

# PENGGUNAAN ALGORITMA SEMUT DAN CONFIX STRIPPING STEMMER UNTUK KLASIFIKASI DOKUMEN BERITA BERBAHASA INDONESIA

I Putu Adhi Kerta Mahendra – Agus Zainal Arifin – Henning Titi Ciptaningtyas

Jurusan Teknik Informatika, Fakultas Teknologi Informasi  
Institut Teknologi Sepuluh Nopember (ITS) – Surabaya, 60111, Indonesia

email: adhi@cs.its.ac.id

## Abstrak

Algoritma semut adalah solusi universal dan fleksibel yang pada awalnya digunakan pada permasalahan optimasi *Traveling Salesman Problem*. Analogi pencarian rute terpendek oleh semut-semut dan pencarian dokumen-dokumen yang sejenis, telah menjadi stimulus terciptanya suatu metode klasifikasi dokumen teks berbasis algoritma semut, yang terbagi menjadi dua tahap yakni pencarian rute terpendek antar dokumen (*trial phase*) dan pembentukan *cluster-cluster* (*dividing phase*).

Pada paper ini, metode klasifikasi dokumen teks berbasis algoritma semut diimplementasikan dengan mengambil 253 dokumen berita berbahasa Indonesia sebagai data uji. Selain itu juga dikembangkan *enhanced confix stripping stemmer*, sebagai perbaikan dari algoritma *confix stripping stemmer* untuk *stemming* dokumen berita berbahasa Indonesia.

Hasil uji coba membuktikan bahwa algoritma semut dapat diaplikasikan untuk klasifikasi dokumen berita berbahasa Indonesia, dengan nilai *F-measure* terbaik 0.86. Uji coba juga membuktikan bahwa *enhanced confix stripping stemmer* berhasil mengatasi kesalahan-kesalahan *confix stripping stemmer* dan mampu mereduksi jumlah *term* hingga 32.66%, sedangkan *confix stripping stemmer* hanya mampu mereduksi 30.95% jumlah *term*.

**Kata Kunci :** *ants algorithm, confix stripping stemmer, text document clustering, F-measure*

## 1. Pendahuluan

Meningkatnya popularitas Internet dan *World Wide Web* pada tahun 1990-an, telah membawa era baru dalam penyampaian informasi, dimana Internet telah menjadi media publikasi yang sangat populer. Dengan melihat peluang yang sama, maka perusahaan-perusahaan media cetak juga berusaha memuaskan publik dengan menyediakan artikel-artikel beritanya secara *online*, yang fleksibel serta efektif dari segi waktu dan biaya jika dibandingkan dengan media cetak konvensional seperti surat kabar dan majalah.

Permasalahan yang muncul kemudian adalah meningkatnya kebutuhan para pembaca berita untuk mendapatkan berita-berita yang terkait dengan berita yang dibacanya saat ini, baik dari sisi topik ataupun kejadian dalam berita tersebut. Permasalahan yang timbul menjadi semakin rumit dengan adanya fakta bahwa jumlah simpanan data berita menjadi sangat besar dan tidak terorganisir. Oleh karena itu, diperlukan suatu strategi pengelompokan otomatis dokumen-dokumen berita tersebut. *Categorization* atau *automatic classification* merupakan salah satu solusi untuk permasalahan ini.

Algoritma semut (*Ants Algorithm*) adalah algoritma universal, yang pada awalnya digunakan dalam permasalahan optimasi pencarian rute terpendek, dengan mengambil studi kasus *Traveling Salesman Problem*. Algoritma semut

kemudian dikembangkan ke dalam bidang *unsupervised classification (clustering)* pada dokumen teks, dengan menganalogikan dokumen-dokumen yang ada sebagai *node-node* pada suatu *graph*. Ide pencarian rute terpendek atau *total-similarities* terbesar antar *node-node* dokumen tersebut, menciptakan suatu metode *clustering* baru yang disebut dengan *ant based documents clustering* [6][7][9].

Salah satu penyesuaian dalam klasifikasi teks untuk suatu domain bahasa adalah dengan menyediakan suatu metode *stemming* yang spesifik pada bahasa tersebut. Sayangnya, perhatian terhadap domain Bahasa Indonesia masih tergolong minim, walaupun peringkat jumlah penduduknya terbanyak keempat di dunia dengan potensi pengguna Internet-nya yang semakin meningkat. *Confix stripping stemmer*, adalah salah satu diantara beberapa metode *stemming* yang spesifik terhadap Bahasa Indonesia.

Dewasa ini, terdapat banyak metode *clustering* yang pada umumnya dapat digolongkan ke dalam *hierarchical* maupun *non-hierarchical clustering* [8][10]. *Ant based documents clustering* tergolong sebagai salah satu metode baru dalam bidang klasifikasi dokumen. Berdasarkan hal ini, maka melalui Tugas Akhir ini, algoritma semut (*ant based documents clustering*) dicoba untuk diimplementasikan melalui suatu perangkat lunak pengklasifikasi dokumen-dokumen teks, dimana

dokumen teks yang diambil adalah dokumen berita berbahasa Indonesia. Karena domain bahasa yang digunakan adalah Bahasa Indonesia, maka aplikasi yang dibangun juga berupaya untuk mengimplementasikan *confix stripping stemmer* yang spesifik untuk Bahasa Indonesia.

## 2. Text Processing

### 2.1 Representasi Dokumen Teks

Untuk mengimplementasikan metode-metode klasifikasi dokumen teks, diperlukan suatu transformasi yang dapat mengubah teks-teks *digital* menjadi suatu model yang lebih efisien dan dapat dimengerti sehingga proses analisa dapat dilakukan [1]. *Vector space model* adalah salah satu pendekatan yang paling banyak digunakan dalam merepresentasikan dokumen teks. Dalam model ini, setiap dokumen  $d_j$  ditransformasikan menjadi suatu vektor [3]:

$$d_j = (w_{1j}, w_{2j}, \dots, w_{ij}), \quad (1)$$

dimana  $w_{ij}$  adalah bobot *term* ke- $i$  pada dokumen  $j$  bersangkutan.

Bobot setiap *term* dapat direpresentasikan secara binari (*true* atau *false*), frekuensi, atau dengan frekuensi dan frekuensi invers-dokumennya (TF-IDF). Metode TF-IDF klasik telah menunjukkan performa yang lebih baik jika dibandingkan dengan metode binari dan frekuensi [10], yang dinyatakan sebagai berikut:

$$w_{ij} = tf_{ij} \cdot \log_2 \left( \frac{N}{df_i} \right), \quad (2)$$

dimana  $w_{ij}$  adalah bobot *term*  $i$  pada dokumen  $j$ ,  $tf_{ij}$  adalah frekuensi *term*  $i$  pada dokumen  $j$ ,  $N$  adalah jumlah dokumen yang diproses dan  $df_i$  adalah jumlah dokumen yang memiliki *term*  $i$  di dalamnya.

### 2.2 Confix Stripping Stemmer

*Confix stripping (CS) stemmer* adalah metode *stemming* pada Bahasa Indonesia yang diperkenalkan oleh Jelita Asian [2] yang merupakan pengembangan dari metode *stemming* yang dibuat oleh Nazief dan Adriani (1996).

Pada dasarnya, algoritma ini mengelompokkan imbuhan ke dalam beberapa kategori sebagai berikut:

1. *Inflection Suffixes* yakni kelompok-kelompok akhiran yang tidak mengubah bentuk kata dasar. Kelompok ini dapat dibagi menjadi dua:
  - *Particle (P)* atau partikel, termasuk di dalamnya adalah partikel “-lah”, “-kah”, “-tah”, dan “-pun”.
  - *Possessive Pronoun (PP)* atau kata ganti kepunyaan, termasuk di dalamnya adalah “-ku”, “-mu”, dan “-nya”.
2. *Derivation Suffixes (DS)* yakni kumpulan akhiran yang secara langsung dapat

ditambahkan pada kata dasar. Termasuk di dalam tipe ini adalah akhiran “-i”, “-kan”, dan “-an”.

3. *Derivation Prefixes (DP)* yakni kumpulan awalan yang dapat langsung diberikan pada kata dasar murni, atau pada kata dasar yang sudah mendapatkan penambahan sampai dengan 2 awalan. Termasuk di dalamnya adalah awalan yang dapat bermorfologi (“me-”, “be-”, “pe-”, dan “te-”) dan awalan yang tidak bermorfologi (“di-”, “ke-” dan “se-”).

Berdasarkan pengklasifikasian imbuhan-imbuhan tersebut, maka bentuk kata dalam bahasa Indonesia dapat dimodelkan sebagai berikut:

**[DP+[DP + [DP+]]] Kata Dasar [[+DS][+PP][+P]]**

Dengan batasan-batasan sebagai berikut :

- Tidak semua kombinasi imbuhan diperbolehkan. Kombinasi imbuhan yang dilarang dapat dilihat pada Tabel 1.
- Penggunaan imbuhan yang sama secara berulang tidak diperkenankan.
- Jika suatu kata hanya terdiri dari satu atau dua huruf, maka proses *stemming* tidak dilakukan.
- Penambahan suatu awalan tertentu dapat mengubah bentuk asli kata dasar, ataupun awalan yang telah diberikan sebelumnya pada kata dasar bersangkutan (bermorfologi).

Algoritma *CS stemmer* bekerja sebagai berikut:

1. Kata yang hendak di-*stemming* dicari terlebih dahulu pada kamus. Jika ditemukan, berarti kata tersebut adalah kata dasar, jika tidak maka langkah 2 dilakukan.
2. Cek *rule precedence*. Apabila suatu kata memiliki pasangan awalan-akhiran “be-lah”, “be-an”, “me-i”, “di-i”, “pe-i”, atau “te-i” maka langkah *stemming* selanjutnya adalah (5, 6, 3, 4, 7). Apabila kata tidak memiliki pasangan awalan-akhiran tersebut, langkah *stemming* berjalan normal (3, 4, 5, 6, 7).
3. Hilangkan *inflectional particle* P (“-lah”, “-kah”, “-tah”, “-pun”) dan kata ganti kepunyaan atau *possessive pronoun* PP (“-ku”, “-mu”, “-nya”).
4. Hilangkan *Derivation Suffixes* DS (“-i”, “-kan”, atau “-an”).
5. Hilangkan *Derivational Prefixes* DP {“di-”, “ke-”, “se-”, “me-”, “be-”, “pe-”, “te-”} dengan iterasi maksimum adalah 3 kali:
  - a) Langkah 5 ini berhenti jika:
    - Terjadi kombinasi imbuhan terlarang seperti pada Tabel 1.
    - Awalan yang dideteksi saat ini sama dengan awalan yang dihilangkan sebelumnya.
    - Tiga awalan telah dihilangkan.
  - b) Identifikasikan tipe awalan dan hilangkan. Awalan ada dua tipe:

- Standar: “di-”, “ke-”, “se-” yang dapat langsung dihilangkan dari kata.
  - Kompleks: “me-”, “be-”, “pe-”, “te-” adalah tipe-tipe awalan yang dapat bermorfologi sesuai kata dasar yang mengikutinya. Oleh karena itu, gunakan aturan pada Tabel 2 untuk mendapatkan pemenggalan yang tepat.
- c) Cari kata yang telah dihilangkan awalnya ini di dalam kamus. Apabila tidak ditemukan, maka langkah 5 diulangi kembali. Apabila ditemukan, maka keseluruhan proses dihentikan.
6. Apabila setelah langkah 5 kata dasar masih belum ditemukan, maka proses *recoding* dilakukan dengan mengacu pada aturan pada Tabel 2. *Recoding* dilakukan dengan menambahkan karakter *recoding* di awal kata yang dipenggal. Pada Tabel 2, karakter *recoding* adalah karakter setelah tanda hubung (‘-’) dan terkadang berada sebelum tanda kurung. Sebagai contoh, pada kata “menangkap” (aturan 15), setelah dipenggal menjadi “nangkap”. Karena tidak valid, maka *recoding* dilakukan dan menghasilkan kata “tangkap”. Perlu diperhatikan bahwa aturan ke-22 tidak ditemukan dalam tesis Jelita Asian [2].
7. Jika semua langkah gagal, maka input kata yang diuji pada algoritma ini dianggap sebagai kata dasar.

Apabila pada kata yang hendak di-*stemming* ditemukan tanda hubung (‘-’), maka kemungkinan kata yang hendak di-*stemming* adalah kata ulang. *Stemming* untuk kata ulang dilakukan dengan memecah kata menjadi dua bagian yakni bagian kiri dan kanan (berdasarkan posisi tanda hubung ‘-’) dan lakukan *stemming* (langkah 1-7) pada dua kata tersebut. Apabila hasil *stemming* keduanya sama, maka kata dasar berhasil didapatkan.

Pada Tabel 2, simbol *C* merupakan konsonan, *V* menandakan vokal, *A* merupakan vokal atau konsonan, dan *P* merupakan partikel.

### 2.3 Ants Algorithm

*Ant System* pada awalnya digunakan untuk menyelesaikan permasalahan *Traveling Salesman Problem* (TSP). Dianggap bahwa  $d_{ij}$  sebagai jarak antara kota  $i$  dan  $j$ . Dalam kasus *Euclidean TSP* maka,  $d_{ij}$  adalah jarak *euclidean* antara kota  $i$  dengan kota  $j$  ( $d_{ij} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2}$ ).

Diasumsikan bahwa  $m$  adalah jumlah total semut yang dipakai. Setiap semut akan memiliki karakteristik sebagai berikut:

- Semut akan memilih kota (*node*) yang akan dilewati berikutnya, dengan fungsi probabilitas antara jarak yang harus ditempuh dan seberapa level jejak *pheromone* yang ada.
- Setiap semut diberikan suatu ingatan atas kota-kota

Tabel 1. Kombinasi Imbuhan Terlarang

Awalan ( <i>prefix</i> )	Akhiran ( <i>suffix</i> ) yang dilarang
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan
te-	-an

Tabel 2. Aturan Pemenggalan Awal

Aturan	Format Kata	Pemenggalan
1	berV...	ber-V...   be-rV...
2	berCAP...	ber-CAP... dimana C!=‘r’ & P!=‘er’
3	berCAerV...	ber-CaerV... dimana C!=‘r’
4	belajar	bel-ajar
5	beC <sub>1</sub> erC <sub>2</sub> ...	be-C <sub>1</sub> erC <sub>2</sub> ... dimana C <sub>1</sub> !=‘r’ ‘l’
6	terV...	ter-V...   te-rV...
7	terCerV...	ter-CerV... dimana C!=‘r’
8	terCP...	ter-CP... dimana C!=‘r’ dan P!=‘er’
9	teC <sub>1</sub> erC <sub>2</sub> ...	te-C <sub>1</sub> erC <sub>2</sub> ... dimana C <sub>1</sub> !=‘r’
10	me{l l w y}V...	me-{l l w y}V...
11	mem{b f v}...	mem-{b f v}...
12	mempe...	mem-pe...
13	mem{rV V}...	me-m{rV V}...   me-p{rV V}...
14	men{c d j z}...	men-{c d j z}...
15	menV...	me-nV...   me-tV
16	meng{g h q k}...	meng-{g h q k}...
17	mengV...	meng-V...   meng-kV...
18	menyV...	meny-sV...
19	mempV...	mem-pV... dengan V!=‘e’
20	pe{w y}V...	pe-{w y}V...
21	perV...	per-V...   pe-rV...
23	perCAP	per-CAP... dimana C!=‘r’ dan P!=‘er’
24	perCAerV...	per-CAerV... dimana C!=‘r’
25	pem{b f V}...	pem-{b f V}...
26	pem{rV V}...	pe-m{rV V}...   pe-p{rV V}...
27	pen{c d j z}...	pen-{c d j z}...
28	penV...	pe-nV...   pe-tV...
29	peng{g h q}...	peng-{g h q}...
30	pengV...	peng-V...   peng-kV...
31	penyV...	peny-sV...
32	peIV...	pe-IV... kecuali “pelajar” yang menghasilkan “ajar”
33	peCerV...	per-erV... dimana C!=‘r w y l m n’
34	peCP...	pe-CP... dimana C!=‘r w y l m n’ dan P!=‘er’
35	terC <sub>1</sub> erC <sub>2</sub> ...	ter-C <sub>1</sub> erC <sub>2</sub> ... dimana C <sub>1</sub> !=‘r’
36	peC <sub>1</sub> erC <sub>2</sub> ...	pe-C <sub>1</sub> erC <sub>2</sub> ... dimana C <sub>1</sub> !=‘r w y l m n’

yang telah dilewatinya (dengan memakai *tabu list*).

- Setelah menyelesaikan satu perjalanan penuh, maka jalur yang dilewati semut bersangkutan, diberikan sejumlah *pheromone*.

Apabila  $\tau_{ij}(t)$  adalah jumlah *pheromone* pada *edge* kota  $i$  dan  $j$  pada waktu  $t$ , maka setiap kali melakukan satu putaran atau iterasi penuh, jumlah

*pheromone* di setiap *edge* diperbarui melalui formula 3 berikut :

$$\tau_{ij}(t+n) = \rho \tau_{ij}(t) + \Delta \tau_{ij}, \quad (3)$$

dimana  $\rho$  adalah suatu koefisien sedemikian sehingga  $(1 - \rho)$  menyatakan tingkat *evaporation* atau penguapan *pheromone* antara waktu  $t$  dan  $t+n$ . Koefisien  $\rho$  berada pada kisaran  $(0,1]$ . Total jumlah *pheromone* yang ditinggalkan pada *edge* tersebut adalah:

$$\Delta \tau_{ij} = \sum_{k=1}^m \Delta \tau_{ij}^k, \quad (4)$$

dimana  $\Delta \tau_{ij}^k$  adalah kuantitas per panjang jejak *pheromone* yang ditinggalkan pada *edge*  $(i,j)$  oleh semut  $k$  antara waktu  $t$  dan  $t+n$  yang dinyatakan sebagai berikut:

$$\Delta \tau_{ij}^k \begin{cases} Q/L_k, & \text{jika semut } k \text{ memakai edge } i, j \\ 0, & \text{sebaliknya} \end{cases}, \quad (5)$$

dimana  $Q$  adalah konstanta dan  $L_k$  adalah panjang keseluruhan rute yang dilalui oleh semut  $k$ .

Setiap semut dilengkapi *tabu list*, yang berfungsi untuk menyimpan *node-node* yang telah dilaluinya. Setelah satu kali putaran (*cycle*) penuh, maka *tabu list* digunakan untuk mendapatkan rute terpendek dan panjang *total-similarities*-nya. *Tabu list* kemudian dikosongkan, dan putaran berikutnya dilakukan kembali. Peluang semut  $k$  yang berada pada *node*  $i$ , untuk memilih *node*  $j$  berikutnya adalah sebagai berikut:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{k \in \text{allowed}_k} [\tau_{ik}(t)]^\alpha [\eta_{ik}]^\beta}, \quad (6)$$

dimana  $\eta_{ij}$  adalah tingkat penglihatan yang nilainya adalah  $1/d_{ij}$ ,  $\text{allowed}_k = \{N - \text{tabu}_k\}$ , serta  $\alpha$  dan  $\beta$  adalah parameter pengontrol seberapa penting jejak *pheromone* dan penglihatan, dimana  $\alpha \geq 0$  dan  $\beta \geq 0$ .

### 3. Klasifikasi Dokumen Berita Berbahasa Indonesia

#### 3.1 Enhanced Confix Stripping Stemmer

Setelah melakukan beberapa analisa dan percobaan kecil, ditemukan beberapa contoh kata yang gagal di-*stemming* oleh metode *confix stripping stemmer*. Analisa atas beberapa kata yang gagal di-*stemming* tersebut adalah sebagai berikut:

1. Kurangnya aturan pemenggalan awalan untuk kata-kata dengan format “mem+p...” seperti pada kata “mempromosikan”, “mempromoteksi”, dan “memprediksi”.
2. Kurangnya aturan pemenggalan awalan untuk kata-kata dengan format “men+s...”.

Contohnya pada kata “mensyaratkan”, dan “mensyukuri”.

3. Kurang relevannya aturan 17 untuk pemenggalan awalan pada kata-kata dengan format “menge + kata dasar”, seperti pada kata “mengerem”.
4. Kurang relevannya aturan 30 untuk pemenggalan awalan pada kata-kata dengan format “penge + kata dasar”, seperti pada kata “pengeboman”.
5. Kurangnya aturan pemenggalan awalan untuk kata-kata dengan format “peng+k...” seperti pada kata “pengkajian”.
6. Adanya elemen pada beberapa kata dasar yang menyerupai suatu imbuhan. Kata-kata seperti “pelanggan”, dan “pelaku” gagal di-*stemming* karena akhiran “-an” dan “-ku” seharusnya tidak dihilangkan.

Tabel 3. Revisi untuk Tabel 2

Aturan	Format Kata	Pemenggalan
14	men{c[d]j[s]z}...	men-{c[d]j[s]z}...
17	mengV...	meng-V...   meng-kV...   (mengV-... jika V='e')
19	mempA...	mem-pA... dimana A!='e'
29	pengC...	peng-C...
30	pengV...	peng-V...   peng-kV...   (pengV-... jika V='e')

Berdasarkan kegagalan-kegagalan tersebut, maka algoritma *confix stripping stemmer* berusaha dioptimalisasi dengan menambahkan beberapa perbaikan. Algoritma *confix stripping stemmer* yang telah mendapatkan revisi dan tambahan algoritma ini kemudian disebut *Enhanced confix stripping stemmer*. Beberapa revisinya adalah sebagai berikut:

1. Merevisi aturan 14, 17, 19, dan 30 pada Tabel 2 agar *stemming* berhasil pada kata-kata dengan format “mem+p...”, “men+s...”, “menge+...”, “penge+...”, dan “peng+k...”. Revisi aturan ini dapat dilihat pada Tabel 3.
2. Menambahkan suatu langkah tambahan untuk mengatasi kesalahan pemenggalan akhiran yang seharusnya tidak dilakukan. Langkah ini disebut dengan *loopPengembalianAkhiran*. Langkah ini dilakukan apabila proses *recoding* gagal. Pada akhir setiap langkah, dilakukan pemeriksaan ke kamus untuk menguji hasilnya. Langkah *loopPengembalianAkhiran* dideskripsikan sebagai berikut:
  - 1) Kembalikan seluruh awalan yang telah dihilangkan sebelumnya, sehingga menghasilkan model kata seperti berikut:

**[DP+[DP+[DP]]] + Kata Dasar**

lalu proses pemenggalan awalan dilakukan.
  - 2) Kembalikan akhiran sesuai dengan urutan model kata. Ini berarti bahwa

pengembalian dimulai dari DS (“-i”, “-kan”, “-an”), lalu PP (“-ku”, “-mu”, “-nya”), dan terakhir adalah P (“-lah”, “-kah”, “-tah”, “-pun”). Pada setiap pengembalian, lakukan langkah 3) hingga 5) berikut. Khusus untuk akhiran “-kan”, pengembalian pertama dimulai dengan “k”, baru kemudian dilanjutkan dengan “an”.

- 3) Lakukan pencarian di kamus. Apabila ditemukan, proses dihentikan. Apabila gagal, maka lakukan proses pemenggalan awalan berdasarkan aturan pada Tabel 2 (dengan revisi Tabel 3).
- 4) Lakukan *recoding* apabila diperlukan.
- 5) Apabila pencarian di kamus tetap gagal setelah *recoding*, maka awalan-awalan yang telah dihilangkan dikembalikan seperti model kata pada langkah 1).

### 3.2 Ant Based Document Clustering

Algoritma *document clustering* berbasis *Ant System* secara garis-besar terbagi menjadi dua tahap, yakni pencarian rute terpendek antar-dokumen (*trial phase*) dan mengelompokkan dokumen-dokumen berdasarkan urutan rute terpendek yang didapatkan pada fase sebelumnya (*dividing phase*).

Pencarian rute terpendek pada *trial phase* mengadopsi *Ants Algorithm*, dimana probabilitas semut  $k$  ( $p_{ij}^k(t)$ ) untuk memilih *node* (dokumen)  $j$  dari posisi *node* (dokumen)  $i$ , menggunakan formula 7 berikut:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [S_{ij}]^\beta}{\sum_{k \in Z_k} [\tau_{ik}(t)]^\alpha [S_{ik}]^\beta}, \quad (7)$$

dimana  $Z_k$  merepresentasikan himpunan *node-node* (dokumen) yang belum dilewati oleh semut  $k$ ,  $\tau_{ij}(t)$  merepresentasikan intensitas jumlah *pheromone* pada *edge*  $i$  dan  $j$ ,  $\alpha$  berfungsi sebagai parameter pengatur seberapa penting faktor intensitas *pheromone* dalam pemilihan *node*  $j$  yang akan dilewati, dan  $\beta$  sebagai parameter penglihatan.  $S_{ij}$  adalah *cosine distance* antara *node* dokumen  $i$  dan  $j$  yang dinyatakan sebagai berikut [1]:

$$S_{ij} = \frac{\sum_k [w_{ki}] \cdot [w_{kj}]}{\|d_i\|^2 \cdot \|d_j\|^2}, \quad (8)$$

dimana  $w_{ki}$  dan  $w_{kj}$  adalah pembobotan TF-IDF *term*  $k$  pada dokumen  $i$  dan  $j$ .  $\|d_i\|^2$  dan  $\|d_j\|^2$  adalah panjang dari vektor dokumen  $i$  dan  $j$ . Sebagai contoh  $\|d_i\|^2 = (t_1^2 + t_2^2 + t_3^2 + \dots + t_k^2)^{1/2}$ , dimana  $t_k$  adalah *term* ke- $k$  pada vektor dokumen  $d_i$ .

Setelah penelusuran *node-node* oleh semut dalam satu kali iterasi selesai, maka langkah

berikutnya adalah menambah jumlah *pheromone* pada *edge-edge* yang dilewati oleh tiap semut, dan mengevaporasinya. Total jumlah *pheromone* yang ditambahkan pada *edge*  $i,j$  ( $\Delta\tau_{ij}$ ) diatur melalui formula 9 berikut:

$$\Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k, \quad (9)$$

dimana  $\Delta\tau_{ij}^k$  adalah jumlah *pheromone* yang ditinggalkan pada *edge* ( $i,j$ ) oleh semut  $k$  yang ditentukan melalui formula 10 berikut:

$$\Delta\tau_{ij}^k \begin{cases} N * L_k, & \text{jika semut } k \text{ memakai } edge \ i, j \\ 0, & \text{sebaliknya} \end{cases}, \quad (10)$$

dimana  $N$  adalah jumlah total dokumen dan  $L_k$  adalah panjang rute (nilai *total-similarities edge-edge* yang dilalui) yang ditempuh oleh semut  $k$  pada waktu ke- $t$ . Sedangkan proses evaporasi *pheromone* dilakukan berdasarkan formula 11 berikut:

$$\tau_{ij}(t+n) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}, \quad (11)$$

dimana  $\rho$  adalah koefisien yang menandakan tingkat evaporasi *pheromone*.

Perlu diperhatikan bahwa rute terpendek terbaik adalah rute yang menghasilkan *total similarities* terbesar. Hal ini karena nilai jarak *edge* yang digunakan dihitung menggunakan formula *similarities* antar-dokumen yakni *cosine distance*.

Tahap berikutnya yakni *dividing phase*, adalah pembentukan *cluster* berdasarkan urutan dokumen-dokumen yang diperoleh dari tahap *trial phase* sebelumnya. Dokumen pertama dari urutan tersebut, dianggap sebagai *centroid*  $\mu$  dari grup pertama. Proses pengelompokkan dimulai dari  $\mu$  dan dokumen berikutnya dalam urutan (disebut dokumen pembanding  $D$ ) dimana formula 11 ini dipakai sebagai pengujian:

$$\delta < \cos(\mu, D), \quad (12)$$

dimana  $\delta$  adalah koefisien *attachment* dan nilainya berkisar pada (0,1], dan  $\cos(\mu, D)$  adalah *cosine distance* antara dokumen  $\mu$  dengan dokumen  $D$ .

Apabila kondisi tersebut benar, maka dokumen  $D$ , menjadi grup dalam *centroid*  $\mu$ , dan dokumen berikutnya pada urutan menjadi dokumen  $D$  berikutnya. Apabila kondisi tersebut salah, maka dokumen  $D$  saat itu, dijadikan *centroid*  $\mu$  yang baru. Pembandingan dengan dokumen berikutnya terus dilakukan hingga mencapai dokumen terakhir.

### 3.3 Evaluasi Hasil Klasifikasi

Konsep evaluasi *external measure* yakni *recall*, *precision*, dan *F-measure* pada bidang *information retrieval* dapat diadopsi dalam bidang

*clustering* dokumen. Setiap *cluster* yang dihasilkan dianggap sebagai hasil *retrieval* dan kelompok-kelompok (kelas) dokumen yang telah diidentifikasi sebelumnya, yang dianggap sebagai *cluster* ideal yang seharusnya dihasilkan pengklasifikasi [1]. Secara lebih spesifik, untuk setiap kelas  $i$  dan *cluster*  $j$  maka,

$$Recall(i,j) = R_{ij} = n_{ij}/n_i, \quad (13)$$

$$Precision(i,j) = P_{ij} = n_{ij}/n_j, \quad (14)$$

dimana  $n_{ij}$  adalah jumlah dokumen kelas  $i$  pada cluster  $j$ ,  $n_i$  adalah jumlah dokumen pada kelas  $i$ , dan  $n_j$  adalah jumlah dokumen pada cluster  $j$ . Nilai *F-measure* kelas  $i$  dan cluster  $j$  dinyatakan sebagai berikut:

$$F_{ij} = (2 * R_{ij} * P_{ij}) / (R_{ij} + P_{ij}). \quad (15)$$

Nilai keseluruhan *F-measure* didapatkan melalui formula berikut:

$$F = \sum_i \frac{n_i}{n} \max\{F_{ij}\}, \quad (16)$$

dimana  $n$  adalah jumlah keseluruhan dokumen,  $n_i$  adalah jumlah dokumen pada kelas  $i$ , dan  $\max\{F_{ij}\}$  adalah nilai  $F_{ij}$  terbesar yang ditemukan pada kelas  $i$  untuk keseluruhan cluster  $j$ .

Tabel 4. *Corpus* Uji Coba

ID	Event	Jumlah file
1	Kasus sodomi Anwar Ibrahim	26
2	Kasus suap Artalyta (Ayin)	27
3	Bencana gempa bumi di China	19
4	FPI dalam insiden Monas	35
5	Konflik Israel-Palestina	25
6	Dua tahun tragedi lumpur Lapindo	21
7	Badai nargis di Myanmar	24
8	Pengunduran jadwal Pemilu	10
9	Perkembangan harga emas	18
10	Kiprah Khofifah dalam Pilkada Jatim	15
11	Kematian mantan presiden Soeharto	28
12	Festival film Cannes	5

#### 4. Uji Coba

Aplikasi ini dibangun dengan menggunakan Java dan MySQL. *Corpus* yang digunakan sebagai data uji dikumpulkan dari artikel-artikel berita *online* [www.kompas.com](http://www.kompas.com) dan [www.detik.com](http://www.detik.com), yang berkisar dari tanggal 11 Januari 2008 hingga 4 Juli 2008, dengan jumlah total 253 berita yang dikelompokkan ke dalam 12 *event* yang berbeda.

Tabel 5. Klasifikasi Kegagalan *CS Stemmer*

Tipe kesalahan	Contoh Kasus		
	Awal	<i>stemming</i>	Seharusnya
Kesalahan pemenggalan akhiran	diriku	diriku	diri
Tidak ada aturan pemenggalan awalan "mempr-"	memprotes	memprotes	protes
Tidak ada aturan pemenggalan awalan "menge-"	mengemukakan	mengemukakan	muka
Tidak ada aturan pemenggalan awalan "mens-"	mensyaratkan	mensyaratkan	syarat
Tidak ada aturan pemenggalan awalan "penge-"	pengeboman	pengeboman	bom
Tidak ada aturan pemenggalan awalan "pengk-"	pengkajian	pengkajian	kaji
<i>Rule precedence</i>	dikenai	dikenai	kena
Kata turunan	diberitahu	diberitahu	beritahu
Kesalahan deteksi kombinasi imbuhan terlarang	keterlibatan	keterlibatan	terlibat
Kesalahan aturan pemenggalan awalan ke-18	menyatakan	menyatakan	nyata
Kesalahan aturan pemenggalan awalan ke-31	penyanyi	penyanyi	nyanyi
Sisipan	temaram	temaram	taram
<i>Overstemming</i>	penyidikan	sidi	sidik
<i>Understemming</i>	mengalami	alami	alam
Nama orang	Gumai	Guma	Gumai

Tabel 6. Klasifikasi Kegagalan *Enhanced CS Stemmer*

Tipe kesalahan	Contoh Kasus		
	Awal	<i>stemming</i>	Seharusnya
Sisipan	temaram	temaram	taram
<i>Overstemming</i>	penyidikan	sidi	sidik
<i>Understemming</i>	mengalami	alami	alam
Nama orang	Gumai	Guma	Gumai
Kesalahan aturan pemenggalan awalan ke-18	menyatakan	menyatakan	nyata
Kesalahan aturan pemenggalan awalan ke-31	penyanyi	penyanyi	nyanyi
Kata turunan	diberitahu	diberitahu	beritahu

*Corpus* data uji aplikasi ini selengkapnya dapat dilihat pada Tabel 4. Selain *corpus*, digunakan juga *stoplist* dan kamus kata dasar untuk proses *stemming* yang didapatkan dari Tugas Akhir Ari Novan Setiono [11]. Jumlah *stoplist* yang digunakan meliputi 792 *stopword* dan jumlah kata dasar kamus sebanyak 29.337 kata.

Tabel 7. Jumlah *Terms* Hasil Preprocessing

<i>Stoplist Removal</i>	<i>Stemmer</i>	Jumlah <i>Terms</i>
Tidak	Tanpa <i>Stemmer</i>	7327
	CS <i>Stemmer</i>	5059
	Enhanced CS <i>Stemmer</i>	<b>4934</b>
Ya	Tanpa <i>Stemmer</i>	6780
	CS <i>Stemmer</i>	4871
	Enhanced CS <i>Stemmer</i>	<b>4758</b>

Tabel 8. Jumlah Kata Ter-*stemming*

<i>Stoplist Removal</i>	<i>Stemmer</i>	Jumlah kata ter- <i>stemming</i>	Pengurangan <i>terms</i> (%)
Tidak	CS <i>Stemmer</i>	2268	30.95
	Enhanced CS <i>Stemmer</i>	<b>2393</b>	<b>32.66</b>
Ya	CS <i>Stemmer</i>	1909	28.16
	Enhanced CS <i>Stemmer</i>	<b>2022</b>	<b>29.82</b>

#### 4.1 Uji Preprocessing dan *Stemmer*

Tujuan dari pengujian ini adalah untuk melihat keberhasilan proses *preprocessing* dokumen. Dalam fase *preprocessing*, aplikasi dapat menggunakan dua tipe *stemmer*, yaitu CS *stemmer* dan *Enhanced CS stemmer*. Selain itu, tujuan pengujian ini adalah untuk membandingkan kemampuan kedua *stemmer* tersebut pada *corpus* yang digunakan. Pengujian kedua *stemmer* dilakukan dengan atau tanpa proses *stoplist removal* sebelumnya.

Berdasarkan Tabel 7, dapat disimpulkan bahwa ditemukan sebanyak 7.327 kata yang berbeda pada keseluruhan dokumen dalam *corpus*. Jumlah kata-kata yang ter-*stemming* dapat dilihat pada Tabel 8. Beberapa contoh kesalahan *stemming* oleh kedua algoritma *stemmer* ini diklasifikasikan berdasarkan jenis-jenis kesalahannya, dan dapat dilihat pada Tabel 5 dan 6.

#### 4.2 Uji *Trial Phase*

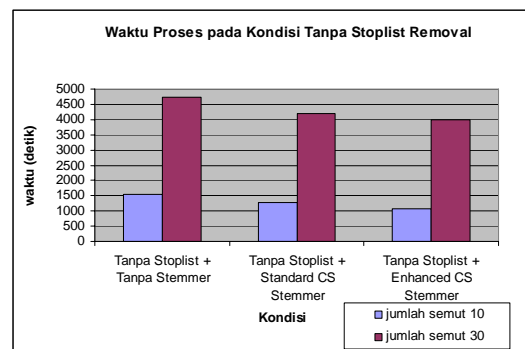
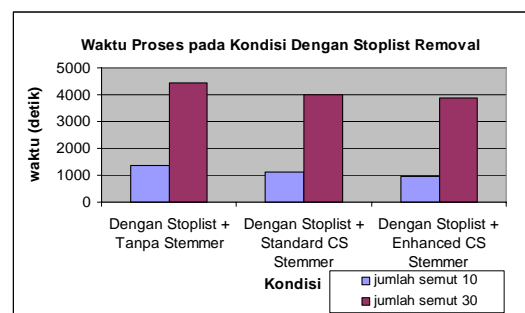
Pencarian rute terpendek pada *graph* dokumen dilakukan pada 6 kondisi *preprocessing* berbeda sebagai berikut:

1. Tanpa *stoplist removal* dan tanpa *stemmer*.
2. Tanpa *stoplist removal* dan CS *stemmer*.
3. Tanpa *stoplist removal* dan *Enhanced CS stemmer*.
4. Dengan *stoplist removal* dan tanpa *stemmer*.
5. Dengan *stoplist removal* dan CS *stemmer*.
6. Dengan *stoplist removal* dan *Enhanced CS stemmer*.

Karena banyaknya jumlah parameter dan tidak adanya informasi atas berapa nilai *total similarities* atau rute terbaik yang sebenarnya, maka parameter-parameter yang diuji mengacu pada keberhasilan eksperimen Marco Dorigo dalam melakukan pencarian rute terpendek pada permasalahan *Traveling Salesman Problem* menggunakan *Ants Algorithm* [5]. Nilai parameter tersebut adalah:  $\alpha \in \{0, 0.5, 1, 2, 5\}$ ,  $\beta \in \{0, 0.5, 1, 2, 5\}$ ,  $\rho=0.5$ , dan jumlah semut  $\in \{10, 30\}$ .

Percobaan dilakukan pada 6 kondisi tersebut dengan jumlah iterasi atau putaran maksimum adalah 1000. Hasil dan konfigurasi terbaik pada tiap kondisi dapat dilihat pada Tabel 9. Melalui pengamatan langsung pada proses *trial phase* pada aplikasi, ditemukan beberapa hal sebagai berikut:

1. Penggunaan jumlah semut yang lebih banyak (30) mampu mencari *total-similarities* terbesar secara lebih cepat, pada iterasi yang lebih singkat, jika dibandingkan dengan penggunaan jumlah semut yang lebih sedikit (10). Akan tetapi, penggunaan jumlah semut yang lebih banyak dapat membawa dampak pada lamanya waktu proses seperti pada Gambar 1 dan 2.
2. Nilai rute terpendek (*total-similarities*) terbaik didapatkan dengan konfigurasi parameter  $\alpha=2$ ,  $\beta=5$ , dan  $\rho=0.5$ .
3. Waktu proses juga dipengaruhi oleh banyaknya *total terms* yang digunakan sebagai representasi vektor dokumen.

Gambar 1. Waktu Proses tanpa *Stoplist Removal*Gambar 2. Waktu Proses dengan *Stoplist Removal*

Tabel 9. *Trial Phase* Terbaik pada 6 Kondisi

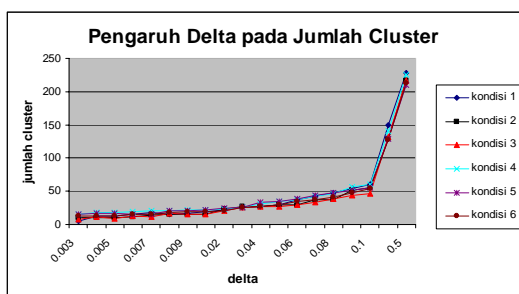
Kondisi	Parameter Trial Phase				Total Similarities
	jumlah semut	$\alpha$	$\beta$	$\rho$	
Tanpa Stoplist + Tanpa Stemmer	30	2	5	0.5	75.1
Tanpa Stoplist + CS Stemmer	30	2	5	0.5	84.0
Tanpa Stoplist + Enhanced CS Stemmer	30	2	5	0.5	85.9
Dengan Stoplist + Tanpa Stemmer	10	2	5	0.5	76.9
Dengan Stoplist + CS Stemmer	30	2	5	0.5	85.3
Dengan Stoplist + Enhanced CS Stemmer	30	2	5	0.5	85.4

### 4.3 Uji *Dividing Phase*

Berdasarkan hasil terbaik yang diperoleh pada tiap 6 kondisi berbeda pada skenario 2 sebelumnya, maka uji coba berikutnya adalah melakukan pembentukan *cluster-cluster* (*dividing phase*) berdasarkan urutan dokumen pada *trial phase* terbaik tiap kondisi tersebut. Hasil terbaik *dividing phase* pada *trial phase* terbaik tiap 6 kondisi tersebut, dapat dilihat pada Tabel 10. Melalui uji coba, dapat dilihat bahwa jika nilai  $\delta$  semakin kecil (mendekati 0), maka jumlah *cluster* juga cenderung makin sedikit, begitu juga sebaliknya jika nilai  $\delta$  mendekati 1. Hal ini dapat dilihat pada Gambar 3.

Tabel 10. *Dividing Phase* Terbaik pada *Trial Phase* Terbaik

Kondisi	$\delta$	jumlah cluster	F-Measure
Tanpa Stoplist + Tanpa Stemmer	0.022	24	0.84
Tanpa Stoplist + CS Stemmer	0.05	29	0.85
Tanpa Stoplist + Enhanced CS Stemmer	0.06	29	0.83
Dengan Stoplist + Tanpa Stemmer	0.024	24	<b>0.86</b>
Dengan Stoplist + CS Stemmer	0.3	26	0.82
Dengan Stoplist + Enhanced CS Stemmer	0.02	21	0.84

Gambar 3. Pengaruh  $\delta$  pada Jumlah Cluster

## 5. Kesimpulan dan Saran

### 5.1 Kesimpulan

1. Algoritma semut dapat diimplementasikan untuk klasifikasi dokumen teks, dalam hal ini adalah dokumen berita berbahasa Indonesia dengan tingkat evaluasi *F-measure* terbaik pada data uji coba adalah 0.86.
2. Peningkatan nilai koefisien *attachment* ( $\delta$ ) sebagai nilai batas pembentukan *cluster*, cenderung memperbesar jumlah grup atau *cluster* yang dihasilkan. Sebaliknya, penurunan nilai  $\delta$  cenderung memperkecil jumlah grup atau *cluster* yang dihasilkan.
3. Algoritma *confix stripping stemmer* berhasil diimplementasikan dan mampu mereduksi hingga 30.95% jumlah *terms* dari total keseluruhan *terms* pada data uji coba.
4. Algoritma *enhanced confix stripping stemmer* berhasil dikembangkan untuk memperbaiki kesalahan-kesalahan *confix stripping stemmer*, dan mampu mereduksi hingga 32.66% jumlah *terms* dari total keseluruhan *terms* pada data uji coba.
5. Penggunaan *stemming* meskipun dapat mengurangi waktu proses klasifikasi dokumen, namun dapat mengurangi performa klasifikasi karena hilangnya *term-term* yang menjadi ciri signifikan suatu dokumen.

### 5.2 Saran

1. Perlunya eksperimen lebih jauh untuk mengetahui konfigurasi parameter yang mampu menghasilkan nilai *F-measure* yang lebih baik.
2. Perlunya riset dan pengembangan metode yang mampu mempercepat waktu proses *trial phase*, misalnya dengan mengimplementasikan metode komputasi paralel.
3. Klasifikasi dokumen berbasis algoritma semut bersifat *unsupervised classification*. Oleh karena itu, dibutuhkan riset lebih lanjut untuk mengembangkan metode klasifikasi dokumen berbasis algoritma semut ini ke arah *supervised classification*.
4. Algoritma semut merupakan algoritma fleksibel yang dapat digunakan dalam berbagai permasalahan termasuk optimasi dan klasifikasi. Penyusunan Tugas Akhir ini, diharapkan dapat memicu riset-riset lain yang berbasis pada algoritma semut.
5. Perlunya implementasi metode yang mampu mengurangi tingginya dimensi *feature* (*terms*) vektor dokumen.
6. Perlunya riset lebih jauh untuk penyempurnaan algoritma *enhanced confix stripping stemmer*, khususnya untuk mengatasi permasalahan *stemming* pada kata bersisipan.
7. *Enhanced confix stripping stemmer* dapat digunakan sebagai *backbone* untuk aplikasi

*automatic thesaurus, recommender system, information retrieval*, dan sistem-sistem lain yang membutuhkan akurasi *stemming* Bahasa Indonesia yang tinggi.

## 6. Daftar Pustaka

- [1] Abdelmalek A., Zakaria E., Michel S., Mimoun M., 2007. "Evaluation and Comparison of Concept Based and N-Grams Based Text Clustering using SOM". TIMC-IMAG Laboratory IN3S, Joseph Fourier University.
- [2] Asian J., 2007. "Effective Techniques for Indonesian Text Retrieval". PhD thesis School of Computer Science and Information Technology RMIT University Australia.
- [3] C.J. van Rijsbergen, "Information Retrieval". Butterworths, London, 1979.
- [4] Deitel H.M dan Deitel P.J. 2003. Java How To Program Fifth Edition. New Jersey: Prentice Hall.
- [5] Dorigo M., Maniezzo V., Colomi A., 1996. "The Ant System: Optimization by Colony of Cooperating Agents". IEEE Transactions on Systems, Man, and Cybernetics-Part B.
- [6] Machnik L., 2006. "ACO Documents Clustering – Details of Processing and Results of Experiments". Annales UMCS Informatica Poland.
- [7] Machnik L., 2005. "Ants in Text Document Clustering". Proceedings of The International Conference on Systems, Computing Sciences and Software Engineering (SCSS 2005).
- [8] Machnik L., 2004. "Documents Clustering Techniques". IBIZA 2004, Annales UMCS Informatica Poland.
- [9] Machnik L., 2006. "Documents Clustering Method based on Ants Algorithm". Proceedings of the International Multiconference on Computer Science and Information Technology, pp.123-130.
- [10] Salton G., 1989. "Automatic Text Processing". Addison Wesley.
- [11] Setiono, Ari Novan. "Implementasi Aplikasi Information Retrieval untuk Pendeteksian dan Klasifikasi Berita Kejadian Berbahasa Indonesia Berbasis Web". Tugas Akhir, Teknik Informatika, Institut Teknologi Sepuluh Nopember Surabaya, 2001.