KOMPRESI DATA TEKS MENGGUNAKAN PENDEKATAN GRAMMAR COMPRESSION DENGAN ALGORITMA SEQUITUR

Ervin [(1)], Umi Proboyekti[(2)], Lucia D. Krisnawati[(3)]

Abstrak: Ukuran media penyimpanan data yang terbatas dan kebutuhan waktu *transfer* data yang cepat merupakan suatu masalah yang dihadapi dalam menyimpan dan men*transfer* data. *Sequitur* merupakan algoritma kompresi yang dapat menyimpulkan konteks tata bahasa apa saja. *Sequitur* memiliki 2 batasan dalam memampatkan data teks yaitu *digram uniqueness* dan *rule utility,* dimana 2 batasan ini akan diterapkan dalam program kompresi yang akan dibangun ini. Melalui penelitian dan analisis yang dilakukan pada karya tulis ini diperoleh bahwa semakin besar *file* yang akan dimampatkan dengan besarnya *compressed characters base yang* digunakan maka peluang keberhasilan pemampatan data teks juga semakin besar tetapi membutuhkan waktu yang cukup lama dalam pemampatan tersebut.

Kata Kunci: *grammar compression, sequitur, digram uniqueness, rule utility*

1. Pendahuluan

Kompresi atau pemampatan merupakan salah satu ilmu pengetahuan di bidang komputer. Pemampatan diperlukan karena adanya keterbatasan ruang memori, keterbatasan media penyimpanan data, dan kebutuhan waktu *transfer* data yang terbatas. Karya tulis ini akan membahas tentang pemampatan pada data teks. Untuk mempermudah melakukan pemampatan pada data teks dapat dilakukan dengan menerapkan beberapa metode, salah satunya adalah menggunakan pendekatan *Grammar Compression*. Algoritma yang dipakai dalam *Grammar Compression* ini adalah *Sequitur*. *Sequitur* menjadi titik awal dalam pengerjaan kompresi teks.

Dengan dibuatnya program kompresi data teks dengan pendekatan *Grammar Compression* ini, diharapkan dapat membantu pengguna komputer dalam mengatasi keterbatasan sistem memori, media penyimpanan data dan kebutuhan waktu *transfer* data.

1.1. Perumusan Masalah

Permasalahan dalam penulisan karya tulis ini dirumuskan sebagai berikut :

1. Bagaimana menerapkan algoritma *Sequitur* untuk pemampatan pada data teks?
2. Apakah ada perbedaan atau tidak antara sebelum memampatkan data teks dan setelah memampatkan data teks dengan menggunakan pendekatan *Grammar Compression* dengan algorima *Sequitur*?

1.2. Metode Penelitian

Metode penulisan yang digunakan untuk menyelesaikan karya tulis ini adalah metode studi pustaka dengan tahapan-tahapan sebagai berikut :

[(1)] *Ervin , Mahasiswa Teknik Informatika, Fakultas Teknik, Universitas Kristen Duta Wacana*
[(2)] *Umi Proboyekti, Dosen Teknik Inforrnatika, Fakultas Teknik, Universitas Kristen Duta Wacana*
[(3)] *Lucia D. Krisnawati, Dosen Teknik Inforrnatika, Fakultas Teknik, Universitas Kristen Duta Wacana*

1. Studi literatur melalui buku-buku pendukung, seperti buku yang berhubungan dengan Pengolahan Bahasa Natural dan Otomata, *Borland Delphi*, dan laporan Tugas Akhir.

2. Informasi metode *Grammar Compression* dan algoritma *Sequitur* melalui situs-situs di internet yang telah tersedia.

2. Landasan Teori

2.1 Context Free Grammar

Tata bahasa bebas konteks (*Context Free Grammar*) adalah suatu sistem formal yang menjelaskan sebuah bahasa dengan menetapkan bagaimana beberapa teks yang sah dapat diperoleh dari perbedaan simbol yang dikenal dengan *axiom* atau kalimat simbol (Rimincha Ammu,2006).

Menurut Rimincha Ammu tata bahasa bebas konteks terdiri dari parameter-parameter dibawah ini , antara lain :

1. Susunan simbol terminal, merupakan karakter dari abjad (*alphabet*) yang terlihat didalam *string* yang dihasilkan oleh sebuah tata bahasa. Yang termasuk terminal adalah kata-kata, karakter, huruf-huruf, atau kutipan, yang merupakan bagian dari keluaran terakhir. Dan koleksi kata merupakan koleksi dari terminal.

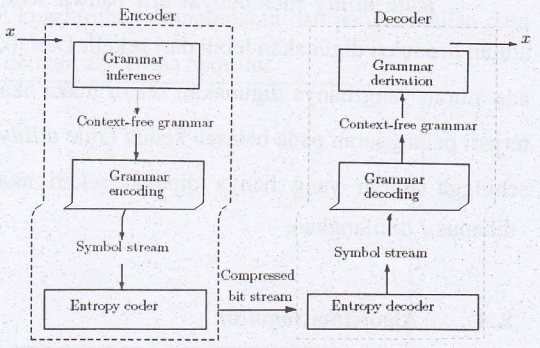
2. Susunan simbol *non-terminal*, merupakan penempatan untuk pola simbol *terminal* yang dapat dihasilkan oleh simbol *non-terminal*. *Non-terminal* adalah simbol internal yang digunakan oleh sebuah tata bahasa untuk menghasilkan keluaran terakhir.

3. Susunan produksi, merupakan aturan untuk menggantikan atau penulisan ulang terhadap simbol *non-terminal* yang terletak disebelah kiri produksi, didalam sebuah *string* dengan *non-terminal* lain atau simbol *terminal* yang terletak disebelah kanan produksi.

4. Sebuah simbol *start*, merupakan simbol variabel khusus yang terlihat pada huruf awal *string* yang dihasilkan oleh sebuah tata bahasa atau ada pada saat proses masuk.

2.2 Grammar Compression

Grammar based compression pertama kali diusulkan oleh Kieffer,Yang dan Nevill Manning (Lehman&Shelat; 2000). Metode *Grammar Compression* ini menunjukkan keberhasilan dalam memampatkan data untuk banyak tipe data yang berbeda. Yang menjadi landasan utama dalam metode ini adalah penggunaan *context free grammar* untuk menghasilkan suatu masukan teks.



Gambar 2.1 Gambaran Keseluruhan *Grammar Compression* (Cherniavsky&Ladner; 2004; hlm.2).

2.3 Sequitur

Menurut Nevill-Manning dan Ian Witten (1997), *Sequitur* merupakan sebuah algoritma waktu linier yang menyimpulkan tata bahasa bebas konteks (*context-free grammar*) ke dalam suatu pemampatan untuk mengurangi masukan yang berulang. Operasi *sequitur* terdiri dari pemastian dua sifat yang berlaku. Ketika menjelaskan algoritma, sifat-sifatnya bertindak sebagai batasan. Algoritma beroperasi menjalankan batasan didalam sebuah tata bahasa yang ketika digram keunikan (*digram uniqueness*) dilanggar, sebuah aturan baru dibentuk, dan ketika batasan aturan kegunaan (*rule utility*)

dilanggar, aturan yang sia-sia dihapus. Berikut ini menguraikan bagaimana dua batasan ini terjadi :

2.3.1 Digram Keunikan (*Digram Uniqueness*)

Digram uniqueness mempunyai arti bahwa tidak ada pasangan dari simbol / digram muncul lebih dari sekali dalam sebuah tata bahasa. Jika hal ini terjadi maka akan melanggar aturan batasan pertama(*digram uniqueness*) sehingga akan membentuk aturan baru (simbol *non-terminal*) yang akan menggantikan simbol / digram yang muncul lebih dari sekali.

2.3.2 Aturan Kegunaan (*Rule Utility*)

Rule utility mempunyai arti bahwa setiap aturan produksi digunakan lebih dari sekali. Dan jika ada aturan yang hanya digunakan sekali maka akan terjadi pelanggaran pada batasan kedua (*rule utility*) sehingga aturan yang hanya dipakai sekali akan dihapus / dihilangkan.

2.4 Algoritma *Sequitur*

Algoritma *Sequitur* beroperasi dengan menjalankan batasan digram keunikan (*digram uniqueness*) dan aturan kegunaan (*rule utility*). Beberapa batasan pelanggaran adalah sangat penting untuk dideteksi secara efisien.

Dibawah ini merupakan algoritma *Sequitur* :

3. Pembahasan

3.1. Perancangan Sistem

Masukan yang akan diterima program adalah *file* data teks yang berformat *.txt* dan *.vtxt*. Format *.txt* adalah untuk masukan *file* data teks yang akan dikompres dan karakteristik masukan data teks yang akan dimampatkan adalah berupa karakter a sampai z dan 0 sampai 9. Sedangkan yang berformat *.vtxt* adalah untuk masukan *file* data teks yang akan didekompres dan karakteristik masukan *file* yang akan didekompres adalah berisi jumlah *non-terminal*, daftar data *non-terminal* dan isi data kompresi, masing-masing dipisahkan oleh tanda spasi (_), isi *file* kompresi yang akan didekompresi dapat dilihat sebagai berikut :"2_€A->ab_€B->€Ac_€B€A€B_", dimana '2' adalah jumlah *non-terminal*, '€A->ab_€B->€Ac' adalah daftar data *non-terminal* dan '€B€A€B' adalah isi data kompresi.

Dibawah ini merupakan contoh tampilan masukan *file* pemampatan data teks yang akan muncul pada *form* pada saat memasukkan data teks sebelum dimampatkan, yaitu sebagai berikut :

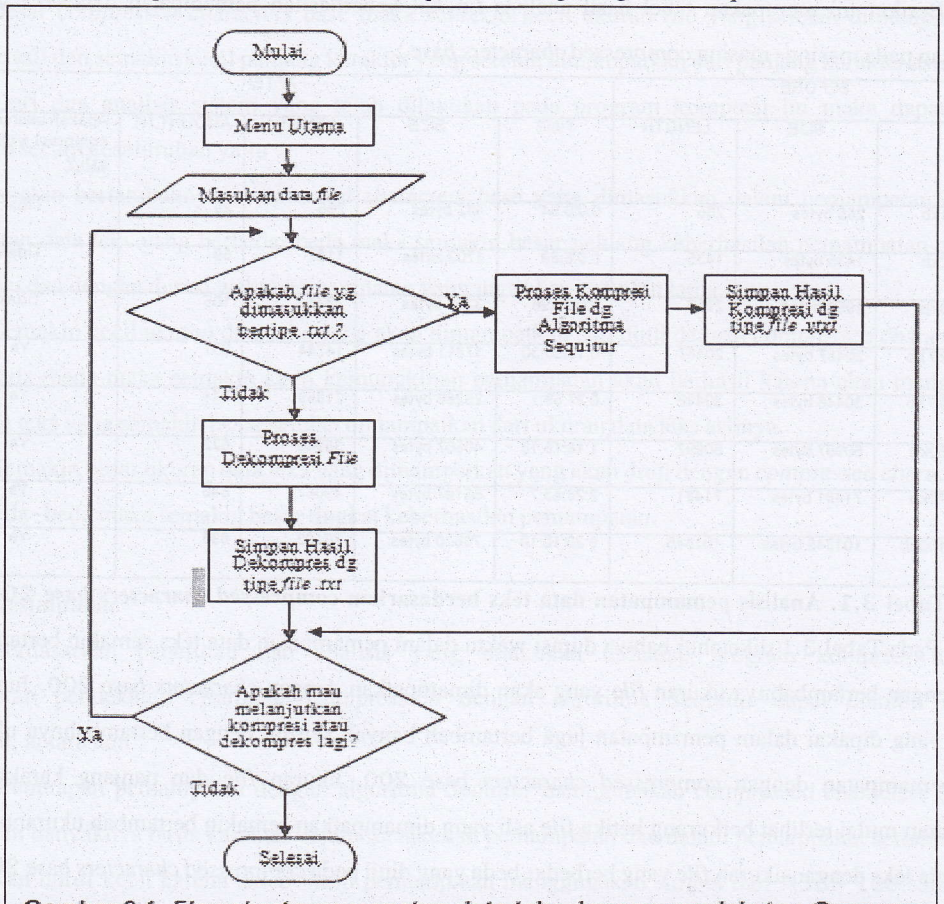
masukan karakter pertama s untuk membuat hasil dari S → abcababcd ;
mengulang
mencocokkan pada *non-terminal* yang ada :
S → AcAab → A→ab
S → AcAA
membuat sebuah *non-terminal* baru :
S → AcAAc A → ab
→ B → Ac
memindahkan *non-terminal* :
S → AcAAc A → ab
S → BAB → B → Ac
memasukkan karakter baru :
S → BABd
sampai tidak ada karakter yang tersisa

Nama *file*	: 'coba'
Tipe *file*	: '.txt'
Besar file	: *{besarnya file sebelum dikompresi}*
Durasi waktu	: *{tergantung lamanya proses kompresi berlangsung}*
Data	: ' latar belakang masalah kompresi merupakan salah satu ilmu pengetahuan di bidang komputer. kompresi adalah pemampatan ukuran file.'

Dibawah ini merupakan contoh tampilan masukan pengembalian *file* data teks yang akan muncul pada *form*, yaitu sebagai berikut :

Nama *file*	: 'coba1'
Tipe *file*	: '.vtxt'
Besar file	: *{besarnya file setelah dikompresi}*
Durasi waktu	: *{tergantung lamanya proses dekompresi berlangsung}*
Data	: ' AGr be$A$Cg $RF $P m$LuSC $F satu $Tmu $Qnge$Ghu$H di bid$Hg KutL. $P ada$Ah QRmSGn ukur$H f$Te '
Daftar NT	: $A→la, $C→kan, $F→sa$Ah, $G→ta, $H→an, $K→komp, $L→er, $P→$Kresi, $Q→pe, $R→ma, $S→pa, $T→il
Jumlah NT	: 12

Gambar 3.1. merupakan *flowchart* dari gambaran keseluruhan pemampatan dan pengembalian data teks yang menggunakan pendekatan *Grammar Compression* dengan algoritma *Sequitur.*



Gambar 3.1. *Flowchart* pemampatan data teks dengan pendekatan *Grammar Compression*

Pada flowchart diatas dapat dilihat bahwa pada menu utama, kita dapat memasukkan data *file*, dimana jika *file* yang dimasukkan adalah bertipe *.txt* maka proses kompresi dengan algoritma *Sequitur* akan dijalankan dan kemudian disimpan dalam tipe *.vtxt* dan jika *file* yang dimasukkan adalah bertipe *.vtxt* maka proses dekompresi akan dijalankan dan kemudian disimpan dalam tipe *.txt.* Setelah semua proses kompresi dan dekompresi telah dijalankan maka akan proses akan bertanya apakah masih ingin melanjutkan kompresi atau dekompresi lagi, dan jika tidak maka proses tersebut telah selesai.

3.2 Analisis Sistem

Pada tahap pengujian ini dibuat 2 analisis sistem yaitu Analisis tentang tingkat keberhasilan pemampatan data teks dengan ukuran *file* yang berbeda-beda diuji berdasarkan pada besarnya *compressed characters base* yang dimasukkan dan Analisis tentang perbedaan tiap *byte compression* yang dimasukkan pada tiap ukuran *file* yang berbeda-beda yang diteliti dari segi waktu, ukuran, panjang karakter dan jumlah non-terminal.

Data pengujian diambil 8 contoh data teks dengan ukuran *file* yang berbeda-beda yaitu 1 kb, 2 kb, 3 kb, 20 kb, 30 kb, 50 kb, 70 kb, 100 kb dan 4 basis karakter kompresi yang berbeda-beda yaitu basis karakter 10, basis karakter 50, basis karakter 100, basis karakter 200.

Berikut ini merupakan tabel hasil analisis tingkat keberhasilan pemampatan data teks yang diuji berdasarkan pada masing-masing *compressed characters base* :

BEFORE			AFTER				
	SIZE	LENGTH	TIME	SIZE	LENGTH	AMOUNT NT	Apakah ukuran file hasil kompresi < ukuran file asli ?
Coba1 1kb	265 bytes	265	0:0:0:54	482 bytes	265	29	Tidak
Coba2 2kb	1435 bytes	1435	0:0:8:89	1703 bytes	1152	69	Tidak
Coba3 3kb	2959 bytes	2959	0:0:29:96	3251 bytes	2371	105	Tidak
Coba4 20 kb	20487 bytes	20487	0:16:30:30	17811 bytes	14744	331	Ya
Coba5 30 kb	30446 bytes	30446	0:27:59:1	25286 bytes	21859	371	Ya
Coba6 50kb	50997 bytes	50997	1:18:16:16	40807 bytes	35906	522	Ya
Coba7 70kb	71481 bytes	71481	2:28:53:7	55197 bytes	49067	646	Ya
Coba8 100kb	101846 bytes	101846	6:20:10:10	78620 bytes	70103	898	Ya

Tabel 3.1. Analisis pemampatan data teks berdasarkan *compressed characters base* 200

Pada Tabel 3.1 diketahui bahwa durasi waktu dalam pemampatan data teks semakin bertambah yang diikuti dengan bertambahnya ukuran *file* yang akan dimampatkan dengan *characters base* 200. Jumlah *non-terminal* yang dipakai dalam pemampatan juga bertambah banyak seiring dengan bertambahnya ukuran *file*. Dalam pemampatan dengan *compressed characters base* 200, ukuran *file* dan panjang karakter setelah pemampatan mulai terlihat berkurang ketika *file* asli yang dimampatkan semakin bertambah ukurannya. Dari 8 contoh data teks dengan ukuran *file* yang berbeda-beda yang diuji pada *compressed characters base* 200 didapat 5 contoh data teks yang berhasil dimampatkan yaitu ukuran *file* dan panjang karakter yang semakin berkurang

dari ukuran *file* dan panjang karakter aslinya.

Berikut ini merupakan tabel analisis perbedaan pemampatan data teks pada tiap *compressed characters base* berdasarkan pada ukuran file teks yang berbeda-beda :

SIZE ORIGINAL	LENGTH ORIGINAL	COMPRESSED CHARACTERS BASE	AFTER			
			TIME	SIZE	LENGTH	AMOUNT NT
101846 bytes	101846	10	0:4:6:56	101423 bytes	100672	101
		50	1:42:26:34	89182 bytes	85019	465
		100	3:28:5:62	82395 bytes	76085	673
		200	6:20:10:10	78620 bytes	70103	898

Tabel 3.2. Analisis perbedaan pemampatan dengan masing-masing *compressed characters base* pada contoh data teks Coba8

Pada Tabel 3.2. diketahui bahwa pemampatan pada data teks Coba8 yang berukuran 100kb dengan *compressed characters base* yang terus bertambah menghasilkan durasi waktu pemampatan yang terus bertambah lama, panjang karakter *file* yang semakin kecil, jumlah *non-terminal* yang terus bertambah banyak dan ukuran *file* teks yang semakin berkurang dari ukuran *file* aslinya. Sehingga dapat diambil kesimpulan bahwa semakin besar *compressed characters base* maka semakin kecil ukuran *file* yang setelah dimampatkan dari ukuran *file* asli dan semakin kecil panjang karakter yang setelah dimampatkan dari panjang karakter asli.

Dari dua analisis sistem yang telah dilakukan pada program kompresi ini maka dapat diambil kesimpulan secara keseluruhan yaitu :

1. *Semakin bertambahnya compressed characters base yang dimasukkan dalam pemampatan data teks* untuk ukuran data teks yang berbeda-beda maka semakin besar peluang keberhasilan pemampatan data yang akan dicapai dan dengan durasi waktu pemampatan yang juga akan semakin lama.
2. Semakin kecil ukuran data teks yang akan dimampatkan dan diuji dengan *compressed characters base* yang berbeda-beda maka semakin kecil kemungkinan pemampatan akan berhasil karena akan menghasilkan ukuran data teks yang semakin besar setelah dimampatkan dari ukuran data teks aslinya.
3. Semakin besar ukuran data teks yang dimampatkan yang akan diuji dengan *compressed characters base* yang berbeda-beda maka semakin besar tingkat keberhasilan pemampatan.

4. Kesimpulan

Berdasarkan penelitian dan analisis yang dilakukan terhadap program kompresi data teks menggunakan pendekatan *Grammar Compression* dengan algoritma *Sequitur* dapat diambil beberapa kesimpulan, antara lain :

1. Penerapan pemampatan dengan algoritma *Sequitur* menggunakan *compressed characters base* atau pengambilan banyaknya basis karakter dalam melakukan pemampatan dan dalam pemampatan semua data teks asli dijadikan huruf kecil karena dalam hasil pemampatan menggunakan simbol dari ASCII 128 sampai 255 ditambah dengan huruf besar A sampai Z.

2. Pengembalian pemampatan data teks dengan algoritma *Sequitur* tidak menghasilkan hasil akhir yang sempurna karena hasil data teks setelah pengembalian semua ditampilkan dalam huruf kecil.

5. Daftar Pustaka

Jurafsky, Daniel & James H.Martin. *Speech and Language Processing*. Prentice-Hall, 2000

Wayner, Peter. *Compression Algorithms for Real Programmers*. San Diego : Academic Press Limited, 2000

Lehman, Eric & Abhi Shelat, http://theory.lcs.mit.edu/~abhi/LehmanShelat-SODA02.ps (diakses 7 November 2006)

Lecturer7, http://mnl.cs.sunysb.edu/class/cse592/2006-fall/lectures/cse592-lecture09.pdf (diakses 7 November 2006)

Ammu, Rimincha, http://www.laynetworks.com/Context%20Free%20Grammar.htm

Manning, Craig G. Nevill & Ian H. Witten, http://sequitur.info/jair/ (diakses 7 November 2006)

6. Biodata Singkat Penulis

Nama : Ervin

Email : ervinchristy@yahoo.com