Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 5, No. 4, November 2020, Pp. 317-324

Attention-based CNN-BiLSTM for dialect identification on Javanese text

Ahmad Fathan Hidayatullah*[1], Siwi Cahyaningtyas[2], Rheza Daffa Pamungkas[3]
Department of Informatics, Universitas Islam Indoneisa, Indonesia[1,2,3]

Article Info

Keywords:
Attention Mechanism, CNN, BiLSTM, Dialect Identification, Deep Learning

Article history:
Received 26 September 2020
Accepted 18 November 2020
Published 30 November 2020

Cite:
Hidayatullah, A., Cahyaningtyas, S., & Pamungkas, R. (2020). Attention-based CNN-BiLSTM for dialect identification on Javanese text. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 5(4). doi:https://doi.org/10.22219/kinetik.v5i4.1121

*Corresponding author.
Ahmad Fathan Hidayatullah
E-mail address:
fathan@uii.ac.id

Abstract

This study proposes a hybrid deep learning models called attention-based CNN-BiLSTM (ACBiL) for dialect identification on Javanese text. Our ACBiL model comprises of input layer, convolution layer, max pooling layer, batch normalization layer, bidirectional LSTM layer, attention layer, fully connected layer and softmax layer. In the attention layer, we applied a hierarchical attention networks using word and sentence level attention to observe the level of importance from the content. As comparison, we also experimented with other several classical machine learning and deep learning approaches. Among the classical machine learning, the Linear Regression with unigram achieved the best performance with average accuracy of 0.9647. In addition, our observation with the deep learning models outperformed the traditional machine learning models significantly. Our experiments showed that the ACBiL architecture achieved the best performance among the other deep learning methods with the accuracy of 0.9944.

1. Introduction

Javanese is one of the languages of the Austronesian family which is the most widely spoken regional language in Indonesia [1]. However, not all Javanese speakers speak the same dialect because Javanese language has several dialect variations in which each dialect represents specific characteristics of certain tribes and regions [2]. Javanese used in different regions may differ in its pronunciation, tone and vocabulary. In this case, there are several factors that cause dialects to differ from one another, such as including geographical, ethnic and social class conditions [1]. Geographically, Javanese is used in everyday conversation by people who live in Central and East Java. Moreover, there are three main different dialects spoken by people in Central and East Java, namely the standard dialect, Banyumas, and East Java. The standard Javanese dialect comes from people who live in Yogyakarta, Solo and their surrounding areas. The Banyumas dialect, or better known as Basa Ngapak, is spoken by people in the western part of Central Java Province, including, Banyumas Regency, Pekalongan Regency, Tegal Regency, Kebumen Regency and the western part of Kedu Regency. The East Javanese dialect covers most of East Java, except Banyuwangi [1].

Based on the explanation above, this study aims to automatically identify Javanese dialects into three classification classes, namely standard Javanese dialects, Javanese dialects, and East Javanese dialects. Dialect identification is a process of identifying dialects of a particular segment of sound or text in any shape and size automatically [3]. Automatic dialect identification on text is a challenging task when compared with language identification, since there are various languages that have closeness and similarity with each other [4].

A number of studies in the field of dialect identification have attracted many researchers. Altamimi and Teahan [5] identified Arabic dialect on Twitter using Prediction by Partial Matching approach. This study proposed both character and word level n-gram as features. Moreover, they also compared three different machine learning algorithms, namely Multinomial Naïve Bayes, LibSVM and K-Nearest Neaighbour (KNN). Fares, et al [3] proposed a deep learning and hybrid frequency based features for Arabic dialectical identification. They built a multiple neural network model using word and document representations. Their experiments showed that the combination between character TF-IDF, word TF-IDF and neural network achieved the best performance with the dev-set F1-score of 66.6%. Xu, et al [6] presented a dialect identification for Chinese language. Their study revealed that feature based PMI (Pointwise Mutual Information) and word alignment were shown the best achievement for Chinese dialect identification. Yang and Xiang [7] also performed Chinese dialect identification. Theiy found that Multinomial Naive Bayes with n-gram feature and Bidirectional LSTM with word embedding have very good performance to discriminate Mandarin dialect varieties. Ali [8] proposed a character-level CNN to distinguish four German dialects. The study found that adding GRU with recurrent layer as an embedding layer before convolutional layer achieved the best performance to classify German dialects. Criscuolo and Aluisio [9] proposed character n-gram and word-level CNN to distinguish several closely-related languages.

The dialect identification task in this study is performed by treating it as a text or document classification task [10]. Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) are the most common deep learning methods for text classification tasks [7]. Other deep learning methods, such as Long-Short Term Memory (LSTM), Bidirectional LSTM (BiLSTM), combination between CNN and LSTM, CNN and BiLSTM have also applied in text classification [11][7]. Currently, with the popularity of attention mechanism [12][13], researchers are proposing attention mechanism to be added in their deep learning models on text classification problems. Wang, et al [14] applied attention-based LSTM for aspect-based sentiment classification. Yang, et al [15] proposed a hierarchical attention network to be applied on document classification. Li, et al [16] proposed a self-attention mechanism on Bidirectional LSTM for sentiment classification. Liu and Guo [17] presented a Bidirectional LSTM and CNN with attention mechanism for text classification.

Based on the success of the implementation of the deep learning and attention mechanism methods, we propose a hybrid deep learning model called attention-based CNN-Bidirectional LSTM (ACBiL) to automatically identify dialect on Javanese text. In this study an attention mechanism is proposed in our CNN-BiLSTM architecture. There are two reasons behind the adding of this attention mechanism. First, we add attention mechanism to handle both long-range and local dependencies [18]. The long-range dependency shows the relationship and gap between related terms or words in a sequence of text [19]. The local dependency represents the syntactic dependency between terms. Second, attention mechanism has the ability to compute faster with less number of parameters than RNN [18][20]. As comparison, we also experiment with other several classical machine learning and deep learning approaches. We follow the previous study that experimented with three different traditional machine learning approaches based on bag of words n-gram features as baseline such as, Multinomial Naïve Bayes (MNB), Support Vector Machine (SVM), and Logistic Regression (LR) [7]. In addition, the term frequency-inverse document frequency (TF-IDF) weighting features are proposed as well for those three methods respectively. As for deep learning methods, we test other several models, including, RNN, GRU, CNN, LSTM, BiLSTM and CNN-BiLSTM.

2. Research Method

2.1 Datasets and Preprocessing

We collected our datasets Twitter using Twitter API and tweepy library provided by Python. The datasets were categorized into three classes, Standard Javanese, East Javanese and Ngapak Javanese. We collected a total of 16500 sentences and we made our dataset balanced with 5500 data for each class. The datasets were labelled manually by 3 persons who understand three Javanese dialects. As for preprocessing, we perform a standard preprocessing tasks for Twitter text data that have been conducted in our previous study [21]. The preprocessing tasks are including removing punctuations, removing non-ASCII characters, removing URLs, removing digits or numbers from string, removing white spaces and lowercase.

2.2 Classical Machine Learning Models

The most popular method for dialect identification task is classical machine learning models based on feature engineering [7]. This study uses three classical machine learning models as baseline, including Multinomial Naïve Bayes (MNB), Support Vector Machine (SVM) and Logistic Regression (LR). We propose n-gram language model and TF-IDF as our feature for the respective traditional machine learning methods. N-gram language model represents *N* sequence of words and then assign probabilities for each word sequences. In this work, we apply both unigram (1-gram) and bigram (2-gram) language model for training the model. TF-IDF is a statistical approach to measure the weight of each term in a corpus by multiplying term frequency and inverse document frequency.

2.3 Deep Learning Models

2.3.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) was built to deal with images and computer vision [8]. Currently, CNN has been applied in natural language processing tasks and it has given promising result [22]. Typically, CNN consists of three layers, such as convolutional, pooling and fully connected layers. In CNN building blocks, convolutional and pooling layers are the first two layers which perform feature extraction, while the fully connected layer maps the extracted features into the final output as classification result [23].

2.3.2 Recurrent Neural Network (RNN)

RNN is a neural network method that was proposed to deal with sequential data [24]. In RNN, the task for each part of an order is repeated, with the output of the current state is depending on the preceding computations. RNN saves information about what has been computed in a memory. The basic formula of RNN is shown by,

$$h_t = f(h_{t-1}, x_t) \tag{1}$$

According to the Equation 1, the current hidden state (h_t) is obtained from a function of the precedent hidden state $h_{(t-1)}$ and the present input (x_t). The variable (h_t) is a latent variable that stores the sequence information.

2.3.3 Long-Short Term Memory (LSTM)

LSTM is generally used to solve the sequence data such as text and proven excellent performance in text classification problem [17]. In LSTM cell state, the information will be removed or added which is regulated by three gates, namely forget gate, input gate and output gate. All of the processes through those three gates are illustrated by the following composite functions.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{2}$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \tag{3}$$

$$c_t = f_t c_{t-1} + i_t \, tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{4}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \tag{5}$$

$$h_t = o_t tanh\,(c_t) \tag{6}$$

Where σ is the sigmoid function, *i, f, o and c* are respectively represent the input gate, forget gate, output gate and cell activation vectors [25]. *W* and *b* respectively denote weight matrices and bias vectors. Equation 2 demonstrates the input gate i_t with its corresponding weight matrix W_{xi}, W_{hi}, W_{ci} and bias vector b_i. Equation 3 illustrates a forget gate f_t with its corresponding weight matrix W_{xf}, W_{hf}, W_{cf} and bias vector b_f. Equation 5 represents the output gate o_t with its corresponding weight matrix W_{xo}, W_{ho}, W_{co} and bias vector b_o. The c_t in Equation 4 stands for the current state which is generated by computing the weighted sum from both previous cell state and current information generated by the cell [25]. Finally, h_t in Equation 6 represents the hidden layer state at time *t.*

2.3.4 Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) is one of the RNN variant architecture which aims to solve the disappearing gradient problem in a standard RNN. It simplifies the LSTM gates into two gates, update and reset gate which are computed by the Equation 7 and Equation 8 [26]. The update gate (z_t) decides how much of previous information needs to be proceed in the future. The reset gate (r_t) is responsible about how much information to forget from the past information. The variable $\hat{h}_t$ in Equation 9 and h_t in Equation 10 represent the candidate activation and output vector respectively.

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{7}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{8}$$

$$\hat{h}_t = tanh(W_h x_t + U_h(r_t \, h_{t-1}) + b_h) \tag{9}$$

$$h_t = (1 - z_t)h_{t-1} + z_t \hat{h}_t \tag{10}$$

2.3.5 Bidirectional LSTM (BiLSTM)

BiLSTM is an improvement of LSTM networks which aims to enhance the model performance on sequence classification problems. Different from LSTM networks, BiLSTM networks train the input sequence using two LSTM networks. BiLSTM combines between two hidden layers, forward and backward hidden layers [17]. By employing both forward and backward hidden layers in BiLSTM, both preceding and subsequent contexts can be captured. As shown by the Equation 11, Equation 12, and Equation 13, BiLSTM calculates the forward hidden sequence $\vec{h}$, backward hidden sequence $h^{\leftarrow}$ and then combined to compute the output sequence y. This calculation is conducted by iterating the backward layer from t=*T* to 1, the forward layer from t=1 to *T* and updating the output layer, thus:

$$\vec{h}_t = H(W_{x\vec{h}}x_t + W_{\vec{h}\vec{h}}\vec{h}_{t-1} + b_{\vec{h}}) \tag{11}$$

$$h^{\leftarrow}{}_t = H(W_{xh^{\leftarrow}}x_t + W_{h^{\leftarrow}h^{\leftarrow}}h^{\leftarrow}{}_{t-1} + b_{h^{\leftarrow}}) \tag{12}$$

$$y_t = W_{\vec{h}y}\vec{h}_t + W_{h^{\leftarrow}y}h^{\leftarrow}{}_t + b_y \tag{13}$$

2.3.6 CNN-BiLSTM Models

In this study, we added dense and dropout layer after max pooling layer for CNN-BiLSTM models. Zhou, et al [27] introduced a combination CNN-BiLSTM model in order to obtain two things, local features of phrases and global information of sentences. In this CNN-BiLSTM model, CNN takes out a higher level sequence representation from word embedding sequences.

2.3.7 Attention-based CNN-BiLSTM (ACBiL)

The basic concept of the ACBiL method is almost similar with the one proposed by Liu and Guo [17]. There are some differences between our proposed architecture and the previous architecture proposed by [17]. First, we did not use pre-trained word vectors to construct word embedding. Instead of using pre-trained word embedding, we put embedding layer in our model to build our word embedding as n-dimensional dense vector. Second, we only employed one attention layer in our architecture, whereas the previous research used two attention layers. Finally, as for the optimization algorithm, we applied Nadam (Nesterov-accelerated Adaptive Moment Estimation) optimizer which combines Adam and NAG [28]. The proposed attention-based CNN-BiLSTM architecture is described below:

1. ***Input layer****:* The input layer is an embedding layer which receives the sentences from dataset in a form of word sequences to the model architecture. The embedding layer aims to capture the semantic information of the sentence. This embedding layer receives n word sequence (w) and these will be represented as a vector in a form of d dimensional word embedding. In this layer, the word sequence (w) is mapped into a sequence vector (x). Each sequence vector (x_w) is a real-valued vector $(X_w \in R^d)$. The length of dimensional word embedding is obtained from the maximum sequence length in our dataset.
2. ***Convolutional layer****:* We apply one-dimension convolutional layer which creates a convolutional kernel to be convolved on a single spatial or temporal dimension to produce a tensor of outputs. In our CNN model, we put a one dimension (1D) convolutional layer in the top of embedding layer. In this 1D convolution operations, the input sequence vector (x) will be mapped into a hidden sequence (h). Let $x_{i:i+w-1}$ represents a concatenation of $x_i, x_{i+1}, \dots, x_{i+j}$, a window of words $x_{i:i+w-1}$ can produce a feature (c_i) by using Equation 14.

   $$c_i = f(k \cdot x_{i:i+w-1} + b) \tag{14}$$

   Where f represents a non-linear function and $b \in \mathbb{R}$ represents a bias term. A new feature (c_i) is created by applying a filter $k \in \mathbb{R}^{w_d}$ to a window of concatenated word embedding with the size w. The final result of feature map is presented in Equation 15.

   $$c = [c_1, c_2, \dots, c_{n-h+1}] \tag{15}$$

   Here $c \in \mathbb{R}^{n-h+1}$, n and h represent the number of filters and the dimensionality of the hidden sequence respectively. In this layer, we apply Rectifier Linear Unit (ReLU) as a non-linear function after every convolution process.
3. ***Max pooling layer****:* We employ 1D max pooling to be applied to the connected output of the multiple convolution c. Max pooling layer performs a down-sampling towards the input representation by getting the maximum value over the window as the extracted feature map. The process of taking the maximum values is performed by converting a sequence c into a single hidden vector c', thus $c' = \max\{c\}$.
4. ***Batch normalization layer****:* Batch normalization layer is added on the top of one-dimension max pooling layer to normalize the inputs of the previous layer.
5. ***Bidirectional LSTM layer****:* The BiLSTM permits the networks to obtain both backward and forward information regarding the sequence at every step. In BiLSTM, sequence long-term dependencies are being captured through extending a standard recurrent neural network using both forward and backward memory states. To avoid overfitting and improve our model performance, we add three regularization parameters, such as kernel regularizer, recurrent regularizer and bias regularizer. Those three regularizations are utilized to apply a penalty during optimization process on the kernel weights matrix, recurrent kernel weights matrix and bias vector respectively.
6. ***Attention layer****:* Attention layer generates a weight vector and perform multiplication to transform a word-level features combination from every time step into a sentence-level feature vector. This work follows the architecture of the Hierarchical Attention Network (HAN) proposed by Yang, et al [15]. The HAN comprises of three components, such as: (i) word encoder, (ii) word-level attention, (iii) sentence encoder and (iv) sentence-level attention. In this study, we perform dialect identification as a document-level classification. We make an assumption that a document

consists of L sentences s_i and there are T_i words in every sentence. The words in the i^{th} sentence is represented as w_{it} with $t \in [1, T]$.

(i) **Word Encoder**. A bidirectional GRU [12] summarizes information to obtain annotation of words and integrate the contextual information in the annotation. The sentence is read from two directions with the bidirectional GRU.

(ii) **Word-level Attention.** In word-level attention, the most informative words that have important contribution in a sentence meaning are extracted by using attention mechanism. This process is computed as,

$$\mathrm{u_{it}} = \tanh\,(\mathrm{W_w h_{it}} + \mathrm{b_w}) \tag{16}$$

$$\alpha_{\mathrm{it}} = \frac{\exp\,(\mathrm{u_{it}^T u_w})}{\sum_{\mathrm{t}} \exp\,(\,\mathrm{u_{it}^T u_w})} \tag{17}$$

$$\mathrm{s_i} = \sum_{\mathrm{t}} \alpha_{\mathrm{it}} \mathrm{h_{it}} \tag{18}$$

In Equation 16, u_{it} is a hidden representation of the word annotation h_{it}. As shown by the Equation 17, the word significance is measured by calculating similarity between u_{it} with a term level context vector u_w and obtain a regularized weight α_{it} using softmax function. In Equation 18, the sentence vector s_i is calculated as a weighted sum of the multiplication between the weight α_{it} and word annotation h_{it}.

(iii) **Sentence Encoder**. Similar with word encoder, bidirectional GRU is applied as well to encode the sentences.

(iv) **Sentence-level Attention.** Almost similar with word-level attention, attention mechanism is utilized and we make use of a context vector u_s in sentence level to gauge the significance of the sentences. In a document, the summarization of all the details of sentences is represented in a document vector v. This calculation is presented in the Equation 19, Equation 20, and Equation 21.

$$u_i = \tanh\,(W_s h_i + b_s) \tag{19}$$

$$\alpha_i = \frac{\exp\,(u_i^T u_s)}{\sum_i \exp\,(\,u_i^T u_s)} \tag{20}$$

$$v = \sum_i \alpha_i h_i \tag{21}$$

7. ***Dense layer**:* Dense layer reshapes the tensor result to have the shape that equals to the number of elements in the tensor.
8. ***Softmax layer**:* This layer predicts the probability distribution for each classes by simply squashing values into a specified range. We put a dense layer with softmax activation function which act as output and prediction layer. The softmax layer is chosen because of its ability on handling multiple classes problem. Suppose that we have the input vector x_i and k output label, the softmax activation function is defined in the Equation 22.

$$softmax(x_i) = \frac{\exp\,(x_i)}{\sum_{j=1}^{k} \exp\,(x_i)} \tag{22}$$

3. Result and Discussion

The dialect classication taks in this work were applied by following the previous study by [29] which modelled as a sentence classification task. Our training process for both traditional machine learning and deep learning models were evaluated using accuracy metrics and implementing k-fold cross validation with the k value of 5. We used Scikit-learn[i] to implement classification tasks with the classical machine learning. We performed three processes, including count vectorizer, tf-idf transformer and classifier. The count vectorizer performs tokenization and build a vocabulary from a collection of texts and encode the new text documents with the vocabulary. N-gram range is used for considering n-values for different word n-grams. We experiemented two different n-gram range, (1,1) means only unigram and (1,2) means consider both unigram and bigram. Tf-idf transformer aims to transform a word frequency matrix into a normalized term frequency or term frequency-inverse document frequency representation. In this task, we applied smoth inverse document frequency weights by adding one to document frequencies to avoid division by zero. In the classifier process, we employed Multinomial Naïve Bayes, Support Vector Machine and Logistic Regression. Table 1 shows the result of the classical machine learning models. From the result, we found tha considering bigram feature did not produce significant result in increasing the accuracy of the model. Among those three traditional methods, applying bigram feature only worked better on Multinomial Naïve Bayes algorithm. Generally, we can see that the Linear Regression models outperformed the other traditional machine learning methods with the average accuracy of 0.963 and 0.9647. Compared with other traditional machine learning methods, the Linear Regression using unigram achieved the best accuracy of 0.9647.

Table 1. The Accuracy of Traditional Machine Learning Models

Traditional Machine Learning Methods	Avg. Accuracy
MNB + Bigram	0.9342
MNB + Unigram	0.9301
SVM + Bigram	0.9471
SVM + Unigram	0.9493
LR + Bigram	0.963
LR + Unigram	**0.9647**

To build the deep learning models, we utilized Keras[ii] deep learning API with Tensorflow[iii] as backend. All of our deep learning models are built based on the previous framework proposed by [7]. The first component of our deep learning model is an embedding layer which functions as an input layer. The embedding layer aims to capture the semantic information of tweets data [30]. The embedding layer receives n tokens (w) and these will be represented as a vector in a form of d dimensional word embedding. We assign three different paramates in the embedding layer, including input dimension, input length and output dimension. The input dimension of our embedding layer is the size of the vocabulary which is the maximum integer index plus one. The output dimension represents the output vector size for every word. The value for the output dimension here is 100. The length of our input is assigned by the maximum sequence length in our dataset.

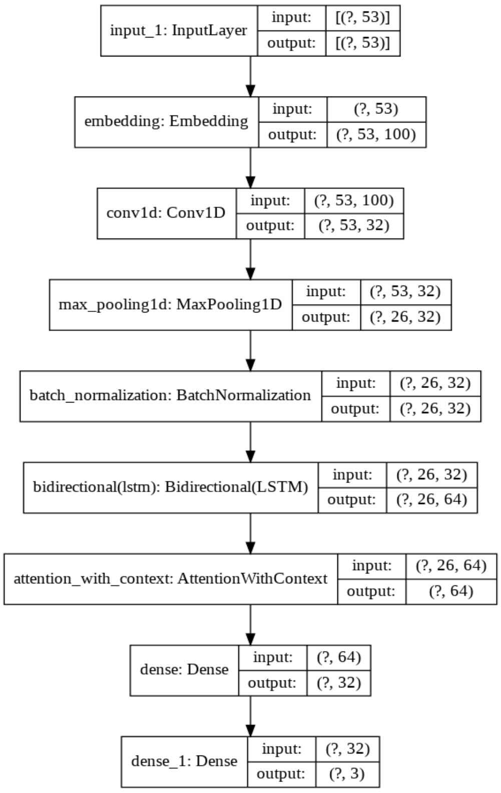

Figure 1. Attention-based CNN-BiLSTM Architecture

In our model, we modified the previous architecture [7] by adding more layers after the embedding layer. We added dense layer with 32-unit dimension and ReLU as the activation function. Moreover, we applied a regularization technique to avoid overfitting in our neural network architecture by adding a dropout layer [31] on the top of dense layer with the dropout rate of 0.2. The next component is sentence encoder layer which is determined by several deep learning models. The sentence encoder layer receives the output from embedding layer then produce a high level vector representation (v). This encoder layer aims to encode the sentences into fixed-size vectors representation. In this layer, we experimented with several different deep learning models, such as RNN, GRU, LSTM, BiLSTM. For RNN, GRU, LSTM and BiLSTM model, the dimensionality of the output space is equal to 32-unit.

In our CNN and hybrid CNN-BiLSTM model, we put a one-dimension convolutional layer in the top of embedding layer. We assigned this 1D convolutional layer with 32 filters, 3 kernel size, 1 strides and ReLu as the activation function. On the top of the convolution layer, a one-dimension max pooling layer was attached with pool size of 2. On the next layer, we put a dense layer with the hidden units of 32 and ReLU as the activation function. After that, we added dropout layer with the dropout rate of 0.2. For the CNN model, we attached a flatten layer before the output dense layer to flatten the feature map into a column. Ultimately, we added a dense layer as the output layer with softmax function.

Figure 1 shows the architecture of our attention-based CNN-BiLSTM (ACBiL). We used the same parameters that were applied in the other deep learning models for the embedding layer, convolution layer and max pooling layer. A batch normalization layer was added to enhance the training performance. After that, we attached a BiLSTM layer with 32 fiters and we applied three different L2 regularization processes on it, including kernel regularizer, recurrent regularizer and bias regularier with the value of 0.01 respectively. We then added a hierarchical attention networks on the top of BiLSTM layer to focus on obtaining the most significant word feature from sentences. In the next layer, we put a fully connected layer with the dimension unit of 32. Finally, a softmax layer was employed as a prediction layer by producing probability distribution for the respective category.

The experiment result of our deep learning models are listed in Table 2. Overall, the result of the deep learning models outperformed the traditional machine learning models significantly. The accuracy result for our deep learning models achieved in around 0.99xx. The accuracy achieved from the RNN was the lowest among the other deep learning methods, which is 0.99. The LSTM reached better performance than BiLSTM with accuracy of 0.9919. CNN-BiLSTM, GRU and CNN achieved an accuracy of 0.9935, 0.9936 and 0.9939 respectively. Finally, we found that our ACBiL performed slightly better than among the other deep learning models with the accuracy of 0.9944.

Table 2. The Accuracy of Deep Learning Models

Deep Learning Methods	Avg. Accuracy
RNN	0.99
GRU	0.9936
LSTM	0.9919
BiLSTM	0.9915
CNN	0.9939
CNN-BiLSTM	0.9935
Attention-based CNN-BiLSTM (ACBiL)	**0.9944**

4. Conclusion

In this study, a deep learning model called ACBiL (Attention-based CNN-BiLSTM) for Javanese text dialect identification is proposed. As comparison, we also experimented with the traditional machine learning and other deep learning methods. For the traditional machine learning, we observed with TF-IDF and n-gram feature. We found that the Linear Regression with unigram performed better that the other traditional machine learning methods with the average accuracy of 0.9647. In addition, our observation with the deep learning models outperformed the traditional machine learning models significantly. Finally, the experiments showed that our ACBiL achieved slightly better performance than the other deep learning methods with the accuracy of 0.9944.

5. Acknowledgement

The authors would like to thank to the Department of Informatics, Universitas Islam Indonesia for the technical and financial support.

References

[1] A. I. Fauzi and D. Puspitorini, “Dialect and Identity: A Case Study of Javanese Use in WhatsApp and Line,” *IOP Conference Series: Earth and Environmental Science*, vol. 175, p. 012111, Jul. 2018. https://doi.org/10.1088/1755-1315/175/1/012111

[2] A. M. Warohma, P. Kurniasari, S. Dwijayanti, Irmawan, and B. Y. Suprapto, “Identification of Regional Dialects Using Mel Frequency Cepstral Coefficients (MFCCs) and Neural Network,” in *2018 International Seminar on Application for Technology of Information and Communication*, Sep. 2018, pp. 522–527. https://doi.org/10.1109/ISEMANTIC.2018.8549731

[3] Y. Fares *et al.*, “Arabic Dialect Identification with Deep Learning and Hybrid Frequency Based Features,” in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, Florence, Italy, 2019, pp. 224–228. http://dx.doi.org/10.18653/v1/W19-4626

[4] T. Jauhiainen, K. Lindén, and H. Jauhiainen, “Language model adaptation for language and dialect identification of text,” *Nat. Lang. Eng.*, vol. 25, no. 5, pp. 561–583, Sep. 2019. https://doi.org/10.1017/S135132491900038X

[5] M. Altamimi and W. J. Teahan, “Arabic Dialect Identification of Twitter Text Using PPM Compression,” pp. 13, 2019.

[6] F. Xu, M. Wang, and M. Li, “Sentence-Level Dialects Identification in the Greater China Region,” *International Journal on Natural Language Computing*, vol. 5, no. 6, pp. 9–20, Dec. 2016. https://doi.org/10.5121/ijnlc.2016.5602

[7] L. Yang and Y. Xiang, “Naive Bayes and BiLSTM Ensemble for Discriminating between Mainland and Taiwan Variation of Mandarin Chinese,” in *Proceedings of VarDial*, Minneapolis, MN, 2019, pp. 120–127. http://dx.doi.org/10.18653/v1/W19-1412

[8] M. Ali, “Character Level Convolutional Neural Network for German Dialect Identification,” in *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, Aug. 2018, pp. 172–177.

[9] M. Criscuolo and S. M. Aluisio, “Discriminating between Similar Languages with Word-level Convolutional Neural Networks,” in *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain, 2017, pp. 124–130. http://dx.doi.org/10.18653/v1/W17-1215

[10] Ç. Çöltekin, T. Rama, and V. Blaschke, “Tübingen-Oslo Team at the VarDial 2018 Evaluation Campaign: An Analysis of N-gram Features in Language Variety Identification,” pp. 11.

[11] M. Elaraby and A. Zahran, “A Character Level Convolutional BiLSTM for Arabic Dialect Identification,” in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, Florence, Italy, 2019, pp. 274–278. http://dx.doi.org/10.18653/v1/W19-4636

[12] D. Bahdanau, K. Cho, and Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” *arXiv:1409.0473 [cs, stat]*, 2014.

[13] M.-T. Luong, H. Pham, and C. D. Manning, “Effective Approaches to Attention-based Neural Machine Translation,” *arXiv:1508.04025 [cs]*, Sep. 2015, Accessed: Sep. 16, 2020. http://dx.doi.org/10.18653/v1/D15-1166

[14] Y. Wang, M. Huang, X. Zhu, and L. Zhao, “Attention-based LSTM for Aspect-level Sentiment Classification,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, 2016, pp. 606–615. http://dx.doi.org/10.18653/v1/D16-1058

[15] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, “Hierarchical Attention Networks for Document Classification,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California, 2016, pp. 1480–1489. http://dx.doi.org/10.18653/v1/N16-1174

[16] W. Li, F. Qi, M. Tang, and Z. Yu, “Bidirectional LSTM with self-attention mechanism and multi-channel features for sentiment classification,” *Neurocomputing*, vol. 387, pp. 63–77, Apr. 2020. https://doi.org/10.1016/j.neucom.2020.01.006

[17] G. Liu and J. Guo, “Bidirectional LSTM with attention mechanism and convolutional layer for text classification,” *Neurocomputing*, vol. 337, pp. 325–338, Apr. 2019. https://doi.org/10.1016/j.neucom.2019.01.078

[18] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, “DiSAN: Directional Self-Attention Network for RNN/CNN-Free Language Understanding,” in *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018, pp. 10. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewFile/16126/16099

[19] S. A. Chowdhury and R. Zamparelli, “RNN Simulations of Grammaticality Judgments on Long-distance Dependencies,” in *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA, Aug. 2018, pp. 133–144.

[20] Y. Liu, C. Sun, L. Lin, and X. Wang, “Learning Natural Language Inference using Bidirectional LSTM model and Inner-Attention,” *arXiv:1605.09090 [cs]*, May 2016, Accessed: Aug. 05, 2020.

[21] A. F. Hidayatullah and M. R. Ma’arif, “Pre-processing Tasks in Indonesian Twitter Messages,” *Journal of Physics: Conference Series*, vol. 801, p. 012072, Jan. 2017.https://doi.org/10.1088/1742-6596/801/1/012072

[22] X. Zhang, Zhao, Junbo, and Y. LeCun, “Character-level Convolutional Networks for Text Classification,” *dvances in neural information processing systems*, pp. 649–657, 2015.

[23] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” *Insights Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018. https://doi.org/10.1007/s13244-018-0639-9

[24] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Gated Feedback Recurrent Neural Networks,” in *International conference on machine learning*, 2015, pp. 2067–2075.

[25] A. Graves, N. Jaitly, and A. Mohamed, “Hybrid speech recognition with Deep Bidirectional LSTM,” in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, Olomouc, Czech Republic, Dec. 2013, pp. 273–278. https://doi.org/10.1109/ASRU.2013.6707742

[26] Y. Zhao, Y. Shen, and J. Yao, “Recurrent Neural Network for Text Classification with Hierarchical Multiscale Dense Connections,” in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Macao, China, Aug. 2019, pp. 5450–5456. https://doi.org/10.24963/ijcai.2019/757

[27] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau, “A C-LSTM Neural Network for Text Classification,” *arXiv:1511.08630 [cs]*, Nov. 2015, Accessed: Sep. 15, 2020.

[28] T. Dozat, “Incorporating Nesterov Momentum into Adam,” in *ICLR Workshop*, 2016, vol. 1, pp. 2013–2016.

[29] C. Guggilla, “Discrimination between Similar Languages, Varieties and Dialects using CNN- and LSTM-based Deep Neural Networks,” in *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects*, Dec. 2016, pp. 185–194.

[30] Q. Zhou and H. Wu, “NLP at IEST 2018: BiLSTM-Attention and LSTM-Attention via Soft Voting in Emotion Classification,” in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Brussels, Belgium, Oct. 2018, pp. 189–194. http://dx.doi.org/10.18653/v1/W18-6226

[31] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.