KOMPRESI MULTILEVEL PADA METAHEURISTIC FOCUSED WEB CRAWLER

Dian S. Santoso [1)] dan R.V. Hari Ginardi [2)]
[1, 2)] Departemen Informatika, Institut Teknologi Sepuluh Nopember
Jalan Raya ITS, Kota Surabaya Jawa Timur 60111
e-mail: dianmhs16@mhs.if.its.ac.id [1)], hari@its.ac.id[2)]

ABSTRAK

Focused Web Crawler merupakan metode pencarian website yang sesuai dengan pencarian yang diinginkan oleh user. Untuk mendapatkan kecocokan yang baik, waktu yang dibutuhkan oleh metode Focused Web Crawler lebih lama dibandingkan dengan metode pencarian web crawler pada umumnya yang menggunakan algoritma DFS (Depth First Search) maupun BFS (Breadth First Search). Untuk mengatasi hal tersebut, dikembangkan teknik pencarian Focused Web Crawler dengan menggunakan metode metaheuristic pencarian cuckoo yang dipadukan dengan pencarian pada data history pencarian yang disimpan. Namun, dengan adanya penyimpanan data pada setiap kali pencarian link maka data akan semakin bertambah. Oleh karena itu diperlukan sebuah cara untuk mengurangi kebutuhan ruang penyimpanan. Cara yang dilakukan untuk mengurangi ruang penyimpanan dan tidak mengurangi nilai informasi dari data penyimpanan sebelumnya adalah dengan melakukan kompresi data. Penelitian ini mengusulkan metode kompresi data dengan melakukan kompresi multilevel menggunakan dua metode kompresi, yaitu pengurangan prefix dan postfix kata dan kompresi string berbasis kamus dengan melakukan pembuatan indeks kamus kata. Hasil kompresi string kamus kata berupa data encode. Pengujian hasil kompresi data dilakukan dengan perbandingan hasil pencarian link menggunakan metode KMP (Knutt Morris Pratt) pada data yang belum terkompresi dengan data yang telah terkompresi. Hasil pengujian menunjukkan maksimum presisi mencapai nilai 1, recall sebesar 0,73, serta rasio kompresi file rata-rata sebesar 36,4%.

***Kata Kunci**: Focused Web Crawler, Knutt Morris Pratt, Kompresi Berbasis Kamus, Kompresi Multilevel, Pencarian Cuckoo*

ABSTRACT

Focused Web Crawler is a method for finding websites that match the search properties desired by the user. This method is slower than the method for web crawlers in general, namely the DFS (Depth First Search) algorithm and the BFS (Breadth First Search). To overcome this, the Focused Web Crawler search is enhanced with the metaheuristic cuckoo search. This method combined with the search on history search data stored in the system. But it stores more data every time the link searches, which increase the required space for stored data. A data compression method is proposed, which reduces data size and keeps the value of stored information. This proposed compression method is a multilevel compression which uses two compression methods, namely reducing prefix and postfix words, and string-based dictionary compression by making dictionary index words. The results of the word dictionary string compression are encoded data. The performance of this data compression is evaluated by comparing the results of search links using the KMP (Knutt Morris Pratt) method of uncompressed data with compressed data. The result shows the maximum precision with a value of 1 while the recall is 0.73. It is also found that the average file compression ratio is 36.4%.

***Keywords**: Cuckoo Search, Dictionary Based String Compression, Focused Web Crawler, Knutt Morris Pratt, Multilevel Compression*

I. PENDAHULUAN

METODE pencarian *link* URL (*Uniform Resource Locator*) *website* dengan memperhatikan isi dokumen *website* yang sesuai dengan pencarian kata atau dikenal dengan istilah *Focused Web Crawler* sudah mulai banyak dikembangkan oleh para peneliti. *Focused Web Crawler* dikembangkan agar pencarian di dalam data web yang sangat besar dapat dilakukan secara efisien, baik waktu, memori pada aplikasi maupun *database* untuk hasil pencarian. Beberapa penelitian yang dilakukan antara lain adalah dengan mencari kesesuaian *website* dengan kata yang dicari oleh pengguna. Dalam penelitian yang telah dilakukan, untuk mencari kesesuaian isi *website* dapat dilakukan melalui pencocokan *keyword* dengan *link* URL web yang ada [1][2]. Pencarian kesesuaian dengan menggunakan *link* URL dalam *Focused Web Crawler* masih memiliki kekurangan karena tidak semua *link* URL merepresentasikan isi *website* [3].

Algoritma genetika digunakan untuk mengatasi tidak terkunjunginya *link* URL *website* pada proses pencarian di mana *link* URL tidak sesuai dengan *keyword* pencarian, tetapi memiliki konten yang sesuai dengan *keyword* pencarian [4]. Pencarian *link* URL dengan metode ini lebih meminimalkan *link* URL yang tidak terkunjungi daripada metode *BFS*. Optimasi pencarian *link* URL juga dilakukan dengan melakukan penghitungan relevansi konten *website* [5]. Dari percobaan yang dilakukan, berdasarkan nilai relevansi konten, hasil yang didapatkan mampu mengurangi kesalahan pencarian *link* URL terkait. Untuk menambah keakuratan penghitungan relevansi konten pada sebuah *website*, dilakukan penghitungan relevansi topik. Meskipun hasil yang lebih baik bisa didapatkan, waktu yang dibutuhkan untuk melakukan penilaian kerelevanan *website* yang dikunjungi menjadi lebih

lama. Optimasi untuk mengatasi lamanya proses pencarian dilakukan dengan pendekatan metode pencarian secara *heuristic* dengan mengetahui apa yang dicari, dari mana pencarian dilakukan dan data mana yang tidak dibutuhkan dengan menggunakan metode optimasi pencarian "*Cuckoo Search*" [6]. Setelah pencarian berhasil dilakukan, ditambahkan teknik pengenalan pola pada web yang sudah disimpan sebagai ekstraksi web dengan pencocokan *string* dengan menggunakan algoritma *Knutt Morris Pratt*. Metode ini membutuhkan ruang penyimpanan lebih banyak karena harus menyimpan *link* URL pencarian yang sudah dilakukan sebelumnya untuk menjadi acuan pencarian selanjutnya (sistem *history*). Untuk mengatasi hal tersebut, ditentukan batasan *overhead limit* namun belum menunjukkan hasil yang maksimal. Cara lain yang bisa dilakukan untuk mengatasi besarnya kebutuhan ruang penyimpanan tersebut adalah dengan menggunakan teknik kompresi data. Teknik kompresi data yang dipilih adalah teknik kompresi yang dapat menjamin bahwa hasil pencarian tetap memiliki nilai kebenaran yang mendekati pencarian sebelum terkompresi. Selain itu, teknik kompresi diharapkan memiliki rasio yang baik untuk mengurangi ukuran data *history link* URL pencarian tanpa mengurangi nilai informasi pada data yang disimpan.

II. STUDI LITERATUR

A. *Focused Web Crawler*

Focused Web Crawler merupakan metode pencarian *website* yang sesuai dengan pencarian yang diinginkan oleh *user*. Strategi-strategi yang diterapkan dalam *Focused Web Crawler* [7] mencakup strategi prioritas, strategi pembelajaran, strategi evaluasi, serta strategi *training*. Strategi pembelajaran dilakukan saat *crawler* sedang *offline* atau *reinforcement* atau penambahan dan pembelajaran data *crawler* saat *online.* Strategi evaluasi dilakukan dengan alat pengklasifikasi. Strategi *training* pada data fitur akan mempermudah proses pembelajaran mesin yang mencakup dua macam tipe data fitur yakni data teks dan data bukan teks. Terdapat dua tipe pencarian *Focused Web Crawler* terhadap *input* yang diberikan oleh pengguna yakni pencarian berdasarkan *link* URL yang ada di dalam web *page* dan berdasarkan konten di dalam web *page*. Metode pencarian *cuckoo* banyak digunakan karena memiliki hasil optimasi pencarian yang baik serta tidak begitu memerlukan banyak perhitungan. Pencarian ini lebih menekankan pada pencarian dari data *history* yang mirip sebagai kamus pencarian. *Focused Web Crawler* memanfaatkan metode pendekatan *metaheuristic* dengan metode pencarian *cuckoo* [6] untuk mengurangi waktu pencarian dan menambah keakuratan pencarian berdasarkan *link* URL *website*.

B. *Pendekatan Metaheuristic*

Metaheuristic merupakan metode tingkat lanjut dari metode *heuristic*, di mana metode *heuristic* adalah sebuah teknik yang digunakan untuk menyelesaikan masalah hingga ditemukan sebuah solusi [8]. Sesuai dengan pengertian pendekatan *metaheuristic* [9] [10], sebuah algoritma optimasi pencarian yang dinamai *cuckoo search* diciptakan oleh Yang dan Deb. Algoritma ini terinspirasi oleh parasit yang dilakukan spesies burung *cuckoo* di mana *cuckoo* menitipkan telur yang dimiliki pada sarang burung lain [11]. Dalam *cuckoo search* ada tiga aturan ideal [12] yaitu: setiap *cuckoo* meletakkan satu telur pada satu waktu dan meletakkannya pada sarang acak. Sarang terbaik dengan kualitas telur terbaik akan terbawa pada generasi berikutnya (akan hidup). Jumlah sarang *host* tetap, dan telur yang dititipkan akan diketahui induk pada sarang inang dengan nilai probabilitas sebesar 0 dan 1. Dari sini akan ditemukan sarang mana yang merupakan sarang terburuk dan tidak akan digunakan sebagai acuan untuk dilakukan penitipan telur selanjutnya.

C. *Kompresi Data*

Kompresi data adalah sebuah teknik untuk memadatkan data sehingga ruang yang dibutuhkan lebih kecil untuk penyimpanan sebuah data. Setiap jenis data (gambar, audio, video dan teks) memiliki teknik kompresi yang berbeda [13]. Pada kompresi data teks [14], kompresi *irreversible* atau *lossy* (tidak dapat dikembalikan lagi) bisa digunakan dengan kondisi tertentu, namun tidak disarankan karena kemungkinan data hilang atau tidak sama dengan data *input*. Teknik yang disarankan untuk kompresi data teks adalah dengan kompresi *reversible* atau *lossless*. Dari studi literatur, beberapa teknik kompresi data teks [15][16][17] mencoba untuk mengurangi kapasitas penyimpanan, pada saat data berada di dalam aplikasi tanpa menghilangkan informasi apa pun dari data yang ada (kompresi *lossless*). Dari ketiga penelitian teknik kompresi di atas, kompresi selalu mengubah simbol atau karakter menjadi bit yang lebih kecil daripada sebelumnya, dan untuk penulisan di dalam penyimpanan, setiap bit tersebut akan dikodekan ke dalam kode 8 bit per 1 *byte* yang disebut proses *encoding*.

Tahap selanjutnya yang dilakukan adalah pencarian *string* per *byte* dengan metode pencocokan *string* KMP. Untuk itu diperlukan teknik kompresi yang menyimpan hasil kompresi per *byte* yang memiliki nilai statis. Nilai statis diperlukan karena setelah indeks pencarian ditemukan, maka hasil bisa sesuai dengan model yang dibuat dan mengembalikan informasi secara tepat. Untuk itu dicari pendekatan metode lain yang memungkinkan untuk membuat sebuah teknik kompresi yang dapat mengurangi ukuran data dan mampu dilakukan pencarian dengan algoritma pencarian *string* KMP. Teknik kompresi ini [18] menggunakan pendekatan kamus kata yang dibuat

berdasarkan jenis kata (tambahan, akhiran, dan kata dasar). Teknik tersebut menunjukkan efisiensi urutan penyimpanan indeks kata pada pembuatan kamus kata, serta dapat dijadikan acuan pembuatan kamus kata yang menggunakan jenis kata (awalan, akhiran, dan kata dasar) bukan per karakter atau per huruf. Selain itu untuk penyimpanan data kompresi di dalam sistem dilakukan dengan menulis indeks ke dalam 2 *byte* data secara terus menerus.

D. Algoritma KMP (Knutt Morris Pratt)

Knutt Morris Pratt merupakan salah satu algoritma pencarian *string* yang dikembangkan oleh Knuth, Pratt, dan Morris yang diperkenalkan pada tahun 1977. Persoalan dalam pencarian *string* adalah sebuah teks dengan panjang "n", dan pola yaitu sebuah *string* dengan panjang "m" karakter di mana ("m" kurang dari "n") yang akan dicari di dalam teks. Dasar ideologi algoritma KMP adalah kapan pun saat terjadi ketidakcocokan (setelah beberapa pencocokan), algoritma sudah mengetahui berapa karakter dalam teks. Misalkan pada pencarian kata "formasi" dari teks "info, *inform*, dan informasikan". Langkah awal yang dilakukan adalah pencocokan pola "formasi" dengan teks dari kiri ke kanan. Yang dicocokkan pertama adalah huruf awal pola dengan huruf awal pada teks, karena huruf awal pola dimulai adalah huruf "f" dan huruf awal teks dimulai dengan huruf "i" maka huruf pertama pada teks dilewati dan dilanjutkan pada huruf kedua dan seterusnya sampai ditemukan pola yang cocok dan mencatat indeks ke berapa pola ditemukan [6].

III. METODE PENELITIAN

Desain sistem keseluruhan ditunjukkan pada diagram alir pada Gambar 1. Dalam desain sistem ini beberapa proses merupakan penelitian yang sebelumnya dilakukan oleh peneliti lainnya. Proses yang merupakan kontribusi penelitian ini dan metode yang diusulkan ditandai dengan kotak merah. Di dalam penelitian ini dua tahapan utama yang dilakukan yaitu proses kompresi *multilevel* dan proses pencarian *link* URL. Proses kompresi *multilevel* dilakukan untuk mengurangi ukuran *file history* pencarian *link* URL, sedangkan proses pencarian *link* URL merupakan proses untuk membuktikan bahwa metode kompresi yang diusulkan tidak mengurangi nilai kebenaran informasi hasil *link* URL *output*. Proses kompresi *multilevel* terdiri dari dua proses yaitu penghilangan *prefix* dan *postfix* kata, dan pembuatan kompresi *string* berbasis kamus. Sedangkan untuk proses pencarian *link* URL terdiri dari dua proses yaitu proses *encoding* kata masukan dan proses *decoding* hasil pencarian.

A. Praproses Pengambilan Data Link URL

Praproses dilakukan untuk mendapatkan data *link* URL hasil pencarian dengan web *crawler*. Pada praproses ini dilakukan pencarian *link* URL pada *website* dengan metode *Focused Web Crawler* dengan menggunakan *cuckoo search*. Data hasil pencarian berupa *link* URL dengan *tag* nama *link* URL yang akan digunakan sebagai data *history* pencarian web. Pengambilan data *link* URL menggunakan bahasa pemrograman *java* versi 8. Populasi awal dihasilkan dari *link* URL berita yang telah ditentukan sebelumnya seperti https://www.detik.com/, https://www.tribunnews.com, https://www.cnnindonesia.com/, https://www.viva.co.id/, dan http://www.liputan6.com/. Data yang dihasilkan akan digunakan sebagai *history* pencarian pada pencarian *metaheuristic*. Contoh data yang akan disimpan di dalam *history* pencarian ditampilkan pada Tabel I.

Gambar. 1. Diagram alir desain sistem

TABEL I
HISTORY PENCARIAN

Tag Nama	Alamat *Link* URL
Liputan Khusus	https://www.liputan6.com/news/liputankhusus
Zona MPR RI	https://www.liputan6.com/news/zona-mpr-ri
Cek Fakta	https://www.liputan6.com/news/cek-fakta
Divonis 15 Tahun Penjara, Setya Novanto Tak Ajukan Banding	https://www.liputan6.com/news/read/3496583/divonis-15-tahun-penjara-setya-novanto-tak-ajukan-banding
Internasional	https://news.detik.com/internasional
Tag Nama	https://news.detik.com/berita-jawa-timur/d-4000690/penjelasan-mensos-soal-stiker-cagub-jatim-di-program-pkh
Liputan Khusus	https://www.liputan6.com/news/read/3496570/100-lebih-wn-china-ditangkap-di-bali-terkait-kejahatan-siber

```
==========================================================
Begin
//proses persiapan link dan kompresi 1
set  content  ⟵  link hasil crawl pada proses sebelumnya
content       ⟵  delete content  prefix(http://, https://) dan postfix (karakter "/" atau "#")

//proses encoding berbasis kamus (kompresi 2)
kata          ⟵  tokenizing content
While kata memiliki lagi
if kata != null
        if map memiliki kata
         map  ⟵  put kata, map get kata + 1
        else
         map  ⟵  put kata, 1
        end if
end if
tabledata     ⟵  add urutan data map value menurut frek kata
dic_encode    ⟵  add tabledata (string, integer)
dic_decode    ⟵  add tabledata (integer, string)
t_encode      ⟵  write tabledata(string)
t_encode      ⟵  write tabledata(integer)
End While
//proses penyimpanan hasil encoding
While url != null
in            ⟵  add url link pencarian
enc           ⟵  tokenizing in
if enc not in dic_encode
      remap   ⟵  kamus kata
else
      b       ⟵  convert integer ke bentuk 2 byte
       encode ⟵  write b
end if
End While
End
==========================================================
```

Gambar. 2. *Pseudocode* kompresi data *link* URL

B. Kompresi Data Link URL

Proses selanjutnya adalah kompresi data *link* URL yaitu data hasil pencarian dengan *Focused Web Crawler* pada tahap sebelumnya. Metode kompresi yang digunakan dimulai dengan penghilangan kata (*prefix* dan *postfix*) dan selanjutnya dilakukan kompresi *string* berbasis kamus. Alur kompresi data ditunjukkan pada Gambar 2.

1) Persiapan Data Link URL

Langkah pertama yang dilakukan untuk kompresi data adalah persiapan data. Data yang akan dikompresi adalah data *history link* URL web berupa kumpulan data *string* yang sebelumnya telah disimpan di dalam aplikasi.

2) Penghilangan Prefix dan Postfix

Pada proses penghilangan *prefix*, kata yang dihilangkan adalah http:// dan https://. Penghilangan *postfix* dilakukan dengan menghilangkan kalimat atau kata setelah karakter "#" dan menghilangkan karakter "/" atau "\" yang terdapat pada akhir *link* URL. Penghilangan awalan dan akhiran ini bisa dilakukan karena tanpa menggunakan kata

yang dihilangkan, *link* URL masih tetap dapat diakses dan memiliki nilai yang sama. Selain itu dalam proses ini juga dilakukan pengubahan *link* URL menjadi *lower case*, hal ini dilakukan agar jenis kata yang akan disimpan sebagai kamus kata memiliki macam yang lebih sedikit.

3) Encoding Berbasis Kamus

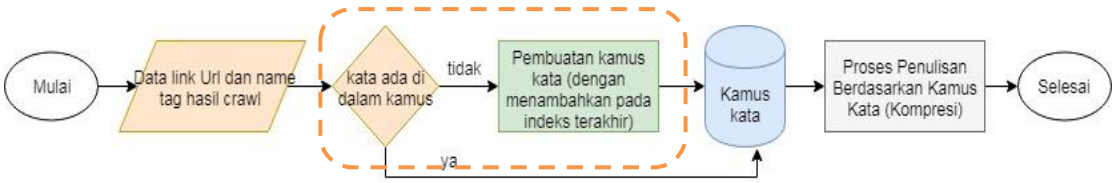
Langkah selanjutnya adalah melakukan *encoding* pada data yang telah dilakukan pengurangan awalan dan akhiran. Tahapan *encoding* dimulai dengan memasukkan *link* URL yang kemudian dilanjutkan dengan melakukan proses *tokenizing* untuk mengambil kata yang dipisahkan oleh karakter-karakter seperti "/", ",", " ", ".", ":", "=", "\", "-", "_", "!", "?", "&", "%", "$", "#", dan seterusnya. Setelah dilakukan pemisahan kata menurut karakter pemisah, tahap selanjutnya adalah membuat kamus kata. Proses pembuatan kamus kata berbeda untuk tahap penyiapan data awal serta proses selanjutnya. Pada penyiapan data awal, kompresi data *link* URL dilakukan dari hasil pencarian *link* URL sebanyak 5.000 *link crawl*. Hasil disimpan di dalam *file* txt. Hal ini dilakukan untuk mempermudah sistem dalam melakukan pembuatan kamus kata dengan menghitung frekuensi kemunculan kata kompresi dalam satu waktu.

Selanjutnya, proses kompresi akan dilakukan setiap kali proses pencarian atau *crawling* selesai dilakukan dan hasil *link* URL dan *tag* nama telah ditemukan. Cara yang dilakukan adalah dengan memeriksa kamus kata apakah sudah memiliki kata tersebut. Jika belum ada, maka kata akan diberikan indeks terakhir kamus kata ditambah 1, dan kemudian disimpan ke *history link* URL. Dengan adanya pengurutan indeks sesuai dengan frekuensi kata, maka kata yang sering muncul dalam pencarian akan lebih cepat didapatkan karena kata terbanyak dimasukkan ke dalam indeks terkecil dan sebaliknya. Alur kompresi tahap kedua dapat dilihat dalam Gambar 3 dan Gambar 4, tanda garis putus-putus merah merupakan step yang membedakan kedua proses.

Dalam pembuatan kamus kata, yang dilakukan adalah membuat data tabel yang berisi kata dan kode pada tiap kata yang ditemukan. Kode tabel didapatkan dari nilai frekuensi tiap-tiap kata, setelah diurutkan barulah nilai urutan tersebut dijadikan kode kata pada tabel kamus kata. Tabel frekuensi ditunjukkan pada Tabel II, dan tabel kamus kata ditunjukkan pada Tabel III.

Tabel III merupakan hasil pengurutan penyimpanan kata, berdasarkan frekuensi kata yang ditemukan pada Tabel II. Semakin banyak kata ditemukan di dalam masukan, maka kata tersebut akan mendapat nilai kode atau indeks terkecil, dan akan dicatat ke dalam kamus kata terlebih dahulu. Dalam pembuatan kamus kata ini, kode yang muncul adalah *byte*. Setiap kata akan menjadi 2 *byte* dalam penyimpanan, ini dilakukan karena kode yang disimpan merupakan representasi dari variasi nilai yang dapat dituliskan ke dalam bilangan biner. Dengan melakukan penyimpanan menggunakan 2 *byte* maka banyaknya jumlah jenis kata yang bisa digunakan adalah 2 ^ 16. Bit 2 ^ 16 memiliki variasi nilai bilangan biner sebanyak 0 – 65.535 jenis.

Gambar. 3. Pembuatan kamus kata awal proses



Gambar. 4. Alur pembuatan kamus kata proses selanjutnya

TABEL II
JUMLAH FREKUENSI KATA

Kata	Jumlah
liputan6	10
detik	3
news	10
viva	6
.	20
tribun	1
com	10

TABEL III
KAMUS KATA

Kata	Kode
.	1
news	2
liputan	3
com	4
viva	5
detik	6
tribun	7

```
================================================================================
Begin
//proses input kata pencarian
set  kata
set  pembatas
part                ◄── split kata
//proses encoding kata pencarian
ListKata            ◄── add part
for all ListKata[i] do
      enc           ◄── get dic_encode kata
      b             ◄── convert nilai integer enc ke bentuk byte
      pat           ◄── add b ke bentuk string
end for
 txt                ◄── add data pada history link
//proses pencarian KMP & decoding
ListindexBatas      ◄── add indexkatahasilpencarian KMP dengan inputan (pembatas, txt)
ListindexKata       ◄── add indexkatahasilpencarian KMP dengan inputan (pat, txt)
for ListindexKata [i]
      rentang       ◄── mencari dua nilai terdekat dari ListindexBatas dengan inputan ListindexKata [i]
      hasil         ◄── add hasil decode byte yang ada pada index rentang.
End for
End
================================================================================
```

Gambar. 5. *Pseudocode* pencarian hasil *link* URL

4) Simpan Data Hasil Encoding

Pada langkah ini, *link* URL dan *tag* nama yang telah dikumpulkan diubah dengan menulisnya sesuai indeks kata pada kamus kata yang sudah dibuat. Setelah indeks kata ditemukan barulah indeks diubah menjadi 2 *byte* dan ditulis ke dalam *file* (*test-encode*.txt), *file* ini merupakan data hasil *encoding*.

C. Pencarian Hasil Link URL

Langkah ini menunjukkan bagaimana mencari *link* URL dari data *history* yang telah dilakukan proses *encode* agar dapat memunculkan hasil pencarian. Gambar 5 merupakan *pseudocode* pencarian hasil *link* URL.

1) Input Kata Pencarian

Pada langkah ini kata pencarian yang digunakan adalah kata pencarian berupa *string* yang dimasukkan oleh pengguna. *String* yang dimasukkan dapat berupa sebuah kata atau lebih.

2) Encoding Kata Pencarian

Langkah selanjutnya adalah memasukkan kata pencarian, misalnya "detik". Kata yang dimasukkan akan melalui proses *encoding* menggunakan tabel kamus kata. Kata "detik" misalnya, maka akan diubah menjadi angka 6. *String* "detik.com" akan ditulis *ke dalam* penyimpanan sebagai 614. Penulisan dilakukan dengan menyimpan konversi 2 *byte* pada masing-masing karakter angka. Digit angka 4 akan disimpan dalam pola biner 0000000000000110.

3) Pencarian Menggunakan KMP (Knutt Morris Pratt)

Tahap selanjutnya adalah pencarian kata pada data *history* yang sesuai, yaitu dengan melakukan pencocokan kata yang bisa disebut dengan pola pada seluruh kata atau pola yang ada di dalam data *history*. Pencocokan dilakukan dengan memasukkan kata pencarian yang telah melalui proses *encoding*, lalu mencocokkan satu per satu dari awal kata hingga akhir kata dalam data *history link* URL yang disimpan dalam bentuk *encoded*. Karena pada kompresi *multilevel* ini penyimpanan setiap indeks kata disimpan ke dalam 2 *byte binary* maka pencarian dengan menggunakan algoritma KMP juga dilakukan dengan melakukan pencocokan per 2 *byte binary*.

4) Decoding Kata Masukan

Proses selanjutnya adalah proses *decoding* kata yang dimasukkan, yang ditemukan di dalam *file* yang telah melalui proses *encoding*. Karena proses pencocokan *string* menggunakan algoritma KMP hanya menemukan letak

indeks dari kata masukan yang dicocokkan, maka perlu adanya skenario agar *link* URL yang mengandung kata itu bisa di ambil sehingga hasil akhir keluaran berupa *link* URL dan *tag* nama. Cara yang dilakukan adalah dengan menambahkan *string* pembatas antar *link* URL yaitu karakter "|". Jika letak *string-string* pembatas *link* URL telah ditemukan (dengan menggunakan pencocokan KMP) selanjutnya akan dilakukan perhitungan dua jarak terdekat antara indeks letak kata masukan dengan letak indeks dua *string* batasan *link* URL. Berikut adalah contoh masukan halaman *link* URL web beserta *tag* nama, di mana untuk pembatas adalah karakter "|". Pembatas ditandai kotak merah dengan garis putus-putus. Contoh di bawah merupakan 2 *link* URL dan *tag* nama. Indeks ke 0 merupakan karakter "|" dan indeks ke 88 adalah "|" pada akhir kata. Gambar 6 merupakan visualisasi indeks pada pembatas *link* URL dan kata yang dimasukkan di dalam data.

Kata yang dimasukkan misalnya adalah kata "hijab" maka akan dilakukan pencarian menggunakan KMP dan diperoleh dua indeks yakni indeks ke 76 dan indeks ke 83. Selain pencarian letak indeks masukan kata, indeks pembatas *link* URL juga dicari. Untuk contoh di atas, indeks perhitungan rentang indeks untuk pembatas *link* URL adalah ada pada indeks berikut 0, 46, dan 88. Setelah itu perhitungan nilai terdekat dilakukan dengan posisi pencarian adalah kata masukan, dan untuk pencarian 2 titik terdekat dilakukan menggunakan data indeks pembatas *link* URL. Gambar 7 memberikan contoh perhitungan pencarian rentang indeks yang menunjukkan *link* URL di dalam data teks. Setelah dilakukan pencarian dua nilai indeks *link* URL terdekat dari indeks kata masukan. Hasil rentang diambil dan dilakukan proses *decoding binary* dengan kamus kata yang telah dibuat. Proses *decoding* dimulai dengan pembacaan tiap angka dan mengambil *byte* untuk dikonversikan ke dalam kata atau karakter sebenarnya.

D. Pengujian

Evaluasi performa metode yang diusulkan dilakukan dengan menghitung nilai presisi dan *recall*, perhitungan rasio kompresi, dan perhitungan kecepatan pencarian. Penghitungan nilai presisi dilakukan dengan rumusan sebagai berikut:

$$p = a/(a+b) \tag{1}$$

di mana p = presisi, a = *page* yang memiliki konten sama dengan pencarian (bernilai benar), dan b = web *page* yang memiliki konten berbeda dengan pencarian (bernilai salah). Penghitungan *recall* dilakukan sebagai berikut:

$$r = a/c \tag{2}$$

di mana r = *recall*, a = *page* yang memiliki konten sama dengan pencarian (bernilai benar), dan c = seluruh *page* yang berhasil ditemukan di dalam data set. Selanjutnya adalah penghitungan rasio kompresi sebagai berikut:

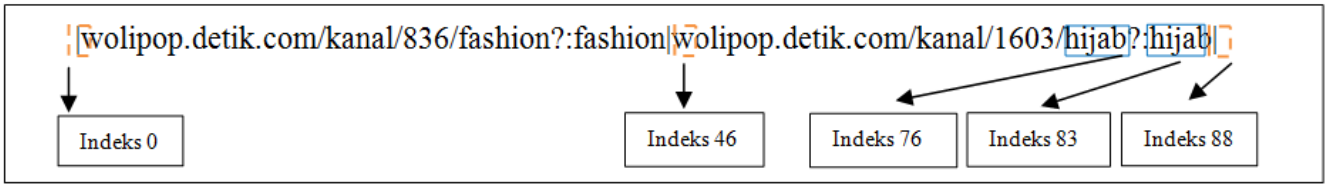
$$rasio\ kompresi = 100\ x\ \left(1 - \frac{ukuran\ setelah\ kompresi}{ukuran\ sebelum\ kompresi}\right) \tag{3}$$

Selanjutnya adalah untuk penghitungan kecepatan pencarian kata dilakukan dengan rumusan sebagai berikut:

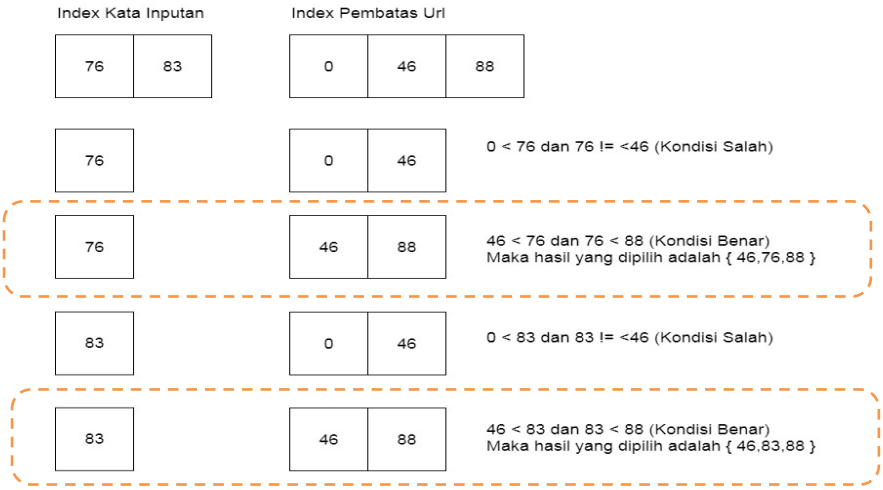
$$waktu\ proses = waktu\ proses\ selesai - waktu\ mulai \tag{4}$$

IV. HASIL DAN PEMBAHASAN

Pengujian yang dilakukan meliputi: perhitungan rasio keberhasilan metode kompresi, ketepatan hasil pencarian kata meliputi perhitungan nilai presisi dan juga *recall*, serta perhitungan lama waktu proses pencarian kata. Pengujian akan dilakukan pada dua metode pencarian yaitu *Focused Web Crawler* + *Cuckoo Search* + KMP (penelitian sebelumnya) atau disingkat dengan FWC dan *Focused Web Crawler* + *Cuckoo Search* + *Kompresi multilevel* + KMP atau disingkat dengan FWCM.



Gambar. 6. Visualisasi indeks pembatas *link* URL dan kata masukan



Gambar. 7. Pengambilan rentang

TABEL IV
JUMLAH HALAMAN YANG DIHASILKAN SETIAP KATA MASUKAN

Jenis Masukan (Jumlah Kata Masukan)	Kata	Jumlah *Link* URL		Selisih
		FWC	FWCM	
1	lari	278	76	202
1	nasi	15320	43	15277
1	menunggu	89	86	3
1	pemadaman	23	20	3
1	kesehatan	50	230	-180
2	orang tewas	59	144	-85
2	investasi migas	16	16	0
2	ratusan tahanan	2	2	0
2	mengulik konsolidasi	2	2	0
2	ikuti kejuaraan	14	14	0
3	buka konser celine	6	6	0
3	kpk tahan anggota	1	1	0
3	untuk ibu hamil	3	3	0
3	sahabat putri diana	6	6	0
3	meminimalisir nyeri rahang	2	2	0

A. Data Uji Coba

Data uji coba yang dipakai dalam penelitian ini adalah data hasil *Focused Web Crawler* dari beberapa *website* berita seperti https://www.detik.com/, https://www.tribunnews.com, https://www.cnnindonesia.com/, https://www.viva.co.id/, dan http://www.liputan6.com. Dari data *link* URL sumber dilakukan kunjungan sebanyak 6.000 *link* URL dan mengambil setiap *link* URL dan *tag* nama untuk pembuatan bank data pada proses awal.

B. Uji Coba Presisi dan Recall

Uji coba dilakukan dalam pencarian beberapa kata dan frasa. Hasil yang diperoleh diperiksa secara manual dengan melihat kebenaran penemuan kata yang dicari dari hasil *link* URL yang didapatkan oleh pencarian. Tabel IV, Tabel V, Tabel VI dan Tabel VII menunjukkan hasil pencarian pada dua metode pencarian. Pencarian dilakukan terhadap beberapa masukan kata yaitu percobaan yang dilakukan dari satu kata, dua kata dan tiga kata yang dimasukkan oleh *user*. Dalam percobaan ini, presisi dan *recall* dievaluasi terhadap hasil *link* URL yang ada pada 5.000 halaman web hasil proses *crawl* dengan total *link* URL yang tersimpan adalah 104.586. Tabel IV menunjukkan jumlah *link* URL yang didapatkan pada setiap masukan kata pada kedua metode.

Dari pengamatan pada hasil *link* URL yang ditemukan, pada beberapa jumlah masukan kata (1, 2, dan 3 kata) terdapat perbedaan jumlah yaitu pada satu masukan kata. Dari jumlah hasil *link* URL yang didapat, rata-rata menunjukkan bahwa pencarian tanpa kompresi *multilevel* mendapatkan hasil yang lebih banyak. Namun untuk jumlah masukan yang lebih banyak (2 atau 3 kata) hasil *link* URL yang didapatkan adalah sama. Dari pengamatan yang dilakukan, perbedaan jumlah hasil *link* URL yang terjadi disebabkan oleh jenis *link* URL yang dihasilkan. Terdapat dua jenis hasil yaitu *link* URL kata di mana hasil pencocokan kata sama persis dengan masukan, dan *link* URL kata di mana hasil pencocokan kata mengandung kata masukan. Tabel V menunjukkan data hasil *link* URL kedua metode pencarian. Kata yang sama persis dengan masukan kata, dimasukkan dalam *field* sesuai dengan masukan. Untuk penemuan *link* URL yang mengandung kata masukan, dimasukkan dalam *field* kata masukan ada di dalam kata lain.

TABEL V
PERBANDINGAN JUMLAH *LINK* URL HASIL TEMUAN (MEMILIKI NILAI BENAR)

Jenis Masukan (Jumlah Kata Masukan)	Kata	Sesuai dengan Masukan		Masukan Ada di Dalam Kata Lain	
		FWC	FWCM	FWC	FWCM
1	lari	81	76	178	1
1	nasi	38	43	15020	2
1	menunggu	82	86	22	16
1	pemadaman	20	20	10	8
1	kesehatan	49	0	8	8
2	orang tewas	58	119	0	0
2	investasi migas	7	5	6	6
2	ratusan tahanan	0	2	0	0
2	mengulik konsolidasi	0	2	0	0
2	ikuti kejuaraan	0	0	0	0
3	buka konser celine	6	6	0	0
3	kpk tahan anggota	0	0	1	1
3	untuk ibu hamil	3	0	0	0
3	sahabat putri diana	6	0	0	0
3	meminimalisir nyeri rahang	2	0	0	0

TABEL VI
PRESISI SETIAP KATA MASUKAN

Jenis Masukan (Jumlah Kata Masukan)	Kata	Sesuai dengan Masukan		Masukan Ada di Dalam Kata Lain	
		FWC	FWCM	FWC	FWCM
1	lari	0,29137	1,00000	0,64029	0,01316
1	nasi	0,00200	1,00000	0,98000	0,04700
1	menunggu	0,92135	1,00000	0,24719	0,18605
1	pemadaman	0,86957	1,00000	0,43478	0,40000
1	kesehatan	0,98325	0,43058	0,03349	0,00604
2	orang tewas	0,98305	0,82639	0,00000	0,00000
2	investasi migas	0,43750	0,31250	0,37500	0,37500
2	ratusan tahanan	1,00000	1,00000	0,00000	0,00000
2	mengulik konsolidasi	1,00000	1,00000	0,00000	0,00000
2	ikuti kejuaraan	1,00000	0.00000	0,00000	0,00000
3	buka konser celine	1,00000	1,00000	0,00000	0,00000
3	kpk tahan anggota	0,00000	0,00000	1,00000	1,00000
3	untuk ibu hamil	1,00000	0,00000	0,00000	0,00000
3	sahabat putrid diana	1,00000	0,00000	0,00000	0,00000
3	meminimalisir nyeri rahang	1,00000	0,00000	0,00000	0,00000

TABEL VII
RECALL SETIAP KATA MASUKAN

Jenis Masukan (Jumlah Kata Masukan)	Kata	Sesuai dengan Masukan		Masukan Ada di Dalam Kata Lain	
		FWC	FWCM	FWC	FWCM
1	lari	0,00397	0,00873	0,00579	0,00008
1	nasi	0,00186	0,73685	0,00328	0,00015
1	menunggu	0,04029	0,03276	0,01081	0,00610
1	pemadaman	0,00983	0,00762	0,00491	0,00305
1	kesehatan	0,20197	0,16305	0,00688	0,00229
2	orang tewas	0,01659	0,03363	0,00000	0,00000
2	investasi migas	0,00200	0,00141	0,00172	0,00170
2	ratusan tahanan	0,02667	0,02740	0,00000	0,00000
2	mengulik konsolidasi	0,18667	0,17808	0,00000	0,00000
2	ikuti kejuaraan	0,18667	0,00000	0,00000	0,00000
3	buka konser celine	0,01749	0,01749	0,00000	0,00000
3	kpk tahan anggota	0,00000	0,00000	0,00292	0,00292
3	untuk ibu hamil	0,00875	0,00000	0,00000	0,00000
3	sahabat putri diana	0,01749	0,00000	0,00000	0,00000
3	meminimalisir nyeri rahang	0,02273	0,00000	0,00000	0,00000

Tabel V merupakan tabel pencatatan hasil *link* URL yang relevan. Pada hasil pengamatan data, perbedaan hasil *link* URL relevan terbanyak pada kedua metode terjadi pada masukan satu kata. Sedangkan untuk masukan kata lainnya tidak terlalu memiliki perbedaan yang signifikan. Pada kedua metode FWC dan FWCM menunjukkan bahwa jumlah *link* URL relevan lebih banyak ditemukan untuk *link* URL hasil yang sesuai dengan kata masukan. Sedangkan untuk rata-rata *link* URL relevan yang ditemukan metode FWC memiliki temuan *link* URL relevan rata-rata lebih banyak dari metode FWCM. Beberapa contoh percobaan metode FWCM menunjukkan bahwa *link* URL yang ditemukan bukan *link* URL yang relevan, percobaan itu antara lain: “untuk ibu hamil” dan ”sahabat putri diana”. *Link* URL tidak relevan terjadi karena walaupun kata masukan ditemukan di dalam data yang telah melalui

proses *encoding* dan berhasil dilakukan proses *decoding*, namun tidak menghasilkan potongan *link* URL dan *tag* nama yang utuh. Hal ini terjadi karena pada metode kompresi *multilevel* (FWCM) apabila indeks pembatas *link* URL dan *tag* nama hasil pencarian KMP menghasilkan bilangan ganjil maka *link* URL dan *tag* nama tidak utuh, *link* URL tidak utuh ini dikatakan bukan *link* URL yang relevan. Selanjutnya, hasil Tabel V digunakan untuk menghitung presisi dan *recall* masing-masing percobaan pencarian kata. Penghitungan presisi dan *recall* hasil uji coba pada setiap kata masukan akan ditunjukkan pada Tabel VI dan Tabel VII. Tabel VI merupakan tabel penghitungan presisi masukan kata pada hasil *link* URL temuan yang sesuai dengan kata masukan (*link* URL temuan mengandung kata masukan atau masukan kata ada pada kata lain). Sedangkan Tabel VII adalah tabel penghitungan *recall* pada kedua jenis hasil *link* URL temuan.

Hasil penghitungan presisi dan *recall* sangat bergantung pada relevansi halaman *link* URL yang ada pada Tabel V. Metode dengan kompresi *multilevel* menunjukkan presisi yang lebih baik untuk jenis hasil *link* URL yang sama dengan masukan kata daripada hasil *link* URL yang mirip dengan masukan kata. Untuk jenis jumlah masukan kata, semakin banyak jumlah masukan maka nilai presisi kedua metode menunjukkan nilai yang sama kecuali pada kasus metode *multilevel*, di mana temuan kata masukan pada *link* URL yang tidak lengkap dianggap sebagai *link* URL yang tidak relevan. Nilai *recall* yang diperoleh dari *link* URL relevan yang dibagi dengan semua *link* URL hasil *crawler* menunjukkan bahwa metode FWCM memiliki nilai *recall* yang tinggi pada jenis *link* URL hasil yang sama dengan kata masukan. Untuk *link* URL hasil yang mengandung kata masukan masih menghasilkan nilai *recall*, seperti percobaan pada kata "lari" memiliki hasil 0,00008 yang memiliki selisih nilai lebih kecil sebesar 0,00571 daripada FWC yaitu sebesar 0,00579.

TABEL VIII
RASIO KOMPRESI DATA *LINK* URL HASIL *CRAWLING*

Banyak Halaman	Ukuran Sebelum Kompresi (*byte*)	Ukuran Setelah Kompresi (*byte*)	Rasio
1.000	3.010.560	1.912.832	36,5
2.000	5.914.624	3.764.224	36,4
3.000	8.912.896	5.660.672	36,5
4.000	11.751.424	7.483.392	36,3
5.000	14.839.808	9.445.376	36,3
6.000	17.719.296	11.284.480	36,3

TABEL IX
PERBANDINGAN HASIL *DECODING* DENGAN METODE KOMPRESI *MULTILEVEL*

Banyak Halaman	Ukuran Sebelum (*byte*)	Ukuran Setelah (*byte*)	Selisih (*byte*)	Hasil *Link* URL
1.000	3.010.560	3.010.560	0	sesuai
2.000	5.914.624	5.910.528	4.096	sesuai
3.000	8.912.896	8.904.704	8.192	sesuai
4.000	11.751.424	11.743.232	8.192	sesuai
5.000	14.839.808	14.827.520	12.288	sesuai
6.000	17.719.296	46.620.672	-28.901.376	Tidak sesuai

TABEL X
PERBANDINGAN WAKTU PENCARIAN METODE *FOCUSED WEB CRAWLER* (SAMPAI MUNCUL HASIL *LINK* URL DAN *TAG* NAMA)

Kata	FWCM (*milisecond*)	FWC (*milisecond*)	Selisih
dipatuk king cobra	3.963	541	3.422
meminimalisir nyeri rahang	564	548	16
merusak terumbu karang	614	615	-1
jus buah kemasan	548	508	40
janji perbaiki aplikasi	2.048	830	1.218
tak berpotensi tsunami	641	600	41
berhasil dievakuasi nasional	666	577	89
tingkatkan penerimaan negara	716	550	166

TABEL XI
PERBANDINGAN WAKTU PENCARIAN INDEKS KATA DENGAN KMP (TANPA MUNCUL HASIL *LINK* URL DAN *TAG* NAMA)

Kata	FWCM (*millisecond*)	FWC (*millisecond*)	Selisih
dipatuk king cobra	519	380	139
meminimalisir nyeri rahang	415	388	27
merusak terumbu karang	413	470	-57
jus buah kemasan	475	478	-3
janji perbaiki aplikasi	462	500	-38
tak berpotensi tsunami	488	544	-56
berhasil dievakuasi nasional	486	542	-56
tingkatkan penerimaan negara	492	502	-10

C. Uji Coba Rasio Kompresi

Dalam tahap ini, dilakukan percobaan untuk melakukan kompresi terhadap data *link* URL yang telah dikumpulkan untuk kemudian dilakukan penyimpanan *history* pada *link* URL. Tujuan dari tahap ini adalah untuk mengetahui seberapa besar rasio dari metode kompresi yang diajukan. Dalam uji coba ini beberapa *file* kumpulan data *link* URL dan *tag* nama dengan jumlah *link* URL berbeda dari 1.000, 2.000, 3.000, 4.000, 5.000, dan 6.000 akan dibandingkan seberapa banyak ukuran yang dapat dikurangi. Untuk hasil kompresi pada masing-masing hasil *crawl* pada jumlah *link* URL yang berbeda akan ditampilkan pada Tabel VIII.

Semua hasil percobaan menunjukkan bahwa metode yang diusulkan mampu menghemat ukuran *file* rata-rata dengan rasio 36%. Selain mencatat keberhasilan kompresi, akan dilakukan percobaan untuk melihat kebenaran hasil kompresi yaitu dengan melakukan proses *decoding* pada *file* terkompresi. Parameter yang digunakan untuk penilaian adalah ukuran *file* sebelum dan sesudah beserta kesesuaian nilai hasil *decoding*. Hasil evaluasi kebenaran hasil ditampilkan di dalam Tabel IX.

Pada tabel perbandingan ukuran *file* sebelum dan sesudah, rata-rata ukuran sebelum memiliki *size* yang lebih besar namun selisih ukuran tidak terlalu jauh berbeda. Perbedaan terjadi karena untuk proses *decoding* pada data yang telah melalui proses *encoding*, untuk karakter pergantian baris di dalam *file* teks tidak dilakukan, sehingga hanya mengurangi sedikit ukuran *file* asli dengan isi *file* yang sesuai. Kompresi data hasil *crawl* 6.000 *link* URL jenis kata yang dimasukkan dalam kamus kata, menghasilkan 66.239 jenis kata yang telah melebihi batas maksimal jenis kata yaitu sebesar 65.536. Oleh karena itu terjadi masalah karena pembacaan untuk konversi integer menjadi *byte* maupun sebaliknya. Pada proses *encoding* maupun *decoding* hanya disediakan sampai 2 *byte* dengan maksimum jumlah nilai indeks adalah 2 ^ 16 atau sebanyak 65.536. Pada hasil pengamatan dari percobaan beberapa *link* URL di atas menunjukkan kesalahan *link* URL terjadi saat mencapai 5.951 *crawl link* URL, sehingga maksimum halaman yang dapat terkompresi dalam kondisi data *link* URL pada saat kompresi hasil 5.951 *crawl link* URL data sekitar 123.992 *link* URL dan *tag* nama.

D. Uji Coba Waktu Pencarian

Dalam uji coba waktu pencarian, data masukan yang digunakan adalah kata masukan yang terdiri dari tiga kata masukan. Tiga kata masukan dicoba karena kata masukan ini memiliki hasil pencarian *link* URL dengan jumlah yang sama besar, sedangkan untuk kata masukan lain masih banyak perbedaan jumlah kata pencarian terutama jika kata masukan hanya 1 dan merupakan kata dasar. Dari pencarian kata akan dilakukan pencatatan waktu dan akan ditampilkan pada Tabel X.

Pencatatan waktu pencarian dihitung dimulai dari kata masukan dimasukkan oleh *user* sampai ditemukan *link* URL *website* yang memiliki kata masukan. Dari hasil yang didapatkan, terlihat bahwa metode *Focused Web Crawler* memiliki waktu yang lebih lama untuk sama-sama menghasilkan *link* URL dan *tag* nama. Ini terjadi karena perbedaan langkah dalam menghasilkan *link* URL dan *tag* nama. Pada metode *Focused Web Crawler* untuk dapat menampilkan hasil output *link* URL seperti sebelum terkompresi, setelah melakukan pencarian pada data yang mengalami proses kompresi harus dilakukan proses *decoding*. Sedangkan pada metode tanpa kompresi *multilevel*, setelah pencarian kata masukan dilakukan, hasil *link* URL dan *tag* nama merupakan hasil akhir yang bisa dibaca oleh *user* tanpa perlu adanya proses *decoding*.

Metode *Focused Web Crawler* dengan kompresi *multilevel* berguna untuk mengurangi jumlah ukuran *byte* data *history link* URL. Dari pemikiran ini maka akan dilakukan pengamatan pada waktu pencarian dengan algoritma KMP sampai ditemukan indeks kata masukan di dalam teks, tanpa menampilkan hasil *output link* URL. Percobaan menggunakan data masukan sama seperti pencarian untuk menghasilkan *link* URL dan *tag* nama sebelumnya. Berikut hasil pengamatan waktu pencarian yang ditampilkan pada Tabel XI.

Pencatatan waktu pencarian dihitung dimulai dari kata masukan dimasukkan oleh *user* sampai ditemukan indeks kata ditemukan pada data yang disimpan. Dari hasil pengamatan, waktu untuk proses pencarian dengan KMP pada metode *Focused Web Crawler* dengan kompresi *multilevel* rata-rata memiliki nilai waktu yang lebih sedikit daripada *Focused Web Crawler* tanpa *multilevel*. Dari percobaan ini menjelaskan bahwa yang membuat pencarian *Focused Web Crawler* dengan kompresi *multilevel* lebih lama adalah proses untuk melakukan *decoding* pada data yang hasil proses *encoding* yang telah disimpan.

V. KESIMPULAN

Hasil pencarian *link* URL dengan metode *Focused Web Crawler + Cuckoo Search +* kompresi *multilevel* dapat digunakan untuk melakukan pencarian pada beberapa macam masukan kata (1, 2, atau 3 kata), baik masukan berupa kata dasar maupun kata berimbuhan. Nilai *recall* dan presisi metode pencarian dengan kompresi *multilevel* memiliki nilai yang lebih tinggi daripada metode pencarian tanpa kompresi *multilevel* untuk *link* URL temuan yang mengandung kata yang sama dengan kata masukan. Sedangkan untuk *link* URL yang mengandung kata yang mengandung atau terdiri dari kata masukan, nilai presisi dan *recall* metode pencarian dengan kompresi *multilevel* lebih sedikit daripada metode tanpa kompresi *multilevel*. Metode kompresi *multilevel* yang diusulkan menghemat

rata-rata sekitar 36,4% kebutuhan ruang penyimpanan. Metode kompresi *multilevel* berbasis kamus kurang cocok untuk data yang memiliki kemungkinan banyak jenis data (data *distinct*). Untuk evaluasi waktu pencarian *link* URL dan *tag* nama, metode *Focused Web Crawler* dengan kompresi *multilevel* memerlukan waktu lebih lama dibandingkan metode *Focused Web Crawler* tanpa kompresi *multilevel*. Faktor penyebabnya adalah karena terdapat proses untuk melakukan *decoding* pada hasil pencarian dengan KMP pada data yang telah melalui proses *encoding*. Sedangkan untuk proses pencarian kata dengan KMP (tidak termasuk proses *decoding*) metode dengan kompresi *multilevel* menunjukkan waktu yang lebih cepat. Ini terjadi karena jumlah *byte* data *link* URL untuk dilakukan pencocokan telah berkurang akibat proses kompresi.

DAFTAR PUSTAKA

[1] M. Kan, "Fast webpage classification using URL features," dalam *Proceedings of the 14th ACM international conference on Information and knowledge management*, 2005, hal. 325–326.

[2] S. K. Dwivedi dan C. Arya, "News web page classification using url content and structure attributes," dalam *Proceedings on 2016 2nd International Conference on Next Generation Computing Technologies, NGCT 2016*, 2017, no. October, hal. 317–322.

[3] G. Pant dan P. Srinivasan, "Link contexts in classifier-guided topical crawlers," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, hal. 107–122, 2006.

[4] B. W. Yohanes, Handoko, dan H. K. Wardana, "Focused Crawler Optimization Using Genetic Algorithm," *Telkomnika*, vol. 9, no. 3, hal. 403–410, 2011.

[5] B. Ganguly dan D. Raich, "Performance optimization of focused web crawling using content block segmentation," dalam *Proceedings - International Conference on Electronic Systems, Signal Processing, and Computing Technologies, ICESC 2014*, 2014, hal. 365–370.

[6] J. Dewanjee, "Heuristic Approach for Designing a Focused Web Crawler using Cuckoo Search," dalam *Int. J. Comput. Sci. Eng.*, vol. 04, no. 09, hal. 59–63, 2016.

[7] I. Avraam, "A Comparison over Focused Web Crawling Strategies," dalam *Panhellenic Conference on Informatics*, 2011, hal. 245–249.

[8] Wikipedia, "Heuristik," 2018.

[9] G. E. S. S, "Kecerdasan Buatan (Metode Heuristic)," tidak dipublikasikan.

[10] P. C. Local, P. Hybridization, A. C. See, dan R. Further, "Metaheuristic," 2018.

[11] X. Yang, S. Deb, dan A. C. B. Behaviour, "Cuckoo Search via L ´ evy Flights," dalam *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 2009, pp. 210–214.

[12] X. Yang, S. Deb, N. World, dan M. A. Random, "Cuckoo search," 2018.

[13] D. Salomon, "Introduction ," dalam *Data compression*, edisi ke-3, New York, United States of America, 2004, bab I, hal. 1-14.

[14] D. a Lelewer dan D. S. Hirschberg, "Data Compression," *ACM Comput. Surv.*, vol. 19, no. 3, pp. 261–296, 2004.

[15] M. M. Kodabagi, "Multilevel Security and Compression of Text Data using Bit Stuffing and Huffman Coding," dalam *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, 2015, hal. 800–804.

[16] K. Kalajdzic, S. H. Ali, dan A. Patel, "Rapid lossless compression of short text messages," *Comput. Stand. Interfaces*, vol. 37, no. JUNE, hal. 53–59, 2015.

[17] A. Mahmood dan K. M. A. Hasan, "An Efficient 6 Bit Encoding Scheme for Printable Characters by Table Look Up," dalam *International Conference on Electrical, Computer and Communication Engineering (ECCE), February 16-18, 2017, Cox's Bazar, Bangladesh An*, 2017, hal. 468–472.

[18] S. Kanda, K. Morita, dan M. Fuketa, "Practical String Dictionary Compression Using String Dictionary Encoding," dalam *2017 International Conference on Big Data Innovations and Applications (Innovate-Data)*, 2017, hal. 4–11.