Рукопись. Версия от 01.04.2002

УДК 621.391

М.А. Смирнов
Санкт-Петербургский государственный университет аэрокосмического приборостроения

МЕТОДЫ ПОВЫШЕНИЯ СТЕПЕНИ СЖАТИЯ ТЕКСТОВ НА ЕСТЕСТВЕННЫХ ЯЗЫКАХ ДЛЯ АЛГОРИТМОВ НЕИСКАЖАЮЩЕГО СЖАТИЯ ДАННЫХ

Показывается возможность заметного увеличения степени сжатия текстов на естественных языках за счет учета грамматики языка без непосредственного построения соответствующей вероятностной модели. С целью повышения эффективности экономного кодирования текстовых данных предлагается простая схема предварительной обработки, особенность которой состоит в расстановке маркеров (тегов) принадлежности слова к некоторой части речи.

Специализированные системы неискажающего сжатия текстов на естественных языках (ЕЯ) представляют большой практический интерес, поскольку данные такого типа составляют значительную долю информации, хранимой на внешних носителях и передаваемой по электронным сетям. Типичными областями применения таких систем являются электронные библиотеки и информационно-поисковые системы. Большие массивы текстовой информации циркулируют по Интернет, при этом стандарт на сжатие данных такого рода отсутствует [1].

Универсальные алгоритмы сжатия не учитывают многих особенностей текстов, что приводит к внесению значительной избыточности в закодированное представление. С другой стороны, лингвистические модели, учитывающие специфику текстов с точки зрения морфологии и синтаксиса, обычно уступают в точности статистическим моделям, которыми оперируют алгоритмы универсального сжатия явным или неявным образом [2]. Анализ состояния проблемы показывает, что в настоящее время отсутствует общепризнанная модель текста на ЕЯ [2]. Разработка специализированных алгоритмов сжатия сопряжена, помимо очевидных материальных и временны́х затрат, с трудностями адаптации в случае изменения языка или даже тематики текстов. Поэтому в настоящее время для компрессии текстовых данных обычно применяют технику предварительной обработки (препроцессинга) в сочетании с универсальным алгоритмом сжатия [1, 2, 3, 4]. Предварительная обработка позволяет существенно упростить алгоритм работы кодера и декодера, дает возможность создания на основе универсальных алгоритмов гибкой специализированной системы сжатия текстов определенного типа.

Предварительная обработка данных выполняется до их сжатия как такового и призвана улучшить степень сжатия. Схема кодирования в этом случае приобретает вид:

исходные данные → препроцессор → кодер → сжатые данные,

а схема декодирования:

сжатые данные → декодер → постпроцессор → восстановленные данные.

В результате препроцессинга входной поток должен так видоизменяться, чтобы степень сжатия преобразованных данных была в среднем выше степени сжатия исходных, “сырых” данных. При этом преобразование должно быть обратимым.

Универсальными и обеспечивающими наибольшее улучшение сжатия являются техники словарной замены последовательностей букв исходного текста на индексы совпадающих с этими последовательностями фраз словаря [1, 4]. Современным представителем методов препроцессинга путем словарной замены является алгоритм Length Index Preserving Transform (LIPT) — “преобразование с сохранением индекса длины” [1]. В LIPT-словарь в качестве фраз включаются самые часто используемые слова $\boldsymbol{W}$, определяемые исходя из анализа большого количества текстов на заданном языке и, возможно, определенной тематики. Под словом понимается последовательность букв, ограниченная с двух сторон символами, не являющимися буквами (“не-букв”). Весь словарь $\boldsymbol{D}$ делится на части $\boldsymbol{D_i}$ (подсловари). Если длина $\boldsymbol{L(W)}$ слова $\boldsymbol{W}$ равна $\boldsymbol{i}$, то это слово включается в подсловарь с номером $\boldsymbol{i}$:

$$\boldsymbol{D_i} = \cup_{\boldsymbol{W}:\boldsymbol{L(W)}=\boldsymbol{i}} \boldsymbol{W}.$$

В пределах подсловаря фразы сортируются в порядке убывания частоты; самое часто используемое слово получает минимальный индекс 0. Каждое слово исходной последовательности, с которым совпадает какая-та фраза словаря, кодируется как

флаг	длина фразы $\boldsymbol{i}$ (номер подсловаря)	индекс фразы в подсловаре $\boldsymbol{D_i}$

В качестве алфавита для записи длины и индекса авторами алгоритма предлагается использовать алфавит языка. Например, если обрабатывается текст на английском языке, то индекс 1 соответствует “a”, 2 — “b”, …, 26 — “z”, 27 — “A”, …, 52 — “Z”, 53 — “aa”, 54 — “ab”, и т.д. Роль флага исполняет символ “*”. Варианты преобразования слова “mere” в зависимости от индекса соответствующей ему в подсловаре $\boldsymbol{D_4}$ фразы приведены в табл. 1 (для большей доходчивости различные части кодового слова фразы разделены подчеркиванием).

Таблица 1

Индекс	Кодовое слово фразы
0	* _d
29	* _d_C
56	* _d_ad

Индекс записывается в позиционной системе счисления, и первая буква соответствует старшему порядку. Если индекс нулевой, то он не передается. Если слово отсутствует в словаре, то оно без изменений копируется в файл преобразованных данных. Конец последовательности, задающей индекс, нет нужды указывать явно, поскольку за любым словом следует какая-либо “не-буква”, которая и становится маркером конца записи индекса. Однозначность обратного преобразования обеспечивается тем, что алфавит индекса не пересекается с алфавитом “не-букв”. Для передачи символа “*” исходного текста используется специальный символ ухода, за который авторами алгоритма выбран знак “\”. Поэтому символ “*” представляется как последовательность “*”. С помощью такого же механизма передается и символ “\”. Чем реже встречаются знаки “*” и “\” в исходном тексте, тем меньше вносится в преобразованный текст служебной информации, и тем сильнее последующее сжатие.

В рамках оригинального алгоритма LIPT также используется обратимое преобразование заглавных букв в строчные, выполняемое перед словарной заменой.

Были изучены следующие варианты модификации LIPT:

1) изменение алгоритма формирования кодовых слов;
2) частичный учет синтаксической структуры текста с помощью разбиения словаря LIPT по признаку принадлежности слова к определенной части речи (лексико-грамматическому разряду).

Разработанные алгоритмы сравнивались между собой и с оригинальным LIPT по критерию степени сжатия набора преобразованных текстов с помощью трех архивато-

ров. Каждый из использованных архиваторов реализует один из трех основных типов алгоритмов универсального сжатия без потерь — алгоритмы на основе сортировки блоков (Burrows-Wheeler Transform, далее BWT), словарные алгоритмы типа LZ77 (Ziv-Lempel-77, или Зив-Лемпел-77) и статистические алгоритмы типа PPM (Prediction by Partial Matching, или предсказание по частичному совпадению). Были отобраны следующие программы сжатия: Bzip2, вер. 1.00 (алгоритм типа BWT); RAR, вер. 2.50 (алгоритм типа LZ77); HA, вер. 0.999c, параметр a2 (алгоритм типа PPM).

Существуют оценки, что тексты на английском языке составляют до 80% от всей текстовой информации, доступной в Интернет [2]. Кроме того, оригинальный алгоритм LIPT разрабатывался в первую очередь для обработки англоязычных текстов. В связи с этим сравнение проводилось на английских текстах. Набор текстов был составлен из девяти файлов, каждый из которых входит в какой-либо из распространенных тестовых наборов для сравнения программ универсального сжатия [5, 6, 7].

Ниже приводятся результаты сравнения трех разработанных автором модификаций LIPT:

1) алгоритм 1 — алфавиты длины и индекса соответствуют оригинальному LIPT, но словарь $\boldsymbol{D}$ разбивается на подсловари не только по критерию длины слов, но и по критерию соответствия слову определенной части речи;
2) алгоритм 2 — алфавиты длины слова и индекса отличаются от алфавита букв и не пересекаются между собой; разбиение $\boldsymbol{D}$ на подсловари соответствует оригинальному LIPT;
3) алгоритм 3 — используются такие же алфавиты, что и в алгоритме 2; $\boldsymbol{D}$ разбивается на подсловари так же, как и в алгоритме 1.

Словарь LIPT был построен на основании анализа примерно 50 Мбайт английских текстов различного характера. Общий объем словаря составил 53 тыс. фраз, или 480 кбайт. Максимальная длина фразы была ограничена 25 буквами. Для реализации алгоритмов 1 и 3 примерно 12 тысячам чаще всего использовавшимся слов был присвоен атрибут $\boldsymbol{P}$ принадлежности к определенной части речи, например: $\boldsymbol{P}(\text{"}\boldsymbol{on}\text{"}) = 1$ (предлог), $\boldsymbol{P}(\text{"}\boldsymbol{you}\text{"}) = 2$ (местоимение) и т.п. Всего использовалось 9 лексико-грамматических разрядов: предлог, местоимение, существительное, прилагательное, глагол, артикль, союз, наречие и “прочее”. Если слово может относиться к нескольким частям речи, то выбиралась чаще всего употребляемая форма, предлагаемая в [8].

Для алгоритмов 1 и 3 в подсловарь $\boldsymbol{D}_i^j$ записываются только слова длины $\boldsymbol{i}$, отнесенные к $\boldsymbol{j}$-ой части речи

$$\boldsymbol{D}_i^j = \cup_{\boldsymbol{W}:\boldsymbol{L}(\boldsymbol{W})=i,\, \boldsymbol{P}(\boldsymbol{W})=j} \boldsymbol{W}\,.$$

Слова, не получившие лексико-грамматического атрибута, трактуются как существительные. Используется следующая схема формирования кодового слова фразы

флаг	часть речи ($\boldsymbol{j}$)	длина ($\boldsymbol{i}$)	индекс в подсловаре $\boldsymbol{D}_i^j$

Например: “mere” → “<флаг><прилагательное>< $\boldsymbol{L}(\boldsymbol{W}) = 4$ ><индекс>”,

где индекс определяет положение фразы в подсловаре 4-буквенных прилагательных. Атрибуты $\boldsymbol{i}$ и $\boldsymbol{j}$ передаются с помощью одного байта каждый, индекс — с помощью такого же способа, что и в оригинальном LIPT.

С целью уменьшения влияния посторонних факторов на результат сравнения эффективности собственно LIPT-подобных алгоритмов при исследовании не выполнялось преобразование заглавных букв. В исходных файлах тестового набора все заглавные буквы были необратимым образом заменены на строчные. Результаты экспериментов сведены в табл. 2 (агрегированная информация) и табл. 3 (детальная информация о степени сжатия файлов). Степень сжатия вычислялась как

$$K = \frac{N_1}{N_2},$$

где N_1 — размер исходного файла в байтах;

N_2 — размер сжатого преобразованного файла в байтах.

Средняя степень сжатия $\overline{K}$ равна среднему невзвешенному степеней сжатия отдельных файлов набора. Символом Δ обозначен выигрыш в сжатии в процентах относительно оригинального алгоритма LIPT. Наилучшие по критерию $\overline{K}$ значения выделены жирным шрифтом в рамках каждого типа архиватора.

Таблица 2

	BWT		LZ77		PPM	
	$\overline{K}$	Δ	$\overline{K}$	Δ	$\overline{K}$	Δ
Оригинальный LIPT	3.93	0%	**3.26**	**0%**	3.95	0%
Алгоритм 1	3.93	0.1%	3.14	-3.6%	4.00	1.3%
Алгоритм 2	**4.05**	**3.0%**	3.08	-5.6%	4.03	2.1%
Алгоритм 3	4.03	2.5%	3.00	-8.0%	**4.05**	**2.6%**
Без LIPT (справочно)	3.70		2.96		3.70	

Таблица 3

	LIPT			Алгоритм 1			Алгоритм 2			Алгоритм 3		
	BWT	LZ77	PPM	BWT	LZ77	PPM	BWT	LZ77	PPM	BWT	LZ77	PPM
Alice29	3.91	**3.29**	3.96	3.95	3.19	4.00	**4.03**	3.11	4.02	4.02	3.05	**4.03**
Asyoulik	3.46	**2.94**	3.54	3.46	2.83	3.59	**3.58**	2.75	3.60	3.57	2.69	**3.62**
Bible	4.98	**3.92**	4.72	4.96	3.80	4.74	**5.07**	3.72	**4.78**	5.06	3.64	**4.78**
Book1	3.52	**2.89**	3.59	3.54	2.78	3.67	**3.64**	2.72	3.70	3.63	2.64	**3.74**
Book2	4.09	**3.44**	4.04	4.10	3.31	4.07	**4.19**	3.27	4.13	4.17	3.17	**4.14**
Condoyle	3.89	**3.19**	3.94	3.88	3.08	4.01	**4.00**	3.01	4.03	3.98	2.94	**4.07**
Lcet10	4.33	**3.55**	4.32	4.32	3.41	4.38	**4.44**	3.36	4.41	4.42	3.26	**4.44**
Paper2	3.69	**3.23**	3.79	3.67	3.10	3.84	**3.82**	3.04	3.84	3.79	2.95	**3.85**
Plrabn12	3.52	**2.89**	3.63	3.51	2.77	3.70	**3.64**	2.71	3.76	3.62	2.63	**3.79**
$\overline{K}$	3.93	**3.26**	3.95	3.93	3.14	4.00	**4.05**	3.08	4.03	4.03	3.00	**4.05**

Таким образом, при применении универсального алгоритма сжатия на основе BWT целесообразно использовать для препроцессинга алгоритм 2 (выигрыш относительно оригинального LIPT равен 3.0%), в случае словарной схемы семейства LZ77 — оригинальный LIPT, в случае PPM — алгоритм 3 (выигрыш относительно оригинального LIPT составляет 2.6%). В LZ77 неэффективно учитывается корреляция между последовательностями символов, и основной выигрыш достигается за счет уменьшения длин совпадения и величин смещений. Поэтому более изощренные способы формирования кодового слова фразы, позволяющие явным образом отразить структуру текста, приводят к ухудшению сжатия в случае LZ77.

С целью проверки неслучайности полученного результата процесс формирования словаря в алгоритме 3 был преобразован так, что P присваивались случайные значения из поддерживаемого множества лексико-грамматических разрядов. Сравнение $\overline{K}$ алгоритма 3 и его модификации приведены в табл. 4.

Таблица 4

	BWT	LZ77	PPM
Алгоритм 3	**4.03**	3.00	**4.05**
Преобразованный алгоритм 3	3.96	**3.01**	3.97

Из табл. 4 следует, что нет оснований отвергать гипотезу о том, что улучшение сжатия в случае алгоритма типа PPM связано с эксплуатацией информации о синтаксических особенностях текста, выраженной явно после применения алгоритма 3. Дополнительные эксперименты, проведенные с использованием других PPM-компрессоров и наборов тестовых данных большего объема, взятых из архива проекта “Гуттенберг” [9], подтверждают стабильность выигрыша при применении алгоритма 3.

В табл. 5 представлена статистика по сжатию текста Alice29 с помощью PPM-компрессора. Параметр ***N*** характеризует число байт, закодированных с помощью контекста определенного порядка, ***K*** — степень их сжатия. Статистика была собрана с помощью PPM-компрессора PPMN, разработанного автором [10, 11, 12].

Таблица 5

Порядок контекста	Без LIPT			Оригинальный LIPT			Алгоритм 3		
	N , байт	***N*** , %	***K***	***N*** , байт	***N*** , %	***K***	***N*** , байт	***N*** , %	***K***
4	117194	77.1%	4.65	100446	76.7%	4.00	125503	83.3%	4.42
3	18434	12.1%	2.79	15031	11.5%	2.72	10876	7.2%	4.12
2	11316	7.4%	2.46	7463	5.7%	3.68	6985	4.6%	3.94
1	4338	2.9%	2.01	6051	4.6%	1.88	5531	3.7%	2.25
0	807	0.5%	1.75	1909	1.5%	1.48	1859	1.2%	1.27
Всего	152089			130900			150754		

Из табл. 5. видно, что выигрыш алгоритма 3 достигается за счет увеличения сжатия символов, кодируемых на основании статистики контекстов 4-го и 3-го порядков.

Было проведено исследование целесообразности совместного использования алгоритмов типа LIPT и других техник препроцессинга, а именно: преобразования заглавных букв и преобразования символов-разделителей.

Заглавные буквы существенно увеличивают число различных последовательностей, встречающихся в тексте и, соответственно, приводят к ухудшению сжатия по сравнению с тем случаем, если бы их не было вообще. Поэтому если слово начинается с заглавной буквы, то оно преобразовывается как

флаг	первая буква, преобразованная в строчную	оставшаяся часть слова

Слову, состоящему целиком из заглавных букв, соответствует отображение

флаг 2	последовательность букв слова, преобразованных в строчные

Суть преобразования символов-разделителей сводится к следующему [4]. Символы-разделители — знаки препинания и символы форматирования — в большинстве случаев плохо предсказываются на основании контекстно-зависимой статистики. Особенно плохо предсказываются символы конца строки (СКС), т.е. пара символов {перевод каретки, перевод строки} CR/LF или символ перевода строки LF. Степень сжатия для BWT- и PPM-компрессоров может быть улучшена, если обратимым образом видоизменить знаки препинания и СКС, “выделив” их из потока букв. Наиболее эффективным и простым способом модификации является добавление пробела перед разделителями [4]. Например: “очевидно,” → “очевидно_,”, но при этом “очевидно_,” → “очевидно__,”. Положительный эффект от использования отображения объясняется тем, что пробел встречается в таких же контекстах, что и преобразуемые знаки. Поэтому в результате преобразования уменьшается количество используемых контекстов и, следовательно, увеличивается объем накапливаемой статистики и точность оценок; кроме того, пробел сжимается часто сильнее, чем соответствующий знак препинания или СКС, и при этом предоставляет несколько лучший с точки зрения точности предсказания контекст для последующего разделителя.

Если все СКС преобразовать в пробелы, то сжатие текстов улучшится. Этого можно достигнуть, искусственно разбив исходный файл на два блока: собственно

текст, в котором СКС замещены на пробелы, и сведения о расположении СКС в файле, т.е., фактически, информация о длинах строк. Если расположение СКС достаточно регулярно, то сумма размеров сжатого блока преобразованного текста и сжатого блока длин строк будет меньше размера архива исходного файла [4].

В табл. 6 приведены сведения о степени сжатия описанного выше тестового набора, файлы которого были преобразованы следующим образом: препроцессинг 1 — оригинальный LIPT; препроцессинг 2 — преобразование заглавных букв и символов-разделителей; препроцессинг 3 — преобразование заглавных букв и символов-разделителей в сочетании с алгоритмом 2 для BWT, оригинальным LIPT для LZ77 и алгоритмом 3 для PPM.

Таблица 6

	Без препроцессинга	Препроцессинг 1	Препроцессинг 2	Препроцессинг 3	Выигрыш для препроцессинга 3 относительно варианта без препроцессинга
BWT	3.64	3.94	3.73	**4.05**	11.1%
LZ77	2.89	3.16	2.96	**3.20**	10.7%
PPM	3.64	3.94	3.75	**4.03**	10.8%

Таким образом, совместное использование описанных техник препроцессинга позволяет усилить на 10-11% неискажающее сжатие текстов на ЕЯ в рамках использованного тестового набора. Перспективными являются дальнейшие исследования о возможности использования при препроцессинге и сжатии данных сведений о синтаксических и морфологических особенностях текстовой информации. Представляет интерес и возможность учета фонетического аспекта языка.

Библиографический список

1. Awan F., Motgi N.Zh.N., Iqbal R., Mukherjee A. LIPT: A Lossless Text Transform to Improve Compression// Proceedings of International Conference on Information and Theory: Coding and Computing, IEEE Computer Society, Las Vegas Nevada, April 2001.
2. Teahan W.J. Modelling English Texts: PhD thesis/ Department of Computer Science, The University of Waikato, Hamilton, New Zealand, May 1998.
3. Chapin B., Tate S. Higher Compression from the Burrows-Wheeler Transform by Modified Sorting// Proceedings of Data Compression Conference, Snowbird, Utah, March 1998.
4. Grabowski Sz. Text preprocessing for Burrows-Wheeler block sorting compression// Conf. proc. of VII Konferencja "Sieci i Systemy Informatyczne" (7th Conference "Networks and IT Systems"), Lodz, Oct. 1999, pp. 229-239.
5. Calgary Compression Corpus. http://links.uwaterloo.ca/calgary.corpus.html
6. Canterbury Compression Corpus. http://corpus.canterbury.ac.nz
7. Compressors Comparison Test Вадима Юкина. http://www.compression.ru/ybs/
8. Новый Большой англо-русский словарь: В 3 т./ Под общ. ред. Апресяна Ю.Д. Изд. 6-е изд., стереотип. М.: Русский язык, 2001. 2484 с.
9. Project Gutenberg. http://www.promo.net/pg/
10. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.: ДИАЛОГ-МИФИ, 2002. – 384 с. http://www.compression.ru/book/
11. Оценка вероятности ухода в алгоритмах PPM/ Смирнов М.А.: Санкт-Петербургский государственный университет аэрокосмического приборостроения, СПб., 2000. – 15 с., – Рус. – Деп. в ВИНИТИ 31.10.2000г., № 2741-В00.
12. Смирнов М.А. (1999). PPMN – PPM-компрессор. http://www.compression.ru/ms/