Emotion Classification on Youtube Comments using Word Embedding

Julio Savigny
School of Electrical and Informatics Engineering
Institut Teknologi Bandung
Bandung, Indonesia
jsavigny@gmail.com

Ayu Purwarianti
School of Electrical and Informatics Engineering
Institut Teknologi Bandung
Bandung, Indonesia
ayu@stei.itb.ac.id

***Abstract*— Youtube is one of the most popular video sharing platform in Indonesia. A person can react to a video by commenting on the video. A comment may contain an emotion that can be identified automatically. In this study, we conducted experiments on emotion classification on Indonesian Youtube comments. A corpus containing 8,115 Youtube comments is collected and manually labelled using 6 basic emotion label (happy, sad, angry, surprised, disgust, fear) and one neutral label. Word embedding is a popular technique in NLP, and have been used in many classification tasks. Word embedding is a representation of a word, not a document, and there are many methods to use word embedding in a text classification task. Here, we compared many methods for using word embedding in a classification task, namely average word vector, average word vector with TF-IDF, paragraph vector, and by using Convolutional Neural Network (CNN) algorithm. We also study the effect of the parameters used to train the word embedding. We compare the performance of the classification with a baseline, which was previously state-of-the-art, SVM with Unigram TF-IDF. The experiments showed that the best performance is achieved by using word embedding with CNN method with accuracy of 76.2%, which is an improvement from the baseline.**

Keywords—comments; classification; word embedding; CNN; emotion;

I. INTRODUCTION

Youtube is rapidly growing and is one of the most visited site in Indonesia [1]. Many people rely on Youtube to make money as content creators. Content creators need to cater their audiences, one of the ways is to read the reaction in the comments. The comments may contain emotion that can be detected automatically. Then, content creators can gain insight by knowing the emotion reactions in their videos and using that insight to create their future videos more wisely.

To classify the emotion automatically, a text classification system can be build using text processing techniques. The text classification approach is to build classifiers from labeled texts [2], and be considered as a supervised classification task. The technique can be described as a statistical or machine-learning approach. In a machine-learning approach, aside from the learning algorithms, the feature representation of a text is also affecting the performance of the classifier. Usually, the text is represented as a bag-of-words [3], where each document is represented by the occurrences of words, which can be furthermore weighted by using the term frequency and inverse document frequency (TF-IDF). In this paper, we will study further another type of document representation using word embedding.

Word embedding have been popular in the NLP community, mainly because it can capture semantics of the words [4]. This unique feature leads to its usage in many text classification tasks, such as sentiment analysis [5], election classification [6], etc. The usage of word embedding as a feature for emotion classification is rare and is never used for Indonesian language.

However, word embedding is a representation of a word, not a document. To represent a text document as a word embedding feature for a classifier, there are various methods that can be used, such as averaging the word vectors [6], using paragraph vector method [7], and also by using Convolutional Neural Network [8]. So, a research is needed to be done to determine the best method for using word embedding in emotion classification task on Youtube comments.

II. RELATED WORKS

Several studies have been conducted on emotion classification in text [9][10]. Many methods are used for different emotion classification tasks and different corpora. Word embeddings are mainly used on tweets, not Youtube comments and rarely implemented for a fine-grained text classification, such as emotion classification.

There is a research on emotion classification for Youtube comments in Thai language [10]. Six basic emotion category are used. The data used are Youtube comments collected from videos with advertisement (AD) and music video (MV) category. Unigram with TF-IDF weighting are used as features. The researcher then conducted experiments using different classifier algorithm, namely SVM, Multinomial Naïve Bayes, and Decision Tree. Best accuracy is achieved with Multinomial Naïve Bayes for MV dataset (84.48%) and SVM for AD dataset (76.14%).

Another research in emotion classification task is by Atmadja & Purwarianti [9]. The data used in this research are 7,622 tweets in Bahasa Indonesia. In this research, rule based and statistical based classification were compared. In the rule based method, Synesketch algorithm is used and word lexicon such as WordNet-Affect is utilized. In the statistical based

978-1-5386-3001-3/17/$31.00 ©2017 IEEE.

method, the research compared n-grams with TF-IDF weighting (unigram, bigram, trigram) features. The features are also selected further with information gain and minimum frequency algorithm to reduce the dimensionality. Various preprocessing techniques are also implemented and compared. Comparation between various learning algorithm, SVM, Naïve Bayes, and Decision Tree is also conducted. The best accuracy is 83.156%, obtained by using statistical method with SVM algorithm and unigram with TF-IDF features.

Another studies have also been conducted on using word embedding in text classification tasks. Yang, et al. conducted a research on using word embedding on twitter election classification task [6]. The dataset is sampled from collected tweets about Venezuela parliamentary election, and then manually labelled as "Election-related" and "Not Election-related". The word embedding models are then constructed by using Word2Vec implementation on twitter data and Wikipedia data. The word embedding models are constructed by differing various parameters, namely context window sizes *W*, and dimension sizes *D*. Then the word embedding models are passed to a CNN architecture. To compare the classifiers, baselines were used. The baselines are Random Classifier, SVM with TF-IDF features, and SVM with word embedding (averaging the word vectors). The best performance is acquired by the CNN model using word embedding with $D = 800$ and $W = 5$, which resulted in f1-score of 0.771.

III. Word Embedding Methods

Word embeddings are the representation of words. To represent a Youtube comments as a word embedding feature, we can use various methods. In this study, we will use four different methods to represent a Youtube comment as a word embedding feature, which are:

- Average Word Vector, word vector of each word in the comment is averaged along each dimension to construct the feature vectors of the comment, as in (1). Where $\boldsymbol{D}$ is the document vector of the comment, $\boldsymbol{W}_i$ is the word vector of the word i, and N is the number of words in the comment.

$$\boldsymbol{D} = \frac{\sum_{i=1}^{N} \boldsymbol{W}_i}{N} \quad (1)$$

- Average Word Vector with TF-IDF, word vector of each word in the comment is multiplied by TF-IDF value of the word and then averaged along each dimension to construct the feature vectors of the comment, as in (2). Where $\boldsymbol{D}$ is the document vector of the comment, $\boldsymbol{W}_i$ is the word vector of the word i, and N is the number of words in the comment.

$$\boldsymbol{D} = \frac{\sum_{i=1}^{N} \boldsymbol{W}_i \times TF\text{-}IDF(\boldsymbol{W}_i)}{N} \quad (2)$$

- Paragraph Vector, using Doc2Vec [7]. This method directly learns the vector representation of a document.
- Convolutional Neural Network (CNN). The comment is represented as an array of word vectors, or matrix, of each word inside the comment. The matrix is in the size of $S \times D$ where *S* is the length of the comment and *D* is the dimensionality of the word vector. The matrix then passed to a CNN architecture proposed by Kim Y. [8] as an embedding layer for a convolution process.

IV. Experimental Setup

A. Dataset

The data is collected from 10 Youtube videos from various genre by using Youtube v3 API. The 8,115 comments collected then stored and manually labeled by 2 people, and the third person as a tie-breaker if there is a tie in the labels. The annotators were given forms containing all comments need to be labelled, and then label each comment separately (independent from the other annotator). The labels used are 6 basic emotions (happy, sad, surprised, disgust, fear, angry) proposed by Paul Ekman [11], and one neutral label for comments that doesn't contain any emotion. Table I shows the label's distribution over the data collected.

TABLE I. Distribution of Data

Label	Total Comments
Happy	1689
Sad	571
Surprised	427
Disgust	270
Fear	654
Angry	846
Neutral	3658

In addition to the labelled data, another 43,151 Youtube comments are collected to be used as a corpus to train the word embedding models. The data used to train the word embedding doesn't need to be labelled.

B. Preprocess

The data collected are not in a formal form. There are many irregularities in the data. So, we preprocess the data first. The preprocess is implemented mainly using python programming language, by doing some regular expression matchings.

Various preprocessing techniques are employed, such as:

- Base processing, contains casefolding (lowercase conversion), removal of punctuation, and converting url.
- Emoticon Conversion, emoticons in the text are converted to a word that represent the emotion.
- Number removal, numbers are assumed to be irrelevant for emotion classification. Therefore, every number in the data is removed.
- Slang Dictionary, informal / slang form of a word is converted to a formal form using a slang dictionary. This process is implemented using InaNLP [15].
- Duplicate character removal, Indonesian Youtube comments usually contain mistyped and non-standard word that have more than one letter and is converted to one letter e.g. "kerreeeen" is converted to "keren".
- One character removal, a word that only contain one character is removed.

C. Baseline

To evaluate the various word embedding models, we use a baseline, which obtain the best performance in previous research [9]. Support Vector Machine is used as the classifier and Unigram with TF-IDF weighting scheme is used as the features.

Experiment on feature selection using Information Gain is also conducted. The feature selection experiment is conducted to study the effect of the feature size to the performance. Feature selection is also reducing the dimensionality of the feature.

D. Word Embedding

In this study, there are several implementations that is used to train word embedding, namely Word2Vec [12], Doc2Vec [7], and GloVe [13]. For each of the implementations, we generate a set of word embedding models by fine tuning the parameters. In Word2Vec, the parameters and its values are Architecture A = {skip-gram, continuous bag-of-words}, context windows size W = {5, 7, 9, 11, 13, 15}, dimensionality D = {100, 200, 300, 400, 500}, negative sampling N = {5, 10, 15, 20, 25, 30}, and iteration I = {5, 10, 15, 20, 25, 30}. For Doc2Vec, the parameters and its values are Architecture A = {distributed memory, distributed bag-of-words}, context windows size W = {5, 7, 9, 11, 13, 15}, dimensionality D = {100, 200, 300, 400, 500}, negative sampling N = {5, 10, 15, 20, 25, 30}, and iteration I = {5, 10, 15, 20, 25, 30}. For GloVe, the parameters and its values are context windows size W = {5, 7, 9, 11, 13, 15}, dimensionality D = {100, 200, 300, 400, 500}, and iteration I = {5, 10, 15, 20, 25, 30}.

Aside from the parameter tuning on the word embedding models, experiment on the word embedding methods is also conducted. There are four methods, Average Word Vector, Average Word Vector with TF-IDF, Paragraph Vector, and Convolutional Neural Network. The parameters of the CNN are based on the previous research [8].

V. Experimental Results

A. Baseline

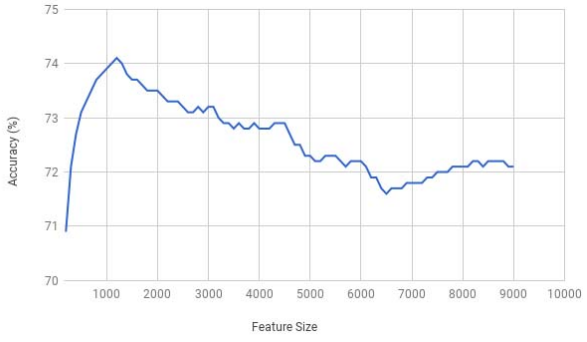

Fig. 1. Experimental Result in Feature Selection using Information Gain

The features used are unigram with TF-IDF. The vocabulary count of the corpus data is about 9000, so to reduce features and select the most important ones we use the range of the features from 200-9000, sorted by each feature's Information Gain, and then validated using 10-fold cross validation. The algorithm used are Support Vector Machine.

Fig. 1 shows the effect of the features selected to the accuracy. We can see that accuracy is increased up until ~1200 most important features are selected, and generally decreased after that. It can be assumed that there are many words or features that are not important and does not discriminate between the labels well. The accuracy with 1200 features is about 74.1% and it is set as the baseline accuracy.

B. Word Embedding

To tune the parameters of the word embedding models, we construct the models, and then use SVM and 10-fold cross validation to validate. The word embedding method used for this is average word vector, except for Doc2Vec model, which use paragraph vector method. For each parameter we set the other control parameters statically, then varying the parameter to be tuned. The best parameter then used for tuning the subsequent parameter. For the Architecture, in Word2Vec, skip-gram is generally gives the better performance, with about ~3% better accuracy. In Doc2Vec, the distributed bag-of-words (DBOW) is better, with ~1.6% better accuracy.

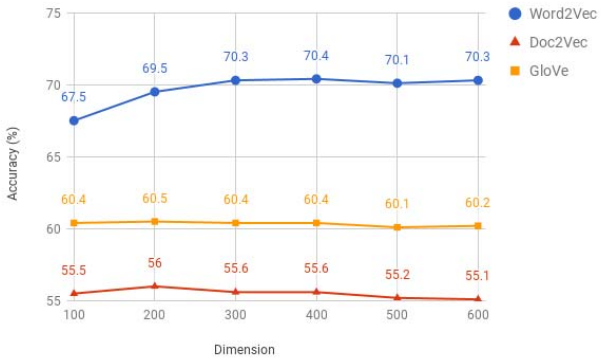

Fig. 2. Effect of dimensionality on accuracy.

For the effect of dimensionality D, we can see in the Fig.2. There it is shown that in Word2Vec, generally, the bigger the size of the dimension, the better the accuracy. The best size of dimensionality in Word2Vec is about 400. In the Doc2Vec and GloVe however, the same behaviour cannot be stated. The accuracy is increased up until D = 200, then it slightly declines.

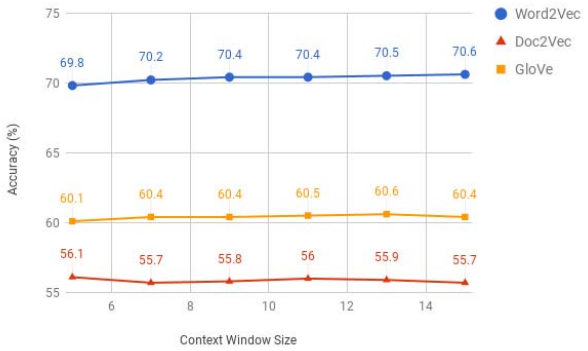

Fig. 3. Effect of context window size on accuracy.

Fig.3. Illustrate the effect of context window size *W* on accuracy. We can see that, generally, context window has insignificant effect on the accuracy. In Word2Vec, accuracy is generally better the larger the context window size, and is generally flat on other models.

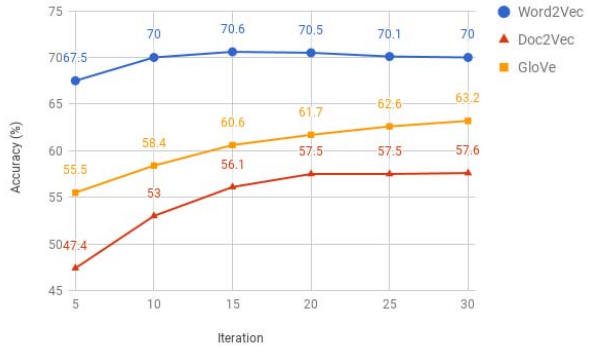

Fig. 4. Effect of iteration on accuracy.

For the effect of iteration *I* on the accuracy, we can see in the Fig.4. It is shown that generally, the larger the iteration, the better the accuracy. In Word2Vec, the improvement seems to stabilize when the *I* = 15. In Doc2Vec, it seems to stabilize when *I* = 20, altough *I* = 30 resulted in better accuracy. In GloVe, the iteration has not yet stabilized when *I* = 30, and if time permits, it's better to increment the iteration further.

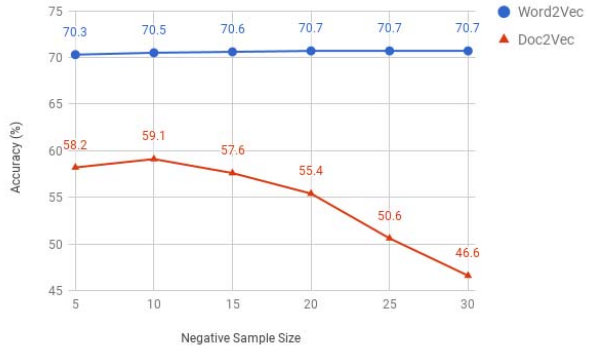

Fig. 5. Effect of negative sample on accuracy.

In Fig.5 we can see the effect of negative sample *N* on the accuracy of Word2Vec and Doc2Vec models. In the Word2Vec model, the effect of negative sample is negligible, although the accuracy is increased slightly when the negative sample is bigger. In Doc2Vec however, the negative sample size has a negative correlation with the accuracy. The accuracy is increased until *N* = 10, then decreases.

After the parameter tuning is done, we set the parameter with the best value that gives the best accuracy. For Word2Vec, the Architecture *A* = skip-gram, dimensionality *D* = 400, context window size *W* = 15, iteration *I* = 15, negative sample size *N* = 20. For Doc2Vec, the Architecture *A* = dbow, dimensionality D = 200, context window size *W* = 15, iteration *I* = 30, negative sample size *N* = 10. For GloVe, the parameters is dimensionality *D* = 200, context window size *W* = 13, and iteration *I* = 30. Generally speaking, Word2Vec gives a much better performance than GloVe with the average vector method, so we will use only Word2Vec and Doc2Vec for subsequent experiments.

After setting the parameters of the word embedding, we then proceed to compare the methods of representing the comments as embedding features. We build the features using four different methods as stated before and then use 10-fold cross validation to validate. SVM is used to train and classify the Average Vector, Average Vector with TF-IDF, and Paragraph Vector methods, and CNN algorithm is used for the CNN method.

TABLE II. ACCURACY OF WORD EMBEDDING METHODS

Method	Accuracy
Average Vector	70.7%
Average Vector with TF-IDF	69.5%
Paragraph Vector	59.1%
CNN	73.4%

In Table II we can see that the best accuracy is obtained by the CNN method. The accuracy of all the methods still can't beat the accuracy that was given by the baseline. In an attempt to increase accuracy, a feature selection experiment is conducted. In the input data, only words that is included in the 1200 best words from the feature selection experiment in baseline are considered.

TABLE III. ACCURACY OF WORD EMBEDDING METHODS WITH FEATURE SELECTION

Method	Accuracy
Average Vector	71,5%
Average Vector with TF-IDF	71,1%
Paragraph Vector	59,5%
CNN	76,2%

Table III shows the result of the feature selection on the accuracy of various methods. We can see that the accuracy are better for every method, compared with the accuracy without feature selection. One of the method, the CNN method, gives a better accuracy, 76.2%, compared to 74.1% of baseline accuracy. Still, there are rooms for improvements. The CNN methods works better for this task because

We analyzed that from the error rate, some of the errors are caused by some ambiguous emotion in the comments. Also, the lack of negation handling, such as "tidak sedih" is classified as sad because it contains kata "sedih" but doesn't consider the negation context. Unoptimized preprocess technique is also causing errors, some word like "waokoawkokwaok" can be considered as a laughing, thus happy emotion, but is not converted to its basic form "tertawa". Also, angry comments, which usually typed in all capitalized form, can be misclassified

as a neutral one because of the casefolding in the preprocess technique.

VI. Conclusion And Future Works

From the experiments that have been done, we may conclude that by using word embedding on emotion classification task on Youtube comment can increase the accuracy of the classification. The best method to use word embedding for the classification is by using Convolutional Neural Network algorithm. The accuracy obtained by using word embedding with CNN is 76.2%, which is better than using SVM and unigram with TF-IDF features. CNN method is also gives a better accuracy than other word embedding methods. This could be explained by the fact that the neural network approach can capture more features, such as the sequence of the words inside a comment (by convoluting the word matrix, it is dependent of sequence of the words), than the others, which is just naïve representation of word vectors (couldn't capture the word orders). For the word embedding models, to obtain the best result, it is recommended to fine-tune the parameters for every classification task and dataset.

There are still room for improvements, and this research can be developed further. In future works, one can handle the negation in the text, and also add another feature, such as capital letter feature. Another emotion category, such as dimensional model, can also be used. Also, the word embedding model can be improved by using a much larger corpus to train. The labelled data can also be larger and more balanced to give a better result. The preprocessing techniques can also be optimized. Furthermore, the CNN method can also be combined with SVM algorithm. Rather than using the softmax layer as the classifier for the features generated by CNN, SVM also can be used.

References

[1] SimilarWeb Ltd. (2016, November 1). *Top Websites in Indonesia - SimilarWeb Website Ranking*. Retrieved from Similar Web: https://www.similarweb.com/top-websites/indonesia

[2] Danisman, T., & Alpkocak, A. (2008). Feeler: Emotion Classification of Text Using Vector Space Model. *AISB 2008 Convention Communication, Interaction and Social Intelligence*, 53..

[3] Joachims, T. (2002). Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms. Norwell: Kluwer Academic Publishers.

[4] T. Mikolov, W.T. Yih, G. Zweig. Linguistic Regularities in Continuous Space Word Representations.

[5] Yin, Y., & Jin, Z. (2015). Document Sentiment Classification based on the Word Embedding. 4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering.

[6] Yang, X., Macdonald, C., & Ounis, I. (2016). Using Word Embeddings in Twitter Election Classification. *CoRR*.

[7] Le, Q., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents. *ICML*.

[8] Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *CoRR*.

[9] Atmadja, A. R., & Purwarianti, A. (2015). Comparison on the rule based method and statistical based method on emotion classification for Indonesian Twitter text. *2015 International Conference on Information Technology Systems and Innovation (ICITSI)*.

[10] Sarakit, P., Theeramunkong, T., & Haruechaiyasak, C. (2015). Classifying emotion in Thai youtube comments. *Information and Communication Technology for Embedded Systems (IC-ICTES), 2015 6th International Conference of*. IEEE.

[11] Ekman, P. (1972). Universals and Cultural Differences in Facial Expressions of Emotions. *Nebraska Symposium on Motivation*, 207-282.

[12] Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *NIPS*.

[13] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *EMNLP*, (pp. 1532-1543).

[14] Purwarianti, A., Andhika, A., Wicaksono, A. F., & Afif, I. (2016). InaNLP: Indonesia natural language processing toolkit, case study: Complaint tweet classification. 2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA), 1-5.