Journal of Theoretical and Applied Information Technology

15th September 2023. Vol.101. No 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

NER-NET: A NEW DEEP LEARNING ARCHITECTURE WITH HYBRID OPTIMIZATION FOR NAMED ENTITY RECOGNITION

P SHEELA GOWR[1], N KUMAR[2]

[1]Research Scholor, Department of CSE,Vels Institute of Science, Technology & Advanced Studies Chennai, India
[2]Professor, Department of CSE, Vels Institute of Science, Technology & Advanced Studies Chennai, India
E-mail: [1]psheelagowr85@gmail.com, [2]kumar.se@velsuniv.ac.in

ABSTRACT

Identification of textual and visual information in a document plays a pivotal role in comprehending the information contained in it, thereby providing an effective way to analyze the document. Information Extraction (IE) is an attractive research area that mainly targets developing techniques for analyzing rich documents. Named Entity Recognition (NER) is the prime process in IE, and it is a sequence-labeling process, wherein unstructured data is considered and the data is mapped to pre-defined labels. In this work, a novel deep learning architecture is developed for performing NER, and it is aimed at identifying the numerical entities in the input text. Here, a novel deep learning network named NER-Net is proposed which includes various layers, like tokenization, Inverse Document Frequency (IDF), Convolutional Neural Network (CNN), Bidirectional Long Short Term Memory (BiLSTM), attention, and NER layers. Further, a Fire Hawk African Vultures Optimization Algorithm (FHAVOA) is proposed for optimally tuning the layer dimension of the NER-Net. Moreover, the evaluation of NER-Net-FHAVOA shows that accuracy of 0.936, Mean Square Error (MSE) of 0.087, Root MSE (RMSE) of 0.294, Positive Predictive Value (PPV) of 0.909, and Negative Predictive Value (NPV) of 0.877 are attained, thus revealing its superiority.

Keywords: *Fire Hawk Optimization, African Vultures Optimization Algorithm, NER-Net, Convolutional Neural Network, Bidirectional Long Short Term Memory*.

1. INTRODUCTION

The process of extracting structured data from semi-structured and unstructured texts is known as Information Extraction (IE) and has been existing for quite a long time. It is associated with the task of processing human language, and so scholars have been using approaches based on Natural Language Processing (NLP) extensively for addressing this. A few of the NLP subtasks that are linked with IS are semi-structured data mining (e.g., table extraction), template filling, Relation Extraction (RE), coreference resolution, and NER. The growth in the usage of various communication channels and the internet has led to the evolution of the input in IE systems from simple textual data to speech, images, and posts/multimedia documents. In this new scenario, scholars are forced to discover innovative methods for using the data attained from the new sources [3]. Identifying the most prominent terms in the textual information is a necessity to extract any valuable details, like the relationship between objects. Named entities refer to phrases or terms that are meaningful and that can be differentiated from similar objects. NER is one such approach that performs automatic identification of named entities in the text and classifies them into various kinds of predetermined entities [11]. It is done to distinguish references of firm designators from the textual data associated with predetermined semantic classes like organization, location, person, and so on. Apart from acting as a standalone tool for IE, NER also contributes a prime part in a multitude of NLP tasks, like knowledge base construction [22], machine translation [23], question answering [24], automatic text summarization [25], information retrieval [26], text understanding [27], etc. [9]. It can also be considered as the process of recognizing named entities, like biological proteins, clinical procedures, time, drugs, and so on from the text. It is most commonly applied as

the primary process in topic modeling, co-reference resolution, and other NLP tasks [10].

NER is considered to be a vital process in NLP, as the semantic details in the text that are essential in the downstream NLP process can be captured using NER [7]. The NER process was first systematized at the Sixth Message Understanding Conference in 1996 by Grishman and Sundheim, and thereafter a large number of NER processes have been organized. In the primary stages, these processes were carried out considering ontologies, orthographic features, lexicons, and handcrafted rules. In the later stages, Machine Learning (ML), and feature engineering were utilized in the NER systems [10]. Conventional NER systems typically employ ML methods, like Conditional Random Field (CRF) and Hidden Markov Model (HMM). Though these models attain excellent results, their heavy reliance on task-specific resources or hand-crafted features makes them difficult to be applied to various languages or domains, and the process is expensive [12]. Additionally, these systems require a huge quantity of labeled information during the training process. As the applications of NER are spread over several fields, like biology, social media, news, and so on, labeling the training data in all domains is extremely laborious, and there is a shortage of adequate labeled data in all fields. Hence, it is essential to create efficacious algorithms that are applicable in low-resource domains that have insufficient labeled data [8]. In NER research, neural systems have become the foremost applied techniques over recent years. Neural models are devoid of the requirement of assimilating external lexicons and determining hand-crafted features and can learn distributed representations from large-scale unlabeled texts [13].

Recently, a highly practical solution is devised by the utilization of the Deep Neural Network (DNN), which effectively learns the statistical features from very large datasets by summarizing and extracting the features for particular processes. Using these approaches, breakthroughs are produced in various processes, like question-answering systems, information retrieval, NER, syntactic analysis, text classification, etc. [4]. The neural network comprises multiple layers, which processed the input, and each layer performs certain mathematical functions on the input thereby transforming the features in the database [15]. In NLP tasks, Recurrent Neural Networks (RNNs) are most commonly used owing to their sequential nature, which is in agreement with the characteristics of human language. One of the extensively employed RNN models is Bidirectional Long Short-Term Memory networks (Bi-LSTM) to be specific, as it has a high supremacy in learning the contextual depiction in the documents. Bi-LSTM is generally used as the encoder in NER models [5]. In addition to this, LSTMs are generally utilized for learning the vector representation of all words in a sentence, and the result generated when applied to CRF is found to enhance the efficacy of NER [14]. But RNN carries out computation based on the location of the input sequence and so only the sequential processing of the sentence is possible, which brings in dependency of the present computations on the previous ones. This can be avoided by the usage of CNN, which unlike the RNN process the words in a feed-forward manner instead of combining representations of all words in the sentence in an incremental fashion. This aspect of CNN enables it in exploiting Graphics Processing Unit (GPU) parallelism [12].

In this work, a novel deep learning architecture named NER-Net is developed for performing NER, which comprises various layers, like tokenization, IDF, CNN, BiLSTM, attention, and NER layers. Here. the input sentence is first tokenized at the tokenization layer and converted to character-level and word-level encodings, and a matrix is generated from the encoded output in the IDF layer. Later, the matrix is applied to the CNN and BilSTM layers for generating local and contextual features. Thereafter, a tagging score is computed for each token in the attention layer and the score is mapped to each class in the NER layer for identifying the numerical entity. Moreover, the layer size of the NER-Net is determined by using the proposed FHAVOA.

The key contributions of this work are listed below:

- ***Proposed NER-Net for NER:*** A novel deep learning network named NER-Net is developed in this research for identifying the numerical entities in the input sentences. The NER-Net comprises various layers, like tokenization, IDF, CNN, BiLSTM, attention, and NER layers.

- ***Proposed FHAVOA for layer dimension determination:*** A optimization technique named FHAVOA is devised by combining the Fire Hawk Optimization (FHO) algorithm and the African Vultures Optimization Algorithm (AVOA) for determining the layer dimension of the NER-Net.

The residual part of the work is arranged as follows: section 2 details the available NER works, section 3 explains the architecture of the NER-Net, section 4 outlines the experimental results, and section 5 concludes the research with suggestions for future work.

2. MOTIVATION

NER is a key process in various NLP applications, and these applications are developed for specific domains or languages, which results in the insufficient availability of labeled datasets. Further, labeling the data is extremely expensive and time-consuming, thereby requiring effective methods to carry out NER. In this segment, the already available methods of NER are detailed with their merits and issues met that formed the stimulus for creating the NER-Net.

2.1. Literature Review

Cho, M., *et al.* [1] developed a biomedical NER (bioNER) method using CNN and BiLSTM based on combinatorial feature embedding for representing biomedical word tokens. This system was founded n CRF and BiLSTM, and the process of NER was augmented through the integration of BiLSTM and CRF. Further, the long-term dependence issue of the LSTM was alleviated using an attention model. This technique was effective in the precise identification and classification of the entities and handled the partial match issues. However, the method had trouble in identifying entities comprising Greek letters, special characters, numbers, and letters easily. Francis, S., *et al.* [2] presented a Transfer Learning (TL) for NER, and this scheme was a generic extraction technique, which concentrated on biomedical and financial documents. Here, Word and Character Level Neural Networks with and without Attention were developed for performing NER on biomedical and financial documents, respectively. As TL was applied, this technique effectively addressed the data scarcity issues, although the availability of a large number of labeled data, a degraded performance was observed. Oral, B., *et al.* [3] devised a BiLSTM-CRF with a graph-based relation extraction algorithm for extracting information based on NER. This approach initially utilized BiLSTM-CRF for extracting the named entities, and then binary relations were predicted using a graph-based relation extraction algorithm. This method was successful in addressing the data sparsity and minimized time consumption, however, endured a high computational cost. Wang, J., *et al.* [4] developed a technique named Adversarial Trained LSTM-CNN (ASTRAL) for augmenting the efficiency of NER. Here, Gated-CNN was applied for fusing the data contained in the neighboring words to utilize the spatial information efficiently. Further, the overfitting problem was dealt with through the application of adversarial training. This model had good generalizability and robustness, although it was unproductive in excerpting particular words requiring circumstantial knowledge.

Yan, H., *et al.* [5] introduced a Transformer Encoder for NER (TENER) for modeling features at the word and character levels. Here, un-scaled, distance-aware, and direction-aware attention were introduced to the transformer encoder for making it applicable to NER tasks. The TENER had a fast convergence and was effective in extracting the intricate patterns from characters, but with scaled attention, a deterioration in performance was observed. Lin, B.Y., *et al.* [6] created a Trigger Matching Network (TMN) for learning the labels effectively using NER. The TMN comprised various modules, like a semantic-based trigger matching module and trigger encoder for learning and the base sequence tagger was utilized for learning the NER tag labels using trigger vectors. The TMN was generalizable in tagging named entities even to unseen sentences and was cost-effective, however, it failed to consider the structured inductive bias to improve the trigger modeling. Jie, Z. and Lu, W., [7] developed a Dependency-Guided LSTM-CRF (DGLSTM-CRF) for encoding and capturing the properties of dependency trees for NER. Here, the LSTM was utilized for capturing the contextual information from the dependency-encoded input representation, and the CRF was utilized to produce the context representations. This model was effective in identifying long entities, although it was not suitable for real-time applications.

Peng, Q., *et al.* [8] devised an entity-aware adversarial training model for cross-domain NER. Here, knowledge was transferred to the target domain from the source based on adversarial training, and the difference of entity in cross-domain was minimized using an entity-aware attention module. This method was successful in handling low-resource situations, however, in the case of complex and random source domains, a substantial increase in performance was not recorded.

2.2. Challenges

Several methods have been devised for NER, and a few of the issues confronted by the techniques deliberated in literature as given below,

- In [1], the CNN and BiLSTM method was introduced for performing bioNER automatically and it was efficacious in capturing the meaning data with high accuracy. However, the performance was limited by the presence of long and complex words, and abbreviations.
- The main problem met by the TL approach proposed in [2] for NER on biomedical and financial documents was that though the technique had a reduced training time, and generalizability, the method was not implemented in real-time.
- The BiLSTM-CRF [3] was developed for extracting information from documents based on NER. This method was effectual in minimizing the effort of human operators, however, it failed to combine word positions with language models and document layout with graph convolutional networks for improving performance.
- In [4], a NER system named ASTRAL was introduced and it produced superior results while identifying named entities from text, like comments, books, news, and so on. However, it was futile in addressing the issue of insufficient training data by considering data enhancement techniques, like distant supervision.
- The NER datasets generally have smaller dimensions, which result in overfitting issues when processed with DL techniques. Further, these methods easily identify familiar words but find it difficult to comprehend unaccustomed ones, so attaining high generalizability is a major issue in attaining stability.

3. PROPOSED NER ARCHITECTURE FOR NAMED ENTITY RECOGNITION

With the huge quantity of unstructured data that is generated on a day-to-day basis, it is necessary to formulate effective techniques for IE and retrieval. NER is the process of structuring data into pre-determined labels. This segment details the NER-Net proposed in this work for carrying out NER. Here, the input text sentence is fed as input to the NER-Net, which is a new time series deep learning architecture that is devised based on the integration of CNN [16] and BiLSTM [17]. The text input is given to the Tokenized layer for generating tokens from the input sentence, and the output generated is forwarded to the IDF layer for obtaining the word-level and character-level features in the input text. Further, the obtained word-level and character-level features are subjected to CNN, which is employed for extracting features in the text. Thereafter, the features generated by the CNN are subjected to the BiLSTM, and the vector representation produced is applied to the Attention layer, which calculates the tagging score. Finally, the NER layer on top of the Bi LSTM is predict the label of each word token. Here, the layer size and connections are determined using the proposed FHAVOA, which is created by combining the FHO [18] algorithm and the AVOA [19]. Figure 1 depicts the structural view of the NER-Net proposed for NER.

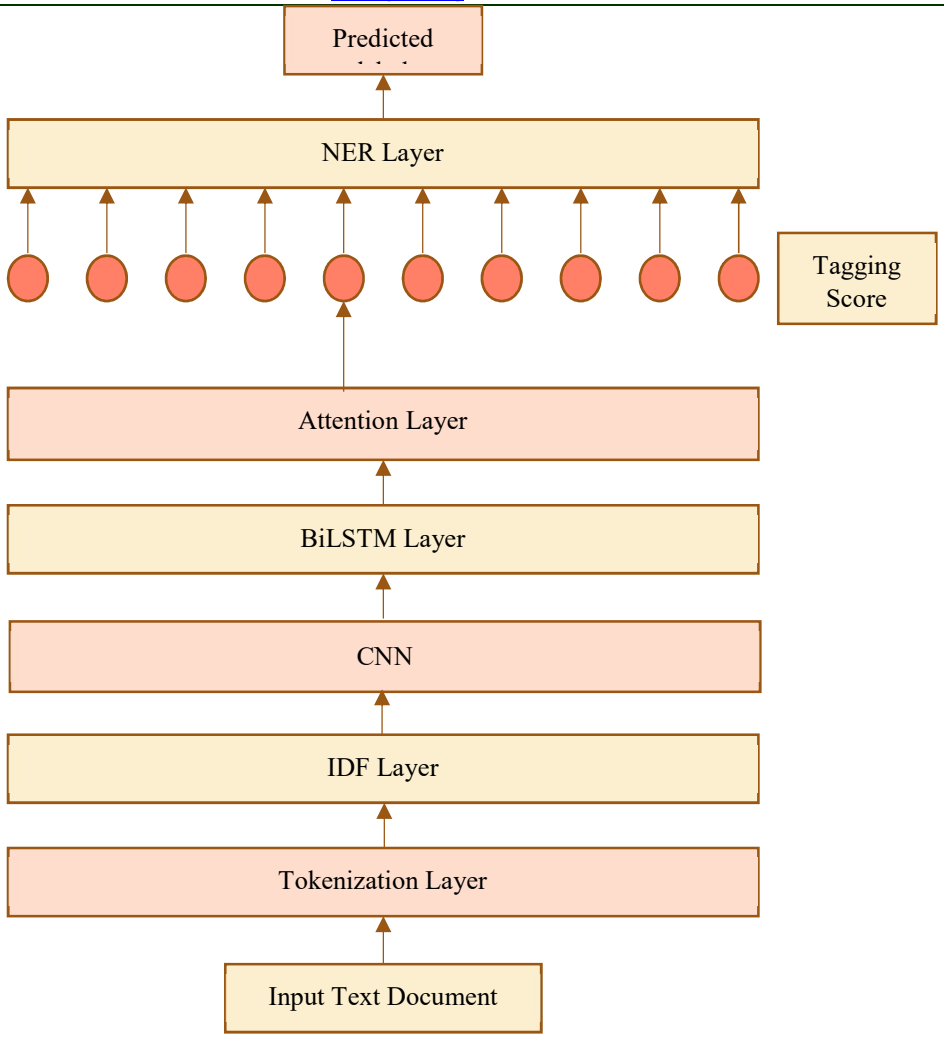


Figure 1. Structural view of the proposed NER-Net

3.1. NER architecture

A novel network named NER-Net is proposed in this work for carrying out identifying numerical values in the input. A sentence is considered as the input to the NER-Net, which is applied to the tokenization layer, where the input sentence is converted into tokens or words. Thereafter, the tokens generated are forwarded to the IDF layer, which comprises two encoders, such as the word-level encoder and the character-level encoder. The character level encoder generates a vector corresponding to the tokens/words obtained from the tokenization layer. Similarly, the word-level encoder creates a vector by combining two adjacent tokens. The vector generated by both modules are concatenated and applied to the CNN layer, which contains multiple layers, and it extracts feature vectors from the input text. The output of the CNN layer is fed to the BiLSTM, which generates a fixed-size vector based on the character sequence embedding. Further, an Attention layer is applied for generating the tagging score, based on which a label is predicted by the NER layer, and this process is explicated in the ensuing sections, and the general architecture is depicted in figure 2.

Journal of Theoretical and Applied Information Technology
15th September 2023. Vol.101. No 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

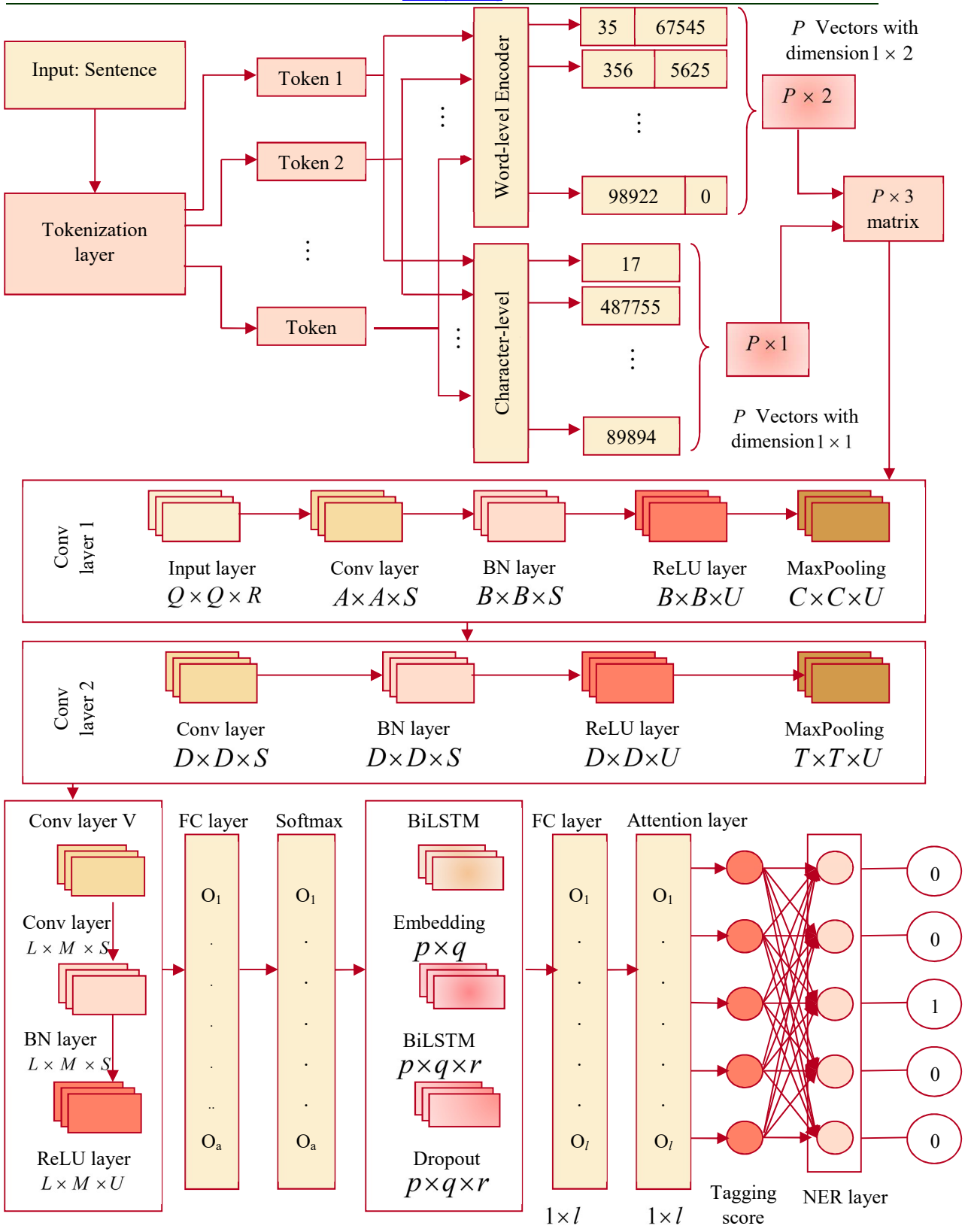


Figure 2. General architecture of the proposed NER-Net

3.1.1. Tokenization layer

The primary process in identifying the numerical entity in the sentence for NER based on the NER-Net is tokenization. Tokenization is the primary process in all NLP processes, and here, the input sentence is converted into multiple tokens that can be punctuation marks or words. These tokens/words obtained are then subjected to the character-level and word-level encoders in the IDF layer. Consider an input sentence F be applied to the tokenization layer and a total of K tokens is generated during the tokenization process, which is expressed as

$$F = \{token_1, token_2, ...token_b, ..., token_K\} \quad (1)$$

Here, $token_b$ represents the b^{th} token in the sentence, and the tokens $\{token_1, token_2, ...token_b, ..., token_K\}$ are applied to the IDF layer for generating word-level and character-level encodings.

3.1.2. Character-level encoder

The token $\{token_1, token_2, ...token_b, ..., token_K\}$ generated from the tokenization layer are applied to the character-level encoder, which generates a vector corresponding to each input token received. The character-level encoders are utilized as they are efficient in encoding multiple languages without any increase in the model dimensionality. Further, this encoding enables the NER-Net is dealing with Out of Vocabulary (OOV) words. Here, the vector generated has a dimension of 1×1, and as a total of K tokens are applied to the character-level encoder, a token of K vectors each with the dimension of 1×1 is produced during the character-level encoding. Assume that the embedding of $token_b$ be indicated as d_b, and thus the character level encoding output obtained is represented by the expression below,

$$L = \{d_1, d_2, .., d_b, .., d_K\} \quad (2)$$

here, L indicates the matrix produced by the character-level encoder, and has a dimension $K \times 1$.

3.1.3. Word-level encoder

The word-level encoder encodes the words/tokens $\{token_1, token_2, ...token_b, ..., token_K\}$ generated in the sentence to a real-valued vector. It is a distributed word representation used for mapping words into word-level representations. The word-level encoding is generated by considering the two neighboring tokens in the sentence, and in the case of the final token, a value of '0' is appended to the last token, and the vector is produced. It is mainly used for capturing the semantic relationship among the tokens. The real-valued vector generated during the word-level encoding will have a dimension of 1×2, and the sequence thus generated is formulated as,

$$M = \{c_1, c_2, .., c_j, .., c_K\} \quad (3)$$

Here, c_j refers to the j^{th} word encoding, and a total of K word embeddings are generated in this process. The output produced by the word-level encoder is specified as M and has a dimensionality of $K \times 2$.

3.1.4. Matrix formation

Once the tokens are encoded at word-level and character-level, a matrix is generated from the encoded vectors. The character-level encoded output L and the word-level encoded output M are concatenated to form a matrix I, which is formulated as,

$$I = L \oplus M \quad (4)$$

Here, $\oplus$ represents the concatenation operator, and the matrix I produced has a dimension of $K \times 3$, as it is obtained by concatenating two matrices with dimensions $K \times 1$ and $K \times 2$. The matrix I is then subjected to the CNN layer for extracting the features from the input.

3.1.5. CNN layer

The matrix I generated is then subjected to the CNN layers for excerpting the local-level features in the input [28],[1]. Here, the word and character encodings are directly applied to the CNN, for determining features. The CNN utilized here comprises a total of V layers, and each layer is connected successively. The CNN layer contains

different layers, like the convolutional (conv) layer, Batch Normalization (BN), Rectified Linear Unit (ReLU), MaxPooling (maxPool), Fully Connected (FC), and softmax classifier. The conv layer is responsible for extracting the features from the input data, it comprises a number of kernels (filters), which are convolved with the input data to produce feature maps. The BN layer is responsible for speeding up the training process, and it helps the CNN to process the data independently in batches rather than processing the input data fully. Further, a ReLU activation is applied in the CNN, which is an activation function that is responsible for activating the neurons, thereby determining the significance of the input applied to a neuron. ReLU is employed for making the computations faster, and this function will not activate all neurons simultaneously. Here, the output thus produced is depicted as,

$$z_j = \alpha\left(\omega \cdot I\left[j, j+i-1\right] + \beta\right) \quad (5)$$

Here, α refers to the ReLU function, ω and β denotes the weight and bias values, and the feature map thus produced is formulated as,

$$Z = \left[z_1, z_2, .., z_{K-j+1}\right] \quad (6)$$

At last, a max-pooling function is also used in the CNN for reducing the dimensionality of the generated feature maps. In addition to this, an FC and a softmax classifier are applied in the final stage, wherein the FC layers connect all the neurons in one layer to another and it produces a final representation of the features generated from the matrix I . The softmax classifier generates the probability of the local features, and the vector obtained is given by,

$$O = \left[O_1, O_2, ..., O_a\right] \quad (7)$$

3.1.6.. BiLSTM layer

The output of the CNN layer O is forwarded to the BiLSTM layer for extracting the contextual data from the input sentence. The BiLSTM [1] is used as it is effective in preserving future as well as past information. The bidirectional aspect of BiLSTM is based on the notion that any word in a sentence is related to both the succeeding and preseeding words. The conceptual information of any word in the sentence is captured using the BiLSTM, and so the combination of CNN and BiLSTM enables the encoding of tokens on the basis of both contextual and local features. The BiLSTM utilized here contains layers, like embedding, BiLSTM, and dropout layers. The embedding layer is used to learn the word embeddings by mapping the token index sequence to corresponding embedding vectors. Next, from the embedding vectors, the dependence of the tokens in the sentence in the long term is captured by the BiLSTM layers. Further, dropout is used to avoid overfitting, and it is applied to drop complete rows of the embedded matrix, thereby discouraging the network from relying upon particular tokens and introducing language-specific regularisation. The output of the BiLSTM is then passed to the attention layer.

3.1.7. Attention layer

The attention mechanism [1] is introduced in NLP to enhance the efficiency of the NLP models by targeting the most appropriate position in the sentence rather than the inappropriate one. It is based on the principle that humans pay high attention to the significant words in the document while reading them. The application of the attention layer lightens the long-term dependency problems which arise while handling long input sequences. In this layer, the score function is computed for all tokens generated from the input sentence. The score is determined for the target token $token_b$ based on the similarity with other tokens in the input sentence $\{token_1, token_2, ...token_b, ..., token_K\}$, as follows,

$$Score\left(token_b, token_i\right) = \omega_a \left| token_b - token_i \right| \quad (8)$$

where, ω_a designates the trainable weight matrix. Later, a context vector is generated for the target token by calculating the weighted sum of the attention weight $\delta_{z,b}$ multiplied by hidden states λ_z , using the expression below,

$$g_z = \sum_b \delta_{z,b} \lambda_z \quad (9)$$

Thereafter, a word representation is created by concatenating the hidden states and eth context

vector for all target tokens $O_z = [\lambda_z : g_z]$, and from this, the tagging score is determined.

3.1.8. NER layer

After determining the tagging score, it is applied to the NER layer, which computes the probability of the K tagging scores generated. The NER layer comprises multiple neurons, and the tagging score generated for each token is applied to all neurons. Thereafter, the relative probability of each tagging score is computed, and based on the probability value, the named entity class is recognized for the corresponding target token.

3.2. Optimization for layer size determination

A major problem associated with NER-Net is determining the layer size of the various layers in the NER-Net, and to address this, a novel technique named FHAVOA is designed. The proposed FHAVOA is created by combining the FHO [18] algorithm and the AVOA [19]. Here, the FHAVOA is used to determine the number of CNN, and BiLSTM layers, as well as the number of BiLSTM cells, and the kernel size of various layers in CNN.

3.2.1. Solution encoding

Solution encoding is used to represent the solutions of any optimization technique pictorially. The FHAVOA is used to determine the number of BiLSTM layers, CN layers, BiLSRM cells, and the kernel size of various layers in CNN layer, and these aspects are represented using the solution encoding represented in figure 3.

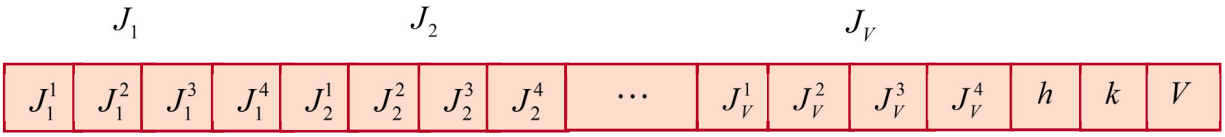


Figure 3. Solution encoding of FHAVOA

The parameters that have to be optimally tuned using the FHAVOA are depicted using solution encoding. Here, J_1, J_2, and J_V represents the kernel size CNN layer 1, layer 2, and layer V respectively. Further, J_1^1, J_1^2, J_1^3, and J_1^4 stipulates the kernel size of the conv, BN, ReLU, and maxpooling layer in the first CNN layer. Moreover, h and V signifies the number of BiLSTM and CNN layers, and k denotes the number of BiLSTM cells.

3.2.2. Proposed FHAVOA for layer size determination

The layer size of the NER-Net is optimally tuned using the proposed FHAVOA algorithm, which is formulated by assimilating FHO [18] in AVOA [19]. The FHO is a population-based optimization technique that is conceived based on the foraging conduct of fire hawks, which includes brown falcons, black kites, and whistling kites. The FHO aims to determine solution to optimization problems based on the behavior of fire hawks while catching prey by carrying fire sticks in their talons and beaks. The FHO is realized using various stages, like computation of distance, setting fires, prey movement inside and outside the territory, and safe territory identification. The FHO is computationally efficient and has a fast convergence rate, although it did not consider solving complex constraint problems. On the other hand, AVOA [19] is devised based on the hunting and navigating conduct of African vultures. This algorithm is executed in four stages: identifying the ideal vulture, computing the starvation rate, exploration, and exploitation. AVOA has high flexibility and is effective in solving highly complex optimization issues. Thus, by incorporating AVOA in FHO, the proposed FHAVOA can effectively solve constrained problems with a high convergence rate. The following steps detail the execution of the FHAVOA.

1) Initialization: The prime step in the execution of the FHAVOA is to initialize the location of eth fire hawks, which indicate the solutions in the search space, as follows,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$Y=\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_k \\ \vdots \\ Y_J \end{bmatrix}=\begin{bmatrix} Y_1^1 & Y_1^2 & \cdots & Y_1^l & \cdots & Y_1^w \\ Y_2^1 & Y_2^2 & \cdots & Y_2^l & \cdots & Y_2^w \\ \vdots & \vdots & \ddots & \vdots & \iddots & \vdots \\ Y_k^1 & Y_k^2 & \cdots & Y_k^l & \cdots & Y_k^w \\ \vdots & \vdots & \iddots & \vdots & \ddots & \vdots \\ Y_J^1 & Y_J^2 & \cdots & Y_J^l & \cdots & Y_J^w \end{bmatrix} \quad \begin{cases} k=1,2,...,J \\ l=1,2,..,w \end{cases} \tag{10}$$

where, J denotes the population size, w is the problem dimensionality, and Y_k signifies the position of the k^{th} fire hawk, the initial position of the l^{th} decision variable of k^{th} solution candidate is given by,

$$Y_k^l(0)=Y_{k,\min}^l+rand\left(Y_{k,\max}^l-Y_{k,\min}^l\right), \quad \begin{cases} k=1,2,...,J \\ l=1,2,..,w \end{cases} \tag{11}$$

Here, $rand$ indicates an arbitrary number randomly distributed in $[0,1]$, $Y_{k,\max}^l$ and $Y_{k,\min}^l$ denotes the maximum and minimum limits of the l^{th} decision variable of k^{th} solution.

2) Fitness estimation: The NER-Net should function in such a way that the entity should be recognized with the least error, and so the optimization is aimed at determining the layer dimension, which produce the least error, and so MSE is considered as the fitness, and is expressed as,

$$Fit=MSE=\frac{1}{n}\sum_{i=1}^{n}\left(H_i^*-H_i\right)^2 \tag{12}$$

where, n is the sample count, H_i^* and H_i denotes the anticipated and actual outcomes of the NER-Net.

3) Identify prey and fire hawk: After the objective is evaluation, few of the solution with best objectives are considered as fire hawks, and the rest are assumed as prey. The fire hawks spread around the prey for making the hunting process easy, and the prey and fire hawks are represented as,

$$FH=\begin{bmatrix} FH_1 \\ FH_2 \\ \vdots \\ FH_s \\ \vdots \\ FH_u \end{bmatrix}, \qquad s=1,2,..,u \tag{13}$$

$$PR=\begin{bmatrix} PR_1 \\ PR_2 \\ \vdots \\ PR_t \\ \vdots \\ PR_v \end{bmatrix}, \qquad t=1,2,..,v \tag{14}$$

Here, u and v indicate the total count of Fire Hawks and prey, and FH_s represents the s^{th} fire hawk, and PR_t designates the t^{th} prey.

4) Compute distance: Later the distance among the prey and Fire hawks is computed, in order to determine the nearest prey, thereby differentiating the territorial domain of the prey. Here, the prey closest to the optimal Fire hawk is identified, and the other preys are used to determine the territory of the remaining birds. The distance is measured as,

$$G_t^s=\sqrt{\left(x_2-x_1\right)^2+\left(y_2-y_1\right)^2}, \quad \begin{cases} s=1,2,...,u \\ t=1,2,...,t \end{cases} \tag{15}$$

Here, the distance among the t^{th} prey and the s^{th} fire hawk is designated as G_t^s, and (x_1,y_1)

represents the position of the s^{th} fire hawk, and (x_2, y_2) characterizes the location of the t^{th} prey.

5) Update location: The fire hawks set fire in the chosen area by collecting burning sticks from the primary fire in their territory or from other territories, thereby making the prey flee the specific area. This behaviour is used to update the location of the solution, and is formulated using the succeeding expression.

$$FH_s^n = FH_s + \left(e_1 \times N - e_2 \times FH_{nr}\right) \ , s = 1,2,..,u \tag{16}$$

Here, FH_s^n refers to the updated position of the s^{th} fire hawk, e_1 and e_2 are random numbers uniformly distributed in $[0,1]$, FH_{nr} refers to another fire hawk, N denotes the global best solution or the main fire.

Now consider $FH_s^n = FH(x+1)$, $FH_s = FH(x)$, and $FH_{nr} = FH_{nr}(x)$, then the equation (16)can be rephrased as,

$$FH(x+1) = FH(x) + \left(e_1 \times N - e_2 \times FH_{nr}(x)\right) \tag{17}$$

The ability of FHO in solving complex problems can be improved by incorporating AVOA, and from AVOA [19],

$$X(x+1) = \eta(x) - \vartheta(x) \times W \tag{18}$$

$$\vartheta(x) = \left|\rho \times \eta(x) - X(x)\right| \tag{19}$$

Considering $\eta(x) > X(x)$,

$$X(x+1) = \eta(x) - \left(\rho \times \eta(x) - X(x)\right) \times W \tag{20}$$

$$X(x+1) = \eta(x)\left(1 - \rho \times W\right) + X(x) \cdot W \tag{21}$$

Let $X(x+1) = FH(x+1)$, $X(x) = FH(x)$, and $\eta(x) = FH_{best}(x)$, then

$$FH(x+1) = FH_{best}(x)\left(1 - \rho \times W\right) + FH(x) \cdot W \tag{22}$$

$$FH(x) = \frac{FH(x+1) - FH_{best}(x)\left(1 - \rho \times W\right)}{W} \tag{23}$$

Applying equation (23) in equation (17),

$$FH(x+1) = \frac{FH(x+1) - FH_{best}(x)\left(1 - \rho \times W\right)}{W} + \left(e_1 \times N - e_2 \times FH_{nr}(x)\right) \tag{24}$$

$$FH(x+1) - \frac{FH(x+1)}{W} = \left(\frac{FH_{best}(x)\left(\rho \times W - 1\right)}{W}\right) + \left(e_1 \times N - e_2 \times FH_{nr}(x)\right) \tag{25}$$

$$FH(x+1)\left(\frac{W-1}{W}\right)=\frac{FH_{best}(x)(\rho\times W-1)(e_1\times N-e_2\times FH_{nr}(x))\cdot W}{W} \quad (26)$$

$$FH(x+1)=\frac{FH_{best}(x)(\rho\times W-1)(e_1\times N-e_2\times FH_{nr}(x))\cdot W}{W-1} \quad (27)$$

Here, FH_{best} refers to the best vulture in the x^{th} iteration, and W refers to the satiation rate of the vulture. The equation (28) is utilized to update the position of the Fire hawks in FHAVOA.

6) Prey movement: As the fire hawks sets fire, the prey starts moving inside the Fire hawk's territory, as it hides, runs away or towards the fire hawk, and this movement is formulated as,

$$PR_y^n=PR_y+(e_3\times FH_s-e_4\times SF_s),\quad y=1,2,...o \quad (28)$$

In some cases, the prey run towards the territorial area of other fire hawks, and this movement is modelled as,

$$PR_y^n=PR_y+(e_5\times FH_{other}-e_6\times SF),\quad y=1,2,...,o \quad (29)$$

Here, PR_y^n refers to the updated position of the t^{th} prey, $e_i, i=3,4,5,6$ are random numbers uniformly distributed in $[0,1]$, SF_s and SF represents the safe place in and outside the fire hawks territory, which are formulated as,

$$SF_s=\frac{\sum_{y=1}^{v}PR_y}{v},\ ,\quad y=1,2,...o \quad (30)$$

$$SF_s=\frac{\sum_{m=1}^{\gamma}PR_m}{\gamma},\ ,\quad y=1,2,...\gamma \quad (31)$$

where, PR_m indicates the m^{th} prey, and PR_y refers to the y^{th} prey engulfed by the s^{th} fire hawk.

7) Fitness estimation: After updating the position, the fitness is recomputed using equation (12), and the one with minimal fitness is considered as the ideal one.

8) Termination: The above process is executed repetitively till the stopping criteria is reached. Algorithm1 depicts the pseudocode of the FHAVOA.

Algorithm 1. Pseudocode of FHAVOA
Begin
Compute initial positions of search agents $Y_k\ k=1,2,..,J$
Estimate fitness with equation (12)
Identify the global best solution N
While $x<x_{\max}$
Generate a random number u (Fire hawk count)
Identify PR and FH
Compute G_t^s using equation (15)
Identify the fire hawks territory
for $s=1:u$

Update fire hawk's position using equation (27)
for $y = 1 : o$
Determine safe place using equation (30)
Update prey's position using equation (28)
Determine safe place using equation (31)
Update prey's position using equation (29)
end
end
Estimate fitness with equation (12)
Identify the global best solution N
endwhile
Return N
End

Thus the FHAVOA effectively determines the layer size and kernels of various layers in the NER-Net. Further, the inclusion of AVOA in FHO effectively increased the convergence speed of the FHAVOA.

3.3. Proposed architecture

This section depicts the architecture of the NER-Net, when an input sentence "My aadhar number is 12345" is applied, and the obtained architecture is portrayed in figure 4. At first, the input sentence is converted into five tokens "My", "aadhar", "number", "is", and "12345". Later, the generated tokens are applied to the word-level and character-level encoders. The word level encoder generates five vectors corresponding to the five tokens, each of dimension 1×1. Here, the tokens "My", "aadhar", "number", "is", and "12345" are converted into unit vectors with real values "34", "124532", "392376", "24", and "67890". Similarly, the tokens applied to the word-level encoder are also converted into real-valued vectors, however, here vectors of size 1×2 are produced. Here, the vectors are generated by considering a pair of adjacent tokens. For instance, consider the adjacent tokens "My" and "aadhar", the real-valued vector is produced by considering these two tokens, and a vector $[15 \quad 2536]$ is produced. In the similar way, the vectors are generated for all token pairs, but the token "12345" has no succeeding term and so "0" is appended to the token and a corresponding vector $[6789 \quad 0]$ is generated. A total of 5 vectors each are generated by the character-level and word-level encoders, and thus two vectors with dimension 5×1 and 5×2 are produced. These vectors are concatenated to obtain a 5×3 matrix, which is subjected to the CNN for generating low-level features. A total of V layers are available, which comprises different layers, like conv, BN, ReLU, MaxPool, FC, and Softmax layers. The dimensions of the layers optimally tuned using the FHAVOA is depicted in the architecture depicted in figure 4 which portrays the NER-Net architecture. Thereafter, BiLSTM layer is used to extract the contextual features in the input data. Finally, an attention layer is applied and accordingly a tagging score is produced. Thereafter, NER layer determines the relative probability of the tokens, and as the NER-Net used here is used for recognizing the numerical entity, and a maximum probability is generated for the token 5, it is assigned a value of 1, and the remaining entities are allocated a value '0'.

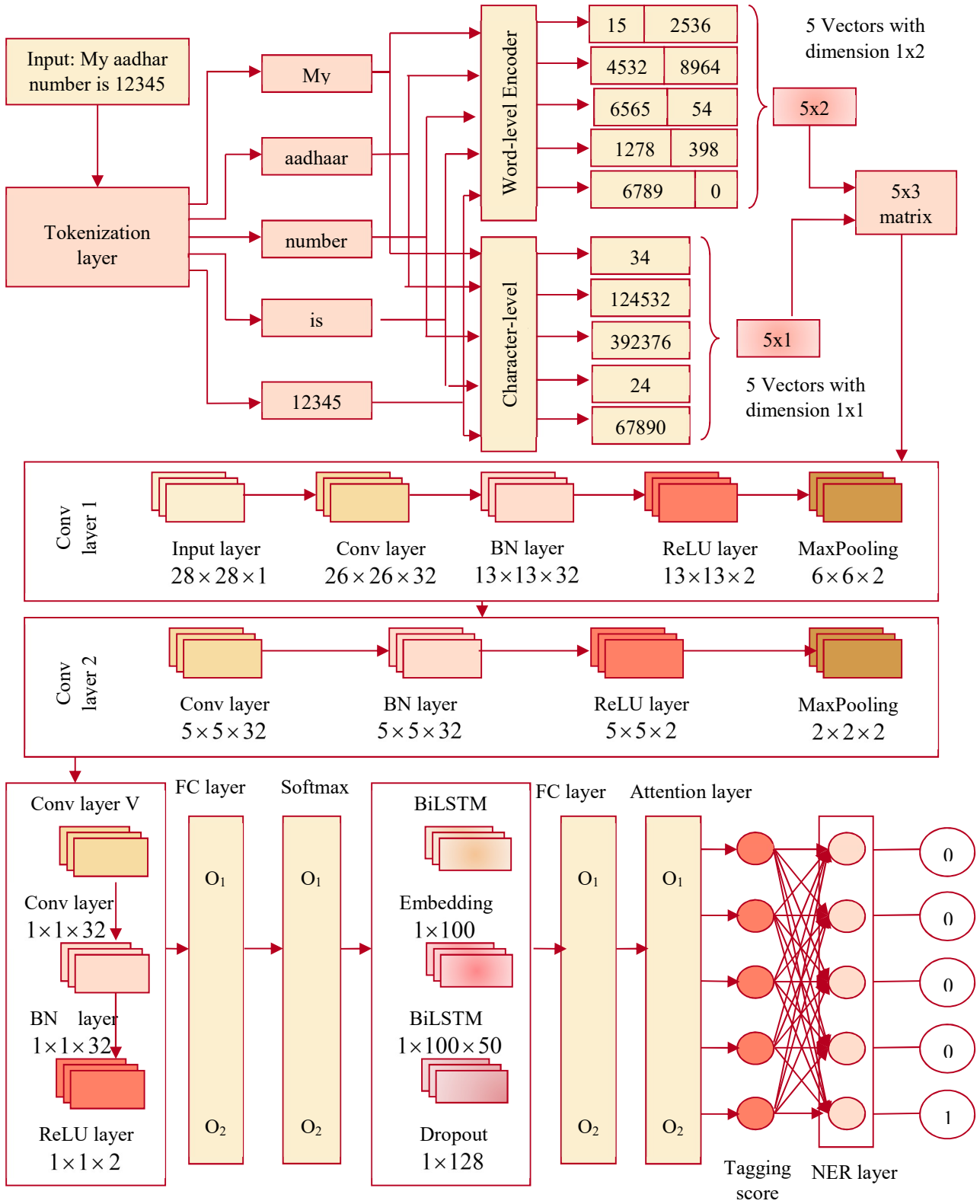


Figure 4. NER-Net architecture for specific input sentence

4. RESULTS AND DISCUSSION

The experimental results obtained during the analysis of the proposed NER-Net-FHAVOA are exemplified in this segment, along with the evaluation metrics used for analysis.

4.1. Experimental set-up

The experiments of the NER-Net are conducted on a system by implementing it in a Python environment.

4.2. Dataset description

The NER-Net's performance is evaluated considering the two data sets mentioned below;

i) NER_dataset: This dataset [20] is one of the cleanest databases available, and it is of size 1M x 4 and contains various columns, such as Tag, Pos, word, Sentence, and all entities are congregated by #Sentence. Here, the English dictionary words taken from the sentence are contained in the word column, and Parts of speech are available in POS. Further, Standard named entity recognition tags, like location, facility, person, organization, etc. is present in the tag column.

ii) Entity-recognition-datasets: This database [21] encompasses datasets obtained from various fields, which are annotated with a multitude of entity kinds, and are utilized for NER and entity recognition. Here, various domains, such as news, wiki, Wikipedia, medical, anatomical, Twitter, social media, and so on.

4.3. Evaluation measures

The NER-Net's efficiency is evaluated in view of a multitude of parameters, lik accuracy, PPV, NPV, RMSE, and MSE, and these parameters are detailed below.

i) Accuracy: This factor is used to estimate the superiority of the NER-Net in detecting the named entity accuractely, and is computed by,

$$Accuracy = \frac{P_1 + N_1}{P_1 + N_1 + P_2 + N_2} \quad (32)$$

where, P_1 refers to true positive, and P_2 signifies true negative, while N_1 denotes true negative and N_2 represents false negative.

ii) PPV: It determines the ratio of the precisely detected numerical entity to the total entities that are categorized as numerical, and is expressed as,

$$PPV = \frac{P_1}{P_1 + P_2} \quad (33)$$

iii) NPV: This factor characterizes the ratio of entities that are accurately identified as non-numerical to the total count of entities that are categorized as non-numerical, and is formulated as,

$$NPV = \frac{N_1}{N_1 + N_2} \quad (34)$$

iv) MSE: MSE indicates the average of the squared difference between the expected and the actual output produced by NER-Net, and is given by equation (12).

v) RMSE: This metric is computed by taking the square root of MSE, and is expressed as,

$$RMSE = \sqrt{MSE} \quad (35)$$

4.4. Comparative techniques

The NER-Net is examined for its efficacy by testing it with the results produced by the available techniques of NER, like CNN-BiLSTM [1], TL [2], BiLSTM-CRF [3], and ASTRAL [4], on the basis of the five metrics.

4.5. Comparative assessment

The quantitative results of the NER-Net are assessed by considering two scenarios: using the data obtained from NER_dataset, and the data gathered from the Entity-recognition dataset by altering the training data.

i) Assessment using NER_dataset

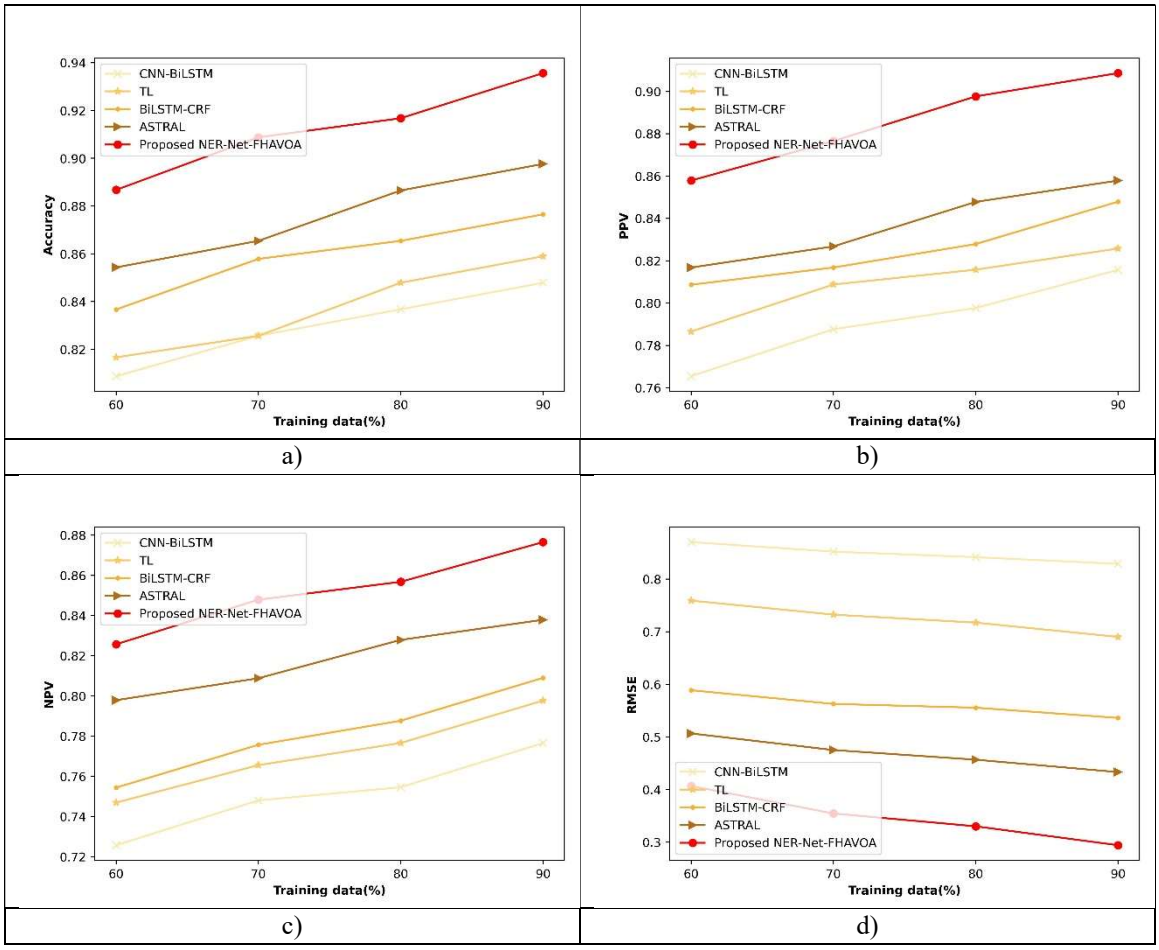
Figure 5 indicates the assessment of the NER-Net-FHAVOA based on the data acquired from the NER_dataset. In figure 5a), the NER-Net-FHAVOA is examined considering the accuracy parameter. The NER-Net-FHAVOA achieved an

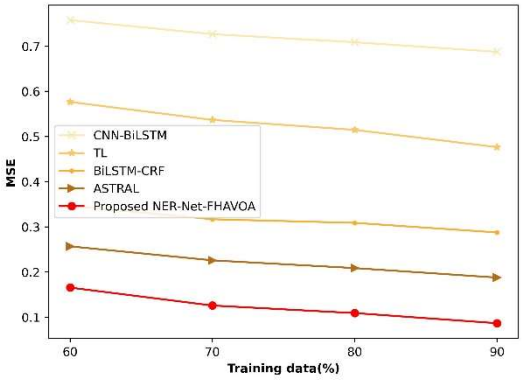
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

accuracy of 0.909, while the value figured by the prevailing techniques, like CNN-BiLSTM, TL, BiLSTM-CRF, and ASTRAL is 0.826, 0.826, 0.858, and 0.865, with 70% training data. The NER-Net-FHAVOA is thus deduced to attain a performance improvement of 9.14%, 9.14%, 5.60%, and 4.77%. Further, the examination of the NER-Net-FHAVOA in view of PPV is exhibited in figure 5b). The values of PPV computed by CNN-BiLSTM, TL, BiLSTM-CRF, ASTRAL, and NER-Net-FHAVOA is 0.788, 0.809, 0.817, 0.827, and 0.877, respectively with 70% training data.

This shows that the NER-Net-FHAVOA succeeded in attaining a higher value of PPV by 10.14%, 7.73%, 6.82%, and 5.68%. Likewise, the NPV-based analysis of the NER-Net-FHAVOA is demonstrated in figure 5c). The NER-Net-FHAVOA figured an NPV value of 0.848 with 70% training data. But the NPV values measured by the existing techniques, such as CNN-BiLSTM is 0.748, TL is 0.766, BiLSTM-CRF is 0.776, and ASTRAL is 0.809, which is lower by 11.78%, 9.71%, 8.52%, and 4.61% than the NPV of the NER-Net-FHAVOA. In figure 5d), the investigation of the NER-Net-FHAVOA with regard to RMSE is illustrated.

The RMSE value attained with 70% training data by CNN-BiLSTM, TL, BiLSTM-CRF, ASTRAL, and NER-Net-FHAVOA is 0.852, 0.733, 0.563, 0.475, and 0.355, respectively. Similarly, the NER-Net-FHAVOA is examined for its efficacy based on MSE, and this is displayed in figure 5e). With 70% training data, the MSE value realized by NER-Net-FHAVOA is 0.126, but the MSE values attained by CNN-BiLSTM is 0.727, TL is 0.537, BiLSTM-CRF is 0.317, and ASTRAL is 0.226.





e)

Figure 5. Analysis considering NER-dataset based on a)accuracy, b) PPV, c) NPV, d) RMSE, and e) MSE

ii) Assessment using Entity-recognition dataset

The quantitative analysis of the NER-Net-FHAVOA using the data from Entity-recognition dataset is depicted in figure 6. The evaluation of the NER-Net-FHAVOA concerning accuracy is exhibited in figure 6a). The methods, like CNN-BiLSTM, TL, BiLSTM-CRF, ASTRAL, and NER-Net-FHAVOA totalled an accuracy of 0.788, 0.827, 0.837, 0.858, and 0.909, with 80% training data, which reveals that the NER-Net-FHAVOA is effective in attaining a superior accuracy by 13.32%, 9.01%, 7.91%, and 5.59%. Figure 6b) presents a PPV-based appraisal of the NER-Net-FHAVOA. For 80% of training data, the value of PPV attained by NER-Net-FHAVOA is 0.865, while the approaches, like CNN-BiLSTM, TL, BiLSTM-CRF, and ASTRAL quantified PPV values of 0.788, 0.809, 0.817, and 0.837. This depicts that the PPV value attained by NER-Net-FHAVOA is better by 8.99%, 6.55%, 5.62%, and 3.30% than the available methods. In figure 6c), the inspection of the NER-Net-FHAVOA with respect to NPV is displayed. The NPV figured by CNN-BiLSTM is 0.758, TL is 0.777, BiLSTM-CRF is 0.787, ASTRAL is 0.808, and NER-Net-FHAVOA is 0.836, for 80% training data. Hence, the NER-Net-FHAVOA is observed to produce a superior NPV value by 9.31%, 7.07%, 5.88%, and 3.33%. Likewise, the NER-Net-FHAVOA is scrutinized for its efficiency concerning RMSE, which is illustrated in figure 6d). The RMSE value attained by NER-Net-FHAVOA is 0.420, while the values achieved by CNN-BiLSTM is 0.888, TL is 0.767, BiLSTM-CRF is 0.622, and ASTRAL is 0.556. Further, the MSE-oriented scrutiny of the NER-Net-FHAVOA is exhibited in figure 6e). The MSE value attained by CNN-BiLSTM, TL, BiLSTM-CRF, ASTRAL, and NER-Net-FHAVOA is 0.788, 0.588, 0.387, 0.309, and 0.177, respectively, with 80% training data.

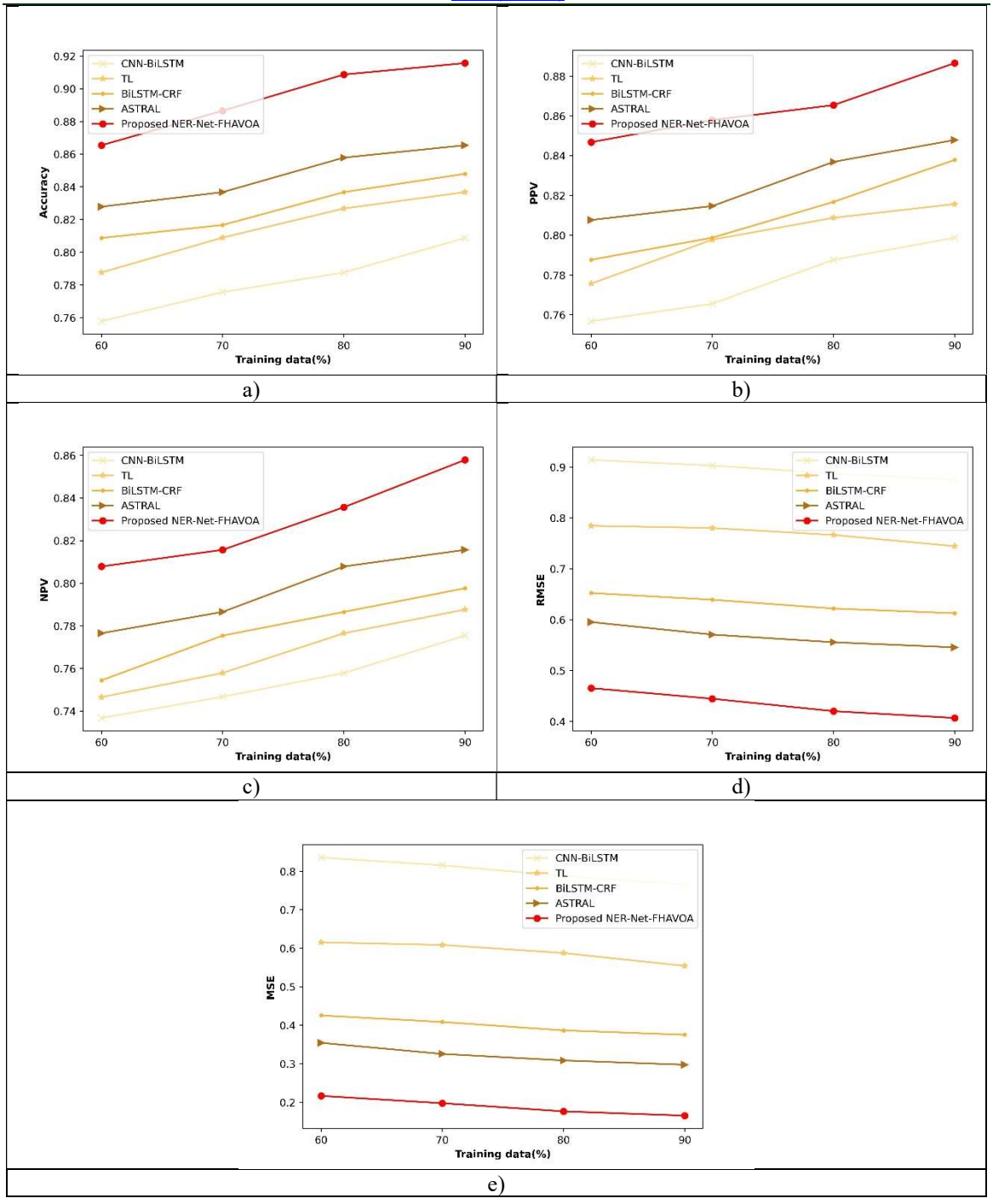


Figure 6. Analysis considering Entity-recognition dataset based on a)accuracy, b) PPV, c) NPV, d) RMSE, and e) MSE

4.6. Comparative discussion

The NER-Net-FHAVOA is investigated based on various metrics to determine the superiority of the approach in recognizing named entities. Here, the investigation is carried out by correlating the method with various approaches of NER, and the quantitative summary is represented in table 1. The quantitative results are provided by considering data acquired from the NER_dataset and the Entity-recognition dataset, and with a training data of 90%. From the table, it can be

deduced that the NER-Net-FHAVOA is effective in attaining a high accuracy of 0.936. PPV of 0.909, NPV of 0.877, low RMSE of 0.294, and MSE of 0.087. But the prevailing methods, like CNN-BiLSTM, TL, BiLSTM-CRF, and ASTRAL attained accuracy values of 0.848, 0.859, 0.877, and 0.898. Likewise, the value of PPV measured by CNN-BiLSTM is 0.816, TL is 0.826, BiLSTM-CRF is 0.848, and ASTRAL is 0.858. Further, a low NPV is attained by CNN-BiLSTM at 0.777, TL at 0.798, BiLSTM-CRF at 0.809, and ASTRAL at 0.838. The CNN-BiLSTM, TL, BiLSTM-CRF, and ASTRAL computed high RMSE values of 0.829, 0.690, 0.536, and 0.433, and MSE values of 0.688, 0.476, 0.288, and 0.188, respectively. This reveals the superiority of the NER-Net-FHAVOA in recognizing named entities, which is mostly due to the utilization of CNN and BiLSTM in determining the local and contextual features, along with the optimal tuning of the layer dimensions using the FHAVOA.

Table 1. Comparative discussion of NER-Net

Dataset	*Metrics*	*CNN-BiLSTM*	*TL*	*BiLSTM-CRF*	*ASTRAL*	*Proposed NER-Net-FHAVOA*
NER_dataset	***Accuracy***	0.848	0.859	0.877	0.898	**0.936**
	PPV	0.816	0.826	0.848	0.858	**0.909**
	NPV	0.777	0.798	0.809	0.838	**0.877**
	RMSE	0.829	0.690	0.536	0.433	**0.294**
	MSE	0.688	0.476	0.288	0.188	**0.087**
Entity-recognition dataset	***Accuracy***	0.809	0.837	0.848	0.865	0.916
	PPV	0.799	0.816	0.838	0.848	0.887
	NPV	0.775	0.788	0.798	0.816	0.858
	RMSE	0.875	0.745	0.613	0.546	0.407
	MSE	0.765	0.554	0.375	0.298	0.165

5. CONCLUSION

This work proposed a novel deep learning architecture named NER-Net for identifying numerical entities in the input sentence. The NER-Net comprises multiple layers, like tokenization, IDF, CNN, BiLSTM, attention, and NER layers. Here, the input sentence is initially applied to the tokenization layer for converting into tokens. Later, the tokens generated are subjected to the IDF layer, wherein two encoders, such as character-level encoder and word-level encoder are used to convert the tokens into character-level and word-level encodings. The character-level and word-level encoded data are then concatenated to form a matrix, which is then forwarded to the CNN, and BiLSM layer for generating local and contextual features. Thereafter, a tagging score is generated by the attention layer, which is mapped to the entity class in the NER layer for identifying the numerical entity. Further, an algorithm named FHAVOA is devised for determining the layer dimensions. Moreover, the analysis shows superior values of accuracy at 0.936. PPV at 0.909, NPV at 0.877, RMSE at 0.294, and MSE at 0.087 are achieved. Further course of action may include the application of TL for addressing the issues related to scarcity of data.

REFERENCES

[1] Cho, M., Ha, J., Park, C. and Park, S., "Combinatorial feature embedding based on CNN and LSTM for biomedical named entity recognition", Journal of biomedical informatics, vol.103, pp.103381, 2020.

[2] Francis, S., Van Landeghem, J. and Moens, M.F., "Transfer learning for named entity recognition in financial and biomedical documents", Information, vol.10, no.8, pp.248, 2019.

[3] Oral, B., Emekligil, E., Arslan, S. and Eryiğit, G., "Information extraction from text intensive and visually rich banking documents", Information Processing & Management, vol.57, no.6, pp.102361, 2020.

[4] Wang, J., Xu, W., Fu, X., Xu, G. and Wu, Y., "ASTRAL: adversarial trained LSTM-CNN for named entity recognition. Knowledge-Based Systems", vol.197, pp.105842, 2020.

[5] Yan, H., Deng, B., Li, X. and Qiu, X., "TENER: adapting transformer encoder for named entity recognition", arXiv preprint arXiv:1911.04474, 2019.
[6] Lin, B.Y., Lee, D.H., Shen, M., Moreno, R., Huang, X., Shiralkar, P. and Ren, X., "Triggerner: Learning with entity triggers as explanations for named entity recognition", arXiv preprint arXiv:2004.07493, 2020.
[7] Jie, Z. and Lu, W., "Dependency-guided LSTM-CRF for named entity recognition", arXiv preprint arXiv:1909.10148, 2019.
[8] Peng, Q., Zheng, C., Cai, Y., Wang, T., Xie, H. and Li, Q., "Unsupervised cross-domain named entity recognition using entity-aware adversarial training", Neural Networks, vol.138, pp.68-77, 2021.
[9] Li, J., Sun, A., Han, J. and Li, C., "A survey on deep learning for named entity recognition", IEEE Transactions on Knowledge and Data Engineering, vol.34, no.1, pp.50-70, 2020.
[10] Yadav, V. and Bethard, S., "A survey on recent advances in named entity recognition from deep learning models", arXiv preprint arXiv:1910.11470, 2019.
[11] Cho, H. and Lee, H., "Biomedical named entity recognition using deep neural networks with contextual information", BMC bioinformatics, vol.20, no.1, pp.1-11, 2019.
[12] Chen, H., Lin, Z., Ding, G., Lou, J., Zhang, Y. and Karlsson, B., "GRN: Gated relation network to enhance convolutional neural network for named entity recognition", In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, No. 01, pp. 6236-6243, July 2019.
[13] Liu, T., Yao, J.G. and Lin, C.Y., "Towards improving neural named entity recognition with gazetteers", In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 5301-5307, July 2019.
[14] Naseem, U., Khushi, M., Reddy, V., Rajendran, S., Razzak, I. and Kim, J., "Bioalbert: A simple and effective pre-trained language model for biomedical named entity recognition", In Proceedings of 2021 International Joint Conference on Neural Networks (IJCNN), IEEE, pp. 1-7, July 2021.
[15] Bose, P., Srinivasan, S., Sleeman IV, W.C., Palta, J., Kapoor, R. and Ghosh, P., "A survey on recent named entity recognition and relationship extraction techniques on clinical texts", Applied Sciences, vol.11, no.18, pp.8319, 2021.
[16] Yao, R., Wang, N., Liu, Z., Chen, P. and Sheng, X., "Intrusion detection system in the advanced metering infrastructure: a cross-layer feature-fusion CNN-LSTM-based approach", Sensors, vol.21, no.2, pp.626, 2021.
[17] Do Xuan, C., Dao, M.H. and Nguyen, H.D., "APT attack detection based on flow network analysis techniques using deep learning", Journal of Intelligent & Fuzzy Systems, vol.39, no.3, pp.4785-4801, 2020.
[18] Azizi, M., Talatahari, S. and Gandomi, A.H., "Fire hawk optimizer: A novel metaheuristic algorithm". Artificial Intelligence Review, pp.1-77, 2022.
[19] Abdollahzadeh, B., Gharehchopogh, F.S. and Mirjalili, S., "African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems." Computers & Industrial Engineering, vol.158, pp.107408, 2021.
[20] NER_dataset is taken from "https://www.kaggle.com/datasets/namanj27/ner-dataset" accessed on January 2023.
[21] Entity-recognition-datasets is taken from "https://github.com/juand-r/entity-recognition-datasets" accessed on January 2023.
[22] Wawrzik, F., Rafique, K.A., Rahman, F. and Grimm, C., "Ontology Learning Applications of Knowledge Base Construction for Microelectronic Systems Information", Information, vol.14, no.3, pp.176, 2023.
[23] Mota, P., Cabarrão, V. and Farah, E., "Fast-paced improvements to named entity handling for neural machine translation", In Proceedings of the 23rd Annual Conference of the European Association for Machine Translation, pp. 141-149, June 2022.
[24] Yin, D., Cheng, S., Pan, B., Qiao, Y., Zhao, W. and Wang, D., "Chinese named entity recognition based on knowledge based question answering system", Applied Sciences, vol.12, no.11, pp.5373, 2022.
[25] Khademi, M.E. and Fakhredanesh, M., "Persian automatic text summarization based on named entity recognition", Iranian Journal of Science and Technology,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Transactions of Electrical Engineering, pp.1-12, 2020.

[26] Brandsen, A., Verberne, S., Lambers, K. and Wansleeben, M., "Can BERT Dig It? Named Entity Recognition for Information Retrieval in the Archaeology Domain", Journal on Computing and Cultural Heritage (JOCCH), vol.15, no.3, pp.1-18, 2022.

[27] Cheng, P. and Erk, K., "Attending to entities for better text understanding", In Proceedings of the AAAI conference on artificial intelligence, Vol. 34, No. 05, pp. 7554-7561, April 2020.

[28] Yadav, A. and Vishwakarma, D.K., "A Deep Language-independent Network to analyze the impact of COVID-19 on the World via Sentiment Analysis", arXiv preprint arXiv:2011.10358, 2020.