

Named Entity Recognition in Bangla

submitted by

Sayan Banerjee

Class Roll No: 002210503041

Exam Roll: MCA244040

Master of Computer Application

under the supervision of

Dr. Sudip Kumar Naskar

in the

Department of Computer Science and Engineering

Faculty of Engineering and Technology

Jadavpur University

2024

Faculty of Engineering and Technology
Jadavpur University

CERTIFICATE OF RECOMMENDATION

To Whom It May Concern

I hereby recommend that the project report entitled “Named Entity Recognition in Bangla” has been carried out by Sayan Banerjee (Reg. No.: 163633 of 2022-23, Roll No.: 002210503041, Exam Roll: MCA244040), under my guidance and supervision and be accepted in partial fulfilment of the requirement for the degree of Master of COMPUTER APPLICATION in DEPARTMENT of COMPUTER SCIENCE and ENGINEERING, JADAVPUR UNIVERSITY during the academic year 2023-2024.

Dr. Sudip Kumar Naskar
Project Supervisor
Dept. of Computer Science and
Engineering
Jadavpur University

Prof. (Dr.) Nandini Mukhopadhyay
Head of the Department
Dept. of Computer Science & Engineering
Jadavpur University

Prof. Dipak Laha
Dean, Faculty Council
of Engineering & Technology
Jadavpur University, Kolkata-032

Faculty of Engineering and Technology
Jadavpur University

CERTIFICATE OF APPROVAL

This is to certify that the project report entitled “**Named Entity Recognition in Bangla**” is a bonafide record of work carried out by Sayan Banerjee, in fulfillment of the requirements for the award of the degree of Master of Computer Application in the Department of Computer Science and Engineering, Jadavpur University. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the project report only for the purpose for which it has been submitted.

Signature of Examiner 1

Date :

Signature of Examiner 2

Date :

Faculty of Engineering and Technology
Jadavpur University

**DECLARATION OF COMPLIANCE WITH
ACADEMIC ETHICS**

I hereby declare that this project report entitled "Named Entity Recognition in Bangla" is an outcome of my genuine and independent work, as part of my Master of Computer Application program.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, all materials and results that are not original to this work have been fully cited and referenced.

Sayan Banerjee

Registration No.: 163633 of 2022-23

Examination Roll No.: MCA244040

Thesis Title :- Named Entity Recognition in Bangla

Signature with Date

ACKNOWLEDGEMENT

With my most sincere and gratitude, I would like to thank Dr. Sudip Kumar Naskar, Department of Computer Science & Engineering, my supervisor, for his overwhelming support throughout the duration of the project. His motivation always gave me the required inputs and momentum to continue with my work, without which the project work would not have taken its current shape. His valuable suggestions and numerous discussions have always inspired new ways of thinking. I feel deeply honored that I got this opportunity to work under him.

I would like to express my sincere thanks to all my teachers for providing sound knowledge base and cooperation.

I would like to thank all the faculty members of the Department of Computer Science & Engineering of Jadavpur University for their continuous support.

Sayan Banerjee

Class Roll No.: 002210503041

Examination Roll No.: MCA244040

Department of Computer Science & Engineering
Jadavpur University

Abstract

Named Entity Recognition (NER) is a crucial task in natural language processing (NLP) that involves identifying and categorizing named entities within a body of text into predefined categories such as persons, organizations, locations, dates, and more. While substantial progress has been made in NER for high-resource languages like English, the development of NER systems for Bangla, a widely spoken language with complex morphology and syntax, remains challenging. In the context of the Bengali (Bangla) language, NER aims to extract and classify named entities specific to this linguistic and cultural context. The application of NER in Bangla is essential for numerous NLP tasks, including information extraction, question answering, and text summarization, which are particularly valuable for Bengali-speaking communities.

Several challenges associated with Bangla language, which make NER a complex task. The language's morphological complexity and syntactic structure pose significant hurdles. Additionally, there is a scarcity of well-annotated datasets for Bangla NER. Variations in how named entities are represented in Bangla, such as different spellings and forms, further complicate the task. The development and effectiveness of Bangla NER systems largely depend on the availability of resources like labeled datasets, language models, and linguistic tools specifically designed for the Bangla.

In this thesis, we are focusing on leveraging advanced deep learning models such as Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM) network, and BERT-based model. The aim is to improve the accuracy and reliability of NER systems in Bangla.

Contents

1. Introduction	1
1.1 Named Entity Recognition.....	1
1.2 Bangla NER.....	2
1.3 Challenges of Bangla NER.....	3
2. Literature Survey	5
2.1 Rule-based NER.....	5
2.2 Machine Learning-based NER.....	6
2.3 Deep Learning-based NER.....	6
3. Methodology	9
3.1 Word Embedding.....	9
3.1.1 Word2Vec.....	9
3.2 Convolution Neural Network.....	11
3.2.1 Convolution Layer.....	13
3.2.2 Pooling Layer.....	14
3.2.3 Fully Connected Layer.....	14
3.3 Bidirectional LSTM.....	15
3.5 BERT.....	17
3.6 Evaluation Metrics.....	19
3.6.1 Confusion Matrix.....	19
3.6.2 Recall.....	20
3.6.3 Precision.....	20
3.6.4 F1-Score.....	20
3.6.5 Accuracy.....	21

4. System Description	22
4.1 Dataset Description	22
4.2 Data Preprocessing	23
4.2.1 Tokenization	23
4.2.2 Word2Vec Model	24
4.2.3 Encoding and Padding	24
4.2.4 Splitting of Dataset	25
4.3 Model Architecture	26
4.3.1 The CNN Classifier	26
4.3.2 The Bidirectional LSTM Classifier	27
4.3.3 The BERT Classifier	28
5. Experiment and Results	30
5.1 Classification Report and Confusion Matrix	30
6. Conclusion and Future Work	34
6.1 Conclusion	34
6.2 Future Work	34

List of Figures

3.1 Different Word2Vec models.....	11
3.2 Basic architecture of an Artificial Neural Network	11
3.3 A simple CNN architecture.....	12
3.4 A simple cell of LSTM.....	15
3.5 Structure of Bidirectional LSTM.....	16
3.6 Masked Language Model (MLM).....	18
3.7 BERT Base Structure.....	19
4.1 Histogram of Sentence lengths.....	25
4.2 Flowchart of BERT architecture.....	29
5.1 Classification Report of CNN model.....	30
5.2 Confusion Matrix for CNN model on Test data.....	31
5.3 Classification Report of BiLSTM model.....	31
5.4 Confusion Matrix for BiLSTM model on Test data.....	32
5.5 Classification Report of BERT model.....	32
5.6 Confusion Matrix of BERT model on Test data.....	33

List of Tables

1.1 List of challenges of Bangla NER.....	4
4.1 Eight Named Entities of B-NER dataset.....	23
5.1 Comparing the performance of different models.....	33

Chapter 1

Introduction

1.1 Named Entity Recognition

Named Entity recognition (NER) is the component of information extraction that aims to identify and categorize named entities within unstructured text. NER involves the identification of key information in the text and classification into a set of predefined categories. The NER task first appeared in the Sixth Message Understanding Conference (MUC6) Sundheim (1995) [1] and involved recognition of entity names (people and organizations), place names, temporal expressions and numerical expressions. In MUC-6, Named entities (NEs) were categorized into three types of label, each of which uses specific attribute for a particular entity type. In the course of system development, people noticed that it is important to recognize information units like names, including person, organization and location names, and numeric expressions including time, date, money and percent expressions. Extracting these entities was recognized as one of the important sub-tasks of Information Extraction. As this task is relatively independent, it has been evaluated separately in several different languages, e.g. Japanese, Chinese and Spanish in MET (Multilingual Entity Tracking) project [2]. Outside the U.S., there have been several evaluation-based projects for NE, as one of the tasks of IREX [3] (Information Retrieval and Extraction Exercise) in Japan [Sekine and Isahara 2000] [IREX HP], and as the shared task in CoNLL in 2002 and 2003 for four languages, English, German, Dutch and Spanish [4]. In data mining, a named entity is a word or a phrase that clearly identifies one item from a set of other

items that have similar attributes. An entity is the thing that is consistently talked about or refer to in the text, such as person names, organizations, locations, time expressions, quantities, percentages and more predefined categories. NER system has various applications across different domains, including question answering, information retrieval and machine translation. NER plays an important role in enhancing the precision of other Natural Language Processing (NLP) tasks like part-of-speech tagging and parsing.

1.2 Bangla Named Entity Recognition

The development of the Named Entity Recognition System in English and a few other languages is remarkable. The English language especially has a large amount of Named Entity annotated corpora. In contrast, in the Natural Language Processing (NLP) community, despite being the 7th most widely spoken language across the world, Bangla is still now considered as a low-resource language. In recent years, a lot of Bangla NLP subtasks have received quite a lot of attention. Due to the recent advancement of NLP, there are so many tools and methods to do word tokenization, word embedding, POS, NER in the English language. NLTK [5](Loper and Bird, 2002), coreNLP (Manning et al., 2014)[6], spaCy (Honnibal and Montani, [7]), AllenNLP [8], are few of the tools.

Tool like iNLTK (Arora, 2020) [9] is an initial approach for different Indic languages including Bengali. But as it groups with other indic languages special monolingual support like easy preprocessing, tokenization, embedding, POS, NER for Bengali is missing.

BNLP is an open-source language processing toolkit [10] for Bengali consisting of tokenization, word embedding, part of speech (POS) tagging, named entity recognition (NER) facilities. It Provides different embedding method to embed Bengali word using the pre-trained model and also provides an option to train an embedding model from scratch. BNLP also provides hands-on start option for POS or NER of Bengali sentences and also provides an option for training Conditional random field (CRF) based POS tagger or NER model from scratch.

1.3 Challenges of Bangla NER

Due to the complex language structure and resource scarcity, Bangla NER appeared to be a difficult task compared to other languages [11]. The common challenges for solving NER in Bangla are discussed below.

A. INCLUSION OF AFFIXES

As Bangla is a morphologically rich language, we need to consider the morphological structure into account to ensure more human-readable output. Affixes play a vital role in developing the contextual meaning of a sentence. When an affix is combined with the base form of a word, it produces a derived or inflected form of that word, which substantially influences the context. Unlike English, affixes are often included with entity words in Bangla without changing the entity class of the word. Since, a Bangla entity with its affix may result in about 150 different forms, it is difficult for NER models to recognize those.

B. FREE ORDER SENTENCE STRUCTURE

Bangla is a highly inflected language from the Indo-Aryan language family. As words within a Bangla sentence can be positioned free-order without losing its context, the same entity word may appear in different positions in a sentence.

C. ABSENCE OF CAPITALIZATION FEATURE

In many languages, especially in English, proper noun generally refers to an entity and always starts with a capital letter. As Bangla language does not require the word capitalization criteria for proper nouns, recognizing those named entities has additional difficulty [12], [13]. In some widely used NER models, the capitalization feature plays a vital role in identifying named entities.

D. WORD-SENSE DISAMBIGUATION

In Bangla language, several words have different meanings despite having similar spellings [14]. It creates additional difficulty for the model to understand which sense of the word is used in a context. For example, the word ‘Kartik’ refers to both a person’s name and a month’s name in the Bangla calendar. Since, Bangla is highly inflected and free-order language, and it does not require the use of preposition as much

like in English, the word-sense disambiguation is quite difficult in Bangla; hence making it more challenging to recognize named entities.

Challenges	Example
Free order sentence structure	সেদিন রাতে বাড়ি ফিরে <u>নজরুল</u> That night <u>Nazrul</u> returned home
	<u>নজরুল</u> সেদিন রাতে বাড়ি ফিরে <u>Nazrul</u> returned home that night
	সেদিন রাতে <u>নজরুল</u> বাড়ি ফিরে <u>Nazrul</u> returned home that night
Capitalization	<u>মনির</u> <u>ঢাকায়</u> থাকে <u>Monir</u> lives in <u>Dhaka</u>
Word-sense disambiguation	<u>পুজা</u> টিভি দেখছে <u>Puja</u> is watching TV
	তিনি <u>পুজায়</u> গ্রামে আসবেন He will come to the village on <u>Puja</u>
Phrases and idioms	বড় লোকের <u>খয়ের খাঁর</u> অভাব নেই Rich people have no shortage of <u>flatterers</u>

Table 1.1: List of challenges of Bangla NER

E. PHRASES AND IDIOMS

Phrases and idioms in Bangla often contain words that do not refer to their actual meaning. For example, in Figure 1.1, the word ‘Khoyer Kha’ generally means a person’s name in Bangla. Nevertheless, when it is used as an idiom, it refers to a flatterer. ‘Alaler Ghorer Dulal’ is another example of a different meaning. Although ‘Alal’ and ‘Dulal’ are person names, it refers to a spoilt kid of a rich dad. In general, NER systems face difficulty in identifying the named entities within phrase and idioms, but the word-sense disambiguation issues as discussed above makes it much more difficult for Bangla.

Chapter 2

Literature Survey

Over many years lots of research work has been done with NER in different languages. In recent years, a lot of Bangla NER tasks have received quite a lot of attention. NER systems can use a Rule-based approach, Machine Learning (ML) or Deep Learning (DL) approach.

2.1 Rule-based NER

Rule-based approaches consist of the set of rules which are hand-crafted by experts. The rules are based on syntactic-lexical patterns, linguistic and domain related knowledge [15]. Rule-based Named Entity Recognition and Classification systems are worked on domain specific features for obtaining the sufficient accuracy and highly efficiency. These systems have some limitations that they are some costly, non-portable and also domain-specific. Moreover, these systems need human expertise for the knowledge of the domain with programming skills. These systems do not transfer across domains. Rule-based NER systems developed only for one domain they do not portable into another domains.

In the paper [16], the authors have experimented with the combination of two approaches: Dictionary-based NE detection and Rule-based NE detection. They created the dataset from Bangla daily newspaper. Then they took a portion for the years 2001-2003 for training their system and took a portion from 2004 for testing. Their model was able to achieve 85.50% recall, 94.24% precision, and 89.51% F1-score.

2.2 Machine Learning-based NER

The ML-based systems use statistical-based models for detecting the entity names. These models try to make a feature-based representation of the observed data. By this approach, a lot of limitations of dictionary and rule-based approaches are solved. The representative machine-learning approaches used in NER are Hidden Markov Model (BBN's IdentiFinder in [17]), Maximum Entropy (New York University's MEME in [18]), Decision Tree (New York University's system in [19], 1998) and Conditional Random Fields (CRFs). Support Vector Machines (SVMs) based NER system was proposed by Yamada et al. (2002) for Japanese [20]. His system is an extension of Kudo's chunking system (Kudo and Matsumoto, 2001) that gave the best performance at CoNLL-2000 shared tasks [21].

This paper [22] reports about the development of a NER system for Bengali using Support Vector Machine (SVM). The training set consists of approximately 150K words and has been manually annotated with the sixteen NE tags. Experimental results of the 10-fold cross validation test show the effectiveness of the proposed SVM based NER system with the overall average Recall, Precision and F1-Score of 94.3%, 89.4% and 91.8%, respectively.

In another study [23], the author proposed a machine learning-based model for the NER system. To categorize the named items from the Bengali language, they used a Maximum Entropy Markov Model (MEMM). According to them, the Maximum Entropy Markov Model can handle complex sequences better than previous approaches. Additionally, to improve their MEMM performance, they defined a rich set of linguistic features for their model's training. They have gathered a few articles in Bangla from different Bangla news websites. They searched articles where they identified numerous named entities since they are working with the designated entity here. Their corpus had 10,000 words that were taken from 690 sentences in total. They received an F1 score of 68.98% from the MEMM model alone and an F1 score of 71.59% from the combined model.

2.3 Deep Learning-based NER

Deep learning is used in computing and modeling multiple levels of abstraction data through multiple processing layers, and then output people expected results. Deep learning is widely used in computer vision, artificial intelligence,

genomics, and biomedical science, and other domains and brought a breakthrough in image, video, audio and speech processing (LeCun, Bengio & Hinton, 2015) [24].

In this paper [25], they have focused on information extraction by using two tasks: NER and Relation Extraction. In this work, the NER system has been implemented for Bangla language system. For the dataset, a unique dataset is created. More than 71,000 Bangla sentences have also been collected, annotated, and categorized into groups in their dataset using the IOB tagging scheme. They tested two different models for the classification task. At first, they used Convolutional Neural Network (CNN) based models. But it was outperformed by their second approach which was made with a Densely Connected Network (DCN). Their model worked resulting in having an F1 score of 63.37%.

In their research [26], the authors focused on POS tagging and Name Entity Recognition in the Bangla language, emphasizing that NER is a crucial aspect in NLP for information extraction. To accomplish this, they created two distinct datasets for POS tagging and NER by compiling news articles from different varieties on different Bangla online news portals, utilizing standard tags and tagging methods. To solve these two tasks separately, a different deep neural network has been used. Their approaches were end to end, there are no handcrafted features like word suffixes or affixes, gazetteers, or dictionaries. For this work, deep neural network models have been implemented - Bi-directional Long short-term memory (BiLSTM), Convolutional Neural Network (CNN), Conditional Random Field (CRF), and LSTM. They achieved a 0.6285 F1 score with their model in the NER task.

In another study [27], the author presented a novel method or Named Entity Recognition (NER) using a Bidirectional Long Short-Term Memory (LSTM) Conditional Random Field (CRF) model. By employing both forward and backward contextual information through bidirectional LSTM layers and including a CRF layer to represent the dependencies among the predicted entity labels, the authors address the difficulties of NER. The dataset they used contains four different types of named entities: Person (PER), Organization (ORG), Location (LOC) and Miscellaneous (MISC). The proposed model's efficiency is demonstrated in the study through experiments on benchmark datasets, highlighting both its capacity for cutting-edge performance in NER tasks and its potential for numerous applications in natural language

processing. Their model was able to achieve a 90.84% F1 score.

According to the author of the [28] paper, they experimented with eleven different architectures consisting of BERT (Bidirectional Encoder Representations from Transformers) layer, BiLSTM layers, linear layers, a dropout layer in between and a CRF layer. They used the dataset from Karim et al. [29] where 72000 Bangla sentences were annotated. Their highest score was achieved with BERT + BiLSTM + CRF + CW with a macro averaged F1 score of 65.96% and micro-averaged F1 score of 90.66%.

This [30] paper introduced us with BanglaBERT and BanglishBERT. BanglaBERT has a base and a large model both of which are based on BERT. In their paper, they discussed how they collected data for pre-training the models. They pre-trained BanglaBERT using ELECTRA. For zero-shot cross-lingual transfer, their BanglishBERT achieved 55.56 Micro F1 score and supervised fine-tuning, their BanglaBERT model gained 77.78% Micro F1 score in MultiCONER dataset.

Chapter 3

Methodology

3.1 Word Embedding

In natural language processing (NLP), a **word embedding** is a representation of a word. The embedding is used in text analysis. Typically, the representation is a real-valued vector that encodes the meaning of the word in such a way that the words that are closer in the vector space are expected to be similar in meaning [31]. Word embeddings can be obtained using language modeling and feature learning techniques, where words or phrases from the vocabulary are mapped to vectors of real numbers. Methods to generate this mapping include neural networks, dimensionality reduction on the word co-occurrence matrix, probabilistic models, explainable knowledge base method, and explicit representation in terms of the context in which words appear. Word and phrase embeddings, when used as the underlying input representation, have been shown to boost the performance in NLP tasks such as syntactic parsing and sentiment analysis [32].

3.2.1 Word2Vec

Word2vec is a shallow two-layered neural network, that is used to produce the relative model of word embedding. It works by collecting a large number of vocabulary based datasets as input and outputs a vector space, such that each word in the dictionary maps itself to a unique vector. This allows us to represent the relationship between words. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence.

Word2vec was developed by Tomas Mikolov and colleagues at Google and published in 2013 [33].

Broadly, there are two models Continuous Bag of Words (CBOW) and Skip-Gram. Both neural network architectures essentially help the network learn how to represent a word. This is unsupervised machine learning, and labels are needed to train the model. Either of these two models, can create the labels for the given input and prepare the neural network to train the model and perform the desired task.

1. Continuous Bag of Words (CBOW)

Continuous Bag of Words (CBOW) is a type of neural network architecture used in the Word2Vec model. The primary objective of CBOW is to predict a target word based on its context, which consists of the surrounding words in a given window. Given a sequence of words in a context window, the model is trained to predict the target word at the center of the window.

In the CBOW architecture, the input to the model is a fixed-size window of context words, typically represented as one-hot encoded vectors. These context word vectors are averaged or summed to create a single vector representation, which serves as the input to the neural network. The output layer of the neural network consists of a softmax function that predicts the probability distribution over the entire vocabulary, indicating the likelihood of each word being the target word. During training, the weights of the neural network are adjusted to minimize the cross-entropy loss between the predicted probability distribution and the actual target word. The objective is to maximize the likelihood of correctly predicting the target word given its context words. CBOW is computationally efficient compared to Skip-gram because it aggregates context information from multiple words into a single vector representation, reducing the complexity of the model.

2. Skip-gram

Skip-gram predicts the surrounding context words within specific window given current word. The input layer contains the current word and the output layer contains the context words. The hidden layer contains the number of dimensions in which we want to represent current word present at the input layer. the model is trained to maximize the likelihood of predicting the context words within a fixed window size around the target word. By iteratively adjusting the weights of the neural network based on the objective of

maximizing the probability of correctly predicting the context words. Skip-gram effectively captures semantic relationships between words based on their contextual usage in large text corpora.

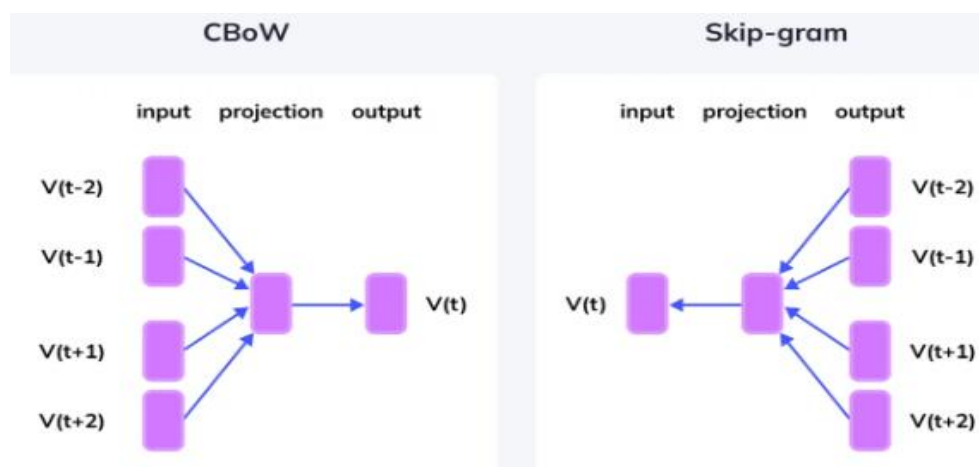


Figure 3.1: Different Word2Vec Models

3.2 Convolutional Neural Networks

Artificial neural networks (ANNs), which are modeled after the way neurons in the human brain process information, work together to learn from the input given in order to provide an optimum outcome. Artificial neural networks are made up of several interconnected processing nodes, or neurons, that are linked together via the neurons' tails. The input layer, the top layer, is loaded with the input. Typically, the input takes the form of a multidimensional vector. Most learning takes place in the hidden layers. Learning is the act of making decisions based on the output from previous layers and determining whether a change degrades or enhances the result.

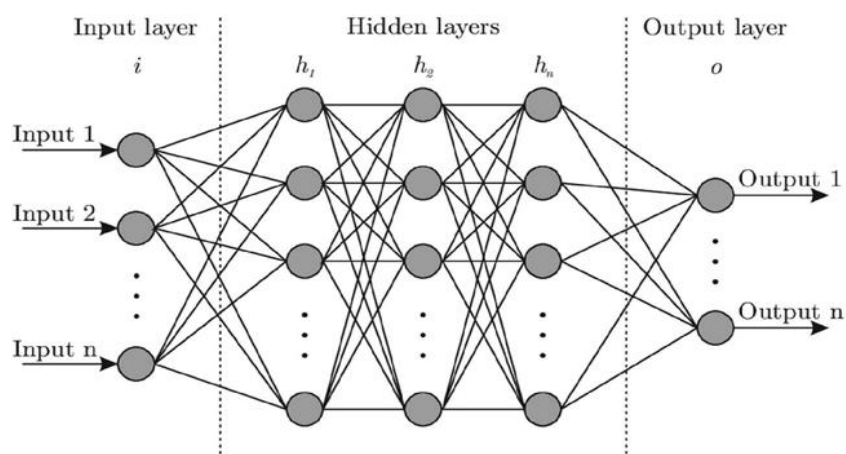


Figure 3.2: Basic architecture of an Artificial Neural Network (ANN)

Convolutional neural networks (CNN) are composed of many such individual computational nodes or neurons that may improve themselves by learning repeatedly, similar to Artificial Neural Networks. Every neuron in a convolutional neural network has the ability to enhance and improve itself through learning, which is another similarity to artificial neural networks. The whole network reflects one weight or one perceptual score function, starting with the multidimensional vectors as input and continuing through to the end findings of the class probabilities.

Similar to artificial neural networks, it has fully connected layers up to the last layer, which has loss functions related to the classes. The Convolutional Neural Network's architecture may be configured in a way that best fits the specific type of data that has to be handled. Contrary to artificial neural networks, the layers of neurons in a convolutional neural network are arranged in three dimensions: height, breadth, and depth. The depth is a third dimension known as the activation volume. In convolutional neural networks, each layer's neurons only interact with a little portion of the layer below it.

The convolutional layer, Pooling layer, and Fully-Connected layer are the three different types of layers that make up a convolutional neural network. The first layer of a convolutional neural network is called the convolutional layer. A convolutional layer may be followed by pooling layers, more convolutional layers, or a mix of the two. The fully linked layer is the last layer. Each layer of the CNN adds to its complexity. Every layer gives CNN the capacity to recognize larger portions of the image. Early layers emphasize basic traits. The convolutional neural network (CNN) begins to recognize bigger, more complicated parts or forms of the item as it advances through the different layers until it can eventually identify the desired object.

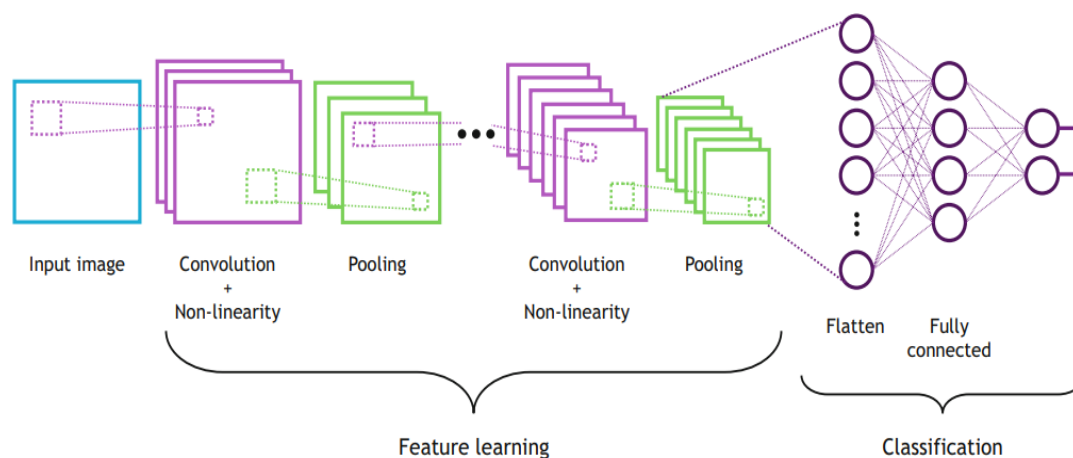


Figure 3.3: A simple CNN Architecture

3.2.1 Convolution Layer

A Convolutional Layer in a Convolutional Neural Network (CNN) is a core building block designed to process and analyze visual data, such as images, by preserving the spatial relationships between pixels. This layer performs a mathematical operation called convolution, which combines two functions to produce a third one that represents how the shape of one is modified by the other. In the context of CNNs, the convolution operation is used to detect various features in an image, such as edges, textures, and patterns. The convolution operation involves sliding a small matrix of weights, known as a filter or kernel, across the input image. Each position of the filter on the image results in a single value, which is computed as the sum of the element-wise product of the filter and the overlapping input values. This process is repeated for all positions of the filter on the image, generating a two-dimensional activation map, or feature map. The values in this feature map represent the presence and intensity of the feature detected by the filter at each spatial location of the image.

Filters are typically small in spatial dimensions but extend through the full depth of the input volume if the input is multi-channel (e.g., an RGB image with three color channels). The filter's weights are learned through the training process, enabling the CNN to automatically discover relevant features for the given task. Multiple filters are used in a convolutional layer, each detecting different features, resulting in multiple feature maps for a single input. To manage the spatial dimensions of the output, CNNs use several techniques. Padding, for instance, involves adding extra pixels around the border of the input image to control the spatial size of the output feature map. Stride refers to the number of pixels by which the filter moves across the image. A stride of 1 means the filter moves one pixel at a time, while a higher stride value reduces the spatial dimensions of the output feature map.

Activation functions, such as ReLU (Rectified Linear Unit), are applied to the output of the convolution operation to introduce non-linearity, enabling the network to learn more complex patterns. ReLU, which stands for Rectified Linear Unit, is an activation function commonly used in neural networks, including Convolutional Neural Networks (CNNs). It introduces non-linearity to the network by applying a simple thresholding operation to the output of each neuron.

Mathematically, the ReLU function is defined as:

$$f(x) = \max(0, x) \quad (3.1)$$

If the input value x is greater than zero, the output of the ReLU function is equal to the input value itself. If the input is negative, the output is zero. Visually, this operation replaces any negative values in the input with zero, while leaving positive values unchanged.

3.2.2 Pooling Layer

In a Convolution Network design, it is typical to sporadically introduce a Pooling layer between succeeding Convolution layers. Its purpose is to gradually shrink the dimension of the representation in order to constrict over-fitting and decrease the number of parameters and computations in the network. Another name for pooling layers is down-sampling. A filter is moved through the whole input by this layer, much way the convolutional layer does. However, this filter does not have any associated weights; instead, the receptive field is subjected to an aggregation function. The output array is filled as a result. Pooling often involves two types.

1. **Max Pooling:** As it moves over the input, the filter chooses the pixel with the greatest value to deliver to the output. This is mostly employed.
2. **Average Pooling:** As it moves over the input, the filter computes the average number of the receptive field.

Since it reduces the input's spatial dimension, the pooling layer is often applied separately to every depth slice of the input volume following a convolution layer.

3.2.3 Fully Connected Layer

In Convolutional Neural Networks (CNNs), a Dense layer, also known as a fully connected layer, plays a crucial role in the final stages of the network. After the convolutional and pooling layers have extracted and reduced spatial features from the input data, these features are flattened into a one-dimensional vector, which is then fed into one or more Dense layers. In a Dense layer, each neuron is connected to every neuron in the previous layer, enabling it to combine the extracted features and make predictions. The layer performs a

weighted sum of its inputs, adds a bias term, and passes the result through an activation function such as ReLU or softmax. This allows the network to capture complex relationships and make high-level decisions. Dense layers are essential for tasks like classification, where they integrate the learned features into a final decision, producing a probability distribution over possible classes. The weights and biases in the Dense layer are learned during training through back-propagation, optimizing the network's performance on the given task.

3.3 Bidirectional LSTM

A Recurrent Neural Network (RNN) type, Long Short-Term Memory (LSTM) [34], has the capability to efficiently capture long-term dependencies in sequential data. A standard RNN processes sequential data by passing the information from one time step to the next through a hidden state, however, this hidden state can only remember information for a short period of time. This makes it difficult for RNNs to accurately model data with long-term dependencies, such as in natural language processing tasks where the meaning of a word depends on the context of the entire sentence. LSTMs address this problem by introducing an additional memory cell, which can store information over a longer period of time. A memory cell in LSTM is managed by three gates: input, forget, and output gates. The input gate decides the portion of newly arrived data being saved in the memory cell, the forget gate determines the amount of previous information to be kept and the output gate controls the amount of information to be passed on to the following time step. Additionally, LSTMs introduce a new structure called the hidden state which is used to store information that is passed from one-time step to the next. At each time step, the hidden state is revised according to the current input, previous hidden state and current memory cell state.

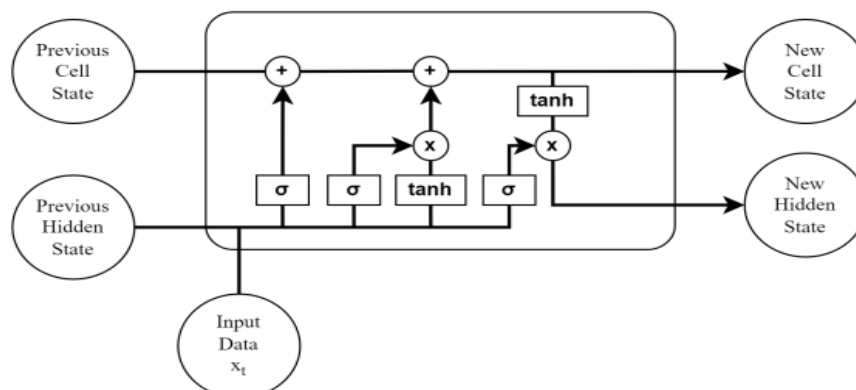


Figure 3.4: A simple cell of LSTM

The ability of LSTMs to efficiently capture long-term dependencies in sequential data makes them useful for a variety of tasks, including speech recognition, natural language processing, and other sequential data processing tasks.

In a bidirectional LSTM (BiLSTM), there are two separate LSTM networks: one that processes the data in the forward direction and one that processes the data in the backward direction. The output from both LSTMs is then concatenated and passed through a fully connected layer to produce the final output. Because a BiLSTM processes the data in both directions, it is able to capture context from both the past and future time steps. This is particularly useful in the Named Entity Recognition (NER) where the model needs to understand the context of the name to identify whether it is a person, location, or organization. A BiLSTM would be able to take into account the context from both past and future time steps to identify the entity correctly.

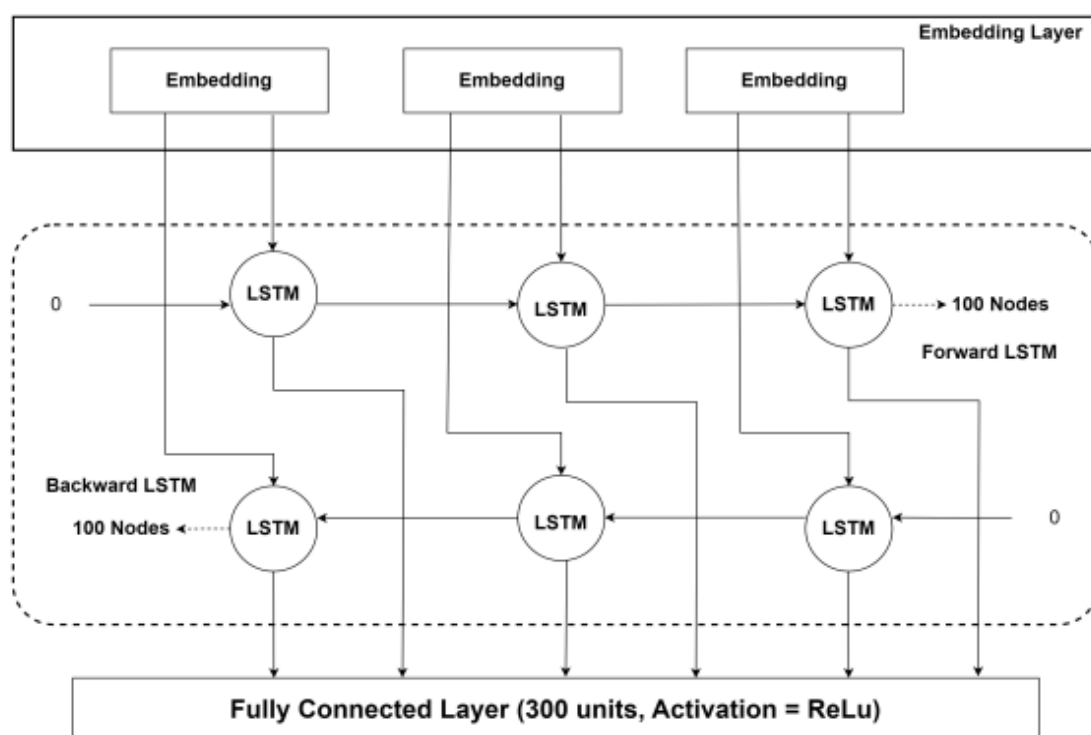


Figure 3.5: Structure of Bidirectional LSTM

In Named Entity Recognition (NER) tasks, Bidirectional Long Short-Term Memory (BiLSTM) architectures play a pivotal role in accurately identifying and classifying entities within a sequence of text. In this context, BiLSTMs process input sequences, typically sentences or documents, bidirectionally,

leveraging both past and future context to make predictions about each word's entity label. This bidirectional approach enables the model to consider not only preceding words but also succeeding ones when predicting entity labels, enhancing its ability to handle complex linguistic structures and ambiguous contexts. The outputs from both directions are combined, providing a comprehensive representation of each word's context for entity classification. BiLSTM-based NER models have demonstrated superior performance in various domains, including biomedical text mining, information extraction, and natural language understanding tasks, showcasing their efficacy in accurately identifying named entities in unstructured text data.

3.4 BERT

The full form of BERT is Bidirectional Encoder Representations from Transformers. It is a pre-trained deep learning model which has revolutionized Natural Language Processing tasks like sentiment analysis, question answering, named entity recognition, etc. BERT was introduced by Google AI in 2018 and since then it has achieved many state-of-the-art results in various NLP tasks.

BERT is based on transformer architecture. Transformer architecture is highly effective in capturing contextual relationships in language which is a major advantage in NER tasks. BERT is a deep bidirectional transformer model that consists of multiple layers of self-attention mechanisms and a feed-forward neural network. To consider the dependencies between all the words in a sentence, the self-attention mechanism plays an important role. BERT is trained using a large amount of unlabeled text data in an unsupervised manner. This unsupervised manner is called masked language modeling (MLM).

In MLM, BERT randomly masks out a certain percentage of words in a sentence. Then, it tries to predict the original masked words from the context of the surrounding word. Because of this unique way, BERT learns a deep understanding of language semantics and syntax. Furthermore, BERT also performs an NSP (next sentence prediction) task during the pre-training phase. In this phase, it takes a pair of sentences as input. This phase allows BERT to understand the relationships between sentences and captures the ability to understand discourse and coherence.

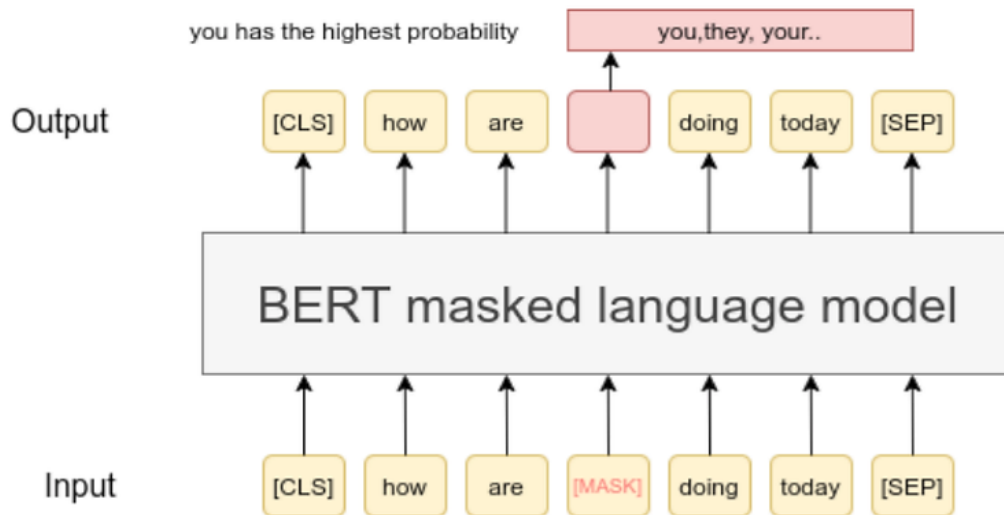


Figure 3.6: Masked Language Model (MLM)

To help the model distinguish between the two sentences in training, the input is processed in the following way before entering the model:

1. A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
2. A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2.
3. A positional embedding is added to each token to indicate its position in the sequence. Positional embedding describes the location or position of an entity in a sequence so that each position is assigned a unique representation.

To predict if the second sentence is certainly connected to the first, the entire input sequence goes through the Transformer model. The output of the [CLS] token is transformed into a 2×1 shaped vector using a simple classification layer, which consists of learned matrices of weights and biases. The probability of the second sentence being the next sequence is calculated using softmax activation function. When training the BERT model, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) are trained together, with the goal of minimizing the combined loss function of these two strategies.

After the pre-training phase, BERT now can be trained for a specific task which usually is done in labeled data. This phase is called fine-tuning. In this phase, the model is further trained with task-specific input and output layers.

We can also use the last layer's output as embeddings for other neural network models. This transfer learning approach allows BERT to perform at or near state-of-the-art levels performance in less task-specific labeled data.

Based on the number of encoders, BERT has a base model and a large model. The BERT base model has 12 encoders each producing (1x768) sized vector for each token and BERT large has 24 encoders that produce (1x1024) sized vector for each token. The structure of the BERT base model can be found in figure 3.7.

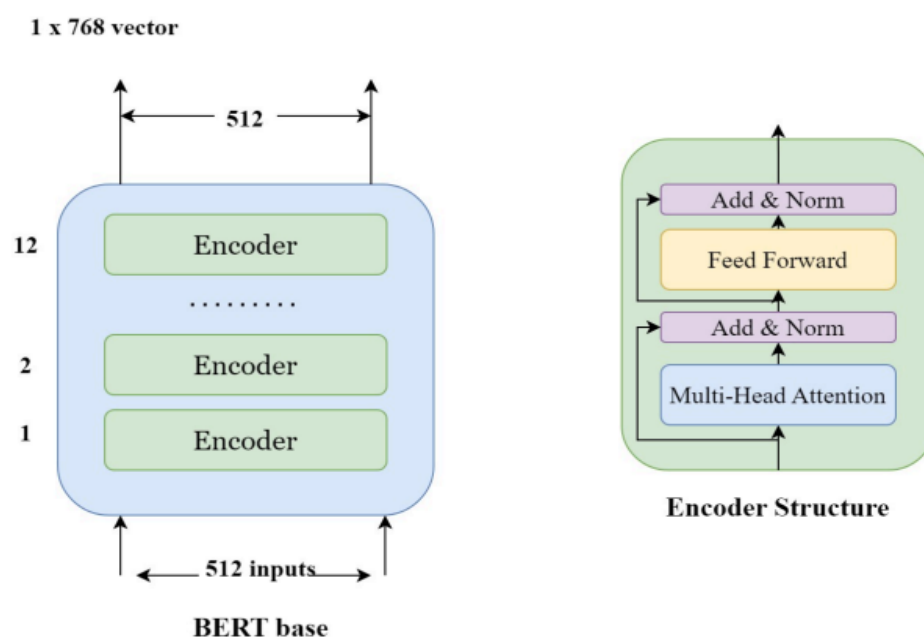


Figure 3.7: BERT Base Structure

3.5 Evaluation Metrics

3.2.1 Confusion Matrix

A table called a confusion matrix is used to describe how well a classification system performs. The output of a classification algorithm is visualized and summarized in a confusion matrix. Confusion matrices are incredibly helpful for determining Recall, Precision, F1-Score, and other crucial performance evaluation measures. Four fundamental properties (numbers) make up the confusion matrix, which is used to provide the classifier's measurement parameters.

These are the four numbers:

1. TP (True Positive): — The number of tags that were accurately identified as belonging to a certain class of outputs, or that are True Positives.
2. TN (True Negative): — The number of tags that are True Negatives, or that were accurately identified as not being under the same category as a certain class.
3. FP (False Positive): — The number of tags that are False Positives, or that were wrongly identified as falling under one category but didn't really do so. Another name for FP is a Type I mistake.
4. FN (False Negative): — The number of tags that are False Negatives, or those that were mistakenly classified as Negatives while they were actually Positives. Another name for FN is a Type II mistake.

3.2.2 Recall

This number lets us know how many inputs, out of all those that belonged to a certain class, were correctly identified as belonging to that class. A well-designed model must have a high Recall value. We can determine the model's recall using the confusion matrix and the values of TP and FN by applying the formula below.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (3.2)$$

3.4.3 Precision

The Precision number provides an estimate of the proportion of inputs that genuinely belonged to each of the classes to which they were assigned. The following formula is used to determine this using the values of TP and FP.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (3.3)$$

3.4.4 F1-Score

Comparing the performance of two models that have low recall and excellent precision, or vice versa, is difficult. In this situation, F1-Score is useful. It

calculates the F- Score by finding the Harmonic Mean of the two values after using both the Recall value and the Precision value. This simultaneously assesses recall and precision. This is demonstrated by the F1-Score calculation equation.

$$F1\text{-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.4)$$

3.4.5 Accuracy

The ratio of properly categorized class labels (TP+TN) to all class labels (TP+TN+FP+FN) is used to measure an algorithm's accuracy.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.5)$$

Chapter 4

System Description

4.1 Dataset Description

B-NER, a Novel Bangla Named Entity Recognition dataset, is used for training and testing the NER models. In this paper [35], the authors have presented the B-NER dataset, which is a fine-grained largest Bangla NER dataset using the BIO tagging scheme. They have collected and filtered raw corpus data from various sources, including blogs, newspaper articles, and Wikipedia. Python built-in library beautifulsoup is used for crawling newspapers, blogs and forums websites. The dataset consists of 2,97,417 rows and 3 columns (Sentence#, Word, Tag). There are a total 22,144 unique sentences, 34,236 unique words and 17 unique tags in the dataset. The Named Entities are grouped into 8 different classes. Miscellaneous words that do not belong to any of the previously stated eight categories are tagged with the letter “O”.

For tagging named entities, researchers proposed and used various tagging schemes, such as BILUO, BIO, BIO2, IO, BIL2, BIOES, IOE, IOE2, etc [36]. Each of the annotation strategies has its own benefits and flaws. Most schemes deliver distinct benefits to high-resource language. For post-positional languages like Bengali, Hindi, and Japanese, the BIO tagging scheme is widely adopted by researchers [37]. B-NER dataset was designed following Groningen Meaning Bank (GMB) [38] and used the BIO tagging scheme, which was first introduced by Ramshaw and Marcus [39].

According to the BIO tagging scheme, if a chunk is found belonging to any entities, then the beginning token is tagged as the “B-entity name” format, and the remaining tokens are tagged as the “I-entity name” format.

Tags	Entity Names	Examples
geo	Geographical Entity	এশিয়া (Asia), ঢাকা (Dhaka)
org	Organization	গুগল (Google)
per	Person	লিওনেল মেসি (Lionel Messi)
gpe	Geopolitical Entity	ভারতীয় (Indian)
tim	Time indicator	রবিবার (Sunday)
art	Artifact	টম এন্ড জেরি (Tom and Jerry)
eve	Event	দুর্গা পূজা (Durga Puja)
nat	Natural Phenomenon	পূর্ণিমা (Full Moon)

Table 4.1: Eight Name Entities of B-NER dataset

4.2 Data Preprocessing

The data preprocessing for the Named Entity Recognition (NER) task in Bengali involves a series of meticulous steps to transform the raw dataset into a structured format suitable for training a machine learning model. The process begins with loading the dataset, where each row contains a word, its corresponding tag, and a sentence identifier.

The data is grouped by sentence number to aggregate words and their corresponding tags into cohesive units. This grouping results in each sentence being represented as a list of word-tag pairs. To facilitate further processing, a preprocessing function is applied to the grouped data. This function extracts words and tags from each sentence, creating separate strings for the words and their associated tags.

4.2.1 Tokenization

Tokenization is the process of dividing a text into smaller units known as tokens. Tokens are typically words or sub-words in the context of natural language processing.

Tokenization is a critical step in many NLP tasks, including text processing. The process involves splitting a string, or text into a list of tokens. Banglanltk, a python package for Bengali Natural Language Processing Toolkit, is used to tokenize the sentences.

4.2.2 Word2vec Model

BNLP, natural language processing toolkit for Bengali, provides different pre-trained Bengali models. One of them is word2vec model. In the [10] paper, the author has trained Bengali word2vec model on Bengali Wikipedia and news articles datasets using gensim word2vec pipeline. The model is trained with embedding dimension 100, window size 5, the minimum number of word occurrences 1, and total workers number 8. The author has train it for a total of 50,000 iterations. This pre-trained model has the vocab size as 1,60,405. This model provides a mapping of words to their respective vector representations. An embedding matrix is constructed where each row corresponds to a word vector from the Word2Vec model, with an additional row for padding. The dimensions of the embedding matrix are defined by the number of unique words in the vocabulary and the size of the word vectors, which is 100 in this case.

4.2.3 Encoding and Padding

Sentences are encoded by converting each word into its corresponding index from the Word2Vec model. To ensure uniformity in input size, these encoded sentences are padded to a maximum length, determined by the 95th percentile of the sentence lengths in the dataset. This padding ensures that shorter sentences are extended to the maximum length with zeros, while longer sentences are truncated. The maximum sentence length is considered as 28 for padding. Similarly, tags are encoded into numerical values based on a predefined mapping and padded to match the sentence lengths. The tags are then one-hot encoded to transform them into a format suitable for categorical classification.

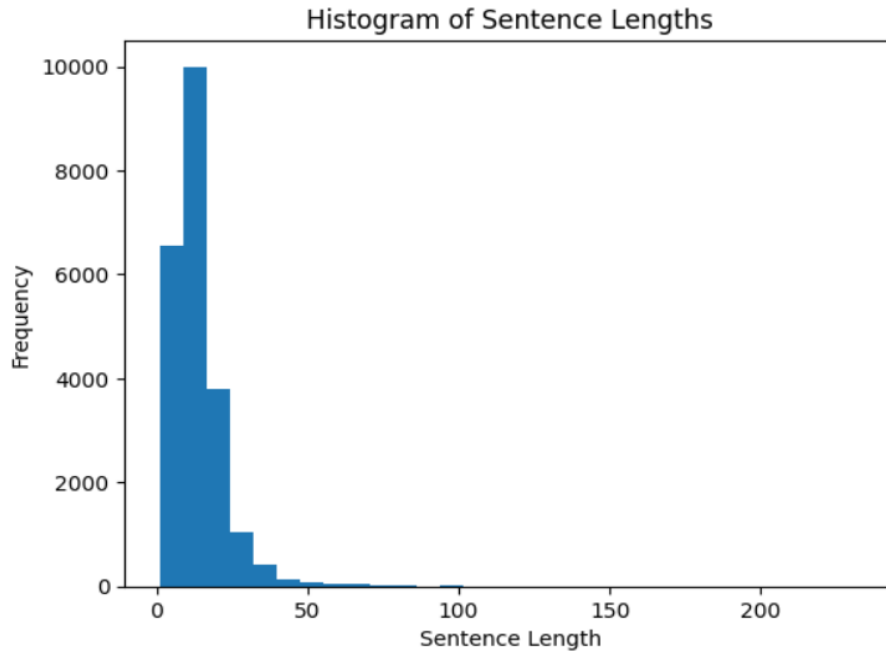


Figure 4.1: Histogram of Sentence lengths

4.2.4 Splitting of Dataset

The data was split into training validation and testing sets.

1. **Training set** contains 16,608 unique sample sentences. This subset was used to train the models, allowing them to learn the patterns and structures in the data.
2. **Validation set** contains 10% of samples from the training set. It enables the fine-tuning of model parameters, prevents over-fitting, and facilitates the selection of the best-performing model. During the training process, the model is trained iteratively on the training set, and its performance is evaluated on the validation set after each epoch or a certain number of training steps.
3. **Testing set** contains 25% of the data (5,536 sample sentences). This subset was used to evaluate the models' performance on unseen data, providing an unbiased assessment of their generalization ability.

4.3 Model Architecture

4.3.3 The CNN Classifier

The Embedding Layer is the first layer in the model and serves a crucial role in converting the input sequence of word indices into dense vectors of fixed size. This transformation is essential for handling natural language data, as it maps each word in the vocabulary to a high-dimensional space where semantically similar words have similar representations. The embedding layer is initialized with pre-trained Bengali Word2Vec embeddings, allowing the model to leverage the pre-learned patterns. The embedding layer has an input dimension equal to the size of the vocabulary plus one (to account for padding), an output dimension of 100.

Following the embedding layer, the Convolutional Layer (Conv1D) applies convolution operations to the input sequence. This layer is designed to automatically and adaptively learn spatial hierarchies in the input data through its filters. Each filter slides over the input sequence and performs a dot product operation, effectively capturing local patterns such as sequences of words that frequently occur together. In the CNN model, the convolutional layer uses 256 filters, each with a kernel size of 1. This configuration means that the layer will generate 256 different feature maps, each highlighting different aspects of the input sequence based on the patterns detected by the filters. The activation function used is ReLU (Rectified Linear Unit), which introduces non-linearity into the model, enabling it to learn more complex patterns and relationships in the data.

The Pooling Layer (MaxPool1D) follows the convolutional layer and is used to reduce the dimensionality of the feature maps while retaining the most important information. Max pooling works by selecting the maximum value within each pool size window, which helps to highlight the most significant features detected by the convolutional filters.

The Fully Connected Dense Layer is the final layer of the model and is responsible for producing the output classification. This layer takes the high-level features extracted by the convolutional and pooling layers and combines them to make a prediction for each word in the input sequence. In this model, the dense layer consists of units equal to the number of unique tags (entities), with a softmax activation function.

The softmax function converts the raw output scores into probabilities, ensuring that the sum of the probabilities for each word across all classes equals one. This enables the model to assign a probability distribution over all possible tags for each word, allowing for a clear and interpretable output that indicates the most likely tag for each word in the sequence. The Adam optimizer and categorical cross-entropy loss function were used for compilation.

4.3.4 The Bidirectional LSTM Classifier

The Bidirectional LSTM model also utilized the pre-trained Bengali Word2Vec embeddings. The architecture included a bidirectional LSTM layer with 200 units followed by a time-distributed dense layer with a softmax activation to output predictions for each token. The Bidirectional LSTM layer is a key component of this model, designed to capture dependencies and patterns in sequential data. Unlike a standard LSTM, which processes sequences in a single direction, a bidirectional LSTM processes the input data in both forward and backward directions. This allows the model to capture context from both past and future states relative to a given time step, providing a more comprehensive understanding of the sequence.

The Time Distributed Dense Layer applies a fully connected (dense) layer to each time step of the sequence independently. This allows the model to produce a separate output for each word in the input sequence. The activation function used is softmax, which converts the outputs into probability distributions over the possible tags for each word. This layer is crucial for sequence labeling tasks like Named Entity Recognition, as it ensures that each word in the sequence receives a tag.

The Model Compilation step finalizes the model configuration, specifying the optimizer, loss function, and metrics to be used during training. The Adam optimizer is chosen for its adaptive learning rate and efficiency in handling sparse gradients, making it well-suited for a variety of deep learning tasks. The loss function used is categorical cross-entropy, which measures the dissimilarity between the true labels and the predicted probability distributions. This loss function is appropriate for multi-class classification problems like Named Entity Recognition, as it provides a robust way to guide the optimization process.

4.3.3 The BERT Classifier

The BERT model, which we have used, employed the “Sagorsarker/bangla-bert-base” pre-trained model [40] and then fine-tuned for token classification. The Bangla-BERT model was trained on two primary datasets: the Bengali Common Crawl corpus, downloaded from OSCAR, and the Bengali Wikipedia Dump Dataset. The data was preprocessed into the BERT format, with each sentence on a new line and new documents separated by an extra newline. The final vocabulary was built using the BNLP package [10], resulting in a vocabulary size of 1,02,025 tokens. The model architecture follows the BERT-Base, uncased variant, which includes 12 transformer layers, 768 hidden units, and 12 attention heads, totaling 110 million parameters. The training was performed on a single Google Cloud GPU, with the model undergoing one million training steps to achieve its current state. “BertForTokenClassification” class is used for token-level predictions. It comprises a BERT backbone, consisting of stacked transformer blocks for contextual understanding, and a token classification head, which assigns probabilities to each token's class using a linear layer followed by softmax activation.

“BertTokenizer” employs Wordpiece tokenization, breaking down words into sub-word units for compatibility with BERT. It maintains a vocabulary of sub-word tokens and adds special tokens like [CLS] and [SEP] for sequence start/end and separation. It converts input text into token IDs and generates attention masks to distinguish between actual words and padding tokens.

We have fine-tuned the model, which involves setting up the Learning Rate, Optimizer, and some other hyper-parameters. AdamW, a variant of the Adam optimizer with weight decay, is used for fine-tuning BERT. The model parameters included learning rate set to $2e-5$, and training was performed for 5 epochs due to computational constraints. The batch size for training and evaluation was set to 16. The model was trained and evaluated using the Simple Transformers library, with automatic saving and loading of the model based on performance.

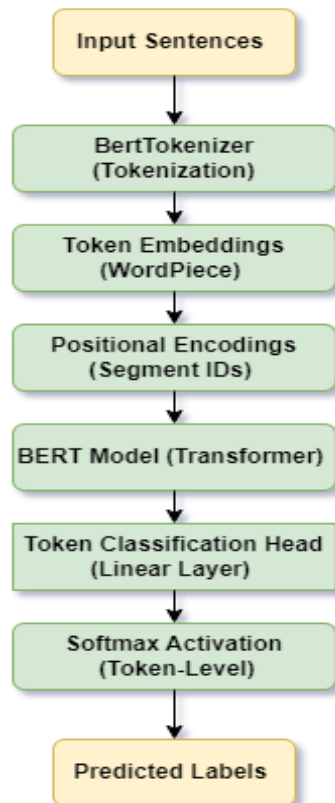


Figure 4.2: Flowchart of BERT architecture

Chapter 5

Experiment and Results

This section provides a description of the results obtained for the Named Entity Recognition (NER) task using different models: Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM), and Bidirectional Encoder Representations from Transformers (BERT).

5.1 Classification Report and Confusion Matrix

A detailed classification report is generated to provide insights into the model's performance across different tags.

1. CNN model

	precision	recall	f1-score	support
B-art	0.88	1.00	0.94	67429
B-eve	0.50	0.23	0.32	104
B-geo	0.84	0.57	0.68	1851
B-gpe	0.96	0.85	0.90	455
B-nat	0.00	0.00	0.00	4
B-org	0.86	0.29	0.43	985
B-per	0.64	0.46	0.54	1788
B-tim	0.87	0.40	0.55	925
I-art	0.83	0.08	0.14	66
I-eve	0.45	0.07	0.12	70
I-geo	0.41	0.11	0.17	147
I-gpe	0.00	0.00	0.00	2
I-nat	0.00	0.00	0.00	2
I-org	0.61	0.18	0.27	754
I-per	0.78	0.52	0.62	1690
I-tim	0.65	0.66	0.66	491
O	0.95	0.84	0.89	47249
accuracy			0.90	124012
macro avg	0.60	0.37	0.42	124012
weighted avg	0.90	0.90	0.89	124012

Figure 5.1: Classification Report of CNN model

		Confusion Matrix															
True Labels	B-art	67393	0	1	0	0	2	4	0	0	0	0	0	0	0	0	29
	B-eve	31	21	0	0	0	1	2	0	0	3	0	0	0	0	0	46
	B-geo	569	0	1055	7	0	40	7	0	0	0	11	0	0	4	3	155
	B-gpe	22	2	5	385	0	4	0	0	0	0	1	0	0	0	0	36
	B-nat	1	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
	B-org	272	1	90	4	0	363	21	0	0	2	0	0	0	16	9	206
	B-per	419	2	6	1	0	1	918	1	0	0	0	0	0	2	151	0
	B-tim	66	0	0	0	0	0	1	441	0	0	0	0	0	0	134	283
	I-art	9	0	0	0	0	0	4	0	7	0	0	0	0	3	1	42
	I-eve	17	2	1	0	0	0	2	0	0	15	0	0	0	1	2	29
	I-geo	62	0	9	0	0	2	3	0	0	0	13	0	0	2	4	52
	I-gpe	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	I-nat	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
	I-org	160	0	11	1	1	23	5	0	0	0	0	0	0	214	6	332
	I-per	305	1	1	5	0	2	346	0	0	0	0	0	0	5	861	3
	I-tim	52	0	0	0	0	0	0	40	0	0	0	0	0	0	332	67
	O	7320	10	40	2	1	45	115	83	1	28	6	0	0	144	42	22
		B-art B-eve B-geo B-gpe B-nat B-org B-per B-tim I-art I-eve I-geo I-gpe I-nat I-org I-per I-tim O															
		Predicted Labels															

Figure 5.2: Confusion Matrix for CNN model on Test data

2. Bidirectional LSTM model

	precision	recall	f1-score	support
B-art	0.99	0.99	0.99	67429
B-eve	0.62	0.32	0.42	104
B-geo	0.74	0.76	0.75	1851
B-gpe	0.97	0.84	0.90	455
B-nat	1.00	0.25	0.40	4
B-org	0.61	0.57	0.59	985
B-per	0.80	0.69	0.74	1788
B-tim	0.91	0.84	0.88	925
I-art	0.53	0.26	0.35	66
I-eve	0.52	0.49	0.50	70
I-geo	0.56	0.46	0.50	147
I-gpe	0.00	0.00	0.00	2
I-nat	0.00	0.00	0.00	2
I-org	0.71	0.51	0.59	754
I-per	0.83	0.84	0.83	1690
I-tim	0.88	0.91	0.90	491
O	0.96	0.97	0.96	47249
accuracy			0.96	124012
macro avg	0.69	0.57	0.61	124012
weighted avg	0.96	0.96	0.96	124012

Figure 5.3: Classification Report of BiLSTM model

		Confusion Matrix																
True Labels	B-art	67086	0	1	0	0	6	9	0	2	1	1	0	0	1	34	1	287
	B-eve	2	33	5	0	0	4	4	3	0	5	0	0	0	0	0	0	48
	B-geo	18	1	1406	4	0	81	14	1	0	0	11	0	0	7	7	3	298
	B-gpe	1	1	9	384	0	5	2	0	0	0	0	0	0	0	0	0	53
	B-nat	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	3
	B-org	14	2	72	0	0	565	17	2	1	2	1	0	0	18	16	0	275
	B-per	35	1	25	1	0	28	1239	3	0	0	0	0	0	4	106	0	346
	B-tim	0	0	5	0	0	4	0	777	0	0	0	0	0	0	1	30	108
	I-art	1	0	0	1	0	0	1	0	17	0	0	0	0	1	0	0	45
	I-eve	0	0	1	0	0	0	0	0	0	34	0	0	0	2	1	1	31
	I-geo	2	0	13	0	0	0	2	0	0	0	67	0	0	5	4	0	54
	I-gpe	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
	I-nat	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
	I-org	14	0	14	0	0	32	0	0	0	0	3	0	0	383	10	0	298
	I-per	22	0	2	0	0	7	104	0	1	0	2	0	0	11	1421	3	117
	I-tim	0	0	6	0	0	0	0	15	0	0	0	0	0	1	0	446	23
	O	579	15	331	6	0	190	160	49	11	23	35	0	0	104	118	20	45608
	Predicted Labels																	
	B-art B-eve B-geo B-gpe B-nat B-org B-per B-tim I-art I-eve I-geo I-gpe I-nat I-org I-per I-tim O																	

Figure 5.4: Confusion Matrix for BiLSTM model on Test data

3. BERT model

	precision	recall	f1-score	support
B-ART	0.54	0.30	0.38	71
B-EVE	0.50	0.48	0.49	94
B-GEO	0.86	0.90	0.88	2041
B-GPE	0.93	0.97	0.95	474
B-NAT	0.56	0.71	0.63	7
B-ORG	0.67	0.63	0.65	1130
B-PER	0.60	0.81	0.69	1880
B-TIM	0.61	0.44	0.51	948
I-ART	0.50	0.02	0.04	52
I-EVE	1.00	0.07	0.13	100
I-GEO	0.86	0.18	0.30	138
I-GPE	0.00	0.00	0.00	4
I-NAT	0.00	0.00	0.00	4
I-ORG	0.56	0.28	0.38	856
I-PER	0.89	0.43	0.58	1827
I-TIM	0.86	0.21	0.34	490
O	0.96	0.99	0.97	49368
accuracy			0.92	59484
macro avg	0.64	0.44	0.47	59484
weighted avg	0.92	0.92	0.92	59484

Figure 5.5: Classification Report of BERT model

		Confusion Matrix																
True Label	B-ART	21	2	3	0	0	16	4	0	0	0	0	0	0	0	0	25	
	B-EVE	0	45	1	2	0	3	3	0	0	0	0	0	0	0	0	40	
	B-GEO	0	0	1847	11	0	41	26	1	0	0	1	0	0	5	3	1	105
	B-GPE	0	0	8	460	0	4	0	0	0	0	0	0	0	0	0	0	2
	B-NAT	0	0	0	0	5	0	1	0	0	0	0	0	0	0	0	0	1
	B-ORG	4	3	139	7	0	710	24	3	0	0	0	0	0	14	5	0	221
	B-PER	4	1	8	0	0	53	1524	0	0	0	1	0	0	3	51	0	235
	B-TIM	1	1	1	0	0	2	0	419	0	0	0	0	0	0	0	16	508
	I-ART	1	0	0	0	0	1	5	2	1	0	0	0	0	3	2	0	37
	I-EVE	0	15	2	0	0	2	3	1	0	7	0	0	0	1	0	0	69
	I-GEO	0	0	37	1	0	3	7	0	0	0	25	0	0	6	3	0	56
	I-GPE	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	1
	I-NAT	0	0	0	0	1	0	3	0	0	0	0	0	0	0	0	0	0
	I-ORG	1	4	15	2	1	67	15	2	0	0	0	0	0	243	6	0	500
	I-PER	2	0	5	2	0	35	777	2	0	0	1	0	0	10	781	0	212
	I-TIM	0	0	0	0	0	0	0	214	0	0	0	0	0	1	0	103	172
	O	5	19	70	8	2	123	133	45	1	0	1	0	0	146	22	0	48793
		B-ART	B-EVE	B-GEO	B-GPE	B-NAT	B-ORG	B-PER	B-TIM	I-ART	I-EVE	I-GEO	I-GPE	I-NAT	I-ORG	I-PER	I-TIM	O
		Predicted Label																

Figure 5.6: Confusion Matrix for BERT model on Test data

Model	Accuracy	Precision	Recall	F1 score
CNN	0.89	0.60	0.37	0.42
BiLSTM	0.96	0.69	0.57	0.61
BERT	0.92	0.64	0.44	0.47

Table 5.1: Comparing the performance of different models

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The first chapter of this thesis provided an introduction to Named Entity Recognition and challenges of Bangla NER. The second chapter included many references to prior research that was relevant to this work. The system architecture and implementations were then discussed in system description chapter.

In this study, we explored and compared the effectiveness of three advanced models—Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM) network, and BERT-based model—for Named Entity Recognition (NER) in Bangla (Bengali). The experimental results revealed that the BiLSTM model outperformed both the CNN and BERT models in terms of accuracy and F1-score, demonstrating its superior capability in capturing sequential dependencies and contextual information necessary for NER tasks.

6.2 Future Work

One primary area of focus is the enhancement of dataset quality and diversity. Expanding the dataset to include more varied and representative samples from different domains, such as legal, medical, and social media texts, can significantly improve model robustness and generalization. Fine-tuning hyper-

parameters and experimenting with different architectures, such as transformer-based models beyond BERT, including GPT-3 or T5, might yield better results.

Finally, creating an ensemble of models that combine the strengths of CNN, BiLSTM, and BERT could provide a more comprehensive approach to entity recognition. These future directions promise to enhance the accuracy and applicability of NER systems for the Bangla language, contributing to the broader field of NLP.

Bibliography

- [1] R. Grishman and B. M. Sundheim, “Message understanding conference— 6: A brief history,” in Proc. 16th Int. Conf. Comput. Linguistics, vol. 1, 1996, pp. 1–6.
- [2] M. Asahara and Y. Matsumoto: “Japanese Named Entity Extraction with Redundant Morphological Analysis”, HLT-NAACL 2003.
- [3] S. Sekine, H. Isahara “IREX: IR and IE Evaluation project in Japanese”, LREC 2000
- [4] Sang, E.T., & Meulder, F.D. (2003). Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. *Conference on Computational Natural Language Learning*.
- [5] Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. CoRR, cs.CL/0205028.
- [6] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit.
- [7] Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.
- [8] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. CoRR, abs/1803.07640.
- [9] Gaurav Arora. 2020. iNLTK: Natural language toolkit for indic languages. In Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS).
- [10] Sagor Sarker. 2021. BNLP: natural language processing toolkit for bengali language. CoRR, abs/2102.00405.
- [11] I. Ashrafi, M. Mohammad, A. S. Mauree, G. M. A. Nijhum, R. Karim, N. Mohammed, and S. Momen, “Banner: A cost-sensitive contextualized model for Bangla named entity recognition,” IEEE Access, vol. 8, pp. 58206–58226, 2020.
- [12] S. Nazah, M. M. Hoque, and M. R. Hossain, “Word sense disambiguation of Bangla sentences using statistical approach,” in Proc. 3rd Int. Conf. Electr. Inf. Commun. Technol. (EICT), 2017, pp. 1–6.

- [13] N. Chinchor and P. Robinson, “Appendix E: MUC-7 named entity task definition (version 3.5),” in Proc. 7th Message Understand. Conf. (MUC7), Conf. Held Fairfax, Virginia, Apr. 1998, pp. 1–21.
- [14] C. Le Meur, S. Galliano, and E. Geoffrois, “Conventions d’annotations en entités nommées-ESTER,” Rapport Technique de la Campagne Ester, 2004. [3] D. Nadeau
- [15] Ben Abacha, A., Zweigenbaum, P.: Medical entity recognition: a comparaison of semantic and statistical methods. In: Proceedings of BioNLP 2011 Workshop, pp. 56–64. Association for Computational Linguistics, Portland, June 2011. <http://www.aclweb.org/anthology/W11-0207>
- [16] B. B. Chaudhuri and S. Bhattacharya. An experiment on automatic detection of named entities in Bangla. [Online]. Available: <https://aclanthology.org/I08-5011/>. [Accessed: 18-Sep-2022]. 2008. url: <https://aclanthology.org/I08-5011/>.
- [17] Bikel, Daniel M., R. Schwartz, Ralph M. Weischedel. 1999. An Algorithm that Learns What’s in Name. Machine Learning (Special Issue on NLP), 1-20.
- [18] Bothwick, Andrew. 1999. A Maximum Entropy Approach to Named Entity Recognition. Ph.D. Thesis, New York University.
- [19] Sekine, Satoshi. 1998. Description of the Japanese NE System Used for MET-2. MUC-7, Fairfax, Virginia.
- [20] Yamada, Hiroyasu, Taku Kudo and Yuji Matsumoto. 2002. Japanese Named Entity Extraction using Support Vector Machine. In Transactions of IPSJ, Vol. 43, No. 1, 44-53.
- [21] Kudo, Taku and Matsumoto, Yuji. 2000. Use of Support Vector Learning for Chunk Identification. In Proceedings of CoNLL-2000.
- [22] Asif Ekbal and Sivaji Bandyopadhyay. 2008. Bengali Named Entity Recognition Using Support Vector Machine. In *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*.
- [23] F. Alam and M. A. Islam. “A proposed model for Bengali named entity recognition using maximum entropy Markov model incorporated with Rich Linguistic Feature Set.” In: Proceedings of the International Conference on Computing Advancements. 2020.

- [24] LeCun, Y., Bengio, Y. and Hinton, G. (2015) Deep Learning. *Nature*, 521, 436–444.
<http://dx.doi.org/10.1038/nature14539>
- [25] R. Karim et al. “A step towards information extraction: Named entity recognition in Bangla using Deep Learning.” In: *Journal of Intelligent & Fuzzy Systems* 37.6 (2019), pp. 7401–7413.
- [26] J. R. Saurav, S. Haque, and F. Chowdhury. “End to end parts of speech tagging and named entity recognition in Bangla language.” In: *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*. 2019.
- [27] Rrubaa Panchendrarajan and Aravindh Amaresan. “Bidirectional LSTM-CRF for Named Entity Recognition.” In: *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*. Hong Kong: Association for Computational Linguistics, Jan. 2018. url: <https://aclanthology.org/Y18-1061>.
- [28] Ashraf et al. “Banner: A cost-sensitive contextualized model for Bangla named entity recognition.” In: *IEEE Access* 8 (2020), pp. 58206–58226.
- [29] Redwanul Karim et al. “A step towards information extraction: Named entity recognition in Bangla using deep learning.” In: *Journal of Intelligent & Fuzzy Systems* 37 (July 2019), pp. 1–13. doi: 10.3233/JIFS-179349.
- [30] Abhik Bhattacharjee, Tahmid Hasan, Wasi Ahmad, Kazi Samin Mubasshir, Md Saiful Islam, Anindya Iqbal, M. Sohel Rahman, and Rifat Shahriyar. 2022. BanglaBERT: Language Model Pretraining and Benchmarks for Low-Resource Language Understanding Evaluation in Bangla. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1318–1327, Seattle, United States. Association for Computational Linguistics.
- [31] Jurafsky, Daniel; H. James, Martin (2000). Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition. Upper Saddle River, N.J.: Prentice Hall. ISBN 978-0-13-095069-7.
- [32] Socher, Richard; Perelygin, Alex; Wu, Jean; Chuang, Jason; Manning, Chris; Ng, Andrew; Potts, Chris (2013). Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank (PDF). EMNLP.
- [33] Mikolov, Tomas; et al. (2013). "Efficient Estimation of Word Representations in Vector Space". arXiv:1301.3781 [cs.CL].
- [34] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-term Memory.” In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. doi: 10.1162/neco.1997.9.8. 1735.
- [35] M. Z. Haque, S. Zaman, J. R. Saurav, S. Haque, M. S. Islam and M. R. Amin, "B-NER: A Novel Bangla Named Entity Recognition Dataset With Largest Entities

and Its Baseline Evaluation," in IEEE Access, vol. 11, pp. 45194-45205, 2023, doi: 10.1109/ACCESS.2023.3267746.

[36] N. Alshammari and S. Alanazi, "The impact of using different annotation schemes on named entity recognition," Egyptian Informat. J., vol. 22, no. 3, pp. 295–302, 2021.

[37] M. K. Malik and S. M. Sarwar, "Named entity recognition system for postpositional languages: Urdu as a case study," Int. J. Adv. Comput. Sci. Appl., vol. 7, no. 10, pp. 141–147, 2016.

[38] J. Bos, V. Basile, K. Evang, N. J. Venhuizen, and J. Bjerva, "The Groningen meaning bank," in Handbook of Linguistic Annotation. Springer, 2017, pp. 463–496.

[39] L. A. Ramshaw and M. P. Marcus, "Text chunking using transformationbased learning," in Natural Language Processing Using Very Large Corpora. Cham, Switzerland: Springer, 1999, pp. 157–176.

[40] S. Sarker. (2020). BanglaBERT: Bengali mask language model for Bengali language understanding," [Online]. Available: <https://github.com/sagorbrur/bangla-bert>