Part-of-Speech Tagging of Odia Language Using Statistical and Deep Learning Based Approaches

TUSARKANTA DALAI, TAPAS KUMAR MISHRA, and PANKAJ K. SA, NIT Rourkela

Automatic part-of-speech (POS) tagging is a preprocessing step of many natural language processing tasks, such as named entity recognition, speech processing, information extraction, word sense disambiguation, and machine translation. It has already gained promising results in English and European languages. However, in Indian languages, particularly in the Odia language, it is not yet well explored because of the lack of supporting tools, resources, and morphological richness of the language. Unfortunately, we were unable to locate an open source POS tagger for the Odia language, and only a handful of attempts have been made to develop POS taggers for the Odia language. The main contribution of this research work is to present statistical approaches such as the maximum entropy Markov model and conditional random field (CRF), as well as deep learning based approaches, including the convolutional neural network (CNN) and bidirectional long short-term memory (Bi-LSTM) to develop the Odia POS tagger. A publicly accessible corpus annotated with the Bureau of Indian Standards (BIS) tagset is used in our work. However, most of the languages around the globe have used the dataset annotated with the Universal Dependencies (UD) tagset. Hence, to maintain uniformity, the Odia dataset should use the same tagset. Thus, following the BIS and UD guidelines, we constructed a mapping from the BIS tagset to the UD tagset. The maximum entropy Markov model, CRF, Bi-LSTM, and CNN models are trained using the Indian Languages Corpora Initiative corpus with the BIS and UD tagsets. We have experimented with various feature sets as input to the statistical models to prepare a baseline system and observed the impact of constructed feature sets. The deep learning based model includes the Bi-LSTM network, the CNN network, the CRF layer, character sequence information, and a pre-trained word vector. Seven different combinations of neural sequence labeling models are implemented, and their performance measures are investigated. It has been observed that the Bi-LSTM model with the character sequence feature and pre-trained word vector achieved a result with 94.58% accuracy.

CCS Concepts: • Computing methodologies → Machine learning; • Information systems → Information retrieval;

Additional Key Words and Phrases: Part of speech (POS), conditional random field (CRF), deep learning, word embedding

ACM Reference format:
Tusarkanta Dalai, Tapas Kumar Mishra, and Pankaj K. Sa. 2023. Part-of-Speech Tagging of Odia Language Using Statistical and Deep Learning Based Approaches. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 22, 6, Article 167 (June 2023), 24 pages. https://doi.org/10.1145/3588900
1 INTRODUCTION

Part-of-Speech (POS) tagging is one of the sequence labeling tasks that involves assigning a grammatical category label to each word based on linguistic and contextual information [6]. The tag or label of a word provides information about the word and its surrounding lexical categories. In general, a POS tagger will classify the sentence into several subcategories based on its parts of speech, including noun, pronoun, adjective, verb, adverb, and so on. POS tags are helpful because they provide linguistic information about how words can be employed in a phrase, sentence, or document. In the field of language processing, POS tagging is an essential preprocessing step for many Natural Language Processing (NLP) frameworks. These NLP frameworks include speech recognition, sentiment analysis, Named Entity Recognition (NER), question answering, word sense disambiguation, chunking, and so on [24]. Determining the grammatical class of a language is challenging because it varies depending on the context in which it is used. Therefore, it is difficult to assign tags to each word of a sentence when some words have more than one grammatical POS label. In English, other European languages, and the majority of South Asian languages, the problem of POS tagging has been thoroughly examined [16, 23, 27, 43]. However, there remains a need for research on Indian languages, particularly the Odia language, as it is challenging to do research on languages with complicated morphological inflection and flexible word order. In addition, the lack of capitalization, gender information, and other aspects complicates POS tagging in Odia.

South Asia is the birthplace of five diverse language groups, including Dravidian, Indo-Aryan, Tibeto-Burman, Austro-Asian, and Andamanese. Odia, which belongs to the Indo-Aryan language family and is also known as Oriya, is spoken in India. It is the state language of Odisha, an Indian state. Odia is one of India’s 22 official languages and 14 regional languages, and it is the sixth Indian language to be recognized as a classical language [3]. Previously, it was a Scheduled Language under the Eighth Schedule of the Indian Constitution. The Odia language originated in the 10th century, and later the language was modified in the 16th and 17th centuries. Like most of the Indian languages, Odia is also a resource-poor language with less computerization. Some of the NLP tools that have been developed are either not available online or have not been made public or are half-furnished [28, 31]. Considering the NLP situation in Indian languages, they are poor as far as the availability of electronically annotated written corpora is concerned. In the field of computational linguistics, there has been a lot of research on other Indian languages, but not much on the Odia language. Thus, the Odia language needs to be explored from the language processing perspective. Therefore, we researched the Odia POS labeling task from an NLP perspective. Some research articles on NLP have recently been introduced in Odia, which will be discussed in the following sections.

Many steps are involved in POS tagging, including designing a tagset, creating a corpus by tagging the text of a given language, constructing an algorithm to classify the given untagged text, and so on. A corpus is an extensive collection of texts or words that is required for any NLP task, and it can be divided and used for training, validation, and testing. It is also difficult to obtain a standard corpora in many languages; therefore, datasets or corpora are often challenging. Thus, many researchers have constructed their own corpus [34, 42]. For the purpose of our experiment, we collected a publicly available corpus through the website of the Technology Development for Indian Languages (TDIL) [1]. This was done as part of phase II of the Indian Languages Corpora Initiative (ILCI) project, which was launched by the Ministry of Electronics and Information Technology of the Government of India at Jawaharlal Nehru University in New Delhi. The corpus is tagged using the Bureau of Indian Standards (BIS) tagset and contains 38 tags. Several POS tagsets have been used for different languages by a number of researchers or research groups.
Various tagset schemes are developed for Indian Languages [7]. The BIS tagset, Indian Language Machine Translation (ILMT) tagset, Linguistic Data Consortium for Indian Languages (LDC-IL) tagset, JNU-Sanskrit tagset (JPOS), Sanskrit consortium tagset (CPOS), and Tamil tagset are some tagsets used for Indian languages. However, since all other datasets used in the current experiments all around the globe have been annotated with **Universal Dependencies (UD)** tagset [4, 36, 40], it was necessary that the Odia dataset also uses the same tagset to maintain uniformity.

A POS tagging technique is necessary to identify unique grammatical POS for each input word. Several algorithms are used for the POS tagging task, including rule-based, probabilistic, deep learning, and hybrid approaches. The system should be able to automatically produce the output as tagged terms from the given input text using the technique.

The motivation behind the work is the wide range of applications of POS tagging, such as NER, sentiment analysis, question answering, and word sense disambiguation, among others. Additionally, a handful of research has been done for developing the POS tagger for morphologically rich and resource-poor languages like Odia. However, we did not find any openly available POS tagger for the Odia language. Moreover, the efficiency of deep learning techniques has not been incorporated into the development of the Odia POS tagger.

To address the wide range of NLP challenges, many researchers are coming up with different deep learning methods. We found from recent literature on POS taggers that some models, such as LSTM, **Bi-directional Long Short-Term Memory (Bi-LSTM)**, and **Convolutional Neural Networks (CNN)**, have been frequently used in the construction of POS taggers for resource-poor languages. In 2021, Warjri et al. [42] proposed a Khasi POS tagger using Bi-LSTM, combinations of Bi-LSTM with **Conditional Random Field (CRF)**, and character-based embedding with Bi-LSTM. Priyadarshi and Saha [35] employed **Recurrent Neural Network (RNN)**-based models to construct a Maithili POS tagger in 2022. These models included the **Long Short-Term Memory (LSTM)**, **Gated Recurrent Unit (GRU)**, LSTM with a CRF (LSTM-CRF), and GRU with CRF (GRU-CRF). However, the efficacy of deep learning techniques has not yet been incorporated into the development of the Odia POS tagger. Our goal in this research is to build an improved Odia POS tagger by comparing the performance of several different deep learning-based models.

First, we employed probabilistic supervised learning methods based on the **Maximum Entity Markov Model (MEMM)** and CRF to prepare a baseline system. In these statistical models, we used contextual features, including the previous word, the next word, the current word, the tag of the previous word, fixed-length suffixes, and prefixes. We examined the effectiveness of our method and investigated the performance of the baseline models for Odia POS tagging with the BIS and UD tagsets using the ILCI Odia corpus. Then we employed deep learning based methods, including Bi-LSTM, CNN, and combination of Bi-LSTM and CNN with CRF. Subsequently, we compare the baseline model with deep learning based approaches used for the Odia POS tagging task.

The following are the main contributions of this article:

1. Design the Odia POS corpus using the UD tagset.
2. Develop a baseline Odia POS tagger using MEMM and CRF.
3. Generate word embedding vector for the Odia POS tagging task using the fastText word embedding approach.
4. Construct a deep learning based Odia POS tagger using CNN and Bi-LSTM.

The rest of the article is organized in the following manner. Section 2 describes the literature survey on POS tagging. Section 3 presents the Odia POS dataset and UD tagset for the Odia language. Section 4 describes the methodologies used to develop POS tagging in the Odia language. Section 5
presents the experimental results of different models. Section 6 describes the performance analysis. Section 7 presents our conclusion and discusses future works in these dimensions.

2 RELATED WORK

This section presents the existing related work of POS tagging in different languages and the presently available work on different approaches. Harris [17] was the first to explore POS tagging and used a rule-based approach to develop the POS tagger. Later, numerous rule-based systems were developed to improve the accuracy and efficiency of POS taggers. However, rule-based systems have significant drawbacks, including a high level of human effort, time consumption, and less learning potential. Statistical-based machine learning methods then became quite popular. Cutting et al. [8] proposed the Hidden Markov Model (HMM) to develop a POS tagger using the Brown Corpus with 50,000 tagged words. Lafferty et al. [21] introduced CRF, another statistical model for POS tagging, and they observed that CRF performs better than other statistical models in POS tagging tasks. Later, neural network based POS tagging was introduced, including CNN, LSTM, Bi-LSTM, RNN, and GRU, among others, and these methods have been recently used for sequence labeling tasks.

Here we discuss some of the recent research work reported on POS tagging in different languages. Santos and Zadrozny [14] introduced a deep neural-based POS tagger, and the model combined the word- and character-level representations. This model achieved good accuracy in the English and Portuguese languages. A compositional character to word Bi-LSTM model was proposed by Ling et al. [22]. Here, the Bi-LSTM network was used to compose the characters to construct character representation of words, and it achieved good accuracy in morphologically rich languages. Plank et al. [33] developed a multilingual POS tagging with Bi-LSTM. The model was experimented on 22 different languages and achieved state-of-the-art performance. This model performed well by using word embedding and character embedding features. Huang et al. [18] proposed a Bi-LSTM-CRF model for sequence labeling tasks. Here, the LSTM network is composed of a CRF layer and improves the model’s performance. This model has produced state-of-the-art results on POS, NER, and chunking datasets. Shrivastava and Bhattacharyya [38] proposed an HMM-based tagger. They achieved an accuracy of 93.12%. Additionally, they used a corpus of size 81,751 tokens for the experiment. Alam et al. [2] developed a POS tagger for the Bangla language using the LSTM network and a CRF layer with a word embedding feature. This model was implemented using a publicly available corpus and achieved an accuracy of 86%. Krishnan et al. [19] presented a POS tagger for Tamil using character-based Bi-LSTM. They got an accuracy of 86.45%. Priyadarshi and Saha [34] proposed a Maithili POS tagger using CRF and word embedding. They used an annotated corpus of 52,190 words and achieved an accuracy of 85.88%. Warjri et al. [42] proposed a Khasi POS tagger using a deep learning based approach on a designed Khasi corpus. They achieved an accuracy of 96.98% by using Bi-LSTM with a CRF layer.

A few efforts have been proposed for the Odia language to build a POS tagger. Das and Patnaik [9] presented an implementation of a single neural network based POS tagging algorithm trained on a manually annotated corpus. A small tag set of five tags was considered for labeling the data. A voting-based selection rule to obtain the output and forward propagation method was used for error correction. The Odia POS tagger using an Artificial Neural Network (ANN) got an accuracy of 81%. Das et al. [10] presented an improved Odia POS tagger using Support Vector Machine (SVM), and it achieved 82% training accuracy. They also used only five tags for manually labeling the dataset having 10k tokens. Lexicon, NER tags, and word suffixes were considered as feature sets and fed to the SVM POS tagger. Ojha et al. [28] reported that SVM and a CRF-based POS tagger for the Odia language achieved an accuracy of 91.3% and 85.5%, respectively. An ILCI dataset with 90k and 2k tokens was used for training and testing, and unigram features were
considered during the training period. Pattnaik et al. [31] developed an HMM-based Odia POS tagger and achieved an accuracy of 91.53%. They implemented the tagger using 0.2 million tokens and 11 tags to annotate the tagset.

From the literature survey, we found that Odia did not gain much attention from researchers for NLP tasks like POS tagging, and only minimal research had been carried out. The construction of the Odia POS tagger did not exploit the effectiveness of deep learning based techniques, and no standard tagging system was used for the construction of the Odia POS dataset. Table 1 describes the comparison of the proposed work with existing research work from the literature survey of POS tagging in the Odia language.

3 ODIA POS CORPUS

In this section, we provide a brief discussion on the dataset and tagset utilized in the development of the Odia POS tagger.

3.1 UD POS Tagset for the Odia Language

This section presents a brief discussion on the UD [25] tagset for Odia POS and a mapping from the BIS tagset to the UD tagset. A tagset contains unique tags that are used to label each word in a text. UD is a framework that provides consistent annotation of POS across an extensive collection of different human languages [25, 26]. The general idea of UD is to give similar constructions across languages and transfer syntactic knowledge across multiple languages. The UD annotation approach is founded on an evolution of Google’s universal POS tags [32], Stanford dependencies [11, 12], and the Interset interlingua for morphosyntactic tagsets [44]. The ultimate goal of UD is to produce a linguistic representation that will be useful for the study of morphosyntactic structures, the interpretation of semantic meaning, and the application of NLP to human languages. UD bears a striking resemblance to the conventional parts of speech. UD is somewhat similar to the conventional parts of speech. It recognizes 17 different classes of words in addition to a variety of other text elements and assigns labels to each of these categories. These categories have a significant amount of linguistic support around the globe.

Researchers have used different tagsets to develop a POS tagger for the Indian language. Some of are the following. Ekbal et al. [15] developed the Bengali POS Tagger using a 26-tag tagset, Suraksha et al. [39] presented a Kannada tagger that contains 19 tags in the tagset, Dhanalakshmi et al. [13] defined a Tamil POS tagset containing 32 tags, and Warjiri et al. [41] introduced a tagger for the Khasi language that has 54 labels. The use of different tagsets complicates the comparative study. However, since all other datasets of various languages around the globe have been used, the corpus is annotated with the UD tagset in the current experiments. Most of the languages follow the UD tagging framework to maintain uniformity among all languages. However, since there is no availability of the Odia dataset annotated with the UD POS tagset, it is necessary that the Odia dataset also uses the same tagset to maintain uniformity. The collected ILCI corpus is annotated...
using the BIS tagset. Thus, we constructed a simple mapping of tags from the BIS tagset to UD tags based on their respective category. The mapping is given in Table 2.

Mapping from the Odia POS BIS tagset to the UD POS tagset is a challenging task. Many of the tags are mapped directly from the BIS tagset to the UD tagset, some of which are nouns, pronouns, verbs, adverbs, adjectives, and so on. A couple of tags made it difficult to map because there is no proper class defined in the UD tagset, like demonstratives, quantifiers, and echo words. General quantifiers do not indicate any precise quantity, which occurs as an intensifier in the Odia language. The most common general quantifiers used in the Odia language are Adhika (much), Kebala (only), and Ahuri (more). Thus, general quantifiers mapped to an intensifier—that is, particles in the UD tagset. The other two quantifiers in the BIS tagset are directly mapped to the UD tagset numeral. The BIS tagset does not have a separate category for determiners and adpositions. However, it does contain demonstratives and postpositions, which do not occur in the UD tagset. Therefore, we classified postposition and demonstrative as adposition and determiner tags, respectively, in the UD tagset. The echo word is one of POS classes in the BIS tagset, but the UD POS tagset does not have echo word category. In Odia sentences, echo words are also used as nouns, adjectives, adverbs, and so on, depending upon their use in a sentence. From the following examples, we observed that echo words could have mapped to different POS categories based on the syntactic structure of the Odia language. The TDIL Odia POS dataset contains 268 echo words. Thus, we manually mapped the echo words to their respective group.

### Example 1:

| BIS Tagset       | UD Tagset       |
|------------------|-----------------|
| jaNashuNA        | num             |
| dhAna            | num             |
| Adhika           | int             |

Translation: Basmati rice is the most famous of them.

### Example 2:

| BIS Tagset       | UD Tagset       |
|------------------|-----------------|
| se bhAratiYa     | conj            |
| aHilYare         | conj            |
| jaNe             | conj            |
| jaNashuNA        | conj            |
| bYaktitva        | conj            |

Translation: He is a well-known person in Indian literature.

### Example 3:

| BIS Tagset       | UD Tagset       |
|------------------|-----------------|
| ahAnaku          | det             |
| saphaKarA       | adposition      |
| anAbanA          | det             |
| ghAsaksu         | adposition      |
| bAhArA           | det             |
| karidiantu       | adposition      |

Translation: Clean up the garden and remove weeds.

**3.2 Corpus Information**

Under phase II of the ILCI project, funded by the Ministry of Electronics and Information Technology, Government of India, Jawaharlal Nehru University, New Delhi, we accessed two publicly accessible corpora via the TDIL website. The first corpus contains monolingual Odia sentences with POS tags, whereas the second corpus contains parallel sentences having POS tags from different domains. Roughly 51,150 sentences are included in the resulting dataset. The translated sentences have been POS tagged according to the BIS tagset. This corpus has the following features: a unique ID, UTF-8 encoding, and text file format. In the corpus, we identified that some special unwanted characters were associated with a number of actual tags. Therefore, the total number of unique tags increased to 87, but only 38 unique tags from the BIS tagset were used to tag the dataset. The corpus also had empty lines, untagged sentences, and redundant space characters. Therefore, it was necessary to do data processing prior to utilizing the data to train the model.
Table 2. Mapping from the BIS Tagset to the UD Tagset

| Sl. No. | BIS Tag  | BIS Category          | UD Tag  | UD Category   |
|---------|----------|-----------------------|---------|---------------|
| 1       | N_NN     | Common Noun           | NOUN    | Noun          |
| 2       | N_NNV    | Verbal Noun           | NOUN    | Noun          |
| 3       | N_NST    | Nloc                  | NOUN    | Noun          |
| 4       | N_NNP    | Proper Noun           | PROPN   | Proper Noun   |
| 5       | PR_PRP   | Personal Pronoun      | PRON    | Pronoun       |
| 6       | PR_PRF   | Reflexive             | PRON    | Pronoun       |
| 7       | PR_PRL   | Relative Pronoun      | PRON    | Pronoun       |
| 8       | PR_PRC   | Reciprocal            | PRON    | Pronoun       |
| 9       | PR_PRQ   | Wh-word               | PRON    | Pronoun       |
| 10      | PR_PRI   | Indefinite Pronoun    | PRON    | Pronoun       |
| 11      | DM_DMD   | Deictic Demonstrative | DET     | Determiner    |
| 12      | DM_DMR   | Relative Demonstrative| DET     | Determiner    |
| 13      | DM_DMQ   | Wh-word Demonstrative | DET     | Determiner    |
| 14      | DM_DMI   | Indefinite Demonstrative | DET   | Determiner    |
| 15      | V_VM     | Main Verb             | VERB    | Verb          |
| 16      | V_VM_VNF | Finite Verb           | VERB    | Verb          |
| 17      | V_VM_VNIF| Non-Finite Verb       | VERB    | Verb          |
| 18      | V_VM_VNG | Gerund                | VERB    | Verb          |
| 19      | V_VAUX   | Auxiliary             | AUX     | Auxiliary     |
| 20      | JJ       | Adjective             | ADJ     | Adjective     |
| 21      | RB       | Adverb                | ADV     | Adverb        |
| 22      | PSP      | Postposition          | ADP     | Adposition    |
| 23      | CC_CCS   | Subordinating Conjunction | SCONJ | Subordinating Conjunction |
| 24      | CC_CCS_UT| Quotative Conjunction | SCONJ   | Subordinating Conjunction |
| 25      | CC_CCD   | Coordinating Conjunction | CCONJ | Coordinating Conjunction |
| 26      | RP_RPD   | Default Particle      | PART    | Particle      |
| 27      | RP_INJ   | Interjection          | INTJ    | Interjection  |
| 28      | RP_INTF  | Intensifier           | PART    | Particle      |
| 29      | RP_CL    | Classifier            | PART    | Particle      |
| 30      | RP_NEG   | Negation              | PART    | Particle      |
| 31      | QT_QTF   | General Quantifier    | PART    | Particle      |
| 32      | QT_QTC   | Cardinal Quantifier   | NUM     | Numeral       |
| 33      | QT_QTO   | Ordinal Quantifier    | NUM     | Numeral       |
| 34      | RD_PUNC  | Punctuation           | PUNCT   | Punctuation   |
| 35      | RD_SYM   | Symbol                | SYM     | Symbol        |
| 36      | RD_RDF   | Foreign Word          | X       | Other         |
| 37      | RD_UNK   | Unknown               | X       | Other         |
After data cleansing, we obtained 48,000 Odia POS-tagged sentences. From the available corpus, we split the dataset into training, validation, and test sets according to 70%, 15%, and 15%, respectively. A few samples taken from the ILCI corpus are shown in Table 3. Details of the dataset are shown in Table 4. The distribution of the ILCI corpus over the UD tagset is presented in Table 5.

### 4 SYSTEM MODEL

The statistical and deep learning models that were utilized to construct the Odia POS tagger are explained next.

#### 4.1 Odia POS Tagger Developed Using a Statistical Method

Several existing models are available to solve sequence labeling tasks like POS tagging, NER, and chunking. HMM, MEMM, and CRF are the stochastic approaches for addressing sequence labeling problems. Popular probabilistic models for solving sequence labeling tasks include the HMM, which is a generative model and defines a joint distribution over label and observation sequences. HMMs rely on feature-independent assumptions. This makes it difficult to include more additional features. Since the Odia language contains non-independent, diverse, and overlapping features, a simple HMM cannot handle these complex and arbitrary features. In this work, we used discriminative models such as MEMM and CRF to build an Odia POS tagger because they allow us to add a much richer variety of features.

The MEMM is a probabilistic model that is typically employed for sequence labeling and information extraction tasks. It is a discriminative model composed of the HMM and Maximum Entropy (MaxEnt) models. The HMM for predicting sequence labels given an observation sequence and the MaxEnt model provide flexibility in the use of features that can be extracted from...
Table 5. Distribution of ILC1 Odia POS Corpus over UD Tagset

| Tag     | Description             | Count   |
|---------|-------------------------|---------|
| NOUN    | Noun                    | 258,726 |
| VERB    | Verb                    | 86,369  |
| PUNCT   | Punctuation             | 68,322  |
| PROPN   | Proper noun             | 58,866  |
| ADJ     | Adjective               | 43,502  |
| NUM     | Numeral                 | 28,904  |
| CCONJ   | Coordinating conjunction| 26,170  |
| PART    | Particle                | 26,048  |
| PRON    | Pronoun                 | 22,426  |
| DET     | Determiner              | 19,337  |
| ADP     | Adposition              | 13,255  |
| ADV     | Adverb                  | 9,387   |
| SCONJ   | Subordinating conjunction| 6,485  |
| X       | Other                   | 1,571   |
| AUX     | Auxiliary               | 1,397   |
| INTJ    | Interjection            | 444     |
| SYM     | Symbol                  | 145     |

the observation sequence. Conditional probability models, such as the MaxEnt model, can be used to predict class labels from a set of features for a particular data point. It determines which label has the highest likelihood by identifying the label with the highest score. To obtain a good Odia POS tagger using the MEMM, it is crucial to select a collection of relevant features. The MEMM is more flexible because it permits the addition of contextual, morphological, and lexical features, among others. Therefore, the MEMM allows for diverse non-independent overlapping features. In both of our baseline models, we employed the same set of features, and the details of the feature set are discussed in Section 5.

Although the MEMM solves all of the problems that the HMM has, it does have a few drawbacks. MEMMs are normalized locally for each observation and hence suffer from the label bias issue. CRF addresses the issue of the label bias problem. The CRF model is the most commonly used sequencing model for labeling tasks. Moreover, the CRF model makes it easy to examine the word before and after the entity by its undirected graph property. CRFs are the type of discriminative model that learns the conditional probability distribution. CRFs are an undirected graph-based method where the node represents input or observation sequence \( x \) that corresponds to label or output sequence \( y \). This model aims to find the output label or tag \( y \) that maximizes the conditional probability \( P(y|x) \) for a given input sequence \( x \). For finding the most probable sequence label from the given word sequence, mathematically we can write it as \( Y = \arg\max P(x|y) \). Let the observation word sequence \( x = (x_1, x_2, x_3, \ldots, x_n) \) and corresponding tag sequence \( y = (y_1, y_2, y_3, y_n) \).
Mathematically, the conditional probability chain structure of the CRF model can be expressed as

$$P(y | x) = \frac{1}{Z_x} \exp \left( \sum_{t=1}^{n} \sum_{k=1}^{K} \lambda_k \cdot f_k(y_{t-1}, y_t, x) \right),$$

where $\lambda$ is the weight associated with each distinct feature $K$, and the model depends on the set of features. $f_k(y_{t-1}, y_t, x)$ denotes the feature function whose value is represented in binary. It may be 0 or 1. $Z_x$ is used for normalization, and the sum of probability of all state sequences is 1. $Z_x$ can be represented as

$$Z(x) = \sum_y \exp \left( \sum_{t=1}^{n} \sum_{k=1}^{K} \lambda_k \cdot f_k(y_{t-1}, y_t, x) \right).$$

All details of the feature set are explained in Section 5.

4.2 Odia POS Tagger Developed Using Deep Learning Approaches

In this section, we discuss the development of the Odia POS tagger using deep learning based approaches. Here, we built a POS tagger for the Odia language by employing a variety of deep learning based models, such as CNN, Bi-LSTM, and a combination of CNN and Bi-LSTM with CRF. The architecture of the deep learning based model used for Odia POS tagging is shown in Figure 1. For the sake of simplicity, we summarized the steps involved in developing the deep learning model for Odia based on POS tagging. Figure 2 illustrates the overall architecture of the Bi-LSTM model used for Odia POS tagging. We have the following:

1. An Odia sentence is taken as an input to the model.
2. CNN or Bi-LSTM neural encoders are used to integrate character sequence information of Odia word as character-level embedding.
(3) We generated a word embedding vector of more than 4.5 million Odia sentences from different resources as initialization for word-level embedding using the fastText word embedding method.

(4) Character-level and word-level embedding are concatenated and fed into a fully connected neural network.

(5) The output of the previous step is input to the word sequence layer, and final word embedding is fed into the word sequence layer.

(6) Finally, the last hidden layer’s output of the word sequence layer is taken as input to the inference layer (softmax or CRF) to predict the possible tags for each input sentence.

### 4.2.1 Character Sequence Layer

For sequence labeling tasks, it is important to consider word morphology. Character-level embedding is taken to represent characters of input words. It is used to deal with language with complex morphology. Character embeddings can be used to represent character features, such as prefixes and suffixes, among other character features.

Example 1: ଜଣ୍ଣିରେମା ଲେଖନା କରାଇ ଆମର ମନ୍ଦିର ଅପାରକାଳକ କରିତା କରିତା ।
Transliteration: chASHlmAnE khajaNA ChA.Da karibA pAi.N sarakAra~Nku anureAdha karithile ।
Translation: The farmers had urged the government to waive the tax.

Example 2: ତେକାରା ଲେଖନା ନିର୍ଦ୍ୱରଣ କରିଲା ।
Transliteration: semAnE anya leAkara jamire chASHa karanti ।
Translation: They cultivate other people’s land.

In most of the cases, the suffix (mAnE) defines a word as a noun or pronoun, as shown in both examples.
To extract the character-level information, we made use of two different kinds of networks: CNN and Bi-LSTM. The CNN approach is an effective method for extracting morphological information of characters from words that are given. Each character within a word token is initially mapped to a character vector. Then, filters of varying sizes are applied to the embedding matrix to capture the key characteristics of nearby inputs. The final stage of the CNN consists of an operation called max-pooling, which extracts a single feature from each of the feature maps. After that, the output characteristics are concatenated to maintain the information that is specific to each word. Moreover, LSTM models are also applied for character-level information extraction. Bi-LSTM is used to record sequence information from left to right (forward LSTM) and right to left (backward LSTM). Using bidirectional hidden states, the model can then preserve both past and future knowledge. In addition, the Bi-LSTM model captures the global characteristics of each word token. Then, the two final hidden states of the forward and backward LSTM are concatenated to get a vector of fixed size representing a word token.

4.2.2 Word Embedding. In many NLP tasks, nearby words play a crucial role. Additionally, the performance of sequence labeling tasks, such as POS tagging, NER, and chunking, depends on the surrounding terms. To accomplish this, we require word embedding for distributed word representation, which maps words into a low-dimensional vector space. Word vectors have the advantage of capturing relationships between words. NLP offers a variety of strategies for word embedding. In our work, we used fastText, one of the embedding techniques developed by Facebook. The key benefit of fastText embeddings is that they take into account the internal structure of words when learning word representations. Therefore, each word is represented as an n-gram of characters, preserves subword information, and can compute valid word embeddings for out-of-vocabulary terms. FastText can provide the vector for unseen words during word embedding training. The word vectors are generated after training the fastText model. To learn word representations, contextually related words often occur with similar surrounding words. However, there are no publicly available pre-trained Odia word vectors. Therefore, we choose to train the models with the fastText model. For training purposes, a huge corpus of Odia text is required. Therefore, we gathered raw corpora from several resources [20, 29, 30, 37]. The collection includes around 4.5 million Odia sentences. The input for fastText training is a large corpus of text, and the output is a vector with several hundred dimensions for each unique word in the corpus. The model contains two network-based variants: continuous bag of words (CBOW), and skip-gram. In this work, the skip-gram approach was chosen for the training process. The input for the skip-gram architecture is a single word, and it generates predictions for words that are contextually connected to that word. Runtime and the quality of the trained model are both impacted by a number of different parameter selections. After conducting the experiment with several different vectors dimension values (ranging from 100 to 150, 200, 250, and 300), we observed that the value of 200 worked well in our experiment. Additionally, we provided the other set of hyperparameters to train the fastText model: context window size of 5, vector dimension of 200, learning rate of 0.1, epoch of 100, minimum word count of 2, and softmax loss function. The word vectors are generated after training the fastText model.

4.2.3 Word Sequence Layer. Like the character sequence layer, the word sequence layer consists of both CNN and Bi-LSTM neural networks. Word representations, which may comprise word embeddings and character sequence representations, are the input of the word sequence layer. Stacking the word sequence layer enables the development of a more robust feature extractor.

Here, we apply a multi-layer CNN on a sequence of words where the words are represented by embeddings. Word representations are generated by concatenating character sequence representations with word embeddings. For each CNN layer, convolution is carried out on the embedded word vectors with varying filter sizes. Word sequence patterns that convey characteristics like
grammatical function can be considered features, and the convolution operation can be interpreted as a window-based method for extracting features. The convolution filter or kernel is applied to each window and can be performed on a few word embeddings. These kernels will move along a list of word embeddings in a sequential fashion so that the complete sequence of words can be processed. This type of convolution is referred to as a one-dimension convolution since the kernel is only moving in one dimension. Here, we utilized a window with a size of three slides along the sequence, extracted local features from the word inputs, and then followed a rectified linear unit (ReLU) function. Following the completion of each CNN layer, batch normalization and dropout are applied.

Bi-LSTM, one of the RNNs, was employed at the word sequence layer (RNN). To overcome the limits of set window size, a bidirectional LSTM captures arbitrary length context data. It is helpful to have access to knowledge about the sequence’s context in the past as well as the sequence’s context in the future for many sequence labeling tasks. However, simple LSTM networks merely take input from the past and are unaware of the context of the future. Bi-LSTM is capable of remembering both the left and right contextual information associated with each word. Two distinct hidden states of the Bi-LSTM are used to capture information from the past and the future, which are then concatenated together to produce the final output.

4.2.4 Inference Layer. This is the final layer of our deep learning based strategy for POS tagging in Odia. Here, the output of the word sequence layer is input to the inference layer, which assigns labels to each word in the sentence or document. In this, we applied CRF and softmax as the output layer. We found in the literature that CRF in the output layer improves the performance of some sequence labeling tasks. The CRF layer acquires knowledge from successive labels by employing a state transition matrix as a set of parameters; with this information, we are able to anticipate the current tag. Because it allows for concurrent decoding, softmax performs better than CRF in certain sequence labeling tasks [22].

5 EXPERIMENTAL RESULTS

5.1 Statistical Model Results

In this section, we provide a brief description of training and testing using the MEMM and CRF models to prepare baseline systems and establish a relevant feature set for interpreting their impact on the Odia POS system. Here we used the same set of features for both of these models. We investigated the efficacy of our approach using the ILCI Odia dataset. Using the ILCI Odia corpus, we evaluated the performance of the models for Odia POS tagging with the BIS and UD tagsets. Our baseline models are trained using LBFGS (Limited-Memory Broyden-Fletcher-Goldfarb-Shanno), a quasi-Newton approach, and this algorithm is used for large-scale numerical optimization problems. For our experiment, we first defined the feature set to train the model. To improve the overall performance of any machine learning task, it is necessary to have a feature set, which should include valuable features. Feature templet contains unigram features to define contextual information during training. A prefix, an identifying number, and a rule string are included in each template. The template type can be determined by the prefix; for example, “U” denotes a unigram template. An identification number is assigned to each template so that they may be distinguished from one another, and a rule string is utilized to guide CRF in generating features.

The information regarding the previous word, the next word, the current word, and the tag of the previous word is contained within the defined feature templates. According to the literature, information about the root word and its morphological inflections is quite helpful for the POS labeling task. The suffix and prefix information are then added to the feature set. For the development of the Odia POS classifier, several experiments were conducted using various combinations
Table 6. Description of the Feature Set

| Feature ID | Feature Template | Description |
|------------|------------------|-------------|
| \( f_1 \)  | “U01:%x[−1,0]%x[0,0]%x[1,0]” | Current word + Previous word + Next word |
| \( f_2 \)  | “U02:%x[−2,0]%x[−1,0]%x[0,0]” | Current word + Previous two words |
| \( f_3 \)  | “U03:%x[0,0]%x[1,0]%x[2,0]” | Current word + Next two words |
| \( f_4 \)  | “U04:%x[0,1]” | POS tag of previous word |
| \( f_5 \)  | “U05:%x[0,2]” | Suffix length 3 of current word |
| \( f_6 \)  | “U06:%x[0,3]” | Suffix length 4 of current word |
| \( f_7 \)  | “U07:%x[0,4]” | Suffix length 5 of current word |
| \( f_8 \)  | “U08:%x[0,5]” | Prefix length 2 of current word |
| \( f_9 \)  | “U09:%x[0,6]” | Prefix length 3 of current word |

of individual features to identify the feature set with the highest accuracy. In Table 6, we present a summary of the many possible feature combinations that are utilized in our experiment. In the construction of the "word features," a variety of different combinations of preceding and following words are used. In addition, we made use of suffix and prefix information, with the length of the suffix varying between three, four, and five characters and the length of the prefix varying between two and three characters.

Models are trained using the regularization technique L2 to generate all features needed to calculate probabilities at the time of tagging. We trained the CRF model multiple times by modifying the parameter, and the optimal result was obtained by setting the -c and -f parameters to 1.5 and 2, respectively. Here, ‘f’ defines the cutoff threshold for the features, whereas ‘c’ specifies the balance between overfitting and underfitting. We evaluated the efficacy of the MEMM- and CRF-based taggers using approximately 276,900 tokens from Corpus I and about 711,000 tokens from combined Corpora I and II on BIS and UD tagsets. In Table 7, we summarize the combination of features represented by feature id and corresponding accuracy. Here we can observe from Table 7 that the accuracy increases as the corpus size grows. In these experiments, MEMM and CRF taggers with defined feature sets obtained a maximum accuracy of 90.97% and 92.08%, respectively. Thus, the performance of the CRF-based Odia POS tagger is better than that of the MEMM-based tagger.

5.2 Deep Learning Model Results

This section provides a brief explanation of the experimental outcomes of the deep learning based technique for the Odia POS tagger. We used the same corpus for this experiment as we did in our initial baseline models. Approximately 276,900 tagged tokens (36,505 unique words) and 671,000 tagged tokens (75,394 unique words) are fed into the systems. For our implementation, 70% of the corpus is used for training, 15% is used for validation, and the remaining 15% is used for testing. For our experiment, we employed CNN and Bi-LSTM, two deep learning based approaches, to construct a POS tagger for the Odia language. The training for both of these models was done with the use of hyperparameter values that gave the best results on the validation set. The selected hyperparameters that are used in our experiments are summarized in Table 8. To initialize the word, we used the pre-trained word embedding dimension of 200 and the character embedding size of 30. The Stochastic Gradient Descent (SGD) optimizer is applied to optimize the parameters.
Table 7. Accuracy of MEMM- and CRF-Based Taggers Using Different Features

| Sl No. | Dataset (Tokens) | Feature Details | Test Accuracy (%) on the BIS Tagset of the CRF Model | Test Accuracy (%) on the UD Tagset of the CRF Model | Test Accuracy (%) on the BIS Tagset of the MEMM Model | Test Accuracy (%) on the UD Tagset of the MEMM Model |
|--------|------------------|----------------|--------------------------------------------------------|------------------------------------------------|------------------------------------------------|------------------------------------------------|
| 1      |                  | $f_1$          | 87.56                                                  | 89.24                                             | 86.32                                             | 88.40                                             |
| 2      |                  | $f_2$          | 87.02                                                  | 88.55                                             | 86.05                                             | 87.73                                             |
| 3      | 193,840 tokens for training | $f_3$          | 87.33                                                  | 88.82                                             | 86.20                                             | 87.98                                             |
| 4      | 41,537 tokens for training    | $f_1 + f_4$    | 87.74                                                  | 89.49                                             | 86.93                                             | 88.51                                             |
| 5      |                  | $f_1 + f_2 + f_3 + f_6 + f_7$ | 87.98                                                  | 89.65                                             | 87.08                                             | 88.62                                             |
| 6      |                  | $f_1 + f_4 + f_3 + f_6 + f_7 + f_8$ | 88.14                                                  | 89.97                                             | 87.22                                             | 88.70                                             |
| 1      |                  | $f_1$          | 89.76                                                  | 91.42                                             | 88.62                                             | 90.35                                             |
| 2      |                  | $f_2$          | 88.87                                                  | 90.15                                             | 88.88                                             | 89.50                                             |
| 3      | 469,950 tokens for training | $f_3$          | 89.01                                                  | 90.59                                             | 88.10                                             | 89.67                                             |
| 4      | 100,704 tokens for testing | $f_1 + f_4$    | 89.77                                                  | 91.50                                             | 88.79                                             | 90.54                                             |
| 5      |                  | $f_1 + f_2 + f_3 + f_6 + f_7$ | 90.41                                                  | 91.83                                             | 88.93                                             | 90.72                                             |
| 6      |                  | $f_1 + f_4 + f_3 + f_6 + f_7 + f_8$ | 90.45                                                  | 92.08                                             | 89.16                                             | 90.97                                             |

with a minimum batch size of 10. The Bi-LSTM and CNN models were trained using 50 and 100 epochs, respectively, and their respective learning rates were 0.015 and 0.005. In addition to this, we used the regularization approaches of l2 regularization and a dropout value of 0.3 to enhance the performance of the model on the unseen data.

To estimate robustness of models with respect to features, we train on seven different combinations on both of the models including Bi-LSTM, CNN, WE + Bi-LSTM, WE + Bi-LSTM + CRF, WE + CNN, WE + CNN + CRF, WE + CharCNN + Bi-LSTM, WE + CharCNN + CNN, WE + CharBi-LSTM + CNN, WE + CharCNN + Bi-LSTM + CRF, WE + CharCNN + CNN + CRF, WE + CharBi-LSTM + Bi-LSTM + CRF, and WE + CharBi-LSTM + CNN + CRF. Both corpora are used to train every model. There are three CRF-based models and four softmax-based models with distinct representations of character and word sequences. Here, token accuracy is used to evaluate system performance. The results of the Odia POS tagger on the BIS and UD tags sets using the deep learning based technique are presented in Table 9, where CharCNN indicates models that use a CNN for character sequence information, CharBi-LSTM indicates models that use Bi-LSTM for character sequence information, and WE indicates word embedding information. Among the various combination of features, the results demonstrate that Bi-LSTM with a pre-trained word vector and character information using a CNN provided the highest accuracy of 94.48%. Additionally, we observed that models with character-level information perform better than models without character-level information. It was found that the accuracy of deep learning based approaches for the Odia POS tagger on BIS and UD tags sets outperforms the accuracy of existing work. Table 10 presents the comparison results of various statistics and deep learning models.
Table 8. Hyperparameters for the Deep Learning Model

| Parameters                        | Value  |
|-----------------------------------|--------|
| Character embedding dimension     | 30     |
| Hidden layer character dimension  | 50     |
| Word embedding dimension          | 200    |
| Hidden layer word dimension       | 300    |
| LSTM layer                        | 2      |
| CNN layer                         | 4      |
| Optimizer                         | SGD    |
| Learning rate for the Bi-LSTM model | 0.015  |
| Learning rate for the CNN model   | 0.005  |
| Learning decay rate               | 0.05   |
| Momentum                          | 0      |
| Dropout                           | 0.3    |
| Batch size                        | 10     |
| Number of epochs for the Bi-LSTM model | 50     |
| Number of epochs for the CNN model | 100    |

5.3 Comparison with Existing POS Taggers in the Odia Language

We compared the performance of our MEMM, CRF, CNN, and Bi-LSTM models with the results of previous work on Odia POS tagging. Table 11 presents the results of other researchers and the results of our model. From the other researchers’ results, we can see that Pattnaik et al. [31] presented the Odia POS tagger on one of the statistical models for sequence labeling tasks—that is, the HMM gives the highest accuracy of 91.53. However, the proposed POS tagger on the MEMM and CRF models gives 90.97% and 92.08% accuracy, respectively, using surrounding word information and prefix and suffix information features to increase the accuracy. We can observe from Table 9 that the result achieved by the CNN and Bi-LSTM models using BIS and UD tagsets on the ILCI corpus gives better accuracy compared to the existing research work.

6 ERROR ANALYSIS

Examining the nature of improperly classified tags is essential to POS tagging. We performed an error analysis on the predicted POS tags using the CNN and Bi-LSTM models. The confusion matrices of Bi-LSTM and CNN for the Odia POS tagger are presented in Appendix A. Table 12 presents the misclassification classes for CNN and Bi-LSTM models derived from the respective confusion matrices. Here we consider the top 5 misclassification labels and calculate their number of instances and error rate.

From the confusion matrices, we observed that the Bi-LSTM model predicts the label more accurately for the majority of classes than the CNN model. In contrast, the CNN model predicts the label more accurately than Bi-LSTM for some classes. We determined from our experiment that both models predict the same label with a 96.14% of accuracy on the validation set. On the validation set, however, the Bi-LSTM and CNN models correctly classify the words with an accuracy of 63.68 and 29.97%, respectively, despite the fact that their predicted labels differ.

Therefore, we took into consideration the label information as a feature in which the models give the same predicted tags, and we trained the model with this additional information using Bi-LSTM and CNN to improve the performance of the Odia POS tagger. Here we concatenated additional tag information of training and validation data with a word embedding vector, and the updated...
Table 9. Comparison of Tagging Performance on POS Tasks for Various Neural Network Models

| Dataset           | Model                                      | Test Accuracy (%) on the BIS Tagset | Test Accuracy (%) on the UD Tagset |
|-------------------|--------------------------------------------|-------------------------------------|-------------------------------------|
| 193,840 tokens for training | CNN                                        | 87.47                               | 88.85                               |
|                   | WE + CNN                                   | 88.72                               | 90.14                               |
|                   | WE + CNN + CRF                             | 88.53                               | 90.12                               |
|                   | WE + CharCNN + CNN                         | 89.31                               | 90.81                               |
|                   | WE + CharBi-LSTM + CNN + CRF               | 89.13                               | 90.67                               |
|                   | WE + CharCNN + CNN + CRF                   | 89.01                               | 90.44                               |
|                   | WE + CharBi-LSTM + CNN + CRF               | 88.91                               | 90.34                               |
| 469,950 tokens for training | CNN                                        | 89.32                               | 91.02                               |
|                   | WE + CNN                                   | 90.74                               | 92.31                               |
|                   | WE + CNN + CRF                             | 90.69                               | 92.14                               |
|                   | WE + CharCNN + CNN                         | 91.29                               | 92.77                               |
|                   | WE + CharBi-LSTM + CNN + CRF               | 91.02                               | 92.43                               |
|                   | WE + CharBi-LSTM + CNN + CRF               | 90.89                               | 92.35                               |
| 193,840 tokens for training | Bi-LSTM                                    | 88.97                               | 90.24                               |
|                   | WE + Bi-LSTM                               | 90.14                               | 91.65                               |
|                   | WE + CNN + CRF                             | 90.11                               | 91.55                               |
|                   | WE + CharCNN + Bi-LSTM                     | 90.84                               | 92.15                               |
|                   | WE + CharBi-LSTM + Bi-LSTM                 | 90.65                               | 92.03                               |
|                   | WE + CharCNN + Bi-LSTM + CRF               | 90.49                               | 91.98                               |
|                   | WE + CharBi-LSTM + Bi-LSTM + CRF           | 90.32                               | 91.83                               |
| 469,950 tokens for training | Bi-LSTM                                    | 90.75                               | 92.65                               |
|                   | WE + Bi-LSTM                               | 92.06                               | 94.02                               |
|                   | WE + CNN + CRF                             | 91.95                               | 93.84                               |
|                   | WE + CharCNN + Bi-LSTM                     | 92.60                               | 94.48                               |
|                   | WE + CharBi-LSTM + Bi-LSTM                 | 92.44                               | 94.23                               |
|                   | WE + CharCNN + Bi-LSTM + CRF               | 92.32                               | 94.17                               |
|                   | WE + CharBi-LSTM + Bi-LSTM + CRF           | 92.21                               | 94.09                               |
Table 10. Comparison of Results (Accuracy (%)) on Different Approaches

| Tagset | Dataset | Statistical Models | Deep Learning Models |
|--------|---------|--------------------|----------------------|
|        |         | MEMM | CRF | CNN | LSTM |
| BIS    | Training | 92.55 | 92.94 | 94.19 | 94.80 |
|        | Test     | 89.16 | 90.45 | 91.29 | 92.60 |
|        | Validation | 89.11 | 90.53 | 91.20 | 92.58 |
| UD     | Training | 93.37 | 94.22 | 95.16 | 96.76 |
|        | Test     | 90.97 | 92.08 | 92.77 | 94.48 |
|        | Validation | 91.08 | 92.07 | 92.84 | 94.51 |

Table 11. Comparison of the Performance of Our Models with Existing Works

| Model | Dataset | Tagset | Accuracy (%) |
|-------|---------|--------|--------------|
| ANN [9] | NA | 5 tags | 81 |
| SVM [10] | 10,000 | 5 tags | 82 |
| CRF [28] | training: 90k, validation: 2k | BIS tagset (38 tags) | 82–86 |
| HMM [31] | 0.2 million tokens | 11 tags | 91.53 |

| Model | Dataset | Tagset | Accuracy (%) |
|-------|---------|--------|--------------|
| MEMM | Training: 469,950, validation: 100,704 | BIS tagset (38 tags) | 89.16 |
| MEMM | Training: 469,950, validation: 100,704 | UD tagset (17 tags) | 90.97 |
| CRF | Training: 469,950, validation: 100,704 | BIS tagset (38 tags) | 90.45 |
| CRF | Training: 469,950, validation: 100,704 | UD tagset (17 tags) | 92.08 |
| CNN | Training: 469,950, validation: 100,704 | BIS tagset (38 tags) | 91.29 |
| CNN | Training: 469,950, validation: 100,704 | UD tagset (17 tags) | 92.77 |
| Bi-LSTM | Training: 469,950, validation: 100,704 | BIS tagset (38 tags) | 92.60 |
| Bi-LSTM | Training: 469,950, validation: 100,704 | UD tagset (17 tags) | 94.48 |

word embedding vector was input to the Bi-LSTM and CNN models. Let \( Z_w \) be the original word embedding produced by fastText for word \( w \). We created 18 different one-hot encoded vectors \( x_0, x_1, \ldots, x_{17} \). Here, \( x_1, x_2, \ldots, x_{17} \) represents each POS tag: \( x_1 = (1, 0, \ldots, 0)_{17} \) corresponds to noun, \( x_2 = (0, 1, \ldots, 0)_{17} \) corresponds to verb, and so on. For word \( w \), if both Bi-LSTM and CNN agreed on the tags (say, \( x_i \)), then we replaced \( Z_w \) with \( Z_w \oplus x_i \). If they disagreed with word \( w \), we replaced \( Z_w \) with \( Z_w \oplus x_0 \) and also concatenated \( x_0 \) with \( Z_w \) for any new words that came into the vocabulary. Here, \( x_0 \) represents \((0, 0, \ldots, 0)_{17}\). This modified word embedding was input to the Bi-LSTM and CNN models. We observed the results across five iterations and found that in the third iteration, the validation accuracy was 94.56%. Its corresponding test accuracy was 94.58%, which was the best result among the five iterations and improved over the original accuracy of 94.48%. We repeated this procedure and monitored model accuracy until model accuracy improved with each iteration. The result is presented in Table 13. We determined from the table that the Bi-LSTM model achieved the maximum accuracy, at 94.58%. Hence, based on the observations, the experimental results
Table 12. Top Five Misclassification Classes for CNN and Bi-LSTM Models

| Model   | Actual Label | Predicted Label | No. of Instances | Error Rate (%) |
|---------|--------------|-----------------|------------------|----------------|
| CNN     | NOUN         | ADJ             | 890              | 12.42          |
|         | ADJ          | NOUN            | 850              | 11.86          |
|         | PROPN        | NOUN            | 760              | 10.61          |
|         | NOUN         | PROPN           | 650              | 9.07           |
|         | NOUN         | VERB            | 541              | 7.55           |
| Bi-LSTM | PROPN        | NOUN            | 694              | 12.71          |
|         | ADJ          | NOUN            | 684              | 12.52          |
|         | NOUN         | ADP             | 582              | 10.65          |
|         | NOUN         | PROPN           | 477              | 8.73           |
|         | NOUN         | VERB            | 362              | 6.63           |

Table 13. Updated Results of the CNN and Bi-LSTM Models

| Iteration | No. of Instances where Results of CNN Equals to Bi-LSTM | Accuracy (%) of the CNN Model | Accuracy (%) of the Bi-LSTM Model |
|-----------|---------------------------------------------------------|--------------------------------|----------------------------------|
|           |                                                         | Validation Testing            | Validation Testing               |
| –         | –                                                       | –                              | –                                |
| 1         | 544,533                                                 | 92.75                          | 92.89                            |
| 2         | 549,850                                                 | 93.03                          | 93.09                            |
| 3         | 550,644                                                 | 93.06                          | 93.13                            |
| 4         | 550,852                                                 | 93.10                          | 93.22                            |
| 5         | 550,907                                                 | 93.06                          | 93.14                            |

indicate that the Bi-LSTM model with various combinations of features achieved better results for Odia POS tagging. The output predicted by the Bi-LSTM model is shown in Table 14. The first row comprises the untagged input text along with its translation and transliteration, and the second row contains the respective labels predicted by the Bi-LSTM system.

7 CONCLUSION AND FUTURE WORK

In this work, we presented various methods for developing POS tagging for the Odia language that is based on statistical and deep learning based approaches. Statistical methods such as the MEMM and CRF, as well as deep learning based techniques like CNN and Bi-LSTM are considered in this research. For example, the MEMM, CRF, Bi-LSTM, and CNN were trained using the ILCI corpus with BIS and UD tagsets. We experimented with various feature set inputs to the MEMM and CRF models, observed the impact of the constructed feature set, and found that the models obtained...
the highest accuracy of 90.97% and 92.08%, respectively. To generate word embedding vectors for the Odia language, we collected raw corpora from several different web resources. These word vectors were integrated into the deep learning model to reduce the number of out-of-vocabulary words. The deep learning based model was comprised of a Bi-LSTM network, a CNN, a CRF layer, character sequence data, and a pre-trained word vector. The Bi-LSTM model with pre-trained word embedding and character sequence feature extracted by a CNN yielded 94.48% accuracy. Moreover, we also achieved an accuracy of 94.58% from the Bi-LSTM model by using the label information. In comparison to other existing studies on the Odia language, the proposed approaches produced more accurate results.

There are significant possible directions for future research. Since our model does not require knowledge that is task- or domain specific, one important direction would be to apply it to data from other domains. In addition, computational linguists dealing with low-resource languages suffer from a lack of resources, such as labeled corpora. In the future, we would like to investigate this possibility to generate more labeled datasets across all domains. Another area of research is to extend this study and apply deep learning to other NLP problems, such as NER, chunking, parsing, and machine translation in low-resource languages.

A APPENDIX

A.1 Confusion Matrix of Different Deep Learning Models

|                      | ADJ | ADP | ADV | AUX | CCONJ | DET | INTJ | NOUN | NUM | PART | PRON | PROPN | PUNCT | SCONJ | SYM | VERB | X |
|----------------------|-----|-----|-----|-----|-------|-----|------|------|-----|------|------|-------|-------|------|-----|------|---|
| **Confusion Matrix** | 4,981 | 19 | 137 | 151 | 51 | 183 | 5 | 34,901 | 114 | 84 | 80 | 796 | 0 | 20 | 582 | 5 |
| **Confusion Matrix** | 12 | 0 | 6 | 0 | 4 | 0 | 4 | 51 | 4,335 | 22 | 0 | 4 | 0 | 0 | 2 | 0 |
| **Confusion Matrix** | 32 | 12 | 9 | 0 | 29 | 30 | 0 | 82 | 14 | 2,485 | 7 | 0 | 4 | 30 | 0 |
| **Confusion Matrix** | 15 | 5 | 15 | 0 | 6 | 140 | 0 | 25 | 2 | 3,152 | 2 | 0 | 7 | 0 | 0 |
| **Confusion Matrix** | 68 | 0 | 0 | 0 | 0 | 0 | 0 | 720 | 23 | 0 | 12 | 7,829 | 0 | 0 | 20 | 6 |
| **Confusion Matrix** | 0 | 0 | 16 | 0 | 71 | 41 | 0 | 12 | 0 | 3 | 2 | 0 | 729 | 0 | 0 | 1 |
| **Confusion Matrix** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Confusion Matrix** | 24 | 3 | 4 | 4 | 5 | 7 | 0 | 301 | 2 | 21 | 0 | 3 | 0 | 0 | 0 | 12,050 |
| **Confusion Matrix** | 0 | 0 | 1 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 119 |

Table A.1. Confusion Matrix of the CharCNN + CNN + CRF Model
### Table A.2. Confusion Matrix of the CharCNN + Bi-LSTM + CRF Model

|       | ADJ  | ADP  | ADV  | AUX  | CCONJ | DET  | INTJ | NOUN | NUM  | PART | PRON | PROPN | PUNCT | SCONJ | SYM  | VERB | X    |
|-------|------|------|------|------|-------|------|------|------|------|------|------|-------|-------|-------|------|------|------|
| ADJ   | 5,382| 8    | 11   | 0    | 0     | 43   | 0    | 668  | 30   | 35   | 13   | 48    | 0     | 0     | 0    | 33   |
| ADP   | 9    | 2,629| 5    | 0    | 1    | 0    | 0    | 125  | 0    | 12   | 7    | 0     | 0     | 0     | 0    | 68   |
| ADV   | 28   | 2    | 1,069| 0    | 11   | 20   | 0    | 194  | 0    | 12   | 7    | 0     | 0     | 10    | 0    | 14   |
| AUX   | 0    | 0    | 0    | 179  | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0     | 0     | 0     | 0    | 68   |
| CCONJ | 0    | 21   | 13   | 0    | 3,689| 24   | 0    | 26   | 0    | 11   | 4    | 0     | 0     | 68    | 0    | 18   |
| DET   | 35   | 0    | 9    | 0    | 9    | 3,665| 0    | 81   | 10   | 19   | 86   | 0     | 33    | 0     | 0    | 0    |
| INTJ  | 0    | 0    | 0    | 0    | 0    | 45   | 17   | 0    | 0    | 0    | 0    | 1     | 0     | 0     | 0    | 0    |
| NOUN  | 741  | 121  | 67   | 0    | 54   | 119  | 5    | 35,711| 93   | 57   | 52   | 576   | 0     | 21    | 0    | 477  |
| NUM   | 21   | 0    | 5    | 0    | 0    | 5    | 0    | 62   | 4,322| 12   | 0    | 9     | 0     | 0     | 0    | 0    |
| PART  | 32   | 15   | 4    | 0    | 16   | 27   | 0    | 64   | 13   | 2,537| 4    | 0     | 0     | 0     | 0    | 22   |
| PRON  | 21   | 3    | 7    | 0    | 7    | 100  | 0    | 35   | 3    | 3,184| 0    | 9     | 0     | 0     | 0    | 0    |
| PROP   | 67   | 0    | 0    | 0    | 0    | 609  | 21   | 0    | 11   | 7,946| 0    | 0     | 20    | 4     | 0    | 0    |
| PUNCT | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 9,905| 0    | 0     | 0     | 0     | 0    | 0    | 0    |
| SCONJ | 0    | 0    | 4    | 0    | 57   | 25   | 0    | 9    | 0    | 2    | 0     | 0     | 0     | 778   | 0    | 0    |
| SYM   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0     | 0     | 0     | 359   | 0    | 0    |
| VERB  | 30   | 2    | 6    | 7    | 7    | 7    | 0    | 299  | 0    | 15   | 0     | 0     | 0     | 0     | 0    | 12,051|
| X     | 0    | 0    | 4    | 0    | 0    | 0    | 76   | 0    | 0    | 19   | 0     | 1     | 0     | 123   | 0    | 0    |

### Table A.3. Confusion Matrix of the CharCNN + CNN Model

|       | ADJ  | ADP  | ADV  | AUX  | CCONJ | DET  | INTJ | NOUN | NUM  | PART | PRON | PROPN | PUNCT | SCONJ | SYM  | VERB | X    |
|-------|------|------|------|------|-------|------|------|------|------|------|------|-------|-------|-------|------|------|------|
| ADJ   | 5,105| 8    | 18   | 0    | 0    | 66   | 0    | 850  | 54   | 57   | 0    | 52    | 0     | 0     | 0    | 0    |
| ADP   | 12   | 1,976| 20   | 0    | 12   | 0    | 0    | 136  | 0    | 17   | 11   | 0     | 0     | 0     | 0    | 0    |
| ADV   | 27   | 18   | 1,052| 0    | 14   | 15   | 0    | 209  | 0    | 10   | 13   | 0     | 21    | 0     | 9    |
| AUX   | 0    | 0    | 168  | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0     | 0     | 0     | 0    |
| CCONJ | 0    | 12   | 17   | 0    | 3,641| 24   | 0    | 50   | 0    | 14   | 11   | 0     | 80    | 0     | 0    |
| DET   | 33   | 0    | 27   | 0    | 18   | 3,569| 0    | 97   | 9    | 36   | 111   | 0     | 47    | 0     | 0    |
| INTJ  | 0    | 0    | 0    | 0    | 0    | 48   | 15   | 0    | 0    | 0    | 0     | 0     | 0     | 0    |
| NOUN  | 890  | 153  | 168  | 1    | 47   | 117  | 0    | 35,264| 95   | 88   | 72   | 650   | 0     | 17    | 0    |
| NUM   | 15   | 0    | 0    | 0    | 0    | 3    | 0    | 55   | 13   | 0    | 0     | 0     | 0     | 0     | 0    |
| PART  | 34   | 13   | 0    | 0    | 23   | 19   | 0    | 90   | 18   | 2,506| 1    | 0     | 0     | 0     | 0    |
| PRON  | 16   | 0    | 22   | 0    | 6    | 89   | 0    | 34   | 0    | 3,194| 0    | 9     | 0     | 0     | 0    |
| PROP   | 69   | 2    | 3    | 0    | 0    | 760  | 17   | 0    | 12   | 7,783| 0    | 0     | 0     | 0     | 0    |
| PUNCT | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 9,906| 0    | 0     | 0     | 0     |
| SCONJ | 0    | 0    | 12   | 0    | 62   | 28   | 0    | 8    | 0    | 0    | 0     | 0     | 0     | 763   | 0    |
| SYM   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0     | 0     | 0     | 359   | 0    |
| VERB  | 26   | 0    | 0    | 17   | 6    | 6    | 0    | 276  | 0    | 11   | 0     | 0     | 0     | 12,082| 0    |
| X     | 0    | 0    | 3    | 0    | 2    | 6    | 0    | 46   | 0    | 11   | 0     | 0     | 0     | 0     | 154  |

ACM Trans. Asian Low-Resour. Lang. Inf. Process., Vol. 22, No. 6, Article 167. Publication date: June 2023.
### Table A.4. Confusion Matrix of the CharCNN + Bi-LSTM Model

|   | ADJ | ADP | ADV | AUX | CON | DET | INTJ | NOUN | NUM | PART | PRON | PROPN | PUNCT | SCONJ | SYM | VERB | X   |
|---|-----|-----|-----|-----|-----|-----|------|------|-----|------|------|-------|-------|------|-----|-----|-----|
| ADJ| 5,384 | 7  | 14  | 0   | 28  | 684 | 23   | 17   | 51  | 0    | 0    | 27    | 0     |      |     |     |     |
| ADP| 6    | 2,038 | 10  | 0   | 6   | 0   | 108  | 0    | 11  | 8    | 0    | 0     | 0     | 0    | 0   | 0   | 0   |
| ADV| 25   | 9   | 1,118 | 10  | 17  | 0   | 173  | 0    | 12  | 9    | 0    | 7     | 0     | 0    | 0   | 0   | 0   |
| AUX| 0    | 0   | 0    | 201 | 0   | 0   | 0    | 0    | 0   | 0    | 0    | 0     | 0     | 0    | 47  | 0   | 0   |
| CON| 6    | 14  | 3,724 | 19  | 0   | 34  | 0    | 6    | 5   | 0    | 0    | 52    | 0     | 12   | 0   | 0   | 0   |
| DET| 31   | 0   | 12   | 0   | 12  | 3,636 | 101  | 12   | 22  | 90   | 0    | 31    | 0     | 0    | 0   | 0   | 0   |
| INTJ| 0    | 0    | 0    | 0   | 0   | 0    | 50   | 13   | 0   | 0    | 0    | 0    | 0     | 0    | 0   | 0   | 0   |
| NOUN| 582  | 74  | 84   | 0   | 34  | 77   | 0    | 36,220 | 81  | 41   | 44   | 477   | 0     | 16   | 0   | 362 | 9   |
| NUM| 13   | 0   | 4    | 0   | 4   | 0    | 50   | 4,356 | 11  | 0    | 0    | 0    | 0     | 0    | 0   | 0   | 0   |
| PART| 25   | 6   | 4    | 0   | 18  | 15   | 0    | 74   | 11   | 2,553 | 0    | 0    | 0     | 0    | 24  | 3   |     |
| PRON| 14   | 0   | 9    | 0   | 5   | 72   | 0    | 37   | 1    | 3,226 | 0    | 6    | 0     | 0    | 0   | 0   |     |
| PROPN| 65   | 0   | 0    | 0   | 0   | 694  | 19   | 0    | 9    | 7,870 | 0    | 0    | 0     | 0    | 17  | 3   |     |
| PUNCT| 0    | 0   | 0    | 0   | 0   | 0    | 0    | 0    | 0    | 0    | 0    | 9,906 | 0     | 0    | 0   | 0   |     |
| SCONJ| 0    | 0   | 6    | 0   | 62  | 15   | 0    | 7    | 0    | 0    | 0    | 0    | 0     | 0    | 785 | 0   | 0   |
| SYM| 0    | 0   | 0    | 0   | 0   | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0     | 0    | 359 | 0   | 0   |
| VERB| 26   | 0   | 8    | 15  | 6   | 5    | 0    | 294  | 0    | 17  | 2    | 0    | 0     | 0    | 12,057 | 0  |     |

ACM Trans. Asian Low-Resour. Lang. Inf. Process., Vol. 22, No. 6, Article 167. Publication date: June 2023.
[17] Zellig Harris. 1962. *String Analysis of Language Structure*. Mouton & Co., The Hague.

[18] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[19] K. S. Gokul Krishnan, A. Pooja, M. Anand Kumar, and K. P. Soman. 2017. Character based bidirectional LSTM for disambiguating tamil part-of-speech categories. *International Journal of Control Theory and Applications* 229 (2017), 235.

[20] Anoop Kunchukuttan, Divyanshu Kakwani, Satish Golla, Gokul NC, Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. AHBharat-IndicNLP corpus: Monolingual corpora and word embeddings for Indic languages. *arXiv preprint arXiv:2005.00085* (2020).

[21] John Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML’01)*.

[22] Wang Ling, Chris Dyer, Alan W. Black, Ramón Fernández, Silvio Amir, Luis Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1520–1530.

[23] Xueze Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1064–1074.

[24] Ruslan Mitkov. 2022. *The Oxford Handbook of Computational Linguistics*. Oxford University Press.

[25] Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC’16)*. 1659–1666.

[26] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajic, Christopher D. Manning, Sampo Pyysalo, Jenna Kanerva, Anna Missilä, Veronika Laippala, and Filip Ginter. 2015. Universal dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*. 4034–4043.

[27] Farhad Nooralahzadeh, Caroline Brun, and Claude Roux. 2014. Part of speech tagging for french social media data. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers (COLING’14)*. 1764–1772.

[28] Atul Ku Ojha, Pitambar Behera, Srishti Singh, and Girish N. Jha. 2015. Training & evaluation of POS taggers in Indo-Aryan languages: A case of Hindi, Odia and Bhojpuri. In *Proceedings of the 7th Conference on Language Resources and Technology: Conference and Workshops*. Human Language Technologies as a Challenge for Computer Science and Linguistics. 524–529.

[29] Shantipriya Parida, Satyajit Sahoo, and Srikanta Patnaik. 2020. OdiEnCorp 2.0: Odia-English parallel corpus for machine translation. In *Smart Intelligent Computing and Applications*. Springer, 495–504.

[30] Shantipriya Parida, Satyajit Sahoo, and Srikanta Patnaik. 2020. OdiEnCorp 2.0: Odia-English parallel corpus for machine translation. In *Proceedings of the WILDRE5–5th Workshop on Indian Language Data: Resources and Evaluation*. 14–19.

[31] Sagarika Pattnaik, Ajit Kumar Nayak, and Srikanta Patnaik. 2020. A semi-supervised learning of HMM to build a POS tagger for a low resource language. *Journal of Information and Communication Convergence Engineering* 16, 4 (2020), 167:23.

[32] Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC’12)*. 2089–2096.

[33] Barbara Plank, Anders Seggaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 412–418.

[34] Ankur Priyadarshi and Sujan Kumar Saha. 2020. Towards the first Maithili part of speech tagging: Resource creation and system development. *Computer Speech & Language* 62 (2020), 101054.

[35] Ankur Priyadarshi and Sujan Kumar Saha. 2022. A study on the performance of recurrent neural network based models in Maithili part of speech tagging. *ACM Transactions on Asian and Low-Resource Language Information Processing* 22, 2 (2022), Article 32, 16 pages.

[36] Sampo Pyysalo, Jenna Kanerva, Anna Missilä, Veronika Laippala, and Filip Ginter. 2015. Universal dependencies for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Computational Linguistics: Technical Papers (COLING’14)*. arXiv:1508.01991.

[37] Goutham Ramesh, Sumanth Doddapaneni, Aravinth Bheemaraj, Mayank Jhonputra, A. K. Raghavan, Ajitesh Sharma, Sujit Sahoo, et al. 2022. Samanantar: The largest publicly available parallel corpora collection for 11 Indic languages. *Transactions of the Association for Computational Linguistics* 10 (2022), 145–162.

[38] Manish Shrivastava and Pushpakh Bhattacharyya. 2008. Hindi POS tagger using naive stemming: Harnessing morphological information without extensive linguistic knowledge. In *Proceedings of the International Conference on NLP (ICON’08)*.
[39] N. M. Suraksha, K. Reshma, and K. M. Shiva Kumar. 2017. Part-of-speech tagging and parsing of Kannada text using Conditional Random Fields (CRFs). In Proceedings of the 2017 International Conference on Intelligent Computing and Control (I2C2’17). IEEE, Los Alamitos, CA, 1–5.

[40] Juhi Tandon, Himani Chaudhry, Riyaz Ahmad Bhat, and Dipti Misra Sharma. 2016. Conversion from Paninian karakas to universal dependencies for Hindi dependency treebank. In Proceedings of the 10th Linguistic Annotation Workshop Held in Conjunction with ACL 2016 (LAW’X’16). 141–150.

[41] Sunita Warjri, Partha Pakray, Saralin Lyngdoh, and Arnab Kumar Maji. 2019. Identification of POS tag for Khasi language based on hidden Markov model POS tagger. Computación y Sistemas 23, 3 (2019), 795–802.

[42] Sunita Warjri, Partha Pakray, Saralin A. Lyngdoh, and Arnab Kumar Maji. 2021. Part-of-speech (POS) tagging using deep learning-based approaches on the designed Khasi POS corpus. ACM Transactions on Asian and Low-Resource Language Information Processing 21, 3 (2021), 1–24.

[43] Yingwei Xin, Ethan Hart, Vibhuti Mahajan, and Jean David Ruvini. 2018. Learning better internal structure of words for sequence labeling. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2584–2593.

[44] Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages.

Received 22 July 2022; revised 17 March 2023; accepted 18 March 2023.