Shahjalal University of Science and Technology
Department of Computer Science and Engineering

Named Entity Recognition in Bengali Language

Md. Tanjiqur Rahman Prince
Reg. No.: 2016331015
4th year, 2nd Semester

Md. Rafsunjani Khan
Reg. No.: 2016331057
4th year, 2nd Semester

Department of Computer Science and Engineering

Supervisor
Md Mahadi Hasan Nahid
Assistant Professor
Department of Computer Science and Engineering

7th July, 2021

Named Entity Recognition in Bengali Language

A Thesis submitted to the Department of Computer Science and Engineering, Shahjalal University of Science and Technology, in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering.

By

Md. Tanjiqur Rahman Prince
Reg. No.: 2016331015
4th year, 2nd Semester

Md. Rafsunjani Khan
Reg. No.: 2016331057
4th year, 2nd Semester

Department of Computer Science and Engineering

Supervisor
Md Mahadi Hasan Nahid
Assistant Professor
Department of Computer Science and Engineering

7th July, 2021

Recommendation Letter from Thesis Supervisor

The thesis entitled *Named Entity Recognition in Bengali Language* submitted by the students

1. Md. Tanjiqur Rahman Prince
2. Md. Rafsunjani Khan

is under my supervision. I, hereby, agree that the thesis can be submitted for examination.

[illegible]
07.07.2021

Signature of the Supervisor

Name of the Supervisor: Md Mahadi Hasan Nahid

Date: 7th July, 2021

Certificate of Acceptance of the Thesis

The thesis entitled *Named Entity Recognition in Bengali Language* submitted by the students

1. Md. Tanjiqur Rahman Prince
2. Md. Rafsunjani Khan

on 7th July, 2021 , hereby, accepted as the partial fulfillment of the requirements for the award of their Bachelor Degrees.

[illegible]
07.07.2021

Head of the Dept.	Chairman, Exam. Committee	Supervisor
Dr. Mohammad Abdullah Al Mumin	Dr. Mohammad Abdullah Al Mumin	Md Mahadi Hasan Nahid
Professor and Head	Professor and Head	Assistant Professor
Department of Computer Science and Engineering	Department of Computer Science and Engineering	Department of Computer Science and Engineering

Abstract

In this era of technology, Natural Language Processing (NLP) has vast presence in almost every field of computer science. And Named Entity Recognition (NER) is a very important sub-task in the field of NLP. NER refers to the task of recognizing and classifying named entities mentioned in unstructured text into tags which are predefined such as person names, organizations, locations, time-date, events, monetary values, units etc. This research report conveys the summary of our in-depth study and analysis of notable previous research works and existing NER systems in Bengali as well as other languages. Afterwards, we implemented three different deep learning architectures. Among those, the Bidirectional Encoder Representations from Transformers (BERT) based architecture performed best in all measurement metrics and produced a state-of-the-art macro f1-score of 73.72% among all currently existing Bengali NER systems. We used deep transfer learning method in that architecture. Other two architectures are Bi-directional Long Short Term Memory (BiLSTM) neural network and a combination of Bi-directional Long Short Term Memory (BiLSTM) and Conditional Random Fields (CRF). All of three architectures were trained on 14,716 Bengali sentences consisting a total of 1,65,715 words and 20,794 unique words. For tagging the words BIO tagging scheme was applied in the used dataset. Five different classes have been used for tagging the words in the dataset. Then, we used our BERT based architecture to predict NER tags of more than 4 lakh words, then using human annotators we created a dataset of about 20 thousand words. We then analysed them to compare our model's performance and also identify the factors that affected the performance.

Keywords: Natural Language Processing (NLP), Named Entity Recognition (NER), Bi-directional Long Short Term Memory (Bi-LSTM), Conditional Random Fields (CRF), Deep Learning, BIO tags, Bidirectional Encoder Representations from Transformers (BERT).

Acknowledgements

We would like to thank the Department of Computer Science and Engineering, Shahjalal University of Science and Technology, Sylhet 3114, for giving us the opportunity to work on this thesis.

We would like to express our heartfelt gratitude to our ever helpful supervisor Md Mahadi Hasan Nahid who has always been there to guide us throughout the research.

We would also love to convey our gratefulness to our seniors Sarowar Hossain , Souvik Roy and Zillur Rahman Sourav who provided us with valuable resources and shared their working experience with us.

We are also grateful to the SUST CSE Developer Network (SCDN) for their open source Bengali Dataset (corpus).

We are also extremely grateful to the countless authors and researchers whose valuable contributions brought the research field of our interest to the current level.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Information extraction from text data is an extremely vital issue in the field of Natural Language Processing (NLP). Named Entity Recognition (NER) is a significant strategy to extract information from text data. This sub-task of information extraction addresses and orders named entity referenced in unstructured text into labels which are predefined such as person names, organizations, locations, time-date, events, monetary values, units, and so forth. As a vital apparatus for information recovery, there are numerous fields in which it has demonstrated its significance. Such fields are text summarization, search result optimization, article captioning, question answering system and some more.

Bengali, being the 6th most communicated in languages all throughout the planet, there has not been a substantial amount of work done in Bengali computational linguistics[1]. In spite of the fact that there has been some work done in Bengali NER beforehand however, it is a long ways behind its English counterpart. If we look closely at those previous works, we notice that both the number of tags and quality of datasets are poor and thus, have huge scope for improvement. We therefore, applied and fine-tuned previously used deep learning methods as well as applied those methods also that have not been previously exploited for Bengali NER and assessed those results. We also established a dataset of impactful standards.

1.1 Motivation

With the rise in use of technology in our day to day lives, the amount of textual information we have to deal with is increasing exponentially. It has reached such a point that manually reading and extracting information is beyond imagination. To overcome this hurdle we require top-notch automated systems to extract information from textual data to aid the comprehension and use of the vast amount of raw data available. Besides information extraction, other fields of Natural Language Processing (NLP) such as designing efficient search algorithms, question answering systems, text summarization, and many other sub-tasks requires Named Entity Recognition. In the English language a generous amount of work has been done in recognizing Named Entities (NE). But sadly there has not been such advancement in case of Bengali language. And this lackness is also affecting the advancement of previously mentioned sub-tasks of NLP, in Bengali language. Then again, this also means that there is a huge possibility of contributing to the advancement of Bengali NER systems.

1.2 Goal

In pursuit of advancement of Bengali NER systems we first aimed at educating ourselves about NER systems by studying some noteworthy works in this field. From the beginning of this research work one of our goal was to design and build a high performing NER system for Bengali language. For this purpose we designed and implemented different architectures. One of those architectures performed really good and produced macro f1-score of 73.72% which is state-of-the-art for Bengali NER systems. In this way we met our first goal. Our another goal was to develop a dataset of significant standard using prediction of our NER system and validating those predictions using human annotators. We also met that goal. In this way we fulfilled our thesis objectives. After accomplishing these goals now we plan to compose a paper constituting our findings for the future researchers. We also aim to extend this research work in the future for betterment of NLP tasks in Bengali language.

Chapter 2

Background Study

2.1 Natural Language Processing

As a sub-field of linguistics, computer science, and artificial intelligence, Natural Language Processing (NLP) is a broad term that refers to the automated manipulation of natural language by computer, such as speech and text[2]. NLP is a multidisciplinary field concerned with both understanding and utilizing natural languages in order to facilitate human–computer interaction. NLP has existed for over 50 years and evolved from the field of linguistics with the advent of computers[3].

Natural language processing problems frequently involve speech recognition, language comprehension, and text generation[3]. NLP is an extremely difficult task because present day computer systems are incapable of analyzing and comprehending analog data provided by a human user. High level programming languages like python are the closest we have come to establishing a type of interaction with the computer, due to the fact that human input has most of the time been ambiguous, unstructured, undefined, and perplexing for a logical machine that understands only true or false. Thus, unambiguous, precise, and structured programming languages were introduced. However, we, as humans, evolve, and so does our technology and the computational power of computers. Computer scientists are currently using machine learning and artificial intelligence to discover patterns in analog data provided directly by humans in natural language and converting it to digital data that can be processed and understood by a logical machine. The majority of current research on natural language processing is focused on Search Auto-correct and Auto-complete, Machine translation, Text Summarization etc.

2.2 Named Entity Recognition

Before understanding the task of Named Entity Recognition (NER) we need to understand what is Named Entity (NE). According to the formal definition, NE is a real-world object, such as persons, locations, organizations, products, etc., that can be denoted with a proper name[4]. For example, "আশরাফুল" is the name of a person, "ঢাকা" is the name of a location, "৭ই আগস্ট" is a date. These are all different types of named entities. There are three broad classes of named entities: name, quantity and date and time[3].

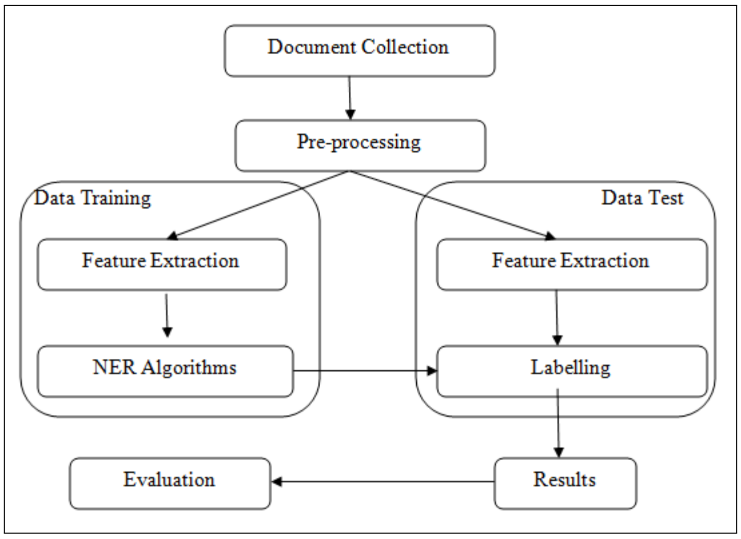

Figure 2.1: Generic Named Entity Recognition System
[5]

The first task of NER is the detection of these named entities from sentences. Then the detected named entities are classified into different classes.

The purpose of a NER system is to identify and classify each token of words in a document into one of several predefined classes including person name, location name, organization name, miscellaneous name (date, time, number, percentage, monetary expressions, etc.) and none-of-the-above[6].

There are many approaches on which an NER system can be based on. All these approaches can be classified into three main types[7]:

2.2.1 Lexicon-based approach

Lexicon-based approach makes use of an external lexicon or gazette to associate chunks of text with entity names[8]. The Lexicon-based methodology rates a document by summing the sentiment scores of all its terms. Each word in the pre-prepared sentiment lexicon should have a corresponding sentiment score. The negation forms of vocabulary words should be added to the lexicon as separate entries and given precedence over their non-negation counterparts. Additionally, simple rules can be used to manage negation terms. This approach has a number of drawbacks. For example, in most cases, the presence of more positive words in online reviews or any other online text sources does not imply that the review is positive or vice versa. In most cases, scoring documents from disparate domains using the same lexicon is impossible.[8]

2.2.2 Rule-based approach

With the aid of language-specific features, rule-based techniques are intuitive in their recognition of named items. There are three ways to approach NER tasks using rules: rules (lexical rules, contextual rules, morphological lexicon), rules combined with dictionaries (Gazetteers), and rules combined with feature selection algorithms. Considering the evident morphology of named entities, regular expressions can match the vast majority of data, time, money, and percentage entities. Though rule-based approaches are no longer widely used, they are still used in some NER architectures[9].

2.2.3 Machine Learning approach

Machine learning is a subset of Artificial Intelligence (AI) that enables computers to learn and develop automatically without being explicitly programmed[10]. In this approach the recognition of NEs shows better results than previously mentioned approaches. Different classification algorithms of machine learning field such as Naive Bayes, Support Vector Machine (SVM), Hidden Markov Model (HMM) etc. have already been used and have shown great performance. Word tokens which are not previously seen by the trained system or not in the gazetteer or lexicon can also be recognized as NEs with more precision. As a result, in the modern NER systems, machine learning approach is the most followed one.

2.2.4 Deep Learning

Deep Learning is a subfield of machine learning that focuses on algorithms inspired by the structure and function of the brain[11]. These algorithms are referred to as artificial neural networks. Deep learning helps computers to do what people do instinctively, which is learning by example. At present day deep learning is being a crucial component of solving many critical problems such as signal processing, natural language processing, computer vision and the list is growing day by day. It is achieving accomplishments that were previously unattainable. As a result, deep learning is getting a lot of attention recently[12].
Now a days, deep learning has a lot of impact on NLP researches. Different algorithms are surpassing the current result and affecting in a better way and giving state of the art result. Like other sub-tasks of NLP, deep learning is also being used in NER. Many deep learning approaches have already been exploited in this problem and those techniques show better results than previously mentioned approaches. Though there has been a revolution in foreign language NER systems because of using deep learning techniques, but there has not been enough development in NER systems for Bengali language as deep learning approaches has not been used much for solving this task in Bengali.
As NER is a classification problem, different deep learning classification algorithms such as Recurrent neural network (RNN), Long short-term memory (LSTM), Convolutional neural network (CNN), different Transfomers architectures such as Bidirectional Encoder Representations from Transformers (BERT) etc. are being used. We have thoroughly studied those algorithms.

2.2.4.1 Artificial Neural Networks (ANN)

The concept of ANNs is based on the premise that the human brain's functioning may be replicated using silicon and wires as live neurons and dendrites by forming the appropriate connections[13].

The human brain is made up of 86 billion neurons, which are nerve cells. Axons link them to thousands of other cells. Dendrites receive information from the external environment and sensory organs. These inputs generate electric impulses that rapidly propagate across the brain network. A neuron then has the option of forwarding the message to another neuron or not forwarding it at all. Multiple nodes comprise ANNs, which mimic biological neurons in the human brain. Neurons are linked together and communicate with one another through connections. The nodes are capable of accepting data as input and performing basic actions on it. Other neurons get the output of these processes. Activation or node value refers to the output of each node.Each connection has a weight assigned to it. ANNs are capable of learning, which occurs through weight value changes. Here, A simple ANN is shown below.

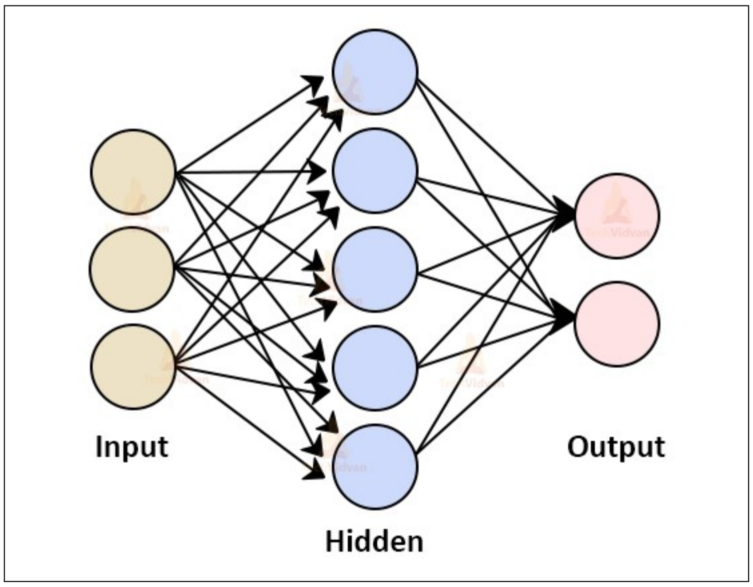

Figure 2.2: A simple ANN Architecture

[14]

The equation of ANN.

$$X_n = f_n(W_n X_{n-1} + b_n) \tag{2.1}$$

Here,

X_n = The output of the nth artificial neuron. f_n = The activation function of the nth artificial neuron. W_n = The weight of the nth artificial neuron. X_{n-1} = The output of the (n-1)th artificial neuron. b_n = bias value of the nth artificial neuron.

2.2.4.2 Convolutional Neural Network(CNN)

Convolutional Neural Network (CNN) contains multiple differentiable layers. Each layer in the network contains a activation function. The perpose of these functions are converting inputs into outputs. CNN contains three main layers. They are:

- Convolutional Layer.
- Pooling Layer.
- Fully Connected Layer.

Besides these layers there are other two more important parameters which are dropout layer and the activation functions. CNN can have hidden layers that can convolve through multiplication. This is a feed forward architecture. Though CNNs are mostly used in Image Processing and Computer Vision domain there are a lot of use of this model in NLP tasks. They are mostly used for classification and feature extraction. Also, CNNs are used as a part of different hybrid models. A simple CNN architecture is shown below.

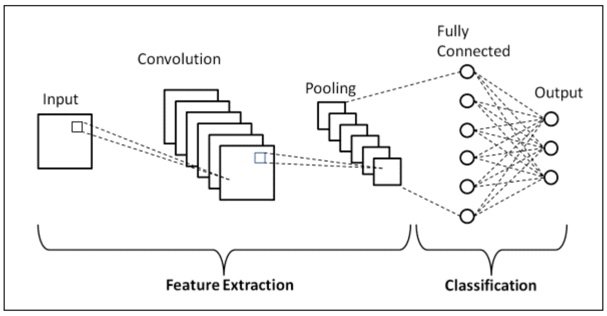

Figure 2.3: A simple CNN Architecture.

[15]

2.2.4.3 Recurrent Neural Network (RNN)

A recurrent neural network (RNN) is a kind of artificial neural network that operates on sequential or time series data. This deep learning algorithm is frequently used to solve ordinal or temporal issues like as language translation, natural language processing (NLP), speech recognition, and picture captioning. Recurrent neural networks utilize their training inputs like feed forward and convolutional neural networks (CNNs). They are characterized by their "memory," which they use to impact the current input and output. While orthodox deep neural networks presume that their inputs and outputs are independent, recurrent neural networks' outputs are reliant on the sequence's previous parts. RNNs allows to use of prior outputs as inputs while maintaining hidden states and earlier inputs in a sequence have an effect on later propagations. The overall working of this architecture is very close to the human memory because of storing and analyzing process of data. Because of this capability, RNNs are hugely used in different NLP tasks.

RNNs contains nodes that are used layer by layer and different weights are used to affect the node outputs and connecting them. Each node contains activation functions. Errors generated from each sequence is same as the sum of all target signal's deviations from respective activations computed by the RNN. The total error in the process is the sum of all errors of individual sequences.

There are three types of layers in RNNs. They are:

- Input Layer.
- Hidden Layer.
- Output Layer.

A RNN architecture is shown below:

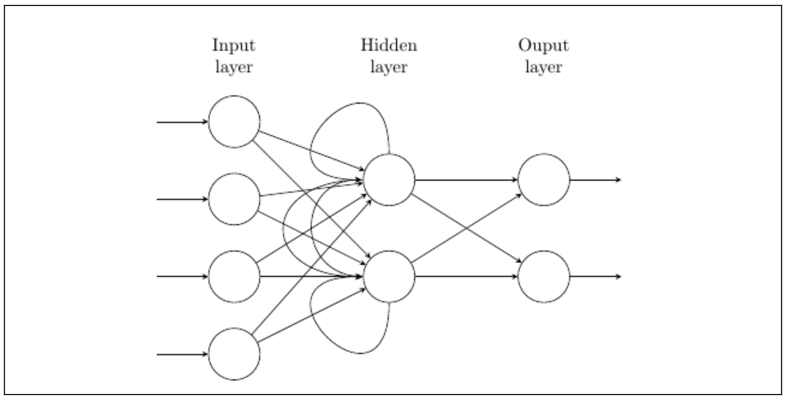

Figure 2.4: A simple RNN Architecture

[16]

Applied equations for a simple RNN are:

$$h_t = Wf\left(h_{t-1}\right) + W^{(hx)}x_{[t]} \tag{2.2}$$

Here,

$x_t =$ Input layer vector.

$y_t =$ Output layer vector.

$h_t =$ Hidden layer vector.

$W =$ Parameter matrix.

2.2.4.4 Long short-term memory (LSTM)

Long short-term memory is a variant of Recurrent Neural Network (RNN). LSTMs are able to learn from long term dependencies. First, Hochreiter and Schmidhuber introduced LSTM back in 1997 and later, many people refined and popularized it.

LSTMs are purpose-built to eliminate the problem of long-term dependencies. Keeping knowledge in nodes for extended periods of time is practically default for them; it is not something they have to struggle to learn. In the process of designing the algorithm memory cells and gates are used that learn to bridge time intervals. The task of removing, adding and retaining a memory from a cell is defined by the gates. As a result the LSTM network

is able to learn which memories are needed to be retained and which to be removed. As a result, the neural network can provide significant performance with great precision for some very critical modern world problem.
Here a LSTM architecture is shown below.

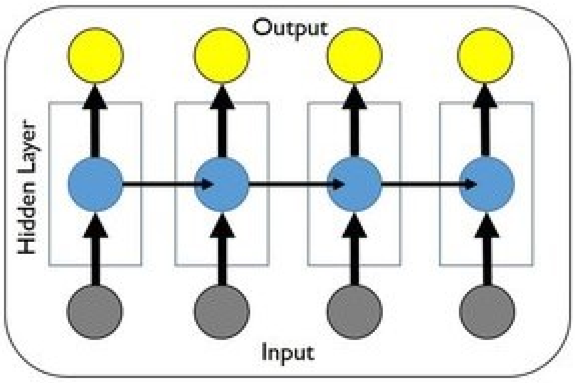

Figure 2.5: A simple LSTM cell or node.
[17]

The equations applied in an LSTM node with a forgotten gate for the forward pass is given below :

$$f_t = \sigma_g \left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{2.3}$$

$$i_t = \sigma_g \left(W_i x_t + U_i h(t-1) + b_i\right) \tag{2.4}$$

$$o_t = \sigma_g \left(W_o x_t + U_o h_{t-1} + b_o\right) \tag{2.5}$$

$$c_t = f_t * c_{t-1} + i_t * \sigma_c \left(W_c x_t + U_c h_{t-1} + b_c\right) \tag{2.6}$$

$$h_t = o_t * \sigma_h \left(c_t\right) \tag{2.7}$$

Here,

$$x_t \in \mathbf{R}^d : \text{ Input vector to the LSTM unit} \tag{2.8}$$

$$f_t \in \mathbf{R}^h : \text{ Forget gate's activation vector} \tag{2.9}$$

$$o_t \epsilon \mathbf{R}^h : \text{ Output gate's activation vector} \tag{2.10}$$

$$i_t \in \mathbf{R}^h : \text{ Input/update gate's activation vector} \tag{2.11}$$

$$h_t \in \mathbf{R}^h : \text{ Hidden state vector also known as output vector of the LSTM unit} \tag{2.12}$$

$$c_t \in \mathbf{R}^h : \text{ Cell state vector} \tag{2.13}$$

2.2.4.5 Bidirectional Long Short-Term Memory (BiLSTM)

Bidirectional Long Short-Term Memory (BiLSTM), is a sequence processing model that consists of two LSTMs. One of them takes the input in a forward direction, and the other in a backwards direction[18].
BiLSTM can use knowledge from both forward and backward directions. Total amount of information available to the network is effectively increased by BiLSTMs which effectively improves the context available to the model. Both forward and backward states are propagated in the forward progression before the output of the node is propagated to next node. In terms of backward propagation the same strategy is applied just in reverse orders. BiLSTM architecture has shown great results in different NLP tasks.

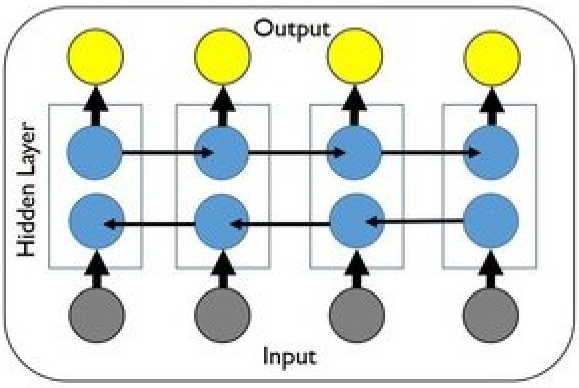

Figure 2.6: A simple Bi-LSTM cell or node.

[17]

2.2.4.6 Transformer Model

Transformer which is a novel architecture, was first introduced in a paper named "Attention is all you need". This architecture uses attention mechanisms to boost up the speed with which the training of these models are done.
Here, Model architecture of Transformer is shown.

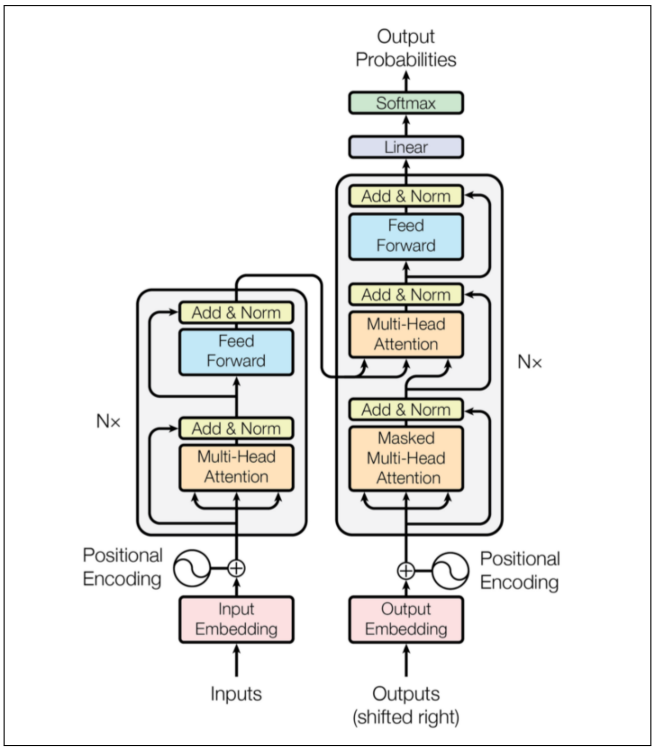

Figure 2.7: Architecture of the Transformer.

[19]

Transformer is an architecture that transforms one sequence to another sequence with the assist of two parts. They are : Encoder and Decoder. But it is different from the previously stated sequence-to-sequence model as it does not applies Recurrent networks like RNN, LSTM, GRU etc [20]. Untill now Recurrent Networks produced best performances in capturing the timely dependencies but in the paper mentioned above proves that transformer architecture can improve the performance in many NLP tasks with attention mechanism and without any kind of Recurrent Networks[19].

In the later work of this thesis we want to deploy Bidirectional Encoder Representations from Transformers (BERT) for our NER task, which is a Transformer based model. And it performed very good in NER task of other languages like English, Korean etc.

2.2.4.7 Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) is transformer based architecture. It is specially structured to pre-train deep bidirectional representations from untagged text data using conditioning on both right and left context jointly in all layers[21]. Architecture pre-training technique has already shown great advancements in different deep learning research fields. In field of NLP language model pre-training has been proved to be very effective. There are two ways we can apply pre-trained language representations to different NLP tasks. They are:

- Feature based.
- Fine-tuning.

BERT consists in the second category. The model architecture of BERT is a multi-layer bidirectional Transformer encoder based on the original design of the architecture described in the paper named "Attention is all you need"[19]. It uses a bidirectional mechanism, where every token can attend to previous as well as next token in the self-attention layers of the Transformer during training time using a "masked language model" (MLM) pre-training objective. BERT architecture comes with two different sizes. They are:

- $\text{BERT}_{\text{Large}}$ (L=24, H=1024, A=16, Total Parameters=340M).
- $\text{BERT}_{\text{Base}}$ (L=12, H=768, A=12, Total Parameters=110M)

Here, L denotes the number of Transformer encoder blocks, H denotes the hidden size and A is for the number of self-attention heads.
There are two steps in BERT framework, one is Pre-training and another is Fine-tuning.
BERT does not uses orthodox right-to-left or left-to-right mechanism during pre-training instead it uses a bi-directional approach. The process of pre-training uses two unsupervised tasks, they are: Masked Language Model (MLM) and Next Sentence Prediction (NSP).

During the MLM task 15% of the input tokens are masked randomly and then prediction of those masked tokens are done. BERT only predict the masked words rather than reconstructing the entire input like de-noising auto-encoders. Though it allows BERT to obtain a bi-directional pre-trained model nature but there is a downside which is creating a mismatch between fine-tuning and pre-training as [MASK] token does not appear during the task of fine-tuning. And for eliminating the mismatch the model does not replace "masked" words always with [MASK] tokens. 15% of the total tokens are chosen randomly for prediction by training data generator and if a token is chosen, 80% of the time the token is replaced with [MASK] tokens, 10% of the time with random tokens and 10% of the time it is unchanged. And after that the chosen token will be used to predict the original token.

In the NSP task the model understands the sentence relationship and the model is pre-trained for binarized next sentence prediction task which is trivially generated from any monolingual corpus. During selection of sentences S1 and S2 for each pre-training example, 50% of the time S2 is actual following sentence of S1 and 50% of the time it is a random sentence. This NSP task has a close relation with representation learning objective described in the paper "Discourse-Based Objectives for Fast Unsupervised Sentence Representation Learning"[22].

Fine-tuning is simple, since the Transformer's self-attention mechanism enables BERT to simulate a wide variety of downstream tasks—whether they include single texts or pairs of texts—by switching out the relevant inputs and outputs. For this task most model hyper-parameters are same as pre-training. The parameters which change during this phase are learning rate, batch size and number of training epochs.

The overall architecture of Pre-training and Fine-tuning process is described in the diagram below:

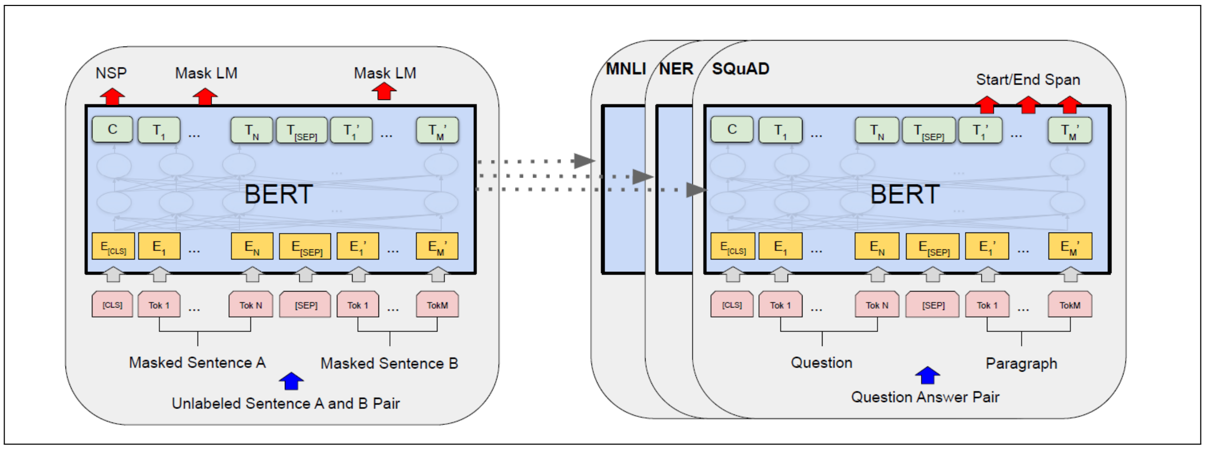

Figure 2.8: Pre-training and Fine-tuning process of BERT.

[21]

BERT takes embedded vectors as input. These embedded input vectors are made using three components which are token embedding, Segment embedding and position embedding. These embeddings are summed up to create the input embeddings. The overall input representation of BERT is illustrated in the diagram below.

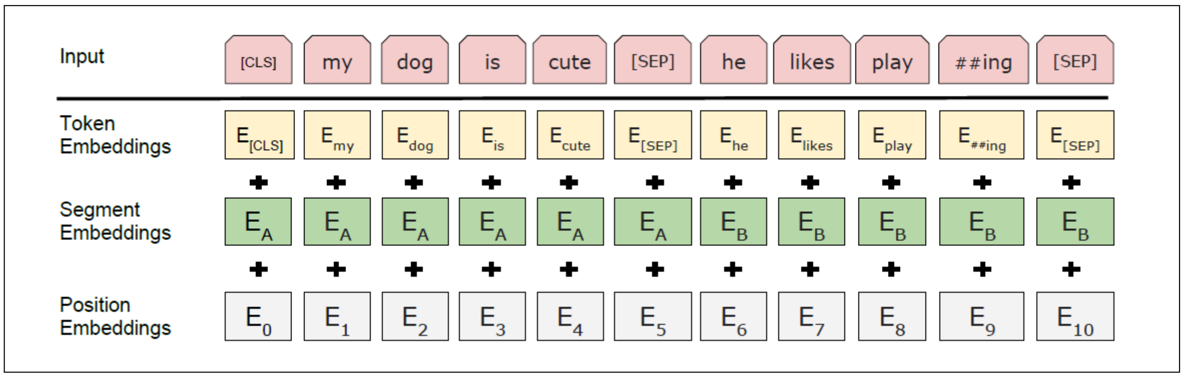

Figure 2.9: Input representation of BERT.

[21]

2.3 Different Tagging Schemes

2.3.1 BIO or IOB

In multi-worded entities B marks the beginning of an entity, I marks the inside or internal part of an entity and O marks the non-entity words, saying us that it is outside the named entity. Moreover, B also marks the single-worded entities.

2.3.2 IOE

In multi-worded entities E marks the ending of an entity, I marks the inside or internal part of an entity and O marks the non-entity words, saying us that it is outside the named entity. Moreover, E also marks the single-worded entities.

2.3.3 IOBES

In the multi-worded entities B marks the beginning of an entity, I marks the inside of an entity, E marks the ending of an entity and O marks the non-entity words. Additionally, S marks the single-worded entities. As this tagging scheme marks boundaries with both B and E tags and marks single entities with a tag S, so it provides more information.

2.3.4 BIES

This tagging scheme marks all the entities in the same way as the IOBES scheme. It also marks the beginning, inside and ending of multi-worded non-entities as B-O, I-O, E-O and marks single-worded non-entities as S-O.

2.3.5 Others

IO, BI, IE are three other tagging schemes. Among all the different tagging schemes the IO tagging scheme is the simplest and only tags I for named entities and O for non-entities. It fails to recognize a sequence of words as a single entity.

The BI tagging scheme is similar to BIO or IOB tagging scheme. It additionally tags the beginning and inside of a non-entity as B-O and I-O.

Similarly, IE tagging scheme is like IOE schemse and adds I-O and E-O tags to the inside and ending words of a non-entity.

2.4 Related Works

2.4.1 English and other Languages

English is the most spoken language of the world. Most of the digital communication happens in English language. So naturally, substantial amount of work has been done in NER in the last few decades. We took a close look at some of the research papers of recent times to gain valuable insights.

The first ever work on NER system[23] was arranged by Grishman and Sundheim (1996) in the 6th Message Understanding Conference. From then on, numerous researches have been carried out in this field. It started from hand-crafted rule, lexicon, lettering and spelling features and ontology based NERs. Then gradually moved towards using feature-engineering and machine learning base NERs to train machines to recognize those features. These NER systems were highly domain specific and language specific due to the dominance of such domain specific and language specific features in the list of features. As time went on, research expanded to develop generalized NER systems that were based on machine learning and took little feature engineering to function. Collobert et al.(2011) is one of the front-runners of work on building this kind of domain independent NER systems[24].

Among some of the early works in such NER systems only labeled seeds were used by Collins and Singer (1999) along with 7 features including orthographic feature, entity context, etc[25]. An unsupervised system was suggested by Etzioni et al.(2005) with a view to increasing the recall rate of NER systems[26].

Based on the two previously mentioned researches Nadeau et al.(2006) built a gazetteer using unsupervised system to extract gazetteer and combining it with commonly available ones and managed to achieve F-scores of 88%, 61%, and 59% on location, person, and organization entities, respectively in MUC-7 (Chinchor and Robinson, 1997)[27].

Light knowledge about the syntax and inverse document frequency (IDF) was used by Zhang and Elhadad (2013) for building an unsupervised NER system on biology (Kim et al.,

2004) and medical (Uzuner et al., 2011) data, achieving 53.8% and 69.5% accuracy, respectively[28]. The seeds were used by their model to find text that might contain named entities. Then it detects noun phrases and removes those with low IDF values, and classifies the remaining ones to predict their tags.

Collobert and Weston (2008) used feature vectors consisting orthographic features, dictionaries and lexicons which were later replaced with a pre-trained word embedding (Collobert et al., 2011), with one of the first proposed word level neural network architectures for NER[24]. Here, the words of a sentence are fed to Recurrent Neural Networks (RNN) and comes out as a list of word embedding. A convolution layer followed by a CRF layer gives us the final prediction. By including gazetteers and SENNA embedding this approach yielded an F1 score of 89.59% on English CoNLL 2003 dataset. Yan et al.(2016) changed things up a bit and highlighted the performance improvement after adding various features like CRF, orthographic case, POS, word embeddings by implemented word level feed forward NN, bi-directional LSTM (bi-LSTM) and window bi-LSTM for NER of English, German and Arabic. This yielded them a 88.91% F1 score on English and 76.12% on German[29].

Arijit Sehanobish and Chan Hee Song used some lexicographical feature of Chinese character like the graphical elements called radicals which often provide valuable semantic insights in combination with Convolutional Neural Network (CNN) models for NER. Their approach yielded a state-of-the-art (SOTA) F1 score of 71.81 on the Weibo dataset, and a SOTA F1 score of 96.49 on the MSRA dataset[30].

Years of research has brought NER systems of many languages like English, Chinese etc. to a satisfactory level. Many standard datasets, pre-trained models, pre-trained embedding models, dictionaries etc are readily available, thus, making the starting point of new researches very smooth.

2.4.2 Bengali Language

Though there have been a lot of work done in the field of NER for other language, works in Bengali language in this field is lacked behind. The reason behind this lacking is mainly because of the limitations in the required resources. Being one of the most spoken and old language there is a huge limitation in corpora. Other obstacles are the pattern or structure of Bengali sentences which is very complex. Besides these limitations there are also other complex reasons behind the lacking of research work in Bengali NER systems. Those limitations are broadly discussed in the challenges chapter.

Despite of these challenges there have been some work done in this field for Bengali language. Among these research works notable works are discussed here.

First published machine learning approach for NER in Bengali was done by Asif Ekbal et al.(2007)[31]. They used a statistical Hidden Markov Model based NER system along with Context Dependency. For handling unknown words they used Viterbi algorithm. They used corpus containing 150,000 wordforms which was initially tagged with a HMM based parts of speech (POS) tagger. Later the corpus was partially tagged with 4 types of NE classes, they are person, location, organization and miscellaneous and the annotation was done in BIE scheme. The corpus was developed from the archive of a leading Bengali newspaper available in web. After exploiting their proposed architecture on the corpus they performed 10 fold cross validation of the systems and got F-score of 84.5% where precision is 79.48% and recall is 90.2%. In 2008, another paper was published By Asif Ekbal, Rejwanul Haque and Sivaji Bandyopadhyay which proposed a new approach called Conditional Random Field (CRF)[32]. They trained on the same dataset using this statistical method and better F-Score, Recall and Precision of 90.7%, 93.8% and 87.8% accordingly using 10 fold cross validation. They also applied different features such as contextual window [-2, +2], prefix and suffix of length up to three, POS information of the window [-1, +1], first word of sentence,current word, NE information of the previous word, different digit features and the various gazetteer lists and these features are best suited for Bengali NER. Another architecture which is based on Support Vector Machine (SVM) was proposed by Asif Ekbal and Sivaji Bandyopadhyay[33]. In this case also they used the same dataset. Along with SVM the proposed NER system architecture uses other different contextual information. They have got Recall, Precision and F-Score of 94.3%, 89.4% and 91.8% which produced state of the art result then. they approximated that the reason behind this great result might be the effectiveness of the architecture to handle the diverse and overlapping features of the highly inflective Bengali language. In the same year a Maximum Entropy based approach was proposed by Mohammed Hasanuzzaman et al.[34] The proposed system uses different contextual information of word along with along with the variety of orthographic word-level features. They also used both language specific and independent features. The whole system was trained on an annotated corpora obtained from the IJCNLP-08 NER Shared Task on South and South East Asian Languages (NERSSEAL). The proposed system produced recall, precision, and F-Score values of 88.01%, 82.63%, and 85.22% respectively. The NER system shows better performance with the use of POS information. Another paper was published by Asif Ekbal et al.in year 2009 showing the performance comparison between three different architectures ME, CRF and SVM[35]. They proposed a multi-engine approach combining the outputs of those classifiers. They also developed two different variants of SVM, one is based on forward parsing and another one is based on backward parsing. Using context pattern learned from an unlabeled corpus of 3 million wordforms and various post processing techniques developed by them, observing the different kinds of errors involved in each classifier models. After they combined all the four systems and made a final multi engine system. In that system they also used three weighted voting techniques while combining those model and the produced multi-engine model produced better results than all previous architectures. The Recall, Precision and F-Score values of the proposed multi engine approach along with voting techniques respectively are 93.98%, 90.63% and 92.28%.

After these research works in Bengali NER, the next notable work done was in year 2017 by Samima Parvez[36]. The author worked on Bengali news data for this task. She proposed a HMM based NER system with Regular Expression and Bengali grammar. In the system she used POS tagging to recognize proper noun which help in recognizing NEs. The proposed system shows F-Score of 90% and precision and Recall of 85.7% and 94.7% respectively. Meanwhile, S. Chowdhury et al.published a paper on Bengali NER using deep learning methods[37]. At first they used baseline CRF architecture with token features. Then they added three more features which are POS, gazetteers and word embedding features. The later architecture outperformed all other architecture. They also deployed Bi-LSTM-CRF combined with these features architecture for the task. But the result could not outperform the CRF with four features architecture. The CRF with four language feature architecture shown F-Score of 58%. They presented a dataset named Bangla Content Annotation Bank (B-CAB) which includes seven NEs with particular focus on Bangladeshi Bengali. They deployed all these architectures in this dataset. They also deployed their architecture on IJCNLP dataset for experimenting the proposed method on Indian Bengali. They then compared the results and found that the proposed architecture performs differently in the two different datasets as Bengali of the two regions are slightly different.

In the same year, A different approach for NER was proposed by Nabil Ibtehaz et al.which uses a partial string matching approach[38]. Their algorithm is based on Breadth First Search (BFS) on Trie data structure and it is augmented with dynamic programming for avoiding repetition. Their algorithm is capable of doing both identifying NEs present on a text and estimating the actual NEs from faulty data. They deployed their technique on a closed domain dataset. The dataset was constructed from different Bengali newspaper mostly Prothom Alo and many others and the domain was sports news. They also claimed that their algorithm can also be used to extract information from texts and can even used as spell checker.

Md Jamiur Rahman Rifat et al.published a review paper in year 2019 describing previous methodologies and, discussing and comparing the performances of the NER systems[39]. They also deployed some deep learning approaches for NE recognition. Among those deep learning approaches Bidirectional Gated Recurrent Unit (BGRU) with CNN model performed better than others with f1 score of 72.66%.

Jillur Rahman Saurav et al.proposed an end to end solution for Bengali NER[40]. In their proposed architecture they did not use any handcrafted features like word suffixes or affixes, gazetteers or any kind of dictionary. They deployed and compared performances of different BiLSTM based deep learning architecture. Among them BiLSTM-CNN-CRF architecture performed best with f1 score of 62.84%. They deployed these models on a dataset prepared by them from different Bengali news portal containing 10,290 sentences and 1,76,029 NE annotated word tokens.

In the year 2020, Imranul Ashrafi et al.used transfer learning technique for the task[41]. They deployed different BERT based architectures on a corpus of 72,000 annotated sentences consisting 8,85,090 number of samples. Among various exploited architectures they found that their BERT-BiLSTM-CRF with class weight architecture performs best for the task producing micro f1-score of 90.64%, macro f1-score of 65.96% and f1-MUC score of 72.04%.

Chapter 3

Training Corpus Acquisition and Exploration

The first requirement of any supervised machine learning algorithms is a large tagged dataset. After we decided on our approach we started searching for available datasets.

3.1 Corpus Acquisition

There was a predominant scarcity of publicly available datasets for Bengali NER and so we looked for datasets available within our university network. Our immediate seniors Souvik Roy and Sarwar Hossain worked on a news corpus prepared by Pipilika and manually tagged the sentences into 5 different named entities[42]. We used their final dataset for training our own model. The corpus of Pipilika contained sentences scraped from different online Bengali newspapers. It consisted of sentences from news articles of different domains ranging from district news to international news and everything in between. There was a lot of variety in writing styles of different authors, but still we may conclude that the sentences were all formal to some extent. The dataset consisted of almost 15 thousand sentences totaling to almost 1.7 lakh words having an average of 11 words per sentence. Out of the 1.7 lakh words there were around 20 thousand unique words. Those unique words were tagged manually into 5 NE classes namely:

Named Entity	Number of words
Person	5625
Location	4230
Organization	2805
Date	1387
Event	386

Table 3.1: Entity-wise word count in training dataset.

At the beginning, we were planning to prepare our own datasets by increasing both the number of tags and the number of tagged words by combining our efforts with that of crowd-sourcing. But due to an elongated lockdown of public universities our plan of crowd-sourcing nipped in the bud.

3.2 Data Exploration

After acquisition of dataset we looked into the dataset to gather knowledge about its composition. We have noticed that the number of tagged words for person name and location name are around 5 thousand followed by number of tagged words for organization name, the date tag has just about 1.5 thousand words whereas the event names has the fewest number of tagged words. As a result our system might be very poor at recognition of event names. But because of scarcity of dataset this is all we could get our hands on. And so we got going by applying different machine learning algorithms and build a prototype NER system.

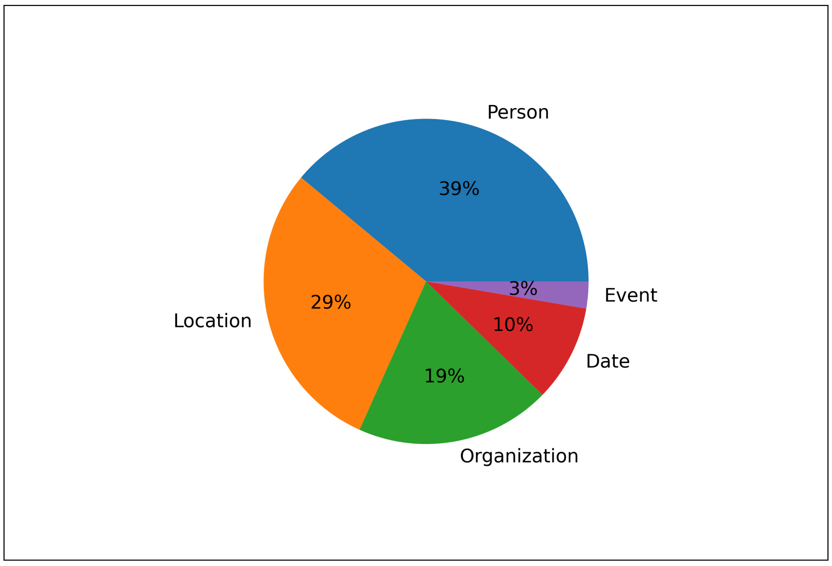

Figure 3.1: Pie-chart of tag distribution in training dataset.

3.3 Used Tagging Scheme

The used dataset is then tagged wih BIO tagging scheme. The tags are given in the table below with examples:

Tag	Meaning	Example
B-DATE	Beginning of a date entity	সোমবার, বুধবার
I-DATE	Inside of a date entity	B-DATE: ২১শে I-DATE: ফেব্রুয়ারি
B-EVE	Beginning of an event entity	ঈদ, রবীন্দ্রজয়ন্তী
I-EVE	Inside of an event entity	B-EVE: একুশে I-EVE: বইমেলা
B-LOC	Beginning of a location entity	ঢাকা, সিলেট
I-LOC	Inside of a location entity	B-LOC: বঙ্গবন্ধু I-LOC: এভিনিউ
B-ORG	Beginning of an organization entity	বিশ্বব্যাংক, জাতিসংঘ
I-ORG	Inside of an organization entity	B-ORG: সুইস I-ORG: ব্যাংক
B-PER	Beginning of a person entity	নজরুল, রবীন্দ্রনাথ
I-PER	Inside of a person entity	B-PER: মুহাম্মদ I-PER: জাফর I-PER: ইকবাল
O	Not a named entity	অনেকদিনের, মহানগরী

Table 3.2: Tags used in in training dataset with examples.

Chapter 4

Implementation and Result Analysis

This chapter describes the experiments we exploited on the previously mentioned dataset and discussion on the achieved result.

4.1 Experiment Discussion

For our experiment we have exploited three different deep learning architectures. At first we started working with BiLSTM architecture. Though the model didn't end up with good results but inability of this model for NER task gave us a good insight of the problem. To overcome the limitations of this architecture, we added a CRF layer on top of the BiLSTM model and this is our tried second approach towards solving the NER task. Though the second approach shown a relatively promising outcome and achieved a balanced F1-score of 82.8% but we kept searching for better solution for solving this task. In the mean time we studied some recent research works on NER task in different foreign languages like English, German etc. where we have seen that using pre-trained BERT architecture and fine-tuning afterwards they have gained state-of-the-art results in those languages. Then we decided to exploit BERT architecture on the dataset. We applied four different variants of pre-trained BERT and fine-tuned afterwards. After using transfer learning technique we have achieved state-of-the-art macro f1-score of 73.72% for Bengali NER task and also the architecture outperformed all previously experimented architectures of ours in all metric systems. The detailed explanation of our architectures and experiments are described in below sections.

We evaluated all our architectures using three metric systems. They are Precision, Recall and F1-Score. These metric systems use the following equations.

$$\text{Precision}_{\,i} = \frac{TP_i}{TP_i + FP_i} \tag{4.1}$$

$$\text{Recall}_{\,i} = \frac{TP_i}{TP_i + FN_i} \tag{4.2}$$

$$F1_i = \frac{2 \times \text{ Precision}_{\,i} \times \text{ Recall}_{\,i}}{\text{Precision}_{\,i} + \text{ Recall}_{\,i}} \tag{4.3}$$

4.1.1 BiLSTM

At first we trained our BiLSTM model on the dataset. The achieved result is not much promising in this approach. We have tried different hyper parameters while training the model and by using the hyper parameters described below we achieved best result among those trial training sessions.

- batch_size = 64
- epochs = 15
- max_sentence_length = 75
- dimension of embedding vector = 40
- hidden units = 50
- activation = relu
- optimizer = adam

For encoding we used One-Hot encoding method. The block diagram of the model is shown below.

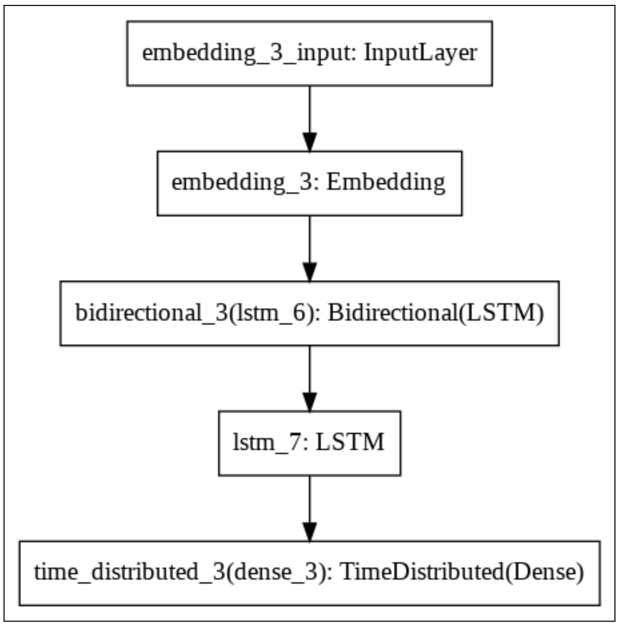
embedding_3_input: InputLayer
embedding_3: Embedding
bidirectional_3(lstm_6): Bidirectional(LSTM)
lstm_7: LSTM
time_distributed_3(dense_3): TimeDistributed(Dense)

Figure 4.1: Diagram of our BiLSTM model.

The achieved result is not that much good. The results are described below.

- Precision = 31.26%
- Recall = 22.84%
- F1-Score = 26.39%

As we can see that the result of our approach is not that much promising. Thus, we are not going to discuss the model elaborately.

4.1.2 BiLSTM-CRF

Our second experiment using BiLSTM-CRF architecture has shown relatively promising results. In this architecture, we kept the hyper parameters same as before, and yielded optimized results in the training process. We got balanced f1-score of 82.8% in this experiment.

The diagram of our BiLSTM-CRF model shown below.

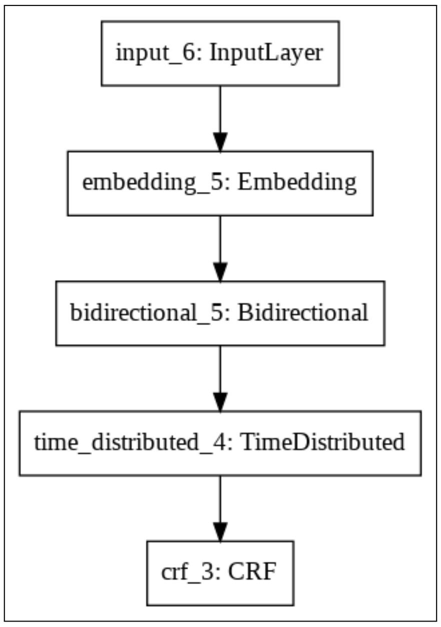
input_6: InputLayer
embedding_5: Embedding
bidirectional_5: Bidirectional
time_distributed_4: TimeDistributed
crf_3: CRF

Figure 4.2: Diagram of our BiLSTM-CRF model.

The learning curve of training process is shown below:

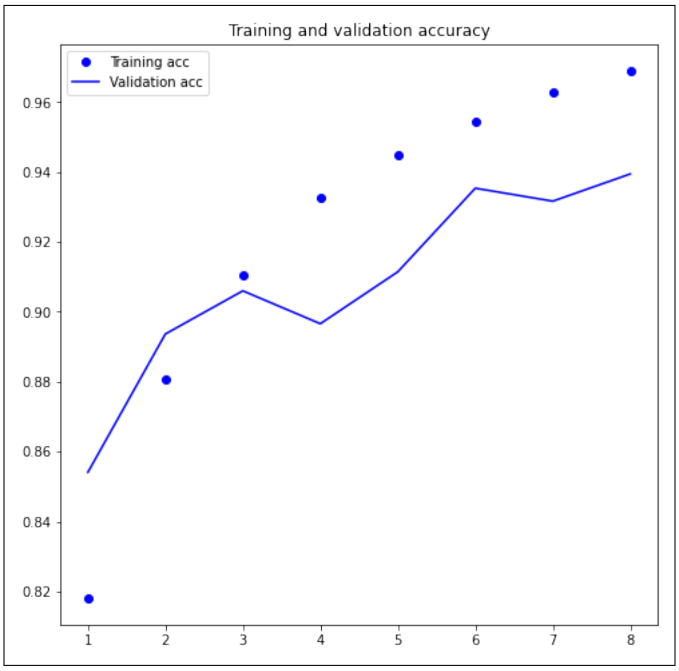

Figure 4.3: Learning curve of the BiLSTM-CRF model.

And the curve of training and validation loss is shown below:

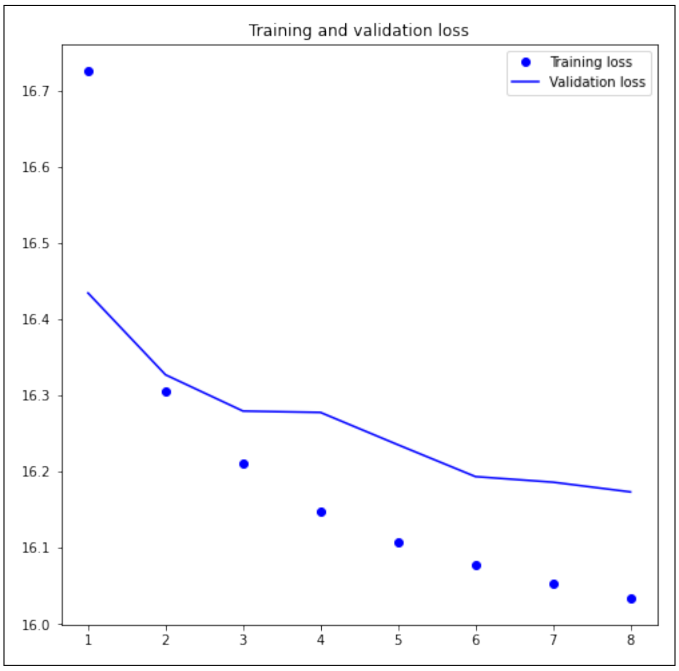

Figure 4.4: Training and validation loss curve of the BiLSTM and CRF model.

The table shown below illustrates the entity-wise result of the BiLSTM-CRF model:

Tag	Precision	Recall	F1-Score	Support
DATE	0.89	0.87	0.88	240
EVE	0.00	0.00	0.00	72
LOC	0.79	0.70	0.74	879
ORG	0.49	0.69	0.57	534
PER	0.90	0.84	0.87	953

Table 4.1: Entity-wise result of BiLSTM-CRF model

The table shown below describes the overall result of the BiLSTM-CRF model:

Metric	Precision	Recall	F1-Score
Accuracy	N/A	N/A	0.99
Macro Avg.	0.65	0.66	0.65
Micro Avg.	0.99	0.99	0.99

Table 4.2: Overall result summary of BiLSTM-CRF model

4.1.3 BERT

Even after getting good results using our BiLSTM-CRF model, we wanted better performance, such that it could be compared with human annotation. Later, on our endeavour for making a high performing NER system, we used BERT architecture and got balanced f1-score of 83.7% which is better than previously stated BiLSTM-CRF architecture. We have tried four different variant of pre-trained BERT and by trial and error method we chose optimized hyper-parameters during the fine-tuning process, that works best for NER tasks in Bengali Language. The chosen hyper-parameters are described below.

- batch_size = 32
- epochs = 14
- max_sentence_length = 75
- optimizer = AdamW
- learning rate = 3e-5
- adam's epsilon = 1e-8
- activation = softmax
- max_grad_norm = 1.0
- weight_decay_rate

The overall architecture of our BERT based NER system is shown below:

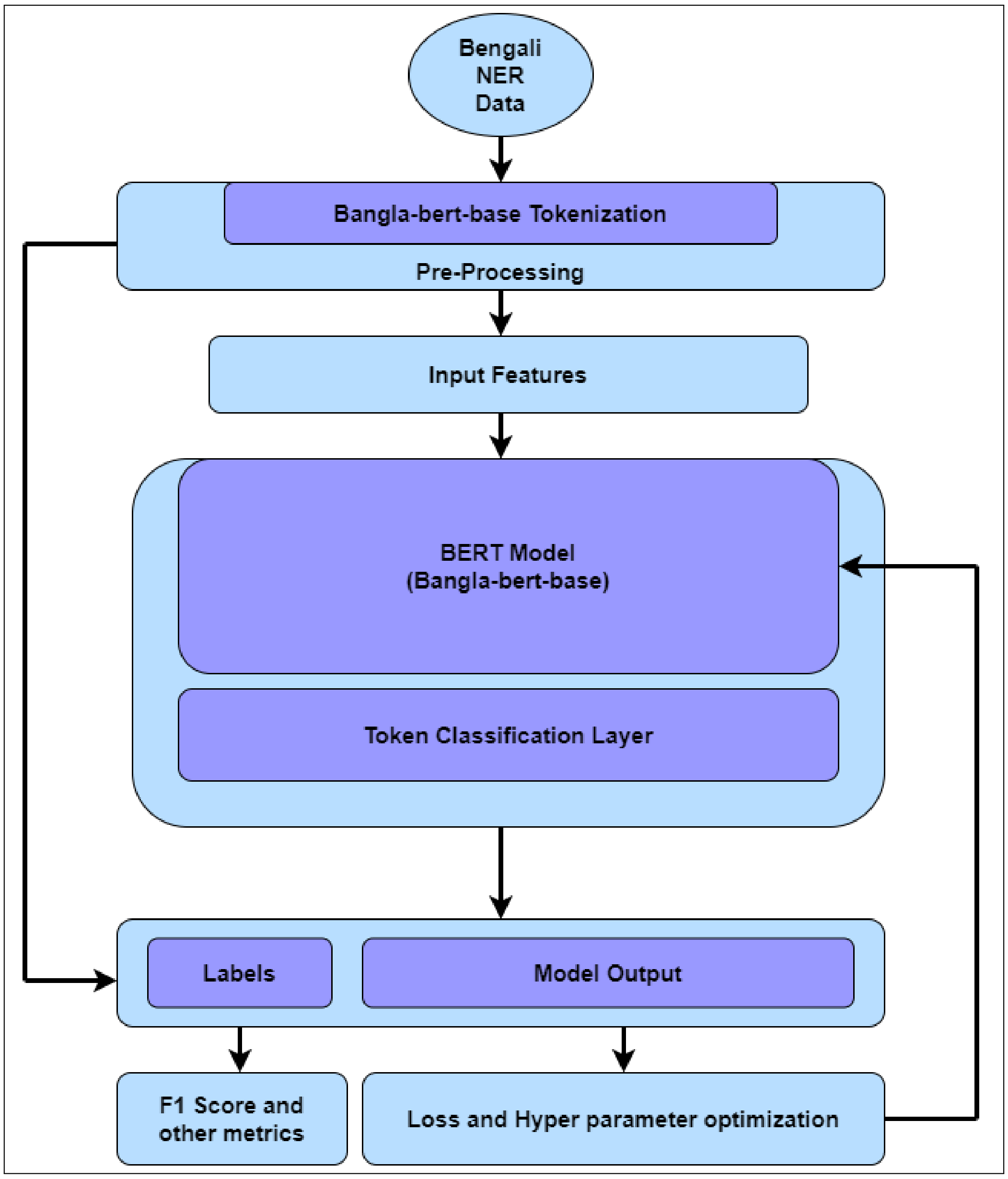
Bengali NER Data
Bangla-bert-base Tokenization
Pre-Processing
Input Features
BERT Model (Bangla-bert-base)
Token Classification Layer
Labels
Model Output
F1 Score and other metrics
Loss and Hyper parameter optimization

Figure 4.5: Architecture of our BERT based NER system.

The pre-trained model and the fine-tuned model has the same design. For pre-trained model we used four different variants of BERT and among them Bangla-bert-base[43] shown huge success. The block diagram of the model is illustrated below:

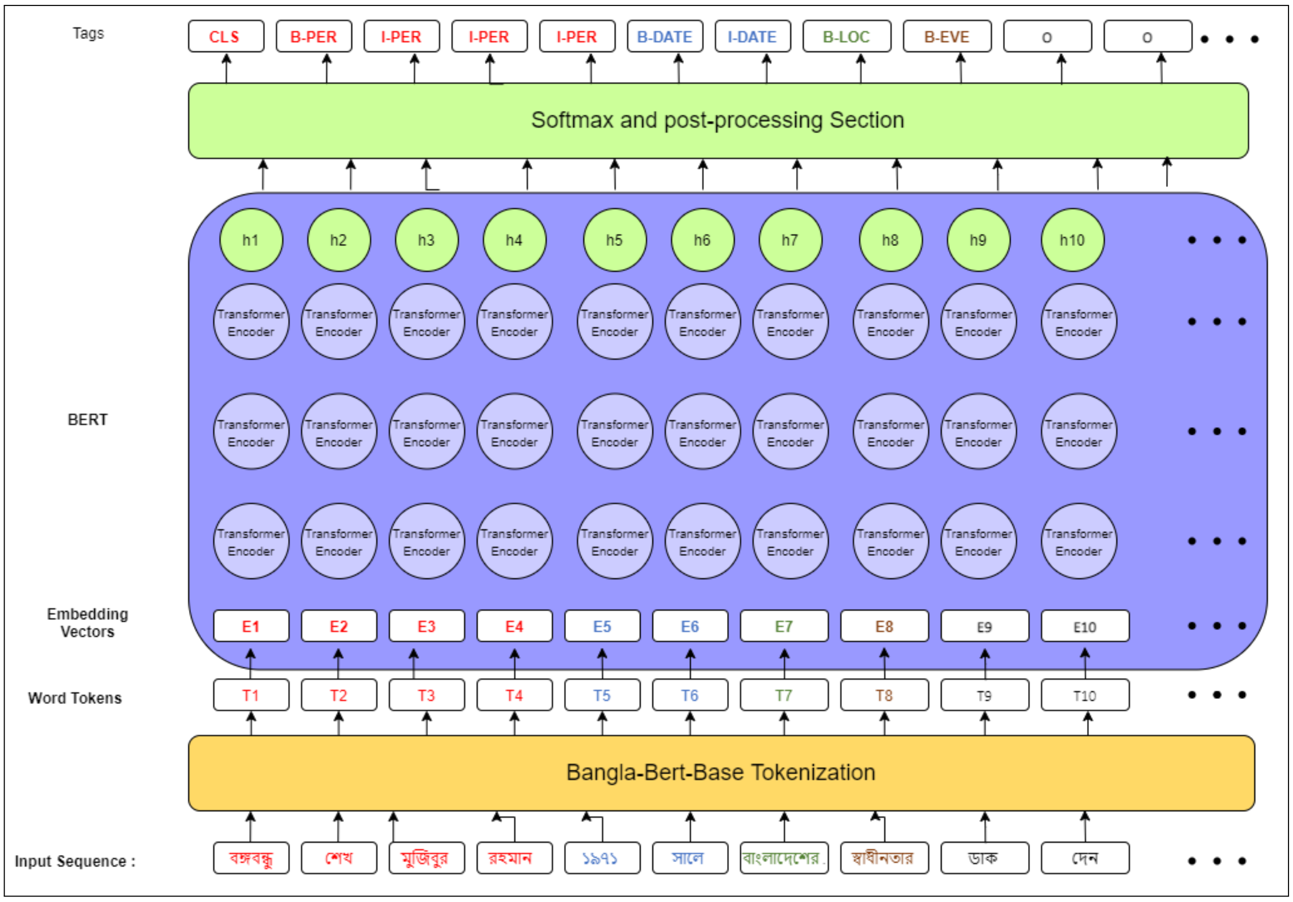

Figure 4.6: Diagram of our used BERT model.

For input representation we applied summation of position embedding, segment embedding and token embedding technique. The input representation process shown below:

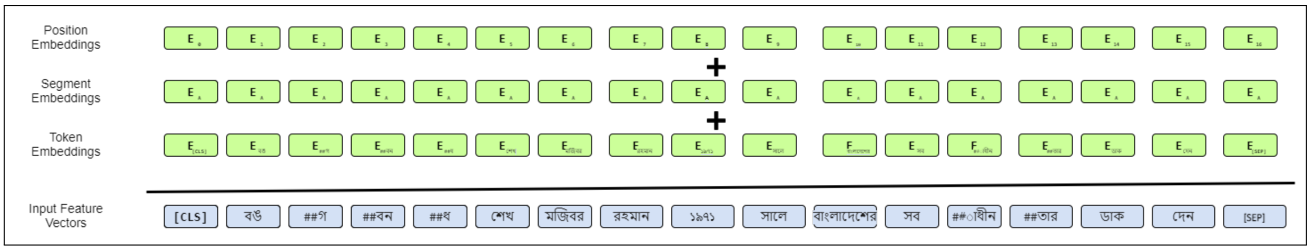

Figure 4.7: Input representation for the BERT based architecture.

We have applied four different pre-trained BERT model in our BERT based architecture on the dataset. They are:

- Bert-base-cased

- Bert-base-uncased
- Bert-base-multilingual-uncased
- Bert-bangla-base

Among these pre-trained models Bert-bangla-base proved to be victorious. There is a huge deviation between the results of Bert-bangla-base and other used pre-trained models. The result comparison is shown below:

Pre-trained model	Precision	Recall	F1-Score
Bert-base-cased	71.64	48.99	58.19
Bert-base-uncased	71.27	30.52	42.74
Bert-base-multilingual-uncased	63.71	34.46	44.73
Bert-bangla-base	82.82	82.48	82.65

Table 4.3: Performance of the BERT based architecture after using different variants of pre-trained model.

Here, we can see that Bert-bangla-based pre-trained model results way ahead of the other pre-trained models. So, naturally we used Bert-bangla-base pre-trained model finally for our architecture for Bengali NER task. So, from now on in this chapter we will discuss about our architecture which uses Bert-bert-base pre-trained model.
The validation accuracy curve of the architecture during training is shown below:

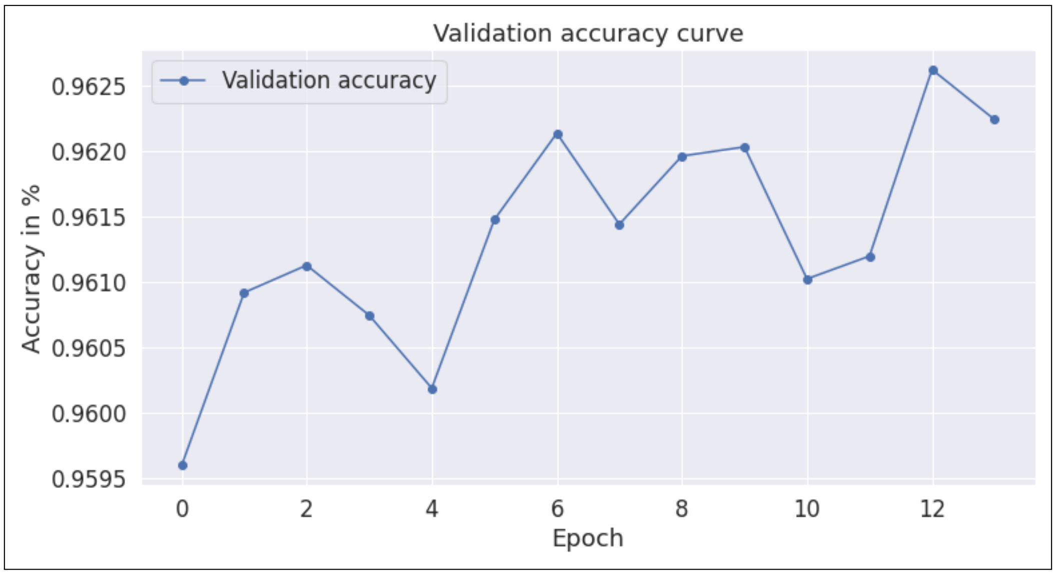

Figure 4.8: Validation accuracy curve of BERT based architecture.

And the training and validation loss during training is shown below:

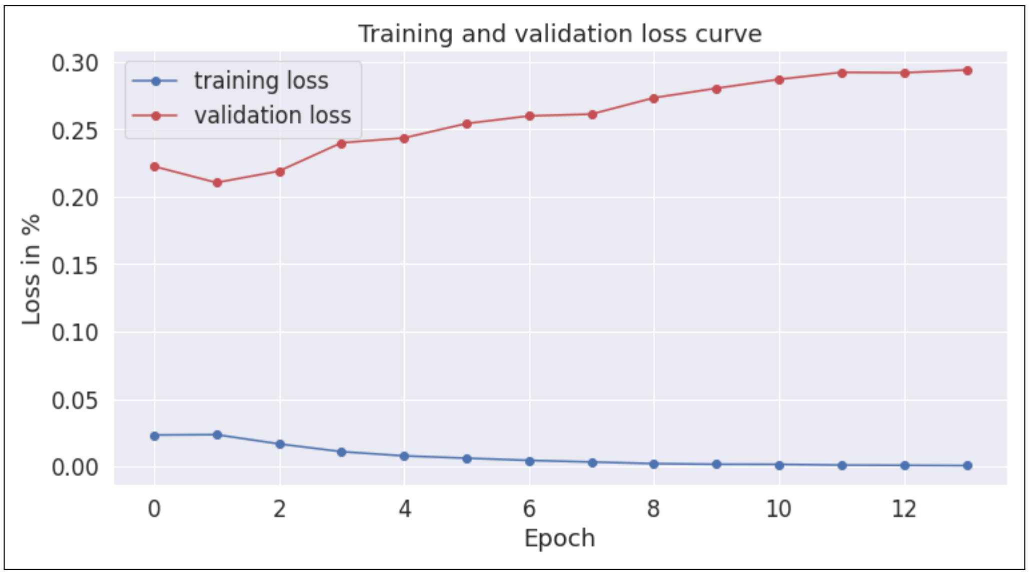

Figure 4.9: Training and validation loss curve of BERT based architecture.

The curve illustrating precision, recall and F-1 score is shown below:

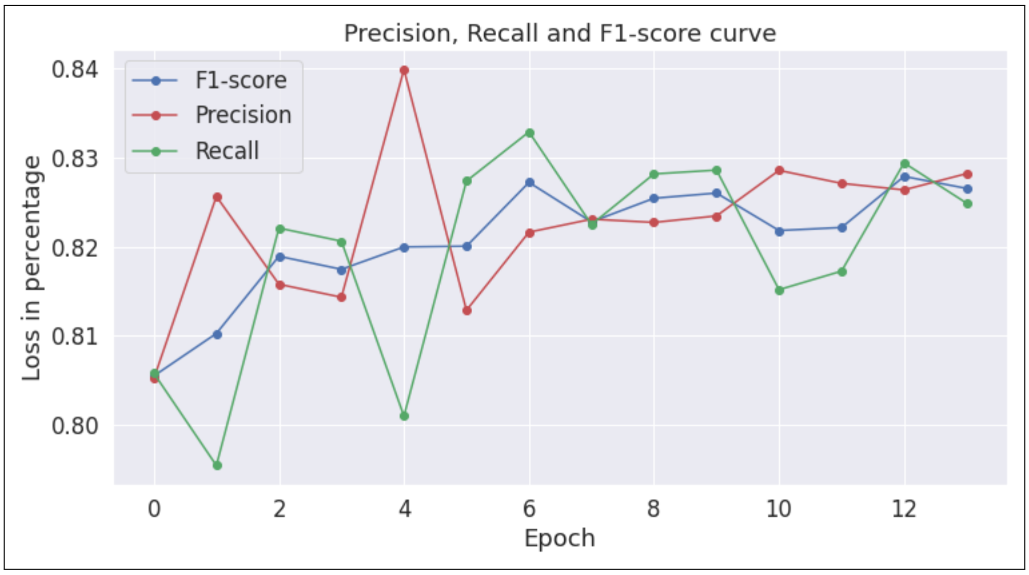

Figure 4.10: Precision, recall and f1-score curve of BERT based architecture.

The table shown below illustrates the entity-wise perfromance of our BERT based architecture.

Tag	Precision	Recall	F1-Score	Support
DATE	0.85	0.89	0.87	240
EVE	0.34	0.35	0.34	72
LOC	0.81	0.83	0.82	879
ORG	0.80	0.80	0.80	534
PER	0.90	0.90	0.90	953

Table 4.4: Entity-wise result of BERT based architecture.

And the table shown below describes the overall result of the BERT based architecture.

Metric	Precision	Recall	F1-Score
micro avg	0.83	0.84	0.84
macro avg	0.74	0.75	0.75
weighted avg	0.83	0.84	0.84

Table 4.5: Overall result summary of BERT based architecture.

4.2 Result Discussion

In this section we are going to compare and discuss about the results of our exploited architectures. For comparing the results we are going to use macro average precision, recall and f1-score as we found it is a standard way to compare the performance of different architectures for NER task. The below table shows a comparison between our architectures.

Architecture	Precision	Recall	F1-Score
BiLSTM	0.31	0.22	0.26
BiLSTM-CRF	0.65	0.66	0.65
BERT	**0.74**	**0.75**	**0.84**

Table 4.6: Comparison between our exploited architectures.

To calculate the macro average precision, recall and f1-score we used the equations described below.

$$\text{precision}_{\text{macro}} = \sum_{\text{classes}} \frac{\text{precision of class}}{\text{number of classes}} \tag{4.4}$$

$$\text{recall}_{\text{macro}} = \sum_{\text{classes}} \frac{\text{recall of class}}{\text{number of classes}} \tag{4.5}$$

$$F1_{\text{macro}} = \sum_{\text{classes}} \frac{\text{F1of class}}{\text{number of classes}} \tag{4.6}$$

Here, we can clearly conclude that BERT based architecture performs better than the other two. Despite being trained on a relatively smaller dataset it has shown great results. So, we can say that if we could train the model on a larger and more diverse dataset it would perform more accurately.

Chapter 5

Developing Our Own Dataset

5.1 Development of Our Dataset

Even after facing different issues at the initial stage, we started to work on making our own dataset once we had put together a promising deep neural network architecture. To do that we followed these steps:

1. Took a small part of a huge corpus prepared by SUST CSE Developer Network[44] which contains news articles from various sections of different online Bengali news portal.
2. Tokenized the articles from the corpus.
3. Used our best performing model to predict the NER tags of more than 4 lakh words.
4. Stored the predictions along with the respective token.
5. Verified and validated a fraction of our predicted dataset by using human annotators including ourselves.

Though it consists of just around 20 thousand tagged words. It helped us in two ways:

- Visualize the performance of our model by comparing it with human's NER tagging accuracy.
- The dataset created in the process could be merged with existing datasets and used for training models

Word	Prediction	Human Annotation
ব্রিসবেনের	O	B-LOC
বাংলাদেশ-অস্ট্রেলিয়ার	B-LOC	B-EVE
ম্যাচটি	O	I-EVE
মাশরাফির	I-PER	B-PER
অক্টোবরেই	O	B-DATE
ওয়েস্ট	O	B-ORG
ইন্ডিজকে	I-PER	I-ORG
রোববার	B-DATE	B-DATE
পুলিশ	O	B-ORG
জৈন্তাপুর	B-LOC	B-LOC
উপজেলার	I-LOC	O
লক্ষীপ্রাসাদ	I-LOC	B-LOC
গ্রামের	O	O
বিডিনিউজ	B-ORG	B-ORG
টোয়েন্টিফোর	B-LOC	I-ORG
ডটকমকে	I-ORG	I-ORG
চাঁপাইনবাবগঞ্জ	B-LOC	B-ORG
সদর	I-LOC	I-ORG
হাসপাতাল	I-LOC	I-ORG
২৮শে	B-DATE	B-DATE
সেপ্টেম্বর	I-DATE	I-DATE
এরশাদবিরোধী	O	B-EVE
আন্দোলনের	O	I-EVE
নূরুল	B-PER	B-PER
হুদা	I-PER	I-PER
বাবুল	B-PER	I-PER
ডিবির	O	B-ORG
হাওর	I-LOC	B-LOC
এলাকা	O	O

Table 5.1: System prediction and human validation comparison in our dataset.

We have shown a section of our dataset having the words and their respective predictions and human annotations in the table above.

5.2 Analysis of our Dataset

We compared the frequency of unique word - corresponding NER tag pair such as পুলিশ B-ORG,হোসেনে I-PER in prediction and human annotation. And we found out that only about 30 such pairs out of about 7 thousand unique pairs had a difference of over or equal to 10 between their frequencies in prediction and human annotation. This finding was very satisfying to us and solidified the fact the predictions were approaching human accuracy. We looked at those 30 word-NER tag pairs and saw that the হাসপাতালে B-ORG was mostly predicted to be a location. Similarly, we noticed that the names of many organizations started with the name of a location or person and were predicted as location or person instead of detecting them as organization. Besides we also found out that the model wrongly predicted many words' tags to be "O", we identified the reason as the over abundance of the tag in comparison with other tags, which resulted in a skewed learning. We also found out that the words tagged as B-DATE, B-PER were identified most easily and effectively by both the system and human annotators and so had the least difference in frequencies.

Besides, we also made a tag-wise frequency comparison of our model's prediction with human annotation. Then we found out the relative error using the formula:

$$\text{Relative Error}_{Tag} = 100 \times \frac{\left|\text{Frequency in Prediction}_{Tag} - \text{Frequency in Human Annotation}_{Tag}\right|}{\text{Frequency in Human Annotation}_{Tag}} \tag{5.1}$$

This relative error, though not an absolute measure, helped us visualize how well our model performed overall in comparison with human annotators in case of each tag. The horizontal bar diagram below depicts our analysis:

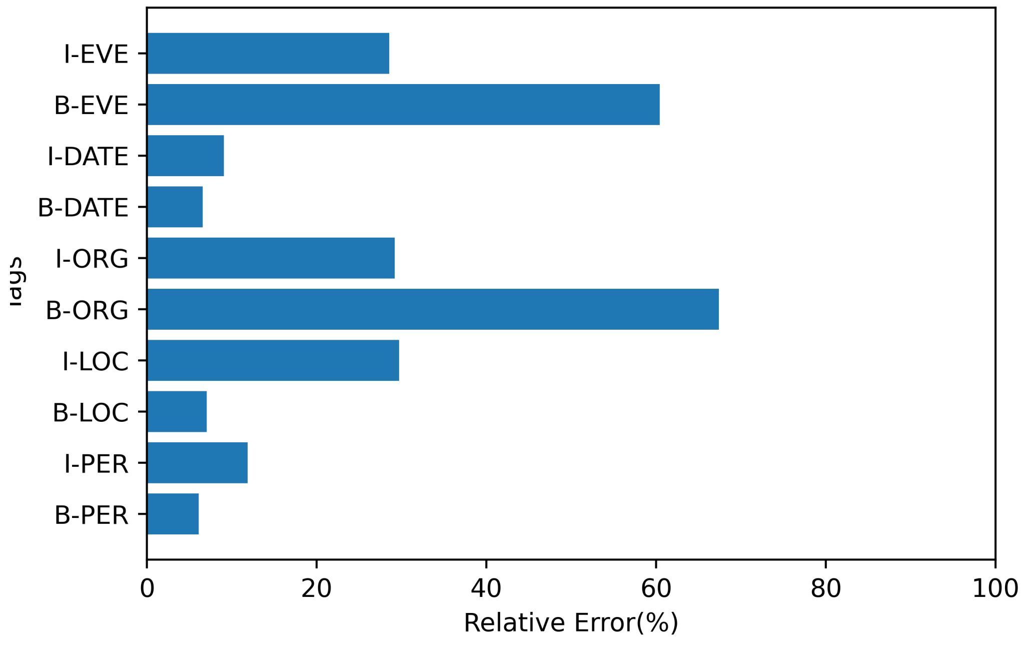

Figure 5.1: Relative Errors(%) in Prediction

From the bar diagram we can see that B-EVE and B-ORG tags had the maximum relative error. The reason behind the high relative error of B-ORG tag was due to the naturally confusing nature of the names of organizations, since organizations can be named after their location, persons which were detected as location and person by our model, which is why it affected the relative errors of I-ORG, B-LOC, I-LOC, B-PER, I-PER. And as for the B-EVE tag, there are very few B-EVE tags in the training dataset, and that is why our model could not learn to recognize this tag properly. A well balanced dataset having enough words for each tag would alleviate this problem.

We can clearly see that B-PER, B-LOC, B-DATE has the lowest relative errors, we cross checked with the word-tag pair frequency chart and found that they are recognized with the highest accuracy. We concluded that it was simply due to the simple and obvious nature of names of these types specially date which makes them easy to recognize.

And last but not least, we found out the frequency of the major tag groups(person, organization, location, date, event) and calculated the ratio of each of these tags in the dataset. The below image shows that ratio in the form of a pie-chart:

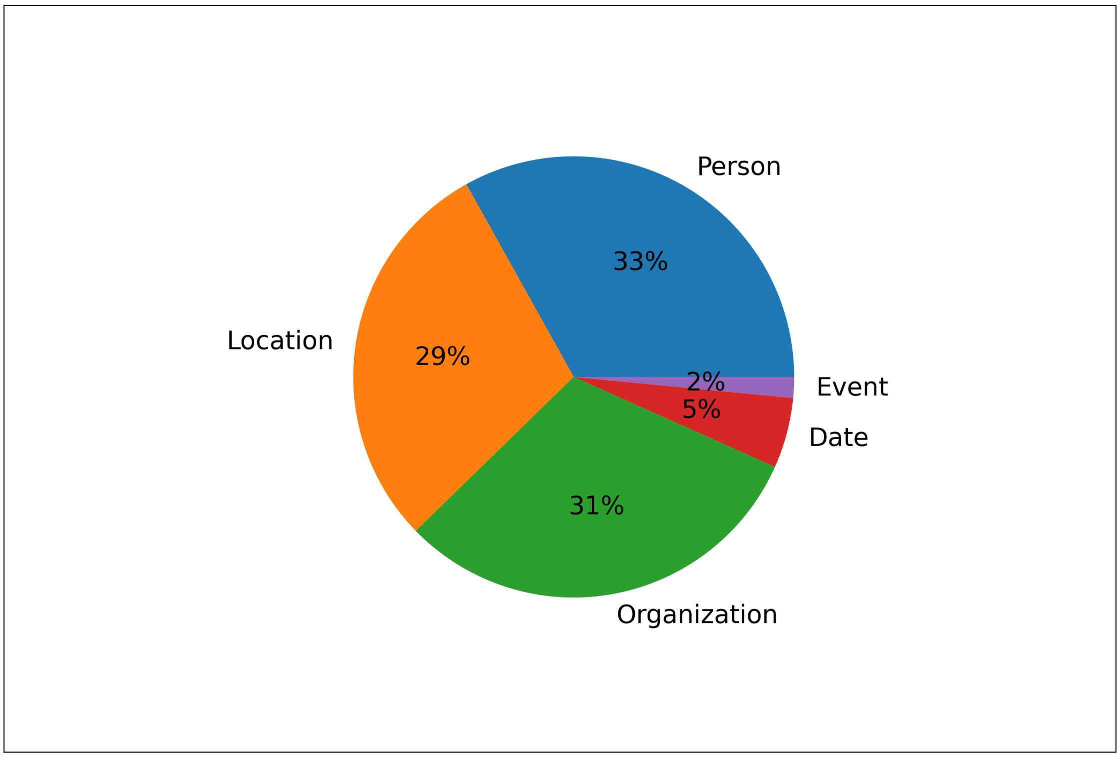

Figure 5.2: Percentage of tags in our dataset.

Chapter 6

Challenges

In the case of English and different European languages it is easier to recognize the named entities in comparison with Bengali language. Because every proper noun in English starts with a capital letter which makes it easier for detection.

Besides, language dependent features of English have been extensively worked on, so pre-trained models and online datasets are also more accessible. Extracting language dependent features of Bengali language is a daunting task.

Person names in Bengali is a lot more diverse where people are named after different birds, animals and many more diverse sources. There is no available name list for Bengali language like that of English.

The Bengali language is very versatile and highly modifiable. Named entities may be present at any position of the sentence due to its free order along with addition of different prepositions, prefix, suffix etc with the original word which makes them difficult to detect.

The tagging of NE's is highly dependent on the context of the sentence. An university can be considered as a location in one case while it will be considered as an organization in another case.

Moreover, the datasets available are not balanced, which causes the learning to be skewed. So the model only learns to recognize some tags properly, for example the PER tag(for person names) are recognized very well by the system but the EVE tags(for event names) are not properly recognized due to the small size of dataset.

The named entities can be both single-worded and multi-worded, which makes the recognition task harder and this is where the tagging scheme steps in.

Because of the context dependency of the NER tags even human annotators cannot tag the words with complete accuracy. As a result, building standard datasets using human annotators requires rigorous cross-checking and revision.

Probably, the biggest hurdle that we had to face is COVID-19. It affected our work in all sorts of ways. Our working rhythm got disrupted again and again. We had communication issues, network and device issues throughout the pandemic. We could not collect enough data for making a large dataset of our own.

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

Throughout our thesis we worked on building a high performing NER system for Bengali language. To achieve that, we applied different deep learning architectures that are currently being applied to build NER systems for English language. We read many research publications on NER systems and related fields and targeted some deep learning architectures to apply to our used dataset to build a highly accurate NER system for Bengali language. Among those architectures we exploited BiLSTM and BiLSTM-CRF in previous semester. Between these two, the BiLSTM-CRF architecture performs significantly better than the other one. In this semester we applied transfer learning technique by using pre-trained BERT for the task. We applied different variants of pre-trained BERT and among those Bert-bangla-base performed the best. After using the pre-trained model and fine-tuning the model we got very impressive results. The architecture produced macro f1-score of 72.73% which is state-of-the-art for Bengali NER. After training the model, NER tag predictions were near those of human annotated NER tags. We observed that the trained models worked really well on the tags that had more data in the dataset like B-PER, B-LOC etc. and had poor results in case of tags having the least data in the training dataset like B-EVE and I-EVE. Moreover, the number of tags in available datasets is still very low, which is why many named entities are missed by current NER systems. So we can say that training the model on a larger dataset having balanced number of tags would result in more precise NER system. And after training the BERT based architecture we used the trained architecture to predict sentences of ”Open Source

Bengali Corpus” in five NEs. After that we cross validated the predictions by human annotators and built a dataset of significant standards.

7.2 Future Scope

We strived for building a highly accurate NER system for Bengali language that can be used effectively in other NLP and IR tasks. By extensively studying many research papers and later applying different deep learning architectures we came to an understanding that our work can be expanded in the following directions:

- Increasing the number of tags: The number of tags in our current dataset is only 5. It would prove beneficial to increase that number and then fine-tune our model to recognize those classes of NE’s
- Increase the size of our developed dataset: We can use our NER system to predict the classes of NE’s in the sentences of balanced corpus, and can then be cross-validated using human validators same as our previously used technique. We can also increase the dataset for specific tags such as B-EVE, I-EVE which did not give satisfactory outcome due to the lack of labelled data. Then, that dataset can be used to train models to get better outcome.
- Increase additional information: Different research papers infer that adding additional information such as lexicographical features, dictionary etc can improve the recognition substantially.
- Going Unsupervised: Since building large datasets are both time and effort consuming and makes the models very domain specific, opting for unsupervised or semi-supervised architecture can expedite the advancements in NER for Bengali language even in the absence of large datasets.

.

References

[1] Wikipedia contributors, “Bengali language — Wikipedia, the free encyclopedia,” 2021, [Online; accessed 17-May-2021]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Bengali_languageoldid=1028227110

[2] J. Brownlee. (2017, 09) What is natural language processing? Accessed: 2020-04-23. [Online]. Available: https://machinelearningmastery.com/natural-language-processing/

[3] Wikipedia. (2020, 04) Named entity. Accessed: 2020-04-02. [Online]. Available: https://en.wikipedia.org/wiki/Named_entity

[4] V. N. Gudivada, “Chapter 12 - natural language core tasks and applications,” in *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*, ser. Handbook of Statistics, V. N. Gudivada and C. Rao, Eds. Elsevier, 2018, vol. 38, pp. 403–428. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169716118300257

[5] S. Kale and S. Govilkar, “Survey of named entity recognition techniques for various indian regional languages,” *International Journal of Computer Applications*, vol. 164, pp. 37–43, 04 2017.

[6] A. Ekbal and S. Bandyopadhyay, “Named entity recognition in bengali: A multi-engine approach,” *Northern European Journal of Language Technology*, vol. 1, pp. 26–58, 01 2009.

[7] V. N. Gudivada and K. Arbabifard, “Chapter 3 - open-source libraries, application frameworks, and workflow systems for nlp,” in *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*, ser. Handbook of Statistics,

V. N. Gudivada and C. Rao, Eds. Elsevier, 2018, vol. 38, pp. 31–50. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169716118300221

[8] V. N. Gudivada, D. Rao, and V. V. Raghavan, "Chapter 9 - big data driven natural language processing research and applications," in *Big Data Analytics*, ser. Handbook of Statistics, V. Govindaraju, V. V. Raghavan, and C. Rao, Eds. Elsevier, 2015, vol. 33, pp. 203–238. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444634924000095

[9] P. Sun, X. Yang, X. Zhao, and Z. Wang, "An overview of named entity recognition," in *2018 International Conference on Asian Language Processing (IALP)*, 2018, pp. 273–278.

[10] E. Team. (2020, 05) What is machine learning? a definition. Accessed: 2020-06-15. [Online]. Available: https://www.expert.ai/blog/machine-learning-definition/

[11] J. Brownlee. (2019, 08) What is deep learning? Accessed: 2021-06-10. [Online]. Available: https://machinelearningmastery.com/what-is-deep-learning/

[12] Mathworks. What is deep learning? 3 things you need to know. Accessed: 2021-06-10. [Online]. Available: https://www.mathworks.com/discovery/deep-learning.html

[13] TutorialsPoint. What are artificial neural networks (anns)? Accessed: 2021-06-11. [Online]. Available: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm

[14] Artificial neural network – applications, algorithms and examples.

[15] Phung and Rhee, "A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets," *Applied Sciences*, vol. 9, p. 4500, 10 2019.

[16] theappsolutions. How business can benefit from recurrent neural networks: 8 major applications. Accessed: 2021-06-12. [Online]. Available: https://theappsolutions.com/blog/development/recurrent-neural-networks/

[17] A. Mohan and D. Gaitonde, "A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks," 04 2018.

[18] paperswithcode. Bidirectional lstm. Accessed: 2021-06-12. [Online]. Available: https://paperswithcode.com/method/bilstm

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

[20] Maxime. (2019, 01) What is a transformer? Accessed: 2021-06-12. [Online]. Available: https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[22] Y. Jernite, S. R. Bowman, and D. Sontag, "Discourse-based objectives for fast unsupervised sentence representation learning," *ArXiv*, vol. abs/1705.00557, 2017.

[23] R. Grishman and B. Sundheim, "Message understanding conference-6: A brief history," in *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, ser. COLING '96. USA: Association for Computational Linguistics, 1996, p. 466–471. [Online]. Available: https://doi.org/10.3115/992628.992709

[24] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. 76, pp. 2493–2537, 2011. [Online]. Available: http://jmlr.org/papers/v12/collobert11a.html

[25] M. Collins and Y. Singer, "Unsupervised models for named entity classification," in *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999. [Online]. Available: https://www.aclweb.org/anthology/W99-0613

[26] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Unsupervised named-entity extraction from the web: An experimental study," *Artificial Intelligence*, vol. 165, no. 1, pp. 91–134, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370205000366

[27] D. Nadeau, P. D. Turney, and S. Matwin, “Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity,” in *Proceedings of the 19th International Conference on Advances in Artificial Intelligence: Canadian Society for Computational Studies of Intelligence*, ser. AI’06. Berlin, Heidelberg: Springer-Verlag, 2006, p. 266–277. [Online]. Available: https://doi.org/10.1007/11766247_23

[28] S. Zhang and N. Elhadad, “Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts,” *Journal of Biomedical Informatics*, vol. 46, no. 6, pp. 1088–1098, 2013, special Section: Social Media Environments. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046413001196

[29] Y. Shao, C. Hardmeier, and J. Nivre, “Multilingual named entity recognition using hybrid neural networks,” 2016. [Online]. Available: http://www8.cs.umu.se/ johanna/sltc2016/abstracts/SLTC_2016_paper_21.pdf

[30] A. Sehanobish and C. H. Song, “Using chinese glyphs for named entity recognition,” *CoRR*, vol. abs/1909.09922, 2019. [Online]. Available: http://arxiv.org/abs/1909.09922

[31] A. Ekbal and S. Bandyopadhyay, “A hidden markov model based named entity recognition system: Bengali and hindi as case studies,” 12 2007, pp. 545–552.

[32] A. Ekbal, R. Haque, and S. Bandyopadhyay, “Named entity recognition in Bengali: A conditional random field approach,” in *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*, 2008. [Online]. Available: https://www.aclweb.org/anthology/I08-2077

[33] A. Ekbal and S. Bandyopadhyay, “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*, 2008. [Online]. Available: https://www.aclweb.org/anthology/I08-5008

[34] M. Hasanuzzaman, A. Ekbal, and S. Bandyopadhyay, “Maximum entropy approach for named entity recognition in bengali and hindi,” *International Journal of Recent Trends in Engineering*, vol. 1, 04 2009.

[35] A. Ekbal and S. Bandyopadhyay, “Named entity recognition in bengali: A multi-engine approach,” *Northern European Journal of Language Technology*, vol. 1, pp. 26–58, 01 2009.

[36] S. Parvez, “Named entity recognition from bengali newspaper data,” *International Journal on Natural Language Computing*, vol. 6, pp. 47–56, 06 2017.

[37] S. Chowdhury, F. Alam, and N. Khan, “Towards bangla named entity recognition,” 12 2018.

[38] N. Ibtehaz and A. Satter, “A partial string matching approach for named entity recognition in unstructured bengali data,” *International Journal of Modern Education and Computer Science*, vol. 10, pp. 36–45, 01 2018.

[39] M. J. Rifat, S. Abujar, S. Noori, and S. Hossain, “Bengali named entity recognition: A survey with deep learning benchmark,” 07 2019, pp. 1–5.

[40] J. Saurav, “End to end parts of speech tagging and named entity recognition in bangla language,” 09 2019.

[41] 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. 58 206–58 226, 2020.

[42] S. R. Sarowar Hossain, *A Web API on Bengali Named Entity Recognition*, 02 2020.

[43] S. Sarker, “Banglabert: Bengali mask language model for bengali language understading,” 2020. [Online]. Available: https://github.com/sagorbrur/bangla-bert

[44] S. CSE, “Open source bengali corpus,” 2019. [Online]. Available: http://184.154.80.146/scdn/corpus/?fbclid=IwAR2M6udXrzohsZK0amqc5K_OW3A9MeGszzyj