Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)
Vol. 3, No. 3, September 2016, hlm. 208-215

TF-IDF-*ENHANCED GENETIC ALGORITHM* UNTUK *EXTRACTIVE AUTOMATIC TEXT SUMMARIZATION*

Dhimas Anjar Prabowo[1], Muhammad Fhadli[2], Mochammad Ainun Najib[3], Handika Agus Fauzi[4] , Imam Cholissodin[5]

[1,2,3,4,5] Program Studi Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: dhimasanjar@gmail.com, muhammadfhadli@ymail.com, m.ec2.a.najib@gmail.com, handika830@gmail.com, imamcs@ub.ac.id

(Naskah masuk: 8 Agustus 2016, diterima untuk diterbitkan: 8 September 2016)

Abstrak

Penelitian ini mengusulkan sebuah implementasi terkait dengan automasi peringkasan teks bertipe ekstraktif dengan menggunakan metode TF-IDF-EGA. Dimana dalam permasalahan peringkasan teks dibutuhkan suatu solusi untuk meringkas teks dengan kalimat ringkasan yang dapat merepresentasikan keseluruhan teks yang ada. Algoritma TF-IDF dikenal mampu untuk menghasilkan ringkasan teks berdasarkan skor yang didapat pada setiap kalimat dalam teks. Namun hasil dari TF-IDF terkadang didapati hasil ringkasan yang terdiri dari kalimat yang tidak deskriptif, hal ini dikarenakan dalam peringkasannya TF-IDF hanya memilih beberapa kalimat yang memiliki skor tertinggi dan biasanya kalimat dengan skor tertinggi merupakan kalimat yang berisi kata-kata penting/kata-kata ilmiah tertentu sehingga kalimatnya tidak deskriptif. Algoritma EGA mampu untuk mengatasi permasalahan tersebut dengan cara memilih kalimat ringkasan yang memiliki nilai probabilitas tertentu sebagai hasil peringkasan teks.

Kata kunci: *peringkasan teks, automasi ekstraktif, TF-IDF, EGA, algoritma evolusi, meta-heuristik.*

Abstract

This research proposed an implementation related to extractive automatic text summarization using TF-IDF-EGA method. Which in summarization problem required a solution to summarize text with a sentence summary that could represent the whole data text. TF-IDF algorithm was usually known to be used for generating summary by its sentence scores. However the result from TF-IDF tends to generate a summary that consist of non descriptive sentences, this is due its summarize that only choose sentences with maximum score and usually sentences with maximum score is consist of significant words on a form of scientific word. EGA could solve this problem by choosing summary by its sentence probability values as a the whole text summary.

Keywords: *text summarization, extractive automation, TF-IDF, EGA, evolutionary algorithm, meta-heuristic.*

1. PENDAHULUAN

Bersama dengan pertumbuhan data dokumen di internet yang semakin besar, user menginginkan data relevan dalam satu tempat tertentu tanpa perlu membaca terlebih dahulu sekian banyak dokumen yang ada, dimana hal ini merupakan sebab dari munculnya *Automatic Text Summarization* (Meena & Gopalani, 2015). Problem ini juga ditemui di Fakultas MIPA Universitas Brawijaya, dimana *biosystem* disana memerlukan suatu fitur peringkasan teks untuk meringkas dokumen penelitian dari berbagai hasil pencarian terhadap senyawa dalam tumbuhan herbal tertentu. Peringkasan teks ini akan mampu memudahkan peneliti/pembacanya untuk mendapatkan inti dari dokumen penelitian terkait dari senyawa dalam tumbuhan herbal yang diteliti (Fhadli et al., 2016).

Dalam *biosystem* yang telah ada, peringkasan teks yang digunakan adalah dengan metode *Term Frequency-Inverse Document Frequency* (TF-IDF). Metode ini mampu untuk mengukur skor/berat dari setiap kalimat dalam dokumen, dimana 2 kalimat dengan nilai skor/bobot tertinggi akan dipilih sebagai ringkasan dari keseluruhan dokumen (Fhadli et al., 2016). Namun, pemilihan berdasarkan data terbesar ini membuat kalimat dari ringkasan yang dihasilkan terkadang terbaca tidak nyambung. Hal ini disebabkan karena kebanyakan kalimat yang memiliki skor tertinggi merupakan kalimat yang mengandung kata yang penting, dimana dalam dokumen berupa penelitian yang digunakan sering berisi dengan simbol ilmiah tertentu yang akan terbaca oleh sistem sebagai kata penting. Sehingga kalimat pertama dari ringkasan akan sering terbaca tidak nyambung dengan kalimat selanjutnya.

Dalam kurun waktu beberapa tahun terakhir telah dikemukakan beberapa metode atau teknik untuk mengatasi *problem* peringkasan teks, seperti *clustering*, *swarm intelligence*, *evolutionary algorithm* (EA), serta metode *hybrid* tertentu. Sebagai contoh untuk bidang EA adalah *Hybrid fuzzy* GA-GP, dimana GA digunakan untuk bagian *string*

(membership functions), dan GP digunakan untuk bagian struktural, sedangkan *fuzzy inference* digunakan untuk menghilangkan ketidakpastian atau keambiguan dalam menyeleksi nilai (Kiani & Akbarzadeh, 2006). Kemudian ada *Differential Evolution Algorithm*, dimana metode ini digunakan untuk mengoptimasi proses *clustering* data dan untuk meningkatkan kualitas dari ringkasan teks yang dihasilkan (Abuobieda et al., 2013). Lalu pada penelitian yang dilakukan oleh (Meena & Gopalani, 2015) tentang algoritma evolusi dalam automasi ekstrasi peringkasan teks, yang membuktikan bahwa algoritma evolusi mampu untuk meningkatkan kualitas dari hasil peringkasan teks menggunakan metode *Genetic Algorithm* (GA). Dimana fungsi fitness dalam GA dapat menemukan kalimat dengan bobot yang lebih optimal dibandingkan dengan metode peringkasan teks biasa. Penelitian lain dilakukan oleh (Ghareb et al., 2015) tentang *enhanced* GA berbasis *hybrid feature selection* (FS) untuk pengkategorisasian teks dokumen, yang membuktikan bahwa algoritma yang dikemukakan tersebut mampu men-*generate* hasil yang lebih baik dibandingkan dengan algoritma kategorisasi biasa dan algoritma GA sendiri. Dimana kombinasi *hybrid* FS dengan EGA ini mampu untuk mereduksi dimensi dengan reduction rate serta categorization rate yang lebih tinggi/baik.

Dalam projek akhir ini akan mencoba memperbaiki hasil dari metode TF-IDF yang telah ada pada *biosytem* Fakultas MIPA Universitas Brawijaya dengan melibatkan metode *Enhanced Genetic Algorithm* (EGA) dalam menghasilkan ringkasan teks dari dokumen hasil pencarian yang ada. Dimana dalam prosesnya terdapat 2 tahap utama, yaitu penghitungan nilai skor dari setiap kalimat dalam dokumen menggunakan metode TF-IDF, lalu selanjutnya akan dilakukan pemilihan kalimat ringkasan dengan menggunakan metode EGA. Dimana diharapkan peringkasan teks hasil dari metode ini dapat terbaca lebih baik dibandingkan dengan hasil ringkasan teks menggunakan metode TF-IDF saja.

2. DASAR TEORI

2.1 Penjelasan Dataset

Pada studi kasus ini, digunakan data teks abstrak penelitian yang berasal dari hasil pencarian *biosystem* terhadap senyawa dalam tumbuhan herbal tertentu. Dimana data teks ini berbasis bahasa inggris, dikarenakan *biosystem* menggunakan *query* berbahasa inggris dengan data sumber dari NCBI.

2.2 *Extractive Automatic Text Summarization*

Secara umum, peringkasan teks dapat diartikan sebagai sebuah cara untuk meringkas sekumpulan informasi teks besar menjadi bentuk yang ringkas/singkat dengan proses seleksi informasi penting dengan menyisihkan informasi yang tidak penting dan berlebihan (Saranyamol & Sindhu, 2014). *Extractive Automatic Text Summarization* merupakan suatu proses untuk memilih beberapa kalimat tertentu dari sekumpulan data teks dokumen yang dimana nantinya kalimat yang terpilih ini akan digunakan sebagai ringkasan dari keseluruhan dokumen (Babar & Patil, 2015). Namun, dalam prosesnya diperlukan suatu persiapan data terlebih dahulu atau lebih dikenal sebagai *text preprocessing*, yang akan dijelaskan sebagai berikut:

2.2.1 *Text Preprocessing*

Text preprocessing merupakan suatu proses untuk merubah data teks menjadi sebuah *term* indeks yang akan digunakan dalam proses *weighting*/pembobotan. Dimana pada umumnya meliputi 4 tahap yaitu:

1. Tahap 1, *Parsing*: suatu proses untuk menentukan data teks apa yang akan digunakan. Dimana dalam *penelitian* ini data yang akan digunakan sebagai dokumen adalah setiap kalimat yang terdapat dalam abstrak dokumen hasil dari pencarian *biosystem*, yang bisa didapatkan dengan membagi setiap kalimat dalam abstrak menjadi dokumen.
2. Tahap 2, *Lexing/Tokenization*: suatu proses untuk memisahkan setiap string kata dalam kalimat, dan juga termasuk proses untuk menghapus duplikasi kata, angka, tanda baca, karakter-karakter lain selain huruf alfabet & simbol ilmiah, serta merubah setiap huruf kapital yang ada menjadi huruf kecil/dasar.
3. Tahap 3, *Filtering/Stopword Removal*: suatu proses untuk menghapus kata-kata tidak penting dalam kalimat dan hanya menyimpan kata-kata yang penting saja, dimana dalam proses ini digunakan sebuah kumpulan data teks berisi kata-kata penting yang telah ditetapkan sebelumnya.
4. Tahap 4, setelah 3 tahap sebelumnya telah selesai dilakukan maka setiap kata yang tersisa akan di-*set* sebagai *term* indeks, dimana term indeks ini akan digunakan untuk proses *weighting*/pembobotan.

2.3 Algoritma *Term Frequency-Inverse Document Frequency* (TF-IDF)

TF-IDF merupakan suatu proses *weighting*/pembobotan dimana akan dilakukan perhitungan *weight*/bobot terhadap setiap *term* indeks yang dihasilkan pada tahap *text preprocessing*.

Dimana terdapat 7 tahap dalam pengerjaannya, yaitu sebagai berikut:

1. Tahap 1, *Document Indexing*: suatu proses untuk menentukan *term* indeks (*t*) mana yang akan digunakan sebagai representasi dari dokumen, dimana dalam *penelitian* ini setiap setiap kata yang tersisa hasil tahap *preprocessing* akan digunakan sebagai *term* indeks.
2. Tahap 2, *Term Weighting*: suatu proses untuk men-*generate* sebuah nilai pada setiap term dengan cara menghitung frekuensi kemunculan *term* dalam dokumen (*d*).
3. Tahap 3, *Log-Frequency Weighting*: suatu proses untuk menghitung nilai bobot hasil dari nilai kemunculan *term weighting*, dimana digunakan rumus yang dapat dilihat dalam persamaan 2.1.

$$Wtf_{t,d} = \begin{cases} 1 + log_{10} tf_{t,d}, & jika\ tf_{t,d} > 0 \\ 0, & lainnya \end{cases} \quad (2.1)$$

dengan keterangan:

- $Wtf_{t,d}$ merupakan *log-frequency weighting* pada *term* ke *t*, dokumen ke *d*,
- $tf_{t,d}$ merupakan nilai dari *term weighting* pada *term* ke *t*, dokumen ke *d*.

4. Tahap 4, Document Frequency: suatu proses untuk menghitung banyaknya dokumen yang mengandung *term* ke *t*.
5. Tahap 5, Inverse Document Frequency: suatu proses untuk menghitung nilai *inverse* dari *document frequency*, dimana digunakan rumus yang dapat dilihat pada persamaan 2.2.

$$idf_t = log_{10} \frac{N}{df_t} \quad (2.2)$$

dengan keterangan:

- idf_t merupakan *inverse document frequency* pada *term* ke *t*,
- N merupakan jumlah keseluruhan dokumen yang ada,
- df_t merupakan nilai dari *document frequency* pada *term* ke *t*.

6. Tahap 6, TF-IDF: suatu proses untuk mendapatkan nilai skor setiap terhadap dokumen, dimana digunakan rumus yang dapat dilihat pada persamaan 2.3.

$$W_{t,d} = Wtf_{t,d} \times idf_t \quad (2.3)$$

dengan keterangan:

- $W_{t,d}$ merupakan TF-IDF pada *term* ke *t*, dokumen ke *d*,
- $Wtf_{t,d}$ merupakan *log-frequency weighting* pada *term* ke *t*, dokumen ke *d*,
- idf_t merupakan *inverse document frequency* pada *term* ke *t*.

7. Tahap 7, menghitung nilai skor akhir setiap dokumen, dimana digunakan rumus yang dapat dilihat pada persamaan 2.4.

$$Ws_j = \sum_{i=1}^{Nterm} Wtd_{i,j} \quad (2.4)$$

dengan keterangan:

- Ws_j merupakan skor dari dokumen ke j,
- Nterm merupakan jumlah banyaknya term,
- $Wtd_{i,j}$ merupakan nilai dari TF-IDF pada term ke *i*, dokumen ke *j*,

Dan perlu diingat bahwa setiap kalimat dari kumpulan data teks dianggap sebagai dokumen dalam perhitungan.

2.4 Algoritma *Enhanced Genetic Algorithm* (EGA)

EGA merupakan suatu metode pengembangan lanjut dari *Genetic Algorithm* (GA) yang termasuk dalam bidang meta-heuristik *Evolutionary Algorithm* (EA). Secara umum alur tahap pengerjaan dalam EGA cukup mirip dengan GA, hanya saja terdapat perbedaan yaitu terletak pada operator reproduksi mutasinya. Dalam *penelitian* ini operator reproduksi mutasi akan dikembangkan guna untuk mengatasi problem randomisasi nilai *string chromosome* terpilih yang dianggap beresiko untuk memberikan efek negatif terhadap keragaman populasi dan juga memberikan hasil randomisasi yang tidak baik (Ghareb et al., 2015).

2.4.1 Inisialisasi populasi

Tahap dimana akan dilakukan randomisasi *string chromosome* sejumlah banyak populasi yang ditetapkan (*popSize*), dimana setiap *string chromosome* ini akan merepresentasikan solusi akhir dari permasalahan. *String chromosome* terdiri atas sejumlah *subset* nilai skor hasil proses perhitungan metode TF-IDF, sehingga setiap *subset* merupakan representasi dari nilai skor kalimat yang ada dalam dokumen. Dimana dalam problem peringkasan teks ini akan dihasilkan 3 kalimat tertentu yang akan terpilih sebagai hasil ringkasan teks. Jadi, yang dirandomisasi adalah posisi letak kalimat dalam *string chromosome*. Sebagai gambaran, setiap *string chromosome* dapat digambarkan seperti yang ditunjukkan pada Gambar 2.1.

	<---------*string chromosome*--------->		
P1 =	[Ws_j,	Ws_j,	Ws_j]

Gambar 2.1 Contoh representasi *string chromosome*

2.4.2 Fungsi *fitness*

Sebuah fungsi *fitness* diperlukan untuk mengevaluasi kualitas dari *chromosome* dalam populasi. Dimana jika nilai *subset* dalam *chromosome* baik maka akan memiliki kemungkinan yang lebih tinggi untuk dapat lolos kedalam populasi berikutnya.

Rumus *fitness* yang digunakan dapat dilihat apa persamaan 2.5.

$$fitness_i = \sum_{j=1}^{Nsubset} (x_j \times Ws_j) \quad (2.5)$$

dengan keterangan:

- Nsubset merupakan banyaknya *subset* dalam *chromosome*
- x_j merupakan sebuah nilai x pada *subset* ke *j* dengan ketentuan jumlah keseluruhan dari x_j adalah 1
- Ws_j merupakan nilai skor kalimat pada *subset* ke *j*

2.4.3 Operator reproduksi

Sebuah operator yang digunakan untuk mempertahankan keragaman dari populasi, terdapat dua operator yang digunakan yaitu sebagai berikut:

1. *Crossover*: teknik yang digunakan adalah teknik *one-cut-point crossover*, yaitu dengan cara memilih dua *chromosome* induk dalam populasi lalu menukarkan isi nilai subset *chromosome*-nya mulai dari titik indeks tertentu. Hal ini dilakukan secara berulang hingga jumlah *offspring* memenuhi jumlah banyaknya *offspring* yang diperlukan dari hasil *crossover rate*.
2. Mutasi: teknik yang digunakan adalah mutasi yang telah di-*enhanced*, yaitu dengan cara memilih satu *chromosome* induk dalam populasi lalu mengganti nilai *subset* terendah dalam *chromosome* induk dengan nilai *subset* tertinggi dari *chromosome* terbaik dari populasi sebelumnya. Hal ini dilakukan secara berulang hingga jumlah *offspring* memenuhi jumlah banyaknya *offspring* yang diperlukan dari hasil *mutation rate*. Enhance mutasi ini kami adaptasi dari penelitian milik (Ghareb et al., 2015).

2.4.4 *Selection*

Sebuah proses seleksi yang dilakukan setelah proses evaluasi *fitness* dari hasil reproduksi. Teknik seleksi yang digunakan adalah dengan menggunakan *Roulette Wheel Selection* (RWS), teknik ini dilakukan dengan menghitung nilai probabilitas seleksi (*prob*) setiap *chromosome* berdasarkan nilai *fitness*-nya. Dari nilai *prob* ini dapat dihitung nilai probabilitas kumulatif (*probCum*) yang akan digunakan dalam proses seleksi. Rumus perhitungannya dapat dilihat pada persamaan 2.6 & 2.7.

$$prob_k = \frac{fitness_k}{totalFitness} \quad (2.6)$$

dengan keterangan:

- $prob_k$ merupakan probabilitas untuk *chromosome* ke *k*
- $fitness_k$ merupakan nilai *fitness* pada *chromosome* ke *k*
- totalFitness adalah jumlah keseluruhan *fitness chromosome* dalam populasi

$$probCum_k = \sum_{j=1}^{k} prob_k \quad (2.7)$$

dengan keterangan:

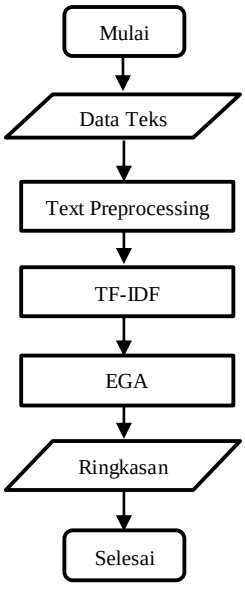
- $probCum_k$ merupakan probabilitas kumulatif untuk *chromosome* ke *k*
- $prob_k$ merupakan nilai probabilitas pada *chromosome* ke *k*

Setelah didapatkan nilai *probCum* untuk setiap chromosome maka dilakukan seleksi dengan men-*generate* nilai random *r* dengan batas *r*[0, 1], lalu memilih dan cek *chromosome* ke 1 hingga ke *k popSize*, dengan kondisi jika memenuhi r <= $probCum_k$ maka *chromosome* ke *k* tersebut terpilih untuk masuk ke populasi berikutnya. Proses randomisasi dan pengecekan ini dilakukan hingga terbentuk populasi baru sebanyak inisialisasi *popSize*.

3. PERANCANGAN DAN IMPLEMENTASI

3.1 Perancangan Alur Proses Algoritma

Dengan dasar teori dari poin 2, maka rancangan alur proses dari *extractive automatic text summarization* dengan TF-IDF-EGA dapat dilihat pada Gambar 3.1.



Gambar 3.2 Diagram Alur Proses TF-IDF-EGA

Dengan detail prosesnya adalah sebagai berikut:

1. Menentukan data teks input
2. Melakukan *preprocessing* teks

a) Melakukan *parsing*
b) Melakukan *lexing/tokenisasi*
c) Melakukan *filtering/stopword removal*
d) Menentukan term indeks

3. Proses TF-IDF
   a) Menghitung nilai *term frequency*
   b) Menghitung nilai *log-term frequency*
   c) Menghitung nilai *document frequency*
   d) Menghitung nilai *inverse document frequency*
   e) Menghitung nilai TF-IDF
   f) Menghitung nilai skor kalimat
4. Proses EGA
   a) Inisialisasi populasi awal
   b) Evaluasi nilai *fitness*
   c) Reproduksi *crossover* dan mutasi
   d) Evaluasi nilai *fitness* keseluruhan individu populasi dan individu hasil reproduksi
   e) Seleksi *roulette wheel selection* (RWS)
   f) Menentukan *global best* sebagai solusi terpilih
5. Menampilkan hasil ringkasan dengan urutan indeks kalimat dari *global best*

3.2 Perancangan Uji Coba

Setelah tahap implementasi selesai akan dilakukan serangkaian uji coba untuk dapat mengevaluasi dan menganalisis hasil yang ditemukan oleh sistem, dimana uji coba tersebut adalah sebagai berikut:

3.2.1 Uji Coba Batas Parameter Iterasi

Uji coba batas parameter iterasi adalah uji coba yang dilakukan untuk mengetahui seberapa banyak iterasi yang dibutuhkan agar dapat dihasilkan solusi peringkasan yang optimal. Dimana nilai *fitness* akan menjadi tolak ukur utama dalam menentukan nilai optimalnya. Rancangan uji coba parameter populasi dapat dilihat pada Tabel 3.1:

Tabel 3.1 Rancangan Uji Coba Parameter Iterasi

Jumlah Iterasi	Jumlah Populasi	Nilai *Fitness* Percobaan Ke-i			Rata Rata *Fitness*
		1	2	3	
10	10				
	30				
25	10				
	30				
50	10				
	30				

3.2.2 Uji Coba Kombinasi Parameter Reproduksi

Uji coba kombinasi parameter reproduksi adalah uji coba yang dilakukan untuk mengetahui kombinasi nilai *crossover rate* (cr) dan *mutation rate* (mr) terbaik untuk menghasilkan kombinasi jumlah *offspring* terpilih guna mendapatkan hasil yang optimal. Rancangan uji coba kombinasi parameter reproduksi dapat dilihat pada Tabel 3.2:

Tabel 3.2 Rancangan Uji Coba Kombinasi Parameter Reproduksi

Nilai mr	Nilai cr	Nilai *Fitness* Percobaan Ke-i			Rata Rata *Fitness*
		1	2	3	
0,1	0,3				
	0,8				
0,2	0,3				
	0,8				
0,3	0,3				
	0,8				

3.3 Implementasi

3.3.1 Implementasi *Preprocessing* Teks

Implementasi proses *preprocessing* teks akan dilakukan dengan mengikuti prosedur alur tahapan pengerjaan *preprocessing* teks dapat dilihat pada Tabel 3.3.

Tabel 3.3 Prosedur alur tahapan pengerjaan *preprocessing* teks

Tahap 1: Lakukan *parsing* data teks dengan memisahkan setiap kalimat yang ada menjadi dokumen
Tahap 2: Lakukan *lexing/tokenization* dengan memisahkan setiap string kata dalam kalimat, menghapus duplikasi kata, angka, tanda baca, karakter-karakter lain selain huruf alfabet & simbol ilmiah, merubah setiap huruf kapital yang ada menjadi huruf kecil/dasar
Tahap 3: Lakukan *filtering/stopword removal* dengan menghapus kata-kata tidak penting dari teks hasil *lexing*
Tahap 4: Membuat *term* indeks dengan menggunakan setiap kata yang tersisa hasil *filtering*

3.3.2 Implementasi TF-IDF

Implementasi proses TF-IDF akan dilakukan dengan mengikuti prosedur alur tahapan pengerjaan TF-IDF dapat dilihat pada Tabel 3.4.

Tabel 3.4 Prosedur alur tahapan pengerjaan metode TF-IDF

Tahap 1: Hitung nilai *term frequency*/kemunculan setiap *term* indeks pada setiap kalimat
Tahap 2: Hitung nilai *log-term frequency* setiap *term* indeks pada setiap kalimat
Tahap 3: Hitung nilai *document frequency*/banyak kalimat yang mengandung *term* indeks
Tahap 4: Hitung nilai *inverse document frequency* dari hasil *document frequency*

Tahap 5: Hitung nilai TF-IDF dari hasil *log-term frequency* dan hasil *inverse document frequency*
Tahap 6: Hitung nilai skor setiap kalimat dengan menjumlahkan nilai TF-IDF dari setiap kalimat

3.3.3 Implementasi EGA

Implementasi proses EGA akan dilakukan dengan mengikuti prosedur alur tahapan pengerjaan EGA dapat dilihat pada Tabel 3.5.

Tabel 3.5 Prosedur alur tahapan pengerjaan metode EGA

Tahap 1: Inisialisasi populasi sebanyak *popSize*
Tahap 2: Evaluasi nilai *fitness* dari setiap *chromosome* dalam populasi
Tahap 3: Lakukan proses reproduksi dengan cara sebagai berikut:

a) Untuk operator *crossover*:
- Hitung jumlah *offspring* yang akan dihasilkan dari *crossover rate*
- Pilih dua *chromosome* induk dari populasi dengan *subset* yang berbeda
- Tukar nilai *subset* kedua *string chromosome* induk mulai dari indeks yang terpilih
- Lakukan hingga didapatkan jumlah *offspring* yang memenuhi jumlah hasil *offspring* dari *crossover rate*

b) Untuk operator mutasi:
- Hitung jumlah *offspring* yang akan dihasilkan dari *mutation rate*
- Pilih satu *chromosome* induk dari populasi
- Ganti nilai *subset* terendah dalam *chromosome* induk dengan nilai *subset* tertinggi dari *chromosome* terbaik dari populasi sebelumnya
- Jika *subset* yang didapati sama atau keseluruhan *subset*-nya sama maka ganti keseluruhan *subset* dengan *subset* milik *chromosome* terbaik populasi sebelumnya
- Lakukan hingga didapatkan jumlah *offspring* yang memenuhi jumlah hasil *offspring* dari *mutation rate*

Tahap 4: Masukkan *chromosome* hasil reproduksi kedalam populasi *chomosome* induk
Tahap 5: Update populasi dengan *roulette wheel selection* (RWS) hingga didapatkan populasi baru sebanyak *popSize*
Tahap 6: Ambil chromosome dengan nilai *fitness* tertinggi sebagai hasil akhir peringkasan teks

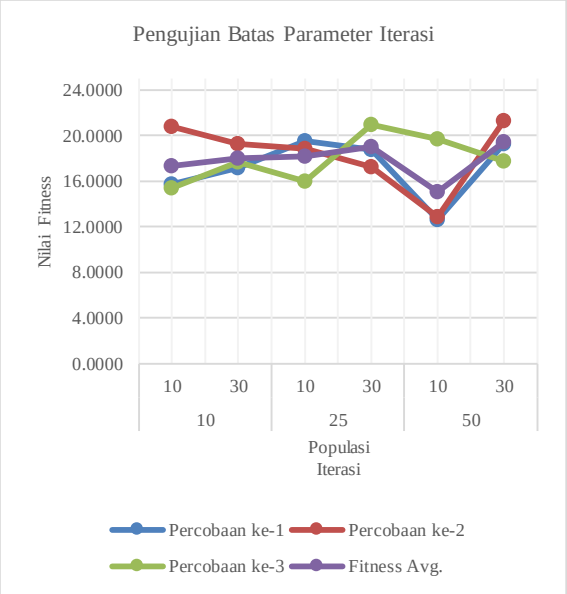
4. PENGUJIAN DAN ANALISIS

4.1 Pengujian Batas Parameter Iterasi

Pengujian ini dilakukan dengan cara menjalankan program dalam beberapa kondisi parameter iterasi dan populasi seperti yang dapat dilihat pada Tabel 3.1, lalu dalam setiap kondisi tersebut dilakukan percobaan penjalanan program sebanyak 3 kali percobaan, dimana dari ketiga percobaan setiap kondisi tersebut akan dihitung nilai rata-rata fitnessnya. Hasil pengujian batas parameter populasi dapat dilihat pada Tabel 3.6 dan sebagai ilustrasi gambaran hasil pengujian batas parameter iterasi dapat dilihat pada Gambar 3.2.

Tabel 3.6 Hasil Pengujian Batas Parameter Iterasi

Nilai Iter.	Nilai Pop.	Nilai *Fitness* Percobaan Ke-i			Rata-Rata Fitness
		1	2	3	
10	10	15.7409	20.8163	15.3994	17.3189
	30	17.1392	19.2701	17.6866	18.0320
25	10	19.5576	18.8977	15.9740	18.1431
	30	18.7898	17.2668	20.9458	19.0008
50	10	12.6497	12.8629	19.6743	15.0623
	30	**19.2701**	**21.2810**	**17.7628**	**19.4380**



Gambar 3.2 Grafik Hasil Pengujian Batas Parameter Iterasi

Dari grafik diatas dapat dilihat bahwa dalam setiap percobaan nilai *fitness* cenderung naik turun dengan alur yang berbeda-beda, hal ini dikarenakan dalam perhitungan fungsi *fitness* digunakan semuah nilai acak yang berbeda-beda untuk setiap subsetnya namun berjumlah 1. Dan dapat dilihat pula bahwa nilai rata-rata *fitness* tertinggi didapatkan dalam kondisi parameter jumlah iterasi sebesar 50 dan jumlah populasi sebesar 30, sedangkan nilai rata-rata *fitness* terendah didapat dalam kondisi parameter jumlah iterasi sebesar 50 dan jumlah populasi sebesar 10.

Jika jumlah iterasi besar dan jumlah populasi kecil maka *fitness* akan menghasilkan nilai yang

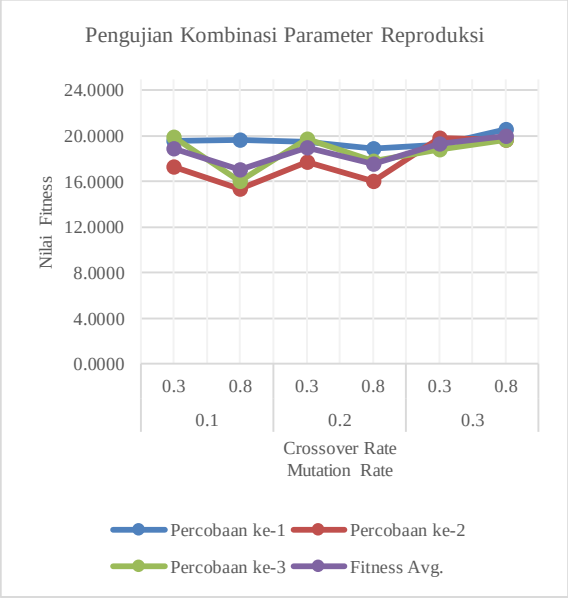
kecil. Berdasarkan analisis penulis, nilai *fitness* akan semakin besar jika interval nilai antara jumlah iterasi dan jumlah populasi tidak terlalu jauh. Untuk meningkatkan nilai *fitness*, jumlah iterasi dan jumlah populasi harus diperbesar.

4.2 Pengujian Kombinasi Parameter Reproduksi

Pengujian ini dilakukan dengan cara menjalankan program dalam beberapa kondisi parameter mr dan cr seperti yang dapat dilihat pada Tabel 3.2, lalu dalam setiap kondisi tersebut dilakukan percobaan penjalanan program sebanyak 3 kali percobaan, dimana dari ketiga percobaan setiap kondisi tersebut akan dihitung nilai rata-rata fitnessnya. Dan untuk parameter iterasi serta populasi dalam pengujian ini menggunakan jumlah besar iterasi dan populasi terbaik dari pengujian batas parameter iterasi. Hasil pengujian batas parameter populasi dapat dilihat pada Tabel 3.7 dan sebagai ilustrasi gambaran hasil pengujian batas parameter iterasi dapat dilihat pada Gambar 3.3.

Tabel 3.7 Hasil Pengujian Kombinasi Parameter Reproduksi

Nilai mr	Nilai cr	Nilai *Fitness* Percobaan Ke-i			Rata-Rata Fitness
		1	2	3	
0.1	0.3	19.5400	17.3422	19.8957	18.9260
	0.8	19.6743	15.3929	16.0351	17.0341
0.2	0.3	19.5184	17.7185	19.7170	18.9846
	0.8	18.8977	16.0350	17.7781	17.5703
0.3	0.3	19.2701	19.8021	18.8233	19.2985
	0.8	**20.6107**	**19.6763**	**19.6989**	**19.9953**



Gambar 3.3 Grafik Hasil Pengujian Kombinasi Parameter Reproduksi

Dari grafik diatas dapat dilihat bahwa dalam setiap percobaan nilai *fitness* cenderung naik turun dengan alur yang sama, hal ini dikarenakan nilai parameter populasi yang digunakan adalah hasil terbaik dari pengujian sebelumnya, dimana parameter tersebut akan mempengaruhi jumlah besar *offspring* masing-masing operator reproduksi. Dan dapat dilihat pula bahwa nilai rata-rata *fitness* tertinggi didapatkan dalam kondisi parameter jumlah mr sebesar 0.3 dan jumlah cr sebesar 0.8, sedangkan nilai rata-rata *fitness* terendah didapat dalam kondisi parameter jumlah mr sebesar 0.1 dan jumlah cr sebesar 0.8.

Nilai *fitness* pada cr sebesar 0.3 selalu lebih besar daripada nilai *fitness* pada cr 0.8 kecuali pada saat nilai mr 0.3. Penggunaan nilai cr 0.8 menghasilkan *fitness* yang naik naik setiap kali nilai mr ditingkatkan. Sehingga kita bisa meningkatkan nilai cr dan mr jika menginginkan nilai *fitness* yang besar. Nilai *fitness* yang maksimal adalah pada saat cr 0.3 dan mr 0.1

5. KESIMPULAN DAN SARAN

Algoritma TF-IDF-EGA mampu untuk menyelesaikan permasalahan pemilihan kalimat ringkasan pada TF-IDF yang sebelumnya hanya mengambil kalimat dengan skor tertinggi (dimana terkadang kalimat dengan skor tertinggi bukan merupakan kalimat deskriptif) menjadi pengambilan kalimat ringkasan dengan nilai *fitness* tertinggi namun dengan ringkasan kalimatnya dapat berupa kalimat dengan skor rendah (yang biasanya merupakan kalimat deskriptif).

Dari pengujian yang dilakukan, didapatkan bahwa parameter yang dapat menghasilkan nilai *fitness* terbaik adalah dengan jumlah iterasi sebesar 50, jumlah populasi sebesar 30, nilai *crossover rate* sebesar 0.3, dan nilai *mutation rate* sebesar 0.8. Dimana semakin tinggi nilai parameter iterasi dan populasi dengan parameter yang tidak terlalu jauh akan semakin meningkatkan nilai rata-rata hasil *fitness*, begitu pula untuk parameter reproduksi.

Penelitian ini dapat dikembangkan dengan penggunaan *random injection* yang dapat mengatasi problem solusi *subset chromosome* yang terkadang cenderung mengalami konvergensi dini. Kemudian penelitian ini dapat dikembangkan pula dengan cara me-*repair* operasi *crossover* untuk menghindari terpilihnya kalimat yang sama dalam hasil peringkasannya. Lalu penelitian ini juga dapat dikembangkan dengan cara mengubah alur proses peringkasannya menjadi bersifat abstraktif, dimana setiap kalimat yg panjang dapat dipisah menjadi beberapa kalimat tertentu.

6. DAFTAR PUSTAKA

MEENA, K.Y. & GOPALANI, D., 2015. Evolutionary Algorithms for Extractive Automatic Text Summarization. *Intelligent Computing, Communication & Convergence*, (48), pp.244-49.

KIANI, A. & AKBARZADEH, M.R., 2006. Automatic Text Summarization Using:

Hybrid Fuzzy GA-GP. *Fuzzy Systems*, pp.977-83.

GHAREB, A.S., BAKAR, A.A. & HAMDAN, R.A., 2015. Hybrid feature selection based on enhanced genetic algorithm for text categorization. *Expert Systems With Applications*, (49), pp.31-47.

FHADLI, M., PRABOWO, D.A. & PRASETYA, A., 2016. *Pencarian Senyawa Dalam Tumbuhan Herbal dan Search Engine Senyawa Tumbuhan Dengan Summarization Menggunakan Metode TF-IDF*. Laporan KKN. Malang: Universitas Brawijaya Program Studi Teknik Informatika.

ABUOBIEDA, A., SALIM, N., BINWAHLAN, M.S. & OSMAN, A.H., 2013. Differential Evolution Cluster-based Text Summarization Methods. *International Conference On Computing, Electrical and Electronic Engineering (ICCEEE)*, pp.244-48.

BABAR, S.A. & PATIL, P.D., 2015. Improving Performance of Text Summarization. *International Conference on Information and Communication Technologies (ICICT 2014)*, (46), pp.354–63.

SARANYAMOL, C.S. & SINDHU, L., 2014. A Survey on Automatic Text Summarization. *International Journal of Computer Science and Information Technologies*, 5(6), pp.7889-93.