Hybrid Nearest Neighbors Ant Colony Optimization for Clustering Social Media Comments

Lucky Lucky and Abba Suganda Girsang
Computer Science Department, BINUS Graduate Program-Master of Computer Science
Bina Nusantara University, Jakarta, Indonesia 11480
E-mail: lucky@binus.ac.id, agirsang@binus.edu

Keywords: ant colony optimization, nearest neighbors, clustering, text mining, social media comments, twitter

Received: February 8, 2019

Ant colony optimization (ACO) is one of robust algorithms for solving optimization problems, including clustering. However, high and redundant computation is needed to select the proper cluster for each object, especially when the data dimensionality is high, such as social media comments. Reducing the redundant computation may cut the execution time, but it can potentially decrease the quality of clustering. With the basic idea that nearby objects tend to be in the same cluster, the nearest neighbors method can be used to choose the appropriate cluster for some objects efficiently by considering their neighbor's cluster. Therefore, this paper proposes the combination of nearest neighbors and ant colony optimization for clustering (NNACOC) which can reduce the computation time but is still able to retain the quality of clustering. To evaluate its performance, NNACOC was tested using some benchmark datasets and twitter comments. Most of the experiments show that NNACOC outperformed the original ant colony optimization for clustering (ACOC) in quality and execution time. NNACOC also yielded a better result than k-means when clustering the twitter comments.

Povzetek: Predstavljen je hibridni algoritem najbližjih sosedov z optimizacijo mravljinčnih kolonij za grupiranje komentarjev socialnih medijev.

1 Introduction

Nowadays, clustering plays an important role in many applications, such as business intelligence and analytic [1], public health and security [2], as well as the energy saving of internet of things [3], [4]. Clustering also has been implemented in many cases of text mining. According to [5], with the rapid growth of social media usage, petabytes of data had been generated; most of them are in the form of text, blogs, Twitter comments, Facebook feeds, chats, e-mails, and reviews. Therefore, clustering the social media comments has drawn many interests from government to businesses for reading people's opinions quickly and accurately. Some of the examples are detecting public concern about Ebola virus in United States of America using Twitter comments [6], and analyzing the engagement level of three largest pizzas chains with their customers through Facebook and Twitter comments [7].

In the study by [8], most of clustering methods can be considered as optimization problems for finding the most optimal data partitioning based on the objective function. One of the most popular clustering algorithms is k-means which was introduced in 1955 and is still widely used until now [8]. However, according to [9], [10], k-means algorithm sometimes fall into local optima. Therefore, some researchers proposed metaheuristic approach for solving clustering problems such as artificial bee colony (ABC) [11], [12] and ant colony optimization (ACO) [13]–[21].

1.1 Basic concept

1.1.1 ACO algorithm

ACO algorithm was proposed by Dorigo [22] for choosing the shortest path in the Traveling Salesman Problem (TSP). ACO algorithm simulates the behavior of ants when going away from their nest to the source of food and going back to the nest. Those ants use pheromone trail they drop on each travel to communicate and find the shortest path as illustrated in Figure 1.

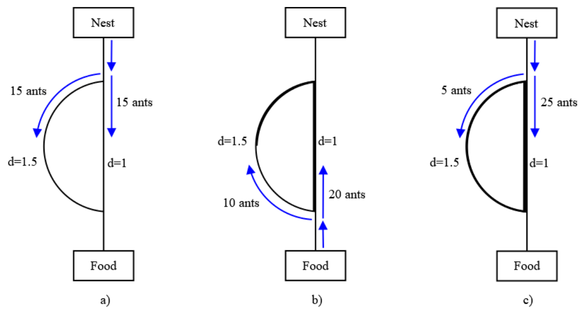

Figure 1: The basic principle of ACO algorithm.

There are two alternative routes between nest and food source; one has a shorter distance equals to 1 (*d=1*) and the other one has a longer distance equals to 1.5 (*d=1.5*). In part *a*, there is no pheromone on all paths. So, the probability that each ant chooses one of the two ways is equal. Part *b* shows the condition when ants travel back from food source to their nest. Since more ants can go faster through the shorter path, the shorter path will have more pheromone than the other one. The pheromone on the longer path also becomes weaker because the evaporation occurs in each tour. Therefore, the shortest path has a bigger chance to be chosen. In part *c*, the number of ants choosing the shorter path increases as the pheromone level on it goes higher while the pheromone level on the longer path goes lower. As the cycle is repeated, all of the ants will eventually choose the shorter path.

The probability used by ants to choose their path is shown in (1).

$$\rho_{ij}^{k} = \begin{cases} \frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{k \in allowed_k}[\tau_{ik}]^{\alpha}.[\eta_{ik}]^{\beta}} & if\ j \in allowed_k \\ 0 & otherwise \end{cases} \quad (1)$$

In (1), τ_{ij} is the level of pheromone between node *i* and *j*. The η_{ij} is the heuristic information between node *i* and *j*. In the TSP case, it is the inverse distance between node *i* and *j*. The α and β are the weight of importance for the pheromone level and heuristic information.

After generating the solution, the pheromone on each edge will be updated to improve the quality of the best solution found using (2).

$$\tau_{ij}(t) = (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad (2)$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & if\ k-th\ ant\ uses\ edge\ (i,j)\ in\ its\ tour \\ 0 & otherwise \end{cases} \quad (3)$$

In (2), ρ is the pheromone evaporation coefficient. Then in (3), $\Delta\tau_{ij}^{k}$ is the pheromone deposited by *k*-th ant when walking through the node *i* to *j*, Q is the pheromone constant, and L_k is the length of tour of the *k*-th ant.

1.1.2 ACO for clustering

ACO for Clustering (ACOC) was firstly introduced by Shelokar [13]. The basic idea of this technique is to represent the solution into a string containing cluster number assigned to each data. Table 1 shows an example of the solution string of a dataset with N = 8 object, K = 3 cluster number, and using M = 3 ants for constructing the solution.

SM	**N**							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
S1	2	2	1	1	3	3	3	3
S2	1	2	2	1	2	3	2	3
S3	1	1	1	2	2	3	3	3

Table 1: Illustration of solution strings in ACOC.

The basic principle of the solution construction in ACOC is almost similar to ACO, except in ACOC there is no heuristic information (η_{ij}) used. There is only pheromone matrix from each object to each cluster for computing the possibility of each ant to choose a cluster that should belong to each object. The rule for choosing cluster number is defined in (4) where *j* is the cluster number in *K* clusters.

$$\rho_{ij}^{k} = \frac{\tau_{ij}}{\sum_{k}^{K}\tau_{ik}}, j = 1 \ldots .K \quad (4)$$

The rule for pheromone evaporation is the same as (2) while the pheromone deposition rule is shown in (5).

$$\Delta\tau_{ij}^{k} = \frac{1}{F^{k}} \quad (5)$$

In (5), F^k is the fitness function of the solution generated by *k*-th ant. To obtain the fitness function, the centroids of each cluster in the generated solution must be calculated using the mean function. After that, the Sum of Squared Error (SSE) of Euclidean distances between each object and the centroid is calculated for measuring the clustering quality. The best solution should have the most minimal SSE of euclidean distances.

The fitness function formula is shown in (6) where *K* is the number of clusters, *N* is the number of objects, and *n* is the number of attributes of an object. The x_{iv} is the v^{th} dimension value of i^{th} object attribute and m_{jv} is the v^{th} dimension value of the centroid for j^{th} cluster. The w_{ij} is the weight that indicates if the i^{th} object belongs to j^{th} cluster. The value is 1 if the i^{th} object belongs to j^{th} cluster or 0 otherwise.

$$F(w,m) = \sum_{j=1}^{K}\sum_{i=1}^{N}\sum_{v=1}^{n} w_{ij}\left\| x_{iv} - m_{jv} \right\| \quad (6)$$

On each iteration, the pheromone matrix between each object and cluster is updated. The bigger the value of the pheromone between an object and certain cluster, the bigger the chance that the object will be assigned to that cluster.

ACOC also implements the elitist ant strategy, which means that only *n*-best ants or solutions will be permitted to deposit the pheromone. The value of *n* is usually 20% from the total number of ants. Besides that, ACOC also uses the local search to improve its generated solution. The local search, which is similar to mutation in Genetic Algorithm, is only performed on 20% of ants with the best fitness value. The process starts from generating N random numbers sequentially where N is the number of objects in a solution. If the generated random numbers were smaller than the pre-determined threshold parameter, the objects in the same sequence as those generated random numbers must change its cluster to a different one. After that, the fitness of the mutated solution will be calculated. If it had a better fitness, then it will replace the current solution. Otherwise, the current solution will be used.

1.2 Related works

According to [23], the ant algorithm for clustering can be divided into two groups, ant-based sorting and ACO based clustering. Ant-based sorting algorithm uses two dimensions grid (x, y) plane. In that algorithm, the objects are scattered randomly at first. After that, the artificial ants will pick the object which is dissimilar to its neighborhood, move it to another location with similar objects, and then drop it. The studies [15], [18]–[20] also used this basic concept in their proposed solution. Although the ant-based sorting does not need a cluster number to be defined at priori, it needs post-processing to identify the generated clusters and requires high processing time [23]. That was proven in some studies where the cluster number should be analyzed visually after the clustering was done [19]. Besides that, the iteration number could reach 15000 for clustering Iris dataset [15].

Another ant algorithm for clustering is ACO based clustering which uses the same concept of solution string to represent the clustering solution as explained in section 1.1.2. The solution string is constructed on each iteration and evaluated by the objective function to find the most optimal one. Although the cluster number must be defined at priori, ACO based clustering is more efficient in computation than ant-based sorting. Also, it does not need post-processing after the clustering is done [23]. Aside of ACO, some of the proposed clustering algorithms also use the same concept of solution string as ACO based clustering [12], [24]–[26].

The first implementation of ACO based clustering is ACOC [13]. After that, ACOC has been improved in some studies such as [14] which modified the original ACOC by keeping the identified best solution as the initial solution for the next iteration and adding the capability to determine the optimal cluster number automatically using Jaccard index. However, the research shows that the algorithm spends more time to run. The research [16] takes a different approach by combining the ACOC with k-means algorithm. K-means is used to generate the initial solution to be explored later by ACO. However, the algorithm is only tested for processing the financial services data. Furthermore, the research [17] also uses the concept of ACO based clustering; however, its focus is for building the classification model based on the training dataset which is clustered using ACO.

The recent research proposes fast ant colony optimization for clustering (FACOC) to improve the efficiency of computation in ACOC [21]. FACOC uses a threshold value to determine if a cluster number became common for an object after it is being chosen for several times. If a cluster number for an object became common, on the next iteration, that cluster number for that object will be simply chosen without computing the probability anymore. This can cut the redundant computations, so that the execution time can be faster. Furthermore, the object with common cluster number will not be affected by local search. However, the result shows that FACOC outputs lower clustering quality than ACOC.

1.3 Problem definition

ACOC uses the probabilistic calculation for choosing the proper cluster for each object based on the strength of the pheromone. That calculation must be done for each object on each iteration. Because of that, the computation in ACOC is high, especially when it is used for clustering large datasets with high dimensionality, such as text and social media comments. Even though the method for reducing the redundant computations has been proposed in FACOC, its performance shows the degradation of the clustering quality. Therefore, the problem that this research tries to solve is how to reduce the computation time of ACOC and retain the clustering quality at the same time.

1.4 The objective and contribution

This paper proposes NNACOC, the hybrid of nearest neighbors and ACOC algorithm which is more efficient than ACOC but still able to retain the clustering quality. To achieve that objective, NNACOC uses the nearest neighbors algorithm to construct the list of nearest neighbors of each object. That list enables the algorithm to assign the same cluster for the current object and its nearest neighbors at the same time. By doing that, the computation for choosing cluster number probabilistically for those nearby objects can be reduced.

The idea that the nearest neighbors can be used for retaining the clustering quality is based on studies which indicate that the good clusters should have the most minimal sum of the euclidean distance between the objects and their cluster's centroid [11]–[13], [21], [27]. Based on that, the objects within the same cluster must be neighboring and near to each other. Therefore, the list of nearest neighbors of each object can be used by the algorithm to assign the appropriate cluster for an object and retain the clustering quality.

The remainder of this paper is organized as follows. Section 2 presents the detail of the proposed algorithm and the description of the datasets for the evaluation. Section 3 discusses the evaluation and its result. Section 4 is about the discussion of the result. Finally, the conclusions and future works are presented in Section 5.

2 Materials and methods

2.1 *N*-nearest neighbors construction

The important part in NNACOC is the list of *n*-nearest neighbors for each object in dataset. The *n* is the pre-determined number of nearest neighbors of the current object which will possibly be assigned with the same cluster number as the current object. When using relatively small dataset, it is still fine to construct the *n*-nearest neighbors for each object one by one using the brute force method. However, when the dataset is large, using the brute force method for constructing the *n*-nearest neighbors list is not feasible because it can be very resource and time consuming in computation.

To overcome the problem, some techniques are introduced for improving the speed of *n*-nearest neighbors construction. One of them is the ball tree algorithm. According to [28], ball tree is an improvement of k-nearest neighbors for faster execution which can be used for handling high dimensional entities. Text clustering usually deals with large number of features vector, which means the dimensionality is high. Thus, the ball tree algorithm is chosen for *n*-nearest neighbors construction in this research.

Before the *n*-nearest neighbors construction can be performed, the text data must be vectorized or transformed into a vector space model. One of the most common vectorizing methods is term frequency - inverse document frequency (TF-IDF). The term importance is based on its occurrence frequencies in a document (TF). Then, it is normalized or reduced by its occurrence frequencies across the document collection (IDF). The TF-IDF method is also used in some of text clustering studies for vectorizing the text document [18], [24]–[26].

2.2 The NNACOC algorithm

The outline of proposed NNACOC algorithm is presented as followsTo make it easier to understand, the pseudocode in Figure 2 is explained in the following step by step illustration.

Step 1. At line 1, the *N* nearest neighbors list for each object is constructed. Where *N* is the number of nearest neighbors which will be assigned to the same cluster automatically when an object is assigned to a cluster.

Let assume that there are 8 objects, *N* = 2, and the constructed *N* nearest neighbors list is shown in Table 2

Step 2. The iteration for constructing solutions is started. Each ant visits all objects one by one and the content of affected objects is checked. If the current object is not in the affected objects list, the cluster for the current object is selected using (4) (line 8 – 10).

```
1.  construct N nearest neighbors list
2.
3.  while termination condition is not met
4.    initialize empty solution string and affected objects
5.
6.    foreach ant in all ants
7.      foreach object in all objects
8.        if object is not in affected objects
9.          assign cluster to current object using eq. (4)
10.         update solution string
11.
12.         calculate the probability to affect neighbors
13.         if should affect neighbors
14.           get N nearest neigbors of current object
15.           foreach neighbor object in N nearest neigbors
16.             assign cluster to neighbor object
17.             update solution string and affected objects
18.
19.     calculate fitness of solution string using eq. (6)
20.
21.   fetch L best solutions from solution string
22.
23.   foreach solution in L best solutions
24.     calculate local search probability
25.     if should do local search
26.       initialize empty new solution
27.
28.       get affected objects in solution
29.       foreach object in affected objects
30.         if object has more than one cluster
31.           choose new cluster randomly then assign to object
32.           update new solution string
33.
34.       if new solution string is not empty and better than ith solution
35.         replace current solution in L best solution with new solution string
36.
37.   evaporate pheromone using eq. (2)
38.
39.   foreach solution in L best solution
40.     deposit pheromone using eq. (5)
41.
42. display the best solution
```

Figure 2: The pseudocode of NNACOC.

Let assume that the affected objects list is empty, the ant is on object A, and then the chosen cluster is 1. So, object A is assigned to cluster 1.

Step 3. After that, the probability calculation is done to decide if the cluster assignment in step 2 should affect the neighbors. It is done by randomizing the float number between 0 and 1. If the result is smaller or equal to pre-determined threshold, then the cluster assignment should affect its neighbors too (line 12 – 17).

Let assume that the cluster assignment should affect the neighbors. Based on the Table 2, C and E are the nearest neighbors of A. Therefore, C and E will be automatically assigned to cluster 1. If ant visit object C or E on the next iteration, then process in line 9 to 17 is skipped. This is how the redundant computations can be reduced.

When visiting each object, there is a possibility that the affected object is assigned to more than one cluster. For example, the ant is on object B then assigns it to cluster 2. The probability calculation indicates that the nearest neighbors should be affected to. Then, based on Table 2, D and E are assigned to cluster 2 as well. Object E has been assigned to cluster 1 previously. So, object E has the possibility to belong to cluster 1 or cluster 2. This condition can be described in Table 3.

Step 4. After each ant finished constructing the solution, some of the best solutions are selected using the same elitist ants strategy as [13] to be processed in local search (line 24 – 35). The same probability calculation as in step 2 is done to decide whether the local search should be done or not. The local search itself is done by randomly selecting other possible clusters to the object that has the possibility to belong to more than one cluster only. If the newly mutated solution has a better fitness than the current solution, it will replace the current solution.

For the example, based on Table 3, object E is currently assigned to cluster 1 because that cluster is chosen at the first time, but it is also possible to belong to cluster 2. To give a better visualization, Figure 3 illustrates the current solution in 2D representation.

The local search process mutated the current solution by replacing the cluster of object E with the different

Object	Nearest Neighbors
A	C, E
B	D, E
C	A
D	B
E	A, B
F	G, H
G	F, H
H	F, G

Table 2: List of the nearest neighbors for each object.

Object	Clusters
C	1
D	2
E	1,2

Table 3: The list affected objects with its possible clusters.

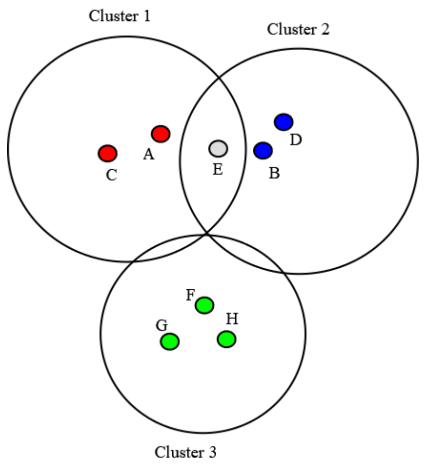

Figure 3: The 2D visualization of clustering solution.

cluster than the current one. In this case, it is cluster 2. If the candidate is more than one, it will be selected randomly. After that, the new solution is evaluated. If it is better than the current solution, then it will replace the current one. Otherwise, the current one is kept.

Step 5. Then, the pheromone evaporation is done to reduce the possibility of bad solution from being chosen (line 37). After that, pheromone deposition is done only for the pre-determined number of the best solutions to increase the possibility of choosing good solution (line 39 – 40).

Step 6. When the termination condition is met, such as reaching certain iteration or maximum execution time, the best solution found is displayed (line 42).

The block diagram of the proposed clustering system can be seen in Figure 4. It contains three main parts, which are dataset collection, nearest neighbors construction, and clustering using NNACOC. The dataset collection part is divided into two parts, one for collecting the benchmark dataset and the other one for collecting Twitter comments dataset which contains data collection process, and data cleansing and pre-processing.

2.3 Fitness function

For measuring the clustering quality on numerical dataset, NNACOC uses the same fitness function as ACOC and FACOC which is defined in (6). However, for measuring clustering quality on social media comments, NNACOC uses the sum of cosine distance instead of SSE of euclidean distance between comments and their cluster centroids. The cosine distance can be defined as 1 – cosine similarity. The equation for cosine similarity is defined in (7), where X and Y are the vectorized texts to be compared, x_i and y_i are the words vector or bag of words from X and Y.

$$cos(X,Y) = \frac{\sum_i x_i . y_i}{\sqrt{\sum_i (x_i)^2} . \sqrt{\sum_i (y_i)^2}} \quad (7)$$

The cosine similarity itself is the most common method for calculating the similarity between texts that

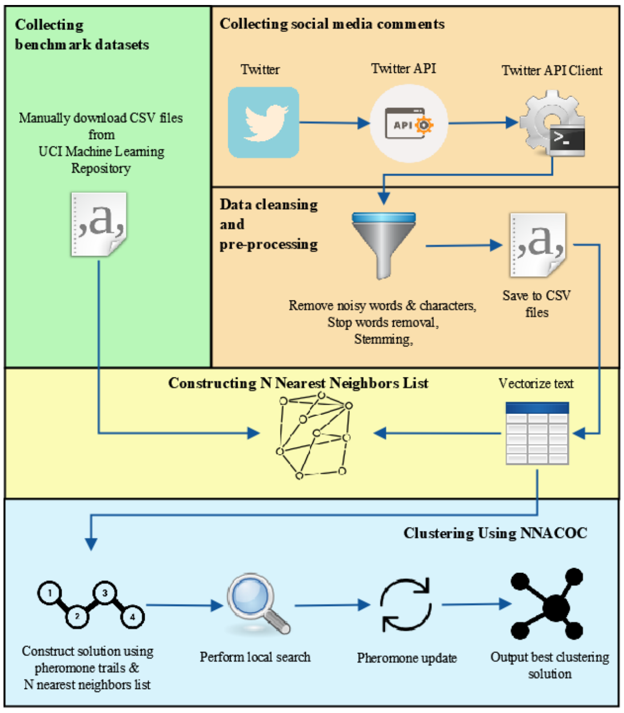
Collecting benchmark datasets
Manually download CSV files from UCI Machine Learning Repository
Collecting social media comments
Twitter
Twitter API
Twitter API Client
Data cleansing and pre-processing
Remove noisy words & characters, Stop words removal, Stemming,
Save to CSV files
Constructing N Nearest Neighbors List
Vectorize text
Clustering Using NNACOC
Construct solution using pheromone trails & N nearest neighbors list
Perform local search
Pheromone update
Output best clustering solution

Figure 4: Block diagram of the proposed system.

has been previously converted into vector-space model. Cosine similarity is also used in some text clustering studies for evaluating the clustering quality by calculating the similarity between the vectorized text and its cluster's centroid [18], [24], [25].

2.4 Dataset collecting

There are two kinds of datasets used in this research. One is for evaluating the standard clustering and the other one is for evaluating the social media comments clustering. Four benchmark datasets for standard clustering are collected from [29] as shown in Table 4.

For text clustering, the Twitter comments in Bahasa Indonesia, which were collected using certain hashtags (#politik, #keuangan, #teknologi, #traveling, #kesehatan, #kuliner, and #olahraga), are populated into four different datasets. Then, because of the noisy characteristics of social media comments, the dataset is populated and pre-processed using the steps illustrated in Figure 4, in the section of collecting social media comments dataset.

The steps in that section can be described as follows.

Step 1. In data collecting section, the Twitter Search API is used by the Twitter API client script for retrieving the comments with certain hashtags.

Step 2. After that, in data cleaning and pre-processing section, the collected comments are cleaned using regular expression from noisy words and characters such as hashtags, mention, URL, emoticons, and repeating characters in a word, as shown in Table 5. Then, the comments are normalized using the program for stop words removal [30] and stemming [31]. Finally, the data is saved into a CSV file.

The specification of Twitter comments datasets is shown in Table 6. For each dataset, the number of hashtags is assumed as the number of clusters. Each of them also has number of attributes, which is the total of feature vectors generated by the TF-IDF algorithm.

3 Evaluation and results

3.1 Evaluation environment

For the evaluation, ACOC and NNACOC are implemented in Python 3 programming language. The k-means algorithm is used as the additional comparison which is also implemented using Python 3. Those algorithms are tested in a laptop with Intel i3 processor, 4 gigabytes RAM, and Arch Linux as the operating system.

3.2 Parameter settings

Some of the parameters used in this research are the same for ACOC and NNACOC, as shown in Table 7, while some are specific for NNACOC only. Others are shown in Table 8.

We use the elitist ant strategy, which according to [13], the ideal value for e is about 20% of m. As explained in section 1.1.2 on the basic concept of ACOC, only the elitist ants (m ants with best solution) are permitted to deposit pheromone. Then, only their solutions will be used in local search. Furthermore, the p_{ls} value is used to control the probability of the mutation in local search process.

Dataset Name	Number of Clusters	Number of Samples	Number of Attributes
Iris	3	150	4
Wine	3	178	13
Breast Cancer (Wisconsin)	2	699	9
Contraceptive Method Choice	3	1473	9

Table 4: Benchmark datasets for clustering.

Regex Pattern	Target
(?:\#+[\w_]+[\w\'_\-]*[\w_]+)	Hashtag
(?:@[\w_]+)	Mention
http[s]?://(?:[a-z]|[0-9]|[$- _@.&amp;+]|[!*\(\),]|(?:%[0-9a-f][0-9a-f]))+	URL
(?:[:=;][oO\-]?[D\)\]\(\]/\\OpP])	Emoticon
([a-z])\1\1+	Repeating characters in a word

Table 5: Regex pattern for cleaning the comments.

Dataset Name	Hashtags in Dataset	Total Tweets	Total Feature Vectors
111 tweets with 3 hashtags	#politik (41 tweets) #teknologi (37 tweets) #olahraga (33 tweets)	111	453
522 tweets with 3 hashtags	#politik (205 tweets) #teknologi (153 tweets) #olahraga (164 tweets)	522	1511
738 tweets with 5 hashtags	#politik (167 tweets) #keuangan (149 tweets) #teknologi (127 tweets) #kesehatan (170 tweets) #olahraga (125 tweets)	738	1877
1013 tweets with 7 hashtags	#politik (167 tweets) #keuangan (149 tweets) #teknologi (127 tweets) #traveling (103 tweets) #kesehatan (170 tweets) #kuliner (172 tweets) #olahraga (125 tweets)	1013	2545

Table 6: Datasets for Twitter comments clustering.

Dataset	**ACOC**		**NNACOC**		
	Avg. Performance	**Avg. Execution Time (s)**	**Avg. Performance**	**Avg. Execution Time (s)**	**Nearest Neighbors Construction Time (s)**
Iris	79.01	11.15	**78.94**	**8.88**	0.31
Wine	2370689.69	14.09	2370689.69	**10.91**	0.24
Breast Cancer Wisconsin	19516.14	42.85	**19444.15**	**32.43**	0.31
Contraceptive Method Choice	36458.68	99.56	**26322.97**	**76.59**	0.33

Table 7: The average of performance and execution time between ACOC and NNACOC on numerical datasets.

The value of *nn* decides how many neighbors can be affected after the cluster number assignment of the current object. We found that 20 is the most optimal value despite the total number of objects. Setting the value of *nn* too low can slow down the speed, while setting it too high can degrade the performance. The parameter *q1* controls the probability if the cluster number assignment to an object also affects its neighbors. The purpose of *q1* is to make the solution construction more dynamic, which means that the cluster number assignment to an object may, but not always, affect its neighbors.

Because of the inherent random behaviour of ACO, the evaluation is done in 10 trials for each dataset and algorithm. In each trial, the maximum iteration number for clustering the numerical datasets is set to 2000. Meanwhile, for the Twitter comments datasets the maximum iteration number is set to 3000 because the number of attributes of the vectorized Twitter comments is much bigger. So, it is assumed that more iteration is needed to achieve the optimal result.

3.3 Results

The result of the average performance and execution time between ACOC and NNACOC for numerical datasets can be seen in Table 9 and the result for Twitter comments datasets can be seen in Table 10. As previously explained, the clustering performance is measured using SSE of euclidean distance for numerical dataset and sum of cosine distance for Twitter comments dataset.

As the additional performance comparison, k-means clustering is also applied to the datasets in Table 9. The result can be seen in the following list.

- Iris: 78.94
- Wine: 2370689.69
- Breast Cancer Wisconsin: 19323.17
- Contraceptive Method Choice: 23691.19

Parameter Name	Value
Number of ants (m)	25
Number of elitist ants (e)	5
Pheromone evaporation rate (ρ)	0.1
Local search probability (p_{ls})	0.01

Table 8: Parameter settings for ACOC and NNACOC.

Parameter Name	Value
Nearest neighbors for each object (nn)	20
Probability to assign nearest neighbors with the same cluster ($q1$)	0.3

Table 9: Parameter settings specific to NNACOC.

Dataset	ACOC		NNACOC		
	Avg. Performance	**Avg. Execution Time (s)**	**Avg. Performance**	**Avg. Execution Time (s)**	**Nearest Neighbors Construction Time (s)**
111 tweets with 3 hashtags	80.35	34.56	**80.22**	**29.59**	0.03
522 tweets with 3 hashtags	407.92	403.72	**407.12**	**317.79**	1.28
738 tweets with 5 hashtags	582.21	638.20	**581.85**	**505.36**	1.45
1013 tweets with 7 hashtags	817.39	1061.17	**813.24**	**826.15**	3.79

Table 10: The average of performance and execution time between ACOC and NNACOC on Twitter comments datasets.

The dataset in Table 10 is also tested using k-means as the additional performance comparison. The result can be seen in the following list.

- 111 tweets with 3 hashtags: 82.8
- 522 tweets with 3 hashtags: 412.67
- 738 tweets with 5 hashtags: 586.69
- 1013 tweets with 7 hashtags: 825.86

3.4 Discussion

The results in the previous section indicate that NNACOC is not only faster than ACOC, but also able to output the equal or even better quality of clustering result than ACOC in most trials. Even though its quality loses to k-means in some numerical data clustering, NNACOC outperforms k-means when clustering the text datasets which have a more complex pattern. The original ACOC also performs better than k-means in text clustering but it loses to NNACOC in all trials.

We also observe how the optimal result is progressing on every iteration, both for ACOC and NNACOC. We display the charts of the iteration progress for each dataset in Figure 5 to Figure 12.

Based on the iterations progress charts for each dataset in Figure 5 to Figure 12, we found that NNACOC always starts with a better solution than ACOC. With that better start, NNACOC has a bigger chance to reach the optimal solution quicker in less iteration than ACOC.

The success key of NNACOC is certainly the nearest neighbors list, which is used in solution construction process. It can reduce the redundant probabilistic calculations by automatically assigning the same cluster to *n*-nearest neighbors of certain objects. Because the objects in the same clusters are usually near to each other, the cluster assignment can be done appropriately, so that the clustering quality can be ensured. The local search process also plays an important role in selecting the most suitable cluster if an object has the possibility to belong to more than one cluster.

However, the automatic cluster assignment should not always happen. The reason is to avoid that the solution falls into local optimum, by giving a chance for the algorithm to explore better possible solutions. So, we set 30% for its probability to happen. The number of how many nearest neighbors can be automatically assigned to a cluster also needs to be set carefully. Too low value can slow down the speed because the redundant probabilistic calculations will be increased. Too high value potentially includes many inappropriate objects which should belong to different cluster. So, it can degrade the clustering quality. We found that 20 is the most optimal number without being affected by the total of objects.

In Table 9 and Table 10, we can also see that the nearest neighbors construction process at the very beginning run of the NNACOC algorithm needs extra time than the ACOC. However, when that extra time is added to the average of NNACOC execution time, the total of

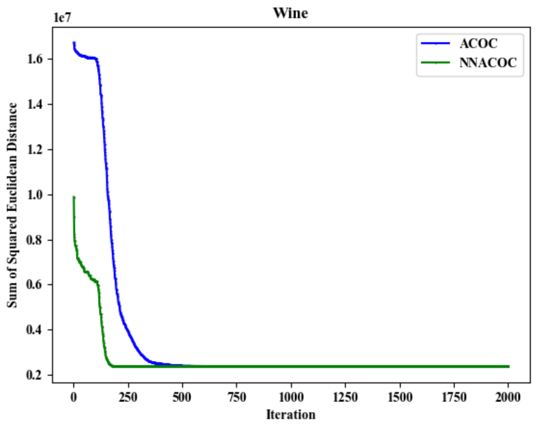

Figure 5: Clustering progress on Wine dataset.

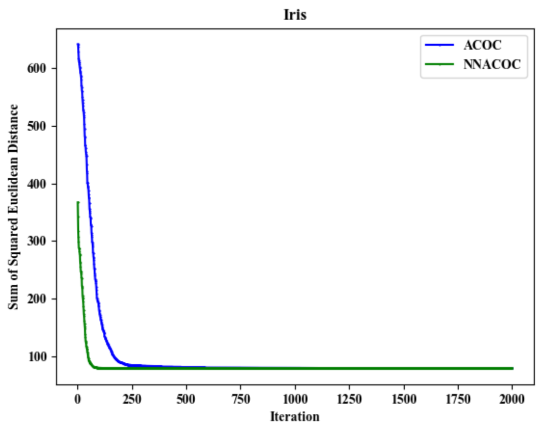

Figure 6: Clustering progress on Iris dataset.

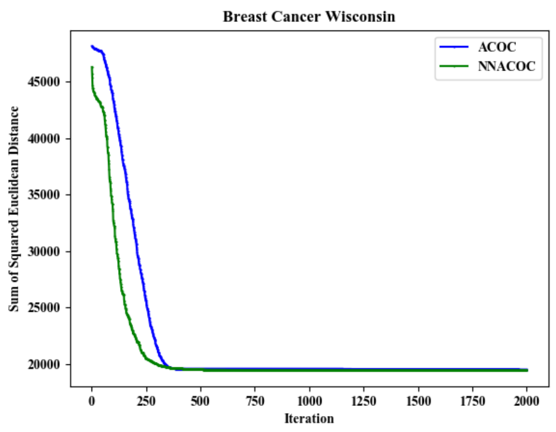

Figure 7: Clustering progress on Breast Cancer Wisconsin dataset.

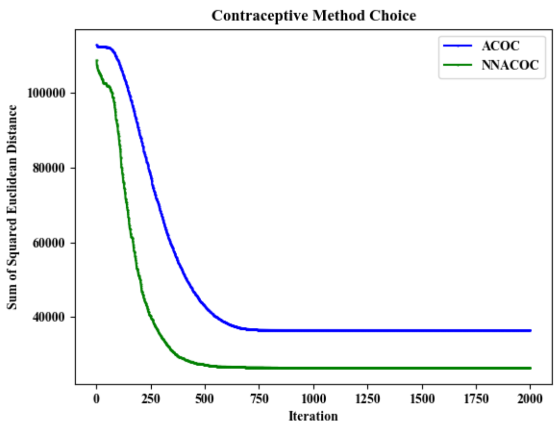

Figure 8: Clustering progress on Contraceptive Method Choice dataset.

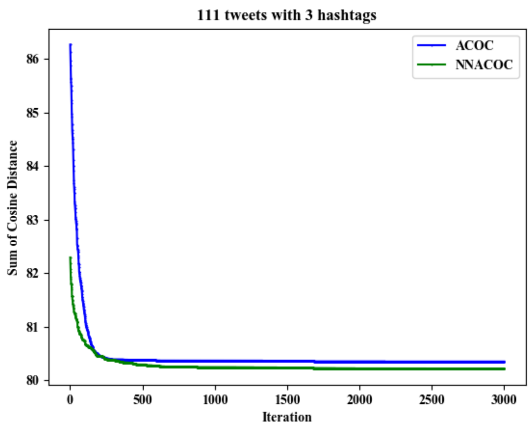

Figure 9: Clustering progress on “111 tweets 3 with hashtags” dataset.

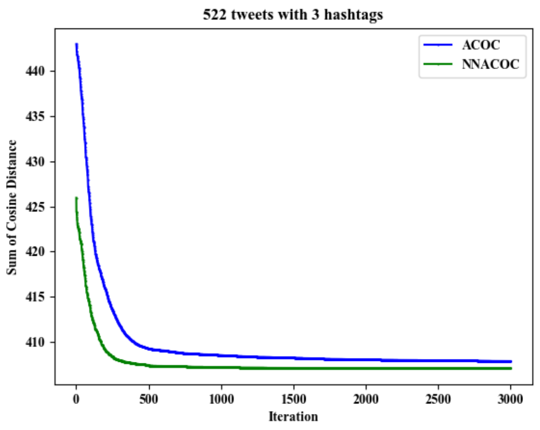

Figure 10: Clustering progress on “522 tweets 3 with hashtags” dataset.

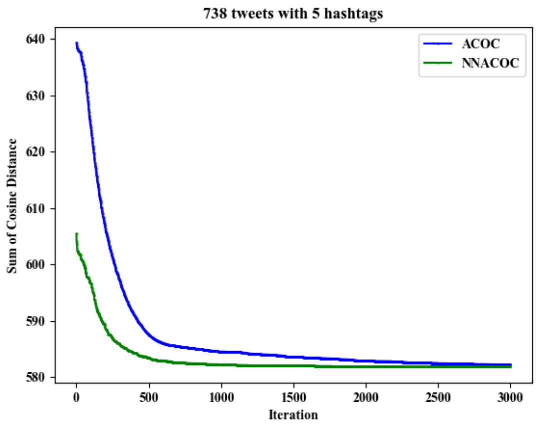

Figure 11: Clustering progress on “738 tweets with 5 hashtags” dataset.

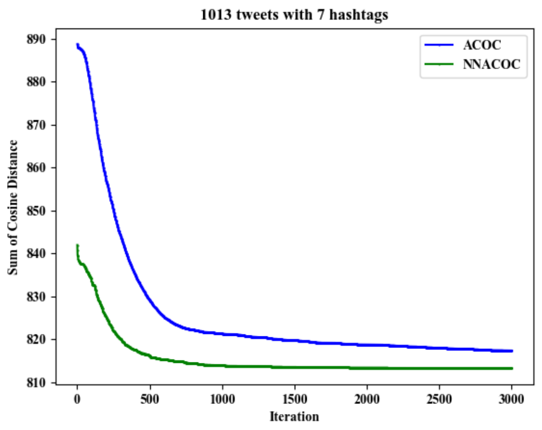

Figure 12: Clustering progress on “1013 tweets with 7 hashtags” dataset.

execution time is still faster than the average execution time of ACOC in all test cases. Moreover, according to our experiments, the extra times needed to construct the nearest neighbors range from 0.03 seconds to 3.79 seconds depending on the size and complexity of the datasets. But, NNACOC can reduce more execution times ranging from 2.62 seconds to 235.02 seconds. The details are shown in Table 11 and Table 12.

To give a better visualization of the execution time comparison between ACOC and NNACOC, we present the following chart in Figure 13 and 14.

Based on the charts, we can see that the difference of total execution time between ACOC and NNACOC becomes bigger when the dataset gets larger. There is also a blue bar at the bottom of each NNACOC chart for showing the nearest neighbors construction time in NNACOC. However, its appearance is almost

Dataset	Nearest Neighbors Construction Time (s)	Reduced Clustering Time (s)
Iris	0.31	2.62
Wine	0.24	3.18
Breast Cancer Wisconsin	0.31	10.42
Contraceptive Method Choice	0.33	22.97

Table 11: The additional time for nearest neighbors. construction versus the reduced clustering time in NNACOC on numerical datasets.

Dataset	Nearest Neighbors Construction Time (s)	Reduced Clustering Time (s)
111 tweets with 3 hashtags	0.03	4.97
522 tweets with 3 hashtags	1.28	85.93
738 tweets with 5 hashtags	1.45	132.84
1013 tweets with 7 hashtags	3.79	235.02

Table 12: The additional time for nearest neighbors construction versus the reduced clustering time in NNACOC on Twitter comments datasets.

unnoticeable. This is a good sign which means that the additional time for the nearest neighbors construction is not significant compared to the reduced execution time.

4 Conclusion

Based on the evaluation result of NNACOC algorithm, it can be concluded that the nearest neighbors algorithm can be used to improve the ACO based algorithm for clustering, especially when clustering the large datasets. With the help of nearest neighbors information for each object, the clustering using ACO can be done faster and more efficient without sacrificing the performance. The evaluation results also show that both ACOC and NNACOC are able to output a better clustering quality than k-means when tested against text dataset. However, even though NNACOC is faster than ACOC and performs better than k-means in text clustering, it still cannot match the speed of k-means in the same case. So, the next challenge for future research is to investigate how to speed up the ACO based clustering algorithm to have the same speed as k-means, or at least close to it, without losing its ability to retain the clustering quality.

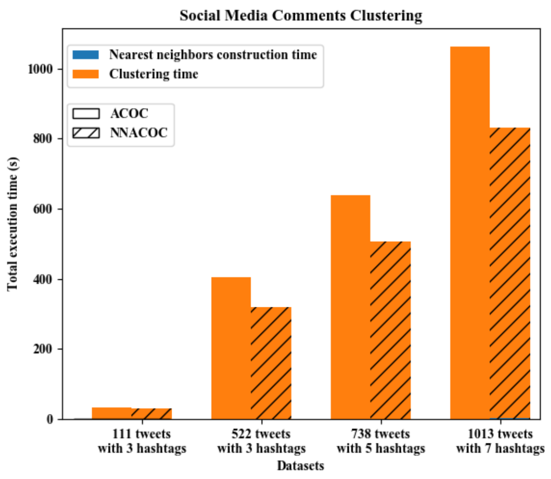

Figure 13: The execution time of ACOC and NNACOC on Twitter comments datasets.

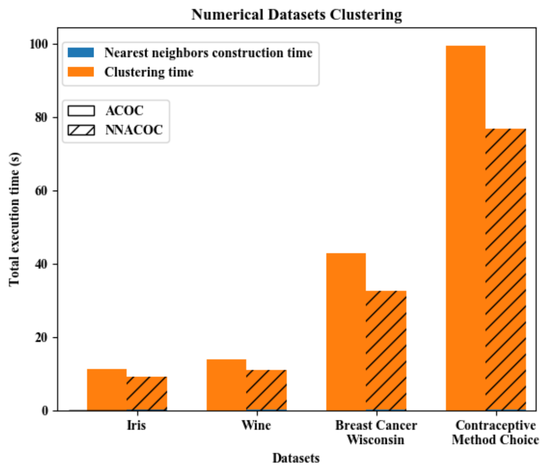

Figure 14: The execution time of ACOC and NNACOC on numerical datasets.

5 References

[1] S. Rajagopal, "Customer data clustering using data mining technique," *International Journal of Database Management Systems*, vol. 3, no. 4, 2011. https://doi.org/10.5121/ijdms.2011.3401

[2] H. Chen, R. H. L. Chiang, and V. C. Storey, "Business intelligence and analytics: from big data to big impact," *MIS Quarterly*, vol. 36, no. 4, pp. 1165–1188, 2012. https://doi.org/10.2307/41703503

[3] H. Chen, Z. Lv, R. Tang, and Y. Tao, "Clustering Energy-Efficient Transmission Protocol for Wireless Sensor Networks Based on Ant Colony Path Optimization," in *2017 International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2017 .https://doi.org/10.1109/cits.2017.8035280

[4] M. Abdelhafidh, M. Fourati, L. C. Fourati, A. Ben Mnaouer, and M. Zid, "Linear WSN Lifetime Maximization for Pipeline Monitoring using Hybrid K-means ACO Clustering Algorithm," in *2018 Wireless Day (WD)*, 2018 .https://doi.org/10.1109/wd.2018.8361715

[5] N. Hardeniya, J. Perkins, D. Chopra, N. Joshi, and M. Iti, *Natural Language Processing: Python and NLTK*. Birmingham: Packt Publishing, 2016.

[6] [6] A. J. Lazard, E. Scheinfeld, J. M. Bernhardt, G. B. Wilcox, and M. Suran, "Detecting themes of public concern: A text mining analysis of the Centers

for Disease Control and Prevention's Ebola live Twitter chat," *American Journal of Infection Control*, 2015 .https://doi.org/10.1016/j.ajic.2015.05.025

[7] W. He, S. Zha, and L. Li, "Social media competitive analysis and text mining: A case study in the pizza industry," *International Journal of Information Management*, vol. 33, pp. 464–472, 2013. https://doi.org/10.1016/j.ijinfomgt.2013.01.001

[8] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010 .https://doi.org/10.1016/j.patrec.2009.09.011

[9] Y. Kao and K. Cheng, "An ACO-Based Clustering Algorithm," in *ANTS 2006: Ant Colony Optimization and Swarm Intelligence*, 2006, pp. 340–347. https://doi.org/10.1007/11839088_31

[10] B. C. Mohan and R. Baskaran, "A survey: Ant Colony Optimization based recent research and implementation on several engineering domain," *Expert Systems with Applications*, vol. 39, pp. 4618–4627, 2012 .https://doi.org/10.1016/j.eswa.2011.09.076

[11] F. Zabihi and B. Nasiri, "A Novel History-driven Artificial Bee Colony Algorithm for Data Clustering," *Applied Soft Computing*, vol. 71, 2018. https://doi.org/10.1016/j.asoc.2018.06.013

[12] A. S. Girsang, Y. Mulyono, and F. Fanny, "Fast Artificial Bee Colony for Clustering," *Informatica*, vol. 42, no. 2, 2018.

[13] P. S. Shelokar, V. K. Jayaraman, and B. D. Kulkarni, "An Ant Colony Approach for Clustering," *Analytica Chimica Acta*, vol. 509, no. 2, pp. 187–195, 2004. https://doi.org/10.1016/j.aca.2003.12.032

[14] X. Liu and H. Fu, "An Effective Clustering Algorithm With Ant Colony," *Journal of Computers*, vol. 5, no. 4, pp. 598–605, 2010. https://doi.org/10.4304/jcp.5.4.598-605

[15] W. Gao, "Improved Ant Colony Clustering Algorithm and Its Performance Study," *Computational Intelligence and Neuroscience*, vol. 2016, no. 19, 2016 .https://doi.org/10.1155/2016/4835932

[16] D. Fogarty, A. George, and S. N. Bhaduri, "New Methods in Ant Colony Optimization Using Multiple Foraging Approach to Increase Stability," in *Advanced Business Analytics*, Singapore: Springer, 2016, pp. 131–138 .https://doi.org/10.1007/978-981-10-0727-9_10

[17] K. M. Salama and A. M. Abdelbar, "Learning cluster-based classification systems with ant colony optimization algorithms," *Swarm Intelligence*, vol. 11, no. 3–4, 2017. https://doi.org/10.1007/s11721-017-0138-5

[18] H. Pang, H. Zhao, W. Li, and Y. Ma, "The Research on the Improved Ant Colony Text Clustering Algorithm," in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, 2017. https://doi.org/10.1109/icbda.2017.8078833

[19] T. M. Pacheco, L. B. Gonçalves, V. Ströele, and S. S. R. F. Soares, "An Ant Colony Optimization for Automatic Data Clustering Problem," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, 2018. https://doi.org/10.1109/cec.2018.8477806

[20] C. Fahy, S. Yang, and M. Gongora, "Ant Colony Stream Clustering: A Fast Density Clustering Algorithm for Dynamic Data Streams," in *IEEE Transactions on Cybernetics*, 2018 .https://doi.org/10.1109/tcyb.2018.2822552

[21] A. S. Girsang, W. T. Cenggoro, and K.-W. Huang, "Fast Ant Colony Optimization for Clustering," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12, no. 1, 2018. https://doi.org/10.11591/ijeecs.v12.i1.pp78-86

[22] M. Dorigo, V. Maniezzo, and A. Colorni, "The Ant System: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, vol. 26, no. 1, pp. 1–13, 1996. https://doi.org/10.1109/3477.484436

[23] A. M. Jabbar, K. R. Ku-Mahamud, and R. Sagban, "Ant-based sorting and ACO-based clustering approaches: A review," in *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 2018 .https://doi.org/10.1109/iscaie.2018.8405473

[24] L. M. Abualigah and A. T. Khader, "Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering," *The Journal of Supercomputing*, vol. 73, no. 11, 2017. https://doi.org/10.1007/s11227-017-2046-2

[25] L. M. Abualigah, A. T. Khader, and E. S. Hanandeh, "Hybrid clustering analysis using improved krill herd algorithm," *Applied Intelligence*, vol. 48, no. 11, 2018. https://doi.org/10.1007/s10489-018-1190-6

[26] N. Kushwaha and M. Pant, "Link based BPSO for feature selection in big data text clustering," *Future Generation Computer Systems*, vol. 82, 2018. https://doi.org/10.1016/j.future.2017.12.005

[27] E. Queiroga, A. Subramanian, and L. dos A. F. Cabral, "Continuous Greedy Randomized Adaptive Search Procedure for data clustering," *Applied Soft Computing*, vol. 72, 2018 .https://doi.org/10.1016/j.asoc.2018.07.031

[28] N. Bhatia and Vandana, "Survey of Nearest Neighbor Techniques," *International Journal of Computer Science and Information Security*, vol. 8, no. 2, pp. 302–305, 2010.

[29] "UCI Machine Learning Repository." [Online]. Available: https://archive.ics.uci.edu/ml/index.php. [Accessed: 30-Sep-2018].

[30] GitHub Inc., "GitHub - masdevid/ID-Stopwords: Stopwords collection of Bahasa Indonesia collected from many sources." [Online]. Available: https://github.com/masdevid/ID-Stopwords. [Accessed: 30-Sep-2018].

[31] GitHub Inc., "GitHub - sastrawi/sastrawi: High quality stemmer library for Indonesian Language (Bahasa)." [Online]. Available: https://github.com/sastrawi/sastrawi. [Accessed: 30-Sep-2018].