Contextual Embedding Comparison for Out-of-vocabulary Handling in Indonesian POS Tagging

Muhammad Alfian[1], Umi Laili Yuhana[*1], Daniel Siahaan[1], Harum Munazharoh[2], Eric Pardede[3]
[1]Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
[2]Department of Indonesian Language and Literature, Universitas Airlangga, Surabaya, Indonesia
[3]Department of Computer and Information Technology, La Trobe University, Melbourne, Australia
E-mail: 7025221023@student.its.ac.id [1], yuhana@its.ac.id [1], do.siahaan@its.ac.id [1], harum.m@fib.unair.ac.id [2], e.pardede@latrobe.edu.au [3]
*Corresponding author

Keywords: Out-of-vocabulary, part-of-speech-tagging, education, contextual embedding, low resource language

Received: August 28, 2025

Out-of-vocabulary (OOV) problems remain a significant challenge in part-of-speech (POS) tagging. These problems affect not only tagging performance, but also downstream tasks, particularly in educational case studies. This issue is related to the limited availability of datasets for low-resource languages (LRLs), the absence of representative features, and the complexity of grammatical variation. Current approaches perform well in recognizing patterned OOV words, but often fail with unpatterned OOV words, such as proper nouns and polysemous words. To address this issue, this study employs contextual embeddings to represent OOV words, improving model recognition. Two types of embeddings are compared: static embeddings (Word2Vec, GloVe, and FastText) and contextual embeddings (ELMo, BERT, and Flair). These embeddings provide appropriate representations for OOV words. We evaluate models using accuracy and the macro F1 score on a curated Indonesian corpus of 30,960 words. The model was evaluated using the k-fold cross-validation method with both OOV and in-vocabulary (IV) word scenarios. The results of the experiment show that models with contextual embeddings outperform those with static embeddings. Flair achieved the highest level of accuracy (95.65%), while BERT and ELMo achieved similar levels of 92.73% and 91.61% respectively. Our proposed model was effective in handling OOV cases, achieving an accuracy of 88.12%, which is a 25.15% improvement over the baseline model. However, it still struggles with redundant words and capitalized letters. Future research should explore integrating form-based and contextual information to improve performance.

Povzetek: Študija za označevanje besednih vrst v jezikih z malo viri izboljša obravnavo OOV z uporabo kontekstualnih modelov, ki prekašajo statične pristope in opozarjajo na potrebo po združitvi oblikoslovnih in kontekstnih znakov.

1 Introduction

Part-of-speech (POS) tagging is a fundamental task in Natural Language Processing (NLP) because it provides essential information about sentence structure that influences subsequent processing. POS refers to the grammatical categories of words, such as verbs, adjectives, adverbs, and nouns. In this step, each word is automatically assigned the appropriate syntactic category based on its context [1], [2]. POS tagging has been shown to enhance the performance of various NLP tasks, including term extraction [3], sentence parsing [4] and text summarization [5]. In the field of education, it is also applied as a preprocessing step for syntactic analysis, serving as input for automatic grammar correction models [6] and automatic question generation [4]. By identifying word classes, text analysis becomes more detailed and precise [7]. However, progress in POS tagging has slowed in recent years due to persistent challenges in handling out-of-vocabulary (OOV) words which remain difficult to predict accurately [8], [9]. OOV words are a common challenge in NLP, particularly in POS tagging. OOV refers to words that are absent from the training vocabulary but appear during testing [10]. This issue can significantly reduce the performance of POS tagging models. Addressing OOV not only enhances tagging accuracy but also improves the performance of other NLP tasks, including Named Entity Recognition (NER) [11], sentence parsing [4], text classification [5-7], text summarization [8-11], and machine translation [18].

As shown in our previous systematic literature review [19], the OOV phenomenon is prevalent in low-resource languages (LRLs) such as Indonesian [20], Yoruba [21], and Ainu [22]. The limited size of available corpora prevents models from fully learning word-class patterns in these languages, making them more sensitive to novel word forms. Several studies have also linked OOV challenges to the lack of representative features for word representation [8], while others highlight linguistic factors, particularly grammatical variations. OOV words may arise from language and spelling variations [9]. For instance, in Indonesian, the word *makan* ("eat") is a verb (VB), but with the affix *-an* it becomes *makanan* ("food"), which is labeled as a noun (NN). Such morphological processes often lead to OOV cases when derived forms are absent from dictionaries. OOV words can also emerge as neologisms driven by language evolution, such as emoticons (*":)"*), expressive forms (*"Haaaa"*), abbreviations (*"krn"*), and slang (*"Tq"*). With the rise of social media, OOV words increasingly originate from code-switching and dialectal mixing [23], [24], where words from languages or dialects not included in dictionaries are also likely to be treated as OOV.

Several researchers have proposed strategies for handling OOV, including preprocessing strategies [25], [26], [27], hand-crafted features [20], [28], [29], and learned features [29], [30], [31]. Preprocessing is typically carried out using two methods: misspelling correction [26], [27] and vocabulary extension [25], [26]. Misspelling correction is straightforward and effective but only addresses misspelled OOV words, leaving other types unresolved. Vocabulary extension can also mitigate OOV cases; however, it requires access to large corpora to improve model performance. In low-resource languages, expanding corpora is often costly and time-consuming.

Several other researchers use a hand-crafted features approach to handle OOV [20], [28], [29]. This approach is effective in recognizing OOV words arising from morphological processes and works well in morphologically rich languages (MRLs). However, it is limited in handling unaffixed OOV words such as neologisms and is less applicable to highly inflected languages. The state-of-the-art approach relies on learned features. Neural network–based methods have been widely used to capture word patterns at different levels, including word-level [30], subword level [29], and character level [31]. Among these, character embeddings are reported to be the most effective in recognizing OOV word forms based on character patterns. Nevertheless, this method is less effective in handling previously unseen characters or word forms absent from dictionaries, such as proper nouns and foreign words from other languages or dialects.

Previous research has shown that pretrained embeddings can help address OOV by providing accurate vector representations learned from large-scale corpora. For example, ELMo has been used to represent OOV words and was shown to improve model performance [32]. Other studies have primarily relied on static embeddings such as word2vec [30], GloVe [23], or Fasttext [33]. Among these, FastText is the most widely used, as it represents words based on subword information. However, FastText primarily relies on surface forms, without considering the contextual meaning of words within sentences. Contextual information is essential for predicting labels of words with different forms but the same meaning and class. For instance, the Indonesian words *diri* ("self") and *sendiri* ("self") share both meaning and word class. A contextual approach therefore has strong potential to overcome these limitations, yet it has received little attention in addressing OOV in POS tagging.

Therefore, this study proposes contextual embeddings to represent OOV words with contextual information. To the best of our knowledge, this approach is the first applied to handle OOV words in Indonesian. We compare static embeddings (Word2Vec, GloVe, FastText) and contextual embeddings (BERT, Flair, ELMo) to examine the impact of contextual representations on POS tagging performance. The comparison also aims to analyze the strengths and limitations of each embedding in handling OOV words and to identify the most effective pretrained model. To maximize performance, we conduct hyperparameter tuning on six aspects: embedding size, BiLSTM hidden size, dropout rate, training batch size, test batch size, and learning rate. This process is carried out to determine the optimal configuration that yields the highest performance. The study uses a corpus from Fu's research [34], which has been reselected and revalidated.

To address these goals, we formulate the following research questions:

RQ1: Does contextual embedding improve Indonesian POS tagging performance, especially for OOV words?

RQ2: Which embedding type achieves the best performance for OOV handling in Indonesian POS tagging?

RQ3: How can hyperparameter tuning enhance the performance of contextual embedding models in Indonesian POS tagging?

2 Previous works

Several researchers have explored strategies for handling OOV. This study classifies existing approaches into three categories: preprocessing strategies, hand-crafted features, and learned features. We summarize previous research in Table 1. A detailed explanation of each strategy is provided below.

2.1 Preprocessing strategies

A preprocessing strategy is applied to manipulate data before it is processed in the POS tagging model. Some researchers address OOV words by correcting misspellings [26], [27], [35]. Keiper et. al [26] and Jettakul et. al. [27] corrected spelling by replacing OOV words with the most similar words in the vocabulary, where word similarity is calculated using the Levenshtein distance method. Candidate words are selected based on the smallest distance and compared against a predefined threshold. Meanwhile, Millour and Fort [35] automatically generated spelling variant rules using AlphaMALIG to correct OOV words. This approach reduces the percentage of OOV from 24% to 22% [35]. However, spelling correction only succeeds for 29% of words, while 41% fail to be corrected, and the remaining 29% cannot be processed due to diverse noise levels in the data [27]. Preprocessing can be executed efficiently because of its simplicity, but its effectiveness is limited to a specific issue, requiring additional approaches to comprehensively address OOV challenges.

Table 1: Summary of previous works in handling OOV in POS Tagging

Method – Languages	**Categories – Metrics**	**Limitations**
Static Word Embedding [38] – Indonesian	Word Level – Accuracy	The model tends to mislabel minority classes and unknown words (noun or verb).
Multi-level OOV Handling [8] – English	Preprocessing strategies, Multi-level – Accuracy	The model reveals limitations in addressing misspellings and neologisms.
Hybrid HMM-BiLSTM [48] – Persian	Word Level – Accuracy	Hybrid models that rely on large-scale data pose challenges for low-resource languages.
Context Char Transformer Encoder (CCTE) [31] – English	Character Level – F1	The model is limited to OOV types beyond misspellings.
BiLSTM+CRF [23] – Bengali-Eng-lish, Hindi-English, Telugu-English	Multi-level - Precision, Recall, F1, Accuracy	The model struggles to distinguish between nouns, verbs, and proper names.
TSWR + OOVR [30] – Korean	Multi-level – Accuracy	The model requires substantial computational cost and prolonged training time.
Bayesian HMM [49] – Turkish, Hu-ngarian, Finnish, Basque, English	Morphology, Word Level – NMI, VI	The model is less effective for inflectional languages such as English.
Semi-supervised Tree Models [25] – Indo-European (8) and Uralic (1)	Vocab Extension, Morphology – Accuracy	The small corpora limit the learning of morphological rules for minority labels.
Semi-markov CRF [41] – Chinese, Vietnamese, English, Japanese	Character Level – F1, Accuracy	The model is limited to 23-character words and struggles with inflected languages.
ELMo [32] – 57 languages in UD	Character Level – Accuracy	The study does not provide detailed insights into the model's limitations.
Word and Character Embedding [33] – Italian	Multi-level – Accuracy	The model shows limited effectiveness for spoken or informal language.
Augmentation + Variation Rule [35] – Alsatian	Spelling Correction, Morphology – Accuracy	The model remains ineffective in handling short words (<4 characters).
Morph + CNN + Attention [29] – Greek	Orthography, Subword Level – Macro F1	The model remains less adaptive to neologisms and misspellings.
Multilingual POS Tagging [43] – English, Italian	Multi-level – Accuracy	The model remains limited in capturing spoken or informal language.
BERToov [50] – Indo-European (14), Uralic (1), Afro-Asiatic (1)	Subword level – Accuracy	The model distorts BERT's knowledge and its performance is inconsistent.
Two-level Backoff [42] – Thai, Chinese	Multi-level – Accuracy	The model suffers from overfitting with small mini-batch sizes.
BiLSTM-CRF [27] – Thai	Spelling Correction, Morpho-logy – Macro F1	The model exhibits high error rates, particularly in fully and middle random cases.
Probabilistic POS Tagging [21] – Yoruba	Morphology – Accuracy, Precision, Recall	Limited corpus size and tagset availability impact model performance
Morph + BiLSTM [28] – Russian	Morphological, con-text, word embedding – Accuracy	The model remains limited in neologisms, proper names, misspellings, and informal words.
TPANN [51] – English	Multi-level – Accuracy	The model struggles on ARK-Twitter due to proper names and weak semantics.
HMM + MA [20] – Indonesian	Morphology – Accuracy	The model remains insufficient for OOV words without affixes.
character-based LSTM [40] – Kazakh, English, Finish, and Russian	Character Level – Accuracy	The model is less effective for inflectional languages like English.
Norm + Lex [26] – German	Vocab Extension, Spelling Correction, Morphology – Accuracy	The model fails to normalize short function words or effectively handle nouns and named entities.

NPT [37] – Farsi	Word Level – Accuracy	Morphological features fit inflectional languages suboptimally
WE-CRF [52] – English	Morphology, Word Level – Accuracy	The model risks overfitting when updating representations of low-frequency OOV words.
FSA-based word representations [53] – English	Character Level	The model fails to disambiguate categories such as adjectives and adverbs.

vocabulary to recognize more words [25], [26]. Janicki [25] added unlabeled words from development data, which improved model performance. Keiper et al. [26] utilized an external lexical corpus separate from the training data, raising accuracy to 90%. However, this strategy demands a large and continuously updated vocabulary to accommodate new words. While effective when the vocabulary size aligns with general language use, its effectiveness diminishes in limited data settings.

2.2 Hand-crafted features

Hand-crafted features are designed by experts based on domain knowledge [36]. To address OOV, some studies employ *morphological* and *orthographic* features. Morphological features capture affix information (prefixes and suffixes) using morphological analysis tools. For instance, Muljono et al. [20] applied MorphInd to extract affixes and clitics in Indonesian, Anastasyev et al. [28] used ABBYY Compreno for Russian, and Partalidou et al. [29] employed tools from the Institute of Language and Speech Processing (ILSP) for Greek. Similarly, Passban et al. [37] utilized stemming to extract affixes and variations in Farsi. Orthographic features, on the other hand, capture surface-level written information such as word shape [29], punctuation, and word case [28].

Feature-based methods rely on linguistic information, such as morphological and orthographic cues, to assign labels. Although these cues seem simple, studies by Ayogu et al. [21] and Muljono et al. [20] showed that morphological features, such as affixes, can enhance HMM accuracy. In neural models, Passban et al. [37] demonstrated that combining prefix, suffix, and variation features raised MLP accuracy to 97.38% (+2.73%). Similarly, Anastasyev et al. [28] found that combining morphological and orthographic features improved BiLSTM results. Morphological features generally outperform context-based ones [37], particularly in agglutinative and morphologically rich languages with stable spelling and grammar, such as Indonesian, which also contains clitics [20].

Although useful, linguistic information does not always improve model performance, particularly in highly inflected languages such as Greek [29]. Since morphological features rely on linguistic expertise, adapting them to each language requires additional effort. Anastasyev et al. [28] also demonstrated that relying solely on linguistic information may decrease accuracy because it fails to capture word context. Nevertheless, there is still potential to investigate more fine-grained orthographic features that have not yet been widely applied in POS tagging.

2.3 Learned features

Learned features are obtained from datasets through training processes designed to accomplish a specific task [36]. In the context of NLP, textual data is transformed into numerical vectors known as embeddings. Various approaches exist for constructing word embeddings, such as word-level, subword-level, character-level, and multi-level representations.

Yu et al. [30] employed Word2Vec to represent words, which offers two architectures: Skip-gram and CBOW. The Skip-gram model learns word vectors by predicting surrounding context words [30], while CBOW predicts a target word by averaging the vectors of its context words [37]. Yu et al. [30] further introduced a Skip-gram variation to construct Task-Specific Word Representations (TSWR) tailored for several NLP tasks. To address the OOV problem, they applied a mapping strategy to derive suitable word representations from Skip-gram vectors.

In addition, Yu et al. [30] represent words at the subword level using FastText. FastText extends Word2Vec by constructing word vectors from the average of subword (character n-gram) representations [29].For instance, the vector of the word *apple* is computed from the sum of its n-grams, such as “<ap”, “app”, “appl”, “apple”, “apple>”, “ppl”, “pple”, “pple>”, “ple”, “ple>”, and “le>”, where angle brackets mark word boundaries [31]. Fasttext can represent OOV words better than word level embedding [38]. In text classification, subword level representation effectively handled OOV [39]. Beyond

FastText, another study trained a BiLSTM model to learn representations of OOV words [30].[30][30][37][30][30][29][31][38][39][30]Meanwhile , character-level representation is widely used for handling OOV in deep learning–based models. Won et al. [31] explored the use of CNNs to construct character embeddings, while Makazhanov & Yessen-bayev [40] and Kemos et al. [41] investigated LSTM and BiLSTM architectures. Some studies also introduce variations by adding position markers at the beginning and end of words. In addition, Makazhanov & Yessenbayev [40] converted characters into 8-bit ASCII codes.

Several studies combine multiple levels of representation within a single framework. Marulli et al. [33] and Bhattu et al. [23] integrated word embedding with character embedding, while other works compared the effectiveness of different embedding methods, such as word2vec, fasttext [33], and glove [23]. Beyond word and character features, Boonkwan & Supnithi [42] enriched their model by incorporating morphological context through the BiRNN method. Pota et al. [43] derived word embeddings from character embeddings trained on pretrained embeddings using BiLSTM. Moudjari et al. [24] developed dialect-specific embeddings for Modern Standard Arabic, which encompasses multiple dialects. Yu et al. [30] proposed an Out-of-Vocabulary Representation (OOVR) by combining a modified word embedding (TSWR) with subword representation.

The learned features approach addresses OOV by leveraging embeddings trained on large datasets. Word-level embeddings are widely used due to their simplicity and efficiency, with Passban et al. [37] showing that CBOW (93.11% accuracy) surpassed GloVe (90.09%). While word-level embeddings capture semantic relations effectively, they fail with unseen words, morphological changes, and spelling errors. Subword-level embeddings overcome these issues and need a smaller vocabulary by modeling subword units, often outperforming static word vectors [29] Nonetheless, they struggle to capture broader sentence context and perform poorly with very short words (fewer than three characters).

Character-level embeddings can naturally address the OOV problem by capturing intrinsic features of words at the character level. They are more robust to morphological variations and misspellings, particularly in short words. However, character-level embeddings require more complex architectures and result in larger model sizes, which lead to longer training times. While they can effectively capture fine-grained information within words, they struggle to represent broader sentence-level context. Bhattu et al. [23] demonstrated that combining multiple levels of embeddings can achieve better performance than relying on a single type. This approach leverages embeddings trained on large-scale data to provide vector representations for OOV words, with a primary emphasis on character-level embeddings.

Character-level embedding focuses solely on learning the arrangement of characters within a word, without taking into account the surrounding context in which the word appears. As a result, the model may still fail to handle words that do not follow previously learned form patterns, such as foreign words from different dialects or languages. In contrast, Liu et al. [32] employed ELMo to represent OOV words by incorporating both sentence context and internal word structure. This method has been shown to enhance the performance of POS Tagging in addressing OOV. Nevertheless, only a limited number of studies have investigated the impact of contextual embeddings on improving POS Tagging performance, particularly in handling OOV. In particular, no study has utilized contextual embedding to handle OOV words in Indonesian.

Therefore, this study proposes a contextual embedding approach to address OOV. To the best of our knowledge, this approach is the first applied to handle OOV words in Indonesian. Several types of embeddings are compared to determine the most effective for improving Indonesian POS Tagging performance in handling OOV. In addition, hyperparameter tuning is conducted to identify the optimal configuration for maximizing performance gains.

3 Dataset preparation

This study employs a dataset from Fu et al. [34], consisting of 21,024 sentences and 355,021 words. However, our observations indicate that the dataset contains labeling errors [44], as it was created using a semi-automatic approach without revalidation. To address this, we revalidated a subset of sentences to ensure consistent and correct labels. The validation involved 25 selected linguists with an inter-annotator agreement of 0.868 using the Fleiss Kappa method, which demonstrate excellent consistency among annotators. They worked in groups of 3-4 people each. They used a voting method, with the majority vote (>50%) determining the most appropriate label for sentences containing the same word. When they encountered difficulties or deadlocks, one expert served as a reference for the entire group. As a result, we obtained 30,960 words from 3,175 validated sentences. The label distribution of the dataset is presented in Table 2, showing a

Table 2: Dataset distribution per label

Fold	Label	Words	Percentage
Adjective	JJ	859	2.77%
	JJS	24	0.08%
Adverb	MD	568	1.83%
	RB	2,065	6.67%
Noun	NN	6,912	22.33%
	NNP	5,403	17.45%
	SP	12	0.04%
Pronoun	PRD	999	3.23%
	PRF	61	0.20%
	PRI	6	0.02%
	PRL	1	0.00%
	PRP	901	2.91%
	WH	120	0.39%
Verb	VB	3,863	12.48%
	VO	32	0.10%
Conjunction	CC	24	0.08%
	SC	179	0.58%
	CD	758	2.45%
Determiner	DT	163	0.53%
	OD	39	0.13%
Interjection	UH	179	0.58%
Particle	P	114	0.37%
Preposition	IN	2,141	6.92%
	PO	24	0.08%
Punctuation and symbol	SYM	195	0.63%
	Z	4,983	16.09%
Miscellaneous	FW	113	0.36%
	ID	211	0.68%
	X	11	0.04%

Table 3: OOV words distribution per fold

Fold	Train	Validation	Test	OOV
1	22,105	2,458	6,397	1,037
2	22,309	2,564	6,087	1,056
3	22,237	2,507	6,216	1,106
4	22,288	2,536	6,136	1,010
5	22,253	2,583	6,124	999

predominance of noun groups (NN and NNP), punctuation (Z), and verbs (VB).

For model evaluation, the dataset was divided into training and testing sets with a 4:1 ratio. The training set was further split into training and validation subsets with a 9:1 ratio. To ensure robustness, we employed a five-fold cross-validation approach, allowing the model to be trained and tested on different data partitions. The distribution of training, validation, and testing data is presented in Table 3. Notably, approximately 16% (1,037 words) of the testing data were identified as OOV words.

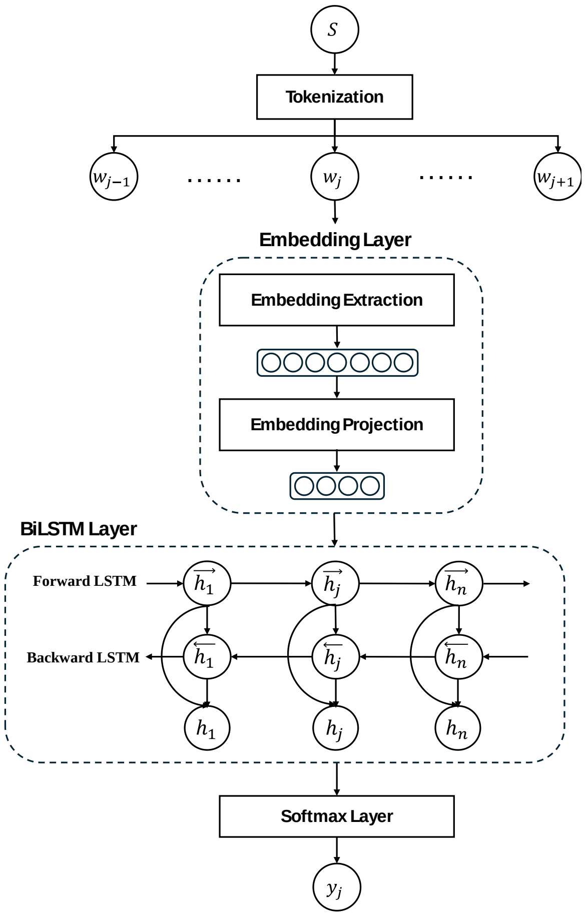


Figure 1: POS Tagging model architecture with Contextual embedding

4 Proposed model

This study introduces a vector representation based on contextual embedding to handle OOV in POS Tagging. The proposed approach is structured into four stages: tokenization, embedding layer, BiLSTM layer, and softmax layer, as illustrated in Figure 1. In the final stage, the model is trained with hyperparameter tuning and evaluated under multiple scenarios. Each stage is described in detail in the following subsections.

4.1 Tokenization

Tokenization is the process of segmenting a sentence (S) into words (w). In this study, tokenization is performed by splitting words based on spaces, while also separating punctuation marks unless they are inherently part of the word. Examples include reduplicated forms such as *barang-barang* (“goods”), legal references such as *39/M-Dag/Per/9/2009*, and dates such as *8/12/2005*. The original word form is preserved to avoid semantic distortion that may occur when a token is divided into excessively small units.

4.2 Embedding layer

The embedding layer converts words into vectors so they can be processed by the BiLSTM method. In the baseline model, the lookup embedding method generates a static vector with random values for each word. In this study, we propose an embedding layer consisting of two processes: embedding extraction and projection. Embedding extraction converts words into numeric vectors. In this study, three static embeddings (Word2Vec, GloVe, and FastText) and three contextual embeddings (BERT, ELMo, and Flair) were employed.

Static embedding assigns a unique vector representation to each word. These vectors are derived from training on a large corpus using specific algorithms. The methods and datasets employed in this study are described in the following subsections.

Word2vec[1] was trained using the Indonesian CoNLL17 corpus, which contains 2,899,107 words. The model is commonly trained using two alternative algorithms: Continuous Bag of Words (CBOW) and Skip-gram. CBOW predicts a target word based on its surrounding context, whereas Skip-gram predicts the surrounding context given a target word. In this study, the Skip-gram algorithm was employed to generate word vector representations, with each word represented in a 100-dimensional vector space.

GloVe[2] was trained on the Common Crawl corpus consisting of 840 billion tokens. Unlike Word2Vec, which learns context from neighboring words, GloVe captures context based on the probability of word co-occurrence within the corpus. This allows GloVe to represent both global and local statistical information, enabling it to capture broader semantic relationships than Word2Vec. The resulting vector representation of GloVe has 300 dimensions.

Fasttext[3] is an extension of Word2Vec that was trained on the Wikipedia corpus and Common Crawl, with the number of tokens ranging from 10 to 100 billion depending on the language. Similar to Word2Vec, FastText can be trained using either the skip-gram or CBOW approach. However, unlike Word2Vec, FastText represents words as a collection of subword units by using character n-grams (typically with a length of 5) and a context window size of 5. This enables FastText to

[1] https://vectors.nlpl.eu/repository/20/50.zip
[2] https://nlp.stanford.edu/data/glove.840B.300d.zip
[3] https://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.id.300.vec.gz

generate embeddings for rare and even out-of-vocabulary words. The resulting vector representation has a dimensionality of 300.

Contextual embeddings, on the other hand, assign a unique representation to each word depending on its context. The vector is dynamically generated based on the entire sentence in which the word appears. These embeddings are trained with more complex models and massive datasets, enabling them to produce context-dependent representations.

BERT[4] was trained on the large-scale Indo4B dataset containing 4 billion words from 12 corpora. Its input consists of subwords generated using the WordPiece method. Each subword is converted into an initial embedding, which is a combination of token embedding, position embedding, and token type embedding. These embeddings are then processed through a transformer architecture with multi-head self-attention and 12 layers of feedforward networks. BERT learns contextual representations through Masked Language Modeling (MLM) to predict missing words and Next Sentence Prediction (NSP) to capture sentence-level relationships. Since the output of BERT is at the subword level, the vectors of all subwords within a word are concatenated and averaged to form a single word representation. The final output is a vector representation with 768 dimensions.

ELMo[5] was trained on the Indonesian CoNLL17 corpus, which contains 2,899,107 words. Its input begins with character-level embeddings derived from character convolutions using the CNN method. The model is then trained with two LSTM layers, each processing the sequence in opposite directions—forward and backward—enabling ELMo to capture context-dependent nuances of word meaning. The final word representation is constructed through a weighted sum of three components: the character embedding, the forward LSTM, and the backward LSTM. The weights are optimized using the softmax activation function to balance the contribution of each component. The resulting ELMo embeddings have a dimensionality of 1024.

Flair[6] was trained on datasets from Wikipedia and OPUS containing 174,467,241 words. Flair employs a bidirectional LSTM architecture that processes sentences both forward and backward at the character level. The representation of each word is derived from character embeddings: in the forward layer, the final character of the word is used, while in the backward layer, the initial character is considered. This design enables Flair to capture contextual information from both preceding and succeeding words in a sentence. Each layer produces a 2048-dimensional vector, resulting in a combined word representation of 4096 dimensions.

The dimensionality of word embeddings varies significantly, ranging from 100 to 4096. Such differences create challenges in ensuring fair comparisons, and embeddings with very high dimensions can increase model complexity and lead to overfitting. To address this, a projection layer is applied to standardize and reduce the embedding dimensions. By applying this approach, the risk of overfitting from disproportionately large vectors is minimized. The final projection produced embeddings with 100 dimensions.

4.3 BiLSTM layer

This study employs the BiLSTM method as a POS Tagging model, following the approach proposed by Kurniawan & Aji [45]. BiLSTM operates with LSTM cells trained in two directions: forward ($\overrightarrow{h_t}$) and backward ($\overleftarrow{h_t}$), as shown in (1) and (2). Figure 1 illustrates the BiLSTM architecture, where the forward and backward layers are concatenated as in (3). a softmax layer is used for classification to interpret the BiLSTM output (h_t) as word classes (T_t). We employ a two-layer BiLSTM architecture to more effectively capture sentence context. Two-layer BiLSTM has been shown to produce better performance than 1 or 3 layers [46]. Each LSTM cell contains 100 neurons, producing a 200-dimensional vector representation.

$$\overrightarrow{h_t} = LSTM_{forward}(x_t, \overrightarrow{h_{t-1}}) \quad (1)$$

$$\overleftarrow{h_t} = LSTM_{backward}(x_t, \overleftarrow{h_{t+1}}) \quad (2)$$

$$h_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t} \quad (3)$$

4.4 Softmax layer

We employ a classification layer to decode the BiLSTM outputs into word classes for POS tagging. A softmax layer with greedy decoding is used to select the word class by computing the highest probability $P(y)$ from the word class vector (V^y). The input to this layer is the BiLSTM output, denoted as h_t. The predicted word class y is expressed as shown in (4):

$$P(y = j|x) = \frac{e^{h_j}}{\sum_{k=0}^{K} e^{h_k}} \quad (4)$$

During training, the model maximizes the likelihood of the correct tag sequences. The final output sequence of tags is determined based on the highest score, computed as shown in (5):

$$y^* = \underset{y' \in y}{\operatorname{argmax}}\ S(x, y') \quad (5)$$

4.5 Evaluation

This study evaluates the models using two metrics to assess the reliability of each embedding in handling OOV words.

[4] https://huggingface.co/indobenchmark/indobert-base-p1
[5] https://vectors.nlpl.eu/repository/20/158.zip
[6] https://flairnlp.github.io/docs/tutorial-embeddings/flair-embeddings

The first metric is *accuracy*, defined as the ratio of correctly predicted word classes to the total number of words (also referred to as Detection Accuracy). *Accuracy* measures the overall performance of the model without considering the distribution of data or label imbalances. It is calculated based on the number of correctly predicted positive labels (True Positives, TP) and negative labels (True Negatives, TN), divided by the total number of instances, as shown in (6):

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (6)$$

Table 4: Embedding performance comparison results

Category	Embedding	General		IV		OOV	
		Acc	F1	Acc	F1	Acc	F1
Baseline Model		85.81%	67.11%	90.43%	73.06%	62.97%	26.48%
Static Embedding	Word2vec	70.68%	57.59%	74.73%	62.52%	50.60%	24.43%
	GloVe	76.21%	59.14%	83.11%	64.28%	41.97%	21.67%
	Fasttext	92.03%	67.56%	93.98%	72.03%	82.40%	33.91%
Contextual Embedding	ELMo	91.61%	64.78%	92.54%	68.92%	87.06%	43.42%
	Flair	**95.63%**	**81.50%**	**97.14%**	**87.07%**	**88.12%**	**53.46%**
	BERT	92.73%	69.74%	94.35%	74.77%	84.73%	43.10%

The second metric is the *macro F1-score*, chosen because it evaluates the model's performance across all word classes, regardless of their distribution. This metric is particularly useful for assessing performance on minority labels and mitigating bias toward majority labels. The *macro F1-score* is calculated as the average F1 score across all labels (n), as shown in (7). The F1 score for each label is computed using the number of correctly predicted positive instances (True Positives, TP) divided by TP plus half of all misclassified instances (False Positives and False Negatives), as shown in (8).

$$macro\ F1 - score = \frac{\sum_{i=1}^{n} F1_i}{n} \quad (7)$$

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP+FN)} \quad (8)$$

This study evaluated the models under three scenarios: a general scenario, which tests all words in the dataset, and two specific scenarios, focusing on particular subsets of words. The specific scenarios are divided into In-Vocabulary (IV) and OOV tests. IV evaluates words that appear in the training data (i.e., in the dictionary), while OOV assesses words that are not present in the training data.

4.6 Hyperparameter tuning

This study performed hyperparameter tuning to identify the optimal configuration for the POS Tagging model. The hyperparameters considered included learning rate, dropout, batch size, BiLSTM hidden size, and embedding size. Tuning was conducted using Optuna with 30 trials. The objective was to maximize accuracy and macro F1-score while minimizing the loss.

The learning rate is a hyperparameter that controls the speed at which the model learns. A high learning rate may cause instability and prevent the model from converging, whereas a low learning rate can slow down the learning process. In this study, learning rates ranging from 0.0001 to 0.1 were explored.

Dropout is a regularization hyperparameter that randomly deactivates nodes in a model, preventing over-reliance on dominant nodes. This technique helps make the model more robust to variations in the data. The dropout value indicates the proportion of nodes deactivated in a layer, with higher values resulting in more nodes being deactivated. In this study, dropout values ranging from 0.1 to 0.5 were evaluated.

Batch size is a hyperparameter that determines the number of data samples processed in one iteration. It is set separately for both training and testing data. Smaller batch sizes typically lead to faster updates, while larger batch sizes require more memory for temporary storage. In this study, batch sizes following a power-of-two pattern (8, 16, 32, 64, 128, 256) were evaluated.

BiLSTM hidden size is a hyperparameter that determines the dimensionality of the BiLSTM layer for capturing contextual patterns. Larger hidden sizes enable the BiLSTM to store more contextual information, while smaller sizes limit its capacity. In this study, hidden sizes ranging from 100 to 300 were evaluated.

Embedding size is a hyperparameter that determines the dimensionality of the vectors generated by the embedding layer, similar to the BiLSTM hidden size. It controls the number of values in the projected embedding representation. Larger embedding sizes allow the model to capture more contextual information, while smaller sizes limit this capacity. In this study, embedding sizes ranging from 20 to 150 were evaluated.

Several hyperparameters were tested simultaneously to identify the optimal configuration for the POS Tagging model. The contribution of each hyperparameter to variations in accuracy, macro F1-score, and loss was analyzed using the fANOVA approach. fANOVA computes the Fraction of Variance Explained (FoVE), quantifying the impact of each hyperparameter on changes in the target metric. Finally, the performance of the proposed model was compared with that of models from previous studies.

5 Experiment and results

The experiments were conducted on a shared computer equipped with an Intel i7-12700 (2.10 GHz) processor, 96 GB of RAM, and an NVIDIA GeForce RTX

3080 Ti GPU. This study uses the Adam optimizer to update weights iteratively based on training data. The model is trained for 50 epochs using an early stopping mechanism. Training stops if the dev loss increases for five consecutive epochs. The number of epochs for each fold varies, ranging from 28 to 44. However, the baseline model across all folds has the maximum number of epochs (50). This also affects the training duration of each model. The baseline and static embedding models require approximately three to four minutes of training time per fold. Meanwhile, the contextual embedding model requires between 30 and 70 minutes per fold, depending on the number of epochs.

The results from each fold were averaged to obtain the overall accuracy and macro F1-score, as presented in Table 4. Static embeddings generally underperform compared to the baseline model. Although FastText showed higher overall performance (+6.22%) relative to the baseline, its performance decreased slightly in the IV scenario (-1.03%). In contrast, contextual embeddings tended to outperform the baseline model. Among them, Flair achieved the best performance across both general and specific evaluations, including IV and OOV cases. However, in some scenarios, ELMo performed slightly worse than the baseline model.

5.1 Static embedding

Model performance tended to be lower when using Word2Vec (-15%) and GloVe (-10%) embeddings. Both are static embeddings trained on large amounts of word-level data. The reduced performance is primarily due to Word2Vec's inability to generate vectors for words outside its vocabulary. This is evidenced by the comparison of prediction results shown in Table 5, which illustrates the correlation between prediction accuracy and the presence of words in the embedding vocabulary.

Based on the Table 5, Word2Vec and GloVe were able to recognize only 24% (1,582 words) of the corpus used in this study. Among the unrecognized words, Word2Vec correctly predicted only 16% (256 words), while 83% (1,326 words) were predicted incorrectly. In contrast, GloVe correctly predicted 75% (1,200 words) and mispredicted 25% (382 words). Word2Vec's poor performance is attributed to its inability to provide vector representations for unknown words; the model uses random vectors for these words, resulting in inaccurate label predictions. For example, in sentence S-1, the words "*Cipinang*" and "*Melayu*" lacked vector representations in Word2Vec, causing the model to incorrectly assign them the label JJ (adjective).

Meanwhile, most of the GloVe model's prediction errors occurred for words that were present in the GloVe dictionary. The model was able to correctly predict unrecognized words as NN (nouns). Conversely, some words that existed in GloVe were predicted incorrectly. This is because GloVe was trained on English data, resulting in vector representations that did not align well with the Indonesian vocabulary. The vector representations generated by GloVe are illustrated in Figure 2. The vectors were visualized using t-SNE with a perplexity of 30, reducing the dimensionality to two dimensions for easier interpretation. For example, the words "*wanita*" (woman), "*warga*" (inhabitant), and "*negeri*" (country) were labeled NN (nouns), while the words "foreign," "direct," and "card" were labeled FW (foreign words). Meanwhile, the word "*Pondasi*" (foundation), which was labeled NN, was predicted as FW in sentence S-2. The GloVe representation for "*Pondasi*" was closer to words labeled FW than to words labeled NN, leading the model to assign it the FW label.

Table 5: Comparison of Word2vec and Glove prediction results

Prediction Status	**Word2vec**		**GloVe**	
	Avail.	**Unavbl.**	**Acc**	**Avail.**
True	4,171	256	3,658	1,200
False	643	1,326	1,156	382

Table 6: Calculation of the distance between the words *diri* (self) and sendiri (self)

Word	**Similarity**		
	Word2vec	**Glove**	**Fasttext**
Diri	0.95278	0.90645	0.68872
Sendiri			

Table 7: Statistical analysis of the words *diri* (self) and *sendiri* (self)

Fold	***diri***		***sendiri***	
	True	**False**	**Acc**	**True**
1	6	0	3	4
2	2	0	5	0
3	6	1	5	0
4	2	0	5	3
5	5	1	4	0
Total	21	2	22	7
Accuracy	**91.30%**		**75.86%**	

(S-1) *Kelurahan* (NNP) ***Cipinang*** (NNP) ***Melayu*** (NNP) **Cipinang** (NNP) **Melayu** (NNP) Sub-district (NNP)

(S-2) ***Pondasi*** (NN) *itu* (PRD) *diletakkan* (PRD) *di* (IN) *dekat* (JJ) *taman* (NNP)
The foundation (NN) was (PRD) placed (PRD) in (IN) near (JJ) the park (NNP)

Meanwhile, the FastText model generally outperformed the default model, achieving an improvement of approximately 7%. FastText is a static embedding trained on large amounts of data at the subword level. Unlike Word2Vec and GloVe, FastText generates vector representations for combinations of subwords in Indonesian, enabling it to recognize word forms effectively. This allows FastText to provide vector representations even for words that are not explicitly present in its vocabulary. For example, the word "*dikembalikkannya*" (returned) in sentence S-3 is a verb with multiple affixes "di-*",* "*-kan*," and "*-nya*." FastText successfully identified it as a VB (verb) because it can recognize commonly used affix patterns in verbs.

However, strong word form recognition can also lead to prediction errors in PRF labels (reflexive pronouns). For instance, the words "*diri*" (self) and "*sendiri*" (self) under PRF labels have different vector representations, as shown in Table 6. The similarity value between these two words is lower compared to other embeddings, which causes the FastText model to fail in correctly predicting the label for "*sendiri*" (self). These prediction errors did not occur by chance; they were repeated. We analysed them statistically, as shown in Table 7. The prediction error for the word 'sendiri' occurred in folds 1 and 4. This resulted in the model achieving 75.86% accuracy in these

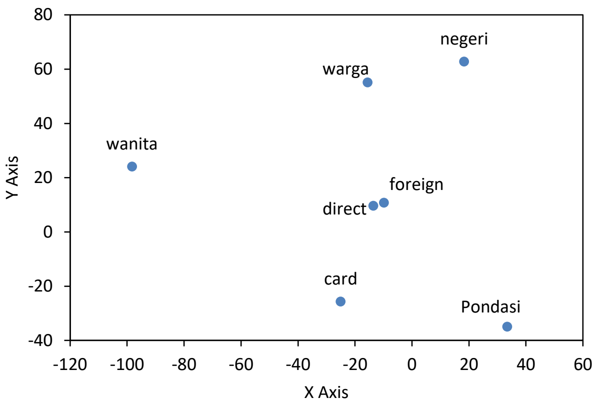


Figure 2: Illustration of the vector representation generated by GloVe

challenging cases. However, subword-based word representation struggles to represent words with the same meaning but different forms, such as 'diri' and 'sendiri'.

(S-3) *PDIP* (NN) *mendorong* (VB) ***dikembalikannya*** (VB) *kewenangan* (NN)
PDIP (NN) encourages (VB) **to return** (VB) the authority (NN)

Although the model's overall performance improved, FastText performed worse than the baseline model in handling IV cases. This is because FastText provides vector representations without considering the context of words within a sentence. Contextual information is crucial for correctly predicting labels for ambiguous words that have multiple possible labels. For example, in sentence S-4, the word "bawah" (under) functions as a noun because it is part of the phrase "parkir bawah tanah" (underground parking). However, the model predicted it as IN (preposition) because FastText identified "bawah" primarily as a preposition.

(S-4) *Parkir* (NN) ***bawah*** (NN) *tanah* (NN) *akan* (MD) *dibangun* (VB)
under (NN) ground (NN) parking (NN) will be (MD) built (VB)

5.2 Contextual embedding

ELMo performed well in handling OOV cases, achieving the second-best performance after Flair. ELMo captures context at the character level using two LSTM layers, which enables it to provide effective vector representations for previously unseen words, achieving an accuracy of 87.06%. For example, in sentence S-5, the word "Nudirman" was correctly labeled as NNP (proper name) even though the model has never encountered it as a person's name. However, ELMo performed less effectively on IV cases, with a macro F1 score of 68.92%, which is lower than the baseline model. ELMo underperforms on several labels, including NNP, RB and MB. The performance of several other minority labels, including PRF, SC, DT, OD, UH, P and ID, also declines. Consequently, ELMo's macro F1-score is lower than that of the baseline model. ELMo has a tendency to over-contextualise known words (IV). For example, the word 'Guangzhou' in the sentence S-6. In the training data, the word 'Guangzhou' is labelled NNP. However, during testing, ELMo mislabels it as NN because it captures contextual noise from surrounding NN-labelled words. This limitation arises because ELMo's two LSTM layers do not interact with each other, restricting the directional context processing. Additionally, the LSTM method tends to be biased towards frequently occurring labels in sentence.

(S-5) ***Nudirman*** (NNP) *pun* (P) *kembali* (VB) *menanggapi* (VB)
Nudirman (NNP) also (P) responded (VB)

(S-6) Kereta (NN) api (NN) **Guangzhou** (NNP) jalannya (NN) ke (IN) arah (NN) mana (WH)
Guangzhou Railway (NN) train (NN) route (NN) to (IN) which (NN) direction (WH)

The Flair model demonstrated a substantial performance improvement (+25%), particularly in OOV cases. Its performance also improved in ambiguous cases (+2%). Overall, the model achieved a significant gain (+10%) by leveraging Flair's contextual representation. Flair can recognize both OOV and IV words effectively because it captures word forms using a character-level BiLSTM and word context using a sequential BiLSTM. This dual-level modeling enables Flair to accurately predict labels for OOV and ambiguous words. For example, in sentence S-7, the word "terkalahkan" (defeated) was correctly labeled as VB because it contains typical verb affixes such as "*ter*-" and "-kan." Flair was also capable of distinguishing different meanings of the word "*baru*." In sentence S-8, "*baru*" (just) carries the meaning of "not long ago" and was labeled RB (adverb), whereas in sentence S-9, "*baru*" (new) conveys the meaning of "beginning" and was labeled JJ (adjective).

(S-7) Kami (PRP) tak (RB) **terkalahkan** (VB)
We (PRP) are undefeated (VB)

(S-8) *Kasus* (NN) *ini* (PRD) ***baru*** (RB) *pertama* (OD) *kali* (NN)
This (PRD) case (NN) is the first (OD) time (NN)

(S-9) *Memasuki* (VB) *awal* (NN) *tahun* (NN) ***baru*** (JJ)
Entering (VB) the beginning (NN) of the **new** (JJ) year (NN)

Although Flair performs well in most cases, it still struggles to predict certain OOV words with specific characteristics. For instance, the capitalized word "*Kuba*" in sentence S-10 should be labeled as NNP because it is a country name. In Indonesian, capitalized words generally indicate proper nouns. However, Flair failed to recognize this pattern and labeled "*Kuba*" as NN (common noun). Additionally, Flair was unable to correctly label reduplication words, such as "*Jarang-jarang*" (rarely) in sentence S-11, which was labeled NN instead of the correct JJ (adjective). While Flair effectively recognizes general word patterns, it has not yet mastered reduplication patterns. Since such words are relatively rare, specific guidance or features may be required to help Flair correctly identify them.

(S-10) *Hubungan* (NN) *AS* (NNP) – (Z) ***Kuba*** (NNP) *memanas* (VB)
US (NNP) – (Z) **Cuba** (NNP) relations (NN) heat up (VB)

(S-11) ***Jarang-jarang*** (JJ) *Gus* (NNP) *Dur* (NNP) *menolak* (VB) *berkomentar* (VB)
Rarely (JJ) Gus (NNP) Dur (NNP) refuses (VB) to comment (VB)

The BERT model achieved the second-best performance overall, after Flair. However, in OOV cases, BERT is outperformed by ELMo. Unlike other contextual embeddings, BERT uses a transformer-based Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) approach to capture the context of words in a sentence. BERT learns word patterns based on subword arrangements, allowing it to recognize the morphological forms of OOV words. For example, in sentence S-12, BERT correctly identified "*perumahan*" (housing) as NNP because it contains noun-specific affixes such as "*pe-*" and "*-an*". Furthermore, BERT also assigned the NNP label correctly to words like "*Pandau*" and "*Permai*" by leveraging the contextual relationship with surrounding words.

(S-12) ***Perumahan*** (NNP) ***Pandau*** (NNP) ***Permai*** (NNP)
Pandau (NNP) **Permai** (NNP) **Housing** (NNP)

However, BERT's performance did not show a significant improvement over the default model in the IV case. This is because BERT failed to correctly label certain word classes, such as OD, PRF, and CC. For instance, in sentence S-13, the word "*pertama*" (first) was labeled as CD (numerical), whereas the correct label is OD (ordinal). Similarly, in sentences S-14 and S-15, the word "*sendiri*" (alone) was incorrectly labeled as RB (adverb), and "*Selain*" (apart) was labeled as SC (coordinating conjunction), both of which do not match their true labels.

(S-13) *Perhitungan* (NN) *suara* (NN) *pilpres* (NN) *putaran* (NN) ***pertama*** (OD)
First (OD) round (NN) presidential (NN) election (NN) vote (NN) count (NN)

(S-14) *Peristiwa* (NN) *lori* (NN) *jalan* (VB) ***sendiri*** (PRF)
Incident of (NN) a lorry (NN) driving (NN) **on its own** (PRF)

(S-15) **Selain** (CC) buku (NN) , (Z) DPRD (NNP) juga (RB) menganggarkan (VB) pengadaan (NN) laboratorium (NN)
Apart from (CC) books (NN), (Z) DPRD (NNP) also (RB) budgeted (VB) for the procurement of

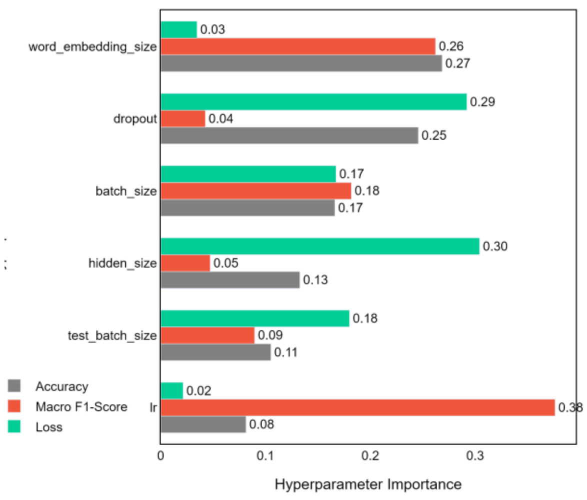


Figure 4: Hyperparameter importance graph against evaluation metrics

(NN) laboratories (NN)

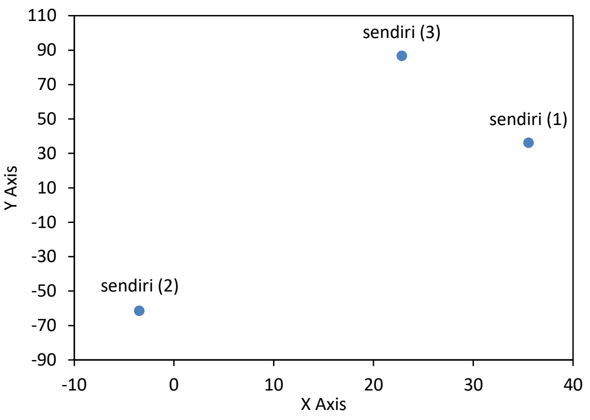


Figure 3: Illustration of a vector representation of the word "*sendiri*" (self) generated by BERT

This prediction error occurs because BERT represents words based on their context and position within the sentence. As a result, the vector representations of the same word vary from sentence to sentence. For example, Figure 3 shows the vector representations of the word "*sendiri*" (self) from three different sentences. Although the correct label is PRF, the vectors differ across sentences. In the first and third sentences, "*sendiri*" was correctly labeled, whereas in the second sentence, it was mislabeled because it appeared far from other occurrences of the word.

The experiments demonstrate that contextual embeddings have promising potential to enhance POS tagging performance in both IV and OOV cases. As their rankings indicate, contextual embeddings (Flair, BERT and ELMo) perform better than static embeddings. In general, both Flair and BERT perform well in terms of accuracy and macro F1-score metrics. To demonstrate the difference between the two models, we performed a t-test. The results of this test show that the t-statistic value for the difference between the BERT and Flair models is -24.59, with a p-value of 0.001. This proves that the BERT model differs from Flair by an average of 24.59, with Flair obtaining the highest value. A p-value of less than 0.05 indicates a statistically significant difference. Nevertheless, Flair still has limitations when it comes to handling certain out-of-vocabulary (OOV) words, particularly when it comes to distinguishing between noun (NN) and noun plural (NNP) labels. It is also not yet capable of correctly processing reduplicated word forms.

5.3 Hyperparameter tuning

To achieve maximum performance, this study optimized six hyperparameters: word embedding size, dropout, training batch size, BiLSTM hidden size, test batch size, and learning rate (lr) to determine the best configuration for the Flair-based POS Tagging model. Hyperparameter tuning was conducted using the Optuna library with 30 trials. The optimal configuration yielded a test macro F1 score of 84.66% and an accuracy of 95.28%, surpassing the model's performance prior to tuning. The best-performing configuration consisted of an embedding size of 67, BiLSTM hidden size of 167, dropout of 0.26, training batch size of 8, testing batch size of 128, and a learning rate of 0.087235.

In addition, we analyzed the importance of each hyperparameter with respect to the target metrics, as shown in Figure 4. The analysis indicates that embedding size and dropout have the greatest influence on accuracy, with importance scores of 0.27 and 0.25, respectively. Meanwhile, hidden size and dropout have the strongest impact on loss, with importance scores of 0.30 and 0.29. Notably, higher dropout values correspond to lower loss.

Dropout regulates how many neurons are deactivated in each layer to prevent the model from over-relying on a single neuron in the embedding and BiLSTM layers. A low dropout value results in weak regularization, allowing the model to memorize specific patterns, which can lead to overfitting and lower accuracy. Conversely, a high dropout value enforces stronger regularization, allowing the model to learn more generalizable features. This makes the model more robust to new data, as reflected in the increased accuracy with a higher dropout value.

The hyperparameter with the greatest influence on the macro F1 score is the learning rate, which has an importance score of 0.38. Adjusting the learning rate significantly affects the performance of the BiLSTM model. However, no clear pattern was observed between changes in the macro F1 score and the learning rate. We hypothesize that the Flair model generates contextual embedding values with inherent randomness, so variations in the learning rate strongly impact other metrics. The fANOVA analysis confirms this significance, as changes induced by learning rate variations are highly influential.

6 Discussions

Our experiment used an embedding architecture with embedding projection. This approach prevented the model from becoming too complex due to receiving large vector inputs from pretrained embedding. To prove the effect of embedding projection, we conducted an ablation study of models with and without embedding projection, as shown in Table 8.

Table 8: Embedding performance with (w) and without (wo) projection layer.

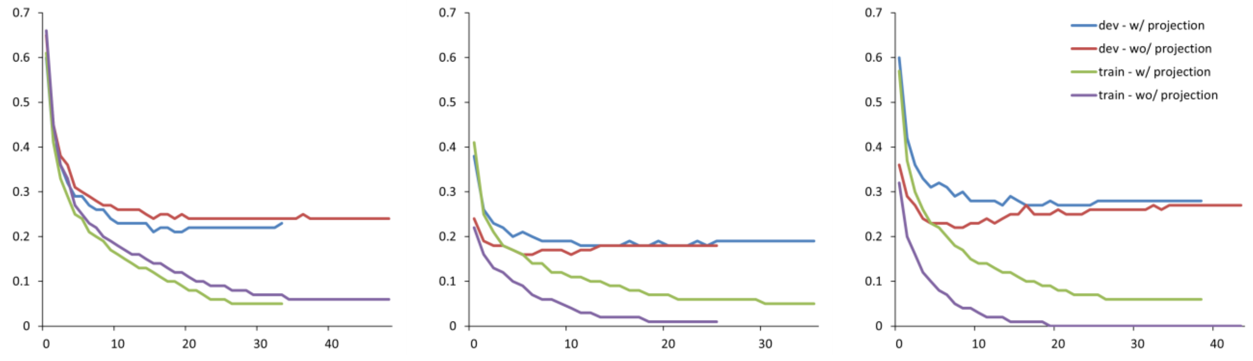
Model	**w/ Projection**		**wo/ Projection**	
	Acc	**F1**	**Acc**	**F1**
Word2vec	70.68%	**57.59%**	**72.30%**	55.06%
GloVe	76.21%	59.14%	**81.78%**	**65.41%**
Fasttext	**92.03%**	**67.56%**	86.95%	61.16%
BERT [54]	92.73%	69.74%	**93.95%**	**77.93%**
ELMo	91.61%	64.78%	**93.88%**	**74.47%**
Flair	95.63%	81.50%	**95.77%**	**82.79%**

Based on Table 8, models that do not use embedding projection perform better than those that do. This is because the projection process removes some important information. Contextual embeddings, such as those produced by BERT, ELMo and Flair, contain syntactic and semantic information and have high dimensions of 768, 1024 and 4096 respectively. However, FastText produces the opposite result. FastText performs better when using embedding projection than when not using it. This is because FastText builds representations using subword n-grams, meaning its vectors sometimes contain redundant information. Embedding projection helps FastText reduce this redundant information, making the vector representation more compact and stable.

However, models without an embedding projection can make the model more complex, as the vector dimensions of pre-trained embeddings, especially contextual embeddings, are quite high. More complex models are more likely to overfit, which means that they appear to perform well but fail to handle new data. We also evaluated the model using a graph comparing train and dev loss to determine the impact of embeddings on model training stability as shown in Figure 5. FastText, with a dimension of 300, tends to be more stable with and without embedding projection. Meanwhile, ELMo and BERT show significant differences in loss. The training loss for the model without embedding projection reached its lowest point. However, the dev loss did not decrease as much. In fact, after the 10th epoch, the dev loss tended to increase. This indicates overfitting, whereby the model memorises the training data too much and fails to generalise. Consequently, the model is unable to predict the dev data correctly.

The results of the experiment revealed that the Flair embedding model outperformed others. This is because the Flair model was trained using 174 million words, which is significantly more than the 4 million and 2 million words used to train the BERT and ELMo models, respectively. Additionally, the Flair architecture uses BiLSTM with character input, enabling Flair to recognise word and sentence patterns more effectively. Character embedding allows Flair to learn word structures based on letter sequences. Meanwhile, the BiLSTM method enables Flair to capture the context of words in sentences from two directions (forward and backward).

We compared our proposed model with previous POS Tagging models, including HMM, MEMM, BiLSTM, BiLSTM+CRF, and the most recent BiLSTM+CRF models incorporating morphological and character features. The comparison results are presented in Table 9. Based on the results in Table 9, our proposed BiLSTM model with contextual embedding (BiLSTM + context) outperformed previous POS Tagging models, achieving an accuracy of 95.63%. This performance surpasses that of the model reported by Kurniawan & Aji [45], which had an accuracy of 91.49%. Our proposed model improves the accuracy of previous research by Kurniawan [45] by 4.14% in general evaluations and by 12.38% in out-of-vocabulary (OOV) evaluations. Our proposed model also demonstrates superior performance compared to traditional machine learning models such as HMM (83.51%), MEMM (85.66%), and CRF (81.87%).



(a) Fasttext (b) ELMo (c) BERT

Figure 5: Grafik perbandingan train loss dan dev loss pada model with (w/) dan without (wo/) dengan projection embedding.

Meanwhile, models with embedding projection tend to have a smaller gap between training and development loss. This indicates that these models tend to be stable, as the information obtained from pre-trained embeddings is not processed immediately within the model. However, this information is adjusted and dimension-reduced so that the vector dimension is reduced, thereby reducing the model's complexity. Although models with embedding projection perform worse than models without, the former are better at handling data outside the training data. When handling OOV words, it is important to consider not only performance aspects such as accuracy and F1 matrices, but also the model's ability to generalise when handling new data.

Table 9: Embedding performance comparison results

Model	General		IV		OOV	
	Acc	F1	Acc	F1	Acc	F1
HMM	83.51%	61.23%	94.91%	75.84%	27.13%	7.96%
MEMM	85.66%	49.69%	88.08%	53.56%	73.64%	20.04%
CRF	81.87%	55.16%	85.85%	59.49%	62.19%	23.53%
CRF [54]	90.60%	66.96%	92.80%	74.74%	79.70%	26.74%
BiLSTM	85.81%	67.11%	90.43%	73.06%	62.97%	26.48%
BiLSTM + CRF	85.27%	66.61%	90.17%	72.32%	60.95%	26.53%
BiLSTM + CRF [45]	91.49%	75.49%	94.67%	83.41%	75.74%	35.88%
BiLSTM + Context (prop)	**95.63%**	81.50%	**97.14%**	87.07%	**88.12%**	**53.46%**
BiLSTM + Context fine-tuned (prop)	95.41%	**82.42%**	97.05%	**89.43%**	87.30%	52.63%

Additionally, it significantly improves the handling of OOV words, achieving a +5% performance gain.

The fine-tuned model also demonstrated a higher macro F1-score compared to the unfine-tuned model, indicating its improved ability to predict minority labels. For instance, the word "*rasanya*" (feel like), which carries the SP label in sentence S-16, appears only once in the dataset, and the SP label is rarely used. Nevertheless, the model successfully captured the pattern of SP followed by the subword "-*nya*" and labeled it correctly. Similarly, the model accurately labeled other minority labels, such as the word "*disebutnya*" (said) in sentence S-17 with the VO label. This word contains a clitic affix attached to the verb, and the model is able to recognize this pattern and assign the correct label.

(S-16) ***Rasanya*** (SP) *aku* (PRP) *harus* (MD) *berhenti* (VB) *sekolah* (NN)
I feel (SP) that I (PRP) should (MD) stop (VB) school (NN)

(S-17) Pelaku (NN) penyerangan (NN) **disebutnya** (VO) adalah (MD) pemuda (NN)
The perpetrator (NN) of the attack (NN) was said (VO) to be (MD) a young man (NN)

The hyperparameter tuning in this study was conducted on a single fold with one run per configuration. Consequently, the reported results should be interpreted as indicative trends rather than statistically rigorous comparisons. Future research should perform repeated runs or multi-fold validation to enable confidence interval estimation and statistical testing.

However, the fine-tuned model still faced the same limitations as the previous model. It struggled to predict certain OOV words with specific characteristics, such as capitalized words and reduplications. In Indonesian, capital letters are often used at the beginning of sentences, in direct quotations, and for personal names, titles, nicknames, greetings, religions, nationalities, ethnicities, languages, months, days, events and places. Since capital letters are frequently used in object and personal names, we analysed the confusion matrix of the two labels (NN and NNP), as shown in Table 10.

While most words labelled NN (95.17%) can be labelled correctly, there are still 44 words labelled NN that are incorrectly labelled as NNP, as well as 24 words that are labelled with other labels. Similarly, most words labelled NNP (95.94%) can be labelled correctly, but 31 words labelled NN are incorrectly labelled as NNP, and 16 words are labelled with other labels. For instance, the word *'Indah'* in sentence S-18, which is labelled JJ. This word is usually used to describe nouns. However, in that sentence, it refers to a person's name. The model failed to label it correctly because the word *'indah'* has multiple meanings, person's name (proper noun) and beautiful (adjective). In this case, the Flair model tends to produce vector representations similar to those of adjectives.

In addition to representation issues, there are problems related to annotation errors in the corpus, which result in suboptimal evaluation results. For instance, the word 'Bang' (Bro) in sentence S-19 is labelled as an NNP. This is correct because it is a nickname or greeting. However, the label for 'Bang' in the corpus is 'NN', so more careful label validation is needed to produce a high-quality corpus. Furthermore, discussions with language experts are required to establish guidelines for the application of NN and NNP labels to prevent similar labelling errors from occurring again.

(S-18) ***Indah*** (NNP) *menunjukkan* (VB) *ruangan* (NN)
Indah (NNP) shows (VB) the room (NN)

(S-19) ***Bang*** (NN) *taksinya* (NN) *pakai* (VB) *argo* (NN) *tidak* (RB)
Bro (NN) his taxi (NN) uses (VB) the meter (NN) doesn't (RB)

We also analysed prediction errors in reduplicated words using the confusion matrix shown in Table 11. Reduplicated words have a unique form compared to other words. They are written by repeating the base word and adding a hyphen between the two parts. Based on Table 11, reduplicated words with the labels 'NN' and 'NNP' can be distinguished with 100% accuracy. This is because reduplicated words with these labels have simple patterns. The words are repeated without undergoing any changes. For example, the word '*anak-anak*' (children) in sentence S-20 is formed from the word '*anak*' (child). Although it has undergone a change in form, the words ‘anak’ and ‘anak-anak’ still have the same label (NN).

However, reduplicated words with the labels VB, JJ, RB and IN cannot yet be predicted correctly. This is because the word is rarely used and changes when repeated. For example, the word '*bolak-balik*' (back and forth) in sentence S-21 is formed from the word '*balik*'

Table 10: Confusion matrix of capitalized words

Actual \ Predict	NN	NNP	Other
NN	**1341**	44	24
NNP	31	**1113**	16

Table 11: Confusion matrix of reduplication words

Act \ Pred	NN	NNP	VB	JJ	RB	IN
NN	**19**	0	0	0	0	0
NNP	0	**2**	0	0	0	0
VB	1	0	**3**	1	2	0
JJ	2	0	0	**4**	1	0
RB	0	0	0	1	**6**	0
IN	1	0	0	0	0	**0**

(back). The vowels change when forming a reduplicated word. Although both words are verbs, the model predicts them as adjectives. Another example is the word '*bersama-sama*' (together) in sentence S-22, which is formed from the word '*sama*' (same). The prefix '*ber-*' is added to the base word, resulting in a longer reduplicated word. Additionally, the label changes from an adjective (JJ) to a verb (VB). The Flair model is unable to handle this case because reduplicated words are rare. Furthermore, reduplicated words can undergo vowel changes and affixation, which can alter the label of the base word.

(S-20) ***Anak-anak*** (NN) *sudah* (RB) *siap* (VB)
Children (NN) are (RB) ready (VB)

(S-21) *Tamsil* (NNP) *harus* (MD) ***bolak-balik*** (VB) *memenuhi* (VB) *panggilan* (NN)
Tamsil (NNP) must (MD) go back and forth (VB) to answer (VB) the call (NN).

(S-22) *Warga* (NN) *melakukan* (VB) *penyisiran* (NN) *Pantai* (NN) *secara* (IN) ***bersama-sama*** (VB)
Residents (NN) conducted (VB) a cleanup (NN) of the beach (NN) with (IN) **together** (VB).

This highlights the need for additional information, such as orthographic cues or word shape, to help Flair correctly recognize these words. In this study, our focus was on maximizing Flair embedding performance through hyperparameter tuning. Meanwhile, other contextual embeddings, such as ELMo and BERT, have the same potential as Flair for overcoming OOV issues. However, their performance is lower than Flair's in the default hyperparameter setting. This potential remains unexplored in this study and could be a focus of future research. In addition, the scope of this study is limited to testing one language only: Indonesian. It would be interesting to discuss whether future research should involve cross-lingual testing, given that each language has its own grammar.

Future research will focus on leveraging such shape information to further improve the performance of contextual embedding–based models. Previous research has shown that morphological and orthographic information, such as prefixes, suffixes, capital letters, and surface forms, can improve model performance when using HMM [25] and CRF [21] methods. We also plan to use the latest character-level embedding approach to capture morphological information more comprehensively. We will apply the attention mechanism to capture global context information in order to address the issues of polysemy and proper nouns that are still problematic in POS tagging. In addition, an adaptive approach using fuzzy logic [47] is also promising when dealing with uncertain conditions, such as those encountered in OOV. These methods present further potential for development to overcome the limitations of contextual models.

7 Conclusion

OOV remains an unresolved problem in POS tagging, primarily due to limited datasets in low-resource languages (LRLs) and the lack of representative features. This challenge is further complicated by the high complexity of grammatical variations. State-of-the-art approaches often rely on character-level embeddings to recognize OOV word forms. However, such information is still insufficient for handling unpatterned OOV words, such as proper nouns and polysemous terms. To address this limitation, this study employed contextual embeddings for more effective OOV handling.

This study compared two types of embeddings—static (Word2Vec, GloVe, FastText) and contextual (ELMo, BERT, Flair)—to evaluate their effectiveness in handling OOV cases. The best-performing embeddings were further fine-tuned to optimize the model by adjusting several hyperparameters, including embedding size, BiLSTM hidden size, dropout rate, training batch size, test batch size, and learning rate. We benchmarked the proposed model against previous approaches based on machine learning (HMM, MEMM, CRF) and deep learning (BiLSTM, BiLSTM+CRF). Model performance was assessed using accuracy and macro F1-score, under two evaluation scenarios: general evaluation and specific evaluation for in-vocabulary (IV) and out-of-vocabulary (OOV) words.

The evaluation results indicated that models with contextual embeddings outperform other approaches. Among the contextual embeddings, Flair achieved the highest performance with an accuracy of 95.65%. Our proposed model also proved effective in handling OOV cases, reaching 88.12% accuracy. Fine-tuning experiments further revealed that hyperparameters such as embedding size, dropout rate, hidden size, and learning rate significantly affect both accuracy and macro F1-score. Despite Flair’s strong performance, several OOV cases remained challenging, particularly word reduplication and capitalized OOV words. Future research should therefore explore the integration of shape-based features to enhance the performance of contextual embedding models.

Acknowledgement

This research was supported in part by Institut Teknologi Sepuluh Nopember [scholarship number 1333/IT2/T/HK.00.01/2022] and the Ministry of Education, Culture, Research, and Technology of Indonesia [grant number 038/E5/PG.02.00.PL/2024]. The authors would like to thank the 25 students from the language and literature department of Airlangga University for assisting in analyzing the data and discussing Indonesian grammar.

References

[1] A. Chiche and B. Yitagesu, "Part of Speech tagging: A systematic review of Deep Learning and Machine Learning approaches," *J Big Data*, vol. 9, no. 1, 2022, doi: 10.1186/s40537-022-00561-y.

[2] M. Alfian, U. L. Yuhana, and D. Siahaan, "Indonesian Part-of-Speech tagger: A comparative study," in *2023 10th International Conference on Advanced Informatics: Concept, Theory and Application (ICAICTA)*, IEEE, Oct. 2023, pp. 1–6. doi: 10.1109/ICAICTA59291.2023.10390353.

[3] A. Kalykulova and A. Nugumanova, "T-Extractor: A Hybrid Unsupervised Approach for Term and Named Entity Extraction Using Rules, Statistical, and Semantic Methods," *Informatica (Slovenia)*, vol. 49, no. 2, pp. 299–318, 2025, doi: 10.31449/inf.v49i2.8148.

[4] S. F. Kusuma, D. O. Siahaan, and C. Fatichah, "Automatic question generation with various difficulty levels based on knowledge ontology using a query template," *Knowl Based Syst*, vol. 249, p. 108906, Aug. 2022, doi: 10.1016/j.knosys.2022.108906.

[5] M. Z. Abdullah and C. Fatichah, "Feature-based POS tagging and sentence relevance for news multi-document summarization in Bahasa Indonesia," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 1, pp. 541–549, 2022, doi: 10.11591/eei.v11i1.3275.

[6] L. Hu, Y. Tang, X. Wu, and J. Zeng, "Considering optimization of English grammar error correction based on neural network," *Neural Comput Appl*, vol. 34, no. 5, pp. 3323–3335, Mar. 2022, doi: 10.1007/S00521-020-05591-2/FIGURES/17.

[7] D. Hoesen and A. Purwarianti, "Investigating Bi-LSTM and CRF with POS Tag Embedding for Indonesian Named Entity Tagger," *Proceedings of the 2018 International Conference on Asian Language Processing, IALP 2018*, pp. 35–38, 2019, doi: 10.1109/IALP.2018.8629158.

[8] J. V. Lochter, R. M. Silva, and T. A. Almeida, "Multi-level out-of-vocabulary words handling approach," *Knowl Based Syst*, vol. 251, Sep. 2022, doi: 10.1016/j.knosys.2022.108911.

[9] P. Kolachina, M. Riedl, and C. Biemann, "Replacing OOV Words For Dependency Parsing With Distributional Semantics," in *NoDaLiDa 2017 - 21st Nordic Conference of Computational Linguistics, Proceedings of the Conference*, 2017, pp. 11–9.

[10] S. Garcia-Bordils *et al.*, "Out-of-Vocabulary challenge report," in *Computer Vision -- ECCV 2022 Workshops*, 2023, pp. 359–375. doi: 10.1007/978-3-031-25069-9_24.

[11] X. Cai, S. Dong, and J. Hu, "A deep learning model incorporating part of speech and self-matching attention for named entity recognition of Chinese electronic medical records," *BMC Med Inform Decis Mak*, vol. 19, 2019, doi: 10.1186/s12911-019-0762-7.

[12] Imamah, U. L. Yuhana, A. Djunaidy, and M. H. Purnomo, "Development of text classification based on difficulty level in adaptive learning system using Convolutional Neural Network," *International Electronics Symposium 2021: Wireless Technologies and Intelligent Systems for Better Human Lives, IES 2021 - Proceedings*, pp. 238–243, Sep. 2021, doi: 10.1109/IES53407.2021.9594021.

[13] F. Gargiulo, S. Silvestri, M. Ciampi, and G. De Pietro, "Deep Neural Network for hierarchical extreme multi-label text classification," *Applied Soft Computing Journal*, vol. 79, pp. 125–138, 2019, doi: 10.1016/j.asoc.2019.03.041.

[14] S. Chotirat and P. Meesad, "Part-of-Speech tagging enhancement to Natural Language Processing for Thai WH-Question classification with Deep Learning," *Heliyon*, vol. 7, no. 10, 2021, doi: 10.1016/j.heliyon.2021.e08216.

[15] S. K. Nambiar, S. Peter David, and S. Mary Idicula, "Abstractive summarization of text document in Malayalam language: enhancing attention model using POS tagging feature," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 22, no. 2, 2023, doi: 10.1145/3561819.

[16] W. Liu and L. Wang, "POS-tagging enhanced Korean text summarization," in *Intelligent Computing Methodologies*, Springer International Publishing, 2017, pp. 425–435. doi: 10.1007/978-3-319-63315-2_37.

[17] W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed, "Automatic Text Summarization: A comprehensive survey," Mar. 01, 2021. doi: 10.1016/j.eswa.2020.113679.

[18] V. H. Vu, Q. P. Nguyen, K. H. Nguyen, J. C. Shin, and C. Y. Ock, "Korean-Vietnamese neural machine translation with named entity recognition and part-of-speech tags," *IEICE Trans Inf Syst*, vol. E103D, no. 4, 2020, doi: 10.1587/transinf.2019EDP7154.

[19] M. Alfian, U. L. Yuhana, D. Siahaan, H. Munazharoh, and E. Pardede, "Out-of-Vocabulary Handling in Part-of-Speech Tagging: A Semantic Web-Driven Systematic Review," *Int*

J Semant Web Inf Syst, vol. 21, pp. 1–36, Sep. 2025, doi: 10.4018/IJSWIS.388421.

[20] Muljono, U. Afini, and C. Supriyanto, "Morphology analysis for Hidden Markov Model based Indonesian Part-of-Speech tagger," in *2017 1st International Conference on Informatics and Computational Sciences (ICICoS)*, 2017, pp. 237–240. doi: 10.1109/ICICOS.2017.8276368.

[21] I. I. Ayogu, A. O. Adetunmbi, B. A. Ojokoh, and S. A. Oluwadare, "A comparative study of hidden Markov model and conditional random fields on a Yorùba part-of-speech tagging task," in *Proceedings of the IEEE International Conference on Computing, Networking and Informatics, ICCNI 2017*, 2017. doi: 10.1109/ICCNI.2017.8123784.

[22] K. Nowakowski, M. Ptaszynski, F. Masui, and Y. Momouchi, "Improving Basic Natural Language Processing Tools for the Ainu Language," *Information 2019, Vol. 10, Page 329*, vol. 10, no. 11, p. 329, Oct. 2019, doi: 10.3390/INFO10110329.

[23] S. N. Bhattu, S. K. Nunna, D. V. L. N. Somayajulu, and B. Pradhan, "Improving code-mixed POS tagging using code-mixed embeddings," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 19, no. 4, p. 1, 2020, doi: 10.1145/3380967.

[24] L. Moudjari, F. Benamara, and K. Akli-Astouati, "Multi-level embeddings for processing Arabic social media contents," *Comput Speech Lang*, vol. 70, 2021, doi: 10.1016/j.csl.2021.101240.

[25] M. Janicki, "Semi-supervised induction of POS-tag lexicons with tree models," in *International Conference Recent Advances in Natural Language Processing, RANLP*, 2019, pp. 507–515. doi: 10.26615/978-954-452-056-4_060.

[26] L. Keiper, A. Horbach, and S. Thater, "Improving POS tagging of German learner language in a reading comprehension scenario," in *Proceedings of the 10th International Conference on Language Resources and Evaluation, LREC 2016*, 2016.

[27] A. Jettakul, C. Thamjarat, K. Liaowongphuthorn, C. Udomcharoenchaikit, P. Vateekul, and P. Boonkwan, "A comparative study on various Deep Learning techniques for Thai NLP lexical and syntactic Tasks on noisy data," in *Proceeding of 2018 15th International Joint Conference on Computer Science and Software Engineering, JCSSE 2018*, 2018. doi: 10.1109/JCSSE.2018.8457368.

[28] D. G. Anastasyev, A. I. Andrianov, and E. M. Indenbom, "Part-of-speech tagging with rich language description," in *Komp'juternaja Lingvistika i Intellektual'nye Tehnologii*, 2017.

[29] E. Partalidou, E. Spyromitros-Xioufis, S. Doropoulos, S. Vologiannidis, and K. I. Diamantaras, "Design and implementation of an open source Greek POS Tagger and Entity Recognizer using spaCy," in *Proceedings - 2019 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2019*, 2019, pp. 337–341. doi: 10.1145/3350546.3352543.

[30] H. Yu, J. An, J. Yoon, H. Kim, and Y. Ko, "Simple methods to overcome the limitations of general word representations in natural language processing tasks," *Comput Speech Lang*, vol. 59, pp. 91–113, 2020, doi: 10.1016/j.csl.2019.04.009.

[31] M. S. Won, Y. S. Choi, S. Kim, C. W. Na, and J. H. Lee, "An embedding method for unseen words considering contextual information and morphological information," in *Proceedings of the ACM Symposium on Applied Computing*, 2021, pp. 1055–1062. doi: 10.1145/3412841.3441982.

[32] Y. Liu, W. Che, Y. Wang, B. Zheng, B. Qin, and T. Liu, "Deep contextualized word embeddings for universal dependency parsing," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 19, no. 1, pp. 1–17, 2019, doi: 10.1145/3326497.

[33] F. Marulli, M. Pota, and M. Esposito, "A comparison of character and word embeddings in bidirectional LSTMs for POS tagging in Italian," in *Smart Innovation, Systems and Technologies*, 2019, pp. 14–23. doi: 10.1007/978-3-319-92231-7_2.

[34] S. Fu, N. Lin, G. Zhu, and S. Jiang, "Towards Indonesian Part-of-Speech tagging: Corpus and models," *2018 International Conference on Asian Language Processing (IALP)*, vol. 1, pp. 303–307, 2018.

[35] A. Millour and K. Fort, "Unsupervised data augmentation for less-resourced languages with no standardized spelling," in *International Conference Recent Advances in Natural Language Processing, RANLP*, 2019, pp. 776–784. doi: 10.26615/978-954-452-056-4_090.

[36] G. Antipov, S. A. Berrani, N. Ruchaud, and J. L. Dugelay, "Learned vs hand-crafted features for pedestrian gender recognition," *MM 2015 - Proceedings of the 2015 ACM Multimedia Conference*, pp. 1263–1266, Oct. 2015, doi: 10.1145/2733373.2806332.

[37] P. Passban, Q. Liu, and A. Way, "Boosting neural Pos tagger for farsi using morphological information," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 16, no. 1, pp. 1–15, 2016, doi: 10.1145/2934676.

[38] M. Alfian, U. L. Yuhana, D. Siahaan, H. Munazharoh, and E. Pardede, "Handling Out-of-Vocabulary in Indonesian POS Tagging: A Comparative Study," in *2025 International Conference on Smart Computing, IoT and Machine Learning, SIML 2025*, Surakarta: Institute of Electrical and Electronics Engineers Inc., Jul. 2025, p. 1. doi: 10.1109/SIML65326.2025.11080832.

[39] Y. Kimura, T. Komamizu, and K. Hatano, "An Automatic Labeling Method for Subword-Phrase Recognition in Effective Text Classification," *Informatica (Slovenia)*, vol. 47, no. 3, 2023, doi: 10.31449/inf.v47i3.4742.

[40] A. Makazhanov and Z. Yessenbayev, "Character-based feature extraction with LSTM networks for POS-tagging task," in *Application of Information and Communication Technologies, AICT 2016 - Conference Proceedings*, 2017. doi: 10.1109/ICAICT.2016.7991654.

[41] A. Kemos, H. Adel, and H. Schütze, "Neural semi-Markov conditional random fields for robust character-based part-of-speech tagging," in *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 2019, pp. 2736–2743.

[42] P. Boonkwan and T. Supnithi, "Bidirectional deep learning of context representation for joint word segmentation and POS tagging," in *Advances in Intelligent Systems and Computing*, 2018, pp. 184–196. doi: 10.1007/978-3-319-61911-8_17.

[43] M. Pota, F. Marulli, M. Esposito, G. De Pietro, and H. Fujita, "Multilingual POS tagging by a composite Deep Architecture based on Character-Level features and on-the-fly enriched Word Embeddings," *Knowl Based Syst*, vol. 164, pp. 309–323, 2019, doi: 10.1016/j.knosys.2018.11.003.

[44] M. Alfian, U. L. Yuhana, D. Siahaan, and H. Munazharoh, "Annotation Error Detection and Correction for Indonesian POS Tagging Corpus," *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, vol. 16, no. 1, p. 41, Jun. 2025, doi: 10.24843/lkjiti.2025.v16.i01.p04.

[45] K. Kurniawan and A. F. Aji, "Toward a standardized and more accurate Indonesian Part-of-Speech tagging," *Proceedings of the 2018 International Conference on Asian Language Processing, IALP 2018*, pp. 303–307, 2019, doi: 10.1109/IALP.2018.8629236.

[46] N. Reimers and I. Gurevych, "Reporting score distributions makes a difference: Performance Study of LSTM-networks for sequence tagging," in *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 2017. doi: 10.18653/v1/d17-1035.

[47] A. Boulkroune, F. Zouari, and A. Boubellouta, "Adaptive fuzzy control for practical fixed-time synchronization of fractional-order chaotic systems," *JVC/Journal of Vibration and Control*, 2025, doi: 10.1177/10775463251320258.

[48] S. Besharati, H. Veisi, A. Darzi, and S. H. H. Saravani, "A hybrid statistical and deep learning based technique for Persian part of speech tagging," *Iran Journal of Computer Science*, vol. 4, no. 1, p. 35, 2021, doi: 10.1007/s42044-020-00063-1.

[49] N. Bölücü and B. Can, "Unsupervised joint PoS tagging and stemming for agglutinative languages," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 18, no. 3, pp. 1–21, 2019, doi: 10.1145/3292398.

[50] B. Wang, A. Wang, F. Chen, Y. Wang, and C. C. J. Kuo, "Evaluating word embedding models: Methods and experimental results," 2019, *Cambridge University Press*. doi: 10.1017/ATSIP.2019.12.

[51] T. Gui, Q. Zhang, H. Huang, M. Peng, and X. Huang, "Part-of-speech tagging for twitter with adversarial neural networks," in *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 2017, pp. 2411–2420. doi: 10.18653/v1/d17-1256.

[52] L. Qu, G. Ferraro, L. Zhou, W. Hou, N. Schneider, and T. Baldwin, "Big data small data, in domain out-of domain, known word unknown word: The impact of word representations on sequence labelling tasks," in *CoNLL 2015 - 19th Conference on Computational Natural Language Learning, Proceedings*, 2015, pp. 83–93. doi: 10.18653/v1/k15-1009.

[53] J. Wulff and A. Søgaard, "Learning finite state word representations for unsupervised Twitter adaptation of POS taggers," in *ACL-IJCNLP 2015 - Workshop on Noisy User-Generated Text, WNUT 2015 - Proceedings of the Workshop*, 2015, pp. 162–166.

[54] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. Pearson Education, 2024.