ABSTRACT  Word embedding is an efficient feature representation that carries semantic and syntactic information. Word embedding works as a word level that treats words as minor independent entity units and cannot handle words that are not in the training corpus. One solution is to generate embedding from more minor parts of words such as morphemes. Morphemes are the smallest part of a word linguistic that has meaning in the grammatical unit of languages. This study aims to build a morpheme embedding model for Bahasa Indonesia (in English: Indonesian Language) in sort: Bahasa. However, there were many morphological rules in Bahasa, such as inflectional and derivational affixes. This implies that all rules in word segmentation will increase the computational complexity. Moreover, the rules were not regular and similar for all words in Bahasa. Therefore, this study modified a Byte Pair Embedding (BPE) algorithm to generate morpheme embedding appropriate to the morphology of Bahasa. The study implemented a simple method by filtering the BPE segmentation results with the list of Bahasa’s morphemes. This process has proven to anticipate the limitation of a conventional BPE algorithm that produces intermediate junk tokens that are not meaningful. Based on three evaluation scenarios, the model in the study can handle OOV and carry semantic and syntactic information in the embedding value of the words.

INDEX TERMS  Morphological embedding, Bahasa Indonesia linguistic, word segmentation.

I. INTRODUCTION  One of the essential parameters for improving machine learning performance is the selection of an appropriate input feature representation. Recent research developments in natural language processing (NLP) reveal that one of the most efficient text representations is word embedding [1]–[3]. Word embedding, also known as continuous word representations or distributed representation [4], is a feature representation of the text generated using the neural network method. The representation is derived from large unlabeled corpora based on the co-occurrence of a word and its context words. Word embedding carries semantic and syntactic information. Words with similar meanings will also have similar word embedding. This model has proven to anticipate the shortcomings of discrete text representation [5], where the discrete model is constrained by the sparsity and curse of dimensionality [6]. However, word embedding works as a word level that treats words as minor independent entity units. Therefore, the model cannot produce embeddings for words that are not in the training corpus. In other words, this model cannot handle out-of-vocabulary (OOV). The model ignores the internal structure of word-forming subwords, such as characters or morphemes, and makes this representation unable to capture explicit relationships between syntactic morphology [7], [8].

Some researchers have attempted to solve the OOV problem by increasing the size of the training corpus. However, this method is inadequate considering inefficient storage media and increases the complexity of the computational process. In addition, several previous researchers have revealed that a large corpus size is not the most critical parameter in forming reliable word embedding [9]–[11]. One solution to anticipate OOV problems is to generate embedding from smaller parts of words, such as characters or subwords.
Therefore, words that do not have an embedding value in vocabulary can be arranged based on the word’s subwords. Several previous studies have proven that the method of breaking words into a smaller form can improve downstream NLP performance [12], such as classification [13] and machine translation [14]. Previous studies have also proven that text representation generated using subword embedding such as fastText [15] is better at handling OOV than conventional word embeddings such as word2vec [16] and GloVe [17].

Generally, two models are used to segment a word into smaller units: split into characters or split into subwords. However, a character is not a natural minimum linguistic unit. Forming word representation using a character into a whole word can be too insufficient and coarse [18]. Segmenting a word in its morphological form is a better method. Compounding morphemes into a word is more meaningful for authentic learning and representation, but the method requires complex linguistic rules. Therefore, some researchers choose an unsupervised method for segmenting words such as byte pair encoding (BPE) [14].

BPE is a simple unsupervised segmentation method that breaks words based on the sequence of characters that appear most often in the corpus. BPE only uses frequency as its measure and does not rely on any morphological or linguistic knowledge; thus, it can be implemented as multilingual. This method is proven to anticipate rare and unknown words because the encoding value for unknown words can be formed from the subword values. Owing to its simplicity, BPE has become a pioneer de facto standard for subword segmentation, especially for machine translation processes [14]. However, the drawback of this model is that each word consists of only one unique segmentation.

Several previous studies, such as LM unigram by [19] and BPE dropout by [20], have tried to anticipate the limitation of BPE by establishing subword regularization. Subword regularization is a method that generates several prospects for word segmentation and finds the best segmentation using probability calculation. However, this process makes the segmentation complex more time-consuming.

Another drawback of BPE is that it can produce intermediate junk tokens that are not meaningful [19]. Moreover, some common affixes and punctuation are absorbed into other tokens [21] and cause the affixes to appear in the vocabulary. For example, inflectional affixes such as “-s” and “-ed” appear in many English contexts. Thus, the tokens merge with adjacent units owing to their frequency, and it is a big challenge to generate word segmentation that appropriates morpheme forms grammatically. Because each language has a unique, dependent rule, no single word segmentation model can be generalized to all languages [22].

Bahasa Indonesia (in English: Indonesian Language), in sort Bahasa, is an agglutinative language. In agglutinative, affixation plays an essential role in the addition of prefixes and suffixes to basic morphemes and establishes new words [23]. For example, one of the words in Bahasa, such as *permainan* (in English: game), contains morphemes: a prefix *per*, a base word *main*, and a suffix *an*. Accordingly, if a word can be segmented into morphemes and these morphemes are added to the vocabulary list, the word embedding performance can be optimized. However, there were many morphological rules in Bahasa, such as inflectional and derivational affixes. This implies that all rules in word segmentation will increase the computational complexity. Moreover, the rules were not regular and similar for all words in Bahasa.

This study aims to build a model text representation for Bahasa that can handle OOV and carry semantic and syntactic information. However, the study also wants to maintain simplicity in the computational process; therefore, the study did not implement complex entirely linguistic morphological supervised rules for Bahasa. Instead, the study modified a simple BPE algorithm to generate morpheme embedding appropriate for the morphology of Bahasa. The main contribution of this study is the proposal of a morpheme embedding model for Bahasa that can handle OOV and can carry semantic and syntactic information. To the best of our knowledge, word representation considering the appropriate morpheme information for Bahasa is not yet available.

II. RELATED WORKS

Generating word embedding based on a subword requires a word segmentation process to split a word into its subword. There are several methods for word segmentation, from fully supervised methods such as CHIPMUNK [24] to unsupervised models such as Morfessor [25], BPE [14], Fasttext [15], and Language Model (LM) unigram [19].

Research by Creutz and Lagus [26] focused on observing word segmentation that matches the language’s morphology. The method is unsupervised and does not depend on a particular language. This model consists of two stages: learning and implementation. The learning stage produces a segmentation model, and the implementation stage produces text segmentation based on the learning model. The model used a vocabulary list containing the lexicon of morphemes and grammars and the lexicon’s frequencies in the corpus. The model uses the Viterbi algorithm to find the most likely segmentation of a word into a sequence of morphs. This research uses Finnish and English corpora. This study focuses more on segmentation and does not discuss the representation of subwords in embedding.

Several previous studies [8], [14], [19], [27] proposed subword segmentation to anticipate the problem of rare words or OOV in a closed vocabulary for neural machine translation (NMT). Based on the assumption that a word’s meaning can be reconstructed from its parts, research on NMT also focuses on algorithms for segmenting words into subwords.

Research by Sennrich et al. [14] is arguably one of the pioneers in implementing BPE for word segmentation. BPE was initially a simple data compression method proposed by Gage [28], but it can be utilized for word segmentation. BPE works in a simple way to generate a vocabulary list based on a combination of several characters that appear most often
in the learning corpus. The vocabulary formation process was repeated until the size of the vocabulary was reached. The vocabulary size is a parameter initialized by the user. This method is proven to anticipate rare and unknown words because the encoding value for unknown words can be formed from the subword values. The results of this study show that BPE is a suitable word-segmentation strategy for neural network models. The BPE model has proven to be simple but robust and can be applied to various languages. It does not require supervised data and does not require tokenized processes. This method has also become the forerunner of other subword methods. However, the drawback of this model is that the formation of subwords is only based on the frequency, and the result of segmentation of each word only consists of one unique segmentation.

Fasttext by Bojanowski et al. [15] is a study to anticipate rare words that are not in the vocabulary for word representation formed at the word level, such as word embedding by Mikolov [16] and GloVe by Pennington et al. [17]. By believing that every word can be broken down into a bag of character n-grams, this study represents word vectors based on the sum of these n-gram representations. Furthermore, the subword in the form of character n-grams is trained using the SkipGram model [16]. The evaluation result of the subword representation generated by fastText is better than that of traditional word embedding.

The word segmentation method expressed by Kudo [19] is the unigram LM method. The motivation of this method is to anticipate the BPE problem, which only produces a unique sequence segmentation. The Unigram language model allows segmentation regularization, where each sentence can produce several subword choices. The unigram LM library was built under the name SentencePiece [29]. The aim is to take advantage of segmentation ambiguity as noise to improve the robustness of NMT. Further calculations were carried out to determine the best segmentation sequence from several options. Compared to BPE, this method is more complex because it consists of several complex stages. The initial stage is to form a vocabulary seed from the training corpus until the vocabulary size is met. The Viterbi and EM algorithms generate several subword sequences of a word to fix the vocabulary set. This method also maintains subwords in-unit characters to anticipate an OOV that cannot be formed from subwords.

Research by Heinzerling and Strube [30] generated BPEmb, a collection of pre-trained subword embeddings in 275 languages, including Bahasa Indonesia. The purpose of this study is to provide pre-trained subword embedding using a BPE approach for general domains. This study utilizes Wikipedia data for 275 languages, but only a few languages are observed in detail due to the variety of Wikipedia data for these various languages. This study focuses on the effect of the number of BPE merge operations and the embedding dimensionality. The model evaluation was carried out only for the top five high-resource languages, two high-resource languages without explicit tokenization, and eight medium-to low-resource Asian languages. However, due to observation limitations, Bahasa is not included in this category, so the evaluation for Bahasa Indonesia is not explained further.

Research by Provilkov et al. [20], namely BPE dropout, also anticipates the drawback of the original BPE method, which can only produce a unique sequence for subwords. This study also anticipates the complexity generated by Kudo [19] research on the production of regularization subwords. This study modified the original BPE method, which stochastically corrupts the BPE segmentation procedure, leading to the production of multiple segmentations within the same fixed BPE framework. This method is more straightforward than the LM unigram method.

Research by Zhang et al. [18] proposes a unified subword-augmented embedding framework. This research focuses on NLP textual entailment and machine reading comprehension (MRC) tasks. In segmenting words into subwords, this study performs two stages: goodness measurement and segmentation. Goodness measurement evaluates how likely a subword is appropriate. The segmentation step is for the decoding algorithm. The investigations for the three models of goodness measurement were frequency, accessory variety, and description length gain. The decoding algorithms were Viterbi, maximal matching, and BPE. The study results show that subword-augmented embedding significantly improves various types of text understanding tasks in both English and Chinese benchmarks.

Research by Wu and Zhao [31] extended the original BPE style segmentation to a general unsupervised framework with three statistical measures: frequency (FRQ), accessor variety (AV), and description length gain (DLG).

Some previous relevant studies were also conducted by researchers focusing on specific languages with additional morpheme information. This method is also known as morphology or morpheme embedding. In addition to being a solution to OOV problems, morpheme embedding can also handle word sense and word ambiguity for a specific language [32]. Several studies on morphology embedding that focus on a particular language have been conducted in Portuguese [33], German [34], and Swedish [35]. These studies generally showed promising results. Additional morphological information has also proven helpful for languages with complex morphology, such as Turkish [36], Hebrew [37], and Arabic [32].

III. BPE WORD SEGMENTATION

BPE is a method for data compression introduced by Gage in 1994 [28]. BPE for data compression works by iteratively replacing the sequence of bytes that appears most frequently with an unused single byte. Based on how the BPE works as data compression, it can also be implemented for word segmentation by replacing the sequence of bytes with sequences of characters. BPE as word segmentation was first proposed in [14]. In the initial step, each end of the word in the corpus was marked with special characters. A unique character is
placed at the end of a word, making it easier to restore its original form. For efficiency, the BPE does not consider pairs of crossword boundaries. Further, the BPE splits the entire sentence in the corpus into individual characters. The most frequent adjacent pairs of characters were then merged consecutively. For example, the character symbol (‘A’, ‘B’) is replaced with the symbol AB. This process is called the merge operation process, which produces a new symbol representing a character n-gram [14]. The process is iterated until it reaches the desired vocabulary size and yields a merge list that contains the subword pattern. Further, this subword pattern can be applied to new words, even though the words were not in the training corpus. The BPE only counts the frequency of characters. The most frequent adjacent pairs of characters will be joined early, and common words are compounded as unique symbols. A larger vocabulary size will generate a more merged operation. For character sequences that rarely appear, these adjacent characters become new subwords. The BPE algorithm is described in the following steps in Algorithm 1.

**Algorithm 1 Learn BPE Word Segmentation**

**Input:** List of the word and its frequency $D_1 = \{W_i; \text{freq}\}$, Vocabulary Sizes VS  
**Output:** Vocabulary List $ML$  
1: Initialize a variable $n = 0$  
2: For $n$ to VS:  
3: For $W_i$ in $D_1$:  
4: Split $W_i$ into characters save in list $V$  
5: For $i$ in len($V$)-1:  
6: Pairing $[V[i], V[i+1]]$ as bigram  
7: Calculate co-occurrence bigram in $D_1$ save in $f$  
8: Add pairs bigram and $f$ in new Dictionary $D_2 = \{\text{bigram}; f\}$  
9: Next $i$  
10: Next $W_i$  
11: Get max $f$ for bigram in $D_2$  
12: Merge bigram $I_i$, $I_{i+1}$ add into list symbols  
13: Next $n$  
14: return $V$

### IV. BAHASA INDONESIA MORPHOLOGY

Based on typology, Bahasa is classified as an agglutinative language. In this type, affixation plays an essential role in the morphological process, where prefixes and suffixes are added to base words to form new words. Typology is a method of dividing language based on its morphological structure. Although the purpose of dividing languages based on typology is to find similarities between languages, each language still has its uniqueness, even though it is in the same typology. For example, languages classified as agglutinative are Turkish, Japanese, and Finnish, but they are very different from Bahasa in terms of morphology and grammar. For languages with agglutinative types, the separation of affixes from base words is necessary.

A morpheme is the smallest linguistic and grammatical unit of language that has meaning. Morphemes are language-dependent on certain grammatical rules because each language has its own rules. Some examples of affix derivation in Bahasa are \{meng-\} {-i}, \{ter-\}, \{ber-\}, \{ber-an\}, and \{-kan\}. Bahasa adheres to the subject-verb-object (SVO) language form.

In Bahasa, a word can be formed by one free morpheme, namely the base word, and a word can also be formed from one base word with several bound morphemes such as affixes. Registering a basic word dictionary as a reference is not a good option for agglutinative language because there are many variants for a word in Bahasa. With affixes, basic words can have morphological changes that have derivational and inflectional effects. Therefore, linguistically, observation based on morphology is better than generating and recognizing different word forms in a much larger number than only relying on corpus training [25]. It is necessary to understand the linguistic morphology of Bahasa in generating subword embeddings for Bahasa.

In Bahasa, inflectional transformation can change a class of words in which several additional affixes can change the basic words into different categories. For example, a verb can be changed into a noun after suffix affixes such as \{makan-an\} \rightarrow \{makanan\} (in English eat \rightarrow food), \{main-an\} \rightarrow \{mainan\} (in English play \rightarrow toy), and \{ayun-an\} \rightarrow \{ayunan\} (in English swing \rightarrow swinging). However, not all base words have a similar inflection point. Therefore, a stemming algorithm needs to determine the inflectional form of unregular base words. The stemming algorithm in Bahasa is mostly based on a rule-based algorithm. A previous study revealed that the syntactic regularity of Bahasa is more complicated than that of English because Bahasa uses more complex affixes than English [38]. In Bahasa, affixes contain prefixes, suffixes, infixes, and confixes. Infixed and confixes are combinations of prefixes and suffixes. For example, the verb \{perintah\} (in English: order) can have an affix -\{nya\}; \{intahnya\} (Noun) (in English: her/his order) but can also have prefix \{di\} \rightarrow \{diperintah\} (in English: ordered by), and confixes \{pemintah\} (in English: Government). There are also some absorption affixes from foreign languages, such as \{pro\} and \{anti\}.

#### A. INFLECTIONAL AFFIXES

Inflectional affixes refer to the formation of a new form of the same word. There are two types of inflections in Bahasa [39]:

1) **PARTICLE**

A particle is a type of word in Bahasa with a grammatical meaning but does not have a lexical meaning. Particles were used to affirm the predicate. Particles can be added to the endings of verbs and nouns. For example, the word \{minum\} (in English: drink) becomes \{minumlah\} (in English: drink it!). In particular, for particle \{pun\}, the writing is sometimes separated except for certain conjunctions, such as \{meskipun\} (in English: although) and \{walaupun\} (in English: although). The particles in Bahasa are \{kah\}, \{.lah\}, \{.-tah\}, and \{.-pun\}.
2) POSSESSIVE PRONOUNS
Possessive pronouns in Bahasa are words with grammatical constructions used to indicate a relationship between ownership and belonging. For example, *bukaku* (my book), *bukumu* (your book), *bukunya* (his/her book). In Bahasa, there are also several forms of compound inflection, such as *bukumulah* (your book). The possessive pronouns in Bahasa are “-ku,” “-mu,” and “-nya”.

B. DERIVATIONAL AFFIXES
Derivational is the formation of a new word from another word through derivational affixes. In Bahasa, there are two types of derivational affixes.

1) DERIVATIONAL PREFIXES
Based on the Bahasa morphology for derivational prefixes, addition prefixes could change the word type. For example, adding the prefix *me-* in some nouns would change the word type into a verb. Some words have prefix adaptations depending on the first letter of the word. For example, the first letter of a word is *s*, then *me-* becomes *meny-* and the noun *sapu* (in English: broom [noun]) becomes the verb *menyapu* (in English: broom [verb]). Another example is for a verb *pakai* (in English: use) addition with prefix *pe-* to be a noun called *pemakai* (in English: user). Sometimes, several derivational prefixes can be added at once as a word, for example, *sepentetahuan* (in English: to my knowledge). The prefixes of derivational *di* Bahasa are “be-,” “di-,” “ke-,” “me-,” “pe-,” “se-,” and “te-.”

2) DERIVATIONAL SUFFIXES
A derivational suffix is an addition morpheme at the end of a word. For example, *cinta* → *cintai* (in English: love [noun] → love [verb]), *lapor* → *laporkan* (in English: report → report it), *lapor* → *aporan* (in English: report [verb] → report [noun]). In these examples, the morpheme can change the word type from a noun to a verb. The suffixes of derivations in Bahasa are “-i,” “-kan,” “-an.”

3) DERIVATIONAL CONFIXES
In Bahasa, a basic word can also have a prefix and suffix at once, namely confix. For example, *dalam* → *mendalami* (in English: explore), *laksana* → *melaksanakan* (in English: implement), and *dedikasi* → *didedikasikan* (in English: dedicated). Derivational suffixes in Bahasa are “be-an,” “me-i,” “me-kan,” “di-i,” “di-kan.”

V. METHODOLOGY
The main difference between conventional word embedding and our model is that conventional word embedding is word-level embedding, whereas our model is a subword-level embedding. We implemented the modification of the BPE algorithm so that the segmenting word into a subword process is appropriate with morphemes of Bahasa. We also keep the model simple so that the model will not be as complex as a fully supervised morphology rule-based model. There are six detailed steps for generating morpheme embedding, and the methodology is shown in Fig. 1.

A. CORPUS PREPARATION
The first step was corpus preparation. The corpus is used as a medium to extract vocabulary and semantic and syntactic relations. In this study, we generated a corpus from Wikipedia in Bahasa. There were some steps to prepare the corpus: cleaning and removing the HTML and XML tags, transforming all words into a lower case, converting all numbers into symbols, and removing all punctuation. To separate the sentences in the corpus is marked with a new line, or ‘
’.

B. INFORMATION EXTRACTION
The second step is information extraction. Information extraction is a step in extracting the required information from the corpus. The target of this step is to generate a vocabulary-frequency list and a word-pair list. The first process involved tokenization. Tokenization is identified by white space, considering that Bahasa uses white space to separate tokens or words. Furthermore, all different words from the corpus are listed in the list. We also extracted the frequency of each word appearing in the corpus. This information is essential in generating a vocabulary-frequency list. This study implements a dictionary function that produces key and value pairs, where the key is the word *w* and the value is the number of frequencies *f*.

For *n* is the number of words in the corpus *K*. Where *K* = \{ *w* = \{ *w* \_1, *w* \_2, *w* \_3, ..., *w* \_n \}. The excerpt of the results of this step can be seen in Table 1.

| Tokens | Frequency |
|--------|-----------|
| dan    | 2,461,380 |
| yang   | 2,332,464 |
| di     | 1,852,169 |
| pada   | 1,172,840 |
| dari   | 1,179,450 |
| ini    | 968,285   |
| dengan | 915,142   |
| dalam  | 758,690   |
| untuk | 747,257   |
| bolanggi | ...     |

The word-pair list contains information about the word pair between the target word and the context word from the corpus. Adhere to philosophy: A word is characterized by the company it keeps, or in other words, a word semantic can be extracted from their word contexts; thus, this list will be used as a reference in generating morpheme embedding. The word-pair lists are generated using Algorithm 2.

The input parameters are *words* and *C*. *Words* are tokens in the corpus, and *C* is the number of context words counted from the target word. *C* is also known as the window size. This function produces a listing that contains a list of target words paired with their context words. If a center word is *w* \_i
and a looping process is carried out to find context words for all sentences, then the context words are $X_i = (w_i-c_1, w_i-c_1+1, w_i+c_1, w_i+c_1+1)$. If the number of $C$ is initialized as 2, then the listing is $[w_i], [w_i-c_1, w_i-c_1+1, w_i+c_1, w_i+c_1+1]$. For example, in sentences “fakultas ilmu komputer dan teknologi informasi,” word pair list between target word “komputer” and its context words for $C = 2$ is: ['komputer', ['fakultas', 'ilmu', 'dan', 'teknologi']].

C. MORPHEMES INITIALIZATION
Morpheme initialization is used to determine the n-gram adjacent character list that is eligible as a subword. In this study, the subword must be appropriate to the morpheme of Bahasa. We list all the affixes of Bahasa on the list described in Table 2. We only implemented morphological Bahasa rules based on affixes. The aim is to keep the method as simple as possible. Some rules, such as prefix adaptations, are not implied because they require a complex stemming and lemmatization algorithm implementation. We also excluded affixes from foreign languages. However, the model can still identify a complex affix such as confixes.

D. BPE LEARNING OPERATION
The next step is the BPE learning operation to generate the vocabulary list. The vocabulary list contains training words

FIGURE 1. Methodology.
Algorithm 2 Generate The Word-Pair List

Input: List of words, words
Window size, C
Output: A word-pair list of target and context words, context_words, center_words
1: i = C
2: While i < len(words) - C do
3:   center_words = words[i]
4:   context_words = words[(i - C): i] + words[(i + 1): (i + C + 1)]
5:   yield context_words, center_words
6:   i = i + 1
7: End While

Algorithm 3 Additional Step for BPE Learn Operation

Input: original Vocabulary List OV, Filtered List, FL
Output: new Vocabulary list, V
1: n = 1
2: For n to len(OV) do:
3:   If (2 <= len(OV[n]) <= 4) then:
4:     If FL[n] not in FL
5:       Remove FL[n]
6:       Update OV
7:       V = OV
8:   End if
9: End For
10: return V

TABLE 2. Affixes of Bahasa list.

| Prefixes | Suffixes |
|----------|----------|
| men      | an       |
| ke       | i        |
| ber      | kan      |
| di       | nya      |
| pen      | kah      |
| ter      | lah      |
| se       | ku       |
| pe       | pun      |

E. WORD ENCODING

Word encoding is a process used to convert words in the vocabulary list into vectors. First, the encoding mechanism for words is initiated by generating unique indexes for all tokens in the vocabulary list. Furthermore, all these words were trained using the GloVe algorithm [17]. This step yields a matrix that represents the co-occurrence of adjacent words or subwords. For example, calculating the embedding J to the target subword S, is Xij. Implementing in a case subword pe with context $ yields matrix Xpe,anS. The matrix is a description amount of co-occurrence subword anS, and subword pe appears once. To avoid the complex calculation of co-occurrence for each token, we implemented a loss function formula, as in equation 1, to calculate the word encoding J.

$$ J = \sum_{i,j=1}^{V} f(X_{ij})(S_i^T S_j + b_i + b_j - \log(X_{ij}))^2 $$

b_i and b_j are offset parameters.

The motivation to use the GloVe algorithm is to obtain global information in the form of frequency occurrence. In this study, the training vocabulary was limited to 100,000 vocabularies. The word encodings were stored in a GloVe model format. This model can be loaded quickly during the word segmentation process. For further steps, all tokens need their encoding form, and the model works as an index to find the pair of the token and its encoding form.

Particularly for rare and OOV words not in the vocabulary list, the words will be broken down into subwords. The word is broken down into prefixes, basic words, and suffixes. For example, the word *permainan* can be split into prefix per-, base word main, and suffix an. The encoding for the whole word *permainan* is formed based on the composition encoding per + main + an. Based on this method, the encoding of other words not in the training corpus but have the same affixes as the prefix per and the suffix a still can be predicted. Further, the word encoding for OOV is obtained based on the composition encoding for a whole word, which encodes prefixes, basic words, and suffixes. Each word K is formed from the subword Sk, where $S_k = (s_k)_1^n$ so that $K = \sum S_k$. This study uses the addition function or the sum

and word segmentation patterns for rare or OOV. In this step, we modified the original BPE algorithm with an additional step: the vocabulary-filtered step. First, we implemented the original BPE algorithm to train the Vocabulary-frequency list generated from the previous stage. We initialized a 10,000 vocabulary size. This stage generates a 10,000 vocabulary list. Unlike the original BPE learning operation step, we further filtered the original vocabulary list with additional rules described in Algorithm 3. We filtered the list based on their length size and their presence in the morpheme list of Bahasa, as shown in Table 2. The range of size is between 2 and 4, considering that in Bahasa, most of the morphemes are between the range. Moreover, the filter is addressed to Bahasa affixes, which have a length of no more than four characters. We keep the single characters to prepare the rare words that occur in the training corpus, which can be constructed from characters. We removed words that occurred less than 5 in the entire corpus. Further, the new vocabulary list is indexed based on alphabetical order to make it easier to manage in a further step.

For example, the algorithm is applied for three words of Bahasa, which are *permainan*, *peraturan*, and *pertemanan*. The algorithm yields subwords *perma* $an$, *pe* $rata* $ruan* $per* $tem* $a* $na* $per* $main* $an* $per* $atur* $an* $per* $tem* $an* $an* $per* $main* $an* $per* $atur* $an* $per* $tem* $a* $na* $per* $main* $an* $per* $atur* $an* $per* $tem* $an* $an* $per* $main* $an* $per* $atur* $an* $per* $tem* $a* $na* $per* $main* $an* $per* $atur* $an* $per* $tem* $an* $an* $per* $main* $an* $per* $atur* $an* $per* $tem* $a* $na* $per* $main* $an* $per* $atur* $an* $per* $tem* $an* $an* $per* $main* $an* $per* $atur* $an* $per* $tem* $a* $na* $per* $main* $an* $per* $atur* $an* $per* $tem* $an* $an* $per* $main* $an* $per* $atur* $an* $per* $tem* $a* $na* $per* $main* $an* $per* $atur* $an* $per* $tem* $an* $an* $per* $main* $an* $per* $atur* $an* $per* $tem* $a* $na* $per* $main* $an* $per$
of all subword encoding values to produce word encoding. The illustration for generating the final word encoding is shown in Fig 2.

F. MORPHEME EMBEDDING

Morpheme embedding is the process of generating the embedding value of each word in the vocabulary list. In this study, we implemented the Word2vec SkipGram algorithm [16] to generate the embedding. The encoding of each word as input for layer input and as predicted output in layer output is taken from the word encoding model in the previous step. Illustration word embedding generation is shown in Fig 3. In the Word2vec model, \( h \) is the output between the input layer and the hidden layer obtained by the formula \( h = W^T X_k \), where the value of \( W \) is the weight value input randomly with the number of dimensions. In this study, 300 dimensions were used. Unlike conventional word embedding, where word representation for the input layer uses one-hot-encoding, in this study, we use word encoding generated from the previous process. Output \( h \) in the form of a matrix with a size of \( V \times 300 \), and \( V \) is the amount of vocabulary.

Output \( Y \) is calculated by the formula \( Y = W'h \); then, the softmax equation is implemented to calculate the posterior word distribution in the vocabulary. The softmax equation for \( P \) is given by Equation 2.

\[
P = \frac{(w_t|w_{t-1}, \ldots, w_{t-n+1})}{\sum_i e^{y_i}}
\]  

The optimization of \( W \) and \( W' \) values was carried out based on the actual \( Y \) values in the word-pair list. Optimization was carried out using the stochastic gradient descent (SGD) formula with a learning rate of 1. In this study, the extracted word-embedding value was the \( W \) value. The determination of several hyper-parameter values was based on the results of several previous studies. The result of this process produces a pre-trained word model in a vector format.

VI. RESULT AND EVALUATION

The main difference between conventional word embedding and our model is that conventional word embedding is word-level embedding, whereas our model is a subword-level embedding. The results of this study are a model to generate a feature representation for Bahasa. Unlike conventional word embedding at the word level with limitations in handling OOV, our method attempted to solve this limitation by implementing features at the subword level. The subword is obtained by a word-segmentation process using the BPE algorithm. Unlike conventional BPE for subword segmentation, our method attempted to remove unwanted junk tokens obtained by the BPE algorithm, which are inappropriate for Bahasa morphology. We keep all subwords obtained by BPE in the eligible list as morphemes of Bahasa. We removed tokens with occurrence frequencies below 5. This process removes misspelling words and removes foreign words from the corpus. The results showed that vocabulary size reduced by approximately 20%, from 10,000 to 7660. We evaluate our subword embedding in three evaluations: how good the model segmented word and handled OOV, how good the model captures semantics, and how good the model captures the syntactic.

**A. WORD SEGMENTATION EVALUATION**

In word segmentation evaluation, two scenarios were applied. We test word segmentation for the 100 most frequent words in the training corpus using our model in the first scenario. We also implemented the original BPE word segmentation as a baseline. The results of the original BPE and modified BPE yield the same result. The words were not split into subwords because they were in the vocabulary list. In the second scenario, we trained the model with fewer vocabulary sizes. Both models tested rare words that were not included in the vocabulary list. We also tested misspelling words and sticking tokens that were not tokenized, such as rumahkupengakunanya and jayalahkemakmuranindonesiaku. Detailed evaluation results are presented in Table 3.

Based on Table 3, both models yield the same word segmentations for some words, such as melupakun, meglupakan, medan, and medan. However, for other sample words, both models yielded different word segmentation. In our model, the subword is split into the morphemes of Bahasa. Meanwhile, in the original BPE, the result was more random.
Our model yields result as expected, where it can handle OOV like the original BPE but is also more appropriate with the morphology of Bahasa. This scenario proves that the modified BPE yields the same result with fewer vocabulary sizes than the original BPE.

**B. SEMANTIC EVALUATION**

Ideally, the evaluation of text representation in capturing semantics uses extrinsic evaluation. Extrinsic evaluation involves implementing the representation in downstream NLP applications, such as machine translation, classification, or clustering. However, extrinsic evaluation is very time-consuming, inefficient, and expensive. Another method is intrinsic evaluation, which evaluates specific intermediate subtasks, such as analogy task completion. Generally, the analogy task is built on categories such as capital, big cities, countries, islands, world figures, hypernyms, hyponyms, meronyms, and else. However, intrinsic evaluation requires a collection of word analogies for each category. Unfortunately, the provided standard word analogies task set is language-dependent and not yet provided for Bahasa. Therefore, we modified the English evaluation benchmark [40] and [41] to fit the word-embedding model for Bahasa. We calculate and find the ten nearest neighbors of these words using the Euclidean distance, as in equation 3.

\[ D(a, b) = \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2} \]  

where D is the distance between the word a and the embedding of word b, n is the number of embedding dimensions, and i is the value of each dimension. Words with close relatedness will also have a close embedding value. The distance D between two closely related words is closer to zero. For example, the embedding of chicago will be closer to embedding another big city in the world, such as washington and kansas. The intrinsic evaluation contains some categories to reveal the relatedness between words, as described in Table 4. We utilized human judgment or the gold standard to state the expected nearest words for each random word in the evaluation step. We performed experiments with the ten most frequent words for each category and expected among these nearest neighbors appropriate with the expected result of the gold standard. The category and expected results are shown in Table 4.

We performed experiments using the original BPE and our model. The result is the original BPE, and our model yields the same result. For example, one of the most frequent tokens for Big Cities Category in the vocabulary is washinton. Both models obtained the same result for the neighbors words: chicago, houston, kansas, smithsonian, dulles, irish, seattle, washington, baltimore. We got eight expected results from 10 nearest words of the Big Cities Category for token washington. Based on the gold standard determination, dulles and
A. Amalia et al.: Morpheme Embedding for Bahasa Indonesia Using Modified Byte Pair Encoding

TABLE 5. The example result of the semantic evaluation.

| Input Tokens | Category   | Obtained Result               |
|--------------|------------|-------------------------------|
| washington   | Big city   | smithsonian, dulles, irish   |
| jakarta      | Capital    | semarang, surabaya, java      |
| indonesia    | Country    | aggeris (United Kingdom)     |
| kakek (grand father) | Family Member | cucu (grand son), ayah      |
| presiden (President) | World Figure | presiden (president)        |
| sumatera | Island | medan, palembang, lampung |
| reptil (reptile) | Hypernym | martim (maritime), kualistian |
| warna (colors) | Hypernym | hijau (green), merah (red)  |
| baja (steel) | Meronym | zaria, aeng, tupelo, irang |

TABLE 6. Syntactic evaluation.

| Input Tokens | Affixes Categories | Expected Result |
|--------------|--------------------|-----------------|
| smithsonian  | Prefix             | Prefix = another token |
| kakek (grand father) | Confix           | Prefix = another word + suffix |

TABLE 7. The example result of the syntactic evaluation.

| Input Tokens | Affixes category | Obtained Result |
|--------------|------------------|-----------------|
| teringgi (highest) | Prefix (ter)    | teringgi (highest) |
| bukunya (her/his book) | Suffix (_nya) | bukunya (her/his song) |

FIGURE 4. Semantic Evaluation Result.

smithsonian is not the name of Big Cities and does not fulfill the expected result. Furthermore, we tried another input token for Big Cities Category and took the average expected result to determine the evaluation accuracy. The same experiments were done for another category. The examples of another input token and the result are shown in Table 5. We mark words that do not meet the gold standard with bold and underlines font format. The evaluation accuracy for each category are presented in Fig. 4.

Based on Fig 4, we can see that the model yields good results in some categories. The model obtains the best result in Big Cities, Capital, Countries, Family Members, Islands, World Figures, and Hypernyms. However, our model does not yield good results for Hyponym and Meronym.

C. SYNTACTIC EVALUATION

We also evaluated the words with the most frequent affixes tokens using the same method as semantic evaluation; the expected result is whether the model can capture the other words with similar affixes. We evaluated ten of the most frequent affix tokens for each syntactic category. The syntactic categories are prefix, suffix, and confix. The syntactic category and the expected result described in Table 6. Similar to the result of the semantic evaluation, the modified BPE and original BPE yielded the same result. The examples of input tokens and the obtained result are shown in Table 7, and the evaluation accuracy results are presented in Fig. 5. Based on Fig. 5, we can see that most of the obtained results are consistent with the expected result. The model can produce word embedding using syntactic information. The word with affixes has an embedding that is close to another word that
This paper investigates a simple method to learn word representation by considering subword information appropriate to Bahasa morphology. We modified a simple BPE algorithm to produce subwords, and we filtered the results using the list of Bahasa morphemes. Our model has proven that it can handle OOV and is also capable of carrying semantics and syntactic information in the embedding value of words. This method produces a word representation model for Bahasa Indonesia with a more efficient vocabulary size list because the junk tokens from the BPE segmentation process are eliminated. However, to simplify the computation, not all the morphology rules of Bahasa are implemented. One of the advantages of our model is that it does not require large amounts of complex computations, such as the stemming and lemmatization process in determining the base words. However, the model still has limitations in identifying prefix absorption. Word segmentation into subwords is not always accurate, especially for affixes with irregular inflectional and derivational absorption. Future work is needed to compare the model with the model using the full morphology rules. More in-depth observations are needed to determine hyperparameters such as the number of dimensions, learning rate, and subword encoding algorithms to improve the performance of the subword embedding model. Future work is also needed to produce an appropriate quantitative evaluation like an intrinsic evaluation for word representation in Bahasa Indonesia.

**REFERENCES**

[1] R. Salakhutdinov and G. Hinton, “An efficient learning procedure for deep Boltzmann machines,” *Neural Comput.*, vol. 24, no. 8, pp. 1967–2006, 2012.

[2] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, “Why does unsupervised pre-training help deep learning?” *J. Mach. Learn. Res.*, vol. 9, pp. 201–208, Jan. 2010.

[3] W. Ling, Y. Tsvetkov, S. Amir, R. Fernandez, C. Dyer, A. W. Black, I. Trancoso, and C.-C. Lin, “Not all contexts are created equal: Better word representations with variable attention,” in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1367–1372.

[4] M. Bansal, K. Gimpel, and K. Livescu, “Tailoring continuous word representations for dependency parsing,” in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, 2014, pp. 809–815.

[5] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[6] M. Popov, B. Kulnitsky, I. Perezhizhin, V. Mordkovich, D. Ovsyannikov, S. Perfilov, L. Borisova, and V. Blank, “Catalytic 3D polymerization of C60,” *Fullerenes, Nanotubes Carbon Nanostruct.*, vol. 26, no. 8, pp. 465–470, Aug. 2018.

[7] Y. Xu and J. Liu, “Implicitly incorporating morphological information into word embedding,” 2017, arXiv:1701.02481.

[8] M. T. Luong, R. Socher, and C. D. Manning, “Better word representations with recursive neural networks for morphology,” in *Proc. 17th Conf. Comput. Natural Lang. Learn.*, 2013, pp. 104–113.

[9] E. Dusserre, “Bigger does not mean better! We prefer specificity,” in *Proc. 12nd Int. Conf. Comput. Semantics (IWCS)*, 2017, pp. 1–6.

[10] E. Altszuler, S. Ribeiro, M. Sigman, and D. F. Slezk, “Comparative study of LSA vs Word2vec embeddings in small corpora: A case study in dreams database,” *Conscious. Cogn.*, vol. 56, pp. 178–187, Oct. 2017.

[11] Y. Gu, G. Leroy, S. Pettygrove, M. K. Galindo, and M. Kurzius-Spencer, “Optimizing corpus creation for training word embedding in low resource domains: A case study in autism spectrum disorder (ASD),” in *Proc. AMIA Symp.*, 2018, pp. 508–517.

[12] S. Park, J. Byun, S. Baek, Y. Cho, and A. Oh, “Subword-level word vector representations for Korean,” in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 2429–2438.

[13] X. Zhang and Y. LeCun, “Which encoding is the best for text classification in Chinese, English, Japanese and Korean?” 2017, arXiv:1708.02657.

[14] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 1–11.

[15] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, Dec. 2017.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” 2013, arXiv:cs/1301.3781.

[17] J. Pennington, R. Socher, and C. D. Manning, “GLOVE: Global vectors for word representation,” in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[18] Z. Zhang, H. Zhao, K. Ling, J. Li, Z. Li, S. He, and G. Fu, “Effective subword segmentation for text comprehension,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 11, pp. 1664–1674, Nov. 2019.

[19] T. Kudo, “Subword regularization: Improving neural network translation models with multiple subword candidates,” in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 1–10.

[20] I. Provilkov, D. Emelianenko, and E. Voita, “BPE-dropout: Simple and effective subword regularization,” 2019, arXiv:1910.13267.

[21] K. Bostrom and G. Durrett, “Byte pair encoding is suboptimal for language models with multiple subword candidates,” in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, 2018, pp. 2429–2438.

[22] Y. Zhu, I. Vulić, and A. Korhonen, “A systematic study of leveraging subword information for learning word representations,” in *Proc. Conf. North.*, 2019, pp. 1–21.

[23] V. Fromkin, R. Rodman, and N. Hyams, *An Introduction to Language*. Cengage Learn., 2018.

[24] R. Cotterell, T. Müller, A. Fraser, and H. Schätzle, “Labeled morphological segmentation with semi-Markov models,” in *Proc. 19th Conf. Comput. Natural Lang. Learn.*, 2015, pp. 164–174.

[25] M. Creutz and K. Lagus, “Unsupervised models for morpheme segmentation and morphology learning,” *ACM Trans. Speech Lang. Process.*, vol. 4, no. 1, pp. 1–34, Jan. 2007.

[26] M. Creutz and K. Lagus, “Unsupervised morpheme segmentation and morphology induction from text corpora using morfessor 1.0,” *Comput. Inf. Sci., Helsinki Univ. Technol., Otaniemi, Uusimaa, Tech. Rep. A81, 2005.

[27] D. Ataman, M. Negri, M. Turchi, and M. Federico, “Linguistically motivated vocabulary reduction for neural machine translation from Turkish to English,” *Prague Bull. Math. Linguistics*, vol. 108, no. 1, pp. 331–342, Jun. 2017.

[28] P. Gage, “A new algorithm for data compression,” *C Users J.*, vol. 12, no. 2, pp. 23–28, 1994.

[29] T. Kudo and J. Richardson, “SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,” 2018, arXiv:1808.09226.
[30] B. Heinzerling and M. Strube, “BPEMB: Tokenization-free pre-trained subword embeddings in 275 languages,” in Proc. 11st Int. Conf. Lang. Resour. Eval., 2019, pp. 2989–2993.

[31] Y. Wu and H. Zhao, “Finding better subword segmentation,” Chin. Comput. Linguist. Nat. Lang. Process., vol. 2, pp. 53–64, Dec. 2018.

[32] R. A. Salama, A. Youssef, and A. Fahmy, “Morphological word embedding for Arabic,” Proc. Comput. Sci., vol. 142, pp. 83–93, Dec. 2018.

[33] N. Hartmann, E. Fonseca, S. Christopher, M. Treviso, J. Rodrigues, and S. Aluisio, “Portuguese word embeddings: Evaluating on word analogies and natural language tasks,” 2017, arXiv:cs/1708.06025.

[34] R. Cotterell and H. Schütze, “Morphological word-embeddings,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., 2015, pp. 1287–1292.

[35] A. Basirat and M. Tang, “Lexical and morpho-syntactic features in word embeddings—A case study of nouns in Swedish,” in Proc. 10th Int. Conf. Agents Artif. Intell., 2018, pp. 663–674.

[36] R. Cotterell, H. Schätz, and J. Eisner, “Morphological smoothing and extrapolation of word embeddings,” in Proc. 34th Annu. Meeting Assoc. Comput. Linguistics, vol. 3, 2016, pp. 1651–1660.

[37] O. Avraham and Y. Goldberg, “The interplay of semantics and morphology in word embeddings,” in Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics, 2017, pp. 422–426.

[38] M. Adriani and H. E. Williams, “Stemming in-donesian: A confix-stripping approach,” ACM J. Educ. Resour. Comput., vol. 6, p. 33, Dec. 2007.

[39] A. M. Moeliono and S. Dardjowidjojo, “Tata Bahasa Baku Bahasa Indonesia (the standard Indonesian grammar),” in Proc. 10th Int. Conf. Lang. Resour. Eval., 2019, pp. 2989–2993.

[40] T. Mikolov, I. Sutskever, and K. Chen, “Distributed representations of words and phrases and their compositionality,” in Proc. Adv. Neural Inf. Process. Syst., vol. 1, 2013, pp. 3111–3119.

[41] A. Gladvkova, A. Drozd, and S. Matsuoka, “Analogy-based detection of morphological and semantic relations with word embeddings: What works and what doesn’t,” in Proc. NAACL Student Res. Workshop, 2016, pp. 8–15.

**AMALIA AMALIA** (Member, IEEE) received the bachelor’s degree in electrical engineering from Universitas Syiah Kuala (USK), in 2002, and the master’s degree in electrical engineering and the Ph.D. degree in computer science doctoral program from Universitas Sumatera Utara (USU), in August 2011 and October 2021, respectively. She is currently a Lecturer with the Department of Computer Science, USU, and works on research related to natural language processing.

**OPIM SALIM SITOMPUL** (Member, IEEE) received the Ph.D. degree in information science from Universiti Kebangsaan Malaysia, Selangor, in 2005. He is currently a Professor with the Department of Information Technology, Universitas Sumatera Utara, Medan, Indonesia. His skill and expertise are in AI, data warehousing, and data science. His recent projects are in natural language generation (NLP) and AIoT. The result of his work can be seen in his most recent publication is “Template-Based Natural Language Generation in Interpreting Laboratory Blood Test.”

**TEDDY MANTORO** (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the School of Computer Science, The Australian National University (ANU), Canberra, Australia. He is currently a Computer Science Professor with Sampoerna University, Jakarta, Indonesia. He has conducted intensive work in an intelligent environment that uses computational intelligence. He developed the concept and theory of context-aware computing for the intelligent environment, and as a proof of concept, he and his laboratory developed many prototypes that led to many awards. His research interests include information security, pervasive computing, and intelligent environment/the IoT. He received five Gold, nine Silver, and 11 Bronze medals from the National and International IT Innovation Competitions, since 2009.

**ERNA BUDHIARTI NABABAN** (Member, IEEE) received the Ph.D. degree in science and management systems from Universiti Kebangsaan Malaysia, Selangor, in 2010. She is currently working with the Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Indonesia. She does research in scheduling (optimization), artificial intelligence, machine learning, and natural language generation. Her most recent publication is “Template-Based Natural Language Generation in Interpreting Laboratory Blood Test.”