ISSN 2347 - 3983
Volume 8. No. 3, March 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter04832020.pdf
https://doi.org/10.30534/ijeter/2020/04832020

Classification of Traffic Accident Information Using Machine Learning from Social Media

Dody Agung Saputro[1], Abba Suganda Girsang[2]
[1]Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia, 11480, dody.saputro@binus.ac.id
[2]Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia, 11480, agirsang@binus.edu

ABSTRACT

With the increasing number of accidents in Indonesia, analysis of accident data is still need to be considered and analyzed. Moreover, traffic accident information from social media such as Twitter is easy to obtain when compared to other data source from police or government institutions. In this work, we tried to create a traffic accident dataset. We crawling on Twitter, then do a text processing which involves case folding, filtering, stemming, tokenizing, and stop word removal. To find out whether the tweet is true about accident information, then we do a word embedding using FastText method and classification with SVM, KNN, and Naïve Bayes algorithms then testing the accuracy of models using K-fold validation method. The results show that the best accuracy of the model is up to 85% with SVM method. This model is then applied to a tweet dataset containing 142,168 tweets taken through the crawling process since April 2019 and can be used for further research on https://www.dodyagung.com/dataset/accident.

Key words : Classification, Machine Learning, Social Media, Traffic Accident

1. INTRODUCTION

The number of traffic accidents in Indonesia that cause death victims, serious injuries, and minor injuries is still relatively high and reported annually. In 2017, 103,493 cases of accidents were recorded nationwide with 25,865 fatalities [1]. In the previous 5 years from 2011 to 2016, the average of traffic accidents was 104,626 times and the average of the death toll was 28,022 annually [2].

With current technological advances, information about traffic accidents can easily be found through social media which is usually become a place for accident reporting. If on online news portals the content goes through an editorial process and contains journalistic elements, while on social media it is more self-reporting that involves the public. One of the most widely used social media is Twitter, which has 330 million average monthly active users [3]. With so much information found on Twitter social media, then that information can be obtained and then processed and classified according to certain categories, especially information relating to traffic accidents. This can make social media especially Twitter as a complementary source of traffic accident information. Usually the officially available datasets originate from the police or authorized government institutions.

This traffic accident information dataset or corpus does not use existing data sources, but using a new data by performing an automatic search by a scheduled machine. The used source is the results of Twitter crawl process. To achieve the best results a text preprocessing process is carried out and then weighting is done using the FastText method by Facebook which is a next development of the Word2Vec method that developed by Google. After that the classification process is done using several supervised algorithms such as Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Naïve Bayes and the model that has the best accuracy is evaluated using the K-Fold Cross Validation method with the number k=10. Next, the prediction process is continued by using the remaining available data. The labels used in this classification are {1} for true positive results of traffic accident news and {0} for negative results of traffic accident news.

Some of the contributions made from this research are as follows:

1. Crawl traffic accident information on Twitter then doing text preprocessing and FastText word embedding to create our own corpus or dataset.
2. Classification uses the SVM, KNN, and Naïve Bayes algorithms and builds its model to find out whether the sentence is a traffic accident information or not.
3. Publication of raw data from crawling, text processing, weighting, classification and extraction of traffic accident information in the form of corpus or dataset that can be used for further research on https://www.dodyagung.com/dataset/accident

2. THEORITICAL BASIS AND RELATED WORKS

The process of getting information from web pages can be done through web crawling processes and through the Really Simple Syndication (RSS) format. Some web crawling methods such as By HTTP Get Request and Dynamic Web Page and By the use of filters are the most preferred methods [4]. In addition, the RSS format is a form of content syndication from Extensible Markup Language (XML) based websites that can also be used [5]. Several previous studies on data mining used the RSS dataset to get content from websites directly, such as system recommendation automation on the website RSS Reader based on user click hyperlinks using the K-Nearest Neighbor algorithm [6]. Information in the form of a collection of sentences that have been obtained, then the entity can be extracted.

Entity extraction in a collection of sentences is an attempt to detect and classify entities, such as the name of a person, organization, place, time, and so on. This is often referred as Named Entity Recognition (NER). Several studies on NER have been conducted, including the creation of a NER automation tool named Stanford NER [7][8] and OpenNLP [9]. In addition to complete entity extraction, we can also use certain rule-based algorithms to extract entities specifically. The advantage of the extraction process will be faster and focus on certain entities, such as only extracting the place name from a sentence. In addition, the limitations of tools that still support certain languages are the reason why rule-based algorithms are more suitable. One of the studies that carried out rule-based extraction of Indonesian corpus entities was the automatic extraction of the place, date and number of victims of tropical diseases from web pages [10]. The result of the entity from this extraction process can then be further processed, for example by conducting a classification process to group categories from sentences or texts.

Web classification by categorizing the label that has been determined is one step of the web mining process [11]. Several studies on web classification have been carried out, including comparing feature selections and learning methods based on the Naive Bayes algorithm, K-nearest neighbor, C4.5, and FURIA [11], using rule-based classification to detect phishing web URLs [12], classification of academic Webometric website links to find links between links [13], news classification on Twitter using SVM [14] and optimization of web classification using Firefly algorithm based on Naïve Bayes [15].

Some studies that have focused on the proposed corpus or dataset have been done before. An Ubuntu Dialogue Corpus has been introduced which has 1 million dialogues, 7 million sayings and 100 million words [16]. The approaches used are Term Frequency and Inverse Document Frequency (TF-IDF), Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM). The three algorithms are applied to the training corpus to produce a response determination model automatically without manual labeling. Benchmark is done on the three algorithms and produces an evaluation matrix where the LSTM algorithm has the best recall, which is 87.8% for 1 in 2 Recall@ 1,60.4% for 1 in 10 Recall@1, 74.5% for 1 in 10 Recall@2, and 92.6% for 1 in 10 Recall@5. The dataset and code used are publicly published on GitHub.

In another study, it was designed and annotated a Violent Scene Detection (VSD) corpus, a dataset containing detection of physical violence in Hollywood films [17]. Some definitions of the category of physical violence are discussed in detail that are used to determine which films have elements of physical violence, where the results are published as public datasets. The dataset is also analyzed in detail about the relationship between audio and visual that describes the scene of violence. The evaluation process is done by using the Benchmark Multimedia MediaEval framework. Measuring to produce the best value of MAP@100 in 2012 reached 65%, when compared to MAP@100 in 2011 it only reached around 40%.

Several applications of classification on social media to produce datasets have also been carried out before, including the classification of tweets on Twitter social media based on two English languages and Roman-Urdu [18]. Classification contains the classification of tweets that contain political elements. The datasets collected were 89,701 tweets, with the classification results of 26.4% being political tweets in Roman-Urdu language and 73.6% being political tweets in English. From the results of the evaluation conducted, the accuracy of the classification reached 93% with a F-measure level of 97%.

Larger Corpus or datasets have been studied to be able to analyze and evaluate an event detection on social media Twitter [19]. The number of tweets studied reached 120 million with a span of 4 weeks. The event definition was proposed based on the characteristics of Twitter tweets and made some relevance judgment for event detection. The approach is done by using event detection which has been proposed and added from Wikipedia Current Events Portal to produce several events. The results obtained include 150 thousand relevance judgments with 500 events.

Rule-based extraction methods have been carried out to obtain information on the place, date and number of victims of disease in the tropics [10]. The approach is done using the Indonesian word structure and corpus taken from online news portals. From the results of the evaluation, the rule-based extraction algorithm has an accuracy of 99.8%. For classification using SVM algorithm with an accuracy of 82%, 96.41%, and 93.38% for determining the place, date and number of victims of the sentence. In addition, some other work that compares between classification methods shows that SVM has a high accuracy when compared to other methods [20], [21]. The use of the SVM method for plant leaf disease detection also provides accurate results [22].

Research involving traffic accident cases has been investigated previously for the classification of large

imbalance data in order to be able to predict traffic accidents [23]. The imbalance data in question is an imbalance between data relating to traffic accident information and those that have nothing to do, which are corrected using the sampling method. The steps taken are preprocessing data, making training data to be done over sampling, clustering using the K-means algorithm, and classification using logistic regression. Data analysis using the help of Hadoop Framework and MapReduce. The results obtained showed precision of 80.56% and 42.71%.

3. PROPOSED METHOD

The stages of can be illustrated in Figure 1 with the following steps: 1) Formulating a problem about how to create a corpus or dataset of traffic accident information itself from Twitter and make a classification model using FastText weighting and SVM, KNN algorithm, and Naïve Bayes; 2) Conducting literature reviews and literature studies; 3) Crawl data from the Twitter API and save it to a dataset; 4) Perform word embedding on tweets and then use SVM, KNN, and Naïve Bayes classifiers and best results are selected 6) Make predictions and save results to a dataset; 7) Evaluate and make conclusions based on the process that has been done.

Tools used in this study are Python 3.6.9 with sklearn libraries for machine learning, gensim for word embedding, pandas for dataframe management, numpy for matrix management, pymysql for database management, tweepy for connections to Twitter, nltk and Sastrawi for text preprocessing. The other devices used are Cloudlinux Based Shared Hosting for storing data in MySQL and running cronjob Twitter automation process with specifications 1 CPU Core Intel Xeon Gold and 256MB RAM, and Google Colab Cloud Notebook platform with 12.7 GB TPU (Tensor Processing Unit) RAM, 107 GB space. All scripts are written in Python 3.6.9.

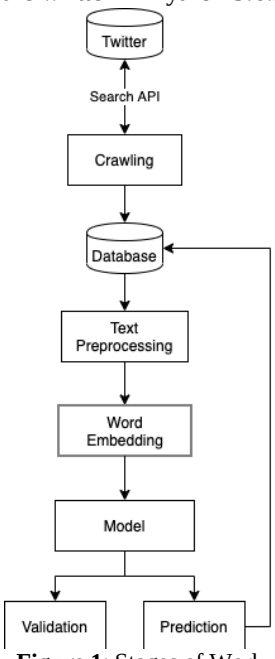

Figure 1: Stages of Work

3.1 Crawling from Twitter Social Media

Twitter is a text-based social media that has a feature to send messages up to a maximum of 280 characters [24]. This social media is streamed, meaning the message is shaped like a timeline that flows so the information is current. There are several ways to get information from Twitter, one of which is using the Application Programming Interface (API) provided on the Twitter Developer page. From the several API categories that have been provided, one method that can be utilized is the Search API [25]. Endpoint addresses and details of using the Search API can be seen in Table 1.

Table 1: The Detail of Search API

Key	Value
URL endpoint	https://api.twitter.com/1.1/search/tweets.json
Method	GET
Response formats	JSON
Authentication	Yes, OAuth based
Rate limited	Yes
Requests / 15-min window	180 (user auth) and 450 (app auth)
Parameter	q, geocode, lang, locale, result_type, count, until, since_id, max_id, include_entities

The search process uses the App Auth credential so that it can make 450 requests in 15 minutes. The automation process is carried out using linux cronjob and the script is run every 15 minutes. The search keyword used is '"*kecelakaan*"-filter: retweets AND -filter: replies'. The retweet and replies filters are used to disable retweet and reply results. If we do not use these filters, the data generated will contain a lot of duplication.

Because the data that has been stored does not have a class at all, manual labeling is needed as the initial process of data that will be used for the training process. Therefore 1000 data were taken from the database to do the training and modeling process sampling. The rest of the data will be used for the prediction process after the training process is complete and produces sufficient accuracy.

3.2 Doing Text Preprocessing

a. Case Folding

This is done using the help of the built in String library from Python, which is lower() so that uppercase letters will be lowercase.

b. Filtering

In filtering, several steps are performed to remove unneeded characters, such as emoticons, website addresses, numbers, punctuation marks, double spaces and new lines. The process is still done by using the built in String library and the re library from Python.

c. Stemming

The Indonesian Stemming process is carried out using the Sastrawi library which was built based on Nazief and Andriani's Algorithm which proved to be quite good in handling the Indonesian language stemming process [26],

[27], because the NLTK library that is used for the English Stemming process does not support Indonesian yet.

d. Tokenizing

The Tokenizing process uses the NLTK library to break sentences into lists with a space character separator.

e. Stop Word Removal

The Stop Word Removal process uses the NLTK library and also the Sastrawi library, because the NLTK library already supports the Indonesian Language corpus. Using two libraries will strengthen the stop word removal process because they will complement each other's shortcomings.

3.3 Word Embedding with FastText

The FastText word embedding process is carried out using the pre-trained Indonesian language models provided, which are available from the website address of FastText [28]. This pretrained model is trained based on Common Crawl and Wikipedia using FastText and CBOW algorithm with position-weights, dimension 300, ngram length 5, window size 5 and negative 10. The size of the cc.id.300.vec.gz file is 1 GB compressed and 4 GB extracted.

3.4 Training Using the SVM, KNN, and Naive Bayes Algorithms

Support Vector Machine (SVM) is one method to classify using supervised learning data. Each data is formulated with $Xi \in Rd$, i = 1,2, ..., l with l is a lot of data. The label is formulated with $yi \in \{-1, +1\}$. -1 is a negative class and +1 is a positive class. Dimension d which separates perfectly the two classes is formulated as wx + b = 0. Data xi will be classified as -1 if it satisfies the equation $wx + b \leq -1$ and conversely the data xi will belong to +1 if it satisfies the equation $wx + b \geq +1$. The most optimal margin can be obtained from the maximum distance between the separator with the closest pattern, formulated with 1/||w|| where ||w|| is normalization of weight vector.

In addition to SVM, the K-Nearest Neighbor (KNN) method is also applied to classify. The first step is to determine the K value as the closest number of neighbors. After that, the Euclidian distance is calculated as equation (3).

$$D(a,b) = \sqrt{\sum_{k=1}^{d}(a_k - b_k)^2} \quad (3)$$

where D is a scalar distance from a to b, a is training data, b is testing data, d is the data dimension, and k is the attribute value.

The application of the Naïve Bayes classification method is done by applying the Bayes theorem which is formulated by equation (4).

$$P(X|Y) = \frac{P(Y|X)\,P(X)}{P(Y)} \quad (4)$$

where X is the hypothesis, Y is the undefined data class, P (X) is the probability of the hypothesis, P (Y) is the probability of Y, P (X|Y) is the probability of the hypothesis based on condition Y, and P (Y|X) is the probability Y when the condition of hypothesis X. During training, the calculation of P (Vj) is carried out with the equation (5).

$$P(Vj) = \frac{|doc_j|}{Example} \quad (5)$$

where docj is a lot of documents that have category j and |Example| are many documents in the example used for training. Then do the calculation of P (Wk | Vj) with the equation (6).

$$P(Wk \mid Vj) = \frac{n_k+1}{n+|vocabulary|} \quad (6)$$

with nk is the frequency the word wk appears in the category Vj and |vocabulary| documents is the amount of data from training data. After that, in the classification process Vmap is calculated on data that is not yet known by the equation (7).

$$V_{map} = \underset{V_j \epsilon V}{\operatorname{argmax}} P(V_j) \prod_i P(a_i \mid V_j) \quad (7)$$

3.5 Evaluation Using K-Fold Cross Validation

Testing the accuracy of text classification can be done using k-fold cross validation [29], [30]. Cross validation is a statistical method for evaluating learning algorithm by dividing data into two segments, one segment is used for the training stage and one segment is used to validate the model [31]. The general form of cross validation is k-fold cross validation [31]. In k-fold cross-validation, the data are divided into k subsection with relatively equal amounts of data between subsections [32]. The training and testing process is performed on k number of iterations and on each iteration, different subsection is used for the testing process, while the other k-1 subsections are used for the training process [31]. The final evaluation is the average accuracy result of each validation step k [30]. The advantages of the k-fold cross validation method are that in this method, the way data placed does not affect because each data will appear once in the test data and appear as much as k-1 times in the training data [33]. Compared to other k values, 10- fold cross validation is the value of k accepted as the most reliable method because it can provide accurate error estimation of a model of various algorithms and applications [29], [30].

The evaluation process of the training results uses the K-Fold Cross Validation method with a value of k=10. The dataset used is from a dataset of 1000 data sets. This method will divide the dataset into 10 sections and 10 times alternating positions as 90% training fold and 10% validation fold, while fitting classification algorithms on each fold 10 times.

3.6 Process Prediction Based on the Best Algorithm

From the training algorithms that have been carried out are SVM, KNN, and Naïve Bayes, the best algorithm will be selected with the highest value. The process is done the same as when the training process is done, the difference is if during the training process using data sampling that already has a class labeled 0 and 1 the results of manual labeling, then here using data that does not have a class to be able to find the

class. The preprocessing and word embedding process use the same function that has been previously defined. Only one additional process that must be carried out in this prediction process is the removal of preprocessing results that have no value or that have an empty list, because the word embedding process will produce an error when the input data is an empty list.

4. RESULT AND DISCUSSION

The data crawling process is carried out from 4 April 2019 until 28 February 2020 automatically using the Python script cronjob on the server. The total number of tweets successfully obtained from the keyword ‘"*kecelakaan*" -filter: retweets AND -filter: replies’ is 142,168 tweets and a total size of 677MB with type InnoDB and Collation utf8mb4_unicode_ci. Graph of the number of tweets per month can be seen in Figure 2 with the highest number in September 2019 totaling 19,874 tweets while the lowest number was in April 2019 of 8,301 tweets. Some of example result is shown at Table 2.

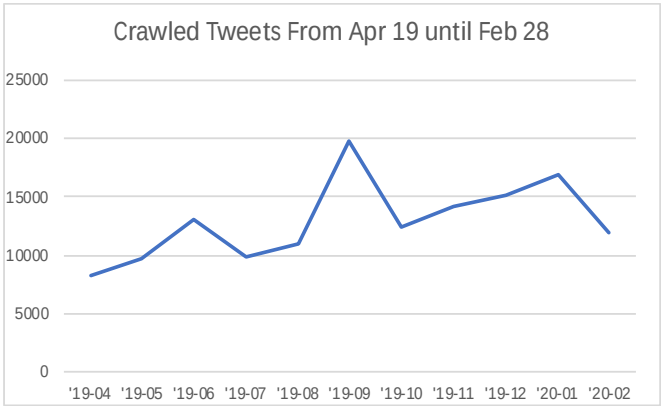

Figure 2: Monthly crawl statistics from Twitter

Table 2: Example Result of Crawl Process

Column	Example
id_str	1113812743138697216
created_at	2019-04-04 21:37:06
crawled_at	2019-04-09 12:30:13
screen_name	vtvindonesia
full_text	*Pelajar SMP Tewas Kecelakaan di Jalinsum Bandarlampung https://t.co/Tgc8D6k3m0 https://t.co/3jmozMMLv0* (Middle School Student Killed in Accident at Jalinsum Bandarlampung https://t.co/Tgc8D6k3m0 https://t.co/3jmozMMLv0)
full_tweet	{"created_at": "Thu Apr 04 14:37:06 +0000 2019", "id": 1113812743138697216, "id_str": "1113812743138697216" …….. "possibly_sensitive_appealable": false, "lang": "in"}

With a file size that contains Wikipedia and Creative Common Indonesian Language vector data that reaches 4 GB more, the process of loading this pretrained model requires a long time, which reaches 588,002113 seconds or about 9.8 minutes and RAM usage more than 6 GB, with very high specification owned by Google Colab Cloud (12.7 GB TPU (Tensor Processing Unit) RAM, 107 GB space). However this should be tolerated because this process is usually only done once during initial initialization. Figure 3 shows the process when the load pretrained model is run. Because FastText is basically the development of Word2Vec, the loading process is still compatible with the load function model for Word2Vec.

```
t_start = time.clock()
model = KeyedVectors.load_word2vec_format('cc.id.300.vec')
t_end = time.clock()
print(t_end - t_start)
```

```
/usr/local/lib/python3.6/dist-packages/smart_open/smart_oper
  'See the migration notes for details: %s' % _MIGRATION_NOT
588.002113
```

Figure 3: FastText Pretrained Model Load

Preprocessing sampling time is measured and produces 86.25220600000011 seconds. Of all the sub-processes in function, the analysis results show that the Indonesian Stemming process with the Sastrawi library takes a very long time. This is evidenced by commenting on the Stemming Literary summon line and the execution time is very significant down 1.0142250000000104 seconds. In other words, the Sastrawi Stemming process in the Python library takes more than 85.13784399999997 seconds with very high specifications owned by Google Colab Cloud. Figure 4 shows a sample of the top five data accompanied by the results of the text preprocessing that has been done. It can be seen that punctuation marks such as periods and colons have disappeared, words have become basic forms such as "*menjaga*" (guarding) becoming "*jaga*" (guard), "*kecelakaan*" (accident) become "*celaka*" (wretch) and "*muatan*" (loading) becoming "*muat*" (to load).

The word embedding of sampling data process that using FastText does not require a long time compared to the data preprocessing process. The time needed is around 0.0332260000000133 seconds. The form of vectorized text is a list with the contents of an element in the form of a float data type which is a vector of processed text, as shown in Figure 4.

0.0332260000000133

	is_accident	full_text	processed_text	vectorized_text
0	0	C.Gerakan.bicara pertolongan pertama pada kece...	[cgerakanbicara, tolong, celaka, pkbakat]	[0.019, 0.0027, 0.027800001, 0.061899997, -0.0...
1	1	Terjadi kecelakaan truk muatan besar di Tol Ku...	[celaka, truk, muat, tol, kuncir, serpong, ara...	[0.0181, 0.0013916664, 0.03533333, 0.06543333,...
2	0	Plot twist: Ibunya abis kecelakaan, nemenin ke...	[plot, twist, abis, celaka, nemenin, ugd, diti...	[0.026145456, -0.018009089, 0.03386363 8, 0.091...
3	1	Kecelakaan Maut Bus Eka di Ring Road Sragen ht...	[celaka, maut, bus, eka, ring, road, sragen]	[0.08505715, -0.0021999988, 0.042585712, 0.071...
4	0	Akibat Andra ingin selalu menjaga Emon, dia ja...	[akibat, andra, jaga, emon, dikejarkejar, tony...	[0.0065266667, -0.020466665, 0.016700001, 0.06...

Figure 3: The Result of Processed and Vectorized Text

The process of measuring training time is also measured and the results show that the time required in the SVM, KNN, Naïve Bayes method only requires a short time, which is less than 0.1 seconds, as illustrated in Figure 4. The results of the evaluation and validation of the models that have been done show that the average of the existing models has a pretty good accuracy and validation that is above 75%. Of the three methods, SVM method with Linear kernel has the best results, which is 85%. In other words, the Linear SVM method chosen to be tried is applied to the rest of the data to find out whether the class includes traffic accident information or not.

	classifier	cross_validation	time_elapsed
0	Nearest Neighbors	0.8245833333333333	0.068451
1	Linear SVM	0.85	0.095783
2	Naive Bayes	0.7770833333333333	0.022975

Figure 4: The Result of Training Process

Like the preprocessing process in training data, predictions of data also produce the same performance which both require a long time 63.89657000000011 seconds. The processed text results also show the desired shape that is in accordance with the basic word and the loss of the stopword so that it is clean and ready to be transformed into a vector. The process of word embedding data shows a relatively short time even though given a larger data, which is 0.014255000000048312 seconds. Vectorized text is the conversion of vector shapes from processed text whose the source data comes from a pretrained FastText model that has been loaded into memory.

0.03456000000005588

	id_str	full_text	processed_text	vectorized_text	predicted_text
0	1226524405309808642	Jalan-jalan seharian sama Cica, Cica doyannya ...	[jalanjalan, cica, cica, doyan, luk, luk, mulu...	[0.038515385, -0.028084617, -0.008238464, 0.03...	0.0
1	1143012258202525696	KECELAKAAN LALU LINTAS TERJADI DI TOL CIPALI, ...	[celaka, lintas, tol, cipali, orang, tinggal, ...	[0.056428574, -0.013971428, 0.04628572, 0.0963...	1.0
2	1161906379109441537	Kecelakaan Maut Terjadi di Kilometer 84 Pulau ...	[celaka, maut, kilometer, pulau, gadang, linta...	[0.002574999, -0.028600005, 0.026237499, 0.076...	1.0
3	1210412942925021189	Dishub Sumsel Serahkan Kecelakaan Bus Pada KNK...	[dishub, sumsel, serah, celaka, bus, knkt]	[0.030360002, -0.013940001, -0.0063199997, 0.0...	1.0
4	1165085286943997953	Pernah ngerawat cowo gua yg abis kecelakaan, s...	[ngerawat, cowo, gua, yg, abis, celaka, smpe, ...	[0.05789334, -0.044139996, 0.028913338, 0.1338...	0.0

Figure 5: Predicted Result of Data with Accident Class

Column	Example
id_str	1207958137782669312
full_text	16.35 WIB #Tol_Japek Halim KM 02 - Jatiwaringin KM 04 PADAT, kepadatan volume lalin. Cikarang Pusat Jalur Bawah KM 37 - KM 39 arah Cikampek PADAT, ada Perbaikan jalan di lajur 3/kanan. Karawang Barat KM 46 - KM 49 arah Cikampek PADAT, ada Penanganan kecelakaan di lajur 1/kiri.
processed_text	‘wib', 'toljapek', 'halim', 'km', 'jatiwaringin', 'km', 'padat', 'padat', 'volume', 'lalin', 'cikarang', 'pusat', 'jalur', 'km', 'km', 'arah', 'cikampek', 'padat', 'jalan', 'lajur', 'kanan', 'karawang', 'barat', 'km', 'km', 'arah', 'cikampek', 'padat', 'tangan', 'celaka', 'lajur', 'kiri'
vectorized_text	array([2.49935500e-02, 1.40967802e-03, 6.02483824e-02, -3.32290344e-02, -1.29096825e-02, 2.25064568e-02, -5.33645079e-02, -3.12935486e-02, -5.91225736e-02, -2.72354838e-02, -1.25161288e-02, 1.88258067e-02, -2.65967809e-02, -4.66967784e-02, 7.97451660e-02, -5.05838618e-02, 3.44548412e-02, 8.43870919e-03, 3.56806554e-02, ……., 3.09967790e-02], dtype=float32)
predicted_text	1.0

Table 3: Final Result that Inserted Into Dataset

The final process of the series of stages that have been passed is the program can label itself independently to what category a sentence is, whether including traffic accident information or not, just the word "*kecelakaan*" (accident) but not related to traffic accident information. Figure 5 shows the sentence prediction process based on the Linear SVM model chosen because of its highest level of accuracy. From the five sample data displayed, all of them showed the correct results, namely the traffic accident information in the 1st, 2nd and 3rd index sentences and not the traffic accident information in the 0th and 4th index sentences. The prediction process also requires a short time 0.03456000000005588 seconds. Furthermore, the prediction data is sent back to the database to complete the corpus or dataset. This process will be repeated continuously because of the automatic cronjob scheduler that runs the process of crawling, text preprocessing, word embedding, and predicting so that the corpus or dataset will get richer and more data.

5. CONCLUSION AND FUTURE WORK

Corpus or dataset of traffic accident information has been formed automatically, starting from the crawling, preprocessing and then word embedding process to the prediction of the class in each tweet. SVM Linear Method was chosen to be the best method between KNN and Naïve Bayes with 85% accuracy and proved to be implemented on both existing and future Twitter raw data with high accuracy results. This method can be used to predict the results of crawling data from Twitter social media and has been applied to the dataset that has never been done before. Since April 2019, there are 142,168 data of crawling result, still continue to grow automatically and can be used for further research on https://www.dodyagung.com/dataset/accident.

For future work, this dataset can be obtained and used as a corpus for next research, such as automatic identification of location and time of traffic accident using Natural Language Processing (NLP), or use of other classification methods that are more optimal and accurate, as well as weighting methods that have better performance. It is necessary to do hyperparameter or tuning optimization of the methods that have been used, so that the same method can produce better performance and accuracy. In the other hand, mixing with data sources other than Twitter is also recommended, such as from the police or other authorized institutions to enrich variations of corpus or dataset. Based on the corpus or dataset that has been formed can be explored of which other potential information can be analyzed that can be processed so it can produce new contributions.

REFERENCES

1. Korps Lalu Lintas Kepolisian RI, **Grafik Fatalitas Kecelakaan - Statistika Laka**, 2018. [Online]. Available: http://korlantas.polri.go.id/statistik-2/. [Accessed: 28-Oct-2018].
2. Badan Pusat Statistik, **Jumlah Kecelakaan, Koban Mati, Luka Berat, Luka Ringan, dan Kerugian Materi yang Diderita Tahun 1992-2018**, 2018. [Online]. Available: https://www.bps.go.id/linkTableDinamis/view/id/1134. [Accessed: 28-Oct-2018].
3. Twitter, **Twitter Q1 2019 Earnings Report**, 2019.
4. R. kumar, A. Jain, and C. Agrawal, **Survey of Web Crawling Algorithms**, *Adv. Vis. Comput. An Int. J.*, vol. 3, no. 3, pp. 1–7, Sep. 2016, doi: 10.5121/avc.2016.3301.
5. RSS Advisory Board, **RSS Best Practices Profile**. [Online]. Available: http://www.rssboard.org/rss-profile. [Accessed:

28-Oct-2018].

6. D. A. Adeniyi, Z. Wei, and Y. Yongquan, **Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method**, *Appl. Comput. Informatics*, vol. 12, no. 1, pp. 90–108, 2016, doi: 10.1016/j.aci.2014.10.001.
7. C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, **The Stanford CoreNLP Natural Language Processing Toolkit**, in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014, pp. 55–60, doi: 10.3115/v1/P14-5010.
8. J. R. Finkel, T. Grenager, and C. Manning, **Incorporating non-local information into information extraction systems by Gibbs sampling**, in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05*, 2005, pp. 363–370, doi: 10.3115/1219840.1219885.
9. The Apache Software Foundation, **Apache OpenNLP**. [Online]. Available: https://opennlp.apache.org/. [Accessed: 30-Oct-2018].
10. T. F. Abidin, R. Ferdhiana, and H. Kamil, **Automatic extraction of place entities and sentences containing the date and number of victims of tropical disease incidence from the web**, *J. Emerg. Technol. Web Intell.*, vol. 5, no. 3, pp. 302–309, 2013, doi: 10.4304/jetwi.5.3.302-309.
11. A. Onan, **Classifier and feature set ensembles for web page classification**, *J. Inf. Sci.*, vol. 42, no. 2, pp. 150–165, 2016, doi: 10.1177/0165551515591724.
12. L. McCluskey, F. Thabtah, and R. M. Mohammad, **Intelligent rule-based phishing websites classification**, *IET Inf. Secur.*, vol. 8, no. 3, pp. 153–160, 2014, doi: 10.1049/iet-ifs.2013.0202.
13. P. Kenekayoro, K. Buckley, and M. Thelwall, **Automatic classification of academic web page types**, *Scientometrics*, vol. 101, no. 2, pp. 1015–1026, 2014, doi: 10.1007/s11192-014-1292-9.
14. I. Dilrukshi, K. De Zoysa, and A. Caldera, **Twitter news classification using SVM**, *Proc. 8th Int. Conf. Comput. Sci. Educ. ICCSE 2013*, no. Iccse, pp. 287–291, 2013, doi: 10.1109/ICCSE.2013.6553926.
15. K. Bhatt, **An Improved Optimized Web Page Classification using Firefly Algorithm with NB Classifier (WPCNB)**, *Int. J. Comput. Appl.*, vol. 146, no. 4, pp. 975–8887, 2016, doi: 10.5120/ijca2016910668.
16. R. Lowe, N. Pow, I. Serban, and J. Pineau, **The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems**, in *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2015, pp. 285–294, doi: 10.18653/v1/W15-4640.
17. C.-H. Demarty, C. Penet, M. Soleymani, and G. Gravier, **VSD, a public dataset for the detection of violent scenes in movies: design, annotation, analysis and evaluation**, *Multimed. Tools Appl.*, vol. 74, no. 17, pp. 7379–7404, Sep. 2015, doi: 10.1007/s11042-014-1984-4.
18. H. Afzal and I. Javed, **Creation of Bi-lingual Social Network Dataset Using Classifiers**, in *Machine Learning and Data Mining in Pattern Recognition: 10th International Conference*, 2013, vol. 7988, no. March, doi: 10.1007/978-3-642-39712-7.
19. A. J. McMinn, Y. Moshfeghi, and J. M. Jose, **Building a large-scale corpus for evaluating event detection on twitter**, in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13*, 2013, pp. 409–418, doi: 10.1145/2505515.2505695.
20. H. khafajeh, **Opinion Mining: How efficient are Online classification tools?**, *Int. J. Emerg. Trends Eng. Res.*, vol. 8, no. 2, pp. 557–567, Feb. 2020, doi: 10.30534/ijeter/2020/46822020.
21. D. A.J, **Evaluating the performance metrics of different machine learning classifiers by combined feature extraction method in Alzheimer's disease detection**, *Int. J. Emerg. Trends Eng. Res.*, vol. 7, no. 11, pp. 652–658, Nov. 2019, doi: 10.30534/ijeter/2019/397112019.
22. N. K. Gattim, **Plant Leaf Disease Detection Using SVM Technique**, *Int. J. Emerg. Trends Eng. Res.*, vol. 7, no. 11, pp. 634–637, Nov. 2019, doi: 10.30534/ijeter/2019/367112019.
23. S. H. Park and Y. G. Ha, **Large Imbalance Data Classification Based on MapReduce for Traffic Accident Prediction**, in *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2014, pp. 45–49, doi: 10.1109/IMIS.2014.6.
24. Wikipedia, **Twitter - Wikipedia**. [Online]. Available: https://en.wikipedia.org/wiki/Twitter. [Accessed: 04-Nov-2018].
25. Twitter, **Standard search API — Twitter Developers**. [Online]. Available: https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets. [Accessed: 04-Nov-2018].
26. M. Widjaja and S. Hansun, **Implementation of porter's modified stemming algorithm in an Indonesian word error detection plugin application**, *Int. J. Technol.*, 2015, doi: 10.14716/ijtech.v6i2.456.
27. A. Nugraha, **Fiturebot: CS Chatbot Application using Nazief-Adriani Algorithm**, *Int. J. Emerg. Trends Eng. Res.*, vol. 8, no. 2, pp. 350–354, Feb. 2020, doi: 10.30534/ijeter/2020/18822020.
28. Facebook, **Word vectors for 157 languages**. .
29. M. Koppel, S. Argamon, and A. R. Shimoni, **Automatically Categorizing Written Texts by Author Gender**, *Lit. Linguist. Comput.*, vol. 17, no. 4, pp. 401–412, 2002, doi: 10.1093/llc/17.4.401.
30. S. Karimi, J. Yin, and J. Baum, **Evaluation methods for statistically dependent text**, *Comput. Linguist.*, vol. 41, no. 3, pp. 539–548, Sep. 2015, doi:

10.1162/COLI_a_00230.

31. K. A. Ross *et al.*, **Cross-Validation**, in *Encyclopedia of Database Systems*, Boston, MA: Springer US, 2009, pp. 532–538.
32. R. Kohavi, **A study of cross-validation and bootstrap for accuracy estimation and model selection**, *Proc. 14th Int. Jt. Conf. Artif. Intell. - Vol. 2*, vol. 2, no. 0, pp. 1137–1143, 1995.
33. K. Polat and S. Güneş, **Classification of epileptiform EEG using a hybrid system based on decision tree classifier and fast Fourier transform**, *Appl. Math. Comput.*, 2007, doi: 10.1016/j.amc.2006.09.022.