

## МЕТОДЫ СЖАТИЯ ТЕКСТОВОЙ ИНФОРМАЦИИ

*И. А. Большаков, А. В. Смирнов*

### § 1. Введение

В процессе ускоренной компьютеризации современного общества объемы данных, хранимых на машинных носителях информации, быстро растут. Еще совсем недавно они измерялись килобайтами и мегабайтами, а теперь — гигабайтами и даже более крупными единицами.

Естественно желание хранить эти данные предельно компактно. В человеческом мозгу и в ряде технических систем используется необратимое сжатие информации, т. е. такая ее перекодировка, которая оставляет лишь наиболее существенную смысловую часть. Но поскольку критерий существенности обычно неизвестен или не определен, естествен интерес и к обратимым методам, устраняющим избыточность при кодировании (сжатии) и восстанавливающим ее при декодировании (расжатии). Обратимые методы актуальны, например, для хранения полнотекстовых баз данных в информационно-поисковых системах.

В настоящей работе затрагиваются лишь обратимые методы сжатия. В связи с ними уместно напомнить, что в 1988 году исполняется 40 лет новаторской работе Шеннона [91], заложившей основу статистической теории информации. Последняя ввела понятие эффективного кодирования, на статистических основаниях устраняющего избыточность произвольных цифровых данных. Казалось бы, решение проблемы оптимального сжатия этой работой было намечено раз и навсегда. Потребовалось всего несколько лет, чтобы появились приближающиеся к оптимуму коды Фано—Шеннона и Хаффмена [50], [31], [17]. Однако в дальнейшем выяснилось, что в применении к системам и данным конкретных видов, в частности, к тестовым решение проблемы не завершено. Этому можно дать следующее объяснение:

1) Требуя учета издержки хранения кодировочных таблиц. В теории Шеннона предполагалось, что эти таблицы хра-

нятся в системе связи на передающем и приемном концах, и затраты на хранение (в первую очередь, в виде памяти некоторого объема) во внимание не принимались. Применительно же к системам общего назначения эти затраты принципиально ничем не отличаются от тех, которые минимизируются самим сжатием.

2) Составление кодовых таблиц — нелепая проблема. В теории Шеннона, особенно в иллюстрирующих ее примерах, явно или неявно полагалось, что ансамбли передаваемых сообщений (т. е. кодируемых символов) конечны и невелики (измеряются, скажем, десятками или сотнями). Применительно же к текстам на естественных языках объемы словарей непосредственно образующих текст буквенных цепочек (словформ) измеряются тысячами или десятками тысяч. Набрать достоверную статистику по таким ансамблям оказывается очень сложной, отдельной проблемой.

3) Неустойчивость статистики и кодовых таблиц оказывается еще одним усложняющим фактором. Многочисленными экспериментами показано, что протяженные тексты, рассматриваемые как статистические источники информации, устойчивостью (стационарностью) не обладают, и привлечение в качестве источника, например, книги по новой проблематике может существенно изменить уже накопленную статистику. Ввиду этого нет гарантии и в устойчивой оптимальности кодовых таблиц, на нее опирающихся.

4) Алгоритмическая сложность кодирования/декодирования в ранних работах по теории информации никак не учитывалась. Между тем алгоритмы с большими переборами (их называют комбинаторными) из-за экспоненциального роста количества просматриваемых альтернатив как функции от объема просматриваемых данных плохо укладываются даже в мощные современные компьютеры.

5) Чисто технические особенности современных ЭВМ также влияют на процессы преобразования информации. Отметим, например, байтовую структуру ЭВМ: отдельные биты информации разбиваются на байты — группы, кратные восьми. Операции, не учитывающие эту кратность, выполнимы, но сильно замедляют вычисления.

Указанные особенности говорят, на наш взгляд, о том, что проблему сжатия текстовой информации нельзя считать окончательно решенной и через 40 лет после появления фундаментальной работы Шеннона, причем как в практическом, так и в теоретическом аспектах. Поэтому не удивительно, что все эти годы исследователи обращались к теме обратимого сжатия. Динамика появления таких работ на английском языке (по пятилетиям) отражена в таблице.

### Количество англоязычных работ по пятилетиям.

| 1948—52 | 1953—57 | 1958—62 | 1963—67 | 1968—72 | 1973—77 | 1978—82 | 1982—87 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| 4       | 0       | 5       | 7       | 15      | 32      | 25      | 17      |

Пик исследований в западных странах пришелся на середину 70-х годов, когда стоимость хранения данных была еще относительно высока. Но особого снижения интенсивности исследований нет и в последние 10 лет, для которых характерно быстрое удешевление накопителей информации.

Настоящая статья представляет аналитический обзор работ по обратимому сжатию, появившихся за последние примерно 30 лет. Хотя мы стремились к полноте покрытия развиваемых в литературе концепций и методов, библиографическая полнота оказалась не достижимой, даже при использовании современных методов поиска информации. Большинство обнаруженных работ написаны на английском языке. Советских работ по сжатию текстовой информации довольно мало, и с учетом медленного убывания стоимости накопителей информации в странах СЭВ проблема сжатия явно остается актуальной.

Проблему обратимого сжатия можно рубрицировать исходя из объектов сжатия, каковыми являются:

- числовые данные,
- упорядоченные текстовые данные (словари),
- специальные тексты (на формализованных языках),
- естественноречевые тексты общего вида,
- структурированные данные,
- изображения.

Основной упор мы делаем на тексты общего вида. Сжатие числовых данных, словарей и специальных текстов рассматривается для полноты и сопоставлений. Структурированные данные состоят из числовых и текстовых, так что новых особенностей не привнесут. По части же сжатия изображений было осознано, что эта тема актуальна, развивается в последние годы ускоренно, но из-за специфичности требует отдельного изучения.

Везде ниже будет нужна количественная мера сжатия — так называемый коэффициент сжатия. Для текстов он определяется у различных авторов по-разному, либо в виде отношения разности длин первоначального и сжатого текстов к длине первоначального варианта, либо просто отношения длин первоначального и сжатого текста. Мы везде исходим из второго определения.

## § 2. Сжатие нетекстовой и структурированной информации

Наиболее популярными методами сжатия числовых данных являются разностное кодирование, кодирование повторений и подавление ведущих нулей [5], [109].

Суть разностного кодирования заключается в хранении, вместо абсолютных значений, разностей двух смежных чисел или отклонения чисел от их среднего значения. Первый вариант эффективен для медленно меняющихся последовательностей, второй — когда максимальное отклонение от среднего значительно меньше абсолютного значения среднего. Вторым вариантом может быть обобщен на случай, когда последовательность чисел имеет тенденцию группироваться в сильно разбросанные кластеры [112].

Кодирование повторений, тоже широко используемое, заключается в замене цепочки одинаковых символов кодом этого символа и числом повторений [5], [15], [38]. Подавление ведущих нулей подразумевает отбрасывание незначащих нулей в старших разрядах.

Существуют также простые способы сжатия десятичных данных, основанные на более компактном их представлении в двоичном формате [123], [18].

Под словарями будем понимать списки неповторяющихся цепочек символов в алфавитном или ином строгом порядке. Такой словарь можно рассматривать как монотонную последовательность чисел и для его сжатия применять метод разностного кодирования. Здесь он заключается в отбрасывании у каждого слова начальных букв, совпадающих с начальными символами предыдущего слова и замене их на число отброшенных букв [26]. Аналогичная процедура может быть проделана для обратного словаря, что обеспечивает сжатие за счет сходства окончаний [5].

Однако такой метод неудобен тем, что при декодировании любого конкретного слова требуется последовательно декодировать все предшествующие слова. Поэтому порой используются отдельные перечни наиболее часто встречающихся частей слов (суффиксы, префиксы), где каждой из них ставится в соответствие более короткий код, заменяющий ее в словаре. Важнейшим является здесь алгоритм выбора «подходящих», т. е. достаточно длинных и часто встречающихся подцепочек. Аналогичная задача при сжатии произвольной текстовой информации рассмотрена ниже. Здесь же отметим, что для задач этого класса эффективного алгоритма поиска оптимального решения не существует [34], поэтому используются эвристические алгоритмы, опирающиеся на упорядоченность слов [73], [33], [34].

Когда составляющие словаря образуют сильно обособленные группы слов, можно разделить весь словарь на подслова-

ри, присвоив каждому из них свой индекс, и кодировать слова независимо в каждом из них кодами минимальной длины, а слова из различных подсловарей различать этими индексами [112]. Такой метод является модификацией описаного ранее метода кодирования числовых данных.

К специальным отнесем тексты на формальных языках, отличающиеся ограниченным словарем, замкнутой грамматикой, поддающиеся однозначному морфологическому и синтаксическому анализу. Сюда прежде всего относятся тексты на языках программирования, машинные коды, различные формулы и обозначения, например формулы химических соединений. К специальным текстам отнесем также ограниченное подмножество фраз естественного языка в таких четко формализованных задачах, как организация реплик в интерактивных системах, выдача сообщений при компиляции и т. п.

Для данного типа информации пригодны многие методы сжатия общей текстовой информации, о которых пойдет речь ниже. Некоторые способы сжатия текстов общего вида были разработаны и тестированы именно на специальных текстах [99, 100, 49, 24]. Наряду с этим ограниченность терминологии специальных текстов, их контекстно-свободная грамматика, а также специфика задач интерактивного обмена информацией в диалоговых системах и в системах искусственного интеллекта позволяют осуществить здесь весьма экономное кодирование [55, 43], основанное на выделении длинных часто повторяющихся фрагментов.

Под структурированными мы понимаем данные, содержащие текстовую и нетекстовую информацию и хранящиеся в определенном формате, приемлемом для тех или иных прикладных задач, например, для документального и фактографического поиска информации. Характерным примером структурированных данных являются библиографические описания. Проблеме сжатия информации в базах данных посвящено много работ [69], [82], [30], [2], [48], [87], [39], [85], [41], [70], [45], [75], [57], [90], [9], [36], [27], [5], [67], [107], [19], [109], [118].

Разнородность данных обуславливает различные типы информационной избыточности, поэтому любой универсальный метод сжатия, например, метод Хаффмена, работает хуже, чем комбинация методов, приспособленных к своим подгруппам данных. Для получения как можно большего сжатия целесообразно применять для числовых полей методы, описанные выше (разностное кодирование, кодирование повторений, подавление ведущих нулей), в совокупности с описываемыми далее методами сжатия текстовой информации. По некоторым оценкам [2], [81], комбинация этих методов дает сокращение объема данных в 1,5—4 раза, по другим оценкам — даже до 6 раз [6]. Достигнутые коэффициенты сжатия структурированных данных существенно различаются у различных авторов, что

связано, повидимому, с разнородностью данных [82], [19], [46], [47], [48], [66], [25], [90], [64].

Полезно отметить, что кодирование повторов и текстовое сжатие в базах данных предполагает отказ от формата записей фиксированной длины, что снижает удобство использования сжатой базы данных в информационно-поисковых задачах.

Сжатие характеризуется не только достигаемым коэффициентом сжатия, но и временем требуемых преобразований. С этой точки зрения использование простых и достаточно универсальных методов сжатия, не претендующих на достижение высокого коэффициента сжатия, может быть вполне оправдано [32], [44], [65], [81], [42], [35], [7], [8], [72], [28], [103].

В структурированных данных наряду с типами информационной избыточности, характерных для текстовых и нетекстовых данных, существует особый, позиционный тип избыточности [5]. Он связан с дублированием информации для идентификации структуры данных. Примером служат поля, не заполняемые при отсутствии сведений. Во избежание такой избыточности применяется индексация и подавление нулевых полей [75], [90], [5].

### § 3. Общие положения о сжатии текстов

Принципиальная возможность сжатия текстовой информации связана с тем, что составляющие текста — буквы и словоформы — сильно отличаются по частоте встречаемости в тексте, в то время как их длины слабо связаны с частотой. Имеется статистическая зависимость и между смежными элементами текста, т. е. появление одной буквы (слова) с большой вероятностью ведет к появлению другой буквы (слова). На языке теории информации Шеннона это означает, что энтропия текста (т. е. информационная емкость кодирующих текст символов) отлична от максимально возможной [91], [92]. Согласно эмпирическому закону Ципфа [113] наблюдается примерно обратная зависимость между частотой встречаемости слов в тексте и их рангами, т. е. номерами в порядке убывания частот. Неравномерность появления элементов текста имеет место не только на уровне букв и слов, но и на уровне частей слов (суффиксов, префиксов) или групп слов [92].

Многообразие информационной избыточности обусловило многообразие подходов к сжатию текстов на естественном языке и классификаций этих подходов.

Все алгоритмы сжатия можно классифицировать по используемому методу кодирования и характеру использования статистики и грамматики текста. Методы кодирования можно разделить на четыре класса [18], [116] в зависимости от того, какие группы символов кодируются, постоянной или переменной дли-

ны, и какие коды используются, постоянной или переменной длины.

По использованию статистики и грамматики алгоритмы сжатия можно разделить на семантически зависимые и семантически независимые [75], [19]. Семантически зависимые (лингвистические) методы опираются так или иначе на грамматику естественного языка для выделения в текстах элементов, подлежащих кодированию (как правило, это словоформы или аффиксы).

Семантически независимые методы сжатия в свою очередь можно разделить на те, которые используют, и те, которые не используют априорные сведения о статистике текста. В соответствии с этим существует два типа алгоритмов сжатия: однофазные и двухфазные, которые будем называть соответственно адаптивными и статистическими. Заметим, что термин «адаптивность» часто относят не только к однофазным алгоритмам, но и к описанным ниже алгоритмам автоматической генерации кодовых таблиц и сегментации текста.

Алгоритмы первого типа [94] не используют для сжатия никаких априорных сведений о статистике текста. Кодирование производится в процессе однократного сканирования текста. Оно сводится к замене цепочек символов текста на встроеные указатели, адресованные к той части текста, где такие цепочки уже встречались [80], [103], [114]. В этом случае говорят о внутренней адресации [94].

В алгоритмах второго типа [87], [48], [25] на первой фазе производится статистический анализ текста и составление кодовой таблицы, на второй — кодирование самого текста, т. е. сегментация текста на составляющие полученной таблицы и присвоение им соответствующих кодов. В этом случае говорят о внешней адресации.

Статистические алгоритмы можно в свою очередь классифицировать по тому, каков способ генерации кодовой таблицы, т. е. создается ли она заново для каждого текста и хранится вместе с его закодированным вариантом или является общей для целого класса текстов. В соответствии с этим будем говорить о локальных или глобальных кодовых таблицах.

## § 4. Конкретные алгоритмы сжатия текстов

**1. Адаптивные алгоритмы.** Адаптивные алгоритмы сжимают текст в процессе однократного его сканирования. Кодирование заключается в нахождении повторяющихся участков текста и замене каждого участка указателем, адресованным к той части текста, где такой участок уже встречался [114], [115], [80], [90], [103], [94]. Для декодирования в этом случае кодовой таблицы не требуется.

Существует два типа встроенных указателей, указывающих на предшествующие или последующие участки. Алгоритмы, использующие указатели назад, могут работать с непрерывным входным потоком данных, генерируя непрерывной выходной поток сжатой информации [114], [115]. На каждом шаге алгоритма входной текст заполняет буфер фиксированной длины, внутри которого производится идентификация одинаковых подстрок максимально возможной длины. При нахождении двух таких подстрок вторая заменяется указателем, адресованным в начало первой.

Алгоритмы с указателями вперед [99], [94] могут работать лишь с текстами конечной длины, поскольку требуют обратного сканирования текста. Здесь также используется поиск совпадающих подстрок в буфере переменной длины с уже закодированным текстом. Хотя алгоритмы первого типа могут оказаться для ряда задач более приемлемыми (например, для каналов связи), алгоритмы второго типа при опоре на оптимизирующие методы динамического программирования дают в целом лучшее сжатие.

Одной из характерных черт адаптивных алгоритмов является достаточная их универсальность, т. е. возможность работать с любыми, не только текстовыми данными, ненужность начальной информации о характере данных и их статистике. Эта черта снижает эффективность сжатия и достигаемое сжатие, как правило, меньше полученного другими методами [94], [103]. Но часто адаптивные алгоритмы просты и все же приемлемы по эффективности.

Теоретически показано, что не существует адаптивного алгоритма, производящего оптимальное сжатие текста с потреблением ресурсов времени и памяти, ограниченных хотя бы полиномиальной функцией от объема входных данных [94]. Поэтому применяются компромиссы между временем поиска решения и достигаемым сжатием.

Поскольку адаптивные алгоритмы не используют априорных сведений о статистике, они сравнительно медленны, а произведение времени кодирования на достигаемый коэффициент сжатия на текстовых данных у них в целом ниже, чем у статистических алгоритмов [90], [103]. Время кодирования и объем используемой памяти, как правило, зависят у них от длины входного текста. Это прежде всего относится к алгоритмам с обратным сканированием [100], [94].

С другой стороны, эффективность алгоритмов прямого сканирования зависит от некоторых внутренних параметров самого алгоритма [114], [115], [80], [103]. Одним из характерных параметров является так называемая длина адаптивности [103], которая характеризует «память» алгоритма к ранее введенной части текста. В итоге статистически неоднородные тексты сжимаются плохо.

Достижимое сжатие сильно зависит от машинной реализации алгоритма. Теоретические оценки [94] и экспериментальные данные [90, 103] показывают, что коэффициент сжатия текстовых данных этим методом лежит в пределах 1,8—2,5.

**2. Статистические алгоритмы.** Под обслуживающим систему сжатия словарем будем понимать таблицу, используемую алгоритмом сжатия для кодирования/декодирования текста. Словарь может быть локальным и глобальным [6]. Локальный словарь строится для каждого отдельного текста и хранится вместе с сжатым его вариантом. Глобальный составляется на основе нескольких эталонных текстов и служит для сжатия любого текста, по предположению обладающего статистическими свойствами эталонной группы.

Простейшей формой словаря является кодовая таблица символов алфавита, ставящая в соответствие каждому символу свой код. Коды выбираются с таким расчетом, чтобы общая длина закодированного ими текста была минимальной. Такую же таблицу можно составить для всех или наиболее часто встречающихся комбинаций из двух, трех и т. д. букв, т. е. фрагментов с фиксированным числом символов.

Другой формой словаря может являться словарь фрагментов переменной длины. Такой словарь может быть составлен на основе морфологического анализа текста или с помощью статистического алгоритма, выделяющего согласно некоторому критерию группы символов с наибольшей информационной избыточностью и помещающему их в словарь. В последнем случае фрагменты могут включать знаки пунктуации и не соответствовать словам естественного языка [86], [85]. Выбранным фрагментам присваиваются коды, которые затем заменяют их место в тексте. Коды опять выбираются для достижения максимального сжатия.

Вкратце рассмотрим методы, основанные на словарях фрагментов фиксированной длины и, несколько подробнее, методы, связанных со словарями фрагментов переменной длины.

**2.1. Кодирование фрагментов фиксированной длины.** Словари фрагментов фиксированной длины, состоят из отдельных символов, пар символов (биграмм), или, в общем случае,  $K$ -грамм. Используются два основанных на них вида кодов, фиксированной и переменной длины [19].

Простейшее кодирование кодами фиксированной длины состоит в уменьшении разрядности кода отдельных букв. Например, вместо общепринятого восьмибитового кода в тексте из латинских букв достаточно использовать шестибитовый код, уже дающий экономию на 25%. Если же ограничиться только строчными буквами, а для переходов к прописным буквам и цифрам использовать специальные переключающие символы, то можно использовать здесь и пятибитовый код, но эффективность его невелика.

Усовершенствовать метод можно, присоединив к обычным символам наиболее часто встречающиеся биграммы и произво-ив им неиспользуемые комбинации восьмибитовых кодов [37], [12], [25], [97], [107]. Этот метод прост, но позволяет сжать текст не более чем вдвое [90].

Существует другой подход, который не использует статисти-ческие свойства текста. При кодовой таблице из 40 составля-ющих (объем алфавита плюс несколько ходовых знаков),  $40 \cdot 40 \cdot 40$  комбинаций из трех таких составляющих могут быть закодированы 16 битами (вместо  $3 \cdot 8 = 24$ ) и помещены в одно-машинное слово микрокомпьютера. Максимальное количество символов в цепочке может быть таким, чтобы закодированное слово поместилось в машинном слове крупной ЭВМ [44], [25]. Такой способ кодирования получил название кода Бодо. Суще-ствуют различные варианты такого способа для числового кодирования в системах искусственного интеллекта [43], [18], [116].

Использование кодов переменной длины состоит в кодиро-вании более часто встречающихся символов более короткими битовыми комбинациями, а более редких — длинными. Коды должны обладать свойством префиксности, т. е. ни одна кодо-вая комбинация не должна быть начальной частью другой (разделители между кодами не используются) [20]. Наиболее известным и широко распространенным методом такого коди-рования остается метод Хаффмена. Он позволяет автоматиче-ски генерировать оптимальные префиксные коды по частотному распределению кодируемых символов. Коды Хаффмена являют-ся оптимальными в том смысле, что не существует способа посимвольного кодирования, дающего больший коэффициент сжатия [98], [74].

Поскольку частотное распределение букв алфавита мало зависит от текста, перекодировочная таблица для кодов Хаф-фмена может быть создана один раз и затем использована для кодирования любых текстов данного языка или иных данных с фиксированным частотным распределением символов. Для дан-ных, подверженных сильным статистическим изменениям, при-менением адаптивный алгоритм Хаффмена [31], [35], который позволяет автоматически перестраивать кодovou таблицу при изменении частот символов. Коэффициенты сжатия с помощью посимвольного кодирования Хаффмена относительно невелики и, как правило, не превосходят двух [90], лишь иногда достигая трех [82], [64].

Как уже отмечалось, избыточность информации заключает-ся еще в корреляции между символами (словами). Метод Хаф-фмена сохраняет эту избыточность. Существуют модификации метода, позволяющие учесть взаимозависимости. Наиболее про-стая из них [7], [8] используется, когда все символы можно раз-делить на небольшое число групп с сильной корреляцией вну-

три групп и слабой корреляцией между ними. Это иногда имеет место для числовых и буквенных символов текста.

Другой способ учета корреляций основан на модели текста в виде марковской последовательности символов [121], [95]. Генерация оптимальных кодов для любой цепочки из  $K$  символов требует знания вероятности появления любого заданного символа в качестве  $(K+1)$ -го. Вычисление этих вероятностей весьма трудоемкая задача. Поэтому результаты приведены лишь для  $K=2$ , а коэффициенты сжатия лишь чуть превосходят 2. Решить задачу сжатия с эффективным учетом произвольных межсимвольных корреляций на основе такого метода пока не удалось.

К другим недостаткам хаффменовских методов относится относительная сложность декодирования — необходимость анализа битовой структуры префиксных кодов, замедляющая процесс декодирования. Здесь существуют довольно эффективные по скорости реализации алгоритма [72], [51], но с малой надежностью. Дело в том, что при неправильном прочтении хотя бы одного бита в цепочке префиксных кодов искажается вся последующая информация («потеря синхронизации»). Во избежание этого необходимо конструировать самосинхронизирующиеся префиксные коды [104], [11], но они снижают сжатие.

Метод Хаффмена позволяет строить и целобайтовые коды (с длиной, кратной байту), что удобнее для современных ЭВМ. Но такие коды эффективны только для словарей достаточно больших размеров, содержащих не только буквы алфавита, но и более длинные фрагменты текста, о чем речь пойдет позднее.

Дальнейшим развитием метода Хаффмена являются арифметические коды [83], [29], [76], [77], [78], [79], [54], [56], [68], [116]. Они происходят из так называемых конкатенационных, или блочных кодов. Суть последних заключается в том, что выходной код генерируется для цепочки входных символов фиксированной длины без учета несимвольных корреляций. В основе метода лежит представление каждой цепочки  $K$  входных символов  $(A_1, \dots, A_k)$  в виде числа, получаемого как сумма  $K$  слагаемых вида

$$p(A_1)p(A_2) \dots p(A_{i-1})P(A_i), \quad i=1 \dots k,$$

где  $p(S)$  — вероятность символа  $S$ , а  $P(S)$  — кумулятивная вероятность символа  $S$ , равная сумме вероятностей всех символов  $A_i$ , для которых  $p(A_i)$  больше  $p(S)$ . Чем больше символов в цепочке, тем большая разрядность числа необходима для ее кодирования. Давая некое улучшение по сравнению с кодированием одиночных символов, конкатенационные коды не могут существенно повысить энтропию выходного текста из-за неучета межсимвольных корреляций.

В работах Колмогорова [122] и Риссанена [76—79] арифметические коды получили развитие в виде теории универсаль-

ного кодирования, в которой ограничение на статистическую независимость символов снимается. Кодирование также происходит поблочно, но код для каждого блока (цепочки) находится на основе выбора в некотором классе статистических моделей текста, тип которого наиболее подходит для данного блока, и подстановки соответствующего кода из этой модели. Выбор моделей дает возможность учесть взаимозависимость символов в тексте, но ведет к необходимости кодирования самих моделей, что при большом их количестве снижает эффективность кодов. Построение самих моделей относится пока к области теории. Практическое приложение нашли лишь конкатенационные коды, основанные на статистике отдельных символов.

Учет межсимвольных корреляций приводит к необходимости создания и поддержания словаря, во много раз превышающего размеры кодовой таблицы посимвольного кодирования. Однако их использование позволяет достигать больших коэффициентов сжатия, чем описанные выше методы.

**2.2. Кодирование фрагментов переменной длины.** На рубеже 60—70-х годов интерес к проблеме сжатия текстовой информации возрос, и особенно для информационно-поисковых применений [89], [117], [71], [93], [61], [48], [53], [96]. Очевидно кодируя группы символов в виде фрагментов переменной длины и используя словари фрагментов, можно получить более высокие коэффициенты сжатия по сравнению с посимвольным кодированием.

Словари фрагментов переменной длины можно условно разделить на словари вокабул и словари произвольных фрагментов. Такое деление отражает два подхода к составлению словаря, лингвистический и статистический. Элементами словаря вокабул являются естественные лексические единицы — словоформы или морфемы (как правило, основы). Такие единицы выделяются морфологическим анализом текста. Словари произвольных фрагментов составляются на основе статистического анализа текста, согласно неким критериям выделяющего из текста фрагменты, кодирование которых дает максимальное сжатие.

Словари, получаемые лингвистическими и статистическими методами, отличаются и наполнением и частотами своих составляющих. В первом случае распределение частот по рангам приближенно подчиняется закону Ципфа, во втором случае зависит от используемого алгоритма и критериев отбора слов. Наиболее выгоден с точки зрения сжатия был бы словарь с равномерным распределением составляющих, когда оптимальны коды фиксированной длины и достигаемый коэффициент сжатия близок к максимально возможному [59]. Для достижения максимального сжатия в случае словаря вокабул необходимо пользоваться кодами переменной длины, причем частота обращения к словарю при кодировании/декодировании возрастает [48], [25].

Еще одним важным преимуществом равной частотности является возможность контроля размеров словаря [87], [48], [59]. Количество фрагментов при построении словаря обычно берется в качестве исходного параметра. Можно разместить весь словарь в памяти быстрого доступа и увеличить скорость кодирования/декодирования. При кодировании же вокабул ситуация иная. Обычно размеры словаря достаточно велики (до десяти и более тысяч) и в общем случае не постоянны. Как показали исследования [62], [86], объем словаря неуклонно возрастает с увеличением размеров текста, на основании которого словарь был составлен, согласно степенной функции с показателем, меньшим единицы. Иными словами, в естественноречевом тексте вероятность встретить незнакомое слово всегда отлична от нуля.

Перечисленные преимущества равночастотных словарей направили усилия многих исследователей на поиск алгоритмов, выделяющих из текста равновероятные фрагменты [21], [86], [87], [85], [48], [59], [25]. Однако эта задача вычислительно сложна, и лингвистический подход остается разумной альтернативой, давая пока и большее сжатие [25].

Алгоритмы сжатия, основанные на использовании словарей вокабул, двухфазны. Как первая, так и вторая фаза опираются на морфологический анализ текста. Поэтому подходы для высоко- и низкофлективных языков различаются количественно и качественно.

Простейшей формой автоматической генерации словаря словоформ является составление перечня всех словоформ текста с указанием для каждой числа повторения. Слова в тексте вычленяются по естественным разделителям — пробелам, знакам пунктуации. Однако все словоформы одной и той же лексемы рассматриваются как независимые составляющие, что ведет к заметному росту словаря. Если же опираться на морфологический анализ, то неоправданно усложняется составление словаря. Поэтому обычно используются уже имеющиеся словари, из которых по некоторым критериям отбираются слова для кодовой таблицы [102]. Образованные таким способом машинные варианты содержат, как правило, аффиксы [105]. Хотя объем словаря зависит от размеров текста, этот объем обычно искусственно ограничивают для повышения быстродействия и экономии памяти. Поэтому окончательный машинный словарь для сжатия, как правило, меньше исходного варианта.

Упомянутые критерии отбора вычисляют для каждого элемента словаря тот коэффициент сжатия, который может быть достигнут за счет перекодирования. Обычно этот критерий задается разностью между количеством бит, занимаемых словом в тексте до и после кодирования. Таким образом каждому слову присваивается свой вес, и отбираются слова с наибольшим весом [102].

Глобальный коэффициент сжатия зависит от размеров словаря. При сжатии английского текста со словарем в 1024 единиц достигается коэффициент, равный 2 [102], в то время как при использовании более емких словарей можно достичь 3—4 [25].

Словарь может быть как статическим, так и динамическим. В первом случае однажды выбранные составляющие сохраняются неизменными, во втором возможно периодическое обновление словарного состава, в том числе слияние отдельных лексем, совместное появление которых в тексте высоковероятно. Для этой цели могут использоваться специальные алгоритмы, производящие автоматический анализ статистики словарных составляющих и их сочетаний [105].

Кодами при сжатии служат битовые комбинации переменной или постоянной длины. Более удобны коды постоянной длины, дающие простую идентификацию кода в тексте и адресацию к кодовой таблице. На практике используются оба метода кодирования [10].

При кодировании кодами фиксированной длины используются, как правило, двухбайтовые коды (16 бит). Первые несколько бит из шестнадцати служат идентификатором кода, позволяющим отличить код словарной составляющей от кодов обычных символов текста (предполагается, что не все элементы текста могут быть закодированы с помощью словаря). Остальные биты являются адресом в кодовой таблице. Такое кодирование удобно для информационно-поисковых систем, но дает небольшой коэффициент сжатия (около двух) [1]. Часть битов может также выделяться для идентификации словоформы, производной от данной основы [102, 6]. Заметим, что в низкофлективных языках вроде английского для идентификации конкретной словоформы иногда бывает достаточно указать ее окончание, для чего хватает трех бит кода, но в высокофлективных языках такая возможность сомнительна.

При кодировании кодами переменной длины наиболее эффективен с точки зрения сжатия информации метод Хаффмена или его модификации [31], [35], [7], [8]. Его применение в принципе обещает наибольшее сжатие, но многое здесь зависит от качества словаря. На практике этот метод не нашел широкого применения из-за сложности процесса декодирования, в котором для идентификации кодовой комбинации используется поиск по двоичному дереву [25], [72].

Получили распространение и целобайтовые коды переменной длины [10], [46], [48]. Как и для иных кодов переменной длины, в этом случае необходим механизм идентификации начала и конца кода. Такая идентификация осуществляется по префиксу или по начальному биту. В первом случае длину показывают первые несколько бит каждого кода. Поскольку максимальная длина кодов не превосходит трех — четырех байт, бывает дос-

таточно двух бит. Комбинация оставшихся битов используется для адресации перекодировочной таблицы. В альтернативном варианте начальный бит каждого байта обозначает, является ли этот байт первым байтом нового кода или очередным байтом старого. В случае префиксной идентификации коды более экономны, а идентификация по начальному биту дает большую устойчивость кодов к помехам.

Задача сегментации не вызывает трудностей, если словарь содержит все возможные словоформы. Но для высокофлективных языков такой метод весьма непрактичен. Поэтому словарь, как правило, содержит части слов — основы, префиксы, окончания. В этом случае для установления соответствия между текстом и словарем необходим морфологический анализ. Однако существующие морфологические анализаторы не нашли пока широкого применения для сжатия информации из-за наличия более эффективных и простых алгоритмов сегментации текста, о которых еще пойдет речь.

В основу создания словарей произвольных фрагментов положена идея выявления набора символьных цепочек с максимальной энтропией [86], [87], [60], [59], [25]. Выявленная цепочка (фрагмент) не обязательно совпадает со словом в обычном понимании.

Впервые предложенная в 1969 году [22], идея оказалась весьма плодотворной в автоматизированных информационно-поисковых системах [3], [48], [53], [19]. Это связано с небольшим и легко контролируемым размером словаря, фиксированной длиной кодов, языковой независимостью. Словарь фрагментов обладает тем преимуществом, что стабилен по отношению к изменяющейся статистике текста и требует в среднем меньше обращений при информационном поиске [58], [4], [14], [23], [25]. Однако реализация равночастотного словаря фрагментов трудна. Наряду с равной вероятностью и статистической независимостью необходимо, чтобы выбранные фрагменты обладали еще свойством полноты, т. е. любая часть текста была представлена конкатенацией выбранных фрагментов. Практически это означает, что набор фрагментов должен включать все буквы алфавита, но это нарушает равномерность.

Многие алгоритмы автоматической генерации словаря используют предварительный статистический анализ  $K$ -грамм [101]. Целью является построение распределения  $K$ -грамм по частотам. Обычно это осуществляется так. задается значение длины цепочки  $K$ . Затем текст сканируется от первого символа до  $(M-K)$ -го, и на каждом  $P$ -м шаге сканирования в список  $K$ -грамм заносится цепочка символов текста от  $P$ -го до  $(P+K)$ -го. Если в списке уже присутствует такая комбинация символов, то значение ее частоты увеличивается на единицу.

После того как список  $K$ -грамм создан, из него выделяется подгруппа как можно более равновероятных  $K$ -грамм, обладаю-

щих свойством полноты. Они и включаются в словарь. Для этого обычно задают частотный порог, и, просматривая по определенному алгоритму  $K$ -граммы различной длины, находят наиболее близко лежащие к этому порогу сверху или снизу [21], [52], [86], [3], [59]. Так как  $K$ -граммы не являются статистически независимыми, то при включении в словарь каждой новой  $K$ -граммы производят пересчет частот остальных  $K$ -грамм, что довольно трудоемко [86], [66]. Генерируемые этими алгоритмами словари все же плохо удовлетворяют критериям равновероятности [106].

Существует также итеративный подход к генерации словаря фрагментов [25], [109], [106], [81]. Предварительный набор статистики  $K$ -грамм здесь не производится. Задается лишь первоначальный словарь фрагментов, как правило, в виде обычного алфавита, и частотный порог. Затем на каждом шаге алгоритма производится выявление наиболее часто встречающихся в тексте пар фрагментов, и, если частота такой пары превышает порог, два эти фрагмента сливаются и включаются в словарь. После этого производится перекодирование текста и пересчет частот фрагментов. Если частота фрагмента при таком пересчете становится слишком малой, фрагмент может быть удален из словаря, но условие полноты при этом нарушаться не должно. Очевидным недостатком этого подхода является необходимость многократного сканирования и перекодирования текста.

Получить в строгом смысле равновероятный и полный набор фрагментов оказалось в общем случае невозможным. Более того, несмотря на значительные усилия различных исследователей [21], [52], [86]; [3], [59], [106], генерируемые до сих пор наборы фрагментов далеки от равновероятных. Поэтому вместо принципа равновероятности часто опираются на принцип максимального сжатия [110], [109], [81]. Речь идет о достижении максимального коэффициента сжатия по совокупности фрагментов, а не по каждому в отдельности. Для поиска решения применяют динамическое программирование. Как показали, однако, исследования [33], [34], задача создания не только словаря фрагментов, но и словаря аффиксов не поддается эффективной алгоритмизации. Это означает, что не существует алгоритма, позволяющего найти оптимальный (в смысле максимального коэффициента сжатия) набор фрагментов, который расходует ресурсы памяти и времени, ограниченные хотя бы полиномиальной функцией от размера входного текста.

В связи с этим в большинстве случаев используются эвристические алгоритмы, представляющие собой компромисс между оптимальным решением и требуемым быстродействием [63], [66], [65], [118], [109], [81]. Как правило, эти алгоритмы итеративны и основаны либо на повторном сканировании текста для выявления наиболее часто встречающихся фрагментов (пар фрагментов) [81], [110], либо на работе с отсортирован-

ными списками фрагментов и пересчете частот по мере изменения списка [66]. В последнем случае алгоритм несколько выигрывает в быстродействии, но требует расхода большей памяти.

Наряду с итеративными алгоритмами существует подход, использующий для составления словаря однократное сканирование текста [65]. Он заимствует идеологию адаптивных алгоритмов. На каждом шаге производится поиск фрагмента наибольшей длины из словаря фрагментов, совпадающего со входной подстрокой. Если такой фрагмент найден, то анализируется возможность его слияния с фрагментом, найденным на предыдущем шаге. В результате либо в словарь включается новый составной фрагмент, либо частота найденного ранее фрагмента увеличивается на единицу. По мере заполнения словаря происходит его периодическая чистка, в процессе которого удаляются фрагменты с частотой, меньшей установленного порога. Наряду с относительной простотой и хорошим быстродействием алгоритму присущи черты адаптивных алгоритмов, такие как ограниченная «память» к ранее введенному тексту и множество настроечных параметров, существенно влияющих на эффективность.

В ряде случаев и здесь полезно применять коды переменной длины (например, коды Хаффмена). Это тем более оправдано, чем сильнее отличие распределения частоты фрагментов от равномерного.

В условиях контроля размера словаря фрагментов обычно используются небольшие словари — не более тысячи составляющих. Это позволяет размещать словарь в оперативной памяти ЭВМ для ускорения процесса кодирования/декодирования, уменьшать затраты на построение словаря и добиться его достаточной универсальности. Как показывают исследования [14], [25], словарь размером более 512 фрагментов оказывается тематически зависимым. В то же время словарь из 256 фрагментов достаточно стабилен по отношению к текстам различной тематики и позволяет быстро осуществлять двухкратное сжатие текста [87], [48], [23], [25].

**2.3. Сегментация текста.** Сегментация текста при использовании словаря фрагментов и кодов фиксированной длины заключается в разбиении текста на минимально возможное число непересекающихся подстрок. При таком разбиении текст имеет минимальные размеры после подстановки кодов вместо фрагментов. Но в общем случае сегментация текста может быть произведена неоднозначно, так как различные фрагменты, представляющие собой цепочки символов, пересекаются внутри текста. Поэтому возникает задача оптимального разбиения текста. Показано, что эта задача сводится к известной из динамического программирования задаче поиска кратчайшего расстояния [99], [94].

В одном из вариантов алгоритма решения этой задачи используется обратное сканирование текста [99]. На каждом шаге алгоритм продвигается назад по тексту, от  $P$ -го символа к  $(P-1)$ -му. При этом весь текст справа от  $P$ -го символа представлен в закодированной, а слева — в первоначальной форме. Из всех фрагментов словаря, совпадающих с отрезком текста, начинающимся с  $P$ -го символа, выбирается такой, замена которого на соответствующий код дает минимальную длину правой части текста. Этот фрагмент назовем локально оптимальным, а значение минимальной длины правой части текста, соответствующее  $P$ -му символу, назовем  $P$ -м значением функции оптимизации. На каждом шаге алгоритма для нахождения оптимального фрагмента необходимо знать по крайней мере  $M$  значений функции оптимизации, соответствующих  $(P+1)$ -й,  $(P+2)$ -й, ..., ...,  $(P+M)$ -й позициям исходного текста, где  $M$  — количество символов в наиболее длинном фрагменте словаря. Достижимый при этом коэффициент сжатия будет определяться отношением первоначальной длины текста к значению функции оптимизации при  $P=1$ .

Поиск оптимальной сегментации текста является задачей экспоненциальной сложности и связан с большими затратами ресурсов памяти и времени [94], [34]. Поэтому пользуются вниманием эвристические алгоритмы, дающие в общем случае не оптимальное решение, но расходующие меньше ресурсов [87], [110], [25]. Эвристические алгоритмы мало уступают по коэффициентам сжатия алгоритмам, ищущим оптимальное решение [87].

Простейшей и наиболее популярной формой такого алгоритма является алгоритм с прямым сканированием текста [87]. На каждом шаге из словаря фрагментов выбирается наиболее длинный фрагмент, совпадающий с первыми символами входной цепочки. После этого часть входной цепочки, совпадающая с фрагментом, заменяется на код фрагмента, а алгоритм продвигается вперед по тексту на длину фрагмента.

Более быстрый способ сегментации основан на использовании частот и взаимных корреляций фрагментов текста [109]. Строится направленное дерево, каждый узел которого соответствует одному фрагменту словаря. На первом уровне располагаются все символы алфавита в порядке убывания их частот. Фрагменты, соответствующие узлам каждого данного уровня, являются конкатенацией фрагментов, соответствующих узлам следующего уровня. Каждый узел соединен ветвями с теми узлами следующего уровня, фрагменты которых содержат в себе фрагмент данного узла. При этом каждой ветви приписывается порядковый номер, определяемый условной вероятностью появления в тексте фрагмента более высокого уровня при наличии в нем фрагмента более низкого уровня. Эти условные вероятности рассчитываются при составлении сло-

варя фрагментов и используются для направленного поиска по графу в процессе сегментации текста.

Сам поиск происходит следующим образом. На каждом шаге идентифицируется фрагмент первого уровня, совпадающий с первым символом (или символами, если первый уровень содержит конкатенации фрагментов) входного текста. Это осуществляется просмотром первого уровня в порядке убывания частот фрагментов. Затем в порядке убывания номеров ветвей, ведущих от найденного фрагмента первого уровня к фрагментам второго уровня, идентифицируется фрагмент второго уровня (если такой имеется), совпадающий с первыми символами входного участка текста. Продвижение по дереву продолжается до тех пор, пока на очередном уровне не будет зафиксировано отсутствие фрагмента, полностью совпадающего с первыми символами входного участка, или не будет достигнута вершина дерева. После этого код последнего найденного фрагмента подставляется на место первых символов входного участка, совпадающих с фрагментом, и алгоритм продвигается вперед по тексту на длину фрагмента. Быстродействие этого алгоритма обусловлено использованием статистических свойств фрагментов текста при поиске по графу.

## § 5. Некоторые сопоставления и перспективы

До сих пор рассматривались вопросы, связанные со сжатием текстов. Однако в приложениях чаще приходится иметь дело с обратной операцией. Декодирование текстов необходимо не только при выдаче текста потребителю, но и при автоматическом поиске по запросу. Последнее обстоятельство связано с невозможностью непосредственного поиска информации в закодированном тексте. Поэтому скорость декодирования оказывается при сопоставлении различных методов сжатия более критическим показателем.

Декодирование осуществляется наиболее просто и быстро в случае кодов фиксированной длины со встроенными указателями, используемыми в адаптивных алгоритмах и алгоритмах, основанных на словарях произвольных фрагментов [10], [94], [6]. В этом случае код непосредственно используется для адресации в кодовой таблице.

Более сложно обстоит дело с кодами переменной длины. Здесь адресация не может быть произведена напрямую, поэтому декодирование идет медленнее. Так, для метода Хаффмена соотношение скоростей кодирования и декодирования колеблется около 2,5 : 1, в то время как для методов с использованием словаря фрагментов это отношение составляет 1 : 3 [25]. Методы, основанные на использовании целобайтовых префиксных кодов [10], [46], [48], занимают промежуточное положение.

При использовании глобальных словарей небольших размеров применение описанных эвристических алгоритмов сегментации, наряду с ускоренными методами идентификации фрагментов [6], могут существенно повысить скорость декодирования.

Если расположить рассмотренные методы сжатия текстовой информации в порядке возрастания достигаемого сжатия, то сначала идут методы, опирающиеся на словари фрагментов постоянной длины, затем методы со словарями произвольных фрагментов и, наконец, методы со словарями вокабул. В последних двух случаях коэффициент сжатия будет, правда, сильно зависеть от типа словаря. Адаптивные алгоритмы занимают особое место. Их эффективность во многом определяется машинной реализацией и настроечными параметрами алгоритма, но в целом они уступают случаям словарей фрагментов переменной длины.

Если оценивать алгоритмы сжатия по критерию быстродействия, то необходимо отдельно учитывать скорости кодирования и декодирования. Методы с глобальными словарями дают значительно большую скорость кодирования, чем методы с локальными словарями или адаптивные. С другой стороны, методы, использующие коды фиксированной длины, дают большую скорость декодирования. Если скорость кодирования и декодирования одинаково важны, как это имеет место в системах связи, то, по-видимому, наиболее целесообразны словари небольших размеров и коды фиксированной длины.

Наиболее сложны в реализации многоэтапные методы, опирающиеся на предварительный статистический анализ текста и создание словарей, в первую очередь глобальных. Однако по завершении создания словаря они дают большой выигрыш. Для достижения большого сжатия можно применять комбинации различных методов.

Наш анализ свидетельствует, что при всей обширности литературы по проблеме сохраняется актуальность поиска новых способов кодирования и развития статистической теории сжатия, учитывающей сложности алгоритмов кодирования/декодирования, стоимость хранения словарей и устойчивость (робастность) к изменениям статистики ансамбля кодируемых сообщений.

## ЛИТЕРАТУРА

1. Aïnon R. N., Storing text using integer codes. Proc. of 11-th Int. Conf. on Comput. Ling. COLING'86, 25—29 Aug. 1986. Bonn, 1986, 418—420
2. Alsberg P. A., Space and time savings through large data base compression and dynamic restructuring. Proc. IEEE, 1975, 63, № 8, 1114—1112
3. Barton I. J., Creasy S. E., Lynch M. F., Snell M. J., An information theoretic approach to text searching in direct access systems. Commun. ACM, 1974, 17, 345—350
4. Barton I. J., Lynch M. P., Petrie J. H., Snell M. J., Variable—length character string analysis of three databases and their application for file

- compression. In: *Informatisc I. Proc. 1-st Informatics Conf. Durham Univ.*, 1973, London Aslib, 1974, 154—162
5. *Bassiouni M. A.* Data compression in scientific and statistical databases. *IEEE Trans. on Software Engineering*, 1985, *11*, № 10, 1047—1058 (PJKMar, 1986, 6Г543)
  6. *Bassiouni M. A., Ok B.* Double encoding — a technique for reducing storage requirements of text. *Inform. systems*, 1986, *11*, № 2, 117—184
  7. *Bassiouni M. A., Hazboun K. A.* A multi—group technique for data compression. *Proc. ASM SIGMOD Int. Conf. on Manag. of Data*, 1982, 284—292
  8. *Bassiouni M. A., Hazboun K. A.* Utilization of character reference locality for efficient storage of databases. *Proc. 2-nd Int. Workshop on Statistical Database Management, Los Altos CA*, 1983, 338—334
  9. *Batory D.* Index Coding: a compression technique for large statistical databases. *Proc. 2-nd Int. Workshop on Stat. Database Manag.*, 1983, 306—314
  10. *Bemer R. W.* Do it by the numbers — digital shorthand. *Commun. ACM*, 1960, *3*, № 10, 530—536
  11. *Bobrov L. S., Hakimi S. L.* Graph theoretic prefix codes and their synchronizing properties. *Inform. & Control*, 1969, *16*, № 1, 70—94 (PJKMar, 1970, 5B323)
  12. *Bookstein A., Faulty G.* A mathematical model for estimating the effectiveness of bigram coding. *Inform. Processing & Manag.*, 1976, *12*, № 2, 111—116
  13. *Booth A. D.* A «law» of occurrences of words of low frequency. *Inform. & Control*, 1967, *10*, 386—393 (PJKMar, 1968, 12B104)
  14. *Brack E. V., Cooper D., Lynch M. F.* The stability of symbol sets produced by variety generation from bibliographic data. *Program*, 1978, *12*, № 2, 64—77
  15. *Bradley S. D.* Optimizing a scheme for run—length encoding. *Proc. IEEE (letters)*, 1969, *57*, № 1, 108—109
  16. *Bray J. M., Nelson V. P., deMain B. A. D., Irwin J. D.* Data compression techniques ease storage problems. *Comput. Design*, 1985, *24*, № 14, 102—106
  17. *Chen T. Ch., Ho I. T.* Storage efficient representation of decimal data. *Commun ACM*, 1975, *18*, № 1, 49—52
  18. *Choros K., Majewska P., Sieminski A.* Bibliographic data base compression. *Int. Forum on Information and Documentation*, 1987, *12*, № 1, 28—32
  19. *Choueka Y., Fraenkel A. S., Perl Y.* Polynomial construction of optimal prefix tables for text compression. *Proc. 19-th Annual Allerton Conf. on Commun., Control and Computing*, 1981, 762—768
  20. *Clare A. C., Cook E. M., Lynch M. F.* The identification of variable—length equiprevalent character strings in natural language data base. *Computer J.*, 1972, *15*, № 3, 259—262 (PJKMar, 1973, 4B653)
  21. *Colombo D. S., Rush J. E.* Use of word fragments in computer based retrieval systems. *J. of Chemical Documentation*, 1969, *9*, № 1, 47—50
  22. *Conuel J. B.* A Huffman—Shannon—Fano code. *Proc. of IEEE*, 1973, 1046—1047
  23. *Cooper D., Emily M. A., Lynch M. F., Yeates A. R.* Compression of continuous prose texts using variety generation. *J. of ASIS*, 1980, *31*, № 3, 201—207
  24. —, *Lynch M. F.* Compression of Wiswester Line Notation using variety generation. *J. of Chemical Inform. and Computer Sciences*, 1979, *19*, № 3, 165—169
  25. —, —, Text compression using variable — to — fixed length encodings. *J. of ASIS*, 1982, *33*, № 1, 18—31
  26. —, The storage problem. *Mech. Translat.*, 1958, *5*, № 2, 74—83
  27. *Cormack G. V.* Data compression on a database system. *Commun ACM*, 1985, *28*, № 12, 1336—1342
  28. *Cortesi D.* An effective text compression algorithm. *Byte*, 1982, *7*, № 1, 397—403

29. Cover T. M., Enumerative source coding. IEEE Trans. Inform. Theory, 1973, 19, 73
30. Creasy S. E., Lynch M. F., Petrie J. H., Compression of bibliographic data—base using a variable to fixed—length bit string transformations. Program, 1974, 8, № 4, 191—195
31. Faller N., An adaptive system for data compression. Record of the 7-th Assilomar Conf. on Circuits, Systems and Computers, 1973, 593—597
32. Fano R. M., Transmission of information. MIT Press & Wiley, 1961, (PJKMar, 1962, 9B256K)
33. Fraenkel A. S., Mor M., Combinatorial compression and partitioning of large dictionaries. Computer J., 1983, 26, № 4, 336—343
34. —, Mor M., Perl Y., Is the compression by prefixes and suffixes practical? Acta Inform., 1983, 20, № 4, 371—379 (PJKMar, 1984, 8Г523)
35. Gallager R. G., Variations on a theme by Huffman. IEEE Trans. on Inform. Theory, 1978, 24, № 6, 668—674 (PJKMar, 1979, 8B576)
36. Gey F., McCarthy F., Merril D., Holmes H., Computer independent data compression for large statistical data—bases. Proc. 2-nd Int. Workshop on Stat. Database Manag., 1983, 296—305
37. Gilbert E. N., Moore E. F., Variable length binary encoding. Bell Syst. Tech. J., 1959, 38, 913—967 (PJKMar, 1961, 10B161)
38. Golomb S. W., Run—length encoding. IEEE Trans. on Inform. Theory, 1966, 12, 399—401 (PJKMar, 1967, 3B279)
39. Gottlieb D., Hagerth S. A., Lehot P. G. H., Rabinowitz H. S., A classification of compression methods and their usefulness for a large data processing center. Proc. of Nat. Comput. Conf., 1975, Montville N. J.: AFIPS Press, 1975, 44, 453—458
40. Goyal P., Coding methods for text string search on compressed databases. Inform. Systems, 1983, 8, № 3, 231—233
41. Grooms D. W., Data compression. Springfield, VA: NTIS, 1977
42. Guazzo M., A general minimum redundancy source coding algorithms. IEEE Trans. on Inform. Theory, 1980, 26, № 15, 15—25
43. Hagamen W. D., et al., Encoding verbal information as unique numbers. IBM Systems J., 1972, 11, № 4, 278—315
44. Hahn B., A new technique for compression and storage of data. Commun. of ACM, 1974, 17, № 8, 434—436
45. Hallman G., How to squeeze data into smaller spaces. Canadian Datatypes, 1978, 10, № 9, 27—29
46. Heaps H. S., Storage analysis of compression coding for document data bases. INFOR J., 1972, 10, № 1, 47—61
47. —, Data compression for large document data base. Conf. on Large Data Bases Sponsored by NAS/NRC, Committee on Chemical Inform., Nat. Acad. of Sci., 1974
48. —, Data compression of large document data bases. J. of Chemical Inform. & Comput. Sciences, 1975, 15, № 1, 32—39
49. —, Radhakrishnan T., Compaction of diagnosis messages for compilers. Software Practice & Experience, 1977, 7, 139
50. Huffman D., A method for the construction of minimum redundancy codes. Proc. IRE, 1952, 40, 1098—1101
51. Jakobsson M., Huffman coding in bit—vector compression. Inform. Processing Letters, 1978, 7, № 6, 304—307 (PJKMar, 1979, 4B743)
52. Jelinek J., Schneider K. S., On variable—length—to—block coding. IEEE Trans. on Inform. Theory, 1972, 18, № 6, 765—774 (PJKMar, 1973, 6B473)
53. Jewell G. C., Text compaction for information retrieval systems. IEEE Systems, Man, and Cybernetics Society Newsletters, 1976, 6, 4—7
54. Johnes C. B., An efficient coding system for long source sequences. IEEE Trans. Inform. Theory, 1981, 27, 280—291
55. Katajainen J., Penttonen M., Teuhola J. Syntax—directed compression of program files. Software Practice & Experience, 1986, 16, № 3, 269—276

56. *Langdon G. G., Rissanen J.*, A double adaptive file compression algorithm. IEEE Trans. on Commun., 1983, 31, 1253—1255
57. *Lynch C. A., Brownrigg E. B.*, Application of data compression technique to large bibliographic databases. Proc. of the 7-th Int. Conf. on Very Large Data Bases, 1981, 435—447 (PJKMar, 1982, 8B1052)
58. *Lynch M. F., Peirie J. H., Snell M. J.*, Analysis of the microstructure of titles in the INSPEC data—base. Inform. Storage & Retrieval, 1973, 9, № 6, 331—337
59. —, Variety generation—a reinterpretation of Shannon's mathematical theory of communication, and its implications for information science. J. of ASIS, 1977, 28, № 1, 19—25
60. —, *Rawson S. D.*, Equipfrequent character strings—a novel text characterization method. Proc. of the Third Int. Symp. on the Use of Computers in Linguistics and Linguistic Research, Cardiff 1974, Univ. Wales Press, 1976, 47—58
61. *Maggs P. B.*, Compression of legal texts for more economical storage. Jurimetrics J., 1974, 14, 254—261
62. *Mandelbrot B. M.*, On the theory of word frequencies and on related markovian models of discourse. Am. Math. Soc. Symp. Appl. Math., 1960, 72, 190—219 (PJKMar, 1963, 4B557)
63. *Marron B. A., de Maine P. A. D.*, Automatic data compression. Commun. ACM, 1967, 10, № 11, 711—715
64. *Martin J.*, Data compaction in computer data base organization. Prent Hall, Engl. Clifs, N. J., 2-nd Edn., 1977, 572—587
65. *Mayne A., James E. B.*, Information compression by factorizing common strings. Computer J., 1975, 17, № 2, 157—160
66. *McCarthy J. P.*, Automatic file compression. Int. Computing Symp., Davos 1973, North Holland, Amsterdam, 1974, 511—516 (PJKMar, 1975, 7B836)
67. *McIntyre D. R., Pechura M. A.*, Data compression using static Huffman code—decode tables. Commun. ACM, 1985, 28, № 6, 612—616 (PJKMar, 1986, 4Г429)
68. *Miller V. S.*, Data compression algorithms. Proc. Symp. Appl. Math., 1986, 34, 107—118
69. *Mulford I. E., Ridall R. K.*, Data compression technique for economic proceeding of large commercial files. ACM Symp. on Inform. Storage and Retrieval, 1971, Colledge Park, MD: ACM, 1971, 207—215
70. *Nevelainen O., Jacobson M., Berg R.*, Compression of clustered inverted files. In: Mathematical Foundation of Computer Sci. Berlin, Springer, 1978, 393—402 (PJKMar, 1979, 4B843)
71. *Nugent W. R.*, Compression word coding techniques for information retrieval. J. of Library Automation, 1968, 1, № 4, 250—260
72. *Pechura M.*, File archival techniques using data compression. Commun. ACM, 1982, 25, № 9, 605—609 (PJKMar, 1983, 3B1000)
73. *Radhakrishnan T.*, Selection of prefix and postfix word fragments for data compression. Inform. Processing & Manag., 1978, 14, № 2, 97—106
74. *Ramamoorthy C.*, Document compaction by variable length encoding. Proc. of ASIS, 1964, 1, 507—513 (PJKMar, 1964, 2B387)
75. *Reghbati H. K.*, An overview of data compression techniques. Computer, 1981, 14, 71—75
76. *Rissanen J.*, Generalized Kraft inequality and arithmetic coding. IBM J. Res. & Dev., 1976, 20, 198—203 (PJKMar, 1977, 3B572)
77. —, Arithmetic coding as number representations. Acta Pol. Scand. Math., 1979, 31, 44—51 (PJKMar, 1980, 5B589)
78. —, *Langdon G. G.*, Arithmetic coding. IBM J. Res. & Dev., 1979, 23, 149—162
79. —, —, Universal modeling and coding. IEEE Trans. Inform. Theory, 1981, 27, 12—23
80. *Rodeh M., Pratt V. R., Even S.*, Linear algorithm for data compression vi matching. J. of ACM, 1981, 28, № 1, 16—24

81. *Rubin F.*, Experiments in text file compression. Commun. of ACM, 1976, 19, № 11, 617—623 (PJKMar, 1977, 4B719)
82. *Ruth S. S., Kreutzer P. J.*, Data compression for large business files. Datamation, 1972, 18, № 8, 62—66
83. *Schalkwijk J.*, An algorithm for source coding. IEEE Trans. Inform. Theory, 1972, 18, 395
84. *Schieber W. D., Thomas G. W.*, An algorithm for compaction of alphanumeric data. J. of Library Automation, 1971, 4, № 4, 198—206
85. *Schuegraf E. J.*, A survey of data compression methods for nonnumeric records. Canadian J. on Inform. Sci., 1976, 2, № 1, 93—105
86. —, *Heaps H. S.*, Selection of equiprequent word fragments for information retrieval. Inform. Storage and Retrieval, 1973, 9, № 12, 697—711
87. —, —, A comparison of algorithms for database compression by use of fragments as language elements. Inform. Storage and Retrieval, 1974, 10, № 9, 309—319
88. *Schwartz E. S.*, Dictionary for minimum redundancy coding. J. of ACM, 1963, № 10, 413—439
89. —, *Kleinboemer A. J.*, A language element for compression coding. Inform. & Control, 1967, 10, № 3, 315—333
90. *Severance D. G.*, A practitioner's guide to data base compression. Inform. Systems, 1983, 8, № 1, 51—62
91. *Shannon C. E.*, A mathematical theory of communication. Bell System Tech. J., 1948, 27, 379—423
92. —, Prediction and entropy of printed English. Bell Syst. Tech. J., 1951, 30, 50—64
93. *Snyderman M., Hunt B.*, The myriad virtues of text compaction. Datamation, 1970, № 16, 36—40
94. *Storer J., Szymanski T.*, Data compression via textual substitution. J. of ACM, 1982, 29, № 4, 928—951 (PJKMar, 1983, 4B1224)
95. *Tanaka H., Kaneku S.*, Data compression approach to cryptography. Proc. of the 1979 Carnahan Conf. on Crime Countermeasures, Lexington—Kentucky, 1979, 159—163
96. *Ting T. C.*, Compacting homogeneous text for minimizing storage space. Int. J. Computer & Inform. Sciences, 1977, 6, № 3, 211—221
97. *Tropper R.*, Binary—coded text, a text compression method. Byte, 1982, 7, № 4, 398—413
98. *Varn B. F.*, Optimal variable length codes. Inform. & Control, 1971, 19, 289—301 (PJKMar, 1972, 3B374)
99. *Wagner R. A.*, Common phrases and minimum text storage. Commun. of ACM, 1973, 16, № 3, 148—153
100. —, An algorithm for extracting phrases in a space optimal fashion. Commun. of ACM, 1973, 16, № 4, 183—185
101. *Walker V. R.*, Compaction of names by X-grams. Proc. of ASIS, 1969, 6, 129—135
102. *Weiss S. F., Vernor R. L.*, A word—based compression technique for text files. J. of Library Automation, 1978, 11, № 2, 97—105
103. *Welch T.*, A technique for high performance data compression. Computer, 1984, 17, № 6, 8—19
104. *Wells M.*, File compression using variable length encodings. Computer J., 1972, 15, № 4, 308—313
105. *White H. E.*, Printed English compression by dictionary encodings. Proc. of IEEE, 1967, 55, № 3, 390—396
106. *Williams P. W.*, Criteria for choosing subsets to obtain maximum relative entropy. Computer J., 1978, 21, № 1, 57—65
107. *Wisniewski J. L.*, Effective text compression with simultaneous bigram and trigram encoding. J. of Inform. Sci., 1987, № 13, 159—164
108. —, Compression of index—term dictionary in an inverted—file—oriented data base: some effective algorithms. Inform. Process. & Manag., 1986, № 6, 493—501

109. Wolf J. G., Recording of natural language for economy of transmission of storage. Computer J., 1978, 21, № 1, 42—44
110. —, An algorithm for the segmentation of an artificial language dialogue. British J. of Psychology, 1975, 66, 79—90
111. Yavuz D., Zipf's law and entropy. IEEE Trans. on Inform. Theory, 1974, 20, № 5, 650—657 (РЖМат, 1975, 5B1135)
112. Young T. Y., Liu P. S., Overhead storage considerations and a multilinear method for data file compression. IEEE Trans. on Software Eng., 1980, 6, № 4, 340—347
113. Zipf G. K., Human behaviour and the principle of least effort. Addison Wesley, Cambridge, Mass., 1949
114. Ziv J., Lempel A., A universal algorithm for sequential data compression. IEEE Trans. on Inform. Theory, 1977, 23, № 3, 337—343
115. —, Compression of individual sequences via variable rate coding. IEEE Trans. on Inform. Theory, 1978, 24, № 5, 530—536 (РЖМат, 1979, 4B558)
116. Амелькин В. А., Методы нумерационного кодирования. Новосибирск: Наука, 1986
117. Белоногов Г. Г., Котов Р. Г., Автоматизированные информационно-поисковые системы. М.: Советское радио, 1968
118. Бондарь Е. В., Методы сжатия информации при организации базы данных АСУП. Сб. «Методы проектирования и организации АСУ», Киев, 1985, 66—70
119. Бризгал Е. Е., Мейерович Л. Н., Сжатие баз данных патентной информации. Сб. «Методы проектирования и организации АСУ», Киев, 1985, 58—61
120. Кавалерчик Б. Я., Гришкан В. И., О повышении производительности вычислительных систем. УСиМ, 1986, № 4, 23—27
121. Колесников Г. С., Пахомов В. И., Переверткин В. Т., Шамбазов А. В., Об одном алгоритме сжатия текстовой информации. «Мат. обеспеч. вычисл. систем.», М., 1984, 107—117 (РЖМат, 1985, 7Г335)
122. Колмогоров А. Н., Три подхода к определению понятия «количество информации». Проблемы передачи информации. М., 1965, I, вып. 1, 11—14
123. Корунец Е. И., Об одном подходе к сжатию массивов двоичной информации. Сб. «Методы проектирования и организации АСУ», Киев, 1985, 101—107