PASCAL Journal of Computer Science and Informatics
https://jurnal.devitara.or.id/index.php/komputer
E-ISSN: 3047-5074 Volume 1 Nomor 1 Tahun 2023

Comparison of BWT and Sequitur Algorithms for Improving the Compression Ratio of Run-Length Encoding

Rian Syahputra
Universitas Budi Darma, Medan, Indonesia

ARTICLE INFO

Keywords:
text compression;
Run-Length Encoding;
Burrows-Wheeler Transform;
Sequitur;
compression ratio

Email:
ryansyah93@gmail.com

ABSTRACT

Run-Length Encoding (RLE) is effective when identical symbols occur consecutively, a condition that is not consistently present in text. This study compares the Burrows-Wheeler Transform (BWT) and Sequitur as preprocessing mechanisms intended to improve the RLE compression ratio. Three processing paths were examined: direct RLE, BWT followed by RLE, and Sequitur followed by RLE. The evaluation focuses on symbol-length changes at the RLE stage and reports the output-to-input ratio and space saving. Direct RLE on "compression" retained 11 symbols. BWT rearranged the string to "nrsoocimpse", but the RLE result also contained 11 symbols. Sequitur produced a grammar-based representation of "compression compass"; however, the visible sequence submitted to RLE remained unchanged after encoding. Accordingly, all tested RLE stages produced a 100% ratio and 0% space saving. The findings show that preprocessing benefits RLE only when it creates sufficiently long adjacent runs. The experiment is limited to short samples and does not include transform metadata or grammar-dictionary overhead.

Copyright © 2023 PASCAL.
All rights reserved is Licensed under a Creative Commons Attribution- NonCommercial 4.0 International License (CC BY-NC 4.0)

INTRODUCTION

The continuous growth of digital documents increases storage and transmission requirements, making data compression an important component of information processing. Lossless compression reduces the representation size while preserving the original information, which is essential when exact reconstruction is required. Run-Length Encoding (RLE) is one of the simplest lossless techniques because it replaces repeated adjacent symbols with a compact run representation (Salomon, 2007; Saragih & Utomo, 2020).

RLE performs well when the input contains long sequences of identical values. This pattern is common in some images and structured data, but it is less predictable in ordinary text, where characters typically alternate rather than occur in long consecutive runs (Jalaludin & Melita, 2012; Mansyuri, 2021). Consequently, applying RLE directly to a short text string may preserve the original length or may even introduce representation overhead when the run marker and run count occupy additional symbols.

A preprocessing transformation can be used to reorganize redundancy before RLE is applied. The Burrows-Wheeler Transform (BWT) is a reversible block-sorting transformation that permutes the input so that equal symbols tend to become locally clustered, a property that can improve subsequent compression stages (Burrows & Wheeler, 1994; Mahendra et al., 2016). Sequitur takes a different approach: it detects repeated digrams and constructs a compact context-free grammar under the constraints of digram uniqueness and rule utility (Nevill-Manning & Witten, 1997; Darnita et al., 2019; Buulolo, 2017).

Although BWT and Sequitur both reorganize redundancy, they do not necessarily create the same type of local repetition required by RLE. This study therefore compares BWT-RLE and Sequitur-RLE on short text samples, with direct RLE used as a baseline. The contribution is a focused examination of whether each transformation actually creates adjacent runs that reduce the RLE-stage representation, rather than assuming that any prior transformation will automatically improve compression. The comparison is expressed through the observed symbol length, compression ratio, and space saving at the RLE stage.

METHODS

The study uses an experimental comparison of three processing paths: direct RLE, BWT followed by RLE, and Sequitur followed by RLE. The primary sample for direct RLE and BWT-RLE is the text "compression". For Sequitur-RLE, the text "compression compass" is used because it contains repeated

symbol pairs that allow grammar rules to be formed. The purpose is not to evaluate a production-scale compressor, but to observe how each preprocessing method changes the local symbol pattern received by RLE.

The experimental procedure consists of the following steps:

1. Review the principles of text compression, RLE, BWT, and Sequitur.
2. Select the text sample used in each processing path.
3. Apply RLE directly to the baseline sample.
4. Transform the text with BWT or Sequitur and then apply RLE to the transformed representation.
5. Compare the symbol length before and after the RLE stage and calculate the compression ratio and space saving.

For consistency, the compression ratio (CR) in this study is defined as CR = (Lout / Lin) × 100%, where Lin is the number of symbols entering the RLE stage and Lout is the number of symbols produced by RLE. Space saving (SS) is calculated as SS = [1 − (Lout / Lin)] × 100%. Therefore, CR = 100% and SS = 0% indicate that RLE does not reduce the representation. For the Sequitur path, these metrics isolate the RLE stage only; grammar rules required for reconstruction are not counted as compressed payload. The reported values should therefore not be interpreted as complete end-to-end codec ratios.

Sequitur Algorithm

Sequitur incrementally infers a context-free grammar from a symbol sequence by replacing repeated digrams with nonterminal rules. Two constraints control the process. Digram uniqueness prevents the same adjacent symbol pair from appearing more than once in the grammar, while rule utility requires each production rule to be used more than once; otherwise, the rule is expanded and removed. These mechanisms capture repeated phrases hierarchically rather than merely grouping identical adjacent characters (Nevill-Manning & Witten, 1997; Darnita et al., 2019; Buulolo, 2017; Siahaan, 2019).

Burrows-Wheeler Transform (BWT)

BWT converts a block of text into a reversible permutation containing the same symbols in a different order. Conceptually, all cyclic rotations are formed, the rotations are sorted lexicographically, and the final column is taken as the transformed sequence. The transformation itself does not remove information; its value lies in increasing local symbol clustering so that a subsequent compressor can exploit the resulting regularity (Burrows & Wheeler, 1994; Mahendra et al., 2016).

Run-Length Encoding (RLE)

RLE represents a run of identical consecutive values using the value and information about the run length. The method is advantageous only when the run is long enough for the encoded form to be shorter than the literal symbols it replaces. For text with few or short repeated runs, RLE may yield no reduction because the count marker also consumes representation space (Salomon, 2007; Jalaludin & Melita, 2012; Mansyuri, 2021).

RESULTS AND DISCUSSION

Direct RLE Test

The text “compression” was first processed using the RLE notation adopted in the experiment. The resulting representation is: **compression → compre1sion**

Both the input and the RLE output contain 11 symbols. Thus, the direct RLE path produces CR = 100% and SS = 0%. The sample contains only a short repeated region, so replacing that run with the count notation does not decrease the total symbol length. This baseline confirms that the selected text does not naturally provide the long adjacent repetitions on which RLE depends.

BWT-RLE Test

BWT rearranges “compression” into “nrsoocimpse”. This transformation places two occurrences of “o” next to each other, creating a local run that was not present in the same position in the original sequence. Applying RLE produces “nrs1ocimpse”. Nevertheless, the RLE input and output each contain 11 symbols, so

the measured CR remains 100% and SS remains 0%. The result is important for the title objective: BWT does improve local clustering in this sample, but the created run is too short to offset the symbols required by the RLE notation. In other words, clustering alone is insufficient; the run length must also exceed the representation overhead.

Sequitur-RLE Test

For the Sequitur path, "compression compass" is used so that repeated digrams can be detected. The pair "co" is first replaced by a nonterminal, followed by additional substitutions as repeated pairs are identified. After rule-utility simplification, the visible transformed sequence becomes "CreDion CaD". This sequence illustrates the main difference between Sequitur and BWT: Sequitur captures repeated phrases through grammar rules, whereas BWT primarily reorders symbols to alter their local adjacency.

When "CreDion CaD" is passed to RLE, no adjacent identical characters are available for shortening and the output remains "CreDion CaD". At the RLE stage, Lin = 11 and Lout = 11, giving CR = 100% and SS = 0%. Although the visible Sequitur sequence is shorter than the original phrase, it would be methodologically incorrect to treat that difference alone as the final compression gain because the generated grammar rules are also needed to reconstruct the source. This study therefore evaluates only whether Sequitur creates an RLE-friendly sequence, and on the tested sample it does not.

Table 1. Comparison of RLE-Stage Results

Processing Path	Before RLE	After RLE	CR / SS
Direct RLE	compression	compre1sion	100% / 0%
BWT-RLE	nrsoocimpse	nrs1ocimpse	100% / 0%
Sequitur-RLE	CreDion CaD	CreDion CaD	100% / 0%

Comparative Analysis

Table 1 shows an identical RLE-stage outcome for all three paths: the output-to-input ratio is 100% and the space saving is 0%. However, the mechanisms leading to this outcome are different. Direct RLE receives text with almost no exploitable runs. BWT changes symbol order and successfully creates one adjacent repetition, but the run is only two symbols long and therefore does not reduce the representation under the notation used in the experiment. Sequitur identifies repeated digrams and reorganizes them as grammar rules, but the resulting visible sequence does not contain adjacent identical symbols for RLE to exploit.

These observations explain why neither preprocessing method improves the measured RLE ratio on the tested samples. BWT is structurally closer to the requirement of RLE because its block-sorting behavior can place equal characters together; nevertheless, the benefit depends on the number and length of the runs produced. Sequitur targets a different redundancy structure: repeated phrases can be represented by nonterminals even when identical characters are not adjacent. As a result, a grammar that is compact in its own representation does not automatically become a sequence that RLE can further shorten. Therefore, the effectiveness of a transformation should be judged not only by whether it changes the data, but by whether the transformed pattern matches the operating principle of the next compression stage.

The result also clarifies the meaning of "improving the compression ratio" in this study. Improvement would require Lout < Lin after RLE, which would produce CR < 100% and SS > 0%. None of the three tested paths satisfies this condition. The experiment therefore provides a negative but informative result: preprocessing with BWT or Sequitur is not sufficient by itself to guarantee a better RLE ratio on short text. This interpretation is consistent with the characteristics of BWT as a reversible clustering transform and Sequitur as a grammar-inference method, while RLE remains specifically dependent on adjacent runs (Burrows & Wheeler, 1994; Nevill-Manning & Witten, 1997; Salomon, 2007).

Methodological Implications and Limitations

The comparison is intentionally small and should be interpreted as a proof-of-concept rather than a general performance benchmark. Only short text samples are evaluated, no byte-level storage format is implemented, and the additional information required for inverse BWT or Sequitur grammar reconstruction is excluded from the reported RLE-stage lengths. These limitations prevent broad claims about overall codec

efficiency. A stronger evaluation should use multiple files with different entropy and repetition characteristics, report byte-level compressed size, include transform overhead, verify lossless reconstruction, and compare execution time as well as compression ratio. Such an evaluation would determine whether the local behavior observed here persists on realistic datasets.

CONCLUSION

The comparison demonstrates that BWT and Sequitur change text redundancy in different ways, but neither transformation improves the RLE-stage compression ratio for the tested samples. Direct RLE, BWT-RLE, and Sequitur-RLE each produce CR = 100% and SS = 0%. BWT creates a short adjacent run, yet the run is not long enough to overcome the RLE representation overhead. Sequitur captures repeated digrams through grammar rules, but its transformed sequence does not create adjacent identical symbols for RLE. The findings therefore show that preprocessing can improve RLE only when the resulting structure is compatible with run-based encoding. Future work should evaluate longer and more diverse text datasets, include complete transform metadata and grammar overhead, verify decompression, and measure byte-level compression ratio and execution time.

REFERENCES

Darnita, Y., Khairunnisyah, K., & Mubarak, H. (2019). Kompresi data teks dengan menggunakan algoritma Sequitur. Sistemasi, 8(1), 104. https://doi.org/10.32520/stmsi.v8i1.429

Saragih, S. R., & Utomo, D. P. (2020). Penarapan algoritma Prefix Code dalam kompresi data teks. KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer), 4(1), 249–252. https://doi.org/10.30865/komik.v4i1.2691

Jalaludin, A., & Melita, Y. (2012). Implementasi metode Run Length Encoding untuk kompresi citra. Jurnal Teknik, 3(2), 249–254.

Mansyuri, U. (2021). Kompresi data teks dengan metode Run Length Encoding. Jurnal SIMASI, 1(2), 102–109.

Mahendra, A. D., Suryani, E., & Aziz, A. (2016). Analisis perbandingan kinerja kombinasi algoritma BWT-RLE-MTF-Huffman dan BWT-MTF-RLE-Huffman pada kompresi file. Jurnal Teknologi Informasi ITSmart, 1(2), 107. https://doi.org/10.20961/its.v1i2.606

Buulolo, E. (2017). Sequitur algorithm for particular character compression or words always returned. International Journal of Informatics and Computer Science (The IJICS), 1(1), 15–17.

Siahaan, S. (2019). Penerapan algoritma Sequitur pada kompresi record database pada database. JURIKOM (Jurnal Riset Komputer), 6(5), 511–516. https://www.ejurnal.stmik-budidarma.ac.id/index.php/jurikom/article/view/1644

Burrows, M., & Wheeler, D. J. (1994). A block-sorting lossless data compression algorithm (SRC Research Report 124). Digital Systems Research Center.

Nevill-Manning, C. G., & Witten, I. H. (1997). Identifying hierarchical structure in sequences: A linear-time algorithm. Journal of Artificial Intelligence Research, 7, 67–82. https://doi.org/10.1613/JAIR.374

Salomon, D. (2007). Data compression: The complete reference (4th ed.). Springer London. https://doi.org/10.1007/978-1-84628-603-2