УДК. 303.732.4
DOI 10.18413/2411-3808-2019-46-1-161-172

ИСПОЛЬЗОВАНИЕ МЕТОДА СЖАТИЯ ИНФОРМАЦИИ В СЕМАНТИЧЕСКОМ ИНФОРМАЦИОННОМ ПОИСКЕ

USING THE METHOD OF INFORMATION COMPRESSION IN A SEMANTIC INFORMATION RETRIEVAL

В.А. Кудинов[1], Нэй Лин[2]
V.A. Kudinov[1], Nay Lin[2]

[1)] Курская государственная сельскохозяйственная академия имени проф. И.И. Иванова
Россия, 305021, г. Курск, ул. Карла Маркса, 70
[2)] Курский государственный университет,
Россия, 305000, г. Курск, ул. Радищева, 33

[1)] Kursk State Agricultural Academy named after prof. I.I. Ivanova,
Russia, 70 Karl Marx St, Kursk, 305021, Russia
[2)] Kursk State University,
Russia, 33 Radishcheva St, Kursk, 305000, Russia

E-mail: kudinovva@yandex.ru, naylynn16@gmail.com

Аннотация
В современном мире возрастает количество людей, использующих Интернет. При этом объём информации в Интернете увеличивается. Однако получаемая информация является семантически неоднозначной, так как имеется проблема семантической неоднозначности слов. Чтобы решить эту проблему, используют онтологии (семантические базы данных). Кроме того, имеется еще одна проблема. Из-за возрастания количества информации необходим больший объём памяти для ее сохранения, но при этом значительно увеличивается время обработки таких данных. Возникают сложности для информационного поиска (ИП). Для того чтобы решить эту проблему, применяется сжатие данных в задачах ИП. В этой статье предлагается новая модель семантического информационного поиска, использующая метод сжатия на основе кода End Tagged Dense Code-ETDC. Для сжатия статьи Wiki используется метод ETDC, что обеспечивает получение коэффициента сжатия 25%. Для построения файла терминологического словаря необходимы только простые тексты. Кодирование – процесс очень простой и легко реализуется программно. Поэтому время кодирования и декодирования меньше, чем в методе Хаффмана.

Abstract
In the modern world, the number of people using the Internet is increasing. At the same time, the volume of information on the Internet is increasing. However, the information obtained is semantically ambiguous, since there is a problem of semantic ambiguity of words. To solve this problem, use Ontologies (semantic databases). In addition, there is another problem. Due to the increase in the amount of information, more memory is needed to store it, but the processing time of such data is significantly increased. There are difficulties for information retrieval (IP). In order to solve this problem, data compression is used in IP problems. This article offers a new model of semantic information retrieval using the method of compression based on the End Tagged Dense Code-ETDC code. To compress the Wiki article, the ETDC method is used, which provides a compression ratio of 25%. To build a terminological dictionary file, only simple texts are needed. Coding – the process is very simple and easy to implement programmatically. Therefore, the encoding and decoding time is shorter than in the Huffman method.

Ключевые слова: помеченный код Хаффмана, ETDC, wordnet, расширение понятия, онтология, сжатие текста, алгоритм Бойера – Мура.

Keywords: marked Huffman code, ETDC, wordnet, concept extension, ontology, text compression, Boyer – Moore algorithm.

Введение

Среди методов сжатия выделяют методы на основе словарей (Ziv-lempel family, gizp) и статистические методы (код Хаффмана, арифметика, PPM и так далее) [Baloul, Abdullah and Babikir, 2013]. В качестве моделей сжатия выделяют 3 модели: статистическую, полустатистическую и динамическую модели. Методы Хаффмана (простой Хаффман и помеченный Хаффман [Arshad, Saleem, Khan, 2016]) и метод окончаний помеченных густых кодов – ОПГК (End Tagged Dense Code-ETDC) включаются в полустатистические методы [Brisaboa, Iglesias, Navarro and Paramá, 2003]. Ziv-lempel метод [Sharma and Gupta, 2017] представляет собой динамическую модель. Полустатистическая модель – это двухпроходный метод (2 pass method), а динамическая модель – это однопроходный метод (1 pass method). Полустатистическая модель обеспечивает хороший коэффициент сжатия. Популярные Ziv-Lempel методы представляют собой динамические методы и обладают хорошим коэффициентом сжатия. Но они реализуются сжатиями текста только с самого начала. Посредством Ziv-lempel метода нельзя реализовать прямой (произвольный) доступ. Поэтому он не подходит к задачам ИП. Коды традиционного метода Хаффмана реализуются на основе букв (символов) и представляют собой последовательности битов. И хотя коэффициент сжатия при этом получается равным 25%, восстановление и поиск занимает много времени [Brisaboa, Nieves, Antonio, Gonzal and José, 2007]. Поэтому Mouretal (2000) предлагает байт-ориентированный метод Хаффмана (простой Хаффман) [Rupanagudi, 2016].

Байт-ориентированный метод Хаффмана

В этом методе последовательность кода – это байты. Поэтому, хотя его коэффициент сжатия получается равным 30%, восстановление и поиск идет быстрее, чем в ранее представленном коде [Fariña, Navarro and Paramá, 2008]. Но его недостаток – это невозможность прямого (произвольного) доступа. Потому что, когда реализуется восстановление, то оно осуществляется с самого начала цельного текста, что не соответствует требованиям ИП. Чтобы решить эту проблему, появился помеченный код Хаффмана, в котором первый бит первого байта принимается равным 1, а первые биты остальных байтов – 0 (таблица 1). В помеченном коде Хаффмана из-за использования первого бита первого байта как флагового разряда (флагового бита) возможен прямой (произвольный) доступ. Благодаря флаговым разрядам обеспечивается возможность распознавать начало кодового слова (в простом коде Хаффмана нет такой возможности). Поэтому в помеченном коде Хаффмана невозможны ошибочные согласования. Кроме того, скорость поиска и восстановления увеличиваются. Код называется помеченным, так как для распознавания начала или окончания кодового слова, для исключения ошибочных согласований используются метки начала или окончания байта. Коэффициент сжатия помеченного кода Хаффмана получается равным 35%, что хуже, чем у простого кода Хаффмана. Поэтому N. Brisaboa, E. Iglesias, G. Navarro описали новый байт-ориентированный код Хаффмана – код End Tagged Dense Code-ETDC [Valencia, Cerdeira, Iglesias and Rodriguez, 2010]. Коэффициент сжатия ETDC (5, 6%) – лучше, чем помеченного кода Хаффмана и хуже (1%) коэффициента сжатия простого кода Хаффмана. Для него не надо строить деревья Хаффмана, что обеспечивает простое и быстрое кодирование. Поэтому время кодирования этого метода сопоставимо с помеченным кодом Хаффмана и на 40% быстрее простого кода Хаффмана [Brisaboa, Nieves, Antonio, Gonzal and José, 2007]. При этом возможен прямой (произвольный) доступ. В сжатом тексте файлы, использующие ETDC, можно искать на основе использования алгоритма boyer moore [Xiong, 2010].

End Tagged Dense Code (ETDC)

Antonio Farina Martinez (2005) предлагает End Tagged Dense Code (ETDC), основой которого является помеченный код Хаффмана. Но ETDC совсем не зависит от кода Хаффмана. Помеченный код Хаффмана определяет первый бит первого байта как флаговый разряд (флаговый бит) «1», но ETDC использует первый бит последнего байта как флаговый бит, первый бит остальных байтов принимается равным «0» «табл. 1». Благодаря этим флаговым битам ETDC становится префиксным кодом, который однозначно декодируется. В префиксных кодах [Xue, Oelmann, 2006] невозможны ошибочные согласования при реализации поиска. Например, допустим, даны 2 кодовых слова A и B, где длина A короче длины B (|A|<|B|). В этой ситуации «A» не может быть префиксом «B».

Таблица 1
Table 1

Кодовые слово ОПГК
Code words of Tagged Huffman (TH) and ETDC

байты	Помеченный Хаффман	ОПГК
1	1xxxxxxx	1xxxxxxx
2	1xxxxxxx 0xxxxxxx	0xxxxxxx1xxxxxxx
3	1xxxxxxx 0xxxxxxx0xxxxxxx	0xxxxxxx0xxxxxxx 1xxxxxxx
……		
n	1xxxxxxx0xxxxxxx..0xxxxxxx	0xxxxxxx….0xxxxxxx 1xxxxxxx

Таблица 2
Table 2

Кодовые слова формат ПФ и ОПГК
Code words format Tagged Huffman (TH) and ETDC

Ранг слова	Назначенное ключевое слово	байт	слова
0 … $2^7 - 1 = 127$	10000000 ……. 11111111	1 ….. 1	2^7
2^7=128 …. 256 … $2^7 2^7$+2^7-1=16511	00000000:10000000 ……. 00000001:10000000 …….. 01111111:11111111	2 …. 2 …… 2	$2^7 2^7$
$2^7 2^7$+2^7=16512 …… $(2^7)^3$+$(2^7)^2 + 2^7$-1=2113663	00000000:00000000:10000000 …… 01111111:01111111: 11111111	3 …. 3	$(2^7)^3$
…			

В начале ETDC [Öztürk, Mesut and Diri, 2017] осуществляет определение частоты слов и получает файл, в котором находятся отсортированные по частоте слова (терминологический словарь). Затем определяются коды для этих слов. Для осуществления кодирования используют не фактическую частоту слов, а только ранги (индексы) слов по этой частоте. На втором шаге осуществляется сжатие (замена исходных символов их кодами). На основе учета исходных символов с убывающими вероятностями $\{P_i\} 0 \leq i < n$, соответствующими кодовым словам, используется метод ETDC, и в результате формируется последовательность символов из b бит, из которых образуются числа в базе 2^{b-1} (то есть от

0 до $2^{b-1}-1$), за исключением последней, которая имеет значение между 2^{b-1} и 2^b-1, при этом назначение выполняется последовательным образом. В частности, если k число байтов в каждом кодовом слове (k ≥1), тогда:

$$2^{b-1}\frac{2^{(b-1)(k-1)}-1}{2^{b-1}-1} \le 2^{b-1}\frac{2^{(b-1)k}-1}{2^{b-1}-1} \quad (1)$$

где *b*=8.

На самом деле значение b может быть любым, а не только 8.

По этой формуле (1), когда

$k = 1{,}0 \le i < 128$, или $k = 2, 128 \le i < 16512$, или $k = 3{,}16512 \le i < 2113664$ и т.д.

Эта формула считает только ранг слов «i», а не кодовые слова «i».

Кодирование

Кодовое слово (x) рассчитывается по следующему алгоритму кодирования.

Кодирование (i)

1) //вход ранг слова (i) $0 \le i \le n-1$
2) output ((imod 128)+128);//самый правый байт
3) i ← i div 128;
4) while i > 0 //остальные байты
5) i← i-1;
6) output (i mod 128);
7) i ← i div 128;

В этом алгоритме output – это кодовое слов «x», а «n» – это количество «i». Этот алгоритм производит однокодовое слово как последовательность байт слева направо и производит самый первый значимый байт (самый правый байт с началом «1»). Для построения файла терминологического словаря нужны только простые тексты, причем строить кодовое дерево не нужно. Однако в каноническом коде Хаффмана необходимо построить кодовое дерево для кодирования. Поэтому для построения файла терминологического словаря кода ETDC требуется меньше времени, чем реализация метода Хаффмана. В сжатых файлах существуют только кодовые слова «Рис. 1».

Декодирование

Декодирование реализуется в 2 этапа. На первом осуществляется загрузка файла терминологического словаря, который отсортирован по частоте. На втором получают кодовое слово (x) в сжатом файле, которое совпадает с рангом слова (i) файла терминологического словаря. Оценка времени декодирования для кода (x) байта (K) будет $O(K) = O(\log x)$ [Zhang, Deng and Li, 2009]. Чтобы получить ранг слова «i» выполняется итерация по каждому байту кодовых слов (x). Так как первый бит последнего байта равен «1», то его значение больше чем 127. Этот помеченный бит «1» указывает на окончание кодового слова. После завершения итерации получается ранг слова «i». Если найти соответствующее слово [i] в файле терминологического словаря, то декодирование завершится. Например, рассмотрим трехбайтовый код $x = x_1x_2x_3$. Чтобы получить ранг слова (i), нужно произвести вычисления по следующей формуле (2):

$$i = \left(\left((x_1 * 128) + x_2\right) * 128\right) + (x_3 - 128) + base[k] \quad (2)$$

где k=3.

Допустим, $x_1 = 1, x_2 = 2, x_3 = 255$.

Итак: x=1,2,255. base [3] вычисляется по следующей формуле (3):

$$base[3] = \left(\sum_{j=1}^{3} 128^{j-1}\right) - 1 \quad (3)$$

итак, i=(((1*128)+2)*128)+(255-128)+(16512)= 33279.

В файле терминологического словаря находим соответствующее слово с рангом 33279. Таким образом, декодирование завершено. Алгоритм декодирования имеет следующий вид:

Декодирование(base,x)

1) //вход x: кодовое слов для декодирования
2) //выход i: позиция декодированного слова в файле терминологического словаря
3) i← 0;
4) k ← 1;// Байт кодового слова
5) while x[k] <128
6) i← i*128 + x[k];
7) k ← k+1;
8) i ←i* 128 + x[k]-128;
9) i ← i+base[k];
10) return i;

Сжатый файл представляет собой соединение кодовых слов. В этом файле, чтобы осуществлять поиск по принципу соответствия образцу, надо распознавать их с самого начала или определять окончания кодовых слов. Если не распознавать их при реализации поиска по принципу соответствия образцу, то появляются ошибочные согласования. В простом коде Хаффмана невозможно определить начала или окончания соединенных кодовых слов. Поэтому, когда осуществляется поиск, появляются ошибочные согласования. В ETDC определяется «1» первый бит последнего байта каждого кодового слова как флаговый бит. Из-за флагового бита легко распознавать окончание кодовых слов. Поэтому при использовании ETDC невозможны ошибочные согласования. В сжатом файле при осуществлении поиска по принципу соответствия образцу реализуется кодирование образца.

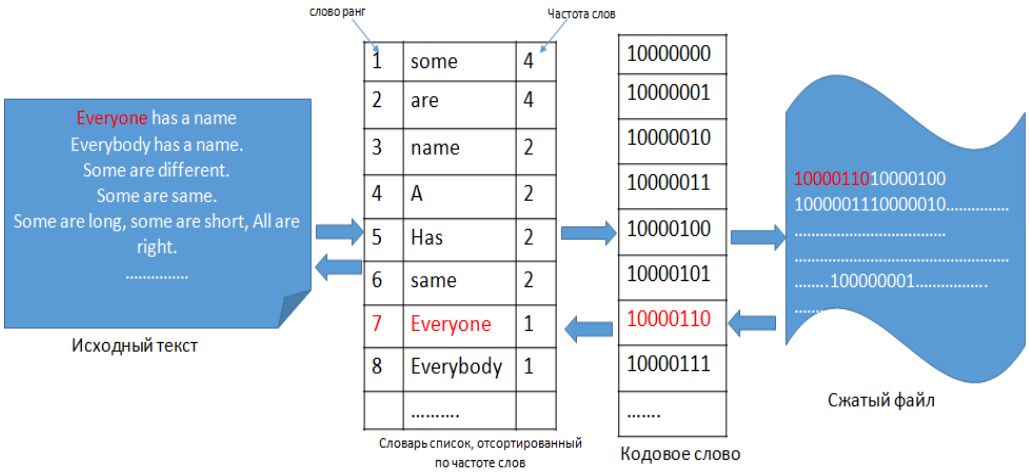

Рис. 1. Процесс кодирования и декодирования ОПГК (ETDC)
Fig. 1. The process of encoding and decoding of the ETDC

Когда появляется полное кодовое слово образца, совпадающее с кодовым словом сжатого текста, то необходимо проверять, имеет ли первый согласованный байт значение в диапазоне $[2^{b-1}, 2^b - 1]$. Это позволяет проверить последний байт предыдущего кодового слова. Здесь основание b=8, значение в диапазоне предыдущего кодового слова – [128,255]. Если диапазон предыдущего кодового слова между [128] и [255], то его первый бит – «1». Посредством этого текущий согласованный байт обеспечивает правильное согласование. А если предыдущее кодовое слово полного кодового слова образца лежит в диапазоне $[0, 2^{b-1} - 1]$ (значит – 0,127), то этот текущий согласованный байт только часть одного кодового слова и будет ошибочное согласование. В такой ситуации кодовые слова сжатого текста длиннее, чем кодовые слова образца (рис. 2). В этой статье предлагается новая модель семантического информационного поиска.

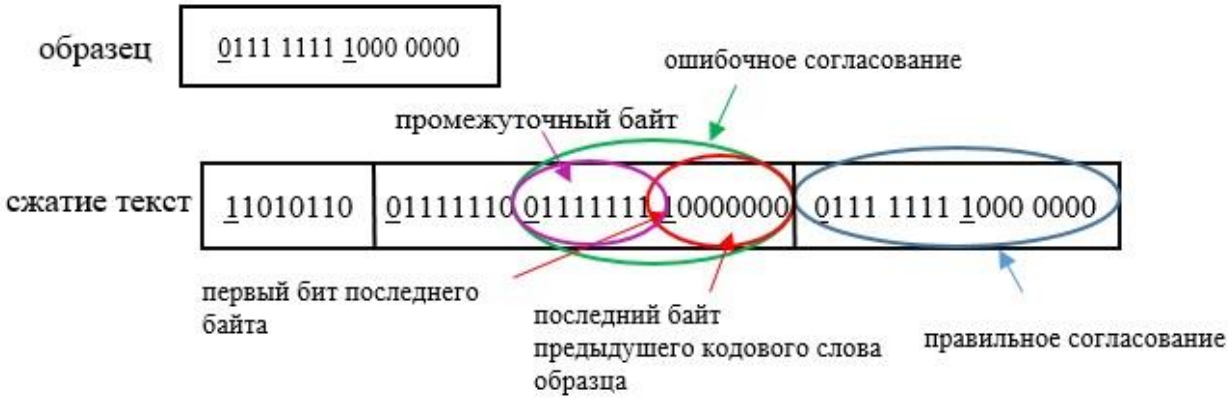

Рис. 2. Поиск по принципу соответствия образцу в ETDC
Fig. 2. Search according to the pattern matching principle in ETDC

Предложенная модель

Предложенная модель реализуется за 5 шагов (рис. 3).
(1) Расширение запроса понятия через Wordnet.
(2) Сжатие онтологии ETDC.
(3) Поиск в сжатом тексте с алгоритмом BoyerMoore.
(4) Декодирование.
(5) Отыскание документов.

Пользователю, чтобы получить соответствующие результаты, необходимо осуществлять расширение понятия запроса. При этом Wordnet [Samhith, Tilak and Panda, 2016] – это всеобъемлющий тезаурус для получения семантической связанности между понятиями и расширениями [Лин, Каунг, 2018]. На этапе предварительной обработки расширения запроса устраняются слова, игнорируемые информационно-поисковой системой, и знаки препинания из запросов, а запрос присоединяется к индивидуальным словам. С помощью <word, POS, POS_NUM>, где POS – часть речи и POS_NUM – номер части речи, определяются эти индивидные слова. У слов в сансете имеются семантические связанности с понятиями запросов. Слова в сансете определяются как двухмерный массив. В каждой строке массива определяются понятия одного сансет [Gadge, Sane, Kekre, 2013]. Таким образом, получаются семантически подобные наборы термов, схожие с понятиями одного терма запроса. Расширенные термы запроса для каждой строки массива объединяются в предложения. Семантические значения этих предложений при случайных объединениях являются не подходящими. Чтобы избежать этого, необходимо фильтровать расширенные запросы. Фильтрация похожа на расширение понятий запросов. Получающиеся термы-кандидаты объединяются в тройки <word,POS,POS_NUM> и находят сансеты в Wordnet, которые соответствуют этим тройкам.

Термы Wordnet сансета определяются как двухмерный массив. Исходное понятие двухмерного массива сравнивается с текущими понятиями двухмерного массива, а расширяются только совпадения с исходными понятиями двухмерного массива. На рис. 4 пунктирная линия – это запрос-кандидат, а сплошная линия – это финальный запрос. N термы исходных запросов соответствуют c^{ith}термам слова сансета. WSD – устранение семантической неоднозначности слов. Wikipedia имеет иерархическую структуру и, из-за подобия заголовка статьи Wiki и понятия онтологии, возможно использовать Wiki как онтологию [Лин, 2017].

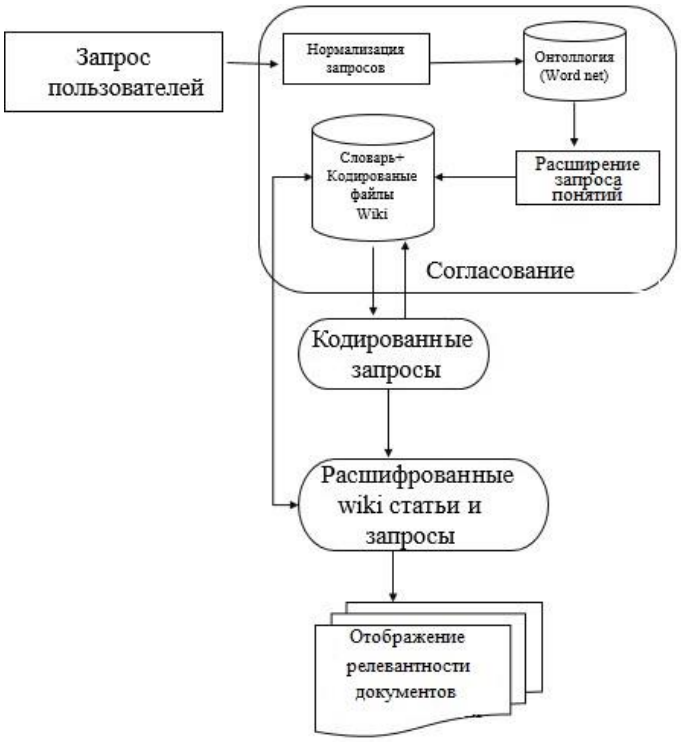

Рис. 3. Семантический информационный поиск с сжатием
Fig. 3. Semantic Information Search with Compression

Сжатие статьи Wikipedia осуществляется методом ETDC. На первом этапе определяется частота слов Wiki-статьи и создается файл терминологического словаря. В этом файле находятся только простые слова, отсортированные по частоте. На этом этапе кодирования определяется ранг слова, осуществляется их сортировка по частоте и образуются кодовые слова.

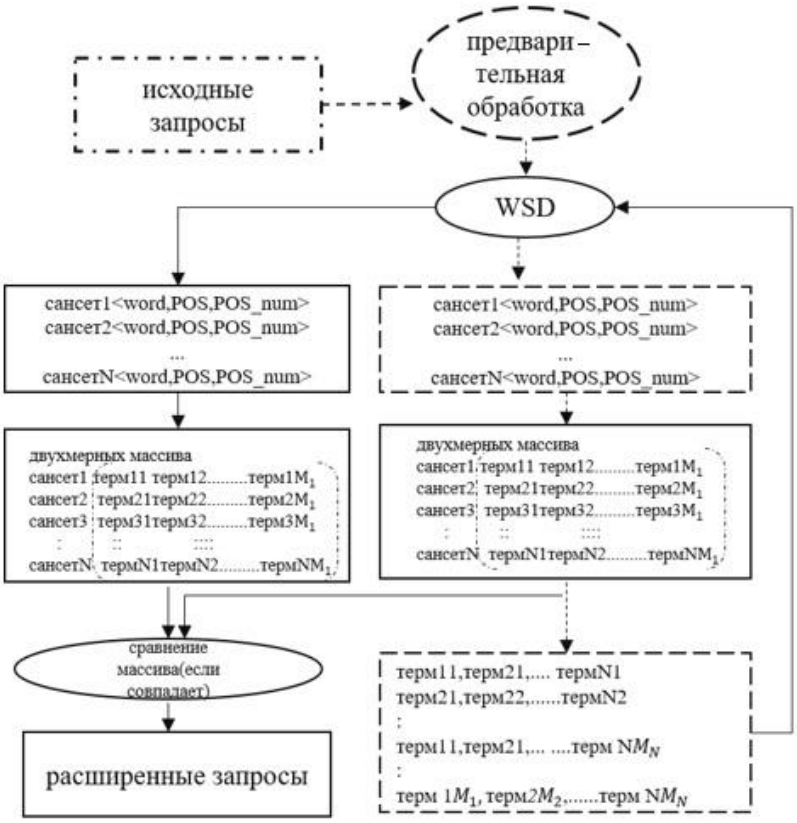

Рис. 4. Расширение запросов через Wordnet
Fig. 4. Guery expansion via Wordnet

На втором этапе термы статей Wiki заменяются получающими кодовыми словами. Таким образом получаются сжатые файлы, в которых находятся только связанные кодовые слова. Запросы пользователей расширяются через Wordnet. Получающиеся расширенные запросы согласовываются с термами файла терминологического словаря и осуществляется кодирование с использованием ETDC. Эти кодовые слова запросов согласовываются с кодовыми словами сжатых файлов с использованием алгоритма Boyre Moore. Осуществляется декодирование кодовых слов и поиск документов, которые включают кодовые слова, соответствующие кодовым словам запросов.

Экспериментальные результаты

На основе описанных моделей были разработаны алгоритмы и осуществлена их программная реализация. Результаты экспериментальной проверки разработанного программного комплекса представлены на рисунках и в таблицах.

Таблица 2а
Table 2a

Сравнение коэффициента сжатия английских wiki-статей при использовании BZip2, LZW и ETDC. Размеры словаря исключены. (СК – коэффициент сжатия, О об – оригинальный объем)
Comparison of the compression ratio of Russian wiki articles and English wiki articles using BZip2, LZW and ETDC. The size of the dictionary is excluded. (CC – Coefficient of compression, O V – Original volume)

Wiki-статья (название статьи)	О Об (кб)	Bizp2 (кб)	LZW (кб)	ETDC (KB)	LZW (СК%)	Bizp2 (СК%)	ETDC (СК%)
Chief of Naval Operations	14	5	6.79	4	48	35	28
Demographics of American Samoa	5	2	2.65	1.07	53	40	21
Sienna Guillory	31	11	15.7	7.8	50	35	25
February	25.3	10	12.9	5	50	40	20
Anselm of Laon	7	4	4	2	57	57	28
Land reform	24	11	12.9	5.7	53	45	23
Dan Simmons	18	7	9.16	5	50	38	27

Таблица 2б
Table 2b

Сравнение коэффициента сжатия английских wiki-статей при использовании BZip2, LZW и ETDC. Размеры словаря исключены. (СК – коэффициент сжатия, О об – оригинальный объем)
Comparison of the compression ratio of Russian wiki articles and English wiki articles using BZip2, LZW and ETDC. The size of the dictionary is excluded. (CC – Coefficient of compression)

Wiki-статья По-русски (название статьи)	О Об(KB)	Bizp2 (кб)	LZW (кб)	ETDC (кб)	LZW (СК%)	Bizp2 (СК%)	ETDC (СК%)
Русский	283	42	78	34	27	14	12.8
Москва	216	42	73	32	33	19	14
Россия	407	78	134	59	32	20.8	14
Москва и Москвичи (Гиляровский)	329	66	106	51	32	20	15
Война и мир (Л. Толстой) / Том III	104	21	36	17	34	20	16

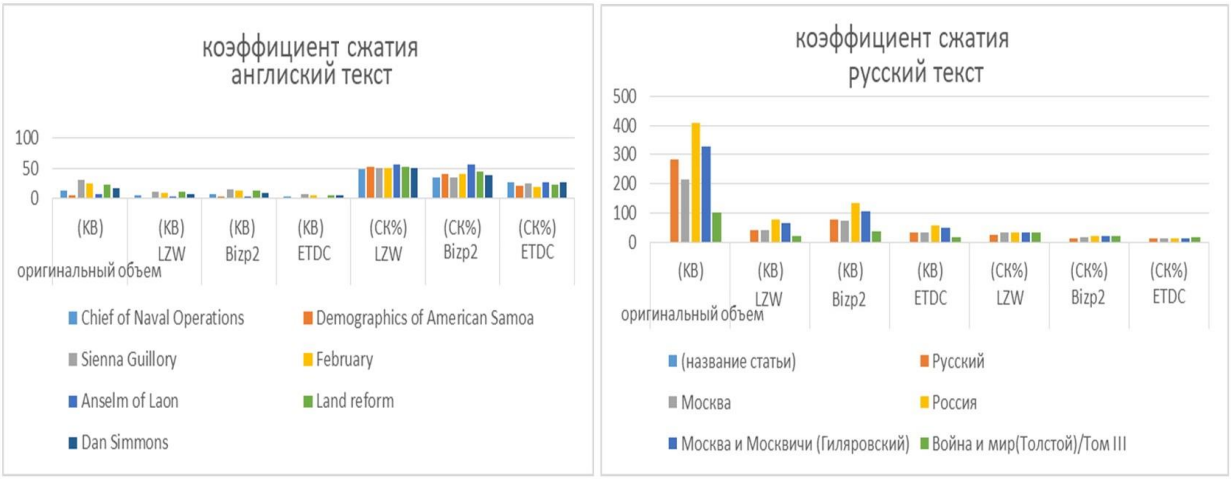

Рис. 5. Сравнение коэффициента сжатия при использовании BZip2, LZW и ETDC.
Размеры словаря исключены. (СК – коэффициент сжатия)
Fig. 5. Comparison of the compression ratio when using BZip2, LZW and ETDC.
The size of the dictionary is excluded. (CC – Coefficient of Compression)

Таблица 3а
Table 3a

Скорость кодирования английского текста при использовании BZip2, LZW и ETDC
Coding speed of English text when using BZip2, LZW and ETDC

Wiki-статья (название статьи)	объем	Bizp2	LZW	ETDC
Chief of Naval Operations	14	0.001	5	0.078
Demographics of American Samoa	5	0.001	2	0.06
Sienna Guillory	31	0.001	23	0.11
February	25.3	0.01	18	0.10
Anselm of Laon	7	0.001	2	0.05
Land reform	24	0.01	18	0.09
Dan Simmons	18	0.01	17	0.09

Таблица 3б
Table 3b

Скорость кодирования русского текста при использовании BZip2, LZW и ETDC
Coding speed of English text when using BZip2, LZW and ETDC

Wiki-статья (название статьи)	объем	Bizp2 (s)	LZW (s)	ETDC (s)
Русский	283	0.2	1.8	1
Москва	216	0.15	2.5	0.99
Россия	407	0.4	5.6	3,36
Москва и Москвичи (Гиляровский)	329	0.3	3.8	2,79
Война и мир (Л. Толстой) /Том III	104	0.1	1.4	0.26

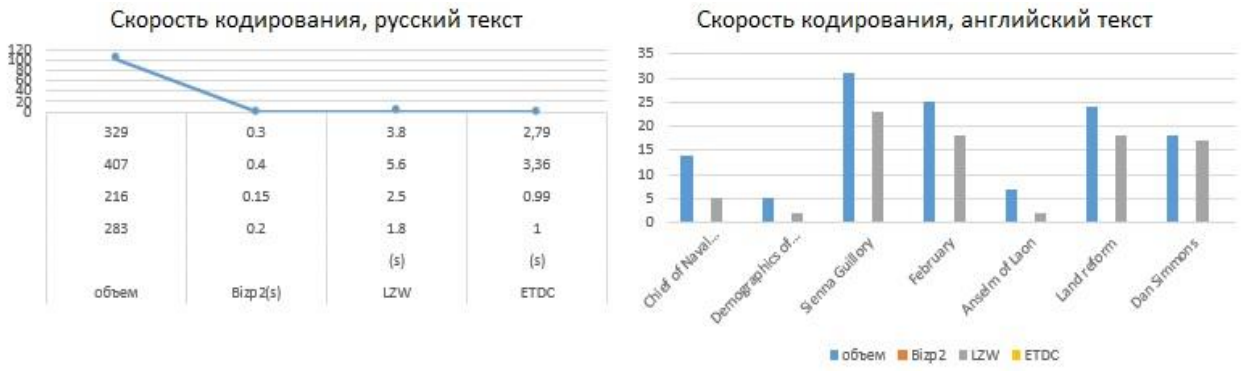

Рис. 6. Скорость кодирования русского текста при использовании BZip2, LZW и ETDC
Fig. 6. Soding speed of the Russian text when using BZip2, LZW and ETDC

Таблица 4
Table 4

Скорость декодирования английского текста при использовании BZip2, LZW и ETDC
Decoding speed of english text when using BZip2, LZW and ETDC

Wiki-статья (название статьи)	Объем (КБ)	Bizp2 (c)	LZW (c)	ETDC (c)
Chief of Naval Operations	14	0.001	5	0.078
Demographics of American Samoa	5	0.001	2	0.06
Sienna Guillory	31	0.001	23	0.11
February	25.3	0.01	18	0.10
Anselm of Laon	7	0.001	2	0.05
Land reform	24	0.01	18	0.09
Dan Simmons	18	0.01	17	0.09

Таблица 5
Table 5

Скорость декодирования русского текста при использовании BZip2, LZW и ETDC
Decoding speed of Russian text when using BZip2, LZW and ETDC

Wiki-статья (название статьи)	объем	Bizp2 (s)	LZW (s)	ETDC (s)
Русский	283	0.2	0.1	0.3
Москва	216	0.1	0.007	0.3
Россия	407	0.4	0.2	0.49
Москва и Москвичи (Гиляровский)	329	0.15	0.2	0.28
Война и мир (Л. Толстой)/Том III	104	0.005	0.08	0.089

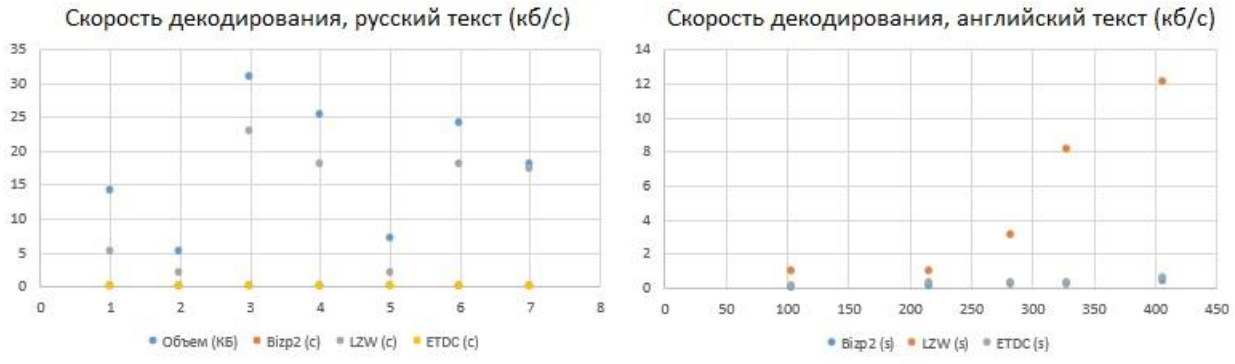

Рис. 7. Скорость декодирования при использовании BZip2, LZW и ETDC
Fig. 7. Decoding speed when using BZip2, LZW and ETDC

Таблица 5а
Table 5a

Скорость поиска в русском сжатом тексте
Speed of search in russian compressed text

Поисковый запрос (объем текста – 1188 кб, количество слов 93038)	Найденное количество ответов	Время поиска (s)
Москва	378	0,25
дума	13	0.2
годы	51	0.14
Толстой	2	0.16
География России	77	0.28
Член совета	22	0,52

Таблица 5б
Table 5b

Скорость поиска в английском сжатом тексте
Speed of search in English compressed text

Поисковый запрос (объем текста – 398 кб, количество слов 51316)	Найденное количество ответов	Время поиска (s)
broadcast	11	0.068
February	25	0.20
naval	43	0.05
germany	21	0.10
Nordic Council	129	0.16
Arm forced	91	0,21

Заключение

Для сжатия статьи Wiki используется метод ETDC, что обеспечивает получение коэффициента сжатия 25%. Для построения файла терминологического словаря необходимы только простые тексты. Кодирование – процесс очень простой и легко реализуется программно. Поэтому время кодирования и декодирования меньше, чем в методе Хаффмана. Главное преимущество метода ETDC – возможность прямого (произвольного) доступа и поиск в сжатом тексте, что невозможно при использовании простых методов Хаффмана и других методов сжатия на основе словарей (Zim-lempel family [Sharma and Gupta, 2017], gizp). Поэтому метод сжатия на основе ETDC возможно использовать для решения задач информационного поиска.

Список литературы
References

1. A. Fariña, G. Navarro and J.R. Paramá, 2008, Word-Based Statistical Compressors as Natural Language Compression Boosters, Data Compression Conference (dcc 2008), Snowbird, UT, 2008, 162–171. doi: 10.1109/DCC.2008.14.

2. Brisaboa N.R., Fariña A., Navarro G., Paramá J.R., 2004, Simple, Fast, and Efficient Natural Language Adaptive Compression. In: Apostolico A., Melucci M. (eds) String Processing and Information Retrieval. SPIRE 2004. Lecture Notes in Computer Science, vol. 3246. Springer, Berlin, Heidelberg.

3. Brisaboa N.R., Iglesias E.L., Navarro G., Paramá J.R. 2003, An Efficient Compression Code for Text Databases. In: Sebastiani F. (eds) Advances in Information Retrieval. ECIR 2003. Lecture Notes in Computer Science, vol. 2633. Springer, Berlin, Heidelberg.

4. Brisaboa, Nieves & Fariña, Antonio & Navarro, Gonzalo & Paramá, José. (2007). Lightweight natural language text compression. Inf. Retr. 10: 1–33. 10.1007/s10791-006-9001-9.

5. F.M. Baloul, M.H. Abdullah and E.A. Babikir, 2013, "ETAOSD: Static dictionary-based transformation method for text compression" 2013, International Conference On Computing, Electrical And Electronic Engineering (ICCEEE), Khartoum, 2013, 384–389.

6. J. Zhang, B. Deng and X. Li, 2009. Concept Based Query Expansion Using Word Net. International e-Conference on Advanced Science and Technology, Dajeon, 2009, 52–55.

7. K. Sharma and K. Gupta, 2017. Lossless data compression techniques and their performance, 2017, International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, 256–261.

8. Нэй Лин, 2017, Кластеризация документов на основе онтологии. Современная наука: проблемы и практики, естественные и технические науки. Царегородцев А.В. 38–42.

Nay Lin, 2017, Klasterizaciya dokumentov na osnove ontologii. Sovremennaya nauka: problemy i praktiki, estestvennye i tekhnicheskie nauki. Caregorodcev A.V. 38–42.

9. Нэй Лин, Каунг Мьят Хту, 2018. Устранение семантической неоднозначности слов. Формирование семантических отношений между текстами на основе использования Wiki and Word Net. Инновации инвестиции. Абдикеев Н.М. 155–160.

Nay Lin, Kaung M'yat Htu, 2018. Ustranenie semanticheskoy neodnoznachnosti slov. Formirovanie semanticheskih otnosheniy mezhdu tekstami na osnove ispol'zovaniya Wiki and Word Net. Innovacii investicii. Abdikeev N.M. 155–160.

10. Xiong, 2010. A Composite Boyer-Moore Algorithm for the String Matching Problem, 2010 International Conference on Parallel and Distributed Computing, Applications and Technologies, Wuhan, 492–496.

11. Sudhir Rao Rupanagudi, Varsha G. Bhat, R Anushree, Y.N. Kavitha, R. Meghana, Arpitha Srinath, J. Pooja, Ramya R. Pai, M. Roopashree, Sandhya Ramesh, C. Anil, S. Anuradha, Arpita V. Murthy, S. Sridevi, 2016. A novel and highly secure encryption methodology using a combination of AES and visual cryptography, Advances in Computing Communications and Informatics (ICACCI) 2016 International Conference. 1682–1688.

12. Rabia Arshad, Adeel Saleem, Danista Khan, 2016. Performance comparison of Huffman Coding and Double Huffman Coding. Sixth International Conference on Innovative Computing Technology (INTECH), 361–364.

13. Tito Valencia; Lorena O. Cerdeira; Eva L. Iglesias; Francisco J. Rodríguez, 2010. Translation table compression under End-Tagged Dense Code, 4th International Universal Communication Symposium. 18–19 Oct. 2010.

14. Shang Xue, B. Oelmann, 2006. Unary prefixed Huffman coding for a group of quantized generalized Gaussian sources, IEEE Transactions on Communications. July 2006. 54(7): 1164–1169.

15. Jayant R. Gadge, S.S. Sane, H.B. Kekre, 2013. Query expansion using WordNet in N-layer vector space model, Nirma University International Conference on Engineering (NUiCONE).

16. K. Samhith, S.A. Tilak and G. Panda, 2016. Word sense disambiguation using WordNet Lexical Categories, International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), Paralakhemundi, 1664–1666.

17. D. Dath and J.V. Panicker, 2017. Enhancing adaptive huffman coding through word by word compression for textual data. International Conference on Communication and Signal Processing (ICCSP), Chennai, 1048–1051.

18. E. Öztürk, A. Mesut and B. Diri, 2017. Multi-stream word-based compression algorithm. International Conference on Computer Science and Engineering (UBMK), Antalya, 34–37.

19. S.T. Klein and D. Shapira, 2002. Searching in compressed dictionaries. Proceedings DCC Data Compression Conference, Snowbird, UT, USA, 142–151.

20. J. Dvorsky, J. Pokorny and V. Snasel, 1999. Word-based compression methods for large text documents, Proceedings DCC'99 Data Compression Conference (Cat. No. PR00096), Snowbird, UT, USA, 523.