IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 12 August 2022, accepted 11 September 2022, date of publication 21 September 2022, date of current version 3 October 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3208269*

RESEARCH ARTICLE

# BaNeP: An End-to-End Neural Network Based Model for Bangla Parts-of-Speech Tagging

**JESAN AHAMMED OVI, MD. ASHRAFUL ISLAM, AND MD. REZAUL KARIM**
Department of Computer Science and Engineering, University of Dhaka, Dhaka 1000, Bangladesh
Corresponding author: Md. Rezaul Karim (rkarim@du.ac.bd)

This work was supported by the Centennial Research Grant (CRG), University of Dhaka.

**ABSTRACT** In Natural Language Processing, Parts-of-Speech tagging is a vital component that significantly impacts applications like machine translation, spell-checker, information retrieval, and speech processing. In languages such as English and Dutch, POS tagging is considered a solved problem (accuracy: 97%). However, for low-resource languages like Bangla, challenges are still there. In this article, we have proposed a novel RNN-based network named BaNeP to determine parts of speech for Bangla words. The proposed network extracts structural features through a bidirectional LSTM-based sub-network, and intricate contextual relations among words of a sentence are identified through an elaborate weighted context extraction procedure. These features are then combinedly utilized to generate the final Parts-of-Speech prediction. Training the model requires only an annotated dataset vanishing the need for any hand-crafted features. Experimental results on the LDC2010T16 dataset show significant accuracy improvement compared to existing Bangla POS taggers.

**INDEX TERMS** Bangla, POS tagging, RNN, sequence labeling.

## I. INTRODUCTION

Part-of-Speech (POS) Tagging, also known as grammatical tagging or word category disambiguation, is a popular natural language processing (NLP) task that refers to mapping words in a text or corpus to corresponding part-of-speech, depending on the structure of the word and its context. Although POS tagging may not be the solution to any particular NLP problem alone, it is a pre-requisite for many NLP applications as it provides a linguistic signal on how a word is being used within the scope of a phrase, or sentence and document. Earlier, when there was no language to communicate, humans used sign language to exchange their thoughts, like how we communicate with our pets. Suppose, when we tell our dog, ''Cooper, we love you'', he responds by wagging his tail. This does not mean he actually understands what we say, but he can read our expressions, and understand our emotions and gesture more than words. As the most intellectual being, human has developed an understanding of many nuances of natural languages more than any other animals on this planet. That is why, when someone says, ''I LOVE you, Copper'' vs. ''My LOVE, let's go for a long driv''e, the word 'LOVE' has a different meaning. In the first phrase, 'LOVE' is used to express the speaker's love for her/his pet, whereas in the second phrase, the speaker uses the word 'LOVE' to indicate her/his dearest person. As humans, we can understand the contextual meaning of the word LOVE in these two phrases, and that is why, responses will be different. Nevertheless, trying to teach our machines to understand these intricate contextual differences becomes onerous. So, in the future, when we develop a home care robot that hears, ''I LOVE you, Copper'', it will understand that LOVE is a verb, and the speaker's emotion toward the dog is expressed. So the robot should pay more attention to caring the Copper vs. ''My LOVE, let's go for a long drive'', where LOVE is a noun, and the robot can understand this is not of its business; it simply leaves the room. This example highlights how important it is to identify Part-of-Speech of a particular word to understand the meaning of the word in a different context. An application like text-to-speech conversion performs POS tagging as a part of preprocessing. For example, in the sentence ''They refuse to permit us to obtain the refuse permit'', the word

The associate editor coordinating the review of this manuscript and approving it for publication was Long Xu.

'refuse' has been used twice with two different meanings. In the first case, 'refuse' is a verb meaning 'deny' while 'refuse' is a noun meaning 'trash' later. These two 'refuse'-es are not homophones. So, it is crucial to identify the proper Part-of-Speech of a word to pronounce it correctly. Other popular NLP applications, such as information retrieval, emotion analysis, spell checking, word sense disambiguation, etc., also perform POS tagging in preprocessing.

Considering the importance of POS tagging in NLP, a tremendous amount of research has been done and still going on to develop an efficient network for languages like English, Dutch and Chinese. Traditional high performance model for those languages are mostly based on Hidden Markov Model (HMM) and Conditional Random Fields (CRF) [1], [2], [3]. However, those models require hand-crafted features and task-specific resources like carefully designed word spelling features, orthographic features and gazetteers. These task-specific resources make the system costly to develop and difficult to adapt to new tasks or domains. Recently, a non-linear neural network with a distributed word representation known as a word embedding system has been broadly applied for higher accuracy. In the past few years, researchers have already developed high accuracy systems with the help of Recurrent Neural Network (RNN) or with its variant such as Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU) for high-resource languages like English [4], [5], [6]. But low-resource languages such as Bangla still lack efficient and accurate POS tagging models. Although, various models have been proposed for Bangla POS tagging [7], [8], [9], practically usable models are still in demand.

Bangla has been ranked 7th among all spoken languages around the world. About 265 million people across the world use the Bangla language. So, NLP applications such as voice recognition, information retrieval, and emotion analysis for the Bangla language will significantly impact this large population. In this direction, this work proposes an RNN-based network for the Bangla language to detect part of speech of a particular word in phrase, sentence or document. More formally, a text corpus is given, and the model's target is to identify the part of speech of each word. For example, consider the following Bangla sentence 'S' of a corpus

কমলে (Komole) ভরে (vore) গেছে (gese) পুকুর (pukur): The pond has been filled with lotus.

The sentence 'S' consists of four words that are 'কমলে', 'ভরে', 'গেছে', 'পুকুর' So, we can consider a sentence as a sequence of words $W_k$, that is

$$S = \{W_0, W_1, W_2, \ldots, W_k, \ldots, W_m\}.$$

Besides, regardless of the language, the structural definition of that word is essential for identifying the proper POS tag of a word. For this reason, the character-level information of a particular word is extracted. That is why, a word $W_k$ can be further considered as a sequence of characters, for example,

$$W = \{C_0, C_1, C_2, \ldots C_n\}.$$

The main objective of the POS tagging method is to find the proper Part-of-Speech for each word $W_k$ of the sentence S. For example, the POS tag for the word the 'পুকুর' will be a common noun. So, we can narrow down our task to take a sequence as input and find a tag for it as we view a word as a sequence of characters, which is ultimately aligned with the sequence labeling procedure, a popular and crucial process in NLP.

Generally, two types of information are processed to determine part of speech of a particular word: word definition or word structure and its sense or contextual relation with other words in the sentence. For example, in the English language, words having suffix -ion, -sion, -tion, -acy, -ment, etc., are commonly categorized as a noun such as population, accuracy, and government. Similarly, words ending with -ly, -ful, -ous, etc., are generally considered as an adjective. But some exceptions exist, such as 'happily' ends with -ly seems to be an adjective, but it is an adverb. So, word definition or structure will not help much to identify the POS tag of a word. The situation becomes more arduous when we need to understand the word sense or purpose of its usage within a particular sentence or phrase. For example,

- NOUN - He carried a log on his back.
- VERB - He did not back me in this case.
- ADJECTIVE - He went through the back door.
- ADVERB - He turned back to look at me.

Here, the word 'back' has different POS tags in different sentences depending on its sense and context. Some words can be converted to a different Part-of-Speech in English, like 'choose' is a verb and its corresponding noun is 'choice'. In this case, the word structure is also changed with the Parts-of-Speech. However, in the previous case, the POS tag for the word 'back' entirely depends on the context, which is quite difficult to understand not only for machines but also for us having intelligence. So, word structure provides some information regarding its POS tag, but the context of the word within the corpus significantly impacts determining the proper POS tag. Imagine the situation when the corpus is a novel, poetry, or some intellectual metaphor [10], [11] written by some world-famous writers. How challenging will it be to extract the word context from such corpus? So, it is evident that the POS tagging process of languages with less structural features and depend heavily on contextual sense will be more challenging. That is why less resourceful languages like Bangla do not have significant POS tagging methodologies with noticeable accuracy. Let us take a more detailed look at the origin and evaluation of the Bangla language to understand its challenges and associated difficulties.

History reveals that the Bangla language can be traced back to 3500 B.C. to the Indo-European language family. It has been assumed by many scholars and linguistics that Bangla was born from Sanskrit. But, others believe that it derives

from Indo-Aryan languages like Magadhi Prakrit and Pali. Modern Bangla also uses words taken from Turkish, Portuguese, Persian or English, for example. In short, Bangla has been enriched by various languages over time. Evaluation of Bangla can be divided into three stages: Old Bangla, Middle Bangla and Modern Bangla. Old Bangla dates back to around 650 C.E., when priests and scholars widely used Sanskrit in literary works in Bangla. Few traces of Bangla literature are found at that time. The oldest text ever found is Charyapada. During the 14th century, Muslim rulers established the Sultanate of Bengal and declared Bangla the region's official court language. Soon, it became the vernacular language of Bengal. However, this Middle version of the Bangla language is still very different from modern Bangla. In the 16th century, the Mughals took over Bengal. The rich Persian language they brought began to influence the language [12], [13]. For example,

- Sky: Asman আসমান
- Heaven: Behesto বেহেস্ত
- Dojokh/Dozokh দজখ
- Land: Jomi জমি/জমিন
- Field: Moydan ময়দান
- Storm: Toofan তুফান
- River/Sea: Doria দোরিয়া

What we call modern Bangla took shape from the dialect spoken in the Nadia region of Bengal around the time of the Battle of Plassey in 1757. However, the language was split into formal and causal variations, known as ''shuddho bhasha'' and ''cholito bhasha''. That introduces more difficulties in the structure of a similar word. For example,

রবি (Robi) বলিয়াছে (boliyase), সে (shey) আজ (aj) এইখানে (eikhane) আসিতে (asite) পারিবে (paribe) না (na): Robi has said that he can not come here today. Here, the words বলিয়াছ: have said, আসিতে : Come, এইখানে: Here, and পারিবে : Can are used in Shuddho bhasha or formal form, have different structures but similar meaning in their corresponding cholito bhasha or causal form. Consider the same sentence in cholito bhasha: রবি (Robi) বলেছে (bollo), সে (se) আজ (aj) এখানে (ekhane) আসতে (aste) পারবে (parbe) না (na). Here, words, বলিয়াছ becomes বলেছে, আসিতে becomes আসতে, এইখানে becomes এখানে and পারিবে becomes পারবে. All of these words have different structures but the same meaning. The situation becomes more onerous when words in the same form have different meanings and structures but similar POS tags. For example, consider the following two phrases,

Phrase 1: সে (se) গেল (gelo): he/she went (a few moments ago).

Phrase 2: সে (se) গিয়েছিল (giyechilo): she/he went (long ago).

Here, we can see the word গেল, গিয়েছিল is in the same formal form but has a different structure. In English, the corresponding word is just 'went', and it requires other words like 'long ago' or a few moments ago to express a similar meaning. There are five main categories of Part-of-Speech in Bangla:

বিশেষ্য, বিশেষণ, সর্বনাম, অব্যয়, ক্রিয়া

Every type has subcategories, just like English and other languages. To make the situation more challenging, almost every word from every subcategory has a ''shuddho/formal'' form and a ''cholito/causal'' form.

After the battle of Plassey, western influence began shaping local cultural norms. As a result, the English language started to impose a significant impact on Bangla. Words such as Chair, Table, Cup, Television, Radio, and many more are used in the Bangla language directly brought from English. Urdu also influenced Bangla after the partition in 1947. Words like Water: pani জল, Salt: lobon নুন, Invitation: daoat আমন্ত্রণ, Bath: gosol স্নান, Wind: batas হাওয়া, etc. are used in Bangla which are actually derived from Urdu language. The above discussion indicates that several languages have enriched the Bangla language over time. That is why words in Bangla bear less structural similarity and require tremendous processing to determine the word relation, context, purpose, and sense within a phrase, sentence, or corpus. Besides, in the English language, the word structure is changed based on tense, such as, 'go' becomes 'went,' but in past tense word remains unchanged with respect to the person. For example, ''I go'' vs. ''He/She goes''. Here, in the present tense, 'go' changes its structure, but in the past tense, it remains unchanged like ''I went'' and ''He/She went''. However, in Bangla, words' structure changed for both person and tense. For example, in the present tense, consider the following two sentences,

আমি (ami) বই (boi) কিনতে (kinte) এসেছি (esheysi): I have come to buy books.

সে (se) বই (boi) কিনতে (kinte) এসেছে (esheyse): He has come to buy books.

Here, we can see the word আসা has two forms depending on the person আমি and সে just like the English language. However, if we convert those sentences in the past tense, unlike in English, the word আসা changes its structure too. For example,

আমি (ami) বই (boi) কিনতে (kinte) এসেছিলাম (eshesilam): I came to buy books.

সে (se) বই (boi) কিনতে (kinte) এসেছিল (eshesilo): He came to buy books - the same word has a different structure in past tense also. Moreover, like other languages, Bangla also suffers from ambiguous situations, with words having the same structure but different parts of speech depending on the context. In the following two sentences, the word কমলে has the same structure but a different pronunciation and POS tag.

কমলে (komoley) ভরে (vore) গেছে (gese) পুকুর (pukur): The pond has been filled with lotus.

দাম (dam) কমলে (komley) কিনব (kinbo): I will buy if the price goes down.

In the first sentence, কমলে is বিশেষ্য (noun), but in the second sentence, it is used as ক্রিয়া (verb). In this article, we aim to consider the above situation and various challenges in the Bangla language to design an efficient model to determine the proper POS tag for words of a particular corpus. Our key contributions are,

- Structural Feature Extraction Network (SFENet) to identify the word structure.
- Context encoder to extract word-level contextual information of a sentence or phrase.
- Weighted Context Generation Network (WCGNet) for each word in a sentence to extract the weighted influence of other words of the sentence (Contextual Feature).
- A prediction generator network that combines structural and contextual features along with discovering inter-word structural dependency in a sentence to generate the final prediction.

The rest of the paper is organized as follows: Section II describes the procedures, networks, models, etc., regarding and relating to POS-Tagging tasks. The architecture of the proposed model has been described in detail in Section III. We analyze the dataset used to evaluate our model in section IV. Performance analysis using various criteria and comparative studies with existing state-of-the-art procedures has been given in section V. Finally, the conclusion of this article has been drawn in Section VI.

## II. RELATED WORKS

This section provides a detailed overview and comparative study of existing methodologies with the proposed model. For a better explanation, our studies are divided into three subsections. Models regarding sequence labeling are discussed first, as POS tagging is a sequence labeling problem. Then an overview of those procedures proposed to handle languages closely related to Bangla is given. Finally, networks that are dedicated to the Bangla language are ventilated.

### A. SEQUENCE LABELING

The task of linguistic sequence labeling can be viewed in two eras. Traditional models are based on the linear probabilistic process that incorporates the Hidden Markov Model (HMM), Conditional Random Field (CRF), Maximum Entropy (ME), etc. Ma and Xia proposed a probabilistic model for unsupervised dependency parsers for languages with no labeled training data, but it requires translated text [14]. Such task-specified models are costly to develop and can not be adapted easily for new domains. Focusing on the increasing demand for Crowdsourcing, Rodrigues *et al.* developed a CRF-based model using the Expectation Maximization algorithm to learn the CRF model parameters [15]. They built their model as a partial task of name entity recognition (NE) and noun phrase chunking. They have evaluated their model on both synthetic data artificially generated by simulating multiple annotators and real data obtained from Amazon's Mechanical Turk. For data with repeating labels, they got a precision of 72.5% and a recall of 67.7%. In the case of non-repeating labels, precision and recall were 65.7% and 62.7%, respectively. Santos and Zadrozny [16] proposed a remarkable neural network-based model. They designed their core architecture named 'Char-WNN' for extracting character-level information using the LSTM-CNN model and achieved state-of-the-art accuracy (97.32%) for the English language on the Penn Treebank WSJ corpus [17]. Later, Huang *et al.* proposed a model that uses BiLSTM for word-level representation and CRF to label sequence [18] jointly. However, these RNN-based models, rather than replacing, combined hand-crafted features for better performance. As a result, the performance of their system drops rapidly when the model solely depends on neural embedding. Focusing on those shortages, a sophisticated neural network-based architecture that incorporates BiLSTM, CNN and CRF was proposed by Ma and Hovy [19]. CNN is used to encode character-level information of a word. Then, character and word level representations are combined to feed BiLSTM and CRF to decode labels for the entire sentence. That is how their model benefits from both word and character level information without requiring task-specific resources, feature engineering, or data preprocessing beyond pre-trained word embedding on the unlabeled corpus. Experiments showed that the proposed model got 97.21% accuracy on POS tagging for Penn Treebank WSJ corpus and 91.21% accuracy on named entity recognition for CoNLL 2003 corpus [20]. In 2019, Akbik A *et al.* came up with the idea of contextual string embedding, where a character language model is trained to leverage its internal state in order to produce a novel word embedding system [21]. They claimed that considering words as a sequence of characters, and their model was trained without any explicit notion of words. Besides, they used different embedded representations of the same word depending on its context. Their model's performance was evaluated on the CONLL 2003 shared task dataset and achieved a state-of-the-art F1 score for English (93.09%) and German (88.32%). Tasi *et al.* [22] proposed BERT based model for sequence labeling task. They were able to design a small, faster, and high-accuracy model for multilingual language that beat the sate-of-the-art baseline [23], [24]. The problem is that in some languages like Bangla, determining the contextual relation of a particular word may require longer text. However, BERT or other transformer-based models can only grasp contextual relation from a user-defined fixed length. An attention-based RNN model has been proposed by Lin *et al.*, which relies on a hierarchical attention neural semi-Markov conditional random fields (semi-CRF) model for the task of sequence labeling [25]. In their proposed model, character and word level information, along with an attention mechanism, is used to determine the sequence label. Their model is evaluated on three sequence labeling tasks: named entity recognition, chunking, and reference parsing. Experimental results showed that their model achieves competitive and robust performance in all three tasks. This is undoubtedly a remarkable work, but their model is designed for generic sequence labeling tasks. The authors did not pay any particular attention to the complex cases that can occur in POS tagging tasks, as mentioned in Section I, especially for the languages that have a minuscule dependency on structural information and demand for significant processing in contextual relation like Bangla language. As a result, their

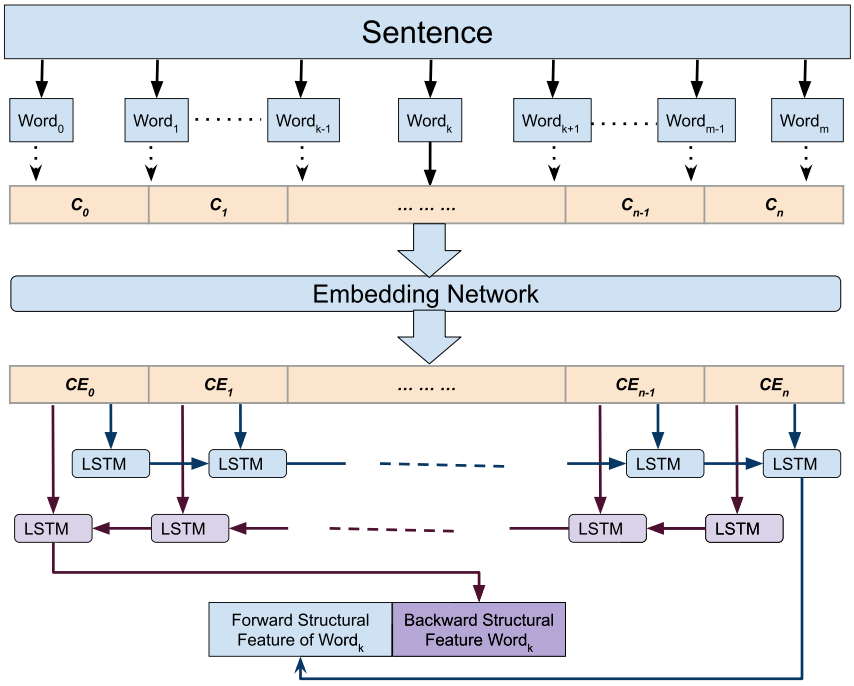


**FIGURE 1.** SFENet.

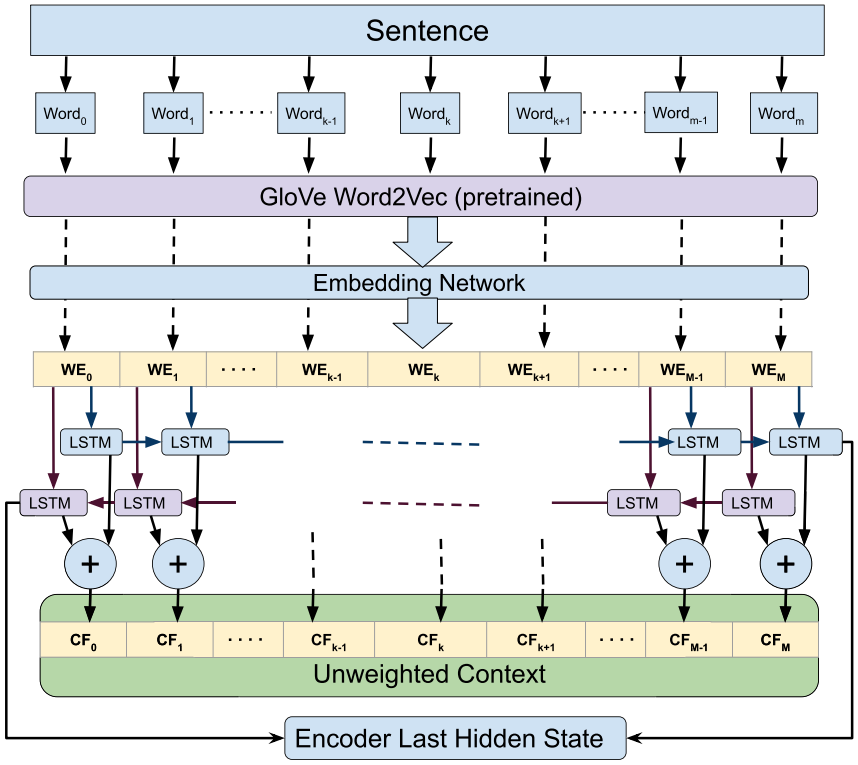


**FIGURE 2.** Context encoder.

model exhibits impressive performance on generic sequence labeling tasks but fails to exhibit that performance in Bangla POS tagging. Performance comparison between this model and ours in the experimental result section (Section V) provides more clear picture in this regard.

### *B. PARTS OF SPEECH TAGGING FOR INDO-ARYAN LANGUAGES*

This section focuses on parts-of-speech taggers of the languages that belong to the Indo-Aryan language family, like Bangla. One of the oldest approaches in Parts-of-Speech

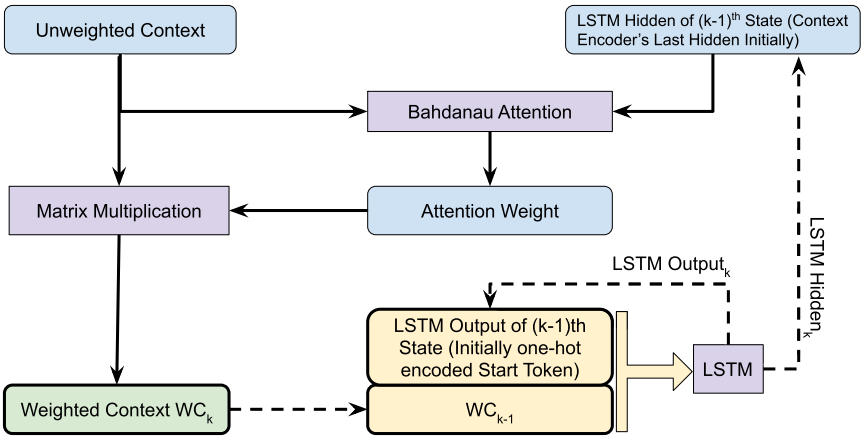


**FIGURE 3.** WCGNet.

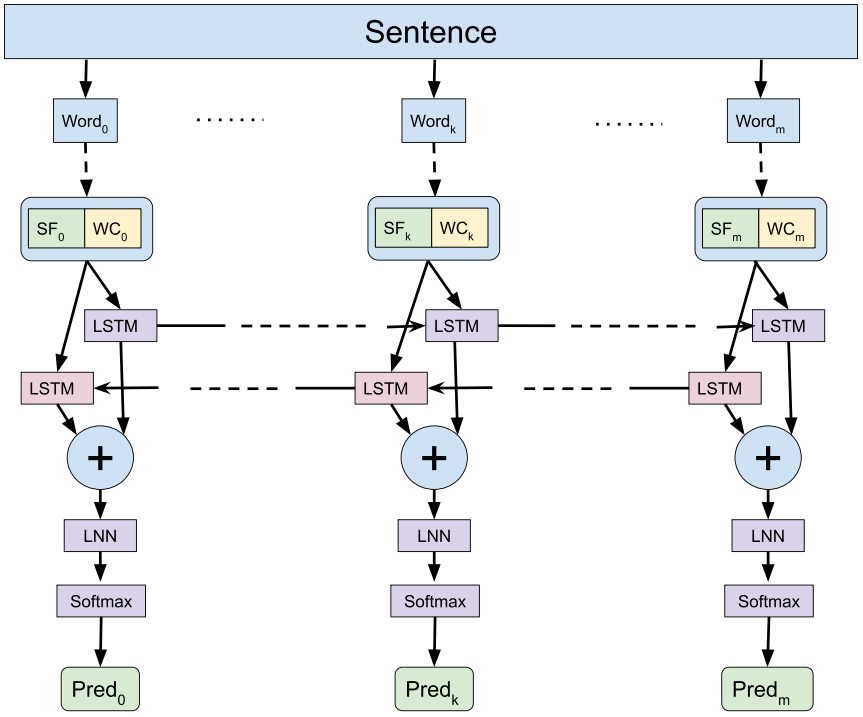


**FIGURE 4.** Prediction generator.

tagging is the Rule-based approach claimed by Kumawat and Jain [26] which is based on a handwritten set of rules. These rules can include punctuation and capitalization patterns in a set of sentences or words. In 2017, Sonai *et al.* [27] proposed a similar rule-based model for the Malay language. Although ruled base models do not require storing a large amount of corpus to estimate probabilistic values or other kinds of parameter, it requires linguistic background and manual constructions of rules. Besides, it can not ensure that every linguistic rule is covered in the rule constructions process. Moreover, it is pretty strenuous to transfer such a model from one language to another. Alongside rule-based approaches, researchers have also focused on designing stochastic methods for POS-Tagging using HMM, CRF, ME, Maximum Likelihood Estimation, Support vector machine, N-gram, etc. Dalal *et al.* applied Maximum Entropy Markov (MEM) with a rich set of features in the Hindi language. They achieved 94.89% accuracy, the highest reported accuracy for Hindi language POS-Tagging at that time. Another research carried out by Pisceldo *et al.* [28] on stochastic POS-Tagging techniques for the Indonesian language. They constructed their model using CRF and ME methods for assigning the

POS tag to a word [28]. Over the years, researchers such as Nunsanga *et al.* [29], Shafi *et al.* [30], Singha *et al.* [31], Dalai *et al.* [32] proposed probabilistic models for Mizo, Urdu, Manipuri and Odia language respectively. Transportation of a language model to another language is easier in the case of such probabilistic models. Besides, on data availability, models can automatically re-adjust the probability of various parameters. Nevertheless, the problem is that such models require a large corpus of data with hand-crafted features, making the system costly to develop. On top of that, such systems fail to generate accurate results when new data is encountered. Apart from these methods, another popular and efficient approach for POS-Tagging tasks is using different artificial neural network-based algorithms like Multi-layer perceptron (MLE), CNN, RNN, GRU, LSTM, BiLSTM, etc. [33]. A neural network-based architecture was proposed to determine the POS tag in the Odia language by Das and Patnaik [34]. They used a simple multi-layer perceptron with a fixed length of context but did not use any character and word-level information. That is why their proposed model failed to determine the complex contextual relation of words within the corpus. As a result, they were able to achieve only 81% accuracy. In 2020, Akhil *et al.* [35] came up with a POS-Tagging model for Malayalam: a low-resource language. In their proposed model, inputs are represented as a one-hot vector fed to LSTM/GRU/BiLSTM layer. Finally, the softmax function is used to generate the prediction. The limitation is that they did not consider the character-level information within a word. Further, as they use a simple architecture of RNN, their model failed to capture the long contextual relation of words.

### C. BANGLA POS TAGGING

In this section, we analyze those methodologies designed for Bangla POS-Tagging. We also extend our discussion over languages almost similar to Bangla, such as Assamese. We start our discussion with a rule-based approach proposed by Chowdhury *et al.* [36]. In their proposed model, the authors consider the suffix structure of a particular word, and they did not determine any relation of words within a sentence. This model is the first step toward an automated morphological analysis of the Bangla language. Over time, several probabilistic models have been developed by researchers for Bangla POS-Tagging system. Among them, approaches proposed by Mukherjee and Mandal, Ekbal *et al.*, and Dandapat *et al.* can be mentioned [37], [38], [39]. Authors of [37] applied traditional probabilistic algorithms like CRF, SVM, HMM, ME for POS-Tagging. Although they got high accuracy (93.12%) for their model, they did not use any standard and sufficiently resourceful dataset. Ekbal A *et al.* conducted their experiment on a corpus having 72,341 tokens with 26 tags (Accuracy: 90.3%) which is a sufficiently large dataset but not good enough. In their proposed model, the authors include NER, lexicon, and word suffix features, which are hand-crafted features that take time to develop. Besides, they fixed the context window of size four, but a particular word within a corpus may have a more extended contextual dependency. The method of Dandapat *et al.*, an HMM and ME-based architecture, got an accuracy of 88.41% on a moderate dataset. An impactful work has been done by Alam *et al.* [40] which is very similar to the approach presented by Huang *et al.* [18]. They designed a BiLSTM-CRF network for Bangla POS-Tagging using a standard publicly accessible dataset from LDC and got a good accuracy of 86%. In their model, character-level char-embeddings representations are employed to capture prefix and suffix information. Later, world-level distributed representation is implemented as it can capture positional information. Finally, both representations are combined and fed to BiLSTM, followed by CRF to generate the final verdict. Authors adopted the similar architecture of the model proposed by Ma and Hovy [19] except for character embedding. Their character embedding is identical to Huang *et al.* [18]. However, both Huang *et al.* and Ma and Hovy carefully design their model for English and similar languages. As discussed earlier in the introduction section (Section I), huge structural differences exist between the English and Bangla languages. Besides, challenges for Bangla and English POS-Tagging are pretty dissimilar. Bangla requires tons of contextual relations to determine the proper tag. In our proposed methodology, we have developed an efficient architecture specifically considering the difficulties of the Bangla language. As the proposed model of Alam F *et al.* is evaluated using a standard dataset with reasonable accuracy, we have considered their model as one of the benchmark approaches and compare our result with them in the comparative performance analysis section (Section V-E). In between, several other researches are also done in Bangla POS-Tagging, but nothing seems convincing in terms of accuracy, dataset they used, and model architecture. For instance, Kabir *et al.* trained a Deep Belief Network (DBN) on the LDC dataset for the Bangla POS tagging task and claimed to achieve an accuracy of 93.33% [41]. But, the authors did not delineate their model's architecture, making it impossible to replicate their network. Even optimal hyper-parameter combinations for the reported accuracy are also not mentioned. In 2018, Uddin *et al.* proposed a feed-forward neural network approach for Bangla POS tagging [42]. They built a tree structure named Trie to capture the structural similarity of the word. But, in the case of POS-Tagging, a similar word may have a different POS tag depending on the context. After that, the authors deploy a simple FNN to calculate a word's probability of having a particular POS tag. Their model did not determine any long-term dependency of a particular word within its context. Moreover, they did not use any remarkable dataset to evaluate their model. A rule-based approach with an approximate accuracy of 94% is proposed by Roy *et al.* [43] where authors constructed several grammatical rules to identify the POS tag of a particular word. As mentioned earlier, rule-based models have lower adaptability to new or unseen contexts, reducing their dependability. Besides, it does not guarantee that all rules have been exhaustively considered.

Considering all the approaches presented in this section, it is evident that a robust POS-Tagger for Bangla is yet to be developed that can practically be used in other NLP-focused applications. Our proposed BaNeP model utilizes only an annotated dataset that drops the need for the hand-crafted feature in the training phase and considers structural and contextual tagging features. A multi-phase contextual feature extraction network enables BaNeP to address the limitations of existing neural network-based approaches and generate the better prediction.

## III. PROPOSED MODEL

The proposed BaNeP model comprises three sub-networks; two are dedicated to extracting features, and the third one generates the final prediction based on the extracted features on a per-word basis. As mentioned earlier, both the structure of a word and the relative position of the word in the context of a sentence with semantic significance play deciding role in Parts-of-Speech identification. BaNeP has two individual feature extraction networks. This section discusses the intuition, rationale, and structure of these feature extraction networks and sheds light on the final prediction generation networks. We present this discussion in a modular fashion instead of demonstrating it as a single combined model because we believe that the feature extraction networks has the potential to be applied in other suitable applications.

### A. STRUCTURAL FEATURE EXTRACTION

Irrespective of language, the character sequence of a word bears significant information about how it can be used in a sentence which plays a crucial role in Parts-of-Speech tagging. BaNeP has a dedicated model called SFENet (Structural Feature Extraction Network) that explores forward and backward relations among the characters in a word and intends to encode the relation as a feature vector.

Figure 1 shows the network architecture of SFENet. Characters of a word are passed through an embedding network to generate a sequence of embedding vectors $CE = [CE_0, CE_1, \ldots, CE_n]$ for the word. A Built-in PyTorch Embedding network has been used here. Each embedded character is then sequentially passed through an RNN cell to generate a hidden state for the next state. Although SFENet uses LSTM cells, GRU cells can also be used in this regard. Bidirectional RNN is preferred here to account for both prefix and suffix-related factors as for a highly both-end inflectional language like Bangla; these structural features are way more important in identifying Parts-of-Speech of a word rather than the root word/lemma. Cell states for the final time-steps in both forward and backward directions hold the structural information about the entire word. Equation 1 and 4 are used to calculate forward and backward cell state. In these equations terms $i_t$, $f_t$, $b_t$, and $W_C$ are used to indicate input gate output, forward gate output and cell state network weight at time $t$ respectively. Superscript $f$ and $b$ indicate forward and backward LSTM, respectively

$$CS_t^f = f_t^f \times CS_{t-1}^f + i_t^f \times \tanh(W_{Cf}[h_{t-1}^f, CE_t] + b_c^f) \tag{1}$$

where,

$$f_t^f = \sigma(W_f^f[h_{t-1}^f, CE_t] + b_f^f) \tag{2}$$

$$i_t^f = \sigma(W_i^f[h_{t-1}^f, CE_t] + b_i^f) \tag{3}$$

Similarly,

$$CS_t^b = f_t^b \times CS_{t-1}^b + i_t^b \times \tanh(W_{Cb}[h_{t-1}^b, CE_{n-t}] + b_c^b) \tag{4}$$

SFENet then concatenates forward and backward cell states at time $t = n$, which essentially holds structural characteristics of the word after training.

$$SF = CS_n^f(CE) + CS_n^b(CE) \tag{5}$$

SFENet does not intend to extract the structure of the entire sentence, and the processing of each word is independent. This creates the opportunity for parallel processing of each word in the sentence (in fact, all words of all sentences in a batch) if computational power is there. Thus character-level processing of SFENet does not create a time-dependent bottleneck for the entire poss-tagging task, especially for contextual feature extraction, which depends on word-level processing. The potential of SFENet stretches beyond Part-of-Speech tagging. Applications like named-entity recognition, lemmatization, and word sense disambiguation require structural feature extractions creating scope for SFENet.

### B. CONTEXTUAL FEATURE EXTRACTION

Almost every language has homonyms, making POS-tagging beyond the scope of the structural feature. Taking context into consideration is thus inevitable. Due to its highly inflectional nature, Bangla complicates the scenario by many folds. POS detection for many languages has become near perfect with simple pre-trained Word2Vec for semantic features with context sense disambiguation through bidirectional LSTM applied on the merged semantic and structural feature. Some languages require an extra CRF (Conditional Random Field) based procedure to properly grab the contextual impact of one word on another in a sentence. However, the inherent complex contextual nature of Bangla can hardly be grabbed with such shallow architecture for context encoding. To address this need for deeper contextual consideration, we present a two-phase weighted context generation procedure taking inspiration from an attention-based encoder-decoder model. In the first phase, a neural network generates unweighted context for the entire sentence, and a second neural network generates weighted context for each word using another neural network.

#### 1) CONTEXT ENCODER

Context encoder summarizes sentence sense as a vector which is then used by WCGNet (Weighted Context Generation Network) to prepare positional significance of each word considering its semantic and contextual factors. Semantic factors are first extracted from a pre-trained GloVe [44] word-to-vector model with 39 million Bangla words, each represented as a 100-dimensional vector. Figure 2 shows the detailed model architecture of how these semantic representations of all words in a sentence are used to prepare sentence context.

The generated vector from GloVe word-to-vector is then passed through the PyTorch embedding network to fine-tune word representation from the perspective of the dataset being used and also to account for new words that are not present in the pretrained word-to-vector file. If the sentence has $M + 1$ words (with padding token), then the output of the embedding network for the sentence is called $WE = [WE_0, WE_1, \ldots, WE_M]$. The context encoder then uses $WE$ as input to a bidirectional recurrent neural network. We have used LSTM as the recurrent unit. Forward and Backward LSTM generates output for each word considering its previous and later words. Forward and backward output calculation at time $t$ can represented as Equation 6 and 7 respectively.

$$O_t^f = \sigma(W_o^f[h_{t-1}^f, WE_t] + b_o^f) \tag{6}$$

$$O_t^b = \sigma(W_o^b[h_{t-1}^b, WE_{M-t}] + b_o^b) \tag{7}$$

In these equations, superscript $f$ is used to indicate forward LSTM, and superscript $b$ is used to indicate backward LSTM. Terms $W_o$ and $h_{t-1}$ are used to indicate the weight of the LSTM output gate and the hidden state of the previous timestamp.

Forward and backward LSTM output for a particular word embedding vector jointly encodes the relative positional significance of that word in the sentence considering the semantic influence of other words. Context Encoder merges these two output vectors of $word_k$ to generate contextual feature $CF_k$.

$$CF_k = O_k^f +\!\!+ O_{M-k}^b \tag{8}$$

If the sentence has $M + 1$ words, then for all words, corresponding contextual features ($CF_0$, $CF_1$, $\ldots$, $CF_{k-1}$, $CF_k$, $CF_{k+1}, \ldots, CF_{M-1}, CF_M$) are concatenated together to generate the contextual representation of the entire sentence. We call this extended vector Unweighted Context of the sentence. The last hidden state of both forward and backward LSTM are also recorded to be later used during weighted context generation.

#### 2) WEIGHTED CONTEXT GENERATOR

Although knowledge about each word considering sentence context is encoded in its contextual feature $CF$, while detecting Parts-of-Speech of that word, directly considering other words also should yield better prediction. Considering this intuition, we have proposed to use the entire unweighted context for prediction generation rather than using only the considered word's contextual feature. However, not all sentence words equally influence the word in consideration. So we here present an approach to generate each word's weighted context. Weighted context is a single vector (different for each word in the sentence) that includes its contextual feature and other words' weighted influence. To achieve this, we have constructed a Weighted Context Generation Network: WCGNet. WCGNet uses Bahdanau attention mechanism [45] for calculating other words' influence strength on the word in consideration for POS prediction. Figure 3 shows the model structure of WCGNet.

WCGNet is inspired by the LSTM-based decoder architecture of the sequence-to-sequence model. Unweighted context from context encoder network passes through Bahdanau attention layer along with WCGNet's previous hidden state. The context encoder's last hidden state is supplied for the first word of the sentence instead of WCGNet's last hidden state as it does not exist for the first word. Equation 9 and 10 show the attention weight calculation mechanism for the first and later words, respectively.

$$\alpha_0 = softmax(FC(\tanh(FC(UC)+FC(ELHS))), axis = 1) \tag{9}$$

$$\alpha_k = softmax(FC(\tanh(FC(UC)+FC(h_{k-1}^{WCGNet}))), axis=1) \tag{10}$$

where,

FC = Fully Connected Layer
UC = Unweighted Context
ELHS = Encoder's Last Hidden State
$h_{k-1}^{WCGNet}$ = WCGNet's previous hidden state

The output of the attention layer is a weight matrix which is then multiplied with unweighted context to generate weighted context $WC_k$ for $Word_k$. The second word in the sentence $WC_0$ (weighted context for the first word) and one-hot encoded Start token are combinedly passed through an LSTM cell. The hidden state of this LSTM cell is used to calculate the weight matrix for $WC_1$. From the third word onward, LSTM cell output from the immediate previous timestamp is used instead of a one-hot encoded start token for the current timestamp's LSTM cell input.

### C. GENERATING PREDICTION

For each word $Word_k$ in a sentence, SFENet generates structural feature $SF_k$, and WCGNet, with the help of Context Encoder, generates weighted context $WC_k$ that holds contextual and semantic information about the word. BaNeP's prediction generator network (Shown in Figure 4) uses $SF_k$ and $WC_k$ combinedly ($F_k = SF_k +\!\!+ WC_k$) to generate the parts-of-speech tag for each word.

From Figure 4, we can see that, analogous to the context encoder, the prediction generator also utilizes a bidirectional LSTM model overall $F_k s$ of a sentence. Although it may seem that BaNeP is repeatedly exploring contextual relation as context encoder already extracted that, this second over the sentence LSTM application is designed to grasp

contextual relation among words having similar structure. This makes BaNeP capable of generating correct Parts-of-Speech even when the maximum words of a sentence are entirely new but have a familiar prefix and suffix inflections on the original lemma. Furthermore, any low-resource language like Bangla will greatly benefit during sequence labeling tasks from this extracted sentence-level structural dependency as the probability of encountering unknown words is relatively high. The operation of the prediction generator network can be explained using Equation 11.

$$Pred_k = softmax(LNN(\sigma(W^f[h^f_{t-1}, F_k] + b^f) \mathbin{+\!\!+} \sigma(W^b[h^b_{M-k-1}, F_k] + b^b))) \quad (11)$$

where,
LNN : Linear Neural Network (possibly multi-layer)

From Equation 11, we can see that combined features $F = [F_0, F_1, \ldots, F_M]$ is fed toa BiLSTM network. For $Word_k$, forward LSTM's output at $t = k$ and backward LSTM's output at $t = M - k$ (backward LSTM generates output for $Word_k$ at time $t = M - k$) is concatenated together. This concatenated feature serves as input to a linear neural network (LNN). This linear neural network can be multi-layer; however, during the experimental phase, we have seen that adding multiple fully connected layers does not bring any noticeable performance improvement. Finally, a softmax function is applied to the output of LNN to generate the probability of each class.

## IV. DATASET ANALYSIS

We analyze the performance of our proposed model using the publicly available LDC dataset, which is designed based on the IL-POST framework [46]. The framework is implemented to address the POS tag task with morph-syntactic details of Indian Languages. The corpus provides two different levels of information for each lexical token: (a) lexical Category and Types and (b) set morphological attributes and their associated values in the context. For example,

মাটি\ NC.0.0.n.n থেকে\ PP.0.n বড়জোর\ JQ.n.n.nnm চার\ JQ.n.n.crd পাঁচ\ JQ.n.n.crd ফুট\ CCL.n উঁচু\ JJ.n.n

Here, each word is followed by a tag, and a backslash separates them. In between the tag and word, there is a blank space. After the blank space, there is at least one blank space before the next word. In the above example, the word মাটি has the POS tag NC means Noun Common, where N denotes the category noun and C denotes the type. 2-4 upper case letters represent the category and type. Table 1 describes the details abbreviation for Bangla Tag set. Attributes of a particular POS tag are denoted by either number or letter, placed after the POS tag, and separated by a dot. The order of the attributes is fixed and can not be changed. For example, consider the attributes of tag NC: Number, Case-marker, Definiteness, and Emphatic. The number can take value from the set: Singular (sg), Plural (pl), and Not-applicable (0). Case-marker can take values from the set: Accusative (acc), Genitive (gen), Locative (loc), Not-applicable (0); Definiteness can take values from yes(y) and no(n), and Emphatic can take values from yes(y) and no(n). So, in the above example, the word মাটি which is neither singular nor plural, so for the number attribute, it takes the value Not-applicable (0). Similarly, other attribute values are Not-applicable, non-definite, and non-emphatic; the complete tag should be

\NC.0.0.n.n

**TABLE 1. Bangla tag set.**

| Category | Type | Tag |
|---|---|---|
| Noun | Common | NC |
| | Proper | NP |
| | Verbal | NV |
| | Spatio-temporal | NST |
| Verb | Main | VM |
| | Auxiliary | VA |
| Pronoun | Pronominal | PPR |
| | Reflexive | PRF |
| | Reciprocal | PRC |
| | Relative | PRL |
| | Wh- | PWH |
| Nominal Modifier | Adjectives | JJ |
| | Quantifiers | JQ |
| Demonstratives | Absolutive | DAB |
| | Relative | DRL |
| | Wh- | DWH |
| Adverb | Manner | AMN |
| | Location | ALC |
| Participle | Relative (Adjectival) | LRL |
| | Verbal (Adverbial) | LV |
| Postposition | | PP |
| Particles | Coordinating | CCD |
| | Subordinating | CSB |
| | Classifier | CCL |
| | Interjection | CIN |
| | Others | CX |
| Punctuation | | PU |
| Residual | Foreign word | RDF |
| | Symbol | RDS |
| | Other | RDX |

The dataset contains 7168 sentences ( 102933 words) which are divided into two parts Bangla1 (3684 sentences, 51091 words) and Bangla 2 (3484 sentences, 51842 words). The authors collected data from Blogs, Multikulti (http://www.multikulti.org.uk), Wikipedia, and A portion of the CIIL corpus under the supervision of Multilingual Systems Group, Microsoft Research Labs India. In our proposed model, we did not use any information other than the POS tag to avoid dependency on hand-crafted features. BaNeP uses only words as input, and during the training phase, it takes true classes (POS tag of each word) in consideration for loss calculation and backpropagation.

## V. EXPERIMENTAL RESULTS

BaNeP has been trained with a wide variety of hyper-parameter combinations to trace down the optimal combination which works best for Bangla POS tagging. Performance of BaNeP with optimal hyper-parameter values is compared with state-of-the-art POS tagging approaches for Bangla BiLSTM-CRF and sequence labeling task ASRNN.

**TABLE 2. Training, validation and test split size.**

| Set | Sentence | Words/tags |
|---|---|---|
| Training | 6459 | 933577 |
| Validation | 381 | 4519 |
| Test | 328 | 4837 |
| Total | 7168 | 102933 |

This section focuses on hyper-parameter tuning for BaNeP and comparative performance analysis of state-of-the-art approaches.

### A. EXPERIMENTAL SETUP

We have trained and evaluated on Nvidia's CUDA-CUDNN environment. Detail configuration of the training and evaluation environment is given below:

CPU: AMD Ryzen 9 5900x
RAM: 128 GB
GPU: NVIDIA RTX 3090
VRAM: 24 GB
OS: Ubuntu 20.04
Language: Python 3.9
Framework: PyTorch 1.8

We have split the entire $LDC2010T16$ dataset into three splits: training, validation, and test set. Sentences were randomly chosen from the entire dataset with a probability of 0.90 for the training-set, 0.05 for the validation-set, and 0.05 for the test-set. Table 2 shows each split's sentence and word count.

### B. EVALUATION METRICS

We calculated the model's accuracy during the validation and testing phase by taking the percentage of correctly identified POS tags. However, the dataset is heavily imbalanced. Out of thirty-two different POS tags, more than 60% of the words fall under the five categories (Common Noun: 30%, Punctuation: 14%, Main verb: 11%, Adjective: 8% and Proper Noun: 7%). To properly report the performance of BaNeP, we have also calculated the precision and recall of each class. Equation 12 and 13 have been used to calculate precision and recall of each class.

$$Precision_i = \forall_{w \in D} \frac{|\{w : PC(w) = TC(w) = i\}|}{|\{w : PC(w) = i\}|} \quad (12)$$

$$Recall_i = \forall_{w \in D} \frac{|\{w : PC(w) = TC(w) = i\}|}{|\{w : TC(w) = i\}|} \quad (13)$$

where,
D = Dataset in consideration
w = word
PC = Predicted Class
TC = True Class

precision and recall both help examine the model's performance on a per word basis, therefore eliminating bias that may occur in case of accuracy measure for larger classes. However, for parts-of-speech tagging, the precision measure is more critical than recall as it indicates how much we can rely on a predicted tag's correctness to be further used as a feature for dependent NLP tasks.

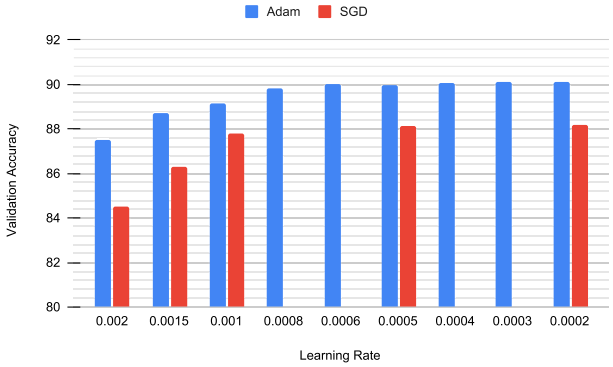


**FIGURE 5. Learning rate tuning.**

### C. HYPER-PARAMETER TUNING

This section discusses the optimal hyper-parameter combination for BaNeP based on experimental results. The hyper-parameters that we have considered for tuning are:

- Optimizer
- Learning Rate
- Context Encoder Hidden size
- Prediction Generator LNN layer count

For all these configurations, we have kept batch size fixed to 64 sentences. SFENet, while extracting structural features, feeds all words of all sentences of a batch simultaneously for parallel calculation. Thus increasing batch size higher than 64 sentences for BaNeP makes the batch size of SFENet challenging to handle as there are some unusually large sentences in the dataset.

#### 1) OPTIMIZER AND LEARNING RATE

We have recorded the accuracy of the BaNeP model for two optimizers: i) Adam and ii) SGD for various learning rates. We started with a higher initial learning rate of 0.002 with an assumption from prior knowledge that LSTM-based models perform well with Adam optimizer compared to SGD, and the optimal learning rate lies near 0.0003. So we started our learning rate tracing at 0.002 and gradually decreased it until 0.0002 to find out which value works for BaNeP best. Figure 5 shows validation accuracy of BaNeP for various learning rates with both Adam and SGD optimizer.

Learning rates we have tried are: [0.002, 0.0015, 0.001, 0.0008, 0.00006, 0.0005, 0.0004, 0.0003, 0.0002] for Adam optimizer. For SGD, we have tried learning rates selectively from the above set, as seen in the figure. Not to our surprise, the initial learning rate did not affect accuracy much. Adam optimizer adjusts learning rate automatically, and so, except for some initial possibility of overshooting from global minima, Adam optimizer is sturdy. The result of validation accuracy also demonstrates that. Validation accuracy varied from 87.51% to 90.10%. The highest validation accuracy was found at learning set to 0.0002, which was not much higher


**FIGURE 6.** Context encoder hidden size.

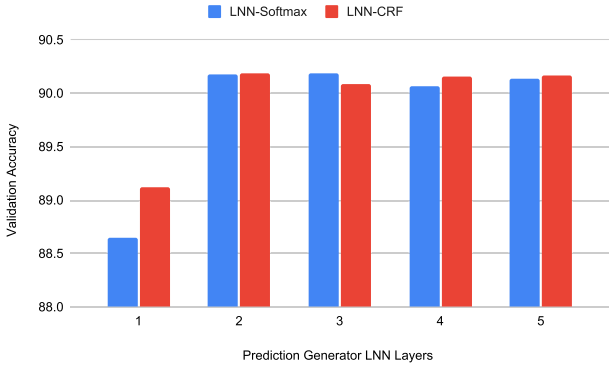


**FIGURE 7.** Prediction generator LNN hidden layer tuning.

than we found for a learning rate of 0.0003, which is 90.09%. Thus we decided to go with a learning rate of 0.0003 as our optimal learning rate with the Adam optimizer.

The effect of the learning rate for the SGD optimizer should have been significant, but we adopted the policy of adjusting the learning rate based on train-loss improvement magnitude. Consequently, the effect of the initial choice for learning was not acute for SGD either. However, in these settings, SGD failed to outperform Adam achieving a maximum of 88.19% accuracy for a learning rate of 0.0002. So for later hyper-parameter tuning, we have used the Adam optimizer with a learning rate set to 0.0003.

### 2) HIDDEN SIZE

Keeping batch size and learning rate fixed to 64 and 0.0003 respectively with Adam optimizer, we conducted training changing Context Encoder's LSTM hidden size. Each word in the packed padded sequence derived from a sentence gets a vector for the forward pass and another for the backward pass. Thus Context Encoder's LSTM hidden size controls how much detail about a word's contextual information will be stored for further process in the WCGNet and ultimately controls the WCGNet's and Prediction Generator's hidden size (Prediction Generators hidden size also depends on SFENet's LSTM hidden size which we kept fixed to 100 during the experiment). Figure 6 shows the effect of hidden size changing as a line graph.

From the figure, we can see that hidden sizes around 200 to 300 performed consistently better compared to other configurations achieving an accuracy of 90.09%, 90.18%, and 90.17%, respectively. A hidden size smaller than 200 is unsuitable, as can be easily inferred from the line trend of Figure 6. Hidden size higher than 300 shows fluctuations in terms of validation accuracy improvement. Hidden size 600 also performed fair (89.73% accuracy), but even with this increased computation, it failed to outperform the hidden size of range [200, 300]. A larger hidden size indicates more detailed information; still, from the figure, we can see that increasing the hidden size does not improve the performance. In fact, the opposite is true. A realistic explanation is that after hidden size 300, overfitting on the training set started to settle in, adversely affecting validation performance. Considering the consistent curve and comparatively higher validation accuracy in the range [200, 300], we finalized 256 as our optimal value for Context Encoder's hidden size.

### 3) LNN HIDDEN LAYERS

In the Prediction Generator network (Figure 4), we can see a block for a linear neural network marked as LNN just before the softmax layer. We have experimented by changing the number of layers in this linear neural network to determine how that affects the performance. Besides, we have also tried a CRF function instead of directly using softmax for class probability calculation. CRF is heavily used in sequence labeling tasks, especially in part-of-speech tagging. We have introduced CRF after LNN to check whether it brings any improvement to our proposed model. Figure 7 shows LNN-Softmax and LNN-CRF performance different single layer LNN to five-layer LNN.

For Single layer LNN, CRF outperformed plain softmax. Single layer LNN with softmax achieved 88.65% accuracy, whereas, with CRF, the validation accuracy was 89.12%. However, for two-layer LNN, softmax and CRF performances were similar: 90.18% for softmax and 90.19% for CRF. When we increased layer count to three, softmax performance remained the same, but CRF performance fell to 90.09%. For CRF and softmax approaches, we have observed that increasing layers more than three does not increase accuracy but falls prey to overfitting. CRF solves this overfitting issue a bit; that is why CRF performance is slightly better than using mere softmax.

The primary role of CRF here is to generate a tag considering the context of the entire sentence. BaNeP already has a two-phase detailed BiLSTM-Attention-based network (Context Encoder and WCG) for contextual consideration. On top of that, Prediction Generator also grasps inter-word structural dependency. These detailed subnetworks combined captured what CRF aims to and possibly more. Thus, augmenting BaNeP with CRF did not significantly improve the model's accuracy. CRF's impact on the network was highest when single-layer LNN was tested. However, for two and three-layer LNN, the impact of CRF is negligible. So we

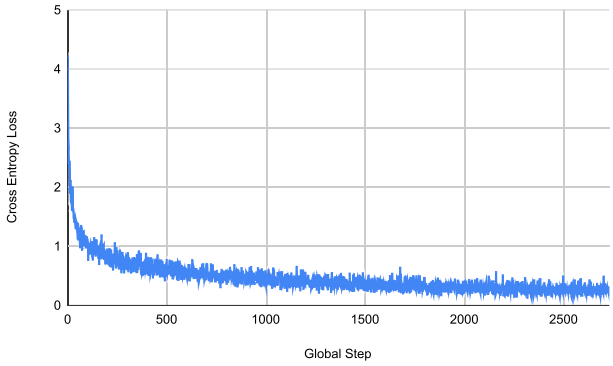


**FIGURE 8.** Train loss.

decided to exclude CRF from the proposed model to avoid unnecessary computational complexity.

So optimal hyper-parameter combination for BaNeP with batch size 64 is:

i) Optimizer : Adam
ii) Learning rate : 0.0003
iii) Context Encoder Hidden size : 256
iv) LNN Hidden Layers : 2

### D. MODEL CONVERGENCE

We have trained the model with the optimal hyper-parameter setting found in Section V-C. Figure 8 shows how cross-entropy loss decreases with backpropagation step for each batch. Our training dataset has 6459 sentences distributed in 101 batches. In the graph, each epoch is shown as 101 global steps. The initial cross-entropy loss on the validation split was 4.23. In the beginning, cross-entropy loss for the train split reduced quickly, from 4.23 to 0.79 in just four epochs (404 global steps).

Speed of convergence took a slow pace afterward and required 20 more epochs to reach 0.13 cross-entropy loss, for which we got the highest validation accuracy (90.18%). Validation accuracy after each train epoch is shown in Figure 9.

The figure shows that similar to cross-entropy loss for training data, validation accuracy reached from 20.03% to 79.72% in just four epochs. After that, accuracy slowly but steadily improved till the 16th epoch (89.77%). After that, we can see minor fluctuations in the achieved accuracy level. This fluctuation occasionally continued, achieving higher accuracy than all previous epochs. We saved model parameters whenever they were higher than the previous validation accuracy. After epoch 22, which gave 90.18% validation accuracy, there was no improvement in the best validation accuracy for the next five epochs. We stopped further training to avoid overfitting the train data and used the saved model of epoch 22 for testing. Table 3 shows the confusion matrix for 328 test sentences containing 4837 tokens to tag. From the confusion matrix, we can see that 1440 tokens have tags NC as a true class, among which 1348 tokens have been classified accurately (recall: 93.61%). However, BaNeP falsely labeled 175 tokens as NC making its precision 88.52%. If we go through the confusion matrix carefully, we can see that among these 175 false positives, 74 entries belong to the NP class. Similarly, among 102 false negative tokens for the NC class, 51 tokens are labeled as NP. The point to notice here is that NP and NP belong to the same category (Noun). A similar pattern between VM (main verb) and VA (auxiliary verb) classes can be found. This indicates that on the category level, as mentioned in Table 1, the accuracy of the proposed BaNeP model is much higher.

### E. COMPARATIVE PERFORMANCE ANALYSIS

This section illustrates how BaNeP performs compared to other existing Bangla POS tagging approaches. We have considered two of the most notable sequence labeling approaches for comparison. One is BiLSTM-CRF (first proposed by Huang *et al.* [18] for sequence labeling and later adopted by Alam *et al.* [40] for Bangla POS-tagging). Another is the recent attention-based sequence labeling approach ASRNN [25] proposed by Lin *et al.* We have also presented a comparison with our CRF variant, referred to as BaNeP-CRF.

#### 1) ACCURACY

BiLSTM-CRF and ASRNN are very robust models that perform decently for sequence labeling tasks. However, the diverse character-level structure makes sentence-level contextual sense disambiguation more important for Bangla, which calls for an elaborate contextual feature to predict Parts-of-Speech accurately. BaNeP is designed to do exactly so. Both BiLSTM-CRF and ASRNN falls behind in such cases for Bangla. Figure 10 and Table 4 shows the test set accuracy comparison among these approaches mentioned above.

As expected, all of these models perform somewhat similarly in terms of accuracy. ASRNN showed marginally better accuracy than BiLSTM-CRF. However, both BaNeP and BaNeP-CRF perform slightly better than ASRNN and BiLSTM-CRF due to handling complex cases requiring deeper contextual consideration. In fact, the reason behind ASRNN outperforming BiLSTM-CRF is also the same. ASRNN has a word-level attention layer that gives it an edge over BiLSTM-CRF for Bangla-POS tagging.

#### 2) PRECISION AND RECALL

As mentioned in Section IV, *LDC*2010*T*16 is a highly unbalanced dataset. So, we cannot rely merely on the accuracy measure. To demonstrate how our proposed model performs for each class compared to ASRNN, we have shown class-wise precision and recall measures in Figure 11 and 12 respectively.

From Figure11 we can see that the precision level for the classes fluctuates mainly between 75 to 95% for both BaNeP and ASRNN. Although the performance level is similar, BaNeP consistently demonstrated slightly higher performance than ASRNN except for proper nouns (NP class). BaNeP predicted 319 tokens to be proper nouns, among which 256 tokens were correctly predicted; the rest are false

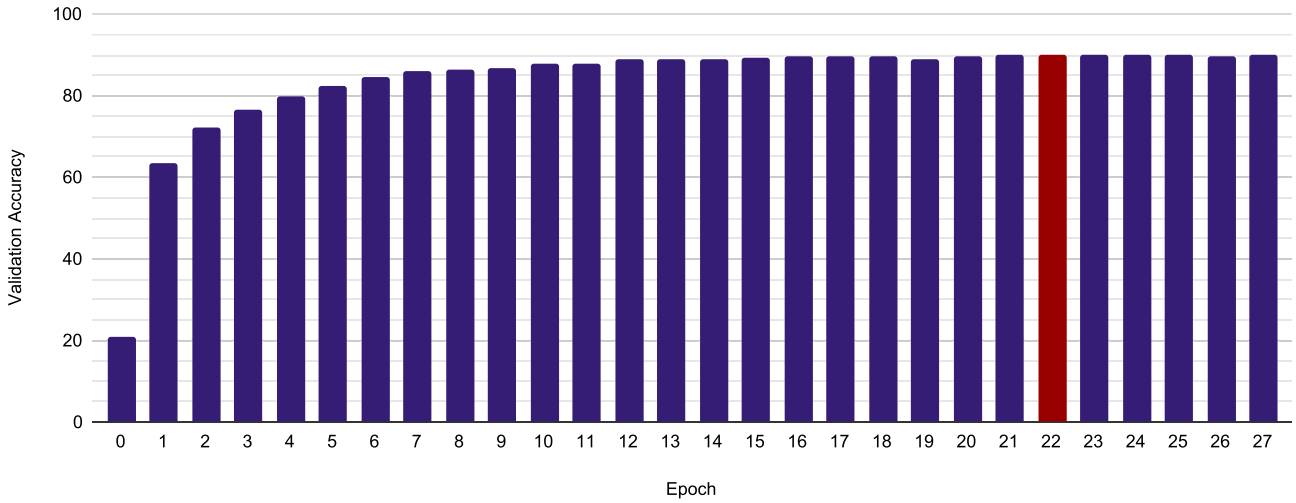


**FIGURE 9.** Validation accuracy.

**TABLE 3.** BaNeP confusion matrix. (Test split, Row header is true class and column header is predicted class.)

| NE | NC | PU | VM | JJ | NP | JQ | PPR | PP | CCD | NV | CX | VAUX | DAB | CSB | NST | ALC | AMN | RDS | RDF | LC | VA | PRL | PWH | RDX | CCL | DRL | PRF | CIN | LV | DWH | PRC | CSD | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NC | 1348 | 0 | 5 | 7 | 51 | 4 | 0 | 3 | 0 | 8 | 3 | 0 | 0 | 4 | 1 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1440 |
| PU | 0 | 645 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 647 |
| VM | 10 | 0 | 474 | 0 | 0 | 0 | 0 | 6 | 0 | 1 | 0 | 14 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 511 |
| JJ | 29 | 0 | 2 | 322 | 10 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 370 |
| NP | 74 | 0 | 1 | 6 | 256 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 339 |
| JQ | 15 | 0 | 0 | 9 | 1 | 177 | 1 | 0 | 2 | 0 | 1 | 0 | 2 | 1 | 0 | 0 | 2 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 213 |
| PPR | 7 | 0 | 0 | 0 | 0 | 3 | 175 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 189 |
| PP | 4 | 1 | 5 | 0 | 0 | 1 | 1 | 114 | 0 | 0 | 3 | 0 | 0 | 1 | 7 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 139 |
| CCD | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 88 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 96 |
| NV | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 84 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 90 |
| CX | 2 | 0 | 0 | 1 | 0 | 2 | 0 | 1 | 2 | 0 | 65 | 0 | 0 | 8 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 88 |
| VAUX | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 62 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 71 |
| DAB | 0 | 0 | 0 | 0 | 0 | 4 | 2 | 0 | 1 | 0 | 0 | 0 | 69 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 76 |
| CSB | 1 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 58 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 64 |
| NST | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 57 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 66 |
| ALC | 5 | 0 | 1 | 1 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 51 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 65 |
| AMN | 6 | 0 | 1 | 2 | 0 | 3 | 0 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 4 | 64 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 84 |
| RDS | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 58 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 61 |
| RDF | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 56 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 57 |
| LC | 1 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 |
| VA | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 29 |
| PRL | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 26 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 29 |
| PWH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16 |
| RDX | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 26 |
| CCL | 2 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 13 |
| DRL | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 0 | 13 |
| PRF | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 12 |
| CIN | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 3 |
| LV | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 7 |
| DWH | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| PRC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| CSD | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

positives. On the other hand, ASRNN predicted 303 tokens to be NP, where the true positive count is 249. Considering the graph carefully, it becomes evident that the gap between BaNeP and ASRNN is comparatively higher for JQ, PP, VAUX, AMN, RDF, LC, VA, RDX, DWH, and CSD classes. All of them are comparatively small classes, proving the reliability of BaNeP to tag or identify smaller classes compared to ASRNN.

In the case of recall, we can see a similar trend, although the average recall level for both BaNeP and ASRNN is a bit lower for all classes except NC class; the largest one among the thirty-two tags present in the dataset. Recall values of class NC are 93.61% and 92.01% for BaNeP and ASRNN, respectively. Analogous to precision comparison, the recall value of only one class (JQ) is higher for ASRNN than BaNeP. There we 213 tokens belonging to JQ class in the test

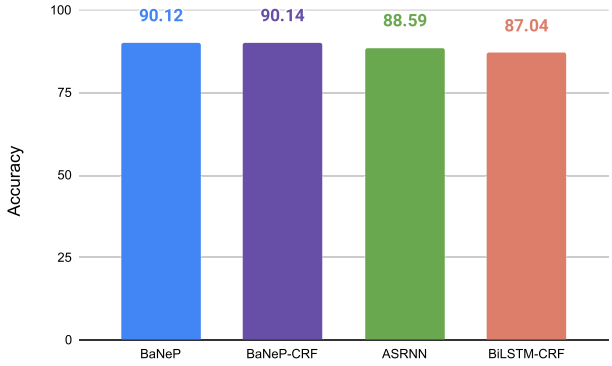


**FIGURE 10.** BaNeP vs BaNeP-CRF vs ASRNN vs BiLSTM-CRF accuracy comparison.

dataset, among which BaNeP was able to identify 177 tokens compared to 187 tokens using ASRNN correctly. The superiority of BaNeP is more evident again for smaller classes like ALC, AMN, VA, RDX, LV, and CSD, where BaNeP's recall is higher than ASRNN by 4.61%, 7.14%, 20.69%, 26.92%, 28.57%, and 100% respectively.

**TABLE 4.** Accuracy comparison.

| | BaNeP | BaNeP-CRF | ASRNN | BiLSTM-CRF |
|---|---|---|---|---|
| Acc % | 90.12 | 90.14 | 88.59 | 87.04 |

From the comparative performance analysis presented in this section, it can easily be said that both BaNeP and ASRNN have shown competitive performance for Bangla POS-tagging application. However, BaNeP managed to be consistent even for smaller classes and cases where contextual information is more important. ASRNN emphasized both character-level structure and word-level context equally. On the other BaNeP emphasized more word-level context, which worked in its favor as Bangla words show low structural patterns.

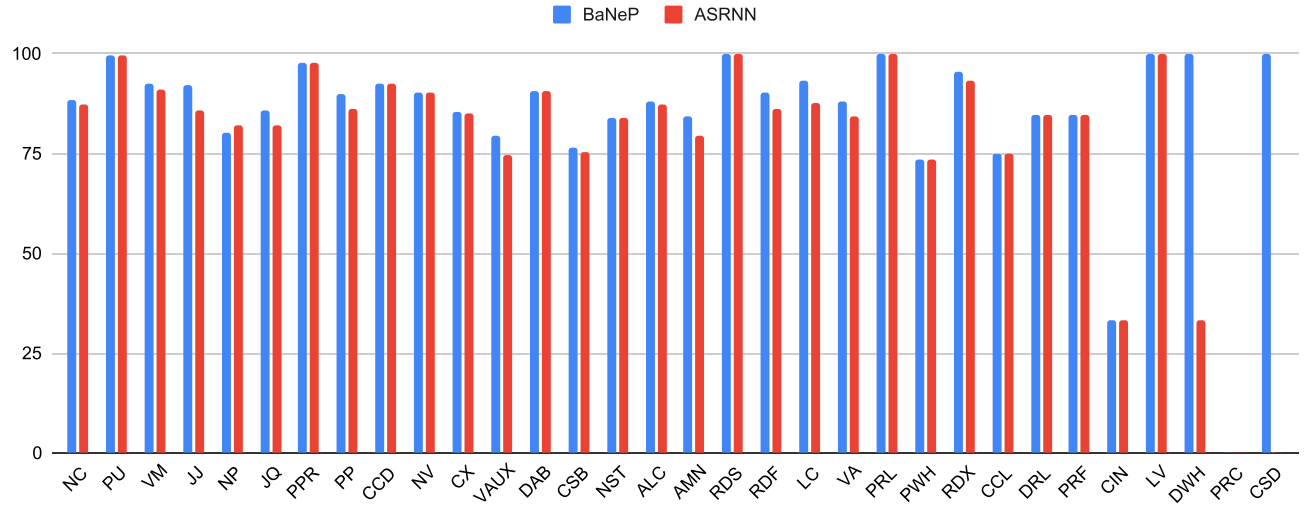


**FIGURE 11.** BaNeP vs ASRNN precision comparison.

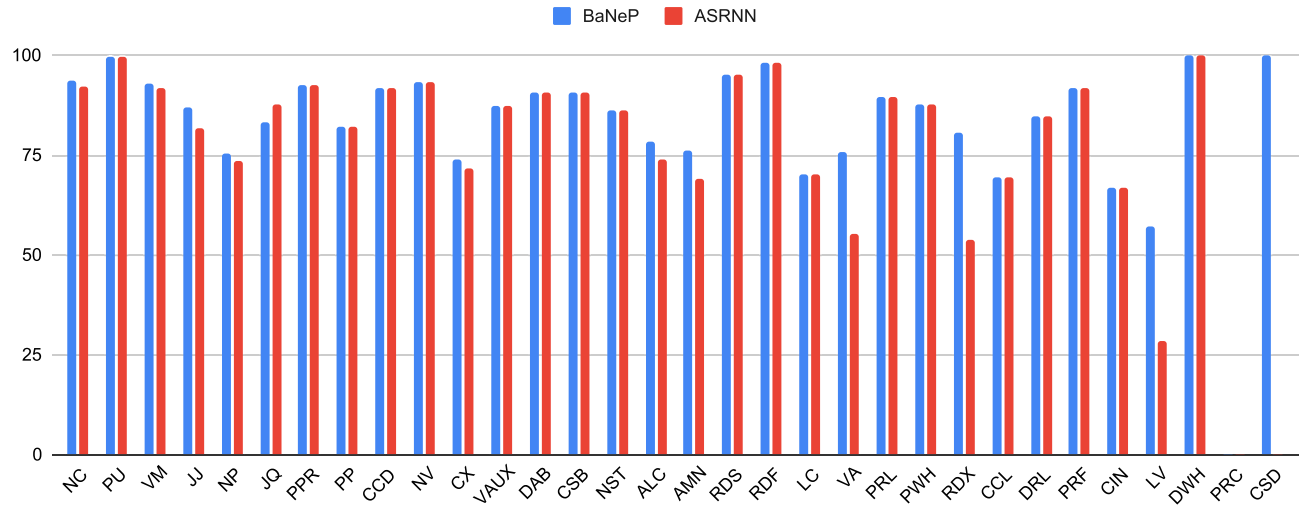


**FIGURE 12.** BaNeP vs ASRNN recall comparison.

## VI. CONCLUSION

This article proposes a novel multi-phase recurrent neural network for Bangla Parts-of-Speech tagging named BaNeP. Model architecture analysis and experimental evaluation demonstrate that BaNeP is highly capable of extracting detailed contextual information that influences a word's Parts-of-Speech. Bangla is a mixed language, originating and being influenced by several other languages, and lacks grammatically sound inflection, making the word's structural features less significant than contextual features. This gave BaNeP an edge for Bangla POS-tagging over existing state-of-the-art algorithms in terms of accuracy. As accuracy alone cannot justify a model's efficiency for cases like POS-tagging where the dataset is bound to be imbalanced, we have also examined precision and recall, which unsurprisingly were higher than existing approaches. Languages like Assamese, Halbic, and Odia, originated from Magadhi Prakrit, show similar traits. So, BaNeP is expected to show similar improvement in POS-tagging.

## ACKNOWLEDGMENT

The authors are thankful to the host institute, University of Dhaka, for APC.

## REFERENCES

[1] P. Awasthi, D. Rao, and B. Ravindran, "Part of speech tagging and chunking with HMM and CRF," in *Proceedings of NLP Association of India (NLPAI) Machine Learning Contest 2006*. India: NLP Association of India, 2006.

[2] S. Warjri, P. Pakray, S. A. Lyngdoh, and A. K. Maji, "Part-of-speech (POS) tagging using conditional random field (CRF) model for Khasi corpora," *Int. J. Speech Technol.*, vol. 24, no. 4, pp. 853–864, Dec. 2021.

[3] A. R. Martinez, "Part-of-speech tagging," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 4, no. 1, pp. 107–113, Jan. 2012.

[4] X. Zheng, H. Chen, and T. Xu, "Deep learning for Chinese word segmentation and pos tagging," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2013, pp. 647–657.

[5] S. Kumar, M. A. Kumar, and K. P. Soman, "Deep learning based part-of-speech tagging for Malayalam Twitter data (special issue: Deep learning techniques for natural language processing)," *J. Intell. Syst.*, vol. 28, no. 3, pp. 423–435, Jul. 2019.

[6] M. E. Basiri, S. Nemati, M. Abdar, E. Cambria, and U. R. Acharya, "ABCDM: An attention-based bidirectional CNN-RNN deep model for sentiment analysis," *Future Gener. Comput. Syst.*, vol. 115, pp. 279–294, Feb. 2021.

[7] J. Mia, "Effectiveness analysis of different pos tagging techniques for Bangla language," in *Smart Systems: Innovations in Computing*. Cham, Switzerland: Springer, 2022, pp. 121–134.

[8] A. Ekbal, R. Haque, and S. Bandyopadhyay, "Maximum entropy based Bengali part of speech tagging," in *Advances in Natural Language Processing and Applications, Research in Computing Science (RCS) Journal*, vol. 33, A. Gelbukh, Ed. Mexico: National Polytechnic Institute, 2008, pp. 67–78.

[9] A. H. Patoary, M. J. Bin Kibria, and A. Kaium, "Implementation of automated Bengali parts of speech tagger: An approach using deep learning algorithm," in *Proc. IEEE Region 10 Symp. (TENSYMP)*, Jun. 2020, pp. 308–311.

[10] A. C. Doyle, *The Adventures Sherlock Holmes*. Hertfordshire, U.K.: Wordsworth Editions, 1992.

[11] W. J. Bate, "The burden of the past and the english poet," in *The Burden of the Past and the English Poet*. Cambridge, MA, USA: Harvard Univ. Press, 2013.

[12] A. Faquire, "Language situation in Bangladesh," *Dhaka Univ. Stud.*, vol. 67, no. 2, pp. 4–5, 2010.

[13] P. Bhattacharya and A. Bhattacharya, "Evolution of the modern phase of written Bangla: A statistical study," in *Mining Intelligence and Knowledge Exploration*. Cham, Switzerland: Springer, 2013, pp. 799–805.

[14] X. Ma and F. Xia, "Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2014, pp. 1337–1348.

[15] F. Rodrigues, F. Pereira, and B. Ribeiro, "Sequence labeling with multiple annotators," *Mach. Learn.*, vol. 95, no. 2, pp. 165–181, May 2014.

[16] C. D. Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1818–1826.

[17] M. P. Marcus, B. Santorini, M. A. Marcinkiewicz, and A. Taylor, *Treebank-3*, vol. 14. Philadelphia, PA, USA: Linguistic Data Consortium, 1999.

[18] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," 2015, *arXiv:1508.01991*.

[19] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," 2016, *arXiv:1603.01354*.

[20] E. F. Sang and F. D. Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," 2003, *arXiv:cs/0306050*.

[21] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proc. 27th Int. Conf. Comput. linguistics*, 2018, pp. 1638–1649.

[22] H. Tsai, J. Riesa, M. Johnson, N. Arivazhagan, X. Li, and A. Archer, "Small and practical BERT models for sequence labeling," 2019, *arXiv:1909.00100*.

[23] B. Bohnet, R. McDonald, G. Simoes, D. Andor, E. Pitler, and J. Maynez, "Morphosyntactic tagging with a meta-BiLSTM model over context sensitive token encodings," 2018, *arXiv:1805.08237*.

[24] A. Smith, B. Bohnet, M. de Lhoneux, J. Nivre, Y. Shao, and S. Stymne, "82 treebanks, 34 models: Universal dependency parsing with multi-treebank models," 2018, *arXiv:1809.02237*.

[25] J. C.-W. Lin, Y. Shao, Y. Djenouri, and U. Yun, "ASRNN: A recurrent neural network with an attention model for sequence labeling," *Knowl.-Based Syst.*, vol. 212, Jan. 2021, Art. no. 106548.

[26] D. Kumawat and V. Jain, "POS tagging approaches: A comparison," *Int. J. Comput. Appl.*, vol. 118, no. 6, pp. 32–38, May 2015.

[27] K. S. M. Anbananthen, J. K. Krishnan, M. S. Sayeed, and P. Muniapan, "Comparison of stochastic and rule-based POS tagging on Malay online text," *Amer. J. Appl. Sci.*, vol. 14, no. 9, pp. 843–851, Sep. 2017.

[28] F. Pisceldo, R. Manurung, and M. Adriani, "Probabilistic—Part of speech tagging for Bahasa Indonesia," in *Proc. 3rd Int. MALINDO Workshop*, 2009, pp. 1–6.

[29] M. V. Nunsanga, P. Pakray, M. Lalngaihtuaha, and L. L. K. Singh, "Stochastic based part of speech tagging in Mizo language: Unigram and Bigram hidden Markov model," in *Edge Analytics*. Cham, Switzerland: Springer, 2022, pp. 711–722.

[30] J. Shafi, H. R. Iqbal, R. M. A. Nawab, and P. Rayson, "UNLT: Urdu natural language toolkit," *Natural Lang. Eng.*, pp. 1–36, Jan. 2022.

[31] K. R. Singha, B. S. Purkayastha, and K. D. Singha, "Part of speech tagging in Manipuri with hidden Markov model," *Int. J. Comput. Sci. Issues*, vol. 9, no. 6, p. 146, 2012.

[32] T. Dalai, T. K. Mishra, and P. K Sa, "Part-of-speech tagging of Odia language using statistical and deep learning-based approaches," 2022, *arXiv:2207.03256*.

[33] 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, pp. 1–25, Dec. 2022.

[34] B. R. Das and S. Patnaik, "A novel approach for Odia—Part of speech tagging using artificial neural network," in *Proc. Int. Conf. Frontiers Intell. Comput., Theory Appl. (FICTA)*. Cham, Switzerland: Springer, 2014, pp. 147–154.

[35] K. K. Akhil, R. Rajimol, and V. S. Anoop, "Parts-of-speech tagging for Malayalam using deep learning techniques," *Int. J. Inf. Technol.*, vol. 12, no. 3, pp. 741–748, Sep. 2020.

[36] M. S. A. Chowdhury, N. M. Uddin, M. Imran, M. M. Hassan, and M. E. Haque, "Parts of speech tagging of Bangla sentence," in *Proc. 7th Int. Conf. Comput. Inf. Technol. (ICCIT)*, 2004, pp. 440–444.

[37] S. Mukherjee and S. K. D. Mandal, "Bengali parts-of-speech tagging using global linear model," in *Proc. Annu. IEEE India Conf. (INDICON)*, Dec. 2013, pp. 1–4.

[38] A. Ekbal, R. Haque, and S. Bandyopadhyay, "Bengali part of speech tagging using conditional random field," in *Proc. 7th Int. Symp. Natural Lang. Process. (SNLP)*, 2007, pp. 131–136.

[39] S. Dandapat, S. Sarkar, and A. Basu, "Automatic—Part-of-speech tagging for Bengali: An approach for morphologically rich languages in a poor resource scenario," in *Proc. 45th Annu. Meeting ACL Interact. Poster Demonstration Sessions (ACL)*, 2007, pp. 221–224.
[40] F. Alam, S. A. Chowdhury, and S. R. H. Noori, "Bidirectional LSTMs—CRFs networks for Bangla POS tagging," in *Proc. 19th Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2016, pp. 377–382.
[41] M. F. Kabir, K. Abdullah-Al-Mamun, and M. N. Huda, "Deep learning based parts of speech tagger for Bengali," in *Proc. 5th Int. Conf. Informat., Electron. Vis. (ICIEV)*, May 2016, pp. 26–29.
[42] M. N. Uddin, M. S. Islam, M. A. Khan, and M.-E. Jannat, "A neural network approach for Bangla POS tagger," in *Proc. Int. Conf. Bangla Speech Lang. Process. (ICBSLP)*, Sep. 2018, pp. 1–4.
[43] M. K. Roy, P. K. Paul, S. R. H. Noori, and S. M. H. Mahmud, "Suffix based automated parts of speech tagging for Bangla language," in *Proc. Int. Conf. Electr., Comput. Commun. Eng. (ECCE)*, Feb. 2019, pp. 1–5.
[44] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.
[45] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent., (ICLR)*. San Diego, CA, USA, May 2015, pp. 1–15.
[46] K. Bali, M. Choudhury, P. Biswas, G. N. Jha, N. Choudhary, and M. Sharma, *Indian Language Part-of-Speech Tagset: Hindi*. Philadelphia, PA, USA: Linguistic Data Consortium, 2010.

**JESAN AHAMMED OVI** received the B.Sc. (Hons.) and M.S. degrees in computer science and engineerig from the University of Dhaka, Bangladesh, in 2015 and 2017, respectively. He has worked for United International University, Bangladesh, as a Faculty Member. He is currently working as a Faculty Member with the Department of Computer Science and Engineering, East West University, Bangladesh. He is also working as a Research Assistant with the Department of Computer Science and Engineering, University of Dhaka. His current research interests include data mining and deep learning (natural language processing).

**MD. ASHRAFUL ISLAM** received the B.Sc. (Hons.) and M.S. degrees in computer science and engineering from the University of Dhaka, Bangladesh, in 2015 and 2017, respectively. He has worked for United International University, Bangladesh, and East West University, Bangladesh, as a Faculty Member. He is currently working as an Assistant Professor with the Department of Computer Science and Engineering, University of Dhaka. His current research interests include data mining, natural language processing, computer vision, and incremental learning.

**MD. REZAUL KARIM** received the B.Sc. degree (Hons.) in applied physics and electronics and the M.Sc. degree in computer science from the University of Dhaka, Bangladesh, the M.Tech. degree in computer and information technology from the Indian Institute of Technology, Kharagpur, India, in January 1998, and the Ph.D. degree from the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Bangladesh, in 2008. He is currently a Professor with the Department of Computer Science and Engineering, University of Dhaka. His research interests include graph drawing, information visualization, algorithm, and natural language processing.