главе исследуются вопросы, связанные с двоичным кодированием стохастических языков. Произвольный стохастический язык – это множество слов (сообщений), на котором задано распределение вероятностей. В качестве стоимости кодирования рассматривается математическое ожидание длины закодированного слова стохастического языка. Целью оптимального кодирования является минимизация стоимости кодирования.

Определение стоимости кодирования в этой главе существенно отличается от определения, рассматриваемого К. Шенноном. Основное отличие состоит в том, что стоимость кодирования определяется на множестве всех слов языка. Поэтому на величину стоимости кодирования влияют как короткие, так и длинные сообщения.

Для произвольного языка с конечным значением энтропии установлены верхняя и нижняя неулучшаемые оценки стоимости оптимального кодирования, зависящие только от энтропии. Показано, что стоимость оптимального кодирования произвольного стохастического языка имеет конечное значение тогда и только тогда, когда конечное значение имеет энтропия языка.

3.1. Основные определения, относящиеся к кодированию стохастических языков

Пусть B- алфавит, и $L \subseteq B^*$ - язык в алфавите B.

Пусть на множестве слов языка L задано распределение вероятностей P. Через $p(a)$ обозначим вероятность слова a. Будем рассматривать только такое распределение вероятностей, для которого $p(a) > 0$ для любого $a \in L$. Множество $\boldsymbol{L} = \{(a, p(a)) : a \in L\}$ называется стохастическим языком. Для стохастического языка будем применять запись $\boldsymbol{L}=(L,P)$.

Пусть $\boldsymbol{L}=(L,P)$ - стохастический язык. Через $F(L)$ обозначим класс всех инъективных отображений из L в $\{0,1\}^*$.

Для $f \in F(L)$ величину

$$C(\boldsymbol{L}, f) = \lim_{N \to \infty} \sum_{a \in L, |a| \le N} p(a) \cdot |f(a)|$$

назовем стоимостью кодирования f.

Если существует конечное значение $C(\boldsymbol{L}, f)$, будем сокращенно записывать

$$C(\boldsymbol{L}, f) = \sum_{a \in L} p(a) \cdot | f(a) |.$$

Величину $C_0(\boldsymbol{L}) = \inf_{f \in F} C(\boldsymbol{L}, f)$ будем называть стоимостью оптимального кодирования, а кодирование f_0, для которого $C(\boldsymbol{L}, f_0) = C_0(\boldsymbol{L})$, будем называть оптимальным.

Оптимальное кодирование для любого стохастического языка $\boldsymbol{L}$ существует и состоит в упорядочении слов в соответствии с невозрастанием их вероятностей и кодировании их в алфавите {0,1} сначала словами длины 1, потом словами длины 2, и т.д.

Под энтропией стохастического языка $\boldsymbol{L}=(L,P)$ будем понимать величину

$$H(\boldsymbol{L}) = \lim_{N \to \infty} - \sum_{a \in L, |a| \le N} p(a) \cdot \log p(a)$$

(логарифм здесь и везде далее берется по основанию 2).

Если существует конечное значение $H(\boldsymbol{L})$, будем писать

$$H(\boldsymbol{L}) = -\sum_{a \in L} p(a) \cdot \log p(a).$$

Отметим, что если $H(\boldsymbol{L})$ конечна, то, так как ряд положительный, он абсолютно сходится, поэтому в нем можно переставлять слагаемые. Следовательно, неважно, как упорядочены слова языка L при вычислении $H(\boldsymbol{L})$.

3.2. Соотношение между стоимостью оптимального кодирования и энтропией для произвольного стохастического языка

Теорема 3.1.

а) Пусть $\boldsymbol{L}=(L,P)$ - произвольный бесконечный стохастический язык, для которого существует конечное значение $H(\boldsymbol{L})$. Тогда стоимость оптимального кодирования $C_0(\boldsymbol{L})$ имеет конечное значение и удовлетворяет неравенствам

$$\frac{1}{2} H(\boldsymbol{L}) \le C_0(\boldsymbol{L}) < H(\boldsymbol{L}) + 1.$$

б) Пусть $L = \{a_1, a_2, \mathbf{K}\}$ - произвольный бесконечный язык.

Существует распределение вероятностей $P^* = \{p_1^*, p_2^*, \mathbf{K}\}$ такое, что для стохастического языка $\boldsymbol{L} = \{(a_i, p_i^*) : i = 1,2, \mathbf{K}\}$ энтропия $H(\boldsymbol{L})$ имеет конечное значение и выполняется равенство

$$C_0(\boldsymbol{L}) = \frac{1}{2} H(\boldsymbol{L}).$$

в) Пусть ***L=(L,P)*** - произвольный бесконечный стохастический язык. Для любого $e > 0$ существует распределение вероятностей $P_e = (p_1(e), p_2(e), \mathbf{K})$, такое, что для стохастического языка $\boldsymbol{L}_e = = \{(a_i, p_i{*}(e)) : i = 1,2, \mathbf{K}\}$ выполняется неравенство

$$C_0(\boldsymbol{L}_e) > H(\boldsymbol{L}_e) + 1 - e.$$

Доказательство.

а) Пусть $L = \{a_1, a_2, \mathbf{K}\}$, $P = (p_1, p_2, \mathbf{K})$ и, для определенности, $p_1 \geq p_2 \geq \mathbf{K}$, где p_i - вероятность слова a_i. Рассмотрим последовательность чисел $D = (d_1, d_2, \mathbf{K}, d_i, \mathbf{K})$, в которой $d_i = \lceil -\log p_i \rceil$ для любого i (здесь $\lceil x \rceil$ - ближайшее целое сверху). Последовательность *D* удовлетворяет неравенству Мак-Миллана:

$$\sum_{i=1}^{\infty} 2^{-d_i} \leq 1, \text{ так как } \sum_{i=1}^{\infty} 2^{-\lceil -\log p_i \rceil} \leq \sum_{i=1}^{\infty} 2^{\log p_i} = \sum_{i=1}^{\infty} p_i = 1.$$

Нетрудно показать, что существует бесконечный префиксный код f_1 с заданным набором длин *D* кодов слов из *L*. Для этого достаточно применить без каких-либо изменений метод построения префиксного кода по заданному конечному набору длин, описанный выше.

Оценим стоимость кодирования для f_1:

$$C(\boldsymbol{L}, f_1) = \sum_{i=1}^{\infty} p_i \cdot \lceil -\log p_i \rceil = \sum_{i=1}^{\infty} p_i \cdot (-\log p_i + e_i) =$$

$$= H(\boldsymbol{L}) + \sum_{i=1}^{\infty} p_i \cdot e_i < H(\boldsymbol{L}) + \sum_{i=1}^{\infty} p_i = H(\boldsymbol{L}) + 1$$

(здесь $e_i = \lceil -\log p_i \rceil + \log p_i$ и, следовательно, e_i <1 для любого *i*).

Так как $C_0(\boldsymbol{L}) \leq C(\boldsymbol{L}, f_1)$, то $C_0(\boldsymbol{L})$ также конечна и $C_0(\boldsymbol{L}) < H(\boldsymbol{L}) + 1$.

Докажем неравенство $\frac{1}{2} H(\boldsymbol{L}) \leq C_0(\boldsymbol{L})$.

Длина оптимального кода для слова a_i равна $\lfloor \log(i+1) \rfloor$.

Рассмотрим разность:

$$\mathrm{H}(\boldsymbol{L}) - 2C_0(\boldsymbol{L}) = \sum_{i \geq 1} p_i \cdot \left(\log \frac{1}{p_i} - 2 \cdot \lfloor \log(i+1) \rfloor \right) = \sum_{i \geq 1} p_i \cdot \log \frac{2^{-2\lfloor \log(i+1) \rfloor}}{p_i}.$$

Используя неравенство $\log x \leq (x-1) \cdot \log e$, получим:

$$\sum_{i\ge 1} p_i \cdot \log \frac{2^{-2\lfloor \log(i+1)\rfloor}}{p_i} \le \log e \cdot \sum_{i\ge 1} p_i \cdot \left(\frac{2^{-2\lfloor \log(i+1)\rfloor}}{p_i} - 1 \right) =$$

$$= \log e \cdot \left(\sum_{i\ge 1} 2^{-2\lfloor \log(i+1)\rfloor} - \sum_{i\ge 1} p_i \right) = \log e \cdot \left(\sum_{i\ge 1} 2^{-2\lfloor \log(i+1)\rfloor} - 1 \right).$$

Оценим сумму:

$$\sum_{i\ge 1} 2^{-2\lfloor \log(i+1)\rfloor} = \sum_{i\ge 1} \left(\frac{1}{4} \right)^{\lfloor \log(i+1)\rfloor} = 2 \cdot \frac{1}{4} + 4 \cdot \frac{1}{4^2} + 8 \cdot \frac{1}{4^3} + \mathbf{K} =$$

$$= \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \mathbf{K} = 1.$$

Окончательно получаем, что

$H(\boldsymbol{L}) - 2C_0(\boldsymbol{L}) \le 0,$ откуда следует $\frac{1}{2} H(\boldsymbol{L}) \le C_0(\boldsymbol{L}).$

б) Рассмотрим произвольный бесконечный язык $L = \{a_1, a_2, \mathbf{K}\}$ и распределение вероятностей $P^* = \{p_1^*, p_2^*, \mathbf{K}, p_i^*, \mathbf{K}\},$ где $p_i = \frac{1}{2^{2\lfloor \log(i+1)\rfloor}}$ для i = 1,2, … .

Нетрудно показать, что $\sum_{i\ge 1} p_i^* = 1,$ и P^* действительно является распределением вероятностей.

Для стохастического языка $\boldsymbol{L} = \{(a_i, p_i^*) : i = 1,2, \mathbf{K}\}$ мы имеем

$$H(\boldsymbol{L}) = \sum_{i\ge 1} \frac{2\lfloor \log(i+1)\rfloor}{2^{2\lfloor \log(i+1)\rfloor}} = \frac{2}{2^2} \cdot 2 + \frac{2 \cdot 2}{2^4} \cdot 4 +$$

$$) = \sum_{i\ge 1} \frac{2\lfloor \log(i+1)\rfloor}{2^{2\lfloor \log(i+1)\rfloor}} = \frac{2}{2^2} \cdot 2 + \frac{2 \cdot 2}{2^4} \cdot 4 +$$

$$+ \frac{3 \cdot 2}{2^{3 \cdot 2}} \cdot 2^3 + \mathbf{K} + \frac{j \cdot 2}{2^{2j}} \cdot 2^j + \mathbf{K} = \sum_{i\ge 1} \frac{2j}{2^j} = 4,$$

т.е. $H(\boldsymbol{L})$ конечна.

Вычислим стоимость оптимального кодирования:

$C_0(\boldsymbol{L}) = \sum_{i\ge 1} \frac{\lfloor \log(i+1)\rfloor}{2^{\lfloor \log(i+1)\rfloor}} = 2,$ следовательно, $\frac{1}{2} H(\boldsymbol{L}) = C_0(\boldsymbol{L}).$

с) Рассмотрим произвольный бесконечный язык $L = \{a_1, a_2, \mathbf{K}\}$ и распределение вероятностей $P',$ построенное по P^* из пункта (б), для которого

$$p_1' = 1 - d \quad \text{и} \quad p_i' = d \cdot p_{i-1}^* \quad \text{для } i > 1, \text{ где } 0 < d < 1.$$

Для стохастического языка $\boldsymbol{L}_d = \left\{\left(a_i, p_i^{'}\right): i = 1,2,\mathbf{K}\right\}$ нетрудно показать, что энтропия $H(\boldsymbol{L}_d)$ конечна и, следовательно, $C_0(\boldsymbol{L}_d)$ также конечна.

Действительно, $H(\boldsymbol{L}_d)$ =

$$= -(1-d)\cdot\log(1-d) + \sum_{i\geq 2} d \cdot \frac{2\cdot\lfloor\log i\rfloor - \log d}{2^{2\lceil\log i\rceil}} =$$

$$= -(1-d)\cdot\log(1-d) + \sum_{i\geq 1} d \cdot \frac{2\cdot\lfloor\log(i+1)\rfloor - \log d}{2^{2\lceil\log(i+1)\rceil}} =$$

$$= -(1-d)\cdot\log(1-d) + d\cdot H(\boldsymbol{L}) - d\cdot\log d = H(d) + 4d < \infty$$

(здесь $\boldsymbol{L}$ - язык из (б) и $H(d) = -d\log d - (1-d)\log(1-d)$). Оценим стоимость оптимального кодирования:

$$C_0(\boldsymbol{L}_d) = \sum_{i\geq 1} p_i^{'}\cdot\lfloor\log(i+1)\rfloor = (1-d) + \sum_{i\geq 2} d\cdot\frac{\lfloor\log(i+1)\rfloor}{2^{2\lceil\log i\rceil}} =$$

$$= (1-d) + \sum_{i\geq 1} d\cdot\frac{\lfloor\log(i+2)\rfloor}{2^{2\lceil\log(i+1)\rceil}} \geq (1-d) + \sum_{i\geq 1} d\cdot\frac{\lfloor\log(i+1)\rfloor}{2^{2\lceil\log i\rceil}} = (1-d) + 2d.$$

$$C_0(\boldsymbol{L}_d) - H(\boldsymbol{L}_d) \geq 1 + d - H(d) - 4d = 1 - 3d - H(d) \to 1 \text{ при } d \to 0,$$

и, следовательно, существует d^*, удовлетворяющее условию:

$$1 - 3d^* - H(d^*) > 1 - e.$$

В качестве $\boldsymbol{L}_e$ возьмем $\boldsymbol{L}_{d^*}$. Для $\boldsymbol{L}_e$ выполняется неравенство

$$C_0(\boldsymbol{L}_e) > H(\boldsymbol{L}_e) + 1 - e.$$

Теорема доказана.

Теорема 3.2. *Пусть* $\boldsymbol{L}=(L,P)$ *– произвольный бесконечный стохастический язык, для которого* $C_0(\boldsymbol{L})$ *имеет конечное значение. Тогда* $H(\boldsymbol{L})$ *также имеет конечное значение.*

Доказательство. Пусть слова из $\boldsymbol{L}$ упорядочены в порядке невозрастания их вероятностей, т.е. $L = \{a_1, a_2, \mathbf{K}, a_i, \mathbf{K}\}$ и $p(a_i) \geq p(a_{i+1})$ для любого i.

Рассмотрим частичные суммы

$$S_n = \sum_{i=1}^{n} p_i \cdot (-\log p_i).$$

Покажем, что существует константа C, для которой $S_n \leq C$ для любого n.

Рассмотрим разность $S_n - 2C_0(\boldsymbol{L})$:

$$S_n - 2C_0(\boldsymbol{L}) = \sum_{i=1}^{n} p_i\cdot(-\log p_i) - 2\cdot\sum_{i=1}^{\infty} p_i\cdot\lfloor\log(i+1)\rfloor =$$

$$= \sum_{i=1}^{n} p_i \cdot \left(\log \frac{1}{p_i} - 2\lfloor \log(i+1) \rfloor \right) < \sum_{i=1}^{n} p_i \cdot \left(\log \frac{1}{p_i} - 2(\log i - 1) \right) =$$

$$= \sum_{i=1}^{n} p_i \cdot \left(\log \frac{1}{p_i \cdot i^2} \right) + 2\sum_{i=1}^{n} p_i .$$

Вторая сумма не превосходит 2. Для оценки первой суммы применим неравенство $\log x \le (x-1) \cdot \log e$. Получим, что

$$\sum_{i=1}^{n} p_i \cdot \left(\log \frac{1}{p_i \cdot i^2} \right) \le \log e \sum_{i=1}^{n} p_i \cdot \left(\frac{1}{p_i \cdot i^2} - 1 \right) =$$

$$= \log e \left(\sum_{i=1}^{n} \frac{1}{i^2} - \sum_{i=1}^{n} p_i \right) < \log e \cdot \sum_{i=1}^{n} \frac{1}{i^2}.$$

При $n \to \infty$ сумма $\sum_{i=1}^{n} \frac{1}{i^2}$ сходится к конечному пределу, обозначим его через S^1. Окончательно получаем, что

$$S_n - 2C_0(\boldsymbol{L}) < \log e \cdot S^1 + 2$$

для любого n. Отсюда следует, что

$$H(\boldsymbol{L}) = -\sum_{i=1}^{\infty} p_i \cdot \log p_i$$

имеет конечное значение. Теорема доказана.

Следствие. *Пусть* $\boldsymbol{L}$ *- произвольный стохастический язык. Тогда стоимость оптимального кодирования* $C_0(\boldsymbol{L})$ *имеет конечное значение тогда и только тогда, когда конечное значение имеет энтропия* $H(\boldsymbol{L})$.

В доказательствах теорем этой главы мы в действительности использовали свойства различных вероятностных распределений. Мы не накладывали ограничений на распределение вероятностей, которое может зависеть от синтаксиса языка.

Таким образом, не всегда энтропия является нижней оценкой стоимости кодирования. Стоимость кодирования может быть меньше энтропии. Все определяется свойствами вероятностного распределения на множестве слов языка сообщений. Эргодические источники с конечным числом состояний, рассматривавшиеся К. Шенноном, определяют не любые распределения вероятностей на словах длины N при $N \to \infty$, а распределения с определенными свойствами. Именно благодаря этим свойствам нижней границей стоимости кодирования является энтропия.

ГЛАВА 4. ВОПРОСЫ КОДИРОВАНИЯ КОНТЕКСТНО-СВОБОДНЫХ ЯЗЫКОВ

Не любые свойства сообщений могут быть описаны с помощью источников с конечным числом состояний.

Пример 4.1. Язык арифметических выражений содержит слова вида

$$((a+b)*a+b(a*(b+c)))+c\,.$$

В арифметических выражениях уровень вложенности скобок может быть любым и не может быть описан средствами источников с конечным числом состояний.

Для описания таких свойств используются контекстно-свободные грамматики. Они определяют класс контекстно-свободных языков, который является ближайшим расширением класса языков, описываемых конечными источниками. КС-языки являются хорошей моделью для описания естественных языков и языков программирования и поэтому представляют практический интерес.

4.1. Основные определения и понятия, связанные с КС-языками и стохастическими КС-языками

Определение 4.1. Стохастической КС-грамматикой называется система $G=\langle V_T, V_N, R, s\rangle$, где V_T, V_N - конечные множества терминальных и нетерминальных символов (терминалов и нетерминалов) соответственно; $s\in V_N$ - аксиома, R - множество правил. Множество R можно представить в виде $R=\cup_{i=1}^{k} R_i$, где k - мощность алфавита V_N и $R_i=\{r_{i1},\mathbf{K},r_{in_i}\}$.

Каждое правило r_{ij} из R_i имеет вид

$$r_{ij}: A_i \xrightarrow{p_{ij}} b_{ij},\ j=1,\mathbf{K},n_i,$$

где $A_i\in V_N$, $b_{ij}\in (V_T\cup V_N)^*$ и p_{ij} - вероятность применения правила r_{ij} (вероятность правила r_{ij}), которая удовлетворяет следующим условиям:

$$0<p_{ij}\le 1 \text{ и } \sum_{j=1}^{n_i} p_{ij}=1.$$

Обычная КС-грамматика отличается от стохастической КС-грамматики отсутствием вероятностей применения правил.

Для слов a и b из $(V_T \cup V_N)^*$ будем говорить, что b непосредственно выводимо из a (и записывать $a \Rightarrow b$), если $a = a_1 A_i a_2$, $b = a_1 b_{ij} a_2$ для некоторых $a_1, a_2 \in (V_T \cup V_N)^*$, и в грамматике G имеется правило $A_i \xrightarrow{p_{ij}} b_{ij}$.

Обозначим через $\Rightarrow^*$ рефлексивное транзитивное замыкание отношения $\Rightarrow$. Через L_G будем обозначать множество слов $\{a : s \Rightarrow^* a, a \in V_T^*\}$.

Пусть $s \Rightarrow^* a$. Левым выводом слова a назовем вывод, при котором каждое правило в процессе вывода слова a из аксиомы s применяется к самому левому нетерминалу в слове. Последовательность правил в левом выводе будем обозначать через $w(a)$. Важное значение имеет понятие дерева вывода. Дерево строится следующим образом.

Корень дерева помечается аксиомой s. Пусть при выводе слова a на очередном шаге в процессе левого вывода применяется правило $A \xrightarrow{p_{ij}} b_{i_1} b_{i_2} \mathbf{K} b_{i_m}$, где $b_{i_l} \in V_T \cup V_N (l = 1, \mathbf{K}, m)$. Тогда из самой левой вершины-листа дерева, помеченной символом A (при обходе листьев дерева слева направо), проводится m дуг в вершины следующего яруса, которые помечаются слева направо символами $b_{i_1}, b_{i_2}, \mathbf{K}, b_{i_m}$ соответственно. После построения дуг и вершин для всех правил грамматики в выводе слова языка все листья дерева помечены терминальными символами и само слово получается при обходе листьев дерева слева направо.

Ярусы дерева будем нумеровать следующим образом. Корень дерева расположен в нулевом ярусе. Вершины дерева, смежные с корнем, образуют первый ярус, и т.д. Дуги, выходящие из вершин j-го яруса, ведут к вершинам $(j+1)$-го яруса.

Высотой дерева будем называть максимальную длину пути от корня к листу, или номер последнего яруса.

Пример 4.2. Рассмотрим грамматику $G_0 = \langle \{x, \bar{x}\}, \{N\}, R, N \rangle$, в которой множество R состоит из двух правил:

$r_1 : N \xrightarrow{p} xN\bar{x}N$,

$r_2 : N \xrightarrow{1-p} l$ (l – пустое слово).

Грамматика G_0 порождает хорошо известный язык Дика. Если символ x интерпретировать как открывающую скобку "(", а символ $\bar{x}$ - как закрывающую скобку ")", то язык Дика - это множество "правильных" последовательностей скобок, обладающих следующими свойствами:

а) для любой начальной подпоследовательности число вхождений "(" не меньше числа вхождений ")";

б) для всей последовательности число вхождений "(" равно числу вхождений ")".

Для слова $a = xx\bar{x}x\bar{x}\bar{x}x\bar{x} \in L_{G_0}$ левый вывод имеет вид $r_1r_1r_2r_1r_2r_2r_1r_2r_2$. Соответствующее ему дерево вывода изображено на рис.4.1. Высота дерева вывода равна 4.

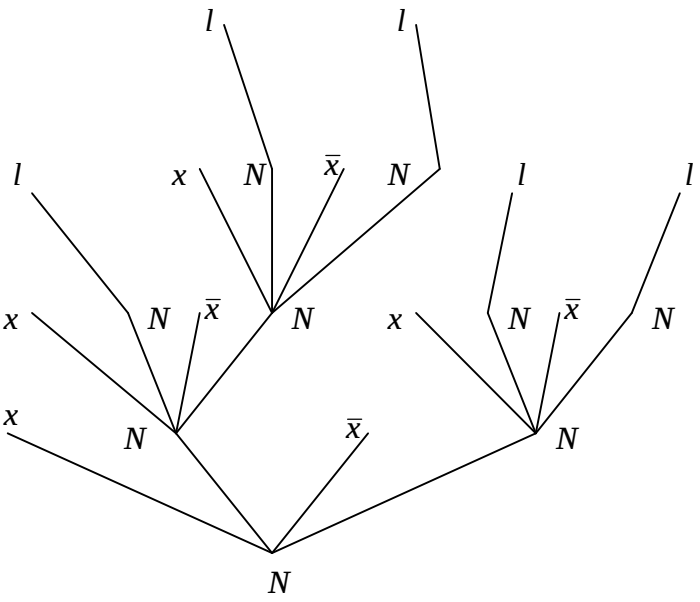


Рис. 4.1. Дерево вывода

Пусть $w(a) = r_{i_1 j_1} r_{i_2 j_2} \mathbf{K}\, r_{i_n j_n}$ - некоторый левый вывод слова $a \in L$ и d_a - соответствующее ему дерево вывода. Определим $p(d_a)$ как произведение вероятностей правил, образующих $w(a)$: $p(d_a) = p_{i_1 j_1} p_{i_2 j_2} \mathbf{K}\, p_{i_n j_n}$.

Вероятность слова a определим как $p(a) = \sum p(d_a)$, где суммирование ведется по всем различным деревьям вывода слова a .

Грамматика G называется согласованной, если

$$\lim_{N \to \infty} \sum_{a \in L_G, |a| \le N} p(a) = 1.$$

В дальнейшем будем рассматривать согласованные КС-грамматики. Согласованная КС-грамматика G индуцирует распределение вероятностей P_G на множестве слов L_G.

Стохастический КС-язык, порожденный согласованной стохастической КС-грамматикой G, есть $\boldsymbol{L}_G = (L_G, P_G)$.

Стохастический язык L называется стохастическим КС-языком, если существует стохастическая КС-грамматика, такая, что $\boldsymbol{L} = \boldsymbol{L}_G$.

Для задачи кодирования важное значение имеет матрица первых моментов, которая определяется следующим образом. Рассмотрим многомерные производящие функции

$$F_i(s_1, s_2, \mathbf{K}\, s_k),\ i=1, \ldots, k,$$

где переменная s_i соответствует нетерминальному символу A_i. Функция $F_i(s_1, s_2, \mathbf{K}\, s_k)$ строится по множеству правил R_i с одинаковой левой частью A_i. Для каждого правила $A_i \xrightarrow{p_{ij}} b_{ij}$ выписывается слагаемое

$$q_{ij} = p_{ij} \cdot s_1^{l_1} \cdot s_2^{l_2} \cdot \mathbf{K} \cdot s_k^{l_k},$$

где l_m - число вхождений нетерминального символа A_m в правую часть правила (m=1, … ,k). Тогда

$$F_i(s_1, s_2, \mathbf{K}, s_k) = \sum_{j=1}^{n_i} q_{ij}.$$

Пусть

$$a_j^i = \frac{\partial F_i(s_1, s_2, \mathbf{K}, s_k)}{\partial s_j}\Big|_{s_1 = s_2 = \mathbf{K}\, s_k = 1.}$$

Квадратная матрица A порядка k, образованная элементами a_j^i, называется матрицей первых моментов грамматики G.

Так как матрица A неотрицательна, то существует максимальный по модулю действительный неотрицательный собственный корень (перронов корень). Обозначим этот корень через r.

Известно необходимое и достаточное условие согласованности стохастической КС-грамматики: стохастическая КС-грамматика при отсутствии бесполезных нетерминалов (т.е. не участвующих в порождении слов языка) является согласованной тогда и только тогда, когда перронов корень матрицы первых моментов не превосходит единицы. Поэтому, рассматривая КС-языки, мы будем предполагать, что $r \le 1$.

Вторые моменты грамматики G определяются как

$$b_{jm}^i = \frac{\partial^2 F_i(s_1, s_2, \mathbf{K}, s_k)}{\partial s_j \partial s_k}\Big|_{s_1 = s_2 = \mathbf{K}\, s_k = 1.}$$

Пусть $A_i \in V_N$. Через $I_1(A_i)$ обозначим множество нетерминальных символов, таких, что для любого $A_j \in I_1(A_i)$ существует слово $a = a_1 A_j a_2 \in (V_N \,\mathbf{U}\, V_T)^*$, для которого $A_i \Rightarrow_* a$. Через $I_2(A_i)$ обозначим множество нетерминальных символов, таких, что для любого $A_j \in I_2(A_i)$ существует $a = a_1 A_i a_2 \in (V_N \,\mathbf{U}\, V_T)^*$, для которого $A_j \Rightarrow_* a$. Таким

образом, $I_1(A_i)$ - это множество нетерминалов, которые встречаются при выводе слов из A_i как аксиомы, а $I_2(A_i)$ - множество нетерминалов, при выводе слов из которых встречается символ A_i. Через $I_0(A_i)$ обозначим пересечение этих множеств, т.е. $I_0(A_i) = I_1(A_i) \cap I_2(A_i)$. Множество нетерминалов $K = A_{i_1}, \mathbf{K}, A_{i_q}$ для которых $I_0(A_{i_j})$ совпадают и $I_0(A_{i_j}) \neq \varnothing$, j=1,…,q, назовем классом. Если $I_0(A_i) = \varnothing$, будем считать, что A_i образует класс $\{A_i\}$.

Для грамматики G множество нетерминалов V_N распадается на непересекающиеся классы. Грамматику G назовем неразложимой, если все нетерминалы из V_N образуют один класс. В противном случае G называется разложимой.

Свойство неразложимости грамматики можно проиллюстрировать на языке теории графов. Пусть k - мощность нетерминального алфавита. Каждому нетерминальному символу A_i поставим в соответствие вершину графа v_i. Вершины v_i и v_j соединим дугой (v_i, v_j), если $A_j \in I_1(A_i)$. Обозначим полученный граф через $\Gamma(G)$.

Для неразложимой грамматики построенный граф $\Gamma(G)$ является сильно связным, т.е. графом, в котором каждая пара вершин связана ориентированным путем.

Неразложимой грамматике соответствует неразложимая матрица первых моментов A. Напомним, что матрица A называется разложимой, если перестановкой рядов она может быть приведена к виду

$$A = \begin{pmatrix} B & 0 \\ C & D \end{pmatrix},$$

где B и D - квадратные матрицы (перестановка рядов - это перестановка строк с такой же перестановкой столбцов). В противном случае матрица A называется неразложимой.

Неразложимая матрица называется периодической с периодом c, если НОД для всех тех t, для которых $a_i^i(t) > 0$, равен c (здесь через $a_j^i(t)$ обозначены элементы матрицы A^t. Если c = 1, матрица называется непериодической.

Неразложимость и непериодичность матрицы A означают существование такого $n > 0$, для которого $A^n > 0$.

Проиллюстрируем понятие неразложимой грамматики на следующем примере.

Пример 4.3. Пусть $G = \langle V_T, V_N, R, s \rangle$ - грамматика, в которой $V_T = \{+, *, (,), x\}$, $V_N = \{E, T, I\}$, аксиомой является нетерминал E и $R = R_1 \cup R_2 \cup R_3$, где

$R_1 = \{r_{11}, r_{12}\}$,

$$r_{11} : E \xrightarrow{p} T,$$
$$r_{12} : E \xrightarrow{1-p} E + T,$$

$R_2 = \{r_{21}, r_{22}\}$,

$$r_{21} : T \xrightarrow{q} I,$$
$$r_{22} : T \xrightarrow{1-q} I * T,$$

и $R_3 = \{r_{31}, r_{32}\}$,

$$r_{31} : I \xrightarrow{g} (E),$$
$$r_{32} : I \xrightarrow{1-g} x.$$

Грамматика G порождает язык правильных арифметических выражений. Примером слова языка, порождаемого G, является $a = (x + x) * x$.

Построим производящие функции по грамматике G:

$$F_1(s) = ps_2 + (1-p)s_1 s_2,$$
$$F_2(s) = qs_3 + (1-q)s_2 s_3,$$
$$F_3(s) = gs_1 + (1-g).$$

Используя производящие функции, найдем матрицу первых моментов:

$$A = \begin{pmatrix} 1-p & 1 & 0 \\ 0 & 1-q & 1 \\ g & 0 & 0 \end{pmatrix}.$$

Матрица A является неразложимой. Нетрудно убедиться, что $A^3 > 0$.

4.2. Связь стоимости оптимального кодирования стохастического КС-языка с энтропией и матрицей первых моментов

Рассмотрим класс КС-языков с однозначным выводом (каждое слово такого языка имеет единственное дерево вывода). В этом случае вероятность слова определяется как произведение вероятностей правил грамматики, образующих левый вывод слова. Поэтому значение вероятности слова определяется выводом слова.

Следующая теорема устанавливает соотношение между стоимостью оптимального кодирования и энтропией КС-языка с однозначным выводом.

Теорема 4.1.

а) Для любого s, $\frac{1}{2} \le s < 1$, *существует стохастический КС-язык* $\boldsymbol{L}_s$ *с однозначным выводом, для которого выполняется равенство*

$$C_0\ (\boldsymbol{L}_s) = s \cdot H\ (\boldsymbol{L}_s).$$

б) Для любоо t, $0 \le t < 1$, *существует стохастический КС-язык* L_t *с однозначным выводом, для которого выполняется равенство*

$$C_0(\boldsymbol{L}_t)\ = H\ (\boldsymbol{L}_t) + t.$$

Опускаем доказательство этой и следующей теорем ввиду их сложности.

Пусть $\boldsymbol{L}$ - стохастический КС-язык, порожденный грамматикой G с однозначным выводом, и A_i - некоторый нетерминальный символ. Через $\boldsymbol{L}_i$ обозначим язык, порожденный грамматикой G_i, которая получается из исходной грамматики G заменой аксиомы на A_i. Таким образом, $\boldsymbol{L}_i$ есть множество слов в терминальном алфавите, выводимых из символа A_i.

Положим $L = L_1$ для исходного языка L. Через $H(R_i)$ будем обозначать энтропию множества правил R_i:

$$H(R_i) = -\sum_{j=1}^{n_i} p_{ij} \cdot \log p_{ij},$$

и через $H(\boldsymbol{L}_i)$ - энтропию языка $\boldsymbol{L}_i$.

Теорема 4.2. *Пусть перронов корень* r *матрицы первых моментов порождающей грамматики* G *с однозначным выводом меньше 1. Тогда величины* $H\ (\boldsymbol{L}_i)$ *конечны и удовлетворяют следующей системе линейных уравнений:*

$$H(\boldsymbol{L}_i) = H(R_i) + \sum_{j=1}^{k} a_j^i \cdot H\ (\boldsymbol{L}_j)\ \ ,(i = 1, \mathbf{K}, k)$$

(здесь a_j^i *- элементы матрицы первых моментов).*

Теорема 4.3. *Пусть перронов корень* r *матрицы* A *равен 1. Тогда* $H(\boldsymbol{L})$ *не ограничена.*

Следствие. *Пусть* $\boldsymbol{L}$ *-стохастический КС-язык, порожденный грамматикой* G *с однозначным выводом. Тогда энтропия* $H(\boldsymbol{L})$ *и стоимость оптимального кодирования* $C_0(\boldsymbol{L})$ *имеют конечное значение тогда и только тогда, когда перронов корень матрицы первых моментов грамматики* G *строго меньше* 1.

Из теоремы 4.3 следует эффективный путь для вычисления энтропии стохастического КС-языка с однозначным выводом при $r < 1$.

Пример 4.4. Пусть $G = \langle \{a,b,c,d\}, \{A_1, A_2\}, R, A_1 \rangle$ и $R = R_1 \cup R_2$, где R_1 содержит два правила:

$$r_{11} : A_1 \xrightarrow{1/2} aA_1b,$$

$$r_{12} : A_1 \xrightarrow{1/2} cA_2d;$$

и R_2 содержит два правила:

$$r_{21} : A_2 \xrightarrow{1/2} cA_2d,$$

$$r_{22} : A_2 \xrightarrow{1/2} cd.$$

Грамматика G с однозначным выводом порождает язык

$$L_1 = \{a^i c^j d^j b^i : i \geq 0, j \geq 1\}.$$

Для этой грамматики

$$A = \begin{pmatrix} 1/2 & 1/2 \\ 0 & 1/2 \end{pmatrix} \text{ и } r = \frac{1}{2} < 1,$$

где A - матрица первых моментов и r - ее перронов корень.

Запишем систему уравнений:

$$H(\boldsymbol{L}_1) = 1 + \frac{1}{2} H(\boldsymbol{L}_1) + \frac{1}{2} H(\boldsymbol{L}_2)$$

$$H(\boldsymbol{L}_2) = 1 + \frac{1}{2} H(\boldsymbol{L}_2).$$

Система имеет единственное решение: $H(\boldsymbol{L}_1) = 4$, $H(\boldsymbol{L}_2) = 2$.

Используя теорему, можно оценить стоимость оптимального кодирования языка L_1:

$$2 \leq C_0(\boldsymbol{L}_1) < 5.$$

4.3. Метод укрупнения правил КС-грамматики

Опишем метод перехода от исходной грамматики G с однозначным выводом к грамматике $G(n)$, используемый в дальнейшем в алгоритмах асимптотически оптимального кодирования стохастических КС-языков.

Пусть $A_i \Rightarrow^* a$. Через $d(a)$ обозначим дерево вывода слова a и через $|a|$ - высоту дерева вывода a. Через L_i^n обозначим множество слов языка L_i, имеющих деревья вывода высоты n, и через M_i^n - множество слов в алфавите $V_N \cup V_T$, выводимых из A_i, для которых высота дерева вывода не превосходит n, и нетерминалами могут быть помечены листья только n-го яруса дерева.

Покажем, что

$$\sum_{a \in M_i^n} p(a) = 1.$$

Очевидно,

$$\sum_{a \in M_i^n} p(a) = \sum_{a \in L_i^{n-1}} p(a) + \sum_{a \in M_i^n \setminus L_i^{n-1}} p(a).$$

Первую и вторую сумму в правой части обозначим через S_1 и S_2 соответственно. Для слагаемых, входящих в S_2, представим $p(a)$ в виде произведения $p(a) = p(b) \cdot p(a_1) \cdot \mathbf{K} \cdot p(a_l)$, где b - слово, которому соответствует дерево вывода $d(b)$, получающееся из $d(a)$ удалением вершин n-го яруса и инцидентных им дуг, а $a_1, \mathbf{K}, a_l$ - слова, соответствующие поддеревьям высоты 1 с корнями на n-1-м ярусе дерева $d(a)$. Будем считать, что $a_1, \mathbf{K}, a_l$ упорядочены слева направо в соответствии с расположением корней их деревьев в $d(a)$. Тогда

$$S_2 = \sum_{a \in M_i^n \setminus L_i^{n-1}} p(a) = \sum_{b, a_1, \mathbf{K}, a_l} p(b) \cdot p(a_1) \mathbf{K}\, p(a_l).$$

Сгруппируем члены суммы, имеющие одинаковые слова $b, a_1, \mathbf{K}, a_{l-1}$, тогда

$$S_2 = \sum_{b, a_1, \mathbf{K}, a_{l-1}} p(b) \cdot p(a_1) \mathbf{K}\, p(a_{l-1}) \cdot \sum_{a_l} p(a_l).$$

Учитывая, что $\sum_{a_l} p(a_l) = 1$, так как слова a_l являются правыми частями правил некоторого множества R_j, получим:

$$S_2 = \sum_{b, a_1, \mathbf{K}, a_{l-1}} p(b) \cdot p(a_1) \mathbf{K}\, p(a_{l-1}) = \mathbf{K} = \sum_{b \in M_i^{n-1} \setminus L_i^{n-1}} p(b).$$

В итоге

$$S_1 + S_2 = \sum_{L_i^{n-1}} p(a) + \sum_{b \in M_i^{n-1} \setminus L_i^{n-1}} p(b) = \sum_{M_i^{n-1}} p(a).$$

Продолжая этот процесс, получим, что

$$\sum_{M_i^n} p(a) = \sum_{M_i^{n-1}} p(a) = \mathbf{K} = \sum_{M_i^1} p(a).$$

Множество M_i^1 есть не что иное, как множество правых частей правил из R_i. Поэтому

$$\sum_{M_i^1} p(a) = \sum_{j=1}^{n_i} p_{ij} = 1.$$

Используя множества M_i^1, мы можем перейти от грамматики G к грамматике $G(n)$ с множеством правил $R(n) = \cup_{i=1}^{k} R_i(n)$ и

$$R(n) = \left\{ A_i \xrightarrow{p'_{ij}} a'_{ij} : a'_{ij} \in M_i^n \right\}$$

где $j = 1, \mathbf{K}, n_i'$, и число правил n_i' в $R_i(n)$ равно числу слов в M_i^n.

Каждому правилу в $R_i(n)$ приписывается вероятность, равная вероятности вывода слова a'_{ij} в исходной грамматике G. В силу доказанного $G(n)$ - стохастическая КС-грамматика. Нетрудно заметить, что $L(G)=L(G(n))$, и если G - грамматика с однозначным выводом, то и $G(n)$ - также грамматика с однозначным выводом.

Для грамматики G с неоднозначным выводом в случае, когда некоторое слово $a \in M_i^n$ имеет несколько различных деревьев вывода высоты, не превосходящей n, достаточно в качестве вероятности соответствующего правила взять сумму вероятностей этих деревьев.

Для $G(n)$ матрица первых моментов совпадает с n-ой степенью матрицы A для G, то есть равна A^n.

Описанный способ укрупнения правил позволяет перейти к эквивалентной грамматике с непериодической матрицей первых моментов.

Периодичность с периодом $c > 1$ для матрицы первых моментов означает, что множество нетерминальных символов V_N разбивается на непересекающиеся подмножества $V_0, V_1, \mathbf{K}, V_{c-1}$. В любом дереве вывода множество нетерминалов, приписанных вершинам яруса, принадлежит некоторому подмножеству разбиения. При переходе от одного яруса к другому множества нетерминалов, соответствующие ярусам, чередуются:

нетерминалы нулевого яруса принадлежат V_0, первого яруса - V_1, и т.д., нетерминалы (c-1)-го яруса принадлежат V_{c-1}, c-го яруса - V_0 и т.д.

Взяв в качестве параметра укрупнения n значение c, мы можем перейти от исходной грамматики G к эквивалентной непериодической грамматике $G(c)$, множеством нетерминалов которой является V_0. Поэтому требование непериодичности матрицы первых моментов является непринципиальным.

Так как неразложимость и непериодичность матрицы A означают, что $A^t > 0$ для некоторого $t > 0$, то, применяя описанный метод укрупнения правил, можно перейти к грамматике с неразложимой матрицей первых моментов, для которой $A > 0$. Поэтому, не уменьшая общности, в дальнейшем, в тех случаях, когда речь идет о неразложимой грамматике, будем полагать, что матрица первых моментов строго положительна.

4.4. Неразрешимость проблем, связанных с алфавитным кодированием КС-языков

В классе КС-языков проблема распознавания взаимной однозначности и задача оптимального алфавитного кодирования алгоритмически неразрешимы. Приведем доказательство этих утверждений.

Теорема 4.4. *Пусть B – алфавит и $|B| \geq 4$. Не существует алгоритма, позволяющего по произвольному КС-языку в алфавите B определить, допускает ли он алфавитную редукцию.*

Доказательство. Сведем к рассматриваемой проблеме комбинаторную проблему Поста.

Пусть $B = \{a,b,c,d\}$ и $x = (x_1, \mathbf{K}, x_n)$, $y = (y_1, \mathbf{K}, y_n)$ - системы непустых слов в алфавите $\{a,b\}$. Примем следующие обозначения:

$$L(x) = \left\{ba^{i_k}\ \mathbf{K}\, ba^{i_1} c x_{i_1}\ \mathbf{K}\, x_{i_k} : k \geq 1, 1 \leq i_j \leq n\right\};$$

$$L(y) = \left\{dba^{i_k}\ \mathbf{K}\, ba^{i_1} c y_{i_1}\ \mathbf{K}\, y_{i_k} : k \geq 1, 1 \leq i_j \leq n\right\};$$

$$L_0 = \{aa, ab, ac, ad, bc, ca, cb\};$$

$$L(x,y) = L(x) \cup L(y) \cup L_0.$$

Если рассматривать слово ba^i как код числа i, то в каждом слове из $L(x)$ слева от символа c последовательно располагаются коды индексов $i_k, i_{k-1}, \mathbf{K}, i_1$, а справа от символа c – слова $x_{i_1}, x_{i_2}, \mathbf{K}, x_{i_k}$.

Для каждой системы x множество $L(x)$ является КС-языком, так как оно порождается КС-грамматикой $G_x = \langle B, \{s\}, P_x, s\rangle$ со следующим множеством правил P_x:

$$\left\{s \to ba^i s x_i, s \to ba^i c x_i : i = 1, \mathbf{K}, n\right\}.$$

Язык $L(y)$ порождается КС-грамматикой $G_y = \langle B, \{s, y\}, P_y, s\rangle$ с множеством правил P_y:

$$\left\{s \to dba^i y\, x_i, s \to ba^i y\, x_i, y\ \to c : i = 1, \mathbf{K}, n\right\}.$$

$L(x,y)$ - КС-язык ввиду замкнутости класса КС-языков относительно операции объединения.

Рассмотрим вопрос, допускает ли $L(x,y)$ алфавитную редукцию. Слова конечного языка L_0 подобраны таким образом, что любая пара букв из B не является контекстно-различимой, и буквы a, b, c – существенные. $L(x,y)$ допускает алфавитную редукцию тогда и только тогда, когда буква d – фиктивная, что верно лишь в том случае, когда для

любой пары слов $a \in L(x)$ и $b \in L(y)$, имеющих одинаковую последовательность кодов $ba^{i_k}\,\mathbf{K}\,ba^{i_1}$, выполняется условие $x_{i_1}\,\mathbf{K}\,x_{i_k} \neq y_{i_1}\,\mathbf{K}\,y_{i_k}$. Таким образом, $L(x,y)$ допускает алфавитную редукцию тогда и только тогда, когда для систем x и y не существует решения комбинаторной проблемы Поста. Из неразрешимости комбинаторной проблемы Поста следует неразрешимость проблемы распознавания алфавитной редукции для КС-языков. Теорема доказана.

Замечание. Из доказательства теоремы следует, что для КС-языков алгоритмически неразрешима проблема выявления фиктивных букв. Нетрудно показать, что для КС-языков алгоритмически неразрешима и проблема выявления контекстно-различимых букв. Для этого достаточно рассмотреть вопрос о контекстной различимости букв d и c в языках

$$L'(x,y) = L'(x) \cup L'(y),$$

$$\text{где } L'(x) = \left\{cba^{i_k}\,\mathbf{K}\,ba^{i_1}cx_{i_1}\,\mathbf{K}\,x_{i_k} : k \geq 1, 1 \leq i_j \leq n\right\},$$

$$L'(y) = \left\{dba^{i_k}\,\mathbf{K}\,ba^{i_1}cy_{i_1}\,\mathbf{K}\,y_{i_k} : k \geq 1, 1 \leq i_j \leq n\right\}.$$

Теорема 4.5. *Пусть B – алфавит и $|B| \geq 4$. Проблема распознавания взаимной однозначности алфавитного кодирования для КС-языков в алфавите B алгоритмически неразрешима.*

Доказательство. Пусть $L \subseteq B^*$ - произвольный КС-язык в алфавите $B = \{b_1, \mathbf{K}, b_m\} (m \geq 4)$. Построим схему f оптимального префиксного кодирования для множества букв $B' = \{b_1, \mathbf{K}, b_{m-1}\}$:

$$f : \begin{cases} b_1 \to v_1 \\ \mathbf{K} \\ b_{m-1} \to v_{m-1}. \end{cases}$$

Рассмотрим следующую схему *f'* алфавитного кодирования:

$$f : \begin{cases} b_1 \to v_1 \\ \mathbf{K} \\ b_{m-1} \to v_{m-1} \\ b_m \to l \end{cases}$$

(здесь l - пустое слово).

Из неразрешимости проблемы распознавания фиктивных букв для КС-языков при $(m \geq 4)$ следует утверждение теоремы.

Из доказательства теоремы вытекает

Следствие. *Пусть* B *– алфавит и* $|B| \geq 4$. *Проблема оптимального алфавитного кодирования для КС-языков в алфавите* B *алгоритмически неразрешима.*

Ввиду того, что основные алгоритмические проблемы кодирования для КС-языков алгоритмически неразрешимы, а решать задачи кодирования надо, выход следует искать по следующим направлениям:

1) Применять более сложные, чем алфавитное, алгоритмы кодирования;
2) Строить алгоритмы асимптотически оптимального кодирования.

4.5. Кодирование стохастических КС-языков в докритическом случае

В этом разделе рассматриваются вопросы экономного кодирования сообщений, являющихся словами стохастического КС-языка. При этом рассматривается случай, когда перронов корень матрицы первых моментов строго меньше 1 (докритический случай). Устанавливаются асимптотические свойства деревьев вывода, и приводится полученная на их основе неулучшаемая нижняя оценка стоимости любого кодирования. Описывается алгоритм асимптотически оптимального кодирования, в основе которого лежит блочное кодирование деревьев вывода.

Приведенные в настоящем разделе результаты являются обобщением результатов К.Шеннона на класс рассматриваемых КС-языков.

4.5.1. Закономерности применения правил грамматики в докритическом случае

Пусть L - язык, порожденный стохастической КС-грамматикой G. Будем полагать, что аксиомой исходной грамматики G является нетерминал A_1. Пусть D_1 - множество деревьев вывода для слов языка L. Рассмотрим D_1^t - множество деревьев из D_1 высоты t.

Через $M_i(t,t)$ обозначим условное математическое ожидание числа вершин на ярусе t, помеченных нетерминалом A_i, в деревьях вывода высоты t.

Теорема 4.6. *Пусть* G *- стохастическая КС-грамматика с неразложимой непериодической матрицей первых моментов, для которой перронов корень меньше единицы, и* D_1^t *- множество деревьев вывода высоты* t.

Тогда для любого $i \in \{1, \mathbf{K}, k\}$ *при* $t \to \infty$ *и* $t - t \to \infty$ *выполняется асимптотическое равенство*

$$M_i(t,t) \sim u_i v_i + B_i,$$

в котором u_i *и* v_i *- i-е компоненты правого и левого собственных векторов для перронова корня* r *соответственно, и* B_i *- константы, определяемые формулой*

$$B_i = \frac{1}{r} \sum_{n,m,s} v_n u_s b_{ms}^l \sum_{t=1}^{\infty} a_i^m (t-1).$$

Здесь b_{ms}^l *- вторые моменты, и* $a_i^m(t-1)$ *- элементы матрицы* A^{t-1}.

Пусть r_{ij} - произвольное правило грамматики G. Через $s_l^{(ij)}$ обозначим число нетерминалов A_l в правой части правила r_{ij}. Условное математическое ожидание числа применений правила r_{ij} в деревьях вывода высоты t на ярусе t будем обозначать через $M_{ij}(t,t)$.

Теорема 4.7. *Пусть* D_1^t *- множество деревьев вывода высоты* t *для слов языка, порождаемого стохастической КС-грамматикой с неразложимой и непериодической матрицей первых моментов, для которой перронов корень меньше единицы.*

Тогда при $t \to \infty$ *и* $t - t \to \infty$ *выполняется следующее асимптотическое равенство:*

$$M_{ij}(t,t) \sim p_{ij}\left(\frac{v_i \sum_{l=1}^{k} u_l s_l^{(ij)}}{r} + B_i \right).$$

В формулировке теоремы p_{ij} - вероятность правила r_{ij}, задаваемая в исходной грамматике, а величины v_i, u_l и B_i имеют тот же смысл, что и ранее.

Таким образом, сделаем несколько выводов.

1. Поскольку все величины в формулировке теоремы являются константами, математическое ожидание числа применений правила r_{ij} на ярусе t стремится к константе при $t \to \infty$, $t - t \to \infty$.

2. Математическое ожидание $M_{ij}(t,t)$ не зависит от аксиомы грамматики.

3. Величина $\sum_{l=1}^{k} u_l s_l^{(ij)}$ определяется правой частью правила r_{ij} и имеет большее значение для тех правил, которые содержат в правой части большее количество нетерминальных символов. Величина B_i имеет одно и то же значение для всех правил грамматики с одинаковой левой частью A_i.

Величину $p_{ij}\left(\frac{v_i \sum_{l=1}^{k} u_l s_l^{(ij)}}{r} + B_i \right)$ в дальнейшем будем обозначать через w_{ij}.

Обозначим через $M(t,t)$ математическое ожидание общего числа нетерминалов на ярусе t. Определим $M(t,t)$. Просуммировав математические ожидания для всех нетерминалов и учитывая нормировку $\sum_{i=1}^{k} u_i v_i = 1$, получаем

Следствие. Пусть $t \to \infty$, $t - t \to \infty$. Тогда

$$M(t,t) \sim 1 + \sum_{i=1}^{k} B_i .$$

Таким образом, пределы величин $M_{ij}(t,t)$, $M_i(t,t)$ и $M(t,t)$ являются константами и не зависят от аксиомы грамматики.

Пусть $S_{ij}(t)$ - число правил r_{ij} в дереве вывода из D_1^t. Рассмотрим величину $\frac{S_{ij}(t)}{t}$ - среднее число правил r_{ij}, приходящееся на один ярус дерева вывода из D_1^t.

Теорема 4.8. *При* $t \to \infty$

$$M\left(\frac{S_{ij}(t)}{t}\right) \to w_{ij} .$$

Теорема 4.9. *Для любого* $e > 0$

$$P\left(\left|\frac{S_{ij}(t)}{t} - w_{ij}(t)\right| > e\right) \to 0 \ \textit{при}\ t \to \infty .$$

Из теоремы следует, что почти все слова КС-языка, имеющие деревья вывода высоты t, при $t \to \infty$ имеют приблизительно одинаковый состав правил в левом выводе и, как следствие, приблизительно одинаковый буквенный состав.

Для иллюстрации результатов рассмотрим простой пример, в котором грамматика содержит один нетерминальный символ.

Пример 4.5. Пусть язык L порождается грамматикой $G = \langle \{x, \bar{x}\}, \{N\}, R, N \rangle$, где множество R содержит два правила:

$r_{11} : N \xrightarrow{p} xN\bar{x}N$,

$r_{12} : N \xrightarrow{1-p} l$ (l - пустое слово).

Для них $s_1^{(11)} = 2$ и $s_1^{(12)} = 0$.

Очевидно, $u_1 = v_1 = 1$ для грамматики с одним нетерминальным символом.

Построим производящую функцию

$$F_1(s) = ps_1^2 + (1-p)s_1^0 = ps_1^2 + (1-p).$$

Определим значения первого и второго моментов:

$$a_1^1 = \frac{\partial F_1(s_1)}{\partial s_1}\Big|_{s_1=1} = 2p,\ \ b_{11}^1 = \frac{\partial^2 F_1(s_1)}{\partial s_1^2}\Big|_{s_1=1} = 2p.$$

Так как матрица первых моментов состоит из одного элемента, его значение является перроновым корнем. Поэтому $r = a_1^1 = 2p.$

Отсюда $r < 1$ при $p < \frac{1}{2}.$ Для нашего примера $B_1 = \frac{1}{1-2p}.$

Найдем пределы математического ожидания среднего числа применений правил r_{11} и r_{12}, приходящегося на один ярус дерева вывода, при $t \to \infty.$

$$w_{11} = p\left(\frac{s_1^{(11)}}{r} + B_1\right) = p\left(\frac{2}{2p} + \frac{1}{1-2p}\right) = 1 + \frac{p}{1-2p},$$

$$w_{12} = (1-p)\left(\frac{s_1^{(12)}}{r} + B_1\right) = (1-p)\left(0 + \frac{1}{1-2p}\right) = 1 + \frac{p}{1-2p}.$$

Для математического ожидания среднего числа правил на одном ярусе дерева вывода имеем:

$$w = w_{11} + w_{12} = 2\left(1 + \frac{p}{1-2p}\right) = 1 + \frac{1}{1-2p}.$$

Отметим, что $w \to 2$ при $p \to 0$ и $w \to \infty$ при $p \to \frac{1}{2}.$

Таким образом, для грамматики *G* в длинных словах порождаемого языка правила r_{11} и r_{12} применяются с одинаковой частотой, не зависящей от начальных вероятностей, заданных для правил грамматики. Так как правило r_{11} порождает при своем применении два нетерминала, очевидно, для того, чтобы поддерживать баланс, необходимо в половине случаев применять правило r_{12}.

4.5.2. Нижняя оценка стоимости кодирования и асимптотически оптимальное кодирование.

Постановка задачи кодирования здесь аналогична той, которую исследовал К. Шеннон для конечного эргодического источника. Задача кодирования рассматривается на множестве слов большой длины. В качестве такого множества берется множество слов стохастического КС-языка, каждому из которых соответствует дерево вывода высоты *t*, при $t \to \infty.$ Мерой эффективности кодирования является его стоимость, или число двоичных разрядов, используемых для кодирования одной буквы слова.

Пусть $\boldsymbol{L}$=(L,P) - стохастический КС-язык с однозначным выводом. Через L^t обозначим множество таких слов из L, что дерево вывода каждого слова имеет высоту t. Для $a \in L^t$ через $p_t(a)$ обозначим условную вероятность появления слова a, т.е. $p_t(a) = \dfrac{p(a)}{P\left(L^t\right)}$. В силу однозначности вывода $P\left(L^t\right) = P\left(D^t\right)$.

Стоимостью кодирования f назовем величину

$$= \lim_{t\to\infty} \frac{\sum_{a\in L^t} p_t(a)\cdot |f(a)|}{\sum_{a\in L^t} p_t(a)\cdot |a|}.$$

(здесь $|x|$ -длина последовательности x).

Величина $C(\boldsymbol{L}, f)$ характеризует число двоичных разрядов, приходящихся на кодирование одного символа слова языка.

Через $f(L)$ обозначим класс всех инъективных отображений из L в $\{0,1\}^+$, для которых существует $C(\boldsymbol{L}, f)$. Стоимостью оптимального кодирования языка $\boldsymbol{L}$ назовем величину

$$C_0(\boldsymbol{L}) = \inf_{f\in F(L)} C(\boldsymbol{L}, f).$$

Теорема 4.10. *Пусть $\boldsymbol{L}$ - язык, порожденный стохастической КС-грамматикой с однозначным выводом, для которой матрица первых моментов неразложима, непериодична и перронов корень строго меньше 1.*

Тогда для любого кодирования $f \in F(L)$ стоимость кодирования $C(L, f)$ удовлетворяет следующему неравенству:

$$C(L, f) \ge \frac{\log r}{h} + \frac{1}{h}\sum_{i=1}^{k} B_i \cdot H\left(p_{i1}, \mathbf{K}, p_{in_i}\right) - \frac{1}{rh}\sum_{i=1}^{k} v_i \sum_{j=1}^{n_i} p_{ij} \log p_{ij} \sum_{l=1}^{k} u_i s_l^{(ij)},$$

где p_{ij} - вероятность правила r_{ij},

$U = (u_1, \mathbf{K}, u_k)$ и $V = (v_1, \mathbf{K}, v_k)$ - соответственно правый и левый положительные собственные векторы для перронова корня r при нормировке $\sum_{i=1}^{k} u_i v_i = 1$,

$s_l^{(ij)}$ - число нетерминалов A_l в правой части правила r_{ij},

B_i - константа, определенная ранее,

h - предел математического ожидания среднего числа терминальных символов на одном ярусе дерева вывода при $t \to \infty$.

Обозначим через $C^*(L)$ нижнюю оценку стоимости кодирования

$$\frac{\log r}{h}+\frac{1}{h}\sum_{i=1}^{k}B_i\cdot H\left(p_{i1},\mathbf{K},p_{in_i}\right)-\frac{1}{rh}\sum_{i=1}^{k}v_i\sum_{j=1}^{n_i}p_{ij}\log p_{ij}\sum_{l=1}^{k}u_i s_l^{(ij)}\ .$$

Теорема 4.11. *Пусть* L *- язык, порожденный стохастической КС-грамматикой* G *с однозначным выводом, для которой матрица первых моментов неразложима, непериодична и перронов корень строго меньше* 1.

Тогда существует последовательность кодирований

$$\{f_n:\ f_n\in F(L), n=1,2,\mathbf{K}\}\ \textit{такая, что}$$

$$C(L,f_n)-C^*(L)\to 0\ \textit{при}\ n\to\infty\ .$$

4.5.3. Алгоритм асимптотически оптимального кодирования

Алгоритм состоит в переходе от исходной грамматики G к грамматике $G(n)$ с "укрупненными" правилами и в применении алгоритма алфавитного кодирования Шеннона (или Хаффмана) к каждому подмножеству $R_i(n)$ правил с одинаковым нетерминалом A_i в левой части правил.

Для кодирования слова a языка достаточно построить левый вывод этого слова в грамматике и затем каждое правило в выводе заменить его элементарным кодом в соответствии с построенной схемой локально-префиксного кодирования.

Проиллюстрируем алгоритм асимптотически оптимального кодирования на примере рассмотренной ранее грамматики.

Пример 4.6. Для грамматики G с двумя правилами

$r_{11}: N\xrightarrow{p} xN\bar{x}N,$

$r_{12}: N\xrightarrow{1-p} l$ (l – пустое слово).

Напомним, что для рассматриваемой грамматики

$$a_1^1=2p\ \text{ и }\ b_{11}^1=2p,$$

и перронов корень $r=a_1^1=2p$. Отсюда $r<1$ при $p<\frac{1}{2}$.

Для математических ожиданий числа применений правил r_{11} и r_{12} установлено, что $w_{11}=w_{12}=1+\frac{p}{1-2p}$.

Для математического ожидания среднего числа правил, приходящегося на один ярус дерева вывода, в пределе при $t\to\infty$ получено следующее значение

$$w_1=1+\frac{1}{1-2p}.$$

Определим значение величины h, учитывая, что число l_1 терминальных символов в правой части правила r_{11} равно двум, и l_2 равно нулю для правила r_{12}:

$$h = l_1 w_{11} + l_2 w_{12} = 2\left(1 + \frac{p}{1-2p}\right) = 1 + \frac{1}{1-2p}.$$

Определим стоимость оптимального кодирования для языка L_G:

$$C_0(\boldsymbol{L}_G) = \frac{\log r}{h} - \frac{1}{h}\left(w_{11}\log p + w_{12}\log(1-p)\right) =$$

$$= \frac{1+\log p}{h} - \frac{1}{2}(\log p + \log(1-p)) = \frac{1-2p}{2(1-p)} + \frac{1}{2(1-p)} \cdot H(p,1-p),$$

где $H(p,1-p) = -(p\log p + (1-p)\log(1-p))$. Отметим, что $C_0(\boldsymbol{L}_G) \to 1$ при $p \to \frac{1}{2}$ и

$$C_0(\boldsymbol{L}_G) \to \frac{1}{2} \text{ при } p \to 0.$$

Пусть $p = 1/8$. Тогда $r = 1/4$ и $C_0(\boldsymbol{L}_G) = 3/7 + 4/7 \cdot H\left(1/8, 7/8\right) \approx 0{,}739$.

Применим алгоритм асимптотически оптимального кодирования для случаев $n = 1$ и $n = 2$. Очевидно, $C(\boldsymbol{L}_G, f_{sh}) = 1$ при $n = 1$, и одно из правил следует кодировать символом 0, а другое - символом 1.

Пусть $n = 2$. Построим правила грамматики $G(2)$, опуская для простоты первый индекс в нумерации правил:

$r_1 : N \xrightarrow{p_1} l,\ p_1 = 7/8 = 0{,}875,$

$r_2 : N \xrightarrow{p_2} x\bar{x},\ p_2 = 1/8 \cdot \left(7/8\right)^2 \approx 0{,}95700,$

$r_3 : N \xrightarrow{p_3} xxN\bar{x}N\bar{x},\ p_3 = \left(1/8\right)^2 \cdot 7/8 \approx 0{,}1367,$

$r_4 : N \xrightarrow{p_4} x\bar{x}xN\bar{x}N,\ p_4 = \left(1/8\right)^2 \cdot 7/8 \approx 0{,}1367,$

$r_5 : N \xrightarrow{p_5} xxN\bar{x}N\bar{x}xN\bar{x}N,\ p_5 = \left(1/8\right)^3 \approx 0{,}00195.$

На рис. 4.2 изображены деревья вывода в грамматике G, соответствующие правилам грамматики $G(2)$.

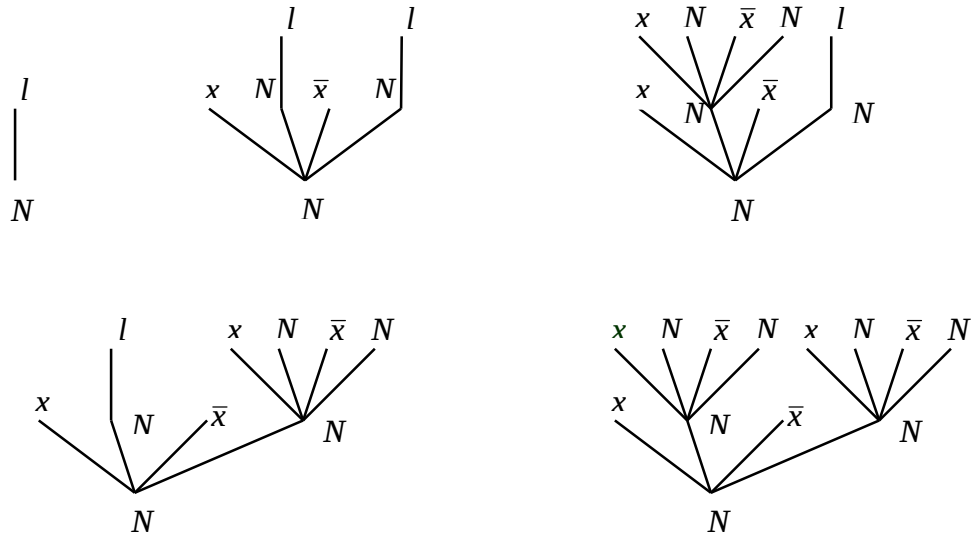


Рис.4.2. Деревья вывода для правил укрупненной грамматики при n=2.

Перронов корень для грамматики $G(2)$ равен $r^2 = \left(1/4\right)^2 = 1/16$. Построим производящую функцию для грамматики $G(2)$:

$$F_1(s_1) = p_1 + p_2 + p_3 \cdot s_1^2 + p_4 \cdot s_1^2 + p_5 \cdot s_1^4.$$

Определим второй момент:

$$b_{11}^1 = \frac{\partial^2 F_1(s_1)}{\partial s_1^2}\Big|_{s_1=1} = 2p_3 + 2p_4 + 12p_5 = \frac{5}{64}.$$

Подсчитаем значение величины B_1:

$$B_1 = \frac{1}{r} \cdot b_{11}^1 \cdot \sum_{r=1}^{\infty} r^{t-1} = \frac{5}{64 \cdot \frac{1}{16} \cdot \frac{15}{16}} = \frac{4}{3}.$$

Определим значения w_{11}, w_{12}, w_{13}, w_{14} и w_{15}:

$$w_{11} \approx 1{,}1666,$$

$$w_{12} \approx 0{,}1276,$$

$$w_{13} \approx 0{,}4557,$$

$$w_{14} \approx 0{,}4557,$$

$$w_{15} \approx 0{,}1276.$$

Отсюда

$$w_1 = \sum_{j=1}^{5} w_{ij} \approx 2{,}3332.$$

Подсчитаем частоты применения правил грамматики $G(2)$:

$$p_1' = \frac{w_{11}}{w_1} = 0{,}5,$$

$$p_2' = \frac{w_{12}}{w_1} = 0{,}0547,$$

$$p_3' = \frac{w_{13}}{w_1} = 0{,}1953,$$

$$p_4' = \frac{w_{14}}{w_1} = 0{,}1953,$$

$$p_5' = \frac{w_{15}}{w_1} = 0{,}0547.$$

Применим алгоритм Хаффмана к полученным частотам. Получим следующую схему кодирования f:

$$f : \begin{cases} v_1 = 0 \\ v_2 = 1010 \\ v_3 = 11 \\ v_4 = 100 \\ v_5 = 1011 \end{cases}$$

Здесь v_i - элементарный код для правила $r_i (i = 1,2,3,4,5)$.

Подсчитаем стоимость кодирования. После несложных преобразований получим, что

$$C\left(L_{G(2)}, f\right) \approx 0{,}957.$$

Таким образом, для n = 2 мы получили меньшее значение стоимости кодирования, чем для n = 1.

В заключение скажем несколько слов о временной сложности построенного алгоритма асимптотически оптимального кодирования. Сложность построенного алгоритма можно характеризовать с двух сторон. Во-первых, можно рассматривать сложность построения схемы локально-префиксного кодирования по заданной грамматике. Во-вторых, можно характеризовать алгоритм сложностью кодирования и декодирования сообщения, являющегося словом языка из рассматриваемого класса.

Известно, что временная сложность алгоритма Хаффмана не более чем квадратично зависит от числа букв в алфавите. Если КС-грамматика содержит k нетерминалов,

алгоритм Хаффмана придется применить k раз, отдельно для каждого множества правил $R_i(i=1,\mathbf{K},k)$.

Пусть l_i - число правил в множестве R_i и $l=\max\{l_1,l_2,\mathbf{K},k\}$. Тогда алгоритм построения локально-префиксного кода по грамматике имеет временную сложность $O(kl^2)$.

При переходе к грамматике $G(n)$ число правил в множестве $R_i(n)$ может расти экспоненциально. Однако строить схему локально-префиксного кодирования для ее последующего использования придется один раз.

Рассмотрим временную сложность кодирования сообщения a , являющегося словом КС-языка. Алгоритм кодирования сообщения можно разбить на два этапа.

Этап 1. Построение левого вывода слова a в грамматике. Для КС-грамматики с однозначным выводом левый вывод строится за время $O(|a|^2)$.

Этап 2. Кодирование левого вывода в соответствии со схемой локально-префиксного кодирования. Временная сложность кодирования на этапе 2 имеет линейный порядок от длины вывода. Так как длина вывода для слова в случае КС-грамматики с однозначным выводом имеет порядок $O(|a|)$, выполнение этапа 2 требует не более $O(|a|)$ операций.

Суммарная временная сложность алгоритма асимптотически оптимального кодирования сообщения длины m равна $O(m^2)$.

Нетрудно показать, что алгоритм декодирования для асимптотически оптимального кодирования также имеет квадратичную временную сложность от длины сообщения.

4.6. Кодирование стохастических КС-языков в критическом случае

В этом разделе рассматриваются вопросы экономного кодирования сообщений, являющихся словами стохастического КС-языка. При этом рассматривается случай, когда перронов корень матрицы первых моментов равен 1 (критический случай). Устанавливаются асимптотические свойства деревьев вывода, и приводится полученная на их основе неулучшаемая нижняя оценка стоимости кодирования. Описывается алгоритм асимптотически оптимального кодирования, в основе которого, так же, как и в докритическом случае, лежит блочное кодирование деревьев вывода.

4.6.1. Закономерности в деревьях вывода слов стохастического КС-языка в критическом случае,

Пусть G - стохастическая КС-грамматика, для которой матрица первых моментов неразложима, непериодична и ее перронов корень равен 1. Через $M_i(t,t)$ обозначим условное математическое ожидание числа вершин, помеченных нетерминалом A_i, в деревьях вывода высоты t на ярусе t (i=1,2, …,k).

Теорема 4.12. *Пусть D_1^t - множество деревьев вывода высоты t для слов языка, порождаемого стохастической КС-грамматикой с неразложимой и непериодической матрицей первых моментов, для которой перронов корень равен единице.*

Тогда для любого e из интервала (0, 1) при $t \to \infty$ и $t \cdot \sqrt{e} \le t \le t \cdot (1 - \sqrt{e})$ выполняется следующее равенство:

$$M_i(t,t) = \frac{v_i B t \cdot (t-t)}{t} \cdot (1 + c_i(t,t,e))(i = 1, \mathbf{K}, k),$$

где $|c_i(t,t,e)| \le c_0 \cdot e$ и c_0 - некоторая константа, не зависящая от t и t .

В формулировке теоремы v_i - i-я компонента левого собственного вектора для перронова корня и B - константа, определяемая формулой

$$B = \sum_{l,m,n} v_n u_l u_m b_{lm}^n.$$

Обозначим через $M_{ij}(t,t)$ условное математическое ожидание числа применений правила r_{ij} на ярусе t в деревьях вывода из D_1^t.

Теорема 4.13. *Пусть D_1^t - множество деревьев вывода высоты t для слов языка, порождаемого стохастической КС-грамматикой с неразложимой и непериодической матрицей первых моментов, для которой перронов корень равен единице.*

Тогда для любого e из интервала (0, 1) при $t \to \infty$ и $t \cdot \sqrt{e} \le t \le t \cdot (1 - \sqrt{e})$ выполняется следующее равенство:

$$M_{ij}(t,t) = \frac{p_{ij} v_i B t \cdot (t-t)}{t} \cdot (1 + c_i(t,t,e))(i = 1, \mathbf{K}, k),$$

где $|c_i(t,t,e)| \le c_0 \cdot e$ и c_0 - некоторая константа, не зависящая от t и t .

Обозначим через $M(t,t)$ условное математическое ожидание общего числа вершин, помеченных нетерминалами, на ярусе t в деревьях вывода из D_1^t.

Следствие. *Для любого* e *из интервала (0,1) при* $t \to \infty$ *и* $t \cdot \sqrt{e} \le t \le t \cdot \left(1 - \sqrt{e}\right)$ *выполняется равенство*

$$M(t,t) = \frac{Bt \cdot (t-t)}{t} \cdot (1 + c_i(t,t,e))(i = 1, \mathbf{K}, k),$$

где $|c_i(t,t,e)| \le c_0 \cdot e$ *и* c_0 *- некоторая константа, не зависящая от* t *и* t *.*

Через $S_{ij}(t)$ обозначим общее число применений правила r_{ij} в дереве вывода из D_1^t.

Теорема 4.14. *Пусть* $t \to \infty$. *Тогда справедливо следующее асимптотическое равенство:*

$$M(S_{ij}(t)) \sim \frac{p_{ij} v_i B t^2}{6}.$$

Для иллюстрации полученных результатов рассмотрим тот же пример, что и для докритического случая.

Пример 4.7. Пусть язык $\boldsymbol{L}$ порождается грамматикой $G = \langle \{x, \bar{x}\}, \{N\}, R, N \rangle$, где множество R содержит два правила:

$r_{11} : N \xrightarrow{p} xN\bar{x}N$,

$r_{12} : N \xrightarrow{1-p} l$ (l - пустое слово).

Напомним,что производящая функция имеет вид

$$F_1(s) = ps_1^2 + (1-p)s_1^0 = ps_1^2 + (1-p).$$

Определим значения $a_1^1 = \frac{\partial F_1(s_1)}{\partial s_1}\Big|_{s_1=1} = 2p$, $b_{11}^1 = \frac{\partial^2 F_1(s_1)}{\partial s_1^2}\Big|_{s_1=1} = 2p.$

Так как матрица первых моментов состоит из одного элемента, его значение является перроновым корнем. Поэтому $r = a_1^1 = 2p.$

Отсюда $r = 1$ при $p = \frac{1}{2}.$

Значит, $p = 1 - p$ и вероятности правил, задаваемые грамматикой, совпадают.

Вычислим константу B, используя приведенную ранее формулу:

$$B = 2p = 1.$$

Найдем математические ожидания числа применений r_{11} и r_{12} в дереве вывода высоты t при $t \to \infty$:

$$M(S_{11}(t)) \sim \frac{pBt^2}{6} = \frac{t^2}{12},$$

$$M\left(S_{12}(t)\right) \sim \frac{(1-p)Bt^2}{6} = \frac{t^2}{12}.$$

Таким образом, для грамматики G в длинных словах порождаемого языка правила r_{11} и r_{12} применяются с одинаковой частотой, что определяется равными исходными вероятностями, заданными для правил грамматики. Это иллюстрирует тот факт, что в критическом случае не происходит перераспределения частот применения правил в длинных словах языка, в отличие от докритического случая.

4.6.2. Нижняя оценка стоимости кодирования и асимптотически оптимальное кодирование в критическом случае

Пусть $\boldsymbol{L}=(L,P)$ - стохастический язык, порождаемый стохастической КС-грамматикой с однозначным выводом. При этом рассматривается случай, когда матрица первых моментов неразложима, непериодична, и ее перронов корень равен единице (критический случай).

Через L^t обозначим множество таких слов из L, что дерево вывода каждого слова имеет высоту t. Для $a \in L^t$ через $p_t(a)$ обозначим условную вероятность появления слова a, т.е. $p_t(a) = \dfrac{p(a)}{P\left(L^t\right)}$. В силу однозначности вывода $P\left(L^t\right) = P\left(D^t\right)$.

Под стоимостью кодирования f, как и в докритическом случае, будем понимать величину

$$C(\boldsymbol{L},f) = \lim_{t\to\infty} \frac{\sum_{a\in L^t} p_t(a)\cdot\left|f(a)\right|}{\sum_{a\in L^t} p_t(a)\cdot\left|a\right|}.$$

(здесь $|x|$ -длина последовательности x).

Через $f(L)$ обозначим класс всех инъективных отображений из L в $\{0,1\}^+$, для которых существует $C(\boldsymbol{L}, f)$.

Для множества слов L^t под энтропией будем понимать величину

$$H(L^t) = -\sum_{a\in L^t} p_t(a)\cdot\log p_t(a).$$

Теорема 4.15. *Пусть* $\boldsymbol{L}$ *- язык, порожденный стохастической КС-грамматикой с однозначным выводом, для которой матрица первых моментов неразложима, непериодична и ее перронов корень равен 1. Тогда*

$$H\left(L^t\right) \sim \frac{Bt^2}{6}\cdot\sum_{i=1}^{k} v_i \cdot H\left(R_i\right),$$

где $V=(\nu_1,\mathbf{K},\nu_k)$ - *левый собственный вектор матрицы первых моментов, соответствующий перронову корню* r *при нормировке* $\sum_{i=1}^{k}\nu_i=1$, *и* $H(R_i)$ - *энтропия множества правил* R_i, $H(R_i)=-\sum_{j=1}^{n_i}p_{ij}\log p_{ij}$.

Следствие.

$$H(L^t)=O(t^2).$$

Теорема 4.16. *Пусть* $\boldsymbol{L}$ - *язык, порожденный стохастической КС-грамматикой с однозначным выводом, для которой матрица первых моментов неразложима, непериодична и перронов корень равен 1.*

Тогда для любого кодирования $f\in F(L)$ *стоимость кодирования* $C(\boldsymbol{L},f)$ *удовлетворяет следующему неравенству:*

$$C(\boldsymbol{L},f)\ge-\frac{1}{h}\sum_{i=1}^{k}\nu_i\sum_{j=1}^{n_i}p_{ij}\log p_{ij}.$$

где $h=\sum_{i=1}^{k}\nu_i\sum_{j=1}^{n_i}p_{ij}l_{ij}$, p_{ij} - *вероятность правила* r_{ij}, $V=(\nu_1,\mathbf{K},\nu_k)$ - *левый положительный собственный вектор для перронова корня* r *при нормировке* $\sum_{i=1}^{k}\nu_i=1$, *и* l_{ij} - *число терминальных символов в правой части правила* r_{ij}.

Нижняя оценка стоимости кодирования $-\frac{1}{h}\sum_{i=1}^{k}\nu_i\sum_{j=1}^{n_i}p_{ij}\log p_{ij}$ является неулучшаемой, так как справедлива следующая теорема.

Теорема 4.17. . *Пусть* $\boldsymbol{L}$ - *язык, порожденный стохастической КС-грамматикой с однозначным выводом, для которой матрица первых моментов неразложима, непериодична и перронов корень равен* 1.

Тогда существует последовательность кодирований

$$\{f_n: f_n\in F(L), n=1,2,\mathbf{K}\}\ \textit{такая, что}$$

$$C(\boldsymbol{L},f_n)-C_0(\boldsymbol{L})\to 0\ \textit{при}\ n\to\infty.$$

Алгоритм блочного кодирования деревьев вывода является асимптотически оптимальным и в критическом случае. Поэтому все утверждения для докритического случая, относящиеся к сложности алгоритма асимптотически оптимального кодирования, остаются справедливыми для критического случая.

Проиллюстрируем некоторые результаты на примере рассмотренной ранее грамматики.

Пример 4.8. Для грамматики $G = \langle \{x, \bar{x}\}, \{N\}, R, N \rangle$, где множество R содержит два правила:

$r_{11} : N \xrightarrow{p} xN\bar{x}N$,

$r_{12} : N \xrightarrow{1-p} l$ (l - пустое слово),

показано, что перронов корень $r = 1$ при $p = 1/2$.

Для математических ожиданий числа применений правил r_{11} и r_{12} установлено, что

$$M(S_{11}(t)) = M(S_{12}(t)) = \frac{t^2}{12}.$$

Вычислим величину h:

$$h = p_{11} \cdot l_{11} + p_{12} \cdot l_{12} = \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 0 = 1.$$

Перейдем к вычислению стоимости оптимального кодирования:

$$C_0(L_G) = \frac{1}{h} \cdot H(R_i) = \frac{1}{h} \cdot (-p_{11} \log p_{11} - p_{12} \log p_{12}) = 1.$$

Применим алгоритм Шеннона на множестве правил грамматики с заданным на нем распределением вероятностей $P = (1/2, 1/2)$. Получим следующую схему кодирования f_{sh}:

$$f_{sh} : \begin{cases} v_{11} = 0, \\ v_{12} = 1. \end{cases}$$

Здесь v_{1i} - элементарный код для правила r_{1i} (i=1,2).

Подсчитаем стоимость кодирования. Очевидно, $C(L_G, f_{sh}) = 1$. Построенное кодирование является оптимальным уже при n =1.

Таким образом, асимптотически оптимальные алгоритмы кодирования для стохастических КС-языков удалось построить, отказавшись от алфавитного кодирования и применив метод кодирования, существенно использующий свойства деревьев вывода слов КС-языка. Отметим, что построенные алгоритмы имеют не более чем квадратичную временную сложность, то есть являются эффективными.

ГЛАВА 5. КРАТКИЙ ОБЗОР МЕТОДОВ ЭКОНОМНОГО КОДИРОВАНИЯ, ИСПОЛЬЗУЕМЫХ В ПРИЛОЖЕНИЯХ

С развитием вычислительной техники и средств связи появились новые алгоритмы экономного кодирования.

При разработке новых алгоритмов основное внимание уделяется эффективности сжатия информации и трудоемкости алгоритмов кодирования и декодирования. В этой главе приводится краткий обзор наиболее известных методов кодирования, применяемых на практике и появившихся в последние годы.

5.1. Арифметическое кодирование

Арифметическое кодирование является одним из наиболее известных методов, применяемых для сжатия текстов. Идея арифметического кодирования была высказана П.Элайесом. В 1976 году Й.Риссанен предложил первый эффективный метод вычисления арифметического кода.

Основная идея метода арифметического кодирования состоит в следующем.

Пусть $B = \{b_1, \mathbf{K}, b_m\}$ - алфавит, на буквах которого задано распределение вероятностей $P = \{p_1, \mathbf{K}, p_m\}$. Рассмотрим B^N - множество сообщений длины N в алфавите B. Каждому слову $a \in B^N$ ставится в соответствие полуинтервал $[l_a, r_a)$ на отрезке [0,1]. Полуинтервалы различных слов из B^N не пересекаются и строятся следующим образом.

Пусть $a = b_{i_1} b_{i_2} \mathbf{K} b_{i_N}$. На первом шаге отрезок [0,1] разбивается на отрезки, пропорциональные вероятностям букв алфавита, и в качестве полуинтервала буквы b_{i_1} выбирается $[l_1, r_1)$, где $l_1 = \sum_{j=1}^{i_1 - 1} p_j$ и $r_1 = \sum_{j=1}^{i_1} p_j$.

На втором шаге полученный полуинтервал $[l_1, r_1)$ снова разбивается на части, пропорциональные вероятностям появления букв алфавита, и т.д.

На k-ом шаге границы полуинтервала $[l_k, r_k)$ определяются по формулам

$$l_k = l_{k-1} + s \cdot \sum_{j=1}^{i_k - 1} p_j,$$

$$r_k = l_{k-1} + s \cdot \sum_{j=1}^{i_k} p_j,$$

где $s = r_{k-1} - l_{k-1}$ - длина предыдущего полуинтервала, полученного для подслова $a_{k-1} = b_{i_1} b_{i_2} \mathbf{K} b_{i_{k-1}}$ слова $a = b_{i_1} b_{i_2} \mathbf{K} b_{i_N}$. Процесс продолжается до тех пор, пока не будет найден полуинтервал $[l_N, r_N)$ для всего слова a.

Далее по построенному полуинтервалу $[l_N, r_N)$ строится двоичный код слова a следующим образом. Внутри $[l_N, r_N)$ выбирается рациональное число, запись которого в двоичной системе счисления содержит наименьшее число разрядов после запятой. В качестве кода слова a выбирается последовательность из 0 и 1, образующая дробную часть числа.

Очевидно, полуинтервалы для слов из B^N не пересекаются, поэтому кодирование на множестве слов B^N будет обладать свойством инъективности..

Пусть S – источник Бернулли (источник с одним состоянием), t – точность арифметических операций в битах и $R(f,S)$ - избыточность арифметического кодирования, т.е. разность между средним числом бит, приходящихся на кодирование одной буквы, и энтропией источника.

Теорема 5.1. *Пусть S – такой источник Бернулли в алфавите $B = \{b_1, \mathbf{K}, b_m\}$, что для любого i выполнены неравенства $\frac{1}{2^{t-3}} \le p_i \le \frac{1}{4}$. Тогда для арифметического кодирования f с параметром t справедливо равенство*

$$R(f,S) \le \frac{\max 1/p_i}{2^{t-3} \ln 2}.$$

Из теоремы следует, что арифметическое кодирование, учитывающее распределение вероятностей букв алфавита, является асимптотически оптимальным при $t \to \infty$.

Проиллюстрируем алгоритм арифметического кодирования на следующем примере.

Пример 5.1. Пусть $B = \{a,b,c,d\}$ и $P = \left(\frac{3}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}\right)$

Построим арифметический код для слова $a = acabbcad$, N=8. Для большей наглядности будем представлять полуинтервалы в десятичной и двоичной системах счисления.

Шаг 1. $a_1 = a$, $l_1 = 0$, $r_1 = \frac{3}{8}$, полуинтервал $[l_1, r_1) = \left[0, \frac{3}{8}\right) = [0;0{,}011)$

Шаг 2. $a_2 = ac$, $s = \frac{3}{8}$, $l_2 = 0 + \frac{3}{8} \cdot \frac{5}{8} = \frac{15}{2^6}$, $r_2 = 0 + \frac{3}{8} \cdot \frac{7}{8} = \frac{21}{2^6}$,

полуинтервал $[l_2, r_2) = \left[\frac{15}{2^6}, \frac{21}{2^6}\right) = [0{,}001111; 0{,}010101)$.

Шаг 3. $a_3 = aca$, $s = \frac{21}{2^6} - \frac{15}{2^6} = \frac{6}{2^6}$, $l_3 = \frac{15}{2^6} + \frac{6}{2^6} \cdot 0 = \frac{15}{2^6}$, $r_3 = \frac{15}{2^6} + \frac{6}{2^6} \cdot \frac{3}{8} = \frac{69}{2^8}$,

полуинтервал $[l_3, r_3) = \left[\frac{15}{2^6}, \frac{69}{2^8}\right) = [0{,}001111; 0{,}01000101)$.

Шаг 4. $a_4 = acab$, $s = \frac{9}{2^8}$, $l_4 = \frac{507}{2^{11}}$, $r_4 = \frac{525}{2^{11}}$,

полуинтервал $[l_4, r_4) = \left[\frac{507}{2^{11}}, \frac{525}{2^{11}}\right) = [0{,}00111111001; 0{,}01000001101)$.

Шаг 5. $a_5 = acabb$, $s = \frac{18}{2^{11}}$, $l_5 = \frac{2055}{2^{13}}$, $r_5 = \frac{2073}{2^{13}}$,

полуинтервал $[l_5, r_5) = \left[\frac{2055}{2^{13}}, \frac{2073}{2^{13}}\right) = [0{,}0100000000111; 0{,}0100000011001)$.

Шаг 6. $a_6 = acabbc$, $s = \frac{18}{2^{13}}$, $l_6 = \frac{8265}{2^{15}}$, $r_6 = \frac{8283}{2^{15}}$,

полуинтервал $[l_6, r_6) = \left[\frac{8265}{2^{15}}, \frac{8283}{2^{15}}\right) = [0{,}010000001001001; 0{,}010000001011011)$.

Шаг 7. $a_7 = acabbca$, $s = \frac{18}{2^{15}}$, $l_7 = \frac{8265}{2^{15}}$, $r_7 = \frac{33087}{2^{17}}$,

полуинтервал $[l_7, r_7) = \left[\frac{8265}{2^{15}}, \frac{33087}{2^{17}}\right) = [0{,}010000001001001; 0{,}01000000100111111)$.

Шаг 8. $a_8 = a = acabbcad$, $s = \frac{27}{2^{17}}$, $l_8 = \frac{264480}{2^{20}}$, $r_8 = \frac{33087}{2^{17}}$,

полуинтервал

$[l_8, r_8) = \left[\frac{264480}{2^{20}}, \frac{33087}{2^{17}}\right) = [0{,}01000000100100100000; 0{,}01000000100111111)$.

В качестве рационального числа для слова a выберем число $0{,}0100000010011 \in [l_8, r_8)$, имеющее минимальное число двоичных разрядов после запятой. Кодом слова a будет последовательность 0100000010011. Для однозначного декодирования необходимо кроме кода знать длину слова a и распределение вероятностей на множестве букв алфавита.

Для реализации изложенного алгоритма требуется арифметика большой точности при кодировании длинных сообщений, поэтому в таком виде алгоритм арифметического кодирования не может применяться на практике. В 1987 году в работе И.Виттена был описан алгоритм арифметического кодирования, использующий вычисления

фиксированной точности, что сделало возможным использование арифметического кодирования для сжатия реальных данных больших размеров.

Приведем описание алгоритма.

Пусть t – длина сообщения. Выберем наименьшее R, удовлетворяющее условиям: $R > 4T$, $R = 2^k$.

Через $C(j)$ обозначим число букв b_j в сообщении длины T, $\sum_j C(j) = T$.

Вместо вероятностей p_j будем использовать величины $C(j)$, а вместо полуинтервала [0, 1) - отрезок целых чисел [0,1, …, R-1].

Для представления чисел рассматриваемого отрезка необходимо не более $\log_2 R = k$ двоичных разрядов.

Текущий отрезок определяется левой и правой границами l и r, ,будем записывать его как [l, r]. Длина s отрезка равна $r - l + 1$. Основная идея алгоритма состоит в том, чтобы всегда сохранять длину отрезка больше, чем $R/4$, расширяя отрезок всякий раз, когда он становится слишком маленьким. Ограничение на длину отрезка гарантирует, что для любой буквы сообщения построится отрезок ненулевой длины.

Каждый раз, когда двоичное представление границ l и r имеет общий префикс, префикс длины 1 отправляется в код слова, и l сдвигается влево на 1 и дополняется справа нулем, в то время как r сдвигается влево и дополняется справа единицей.

Отрезок для очередной буквы b_i слова вычисляется с помощью частот $C(j)$ по формулам:

$$l_k = l_{k-1} + \left\lceil \frac{Cf(i-1)}{Cf(m)} \cdot (r_{k-1} - l_{k-1} + 1) \right\rceil,$$

$$r_k = l_{k-1} + \left\lceil \frac{Cf(i)}{Cf(m)} \cdot (r_{k-1} - l_{k-1} + 1) \right\rceil - 1.$$

При вычислении границ используются накопленные частоты $Cf(i) = \sum_{j=1}^{i} C(j)$.

Если текущий отрезок становится слишком коротким (меньше T), он расширяется до отрезка $\left[2 \cdot \left(l - 2^{k-2}\right), 2 \cdot \left(r - 2^{k-2}\right) + 1\right)$, при этом счетчик ожидания c увеличивается на 1. Эта операция повторяется до тех пор, пока отрезок остается коротким. Когда в код посылается очередной бит (общий префикс длины 1), вслед за ним в код с противоположным значением добавляются биты, число которых равно счетчику ожидания c.

Для однозначного декодирования необходимо дополнительно передать частоты букв алфавита.

Проиллюстрируем этот алгоритм на том же примере.

Пример 5.2. Пусть $B=\{a,b,c,d\}$ и $P=\left(\frac{3}{8},\frac{1}{4},\frac{1}{4},\frac{1}{8}\right)$.

Закодируем слово $a = acabbcad$, T=8.

Предварительно построим таблицу накопленных частот:

Символ	Частота *C(j)*	Накопленная частота *Cf(j)*
a	3	3
b	2	5
c	2	7
d	1	8

Найдем R из условий $R > 4T$, $R = 2^k$. Очевидно, R=64. Поэтому начальный отрезок равен [0, 63]=[000000, 111111]$_2$.

Шаг 1. $a_1 = a$, $l_1 = 0$, $r_1 = 23$, отрезок $[l_1, r_1] = [0;23] = [000000;010111]$.

Общий префикс равен 0, он отсылается в код: f_1=0. После этого производится преобразование границ отрезка.

Новый отрезок $[l_1, r_1] = [000000;101111] = [0;47]$.

Шаг 2. $a_2 = ac$, $l_2 = 30$, $r_2 = 42$. Используя таблицу накопленных частот, находим отрезок для буквы *c*: $[l_2, r_2] = [30;42] = [011110;101010]$. Общего префикса нет, в код ничего не добавляется. Длина отрезка больше T, отрезок не преобразуется.

Шаг 3. $a_3 = aca,$. Находим отрезок для буквы *a*:

$[l_3, r_3] = [30;34] = [011110;100010]$.

Длина отрезка равна 5, что меньше T=8, поэтому в счетчик ожидания *c* прибавляем 1 (c=1) и расширяем отрезок.

Новый отрезок $[l_3, r_3] = [011100;100101] = [28;37]$.

Общего префикса нет, в код ничего не добавляется.

Шаг 4. $a_4 = acab$. Находим отрезок для буквы *b*:

$[l_4, r_4] = [32;34] = [100000;100010]$.

Общий префикс длины 1 равен 1, он отсылается в код, после этого к коду добавляется один ноль, так как счетчик ожидания равен 1: f_4=010. Счетчик ожидания после его использования полагается равным 0. Далее производится преобразование границ отрезка.

Новый отрезок $[l_4, r_4] = [000000;000101] = [2;47]$. Счетчик ожидания равен нулю, Общий префикс 000 добавляем к коду: f_4=010000.

Преобразуем отрезок: $[l_4, r_4] = [000000;101111] = [0;47]$.

Шаг 5. $a_5 = acabb$. Находим отрезок для буквы *b*:

$[l_5, r_5] = [18;29] = [0100010;011101]$.

Общий префикс равен 01, добавляем его к коду: f_4=01000001.

Преобразуем отрезок: $[l_5, r_5] = [000100;110111] = [4;55]$.

Шаг 6. $a_6 = acabbc$. Находим отрезок для буквы *c*:

$[l_6, r_6] = [37;50] = [100101;110010]$.

Общий префикс равен 1, добавляем его к коду: f_6=010000011.

Преобразуем отрезок: $[l_6, r_6] = [001010;100101] = [10;37]$.

Шаг 7. $a_7 = acabbca$. Находим отрезок для буквы *a*:

$[l_7, r_7] = [10;20] = [001010;010100.]$.

Общий префикс равен 0, добавляем его к коду: f_7=0100000110.

Преобразуем отрезок: $[l_7, r_7] = [010100;110001] = [20;49]$.

Шаг 8. $a_8 = acabbcad$. Находим отрезок для буквы *d*:

$[l_8, r_8] = [45;49] = [101101;110001.]$.

Общий префикс равен 1, добавляем его к коду: f_8=01000001101.

Преобразуем отрезок: $[l_8, r_8] = [011010;100011] = [26;35]$.

Счетчик ожидания пуст, число $R/2$ принадлежит отрезку, в качестве символа конца кода добавляем два первых бита числа $R/2$=32=100000.

Окончательно получаем *f*=0100000110110.

Для изложенных вариантов арифметического кодирования требуется дополнительная информация о частотах (вероятностях) букв в сообщении. Поэтому практическая реализация требует двух проходов сообщения: один раз – для сбора статистики, и второй раз – непосредственно для кодирования, основанного на собранной статистике.

Отметим, что существуют адаптивные варианты алгоритма арифметического кодирования, в которых частотные характеристики для букв алфавита накапливаются и корректируются в процессе чтения и одновременного кодирования букв сообщения, и, таким образом, кодирование осуществляется за один проход.

5.2. Алгоритмы Зива-Лемпеля

Алгоритмы Зива-Лемпеля относятся к словарным методам сжатия, основанным на использовании словаря. Алгоритмы разработаны Якобом Зивом и Авраамом Лемпелом. Они известны под названиями LZ77 и LZ78 и опубликованы в 1977 и 1978 годах.

5.2.1. Алгоритм LZ77

Основная идея LZ77 состоит в том, что повторные вхождения некоторой подстроки сообщения заменяются ссылкой на ее первое вхождение.

Алгоритм использует скользящее окно, разделенное на две части: словарь и буфер. Скользящее окно по мере построения кода передвигается вдоль сообщения. Словарь включает уже просмотренную подстроку сообщения фиксированной длины. Буфер включает текущую еще не закодированную подстроку сообщения. Алгоритм пытается найти в словаре фрагмент наибольшей длины, совпадающий с начальной подстрокой буфера, и кодирует, заменяет найденную подстроку буфера с помощью ссылки на место этой подстроки в словаре.

Обозначим длину словаря через W, и длину буфера через M. Для эффективности кодирования размер словаря должен быть существенно больше размера буфера.

Алгоритм LZ77 выдает коды, состоящие из трех элементов:

1) смещение в словаре относительно его начала подстроки, совпадающей с началом содержимого буфера;
2) длина подстроки;
3) первый символ буфера, следующий за подстрокой.

Пример 5.3. Пусть длина словаря W=5, длина буфера M=3. Закодируем с помощью алгоритма LZ77 сообщение $a = abbdcabdcaabdaa$.

Процесс кодирования представим в виде следующей таблицы:

Шаг	Словарь	Буфер	Код
1	-----	abb	(1,0,$"a"$)
2	----a	bbd	(1,0,$"b"$)
3	---ab	bdc	(1,1,$"d"$)

4	*-abbd*	*cab*	(1,0,*"c"*)
5	*abbdc*	*abd*	(5,2,*"d"*)
6	*dcabd*	*caa*	(4,2,*"a"*)
7	*bdcaa*	*bda*	(5,2,*"a"*)
8	*aabda*	*a*	(0,0,*"a"*)

В начале работы словарь пуст, а буфер содержит начальную подстроку сообщения *abb*. На первом шаге строится код (1,0,*"a"*). Так как символы буфера не встречаются в словаре, смещение равно 1 (отсчет ведется с самой правой позиции в словаре, она имеет номер 1, хотя это не принципиально), длина подстроки совпадения равна 0 – это значение второго элемента кода, третий элемент равен *"a"* – это код очередной буквы после найденной подстроки буфера (например, ASCII-код). После выполнения первого шага скользящее окно смещается вправо на 1 позицию.

На шаге 5 максимальное совпадение начальной подстроки буфера со словарем равно *ab*, поэтому смещение равно 5 (это номер самой левой подстроки словаря), длина подстроки равна 2, и следующий после найденной подстроки в буфере – символ *d*. После выполнения шага 5 скользящее окно смещается вправо на 3 позиции.

На шаге 8, так как символ *a* последний в сообщении, словарь не используется.

Длину кода можно оценить следующим образом. Длина подстроки не может быть больше длины буфера M, а смещение не может быть больше размера словаря W. Следовательно, для двоичного кода длины подстроки достаточно $\lceil \log_2 M \rceil$ битов, а для двоичного кода смещения достаточно $\lceil \log_2 W \rceil$ битов.

При использовании ASCII-кода общая длина кода будет равна $N \cdot (\lceil \log_2 W \rceil + \lceil \log_2 M \rceil + 8)$, где N – число шагов. Сжатие алгоритмом LZ77 достигается за счет шагов, аналогичных шагу 5. Различные модификации алгоритма позволяют увеличить степень сжатия.

К недостаткам алгоритма LZ77 следует отнести следующие:

1) невозможность кодирования подстрок, находящихся друг от друга на расстоянии, большем длины словаря;

2) длина подстроки, которую можно закодировать, ограничена размером буфера.

Известны оценки избыточности кодирования LZ77 для источников с конечным числом состояний, приведем одну из них, принадлежащую Х.Морита и К.Кобояши:

$$R(f_{77}, X) = O\left(\frac{\log\log W}{\log W}\right) \text{ при } W \to \infty$$

(здесь X – приизвольный марковский источник и W – длина скользящего окна).

5.2.2. Алгоритм LZ78

Алгоритм LZ78 был разработан в 1978 году. Алгоритм не использует скользящее окно, он хранит словарь из уже просмотренных фраз. В начале работы словарь содержит только одну пустую строку (строку длины 0). Алгоритм считывает символы сообщения до тех пор, пока накапливаемая строка входит целиком в одну из строк словаря. После этого алгоритм строит код, состоящий из индекса (номера) строки в словаре, которая совпадает с кодируемой подстрокой сообщения, и следующего символа сообщения. Затем в словарь добавляется закодированная подстрока сообщения плюс следующий символ.

Размер получаемого кода определяется размером словаря, так как каждый код содержит номер строки в словаре.

Пример 5.4. Закодируем с помощью алгоритма LZ78 сообщение $a = abbdcabdcaabdaa$. Процесс кодирования представим в виде следующей таблицы:

Номер строки в словаре	Словарь	Код	Текущая строка сообщения	Строка **Str**
0	- (пустая строка)	(0, *“a”*)	*abbdcabdcaabdaa*	пустая строка
1	*a*	(0, *“b”*)	*bbdcabdcaabdaa*	пустая строка
2	*b*	(2, *“d”*)	*bdcabdcaabdaa*	*b*
3	*bd*	(0, *“c”*)	*dcabdcaabdaa*	пустая строка
4	*c*	(1, *“b”*)	*abdcaabdaa*	*a*
5	*ab*	(0, *“d”*)	*dcaabdaa*	пустая строка
6	*d*	(4, *“a”*)	*caabdaa*	*c*
7	*ca*	(5, *“d”*)	*abdaa*	*ab*
8	*abd*	(1, *“a”*)	*aa*	*a*

Здесь **Str** – накапливаемая подстрока, которая целиком соответствует какой-либо фразе из словаря (притом максимальная).

На первом шаге построен код (0, *“a”*), так как словарь содержит только пустую строку, имеющую номер 0 в словаре, первый элемент в коде равен 0 (пустая строка является подстрокой любой другой строки). В словарь добавляется строка, получающаяся

конкатенацией пустой строки и следующего после найденной подстроки символа сообщения. Это строка a, она записывается в словарь с номером 1.

На втором шаге так же, как и на первом, максимальная строка словаря, совпадающая с началом текущей строки сообщения, является пустой. Поэтому формируется код (0, *"b"*). В словарь под номером 2 добавляется строка b.

На третьем шаге начальная подстрока сообщения длины 1 встречается в словаре под номером 2. Поэтому создается код (2, *"d"*), в котором первый элемент равен номеру найденной строки в словаре, а второй элемент является кодом следующей буквы. К найденной строке b из словаря приписывается буква d, и строка bd добавляется в словарь.

В рассматриваемом примере наибольший вклад в сжатие сообщения дает шаг 7, поскольку строка совпадения имеет длину 2.

Кодируемое сообщение имеет длину 15. Для сообщения длины 15 словарь может содержать 16 строк, включая пустую строку. Поэтому для записи номера строки в словаре требуется 4 бита. Для записи второго элемента в коде требуется 2 бита, так как алфавит сообщения содержит 4 символа. Для кодирования сообщения потребовалось 8 шагов. Поэтому общая длина кода равна 48 битам.

Алгоритм LZ78 так же, как LZ77, является асимптотически оптимальным для марковских источников с конечным числом состояний.

Известна оценка избыточности для LZ77, полученная С.Савари:

$$R_n(f_{78}, X) \le O\left(\frac{1}{\log n}\right) \text{ при } n \to \infty,$$

где X – произвольный марковский источник и n – длина кодируемого сообщения.

Алгоритм LZ78 нашел свое практическое применение только после реализации LZW, предложенной Велчем (Welch) в 1984 г. Некоторые изменения коснулись устройства словаря и формирования кодов. Рассмотрим эти изменения.

Вначале производится инициализация словаря всеми возможными односимвольными строками (обычно 256 символами расширенного ASCII-кода). В процессе работы словарь разрастается до своего максимального объема V_{max} строк (слов). Обычно объем словаря достигает нескольких десятков тысяч слов. Алгоритм работает практически аналогично алгоритму LZ78: символы сообщения считываются до тех пор, пока накапливаемая подстрока входит целиком в одну из фраз словаря Str. Так как словарь первоначально не пустой, такое слово всегда найдется. Как только эта строка перестанет соответствовать хотя бы одной фразе словаря, генерируется код - индекс строки в словаре, которая до последнего введенного символа содержала входную строку (длина

кода равна $\lceil \log V_{max} \rceil$), а словарь пополняется новым словом: Str + символ S, нарушивший совпадение. И далее алгоритм продолжает свою работу по указанной схеме, начиная с символа S. Таким образом, пропала необходимость передавать один символ в чистом виде (в результате того, что словарь изначально не пустой).

Рассмотрим работу алгоритма на строке *колокол*:

Str	**Код**	**StrS – в словарь**	**Индекс в словаре**
		ASCII	0...255
к	#к	ко	256
о	#о	ол	257
л	#л	ло	258
о	#о	ок	259
ко	256	кол	260
л	#л		

Str – накапливаемая подстрока, которая целиком соответсвует какой-либо фразе из словаря (притом максимальная). Как только появляется символ **S**, нарушающий совпадение, генерируется **Код** – индекс слова в словаре.

StrS – строка Str + символ S, на котором и нарушается совпадение накапливаемой подстроки. Словарь пополняется этим новым словом, которому присваивается соответствующий **индекс**.

Теперь сравним размер исходной строки и закодированной. Для представления кода (а это индекс в словаре) требуется 9 бит, получаем размер закодированной строки = 6*9 = 54 бита. Тогда как размер исходной строки – 8*7 = 56 бит. Как видим, даже на такой короткой последовательности символов достигается сжатие, которое по мере увеличения размера последовательности будет увеличиваться.

Отметим основные проблемы, которые приходится решать при реализации алгоритма LZW.

Чтобы алгоритмы сжатия были эффективными, важна не только степень сжатия информации, но и скорость работы кодера и декодера, поэтому главная проблема – устройство словаря.

Реализация алгоритма построения словаря и поиска в словаре сами по себе достаточно сложные и медленные процедуры. Бинарные деревья позволяют на несколько порядков

ускорить работу архиватора, поэтому они часто используются в реализациях для представления словаря.

Еще одна серьезная проблема алгоритмов семейства LZ78 - переполнение словаря: если словарь полностью заполнен, прекращается его обновление и процесс сжатия может быть заметно ухудшен. Отсюда следует вывод - словарь нужно иногда обновлять, но когда и как значительно? Этому вопросу посвящено множество публикаций. Самый простой способ - как только словарь заполнился, его полностью обновляют. Недостаток очевиден - кодирование начинается на пустом месте, как бы сначала, и пока словарь не накопится, сжатие будет незначительным. Поэтому словарь можно обновлять не сразу после его заполнения, а только после того, как степень сжатия начала падать. Еще более сложными являются эвристические методы обновления словарей в зависимости от частоты использования тех или иных слов.

5.3. Преобразование Барроуза-Уилера

Преобразование Барроуза-Уилера применяется в алгоритмах сжатия для предварительной обработки данных, для того чтобы с помощью перестановки элементов придать исходным данным со сложными зависимостями такие структурные свойства, которые легче смоделировать и учесть при кодировании.

Метод преобразования был опубликован в 1994 году в работе Д.Уилера и М.Барроуза и получил сокращенное название BWT. Преобразование Барроуза-Уилера применяется обычно вместе с методами кодирования, специально для него предназначенными (кодирование длин серий, сжатие с помощью «стопки книг» и др.).

Преобразование BWT - это способ перестановки символов в блоке данных. В нем можно выделить четыре этапа:

1) выделяется блок из входного потока;

2) формируется матрица всех перестановок, полученных в результате циклического сдвига блока;

3) все перестановки сортируются в соответствии с лексикографическим порядком символов каждой перестановки;

4) на выход подается последний столбец матрицы и номер строки, соответствующей исходному блоку.

Пример 5.5. Обычно принято иллюстрировать преобразование на примере блока данных, состоящего из слова «абракадабра». Сформируем матрицу циклических

перестановок слова «абракадабра», где первой строкой будет исходное слово, второй строкой – слово, сдвинутое на один символ влево, и т.д. Получим следующую матрицу:

абракадабра
бракадабраа
ракадабрааб
акадабраабр
кадабраабра
адабраабрак
дабраабрака
абраабракад
браабракада
раабракадаб
аабракадабр

Пометим в этой матрице исходную строку и отсортируем все строки в лексикографическом порядке. Получим следующую матрицу циклических перестановок, отсортированных в соответствии с лексикографическим порядком:

0 аабракадабр
1 абраабракад
2 абракадабра – исходная строка
3 адабраабрак
4 акадабраабр
5 браабракада
6 бракадабраа
7 дабраабрака
8 кадабраабра
9 раабракадаб
10 ракадабрааб

Результатом преобразования Барроуза-Уилера является последний столбец полученной матрицы: «рдакрааааббб».

Покажем, что преобразование BWT является обратимым, т.е. по последнему столбцу отсортированной матрицы и номеру исходной строки однозначно можно восстановить саму исходную строку.

Рассмотрим процесс восстановления отсортированной лексикографически исходной матрицы. Для этого отсортируем все символы последнего столбца.

0 а

1 а

2 а

3 а

4 а

5 б

6 б

7 д

8 к

9 р

10 р

Так как строки матрицы были отсортированы по лексикографическому порядку, отсортировав последний столбец, мы получили первый столбец отсортированной матрицы. Таким образом, мы имеем два столбца матрицы: первый - аааааббдкрр и последний – рдакрааааабб:

0 а . . . . . р

1 а . . . . . д

2 а. . . . . а

3 а. . . . . к

4 а. . . . . р

5 б. . . . . а

6 б. . . . . а

7 д. . . . . а

8 к. . . . . а

9 р. . . . . б

10 р. . . . . б

Строки матрицы были получены в результате циклического сдвига исходной строки. Поэтому символы последнего и первого столбца образуют друг с другом пары. Отсортировав эти пары лексикографически, мы получим первые два столбца матрицы:

0 аа . . . . . р

1 аб . . . . . д

2 аб. . . . . а

3 ад. . . . . к

4 ак. . . . . р

5 бр. . . . . а

6 бр. . . . . а

7 да. . . . . а

8 ка. . . . . а

9 ра. . . . . б

10 ра. . . . . б

Сортируя тройки символов последнего, первого и второго столбцов, восстановим третий столбец, и т.д.

0	ааб . . . . . р	аабр . . . . . р	. . .	аабракада . р	аабракадабр
1	абр . . . . . д	абра . . . . . д	. . .	абраабрак . д	абраабракад
2	абр. . . . . а	абра. . . . . а	. . .	абракадаб . а	абракадабра
3	ада. . . . . к	адаб. . . . . к	. . .	адабраабр . к	адабраабрак
4	ака. . . . . р	акад. . . . . р	. . .	акадабраа . р	акадабраабр
5	бра. . . . . а	браа. . . . . а	. . .	браабрака . а	браабракада
6	бра. . . . . а	брак. . . . . а	. . .	бракадабр . а	бракадабраа
7	даб. . . . . а	дабр. . . . . а	. . .	дабраабра . а	дабраабрака
8	кад. . . . . а	када. . . . . а	. . .	кадабрааб . а	кадабраабра
9	раа. . . . . б	рааб. . . . . б	. . .	раабракад . б	Раабракадаб
10	рак. . . . . б	рака. . . . . б	. . .	ракадабра . б	Ракадабрааб

Восстановив все столбцы матрицы, по номеру исходной строки в матрице найдем саму строку. Таким образом, для восстановления всей матрицы достаточно помнить лишь ее последний столбец.

Рассмотрим вектор обратного преобразования, позволяющий эффективно находить исходную строку, не восстанавливая матрицу. Заметим, что в процессе выявления очередного столбца матрицы мы делали одни и те же действия: получали новую строку, сливая символ из последнего столбца старой строки с известными символами первых столбцов этой же строки. Новая строка после сортировки перемещалась в другую позицию в матрице.

Чтобы получить вектор обратного преобразования, следует определить порядок получения символов первого столбца из символов последнего, т.е. отсортировать матрицу по номерам новых строк.

Номер строки		Номер новой строки	Перенос последнего столбца в начало
2	а . . . . . а	0	аа . . . . . р
5	б . . . . . а	1	аб . . . . . д
6	б. . . . . а	2	аб. . . . . а
7	д. . . . . а	3	ад. . . . . к
8	к. . . . . а	4	ак. . . . . р
9	р. . . . . б	5	бр. . . . . а
10	р. . . . . б	6	бр. . . . . а
1	а. . . . . д	7	да. . . . . а
3	а. . . . . к	8	ка. . . . . а
0	а. . . . . р	9	ра. . . . . б
4	а. . . . . р	10	ра. . . . . б

Вектор обратного преобразования образуют полученные значения номеров строк T={2,5,6,7,8,9,10,1.3,0,4}. Чтобы получить исходную строку, возьмем элемент вектора, соответствующий номеру исходной строки в матрице циклических перестановок, T[2]=6. В качестве первого символа в исходной строке следует взять шестой символ из строки «рдакрааааббб - это символ «а». Для определения второго символа возьмем T[6]=10, это символ «б», и т.д.

6	10	4	8	3	7	1	5	9	0	2
а	б	р	а	к	а	д	а	б	р	А

Главное свойство преобразования Барроуза-Уилера состоит в том, что оно группирует вместе символы, соответствующие похожим контекстам. Практика показывает, что в результате преобразования обычных текстов более половины всех символов следует за такими же.

Чаще всего вместе с преобразованием BWT используется кодирование с помощью алгоритма сжатия «стопкой книг» (англо-язычное название move to front). Алгоритм легко понять, если представить стопку книг, каждая из которых соответствует определенному символу. По мере востребования из стопки вытаскивается нужная книга и помещается сверху. Через некоторое время те книги, которые используются часто, оказываются ближе к верхушке стопки.

Пример 5.6. Рассмотрим наш пример слова «рдакрааааббб», полученного в результате преобразования Барроуза-Уилера. Символы слова принадлежат алфавиту из пяти символов. Начальный список содержит эти символы в следующем порядке: {а,б,д,к,р}. Символ «р» стоит на пятом месте в списке, поэтому первый код становится равным 4. После перемещения символа «р» на первое место список символов принимает вид {р,а,б,д,к}, и т.д.

Символ	Список	Выход
р	{а, б, д,к, р}	4
д	{р, а, б, д,к}	3
а	{д, р, а, б, к}	2
к	{а, д, р, б, к}	4
р	{к, а, д, р, б}	3
а	{р, к, а, д, б}	2
а	{а, р, к, д, б}	0
а	{а, р, к, д, б}	0
а	{а, р, к, д, б}	0
б	{а, р, к, д, б}	4
б	{б, а, р, к, д}	0

В выходной последовательности кодов встречается только четыре различных номера. Применив для их кодирования алгоритм Хаффмана, получим стоимость кодирования, равную двум битам.

Литература

1. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. Том 1.- М.: Мир, 1978.
2. Ватолин Д. и др. Методы сжатия данных. http://compression.graphicon.ru
3. Жильцова Л.П. Об алгоритмической сложности задач оптимального алфавитного кодирования для контекстно-свободных языков // Дискретная математика. - 1989. - Том 1, вып. 2. С. 38 - 51.
4. Жильцова Л.П. Кодирование стохастических контекстно-свободных языков с однозначным выводом // Дискретная математика. - 1994. - том 6, вып. 3. - С. 73 - 88.
5. Жильцова Л.П. Закономерности применения правил грамматики в выводах слов стохастического контекстно-свободного языка // Математические вопросы кибернетики. - М.: Наука. - 2000. - Вып.9. - С. 101 - 126.
6. Жильцова Л.П. О нижней оценке стоимости кодирования и асимптотически оптимальном кодировании стохастического контекстно-свободного языка // Дискретный анализ и исследование операций. - 2001.- Серия 1. - Том 8, N3. Новосибирск: Издательство Института математики СО РАН. - С. 26 - 45.
7. Жильцова Л.П. О кодировании стохастических контекстно-свободных языков // Доклады РАН. - 2002. - Том 383, N 4. - С. 451 - 453.
8. Жильцова Л.П. Закономерности в деревьях вывода слов стохастического контекстно-свободного языка и нижняя оценка стоимости кодирования. Критический случай // Дискретный анализ и исследование операций. - 2003.- Серия 1. - Том 10, N3. - Новосибирск: Издательство Института математики СО РАН. - С. 23 - 53.
9. Кричевский Р.Е. Сжатие и поиск информации. - М.: Радио и связь, 1989.
10. Лидовский В.В. Теория информации. Учебное пособие, 2002. http://compression.ru
11. Марков А.А. Введение в теорию кодирования. - М.: Наука, 1982.
12. Потапов В.Н. Обзор методов неискажающего кодирования дискретных источников. http://compression.ru
13. Фомин А.А. Основы сжатия информации. Санкт-Петербургский госуд.университет. С.-Петербург, 1998.

14. Шеннон К. Математическая теория связи. // Работы по теории информации и кибернетике. - М.: ИЛ, 1963. - С. 243 - 332.
15. Яблонский С.В. Введение в дискретную математику. - М.: Наука,2000.