

# Automated Text Annotation Using Semi-Supervised Approach with Meta Vectorizer and Machine Learning Algorithms for Hate Speech Detection

[Shoffan Saifullah](#)<sup>\*</sup>, [Rafał Dreżewski](#), Felix Andika Dwiyanto, Agus Sasmito Aribowo, Yuli Fauziah, Nur Heri Cahyana

Posted Date: 15 November 2023

doi: 10.20944/preprints202311.0963.v1

Keywords: Hate Speech Detection; Machine Learning; Sentiment Analysis; Semi-Supervised Learning; Self-Learning; Text Mining






Preprints.org is a free multidiscipline platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

## Article

# Automated Text Annotation Using Semi-Supervised Approach with Meta Vectorizer and Machine Learning Algorithms for Hate Speech Detection

Shoffan Saifullah <sup>1,2,\*</sup> , Rafał Dreżewski <sup>1,3</sup> , Felix Andika Dwiyanto <sup>1,4</sup> , Agus Sasmito Aribowo <sup>2</sup>, Yuli Fauziah <sup>2</sup> and Nur Heri Cahyana <sup>2</sup>

<sup>1</sup> Faculty of Computer Science, AGH University of Krakow, Krakow, Poland; saifulla@agh.edu.pl (S.S.), drezew@agh.edu.pl (R.D.), dwiyanto@agh.edu.pl (F.A.D.)

<sup>2</sup> Department of Informatics, Universitas Pembangunan Nasional Veteran Yogyakarta, Yogyakarta, Indonesia; shoffans@upnyk.ac.id (S.S.), sasmito.skom@upnyk.ac.id (A.S.A.), yuli.fauziah@upnyk.ac.id (Y.F.), nur.hericahtyana@upnyk.ac.id (N.H.C.)

<sup>3</sup> Artificial Intelligence Research Group (AIRG), Informatics Department, Faculty of Industrial Technology, Universitas Ahmad Dahlan, Yogyakarta, Indonesia

<sup>4</sup> Department of Electrical Engineering, Universitas Negeri Malang, Malang, Indonesia

\* Correspondence: shoffans@upnyk.ac.id

**Abstract:** Text annotation is an essential element of the natural language processing approaches. The manual annotation process performed by humans has several drawbacks, such as subjectivity, slowness, fatigue, and possibly carelessness. In addition, annotators may annotate ambiguous data. So, we developed the concept of automated annotation to get the best annotations using several machine-learning approaches. The proposed approach is based on an ensemble algorithm of meta-learners and meta-vectorizer techniques. The approach employs a semi-supervised learning technique for automated annotation, aimed at detecting hate speech. This involves leveraging various machine learning algorithms, including Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbors (KNN), and Naive Bayes (NB), in conjunction with Word2Vec and TF-IDF text extraction methods. The annotation process is performed using 13,169 Indonesian YouTube comments data. The proposed model used a Stemming approach using data from Sastrawi and also new data of 2,245 words. Semi-supervised learning uses 5%, 10%, and 20% of labeled data as compared to performing labeling based on 80% of the datasets. In semi-supervised learning, the model learns from the labeled data, which provides explicit information, and the unlabeled data, which offers implicit insights. This hybrid approach enables the model to generalize and make informed predictions even when limited labeled data is available, ultimately enhancing its ability to handle real-world scenarios with scarce annotated information. In addition, the proposed method uses a variety of thresholds for matching words labeled with hate speech ranging from 0.6, 0.7, 0.8, and 0.9. The experiments indicated that the KNN-Word2ec model has the best accuracy value of 96.9% with a scenario of 5%:80%:0.9. However, several other methods have also accuracy above 90%, such as SVM and DT based on both text extraction methods in several test scenarios.

**Keywords:** hate speech detection; machine learning; sentiment analysis; semi-supervised learning; self-Learning; text mining

## 1. Introduction

Hate speech detection is based on the concept of Natural Language Processing (NLP) and sentiment analysis [1,2]. NLP uses the concept of text mining to perform processing of the specific text [3]. The process requires several stages, one of which is text annotation, so the training process requires data that has been labeled. Text annotation done manually by humans has several disadvantages, such as being expensive, prone to errors, inconsistent labeling, subjectivity, and difficulties in processing large datasets. These factors occur because the person doing the annotation

has several shortcomings, such as fatigue, pressure of circumstances and environment, the influence of workload, and the subjectivity. In addition, if several people do the annotation, they must equalize the perception (understanding of words and mindset) when annotating. This process affects the consistency of the annotated labels. These problems can be solved by applying the concept of automatic text annotation using a machine-learning approach. Thus, we developed an automatic annotation model using several machine learning methods such as SVM, DT, KNN, and NB for hate speech detection based on data from social media (YouTube). The data used can impact the learning process and classification of statements as hate speech or not.

Automatic text annotations can detect hate speech by applying machine learning methods with a semi-supervised learning approach [4,5]. Hate speech data are annotated using two categories (hate and not hate) [6–10], and using sentiment analysis methods, data are labeled using 2 or 3 categories, namely (positive and negative) [11,12], or (positive, negative, and neutral) [13–16]. We develop automatic annotations using a dataset with minimal training data that is labeled, and then generate self-learning for the labels. We refer to AraSenCorpus modeling [17], which automates the concept of a semi-supervised framework (long short-term memory with FastText, and machine learning approach) to analyze sentiment with two and three Arabic language classifications. In addition, annotations can also be used for sentiment and emotion detection by applying several methods such as SVM, Random Forest (RF), and NB [17]. Several other methods (such as NB, convolutional neural networks (CNN), Bi-LSTM, and SVM) can be implemented in annotating Turkish text [18].

This research aims to develop the concept of automated text annotations that apply semi-supervised learning and self-learning. The most important original contributions of this research are as follows:

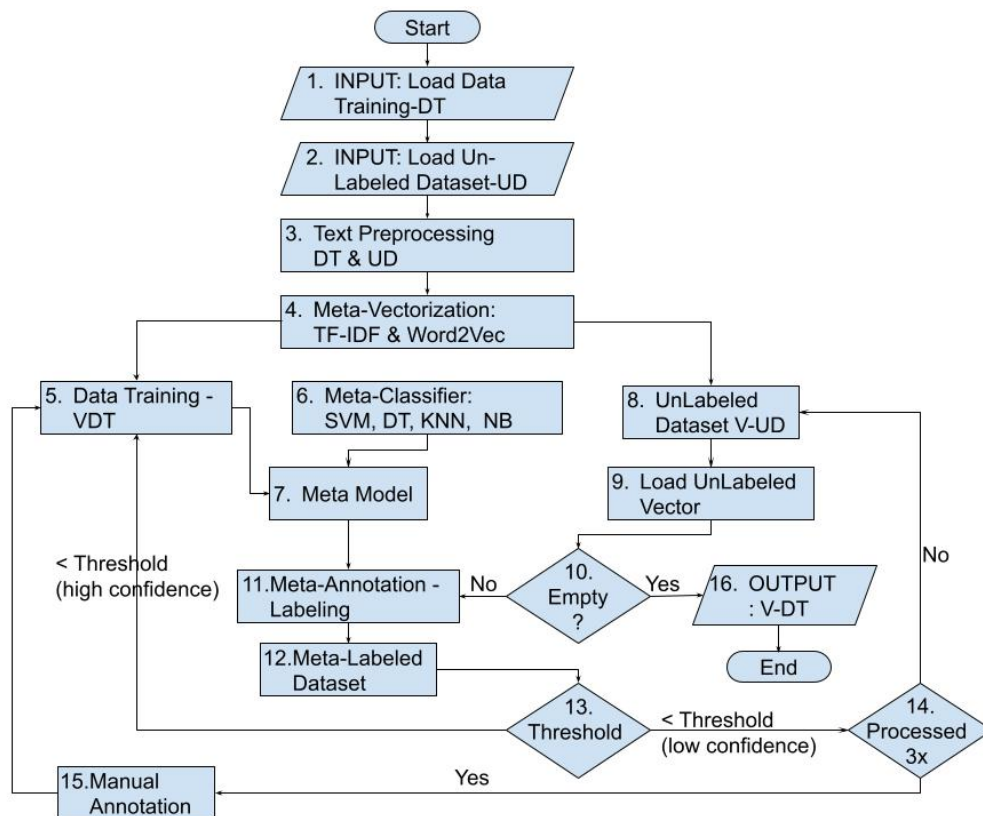
1. Machine learning modeling uses the meta-vectorizer and meta-classifier methods to determine the best model.
2. Carrying out experiments with self-learning scenarios using thresholds of 0.6, 0.7, and 0.8 in the proposed model and labeled datasets with the proportion of 5%, 10%, and 20%, and with a training data to test data ratio of 80%.
3. Experimental evaluation of the machine learning methods proposed in this study and their comparisons based on the mentioned above scenarios.

This study conducted experiments to evaluate the machine learning model based on the meta-vectorizer and meta-classifier concepts proposed herein. The proposed models have different accuracy results in each of the proposed scenarios. The minimally labeled data section (5%) yielded the highest accuracy for the proposed model, indicating that this proportion is the most effective.

The research presented in this paper is organized as follows. In Section 2, we describe the dataset and machine learning methods used in the NLP-based approach for hate speech detection. Section 3 discusses the experiments and results obtained based on the proposed scenarios. Finally, Section 4 provides conclusions of the conducted research.

## 2. Materials and Methods

This research focuses on machine learning algorithms performing automatic annotations using a semi-supervised learning approach. The proposed approach is also used to identify the hate speech based on text processing. The research process was generally carried out using steps presented in Figure 1.



**Figure 1.** The model for hate speech detection using a semi-supervised learning approach based on text annotations.

Based on Figure 1, the semi-supervised hate speech annotation process begins by reading the expert's hate-speech annotated training data (DT) in Step 1. Step 2 is reading the data without annotation (UD). Step 3 is the text preprocessing of the DT and UD datasets. Step 4 is the meta-vectorization process. The meta-vectorization process converts the clean DT and UD datasets into 2 types of vectorization (TF-IDF and Word2Vec). Step 5 is the process of preparing the training data. Step 6 is the preparation of machine learning algorithms, namely SVM, DT, KNN, and NB. Step 7 is the creation of the meta-learning model. Meta-vector and meta-learning models will produce 8 different combinations of vectorization and machine learning methods. Each combination of machine learning methods will be used for the auto-annotation process. Steps 8 and 9 form the process of preparing the dataset of vectors without annotations. In step 10, it will be checked if there are still datasets that have not been annotated and then the process will proceed to step 11, which is the process of annotating the dataset. The annotation process is done by predicting labels using 8 different combinations of these vectors and machine learning methods. The prediction results are also validated to determine their accuracy, which results in a meta-labeled dataset from the auto-annotation process in Step 12. In Step 12, a voting process will be carried out to determine the label for each dataset record. There are 2 types of voting, namely the total score  $W$  (weight) of the vectorization-machine-learning model for hate-speech polarity and the total score  $W$  (weight) of the vectorization-machine-learning model for non-hate speech polarity. In Step 13, if the hate-speech or non-hate speech polarity score exceeds the threshold, then the dataset and its annotations will be moved to the training data. If not, it will be re-annotated in the next cycle. In Step 14, the whole cycle will be repeated 3 times. Subsequently, the remaining Unlabeled Dataset will be passed to Step 15, which is manual annotation. The manual annotation results will be combined with the training data.

## 2.1. Datasets

This research also focuses on the dataset used to detect hate speech [19]. Hate speech in this study is limited to the realm of politics and law in Indonesia. Thus, this research takes several public opinions on the presidential debate video on YouTube social media and opinions about the Covid-19 Pandemic. After reviewing the comments, it was concluded that the opinions expressed about the video may be considered in further research for several reasons:

1. All elements of hate speech are needed, namely hate speech, non-hate speech, and even very negative hate speech.
2. The truth of the contents and existence of this video are guaranteed because it is broadcast by official broadcasting institutions/channels.
3. Public opinion in YouTube video comments is fair because there is no limit to the length of the message, there is no censorship, and it is possible for users to interact with each other (reply).

Thus, this research determines that YouTube video comments related to the 2019 Indonesian presidential debate and about Covid-19 are relevant dataset sources to build a semi-supervised text annotator model for hate speech. The YouTube channels used in this research are official Indonesian television channels, including Kompas TV, News Metro TV, TV-One, IDN Times, CNN Indonesia, and I-News TV.

Viewer comments from videos are used in this research. Public opinion data, contained in the debate video, have the following characteristics:

1. Comments and conversations are free and unstructured and contain emoticons, punctuation, and special characters that reflect the sentiments and emotional state of the user.
2. Some comments directly respond to the contents of the video, some comments respond to other comments (replies), and some reveal the user's side.
3. Comments contain elements of ambiguity, such as satire, polysemy, slang words, stop-words, and metaphors.

Concentrating on six official television channels, the study logs the volume of opinions and comments per video. Not all opinions were downloaded at the data collection stage, but the sampling method was used.

### 2.1.1. Population and Sampling

Based on monitoring of data sources, we know that the size of the dataset is dynamic, meaning that these opinions continue to grow, even though the presidential candidates' debate or the Covid-19 pandemic is already underway. New videos on the official channel also appeared, followed by viewer opinions. Thus, this study concludes that the population size is unknown, and it is impossible to download all the comments from the YouTube repository. So, it was determined during the research that the sampling method used should be purposive sampling. The purposive sampling was chosen for the following reasons:

1. The results of our initial observations found hate speech in YouTube video comments on the topics of the 2019 Indonesian presidential debate and Covid-19. This study assumes that the data are well-matched to our research objectives.
2. Data from YouTube video comments is publicly available and can be downloaded free of charge.

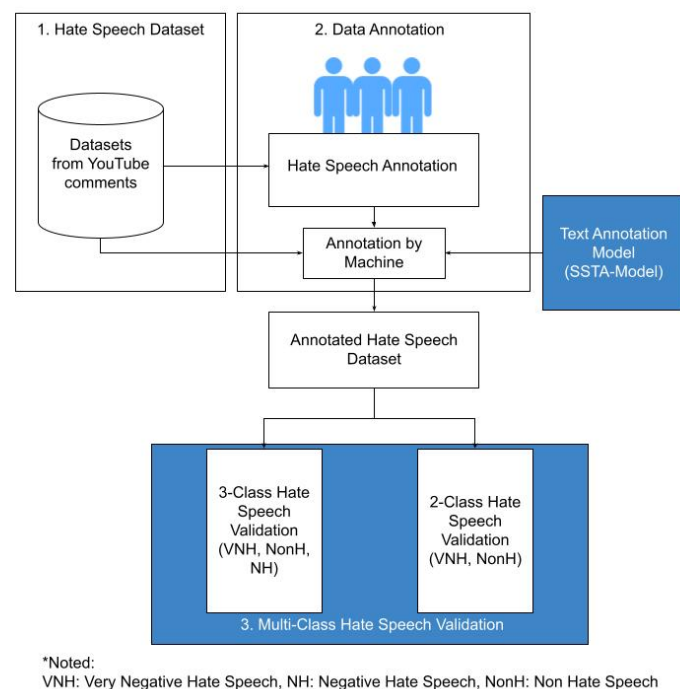
The number of videos related to the sample presidential debates required refers to previous similar studies. The population of comments from videos was collected considering the following criteria:

1. The data collection stage downloads all comments from the presidential debates 1 to 5, each broadcast in full by 2 official channels. So, the comments from 10 videos of presidential debates and 5 news about Covid-19 were downloaded.



- Moreover, comments were also downloaded from the official channel videos that do not feature the complete presidential debate but are deemed necessary to download due to the high number of views (more than 10,000 views), comments (over 1,000 comments), and exciting topics.

The samples that were obtained were described by experts in terms of hate speech, sentiment, and emotions. The results were used as a reference for automatically annotating population datasets using the Semi-Supervised Text Annotator Model, and the validity of the annotations was verified using the prepared machine learning model. The proposed machine learning model was tested for its ability to classify a multi-class dataset, as shown in the process of hate speech categorization (Figure 2).



**Figure 2.** The proposed automatic annotation process in detecting hate speech using a semi-supervised and self-learning approach.

## 2.2. Pre-processing

Based on the dataset, pre-processing should be performed to normalize the text data [20,21]. In our case, the dataset is unstructured data, so to improve the results and make it more structured [22] it is necessary to do pre-processing [23–26]. Text pre-processing is the process of cleaning and preparing text data for use in natural language processing (NLP) tasks. Typically, this involves removing unnecessary characters and words such as punctuation marks and common stop-words [27], as well as formatting the text in a way that makes it easier for NLP algorithms to analyze and understand. Raw text data is often noisy and difficult for algorithms to analyze, and pre-processing can help improve the accuracy and efficiency of NLP models [28–30]. The steps involved in text pre-processing typically include the following:

- Cleaning the text—it involves removing any unnecessary or irrelevant characters, such as punctuation marks or special characters, from the text [31].
- Tokenization—it involves splitting the text into smaller units, called tokens, such as words or phrases [32].
- Removing stop-words—stop-words are common words that do not provide significant meaning [33], such as “the” or “and”, and can be removed from the text to reduce its size and improve the performance of the NLP model.

4. Stemming and lemmatization—these are the techniques that are used to reduce words to their base form [34], called the stem or lemma, which reduces the number of words in the text and improves the model's ability to understand the text.
5. Part-of-speech tagging—it involves identification of the parts of speech in the text, such as a noun or verb, which can be useful for certain NLP tasks [35].
6. Normalization—it involves formatting the text consistently, which includes converting all words to lowercase to make it easier for the NLP model to process the text [36].

Overall, the goal of text pre-processing is to clean and prepare text data for use in NLP tasks to improve the accuracy and efficiency of the NLP model. The specific steps involved in text pre-processing can vary depending on the specific NLP task and the quality of the raw text data.

### 2.3. Meta-vectorization Based on Text Feature Extraction

The result of the preprocessing is the default text that is an input in the feature extraction process. The feature extraction aims to transform the input data into meaningful features [37]. This work uses feature extraction with the concept of meta-vectorization. Vectorization converts input (data) from text into vectors of real numbers that are used by machine learning algorithms. This concept is known as text feature extraction. Meta-vectorization was used in this study to obtain information from the extracted text that met specific requirements. This technique involves applying text feature extraction (characteristics of the dataset) [38]. This study applies Term Frequency-Inverse Document Frequency (TF-IDF) and Word Embedding (Word2Vec) feature extraction [39].

#### 2.3.1. Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF was run to weight words, and each term in a document was given a different weight [40] based on the frequency per document and all documents [41]. TF-IDF, which was applied in this study, has a good performance [22]. TF-IDF is calculated by multiplying two statistics: term frequency (TF) and inverse document frequency (IDF) [42]. Term frequency indicates how often a particular word appears in a document. It is calculated by dividing the number of word occurrences by the total number of words in the document. This method can be calculated by Eqs. (1), (2), and (3).

$$TF(t, d) = \frac{c(t, d)}{\sum_i c(t_i, d)} \quad (1)$$

$$IDF(t) = 1 + \log \frac{D}{d_t} \quad (2)$$

$$TF - IDF(t, d) = TF(t, d) \cdot IDF(t) \quad (3)$$

The value of  $c(t, d)$  indicates the occurrence of the  $t$  term in the document  $d$ , and the denominator shows the total number of terms in the document  $d$ .  $D$  is the total number of documents in the dataset, and  $d_t$  is the number of documents in which the term  $t$  appears.

#### 2.3.2. Word Embedding (Word2Vec)

Word2vec is a neural network-based word embedding method for representing words as vectors (of length  $N$ ) [43]. This method has three layers, namely input, hidden, and output. The input layer has a one-hot encoded vector whose length is the sum of the training data and applies two models: skip-gram and Continuous Bag of Words (CBOW). Skip-grams can represent less frequent words better than CBOW [44]. The input layer utilizes vectors where all values are zeros, except for a single value set to 1. This distinctive value of 1 uniquely characterizes each word representation within the vectors. The neurons of the input layer represent a single word in the vocabulary. In both the CBOW and skip-gram models of Word2Vec, the hidden layer comprises neurons that capture contextual information. In CBOW, these neurons amalgamate context from multiple neighboring words to predict

a target word. In skip-gram, each neuron specializes in comprehending the context around a single target word. In essence, the hidden layer serves to encode the intricate semantic relationships between words, critical for generating meaningful word embeddings. The activation function is linear. The value of the neuron is acquired by multiplying the input value and the weight (Eq. (4)). Meanwhile, multiplying the value of the hidden layer with the weight of the output layer requires Eq. (5). In Word2Vec's CBOW and skip-gram models, output layers differ. CBOW predicts the target word from context, generating a probability distribution. Skip-gram predicts multiple context words for one target, with separate distributions. Both use soft-max to convert outputs into probabilities for accurate prediction. (Eq. (6)).

$$h = W^T x \quad (4)$$

$$h = W^T x \quad (5)$$

$$y = \frac{\exp(u_j)}{\sum_{j'=1}^V \exp(u_{j'})} \quad (6)$$

The  $h$  is the hidden layer, and  $W^T$  is the transpose of the weight vector  $x$ .  $u_j$  is the output path to the hidden layer. The Soft-max activation function ( $y_i$ ) uses the output value of  $u_j$  according to the number of vocabulary words ( $V$ ).

#### 2.4. Meta-Classification Using Machine Learning Algorithms

Based on standard machine learning hate detection, the methods that we use in meta-classification include SVM, DT, KNN, and NB [45].

##### 2.4.1. Support Vector Machine

Support Vector Machines (SVMs) are supervised learning algorithms used in classification and pattern recognition. This method applies the principles of structural risk minimization and linear classifiers in solving linear and non-linear cases to identify hyperplanes from these classifications according to their formation patterns (+1 and -1) and their discriminative alternative boundaries. A hyperplane functions as a class separator (support vector) and determines the best hyperplane (lines in 2 or 3-dimensional space) in the training process that will maximally separate different classes. This concept refers to statistics with accuracy based on specific parameters and functions, which are used in the training process of the SVM classification process. SVM will identify the training data points closest to the hyperplane (support vectors), which are used to define the hyperplane and predict new data [46].

SVM implements the Mercer theorem kernel using Eq. (7), which will optimize quadratic programming with the constraints  $a_j \geq 0$  and  $\sigma_j(a_j y_j)$  by calculating the dot product of the pair (calculated by Eq. (8)) based on the effect of vector  $a$  on the value of  $w$ .

$$\arg \min_x \sum_j a_j - \frac{1}{2} \sum_{j,k} a_j a_k y_j y_k (x_j \cdot x_k) \quad (7)$$

$$h(x) = \text{sign} \left( \sum_j a_j y_j (x \cdot x_j - b) \right) \quad (8)$$

$j$  will consistently hold the value of 0 when the vector is positioned far from its nearest point. In the context of Support Vector Machines (SVM), a training and testing process is conducted to identify and annotate hate speech. This process is guided by training data with minimal labeling and a designated target class. The process involves addressing the constraints defined by Eq. (9) while simultaneously



optimizing the value of  $w$  and  $m$  as specified in Eq. (10). These results will be used to determine parameter  $c$  (penalty parameter) and error value ( $\epsilon$ ) based on  $lab_j(w.x_i) \leq 1 - \epsilon_j, j = 1, 2, 3, \dots, n$ .

$$y(x) = w.x + m \quad (9)$$

$$\text{minimize} \left( \frac{1}{2} |w|^2 + c \sum \epsilon_j \right) \quad (10)$$

#### 2.4.2. Decision Tree (DT)

Decision trees are a type of algorithm used in machine learning and data mining [47]. It is a predictive model commonly used to classify objects or predict outcomes based on specific input characteristics. Decision trees are hierarchical, with a root node representing the entire dataset and branches representing possible outcomes based on the values of the input features. The leaves of the tree represent the final classification or prediction. Decision trees can be constructed from a relatively simple set of rules. A decision tree algorithm is a supervised learning method for classification and regression tasks. Each inner node represents an attribute ("test"), each branch represents the result of the test, and each leaf node represents a class label. The algorithm starts at the root node and traverses the tree from top to bottom, making decisions at each internal node based on the values of input attributes. The final classification or prediction is determined by the leaf node reached at the end of the traversal.

Decision tree can handle both numeric and categorical data. An ensemble decision tree is a type of algorithm that combines predictions from multiple decision trees to improve the overall performance and accuracy of the model [48]. Ensemble methods use multiple learning algorithms to achieve better predictive performance than each algorithm alone. This can be achieved using bagging, boosting, and bootstrapping techniques. Ensemble decision trees are commonly used in various applications, such as image and text classification, regression, and anomaly detection.

#### 2.4.3. K-Nearest Neighbors (KNN)

KNN is a machine learning method that can be used for classification and regression to be implemented in pattern recognition [5]. The KNN learning operates with real-time efficiency, identifying the nearest reference data point in the feature space to classify incoming data [49]. This method assigns labels based on the characteristics shared with nearby data points. The process involves distinguishing test data from training data and organizing them into tuples, each characterized by specific features. Each tuple corresponds to a point in the multidimensional space, forming a reference set. Uncertain tuples are compared with their nearest neighbors to determine their classification based on the closest K-value neighbors.

KNN classification solves the problem by approaching the calculation of the similarity of the two data and selecting the value of  $k$  [50]. In the realm of the K-nearest neighbors (KNN) algorithm, various distance metrics are utilized to assess the similarity or dissimilarity between data points, thereby influencing the KNN algorithm's efficacy. Notable metrics include Euclidean Distance, Mahalanobis Distance, and Minkowski Distance. For the purposes of this study, our exclusive focus is on the Euclidean Distance formula (Eq. (1)), which measures the dissimilarity between data points based on their multidimensional coordinates ( $x_{i_k}$  and  $x_{j_k}$ ) within the dataset  $p$ . This Euclidean Distance calculation, denoted as  $d(i, j)$ , plays a pivotal role in KNN-based decision-making processes, impacting various applications in machine learning and pattern recognition.

$$d(i, j) = \sqrt{\sum_{k=1}^p (X_{i_k} - X_{j_k})^2} \quad (11)$$

The determination of the appropriate value for the variable  $k$  involves a systematic experimentation process. This process commences with an initial assessment, beginning with  $k = 1$ , followed by iterative tests. The objective is to identify the optimal value of  $k$  that yields the best results for each specific scenario. On the other hand, the application of the meta-learner automates the selection of the most suitable  $k$  value. This selection is made based on predefined criteria that aim to maximize the accuracy of the results by considering the relationships between data points and their neighbors.

#### 2.4.4. Naive Bayes (NB)

Naive Bayes is a family of naive probabilistic classifiers based on the application of Bayes' theorem with a strong (simple) assumption of independence between features [51]. The probability of the class is determined by calculating the probability of each feature occurring within that class and dividing it by the probability of the feature itself (Eq. (12)). Naive Bayes classifiers are highly scalable and require a set of parameters proportional to the number of variables (features) in the dataset. Because NB classifiers are simple and can be trained quickly, they are often used for large datasets. They are also commonly used in text classification and natural language processing tasks, where strong independence assumptions are often met.

To make predictions using the Naive Bayes algorithm, we first need to compute class prior probabilities and feature probabilities for each class. These probabilities are typically estimated from the training data using maximum likelihood estimation. Once we have these probabilities, we can use Bayes' theorem to compute the posterior probabilities of each class for a given feature set and find the class with the highest posterior probability of our choice to make a prediction.

An ensemble naive Bayes classifier is an algorithm that combines the predictions of multiple naive Bayes classifiers to improve the overall performance and accuracy of the model [52]. Ensemble methods use multiple learning algorithms to achieve better prediction performance than can be achieved with each algorithm alone. This can be achieved using bagging, boosting, and bootstrapping techniques. The Ensemble Naive Bayes classifier is commonly used in various applications, such as text classification and natural language processing tasks [53].

$$P(class|feature) = \frac{P(feature|class) * P(class)}{P(feature)} \quad (12)$$

### 3. Results

In this section, we present the experimental process and explain the results obtained in this study. The experiments were conducted based on the proposed scenarios using datasets from YouTube comments. This aligns with the goal of self-learning of automated annotation processes with minimal datasets to obtain optimal accuracy according to the proposed method. The model performance was assessed using a confusion matrix.

#### 3.1. Experimental Scenario Setup

In this study, we designed annotation text scenarios using a self-learning approach to detect hate-speech in 3-ways (very negative hate-speech, negative hate-speech, and non-hate-speech) or 2-ways (hate-speech, and non-hate-speech) as shown in Figure 2. The proposed model uses several machine learning methods with the scenario of dividing the dataset (see Table 1). This scenario aims to obtain optimal annotation predictions using minimal data. This scenario has different variations in the number of labeled datasets used and the threshold that identifies the limits of the similarity of the weights of the data being tested.

**Table 1.** The automatic annotation process scenario uses minimal training data and threshold.

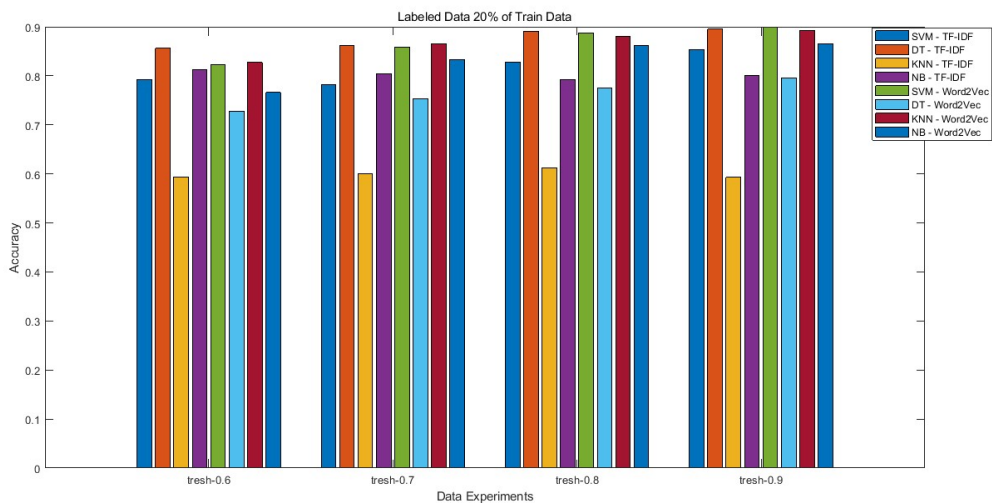
No	Training Data (%)	Unlabeled Data (%)	Threshold
1	20	80	0.6
2	20	80	0.7
3	20	80	0.8
4	20	80	0.9
5	10	90	0.6
6	10	90	0.7
7	10	90	0.8
8	10	90	0.9
9	5	95	0.6
10	5	95	0.7
11	5	95	0.8
12	5	95	0.9

3.2. Experimental Results of the Machine Learning Based Approach

This section presents the results and analysis of machine learning methods’ performance with the meta-learner concept based on feature extraction of meta vectorization. The carried out experiments used the processing scenario as shown in Table 1. The grouping was based on the percentage of training data, namely 20%, 10%, and 5%.

The experiment aimed to determine meta-learners’ performance in automatic annotations using the least amount of labeled training data, to optimize the annotation process. Thus, the annotation process with a small amount of labeled training data can annotate itself using the application of self-learning. Furthermore, in the context of self-learning, we employ threshold parameters to identify and seek similarities within the training dataset. This approach ensures that annotations align closely with the training and testing phases.

The results obtained use accuracy to determine the correctness of the annotation process. Figure 3 presents the results of calculating the accuracy of each combination of machine learning algorithm (SVM, DT, KNN, and NB) with feature method (TF-IDF and Word2Vec). The evaluation uses manual annotations of 20% of 13169 data. This automatically generated annotation will be used for the initial training.

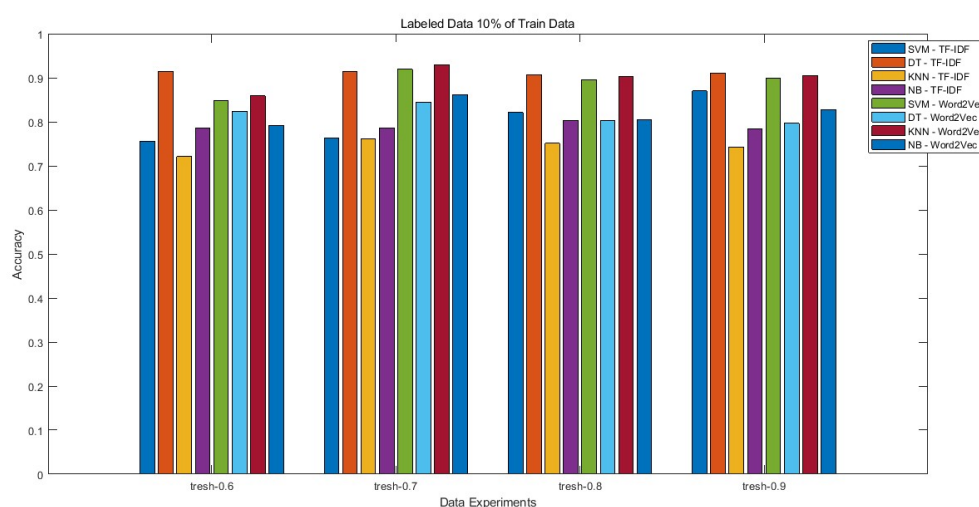


**Figure 3.** The Automatic Annotation Process Proposed in Detecting Hate Speech Using a Semi-Supervised and Self-Learning Approach Based on Scenarios 1, 2, 3, and 4 (Table 1).

Figure 3 shows that the machine learning model with 20% initially labeled data scenario is capable of performing automatic annotations with varying accuracy, with the most considerable value

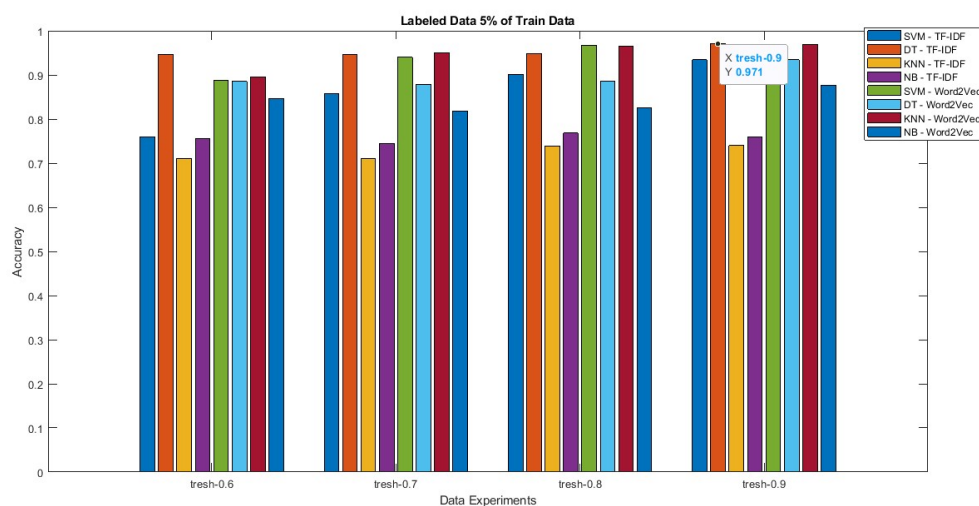
being 90% at the threshold of 0.9 obtained by the SVM-Word2Vec method. The accuracy of each individual method is visible on the chart presented in Figure 3. The evaluation showed that the worst machine learning methods in all scenarios were KNN combined with both feature extraction algorithms (TF-IDF and Word2Vec). The other methods have an accuracy of  $\approx 80\%$  and above. The accuracy of KNN-TF-IDF method has increased as compared to Cahyana et al. [5] from 59.68% to 61.3% (+1.62 p.p.). This enhancement reflects advancements in the preprocessing procedures from prior research and has been validated through several experimental scenarios, as shown in Table 1. The Indonesian language preprocessing was carried out using the Sastrawi library for Python in the stemming process. In addition, we have also added several stop-words to calculate the average vector features. However, it is worth noting that these stop-words are not employed in the Word2Vec method due to the fundamental differences in the calculation process it entails. In addition, the SVM-TF-IDF and SVM-Word2Vec methods, as implemented by Saifullah et al. [54], have been assessed under similar conditions. These methods achieved consecutive accuracies of 63.3% and 89%. As a result, feature extraction using TF-IDF has increased accuracy in performing automatic annotations by 19.5%. This also influences the application of the ensemble concept and the addition of stop-words data in the preprocessing. The Word2Vec method has achieved optimal results, and when a threshold value of 0.9 is added, it leads to a 90% yield, representing a slight 1% increase.

Based on the scenario results, the amount of 20% of the labeled data has not yet obtained optimal accuracy because several methods for detecting hate speech annotations have an accuracy of  $\approx 90\%$  [55] and even more than 95% [45]. So, we conducted a trial to reduce the amount of labeled data to 10% (Figure 4) and 5% (Figure 5) of the total YouTube comment data. We aim to optimize the automatic annotation with minimal amount of data. The text annotation results improved in the case of several methods, such as DT-TF-IDF and KNN-Word2Vec. These methods obtained the accuracy of 90% and more for all threshold experiments (for DT-TF-IDF) and 0.7, 0.8, and 0.9 threshold experiments (for KNN-Word2Vec). In addition, the accuracy of SVM-Word2Vec also increased to 91.9% in the 10%, 90%, and 0.7 parameters of the same scenario. So, the notable accuracy improvement when reducing the labeled data from 20% to 10% data is based on Figure 3 and Figure 4. This increase in accuracy is based on the process of calculating the weight of the available data, where the smaller the amount of data, the higher the accuracy of the process. In addition, the self-learning performance of the meta-learner and meta-vectorizer relies on an ensemble approach to optimize the calculated weights, resulting in enhanced performance.



**Figure 4.** The Proposed Automatic Annotation Process for Detecting Hate Speech Using a Semi-Supervised and Self-Learning Approach Based on Scenarios 5, 6, 7, and 8 (Table 1).

The improvement of selected proposed methods from previous studies [5,54] became a reference for conducting this research. Building upon some proposed methodologies from the previous studies [5], our future work encompasses an exploration of various scenarios. These scenarios involve the reduction of labeled data to 5%, 10%, and 20%, in conjunction with the adjustment of threshold values to 0.6, 0.7, 0.8, and 0.9. This study has achieved this by employing meta-vectorization and meta-learning methods to enhance the annotation process. Furthermore, the application of threshold variations consistently resulted in the highest accuracy at the most significant value, specifically 0.9. This pattern was observed across all variations in the percentage of labeled data, except for the 10% labeled data scenario. This observation is attributed to the robust performance of SVM in various contexts, as documented in references [56,57], including the present research, as evidenced by Figures 3, Figure 4, and Figure 5. These comparisons are readily evident in Figure A1 and Table A1. Notably, SVM consistently exhibits increasing accuracy. However, while the overall accuracy results demonstrate an increase, it is worth noting that certain methods experience a reduction of up to 8% in specific scenarios. The detailed comparison of the increase and decrease in method accuracy based on the scenario can be seen in Table A2.



**Figure 5.** The Proposed Automatic Annotation Process for Detecting Hate Speech Using a Semi-Supervised and Self-Learning Approach Based on Scenarios 9, 10, 11, and 12 (Table 1).

Based on the results of the 10% labeled data scenario, the final experiment is to apply a 5% labeled data scenario for training with each existing threshold. Remarkably, each scenario demonstrates a systematic progression, encompassing thresholds ranging from minor to the most significant. The accuracy of this scenario is best with a threshold value of 0.9. The highest accuracy was obtained for DT-TF-IDF with the value of 97.1%, followed by KNN-Word2Vec (96.9%), SVM-Word2Vec (96.8%), SVM-TF-IDF, DT-Word2Vec (93.4%), and others below 90%. These findings indicate that varying the quantity of data and adjusting the threshold value had a discernible impact on the methods' accuracy, leading to improved performance. In Figure 5, we can observe that DT-TF-IDF consistently achieves a commendable accuracy level exceeding 94%, surpassing both KNN-Word2Vec and SVM-Word2Vec (exact accuracy values are shown in Table A1).

Based on presented scenarios and outcomes, this study adopts a semi-supervised learning approach, as illustrated in Figure 1. It involves the annotation of unlabeled data and substantial refinement of the sample dataset, ultimately leading to the automatic hate speech annotation. This process implements a self-learning mechanism, which leverages prior learning experiences, to address the limitations associated with manual annotations. In terms of algorithm design, our proposed approach is shown in Listing 1. This annotation delineates an approach that employs minimal training



data to maximize accuracy. This is achieved through the incorporation of a threshold parameter designed to align the training dataset with the annotation process.

**Listing 1.** Text Auto-Annotation Based on Semi-Supervised and Self-Learning Approach

---

STEPS :

1. Dataset Input:
    - a. Annotated Dataset (20%) with scenarios training 5%:10%:20%
    - b. UnAnnotated dataset (80%)
  2. Preprocessing
    - a. Annotated Dataset → semi structured Annotated Dataset
    - b. UnAnnotated dataset → semi structured UnAnnotated Dataset
  3. Vectorization using merged semi structured annotated dataset + unanotated dataset : TF-IDF, Word2Vec
  4. Put 10% from semi structured annotated dataset for sample dataset
  5. Create model classifier with 80% data training and 20% data validation. Model using SVM, DT, KNN, NB
  6. Get the best model MTA : vectorizer + machine learning (with best parameter)
 

===== VALIDATE  
SEMI SUPERVISED ANNOTATION BEST MODEL Using SSADF  
=====
  7. Convert Semi Structured Annotated Dataset → Semi Structured Annotated Dataset FEATURE : SSADF
  8. Split SSADF into 80% for Data Training (SSADFT) and 20% for Data Validation (SSADFV)
  9. Create MTA model using SSADFT (as Data Training)
  10. VALIDATE MTA Model using SSADFV (as Data Validation)
 

===== SEMI  
SUPERVISED ANNOTATION using THE BEST MODEL  
=====
  11. Convert Semi Structured UnAnnotated dataset → Semi Structured UnAnnotated Dataset FEATURE : SSUDF
  12. AnnotationResult = Annotate SSUDF using MTA MODEL with SSADFT
  13. percentage = VALIDATE AnnotationResult using SAADFV
  14. SAVE AnnotationResult\_Valid to merge with SSADF
  15. IF percentage < treshold THEN Go To 7.  
Output : SSADF
- 

#### 4. Conclusions

This research paper introduces a novel approach to enhance machine learning techniques by incorporating a meta-learner for the automated annotation of hate speech in Indonesian text. The proposed concept combines a meta-learner and a meta-vectorizer to get an annotation process with a minimal dataset applied. The experimental scenario was carried out using 3 labeled dataset variations, namely 5%, 10%, and 20%. In addition, the labeling process is accomplished by computing the weights of individual model combinations, employing a threshold system with four predefined values: 0.6, 0.7, 0.8, and 0.9. In each scenario, a specific calculation approach is employed to maximize the accuracy of the proposed model. The optimization process for annotating small datasets has proven successful, achieving automatic annotations with only 5% labeling and a threshold of 0.9. Notably, the DT-TF-IDF method achieved the highest accuracy, reaching an impressive 97.1%. In addition, the model that has the best accuracy is followed by KNN-WOrd2Vec (96.9%), SVM-Word2Vec (96.8%), SVM-TF-IDF (93.4%), DT-Word2Vec (93.4%) in the same scenario (5%, 80%, 0.9). The comprehensive findings indicate a consistent trend of improved accuracy with increasing threshold values, with the highest threshold

consistently yielding superior accuracy across all scenarios. This result is inversely proportional to the percentage of labeled training data—lower percentages correlate with higher accuracy value. Based on Table 1, we observe that as the dataset size decreases from largest to smallest, the accuracy value increases. These study findings can serve as a useful reference for improving the accuracy of automatic annotation processes.

This approach has the potential to increase the number of datasets and their corpus in the future, optimizing annotations. In addition, there is a need to explore various methods, including the possibility of applying other machine learning techniques, to develop automated annotation processes further. In future projects, we can explore the development of multiple criteria for hate speech categorization, such as different classification categories like abusive, aggressive, offensive, and others.

**Author Contributions:** Conceptualization, Shoffan Saifullah and Agus Aribowo; Data curation, Shoffan Saifullah, Agus Aribowo, Yuli Fauziah and Nur Cahyana; Formal analysis, Shoffan Saifullah, Rafał Dreżewski and Agus Aribowo; Funding acquisition, Rafał Dreżewski; Investigation, Shoffan Saifullah and Agus Aribowo; Methodology, Shoffan Saifullah and Agus Aribowo; Project administration, Shoffan Saifullah; Resources, Shoffan Saifullah, Agus Aribowo, Yuli Fauziah and Nur Cahyana; Software, Shoffan Saifullah and Agus Aribowo; Supervision, Shoffan Saifullah and Rafał Dreżewski; Validation, Shoffan Saifullah, Rafał Dreżewski and Agus Aribowo; Visualization, Shoffan Saifullah and Agus Aribowo; Writing – original draft, Shoffan Saifullah, Rafał Dreżewski, Felix Dwiyanto and Agus Aribowo; Writing – review & editing, Shoffan Saifullah, Rafał Dreżewski and Felix Dwiyanto..

**Funding:** The research presented in this paper was partially supported by the funds of Polish Ministry of Education and Science assigned to AGH University of Krakow.

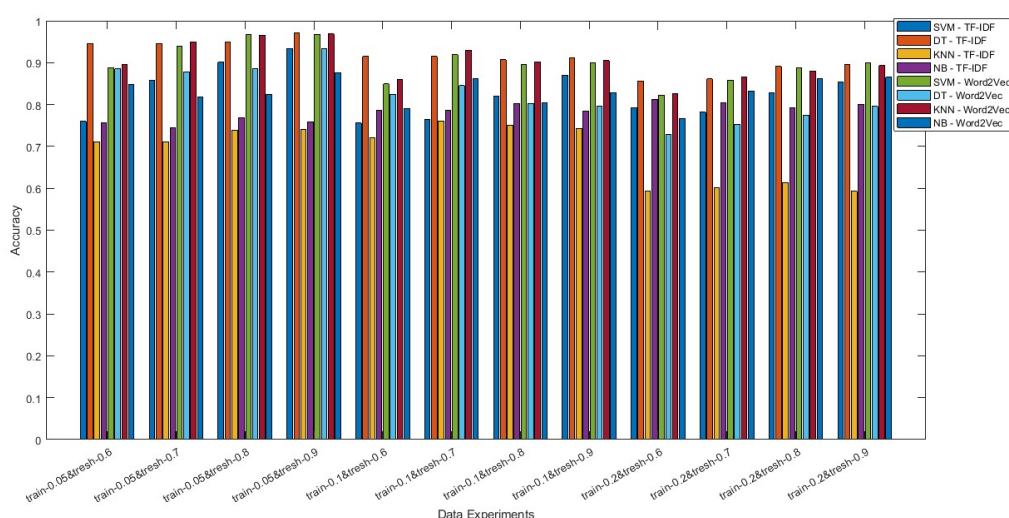
**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare that there is no conflict of interest in conducting research and reporting this manuscript.

## Appendix A



**Figure A1.** The Accuracy results of the Proposed Models in Text Annotations for Hate Speech Detection Using a Semi-Supervised (Self-Learning) Approach based on scenarios labeled data (5%, 10%, and 20%) and threshold (0.6, 0.7, 0.8, and 0.9) .

## Appendix B

**Table A1.** Accuracy Results of the Combination of Meta-Learning and Meta Vectorization Modeling Based on Scenario of Table 1.

No	Labeled Data	Unlabeled Data	Threshold	SVM TF-IDF	DT TF-IDF	KNN TF-IDF	NB TF-IDF	SVM Word2Vec	DT Word2Vec	KNN Word2Vec	NB Word2Vec
1	0.2	0.8	0.6	0.793	0.856	0.594	0.812	0.823	0.728	0.827	0.766
2	0.2	0.8	0.7	0.782	0.861	0.601	0.804	0.858	0.753	0.866	0.833
3	0.2	0.8	0.8	0.828	0.891	0.613	0.793	0.888	0.775	0.88	0.861
4	0.2	0.8	0.9	0.853	0.895	0.593	0.801	0.9	0.796	0.893	0.865
5	0.1	0.9	0.6	0.756	0.915	0.721	0.786	0.849	0.824	0.859	0.791
6	0.1	0.9	0.7	0.764	0.915	0.761	0.786	0.919	0.845	0.93	0.862
7	0.1	0.9	0.8	0.821	0.907	0.751	0.803	0.896	0.803	0.902	0.804
8	0.1	0.9	0.9	0.87	0.911	0.743	0.784	0.899	0.797	0.905	0.828
9	0.05	0.95	0.6	0.76	0.946	0.711	0.756	0.888	0.885	0.896	0.847
10	0.05	0.95	0.7	0.858	0.946	0.711	0.745	0.94	0.878	0.95	0.818
11	0.05	0.95	0.8	0.901	0.949	0.739	0.768	0.967	0.885	0.966	0.825
12	0.05	0.95	0.9	0.934	0.971	0.74	0.759	0.968	0.934	0.969	0.876

## Appendix C

**Table A2.** Comparison of Increase and Decrease in Accuracy in Each Scenario.

Scenario Comparisons	SVM TF-IDF (%)	DT TF-IDF (%)	KNN TF-IDF (%)	NB TF-IDF (%)	SVM Word2Vec (%)	DT Word2Vec (%)	KNN Word2Vec (%)	NB Word2Vec (%)
1-2	-1.10	0.50	0.70	-0.80	3.50	2.50	3.90	6.70
2-3	4.60	3.00	1.20	-1.10	3.00	2.20	1.40	2.80
3-4	2.50	0.40	-2.00	0.80	1.20	2.10	1.30	0.40
5-6	0.80	0.00	4.00	0.00	7.00	2.10	7.10	7.10
6-7	5.70	-0.80	-1.00	1.70	-2.30	-4.20	-2.80	-5.80
7-8	4.90	0.40	-0.80	-1.90	0.30	-0.60	0.30	2.40
9-10	9.80	0.00	0.00	-1.10	5.20	-0.70	5.40	-2.90
10-11	4.30	0.30	2.80	2.30	2.70	0.70	1.60	0.70
11-12	3.30	2.20	0.10	-0.90	0.10	4.90	0.30	5.10

## References

- Alrehili, A. Automatic Hate Speech Detection on Social Media: A Brief Survey. 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA). IEEE, 2019. doi:10.1109/aiccsa47632.2019.9035228.
- Al-Makhadmeh, Z.; Tolba, A. Automatic hate speech detection using killer natural language processing optimizing ensemble deep learning approach. *Computing* **2019**, *102*, 501–522. doi:10.1007/s00607-019-00745-0.
- Rajman, M.; Besançon, R. Text Mining: Natural Language techniques and Text Mining applications. In *Data Mining and Reverse Engineering*; Springer US, 1998; pp. 50–64. doi:10.1007/978-0-387-35300-5\_3.
- Fortuna, P.; Nunes, S. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys* **2018**, *51*, 1–30. doi:10.1145/3232676.
- Cahyana, N.H.; Saifullah, S.; Fauziah, Y.; Aribowo, A.S.; Drezewski, R. Semi-supervised Text Annotation for Hate Speech Detection using K-Nearest Neighbors and Term Frequency-Inverse Document Frequency. *International Journal of Advanced Computer Science and Applications* **2022**, *13*. doi:10.14569/ijacsa.2022.0131020.
- Aman, S.; Szpakowicz, S. Identifying Expressions of Emotion in Text. In *Text, Speech and Dialogue. TSD 2007*; Springer Berlin Heidelberg, 2007; pp. 196–205. doi:10.1007/978-3-540-74628-7\_27.
- Krouska, A.; Troussas, C.; Virvou, M. The effect of preprocessing techniques on Twitter sentiment analysis. 2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA). IEEE, 2016. doi:10.1109/iisa.2016.7785373.
- Savigny, J.; Purwarianti, A. Emotion classification on youtube comments using word embedding. 2017 International Conference on Advanced Informatics, Concepts, Theory, and Applications (ICAICTA). IEEE, 2017. doi:10.1109/icaicta.2017.8090986.

9. Ningtyas, A.M.; Herwanto, G.B. The Influence of Negation Handling on Sentiment Analysis in Bahasa Indonesia. 2018 5th International Conference on Data and Software Engineering (ICoDSE). IEEE, 2018. doi:10.1109/icodse.2018.8705802.
10. Mariel, W.C.F.; Mariyah, S.; Pramana, S. Sentiment analysis: a comparison of deep learning neural network algorithm with SVM and naïve Bayes for Indonesian text. *Journal of Physics: Conference Series* **2018**, *971*, 012049. doi:10.1088/1742-6596/971/1/012049.
11. Mao, R.; Liu, Q.; He, K.; Li, W.; Cambria, E. The Biases of Pre-Trained Language Models: An Empirical Study on Prompt-Based Sentiment Analysis and Emotion Detection. *IEEE Transactions on Affective Computing* **2022**, pp. 1–11. doi:10.1109/taffc.2022.3204972.
12. Dashtipour, K.; Gogate, M.; Gelbukh, A.; Hussain, A. Extending persian sentiment lexicon with idiomatic expressions for sentiment analysis. *Social Network Analysis and Mining* **2021**, *12*. doi:10.1007/s13278-021-00840-1.
13. Imran, A.S.; Yang, R.; Kastrati, Z.; Daudpota, S.M.; Shaikh, S. The impact of synthetic text generation for sentiment analysis using GAN based models. *Egyptian Informatics Journal* **2022**, *23*, 547–557. doi:10.1016/j.eij.2022.05.006.
14. Balli, C.; Guzel, M.S.; Bostanci, E.; Mishra, A. Sentimental Analysis of Twitter Users from Turkish Content with Natural Language Processing. *Computational Intelligence and Neuroscience* **2022**, *2022*, 1–17. doi:10.1155/2022/2455160.
15. Jain, D.K.; Boyapati, P.; Venkatesh, J.; Prakash, M. An Intelligent Cognitive-Inspired Computing with Big Data Analytics Framework for Sentiment Analysis and Classification. *Information Processing & Management* **2022**, *59*, 102758. doi:10.1016/j.ipm.2021.102758.
16. Kabakus, A.T. A novel COVID-19 sentiment analysis in Turkish based on the combination of convolutional neural network and bidirectional long-short term memory on Twitter. *Concurrency and Computation: Practice and Experience* **2022**, *34*. doi:10.1002/cpe.6883.
17. Al-Laith, A.; Shahbaz, M.; Alaskar, H.F.; Rehmat, A. AraSenCorpus: A Semi-Supervised Approach for Sentiment Annotation of a Large Arabic Text Corpus. *Applied Sciences* **2021**, *11*, 2434. doi:10.3390/app11052434.
18. Balakrishnan, V.; Lok, P.Y.; Rahim, H.A. A semi-supervised approach in detecting sentiment and emotion based on digital payment reviews. *The Journal of Supercomputing* **2020**, *77*, 3795–3810. doi:10.1007/s11227-020-03412-w.
19. Ibrohim, M.O.; Budi, I. Multi-label Hate Speech and Abusive Language Detection in Indonesian Twitter. Proceedings of the Third Workshop on Abusive Language Online. Association for Computational Linguistics, 2019. doi:10.18653/v1/w19-3506.
20. Zhang, Z.; Robinson, D.; Tepper, J. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *The Semantic Web*; Springer International Publishing, 2018; pp. 745–760. doi:10.1007/978-3-319-93417-4\_48.
21. Davidson, T.; Warmesley, D.; Macy, M.; Weber, I. Automated Hate Speech Detection and the Problem of Offensive Language. *Proceedings of the International AAAI Conference on Web and Social Media* **2017**, *11*, 512–515. doi:10.1609/icwsm.v11i1.14955.
22. Cahyani, D.E.; Patasik, I. Performance comparison of TF-IDF and Word2Vec models for emotion text classification. *Bulletin of Electrical Engineering and Informatics* **2021**, *10*, 2780–2788. doi:10.11591/eei.v10i5.3157.
23. Abduljabbar, D.A.; Omar, N. Exam questions classification based on Bloom's taxonomy cognitive level using classifiers combination. *Journal of Theoretical and Applied Information Technology* **2015**, *78*, 447–455.
24. Soliman, A.B.; Eissa, K.; El-Beltagy, S.R. AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP. *Procedia Computer Science* **2017**, *117*, 256–265. doi:10.1016/j.procs.2017.10.117.
25. Kumar, C.S.P.; Babu, L.D.D. Novel Text Preprocessing Framework for Sentiment Analysis. In *Smart Intelligent Computing and Applications*; Springer Singapore, 2018; pp. 309–317. doi:10.1007/978-981-13-1927-3\_33.
26. Ramachandran, D.; Parvathi, R. Analysis of Twitter Specific Preprocessing Technique for Tweets. *Procedia Computer Science* **2019**, *165*, 245–251. doi:10.1016/j.procs.2020.01.083.
27. Mohammed, M.; Omar, N. Question classification based on Bloom's taxonomy cognitive domain using modified TF-IDF and word2vec. *PLOS ONE* **2020**, *15*, e0230442. doi:10.1371/journal.pone.0230442.

28. Babanejad, N.; Agrawal, A.; An, A.; Papagelis, M. A Comprehensive Analysis of Preprocessing for Word Representation Learning in Affective Tasks. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020. doi:10.18653/v1/2020.acl-main.514.
29. Albalawi, R.; Yeap, T.H.; Benyoucef, M. Using Topic Modeling Methods for Short-Text Data: A Comparative Analysis. *Frontiers in Artificial Intelligence* **2020**, *3*. doi:10.3389/frai.2020.00042.
30. Arora, M.; Kansal, V. Character level embedding with deep convolutional neural network for text normalization of unstructured data for Twitter sentiment analysis. *Social Network Analysis and Mining* **2019**, *9*. doi:10.1007/s13278-019-0557-y.
31. Elgibreen, H.; Faisal, M.; Sulaiman, M.A.; Abdou, S.; Mekhtiche, M.A.; Moussa, A.M.; Alohal, Y.A.; Abdul, W.; Muhammad, G.; Rashwan, M.; Algabri, M. An Incremental Approach to Corpus Design and Construction: Application to a Large Contemporary Saudi Corpus. *IEEE Access* **2021**, *9*, 88405–88428. doi:10.1109/access.2021.3089924.
32. Rai, A.; Borah, S. Study of Various Methods for Tokenization. In *Applications of Internet of Things*; Springer Singapore, 2020; pp. 193–200. doi:10.1007/978-981-15-6198-6\_18.
33. Manalu, S.R. Stop words in review summarization using TextRank. 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON). IEEE, 2017. doi:10.1109/ecticon.2017.8096371.
34. Zeroual, I.; Lakhouaja, A. Arabic information retrieval: Stemming or lemmatization? 2017 Intelligent Systems and Computer Vision (ISCV). IEEE, 2017. doi:10.1109/isacv.2017.8054932.
35. AlKhawter, W.; Al-Twairesh, N. Part-of-speech tagging for Arabic tweets using CRF and Bi-LSTM. *Computer Speech & Language* **2021**, *65*, 101138. doi:10.1016/j.csl.2020.101138.
36. Sharma, A.; Kumar, S. Ontology-based semantic retrieval of documents using Word2vec model. *Data & Knowledge Engineering* **2023**, *144*, 102110. doi:10.1016/j.datak.2022.102110.
37. Liang, H.; Sun, X.; Sun, Y.; Gao, Y. Text feature extraction based on deep learning: a review. *EURASIP Journal on Wireless Communications and Networking* **2017**, *2017*. doi:10.1186/s13638-017-0993-1.
38. Garouani, M.; Ahmad, A.; Bouneffa, M.; Hamlich, M.; Bourguin, G.; Lewandowski, A. Using meta-learning for automated algorithms selection and configuration: an experimental framework for industrial big data. *Journal of Big Data* **2022**, *9*. doi:10.1186/s40537-022-00612-4.
39. Kamyab, M.; Liu, G.; Adjeisah, M. Attention-Based CNN and Bi-LSTM Model Based on TF-IDF and GloVe Word Embedding for Sentiment Analysis. *Applied Sciences* **2021**, *11*, 11255. doi:10.3390/app112311255.
40. Saifullah, S.; Fauziyah, Y.; Aribowo, A.S. Comparison of machine learning for sentiment analysis in detecting anxiety based on social media data. *Jurnal Informatika* **2021**, *15*, 45. doi:10.26555/jifo.v15i1.a20111.
41. Fauziah, Y.; Saifullah, S.; Aribowo, A.S. Design Text Mining for Anxiety Detection using Machine Learning based-on Social Media Data during COVID-19 pandemic. *Proceeding of LPPM UPN "Veteran" Yogyakarta Conference Series 2020–Engineering and Science Series*, 2020, pp. 253–261. doi:10.31098/ess.v1i1.117.
42. Capelle, M.; Hogenboom, F.; Hogenboom, A.; Frasinca, F. Semantic news recommendation using wordnet and bing similarities. *Proceedings of the 28th Annual ACM Symposium on Applied Computing — SAC'13*. ACM Press, 2013. doi:10.1145/2480362.2480426.
43. Sivakumar, S.; Videla, L.S.; Kumar, T.R.; Nagaraj, J.; Itnal, S.; Haritha, D. Review on Word2Vec Word Embedding Neural Net. 2020 International Conference on Smart Electronics and Communication (ICOSEC). IEEE, 2020. doi:10.1109/icosec49089.2020.9215319.
44. Landgraf, A.J.; Bellay, J. word2vec Skip-Gram with Negative Sampling is a Weighted Logistic PCA, 2017. doi:10.48550/ARXIV.1705.09755.
45. Alkomah, F.; Ma, X. A Literature Review of Textual Hate Speech Detection Methods and Datasets. *Information* **2022**, *13*, 273. doi:10.3390/info13060273.
46. Saifullah, S.; Drezewski, R. Non-Destructive Egg Fertility Detection in Incubation Using SVM Classifier Based on GLCM Parameters. *Procedia Computer Science* **2022**, *207*, 3248–3257. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 26th International Conference KES2022, doi:10.1016/j.procs.2022.09.383.
47. Bansal, M.; Goyal, A.; Choudhary, A. A comparative analysis of K-Nearest Neighbor, Genetic, Support Vector Machine, Decision Tree, and Long Short Term Memory algorithms in machine learning. *Decision Analytics Journal* **2022**, *3*, 100071. doi:10.1016/j.dajour.2022.100071.



48. Kuchipudi, R.; Uddin, M.; Murthy, T.; Mirrudoddi, T.K.; Ahmed, M.; P, R.B. Android Malware Detection using Ensemble Learning. 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS). IEEE, 2023, pp. 297–302. doi:10.1109/ICSCSS57650.2023.10169578.
49. Degirmenci, A.; Karal, O. Efficient density and cluster based incremental outlier detection in data streams. *Information Sciences* **2022**, *607*, 901–920. doi:10.1016/j.ins.2022.06.013.
50. Kesarwani, A.; Chauhan, S.S.; Nair, A.R. Fake News Detection on Social Media using K-Nearest Neighbor Classifier. 2020 International Conference on Advances in Computing and Communication Engineering (ICACCE). IEEE, 2020, pp. 1–4. doi:10.1109/ICACCE49060.2020.9154997.
51. Xu, S. Bayesian Naïve Bayes classifiers to text classification. *Journal of Information Science* **2018**, *44*, 48–59. doi:10.1177/0165551516677946.
52. Mwaro, P.N.; Ogada, D.K.; Cheruiyot, P.W. Applicability of Naïve Bayes Model for Automatic Resume Classification. *International Journal of Computer Applications Technology and Research* **2020**, *9*, 257–264. doi:10.7753/IJCATR0909.1002.
53. Zhang, F.; Fleyeh, H.; Wang, X.; Lu, M. Construction site accident analysis using text mining and natural language processing techniques. *Automation in Construction* **2019**, *99*, 238–248. doi:10.1016/j.autcon.2018.12.016.
54. Saifullah, S.; Cahyana, N.H.; Fauziah, Y.; Aribowo, A.S.; Dwiyanto, F.A.; Drezewski, R. Text Annotation Automation for Hate Speech Detection using SVM-classifier based on Feature Extraction. International Conference on Advanced Research in Engineering and Technology, 2022.
55. Kocoń, J.; Figas, A.; Gruza, M.; Puchalska, D.; Kajdanowicz, T.; Kazienko, P. Offensive, aggressive, and hate speech analysis: From data-centric to human-centered approach. *Information Processing & Management* **2021**, *58*, 102643. doi:10.1016/j.ipm.2021.102643.
56. Maniruzzaman, M.; Rahman, M.J.; Ahammed, B.; Abedin, M.M. Classification and prediction of diabetes disease using machine learning paradigm. *Health Information Science and Systems* **2020**, *8*. doi:10.1007/s13755-019-0095-z.
57. Machova, K.; Mach, M.; Vasilko, M. Comparison of Machine Learning and Sentiment Analysis in Detection of Suspicious Online Reviewers on Different Type of Data. *Sensors* **2021**, *22*, 155. doi:10.3390/s22010155.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.