Knowledge Engineering and Data Science (KEDS)
Vol 4, No 1, July 2021, pp. 38–48

pISSN 2597-4602
eISSN 2597-4637

Indonesian Sentence Boundary Detection using Deep Learning Approaches

Joan Santoso [a, 1, *], Esther Irawati Setiawan [a, 2],
Christian Nathaniel Purwanto [b, 3], Fachrul Kurniawan [c, 4]

[a] *Department of Information Technology, Institut Sains dan Teknologi Terpadu Surabaya*
Jalan Ngagel Jaya Tengah 73 - 77, Surabaya, Indonesia

[b] *Electrical Engineering and Computer Science, National Yan Ming Chiao Tung University*
1001 University Road, Hsincu, Taiwan

[c] *Department of Informatics, Maulana Malik Ibrahim State Islamic University*
Jalan Gajayana No.50, Malang, Indonesia

[1] *joan@istts.ac.id*;* [2] *esther@istts.ac.id;* [3] *chrisnp.ee08@nycu.edu.tw;* [4] *fachrulk@ti.uin-malang.ac.id*
** corresponding author*

ARTICLE INFO

Article history:
Received 7 February 2021
Revised 22 May 2021
Accepted 21 June 2021
Published online 17 August 2021

Keywords:
Bahasa Indonesia
Bidirectional LSTM
Natural Language Processing
Sentence Boundary Detection
Sequence Classification

ABSTRACT

Detecting the sentence boundary is one of the crucial pre-processing steps in natural language processing. It can define the boundary of a sentence since the border between a sentence, and another sentence might be ambiguous. Because there are multiple separators and dynamic sentence patterns, using a full stop at the end of a sentence is sometimes inappropriate. This research uses a deep learning approach to split each sentence from an Indonesian news document. Hence, there is no need to define any handcrafted features or rules. In Part of Speech Tagging and Named Entity Recognition, we use sequence labeling to determine sentence boundaries. Two labels will be used, namely O as a non-boundary token and E as the last token marker in the sentence. To do this, we used the Bi-LSTM approach, which has been widely used in sequence labeling. We have proved that our approach works for Indonesian text using pre-trained embedding in Indonesian, as in previous studies. This study achieved an F1-Score value of 98.49 percent. When compared to previous studies, the achieved performance represents a significant increase in outcomes.

I. Introduction

Sentence segmentation or tokenization is a primary text processing in natural language processing [1]. To begin processing each token of words, we need to detect whether those tokens belong to the same sentence or not. Sentence boundary detection is used to split every sentence in a document. Hence, we can transfer this boundary information to the following process. This kind of task is a crucial one for natural language processing.

The core of detecting sentence boundary is to identify the end of a sentence [2]. A full stop mark "." usually ends the sentence, but not in all cases. For example, the full stop mark may denote an abbreviation, decimal value, or even currency value. Another punctuation marks that may end a sentence are a question mark or exclamation mark. Even a random word may finish a sentence. It needs many rules to encounter all the possibilities as every writer comes with their writing style. Many rules mean a lot of effort and time required.

Several studies use sentence boundary detection for text pre-processing. Walker [3] improves the accuracy of machine translation using a sentence splitter. Liu [4][5] and Roark [6] detect the sentence boundary from a conversation. Goldstein [7] and Erwin [8] also use sentence extractor to summarize a document. Rudrapal [9] uses sentence boundary detection for social media text. Another research by Chang *et al*. [10] use sentence position as a feature for question answering. Sentence boundary

https://doi.org/10.17977/um018v4i12021p38-48

detection can help the pre-processing phase and further improve the performance results. We choose the Deep Learning approach to simplify the learning process without crafting any rules by hand as in the traditional machine learning approach. We decide to use Bidirectional LSTM because of its ability to remember long-term sequences from two-way directions. By using this model, we do not need a handcrafted feature like in previous research. This model only needs the token of words.

There are several reasons why we conduct this research for Bahasa Indonesia. The main reason is the limitation of available tools and resources. Moreover, there is a need for tokenizing sentences. Natural Language Processing approaches can use the tokenizing task as a basis for further tasks. Sentence Boundary Detection is crucial as a pre-processing phase of many natural language processing tasks. One use is on Simultaneous Translation, where Sentence Boundary Detection could detect sentences before the translation process [11]. Sentence Boundary Detection also is needed for chatbot [12], machine translation, named entity recognition, and coreference resolution [13].

Previous researchers have worked on several machine learning approaches on Sentence Boundary Detection, i.e., Unsupervised [14], Rule-Based Method [13][15], Maximum Entropy [16], Hidden Markov Model [17], Conditional Random Field [18], Support Vector Machine [19], and Confusion Networks [20]. We use a deep learning approach to detect the sentence boundary as in our previous work [21]. Sentence Boundary Detection has been studied on other languages like English [22], Portuguese [23], French [24], Vietnamese [25], Chinese [26], Japanese [27], Marathi [28], Kannada [29], Arabic [30], and Urdu [31]. Another study is in Thai with Bi-LSTM CNN Approach [32]. In Indonesian, Sentence Boundary Detection has been presented using Maximum Entropy [33] and Bidirectional LSTM [21].

Our contribution is aimed directly at text processing in *Bahasa Indonesia*. The result of sentence boundary detection might be used for extracting information or even further, like solving another natural language processing problem. To our knowledge, we are the first to propose Sentence Boundary Detection with Deep Learning in Indonesian. After the tokenization process of a document, sometimes the determination of punctuation as the end of a sentence gives ambiguity whether it is the ending of a sentence or not. In this research, the sequential learning method is used to classify each token whether it marks the end of a sentence or not. We use Deep Learning to provide a crucial pre-processing of text that detects each sentence from a text document. Our sentence boundary detector can be used as a feature extractor for later tasks. Furthermore, we also prove that the deep learning model is capable of detecting sentence boundaries. Our approach could achieve a higher F1 Score than the previous approach, and no need to build any handcrafted rules.

II. Method

This section explains the steps of our research framework. The first step is explaining how we build our corpus for sentence boundary detection. This section explains how to get the raw data until processed as a labeled dataset, followed by further discussion of the proposed architecture. The discussion is divided according to each architecture layer: input layer, Bidirectional LSTM cells, and output layer. This section also includes an additional explanation of the used optimization method.

A. The problem in Sentence Boundary Detection for Bahasa Indonesia

This section will explain some problems that occur when detecting sentence boundaries for Bahasa Indonesia [33]. All of them are based on the ambiguities that punctuation marks may not always end a sentence [34]. We have listed each problem with few examples. There are also several points that we discuss to explain each case.

The first problem is writing the title and degree. When writing someone's title, the writer often uses the short version of the title or degree. As seen in the first example, "H" is a title for someone who used to have a pilgrimage. "Ir" is an academic degree for an engineering major. The title "H" stands for "Haji" and the title "Ir" stands for "Insinyur". This example shows the use of a stop mark to shorten the writing of the title or degree. The full stop mark in the title and degree does not end the sentence. This case is different when the title or degree is placed at the end of the sentence. In the second example, the stop mark in "Kom." ends the sentence because it is the last word.

1. *Presiden Ir. H. Joko Widodo berkunjung ke Surabaya.*
   President Ir. H. Joko Widodo visited Surabaya.

2. *Kelas kami diajar oleh Joan Santoso, M.Kom.*
   Our class is taught by Joan Santoso, M.Sc.

Abbreviation for names case comes when writing a long name. The writer usually makes the name shorter by using each word's first character and gives a full stop mark on each abbreviation. This abbreviation is written in uppercase letters. It is hard to list all of the abbreviations for names because many names are used in the document collection. Full stop mark in abbreviations for names does not end a sentence. However, it ends the sentence if the abbreviation is placed at the end of the sentence. This case is similar to the first case in 2.1, which happens in writing someone's title and degree. As seen in the first example, the writer use "W" which stands for "Widodo" to shorten the name. In the second example, the stop mark after "S" stands for "Santoso" ends the sentence because it is the last word.

1. *Presiden Ir. H. Joko W. berkunjung ke Surabaya.*
   President Ir. H. Joko W. visited Surabaya.
2. *Kelas kami kedatangan alumni bernama Joan S.*
   Our class is visited by an alumnus named Joan S.

The third problem is related to common abbreviations. There are some standard abbreviations used in Bahasa Indonesia. For examples: "a.n" (atas nama / by the name of), "s.d." (sampai dengan/until), "d.a." (dengan alamat/placed in), "jl." (jalan/street), "hlm." (halaman/page), etc. A full stop mark in this kind of abbreviation does not end a sentence. Usually, the writer uses these abbreviations in the middle of the sentence. The first example shows that the stop mark in "tgl" is shortened from the original word "tanggal". The second example also using the abbreviation "s.d." to shorten the original word "tanggal". In the third example, the writer could write the original word "Jalan" or just "Jl." for the shorter one.

1. *Dia akan pergi pada tgl. 25 Agustus 2018.*
   He will go on 25 August, 2018.
2. *Dia akan pergi dari Senin s.d. Minggu.*
   He will go from Monday to Sunday.
3. *Dia akan pergi ke Jl. Ngagel Jaya.*
   He will go to Ngagel Jaya Street.

Time separator is considered as the fourth problem. Time can be separated using punctuation marks. The full stop mark in the time separator does not end the sentence. In the first example, the expression of time "10.30" does not end its sentence. It only separates that 10 is the number of hours and 30 is the number of minutes. The second example also uses a full stop mark to separate between hours and minutes. Both the first and second examples provide the use case of stop mark for time expression in a sentence.

1. *Dia telah tiba di Surabaya pukul 10.30 WIB.*
   He had arrived in Surabaya at 10.30 WIB.
2. *Pada pagi hari jam 08.05, sang pembunuh menemui korban.*
   At 08:05 in the morning, the murderer met the victim.

The next problem, the money separator, can be expressed using punctuation marks. In the first example, the full stop mark in the expression "100.000" does not end the sentence. It separates the amount of money. Usually, people separate money per three digits in Bahasa Indonesia to make it easier for the reader. The second example also expresses the money format with the other currency used. "Rp." is the formal way to write Indonesian currency. There is another way to express money in Bahasa Indonesia, as stated in the third example. The only difference is in the use of ",-" to end the money expression.

1. *Buku ini seharga 100.000.*
   This book costs 100,000.
2. *Tas ini seharga Rp. 100.000.*
   This bag costs Rp. 100,000.

3. *Meja ini seharga Rp. 100.000,-.*
   This table costs Rp. 100,000,-.

Another problem is a number separator. A full stop mark is used to separate the number per thousand. It is not only used in expressing money but also when writing any number. For example, "1.123" in the first example contains full stop mark that separate number in expressing the number of people who died from the earthquake. The second example shows the use of a full stop mark to separate the number of smartphones. Almost any numbering expression uses a full stop mark to separate per thousand. This separation is similar to money separation to make the reader easier to read and understand.

1. *Gempa pekan lalu menimbulkan korban sekitar 1.123 jiwa.*
   Last week's quake caused casualties of around 1,123 people.
2. *Ada 1.500.000 ponsel pintar yang terhubung ke server kami.*
   There are 1,500,000 smartphones connected to our server.

The email-formatted text could be problematic, contain more than one full stop mark. The first example shows the standard email formatted text. However, the second example shows that the number of full stop marks in email can be as much as possible. Users can freely choose a custom name for their email. The third example shows that there are a lot of non-formal ways to write an email. In this case, building rules for each case is time-consuming. Moreover, email-formatted text can also be written like in the fourth example. The writer can use "dot" instead of a full stop mark.

1. *Pertanyaan lain dapat dikirimkan ke email christian.np@indocl.stts.edu.*
   Other questions can be sent to christian.np@indocl.stts.edu.
2. *Email kami yaitu people.hrd.tech.123@main.hrd.indocl.stts.edu.*
   Our email is people.hrd.tech.123@main.hrd.indocl.stts.edu.
3. *Email saya adalah christian at indocl.stts.edu.*
   My email is christian at indocl.stts.edu.
4. *Email dia adalah christian.np at indocl dot stts dot edu.*
   His email is christian.np at indocl dot stts dot edu.

Problem number eight is the username formatted text. Sometimes the writer takes quotes from social media and includes the username. There is no limitation on giving a full stop mark in the username. Full stop in username does not end the sentence. The first example shows the use of a full stop mark in the usual username "@christian.np". On the contrary, a username can also contain many full stop marks like the second example. "@christian.n.p.stts.sby" contains several numbers of full stop marks. This case rarely happens, but it is still possible for a username to have many full stop marks.

1. *Akun @christian.np juga mengatakan hal yang serupa.*
   Account @christian.np also said the same thing.
2. *@christian.n.p.stts.sby @joan.s. Ayo pergi ke Bali bulan depan!*
   @christian.n.p.stts.sby @joan.s. Let's go to Bali next month!

Sentence emphasis is often used when the author wants to emphasize some meaning from the text. This kind of writing is often found in drama script writing to express feeling through the writing. In addition, the writer can combine many different punctuation marks according to his or her creativity. This case is the same when handling free structured text from social media. Chat, comment, or post on social media does not have a fixed rule in writing. Users can write anything based on current trends, thus making a problem because the rule for splitting each sentence is different for each time.

Sometimes multiple punctuations can be combined to be a single token. Usually, the last punctuation mark in the token is the one that ends the sentence. A question mark may not finish a sentence when it comes together with another punctuation mark like "?!". As seen in Figure 1, the question mark after the word "Surabaya" does not end the sentence. The exclamation mark after the question mark is the one that ends the sentence. On the other hand, a single punctuation mark may not end the sentence if it is placed in line with another punctuation mark. The last punctuation mark in token "!!!" which is the exclamation mark, is the one that ends the sentence.

Input : *Akankah Bapak Gunawan berkunjung ke Surabaya?! Kami yakin beliau datang!!!*
Output : *Akankah Bapak Gunawan berkunjung ke Surabaya?!*
Kami yakin beliau datang!!!

Fig. 1. Sentence emphasis example.

Berikut 8 nama calon pimpinan KPK hasil seleksi Pansel yang dikirim Presiden ke DPR:
1. Bambang Widjojanto
2. Yunus Husein
3. Abdullah Hehamahua
4. Handoyo Sudradjat
5. Abraham Samad
6. Jaksa Zulkarnain
7. Adnan Pandu Praja
8. Irjen Pol (Purn) Aryanto Sutadi

Fig. 2. Non-punctuation token example from detik.com

The next problem happens in the dialogue text. Some conversations may consist of multiple sentences. When we try to split them up, we lose their context, which is used to determine these sentences belong to whom. The sentences seem to have their context, but those sentences are in the same context. We make an agreement that all spoken words from a person at a particular time will be counted as a single sentence, even if there is more than one sentence inside it. This agreement may be different from other sentence tokenizer tools where the text is tokenized based on the end of a sentence, not the context of the whole text.

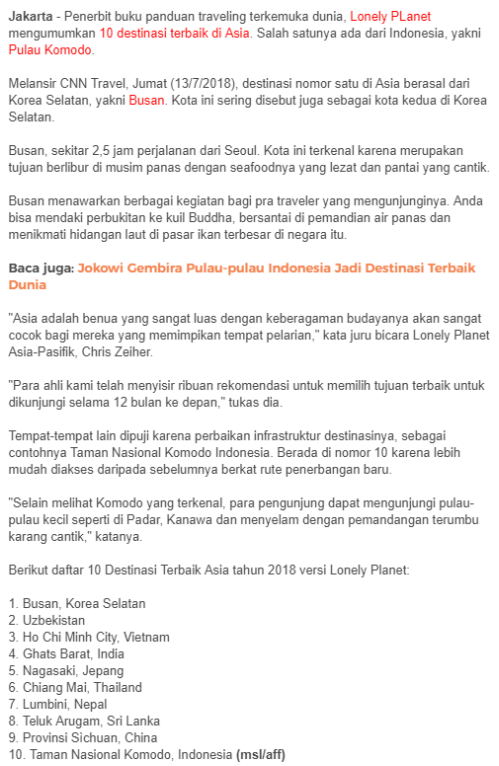
1. *"Siapa namamu?" tanya Joan.*
   "What is your name?" asked Joan.
2. *"Hai! Nama saya Christian NP. Saya senang berkenalan denganmu!" ujar Christian.*
   "Hi! My name is Christian NP. I am glad to know you!" said Christian.

The first example is a common writing style in which dialogue contains only one sentence. The second example is more complex than the previous example. It consists of three different sentences, which are "Hai!", "Nama saya Christian NP.", and "Bagaimana harimu?". We count these three sentences as a single sentence, together with the main sentence. The context is the same because they are all spoken by one person at a particular time.

The last problem is the non-punctuation token. As we analyze our dataset, we found that the end of each sentence is not always a punctuation mark. It may occur when a non-word ends the sentence. This case usually happens when converting a list to plain text. Point by point in a list can either ends with a punctuation mark like a full stop mark or just a word. Figure 2 [35] shows the output of sentence tokenization from a list. Colon mark ":" ends the first sentence as a description of the list. The second sentence until the rest is split according to the number of the list. As we can see, the end of the second sentence until the rest is different. It may be *"Widjojanto", "Husein"*, "Hehamahua", or other words that ends the sentence. On another view, the full stop mark after the index is combined with the current sentence. These numbers are used as an index and do not end the sentence.

B. Data Preparation

Our corpus is built from Indonesian news documents. All news is crawled from two news sites which are Detik and Kompas. Each news is then extracted and parsed to get the text. We remove unused information like ads, pictures, video, and audio because we only need plain text. Then, we conducted post-processing, which converts all list types to readable text and does tokenization at the word level. The product is a token that contains either word or a punctuation mark. In the last step, we split each sentence manually for all documents and crawled those sites from 2011 until 2012. There are 14,142 sentences in total from all documents.



Jakarta - Penerbit buku panduan traveling terkemuka dunia, Lonely PLanet mengumumkan 10 destinasi terbaik di Asia. Salah satunya ada dari Indonesia, yakni Pulau Komodo.

Melansir CNN Travel, Jumat (13/7/2018), destinasi nomor satu di Asia berasal dari Korea Selatan, yakni Busan. Kota ini sering disebut juga sebagai kota kedua di Korea Selatan.

Busan, sekitar 2,5 jam perjalanan dari Seoul. Kota ini terkenal karena merupakan tujuan berlibur di musim panas dengan seafoodnya yang lezat dan pantai yang cantik.

Busan menawarkan berbagai kegiatan bagi pra traveler yang mengunjunginya. Anda bisa mendaki perbukitan ke kuil Buddha, bersantai di pemandian air panas dan menikmati hidangan laut di pasar ikan terbesar di negara itu.

Baca juga: Jokowi Gembira Pulau-pulau Indonesia Jadi Destinasi Terbaik Dunia

"Asia adalah benua yang sangat luas dengan keberagaman budayanya akan sangat cocok bagi mereka yang memimpikan tempat pelarian," kata juru bicara Lonely Planet Asia-Pasifik, Chris Zeiher.

"Para ahli kami telah menyisir ribuan rekomendasi untuk memilih tujuan terbaik untuk dikunjungi selama 12 bulan ke depan," tukas dia.

Tempat-tempat lain dipuji karena perbaikan infrastruktur destinasinya, sebagai contohnya Taman Nasional Komodo Indonesia. Berada di nomor 10 karena lebih mudah diakses daripada sebelumnya berkat rute penerbangan baru.

"Selain melihat Komodo yang terkenal, para pengunjung dapat mengunjungi pulau-pulau kecil seperti di Padar, Kanawa dan menyelam dengan pemandangan terumbu karang cantik," katanya.

Berikut daftar 10 Destinasi Terbaik Asia tahun 2018 versi Lonely Planet:

1. Busan, Korea Selatan
2. Uzbekistan
3. Ho Chi Minh City, Vietnam
4. Ghats Barat, India
5. Nagasaki, Jepang
6. Chiang Mai, Thailand
7. Lumbini, Nepal
8. Teluk Arugam, Sri Lanka
9. Provinsi Sichuan, China
10. Taman Nasional Komodo, Indonesia **(msl/aff)**

1. *Penerbit buku panduan traveling terkemuka dunia, Lonely Planet mengumumkan 10 destinasi terbaik di Asia.*
2. *Salah satunya ada dari Indonesia, yakni Pulau Komodo.*
   Melansir CNN Travel, Jumat (13/7/2018), destinasi nomor satu di Asia berasal dari Korea Selatan, yakni Busan.
3. *Kota ini sering disebut juga sebagai kota kedua di Korea Selatan.*
   Busan, sekitar 2,5 jam perjalanan dari Seoul.
4. *Kota ini terkenal karena merupakan tujuan berlibur di musim panas dengan seafoodnya yang lezat dan pantai yang cantik.*
   Busan menawarkan berbagai kegiatan bagi pra traveler yang mengunjunginya.
5. *Anda bisa mendaki perbukitan ke kuil Buddha, bersantai di pemandian air panas dan menikmati hidangan laut di pasar ikan terbesar di negara itu.*
6. *"Asia adalah benua yang sangat luas dengan keberagaman budayanya akan sangat cocok bagi mereka yang memimpikan tempat pelarian," kata juru bicara Lonely Planet Asia-Pasifik, Chris Zeiher.*
7. *"Para ahli kami telah menyisir ribuan rekomendasi untuk memilih tujuan terbaik untuk dikunjungi selama 12 bulan ke depan," tukas dia.*
8. *Tempat-tempat lain dipuji karena perbaikan infrastruktur destinasinya, sebagai contohnya Taman Nasional Komodo Indonesia. Berada di nomor 10 karena lebih mudah diakses daripada sebelumnya berkat rute penerbangan baru.*
9. *"Selain melihat Komodo yang terkenal, para pengunjung dapat mengunjungi pulau-pulau kecil seperti di Padar, Kanawa dan menyelam dengan pemandangan terumbu karang cantik," katanya.*
10. *Berikut daftar 10 Destinasi Terbaik Asia tahun 2018 versi Lonely Planet:*
11. *1. Busan, Korea Selatan*
12. *2. Uzbekistan*
13. *3. Ho Chi Minh City, Vietnam*
14. *4. Ghats Barat, India*
15. *5. Nagasaki, Jepang*
16. *6. Chiang Mai, Thailand*
17. *7. Lumbini, Nepal*
18. *8. Teluk Arugam, Sri Lanka*
19. *9. Provinsi Sichuan, China*
20. *10. Taman Nasional Komodo, Indonesia*

Fig. 3. Data preparation from news site Detik

Figure 3 [36] displays an example of the data preparation process. The rest of the dataset follows the same process. On the left side is the original HTML formatted text from Detik news. On the right side is the result with 20 sentences in total. Each sentence is split based on its context (as discussed in section 2). There is a section with a list of typed text in the last part, separated one by one per point. The numbering is essential information for further tasks.

C. Sequence Classification for Sentence Boundary Detection

Long Short-Term Memory (LSTM) is established for Part-of-Speech (POS) Tagging, Named Entity Recognition (NER), and Noun Phrase Chunking. Sentence boundary detection can be seen as a sequence classification problem where we want to label every timestep of the input or as a collocation identification problem [37]. Every token of input is predicted, whether it is the end of a sentence or not, based on the previous token. We build an architecture based on the nature of the problem. We pay much attention to the whole sequence rather than individual prediction. Thus, we use bidirectional LSTM to capture the sequential features from both directions (left to right direction and right to left direction).

Figure 4 is the visualization of our system architecture. We divide our architecture into three different layers: input layer, sequence learning, and output layer. The input is a sequence of tokens from a single sentence, and the output is also a sequence of labels. In the input layer, it just simply converts each token into a vector using word embedding. Thus, the word vector is learned by the sequence learning layer. We use Bidirectional LSTM for sequence learning. In the end, all predicted results are converted into final predictions in the output layer. This prediction contains information on which token is identified as the end of the sentence and which is not. We also use one of the optimization methods to help the learning process. We use Adam optimizer for that purpose.

Our proposed input layer is token embedding because it converts from token input into vector. Every token is a string which can be either word or a punctuation mark. We use Skip-Gram Word2Vec as our embedding model. Skip-Gram Word2Vec is capable of giving a semantic representation of a token. It is also capable of providing the similarity of context from different words. However, Word2Vec has a drawback when handling unknown words. Word2Vec cannot provide the vector representation if the word is not trained before. To encounter this problem, we use a random trained vector to represent every unknown word.

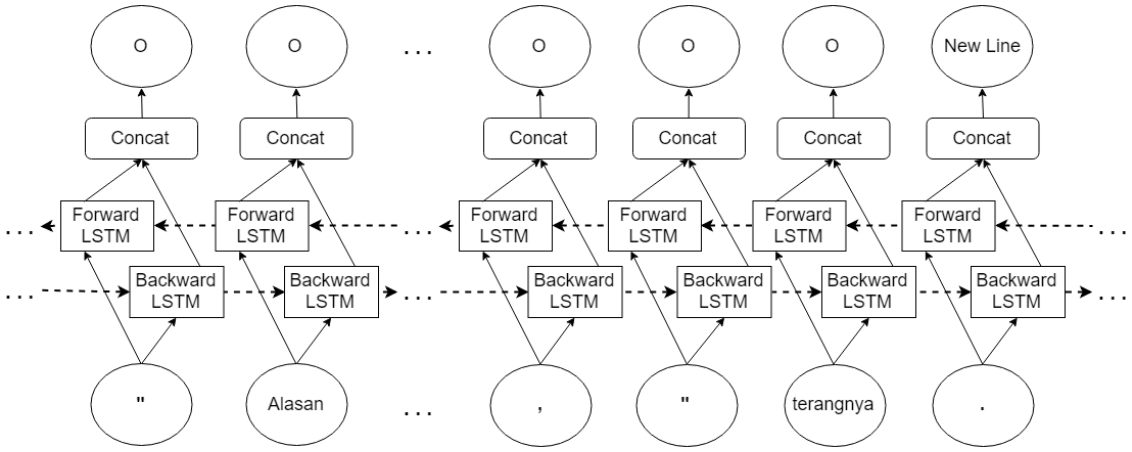


Fig. 4. System architecture

Sequence learning is used to predict the outputs from the given inputs. We use bidirectional LSTM, which uses two different LSTM cells. Each cell acts as a forward learner and a backward learner. Forward LSTM reads input from the first token to the last token, and backward LSTM reads input from the last token to the first token. The results from both of the cells will be concatenated. The gray circle on the figure denotes the input to LSTM Cell. The colorful circle denotes every gate in the LSTM cell, which consists of yellow for the activation gate, green for the input gate, red for the forget gate, and blue for the output gate. The last one is light blue for cell state, which holds long-term memory from several previous calculations.

$$a_t^{fwd} = \tanh(w_a^{fwd}.x_t^{fwd} + u_a^{fwd}.h_{t-1}^{fwd} + b_a^{fwd}) \quad (1)$$

$$i_t^{fwd} = \sigma(w_i^{fwd}.x_t^{fwd} + u_i^{fwd}.h_{t-1}^{fwd} + b_i^{fwd}) \quad (2)$$

$$f_t^{fwd} = \sigma(w_f^{fwd}.x_t^{fwd} + u_f^{fwd}.h_{t-1}^{fwd} + b_f^{fwd}) \quad (3)$$

$$o_t^{fwd} = \sigma(w_o^{fwd}.x_t^{fwd} + u_o^{fwd}.h_{t-1}^{fwd} + b_o^{fwd}) \quad (4)$$

$$c_t^{fwd} = c_{t-1}^{fwd} * f_t^{fwd} + i_t^{fwd} * a_t^{fwd} \quad (5)$$

$$h_t^{fwd} = o_t^{fwd} * \tanh(c_t^{fwd}) \quad (6)$$

Equations (1) to (6) are the mathematical functions for the Forward LSTM cell. (1) is the activation gate, (2) is the input gate, (3) is the forget gate, (4) is the output gate, (5) is the cell state, and (6) is the prediction from the Forward LSTM cell. Equations (7) to (12) are similar to equations (1) to (6). (13) is used as the final prediction of both LSTM Cells which use concatenation function to combine two vectors values.

$$a_t^{bwd} = \tanh(w_a^{bwd}.x_t^{bwd} + u_a^{bwd}.h_{t-1}^{bwd} + b_a^{bwd}) \quad (7)$$

$$i_t^{bwd} = \sigma(w_i^{bwd}.x_t^{bwd} + u_i^{bwd}.h_{t-1}^{bwd} + b_i^{bwd}) \quad (8)$$

$$f_t^{bwd} = \sigma(w_f^{bwd}.x_t^{bwd} + u_f^{bwd}.h_{t-1}^{bwd} + b_f^{bwd}) \quad (9)$$

$$o_t^{bwd} = \sigma(w_o^{bwd}.x_t^{bwd} + u_o^{bwd}.h_{t-1}^{bwd} + b_o^{bwd}) \quad (10)$$

$$c_t^{bwd} = c_{t-1}^{bwd} * f_t^{bwd} + i_t^{bwd} * a_t^{bwd} \quad (11)$$

$$h_t^{bwd} = o_t^{bwd} * \tanh(c_t^{bwd}) \quad (12)$$

$$h_t = \text{concat}(h_t^{fwd}, h_t^{bwd}) \quad (13)$$

The output layer converts every vector result from the sequence learner to be the predicted label using the Softmax function. The function provides a probability distribution for each label and then outputs the label with the largest probability. The output labels are “E” as “EndOfSentence” and “O”. Label “E” or “EndOfSentence” means the current token is the ending of a sentence. Label “O”

(Others) represents that the current token does not end a sentence. Because of its sequential nature, every token input will have a single output label.

In this research, we choose Adam optimizer to obtain an appropriate gradient for each weight in networks. Adam combines adaptive learning rate and momentum. Technically, every weight is updated by using gradient calculated with Adam. Algorithm 1 is the pseudocode of the Adam optimizer. The default value for each hyperparameter is based on the original paper in [38].

Algorithm 1; Adam Optimizer

```
while θt not converged do:
    t = t + 1
    gt = GetGradient(θt-1)
    mt = β1 * mt-1 + (1 - β1)*gt
    vt = β2 * vt-1 + (1 – β2)*gt^2
    mt = mt / (1- β1^t)
    vt = vt/(1- β2^t)
    θt = θt-1 α * mt/(√vt + ϵ)
return θt
```

III. Results and Discussions

We had done several experiments to prove the capability of our proposed architecture. We provide some test cases by fine-tuning a few hyperparameters. Besides, we also report a different approach by using standard LSTM to compare with our Bidirectional LSTM model. We ran different scenarios based on the changing of hyperparameters. Each scenario used the same dataset. We split our corpus by 70% (9,953 sentences) for training and 30% (4,189 sentences) for testing. The random seed was turned off to focus only on the original effect of the hyperparameters setting. There were two big categories based on the model we have tried. We tested every model by changing the hidden unit of LSTM cell, the number of layers, and training iteration.

Table 1 contains all experiments using different kinds of methods. The row represents the method, and the column represents the number of iterations. Every method is either experimented on the LSTM or the word embedding. Based on the results in Table 1, we found that BiLSTM (Bidirectional LSTM) works better than UniLSTM (Unidirectional LSTM). Word embedding gives a small difference in overall accuracy. The number of iterations will increase accuracy but not a lot in the next iteration. We also conduct another trial to identify the effect of word embedding dimension by using 50% of the training document and separate as 70% sentences as training and 30% sentences as testing. The results are as follows 0.9843% for 50 dimensions, 0.9830% for 100 dimensions, 0.9832% for 150 dimensions, 0.9846% for 200 dimensions, 0.9850% for 250 dimensions, 0.9857% for 300 dimensions, 0.9808% for 350 dimensions, 0.9826% for 400 dimensions, 0.9863% for 450 dimensions, and 0.9826% for 500 dimensions. Our final result is 98.49% when using Bi-LSTM model with Word2Vec embedding and 100 iterations.

The second experiment was conducted by comparing the performance of the proposed method with several approaches from previous state-of-the-art research. The problem modeling in this research is sequential tagging for a set of input token sequences. Several sequential tagging methods will be used as a comparison method in this proposed approach, namely Maximum Entropy, Decision Tree, and Naïve Bayes. In addition to using several traditional non-Deep Learning models, the performance of the proposed method is also compared with previous studies using Bi-LSTM by Purwanto *et al*. [21]. The experimental results can be seen in Table 2.

Table 1. Experiments results

Method	Number of Iteration			
	10	20	50	100
UniLSTM	96.79%	96.94%	97.14%	96.81%
BiLSTM	96.95%	97.43%	98.22%	98.47%
UniLSTM + Word2Vec*	96.91%	96.41%	97.44%	97.48%
BiLSM + Word2Vec*	97.09%	98.10%	98.39%	98.49%

*We use Skip-Gram model for Word2Vec

Table 2. Experimental results compared with other studies

No.	Previous Research	Performance
1	Maximum Entropy	87.91%
2	Decision Tree	82.23%
3	Naïve Bayes	86.28%
4	Bi-LSTM by Purwanto *et al*. [21]	96.57%
5	Our Proposed Model	98.49%

Based on the experimental results in previous studies, the best performance of the Bi-LSTM proposed in this study provides the most significant increase of approximately 13% compared to other approaches that do not use Deep Learning. However, compared with the Bi-LSTM that has been proposed by [21], there was an increase of approximately 2%. The reason is that the results of the proposed approach are using two labels and while in [21] approach uses four labels. The use of two labels can give the best results compared to 4 labels in previous studies, especially in sentence boundary detection research.

IV. Conclusion

We have done several experiments to prove the capability of Bidirectional LSTM as the sequence learner to solve sentence boundary detection. We view this task as a sequential problem where every token input is predicted to end a sentence. Based on our experiments, we could reach 98.49% F1 score with Bidirectional LSTM as our sequence learner and train embedding for the word embedding as the best model. We also compare our research with other widely used methods in sequence classification. We conclude that the Bidirectional LSTM is way better than a Unidirectional LSTM. In our case, word2vec does not effectively capture sentence boundaries for Indonesian news documents. Our last trial gives a similar F1 score, whether using low dimension or high dimension embedding size.

Acknowledgment

The authors want to appreciate Institut Sains dan Teknologi Terpadu Surabaya (ISTTS) for supporting this research. Also, we want to thank our laboratory member from Natural Language Processing Laboratory from ISTTS for helping us finish this research.

Declarations

Author contribution

All authors contributed equally as the main contributor of this paper. All authors read and approved the final paper.

Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of interest

The authors declare no known conflict of financial interest or personal relationships that could have appeared to influence the work reported in this paper.

Additional information

Reprints and permission information is available at http://journal2.um.ac.id/index.php/keds.

Publisher's Note: Department of Electrical Engineering - Unversitas Negeri Malang remains neutral with regard to jurisdictional claims and institutional affiliations.

References

[1] D. Jurafsky and H. James, Martin: Speech and Language Processing: *An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, Englewood Cliffs, 2008.

[2] J. Read, R. Dridan, S. Oepen, and L. J. Solberg, "Sentence boundary detection: A long solved problem?," in *Proceedings of COLING 2012: Posters*, 2012, pp. 985–994.

[3] D. J. Walker, D. E. Clements, M. Darwin, and J. W. Amtrup, "Sentence boundary detection: A comparison of paradigms for improving MT quality," in *Proceedings of the MT Summit VIII*, 2001, vol. 58.

[4] Y. Liu, A. Stolcke, E. Shriberg, and M. Harper, "Comparing and combining generative and posterior probability models: Some advances in sentence boundary detection in speech," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004, pp. 64–71.

[5] Y. Liu, A. Stolcke, E. Shriberg, and M. Harper, "Using conditional random fields for sentence boundary detection in speech," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 2005, pp. 451–458.

[6] B. Roark *et al.*, "Reranking for sentence boundary detection in conversational speech," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, 2006, vol. 1, pp. I--I.

[7] J. Goldstein, V. Mittal, J. Carbonell, and M. Kantrowitz, "Multi-document summarization by sentence extraction," in *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic summarization*, 2000, pp. 40–48.

[8] E. Y. Hidayat, F. Firdausillah, K. Hastuti, I. N. Dewi, and A. Azhari, "Automatic text summarization using latent Drichlet allocation (lda) for document clustering," *Int. J. Adv. Intell. Informatics*, vol. 1, no. 3, pp. 132–139, 2015.

[9] D. Rudrapal, A. Jamatia, K. Chakma, A. Das, and B. Gambäck, "Sentence Boundary Detection for Social Media Text," in *Proceedings of the 12th International Conference on Natural Language Processing*, 2015, pp. 254–260.

[10] X. Chang and Q. Zheng, "Offline definition extraction using machine learning for knowledge-oriented question answering," in *International Conference on Intelligent Computing*, 2007, pp. 1286–1294.

[11] R. Zhang and C. Zhang, "Dynamic Sentence Boundary Detection for Simultaneous Translation," *Proceedings of the First Workshop on Automatic Simultaneous Translation*, 2020.

[12] T. A. Le, "Sequence labeling approach to the task of sentence boundary detection," in *ACM International Conference Proceeding Series*, Jan. 2020, pp. 144–148, doi: 10.1145/3380688.3380703.

[13] N. Sadvilkar and M. Neumann, "PySBD: Pragmatic Sentence Boundary Disambiguation," Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.09657.

[14] T. Kiss and J. Strunk, "Unsupervised multilingual sentence boundary detection," *Comput. Linguist.*, vol. 32, no. 4, pp. 485–525, 2006.

[15] J. Wang, Y. Zhu, and Y. Jin, "A rule-based method for Chinese punctuations processing in sentences segmentation," in *2014 International Conference on Asian Language Processing (IALP)*, 2014, pp. 195–198.

[16] J. C. Reynar and A. Ratnaparkhi, "A maximum entropy approach to identifying sentence boundaries," in *Proceedings of the fifth conference on Applied natural language processing*, 1997, pp. 16–19.

[17] B. Jurish and K.-M. Würzner, "Word and Sentence Tokenization with Hidden Markov Models.," *JLCL*, vol. 28, no. 2, pp. 61–83, 2013.

[18] K. Tomanek, J. Wermter, and U. Hahn, "Sentence and token splitting based on conditional random fields," in *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, 2007, vol. 49, p. 57.

[19] Y. Akita, M. Saikou, H. Nanjo, and T. Kawahara, "Sentence boundary detection of spontaneous Japanese using statistical language model and support vector machines," 2006.

[20] D. Hillard, M. Ostendorf, A. Stolcke, Y. Liu, and E. Shriberg, "Improving automatic sentence boundary detection with confusion networks," in *Proceedings of HLT-NAACL 2004: Short Papers*, 2004, pp. 69–72.

[21] C. N. Purwanto, A. T. Hermawan, J. Santoso, and Gunawan, "Distributed Training for Multilingual Combined Tokenizer using Deep Learning Model and Simple Communication Protocol," in *2019 1st International Conference on Cybernetics and Intelligent System (ICORIS)*, 2019, vol. 1, pp. 110–113.

[22] D. Gillick, "Sentence boundary detection and the problem with the US," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Companion Volume: Short Papers, 2009, pp. 241–244.

[23] C. N. Silla and C. A. A. Kaestner, "An analysis of sentence boundary detection systems for English and Portuguese documents," in *International Conference on Intelligent Text Processing and Computational Linguistics*, 2004, pp. 135–141.

[24] C.-E. González-Gallardo and J.-M. Torres-Moreno, "Sentence boundary detection for French with subword-level information vectors and convolutional neural networks," *arXiv Prepr. arXiv1802.04559*, 2018.

[25] H. P. Le and T. V. Ho, "A maximum entropy approach to sentence boundary detection of Vietnamese texts," 2008.

[26] N. Xue and Y. Yang, "Chinese sentence segmentation as comma classification," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Jun. 2011, pp. 631–635, [Online]. Available: https://www.aclweb.org/anthology/P11-2111.

[27] K. Shitaoka, K. Uchimoto, T. Kawahara, and H. Isahara, "Dependency Structure Analysis and Sentence Boundary Detection in Spontaneous Japanese," in *Proceedings of the 20th International Conference on Computational Linguistics*, 2004, pp. 1107–es, doi: 10.3115/1220355.1220514.

[28] N. Wanjari, G. M. Dhopavkar, and N. B. Zungre, "Sentence Boundary Detection For Marathi Language," *Procedia Comput. Sci.*, vol. 78, pp. 550–555, 2016, doi: https://doi.org/10.1016/j.procs.2016.02.101.

[29] D. N and R. K. P, "Article: Sentence Boundary Detection in Kannada Language," *Int. J. Comput. Appl.*, vol. 39, no. 9, pp. 38–41, Feb. 2012.

[30] C.-E. González-Gallardo, E. L. Pontes, F. Sadat, and J.-M. Torres-Moreno, "Automated Sentence Boundary Detection in Modern Standard Arabic Transcripts using Deep Neural Networks," *Procedia Comput. Sci.*, vol. 142, pp. 339–346, 2018, doi: https://doi.org/10.1016/j.procs.2018.10.485.

[31] Z. Rehman, W. Anwar, and U. I. Bajwa, "Challenges in Urdu Text Tokenization and Sentence Boundary Disambiguation," in *Proceedings of the 2nd Workshop on South Southeast Asian Natural Language Processing*

(WSSANLP), Nov. 2011, pp. 40–45, [Online]. Available: https://www.aclweb.org/anthology/W11-3007.

[32] S. Sirirattanajakarin, D. Jitkongchuen, and P. Intarapaiboon, "BoydCut: Bidirectional LSTM-CNN Model for Thai Sentence Segmenter," Sep. 2020, doi: 10.1109/IBDAP50342.2020.9245454.

[33] S. J. Putra, M. N. Gunawan, I. Khalil, and T. Mantoro, "Sentence boundary disambiguation for Indonesian language," in *ACM International Conference Proceeding Series*, Dec. 2017, pp. 587–590, doi: 10.1145/3151759.3156474.

[34] S. Raharjo, R. Wardoyo, and A. E. Putra, "Rule Based Sentence Segmentation of Indonesian Language," *J. Eng. Appl. Sci.*, vol. 13, no. 21, pp. 8986–8992, 2018.

[35] "Siapa Calon Pimpinan KPK yang Akan Dipilih DPR?," Nov. 14, 2011. https://news.detik.com/berita/d-1766855/siapa-calon-pimpinan-kpk-yang-akan-dipilih-dpr (accessed Aug. 09, 2021).

[36] "10 Destinasi Terbaik Asia 2018 Versi Lonely Planet, Ada Komodo," Jul. 13, 2018. https://travel.detik.com/travel-news/d-4113452/10-destinasi-terbaik-asia-2018-versi-lonely-planet-ada-komodo (accessed Aug. 09, 2021).

[37] T. Kiss and J. Strunk, "Viewing sentence boundary detection as collocation identification," in *Proceedings of KONVENS*, 2002, vol. 2002, pp. 75–82.

[38] D. P. Kingma and J. L. Ba, "Adam: a Method for Stochastic Optimization," *Int. Conf. Learn. Represent. 2015*, 2015.