Некоторые модификации алгоритма сжатия BPE

А.Я. Казаков[I,II], А.А. Жихарева [I,III], П.А. Пасечник[I]

[I] Санкт-Петербургский государственный университет промышленных технологий и дизайна, г.Санкт-Петербург, Россия
[II] Санкт-Петербургский государственный университет аэрокосмического приборостроения, г.Санкт-Петербург, Россия
[III] Национальный исследовательский университет «Высшая школа экономики», г.Санкт-Петербург, Россия

Аннотация. В работе обсуждаются проблемы сжатия больших объемов данных с целью передачи их по каналам связи либо с целью их долгосрочного хранения. Предложено несколько вариантов алгоритмов кодирования информации, основанных на известном алгоритме BPE (Byte Pair Encoding), в том числе, блочно-потоковый вариант BPE. Проведен сравнительный анализ полученных результатов с точки зрения совокупности факторов — коэффициента сжатия данных и затраченного на кодирование времени.

Ключевые слова: сжатие данных, кодирование, алгоритмы сжатия, алгоритм Зива-Лемпеля, BPE.

DOI 10.14357/20718632180306

Введение

Сжатие данных с целью их экономичного хранения, обработки и передачи по линиям связи остается одной из самых важных задач прикладной информатики. В настоящее время известен ряд алгоритмов сжатия данных без потерь, все они нашли широкие приложения на практике [1, 2]. Наиболее широко используются алгоритмы Хаффмена и Зива-Лемпеля и их различные варианты. В целом алгоритмы сжатия могут быть классифицированы по свойствам, которыми они обладают. Например, алгоритм Хаффмена требует предварительного изучения статистики текста. Алгоритм Зива-Лемпеля предварительной обработки текста не требует, он может быть реализован в потоковом варианте, по мере поступления последующих символов текста. Еще один из критериев классификации – наличие или отсутствие словаря в процессе кодирования (сжатия) и декодирования. Одним из алгоритмов, в котором предполагается генерация словаря при реализации сжатия, является алгоритм Byte Pairing Encoding (BPE), или Digram coding, который был предложен в работе [3, 4], в работе [5] были предложены различные варианты этого алгоритма. Здесь мы обсудим дальнейшие модификации BPE, обладающие дополнительными возможностями, опишем модификации этого алгоритма, в том числе такие, которые учитывают корреляцию удаленных символов и такие, которые позволяют реализовать потоковую обработку входной строки и не требуют создания словаря.

1. Метод сжатия BPE

Исходным объектом в задаче сжатия является достаточно длинный текст, который можно представлять как строку символов. При этом обычно имеется информация о наборе символов, которые встречаются в строке (исходный алфавит), а также о наборе символов, которые можно употреблять в процессе сжатия (дополнительный алфавит). Процедура сжатия без по-

терь должна удовлетворять следующим естественным условиям:

1. процессы кодирования и декодирования для всех исходных строк взаимно-обратны (это требование является императивом при построении алгоритмов сжатия без потерь);

2. результат кодирования должен быть возможно меньшего объема;

3. процессы кодирования и декодирования должны быть не слишком ресурсоемкими (в смысле требований к необходимой памяти и времени обработки).

1.1. Кодирование

Опишем реализацию исходного варианта алгоритма BPE. На первом шаге просматривается (в выбранном порядке, скажем, слева направо) вся исходная строка с целью определения наиболее часто встречающейся пары последовательных символов. Найденной паре сопоставляется символ из дополнительного алфавита, эта подстановка записывается в словарь. Далее при проходе по строке выбранная пара символов, как только она обнаруживается в строке, заменяется на сопоставленный ей символ. В результате получается преобразованная строка, причем эта строка по построению будет короче исходной. Алгоритм повторяет этот процесс до тех пор, пока дальнейшее сжатие не будет невозможно либо из-за отсутствия часто встречающихся пар, либо из-за исчерпания символов в дополнительном алфавите. В итоге получаем сжатую строку и словарь подстановок, в соответствии с которым проведено кодирование.

1.2. Декодирование

Первым шагом процедуры декодирования является преобразование словаря. Для этого надлежит рассматривать подстановки, записанные в словаре, в обратном порядке (справа налево). Каждую подстановку следует учесть в подстановках, записанных левее данной подстановки. Достигнув при этом последовательном процессе самой левой подстановки, мы получим выражение всех подстановок словаря в символах исходного алфавита. Теперь декодирование выполняется всего за один проход: двигаясь по сжатой строке слева направо, реализуем все подстановки из преобразованного словаря. Таким образом, сжатие исходной строки требует многих проходов, обратный процесс – восстановление исходной строки – может быть организован только за один проход за счет предварительного преобразования словаря.

Нетрудно показать, что данный алгоритм удовлетворяет описанному выше условию взаимной обратности процессов кодирования и декодирования (условие 1).

1.3. Возможные модификации BPE

Из приведенного выше описания BPE следует, что данный алгоритм – это кодирование со словарем, оно использует корреляцию последовательных пар символов. В данной работе мы рассмотрим несколько модификаций BPE, связанных со следующими возможностями. Во-первых, можно рассматривать последовательные тройки символов; точнее, одновременно пары и тройки символов, устанавливая соответствующую иерархию. Во-вторых, можно рассматривать пары символов, не являющихся соседними, учитывая таким образом возможную дальнюю корреляцию символов в исходной строке. В-третьих, можно на каждом из проходов определять не одну заменяемую последовательность символов, а несколько таких кодируемых последовательностей для выполнения серии последовательных замен. В соответствии со сформулированными изменениями мы называем эти модификации:

1. BPTE (Byte Pair–Triple Encoding);
2. BPE с удаленной корреляцией;
3. блочный BPE.

В нашей работе [5] мы подробно обсуждали эти варианты, так что далее мы лишь вкратце осветим особенности их реализации. Основное внимание уделим модификации блочного варианта, которую мы будем называть блочно-потоковым вариантом BPE. Этот вариант алгоритма обладает в определенной степени характеристиками LZW – он потоковый и не требует создания словаря. Для сравнения характеристик этих модификаций мы приводим результаты реализации этих алгоритмов для стандартного набора текстов.

Замечание. Конечность доступного набора кодов является существенным ограничением данного алгоритма. Однако процесс реализации

этого алгоритма можно модифицировать так, чтобы несколько ослабить это ограничение. Рассмотрим процесс формирования словаря на примере. Пусть исходная строка содержит набор символов *...abc...*, причем пару символов *ab* мы устраняем с помощью замены $ab \rightarrow w$, а пару символов *wc* — с помощью замены $wc \rightarrow u$. В последнюю замену можно подставить *ab* вместо *w*. При этом возможна ситуация, когда в преобразованной строке символ *w* будет отсутствовать. В этом случае символ *w* «освобождается» и может быть вновь использован в процедуре сжатия строки.

2. BPTE (Byte Pair–Triple Encoding)

В данной модификации метода BPE предлагается на каждом проходе учитывать как пары, так и тройки соседних символов, и затем заменять набор символов (пару или тройку) с наибольшим весом на символ из дополнительного алфавита. Под (суммарным) весом пары мы понимаем число символов, сокращаемых в строке за одну замену; в этом случае вес каждой пары символов будет равен единице, вес каждой тройки символов будет равен двум. После прохождения строки для каждого набора символов (каждой пары и тройки последовательных символов) получим суммарный вес, равный произведению частоты вхождения набора символов и веса этого набора. Замене подлежит набор с наибольшим суммарным весом. Результатом кодирования является преобразованная (сжатая) строка и соответствующий словарь выполненных замен. Эта информация подлежит хранению и передаче по каналам связи. Декодирование, так же, как и в методе BPE, начинается с преобразования словаря, выполняемого от последней замены к первой с учетом всех предшествующих замен. После чего, проходя сжатую строку слева направо и выполняя замены в соответствии с преобразованным словарем, получим исходную строку.

Рассмотренная модификация обладает тем же недостатком, что и метод BPE: для определения пары или тройки символов с наибольшим весом на каждом проходе требуется организовывать новый поиск. Поэтому следует ожидать, что длительность кодирования в данном варианте модификации не претерпит существенных изменений.

3. BPE с удаленной корреляцией

В реальных приложениях возможны ситуации, когда в исходной строке присутствуют коррелированные пары символов, удаленные друг от друга. В работе [5] описана модификация BPE, позволяющая учитывать удаленную корреляцию пар символов.

Введем параметр метода — *D (distance)*, который может принимать только целые значения большие двух: $D > 2$. Этот параметр определяет длину выделяемого набора последовательных символов, первый и последний символы которого представляют собой пару с удаленной корреляцией. Стандартный BPE соответствует значению параметра $D=2$, когда в заменяемой паре нет пробелов. При проходе по строке определяется набор пар, которые отличаются символами и/или числом пробелов между ними, и частота вхождения каждой пары в строку.

Построение алгоритма сжатия аналогично BPE с учетом соответствующих изменений в процедуру проведения замены. Из всего множества выделенных пар с удаленной корреляцией выберем ту, которая встретилась наибольшее число раз. Сопоставим ей символ из дополнительного алфавита и проведем замену данной пары на код по следующему правилу: первый (левый) символ пары с удаленной корреляцией заменим в строке данных на код пары, а второй (правый) символ пары из строки удалим; при этом символы, стоящие между символами пары, оставим на своих местах. Описанные замены совершаем, двигаясь по строке, как и прежде, слева направо. В результате такой замены получим преобразованную строку меньшей длины по отношению к исходной и запись в словаре о замене, которая помимо заменяемой пары и сопоставленного ей кода содержит число пробелов между символами удаленной пары.

В итоге процедуры сжатия получим сжатую строку и упорядоченный (по порядку совершенных замен) словарь с информацией о заменах следующей структуры: (*код; пара; число пробелов*). Этой информации достаточно для восстановления строки в своем исходном виде [5].

Обсудим вкратце выбор параметра D. Предположим, что исходная строка символов формировалась путем склейки набора подстрок фиксированной длины r, причем подстроки возникали в ходе какого-то естественного процесса. В качестве примера строки такой структуры можно рассмотреть результат разворачивания в строку графического образа, записанного в формате uint8. При этом естественно полагать, что значения яркостей соседних пикселей хорошо коррелированы. Таким образом, параметр D следует выбирать равным $D=r+1$, где r равно радиусу корреляции данных.

Разумеется, данный вариант алгоритма сжатия может быть реализован и с помощью выбора пар и троек удаленных друг от друга символов. Работа с тройками потенциально ускоряет работу алгоритма, однако в этом случае усложняется система записи замены в словарь: для тройки символов следует указывать два числа пробелов — между первым–вторым и вторым–третьим символами.

4. Блочно-потоковый BPE

Сравнивая метод BPE и еще один популярный метод сжатия — LZW, можно отметить следующие характерные особенности этих алгоритмов. К преимуществам BPE можно отнести тот факт, что он в более полной мере учитывает статистику распределения символов в строке, используя для образования новых символов только наиболее часто встречающиеся пары символов. Его реализация приводит к безусловному сокращению исходной строки. Недостатки этого алгоритма состоят в том, что при его реализации требуется много раз проходить вдоль строки. Это приводит к существенному замедлению процесса сжатия. Достоинство LZW состоит в том, что это алгоритм, последовательно преобразующий входящий поток символов. Недостаток метода — в том, что он учитывает все встречающиеся сочетания символов, даже относительно редкие, что приводит иногда даже к увеличению объема файла.

Мы обсудим здесь еще один вариант алгоритма сжатия, который можно рассматривать как блочно-потоковый вариант BPE, сочетающий достоинства BPE и LZW.

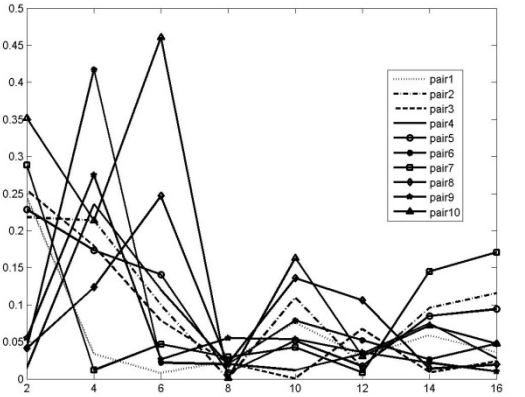

Рис. 1. Зависимость ошибки относительной частоты пары от размера блока (в Кб) на примере первых (по частотности) десяти кодируемых пар

4.1. Блочная обработка

Очевидным недостатком метода BPE является необходимость на каждом проходе перед выполнением замены организовывать новый поиск для определения наиболее часто встречающейся пары символов, что существенно влияет на объем вычислений и, соответственно, на длительность кодирования. В то же время численные эксперименты (при выделении произвольным образом части исходной строки длины 2 - 16 Кб) для обычных текстовых файлов показывают, что относительная ошибка частотности ведущих пар не превосходит 1 или 100 % уже при величине блока 2 Кб, а при размере блока 8 Кб эта относительная ошибка меньше 30 % (Рис. 1).

Таким образом, статистика пар может быть достаточно надежно установлена на относительно небольшом блоке исходной строки. В связи с этим в [5] обсуждался блочный вариант BPE. Здесь мы рассмотрим модификацию этого подхода.

4.2. Блочно-потоковый BPE

В соответствии с результатами, представленными на Рис. 1, частотности пар последовательных символов мы будем определять не по всей входной строке, а по ее части (блоку).

4.2.1. Процесс сжатия

Пусть алфавит исходной строки составляет k символов из n возможных. Остальные $m = n - k$ символов образуют дополнительный алфавит.

Мы будем реализовывать сжатие строки за M шагов (проходов). На каждом шаге будем заполнять не одну, а сразу несколько строк словаря, скажем, K. При этом число проходов M определяется соотношением $M \cdot K = m$ и, очевидно, будет достаточно мало по сравнению с числом свободных символов m.

Объектами нашего алгоритма будут являться исходная строка (подлежащий сжатию файл), создаваемый основной словарь — словарь замен пар символов на соответствующий код и вспомогательный (временный) словарь.

Опишем процедуру заполнения словаря и выполнения замен на каждом шаге алгоритма.

Шаг 1. Рассмотрим в начале строки два последовательно расположенных блока — первый размера $L_1 \sim 2$ Кб, второй размера $L_2 \sim 8$ Кб. Выделим из первого блока $2 \cdot K$ уникальных пар с наибольшей частотой вхождения в него и занесем их во вспомогательный словарь. Во втором блоке нас будут интересовать только эти пары. Обновим частотность пар из вспомогательного словаря в результате прохождения второго блока, после чего выделим из них только K пар с наибольшим значением частот. Занесем их в основной словарь замен, одновременно сопоставляя каждой паре код из числа свободных символов. В результате первого шага мы «прошли» небольшую часть исходной строки и заполнили только часть основного словаря; вспомогательный словарь по окончании первого шага очищается.

Шаг 2. Непосредственно после второго блока выделим в исходной строке два последовательных блока длиной $\frac{(k+K)^2}{k^2} \cdot L_1$ и $\frac{(k+K)^2}{k^2} \cdot L_2$ соответственно. Проведем в первом блоке замены согласно основному словарю. Далее, как и в шаге 1, по этому же блоку сформируем вспомогательный словарь из $2 \cdot K$ пар с наибольшей частотой вхождения в него. Теперь будем двигаться слева направо по второму блоку, при этом для каждой пары символов будем выполнять замену в соответствии с основным словарем (если она там имеется) или определять частоту ее вхождения в текущий блок, если такая пара содержится во вспомогательном словаре. По окончании прохода блока выделим из вспомогательного словаря только K пар с наибольшими значениями частот и добавим их в основной словарь, поставив при этом в соответствие каждой паре код из числа свободных символов, после чего очистим вспомогательный словарь. Таким образом, по окончании второго шага основной словарь будет содержать уже $2 \cdot K$ пар с соответствующими кодами.

Каждый последующий шаг повторяет последовательность действий шага 2 с той лишь разницей, что изменяются размеры исследуемых блоков и число выполняемых в соответствии с основным словарем замен. На j–м шаге будем рассматривать блок исходной строки длины $\frac{(k+(j-1)\cdot K)^2}{k^2} \cdot L_1$ для формирования временного вспомогательного словаря из $2 \cdot K$ пар с наибольшими частотами и следующий за ним блок длины $\frac{(k+(j-1)\cdot K)^2}{k^2} \cdot L_2$ для отбора из вспомогательного словаря в основной только K пар. Увеличение размера блоков соответствует увеличению числа возможных пар символов, создаваемых с помощью символов алфавита и используемых на текущем шаге кодов.

В итоге, после выполнения M шагов будет заполнено $M \cdot K$ строк основного словаря и обработана часть исходной строки общей длиной $\frac{L_1 + L_2}{k^2} \cdot \sum_{j=0}^{M-1} (k + j \cdot K)^2$.

В оставшейся части строки замены проводятся в соответствии со сформированным основным словарем. На этом кодирование исходной строки завершено. Поскольку число шагов в процессе кодирования существенно сокращено, то следует ожидать, что эти изменения дадут значительное уменьшение времени сжатия исходных данных.

4.2.2. Процесс декодирования

Процесс восстановления строки из ее сжатого состояния аналогичен процессу декодирования методом BPE: для декодирования сжатой строки достаточно одного прохода с предварительно преобразованным словарем. Рассмотрим этот вопрос подробнее.

Описанный выше процесс сжатия строки включает одновременное формирование словаря, состоящего из M частей. Сжатая строка и словарь передаются получателю и являются исходным материалом для восстановления исходной строки. Сначала мы преобразуем словарь так, чтобы процесс декодирования реализовался за один просмотр строки. Заметим, что пары в первой части словаря содержат только символы исходного алфавита, так что эта часть словаря не подлежит преобразованию. Затем рассмотрим вторую часть словаря. Если пары в этой части содержат «новые» символы, эти символы в силу особенностей формирования словаря содержатся только в первой части, и мы замещаем эти «новые» символы парами символов исходного алфавита в соответствии с заменами из первой части словаря. Соответственно, «новые» символы в парах из третьей части словаря можно заменить наборами символов из предыдущих (т.е. первой и второй) частей словаря и т.д. Завершив это преобразование словаря, мы получим словарь замен «новых» символов комбинациями из символов исходного алфавита. После этого мы просматриваем сжатую строку слева направо и все «новые» символы заменяем согласно преобразованному словарю, восстанавливая в итоге исходную строку.

4.2.3. Результаты и их обсуждение

Отметим следующие особенности данного алгоритма.

1. Словарь формируется поэтапно, с учетом статистики появления последовательных пар в исходной строке. Кроме того, он формируется в итоге на основе просмотра не всего текста, а только начальной его части длины $\frac{L_1 + L_2}{k^2} \cdot \sum_{j=0}^{M-1} (k + j \cdot K)^2$.

2. Обработка строки происходит последовательно, так что в этом отношении данный алгоритм подобен LZW.

3. Возможен выбор параметров алгоритма L_1; L_2; M; K. Алгоритм можно оптимизировать по этим параметрам.

4. Отметим также, что в силу особенностей процесса кодирования исходной строки в предложенной модификации метода (текущий блок строки подлежит кодированию символами только из предыдущей части словаря, но не из полученной на этом этапе; т.е. текущий блок служит лишь для определения очередной группы замен в словаре), для ее однозначного восстановления достаточно лишь сжатой строки, символов исходного алфавита и информации о длине блоков для формирования отдельных частей словаря. Сам словарь замен может быть сформирован при процедуре декодирования. Таким образом, возможна реализация этой версии BPE как процесса кодирования без словаря.

5. Экспериментальные результаты и анализ

Для численного анализа алгоритмов — BPE и предложенных его модификаций — мы использовали пакет Matlab, который является одним из мощнейших инструментом анализа данных и решения широкого спектра научных и прикладных задач в различных областях. Этот пакет, в частности, имеет в своем составе библиотеки функций, предназначенные для ускорения матричных вычислений, что позволяет существенно сократить время решения стандартных задач и значительно упрощает разработку новых алгоритмов.

Тестирование реализованных алгоритмов проводилось на файлах текстового типа (формата .txt, .c, .html) из стандартных тестовых наборов файлов Calgary corpus и Canterbury corpus. Все вычисления и апробация алгоритмов проведены на персональном компьютере с процессором Pentium (R) Dual-Core CPU T4400 2,20 ГГц и 2 ГБ оперативной памяти.

На графиках (Рис. 2–Рис.5) представлено изменение размера файла в зависимости от числа выполненных по нему проходов при кодировании исходного файла указанным методом. Напомним, что для самого метода BPE и предложенных модификаций — BPTE и BPE с удаленной корреляцией — число проходов по исходной строке соответствует числу проведенных в ней замен (что в свою очередь определяется числом свободных символов и может быть различным для различных исходных данных). Для блочно-потокового BPE число проходов выбирается много меньшим допустимого числа замен и в нашем случае равно 5. Отметим также, что блочно-потоковый метод был реализован в варианте без словаря.

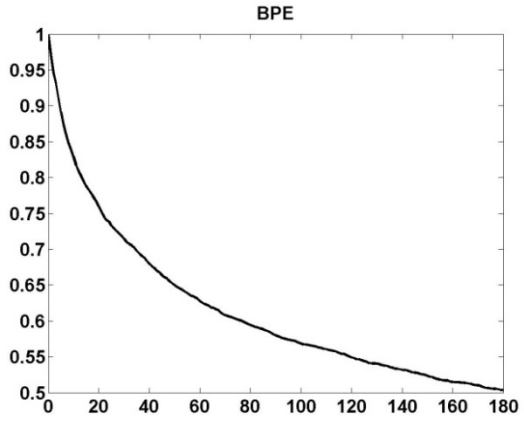

Рис. 2. Изменение коэффициента сжатия в зависимости от числа проходов при сжатии методом BPE файла alice29.txt

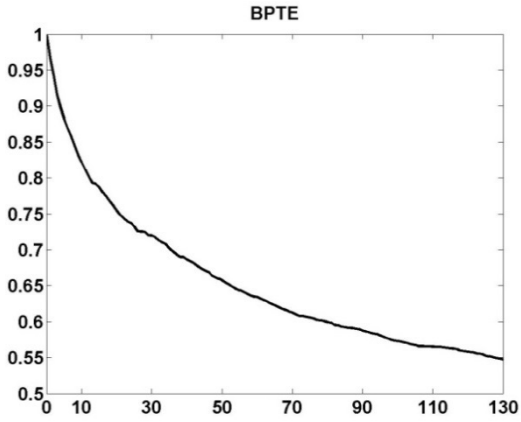

Рис. 3. Изменение коэффициента сжатия в зависимости от числа проходов при сжатии методом BPTE файла alice29.txt

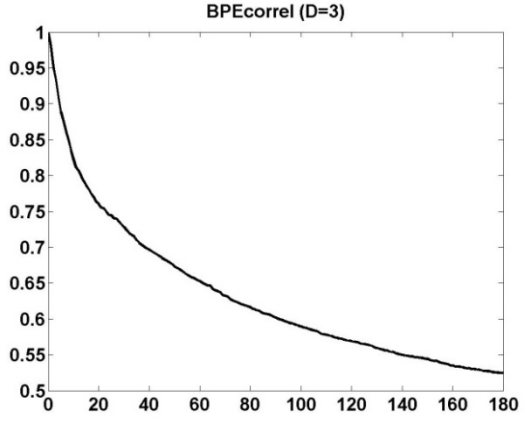

Рис. 4. Изменение коэффициента сжатия в зависимости от числа проходов при сжатии методом BPE с удаленной корреляцией (при значении параметра D=2..3) файла alice29.txt

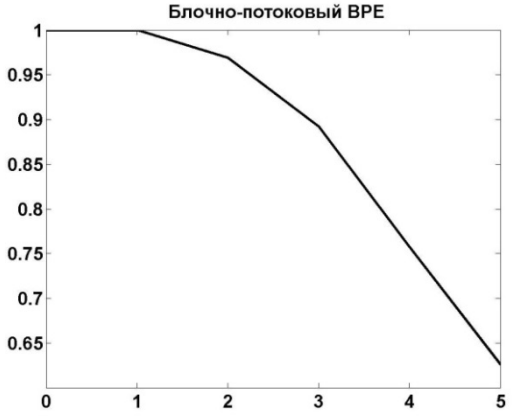

Рис. 5. Изменение коэффициента сжатия в зависимости от числа проходов при сжатии блочно-потоковым BPE файла alice29.txt

Для сравнения предлагаемых методов мы выбрали данные о сжатии одного и того же источника — файла *alice29.txt*, входящего в состав Canterbury corpus.

В Табл. 1 представлены результаты реализации обсуждаемых алгоритмов, примененных к 21 тестовому файлу (размеры исходных файлов составляют от 11 Кб до 770 Кб), а именно, коэффициенты сжатия файлов после применения к исходному файлу описанных методов, а также затраченное на кодирование время в относительных показателях: при этом время кодирования исходного файла классическим методом BPE принято за единицу. Тем самым, значения, большие 1, говорят об увеличении, а меньшие 1 - об уменьшении времени кодирования информации по сравнению со стандартным BPE. В последней строке таблицы приведены усредненные показатели.

Отметим, что блочно-потоковый BPE дает существенное снижение временных затрат на сжатие информации по сравнению со стандартным BPE. В 70 % случаев время кодирования уменьшилось более чем в 10 раз, а для отдельных файлов этот показатель ниже в 50 и более раз (*cp.html, fields.c*, *paper4*, *paper5*).

На Рис. 6 графически представлена совокупность информации о достигнутых показателях — коэффициенте сжатия и затраченном на кодирование времени в среднем для исследуемого набора файлов.

Табл. 1. Показатели эффективности сжатия

Имя файла	*Коэффициенты сжатия*				*Время сжатия (по отношению к BPE)*		
	BPE	*BPTE*	*BPE с удаленной корреляцией (D=3)*	*Блочно-потоковый BPE*	*BPTE*	*BPE с удаленной корреляцией (D=3)*	*Блочно-потоковый BPE*
alice29.txt	0,513	0,546	0,516	0,627	1,96	4,29	0,10
asyoulik.txt	0,550	0,586	0,543	0,685	2,01	5,27	0,09
cp.html	0,495	0,496	0,493	0,881	1,90	5,34	0,02
fields.c	0,450	0,438	0,445	0,950	2,33	5,73	0,02
lcet10.txt	0,536	0,543	0,536	0,588	1,25	2,32	0,36
plrabn12.txt	0,522	0,572	0,526	0,567	1,07	2,07	0,43
bib	0,519	0,523	0,499	0,687	1,87	4,84	0,09
book1	0,556	0,588	0,561	0,587	1,01	1,50	0,57
book2	0,597	0,643	0,679	0,623	0,97	1,25	0,56
news	0,656	0,674	0,671	0,699	1,43	2,66	0,33
paper1	0,604	0,605	0,604	0,799	2,39	5,24	0,04
paper2	0,555	0,574	0,546	0,715	2,20	4,98	0,06
paper3	0,557	0,571	0,567	0,772	2,27	5,35	0,03
paper4	0,553	0,568	0,559	0,930	2,27	5,78	0,01
paper5	0,582	0,583	0,585	0,943	2,52	5,64	0,01
paper6	0,603	0,604	0,618	0,834	2,57	5,33	0,03
pic	0,152	0,184	0,173	0,212	0,41	0,87	0,61
progc	0,635	0,663	0,605	0,836	2,27	4,48	0,03
progl	0,539	0,586	0,587	0,732	2,28	3,81	0,08
progp	0,498	0,490	0,516	0,762	2,69	4,50	0,06
trans	0,600	0,579	0,585	0,736	2,59	4,97	0,09
в среднем	0,537	0,553	0,544	0,722	1,92	4,11	0,17

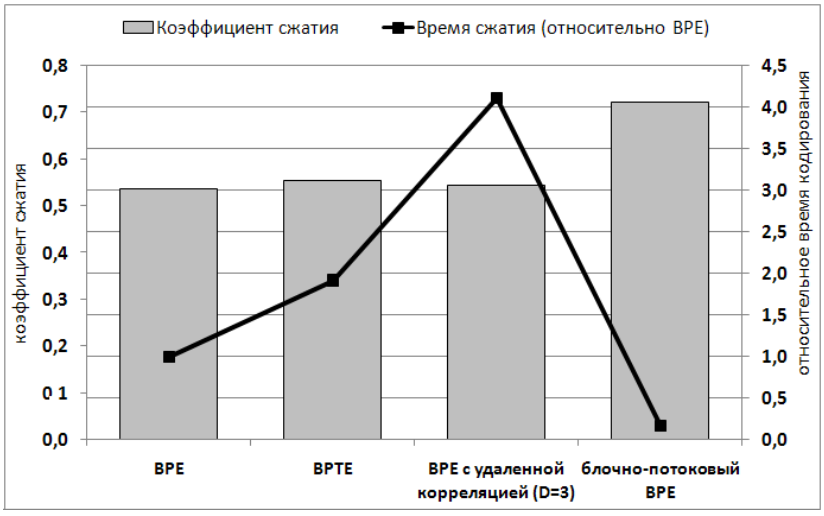

Рис. 6. Изменение показателей оценки эффективности группы методов BPE

Заключение

В практических приложениях требуется иметь широкий набор алгоритмов сжатия, предназначенных для работы с данными различной природы. Здесь мы обсудили несколько возможных модификаций алгоритма BPE. Все эти варианты удовлетворяют ключевому условию алгоритмов сжатия без потерь – кодирование и декодирование взаимнообратны. Мы рассмотрели возможности этих подходов на примере сжатия текстовых файлов из двух тестовых наборов Canterbury и Calgary. Результаты показывают, что блочный вариант BPE позволяет на порядок сократить время кодирования, одновременно несколько снижая параметр сжатия. Мы показали также, что возможна модификация BPE, допускающая отсутствие словаря – он генерируется в процессе кодирования и в процессе декодирования, аналогично тому, как это происходит при реализации LZW.

Литература

1. Sayood, K., Introduction to Data Compression. 2012. 3th ed. — N.Y.:Morgan Kaufmann Publishers, 768 p.
2. Salomon, D., Motta, G., Handbook of data compression. 2010. London: Springer-Verlag, 1361 p.
3. Gage, P., A New Algorithm for Data Compression. 1994. The C Users Journal, Vol. 12(2):23-38.
4. Larsson, J., Moffat, A., Off-line dictionary-based compression. 2000. Proceedings of the IEEE 88 (11): 1722–1732.
5. Казаков А.Я., Жихарева А.А., Пасечник П.А. 2017. Коды сжатия, близкие алгоритму BPE. // Вестник Санкт-Петербургского государственного университета технологии и дизайна, 3:3-9.

Казаков Александр Яковлевич. Высшая школа печати и медиатехнологий Санкт-Петербургского государственного университета промышленных технологий и дизайна, г. Санкт-Петербург, Россия. Заведующий кафедрой, доктор физико-математических наук, профессор. Санкт-Петербургский государственный университет аэрокосмического приборостроения, г. Санкт-Петербург, Россия. Профессор, доктор физико-математических наук, профессор. Количество печатных работ: 98. Область научных интересов: квантовая оптика и квантовая информатика, прикладная информатика, математическая физика. E-mail: a_kazak@mail.ru

Жихарева Алена Аркадьевна. Высшая школа печати и медиатехнологий Санкт-Петербургского государственного университета промышленных технологий и дизайна, г. Санкт-Петербург, Россия. Доцент, кандидат физико-математических наук, доцент. Санкт-Петербургская школа экономики и менеджмента НИУ ВШЭ. г. Санкт-Петербург, Россия. Доцент, кандидат физико-математических наук, доцент. Количество печатных работ: 17. Область научных интересов: сжатие информации, математическое и компьютерное моделирование, имитационное моделирование. E-mail: jikhareva.sg@mail.ru

Пасечник Павел Алексеевич. Высшая школа печати и медиатехнологий Санкт-Петербургского государственного университета промышленных технологий и дизайна, г. Санкт-Петербург, Россия. Аспирант кафедры автоматизации производственных процессов СПбГУПТД, заведующий лабораторией кафедры информационных и управляющих систем СПбГУПТД. Количество печатных работ: 6. Область научных интересов: цифровая обработка изображений, сжатие информации, разработка информационных систем. E-mail: pp@hspm.ru

Some modifications of the BPE algorithm

A.Ya. Kazakov [I,II], A.A. Zhikhareva[I,III], P.A. Pasechnik[I]

[I] Saint Petersburg State University of Industrial Technologies and Design, Saint Petersburg, Russia
[II] Saint-Petersburg State University of Aerospace Instrumentation, Saint Petersburg, Russia
[III] National Research University Higher School of Economics, Saint Petersburg, Russia

Problems of data compression are under consideration. Different modifications of the Byte Pair Encoding (BPE) algorithm are discussed. A comparative analysis of the obtained results is made from the point of view of the set of factors — the compression ratio of data and the time spent on the coding.

Keywords: compression algorithms, data compression, coding, LZW, BPE

DOI 10.14357/20718632180306

References

1. Sayood, K., Introduction to Data Compression. 2012. 3th ed. — N.Y.:Morgan Kaufmann Publishers, 768 p.
2. Salomon, D., Motta, G., Handbook of data compression. 2010. London: Springer-Verlag, 1361 p.
3. Gage, P., A New Algorithm for Data Compression. 1994. The C Users Journal, Vol. 12(2):23-38.
4. Larsson, J., Moffat, A., Off-line dictionary-based compression. 2000. Proceedings of the IEEE 88 (11): 1722–1732.
5. Kazakov, A.Ya, A.A. Zhikhareva, P.A. Pasechnik. 2017. Compression codes close to the BPE algorithm [Vestnik of St. Petersburg State University of Technology and Design]. 3:3-9.

A.Ya. Kazakov. PhD, Professor, Saint Petersburg State University of Industrial Technologies and Design, 191186, 18, Bolshaya Morskaya str., Saint Petersburg, Russia. Saint-Petersburg State University of Aerospace Instrumentation, 67, Bolshaya Morskaya str., Saint Petersburg, 190000, Russia. E-mail: a_kazak@mail.ru

A.A. Zhikhareva. PhD, Saint Petersburg State University of Industrial Technologies and Design, 191186, 18, Bolshaya Morskaya str., Saint Petersburg, Russia. National Research University Higher School of Economics, 16, Soyuza Pechatnikov str, Saint Petersburg, 190121, Russia. E-mail: jikhareva.sg@mail.ru

P.A. Pasechnik. Post-graduate student, Saint Petersburg State University of Industrial Technologies and Design, 18, Bolshaya Morskaya str., Saint Petersburg, 191186, Russia. E-mail: pp@hspm.ru