Named Entity Recognition on Indonesian Twitter Posts Using Long Short-Term Memory Networks

Valdi Rachman, Septiviana Savitri, Fithriannisa Augustianti, Rahmad Mahendra
Faculty of Computer Science
Universitas Indonesia
Depok, Indonesia
{valdi.rachman, septiviana.savitri, fithriannisa.augustianti}@ui.ac.id, rahmad.mahendra@cs.ui.ac.id

***Abstract* — The task of Named-Entity Recognition (NER) can support the higher-level tasks such as question answering, text summarization, and information retrieval. This work views NER on Indonesian Twitter posts as a sequence labeling problem using supervised machine learning approach. The architecture used is Long Short-Term Memory Networks (LSTMs), with word embedding and POS tag as the model features. As the result, our model can give a performance with an F1 score of 77.08%.**

Keywords — Named Entity Recognition, Long Short-Term Memories, Twitter, Indonesian language

I. INTRODUCTION

Named Entity Recognition (NER) is one of the sub-task from Information Extraction. NER task is firstly introduced in Message Understanding Conference-6 (MUC-6) [1]. Terminologies "named entity" originally refer to the names of people, organizations, and geographic locations that are appeared in a text. Named Entity Recognition task can support a number of Natural Language Processing (NLP), such as: question answering, semantic search, and machine translation.

Microblog message becomes more popular due to the huge use of social media. Twitter is one of the popular microblog services. Twitter provides the huge size of tweet data that can be analyzed in order to develop applications, such as sentiment analysis [2] and event detector [3]. NER system that is designed for standard language gives a worse performance for Twitter NER tagging. Tweets are usually informal and ungrammatical text. The word choice can be slang words or pieces of daily conversations that cannot be found in the dictionary.

While NER for foreign languages such as English has been extensively studied, Indonesian NER is still barely tapped, especially on the microblog messages. In this work, we use LSTM as the model architecture for solving NER on Indonesian microblog messages. The features used are word embedding, neighboring word embedding, and POS tag.

The rest of this paper is organized as follows. The related works are explained in Section 2. In Section 3, we discuss the methodology, including the model features and architectures. Our experiments and analyses are described in Section 4, followed by the conclusion and future work in Section 5.

II. RELATED WORKS

The proposed solutions for NER problem can nearly divide into three categories: the rule-based, the machine learning based, and hybrid methods [4]. The rule-based approach has been done by [5] with the NetOwl Extractor that using pattern rule-based and lexicon. The machine learning approach was used in [6] in order to solve nested named entity problem. Jansche and Abney used two-phase hybrid approach for NER problem [7]. They extracted phone number with the two-phase procedure. In the first phase, all the possible candidates were extracted using hand-crafted grammar. The second phase would process the candidates that already extracted from the first phase.

NER researches on Twitter posts for English language have been done by [4], [8] and [9] with different kind of proposed methods. [4] proposed the idea of combining two algorithms, namely K-Nearest Neighbors (KNN) for word level classification and Conditional Random Field (CRF) algorithm. Both algorithms were repeatedly trained until they reached high performance, so the approach can be classified as semi-supervised learning. The evaluation shows the overall result performed F1 of 80,2%.

With a different method, [8] proposed T-NER system by re-built the NLP pipeline started with POS tagging, chunking until named entity recognition. T-NER applied LabeledLDA in order to leverage large numbers of unlabeled data to enrich the corpus from Freebase. The result of [8], compared with Stanford NER system which is reducing error by 41%.

Further research for twitter-specific NER conducted by [9] that proposed TwiNER, a novel two steps unsupervised NER system for Twitter. The first step leveraged on the global context gathered from Wikipedia and Web N-Gram corpus to tokenize tweet into valid phrases using dynamic programming algorithm. The second step constructed a random walk model. Compared with T-NER, TwiNER increased the overall F1 about 30% for global context data.

978-1-5386-3172-0/17/$31.00

[10] conducted research on NER system for Indonesian microblog messages. They viewed NER problem as sequence labeling problem. The idea behind sequence labeling problem was that a tag for a word may be dependent to the previous tags. Thus, viewing NER as a sequence labeling problem enabled the model to consider the context and labels from previous words in order to predict the current word. They proposed various word level and orthographic features, including words, last three letters, word length, pattern function, inside the bracket, part of speech, surrounding words, lookup list, and non-standard word list. They used machine learning approach with Conditional Random Field (CRF) as the algorithm and achieved F1 scores 40,75%, 75,13%, and 56,35% for Person, Place, and Organization, respectively.

The limitation of machine learning model with traditional approach is the used of hand-crafted features which could be an exhausted work. As a solution, the deep learning approach utilized minimal feature engineering for the model. [11] only used word embedding, a dense vector representation of a word, as the only feature for the model. They view NER as a classification problem and use Convolutional Neural Networks (CNN) as the model architecture.

Another deep learning research on NER task was done by [12]. They viewed NER as a sequence labeling problem and used Bi-Directional Long Short-Term Memories (LSTM) as the main architecture. Both [11] dan [12] showed that the performance of deep learning approach with less feature engineering was competitive compared with the previous research that uses hand-crafted features.

In order to increase the performance of NER system for Indonesian language microblog message with less feature engineering, we proposed our method of NER system with deep learning approach. The architecture of this method is Long Short Term Memory (LSTM) with some features. Those features are word embedding, neighboring word embedding, and POS tag.

III. METHODOLOGY

This section explains the data as well as the features and architecture used in our experiments.

A. Data Annotation

We used the data annotated by [10], with a total of 480 tweets. We also contributed to fix the mistakes found in annotation. The data is labeled using BIO format (Begin, Inside, Other).

Table I provides the example of tweet. The tweet contains an ORGANIZATION's name (DPRD is the abbreviation of "Dewan Perwakilan Rakyat Daerah", a governor institution in Indonesia) and a LOCATION's name (Kaltim is the abbreviation of "Kalimantan Timur", it is one of Indonesia's province).

TABLE I. NAMED-ENTITY IN INDONESIAN TWEETS

Dataset	NER Format
Kasus	O
Bansos	O
Libatkan	O
Mantan	O
Anggota	O
DPRD	B-ORGANIZATION
Kaltim	B-LOCATION
,	O
Pembacaan	O
Vonis	O
Ditunda	O

*["Case","Bansos,"involved","former","member","DPRD","Kaltim","the reading of","verdict","postponed"] or "Bansos case involved former member of DPRD (Indonesian governor institution), the reading of verdict is postponed"

B. Features

In this research, the features being used are:

Word Embedding is a feature for mapping each word in a corpus into a vector form. Each unique word mapped into a unique vector. One word can retrieve a vector. The similar word will also retrieve the similar vector form. The approach used is the unsupervised learning called Skip-gram proposed by [8]. In our work, the dimension of word embedding vector is 32.

Neighboring Word Embedding is a feature that utilizes other feature that is Word Embedding. The results of Word Embedding are used as inputs for the Neighboring Word Embedding. This feature consists of one word embedding on the left as well the one on the right of the current word. If the word is located at the beginning of the sentence then the word has no word on the left so that the vector for the left is vector **0**. Conversely, if the word is located at the end of the sentence then the word has no word on the right so that the vector for the right is vector **0**.

Part of Speech (POS) Tag. We used POS Tag with three main tags, which are NNP (Proper Noun), NN (Noun), and O (Other). The rationale of using such POS tags is that named-entity is generally in a form of noun, be it a proper noun or regular noun. Thus, adding information that tells whether a word is a noun will provide the model the candidates of named-entities.

Table II provides an example of Indonesian tweet that is equipped with POS Tag information. POS Tag is then encoded

978-1-5386-3172-0/17/$31.00©

into the one-hot-vectors format in deep learning architecture. Each tag are mapped into a unique vector. NN mapped into [1,0,0], NNP mapped into [0,1,0], and O mapped into [0,0,1].

TABLE II. POS FEATURES OF INDONESIAN TWEETS

Text	Windy	membeli	buku	di	PIM
POS format	NNP	O	NN	O	NNP
ONE-HOT-VECTOR	[0,1,0]	[0,0,1]	[1,0,0]	[0,0,1]	[0,1,0]

*["Windy","buys","book","at","PIM"]

C. Architecture: Bi-Directional Long Short-Term Memories

In this work, we utilize Bi-Directional Long Short-Term Memories (BLSTM) to solve Indonesian language Twitter NER task. Using bi-directional LSTM enables the model to learn the information from the past (left) as well as the future (right). This way, the model can learn from the information from the left and right in order to predict the current word. [13] proposed the idea of stacking the BLSTM networks so that it can learn more in a higher level features.

Figure I shows the model architecture of stacked BLSTM. The input vector $\mathbf{x}_i$ is fed into the first LSTM forward layer which outputs $\mathbf{f}_i$, with i denotes the time step of the current word. At the same time, $\mathbf{x}_i$ is also processed by the first LSTM backward layer which produces $\mathbf{bi}$.

$$\mathbf{f}_i = \text{LSTM\_forward\_1}(\mathbf{x}_i, \mathbf{f}_{i-1})$$

$$\mathbf{b}_i = \text{LSTM\_backward1}(\mathbf{x}_i, \mathbf{b}_{i+1})$$

$$\mathbf{h}_i = \text{Concatenate}(\mathbf{f}_i, \mathbf{b}_i)$$

After that, $\mathbf{h}_i$ is produced by concatenating $\mathbf{f}_i$ and $\mathbf{b}_i$.

$$\mathbf{k}_i = \text{LSTM\_forward\_2}(\mathbf{h}_i, \mathbf{k}_{i-1})$$

$$\mathbf{m}_i = \text{LSTM\_backward\_2}(\mathbf{h}_i, \mathbf{m}_{i-1})$$

Same as the previous steps, $\mathbf{h}_i$ is then fed into the second LSTM forward and LSTM backward layers, producing vector $\mathbf{k}_i$ and $\mathbf{m}_i$.

$$\mathbf{p}_i = \text{Concatenate}(\mathbf{k}_i, \mathbf{m}_i)$$

$$\mathbf{label}_i = \text{Softmax}(\mathbf{p}_i)$$

Finally, $\mathbf{k}_i$ and $\mathbf{m}_i$ are concatenated to be $\mathbf{p}_i$ which is then processed by the Softmax layer to produce the output $\mathbf{label}_i$.

IV. EXPERIMENTS

This Section explains the experiment results as well as the analysis. We use 5-fold cross-validation for all of the experiments. The metrics used are precision, recall, and F1. All experiments are done by using server with GPU Nvidia Tesla M60. The server has 6 cores and RAM size of 56.00 GiB. The operating system used is Ubuntu 16.04.2.

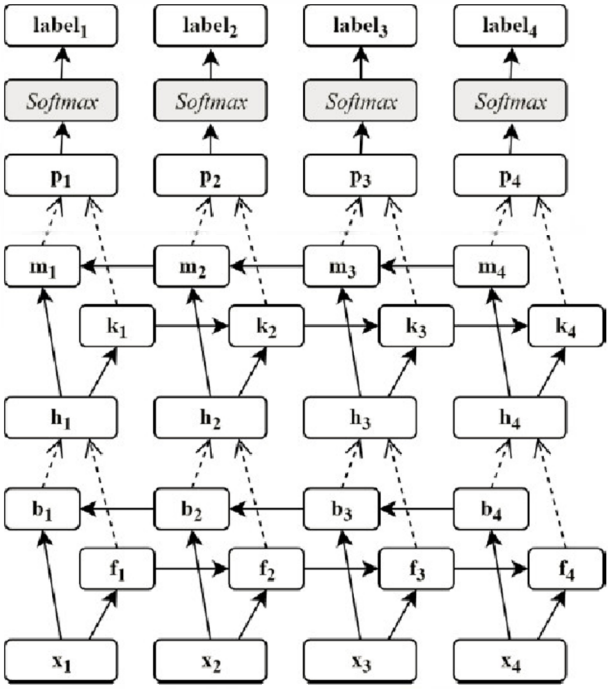

Figure 1. Architecture of stacked BLSTM

We first experiment on feature selection by comparing the performance of different feature combinations. After that, we continue to analyze the improvement that happens when a particular feature is added, including the analysis of the named-entity tags. We also explain the challenge faced by the model on predicting one of the tags.

The feature selection scenario aims to select the best feature combination for our model. The evaluation approach used is a partial match. We experiment with the following combinations:

1. Word Embedding (WE)
2. Word Embedding + Neighboring Word Embedding (WE + NWE)
3. Word Embedding + POS Tag (WE + POS)

Table III shows an example of tweet in which NER system using only feature WE mispredicted the named-entity. The misprediction is corrected when combination of WE + POS features are used. The tweet actually has two named entities: *"Jamie T"* as PERSON and *"Inggris"* as LOCATION. These two entities are not available in the training data (out-of-vocabulary case), thus making it challenging for the model to predict them correctly. In the WE model, both entities are mispredicted as O (Other), showing the incapability of the model to predict unseen entities. On the other hand, the WE + POS model correctly predicts both entities as PERSON and LOCATION, respectively. We suggest that the POS information (Noun and Proper Noun) helps the model to determine the candidate of named entities, therefore providing the model valuable information to predict them.

978-1-5386-3172-0/17/$31.00©2017 IEEE

TABLE III. EXAMPLE OF PREDICTION USING FEATURES WE AND WE + POS

Word	WE (Prediction)	WE + POS (Prediction)
Jamie	O	B-PERSON
T	O	I-PERSON
mengkover	O	O
satu	O	O
lagu	O	O
dari	O	O
grup	O	O
musik	O	O
legendaris	O	O
Inggris	O	B-LOCATION

*["Jamie","T,"covers","one","song","from","group","music","legendary","British"] or " Jamic T covers a song from British's legendary music group"

TABLE IV. PRECISION, RECALL AND F1 SCORE OF ALL FEATURE COMBINATION (%)

Features	Precision	Recall	F1
WE	65.46	62.54	63.96
WE + NWE	66.07	59.41	62.09
WE + POS	**78.33**	**76.56**	**77.08**

Table IV shows the evaluation score of all feature combinations. Our experiments show that the feature combination WE + POS outputs the best performance with an F1 score of 77.08%, followed by WE (63.96%) and WE + NWE (62.09%). The precision and recall scores of WE + POS are also the highest, with the scores of 78.33% and 76.56%, respectively.

We can see that incorporating POS tag feature improves the F1 score by 13.12% (from 63.96% to 77.08%). The POS feature is able to enhance the performances. We suggest that the POS feature (Noun and Proper Noun) helps the model to determine the candidate of the named entities since most of the entities are in a form of Noun or Proper Noun.

Table VI. shows the precision, recall, and F1 scores of each named entity tags of the model WE + POS. These results show that all precision and recall scores are relatively balanced. The highest F1 score is achieved by PERSON tag (82.59%), followed by LOCATION (81.18%) and ORGANIZATION (67.46%).

ORGANIZATION tag has the lowest result compared to other tags. The common examples of the mispredicted ORGANIZATION in our model are illustrated as follows.

TABLE V. PRECISION, RECALL, AND F1 SCORES OF EACH NAMED ENTITY TAGS (FEATURE: WE + POS) (%)

Tag	Precision	Recall	F1
ORGANIZATION	67.19	68.15	67.46
PERSON	84.44	81.42	82.59
LOCATION	83.33	80.12	81.18
Average	78.33	76.56	77.08

Example 1:
*Peh, sekolah2 bagus dlm bidang political science di US kebanyakan gak offer MA **program***
ORGANIZATION

Example 2:
*Hadiahnya ada 10 Voucher GRATIS STREAMING 7 hari di **GENFLIX***
LOCATION

These common mistakes are the challenges that the NER system should overcome in the future works. In the first example, the model incorrectly predicts *"program"* as ORGANIZATION, while it should be labeled as O (Other). On the other hand, Example 2 shows a case whereby the model incorrectly predicts "GENFLIX" as LOCATION, where it should be labeled as ORGANIZATION. We suggest that the model thinks that the word after preposition *"di"* (English: *"at"*) should be labeled as LOCATION, which in some cases may be true. However, in this case, "GENFLIX" is an ORGANIZATION even though it is located after the preposition "di".

V. CONCLUSION

The result of the experiments shows that feature combination of Word Embedding and POS Tag is the best feature combination to NER modeling using stacked BLSTM architecture with an F1 score of 77.08%. Our experiment suggests that POS tag feature (Noun and Proper Noun) adds valuable information to help the model to determine the candidates of the named entities. The experiment results show that the use of POS tag feature in addition to word embedding can improve the F1 score by 13.12%.

For future works, one can try to experiment on various architectures and features for enhancing the NER system on microblog messages. In terms of architecture, one can try to add CRF layer on top of the LSTM architecture, as done by [14]. Another interesting architecture design is by adding attention mechanism on top of the LSTM layer. In addition to architecture, there are also potential features such as character embedding [15] and word shape for enhancing the performance of this NER system.

978-1-5386-3172-0/17/$31.00©

REFERENCES

[1] R. Grishman and B. Sundheim, "Message Understanding Conference-6: A Brief History," *Proc. 16th Conf. Comput. Linguist.*, vol. 1, pp. 466–471, 1996.

[2] M. Trupthi, S. Pabboju, and G. Narasimha, "Sentiment analysis on twitter using streaming API," in *Proceedings - 7th IEEE International Advanced Computing Conference, IACC 2017*, 2017, pp. 915–919.

[3] T. Sakaki, M. Okazaki, and Y. Matsuo, "Tweet analysis for real-time event detection and earthquake reporting system development," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 919–931, 2013.

[4] X. Liu, S. Zhang, F. Wei, and M. Zhou, "Recognizing Named Entities in Tweets," *Proc. 48th Annu. Meet. Assoc. Comput. Linguist.*, vol. 1, no. 2008, pp. 359–367, 2011.

[5] G. Krupka and K. Hausman, "IsoQuest Inc.: Description of the NetOwl (TM) Extractor System as Used for MUC-7," *Proc. Seventh Messag. Underst. Conf.*, pp. 1–10, 1998.

[6] J. R. Finkel and C. D. Manning, "Nested Named Entity Recognition," *Proc. 2009 Conf. Empir. Methods Nat. Lang. Process.*, no. August, pp. 141–150, 2009.

[7] M. Jansche and S. P. Abney, "Information Extraction from Voicemail Transcripts," in *Proceedings of the ACL '02 Conference on Empirical Methods in Natural Language Processing*, 2002, no. July, pp. 320–327.

[8] A. Ritter, S. Clark, Mausam, and O. Etzioni, "Named Entity Recognition in Tweets: An Experimental Study," *Proc. 2011 Conf. Empir. Methods Nat. Lang. Process.*, pp. 1524–1534, 2011.

[9] C. Li *et al.*, "TwiNER," *Proc. 35th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR '12*, p. 721, 2012.

[10] N. Taufik, A. F. Wicaksono, and M. Adriani, "Named entity recognition on Indonesian microblog messages," *Proc. 2016 Int. Conf. Asian Lang. Process. IALP 2016*, pp. 358–361, 2017.

[11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (Almost) from Scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.

[12] V. Athavale, S. Bharadwaj, M. Pamecha, A. Prabhu, and M. Shrivastava, "Towards Deep Learning in Hindi NER: An approach to tackle the Labelled Data Sparsity," *arXiv*, 2016.

[13] J. Zhou and W. Xu, "End-to-end learning of semantic role labeling using recurrent neural networks," *Acl*, pp. 1127–1137, 2015.

[14] Z. Huang, W. Xu, and K. Yu, "Bidirectional {LSTM-CRF} Models for Sequence Tagging," *CoRR*, vol. abs/1508.0, 2015.

[15] M. Rei, G. K. O. Crichton, and S. Pyysalo, "Attending to Characters in Neural Sequence Labeling Models," *CoRR*, vol. abs/1611.0, 2016.

978-1-5386-3172-0/17/$31.00