JPPI Vol 10 No 2 (2020) 157 - 168

Jurnal Penelitian Pos dan Informatika
32a/E/KPT/2017

e-ISSN 2476-9266
p-ISSN: 2088-9402

Doi:10.17933/jppi.2020.100206

Naive Bayes Classifier Optimization on Sentiment Analysis of Hotel Reviews

Optimasi Naive Bayes Classifier Pada Sentiment Analysis Komentar Pelanggan Hotel

Siti Khomsah

Data Science , Institut Teknologi Telkom Purwokerto;
Jl. D I Panjaitan No.128 Purwokerto, Indonesia 53147
siti@ittelkom-pwt.ac.id

Received: 13 November 2020 ; Received in revised from: 14 December 2020; Accepted: 15 December 2020

Abstract

Feature extraction plays an important role in the sentiment analysis process, especially of text data. The Naive Bayes Classifier performs well on low feature dimensions. However, the accuracy provided is not optimal. To acquire optimal machine learning model, information gain method, evolutionary algorithm, and swarm intelligent algorithm are applied. The objective of this study is to determine the performance of the Particle Swarm Optimization (PSO) to optimize the Naive Bayes Classifier. Vectorization of words is carried out using TF-IDF. In order to produce high PSO performance, the PSO-NBC model is tested with several parameters, namely the number of particles (k = 3), setting of the number of iterations and inertia weight, individual intelligence coefficient (c1 = 1), and social intelligence coefficient (c2 = 2). Inert weight is calculated using the formulation (w = 0.5+ Rand ([- 1,1])). In conclusion, PSO is able to solve the problem space of text-based sentiment analysis. PSO is able to optimize the accuracy of Naive Bayes at a value of 89% to 91.76%. PSO performance is determined by the parameters used, especially the number of particles, the number of iterations, and the weight of inertia. A large number of particles accompanied by an increase in inertia weight can increase accuracy. The number of particles 20-30 has reached the optimal accuracy.

Keywords: *Sentiment Analysis, Optimization, Features-selection, Naive Bayes Classifier, Particle Swarm Optimization*.

INTRODUCTION

Customers who booked hotels online will usually post their reviews after their stay. Online reviews in the forms of ratings and comments can influence potential customers. Customer experience is an important factor because it has an impact on hotel reputation and to increase hotel room bookings (Xie & Zhang, 2014). Analysis of consumer experiences can be obtained by focusing on textual reviews in the form of comments on social media (Xiang et al., 2015). Sentiment analysis is a methodology to determine consumer polarity (sentiment). In big data processing, machine learning works better for sentiment analysis but creating accurate machine learning models is not easy.

NBC is proven to be able to produce good model accuracy even with a small amount of data (Feng et al., 2015; and Rasjid & Setiawan, 2017). NBC considers each word or token in the document (comments) to be mutually independent. This differs from the lexicon-based classification model in general. Accuracy performance can still be optimized with various optimization techniques. There are many ways to optimize machine learning models, including techniques, evolutionary machine learning algorithms, and intelligent swarms (Rizaldy & Santoso, 2017; Pramono et al., 2019; and Osman & Zarog, 2019).

Previous research on the analysis of customer sentiment in Purwokerto hotels by Paramitha (2020) employed two-step feature selection (Elin Hanjani Pramitha, 2020). In this study, feature selection in the Naive Bayes Classifier (NBC) model using TF-IDF is then reduced using PSO. In this study, the accuracy of NBC is optimized using Particle swarm optimization (PSO). In several text analysis studies, PSO was able to optimize accuracy (Pandhu Wijaya & Agus Santoso, 2018; and Wardhani et al., 2018).

PSO was first discovered by Kennedy and Eberhart as a stochastic probability problem optimization technique, which takes advantage of herd behavior (Eberhart & Kennedy, 1995). PSO has advantages, among others, a simple algorithm, able to solve complex problems, and continuously produce better solutions, while a herd algorithm such as Genetics requires complex and expensive computations (Hu et al., 2003). The most optimal final solution depends on the objective function implemented. This objective function can be changed according to certain methods. In this study, the objective function is obtained by multiplying the NBC accuracy for each iteration by an alpha value of 0.9. To obtain the optimal solution, PSO uses several parameters, including the number of particles representing the swarm (a group of data), the position vector of each particle in the swarm, the direction of motion of the particles or velocity, the learning rate, and the learning rate inertia. Each particle will produce a solution and reach convergent under the specified conditions. According to Ratnaweera, convergence et al is influenced by the rate of learning and the inertia weight (Yan et al., 2018). The inertia of the rate and the speed of the learning rate can be determined by a constant value or an adaptation value based on a certain function determined by the programmer.

Based on the review of previous studies, this study optimizes NBC using PSO with random function inertia weigh parameter. The contribution of this study is to find the most optimal NBC accuracy using the parameters of constant learning rate, learning rate inertia with dynamic functions, and changes in the number of particles. A major contribution is in finding the optimal number of particles by applying dynamic inertia weights.

METHODOLOGY

1. Pre-modeling Stage

This stage of study begins with data collection followed by labeling, data selection, and data balancing to obtain ready-to-use dataset. It is followed by building a sentiment analysis model. A sentiment model is built using a combination of PSO and Naive Bayes.

Dataset

The study uses data from previous study on the analysis of customer sentiment in Purwokerto hotels (Elin Hanjani Pramitha, 2020). In previous study, data was collected using web scraping techniques. Scraped data are reviews of customers or hotel users from the agoda.com and tripadvisor sites, especially for hotels in the Purwokerto area of Banyumas Regency. The research data consists of customer reviews of Purwokerto hotels from the agoda.com site, with 900 reviews of 33 hotels, and data on customer reviews of Purwokerto hotels on tripadvisor.co.id, with 1,166 reviews of 39 hotels.

Data Labeling

Data labeling is done automatically and computerized. The hotel customer review data from agoda.com include reviews and service rating of 0 to 10, which is given by consumers. Data labeling is carried out based on the data review score. If the score given by customers is 6.0 or more, the review is labeled positive or 1. If less than 6.0, it is labeled negative or 0.

Meanwhile, hotel customer comments from tripadvisor.com, customers also provide service ratings which range from 10 (Very Bad) to 50 (Excellent). Data from Tripadvisor.com is labeled by sorting positive reviews (1) based on review scores of 40 and 50, while negative reviews (0) are based on review scores of 10, 20, and 30.The labeling results is presented in Table 1.

Table 1. Labeled data

Data source	*Positive (1)*	*Negative(0)*
agoda.com	732	168
tripadvisor.co.id	810	269
Total	1,542	437

Data Selection

Information-rich comments are generally written in a long narrative. Only a few comments were written in short characters. Short comments do not evoke sentiment. The data selection was made to discard reviews with characters below 10.

Examples of comments with less than 10 characters.

- OK
- *Mantappp* (great)
- *Libur asik* (nice vacation)

Data Balancing

Before commencing the vectorization and classification stage, it is better to review the balance of dataset. Balanced data is where the number of datasets labeled positive are the same as the number of data labeled negative. Imbalanced data can affect the accuracy of classification from bias, overfitting, or underfitting even though the accuracy value obtained is high. Methods to manage imbalanced data include increasing the amount of sample data (oversampling), reducing the amount of sample data (undersampling) or a combination of both. Cahyana, Khomsah, and Aribowo (2019) said that oversampling is good for imbalanced data, but sensibility and natural conditions of the data must be considered. To improve accuracy, in addition to oversampling, the undersampling method can be applied (Cahyana et al., 2019). In this study, the undersampling method is used because the number of data labeled positive is three times the number of data labeled negative. Datasets labeled negative are reduced so that they are balanced by the amount of data labeled negative.

2. PSO dan Naive Bayes modeling stage

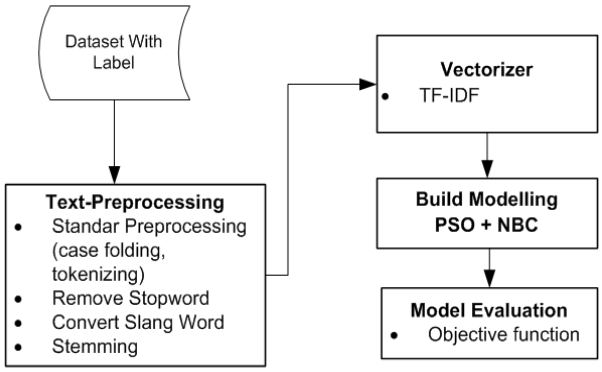

Figure 1. *Proposed Model*

Text Pre-processing

The pre-processing stages in sentiment analysis are generally the same. Before review data is converted into numerical vectors, the text-preprocessing stages are carried out, which include case folding, tokenization, remove stopword, and stemming.

1. Case folding is the process of uniformity of character into lowercase letters. This process is necessary because the same word with different fonts will be considered two different features, this can enlarge the dimensions of the token produced but does not give the meaning of the feature variation. Example case folding include '*Liburan Asik* (Nice Vacation) is changed to ' *liburan asik* (nice vacation)'.
2. Tokenizing is the process of breaking down a sentence (comment) into its constituent single words. For example, '*mantapp banget liburannya* (it was a really great holiday)" will generate three tokens namely '*mantapp I (*great*)*, '*banget* (really)', '*liburannya* (the vacation)'.
3. Stopword removal functions to remove formal words without sentiment. In this study, the stopword removal process uses a built-in function. Every comment in the dataset will be checked automatically (computationally) based on the stopword list that has been compiled (Khomsah & Aribowo, 2020). Examples of stopwords include conjunctions such as '*dan* (and) ', '*ke* (to)', 'dari (from)', *'pada* (in, on, at)', adverbs of time, adverbs of place, and personal pronouns. For example, the comment '*aku menginap pada akhir tahun* (I stayed on yearend', after being subject to a stopword removal, the order of comments changes, '*menginap akhir tahun* (stayed yearend)'.
4. Slangword conversion is the process of converting slang words into standard words. The slangword conversion dictionary uses previous research (Khomsah & Aribowo, 2020). For example, the word '*wuenak* (slang for delicious)' will be converted to '*enak* (delicious)'.
5. Stemming adalah proses mengubah token (kata) menjadi bentuk dasarnya. Misalnya, 'menginap' menghasilkan stem ' inap'. Stemming is the process of changing a token (word) into its basic form. For example, '*menginap* (stayed)' produces the stem '*inap* (stay)'.

Word Vectorizer

Vectorization aims to convert each token in the dataset into a vector value. The method employed is TF-IDF (Salton & Buckley, 1988). TF-IDF is in the form of a word weight value. TF-IDF represents the distribution of each word in a document across the entire document or corpus. The first step is to calculate the TF value with equation (1), calculate the IDF value with equation (2), and calculate the TF-IDF value with equation (3).

$$TF\ (t,d) = 0{,}5 + 0{,}5\ \frac{f\ (t,d)}{\max\{f(w,d):w \in d\}} \quad (1)$$

$$IDF\ (t,d) = log\ \frac{N}{Df(t,d)} \quad (2)$$

$$TF - IDF\ (t,d,D) = tf(t,d)\ x\ idf(t,D) \quad (3)$$

The machine learning model for classifying hotel customer review sentiments in this study is the Naive Bayes Classifier (NBC). Although the previous

study also used NBC as a classifier, it differs from the classifier model in this study. This study optimizes the accuracy of NBC using Particle Swarm Optimization with different parameters.

Building Model

1. Naive Bayes

The Naive bayes model used is the Multinomial Naive Bayes (MNB) type, which is the Naive Bayes type often used for text analysis where data is represented in the form of a word frequency vector, TF-IDF weight (Naive Bayes, n.d.). Distribution is notated with vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ and class *y* , where θ_{yi} is the-*i*-th feature probability $P(x_i \mid y)$ on class *y*. θ_{yi} Parameter is calculated using equation (4)

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \quad (4)$$

where $N_{yi} = \sum_{x \in T} x_i$ is the number of occurences of feature *i* in class *y* and dataset *T*, while $N_y = \sum_{i=1}^{n} N_{yi}$ is the total number of features for class *y*.

2. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is motivated by how birds foraging for food in an area where there is only one source of food. Each bird does not have the information of the right food source. However, every time (iteration) they know the distance of the food source from their places (position). PSO adopts the bird's behavior in looking for food, namely by following the closest bird. A bird states a problem solution in a problem space (Suyanto; et al., 2020).

In general, the term solution is called a particle. Each particle has a fitness value, and it is always evaluated every time (iteration) by the fitness function. Each particle has a speed and direction of flight. The particles (selected solutions) will move (fly) following the optimum particles at that time (close to the source of the solution).

The relationship between particles in the problem space is identified using the neighboring ring or star topology. The neighboring size of the ring topology is equal to three, but sometimes it can be changed to 2,3,4,5 and so on, depending on the width of the problem space dimensions. Star topology has a larger neighbor size because a particle can connect to all particles globally. A particle in PSO consists of three vectors, namely X, P, V and two fitness values, namely fitness X and fitness P.

- X is the current position of the particle in the search space.
- P is the position of the best solution found for the particle and,
- V is the direction of flight of the particle or velocity.
- X-fitness is the vector fitness value X and,
- P-fitness is the vector fitness value P.

To which direction do the particles fly? After a particle obtains a new *Xi*, the particle will evaluate its new position. If *X-fitness* is better *than P-fitness* then *Pi = Xi* and *P-fitness = X-fitness*. This study uses a star topology so as to update the velocity (v) using equation (5)

$$v_{id} = v_{id} + c_1 r[P_{best,id} - x_{id}] + c_2 r[G_{best,id} - x_{id}] \quad (5)$$

Index *i* is the *i*-th particle, and *d* is the *d* dimension. The G_{best} index is used to designate the particle with the best fitness in the neighboring topology used. Meanwhile, the P_{best} index indicates the best fitness

that a particle has achieved so far. Index c_1 refers to the rate of learning components of individual intelligence (cognition), while c_2 is the rate of learning components of the relationship between individuals (social), and r is a random number in the interval [0,1]. The rates of c_1 and c_2 can be fixed at a constant value. This study uses the learning rate c_1 as individual intelligence (cognition). How do particles fly ? Add *v-vector* to *x-vector* to obtain a new *v-vector*, written mathematically as equation (6).

$$x_{id} = x_{id} + v_{id} \quad (6)$$

$$w = 0.5 + \frac{rand(-1,1)}{2} \quad (7)$$

The PSO algorithm begins by generating a random set of particles. This particle size is taken from the dataset. The small particle size will speed up the process of finding a solution. According to Carlisle and Dozier (2001), for a small problem space, optimal PSO performance can be achieved using 10 to 50 particles. As for large problem spaces, 100 particles can be used. However, a small number of particles is often trapped in a local solution, while a large number of particles produces a global solution but the processing time is long (Suyanto; et al., 2020).

The algorithm for this research is binary PSO which is written in Python. Important functions in PSO are shown in Figure 2, Figure 3 and Figure 4.

```
#Generating Particles.
def f_per_particle(m, alpha):
   numpy.ndarray
 #Hitung fungsi objektif
   total_features = X.shape[1]

 # Get the subset of the features from the binary
mask
   if np.count_nonzero(m) == 0:
      X_subset = preprocessing.scale(X)
   else:
      X_subset = preprocessing.scale(X[:,m==1])

 # Perform classification and store performance in P
   Feature_scaler =
MinMaxScaler(feature_range=(0.01, 0.99))
   X_subset =
Feature_scaler.fit_transform(np.array(X_subset))
   mnb.fit(X_subset, y)
   P = (mnb.predict(X_subset) == y).mean()
   particleScore.append(P)
   particleSize.append(X_subset.shape[1])

 # Objective Function
   j = (alpha * (1.0 - P)
      + (1.0 - alpha) * (1 - (X_subset.shape[1] /
total_features)))
   return j
```

Figure 2. Objective Function Algorithm

In Figure 2, the accuracy of each particle is calculated using Naive Bayes and stored in *P*. The objective function is to calculate the optimal solution for each particle. The value of the objective function *j* will continue to change until it reaches a convergent value, that is, when the *j* value of the particle in the next iteration (generation) does not change (constant). If the convergent value is not exceeded, but the specified iteration is complete, the optimization process will stop.

```
# Function for generating a bunch of particles
in each iteration
def f(x, alpha=0.88):
    x: numpy.ndarray of shape (n_particles,
dimensions)
    n_particles = x.shape[0]
    j = [f_per_particle(x[i], alpha) for i in
range(n_particles)]
    return np.array(j)
#m is array (particle dimension).
#alpha is constant weight for trade-off classifier
performance and number of features.
```

Figure 3. Particle Generating Function

```
#Main Function PSO
import matplotlib.pyplot as plt
import random
w=0.5+random.uniform(-1,1)/2
jumpartikel=10
iters=100

 # Initial parameters PSO
options = {'c1': , 'c2': , 'w':, 'k':, 'p':}
#c1 = individual intelligence learning rate
#c2 = Social intelligence learning rate
#w = inertia weight
#k = number of neighbors
#p = distance calculation (2=euclidean)

particleScore = list()
particleSize = list()

# sepecify the dimensions of features ( features
of word that produced by stemming)

dimensions = X.shape[1]

# Create Object PSO named optimizer

optimizer =
ps.discrete.BinaryPSO(n_particles=jumpartikel,
dimensions=dimensions, options=options)

 # Performance  optimizer
cost, pos = optimizer.optimize(f, iters=iters)
```

Figure 4. Main Function

Model Evaluation

The accuracy of the Naive Bayes classification is viewed from data comparison predicted correctly by the MNB to the overall data in particles (Figure 2). Meanwhile, the optimized performance of Naive Bayes is shown by the variable *j*. The accuracy of the Naive Bayes Classifier which is optimized by PSO is indicated by the final value of the objective function.

RESULTS AND DISCUSSION

The analysis sentiment classification model used is the Multinomial Naive Bayes Classifier (NBC) algorithm. For the first step, each token (term) is converted into a numeric vector using TF-IDF. From the TF-IDF calculation, the dimensions are 2,994 features. After obtaining the features with TF-ID, the classification process is continued with Multinomial NBC. Each NBC classification result is optimized with Particle Swarm Optimization (PSO). The PSO parameters used in the experiment are shown in Table 2.

Table 2 Experiment Parameter Setting

Parameter	***Value***
c1 = *individual intelligence learning rate*	1
c2 = *social intelligence learning rate*	2
w = *inertia weight*	equation (7)
k = *number of neighbors*	3
p = *distance calculation option*	2 (Euclidean)
number of particles	10; 20; 30; 40; 50
number of iterations	100 dan 500

The components c1 and c2 are given constant values. Small constant value of under one will speed up the rate but sometimes decrease accuracy. The small number of *k* is expected to produce a good neighborhood relationship. Large amounts of *k* will usually slow down the process. The number of particles was tested using 10 to 50 particles, and each with 100 and 500 iterations. If 10 particles are generated, there will be 10 generations, namely P1, P2, P3, ..., P10. NBC make 10 times classification, then average the accuracy. One P loop is called a generation.

To prove that the PSO is working well for NBC, two experiments are carried out. First, the NBC model is built with the TF-IDF feature selection. The

second model uses NBC in combination with PSO. In the second model, the NBC method works on the particles in the PSO. In the first model experiment, all features and comment data are used. Seventy percent (70%) serve as training data to build models and thirty percent (30%) becomes testing data. The accuracy of the PSO + NBC model is shown in Table 3.

Table 3 NBC +TF-IDF model performance

Model	**Accuracy**
NBC+ TF-IDF	76%

In the NBC optimization model, the experiment is repeated 10 times (10 variations of particles). The results for each particle and iteration are shown in Table 4.

Table 4 PSO-NBC Model Performance

Number of iterations	Number of particles	Accuracy PSO-NBC (%)	Inertia weight(w)
100	10	89.93	0.399
	20	90.38	0.345
	30	90.73	0.414
	40	90.80	0.752
	50	91.76	0.980
500	10	90.50	0.679
	20	90.50	0.217
	30	91.18	0.535
	40	90.50	0.130
	50	90.80	0.396

In an experiment with 100 iterations (Table 4), the accuracy in the initial generation of particle 10, the accuracy has reached 89.93%. This figure actually exceeds the accuracy of NBC without PSO. With the increase in the number of particles from 20, 30, to 40, convergent accuracy is achieved at the value of 90%. The increase in accuracy occurs when 50 particles are used. In the inertia weight column, it can be seen that the greater the weight value, the greater the accuracy.

Meanwhile, the experiment with 500 iterations yields different result. Early generation accuracy, with 10 convergence particles is at 90.50%. However, in contrast to the first experiment (with 100 iterations), the increase in the number of particles is not accompanied by an increase in accuracy. Inertia weight is reduced when the number of iterations is 500. With 500 iterations, the higher the number of particles, the smaller the weight change. However, when using 30 particles, it appears that the accuracy increases with inertia weight better than that with 20 particles. This may happen because of the way PSO generates particles randomly. The random generation of particles does not guarantee that the selected data is good to support optimal accuracy.

The PSO performance rates for NBC optimization are shown in Figure 5, Figure 6, Figure 7, Figure 8. The red line in Figure 5,6,7,8 shows the peak mean accuracy of all particles in each generation. Figure 5 shows that the use of 10 particles will achieve the first accuracy at about the 50^{th} iteration but then decrease and reach the highest average accuracy at the 99^{th} iteration.

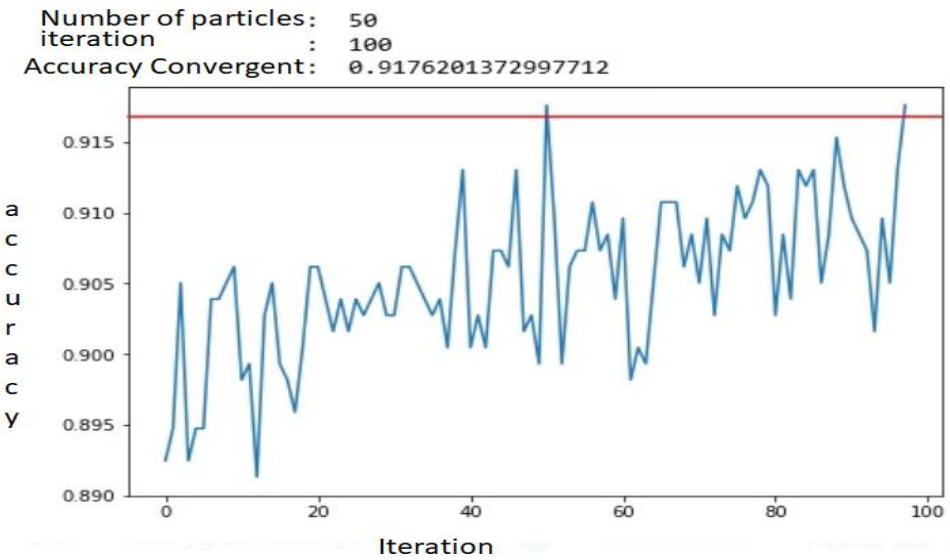

Figure 5. Accuracy Rate With 10 Particles and 100 Iterations

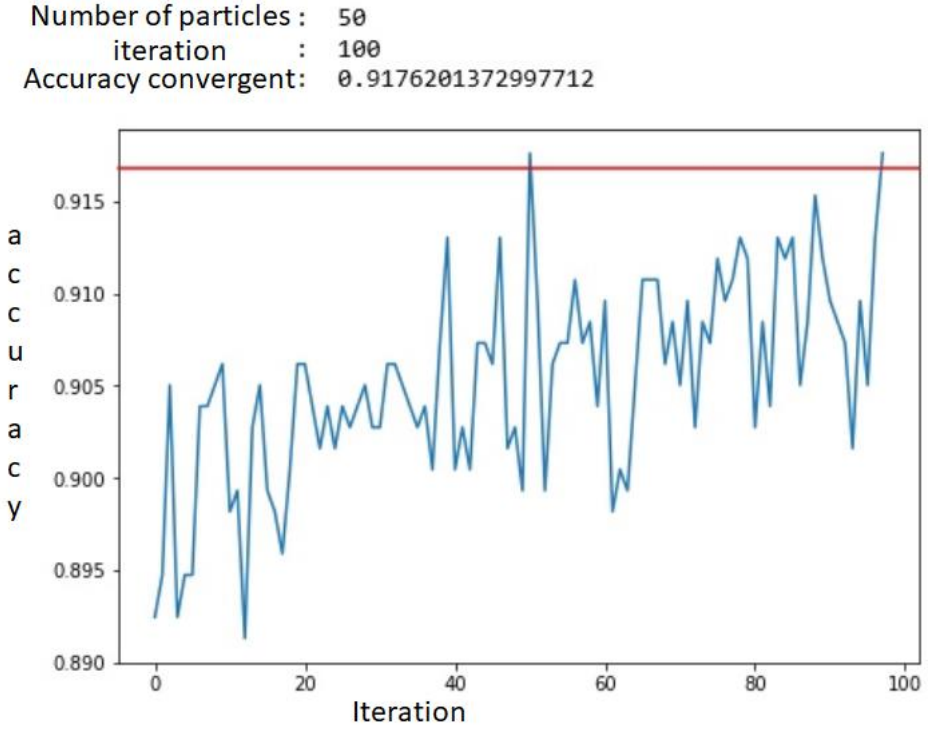

Figure 6. Accuracy Rate With 50 Particles and 100 Iterations

Figure 6 shows the use of 50 particles under the similar conditions as that with 10 particles. In Figure 5 and Figure 6, the convergence of accuracy did not occur consistently despite increase of accuracy.

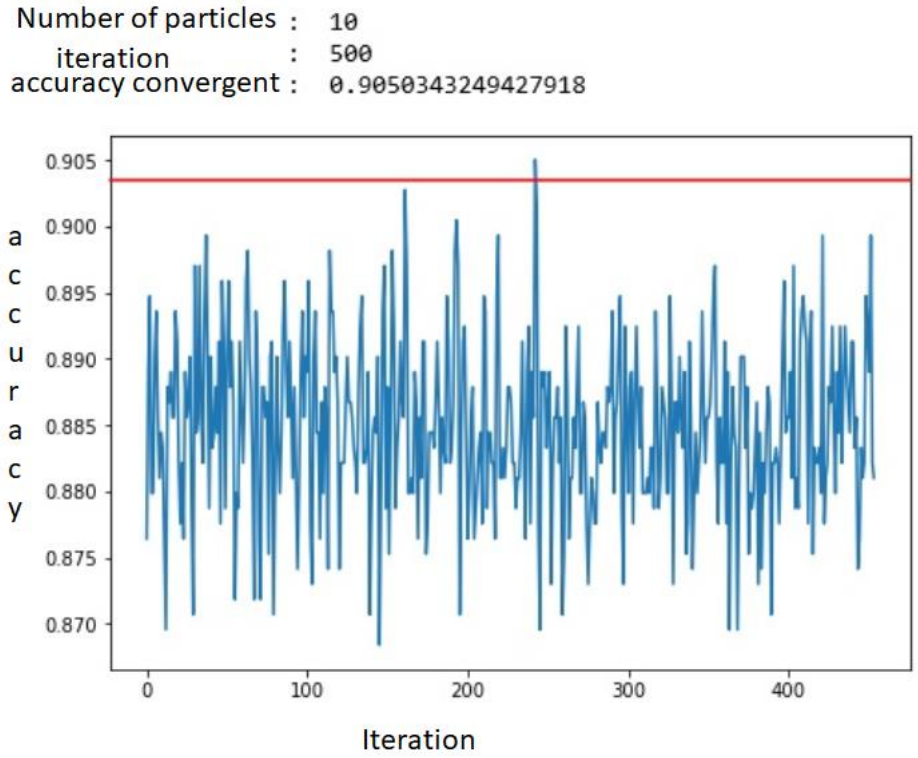

Figure 7. Accuracy Rate With 10 Particles and 500 Iterations

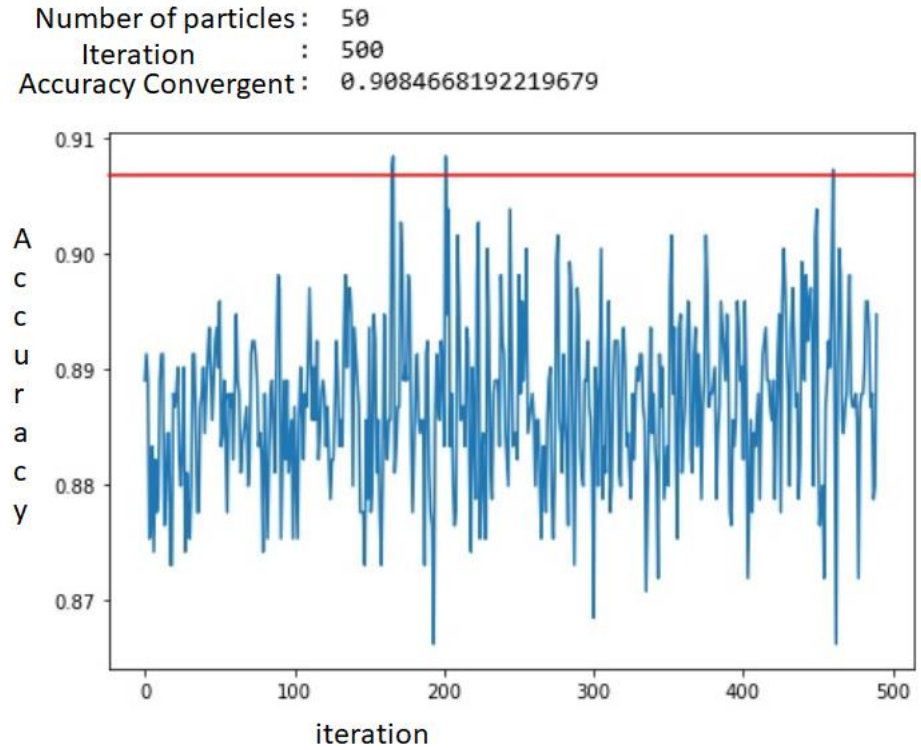

Figure 8. Accuracy Rate With 50 Particles and 500 Iterations

Figure 6 and Figure 7 show the rate of change in the accuracy of each particle. In the early days of flying particles (the first generation), accuracy was seen at 90%. Although the number of particles increases, accuracy does not increase. Usually, a high number of iterations requires high computation time. However, in this study, the computation time is not calculated because the computation time is very relative, depending on the hardware used.

CONCLUSIONS

1. Conclusion

From the experiments and datasets carried out and used in this study, it is proven that Particle Swarm Optimization has been successful in optimizing the Naive Bayes Classifier, achieving the smallest accuracy of 89.93% and the maximum accuracy of 91.76%.

Text analysis yields a large feature dimension. The increase in accuracy of Naive Bayes from 76% to 89.93-91.76% indicates that PSO can solve the problem space of text analysis with certain parameters. The parameter setting significantly affects the convergence rate of accuracy. The use of a smaller number of particles can achieve optimal accuracy (89.93%). Larger number of particles can generate more optimal accuracy, with the highest at 91.76%. However, the combination of the number of iterations and the number of particles is equally large, unable to increase accuracy. The right inertia weight can improve accuracy. The number of particles 20-30 has reached optimal accuracy.

2. Recommendation

It is necessary that PSO performance for feature optimization in sentiment analysis be tested on various types of datasets, across domains, and various parameters. In the sentiment analysis problem space, the selection of the right number of particles, the number of iterations, and the value of the inertia weight component still require further investigation.

ACKNOWLEDGEMENT

The research team would like to thank colleagues at the Infomatics Engineering Study Program of the Purwokerto Telkom Institute of Technology. The research team would like to thank Mr. Agus Sasmito Aribowo, S.Kom., M.Cs as a researcher in the field of sentiment analysis, who has provided an initial criticism and review of the writing of the results of this study, and Elin Hanjani Paramitha who has been willing to share the dataset to complete this study.

REFERENCES

Cahyana, N., Khomsah, S., & Aribowo, A. S. (2019). Improving Imbalanced Dataset Classification Using Oversampling and Gradient Boosting. *Proceeding - 2019 5th International Conference on Science in Information Technology: Embracing Industry 4.0: Towards Innovation in Cyber Physical System, ICSITech 2019*, 217–222. https://doi.org/10.1109/ICSITech46713.2019.8987499

Eberhart, R., & Kennedy, J. (1995). New optimizer using particle swarm theory. *Proceedings of the International Symposium on Micro Machine and Human Science*, 39–43. https://doi.org/10.1109/mhs.1995.494215

Elin Hanjani Pramitha. (2020). *Sentiment Analysis Komentar Pelanggan Hotel Di Purwokerto Menggunakan Naive Bayes Classifier*.

Feng, G., Guo, J., Jing, B. Y., & Sun, T. (2015). Feature subset selection using naive Bayes for text classification. *Pattern Recognition Letters*, *65*, 109–115. https://doi.org/10.1016/j.patrec.2015.07.028

Hu, X., Eberhart, R. C., & Shi, Y. (2003). Engineering optimization with particle swarm. *2003 IEEE Swarm Intelligence Symposium, SIS 2003 - Proceedings*, 53–57. https://doi.org/10.1109/SIS.2003.1202247

Khomsah, S., & Aribowo, A. S. (2020). Model Text-Preprocessing Komentar Youtube Dalam Bahasa Indonesia. *Rekayasa Sistem Dan Teknologi Informasi*, *RESTI*, *4*(10), 648–654. https://doi.org/10.29207/resti.v4i4.2035

Naive Bayes. (n.d.). https://scikit-learn.org/stable/modules/naive_bayes.html

Osman, S. E., & Zarog, M. (2019). Optimized V-Shaped Beam Micro-Electrothermal Actuator Using Particle Swarm Optimization (PSO) Technique. *Micro and Nanosystems*, *11*(1), 62–67.

https://doi.org/10.2174/1876402911666190208162346

Pandhu Wijaya, A., & Agus Santoso, H. (2018). Improving the Accuracy of Naïve Bayes Algorithm for Hoax Classification Using Particle Swarm Optimization. *Proceedings - 2018 International Seminar on Application for Technology of Information and Communication: Creative Technology for Human Life, ISemantic 2018*, 482–487. https://doi.org/10.1109/ISEMANTIC.2018.8549700

Pramono, F., Didi Rosiyadi, & Windu Gata. (2019). Integrasi N-gram, Information Gain, Particle Swarm Optimation di Naïve Bayes untuk Optimasi Sentimen Google Classroom. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *3*(3), 383–388. https://doi.org/10.29207/resti.v3i3.1119

Rasjid, Z. E., & Setiawan, R. (2017). Performance Comparison and Optimization of Text Document Classification using k-NN and Naïve Bayes Classification Techniques. *Procedia Computer Science*, *116*, 107–112. https://doi.org/10.1016/j.procs.2017.10.017

Rizaldy, A., & Santoso, H. A. (2017). Performance improvement of support vector machine (SVM) With information gain on categorization of Indonesian news documents. *Proceedings - 2017 International Seminar on Application for Technology of Information and Communication: Empowering Technology for a Better Human Life, ISemantic 2017*, *2018-January*, 227–231. https://doi.org/10.1109/ISEMANTIC.2017.8251874

Salton, G., & Buckley, C. (1988). Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, *24*(5), 513–523. https://doi.org/https://doi.org/10.1016/0306-4573(88)90021-0

Suyanto;, Arifianto, A., Rismala, R., & Sunyoto, A. (2020). *Evolutionary Machine Learning* (Edisi 1). Informatika.

Wardhani, N. K., Rezkiani, Kurniawan, S., Setiawan, H., Gata, G., Tohari, S., Gata, W., & Wahyudi, M. (2018). Sentiment analysis article news coordinator minister of maritime affairs using algorithm naive bayes and support vector machine with particle swarm optimization. *Journal of Theoretical and Applied Information Technology*, *96*(24), 8365–8378.

Xiang, Z., Schwartz, Z., Gerdes, J. H., & Uysal, M. (2015). What can big data and text analytics tell us about hotel guest experience and satisfaction? *International Journal of Hospitality Management*, *44*, 120–130. https://doi.org/10.1016/j.ijhm.2014.10.013

Xie, K., & Zhang, J. (2014). The Business Value of Online Consumer Reviews and Management Response to Hotel Performance. *International Journal of Hospitality Management*, *43*(October 2017), 1–12. https://doi.org/10.1016/j.ijhm.2014.07.007

Yan, Y., Zhang, R., Wang, J., & Li, J. (2018). Modified PSO algorithms with “Request and Reset” for leak source localization using multiple robots. *Neurocomputing*, *292*, 82–90. https://doi.org/10.1016/j.neucom.2018.02.078