ENHANCED CONFIX STRIPPING STEMMER AND ANTS ALGORITHM FOR CLASSIFYING NEWS DOCUMENT IN INDONESIAN LANGUAGE

Agus Zainal Arifin [1], I Putu Adhi Kerta Mahendra [2], Henning Titi Ciptaningtyas [3]
[1,2,3] Informatics Department, Faculty of Information Technology
Sepuluh Nopember Institute of Technology – Surabaya, 60111, Indonesia
agus.za@its-sby.edu [1], adhi@cs.its.ac.id [2], henning@its-sby.edu [3]

ABSTRACT

Ants algorithm is a universal and flexible solution which was first designed for solving optimization problem such as Traveling Salesman Problem. Analogy between finding the shortest way by ants and finding documents most alike, became a stimulus of ant based text document clustering method. This method consist of two phases, which are finding documents most alike (trial phase) and clusters making (dividing phase).

In this paper, we implemented ant based document clustering method on 253 news documents in Indonesian language. Beside that, we developed enhanced confix stripping stemmer as an improvement of confix stripping stemmer for stemming news documents in Indonesian language.

Result of the experiments proved that ants algorithm can be applied for classification of news document in Indonesian language, with the best F-measure achieved from experiments was 0.86. The experiments also showed that enhanced confix stripping stemmer had been succesfully solved confix stripping stemmer's problems and reduce terms size up to 32.66%, while confix stripping stemmer only reduce 30.95%.

Keywords: ants algorithm, confix stripping stemmer, text document clustering, F-measure.

1 INTRODUCTION

The rapidly growth of internet and world wide web in 1990s had brought a new era of information source, where internet became a popular publication tool. By looking exact the same chance, the news companies also tried to satisfy its reader by providing their news articles online, which is flexible and effective in time and cost point of view, compared to the conventional newspapers and magazines. The next problems arise are the increase of people's need of relevant event or topics from the news he/she reads. It is more complicated along with the fact that the news storage has became a vast and unorganized data. Because of that, strategy of automatic classification for these documents is needed.

Ants algorithm is a universal solution which is first designed to solve Traveling Salesman Problem (TSP). Then, ants algorithm was applied on unsupervised classification (clustering) of text documents, by analogying documents as graph nodes. The idea of finding shortest path of those document nodes has triggered a new clustering method which is known as ant based document clustering method [6][7][9]. In this paper, we try to implementing ants based document clustering for unsupervised classification of news documents in Indonesian language.

Stemming is an adjustment and improvement strategy for text document classification in a specific language. Unfortunately, document classification research in Indonesian language was less than expected, although the density of Indonesian people is considered one of the biggest in the world, along with its growing internet user potential. Unlike English, Indonesian language has more complex category of affixes, which includes prefixes, suffixes, infixes (insertions), and confixes (combinations of prefixes and suffixes). Several Indonesian language stemmer have been developed by Arifin and Setiono (2002) [11], Nazief and Adriani (1996), Vega (2001), and Jelita Asian (2007) [2]. From experiments, Jelita proved that her confix stripping stemmer was the most effective scheme among those stemmers, although stemming failures were still occur in this stemmer. In this paper, we try to fix its failures and propose enhanced confix stripping stemmer, an improvement of confix stripping stemmer.

ISSN 2085-1944

2 TEXT PROCESSING

2.1 Representation of Textual Documents

Before implementing text document classification method, such method to transform digital texts into an efficient and understandable model is needed, so they can be analyzed. Vector space model is the most used approach to represent textual documents. In this model, each documents *dj* will be transformed into a vector [3]:

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

where w_{ij} represents the weight of term *i* on document *j*.

Terms weight can be represented in binary occurence (*true* or *false*), frequency, or based on frequency and its inverse document frequency (TF-IDF). The classical TF-IDF weighting method has shown a better performance than binary or frequency-only weighting method, and calculated as follows [10]:

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

where w_{ij} represents weight of term *i* on document *j*, tf_{ij} represents frequency of term *i* on document *j*, *N* represents number of processed documents and df_j represents number of documents that actually have term *i* at least one.

2.2 Confix Stripping Stemmer

Confix stripping (CS) stemmer is a stemming method for Indonesian language originally proposed by Nazief and Adriani (1996) and extended by Jelita Asian [2]. This method groups affixes into categories :

1. *Inflection Suffixes*, the set of suffixes that do not alter the root word. The inflections are divided into:
   - *Particle (P)*, including "-lah", "-kah", "-tah", and "-pun" particles.
   - *Possessive Pronoun (PP)*, including "-ku" , "-mu", and "-nya".
2. *Derivation Suffixes* (*DS)*, the set of suffixes that are directly applied to root words. Included on this kind of suffix are "-i", "-kan", and "-an".
3. *Derivation Prefixes (DP)*, the set of prefixes that are applied either directly to root words, or to words that have up to two other derivation prefixes. This kind of prefixes divided into complex prefix ("me-", "be-", "pe-", and "te-") and plain prefixes ("di-", "ke-" and "se-").

This classification of affixes leads to an order of Indonesian language word model as follows:

[DP+[DP + [DP+]]] root-word [[+DS][+PP][+P]]

However, there are exceptions and limitations that are incorporated in the rules:

- Not all combinations of affixes are possible. Disallowed prefix and suffix combinations are shown in Table 1.
- Same affix cannot be repeatedly applied.
- If a word has on or two characters only, the stemming is not attempted.
- Prefixes adding may change the root word or previously-applied prefix. For example, the prefix "me-" that applied to the word "tambah" (add), will produce the word "menambah" (adding) in which the "t" is altered to "n".

Confix stripping stemmer works as follows:

Table 1. Disallowed Prefix-Suffix Combinations

Prefix	Disallowed suffixes
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan
te-	-an

1. At start of processing and each step, check the current word against the root word dictionary. If the lookup succeeds, the word is considered to be a stem, and processing stops.
2. Check the rule precedence. If the word has combination of prefix-suffix of "be-lah", "be-an", "me-i", "di-i", "pe-i", or "te-i", then the next steps attempted are (5, 6, 3, 4, 7). Otherwise, the stemming steps executed normally (3, 4, 5, 6, 7).
3. Remove inflectional particle P ("-lah", "-kah", "-tah", "-pun") if existed in the word, and continued by removing possessive pronoun PP ("-ku", "-mu", "-nya").
4. Remove derivation suffixes DS ("-i", "-kan", or "-an").
5. Remove derivation prefixes DP ("di-", "ke-", "se-", "me-", "be-", "pe", "te-") with a maximum of three times iteration:
   a) Stop processing if:
      - The identified prefix forms a disallowed affix combination with the suffix that was removed in previous steps;
      - The identified prefix is identical to a previously removed prefix; or
      - Three prefixes have already been removed.
   b) Identified prefix type and remove it. Prefixes may be of two types:
      - Plain: prefixes "di-", "ke-", and "se-" that can be removed directly.

- Complex: prefixes “me-”, “be-”, “pe”, and “te-” that may alter the root word. Use the rule described in Table 2, 3, 4, or 5 to get the correct prefix removal.

c) This step (Step 5) is re-attempted when dictionary lookup failed on the current word.

6. If, after prefix removal on Step 5, the root word has still not been found, check whether recoding is possible by examining the last column of Table 2, 3, 4, or 5. Recoding is attempted by replacing and adding recoding character on the first letter of the word. Recoding characters in Table 2, 3, 4, and 5 are a lowercase characters following the hyphen (‘-’) and sometimes outside the braces. For example, when removing prefix “me-” on word “menangkap” (to catch), by examining rule 6 on Table 2, there are two possible recoding characters, “n” and “t”. Adding “n” will produce word “nangkap”, and unfortunately it is not a valid word. Adding “t” will produce word “tangkap” (catch) which is the valid stem.
7. If all steps are failed, then the algorithm returns the original unstemmed word.

There is a special condition when hyphen (‘-’) is found in the word to be stemmed. It is indicated that the word may be a plural or repeated word. For these kinds of words, stemming is attempted separately on the sub-word preceding and following the hyphen. The stemming succeeds if the two sub-words have the same root word.

The letter ‘C’ in Table 2, 3, 4, and 5 indicates a consonant, letter ‘V’ indicates a vowel, letter ‘A’ indicates any letter, and letter ‘P’ indicates a short fragment of a word, such as “er”.

2.3 Enhanced Confix Stripping Stemmer

After conducting limited experiments, we obtained some failures made by confix stripping stemmer and classified them as follows:

1. No prefix removal rule for words with construction of “mem+p...”, for example, “mempromosikan”, “memproteksi”, and “memprediksi”.
2. No prefix removal rule for words with construction of “men+s...”, for example, “mensyaratkan”, and “mensyukuri”.
3. No prefix removal rule for words with construction of “menge+...”, for example, “mengerem”.
4. No prefix removal rule for words with construction of “penge+...”, for example, “pengeboman”.
5. No prefix removal rule for words with construction of “peng+k...”, for example, “pengkajian”.
6. Suffix removal failures – sometimes the last fragment of a word resembles certain suffix. For examples, the words like “pelanggan” and “pelaku” failed to be stemmed, because of the “-an” and “-ku” on the last part of the word should not be removed.

Based on those failures, we try to extend confix stripping stemmer, and present our modified confix stripping stemmer that is called enhanced confix stripping stemmer. The improvements deal as follows:

1. Modifying some rules on Table 2 and 3, so that stemming process on words with construction of “mem+p...”, “men+s...”, “menge+...”, “penge+...”, and “peng+k...” can be done. These modifications are listed in Table 6.
2. Adding additional stemming step to solve the suffix removal problem. We called this additional step, *loopPengembalianAkhiran*. This step is performed when recoding (step 6, CS stemmer) failed.

In each process of *loopPengembalianAkhiran*, the dictionary lookup is performed to check the result upon current word. The processes in *loop PengembalianAkhiran* are defined as follows:

1) Restore the word to its pre-recoding form and return all the prefixes that have been removed in the last process, so it will create word model like follows:

[DP+[DP+[DP]]] + Root word

Next, the prefix removal is attempted. If dictionary lookup succeed, then the process stops. Otherwise, the next step is executed.

2) Return the suffixes that have been removed previously. It means that the return starts from DS (“-i”, “-kan”, “-an”) if exist, then followed by PP(“-ku”, “-mu”, “-nya”), and the last is P (“-lah”, “-kah”, “-tah”, “-pun”). On each returning, step 3) to 5) below is attempted. Special case for DS “-kan”, character “k” is restored first and step 3) to 5) is executed. If still failed, then “an” is restored.
3) Prefix removal is performed according to rules defined in Table 2, 3, 4, and 5 (with modifications on Table 6 and 7).
4) Recoding is performed.

If dictionary lookup does not succeed, then return the word to its pre-recoding form and return all the prefixes that have been removed. The next suffix according order in Step 1) is restored and Step 3) to 5) are performed against current word.

Tabel 2. Prefix Removal Rules for Prefix "me-"

Rule	Construction	Prefix Removal
1	me{l|r|w|y}V...	me-{l|r|w|y}V...
2	mem{b|f|v}...	mem-{b|f|v}...
3	mempe...	mem-pe...
4	mem{rV|V}...	me-m{rV|V}... | me-p{rV|V}...
5	men{c|d|j|z}...	men-{c|d|j|z}...
6	menV...	me-nV... | me-tV
7	meng{g|h|q|k}...	meng-{g|h|q|k}...
8	mengV...	meng-V... | meng-kV...
9	menyV...	meny-sV...
10	mempV...	mem-pV... where V!= 'e'

Table 3. Prefix Removal Rules for Prefix "pe-"

Rule	Construction	Prefix Removal
1	pe{w|y}V...	pe-{w|y}V...
2	perV...	per-V... | pe-rV...
3	perCAP	per-CAP... where C!= 'r' and P!= "er"
4	perCAerV...	per-CAerV... where C!= 'r'
5	pem{b|f|V}...	pem-{b|f|V}...
6	pem{rV|V}...	pe-m{rV|V}... | pe-p{rV|V}...
7	pen{c|d|j|z}...	pen-{c|d|j|z}...
8	penV...	pe-nV... | pe-tV...
9	peng{g|h|q}...	peng-{g|h|q}...
10	pengV...	peng-V... | peng-kV...
11	penyV...	peny-sV...
12	pelV...	pe-lV... except "pelajar", return "ajar"
13	peCerV...	per-erV... where C!={r|w|y|l|m|n}
14	peCP...	pe-CP... where C!={r|w|y|l|m|n} and P!= 'er'
15	terC_1erC_2...	ter-C_1erC_2... where C_1!='r'
16	peC_1erC_2...	pe-C_1erC_2... where C_1!={r|w|y|l|m|n}

Table 4. Prefix Removal Rules for Prefix "be-"

Rule	Construction	Prefix Removal
1	berV...	ber-V... | be-rV...
2	berCAP...	ber-CAP... where C!= 'r' and P!= "er"
3	berCAerV...	ber-CaerV... where C!= 'r'
4	belajar	bel-ajar
5	beC_1erC_2...	be-C_1erC_2... where C_1!={ 'r'|'l'}

Table 5. Prefix Removal Rules for Prefix "te-"

Rule	Construction	Prefix Removal
1	terV...	ter-V... | te-rV...
2	terCerV...	ter-CerV... where C!= 'r'
3	terCP...	ter-CP... where C!= 'r' and P!= 'er'
4	teC_1erC_2...	te-C_1erC_2... where C_1!= 'r'

Table 6. Modified Rules for Table 2

Rule	Construction	Prefix Removal
5	men{c|d|j|s|z}...	men-{c|d|j|s|z}...
6	mengV...	meng-V... | meng-kV... | (mengV-... if V='e')
10	mempA...	mem-pA... where A!= 'e'

Table 7. Modified Rules for Table 3

Rule	Construction	Prefix Removal
9	pengC...	peng-C...
10	pengV...	peng-V... | peng-kV... | (pengV-... if V='e')

3 DOCUMENT CLUSTERING USING ANTS ALGORITHM

3.1 Ants Algorithm

Ants algorithm was developed to solve optimization problem of Traveling Salesman Problem (TSP) for the first time [5]. In the case of Euclidean TSP, d_{ij} is euclidean distance between towns i and j (i.e., $d_{ij} = [(x_i-x_j)^2 + (y_i-y_j)^2]^{1/2}$). Let m be the total number of ants. Each ant is a simple agent with the following characteristics:

- it chooses the town to go to with a probability that is a function of the town distance and the amount of pheromone trail presend on the connecting edge.
- each ant equiped with *tabu list* to force it to make legal tours. It means that transitions to already visited towns are disallowed until a tour is completed.
- when it completes a tour, it lays a substance called *pheromone trail* on each edge(i,j) visited.

Let $\tau_{ij}(t)$ be the amount of pheromone trail on edge (i,j) at time t. Pheromone trail on each edge is updated on each tour, cycle, or iteration according to the following formula:

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

where ρ is a coefficient such that (1- ρ) represents the evaporation of pheromone trail between time t and $t+n$. The range of coefficient ρ is (0,1]. Total amount of pheromone trail laid on edge (i,j) defined as follows:

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

where $\Delta\tau_{ij}^{k}$ is the quantity per unit length of pheromone trail laid on edge (i,j) by the k-th ant between time t and $t+n$; it is given by

$$\Delta\tau_{ij}^{k}\begin{cases} Q/L_k & \text{if } k-\text{th ant uses edge } (i,j) \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}, \quad (5)$$

where Q is a constant and L_k is the tour length of the k-th ant.

Each ant equiped with a data structure called the tabu list $tabu_k$, that saves the towns already visited. After each complete cycle, the tabu list is used to compute the ant's current solution, to get the shortest path route and its length. The transition probability from town i to town j for the k-th ant is defined as follows:

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

where η_{ij} is the *visibility* of $1/d_{ij}$, $allowed_k = \{N\text{-}tabu_k\}$, α and β are parameters that control the relative importance of pheromone trail and visibility where $\alpha \geq 0$ and $\beta \geq 0$.

3.2 Ant Based Document Clustering

Ants algorithm inspired new document clustering method, where the processes is divided into two phase, which are finding the shortest path between the documents (trial phase) and separate a group of documents alike (dividing phase) based on previous trial phase result [6][7][9].

Generally, finding shortest path on the trial phase is adopting ants algorithm, where transition probability from document node i to node j of k-th ant ($p_{ij}^{k}(t)$) is defined as follows:

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

where Z_k represents set of nodes (documents) that is not visited yet by k-th ant, $\tau_{ij}(t)$ represents pheromone trail on edge (i,j), α represents parameter that control the importance of pheromone trail, and β represents visibility parameter. S_{ij} represents cosine distance between node (i,j) which is defined as follows [1]:

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

where w_{ki} and w_{kj} are TF-IDF weighting of k-th term on document i and j. $||d_i||^2$ and $||d_j||^2$ are the length of document vector of document i and j. For example, $||d_i||^2 = (t_1^2 + t_2^2 + t_3^2 + ... + t_k^2)^{1/2}$, where t_k is k-th term of document vector d_i.

On each complete cycle, the amount of pheromone on each edge is updated. Inspired from Lukasz works [6][7][9], we modified the method for laying pheromone on edge i,j ($\Delta\tau_{ij}$), which is defined by the following formula:

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

where $\Delta\tau_{ij}^{k}$ is the amount of pheromone trail laid on edge (i,j) by the k-th ant, which is defined as follows:

$$\Delta\tau_{ij}^{k}\begin{cases} N \times L_k & \text{if } k-\text{th ant uses edge } (i,j) \text{ in its tour} \\ 0 & \text{otherwise} \end{cases}, \quad (10)$$

where N is the total documents and L_k is the total route length of k-th ant. The evaporation of pheromone trail is given by the following formula:

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

where ρ is the pheromone evaporation coefficient.

It should be noted that in this trial phase, the best route (shortest path) is actually represented by the route that give the biggest length (total similarities). It is because of the length of the path between node i and j is represented by cosine distance, which is actually used for calculating documents similarity measurement.

In the next phase, dividing phase, the clusters making or documents grouping is performed according to the documents sequence of the route obtained in the previous trial phase. The first document in the sequence is recognized as centroid μ of the first cluster. The next document in the sequence is called the document D. Next, check the following condition:

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

where δ is the attachment coefficient and its range is (0,1], and $cos(\mu,D)$ is cosine distance of document μ and D.

If the condition is true, then the document D becomes the member of centroid μ. The next document after the current document D in the sequence becomes the next document D. However, if the condition is false, then the cluster making for the first cluster is finished, and the cluster making for the next (second) cluster begins with the current document D becomes the centroid of the new cluster. The whole process is repeated and finished when the whole sequence of documents is done.

3.3 Evaluation

External measure evaluation of recall, precision, and F-measure in information retrieval can be implemented in document clustering area, with analogy of each cluster produced as the retrieval result, and the manual-predefined group (class) of documents as the correct result that should be retrieved [1]. Specifically, for each predefined class i and cluster j, the recall and precision is defined as follows:

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

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

where n_{ij} is the number of documents of class i in cluster j, n_i is the number of documents of class i, and n_j is the number of documents of cluster j. The F-measure of class i on all cluster j is defined as follows:

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

With the overall F-measure is calculated by the following formula:

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

where n is the total number of documents, n_i is the number of documents in class i, and $max\{F_{ij}\}$ is the maximum value F_{ij} of class i on all cluster j.

4 EXPERIMENTS

Our system was developed with Java (JDK 1.6.0) and MySQL database (5.0.45) in Microsoft Windows XP SP2 platform, on a machine with a processor AMD Athlon-64 3200+ (2.2 GHz) and 2GB of RAM. Table 7 shows our news document corpus is consisted of 253 news from www.kompas.com and www.detik.com, spanning from 11 January, 2008, to 4 July, 2008, with 12 manually identified events. We also use stoplist with 792 stopwords and Indonesian language dictionary with 29,337 root-words for document preprocessing purposes. We also calculate the medoid vector of each manual identified news group (v_medoid_i) with the following formula:

$$v\_medoid_i = \{\frac{\sum_{j=1}^{Ni} w_{1j}}{a_{1i}}, \frac{\sum_{j=1}^{Ni} w_{2j}}{a_{2i}}, ..., \frac{\sum_{j=1}^{Ni} w_{kj}}{a_{ki}}\}, \quad (17)$$

where w_{kj} represents the weight of k-th term in document j of i-th news group, a_{ki} represents the number of k-th term on i-th news group, and Ni represents total number of document on i–th news group. The real medoid of i-th news group on Table 7 was achieved by calculating the most similar document with the i-th medoid vector, using cosine distance as similarity measure.

4.1 Trial Phase Experiments

We conducted trial phase on six different preprocessing conditions as follows:

1. Without stoplist removal + without stemmer.
2. Without stoplist removal + CS stemmer.
3. Without stoplist removal + enhanced CS stemmer.
4. With stoplist removal + without stemmer.
5. With stoplist removal + CS stemmer.
6. With stoplist removal + enhanced CS stemmer.

Because of too many parameters to handle and no information about the shortest path and its route for our corpus, we conducted the experiments using parameters setting according to the efficacy of Marco Dorigo's experiments in finding shortest path on traveling salesman problem using ants algorithm [5]. The parameters setting were: $\alpha \in \{0, 0.5, 1, 2, 5\}$, $\beta \in \{0, 0.5, 1, 2, 5\}$, ρ=0.5, and number of ants $\in \{10, 30\}$.

We used 1,000 maximum iteration (cycle) for trial phase on the six different conditions. The results and the best parameters setting is shown in Table 12. From the observations, we've found some points:

1. The more ants we used (30 ants), the less iteration we needed to get the best solution. If we used less ants (10 ants), it will required more iteration to get the best solution. However, the use of more ants will slow down the trial phase time, as shown in Figure 1 and 2.
2. The shortest path always obtained with parameters setting of α=2, β=5, and ρ=0.5.
3. The processing time is also affected by number of terms used in the document vector representation.

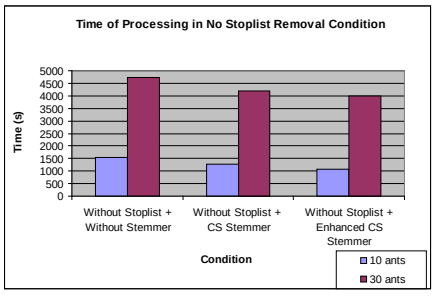

Figure 1. Time of Processing without Stoplist Removal

Table 7. The *Corpus* with Manually Identified Events

ID	Event	Medoid	Count
1	Anwar Ibrahim's sodomy case	Tunangan Saiful Bicara Tentang Kasus Sodomi Anwar Ibrahim	26
2	Artalyta's bribery case	Jadi "Sinterklas", Artalyta Bagi-bagi Makanan di Pengadilan	27
3	Earthquake in China	Hampir 900 Terperangkap dan 107 Tewas di Gempa China	19
4	FPI in Monas incident	Kasus Kekerasan oleh FPI	35
5	Conflict in Israel-Palestine	Menteri Israel Tolak Gencatan Senjata di Gaza	25
6	Two years of Lapindo tragedy	Pengungsi Lapindo Tuntut Jatah Makan Dilanjutkan	21
7	Cyclone Nargis in Myanmar	Yangon Rusak Parah, 351 Orang Tewas	24
8	The election in Indonesia	Jadwal Pemilu Tidak Berubah Cuma Digeser	10
9	Growth of gold price	Berkilaunya Investasi Emas	18
10	Khofifah in East Java Governor Election	Khofifah, Perempuan Lembut Setangguh Lelaki	15
11	Death of Soeharto	Kematian Soeharto Tragedi Bagi Korbannya	28
12	Cannes Movie Festival	The Class Raih Penghargaan Tertinggi Cannes	5

Table 8. Classified Failure Cases of CS Stemmer

Failure Class	Examples		
	Original	Stemmed	Correct
Suffix Removal	**diri**ku	diriku	diri
No prefix removal rule for "mempr-"	mem**protes**	memprotes	protes
No prefix removal rule for "menge-"	menge**muka**kan	mengemukakan	muka
No prefix removal rule for "mens-"	mens**yarat**kan	mensyaratkan	syarat
No prefix removal rule for "penge-"	penge**bom**an	pengeboman	bom
No prefix removal rule for "pengk-"	peng**kaji**an	pengkajian	kaji
Rule precedence	di**kena**i	dikenai	kena
Compound words	di**beritahu**	diberitahu	beritahu
Disallowed prefix-suffix combinations	ke**terlibat**an	keterlibatan	terlibat
Rule 9 of Table 2	me**nyata**kan	menyatakan	nyata
Rule 11 of Tabel 3	pe**nyanyi**	penyanyi	nyanyi
Infixes	**t**em**aram**	temaram	taram
Overstemming	peny**idik**an	sidi	sidik
Understemming	meng**alam**i	alami	alam
Peoples' names	Gumai	Guma	Gumai

Table 9. Classified Failure Cases of Enhanced CS Stemmer

Failure Class	Examples		
	Original	Stemmed	Correct
Infixes	**t**em**aram**	temaram	taram
Overstemming	peny**idik**an	sidi	sidik
Understemming	meng**alam**i	alami	alam
Peoples' names	Gumai	Guma	Gumai
Rule 9 of Table 2	me**nyata**kan	menyatakan	nyata
Rule 11 of Tabel 3	pe**nyanyi**	penyanyi	nyanyi
Compound words	di**beritahu**	diberitahu	beritahu

4.2 Dividing Phase Experiments

We conducted dividing phase experiments by choosing the best result of trial phase on each of six conditions. In this experiments, the cluster making (dividing phase) is attempted by using the documents sequence obtained in the previous trial

phase and F-measure evaluation. The best results on each of the six conditions are showed in Table 13. From the experiments, it can be concluded that if we increase the value of δ (close to 1), the bigger number of clusters we'll received. On the contrary, the decrease of δ value (close to 0) will decrease the number of clusters, as shown in Figure 3.

Table 12. Best Trial Phase on the 6 Conditions

Condition	Trial Phase Parameters				Total Similarities
	Number of ants	α	β	P	
Without Stoplist + without Stemmer	30	2	5	0.5	75.1
Without Stoplist + CS Stemmer	30	2	5	0.5	84.0
Without Stoplist + Enhanced CS Stemmer	30	2	5	0.5	85.9
With Stoplist + without Stemmer	10	2	5	0.5	76.9
With Stoplist + CS Stemmer	30	2	5	0.5	85.3
With Stoplist + Enhanced CS Stemmer	30	2	5	0.5	85.4

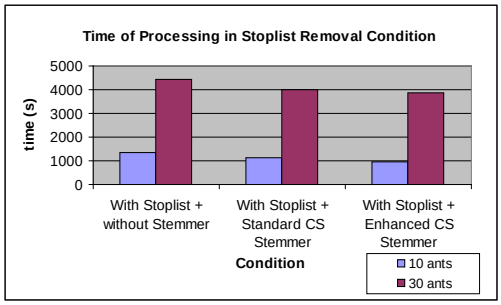

Figure 2. Time of Processing with Stoplist Removal

Table 13. Best Dividing Phase on Best Trial Phase Achieved

Condition	δ	Number of cluster	F-Measure
Without stoplist + without stemmer	0.022	24	0.84
Without Stoplist + CS Stemmer	0.05	29	0.85
Without Stoplist + Enhanced CS Stemmer	0.06	29	0.83
With Stoplist + without stemmer	0.024	24	**0.86**
With Stoplist + CS Stemmer	0.3	26	0.82
With Stoplist + Enhanced CS Stemmer	0.02	21	0.84

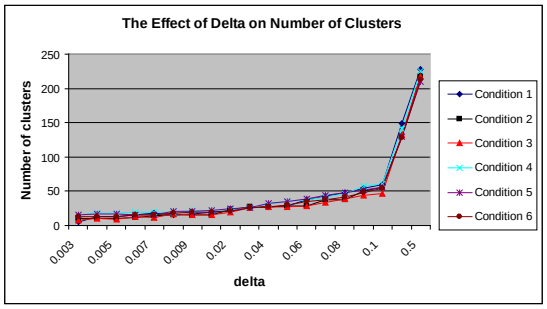

Figure 3. The Effect of δ on Number of Cluster

5 CONCLUSIONS

In this paper, we have proved that ants algorithm can be implemented for classifying news document in Indonesian language, with the best F-measure value achieved from the experiments was 0.86. Also from the experiments result, we've found that the increase of the attachment coefficient (δ), as the threshold value in clusters making, tend to enlarge the number of cluster. On the contrary, the decrease of δ value, tend to reduce the number of clusters.

The preprocessing experiments in this paper showed that confix stripping stemmer had been succesfully implemented and reduce terms size up to 30.95% of the total terms set. Enhanced confix stripping stemmer as an improvement of confix strippin stemmer had been succesfully developed and implemented to solve confix stripping stemmer failures, and able to reduce terms size up to 32.66% of the total terms set.

From the experiments on six different preprocessing conditions, we've showed that although the use of stemming might have been reduce processing time, stemming may reduce the clustering result, because the loss of some terms as significant document features.

6 FUTURE WORKS

We plan to conduct further experiments to get the better parameters setting that could produce better classification result. The fact that the trial phase processing time required lot of time for small data set, and also the high dimensional space of the terms vector, become our next task to solve. Document classification using ants algorithm was an unsupervised classification, which is known as clustering. For better real implementation, we suggest further improvements of ants algorithm in supervised classification area.

Our proposed enhanced confix stripping stemmer can be used as backbone in some applications such as automatic thesaurus, recommender system, information retrieval, and

other system that require high stemming accuracy in Indonesian language. The improvements of our proposed enhanced confix stripping stemmer remains an open problem, especially to solve the problem with infixes (insertions).

REFERENCES

[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) Available at http://www.dcs.gla.ac.uk/Keith/Preface.html

[4] Deitel H.M & Deitel P.J. (2003) Java How To Program Fifth Edition. New Jersey: Prentice Hall.

[5] Dorigo M., Maniezzo V., Colorni 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] Arifin A. Z. & Setiono, Ari Novan. (2002). "Klasifikasi Dokumen Berita Kejadian Berbahasa Indonesia dengan Algoritma Single Pass Clustering". Prosiding Seminar on Intelligent Technology and its Applications (SITIA), Teknik Elektro, Institut Teknologi Sepuluh Nopember Surabaya.

[This page intentionally left blank]