OPEN ACCESS
Indo-JC Indonesia Journal of Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 5, Issue. 2, September 2020. pp. 31-42
doi:10.21108/indojc.2020.5.2.440

Peringkasan Teks Ekstraktif Menggunakan *Binary Firefly Algorithm*

Ade Naufal Ammar [1], Suyanto [2]

School of Computing, Telkom University
Jl. Telekomunikasi Terusan Buah Batu, Bandung, West Java 40257, Indonesia

[1] adenaufal@student.telkomuniversity.ac.id, [2] suyanto@telkomuniversity.ac.id

Abstract

There is a lot of text information on the internet, but humans find it difficult to understand all of this information in a short time. Automatic text summarization is a technology that helps people to read a summarization of the text that produced by an application without human intervention. First, the data from a website is taken by using parsing. Pattern matching is also needed to filter all of HTML tags from the taken data to produce a pure text. Then, application use tokenization to break the text into a group of words. Using binary firefly algorithm, each section of the text was given a weight based on similarity score of the contained words which is determined by TF-IDF matrix. This research takes seven sections of text that have the highest weight similarity scores. Then the summary was evaluated by using ROUGE metric. The results showed that compared to abstractive, extractive summarization method gave a relative improvement of 47.06% on ROUGE-1, 34.4% on ROUGE-2, and 44.92% on ROUGE-L.

Keywords: text summarization, binary firefly algorithm, TF-IDF matrix, ROUGE evaluation

Abstrak

Ada banyak informasi teks yang beredar di internet, tetapi manusia sulit mencerna semua informasi tersebut dalam waktu singkat. Peringkasan teks otomatis merupakan teknologi yang membantu seseorang untuk membaca suatu teks secara ringkas dengan menghasilkan ringkasan secara otomatis dari suatu teks tanpa adanya proses penyuntingan manusia terhadap ringkasan tersebut. Pertama, data dari situs diambil menggunakan teknik *parsing*. *Pattern matching* juga diperlukan untuk menyaring tag HTML dari data yang diambil sehingga menghasilkan teks murni. Setelah itu, dilanjutkan dengan *tokenization* untuk memecah teks menjadi kumpulan kata bermakna. Dengan *Binary Firefly Algorithm*, setiap bagian pada teks diberikan bobot berdasarkan skor kemiripan makna yang terkandung yang ditentukan oleh matriks TF-IDF. Dalam penelitian ini, ringkasan teks dibuat dengan mengambil tujuh bagian teks yang memiliki bobot tertinggi. Ringkasan kemudian dievaluasi menggunakan metrik ROUGE. Hasil penelitian menunjukkan bahwa dibandingkan dengan ringkasan abstraktif, ringkasan ekstraktif memberikan *relative improvement* sebesar 47,06% pada ROUGE-1, 34,4% pada ROUGE-2, dan 44,92% pada ROUGE-L.

Kata Kunci: peringkasan teks, algoritme *binary firefly*, matriks TF-IDF, evaluasi ROUGE

Received on Jan 2020. Accepted on Aug 2020

I. Introduction

SALAH satu kegiatan terpenting manusia untuk memperkaya cakrawala pengetahuannya adalah membaca. Di zaman informasi ini, kemajuan teknologi yang pesat membuat manusia menjadi lebih mudah mencari informasi yang beredar. Akan tetapi, jumlah informasi yang beredar tidak selaras dengan kemampuan rata-rata membaca manusia. Hadirnya teknologi peringkasan teks otomatis membantu manusia dalam menyediakan ringkasan teks. Peringkasan teks umumnya terbagi menjadi dua metode, yaitu abstraktif dan ekstraktif. Peringkasan teks secara ekstraktif lebih sangkil dibandingkan dengan abstraktif karena menghasilkan ringkasan teks dengan memilih kalimat-kalimat yang relevan dengan dokumen aslinya lalu memilih kata-kata yang memiliki nilai skor relevansi paling tinggi. Hasil peringkasan tidak terlalu memperhatikan tata bahasa karena hanya menilai kalimat mana saja yang mewakili intisari dari satu dokumen teks secara keseluruhan [1] [2].

Firefly Algorithm (FA) adalah salah satu metode *swarm intelligence* berbasis metaheuristik untuk optimisasi global. Pemilihan metode FA untuk peringkasan ekstraktif karena menggunakan prinsip ketertarikan (*attractiveness*) yang dapat menjaring kalimat mana saja yang diprediksi lebih berpotensi memberikan intisari dari satu artikel secara keseluruhan. Sejauh ini, baik *swarm intelligence* secara umum maupun FA, telah digunakan dalam beberapa penelitian berbasis teks [13] [14] [15] [16]. Salah satu varian dari FA adalah *Binary Firefly Algorithm* (BFA). Varian FA ini menggunakan fungsi sigmoid dan fungsi transformasi Tanh dalam perhitungannya. Penelitian ini membahas eksperimen percobaan peringkasan teks secara ekstraktif menggunakan BFA pada teks berbahasa Indonesia. Penelitian peringkasan teks ekstraktif pernah dilakukan oleh Abdallah dan Al-Taani dalam jurnalnya yang berjudul "Arabic Text Summarization using Firefly Algorithm" [3]. Perbedaan penelitian tersebut dengan penelitian penulis ialah penulis melakukan modifikasi metode dari *firefly algorithm* menjadi *binary firefly algorithm* guna meningkatkan presisi ringkasan.

II. Literature Review

Sebelum ditemukan *modern neural network*, peringkasan teks abstraktif tidak mendapatkan banyak perhatian dibandingkan dengan peringkasan teks ekstraktif. Akan tetapi, Hongyan Jing [13] berhasil mengekplorasi pemotongan bagian kalimat yang tidak penting untuk membuat ringkasan dan Cheung & Penn [14] berhasil mengeksplorasi perpaduan kalimat menggunakan pohon dependensi.

A. Firefly Algorithm

Algoritme FA terdiri dari beberapa langkah berikut [3]:
1. Inisialisasi populasi kunang-kunang secara acak.
2. Temukan CBS (*current best solution*).
3. Hitung pergerakan kunang-kunang: jika lebih sedikit bercahaya, maka kunang-kunang akan bergerak ke yang lebih terang.
4. Cek kriteria berhenti: jika kriteria terpenuhi, maka urutkan ranking kunang-kunang. Setelah mengurutkan ranking, berikan nilai hasil optimal. Jika tidak, ulangi langkah kedua.

Algoritme FA dimulai dengan serangkaian solusi kandidat acak (berupa ringkasan). Kalimat dalam ringkasan kandidat diurutkan dalam urutan naik untuk mempertahankan urutan kronologis dari ringkasan. Setiap kunang-kunang diinisialisasi dengan posisi-posisi dan ringkasan acak dari kandidat graf solusi. Kemudian proses evaluasi kualitas ringkasan kandidat dilakukan dengan menggunakan fungsi *fitness*. Fungsi *fitness* dihitung dengan mengalikan nilai kemiripan (lihat rumus 20) dengan skor informasi (lihat rumus 8) untuk kalimat yang menyusun ringkasan menggunakan persamaan [3]:

$$F_\alpha = \sum_{i=1}^{n} \sum_{j=i+1}^{n-1} SimScore(S_i, S_j) \times IScore(S_i) \quad (1)$$

dengan F_α adalah nilai *fitness*, *n* adalah jumlah kalimat, *SimScore* adalah nilai kemiripan, *i* adalah kalimat i, *j*

adalah kalimat j, *Si* adalah jumlah kalimat i, *Sj* adalah jumlah kalimat j, dan *IScore* adalah skor informasi.

Algoritme FA bertujuan untuk menemukan *sub-path* dengan jumlah node yang terbatas dari DAG yang memiliki nilai *fitness* maksimum. Tujuan menggunakan algoritme FA adalah untuk menghindari masalah dengan teknik pencarian lokal dan untuk memilih kalimat informatif maksimum bersamaan dengan aliran maksimum yang lebih baik untuk keterbacaan optimal dan minimum redundansi.

B. Binary Firefly Algorithm

Binary Firefly Algorithm merupakan salah satu bentuk varian *Firefly Algorithm*. *Firefly Algorithm* pertama kali dikembangkan oleh Xin-She Yang pada akhir 2007 dan 2008 di Cambridge University [4]. Dinamakan *Firefly Algorithm* karena ia menggunakan pola kelap-kelip dan tingkah laku dari makhluk kunang-kunang (*firefly*). Pada saat itu, FA dikembangkan untuk masalah optimisasi berkelanjutan. Berfokus pada penanganan masalah FS, Zhang et al. mengusulkan FA berbasis biner dengan mengubah dua istilah terakhir dalam persamaan ke dalam vektor probabilitas berikut dengan fungsi sigmoid [5]:

$$x_{(k,j)} = \begin{cases} 1, \dfrac{1}{1+e^{v_{(k,j)}}} > rand \\ 0, \dfrac{1}{1+e^{v_{(k,j)}}} < rand \end{cases} \tag{2}$$

$$x_{(k,j)} = \beta(k,l)\left(x_{(l,j)} - v_{(k,j)}\right) + \alpha(rand - 0.5) \tag{3}$$

dengan rand adalah bilangan acak antara 0 hingga 1 dan e adalah bilangan euler (2.71828).

Dari rumus di atas, dibuatlah fungsi transformasi berikut yang disebut Tanh guna meningkatkan kinerja FFA biner [6]:

$$\tanh\left(\left|v_{(k,j)}\right|\right) = \frac{e^2\left|v_{(k,j)}\right| - 1}{e^2\left|v_{(k,j)}\right| + 1} \tag{4}$$

C. Term Frequency–Inverse Document Frequency (TF-IDF)

TF-IDF bertujuan untuk menentukan seberapa penting suatu kata untuk mewakili dokumen dalam suatu *corpus*. Bobot TF-IDF dalam sebuah kalimat dapat didefinisikan oleh rumus berikut [3]:

$$TF - IDF\left(t_{s_i}\right) = TF(t_{s_i}) \times IDF(t) \tag{5}$$

dengan t adalah jumlah dari berapa kali istilah t muncul pada kalimat i suatu dokumen.

IDF dari istilah t dapat didefinisikan sebagai [3]:

$$IDF(t) = \log N/Nt \tag{6}$$

TF-IDF akhir adalah penjumlahan semua TF-IDF dari setiap kalimat secara keseluruhan [3]:

$$TF - IDF(S_i) = \sum_{t=1}^{n} TF - IDF(t_{s_i}) \tag{7}$$

Tiap kalimat diberikan skor informasi yang merupakan jumlah dari semua skor fitur sebelumnya dari kalimat tersebut. Persamaan yang digunakan untuk menghitung skor informasi adalah sebagai berikut [3]:

$$ISCORE(S) = T + L + Loc + TF - IDF(S) \tag{8}$$

dengan $ISCORE(S)$ adalah skor informasi dari kalimat i. T adalah fitur kemiripan judul (lihat rumus 14), L adalah fitur panjang kalimat (lihat rumus 15), Loc adalah fitur lokasi kalimat (lihat rumus 16) dan $TF - IDF(S)$ adalah TF-IDF dari kalimat i (lihat rumus 19).

D. ROUGE Metric

ROUGE adalah metode evaluasi yang dapat mengukur kualitas dari hasil sistem peringkasan teks otomatis dengan membandingkannya dengan ringkasan buatan manusia [7]. Jenis-jenis metode pada ROUGE adalah ROUGE-N, ROUGE-L, ROUGE-W, ROUGE-S dan ROUGE-SU.

ROUGE-N: ***N-gram Co-Occurrence Statistics***
ROUGE-N berupa perhitungan n-gram antara kandidat dengan referensi ringkasan [7].

$$\frac{\sum_{S\epsilon\{Reference\ Summaries\}}\sum_{gram_n\epsilon S} Count_{match}(gram_n)}{\sum_{S\epsilon\{Reference\ Summaries\}}\sum_{gram_n\epsilon S} Count(gram_n)} \tag{9}$$

ROUGE-L: ***Longest Common Subsequence***
Sebuah urutan $Z = [z_1, z_2, \dots, z_n]$ adalah sub urutan dari urutan $X = [x_1, x_2, \dots, x_n]$ jika ada kenaikan urutan $[i_1, i_2, \dots, i_n]$ dari setiap index X untuk semua $j = 1, 2, \dots, k$ sehingga menghasilkan $x_{ij} = z_j$. Pada dua urutan X dan Y, urutan umum terpanjang (LCS) dari X dan Y adalah urutan umum dengan panjang maksimum. LCS telah banyak dipakai untuk mengidentifikasi persamaan kata dari gabungan teks dengan menggunakan rasio LCSR antara panjang LCS dari dua kata dengan kata terpanjang dari dua kata untuk mengukur keserumpunan mereka [7].

Sentence-Level LCS
Untuk menerapkan LCS dalam evaluasi ringkasan perlu memperhatikan kalimat ringkasan sebagai urutan kata-kata. Intinya adalah bahwa semakin banyak LCS dari dua kalimat ringkasan, semakin mirip kedua ringkasan tersebut. Lin (2005) mengusulkan menggunakan pengukuran F (*F-measure*) berbasis LCS untuk memperkirakan kesamaan antara dua ringkasan X dengan panjang m dan Y dengan panjang n, dengan asumsi X adalah kalimat ringkasan referensi dan Y adalah ringkasan sistem sebagai berikut [7]:

$$R_{lcs} = \frac{LCS(X,Y)}{m} \tag{10}$$

$$P_{lcs} = \frac{LCS(X,Y)}{n} \tag{11}$$

$$F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \tag{12}$$

E. Relative Change

Relative change digunakan untuk menghitung perbedaan antara dua nilai yang mengalami perubahan. *Relative change* dinyatakan dalam persen. Rumus umum dari *relative change* yaitu [8]:

$$Relative\ change(x, x_{reference}) = \frac{(x\ -\ x_{reference})}{x_{reference}} \tag{13}$$

III. RESEARCH METHOD

A. Ringkasan

Ringkasan, menurut Mulyati, merupakan salah satu wujud/bentuk penyingkatan suatu informasi dengan hanya menyajikan informasi atau butir-butir pentingnya [9]. Ringkasan teks membantu pembaca dalam memahami konteks isi dan mendapatkan informasi penting dari suatu teks secara keseluruhan dalam waktu singkat. Pembaca tentu akan lebih menyukai membaca ringkasan tulisan daripada membaca tulisan secara keseluruhan. Namun, tidak semua pemilik informasi teks menyediakan ringkasan.

B. *Dataset*

Dataset yang digunakan adalah dataset IndoSum [10] yang berisikan kurang lebih 19 ribu artikel berita dan rangkuman berbahasa Indonesia. Dataset telah dipra-proses seperti tokenisasi, mengecilkan huruf, menghilangkan tanda huruf, dan mengganti angka dengan huruf nol.

C. Pra-proses

Tahap pra-proses mengikuti algoritme *natural language processing* yang meliputi segmentasi (proses pemecahan teks menjadi kalimat-kalimat), tokenisasi (proses pemecahan kalimat menjadi kata-kata), penghapusan *stop-words* atau kata-kata yang tidak penting, *stemming* (pengelompokan kata-kata menjadi kata dasar), dan terakhir membuat dokumen yang berisi daftar kata yang telah di-*stemming*.

D. Hitung Skor Informasi

Skor informasi untuk setiap kalimat dalam dokumen dihitung menggunakan fitur struktural. Fitur-fitur berikut digunakan untuk menentukan skor dari setiap kalimat [3].

Kemiripan kalimat dengan judul [3]

$$T = \frac{Jumlah\ kata\ di\ kalimat\ S\ yang\ ada\ di\ judul}{Jumlah\ kata\ di\ judul} \tag{14}$$

Nilai kemiripan kalimat dengan judul (T) dihitung dengan cara membagi jumlah kata di kalimat ke-S yang ada di judul dengan jumlah kata di judul. Hasil dari nilai T berupa angka.

Panjang kalimat (L) [3]
Untuk menghitung skor panjang kalimat, dapat menggunakan rumus berikut:

$$L = \frac{Jumlah\ kata\ di\ kalimat\ S}{Jumlah\ kata\ di\ kalimat\ terpanjang} \tag{15}$$

Lokasi kalimat (Loc) [3]
Skor lokasi kalimat relatif pada total banyaknya kalimat pernyataan dapat dihitung dengan menggunakan rumus berikut [3]:

$$Loc = 1 - \frac{Loc\ S}{N} \tag{16}$$

dengan N adalah total banyaknya kalimat dan *Loc S* adalah lokasi dari kalimat ke-S dari dokumen tersebut. Hasil dari nilai *Loc* berupa angka yang tidak memiliki rentang tertentu.

E. Hitung Skor Semantik

Skor semantik merefleksikan seberapa dekat dan mirip dari dua kalimat. *Cosine similarity* digunakan untuk menghitung korelasi kesamaan antara kalimat yang dimaksud. Vektor TF dibuat menggunakan persamaan [3]:

$$\overrightarrow{TF_{si}} = (TF(t_{1s_i}), TF(t_{2s_i}), TF(t_{3s_i}), \dots, TF(t_{ns_i})) \quad (17)$$

dengan $TF(t_{s_i})$ adalah frekunsi kemunculan istilah t pada kalimat i.

Vektor IDF dibuat menggunakan persamaan [3]:

$$\overrightarrow{IDF_{si}} = (IDF(t_{1s_i}), IDF(t_{2s_i}), IDF(t_{3s_i}), \dots, IDF(t_{ns_i})) \quad (18)$$

Vektor TF-IDF dibuat menggunakan persamaan [3]:

$$\overrightarrow{TF-IDF_{si}} = (TF-IDF(t_{1s_i}), TF-IDF(t_{2s_i}), TF-IDF(t_{3s_i}), \dots, TF-IDF(t_{ns_i})) \quad (19)$$

Langkah selanjutnya adalah membangun matriks kemiripan untuk dokumen tersebut. Matriks ini dibutuhkan untuk mengetahui kalimat mana yang akan berguna untuk dipilih berdasarkan semantik. Kemiripan dari kedua kalimat dihitung menggunakan persamaan [3]:

$$SimScore(s_i, s_j) = \frac{\sum_{t=1}^{n} TF-IDF(s_{ti}) \times TF-IDF(s_{tj})}{\sqrt{\sum_{t=1}^{n}(TF-IDF(s_{ti}))^2} \times \sqrt{\sum_{t=1}^{n}(TF-IDF(s_{tj}))^2}} \quad (20)$$

dengan $SimScore(:E)$ adalah $cosine\ similarity$ antara dua kalimat vektor : E dan vektor : F.

F. Bangun Graf Solusi

Digunakan graf bobot DAF dengan setiap simpul merepresentasikan kalimat-kalimat dalam dokumen dan tepi yang merepresentasikan kemiripan antara kedua kalimat. Graf solusi diciptakan melalui dua tahap [3]:
1. Menambahkan setiap kalimat ke dalam dokumen pada daftar simpul.
2. Untuk setiap dua kalimat s dan s_G, jika kalimat s muncul sebelum s_G dan di dokumen dan skor semantiknya tidak nol, maka tambah tepi dari s ke s_G ke dalam daftar tepi.

G. Penerapan BFA

Penentuan kalimat yang hendak ditampilkan pada ringkasan menggunakan *binary firefly algorithm* berlangsung sebagai berikut [11] [12]:

1. Menentukan ukuran dokumen dengan menghitung keseluruhan jumlah kata sebagai acuan jumlah kunang-kunang.
2. Berdasarkan pada ukuran tersebut, tentukan jumlah kunang-kunang untuk membuat populasi.
3. Menginisialisasi posisi kunang-kunang secara acak.
4. Memberikan nilai pada kunang-kunang. Kunang-kunang berupa kata terseleksi yang direpresentasikan dalam bentuk *array* {0,1}. Nilai *array* menunjukkan bahwa suatu kata terdapat atau tidak pada dokumen sebagaimana dapat dilihat pada persamaan di bawah [11]:

$$x_{(i,j)} \begin{cases} 1\ jika\ kata\ j\ terdapat\ pada\ dokumen\ i \\ 0\ jika\ kata\ j\ tak\ terdapat\ di\ dokumen\ i \end{cases} \quad (21)$$

Proses pertimbangan posisi 0 atau 1 mengikuti fungsi berikut [11]:

$$S(x_{(i,j)}) = \frac{1}{(1 + \exp(x_{(i,j)}))} \tag{22}$$

$$Jika\ (rand < S(x_{(i,j)}))\ maka\ x_{(i,j)} = 1, selain\ itu\ x_{(i,j)} = 0 \tag{23}$$

dengan $x_{(i,j)}$ adalah posisi saat ini pada kunang-kunang i di dimensi j (kata ke j pada dokumen). $S(x_{(i,j)})$ merujuk pada kemungkinan kunang-kunang berada di posisi $x_{(i,j)}$ mendapatkan nilai 1. Dalam kata lain, kemungkin kata *j* untuk diambil. Rand adalah angka acak antara 0 dan 1. Pengacakan digunakan untuk meningkatkan eksploitasi dalam seleksi kata yang relevan.

5. Menghitung intensitas posisi kunang-kunang saat ini menggunakan fungsi sebagai berikut [11]:

$$Jika\ TF(x_{(i,j)}) > 2\ maka\ skor = \sum TF(x_{(i,j)}) \tag{24}$$

6. Iterasi dilakukan dan proses pembandingan dimulai. Jika intensitas kunang-kunang i kurang dari intensitas kunang-kunang k, kunang-kunang i berpindah menuju kunang-kunang k. Fungsi ketertarikan $\beta(r)$ berikut digunakan untuk memindahkan kunang-kunang i ke kunang-kunang k [12]:

$$\beta(r) = \beta_0 \times e^{-\gamma r^2} \tag{25}$$

dengan r menunjukkan jarak antara kunang-kunang dan β_0 merepresentasikan ketertarikan pada r = 0 [12]:

$$r_{(i,j)} = |x_i - x_j| \tag{26}$$

dengan x menunjukkan nilai sesungguhnya dari posisi kunang-kunang yang telah dihitung berdasarkan informasi persamaan rasio yang didapat. Jarak dihitung menggunakan mengurangkan tiap kunang-kunang i dari kunang-kunang j. Jarak dari metode ini didapat dari perbedaan dua *binary string* antara kedua kunang-kunang.

7. Tiap kunang-kunang pada populasi bergerak ke arah kunang-kunang lebih terang – dengan kata lain kunang-kunang tertarik pada kunang-kunang lain yang lebih terang. Tahapan ini ditentukan menggunakan persamaan berikut [12]:

$$x_i = x_j + \beta \times (x_j - x_i) + \alpha \times (Rand - \frac{1}{2}) \tag{27}$$

8. Melakukan pengecekan kriteria berhenti. Jika kriteria terpenuhi, maka urutkan ranking kunang-kunang. Setelah mengurutkan ranking, berikan nilai hasil optimal. Jika tidak, ulangi langkah kedua.

IV. RESULTS AND DISCUSSION

Penulis membuat peringkas otomatis ekstraktif dan abstraktif lalu mengujinya menggunakan metrik ROUGE. Hasil dari pengujian tersebut ditunjukkan pada Tabel 1.

TABEL I
HASIL PENGUJIAN RINGKASAN EKSTRAKTIF DAN ABSTRAKTIF

Metric	***Evaluation***	***Summarization Type***	
		Extractive	***Abstractive***
		Mean	***Mean***
ROUGE-1	*Precision*	0,41254	0,09600

	Recall	0,56223	0,40598
	F-measure	0,46717	0,15485
ROUGE-2	*Precision*	0,29444	0,06868
	Recall	0,42680	0,30353
	F-measure	0,34064	0,11178
ROUGE-L	*Precision*	0,40255	0,09284
	Recall	0,54830	0,39175
	F-measure	0,42592	0,14971

Metrik ROUGE menggunakan tiga macam evaluasi yaitu *precision*, *recall* dan *F-measure*. *Precision* menunjukkan perbandingan antara total kata penting pada ringkasan dengan total kata pada ringkasan otomatis. Nilai *precision* didapatkan dari rumus berikut [15]:

$$\frac{jumlah\ kata\ tumpang\ tindih}{jumlah\ kata\ di\ ringkasan\ otomatis} \tag{28}$$

Nilai *precision* ROUGE-1 untuk peringkas ekstraktif penulis sebesar 0,41, ROUGE-2 sebesar 0,29, dan ROUGE-L sebesar 0,4, sedangkan nilai *precision* ROUGE-1 peringkas abstraktif penulis sebesar 0,1, ROUGE-2 sebesar 0,15, dan ROUGE-L sebesar 0,09.

Recall menunjukkan perbandingan antara total kata penting pada ringkasan dengan total kata pada ringkasan manusia. Nilai *recall* didapatkan dari rumus berikut [15]:

$$\frac{jumlah\ kata\ tumpang\ tindih}{jumlah\ kata\ di\ ringkasan\ manusia} \tag{29}$$

Nilai *recall* ROUGE-1 untuk peringkas ekstraktif penulis sebesar 0,56, ROUGE-2 sebesar 0,29, dan ROUGE-L sebesar 0,4, sedangkan nilai *recall* ROUGE-1 peringkas abstraktif penulis sebesar 0,4, ROUGE-2 sebesar 0,3, dan ROUGE-L sebesar 0,4.

F-measure menunjukkan seberapa penting nilai *precision* terhadap *recall* atau sebaliknya. *F-measure* dihitung dengan rumus [15]:

$$\frac{2 \times Precision \times Recall}{(Precision \times Recall)} \tag{30}$$

Nilai *F-measure* ROUGE-1 untuk peringkas ekstraktif penulis sebesar 0,47, ROUGE-2 sebesar 0,34, dan ROUGE -L sebesar 0,42, sedangkan nilai *F-measure* ROUGE-1 untuk peringkas abstraktif penulis sebesar 0,15, ROUGE-2 sebesar 0,11, dan ROUGE-L sebesar 0,15.

Demi mendapatkan nilai ROUGE yang lebih objektif, penulis melakukan 1.400 kali pengujian dan membandingkan hasil ringkasan otomatis dengan hasil ringkasan manusia. Dari 1.400 kali pengujian yang telah dilakukan, penulis mencari rata-rata tiap nilai ROUGE pada evaluasi hasil ringkasan otomatis. Rata-rata dihitung dengan rumus:

$$n\ rata - rata = \frac{n1 + n2 + \cdots + ni}{Si} \tag{31}$$

dengan *n* adalah nilai ROUGE pengujian ke-i, dan Si adalah jumlah pengujian. Untuk mencari perbedaan antara nilai ROUGE dibandingkan dengan rata-ratanya, digunakanlah rumus standar deviasi sebagai berikut:

$$SR = \frac{1}{n}\sum_{i=1}^{n} |xi - \bar{x}| \quad (32)$$

dengan *SR* adalah simpangan rata-rata, *n* adalah banyak nilai, *xi* adalah nilai ke i dan $\bar{x}$ adalah nilai rata-rata.

TABEL II
RATA-RATA DAN STANDAR DEVIASI NILAI ROUGE RINGKASAN EKSTRAKTIF

ROUGE Metric		**Rata-rata**	**Standar Deviasi**
ROUGE-1	*Precision*	0,413	0,163
	Recall	0,562	0,225
	F-measure	0,467	0,177
ROUGE-2	*Precision*	0,294	0,181
	Recall	0,427	0,260
	F-measure	0,341	0,203
ROUGE-L	*Precision*	0,403	0,167
	Recall	0,548	0,230
	F-measure	0,426	0,176

Hasil rata-rata akhir nilai ROUGE-1, ROUGE-2, dan ROUGE-L pada ringkasan ekstraktif maupun abstraktif dapat dilihat pada tabel di bawah:

TABEL III
RATA-RATA NILAI ROUGE-1, ROUGE-2, DAN ROUGE-L PADA RINGKASAN OTOMATIS

ROUGE Metric	**Ringkasan Abstraktif**	**Ringkasan Ekstraktif**	***Relative Improvement***
ROUGE-1	0,219	0,481	47,06
ROUGE-2	0,161	0,354	34,40
ROUGE-L	0,211	0,459	44,92

Pada Tabel 4 dapat dilihat ada *relative improvement* yang cukup besar dari ringkasan ekstraktif terhadap ringkasan abstraktif. Penulis menghitung *relative improvement* menggunakan rumus *relative change* (lihat rumus 13) dan menemukan masing-masing perbandingan nilai ROUGE sebagai berikut:

ROUGE-1

$$\frac{(0{,}481 - 0{,}219)}{0{,}219} \times 100 = 47{,}06 \quad (33)$$

ROUGE-2

$$\frac{(0{,}354 - 0{,}161)}{0{,}161} \times 100 = 34{,}4 \quad (34)$$

ROUGE-L

$$\frac{(0{,}459 - 0{,}211)}{0{,}211} \times 100 = 44{,}92 \quad (35)$$

Dari metrik ROUGE menunjukkan bahwa dibandingkan dengan metode abstraktif, ringkasan otomatis menggunakan metode ekstraktif jauh lebih menyerupai ringkasan manusia. Penyebab utama dari masalah ini adalah ringkasan abstraktif memiliki sejumlah *unknown word* <UNK> dan pengulangan frasa. Hal ini dikarenakan peringkas abstraktif mengandalkan *dictionary word* untuk merangkai kata-kata dalam membentuk suatu ringkasan. Sedangkan peringkas ekstraktif menggunakan kata-kata dalam bacaan untuk menghasilkan ringkasan sehingga tidak akan menghasilkan *unknown word* pada ringkasan yang dibuat. Selain itu, peringkas abstraktif cukup banyak menghasilkan kata baru pada ringkasan dengan kata tersebut tidak ada pada bacaan. Hal ini menyebabkan *precision* ringkasan abstraktif menjadi lebih rendah karena kata-kata pada ringkasan manusia kebanyakan berasal dari bacaan.

V. Conclusion

Penelitian peringkasan teks ekstraktif menggunakan *Binary Firefly Algorithm* telah berhasil dilakukan. Ada banyak algoritme yang digunakan dalam penelitian ini, seperti algoritme NLP, *binary firefly*, TF-IDF, dan ROUGE. Data latih yang digunakan dalam penelitian diambil dari IndoSum yang berisi artikel berita dan rangkuman berbahasa Indonesia. Sedangkan data uji yang digunakan dalam penelitian ini berupa artikel berita dari internet. Pada penelitian penulis terlihat bahwa apabila ringkasan ekstraktif dibandingkan terhadap ringkasan abstraktif, ringkasan ekstraktif memberikan *relative improvement* sebesar 47,06% pada ROUGE-1, 34,4% pada ROUGE-2, dan 44,92% pada ROUGE-L. Hasil penelitian menunjukkan bahwa meskipun peringkas abstraktif lebih rumit dan dianggap mendekati pemikiran orang, hasil ringkasan menggunakan peringkas ekstraktif masih jauh lebih mirip dengan ringkasan manusia dibandingkan dengan ringkasan hasil dari peringkas abstraktif karena peringkas abstraktif mengandalkan *dictionary words* untuk merangkai kata-kata dalam ringkasan. Guna mendapatkan *dictionary words* yang berkualitas, peringkas abstraktif harus sering dilatih dengan data latih. Meskipun demikian, tetap ada kemungkinan ringkasan hasil dari peringkas abstraktif memiliki *unknown word* serta terjadi pengulangan frasa terutama ketika data uji yang digunakan berbeda topik dengan data latih yang selama ini dilatih pada peringkas abstraktif. Akhir kata, dapat disimpulkan bahwa ringkasan ekstraktif lebih baik dari abstraktif.

Acknowledgment

Kami mengucapkan terima kasih kepada orang tua, seluruh kolega di Universitas Telkom, khususnya di Fakultas Informatika, dan teman-teman penulis yaitu Damar Alam Permana, Akbar Anshori, dan Anisa Dewinta Putri, yang turut membantu dalam penyelesaian penulisan jurnal ilmiah ini.

References

[1] N. Moratanch and S.Chitrakala, "A survey on abstractive text summarization," in *International Conference on Circuit, Power and Computing*, March 2016, pp. 1-7.

[2] N Nazari and MA Mahdavi, "A survey on automatic text summarization.," *Journal of AI and Data Mining*, vol. 7, no. 1, pp. 121–135, 2019.

[3] Raed Z., Ahmad T. Al-Taani Al-Abdallah, "Arabic Text Summarization using Firefly Algorithm," 2019.

[4] Xin She Yang and Xingshi He, "Automatic Extractive Text Summarization Using K-Means Clustering," *International Journal of Swarm Intelligence*, vol. 1, no. 1, 2013.

[5] L. Zhang, L. Shan, and J. Wang, "Optimal Feature Selection Using Distance Based Discrete Firefly Algorithm With Mutual Information Criterion," *Neural Comput. Appl.*, 2016.

[6] K., S.P. Simon, N.P. Padhy Chandrasekaran, "Binary Real Coded Firefly Algorithm For Solving Unit Commitment Problem," *Inf. Sci.*, pp. 67-84, 2013.

[7] Chin-Yew Lin, "Rouge: A package for automatic evaluation of summaries.," 2005.

[8] Jeffrey Bennett and William Briggs, *Using and Understanding Mathematics: A Quantitative Reasoning Approach*, 3rd ed. Boston: Pearson, 2005.

[9] Encep Kusumah and Yeti Mulyati, *Menulis 2*.: Penerbit Universitas Terbuka, 2014.

[10] Kemal Kurniawan and Samuel Louvan, "Indosum: A New Benchmark Dataset For Indonesian Text Summarization," 2018.

[11] Souad Larabi, Nada Alalyani Marie-Sainte, "Firefly Algorithm based Feature Selection for Arabic Text Classification," *Journal of King Saud University*, 2018.

[12] Rana F., Ban N. Dhannoon Najeeb, "A Feature Selection Approach Using Binary Firefly Algorithm For Network Intrusion Detection System," *ARPN Journal Of Engineering and Applied Sciences*, vol. 13, no. 6, 2018.

[13] H. Asgari, B. Masoumi, and O. S. Sheijani., "Automatic text summarization based on multi-agent particle swarm optimization," in *Iranian*, Feb 2014, pp. 1–5.

[14] M. S. Binwahlan, N. Salim, and L. Suanmali., "Swarm based text summarization," in *International Association of Computer Science and Information Technology - Spring Conference*, April 2009, pp. 145-150.

[15] Mohammed Salem Binwahlan, Naomie Salim, and Ladda Suanmali, *Fuzzy Swarm Based Text Summarization 1*.

[16] J. Sheeba, D. I. Sowmya, and Pradeep Devaneyan S., "Keyword Extraction Using Swarm Intelligence Techniques," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 4, no. 4, pp. 6742–6749, 2016.