SNESTIK

Seminar Nasional Teknik Elektro, Sistem Informasi, dan Teknik Informatika

https://ejurnal.itats.ac.id/snestik dan https://snestik.itats.ac.id

Informasi Pelaksanaan:
SNESTIK III - Surabaya, 11 Maret 2023
Ruang Seminar Gedung A, Kampus Institut Teknologi Adhi Tama Surabaya

Informasi Artikel:
DOI : 10.31284/p.snestik.2023.4266

Prosiding ISSN 2775-5126

Fakultas Teknik Elektro dan Teknologi Informasi-Institut Teknologi Adhi Tama Surabaya
Gedung A-ITATS, Jl. Arief Rachman Hakim 100 Surabaya 60117 Telp. (031) 5945043
Email : snestik@itats.ac.id

Implementasi Algoritma Levenshtein untuk Kompresi File Audio

Gusti Eka Yuliastuti, Danang Haryo Sulaksono, Citra Nurina Prabiantissa, Achmad Febrianto

Teknik Informatika, Institut Teknologi Adhi Tama Surabaya
e-mail: gustiekay@itats.ac.id

ABSTRACT

MPEG-1 Audio Layer 3 or commonly known as MP3 file is one of the audio data formats that is often used, because the data stored resembles the actual data when recorded. The more MP3 files that are stored, it will require a large storage space. The problems that need to be solved are to provide free space and change the size of the stored data to be smaller. This problem can be solved by compressing. The compression implementation that will be carried out by the author is by applying the Levenshtein Algorithm. This Levenstein algorithm has the advantage of being able to reduce storage space with a compression ratio of up to 100%. The results of testing the performance of the Levenshtein Algorithm obtained an average Ratio of Compression (RC) value of 0.921; the average Compression Ratio (CR) value is 92.18% and the Mean Square Error (MSE) value is 0.0925.

Keywords: *Audio; Compression; File; Levenshtein Algorithm; MP3.*

ABSTRAK

MPEG-1 Audio Layer 3 atau biasa dikenal sebagai *file* MP3 merupakan salah satu format data audio yang sering digunakan, karena data yang disimpan menyerupai data sebenarnya pada saat direkam. Semakin banyak *file* MP3 yang disimpan, maka akan memerlukan tempat penyimpanan yang besar. Permasalahan yang perlu diselesaikan yakni memberikan ruang kosong dan mengubah ukuran data yang disimpan menjadi lebih kecil. Permasalahan tersebut dapat diselesaikan dengan melakukan kompresi. Implementasi kompresi yang akan dilakukan oleh penulis yakni dengan menerapkan Algoritma Levenshtein. Algoritma Levenshtein ini memiliki kelebihan dapat mengurangi ruang penyimpanan dengan rasio kompresi hingga sebesar 100%. Hasil pengujian kinerja Algoritma Levenshtein didapatkan nilai *Ratio of Compression* (RC) rata-rata sebesar

0,921; nilai *Compression Ratio* (CR) rata-rata sebesar 92,18% dan nilai *Mean Square Error* (MSE) sebesar 0,0925.

Kata kunci: Algoritma Levenshtein; Audio; File; Kompresi; MP3.

PENDAHULUAN

MPEG-1 Audio Layer 3 atau biasa dikenal sebagai *file* MP3 merupakan salah satu format data audio yang sering digunakan, karena data yang disimpan menyerupai data sebenarnya pada saat direkam. Meskipun *file* MP3 memiliki ukuran yang tidak terlalu besar dibandingkan format audio lain, namun ukurannya cukup besar jika dibandingkan dengan ukuran *file* dokumen biasa [1]. Semakin banyak *file* MP3 yang disimpan, maka akan memerlukan tempat penyimpanan yang besar. Solusi atas keterbatasan tersebut yakni dengan penyimpanan secara digital. Penyimpanan secara digital menjadi salah satu pilihan yang terbaik, karena tidak membutuhkan tempat penyimpanan yang besar.

Permasalahan yang perlu diselesaikan yakni memberikan ruang kosong dan mengubah ukuran data yang disimpan menjadi lebih kecil. Permasalahan tersebut dapat diselesaikan dengan melakukan kompresi [2]. Kompresi merupakan salah satu cara untuk memampatkan data sehingga ukuran bit yang diperlukan suatu data dalam memori penyimpanan tidak terlalu besar [3]. Jika semua data yang disimpan berukuran kecil, maka akan menimbulkan ruang kosong yang lebih besar. Dengan adanya ruang kosong ini, maka media penyimpanan akan memiliki tempat lebih untuk menyimpan data yang lebih banyak lagi.

Implementasi kompresi yang akan dilakukan oleh penulis yakni dengan menerapkan Algoritma Levenshtein. Algoritma Levenshtein ini memiliki kelebihan dapat mengurangi ruang penyimpanan dengan rasio kompresi hingga sebesar 100% [4]. Hal tersebut sangat membantu dalam hal penghematan ruang [5]. Besarnya hasil rasio kompresi bergantung pada jumlah karakter dan jumlah kejadian dari masing-masing karakternya.

Algoritma Levenshtein merupakan salah satu jenis *lossless compression* [6]. *Lossless compression* adalah proses kompresi *file* dimana *file* hasil kompresi tersebut jika dilakukan dekompresi akan menghasilkan *file* yang sama persis dengan *file* awalnya [7]. Pada Algoritma Levenshtein, semakin besar frekuensi kejadian karakter maka semakin kecil hasil kompresinya. Algoritma Levenshtein ini cocok digunakan untuk mengkompresi data yang memiliki banyak pengulangan [8]. Kekurangan dari Algoritma Levenshtein adalah harus membuat tabel baru dalam *database* untuk menyimpan kode Levenstein yang digunakan untuk mengubah karakter kompresi ke karakter awal. Pembuatan tabel baru ini akan dapat menambah kapasitas memori yang digunakan.

METODE

Kompresi

Kompresi merupakan pengubahan data ke dalam bentuk lain yang memerlukan bit lebih sedikit [9]. Tujuan dari dilakukannya kompresi yakni agar data tersebut dapat disimpan atau dikirimkan dengan lebih efisien [10]. Selain kompresi, terdapat proses kebalikannya yakni dekompresi. Dekompresi adalah proses untuk mengembalikan data baru yang telah dihasilkan oleh proses kompresi menjadi data awal. Data hasil kompresi dapat digunakan dalam jaringan yang lebih rendah sesuai dengan kapasitas.

Teknik kompresi terdiri dari 3 kategori, antara lain: sumber, *entropy* dan *hybrid*. Contoh data untuk kompresi kategori sumber yakni jenis audio *lossy*, dimana terjadi beberapa bagian komponen dari data yang hilang akibat dari proses kompresi. Contoh data untuk kategori *entropy* yakni jenis audio *lossless*, yang berarti tidak ada data yang hilang selama proses kompresi berjalan. Sedangkan contoh data untuk kategori *hybrid*, merupakan kombinasi dari jenis audio *lossy* dan *lossless*.

Algoritma Levenstein

Algoritma Levenshtein atau Levenstein *Coding* merupakan pengkodean universal untuk bilangan bulat non-negatif yang dikembangkan oleh Vladimir Levenshtein pada tahun 1968 [11]. Algoritma ini tidak banyak diketahui baik pada saat pengkodean maupun pembacaan sandi. Proses pengkodean pada Algoritma Levenshtein yakni satu dan nol.

Berikut merupakan tahapan dalam pengkodean angka positif *n*:

1. Atur jumlah variabel C menjadi 1
2. Tuliskan representasi biner dari nomor tanpa awalan "1" ke kode awal
3. Misalkan M adalah jumlah bit yang dituliskan pada langkah kedua
4. Jika M tidak sama dengan 0, tambahkan C dengan 1. Ulangi langkah kedua, dengan memasukkan M sebagai nomor baru
5. Jika M = 0, tambahkan C "1" bit dan 0 ke awal kode

Sedangkan untuk tahapan pendekodean yakni sebagai berikut:

1. Lakukan perhitungan pada jumlah bit C "1" sampai "0" ditemukan
2. Jika hasil perhitungannya adalah C = 0, maka nilainya adalah nol, lalu berhenti. Jika tidak lanjutkan langkah ketiga
3. Atur N = 1, dan ulangi langkah 4(C-1) kali
4. Lakukan pembacaan bit N, tambahkan 1, lalu berikan nilai yang dihasilkan ke N (dengan demikian dapat menghilangkan nilai sebelumnya dari N)
5. String yang diberikan ke N pada iterasi terakhir merupakan hasil dari pembacaan sandi.

Algoritma Levenshtein dapat digunakan untuk melakukan kompresi pada data dengan memanfaatkan perhitungan jarak antar *string*. Penggunaan Algoritma Levenshtein untuk kompresi data akan menghasilkan beberapa nilai evaluasi, antara lain: *Ratio of Compression* (RC), *Compression Ratio* (CR) dan *Mean Square Error* (MSE).

***Ratio of Compression* (RC)**

Ratio of Compression (RC) merupakan nilai perbandingan antara ukuran bit data sebelum dikompresi dengan ukuran bit data yang telah dikompresi [2][8]. Rumus untuk menghitung nilai RC seperti ditunjukkan pada Persamaan 1.

$$Ratio\ of\ Compression\ (RC) = \frac{ukuran\ data\ hasil\ kompresi}{ukuran\ data\ asli} \quad (1)$$

***Compression Ratio* (CR)**

Compression Ratio (CR) merupakan proses perhitungan ukuran data setelah dikompresi dan membaginya dengan ukuran data asli [2][12]. Setelah itu nilai dikalikan dengan prosentase 100%. Rumus untuk menghitung nilai CR seperti ditunjukkan pada Persamaan 2.

$$Compression\ Ratio = \frac{ukuran\ data\ hasil\ kompresi}{ukuran\ data\ asli} \times 100\% \quad (2)$$

***Mean Square Error* (MSE)**

Mean Square Error (MSE) merupakan nilai rata-rata penyimpangan perangkat antara dua data [13], dalam hal ini kedua data tersebut yakni audio asli dan audio hasil kompresi.

Semakin mirip kedua data yang diproses, maka semakin kecil nilai MSE yang dihasilkan. Rumus untuk menghitung nilai MSE seperti ditunjukkan pada Persamaan 3.

$$MSE = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N}[f(x,y) - f^{'}(x,y)]^{2} \quad (3)$$

HASIL DAN PEMBAHASAN

Pengujian kinerja pada implementasi kompresi *file* audio menggunakan Algoritma Levenshtein ini meliputi pengujian sistem dengan *testing data* menggunakan sampel sebanyak 100 data yang berupa data *file* audio berekstensi *mp3. Data dengan nama "Musik" diunduh dari halaman web https://lagubebass.blogspot.com/2018/10/andra-and-backbone-self-titled-full.html. Data *file* audio dengan nama "Musik Download" diunduh dari halaman web https://ilkpop.net/music/download.xhtml. Data dengan nama "Musik Murni" diunduh dari halaman web http://matikiriwap.wapku.net/site-download.html?to-id=3647.

Pengujian dilakukan dengan cara melakukan proses kompresi *file* menggunakan Algoritma Levenshtein dan dilanjutkan dengan melakukan proses dekompresi *file* terhadap hasil kompresinya. Performa dari hasil rekonstruksi diukur berdasarkan nilai *Ratio of Compression* (RC), *Compression Ratio* (CR) dan *Mean Square Error* (MSE) seperti ditunjukkan pada Tabel 1.

Tabel 1 Pengujian Hasil Kompresi

No	Nama File	File Asli	File Hasil	Selisih	RC	CR (%)	MSE
1	Musik (1)	4.52	4.49	0.03	0.993	99.33	0.0004
2	Musik (2)	4.54	4.5	0.04	0.991	99.11	0.0008
3	Musik (3)	4.3	4.27	0.03	0.993	99.30	0.0004
4	Musik (4)	3.51	3.51	0	1	100	0
5	Musik (5)	9.14	9.1	0.04	0.995	99.56	0.0008
6	Musik (6)	7.7	7.5	0.2	0.974	97.40	0.02
7	Musik (7)	8.21	8.1	0.11	0.986	98.66	0.0060
8	Musik (8)	6.3	6.25	0.05	0.992	99.20	0.0012
9	Musik (9)	5.48	5.32	0.16	0.970	97.08	0.0128
10	Musik (10)	4.71	4.67	0.04	0.991	99.15	0.0008
11	Musik Download (1)	3.03	2.84	0.19	0.937	93.72	0.0180
12	Musik Download (2)	3.96	3.87	0.09	0.977	97.72	0.0040
13	Musik Download (3)	3.2	3.02	0.18	0.943	94.37	0.0162
14	Musik Download (4)	2.96	2.78	0.18	0.939	93.91	0.0162

15	Musik Download (5)	4.82	4.87	-0.05	1.010	101.03	0.0012
16	Musik Download (6)	3.4	3.26	0.14	0.958	95.88	0.0098
17	Musik Download (7)	7.47	7.53	-0.06	1.008	100.80	0.0018
18	Musik Download (8)	3.7	3.13	0.57	0.845	84.59	0.1624
19	Musik Download (9)	3.35	3.08	0.27	0.919	91.94	0.0364
20	Musik Download (10)	3.15	3.03	0.12	0.961	96.19	0.0072
21	Musik Murni (1)	2.33	2.05	0.28	0.879	87.98	0.0392
22	Musik Murni (2)	2.28	1.67	0.61	0.732	73.24	0.1860
23	Musik Murni (3)	4.83	4.22	0.61	0.873	87.37	0.1860
24	Musik Murni (4)	0.556	1.03	-0.474	1.852	185.25	0.1123
25	Musik Murni (5)	2.58	2.66	-0.08	1.031	103.10	0.0032
26	Musik Murni (6)	2.26	2.03	0.23	0.898	89.82	0.0264
27	Musik Murni (7)	2.98	2.34	0.64	0.785	78.52	0.2048
28	Musik Murni (8)	0.32	0.452	-0.132	1.412	141.25	0.0087
29	Musik Murni (9)	2.92	2.62	0.3	0.897	89.72	0.0450
30	Musik Murni (10)	2.77	2.16	0.61	0.779	77.97	0.1860

Berdasarkan hasil pengujian pada Tabel 1, rasio kompresi terbaik terjadi pada *file* yang memiliki nilai RC sebesar 1 dimana bobot karakter hampir sebesar atau sama besar dengan ukuran *file* yang sesungguhnya. Hasil *Ratio of Compression* (RC) rata-rata yang didapatkan yakni sebesar 0,921.

File tersebut memiliki nilai CR 100% dimana bobot karakter hampir sebesar atau sama besar dengan ukuran file yang sesungguhnya. Hasil *Compression Ratio* (CR) rata-rata yang didapatkan yakni sebesar 92,18%.

Nilai rata-rata *Mean Square Error* (MSE) cukup rendah yakni sebesar 0,0925. Hal ini menunjukkan bahwa Algoritma Levenshtein cukup baik dalam penerapan kompresi. MSE dihitung dengan tujuan untuk merekonstruksi *file* kompresi dan mengukur kesalahan proses kompresi secara keseluruhan.

KESIMPULAN

Algoritma Levenshtein dapat digunakan untuk melakukan kompresi *file* audio dengan ekstensi mp3. Hal ini terbukti dari hasil pengujian kinerja Algoritma Levenshtein yang dapat diketahui dari nilai *Ratio of Compression* (RC) rata-rata sebesar 0,921 dan nilai *Compression*

Ratio (CR) rata-rata sebesar 92,18%. Kedua perhitungan rasio kompresi tersebut berfungsi untuk menguji kesamaan *file* hasil kompresi dan dapat dikembalikan seperti semula atau melakukan proses dekompresi.

Performa dari Algoritma Levenshtein dapat diketahui berdasarkan hasil *nilai Mean Square Error* (MSE) yakni sebesar 0,0925. Hal ini menunjukkan bahwa Algoritma Levenshtein memiliki performa yang sangat baik dalam hal rendahnya nilai *error*.

DAFTAR PUSTAKA

[1] L. Y. Telaumbanua, E. Bu, and K. Ulfa, "Implementasi Algoritma Code-Excited Linear Prediction (CELP) pada Kompresi File Audio," in *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, 2022, vol. 6, no. November, pp. 317–321, doi: 10.30865/komik.v6i1.5701.

[2] S. H. Silitonga and S. D. Nasution, "Implementasi Algoritma Boldi-Vigna Codes Untuk Kompresi File Audio pada Aplikasi Pemutar Audio Berbasis Web," in *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, 2022, vol. 6, no. November, pp. 586–595, doi: 10.30865/komik.v6i1.5754.

[3] G. E. Purba, "Kompresi File Advanced Audio Coding (AAC) Menggunakan Metode Lempel Ziv Oberhumer (LZO)," *J. Comput. Informatics Res.*, vol. 2, no. 1, pp. 30–36, 2022.

[4] L. V Simanjuntak, "Perbandingan Algoritma Elias Delta Code dengan Levenstein Untuk Kompresi File Teks," *J. Comput. Syst. Informatics*, vol. 1, no. 3, pp. 184–190, 2020, [Online]. Available: https://ejurnal.seminar-id.com/index.php/josyc/article/view/168.

[5] D. P. Yuana and A. Prihanto, "Implementasi Digital Watermarking Pada File Audio Mp3 Menggunakan Metode FHSS (Frequency Hopping Spread Spectrum) Dan Fourier Transform," *J. Informatics Comput. Sci.*, vol. 3, no. 04, pp. 456–461, 2022, doi: 10.26740/jinacs.v3n04.p456-461.

[6] E. Hoogeboom, J. W. T. Peters, R. van den Berg, and M. Welling, "Integer Discrete Flows and Lossless Compression," *Adv. Neural Inf. Process. Syst.*, vol. 32, no. NeurIPS, 2019.

[7] A. Suharso, J. Zaelani, and D. Juardi, "Kompresi File Menggunakan Algoritma Lempel Ziv Welch (LZW)," *Komputasi J. Ilm. Ilmu Komput. dan Mat.*, vol. 17, no. 2, pp. 372–380, 2020, doi: 10.33751/komputasi.v17i2.2147.

[8] D. Asdini and D. P. Utomo, "Analisis Perbandingan Kinerja Algoritma Huffman dan Algoritma Levenstein Dalam Kompresi File Dokumen Format .RTF," in *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, 2022, vol. 6, no. November, pp. 87–99, doi: 10.30865/komik.v6i1.5739.

[9] Y. Darnita, K. Khairunnisyah, and H. Mubarak, "Kompresi Data Teks dengan Menggunakan Algoritma Sequitur," *Sistemasi*, vol. 8, no. 1, p. 104, 2019, doi: 10.32520/stmsi.v8i1.429.

[10] A. N. Alim, H. Yuana, and F. Febrinita, "Aplikasi Kompresi Citra dengan Menggunakan Algoritma Lempel Ziv Welch (LZW)," *J. Mhs. Tek. Inform.*, vol. 6, no. 2, pp. 684–695, 2022.

[11] P. Coates and F. Breitinger, "Identifying Document Similarity Using A Fast Estimation of the Levenstein Distance Based on Compression and Signatures," in *Digital Forensics Conference Europe (DFRWS EU)*, 2022, pp. 1–11.

[12] V. No, J. Hal, and R. Syahputra, "Peningkatan Rasio Kompresi Algoritma RLE Menggunakan Algoritma BWT," *J. Sains dan Teknol.*, vol. 2, no. 1, pp. 10–13, 2022.

[13] A. M. Rizki, G. E. Yuliastuti, A. L. Nurlaili, and F. P. Aditiawan, "Forecasting the Inflation Rate in Indonesia Using Backpropagation Artificial Neural Network," in *2022 IEEE 8th Information Technology International Seminar (ITIS)*, 2022, pp. 1–5.