# Bidirectional LSTMs - CRFs Networks for Bangla POS Tagging

Firoj Alam, Shammur Absar Chowdhury
Department of Information Engineering and Computer Science
University of Trento, Italy
Email: {firoj.alam, shammur.chowdhury}@unitn.it

Sheak Rashed Haider Noori
Department of CSE
Daffodil International University, Bangladesh
Email: drnoori@daffodilvarsity.edu.bd

***Abstract*—Part-of-speech (POS) information is one of the fundamental components in the natural language processing pipeline, which helps in extracting higher-level information such as named entities, discourse, and syntactic structure of a sentence. For some languages, such as English, Dutch, and Chinese, it is considered as a solved problem due to the higher accuracy (97%) of the predicted system. Significant efforts have been made for such languages in terms of making the data publicly accessible and also organizing evaluation campaigns. Compared to that there are very fewer efforts for Bangla (ethnonym: Bangla; exonym: Bengali). In this paper, we present a *knowledge poor approach* for POS tagging, which we evaluated using publicly accessible dataset from LDC. The motivation of our approach is that we did not want to rely on any existing resources such as lexicon or named entity recognizer for designing the system as they are not publicly available and difficult to develop. We have not used any hand-crafted features, rather we employed distributed representations of word and characters. We designed the system using Long Short Term Memory (LSTM) neural networks followed by Conditional Random Fields (CRFs) for designing the model with an inclusion of pre-trained word embedded model. We obtained promising results with an accuracy of 86.0%.**

***Keywords*—Bangla, Deep Learning, POS tagging**

## I. Introduction

In any Natural Language Processing (NLP) pipeline, POS tagging is one of the fundamental tasks and its importance is well known in NLP community. The task of POS tagging is to annotate each word in a sentence into an appropriate predefined set of syntactic categories. Currently, the performance of POS tagging system is very high (above 95%) for several languages, especially, English, German, Italian and Chinese [1], [2]. Traditionally, systems are designed using supervised machine learning techniques by employing manually annotated dataset, where tags are predefined. To design the model using such approaches the idea is to employ many hand-crafted features and task-specific knowledge sources that describe how a *word* is associated with a particular POS tag. Most widely used machine learning algorithms are Hidden Markov Models (HMMs) [3], Conditional Random Fields (CRFs) [4], Maximum Entropy (ME) [5], Maximum Entropy Markov Models (MEMMs) [6], Support Vector Machines (SVMs) [7], [8], and hybrid approaches [9]. Hand-crafted features are mainly orthographic features such as prefix, suffix, the word in context, abbreviation, and numbers. An example of task-specific knowledge source is lexicon that describes whether a word is a noun, pronoun or others. In the last few years, with the advancement of high-performance computing, such as modern graphics processing unit (GPU) [10], the use of neural networks with more than one layer, also termed as "deep learning or deep neural networks", came forward. The success has been proven in many areas including NLP and computer vision [11].

In NLP, the use of distributed word representations in vector space, also called "word embeddings", has recently been popular with the inclusion of deep neural networks. Word embeddings, also known as context predictive model or neural language model, are new techniques to design "distributional semantic models (DSMs)", which differ from traditional DSMs where co-occurrence counts are used [12]. The idea "word embeddings" is to minimize the effort of hand-crafted features. It has been proven that this representation can capture semantic and syntactic information [13]. In word embedding, distributed vector representations are learned from a large corpus by neural network training, and represent them in a low-dimensional continuous space. Research of distributed representations are mainly focused on word-level, and very recently the use of character-level representations has also been reported in the literatures [14], [15], [16], [17]. The motivation of character-level representations is that it can capture word-shape, morphological and orthographic information, which is specially important for POS tagging. For sequence labeling tasks such as POS, and Named Entity Recognition (NER) the use of the word- and character- level information are employed mostly with Long Short Term Memory (LSTM) neural networks, a special kind of Recurrent Neural Networks (RNNs). Its success has been proven in several studies [18], [15].

In this study, we used the word- and character- level distributed representation with bidirectional LSTM neural networks (LSTMs) and followed by a CRFs layer to design the Bangla POS tagging system. The motivation of using such an approach is due to its success in numerous studies in sequence labeling task, as discussed earlier [16], [18], [15].

The structure of this paper is as follows. In Section II, we provide a brief overview of the existing work on Bangla POS tagging. In Section III, we discuss the corpus that we used in this study. We present our sequence labeling system in Section IV. In Section V we discuss the results and finally we present the conclusions in Section VI.

978-1-5090-4090-2/16/$31.00 ©2016 IEEE

ISBN 978-1-5090-4089-6

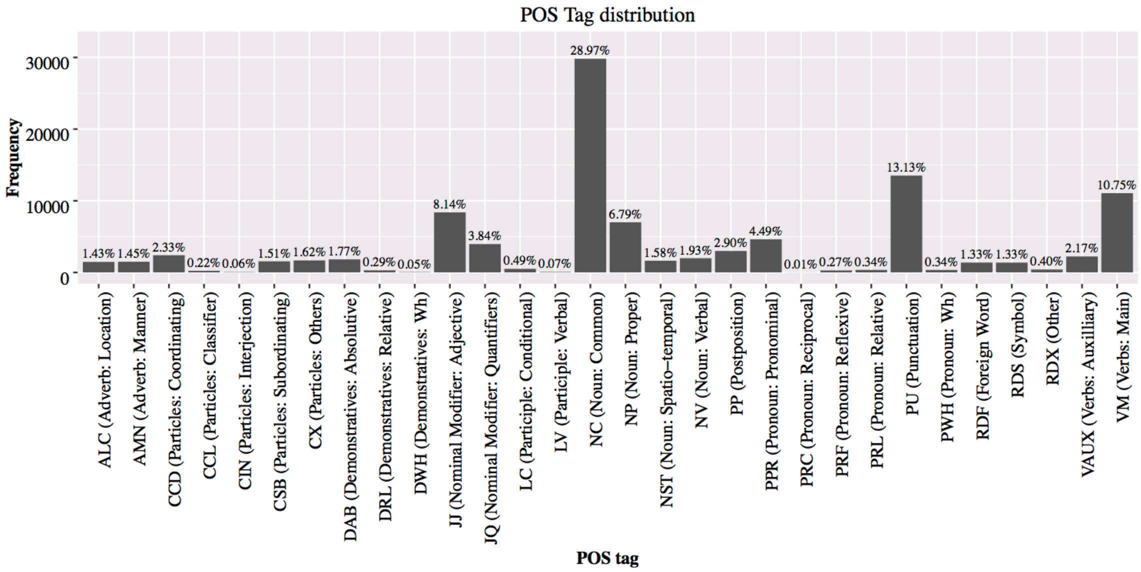

Fig. 1: POS tag distribution (in percentage).

## II. Related Work

There has been many efforts for the Bangla POS tagging systems, which include the development of POS tag set, evaluation campaign, designing automatic systems by investigating different machine learning algorithms [19], [20], [21], [22].

In [19], authors used a corpus containing 26 tags with $72,341$ tokens. The reported system achieved an accuracy of $90.3\%$ with the use of NER, lexicon, word suffix features, context window of size four, and prefix and suffixes of length 3, in which CRFs is used for designing the system. In another study [20], authors used the same corpus and comparatively evaluated the HMMs and ME based systems. The ME based system ($88.1\%$) outperforms the HMMs-based system ($80.3\%$), however, it appeared that it does not beat the performance of CRFs based system mentioned above. In [21], authors report that using the same experimental settings SVMs outperforms HMMs, ME, and CRFs based systems.

Dandapat et al. report that the use of morphological analyzer helps in improving the performance of the tagger [23]. They explored different experimental settings with different feature sets with HMMs and ME, which resulted in an accuracy of up to $88.41\%$. In their experiment, the corpus includes 40 tags with 3625 sentences of approximately $40,000$ words in the training set and the system was evaluated using $400$ sentences of $5000$ words.

Avinesh et al. [24] investigated different approaches using the data provided by SPSAL workshop at IJCAI in 2007. In their study, the achieved an accuracy of $76.08\%$ for Bangla using CRFs by exploiting Transformation-Based Learning.

Compared to the studies using supervised techniques there are very few studies using unsupervised techniques. In [25], Dasgupta et al. investigated bootstrapping algorithm for unsupervised POS induction and report that this approach works well for the resource-scarce condition. For Bangla, they obtained F1 $79\%$ using their approach. It is needed to mention that number of tags are very few (10 tags) in their study.

An extensive survey of Indian languages' POS taggers has been conducted by Kumar et al. [26], in which they reported the POS tagging systems of different Indian languages. It appeared that the accuracy of Bangla POS tagging systems varies from $\sim 77\%$ to $\sim 90.3\%$, which reports the different type of features and using different algorithms such as HMMs, ME, SVMs, or CRFs.

In [27], authors presented an HMMs-based system - designed using a small set of corpus consists of 895 Bangla sentences with 26 tags. With 10 folds cross-validation, the performance of the system is $78.68\%$.

In [28], authors used the same POS tag set, i.e., IL-POSTS. They manually annotated the corpus, which contains $4000$ sentences with approximately $44,048$ words. Their comparative study with different algorithms, such as Global Linear Model (GLM), CRFs, ME, SVMs, and HMMs, shows that GLM performs better than other algorithms with an accuracy of $93.12\%$.

In the current state-of-the-art of the Bangla POS tagging systems, we observed the number of tags varies from 26 to 30 and different statistical machine learning algorithms have been investigated. Better accuracy has been obtained using NER and lexicon based information in addition to the use of prefix and suffixes with 'word in context'. Comparative evaluations are very few due to the lack access to the same dataset.

Compared to the previous studies, our work differs in a number of ways: we focused on the use of publicly accessible data, which can help to replicate the experiment; our presented system does not rely on hand-crafted features; it does not exploit any external resources such as NER or lexicon.

## III. Corpus

We used the corpus that is publicly available through LDC [29], [30], which has developed by Microsoft Research (MSR), India, for the purpose of linguistic research. The corpus consists of three level annotations such as lexical category, type, and morphological attribute. For this study, we only utilized POS tags as a reference annotation. The whole corpus consists of $7398$ sentences corresponding to $1,02,921$ tokens. The text in the corpus collected from blogs, Wikipedia, and other sources in order to have a variation of the text. As a part of the annotation process, they developed a tag set consists of 30 tags. More details of this tag set can be found in the annotation guideline included with the corpus, and can also found in [29].

For this study, we split the corpus into training, development, and test set in order to design and evaluate the system as shown in Table I. In the original distribution, the data set appears in two sets “Bangla 1” and “Bangla 2”. We divided them by maintaining the file numbers and the associated set so that the experiment can be replicable in future. In our data split, we maintained $60\%$, $20\%$ and $20\%$ of the tokens, for the training, development and test set, respectively. It is needed to mention that in our data split there are many unknown words, i.e., words that are not found in the training set. About $\sim 51\%$ token types are unknown in both development and test sets.

TABLE I: Training, development and test data split. “Bangla 1” and “Bangla 2” are just two set in the original LDC distribution.

| Data set | # of Sent | # of Token (%) | Set from Original Source |
|---|---|---|---|
| **Train** | 4575 | 62048 (∼60%) | All from “Bangla 1” data set + (1 to 4) from “Bangla 2” data set |
| **Dev** | 1455 | 20435 (∼20%) | (5 to 10) from “Bangla 2” data set |
| **Test** | 1368 | 20437 (∼20%) | (11 to 17) from “Bangla 2” data set |

In Figure 1, we present the POS tag distribution, in percentage, for the whole corpus. The distribution are very low for some tags, for example, CIN (Particles: Interjection) - 59, DWH (Demonstratives: Wh) - 55, LV (Participle: Verbal) - 72, and PRC (Pronoun: Reciprocal) - 15. Such skewed tag distribution also effects the performance of the automatic sequence labeling (tagging) task.

## IV. Methodology

For the experiment, we design the model using the bidirectional LSTMs - CRFs, in which we exploited word- and character- level distributed representations. We conducted the preliminary experiments using the training and development set and final system was evaluated using the test set. Performance was computed using the precision, recall, F1 and accuracy measures discussed in Section IV-D.

### *A. Bidirectional LSTMs - CRFs Architecture*

In Figure 2, we present the architecture of the neural-network-based system, which we employed for designing the POS tagging model. It is a similar architecture that has been employed for NER by Lample et al. [16]. In this architecture, word level distributed representation [31], so called word-embeddings, and character level representation for word-embeddings are employed. Word-embeddings can capture the positional information, whereas character level representations can capture the prefix and suffix information, which is important for POS tagging, discussed earlier. Both representations are combined and then passed to the bidirectional LSTMs followed by a CRFs layer. In the following subsections, we will discuss each component of the system.

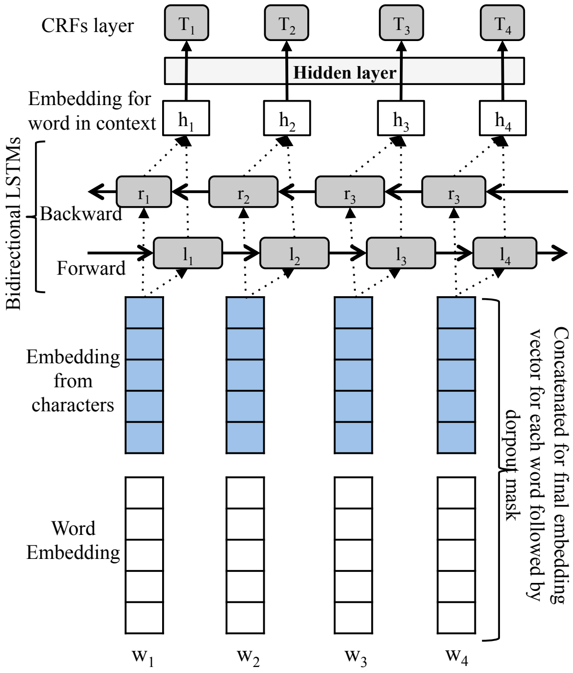
CRFs layer
T1
T2
T3
T4
Hidden layer
Embedding for word in context
h1
h2
h3
h4
Bidirectional LSTMs
Backward
r1
r2
r3
r3
Forward
l1
l2
l3
l4
Embedding from characters
Concatenated for final embedding vector for each word followed by dorpout mask
Word Embedding
w1
w2
w3
w4

Fig. 2: Bidirectional LSTMs -CRFs architecture of the system.

*1) LSTMs:* LSTMs [32] are a variant of RNNs. Due to their power of capturing long-term dependencies, RNNs have been widely used for sequence labeling tasks, which include speech recognition [33], language modeling [34], and machine translation [35]. RNNs maintain previous information using memory cells, upon which they predict the output. For example, it take a sequence of vectors, $(x_1, x_2, ..., x_n)$ as input and produce another sequence $(h_1, h_2, ..., h_n)$ as output. The drawback of RNNs is that it can not capture information from a very long distance because of the *gradient vanishing problem*, which results that it is biased towards most recent inputs. This is where LSTMs came in to overcome the drawback of RNNs and at the same time capture long-term dependencies using gating mechanisms. In Figure 3, we present a single LSTM memory cell for clarity. As presented in the figure it uses several gates, such as input, output, and forget, which control the amount of information to get- in and out to the LSTM cell. Gates are designed using a sigmoid neural net layers and pointwise multiplication operators. This version of LSTM does not use “peephole” [1] connections as can be seen in the figure. The LSTM memory cell is implemented using the following equations:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{2}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{3}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{4}$$

$$h_t = x_o \odot \tanh(c_t) \tag{5}$$

where $\sigma$ is the point-wise logistic sigmoid function $\frac{1}{(1+e^{-x})}$, and $\odot$ is the point-wise multiplication of the two vectors; $i$, $f$, $o$ and $c$ are the input gate, forgate gate, output gate, and memory cell vectors, respectively. The weight

[1] a variant of LSTM that uses some “peephole connections” [36] to use the cell’s $c_{t-1}$ internal previous information.

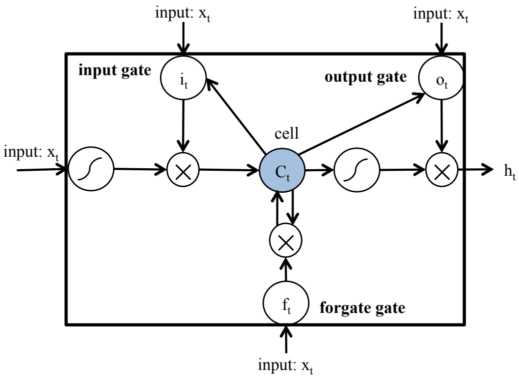

Fig. 3: A LSTM cell, presented for clarity and adapted from [32], [18].

matrix $W_*$ represents the weight vectors of different gates, for example, $W_{xi}$ is the input to input gate matrix. $b_i$, $b_f$, $b_c$, and $b_o$ denote bias vectors. Hence, the LSTM unit takes current input $x$ at time $t$ and previous hidden state $h_{t-1}$ and compute the next hidden state $h_t$. For a sentence with $n$ words with input $(x_1, x_2, ..., x_n)$, the LSTM will compute an output representation of $(h_1, h_2, ..., h_n)$.

*2) BLSTM:* In many different tasks both past and future information, i.e., left and right context, are important in order to predict the current output. The idea of bidirectional LSTM [33] is to utilize the contextual information to predict the current output and its success has been shown in the literature [37]. The idea is to pass the input sequence into two separate LSTMs to compute the representations of $\overrightarrow{h}$ forward and $\overleftarrow{h}$ backward sequences. This forward and backward representations are then combined in order to represent the word vector $h_t = \left[\overrightarrow{h_t}, \overleftarrow{h_t}\right]$.

*3) CRFs:* CRFs [4] are a probabilistic framework, which has a long history for seqence labeling tasks such as NER [38], and POS tagging [39]. The idea is to jointly learn the conditional probability distribution by considering the whole sequence of labels given the observation sequence, and then decode the best label sequences for an input sentence $s = x_1, x_2, ..., x_n$. As shown in Figure 2, we use $h_t$ as the final representation of features for the word, $x_t$, with label $y_t$ to train the CRFs model.

## B. Word Embeddings

To design word-embedding models, we collected text from different sources such as prothom-alo news corpus [40], transcriptions of CRBLP speech corpus [41], Bangla textbook corpus [42], wiki dump[2]. The whole corpus contains text of different genres, such as the magazine, novel, legal document, history, blog, constitution of Bangladesh, and different categories of news. It contains $\sim 84.25$ millions of words, $\sim 3.2$ millions word types, and $\sim 7.5$ millions of sentences.

To design the word embeddings, we utilized gensim implementations [43], which is an implementation of Mikolov et al. [44], [31] word vector model. It contains both continuous bag-of-words (CBOW) and skip-gram algorithms. We designed our model using the CBOW approach with a size of the feature vector 400, a context window size 4, negative-sampling with a value of $k = 4$ and filtered word with a frequency less than equal to two. The resulting trained word-embedding model contains 6 billion words with a vocabulary of size $\sim 1.98$ millions. We will make this model publicly available for research purpose on github[3].

## C. Embeddings from Characters

In Figure 4, we present the bidirectional LSTMs system for character-level embeddings. In [15], this character-level embeddings is designed through convolution neural networks. The reason of using bidirectional LSTMs was that forward LSTMs can capture suffix information with its final state, whereas the final state of the backward LSTMs can represent the prefix information. As shown in the Figure 4, for each character of the word, at first, a random vector was initialized. Then, they were passed to the forward and backward LSTMs to learn the prefix and suffix information. After that, their final states were concatenated to form a vector for the word. This vector was then concatenated with word embeddings from a pretrained model as shown in Figure 2.

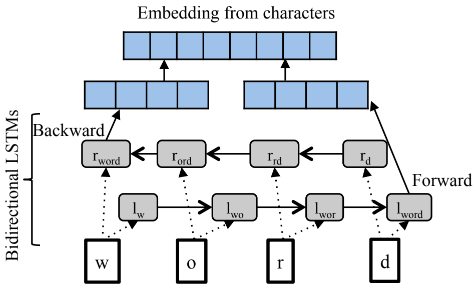

Fig. 4: Character embeddings were used as input to the bidirectional LSTMs, and then their forward and backward outputs were concatenated to have an embedding vector for the word.

## D. Training and Evaluation

For training and evaluating the system, we used the neural network based system presented in Figure 2. The system designs the feature vector for each word at different level to incorporate prefix-, suffix- and position- level information in order to predict the correct tag sequence. At the first step, it utilizes pretrained word embeddings and extract features using a lookup approach. Then, for designing embeddings from characters it utilizes bidirectional LSTMs as discussed earlier. It then concatenates these two embeddings, which resulted in the feature vector for each word. The system also uses a dropout mask (dropout rate = 0.5) on this feature vector for regularization, which it then passes to the bidirectional LSTMs to capture contextual information. Finally, the obtained feature vector from bidirectional LSTMs are passed to the CRFs layer to design the final model for the sequence predictions. The system also consists of a hidden layer between CRFs layer and bidirectional LSTMs. Instead of CRFs layer, the final layer can also be designed using a softmax function as presented in [17].

The neural network model was trained using backpropagation algorithms by updating the parameters on each

[2] wikidump accessed on 20th of July, 2016

[3] https://github.com/nlpresources/Word-embedding-model-for-Bangla

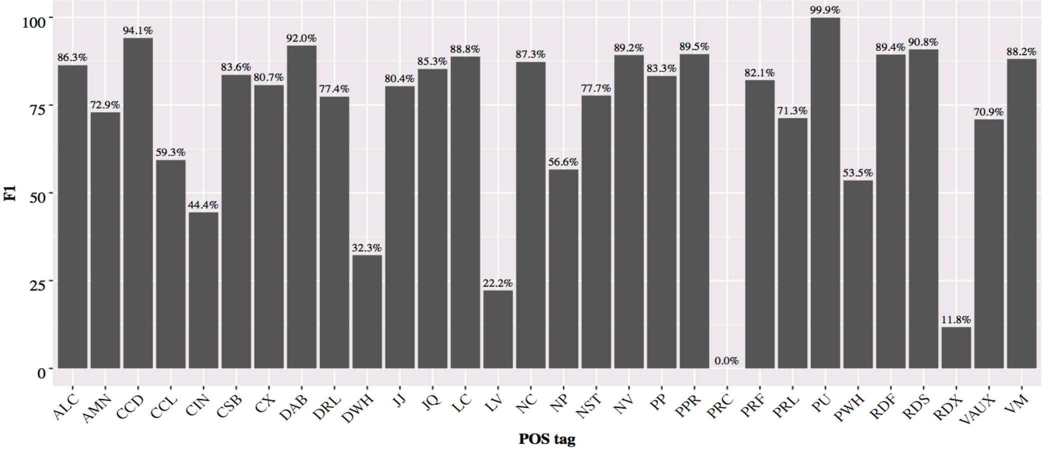

Fig. 5: F1 measures of POS tags.

training instance while Stochastic Gradient Descent (SGD) were used with a learning rate of $0.01$ and a gradient clipping of $5.0$. The dimension of the embeddings from character was set to $128$, and for pretrained word embeddings it is set to $100$. The dimension in forward and backward LSTMs in bidirectional LSTMs was set to $100$, for each. It is needed to mention that there are many words missing in the word embeddings dictionary, in those cases unknown word embeddings were randomly initialized.

To evaluate the performance of the system, we computed precision, recall, and F1 measure for each POS tag $i$ using the formulas as shown in Equation 6. We also calculated the accuracy of the system.

$$\begin{aligned} Precision_i &= \frac{TP_i}{TP_i + FP_i} \\ Recall_i &= \frac{TP_i}{TP_i + FN_i} \\ F1_i &= \frac{2 \times Precision_i \times Recall_i}{Precision_i + Recall_i} \end{aligned} \tag{6}$$

## V. Results and Discussion

In Table II, we present the performance of our system including baseline results on the test set. We computed our baseline results using the token and bigram context as features and used CRFs to train the model. For the sake of simplicity, we present the overall precision, recall, F1 measure, and also accuracy of the system. As can be seen in the table, there is a significant improvement in performance in terms of accuracy and F1 measure using the bidirectional LSTMs compared to the baseline system. On the development set, the F1 measure of the system is $84.68\%$. We have also conducted experiments without using CRFs in the final layer of our system. It appeared that accuray dropped $\sim 2.5\%$.

TABLE II: Results on test set

| Exp | P | R | F1 | Acc |
|---|---|---|---|---|
| **Baseline (Token+CRFs)** | 42.35 | 25.76 | 29.96 | 70.08 |
| **bidirectional LSTMs - CRFs** | 86.25 | 86.25 | 86.25 | 86.00 |

We analyzed the tag-wise performance on the test set as shown in Figure 5. From the figure, we can see that for PRC (reciprocal pronoun) the system completely failed to predict. For PRC tag, the system is not learning anything due their poor distributions ($0.01\%$) in the dataset, as can be seen in Figure 1. The distributions of DWH (Demonstratives: Wh) is also lower, i.e., $0.05\%$, however, the system could able to predict it with an F1 of $22.22\%$. It is interesting to note that the F1 of NP (proper noun) is very low $56.6\%$, which is affected due to the low recall, i.e, $41.49\%$ compared to precision $89.25\%$. The identification of proper noun is itself complex as it is an open-class problem and sometime it is confused with adjective. For example, বর্তমান/bort̪oman/[present] can be considered as a proper noun or adjective depending on the context. The opposite scenario is observed for the tag PWH (question - pronoun), higher recall, $97.14\%$ and low precision $36.96\%$, resulted in an F1 of $53.54\%$.

We have done an analysis to understand what type of error occurs in case of unkown words. It appears that most of the error occurs with proper nouns, adjectives, and then common nouns. For the known words case, it happens with auxiliary verbs.

Our results are not directly comparable with existing work for Bangla due to the different data set and number of tags that has presented in the literature [26], [19]. In [19], authors presented an accuracy of $90.3\%$, in which they exploited many different linguistic knowledges. The performance of English POS tagging system using Penn TreeBank (PTB) dataset with bidirectional LSTMs - CRFs architecture is accuracy=$97.55\%$, which is $11.54\%$ higher than the performance we are presenting for Bangla. This indicates more efforts needed to be done for Bangla to understand the intricacies of the complex dynamics of neural network architectures and the language. In addition, experiments can also be conducted by incorporating linguistic knowledge with neural network architecture.

## VI. Conclusions

In this paper, we present a neural network based POS tagging system for Bangla, which does not require hand-crafted features or any knowledge sources in order to incorporate linguistic information. We used a dataset that is publicly available through LDC. We will made the data split available for sake of replicability of the experiments. We obtained very promising results compared to the baseline and close results compared to the other studies for Bangla, even though it is not directly comparable. In future, we will investigate unlabeled dataset to incorporate more data for training.

## References

[1] C. D. Manning, "Part-of-speech tagging from 97% to 100%: is it time for some linguistics?" in *Int. Conf. on Intelligent Text Processing and Computational Linguistics*. Springer, 2011, pp. 171–189.

[2] E. Giesbrecht and S. Evert, "Is part-of-speech tagging a solved task? an evaluation of pos taggers for the german web as corpus," in *Proceedings of the fifth Web as Corpus workshop*, 2009, pp. 27–35.

[3] T. Brants, "Tnt: a statistical part-of-speech tagger," in *Proceedings of the sixth conference on Applied natural language processing*. Association for Computational Linguistics, 2000, pp. 224–231.

[4] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. of 8th ICML*, vol. 1, 2001, pp. 282–289.

[5] A. Ratnaparkhi *et al.*, "A maximum entropy model for part-of-speech tagging," in *Proc. of EMNLP*, vol. 1. Philadelphia, USA, 1996, pp. 133–142.

[6] A. McCallum, D. Freitag, and F. C. Pereira, "Maximum entropy markov models for information extraction and segmentation." in *Proc. of ICML*, vol. 17, 2000, pp. 591–598.

[7] T. Kudo and Y. Matsumoto, "Chunking with support vector machines," in *Proc. of NACL*. ACL, 2001, pp. 1–8.

[8] H. Yamada and Y. Matsumoto, "Statistical dependency analysis with support vector machines," in *Proceedings of IWPT*, vol. 3, 2003, pp. 195–206.

[9] Y. Altun, I. Tsochantaridis, T. Hofmann *et al.*, "Hidden markov support vector machines," in *ICML*, vol. 3, 2003, pp. 3–10.

[10] J. Nickolls and W. J. Dally, "The gpu computing era," *Micro, IEEE*, vol. 30, no. 2, pp. 56–69, 2010.

[11] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *JMLR*, vol. 11, no. Feb, pp. 625–660, 2010.

[12] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors," in *Proc. of ACL*, vol. 1, 2014, pp. 238–247.

[13] J. Bian, B. Gao, and T.-Y. Liu, "Knowledge-powered deep learning for word embedding," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 132–148.

[14] C. N. dos Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging." in *ICML*, 2014, pp. 1818–1826.

[15] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," *arXiv preprint arXiv:1603.01354*, 2016.

[16] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[17] W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black, and I. Trancoso, "Finding function in form: Compositional character models for open vocabulary word representation," *arXiv preprint arXiv:1508.02096*, 2015.

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

[19] A. Ekbal, R. Haque, and S. Bandyopadhyay, "Bengali part of speech tagging using conditional random field," in *Proc. of SNLP*, 2007.

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

[21] A. Ekbal and S. Bandyopadhyay, "Part of speech tagging in bengali using support vector machine," in *Proc. of International Conference on Information Technology (ICIT)*. IEEE, 2008, pp. 106–111.

[22] S. Dandapat, "Part of specch tagging and chunking with maximum entropy model," in *Proceedings of the IJCAI Workshop on Shallow Parsing for South Asian Languages*, 2007, pp. 29–32.

[23] S. Dandapat, S. Sarkar, and A. Basu, "Automatic part-of-speech tagging for bengali: An approach for morphologically rich languages in a poor resource scenario," in *Proc. of ACL*. ACL, 2007, pp. 221–224.

[24] A. PVS and G. Karthik, "Part-of-speech tagging and chunking using conditional random fields and transformation based learning," *Shallow Parsing for South Asian Languages*, vol. 21, 2007.

[25] S. Dasgupta and V. Ng, "Unsupervised part-of-speech acquisition for resource-scarce languages." in *Proc. of EMNLP-CoNLL*, vol. 7, 2007, pp. 218–227.

[26] D. Kumar and G. S. Josan, "Part of speech taggers for morphologically rich indian languages: a survey," *International Journal of Computer Applications*, vol. 6, no. 5, pp. 32–41, 2010.

[27] K. Sarkar and V. Gayen, "A practical part-of-speech tagger for bengali," in *Proc. of Third International Conference on Emerging Applications of Information Technology (EAIT)*. IEEE, 2012, pp. 36–40.

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

[29] S. Baskaran, K. Bali, T. Bhattacharya, P. Bhattacharyya, G. N. Jha *et al.*, "A common parts-of-speech tagset framework for indian languages," in *Proc. of LREC 2008*. Citeseer, 2008.

[30] M. C. Kalika Bali and P. Biswas, "Indian language part-of-speech tagset: Bengali ldc2010t16," Philadelphia: Linguistic Data Consortium, Tech. Rep., 2010.

[31] T. Mikolov and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, 2013.

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[34] T. Mikolov, S. Kombrink, L. Burget, J. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. of ICASSP*. IEEE, 2011, pp. 5528–5531.

[35] S. Liu, N. Yang, M. Li, and M. Zhou, "A recursive recurrent neural network for statistical machine translation." in *Proc. of ACL*, 2014, pp. 1491–1500.

[36] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *JMLR*, vol. 3, no. Aug, pp. 115–143, 2002.

[37] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. of ICASSP*. IEEE, 2013, pp. 6645–6649.

[38] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons," in *Proc. of HLT-NAACL*, vol. 4. ACL, 2003, pp. 188–191.

[39] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proc. of NACL*. ACL, 2003, pp. 173–180.

[40] Y. Arafat, M. Z. Islam, and M. Khan, "Analysis and observations from a bangla news corpus," in *Proc. of 9th International Conference on Computer and Information Technology (ICCIT 2006)*, 2006.

[41] F. Alam, S. Habib, D. A. Sultana, and M. Khan, "Development of annotated bangla speech corpora," 2010.

[42] Z. Islam, A. Mehler, R. Rahman, and A. Texttechnology, "Text readability classification of textbooks of a low-resource language," in *Proc. of Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, 2012.

[43] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[44] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of the International Conference on Learning Representations*, 2013, available as arXiv preprint arXiv:1301.3781.