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

TUSARKANTA DALAI, Department of Computer Science and Engineering, NIT Rourkela, India
TAPAS KUMAR MISHRA, Department of Computer Science and Engineering, NIT Rourkela, India
PANKAJ K SA, Department of Computer Science and Engineering, NIT Rourkela, India

Automatic Part-of-speech (POS) tagging is a preprocessing step of many natural language processing (NLP) tasks such as name entity recognition (NER), speech processing, information extraction, word sense disambiguation, and machine translation. It is a process of assigning the grammatical class label to each word in a sentence with their respective part-of-speech based on morphological and contextual language information. It has already gained a promising result in English and European languages, but in Indian languages, particularly in Odia language, it is not yet well explored because of the lack of supporting tools, resources, and morphological richness of language. Unfortunately, we were unable to locate an open source POS tagger for Odia, and only a handful of attempts have been made to develop POS taggers for Odia language. Odia is one of the official languages of India that comes under Indo Aryan language family, spoken in Odisha and parts of West Bengal, Jharkhand and Chhattisgarh, with around 45 million native speakers. The main contribution of this research work is to present a conditional random field (CRF) and deep learning-based approaches (CNN and Bidirectional Long Short-Term Memory) to develop Odia part-of-speech tagger. We used a publicly accessible corpus as part of the Indian Languages Corpora Initiative (ILCI) phase-II project, which was initiated by the Ministry of Electronics and Information Technology (MeitY) Government of India. The dataset is annotated with the Bureau of Indian Standards (BIS) tagset. However, most of the languages around the globe have used the dataset annotated with Universal Dependencies (UD) tagset. Hence, to maintain uniformity Odia dataset should use the same tagset. So we have constructed a simple mapping from BIS tagset to UD tagset. CRF, Bi-LSTM, and CNN models are trained using ILCI corpus with BIS and UD tagset. We experimented with various feature set inputs to the CRF model, observed the impact of constructed feature set. The deep learning-based model includes Bi-LSTM network, CNN network, CRF layer, character sequence information, and pre-trained word vector. Character sequence information was extracted by using convolutional neural network (CNN) and Bi-LSTM network. Six different combinations of neural sequence labelling models are implemented, and their performance measures are investigated. It has been observed that Bi-LSTM model with character sequence feature and pre-trained word vector achieved a significant state-of-the-art result.

Additional Key Words and Phrases: Part of speech (POS), Conditional random field (CRF), Deep learning, Word embedding

1 INTRODUCTION

POS tagging is one of the sequence labelling tasks that involves assigning a grammatical category label to each word based on linguistic and contextual information [3]. 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 the its parts of speech including noun, pronoun, adjective, verb, adverb and so on. POS tags are useful 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 pre-processing step for many natural language processing (NLP) frameworks. These NLP frameworks include speech recognition, sentiment analysis, named entity recognition, question answering, word sense disambiguation, and chunking [19]. Determining the grammatical class of a language is challenging because it is difficult to assign tags to each word of a sentence when some words have more than one grammatical POS label. In addition syntax or semantics of many languages may differ, and the grammatical structure of many languages varies depending on the context in which it is used. In English, European languages, and other South Asian languages, the problem of POS tagging has been thoroughly investigated. However, there is still a need for research on Indian languages, particularly the Odia language, as it is difficult 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.

India 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. Previously, it was a Scheduled Language under the Eighth Schedule of the Indian Constitution. Odia language originated in the tenth century and later the language was changed under the influence of the Sanskrit language in the sixteenth and seventeenth 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 half-furnished. Considering the NLP situation in Indian languages, they are poor so 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. So, the Odia language needs to be explored from the language processing prospective. Therefore, we researched into Odia POS labelling task from an NLP perspective. Some research articles on NLP have recently been introduced in Odia, which will be discussed in the following units below.

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 un-tagged 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 standard corpora in many languages; therefore, datasets or corpora are often a challenge. So, many researchers have constructed their own corpus. For the purpose of our experiment, we have collected publicly available corpus through the website of the Technology development for Indian languages (TDIL). This was done as part of the project Indian Languages Corpora Initiative phase-II, 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 BIS tagset contains 38 tags. The tagsets consist of a collection of tags representing the grammatical class information of a specific language. Tags that describe grammatical categories or POS, such as nouns, verbs, adjectives, pronouns, etc. 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 [4] . Bureau of Indian Standards (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 tagset, it was necessary that Odia dataset also uses the same tagset to maintain uniformity. Therefore we have also used UD tagset for tagging the corpus by using a simple mapping from BIS tagset.

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. Therefore, the system should be able to automatically produce the output as tagged terms from the given input text using the technique. We employed supervised learning method based on conditional random fields (CRF), deep learning approaches like CNN and bidirectional long short-term memory (Bi-LSTM), and a combination of deep learning methods with CRF.

The following are the main contributions of this article:

(1) Design the Odia POS corpus using UD tagset.
(2) Develop Odia POS tagger using CRF.
(3) Generate word embedding vector through fastText word embedding method for Odia POS tagging task.
(4) Develop the first deep learning-based Odia POS tagger using CNN and Bi-LSTM.

The remaining parts of this article are organized in the following manner: Section 2 describes literature survey on POS tagging; Section 3 presents the Odia POS dataset and UD tagset for 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 the conclusions and discuss future works in these dimensions.

2 RELATED WORK

This section presents the existing related work of POS tagging on different languages and presently available work on different approaches. Harris [13] was the first to explore POS tagging, and he used a rule-based approach to develop 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. In the nineties, statistical-based machine learning methods were quite popular. Cutting et al. [5] have proposed a statistical-based Hidden Markov Model (HMM) to develop a POS tagger using Brown Corpus with 50,000 tagged words. In 2001 Lafferty et al. [17] introduced another statistical model for POS tagging called conditional random field (CRF), and they observed that CRF performs better than other statistical models in POS tagging tasks. Later neural network-based POS tagging was introduced. It includes CNN, LSTM, Bi-LSTM, RNN, GRU, etc., and these methods are recently used for sequence labelling tasks. In 1994, Helmut [30] introduced a neural network-based POS tagging.

Here we discuss some of the recent research work reported on POS tagging in different languages. In paper [11] the authors 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 Bi-LSTM model was proposed by W. Ling et al.[18]. Here Bi-LSTM network was used to compose the characters to construct character representation of words, and it achieved good accuracy in morphologically rich languages. In [27] the authors developed a multilingual POS tagging with Bi-LSTM. The model was experimented on 22 different languages and achieved a state-of-art performance. This model was performed well by using word embedding and character embedding features. Huang et al. [14] proposed a Bi-LSTM-CRF model for sequence labelling tasks. Here LSTM network is composed with a CRF layer and improves the model's performance. This model has produced a state-of-art result on POS, NER, and chunking datasets. Shrivastava and Bhattacharyya [31] proposed HMM-based tagger. They achieved an accuracy of 93.12%. And they used a corpus of size 81,751 tokens for the experiment. Alam et al. [1] developed a POS tagger for Bangla language using LSTM network and CRF layer with word embedding feature. This model was implemented using publicly available corpus achieved an accuracy 86.0%. In [15] the author presented a POS tagger for Tamil using character-based Bi-LSTM. They got an accuracy of 86.45%. Priyadarshi and Saha [28] proposed a Maithili POS tagger using CRF and word embedding. They have used an annotated corpus of 52,190 words and achieved an accuracy of 85.88%. Warjri et al. [34] proposed a Khasi POS tagger using deep learning-based approach on designed Khasi corpus. They achieved an accuracy of 96.98% by using Bi-lSTM with CRF layer.

A few efforts have been proposed for the Odia language to build POS tagger. [6] Das and Patnaik 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 labelling the data. Voting-based selection rule to obtain the output and forward propagation method was used for error correction. The Odia POS tagger using artificial neural network (ANN) obtained an accuracy 81%. In [7] Das et al. presented an improved Odia POS tagger using support vector machine, and it achieved 82% training accuracy. They also used only five tags for manually labelling the dataset having 10k of tokens. Lexicon, NER tags and word suffixes were considered as feature set and fed to the SVM POS tagger. In [22] Ojha et al. reported support vector machine (SVM) and conditional random field (CRF) based POS tagger for Odia language achieved an accuracy 91.3% and 85.5%, respectively. ILCI dataset with 90k and 2k tokens was used for training and testing and unigram features were considered during the training period.

In this work [25], Pattnaik et al. developed Hidden Markov Model (HMM) based Odia POS tagger and achieved an accuracy 91.53% . They implemented the tagger using 0.2 million tokens and 11 tags to annotate the tagset.

3 ODIA POS CORPUS

In this section, we have provided a brief discussion on the dataset and tagset that were utilised in the development of the Odia POS tagger.

3.1 Universal dependency POS tagset for Odia language

This subsection presents a brief discussion on UD tagset for Odia part of speech and a mapping from BIS tagset to UD tagset. A tagset contains unique tags which are used to label each word in a text. Universal Dependency (UD) is a framework provides consistent annotation of part of speech across an extensive collection of different human languages [20] [21]. 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 part-of-speech tags [26], Stanford dependencies [9] [8], and the Interset interlingua for morphosyntactic tagsets [35]. 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 natural language processing to human languages. UD bears a striking resemblance to the conventional parts of speech. UD is somewhat similar to the conventional parts of speech, and it recognises 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 Ekbal et al. [12] (2007) developed Bengali POS Tagger using 26-tags tagset, Suraksha et al. [32] (2017) presented a Kannada tagger contains 19 tags in tagset, Dhanalakshmi et al. [10] (2009) defined a Tamil POS tagset containing 32-tags, and Warjiri et al. [33] (2019)introduce a Tagger for Khasi language has 54 labels. The use of different tagsets makes it difficult in the comparative study. To resolve the issue, various tagset schemes are developed. However, since all other datasets of various language around the globe have been used the corpus annotated with Universal Dependencies 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 Odia dataset annotated with UD POS tagset, it is necessary that Odia dataset also uses the same tagset to maintain uniformity. The collected ILCI corpus was annotated using the Bureau of Indian Standards (BIS) tagset. So, We constructed a simple mapping of tags from BIS tagset to UD tags based on its respective category. The mapping is given in Table 1.

Mapping from Odia POS BIS tagset to UD POS tagset is relatively straightforward and easy to implement. Many of the tags mapped directly from BIS tagset to UD tagset, some of are noun, pronoun, verb, adverb and adjective and so on. A couple of tags made difficult to map because there is no proper class define in UD tagset, like demonstratives, quantifiers and echo-words. General quantifiers do not indicate any precise quantity and it occurs as intensifier in Odia language. The most common general quantifiers used in Odia language are adhika (much), kebala (only), Ahuri (more) etc. So general quantifiers mapped to intensifier, that is particles in UD tagset. Other two quantifiers in BIS tagset is directly mapped to numeral in UD tagset. BIS tagset do not have determiners and adpositions as a separate category but they have demonstratives and postposition which do not appear in UD. So he label postposition and demonstrative are grouped as adposition and determiner tag in UD tagset respectively. Echo word is one of POS class in BIS tagset but UD POS tagset does not have echo word category. In Odia sentence echo words also used as noun, adjective, adverb etc. depends upon its use in a sentence. From the below examples, we observed that, echo word can have mapped to different POS categories based upon syntactic

structure of Odia language. TDIL Odia POS dataset contains 268 echo words. So we manually mapped the echo words to its respective group.

Example:1 ବାସମତୀ ଧାନ ସେଥିମଧ୍ୟରୁ ଅଧିକ ଜଣାଶୁଣା।

Transliteration: bAsamatI dhAna sethimadhYaru adhika jaNAshuNA.

Here the word ଜଣାଶୁଣା(jaNAshuNA) is a noun in sentence.

Example:2 ସେ ଭାରତୀୟ ସାହିତ୍ୟରେ ଜଣେ ଜଣାଶୁଣା ବ୍ୟକ୍ତିତ୍ୱ ।

Transliteration: se bhAratIYa sAhitYare jaNe jaNAshuNA bYaktitva.

In example:2 word ଜଣାଶୁଣା(jaNAshuNA) is an adjective.

Example:3 ଉଦ୍ୟାନକୁ ସଫାକରି ଅନାବନା ଘାସକୁ ବାହାର କରିଦିଅନ୍ତୁ ।

Transliteration: udYAnaku saphAkari anAbanA ghAsaku bAhAra karidiantu.

Here the word ଅନାବନା(anAbanA) represents an adjective in Odia sentence.

3.2 Corpus Information

Under the project Indian Languages Corpora Initiative phase-II, funded by the Ministry of Electronics and Information Technology, Government of India, Jawaharlal Nehru University, New Delhi, we accessed two publically accessible corpus 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. Both the corpus having approximately 51,150 sentences. In the corpus, we identified that some special unwanted characters were associated to 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. There were also empty lines, untagged sentences, and redundant space characters in the corpus. Therefore, it is necessary to do data processing prior to utilizing the data to train the model. After data cleansing, we obtained 48,000 Odia POS-tagged sentences. From the available corpus, we have split the dataset to Training, validation and test set according to 70%, 15%, and 15% respectively. Deatails of the dataset shown in Table 2. The distribution of the ILCI corpus over the UD tagset is presented in reference Table 3.

4 SYSTEM MODEL

The statistical and deep learning models that were utilised to construct the Odia POS tagger have been explained in this section of the article.

4.1 Odia POS Tagger developed using statistical method

Several existing models are present for sequence labelling tasks like POS tagging, NER and chunking etc. HMM, MEMM and CRF are the stochastic methods used to solve sequence labelling problems. However, from the literature survey, we knew that conditional random field performs better than HMM and MEMM. CRF model is the most commonly used sequencing model for labelling tasks. Moreover, the CRF model makes it easy to examine the word before and after entity by its undirected graph property. CRFs are the type of discriminative model that learns the conditional probability distribution. Conditional random fields is an undirected graph-based method where node represents input or observation sequence x corresponds to label, or output sequence y. This

Table 1. **Mapping from BIS tagset to 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

Table 2. **Odia POS corpus details**

Collected Corpus	Domain	NO. of Sentences	NO. of tokens
Odia Monolingual Text Corpus ILCI-II	Agriculture, Entertainment	19,000	2,76,916
Hindi-Odia Text Corpus ILCI-II	Agriculture, Entertainment	31,500	3,94,438

Table 3. **Distribution of ILCI Odia POS corpus over UD tagset**

Tag	Description	Count
NOUN	noun	258726
VERB	verb	86369
PUNCT	punctuation	68322
PROPN	proper noun	58866
ADJ	adjective	43502
NUM	numeral	28904
CCONJ	coordinating conjunction	26170
PART	particle	26048
PRON	pronoun	22426
DET	determiner	19337
ADP	adposition	13255
ADV	adverb	9387
SCONJ	subordinating conjunction	6485
X	other	1571
AUX	auxiliary	1397
INTJ	interjection	444
SYM	symbol	145

model aims to find the output label or tag y that maximises 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 = argmax P(x|y). Let the observation word sequence $x = (x_1x_2x_3\ldots..x_n)$ and corresponding tag sequence $y = (y_1y_2y_3\ldots\ldots\ldots y_n)$.

Mathematically, The conditional probability chain structure of the crf model can be expressed as

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

Where λ is the weight associated with each distinct features K, and the model depends on the set of features. $f_k\left(y_{t-1}, y_t, x\right)$ denotes feature function whose value is represented in binary. It may be '0' or '1'. Z_x is used for normalization and sum of probability of all state sequences is one. Z_x can be represented as

$$Z(x) = \sum_{y} \exp\left(\sum_{t=1}^{n}\sum_{k=1}^{K} \lambda_k \cdot f_k\left(y_{t-1}, y_t, x\right)\right) \tag{2}$$

All the details of feature set, we have explained in the result section.

4.2 Odia POS Tagger developed using deep learning approach

In this subsection, we discuss the development of 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 have summarised the steps involved in developing the deep learning model for Odia based on POS tagging below. Figure 2 illustrates the overall architecture of the Bi-LSTM model used for Odia POS tagging.

(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 fastText word embedding method.
(4) We obtained character level embedding and word-level embedding are concatenated and fed into a fully connected neural network.
(5) The output of the previous step is taken input to the word sequence layer, and final word embedding is fed into the word sequence layer.
(6) Finally, last hidden layer's output of the word sequence layer was taken 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 labelling 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. In order 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 in order to capture the key characteristics of nearby inputs. The final stage of 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 in order to maintain the information that is specific to each word. Moreover LSTM models are also applied for character-level information extraction. Bidirectional LSTM or Bi-LSTM is used to record sequence information from left to right (forward LSTM) and right to left (backword 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, concatenate the two final hidden states of the forward and backward LSTM to get a vector of fixed size representing a word token.

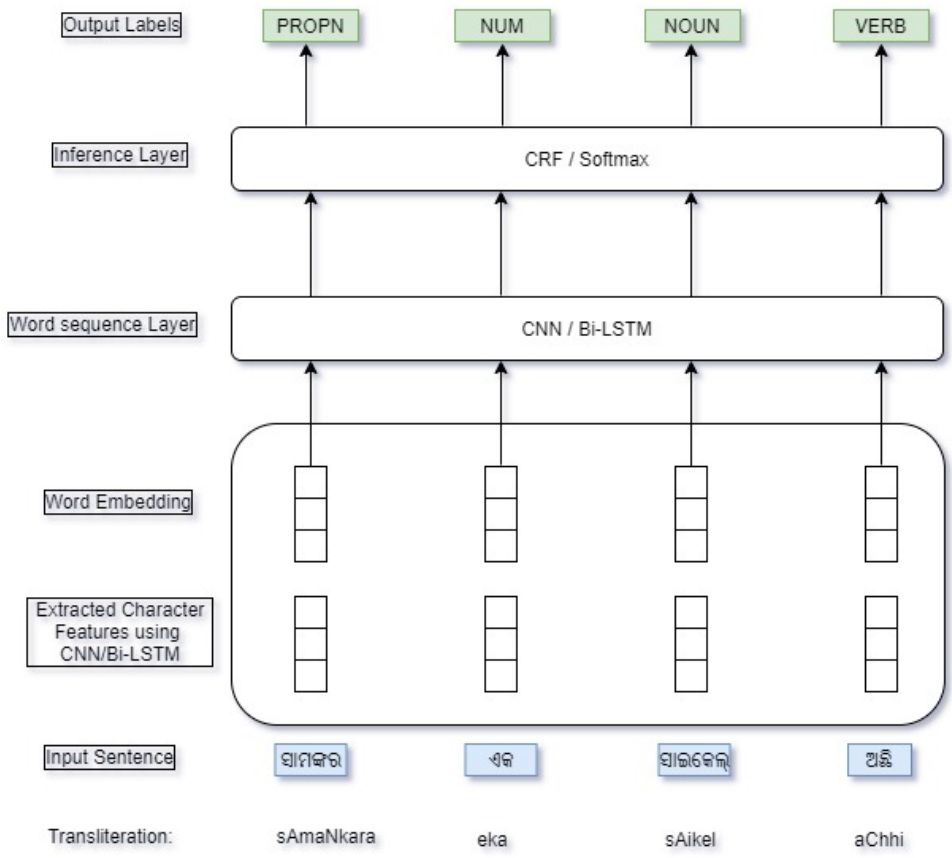

Fig. 1. Architecture of Odia POS tagger using deep learning-based model

4.2.2 ***Word Embedding.*** In many NLP tasks, nearby words play a crucial role. In addition, the performance of sequence labelling tasks, such as POS tagging, NER, and Chunking, is dependent on the surrounding terms. In order to accomplish this, we require word embedding for distributed word representation, which maps words into a low-dimensional vector space. Word vector has the advantage of capturing relationships between words. NLP offers a variety of strategies for word embedding. In our work, we used fastText [2], one of the embedding techniques developed by Facebook. We train the fastText model because each word is represented as an n-gram of characters, it preserves subword information, and it can compute valid word embeddings for out-of-vocabulary terms. Therefore, it 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 number of resources [23] [24] [29] [16] and Odia-specific websites. 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

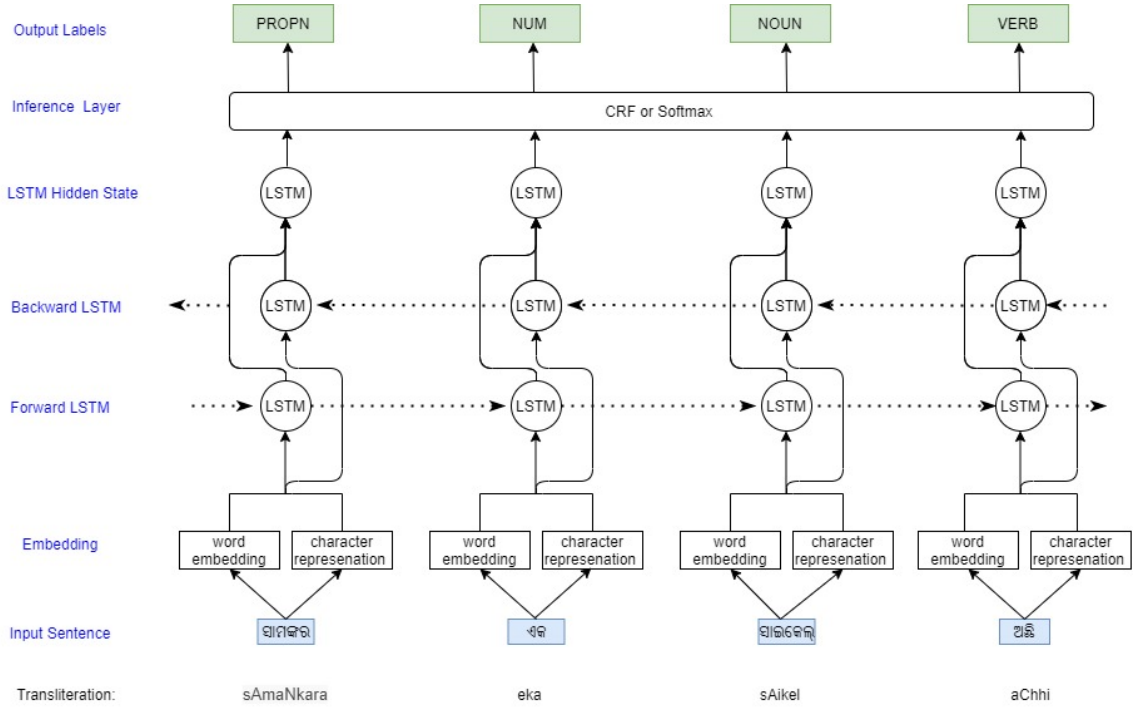

Fig. 2. Architecture of Odia POS tagger using Bi-LSTM model

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. Run time and the quality of the trained model are both impacted by a number of different parameter selections. After conducting the experiment with a number of different vector dimension values (ranging 100, 150, 200, 250, and 300), we observed that value of 200 worked well in our experiment. We also experimented with different window sizes and determined the best one to be 5. The word vectors are generated after training the fastText model.

4.2.3 ***Word sequence layer.*** Similarly to 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. CNN uses a sliding window to collect local characteristics and a max-pooling to encode the aggregated character sequence. Immediately following each CNN layer, batch normalization and dropout are applied.

Bi-LSTM, one of the types of recurrent neural network, was also employed at the word sequence layer (RNN). In order to overcome the limits of a 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 labelling tasks. On the other hand, simple LSTM networks merely take input from the past and are unaware of the context of the future. Bi-LSTM are 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 is 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 labelling tasks. CRF layer acquires knowledge from successive labels by employing 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 labelling tasks [18].

5 EXPERIMENTAL RESULT

5.1 CRF model result

In this subsection, we provided a brief description of training and testing using the CRF model and established a relevant feature set for interpreting their impact on the Odia POS system. We investigated the efficacy of our approach using the ILCI Odia dataset. Using ILCI Odia corpus, we evaluated the performance of the CRF model for Odia POS tagging with the BIS and UD tagsets. Our CRF model is trained using LBFGS (Limited-Memory Broyden-Fletcher-Goldfarb-Shanno), a quasi-Newton approach and this algorithm is used for large-scale numerical optimization problems. We used CRF++-0.58.8 toolkit to prepare the model and the toolkit runs with the CRF algorithm. For our experiment, we first defined the feature set to train the model using the feature templet of the CRF++ toolkit. To improve the overall performance of any machine learning task, it is necessary to have the 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 utilised to guide CRF in the generation of features.

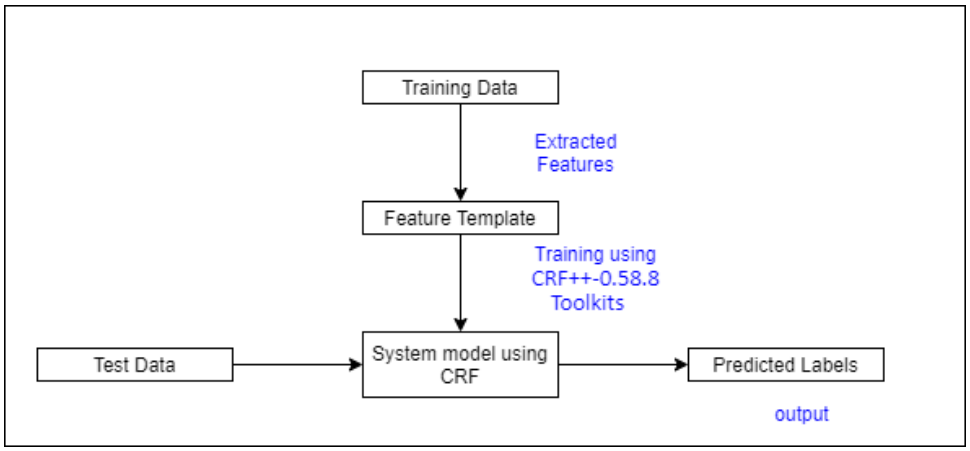

Fig. 3. Flow Chart for CRF approach

The information regarding the previous word, the next word, the current word, as well as the tag of the current word, is contained within the defined feature templates. According to the research, information about the root

word and its morphological inflections is quite helpful for the POS labelling task. The suffix and prefix information was then added to the feature set. For the development of the CRF classifier, a number of experiments were conducted using various combinations of individual features to identify the feature set with the highest accuracy. In Table 4, we have provided a summary of the many possible feature combinations that were utilised in our experiment. In the construction of the word "features," a variety of different combinations of preceding and following words were utilised. 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. Figure 3 presents the flow chart of POS tagger using CRF++-0.58.8 toolkit.

CRF models are trained using the regularisation technique L2 to generate all features needed to calculate probabilities at the time of tagging. We trained the 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 cut-off threshold for the features, whereas 'c' specifies the balance between over-fitting and under-fitting. We evaluated the efficacy of the CRF tagger using approximately 2,76,900 tokens from Corpus I and about 6,711,000 tokens from combined Corpora I and II on BIS and UD tagsets. In Tabel 4, we summarized the combination of features represented by feature id and corresponding accuracy. Here we can observe from Table 5 that the accuracy increases as the corpus size grows. In these experiments, CRF tagger with defined feature sets obtained a maximum accuracy of 92.08%.

Table 4. **Description of 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 Current 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

5.2 Deep learning model results

This subsection provides a brief explanation of the experimental outcomes of deep learning-based technique for the Odia POS tagger. We used the same corpus for this experiment as we did in our initial CRF-model-based approach. Approximately 2,76,900 tagged tokens (36505 unique words) and 6,71,000 tagged tokens (75394 unique words) are fed into the systems. For our implementation, 70% of the corpus was used for training, 15% for validation, and the remaining 15% was 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 hyper-parameter values that gave the best results on the validation set. The selected hyper-parameters that were used in our experiments are summarised in Table 6. To initialize

Table 5. **Accuracy of CRF based tagger using different features**

Sl No.	Dataset	Feature details	Test accuracy on BIS tagset	Test accuracy on UD tagset
1		f_1	87.56	89.24
2		f_2	87.02	88.55
3	1,93,840 tokens for training	f_3	87.33	88.82
4	41,537 tokens for training	$f_1 + f_4$	87.74	89.49
5		$f_1 + f_4 + f_5 + f_6 + f_7$	87.98	89.65
6		$f_1 + f_4 + f_5 + f_6 + f_7 + f_8 + f_9$	88.14	89.97
1		f_1	89.76	91.42
2		f_2	88.87	90.15
3	4,69,950 tokens for training	f_3	89.01	90.59
4	1,00,704 tokens for testing	$f_1 + f_4$	89.77	91.50
5		$f_1 + f_4 + f_5 + f_6 + f_7$	90.41	91.83
6		$f_1 + f_4 + f_5 + f_6 + f_7 + f_8 + f_9$	90.45	92.08

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 in order to optimize the parameters 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 in order to enhance the performance of the model on the unseen data.

To estimate robustness of models with respect to features, we train on six different combination on both of the model including Bi-LSTM, CNN, WE + Bi-LSTM, WE + CNN, WE + CharCNN + Bi-LSTM, WE + CharCNN + CNN, WE + CharBi-LSTM + Bi-LSTM, 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 three Softmax-based models with distinct representations of character and word sequences. Here token accuracy is used to evaluate the system performance. The results of Odia POS tagger on BIS and UD tagset using the deep learning-based technique are presented in Table 7. In Table 7, CharCNN indicates models that use CNN for character sequence information, CharBi-LSTM indicates models that use Bi-LSTM for character sequence information, and WE indicate word embedding information. Among the various combination of features, the results demonstrate that the Bi-LSTM with pre-trained word vector and character information using CNN provided the highest accuracy of 94.58 percent. Also, we observed that models with character-level information perform better than models without character-level information. It is found that the accuracy of deep learning-based approaches for Odia POS tagger on BIS and UD tagsets outperforms the accuracy of existing work.

Table 6. **Hyper-parameters for deep learning model**

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

5.3 Comparison with existing POS Tagger in Odia Language

We compared the performance of our CRF, CNN and Bi-LSTM model with the results of previous work on Odia POS tagging. Table 8 presents the results of other researchers and the results of our model. From the other researchers' results, we can see that Pattnaik et al. [25] presented Odia POS tagger on one of statistical model for sequence labelling task, that is HMM model gives the highest accuracy 91.53. On the other hand, the proposed POS tagger on the CRF model gives 92.08% accuracy using surrounding word information prefix and suffix information features to increase the accuracy. We observed from Table 8, result achieved by the CNN and Bi-LSTM model using BIS and UD tagsets on ILCI corpus gives better accuracy compared to the existing research work.

6 ERROR ANALYSIS

An examination of the nature of improperly classified tags is essential to POS tagging. We therefore performed an error analysis on the predicted POS tags using the CNN and Bi-LSTM models. Therefore, we conducted an error analysis for the predicted POS tags by the CNN and Bi-LSTM models. Figure 4 and Figure 5 presents the confusion matrix of Bi-LSTM and CNN for Odia POS tagger respectively. Table 9 presents the misclassification classes for CNN and Bi-LSTM models derived from the respective confusion matrices. Here, we considered the top 5 misclassification labels and calculated their number of instances and error rate.

From the confusion matrices,we observed that Bi-LSTM model predicts the label more accurately for the majority of classes than the CNN model, whereas 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. The word for which Bi-LSTM and CNN model disagree on the predicted tag, Bi-LSTM correctly labels 63.68% of the words whereas CNN correctly labels 29.97% accuracy of the words. Therefore, we took into consideration the label information as a feature in which the models give the same

Table 7. **Comparsion of tagging perfermance on POS tasks for various neural network model**

Dataset	Model	Test accuracy on BIS tagset	Test accuracy on UD tagset
1,93,840 tokens for training 41,537 tokens for validation	CNN	87.47	88.85
	WE +CNN	88.72	90.14
	WE + CharCNN + CNN	89.31	90.81
	WE + CharBi-LSTM + CNN	89.13	90.67
	WE + CharCNN + CNN + CRF	89.01	90.44
	WE + CharBi-LSTM + CNN + CRF	88.91	90.34
4,69,950 tokens for training 1,00,704 tokens for validation	CNN	89.32	91.02
	WE +CNN	90.74	92.31
	WE + CharCNN + CNN	91.29	**92.77**
	WE + CharBi-LSTM + CNN	91.19	92.60
	WE + CharCNN + CNN + CRF	91.02	92.43
	WE + CharBi-LSTM + CNN + CRF	90.89	92.35
1,93,840 tokens for training 41,537 tokens for validation	Bi-LSTM	88.97	90.24
	WE +Bi-LSTM	90.14	91.65
	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
4,69,950 tokens for training 1,00,704 tokens for validation	Bi-LSTM	90.75	92.65
	WE +Bi-LSTM	92.06	94.02
	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

predicted tags, and we trained the model with this additional information using Bi-LSTM and CNN in order to improve the performance of the Odia POS tagger. Here we concatenated additional tag information of training and validation data with word embedding vector and the updated word embedding vector is input the Bi-LSTM and CNN model. Let Z_w be the original word embedding produced by fastText for word w. We created 18 different

Table 8. **Comparsion of tagging perfermance on POS tasks for various neural network model**

	Model	Dataset	tagset	Accuracy
Previous works	ANN [6]	NA	5tags	81%
	SVM [7]	10,000	5 tags	82%
	CRF [22]	training:90k,validation:2k	BIS tagset (38 tags)	82-86%
	HMM [25]	0.2 million tokens	11 tags	91.53
Our results	CRF	Training:4,69,950 ,validation:1,00,704	UD tagset (17 tags)	92.08%
	CRF	Training:4,69,950 ,validation:1,00,704	BIS tagset (38 tags)	90.45%
	CNN	Training:4,69,950 ,validation:1,00,704	UD tagset (17 tags)	92.74%
	CNN	Training:4,69,950 ,validation:1,00,704	BIS tagset (38 tags)	91.29%
	Bi-LSTM	Training:4,69,950 ,validation:1,00,704	UD tagset (17 tags)	94.48%
	Bi-LSTM	Training:4,69,950 ,validation:1,00,704	BIS tagset (38 tags)	92.60%

Table 9. **Top five misclassification classes for CNN and Bi-LSTM model**

Model	Actual label	Predicted label	Number 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

one hot encoded vectors $x_0, x_1, ...x_{17}$. Here $x_1, x_2, ...x_{17}$ represents each pos tags. For word w , if both Bi-LSTM and CNN agreed on the tags (say,noun), then we replaced Z_w with $Z_w \oplus x_1$. If for word w, they disagree, we replaced Z_w with $Z_w \oplus x_0$ and also we concatenated x_0 for any new words came in vocabulary. Here x_0 represents 0 0 0...0. 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% and its corresponding the 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 10. We determined from the above table that the Bi-LSTM model achieved the maximum accuracy, 94.58 percent. Hence, based on the above mentioned observations, the experimental results indicate that the Bi-LSTM model with various combinations of features achieved better results for Odia POS tagging.

Table 10. **Updated results of CNN and Bi-LSTM model**

Iteration	Number of instances result of CNN equals to Bi-LSTM	Accuracy of CNN model		Accuracy of Bi-LSTM model	
		Validation	Testing	Validation	Testing
-	-	92.61	92.77	94.43	94.48
1	544533	92.75	92.89	94.48	94.54
2	549850	93.03	93.09	94.53	94.56
3	550644	93.06	93.13	94.56	94.58
4	550852	93.10	93.22	94.55	94.51
5	550907	93.06	93.14	94.51	94.52

7 CONCLUSION AND FUTURE WORK

In this work, we have presented various methods for developing part of speech tagging for the Odia language that are based on statistical and deep learning-based approaches. Both a statistical method like conditional random field (CRF) and a deep learning-based technique like convolutional neural network (CNN) and bidirectional long short term memory (Bi-LSTM) were considered in this research. Models such as CRF, Bi-LSTM, and CNN are trained using the ILCI corpus with BIS and UD tagsets. We experimented with different feature set inputs to the CRF model, observed the impact of the constructed feature set, and achieved a 92.08 percent accuracy. We gathered raw corpora from a variety of online resources and Odia-specific websites, and then trained neural word embedding for Odia words. These word vectors are integrated into the deep learning model in order to reduce the number of out-of-vocabulary words. The deep learning-based model is comprised of a Bi-LSTM network, CNN network, CRF layer, character sequence data, and a pre-trained word vector. Bi-LSTM model with pre-trained word embedding and character sequence feature extracted by CNN yielded 94.48 percent accuracy. Moreover, we have also achieved an accuracy of 94.58% by using the label information of both the models input to Bi-LSTM model. In comparison to other existing studies on the Odia language, the proposed approaches produce 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 with a lack of resources, such as labelled corpora. In the future, we would want to investigate this possibility to generate more labelled datasets across all domains. Another area of research is to e extending this study and applying deep learning to other natural language processing (NLP) problems, such as name entity recognition, chunking, parsing, and machine translation.

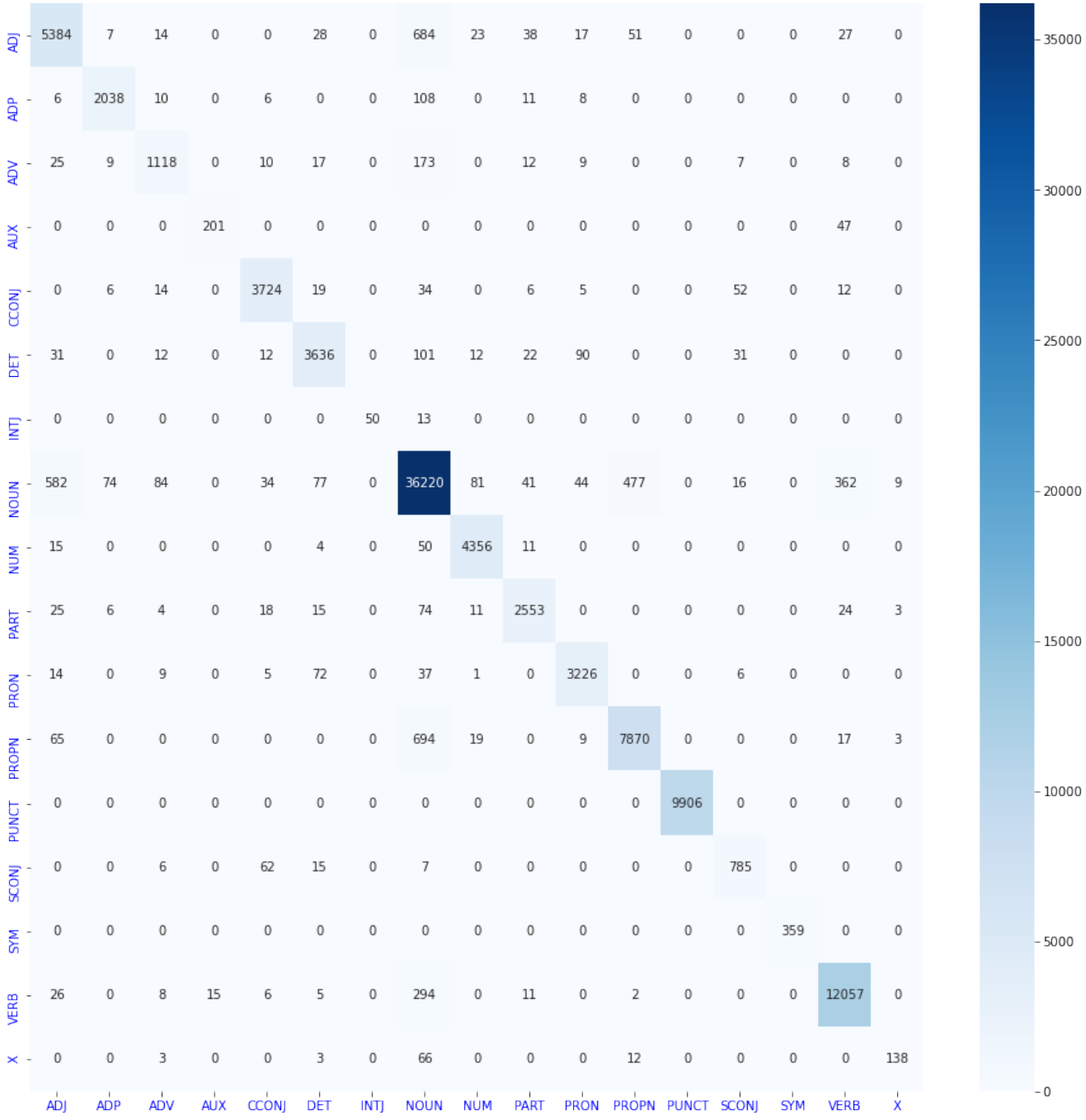

Fig. 4. Confusion matrix of Bi-LSTM model

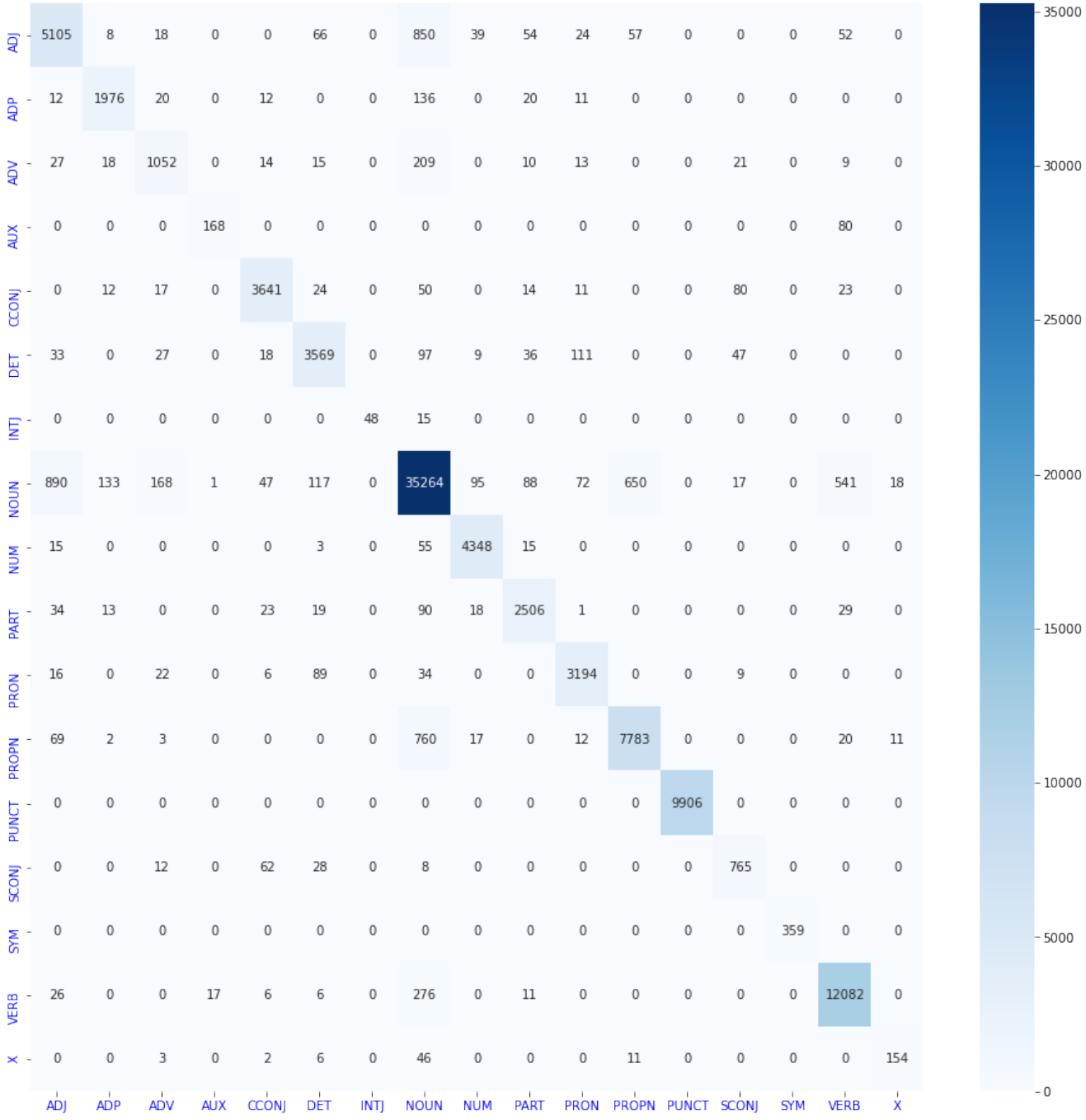

Fig. 5. Confusion matrix of CNN model

REFERENCES

[1] Firoj Alam, Shammur Absar Chowdhury, and Sheak Rashed Haider Noori. 2016. Bidirectional lstms—crfs networks for bangla pos tagging. In *2016 19th International Conference on Computer and Information Technology (ICCIT)*. IEEE, 377–382.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the association for computational linguistics* 5 (2017), 135–146.

[3] Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational linguistics* 21, 4 (1995), 543–565.

[4] Nitish Chandra, Sudhakar Kumawat, and Vinayak Srivastava. 2014. Various tagsets for indian languages and their performance in part of speech tagging Proceedings of 5 th IRF International Conference. *Chennai, 23rd March* (2014).

[5] Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A Practical Part-of-Speech Tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing* (Trento, Italy) *(ANLC '92)*. Association for Computational Linguistics, USA, 133–140. https://doi.org/10.3115/974499.974523

[6] Bishwa Ranjan Das and Srikanta Patnaik. 2014. A novel approach for Odia part of speech tagging using artificial neural network. In *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2013*. Springer, 147–154.

[7] Bishwa Ranjan Das, Smrutirekha Sahoo, Chandra Sekhar Panda, and Srikanta Patnaik. 2015. Part of speech tagging in odia using support vector machine. *Procedia Computer Science* 48 (2015), 507–512.

[8] Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. 4585–4592.

[9] Marie-Catherine De Marneffe and Christopher D Manning. 2008. The Stanford typed dependencies representation. In *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*. 1–8.

[10] V Dhanalakshmi, G Shivapratap, and Rajendran S Soman Kp. 2009. Tamil POS tagging using linear programming. (2009).

[11] Cicero Dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *International Conference on Machine Learning*. PMLR, 1818–1826.

[12] Asif Ekbal, Rejwanul Haque, and Sivaji Bandyopadhyay. 2007. Bengali part of speech tagging using conditional random field. In *Proceedings of seventh international symposium on natural language processing (SNLP2007)*. 131–136.

[13] Zellig Harris. 1962. String analysis of language structure. *Mouton and Co., The Hague* (1962).

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

[15] KS Gokul Krishnan, A Pooja, M Anand Kumar, and KP Soman. 2017. Character based bidirectional LSTM for disambiguating tamil part-of-speech categories. *Int. J. Control Theory Appl* 229 (2017), 235.

[16] Anoop Kunchukuttan, Divyanshu Kakwani, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. AI4Bharat-IndicNLP Corpus: Monolingual Corpora and Word Embeddings for Indic Languages. *arXiv preprint arXiv:2005.00085* (2020).

[17] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001).

[18] Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096* (2015).

[19] Ruslan Mitkov. 2022. *The Oxford handbook of computational linguistics*. Oxford University Press.

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

[21] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. *arXiv preprint arXiv:2004.10643* (2020).

[22] 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 *the proceedings of 7th language & technology conference: human language technologies as a challenge for computer science and linguistics*. 524–529.

[23] Shantipriya Parida, Ondřej Bojar, and Satya Ranjan Dash. 2020. OdiEnCorp: Odia–English and Odia-Only Corpus for Machine Translation. In *Smart Intelligent Computing and Applications*. Springer, 495–504.

[24] Shantipriya Parida, Satya Ranjan Dash, Ondřej Bojar, Petr Motlicek, Priyanka Pattnaik, and Debasish Kumar Mallick. 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.

[25] Sagarika Pattnaik, Ajit Kumar Nayak, and Srikanta Patnaik. 2020. A Semi-supervised Learning of HMM to Build a POS Tagger for a Low Resourced Language. *Journal of information and communication convergence engineering* 18, 4 (2020), 207–215.

[26] Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086* (2011).
[27] Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529* (2016).
[28] Ankur Priyadarshi and Sujan Kumar Saha. 2020. Towards the first Maithili part of speech tagger: Resource creation and system development. *Computer Speech & Language* 62 (2020), 101054.
[29] Gowtham Ramesh, Sumanth Doddapaneni, Aravinth Bheemaraj, Mayank Jobanputra, Raghavan AK, Ajitesh Sharma, Sujit Sahoo, Harshita Diddee, Divyanshu Kakwani, Navneet Kumar, 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.
[30] Helmut Schmid. 1994. Part-of-speech tagging with neural networks. *arXiv preprint cmp-lg/9410018* (1994).
[31] Manish Shrivastava and Pushpak Bhattacharyya. 2008. Hindi POS tagger using naive stemming: harnessing morphological information without extensive linguistic knowledge. In *International Conference on NLP (ICON08), Pune, India.*
[32] NM Suraksha, K Reshma, and KM Shiva Kumar. 2017. Part-of-speech tagging and parsing of Kannada text using Conditional Random Fields (CRFs). In *2017 International Conference on Intelligent Computing and Control (I2C2).* IEEE, 1–5.
[33] 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.
[34] 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. *Transactions on Asian and Low-Resource Language Information Processing* 21, 3 (2021), 1–24.
[35] 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.*