Research on the latest trends in Bangla Named Entity Recognition

by

Niloy Farhan
23341028
Saman Sarker Joy
20101114
Tafseer Binte Mannan
20101256

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
School of Data and Sciences
BRAC University
May 2023

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Niloy Farhan
23341028

Saman Sarker Joy
20101114

Tafseer Binte Mannan
20101256

Approval

The thesis/project titled "Research on the latest trends in Bangla Named Entity Recognition" submitted by

1. Niloy Farhan(23341028)
2. Saman Sarker Joy(20101114)
3. Tafseer Binte Mannan(20101256)

Of Spring, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 2023.

Examining Committee:

Supervisor:
(Member)

Farig Yousuf Sadeque

Dr. Farig Yousuf Sadeque
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Ethics Statement

We hereby declare that this thesis is based on our own findings from our own research discoveries. All other sources used in this work have been properly acknowledged. Furthermore, we confirm that this thesis has not been submitted or presented, either in its entirety or partially for the purpose of receiving a degree from any other educational institution or university.

Abstract

Named Entity Recognition (NER) is a sub-task of Natural Language Processing (NLP) that distinguishes entities from unorganized text into predefined categorization. In recent years, a lot of Bangla NLP subtasks have received quite a lot of attention; but Named Entity Recognition in Bangla still lags behind. In this thesis, we explored the existing state of research in Bangla Named Entity Recognition. We tried to figure out the limitations that current techniques and datasets face, and we would like to address these limitations in our research. Additionally, we developed a Gazetteer that has the ability to significantly boost the performance of NER. We also proposed a new NER solution by taking advantage of state-of-the-art NLP tools that outperform conventional techniques.

Keywords: NER; NLP; Transformers; BERT; Gazetteer;

Dedication

We want to dedicate all of our sacrifices and educational efforts to our great parents, without whom we would be worthless. We also dedicate our thesis to Dr. Farig Yousuf Sadeque sir, who served as our supervisor, guided us, and showed us how to develop our skills and personalities as successful professionals.

Acknowledgement

We extend our gratitude to the Great Almighty for guiding us through the completion of our thesis without any major setbacks. Our appreciation also goes to our supervisor, Dr. Farig Yousuf Sadeque, for his invaluable support and guidance throughout the process. Lastly, we are grateful for the unwavering support of our parents, without whom this achievement would not have been possible.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

$BERT$ Bidirectional Encoder Representations from Transformers

$Bi-LSTM$ Bidirectional LSTM

CNN Convolutional Neural Network

CRF Conditional Random Fields

DL Deep Learning

$ELECTRA$ Efficiently Learning an Encoder that Classifies Token Replacements Accurately

F_1 Balanced F-Score

GRU Gated Recurrent Unit

$LSTM$ Long Short-Term Memory

$MEMM$ Maximum-Entropy Markov Model

ML Machine Learning

NE Named Entity

NER Named Entity Recognition

NLP Natural Language Processing

Chapter 1

Introduction

1.1 Named Entity Recognition in Bangla

Named Entity Recognition (NER) is a significant initial step in Natural Language Processing (NLP) tasks. It has been widely used in machine translation, information extraction, automatic question-answering, natural language understanding, text-to-speech synthesis, etc. NER identifies text segments that make up proper nouns and tags the entities from unstructured text into a predefined class. Generally, the most common entity tags are: the name of a person (PER), the name of a company or organization (ORG), the name of a place (LOC), and the geo-political entity (GPE). But this can be reduced or extended based on the annotated corpus. An example of Bangla NER is shown in figure 1.1.

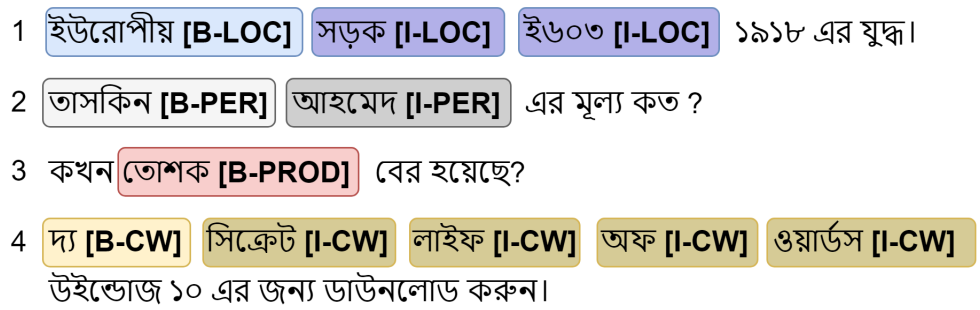


Figure 1.1: NER Example in Bangla

NER systems can use a Rule-based approach, Machine Learning (ML) or Deep Learning (DL) approach, or using Hybrid approach. These are the only guidelines that a rule-based NLP system needs to classify the language and to analyze it [1]. This method can be used for a particular language depending on its grammatical structure. Usually, it is done by using RegEx. RegEx, or regular expression, is a tool for finding matches among characters in a string. This property to detect character combinations is very vigorous. Compared to machine learning techniques, this type of NE (Named Entitiy) Rule Base approach exhibits greater accuracy. But this is hard to maintain as it is language dependent and the set of rules can be huge and complicated. Furthermore, the use of ML and DL is more popular in NER systems as these are easy to train and less expensive. A hybrid NER system is made by merging the rule-based method and the statistical ML or DL methods.

The seventh most popular language in the world is Bangla [2]. But very few notable works have been done in the field of NLP in this language compared to English. We examined the current state of named entity recognition research in Bangla in this thesis. To address these limits in our research, we had investigated the limitations that current approaches and datasets confront. In our research, we had discussed the poor performance of typical deep-learning methods and came up with the idea of a knowledge-based approach. We created Gazetteer which is a set of entity lists of each tag that can be used as an external knowledge source for neural network models which we made publicly available. Our Gazetteer showed promising improvement in the NER classification task. Furthermore, we have experimented with BanglaBERT models and tried to improve them with our custom weighted cross entropy loss function which we have discussed in detail in the upcoming section. Finally, we experimented with different models with different features and strategies and based on our findings, we have proposed a completely new NER solution which is a CRF model with Gazetteer and BanglaBERT embeddings that achieved 0.8267 Macro F_1 score in MultiCoNER 1 dataset, significantly outperforming conventional techniques.

1.2 Problem Statement

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 NLP community, despite being the 7th most widely spoken language across the world, Bangla is still now considered as a low-resource language.

Firstly, the Bangla language does not have a sufficient amount of Named Entity annotated corpus compared to English and other languages. At the time of our research, the existing datasets were mostly extracted from Bangla online newspapers. But these were not enough to represent the vast diversity of the Bangla Language. Another problem that researchers faced while working with Bangla Language is the lack of special features such as Capitalization. Most of the work in Bangla NER has been done using Machine Learning models such as Support Vector Classifier, Maximum Entropy Markov Model (MEMM), CRF (Conditional Random Field), and Deep Learning Models such as Bi-directional LSTM (Long Short-Term Memory), GRU (Gated recurrent unit), CNN (Convolutional Neural Network). As the output tag of NER can have different numbers of entities based on the corpus and task, there was a lack of a general NER system with very high accuracy.

In light of this, the concern might be: whether we could develop a cutting-edge system for Bangla NER. Yes, there has been a great deal of research done in this field in the English language. We, therefore, thought that we could apply the latest NLP methodologies in Bangla.

Hence, the concern this research is attempting to address is:

> *Can we create a reliable Bangla NER System using the state-of-the-art NLP techniques and deep learning model?*

By evaluating the status of Bangla Natural Language Processing, this research provided an answer to the aforementioned query.

1.3 Research Objectives

As the NER plays a vital role in information extraction, machine translation, etc., a reliable system is much needed for Bangla Language. Therefore, our research was aimed to develop a system for Bangla NER which should be able to provide the highest score accuracy and F_1 score by implementing and experimenting with state-of-the-art NLP techniques and models of deep learning such as Transformer Models, etc.

1. Achieve a thorough understanding of Natural Language Processing and its functioning as well as expertise in latest Machine Learning and Deep Learning Techniques.
2. Understand the current drawbacks of NER models and provide solutions.
3. Create a knowledge base that can be used by different models.
4. Develop a model for recognizing named entities in Bangla and evaluate the performance of the model.
5. Provide suggestions for enhancing the performance of the model.

Chapter 2

Literature Review

NER, or Name Entity Recognition, is a subtask within Natural Language Processing (NLP) that is responsible for identifying and extracting specific entities. Over many years lots of research work has been done with NER in different languages. Recently, several supervised learning-based models and rule-based models were proposed.

In the paper [3], the authors have experimented with the combination of three approaches: Dictionary-based NE detection, Rule-based NE detection, and n-gram-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% F_1 score.

In another study [4], 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. This rule base approach has been merged with the MEMM which is helping to improve the score. 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 F_1 score of 68.98% from the MEMM model alone and an F_1 score of 71.59% from the combined model.

In this paper [5], 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 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 F_1 score of 63.37%.

In their research [6], the authors focused on POS tagging and Name Entity Recog-

nition 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 (BLSTM), Convolutional Neural Network (CNN), Conditional Random Field (CRF), and LSTM. They achieved a 0.6285 F_1 score with their model in the NER task.

According to the author of the [7] 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. [8] where 72000 Bangla sentences were annotated. Their highest score was achieved with BERT + BiLSTM + CRF + CW with a macro averaged F_1 score of 65.96% and micro-averaged F_1 score of 90.66%.

There are some notable works done in order to build NER annotated corpus.

The [9] paper is dedicated to a corpus annotated with seven named entities (Person, Location, Organization, Facility, Time, Units, Misc.) with a discreet attention on Bangladeshi Bangla. They have worked on baseline results as well. They collected newspaper stories from several newspapers and annotated them for the dataset.

In the [10] research paper, they worked with a large curated Bangla Articles Dataset and many supervised learning models were proposed for analyzing these data. According to them, there is a very a smaller number of curated Bangla datasets so they themselves curated a huge dataset from Bangla articles from different news sites containing 3,76,226 articles. Applying to pre-process (removal of punctuations, stopwords, etc.) they got a set of words frequency. TF-IDF and Word2Vec approach has been used in this paper. Although this is a large dataset, it is not annotated for the NER task.

From the above discussion, it is observed that in the above research papers mostly deep learning models like Long Short-Term Memory (LSTM), and Bi-directional Long short-term memory (BLSTM) have been used for Bangla name entity recognition. And not every model could give perform reliably enough performance like English, Chinese or other languages. As Bangla NER is a very unique work, the technique should not be limited. Recently transformer models of deep learning have come to light, such new technology should be implemented for a better and more accurate score.

In the [11] research paper, the author proposed a knowledge based system. They were working with the MultiCoNER 1 dataset which is similar to us. They built a multilingual knowledge base which works on Wikipedia. The goal of this base is to provide related context information to the NER model. The system retrieves related

context from the knowledge base when an input sentence is given and augments the context information with the original sentence. This approach wins 10 out of 13 categories in the MultiCoNER task. They were able to achieve 0.8351 Macro F_1 score in the test data.

The [12] 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 pretraining the models. They pretrained BanglaBERT using ELECTRA. For zero-shot cross-lingual transfer, their Banglish-BERT achieved 55.56 Micro F_1 score and supervised fine-tuning, their BanglaBERT model gained 77.78% Micro F_1 score in MultiCONER dataset.

In another study [13], 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 efficacy 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% F_1 score.

Chapter 3

Description of the Data

3.1 MultiCoNER Dataset

The dataset we decided to use in this thesis is MultiCoNER [14]. The MultiCoNER dataset we have utilized, is a comprehensive multilingual dataset for Named Entity Recognition. This dataset comprises 3 different domains (encyclopedia sentences, QA questions, and Web queries) across 11 languages. It has a large amount of data collected from sources such as Wikipedia, queries and questionnaires. The 11 languages included in the dataset range from well-resourced languages like English to those with limited resources like Farsi, in order to represent a diverse range of languages and writing systems. MultiCoNER includes 13 distinct subsets, consisting of 11 monolingual subsets for the aforementioned languages, a multilingual subset (MULTI) and a code-mixed subset (MIX). The subsets were categorized into Monolingual, Multilingual and Code-mixed.

The MultiCoNER dataset includes 11 languages such as 1. Bangla (BN), 2. Hindi (HI), 3. German (DE), 4. Chinese (ZH), 5. Korean (KO), 6. Turkish (TR), 7. Dutch (NL), 8. Russian (RU), 9. Farsi (FA), 10. English (EN), 11 Spanish (ES). Long-tail entity distributions, syntactically complicated entities like movie titles, low-context scenarios (brief, uncased text), and low-context scenarios are just a few of the issues that the dataset is intended to convey. Additionally, this dataset is considered particularly significant and relevant as it is the first time, a multilingual dataset for Named Entity Recognition has been created for Bangla. The dataset is complex and aimed at advanced Named Entity Recognition.

MultiCoNER was built on the WNUT 2017 [15] entity type taxonomy, which enables the identification of a wide range of entities, including those with more complex structures such as creative works. The NER tag-set used in this dataset includes six classes:

1. PER: Names of individuals
2. LOC: Places and physical facilities
3. CORP: Corporations and businesses
4. GRP: Additional group types

5. PROD: Consumption Products.

6. CW: Creative works titles like movies, songs & books.

MultiCoNER dataset used BIO tags which are a widely used annotation scheme for labeling words in a sentence as a named entity or not. The BIO notation stands for ”Begin, Inside, Outside”. The initial word of a named entity is identified as ”B-tag”, where tag is the named entity class, and the subsequent words within the same named entity are labeled as ”I-tag”. Any other words that are not part of a named entity are labeled as ”O”. For example, if we have a sentence ”Barack Obama is the President of the United States.” and we want to extract ”Barack Obama” as ”PER” (person) named entity and ”United States” as ”LOC” (location) named entity, the annotation would be ”B-PER, I-PER, O, O, O, O, B-LOC”. This scheme is useful because it allows the model to not only identify when a named entity starts, but also when it ends. The tag of MultiCoNER is illustrated in figure 3.1

BIO Tag	NER Tag
B = Begin **I** = Inside **O** = Outside	**PER** = Names of People **LOC** = Location or Physical Facilities **CORP** = Corporations and Businesses **GRP** = All Other Groups **PROD** = Consumer Products **CW** = Titles of Creative Works like Movie, Song and Book

Figure 3.1: BIO Tags

3.1.1 Formation of MultiCoNER's Bangla Dataset

The MultiCoNER dataset comprised eleven different languages, and three distinct domains (information database sentences, questions from Question Answerings, and internet inqueries). The dataset is constructed by three different subsets:

LOWNER: This subset is composed of encyclopedic sentences extracted from various localized versions of Wikipedia. Using Wikidata as a base, sentences were chosen with low context and entities were linked to the appropriate entity types, according to the Named Entity Recognition class taxonomy. A manual look-over of four hundred sampled EN sentences revealed that the Named Entity Recognition gold labels are quite accurate in 94% of the cases.

MSQ-NER: This subset is created from the MS-MARCO Question and Answer corpus [16] by extracting question impressions, putting the entities with their NER variety (MultiCoNER NER taxonomy) and translating them into other languages. Entities in the questions are identified using spaCy.

ORCAS-NER: Similarly, this subset is created by extracting templates from Web user queries from the ORCAS dataset [17] and translating them into different languages. Instances are generated from each template by replacing the designated spots with actual named entities in the target languages, creating multiple variations of the original template.

To create the dataset of Bangla, Google Translation API and other automatic translation techniques were used. The total number of Bangla data in MultiCoNER is 133,119, where LOWNER is 15,698, ORCAS-NER is 100,000 and MSQ-NER is 17,421. Translation quality for languages such as Bangla, Chinese, German and Hindi in LOWNER, ORCASNER, and MSQ-NER is exceptional, boasting an accuracy rate of over 90%. Contribution of each domain is presented in the table 3.1

lang	**LOWNER**	**ORCAS-NER**	**MSQ-NER**
BN	15,698	100,000	17,421

Table 3.1: Contribution of Each Domain

3.1.2 Dataset Statistics

To corroborate that the Named Entity Recognition results obtained from this dataset are consistent and can be replicated, it has been divided into three prearranged sets for training, development, and testing. 15,300 training and 800 development data are instances available for each language. The vast majority of instances in the training splits are from the Wikipedia domain (LOWNER), while only 100 instances—50 each from the domains of Web Questions (MSQ-NER) and Web Query (i.ORCAS-NER) - represent domain-adaptation data.

The test splits are substantially bigger. According to MultiCoNER's authors, this is done primarily for two reasons:

1. To evaluate how well the NER models works on unknown and complicated words

2. To evaluate the effectiveness of NER models for cross-domain adaptation

3.1.3 Bangla (bn) Dataset Statistics

Our work focuses solely on the Bangla language. We aimed to work with Bangla NER, hence we have explained the Bangla language part among all the 11 languages included in this MultiCoNER dataset.

There are a total of 149, 219 sentences in the whole BN split of the MultiCoNER dataset. This BN data is separated into three different sets for training, development, and testing purposes.

For the training data split, there are 15,300 sentences (10.25%). To evaluate model generalizability, a random sample of 800 (0.53%) sentences per subset was selected from the LOWNER domain and used as the Development data split. The Testing data set comprised the remaining instances that were not part of the Training or Development data. It has a total of 133,119 sentences (89.21%). As previously stated, the training data was fairly large because it should help us to evaluate how well our NER models work on unknown and complicated words. The class distribution of train, test and dev of Bangla data is presented on table 3.2 and percentage of each class in training and test data is visualized in figure 3.2.

class	PER	LOC	GRP	CORP	CW	PROD	#TOTAL
TRAIN	2,606	2,351	2,405	2,598	2,157	3,188	15,300
DEV	144	101	118	127	120	190	800
TEST	24,601	29,628	19,177	20,066	21,280	20,878	133,119

Table 3.2: Class Distribution of Train Test and Dev of Bangla Data

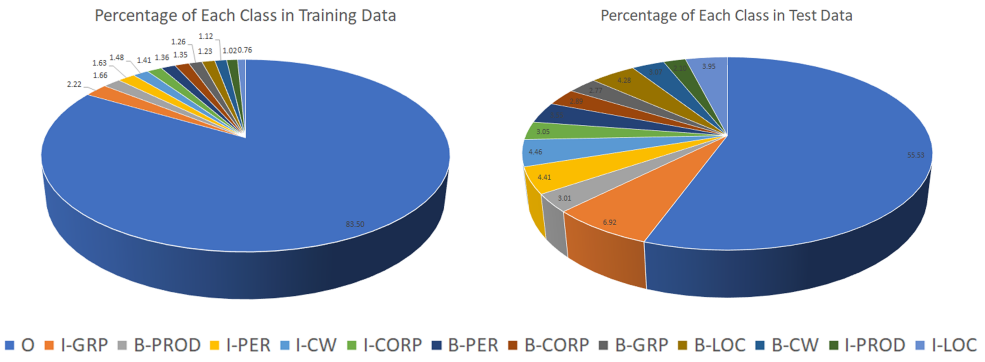


Figure 3.2: Percentage of Each Class in Training and Test Data

3.2 Supplementary Dataset

3.2.1 Gazetteers

The word gazetteer means a geographical dictionary or directory used in conjunction with a map or atlas. However, in NER research area, the word "gazetteer" is used interchangeably for both the set of entity lists and for the processing resource that uses those lists to find occurrences of named entities in texts.

There has been many research on this Gazetteer technique and it has shown promising improvements to tasks like NER. From the paper [18], we can see that the implementation of gazetteer has an impact on the overall performance of their NER system. Without it they were getting a F_1 score of 0.771 in CRF Models, however, with the help of gazetteers it was boosted to 0.816.

3.2.2 Difficulties of Forming Gazetteers in Bangla

It is a huge challenge to form an entire set of gazetteers in Bangla language. The information is mostly incomplete and not updated properly. It can be difficult and time-consuming to gather correct and current information for a gazetteer. It's also difficult to create a thorough and precise gazetteer in Bangla because some regions are off-limits or have scant information available. We know there are many areas in Bangladesh that may have local dialects which are tough to understand and gathered for proper documentation. Additionally, it can be difficult to translate information from regional dialects into Bangla, particularly if there are no accepted spellings or pronunciations. Also the resources in the Bangla language are very limited. Some words are found easily in English but not in Bangla. Bangla language also has a lot of English words that are used the way they are. Bengali names and entities can have various forms and transliterations. There can be multiple ways to represent the same entity due to different spelling conventions, variations in pronunciation, and transliteration from other scripts.

But despite all these challenges and difficulties it has been possible to develop gazetteers and it is now an ongoing process. Some of the major organizations involved in this work include the Bangladesh Bureau of Statistics, the Survey of Bangladesh, and the National Institute of Local Government. Beside this in many research works in NER, a proper set of Bangla gazetteers have been formed and used as well.

3.2.3 Our Bangla Gazetteer Formation

We have created a Gazetteer in this research. The whole workflow of the formation of our gazetteer is given in figure 3.3.

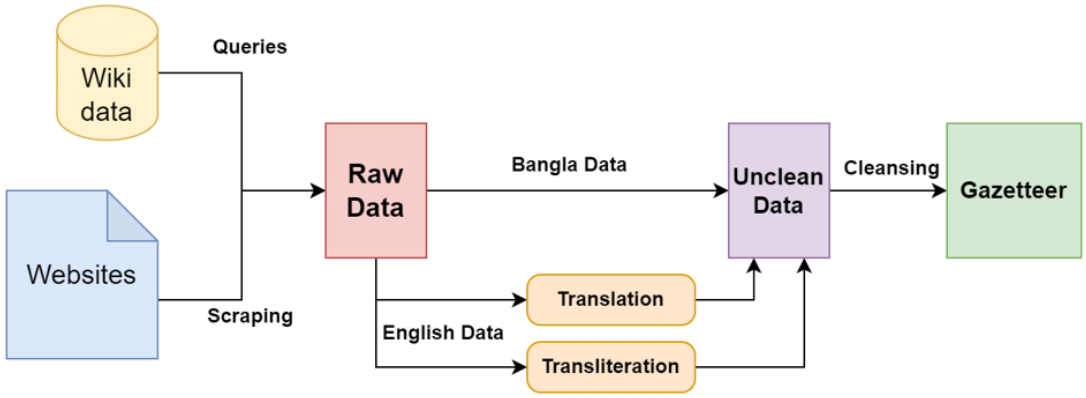


Figure 3.3: Workflow of the formation of our gazetteer

Extracting Bangla Data from Wikidata

We have collected a huge amount of data for each of the tags (LOC, PER, CORP, GRP, PROD, CW). The source we mainly used is Wikidata. Wikidata acts as central storage for the structured data of its Wikimedia sister projects including Wikipedia, Wikivoyage, Wiktionary, Wikisource, and others. We have used Wikidata Query Service where we have put different queries like this and got the desired data.The queries are written in SPARQL language. One example query that we used to make the gazetteer is given in 3.4.

```
LIMIT 1000.
SELECT ?location ?locationLabel
WHERE {
  ?location wdt:P31/wdt:P279* wd:Q486972.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "bn". }
}
```

Figure 3.4: An Example of SPARQL Query for WikiData

Using these queries, we extracted several *CSV* files containing a lot of Bangla data for each of the NER tags we focused on. However, we couldn't get enough data using it for some tags.

Manual Scraping of Bangla Data

Apart from the data we got from Wikidata, we also scraped a lot of websites including Wikipedia and newspaper portals to gather Bangla data. For scraping, we used frameworks like *BeautifulSoup* which is a Python package made for web scraping. However, as those data were randomly gathered, we had to manually hand-pick the data according to our tags. We especially did this for PROD and CW as the number of Bangla Wikidata of these were significantly low.

Using English Data

As we were exploring with Wikidata, we found that the amount of English data on this was enormous. So, like how we gathered the Bangla Data, we also gathered a lot of English data according to the tags required. We used both Wikidata and manual scraping for this as well. Then with the English data, we did two things:

1. **Translation:** We translated the English data to Bangla data as it is. For example, "Hilsa Fish Fry" is translated to "ইলিশ মাছ ভাজা". We did this using the help of *Google API* translation tool.

2. **Transliteration:** Transliteration refers to the method of mapping from one system of writing to another based on phonetic similarity. We transliterated the English data to Bangla data. We did this because some foreign words in Bangla are written based on the foreign phonetics. For example, if we translated "Chicken fry" in Bangla it would be translated into "মুরগি ভাজা". However, in Bangla, we do not use "মুরগি ভাজা" in our sentences instead we use it as "চিকেন ফ্রাই". That's why we kept both the translated and transliterated version in the gazetteer.

Cleansing the Data

As we were working on a large list of words, there were some broken words and some unnecessary punctuations here and there. We cleaned the data by removing all these unnecessary punctuations and broken words. Then we also removed any duplicate words. For doing all these we made some custom Python scripts and also did some manual inspection.

After collecting all 93,749 data, the distribution of each tags is given in table 3.3 and figure 3.5.

NER Tags	Number of data
PER	26296
CW	20446
LOC	16617
CORP	13737
GRP	10517
PROD	6136

Table 3.3: Total Number of Each Tag in our Gazetteer

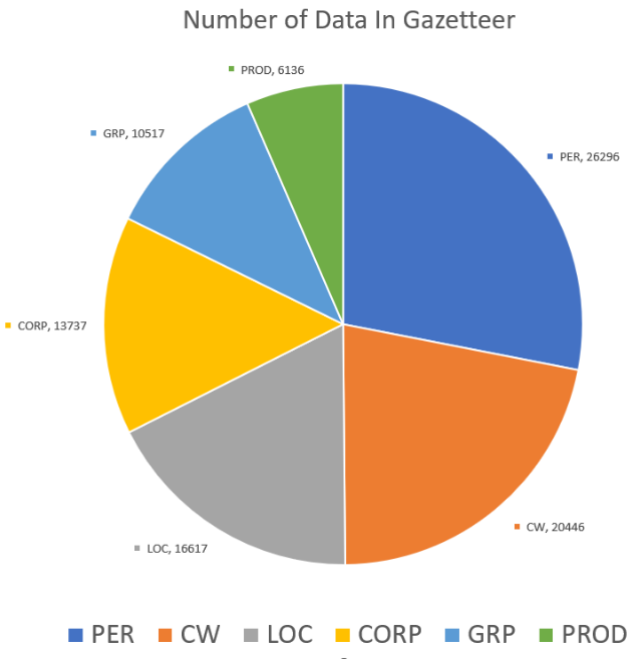


Figure 3.5: Total Number of Each Tag in our Gazetteer

Trie Data Structure

This data structure follows a treelike structure where every node represents a single character. The root of the tree is an empty string and the edges are the characters which stay connected with the nodes. The nodes can have multiple child nodes. The string can be reconstructed which is connected to the nodes by traversing the tree from the root to that node.

This data structure provides the fastest search operations. It is very good at retrieving the strings that include a specific prefix. You can quickly explore all descendants and extract all matching strings by moving up the tree from the root to the node that corresponds to the prefix. Because of this, Trie is especially beneficial for applications and autocomplete systems that require prefix-based searches. Moreover it can sort automatically. As it has a tree-like structure, traversing the Trie from root to leaf nodes will provide strings that are sorted. The figure 3.6shows an example of Trie. We created a Trie Data Structure where we inserted all the gazetteer words. We modified the node class so that it can return us their corresponding NER tags. As we have 93,749 entities, by using Trie DS, we can quickly find if given token is present in gazetteer or not. The time complexity of searching in Trie Data Structure is $O(N)$ where N is the length of the string.

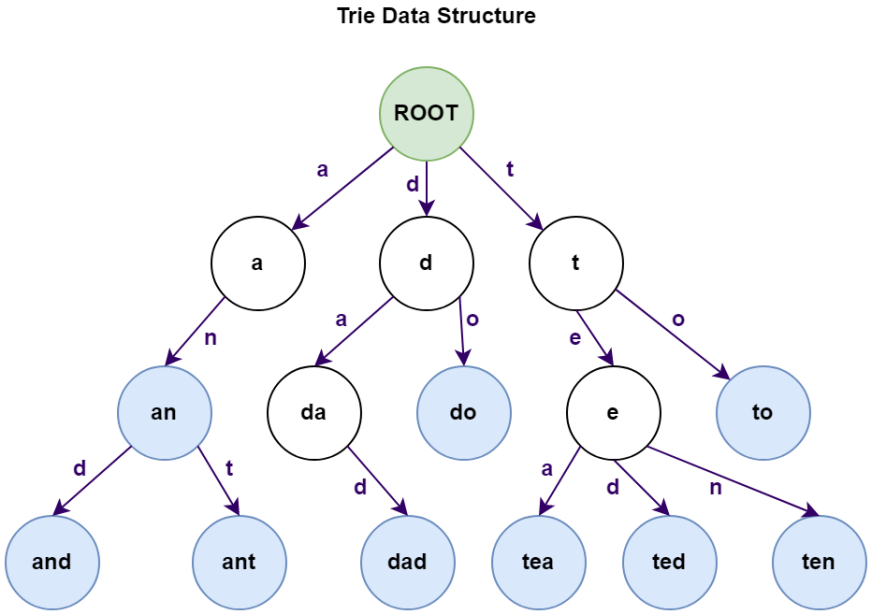


Figure 3.6: Trie Data Structure

We have also implemented a function that takes a Bangla sentence as input and it gives corresponding one-hot vector representation for each word in the sentence. For example - if we search this Bangla sentence "তানভীর হোসেন ব্র্যাক বিশ্ববিদ্যালয়ে পড়েন" we will get the following output like table 3.4.

Sentence	**One Hot Vector**	**Tags**
তানভীর	[0,0,0,0,0,0,0,0,0,0,1,0]	B-PER
হোসেন	[0,0,0,0,0,0,0,0,0,0,0,1]	I-PER
ব্র্যাক	[0,0,0,0,1,0,0,0,0,0,0,0]	B-CORP
বিশ্ববিদ্যালয়ে	[0,0,0,0,0,1,0,0,0,0,0,0]	I-CORP
পড়েন	[0,0,0,0,0,0,0,0,0,0,0,0]	O

Table 3.4: Gazetteer Representation of Words of a Sentence

3.3 Data Analysis

We have performed Exploratory Data Analysis (EDA) in the MultiCoNER dataset. Our findings are presented below:

3.3.1 Large Test Data

At the very beginning of our data analysis, we saw that the amount of test data (133,119 sentences) was overwhelmingly larger than the train data (15,300 sentences), which is mainly done in order to make the model ready to deal with real-world data. However, this means the features from our train data alone are not sufficient enough for our models to perform a good result which can be reflected in the baseline score (XLM-RoBERTa) of MultiCoNER's paper [14].

3.3.2 Irregularities in the Punctuations

We noticed that in the tokens, there were a lot of irregularities in the punctuations which sometimes caused issues. For example, a sentence 'অনেক চকোলেট ! আছে' is sometimes tokenized as <'অনেক'>, <'চকোলেট !'> <'আছে'> <'চকোলেট।'> and sometimes as <'অনেক'>, <'চকোলেট'> <'চকোলেট !'> <'আছে।'> . The problem arises as the tokenizer will detect <'চকোলেট !'> and <'চকোলেট'> as two different tokens which actually shouldn't be the case. This occured when we were preparing our data for our first two models: BiLSTM and Transformer Based Model. The problem occurred a bit differently when we used the BanglaBERT tokenizer. The way we handled that case is explained in section 3.3.6. To make our BiLSTM and Transformer based model to work, we removed the punctuation from those tokens where they are added as prefixes. We were careful while removing it, because some tokens had only one punctuation and were tagged as 'O'. We did not remove those punctuation which were a single token. We also didn't remove the period in abbreviations of the words. The effect of our punctuation removal can be seen in the table 3.5.

Before Removing Punctuation			**After Removing Punctuation**		
Sentence	**Word**	**Tag**	**Sentence**	**Word**	**Tag**
2	(খাবার)	I-PROD	**2**	খাবার	I-PROD
23	যায়।	O	**23**	যায়	O
27	,	O	**27**	,	O
31	ছোট।	O	**31**	ছোট	O
43	কমিউনিটি,	B-CW	**43**	কমিউনিটি	B-CW
123	পি.এইচ.ডি	O	**123**	পি.এইচ.ডি	O

Table 3.5: Before and After of Punctuation Removal

3.3.3 Presence of Foreign Words

In the dataset, we also found words of some other languages such as Hindi, Farsi, English, Chinese, and Russian. The presence of foreign language data in Bangla Dataset can be caused by translation errors as we have discussed in the formation of MultiCoNER Dataset. Some examples of foreign words in train data are given in figure 3.7.

Train Data Row	Sentence
125	কাঠের বৈচিত্র্যময় বিষয়গুলির মধ্যে রয়েছে রাস্তার দৃশ্য, still life এবং পুষ্পশোভিত বিষয়, অন্যদের মধ্যে।
881	একটি ডুব বোম্বারের ক্রনিকলস (хроника пикирующего бомбарди-ровщика, ১৯৬৭) মেজর, এয়ারফিল্ডের কমান্ড্যান্ট হিসেবে

Figure 3.7: Presense of foreign language

3.3.4 Word Frequency

We calculated the word frequency of our dataset. Word frequency determines how many times a word is being used in sentences. As we observed the word frequency of the Bangla Datasets, from figure 3.8, we could see that in the train dataset 'এবং': 4346, 'এর': 3545, 'একটি': 3404, 'তিনি': 2824, 'সালে': 2238 are the top 5 most frequent words with their frequency. So, we saw that a large number of these words are stopwords, which have “O” tags.

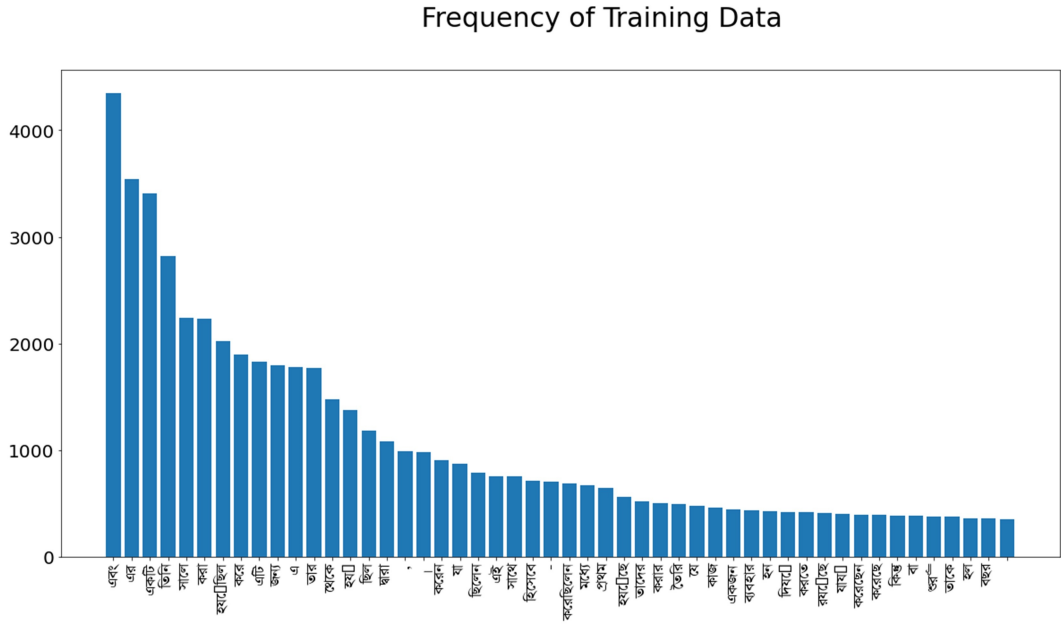
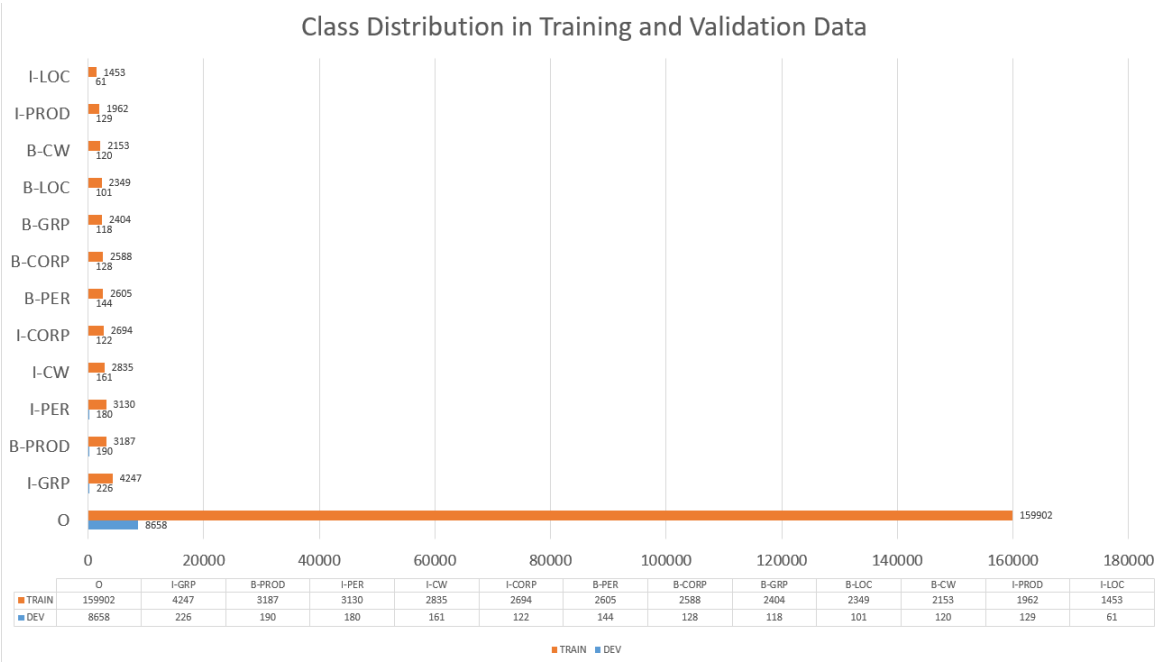


Figure 3.8: Frequency of words in Training Data

3.3.5 Imbalanced Dataset

One common problem of NER is, the most frequent class is 'O'. This causes an imbalance in the dataset. When there is an imbalance in a dataset, the model is more likely to be biased towards the majority class and perform poorly on the minority class. This can lead to poor overall performance, as well as skewed metrics such as accuracy. In the MultiCoNER dataset, this problem was more extreme because 83.5% the 'O' tag in training data. The detailed distribution of classes in train and dev dataset is given in 3.9.



	O	I-GRP	B-PROD	I-PER	I-CW	I-CORP	B-PER	B-CORP	B-GRP	B-LOC	B-CW	I-PROD	I-LOC
TRAIN	159902	4247	3187	3130	2835	2694	2605	2588	2404	2349	2153	1962	1453
DEV	8658	226	190	180	161	122	144	128	118	101	120	129	61

Figure 3.9: Class Distribution of Training and Validation Data

3.3.6 Alignment of tags after tokenization

Models like BanglaBERT base and BanglaBERT large use their BanglaBERT tokenizer. This tokenizer takes a sentence and breaks it into tokens with their input ids and attention masks. However, this created a problem of misalignment of the tags in the sentence. To visualize the alignment problem, the following example is given in table 3.6.

To fix this issue, we have to manually iterate through the dataset, and align the tags based on the tokens generated by the tokenizer. We had to fix the alignment issue by creating a preprocessing function which was executed in the train, test and validation dataset. This alignment function uses BERT tokenizer to tokenize the sentence. Then it analyzes each token with its corresponding tags. As we have noticed, the broken part of the word starts with "#" such as: "হেনরিচ" is broken into "হেনরি" and "##চ". We utilized this "#" to detect the broken word and assign them to their original tag. The table 3.7 shows the output of this alignment function.

Sentence: আরও জনপ্রিয়তা ছিল হেনরিচ ডুমৌলিন এর লেখার কারণে।

Given Sentence in Dataset	**Given Tags in Dataset**	**After Tokenization**	**Predicted Tag By BanglaBERT**	**Comments while measuring F_1**
আরও	**O**	আরও	**O**	**Correct**
জনপ্রিয়তা	**O**	জনপ্রিয়তা	**O**	**Correct**
ছিল	**O**	ছিল	**O**	**Correct**
হেনরিচ	**B-PER**	হেনরি	**B-PER**	**Correct**
ডুমৌলিন	**I-PER**	##চ	**B-PER**	**Misaligned & Incorrect**
এর	**O**	ডুম	**I-PER**	**Misaligned & Incorrect**
লেখার	**O**	##ৌল	**I-PER**	**Misaligned & Incorrect**
কারণে।	**O**	##িন	**I-PER**	**Misaligned & Incorrect**
[PAD]	**O**	এর	**O**	**Misaligned & Correct**
[PAD]	**O**	লেখার	**O**	**Misaligned & Correct**
[PAD]	**O**	কারণে	**O**	**Misaligned & Correct**
[PAD]	**O**	।	**O**	**Misaligned & Correct**

Table 3.6: Allignment Issue after Tokenization

Given Sentence in Dataset	**Given Tags in Dataset**	**Preprocessed Dataset**	**Aligned Tag**
আরও	O	আরও	O
জনপ্রিয়তা	O	জনপ্রিয়তা	O
ছিল	O	ছিল	O
হেনরিচ	B-PER	হেনরি	B-PER
ডুমৌলিন	I-PER	##চ	B-PER
এর	O	ডুম	I-PER
লেখার	O	##ৌল	I-PER
কারণে।	O	##িন	I-PER
		এর	O
		লেখার	O
		কারণে	O
		।	O

Table 3.7: After Fixing Alignment Issue

Chapter 4

Description of the Model

As we were dealing with data where understanding the context was extremely important, we tried two different deep learning architectures which are especially great at time series prediction and Natural Language Processing. A detailed description of our models are given below:

4.1 Two Layer BiLSTM Network

The architecture of the Two Layer BiLSTM Network is shown in figure 4.1.

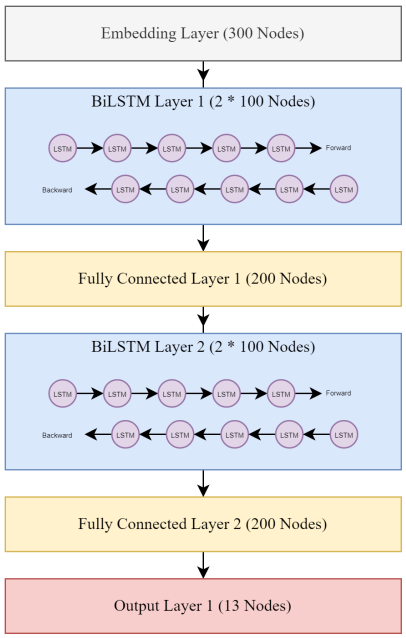


Figure 4.1: Two Layer BiLSTM Network Workflow

At first, we used an embedding layer that has 300 dimensions for each token. Then we used our first BiLSTM layer followed by a Feed Forward Neural Network(FFNN). After that, the model has the second BiLSTM Layer which is again followed by the second FFNN. Both BiLSTM and FFNN layers have 200 dimensions. Finally, we used the time distribution layer to get our tags.

4.1.1 LSTM

A Recurrent Neural Network (RNN) type, Long Short-Term Memory (LSTM) [19], 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. A single cell of LSTM is shown in figure 4.2

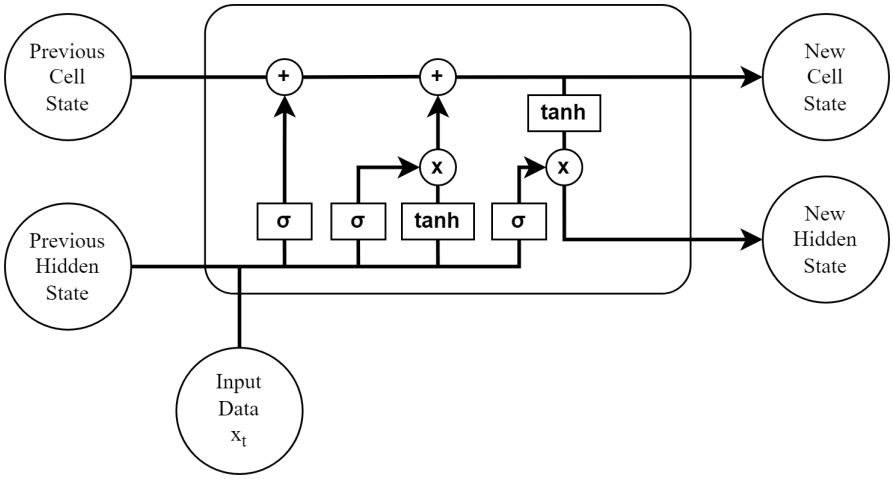


Figure 4.2: A single cell of LSTM

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.

4.1.2 BiLSTM

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. The structure of BiLSTM can be found in 4.3

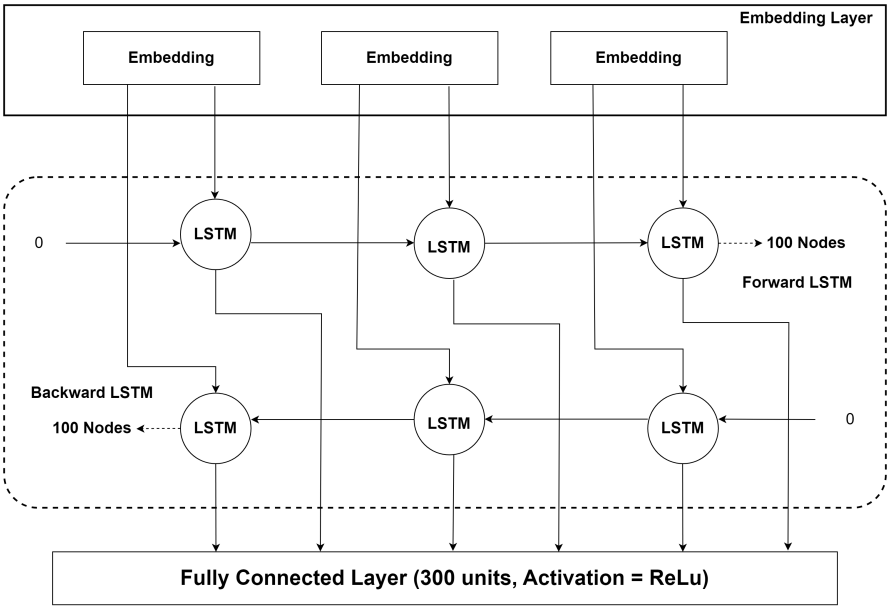


Figure 4.3: Structure of BiLSTM

4.2 Single Transformer-Based Network

The architecture of the Transformer Based Network is shown in figure 4.4.

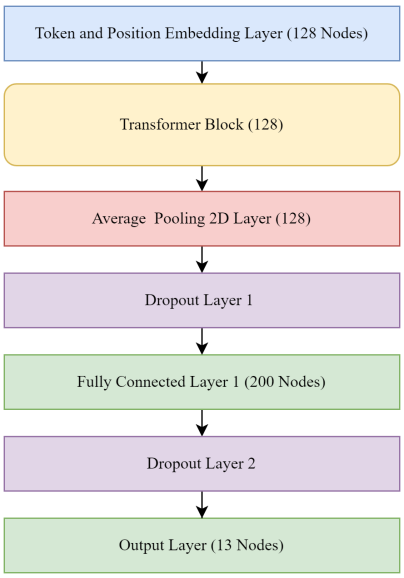


Figure 4.4: Single Transformer-Based Network Workflow

Initially, we used token and position embedding of 128 dimensions for our word token and pos tag of word tokens. Then the model passes it to the transformer block. The output of the transformer block then goes through the average pooling 2d layer. Here, we used a dropout layer to prevent overfitting. Then, It passes through a fully connected layer with 200 nodes which is also a dropout layer. The final layer is our output layer which is another fully connected neural network.

4.2.1 Token and Position Embedding

Token embedding and position embedding are two types of embeddings used in natural language processing tasks. Token embedding is a method applied to stand for words or tokens in a continuous vector space by converting words, which are discrete symbols, into continuous vectors that can be processed by a machine learning model. Position embedding, on the other hand, is a technique used to represent the position of a token in a sequence of tokens, this can be useful in tasks such as language understanding and machine translation, where the position of a word in a sentence can be important for understanding its meaning. Both token embedding and position embedding are commonly used together in neural networks such as Transformer and BERT, to represent the words and their positions in a sentence in a continuous vector space, and are updated during the training process to learn useful representations for the specific task at hand. We used this embedding to process our words and their pos tag simultaneously.

4.2.2 Transformer

A revolutionary deep learning architecture, the Transformer [20], was first mentioned in the paper "Attention Is All You Need" by Google researchers back in 2017. This neural network model is versatile and can be practiced to a range of natural language processing tasks, as well as language translation, text summarization, and language understanding. The Transformer model is built on the principle of self-attention, which enables it to assign different levels of significance to various segments of the input while making predictions. The model uses multiple attention heads to learn different relationships between the input tokens, which allows it to capture more complex patterns in the data.

The Transformer model architecture has an encoder as well as a decoder. The encoder receives the input sentence and passes it through multiple layers of self-attention and feed-forward neural networks to produce a set of hidden representations. The decoder then takes these hidden representations and generates the output sequence.

One of the key innovations of the Transformer model is the use of a technique called positional encoding, which allows the model to take into account the order of the input tokens. This is important for tasks such as language translation, where the meaning of a word depends on its position in the sentence. The structure of the Transformer Model is given in figure 4.5.

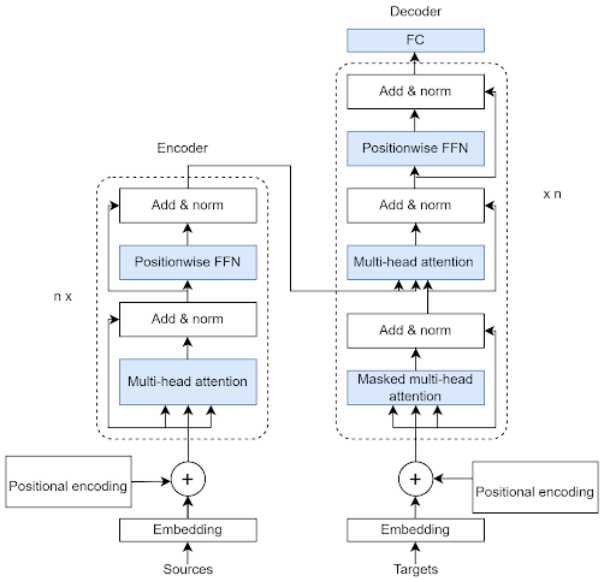


Figure 4.5: Transformer Model

In summary, the Transformer is a neural network architecture that builds upon the concept of self-attention, it uses multiple attention heads to learn different relationships between input tokens, it consists of an encoder and a decoder, it uses a technique called positional encoding to take into account the order of the input tokens, and it is proven to be highly beneficial in many Natural Language Processing tasks.

4.3 Fine-Tuned BanglaBERT

4.3.1 BERT

The full form of BERT [13] 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 which we have mentioned in section 4.2.2. As we discussed before, transformer architecture is highly effective in capturing contextual relationships in language which is a major advantage in named entity recognition 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 time, BERT tries to predict whether the second sentence follows the first sentence or not. This phase allows BERT to understand the relationships between sentences and captures the ability to understand discourse and coherence.

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 and the BERT large model can be found in figure 4.6 and 4.7.

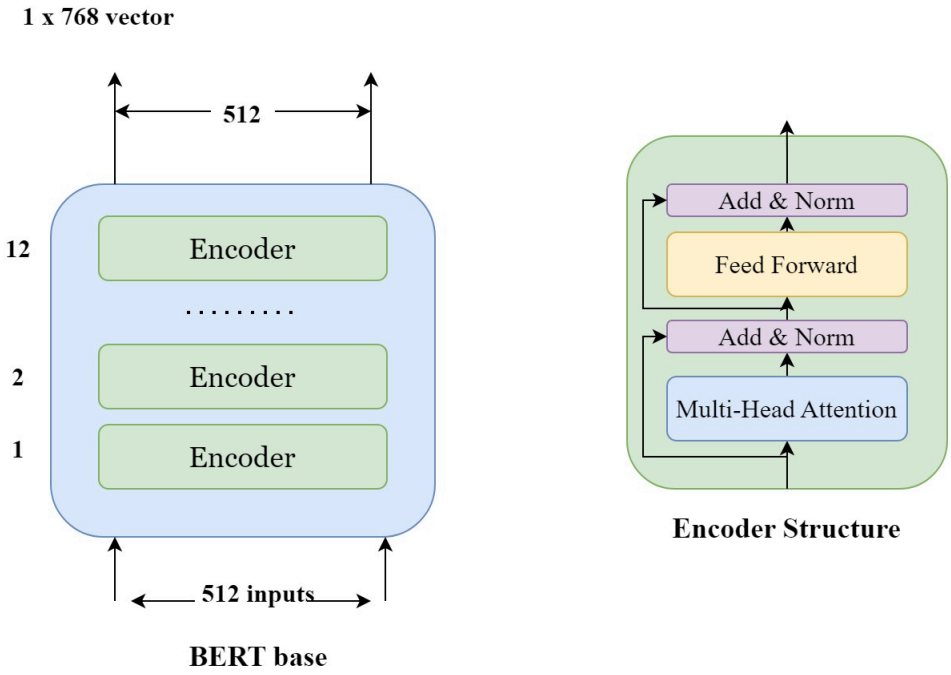


Figure 4.6: BERT Base Structure

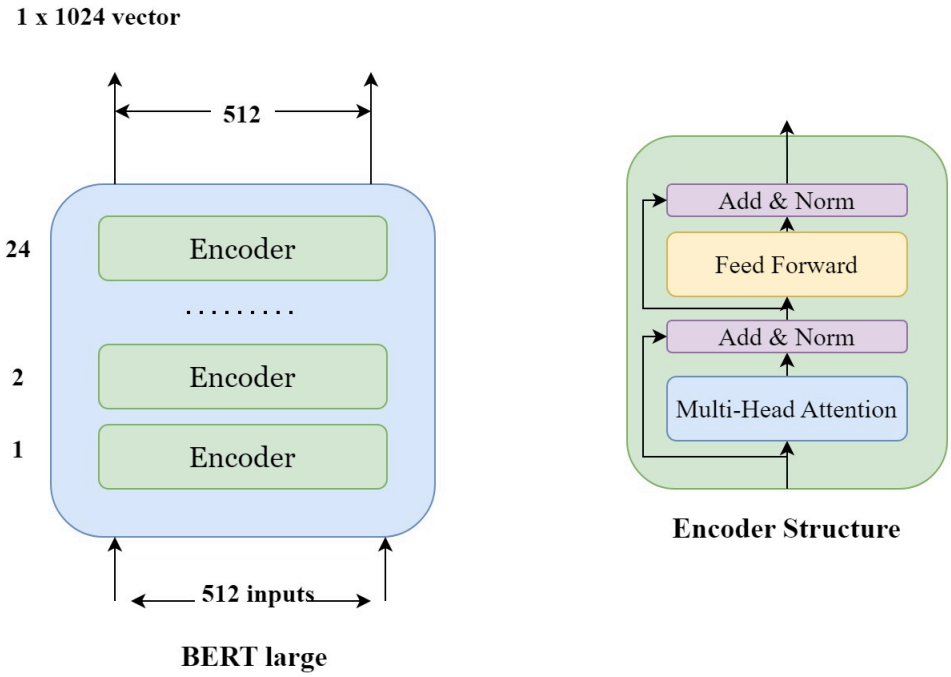


Figure 4.7: BERT Large Structure

4.3.2 ELECTRA

ELECTRA [21] (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) is another important deep learning model. It is introduced by Stanford and Google Brain in 2020. The researchers proposed an alternative approach to pre-training language models. ELECTRA has achieved competitive performance in various NLP tasks and benchmarks, often surpassing or achieving similar performance as BERT while being more computationally efficient.

The main idea behind ELECTRA is to employ a discriminative pre-training instead of generative one like BERT. During pretraining, it replaces a certain percentage of the input tokens with plausible alternatives whereas BERT masks out and predicts randomly selected tokens in a sentence. Now, the task of the discriminator is to determine if the replacements are correct or not. As it is focused on discrimination task, ELECTRA can efficiently learn from the difference between replaced and original tokens.

The core architecture of ELECTRA is based on the same transformer model that is used in BERT. It has stacked layers of self-attention mechanisms and feed-forward neural networks. But, the differentiator of ELECTRA is the generator and the discriminator. The generator proposes replacement tokens, while the discriminator learns to distinguish between replaced and original tokens.

The pre-training phase have some similarities with GAN (Generative Adversarial Networks). During Pre-training, ELECTRA trains both the generator and the discriminator simultaneously. The generator proposes replacement tokens from a generator distribution. The discriminator discriminates every token as either replaced or original. This adversarial training setup makes generators produce more plausible replacements that are difficult for the discriminator to distinguish from original words. The generator and discriminator of ELECTRA are jointly trained using a modified version of MLM.

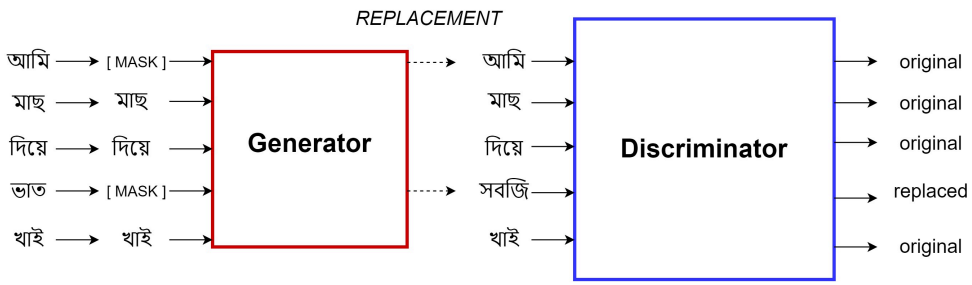


Figure 4.8: ELECTRA Structure

The fine-tuning of ELECTRA on specific tasks is similar to BERT. Its encoder's output can also be used as embeddings for other neural network architecture. It is computationally more efficient than BERT, making it an effective option for NLP tasks.

4.3.3 BanglaBERT

BanglaBERT [12] is a pre-trained language model based on ELECTRA. It is trained on a large corpus of Bangla text. As Bangla is a rather resource-constrained language in the web domain, the researcher of BanglaBERT used data from Wikipedia and 110 Bangla websites (selected by their Amazon Alexa rankings). This data includes blogs, e-books, news etc. The architecture of BanglaBERT is the same as the original BERT model. The base model has 12 layers of transformers encoder and each encoder has an embedding size of 768. On the other hand, the BanglaBERT large model has 24 layers of encoder, each encoder has an embedding size of 1024.

4.3.4 Fine-tuned BanglaBERT Base

As we have mentioned earlier, the base model has 12 layers of transformers encoder and each encoder has an embedding size of 768. The process of fine-tuning is taking a pre-trained model and further training it on a task-specific dataset in order to adjust it for certain applications or domains. The model for Fine-tuned BanglaBERT Base has been improved on a variety of tasks involving the Bangla language, including sentiment analysis, question answering, named entity recognition, text categorization, and machine translation. The structure of the model given in figure 4.9.

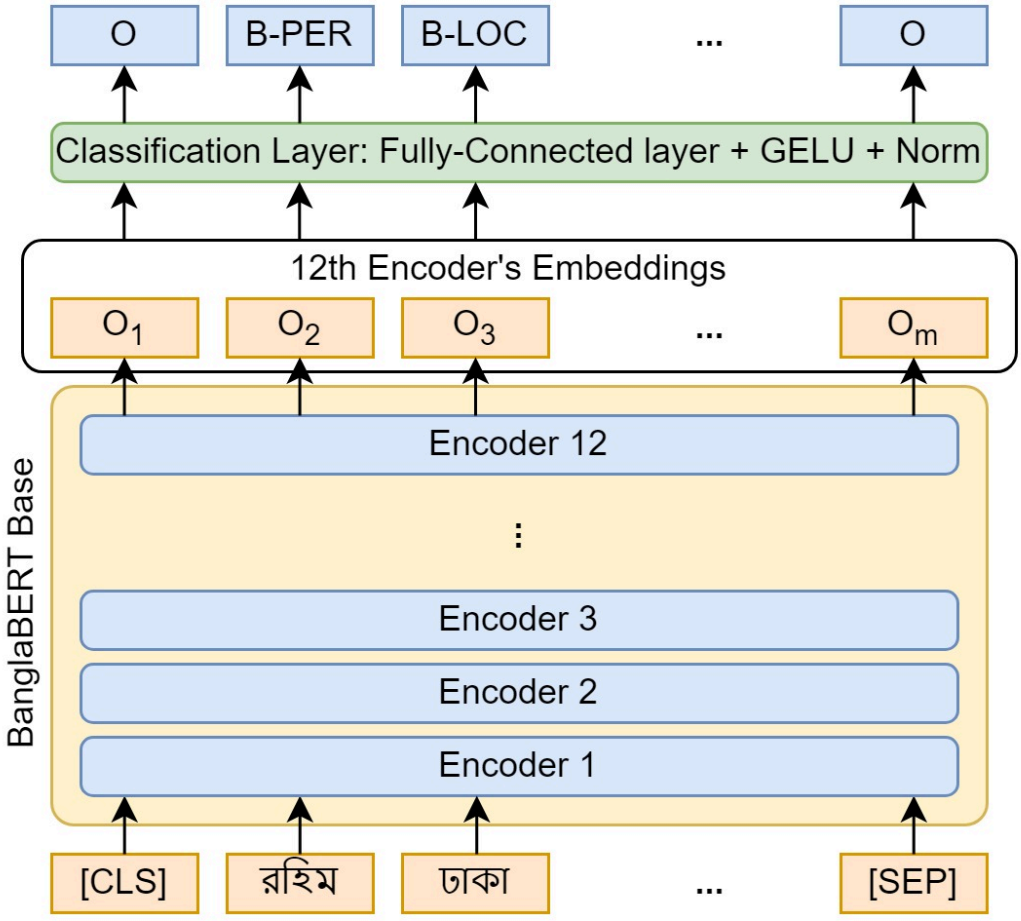


Figure 4.9: Fine-tuned BanglaBERT Base Model

We fine-tuned the pre-trained BanglaBERT Base model on the MultiCoNER dataset to perform Named Entity Recognition (NER) in Bangla. We used BanglaBERT tokenizer to get the input ids and attention masks. Then the data was padded using the *DataCollatorForTokenClassification* function where the max sequence length was 64. The learning rate was set to 2e-5 and the batch size was 32. We used Pytorch's Cross-Entropy loss function. At first, we did not use any custom weights for each tag. Later, we used our custom weights for each tag in the categorical cross-entropy loss function, which can be found in section 4.3.6. We used 6 epochs for fine-tuning the model as the score was not improving after that.

4.3.5 Fine-tuned BanglaBERT Large

The BanglaBERT model has 24 layers of transformers encoder and each encoder has an embedding size of 1024. As it is bigger than the base model, we expect it to perform better in this task. The structure of the model is given in figure 4.10.

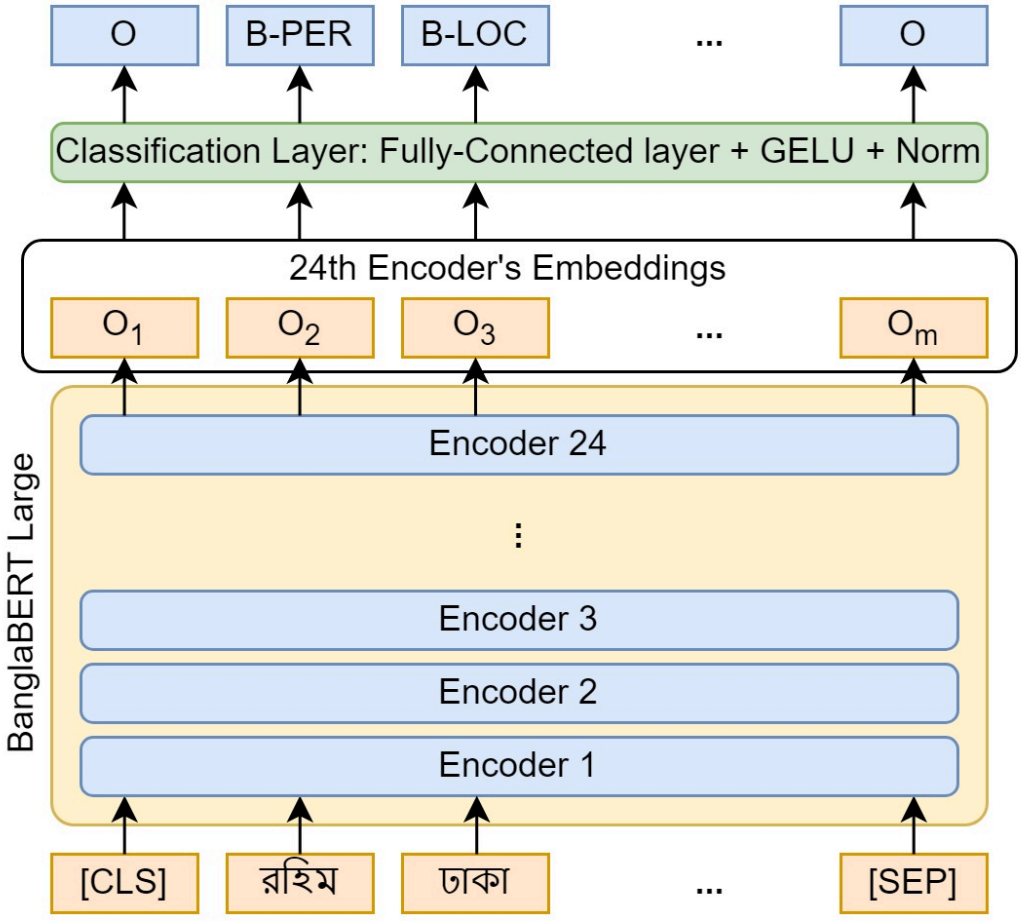


Figure 4.10: Fine-tuned BanglaBERT Large Model

We fine-tuned the pre-trained BanglaBERT large model on the MultiCoNER dataset to perform Named Entity Recognition (NER) in Bangla. Like the base model, we used BanglaBERT tokenizer to get the input ids and attention masks. Then the data was padded using the *DataCollatorForTokenClassification* function where the max sequence length was 64. Similar to the base model, the learning rate was set to 2e-5 and the batch size was 32. We used Pytorch's Cross-Entropy loss function. At first, we did not use any custom weights for each tag. Later, we used our custom weights for the categorical cross-entropy loss function, which can be found in section 4.3.6. We used 6 epochs for fine-tuning the model as the score was not improving after that.

4.3.6 Custom Categorical Cross Entropy Loss Function

As our dataset was imbalanced, we developed a custom loss weights to ensure the model reduces the bias toward more encountering tags. The custom loss weights are calculated using the following formula:

$$\left(1 - \frac{(n+2) * \text{count}(\text{ Tag })}{\text{total}}\right) * 10$$

Here, n represents the number of classes, count(Tag) is representing the number of that tag in the dataset and the total is representing the total number of tokens in the dataset. The added 2 was for normalizing the weights. By using this formula, the weights of each tag in the loss function can be adjusted based on their frequency in the dataset.

Our calculated weights are given in the following table 4.1.

O	1.054019840917585
B-CORP	9.855210087105194
B-CW	9.879546876946478
I-CW	9.841391266206811
B-GRP	9.865504269474840
B-PER	9.854258994168868
B-PROD	9.821698047760531
I-PROD	9.890232685819315
B-LOC	9.868581334857073
I-LOC	9.918709527265783
I-GRP	9.762394605848439
I-PER	9.824887006429389
I-CORP	9.849279742913984

Table 4.1: Custom Loss Weights

4.4 Clustering Algorithm

Clustering Algorithm or Clustering is a machine learning technique that refers to an unsupervised algorithm that is used to group similar types of data points. Some data points, even though they represent different values, have similar features or properties to other data points. Clustering algorithms use these similar types of properties to group the data points into specific groups. These kinds of clustering are necessary when we have a lot of data. There are different kinds of clustering algorithms K-means, Gaussian Mixture Model, BIRCH algorithm, etc.

4.4.1 K-means Clustering Algorithm

K-means clustering is one of the widely used clustering algorithms. The algorithm tries to reduce the variance of data points within a cluster. Formally, it wants to partition n number of data points into k number of clusters. So, before using this algorithm we must know how many clusters we want. The objective of K-Means clustering is to minimize total intra-cluster variance, or, the squared error function:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

In the formula, the first summation is the number of clusters we want. The second summation is the number of cases and then it is followed by the Distance Function which is c_j, the centroid of the cluster.

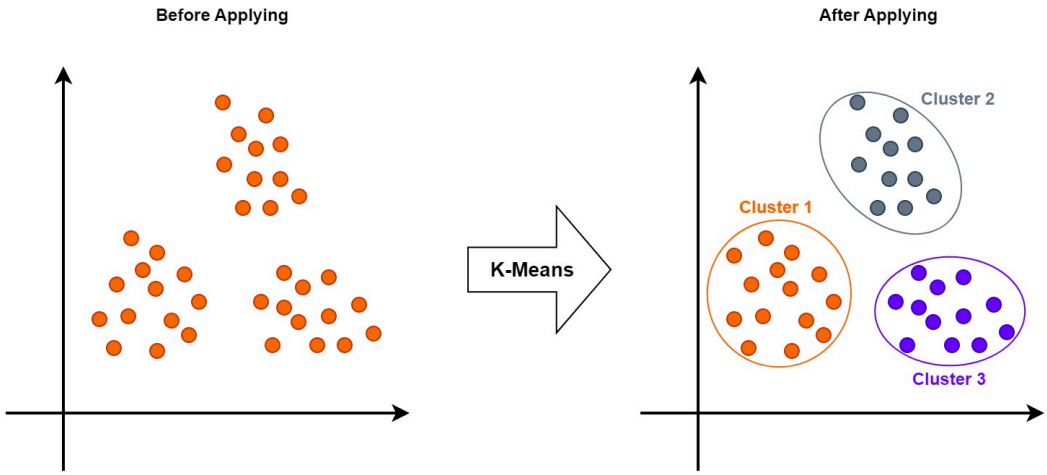


Figure 4.11: K-means Clustering

In figure 4.11, we can see the data points are clustered into 3 groups.

However, we are not making a NER model with this algorithm, rather we use this later as a feature extraction method discussed in 4.5.2.

4.5 Feature-Based Learning

In the field of Natural Language Processing and Machine Learning, feature engineering is a vital aspect when it comes to constructing effective models for complex tasks like Named Entity Recognition (NER). While traditional machine learning methods depend upon predefined inputs that are used by the algorithm to generate predictions, these techniques do not have an inherent ability to discover meaningful patterns or structures within raw data. On the other hand, advanced deep learning frameworks can extract valuable insights and relationships directly from large amounts of data through automatic representation discovery. As a result, incorporating sophisticated techniques for feature engineering into NER models helps enhance their accuracy and efficiency significantly. Ultimately, this process enables researchers to build more robust and accurate systems capable of handling challenges unique to natural language understanding.

4.5.1 Conditional Random Fields

CRF is a probabilistic graphical model dedicated to tasks like named entity recognition, part-of-speech tagging, semantic role labeling, and handwriting recognition. This model is particularly useful in natural language processing (NLP) and computer vision. CRF models are trained using labeled data, where parameters are learned to maximize the likelihood of observed label sequences. During inference, the most likely sequence of labels is computed based on the learned parameters and observed features. CRF is widely used in NLP for capturing contextual information and achieving accurate sequence labeling.

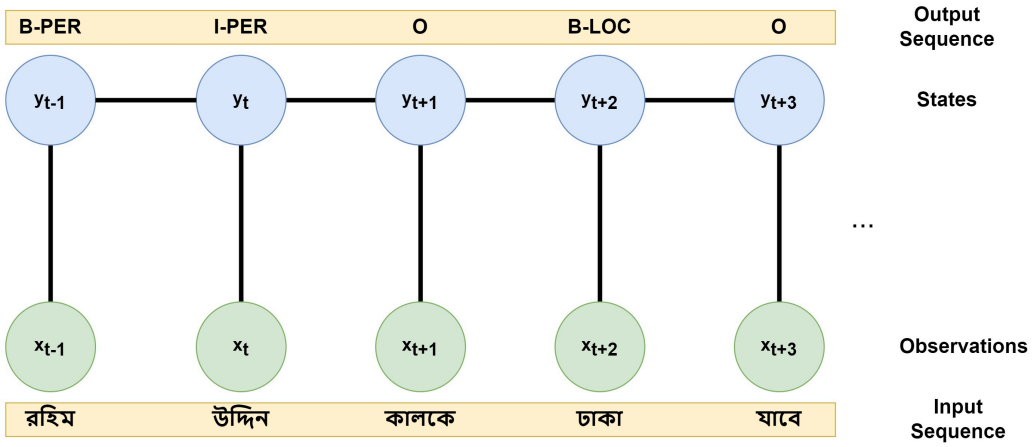


Figure 4.12: CRF

To comprehend how CRF functions in practice, let us consider an example with n words represented by feature vectors x_1, x_2, ..., x_n. Correspondingly, there exists a label sequence y_1, y_2, ..., y_n associated with each word. A feature function denoted by Φ(x, y) takes both the input feature sequence and the label sequence as arguments and generates a vector of weights w that capture the relationship between them. The primary aim of CRF is to locate the most plausible label string given

the input features, which can be expressed mathematically as follows:

$$\arg\max_y P(y \mid x) = \arg\max_y \left(\frac{\exp\left(w^T \Phi(x, y)\right)}{\sum_{y'} \exp\left(w^T \Phi\left(x, y'\right)\right)} \right)$$

4.5.2 Feature Engineering

We approached the task of developing a powerful Natural Language Processing (NLP) system using an array of different feature sets. Through rigorous experimentation and analysis, we sought to optimize the classifier's capability to learn from data and generalize effectively on unseen instances. Selecting appropriate features played a critical role in achieving this goal. Here, we faced several difficulties due to linguistic differences between English and Bangla languages. For instance, while certain features may perform well for English text, they might prove redundant or irrelevant for Bangla, impeding the model's effectiveness. To tackle this challenge, we focused on designing specialized features targeted specifically toward Bangla NER requirements. Below, we presented the detailed descriptions of the feature set that we employed for the NER task.

Suffixes: We recognized the significance of considering cultural nuances while dealing with Bangla text. Many locations in Bangladesh tend to finish with popular suffixes such as "##আলয়", "##পুর", "##লিয়া". Similarly, some names in Bangla often terminate with "#আ". By including suffixes as features, our classifier could capitalize on Bangla-centric patterns, ultimately improving its NER capabilities. For our NER models, we took up to the last 3 suffixes, for example, for the word "ফারহান" we used "হান", "◌ান" and "ন" as the suffix features.

Prefixes: Exploring information beyond just a single word could provide additional insight into the contextual meaning of the term under consideration. Our implementation therefore leveraged preceding and succeeding words' contents as useful prefix information. For our NER models, we took up to the last 3 prefixes, for example, for the word "ফারহান" we used "ফার", "ফা" and "ফ" as the prefix features.

Length: Length can be a helpful feature because it provides an indication of the structure or complexity of certain types of entities. For instance, many place names may have longer lengths due to their geographical specificity. On the other hand, person names, which typically follow a pattern of first name + last name, will also exhibit varying lengths depending on individual preferences and cultural norms. By incorporating length as a feature, NER algorithms can capture patterns associated with different classes, potentially allowing them to learn the subtle differences between diverse entity types.

Neighboring Words: Encountering similar words before or immediately after the current one frequently implies relatedness and shared topics. Harvesting those words (for up to three positions away) as separate features allowed us to investigate how adjacent terms could aid NER tasks. For example, if the next word is "করেছিলেন" it is likely that the word that we are investigating is a person's name. Similarly, if a word is neighbored by words like "ম্যাগাজিন", "গান", "সিনেমা", is likely these words are creative word.

IsDigit: Knowledge of whether a given word consists entirely of numbers can prove useful in discriminating between various types of texts and facilitate filtering out irrelevant content. This feature alone might not contribute significantly toward improving general NER accuracy, especially when considered independently. However, combining it with other indicators can be quite helpful.

IsPunctuation: If a word is entirely punctuation (such as commas, periods, question/exclamation marks, etc.) would not influence the NER output directly, but it can serve as a convenient indicator that a word should be skipped or should be detected as "O" tag as it can never be any other tag.

IsBangla: Even though we were working with Bangla Data, the dataset contains some words that are not Bangla. In particular, there are some occurrences of Hindi, Farsi, Chinese, and English languages. For these words, there is this special feature where if a word is not Bangla, it will indicate that the word is not Bangla which should give the model information to maybe skip these words. We implemented this by converting all the letters of the word into Unicode and then all the Unicode between 2432 to 2559 or 32 (Unicode of space) is considered as a Bangla word, else it is not a Bangla word.

IsStopword: Similarly to IsDigit, IsPunctuation, and IsBangla, if a word is a stopword, it contains little to no value to the NER model hence it should be excused or should be tagged as "O".

Word Frequency: Word frequency plays a crucial role in influencing the performance of Conditional Random Fields (CRF) models in natural language processing tasks such as named entity recognition (NER). Although not directly affecting the outcome itself, word frequencies can indirectly impact model efficacy.

Part of Speech (POS) Tags: Incorporating Part of Speech (POS) tags offers valuable insights into the grammatical composition of sentences and the functional roles played by individual words. An example of a POS tag is given in figure 4.13.

Figure 4.13: Example of POS Tag

POS tags are essential features in Conditional Random Fields (CRF)-based named entity recognition (NER) systems, enabling efficient discrimination between named entities and other linguistic elements. In our implementation, we leveraged the Bengali POS tag model of BNLP [22], a CRF-driven POS tagger built upon the NLTR dataset with an 80.75 F_1 score. However, as the POS tagger was only trained with

2247 sentences, we highly doubt it will bring that much significance to the model as the dataset we are working on is very diverse and has new words. So, to enhance the effectiveness of the POS tags, we incorporated their information for every word and its three neighboring positions prior and posterior.

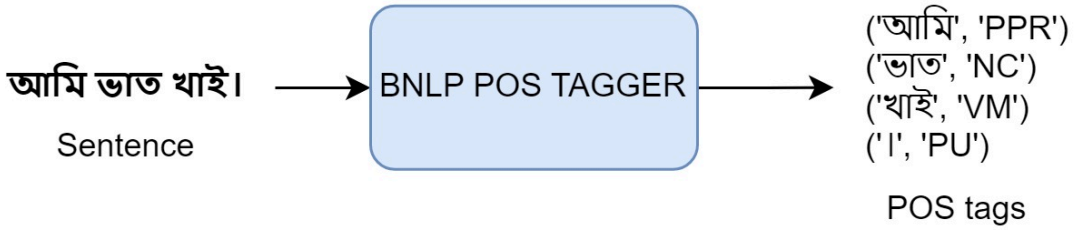


Figure 4.14: Example of POS Tag

BOS and EOS: BOS (Beginning of Sentence) and EOS (End of Sentence) tags serve crucial roles in guiding a Conditional Random Field (CRF) toward identifying named entities accurately. These two special symbols signify the start and end positions of sentences within the text input and provide valuable cues that can inform the CRF's decision-making during the training and inference phases. By encoding local syntax rules through BOS and EOS markers, the CRF inherently learns about common phrase structures associated with various named entities. This allows the model to develop a more sophisticated understanding of how to identify and separate named entities from surrounding text. Additionally, exploiting the spatial distribution patterns of BOS and EOS tags within the input text improves the CRF's interpretative abilities regarding linguistic relationships between phrases.

Gazetteer Information: We used the idea of Gazetteer Lists (discussed in section 3.2.1) to further improve our model. As discussed previously in the Dataset section, the Gazetteer we built is a very big and comprehensive list of PER, LOC, GRP, CORP, CW, and PROD. Utilizing gazetteer entries provides explicit labels for otherwise ambiguous text spans, allowing the CRF to make confident judgments when encountering unseen variations. When incorporating gazetteer entries, the CRF framework becomes aware of existing named entities already documented within the gazetteer's scope. Real-world named entities follow intricate hierarchical structures, reflecting subcategory inclusions, parent-child relations, and other complex organization schemes. Utilizing gazetteers in conjunction with CRFs facilitates learning these interrelationships, allowing the model to make better predictions when confronting less frequent or unusual combinations of entity types. By design, this means the model may avoid misclassifications caused by isolated substrings resembling real names yet lacking any genuine significance in the current context. For implementing this feature, we created is_corp, is_per, is_loc, is_grp, is_prod, is_cw for the current word and its neighboring words. This should drastically help the CRF model to confidently predict words present in the gazetteer list.

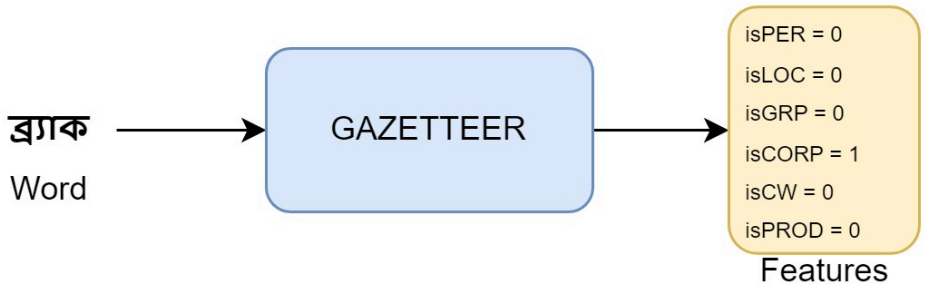


Figure 4.15: Gazetteer as Features

BanglaBERT Large Embeddings Cluster Information: Instead of utilizing raw BanglaBERTLarge Embeddings directly, we adopted a modified strategy tailored to our diverse dataset exhibiting domain shifts and OOV (out-of-vocabulary) words. We anticipated that applying raw BanglaBERTLarge Embeddings might lead to lower robustness against unseen data, hindering the model's generalizability. To address this concern, we decided to employ additional clustering methods to group together semantically related tokens before processing them further. We reasoned that such clusters would be more effective at capturing shared characteristics across similarly named entities, regardless of their exact instantiations. For generating this cluster information, we used the BanglaBERTLarge Embeddings of all the Training data (15,300 sentences) which is of length 1024 for each word. Then we have fit these values to a K-means clustering to group similar words together. The length of the list that was used to fit the K-means algorithm was 2,02.266 which is a huge number. As the words are very diverse and we do not want to lose any significant information, we used a big enough number 1000 for the number of clusters in K-means. Then we predicted this K-means value for each word in our dataset to use as a feature. We trained two different K-Means Clustering Models one with the BanglaBERTLarge's 23rd Layer's Embedding and another with the 24th Layer's Embedding. We then predicted two different clustering values for each of the sentences so that we could use these as features in our CRF model.

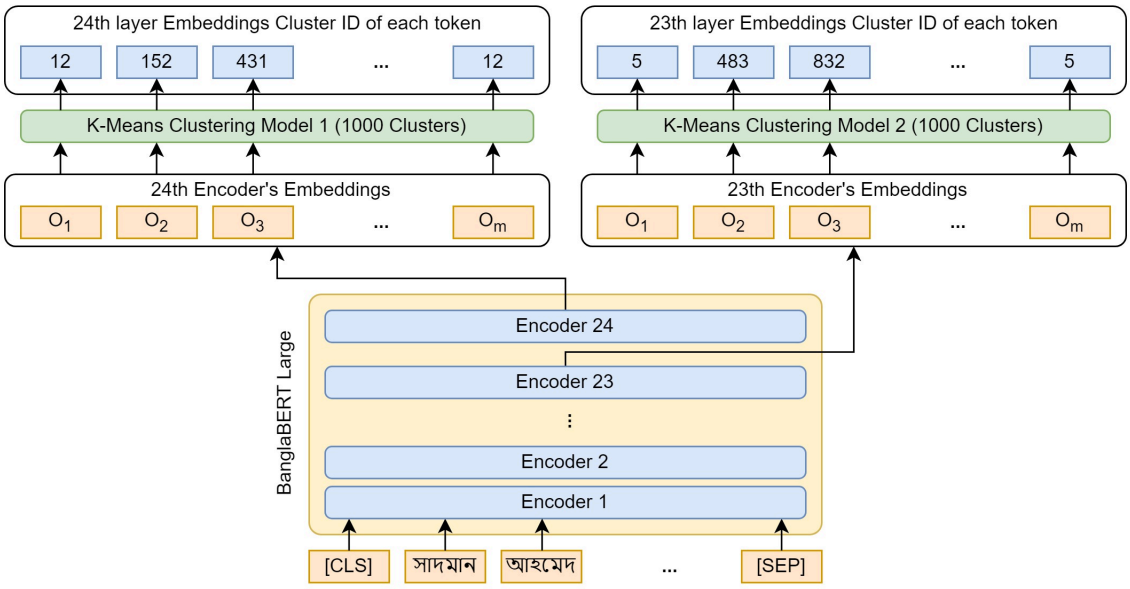


Figure 4.16: BanglaBERT Embeddings with K-means

BanglaBERT Large Softmax Outputs: We also experimented with Fine-tuned BanglaBERT Large's (discussed in section 4.3.5) Softmax outputs as a feature for our CRF model. We anticipate that this may lead to some improvements to our CRF model. However, it's also a possibility that it may worsen the performance of our model if the BanglaBERTLarge's output is not up to the mark. To use this as a feature, we took every sentence of the training dataset and predicted the labels; this prediction gave us Softmax values for each word that was used as the features.

Raw BanglaBERT Large 24th Layer's 1024-Sized Embedding: We wanted to use the BanglaBERT Large model's full potential, so, we also used the raw embedding of BanglaBERT Large as a feature as well. However, the values of the raw embeddings were mostly in the range of 0 to 1. If we train with these values, then the model will be overfitted with train data, so, to overcome this problem, we transformed the embeddings by using the following formula.

$$E_{\text{new}} = R_{\text{round}} \ (E_{\text{raw}} * 100)$$

An example of the converted embedding can be found in table 4.2.

E_{raw}	0.56543	0.56549	-0.353434	1.452232	4.56431
E_{new}	56	56	-35	145	456

Table 4.2: Transforming BanglaBERT Large's Raw Embeddings

We can see, this transformation eliminates the minor differences in embedding values (0.01 scale) which theoretically should help to reduce the overfitting.

4.5.3 CRF Model Strategies

We have discussed in section 4.5.2 all the features that we could engineer from the train data. However, we did not have a single model out of all these features, rather, we experimented with several strategies. These strategies are discussed in table 4.3.

Name	Features (Added feature are in bold)
CRF Model A	Suffix, Prefix, Index, Length
CRF Model B	Suffix, Prefix, Index, Length, **isDigit**, **isPunctuation**, **Frequency**
CRF Model C	Suffix, Prefix, Index, Length, isDigit, isPunctuation, Frequency, **POS**
CRF Model D	Suffix, Prefix, Index, Length, isDigit, isPunctuation, Frequency, POS, **Gazetteer**
CRF Model E	Suffix, Prefix, Index, Length, isDigit, isPunctuation, **isBangla**, **isStopword**, Frequency, POS, Gazetteer
CRF Model F	Suffix, Prefix, Index, Length, isDigit, isPunctuation, Frequency, POS, Gazetteer, **K-means of BanglaBERTLarge Embeddings of 24th Layer**
CRF Model G	Suffix, Prefix, Index, Length, isDigit, isPunctuation, Frequency, POS, Gazetteer, **K-means of BanglaBERTLarge Embeddings of 23rd and 24th Layer**
CRF Model H	Suffix, Prefix, Index, Length, isDigit, isPunctuation, Frequency, POS, Gazetteer, K-means of BanglaBERTLarge Embeddings of 24th Layer, **BanglaBERT Large Softmax Outputs**
CRF Model I	Suffix, Prefix, Index, Length, isDigit, isPunctuation, Frequency, POS, Gazetteer, K-means of BanglaBERTLarge Embeddings 24th Layer, **All BanglaBERT 1024 Layers Information**

Table 4.3: CRF Model Strategies

The flowchart of one of our CRF models is given in figure 4.17.

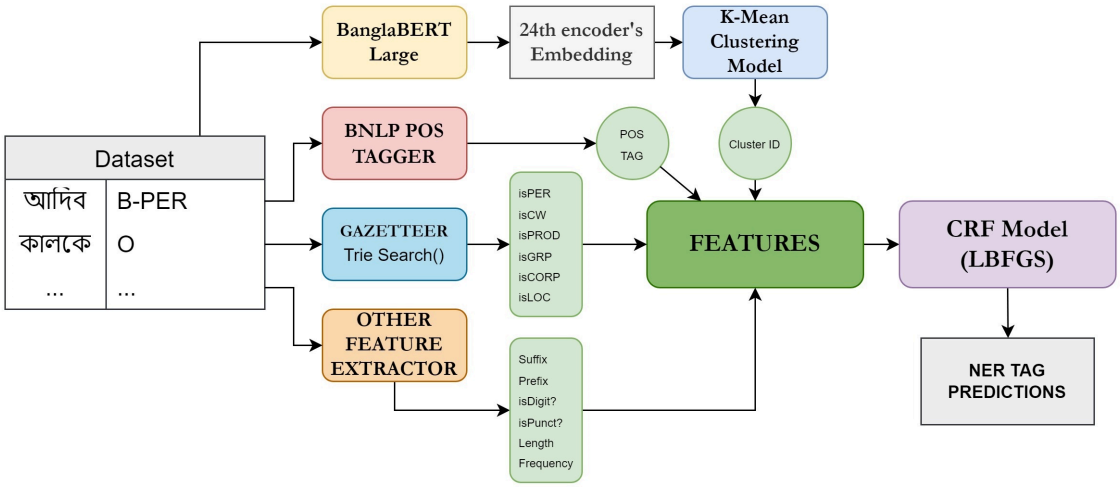


Figure 4.17: CRF Model F

All of our CRF models used the L-BFGS (Limited memory Broyden Fletcher Goldfarb Shanno) algorithm. The *all_possible_transitions* were set to True so that the model generates transition features that associate all the possible label pairs. Both coefficients of *L1* and *L2* regularization were set to 0.1. The maximum number of iterations for optimization algorithms was set to 150.

Chapter 5

Result and Analysis

In this section, we have analyzed our results of our baseline performance in the MultiCoNER 1 dataset at first. Then, we evaluated our BanglaBERT performances with both Default Categorical Cross Entropy Loss Function and Custom Categorical Cross Entropy Loss Function. Later, we have elaborately analyzed the performance of all strategies we experimented with our CRF Model. Finally, we have summarised all of our findings and compared it with other state-of-the-art models.

As the MultiCoNER 1 paper suggested Macro F_1 score to evaluate the dataset, we have evaluated and compared all of the models according to the Macro F_1 score in the test data.

5.1 Baseline Performance

At first, we would like to set our benchmark score by using the Two Layer BiLSTM Network and Single Transformer-Based Network. We also compared our result to the MultiCoNER paper's XLM-RoBERTa (baseline) performance.

Model Names	**Macro-F_1 (Test Data: 133,199 data)**
Two-Layer BiLSTM Network	0.30
Single Transformer-Based Network	0.44
XLM-RoBERTa Baseline	0.391

Table 5.1: Baseline Performances

As we can see in table 5.1, the performance of our baseline performance is very low. As the training data is really insufficient compared to the test data, it is clearly visible that using only a deep learning model without any additional information will not be enough for this dataset. XLM-RoBERTa is pre-trained in 100 languages so it performed a little better as it has some contextual knowledge. Single Transformer-Based Network performed best here as it was a transformer that can understand some context of the text. From this baseline score, the significance of using extra knowledge in this task becomes more apparent.

5.2 BanglaBERT Performance

After setting our baseline benchmarks, we explored a dedicated BERT model for Bangla. At the time of our research, BanglaBERT Base and BanglaBERT Large were published by the CSE department of BUET. As these models showed promising performance in different NLP tasks, we fine-tuned them for our NER task. At first, we used default Categorical Cross Entropy weights and later we experimented with our custom weighted Categorical Cross Entropy function discussed in 4.3.6. The result of this experiment is given in table 5.2.

Model Name	**Macro F_1 (Dev: 800 data)**	**Macro F_1 (Test: 133,199 data)**
BanglaBERT Base	0.4518	0.2710
BanglaBERT Base (Custom Weight LF)	0.6853	0.4858
BanglaBERT Large	0.7778	0.61
BanglaBERT Large (Custom Weight LF)	0.7609	0.5883

Table 5.2: BanglaBERT Performances

The fine-tuned BanglaBERT Large with default Categorical Cross Entropy loss function gets the highest F_1 score (0.61 in test) among the rest of the fine-tuned BanglaBERT models. In this experiment, the implementation of the custom weight loss function has inconsistent results. Our custom loss function impacted positively on the Base model of BanglaBERT by improving 0.2148 of F_1 score. However, it impacted slightly negatively on BanglaBERT Large by reducing 0.0217 score. More experiments need to be done with this custom weight loss function to understand its impact on the other BERT models which is left as future work.

5.3 CRF Performance

As we have discussed in section 4.5.3, we used several strategies for our CRF models. The detailed performance of each model (with added and removed features) can be found in table 5.3.

Model Name	**Added features**	**Removed Features**	**Macro F_1 (Dev)**	**Macro F_1 (Test)**
CRF Model A	Suffix, Prefix, Index, Length		0.671	0.3369
CRF Model B	isDigit, isPunctuation, Frequency		0.6705	0.4379
CRF Model C	POS Tag		0.6829	0.4534
CRF Model D	Gazetteer		0.8103	0.8155
CRF Model E	isBangla, isStopword		0.8221	0.8074
CRF Model F	K-Means of BanglaBERT 24th layer Embedding	isBangla, isStopword	0.815	**0.8267**
CRF Model G	K-Means of BanglaBERT 23rd and 24th layer Embedding		0.8172	0.8216
CRF Model H	BanglaBERT Large Softmax Output	K-Means of BanglaBERT 23rd layer embeddings	**0.8467**	0.8038
CRF Model I	BanglaBERT Large 24th layers 1024 sized embedding		0.8416	0.7398

Table 5.3: CRF Performances

In CRF Model A, we used basic features such as suffix, prefix, index, length. This gave us a score close to the BanglaBERT base model's in dev; however, it performed poorly in test data.

Then we incorporated some new features: isDigit, isPunctuation, and Frequency of the words. This CRF Model B gave us similar results in dev data; however, there was a significant boost of 0.101 F_1 score in the test data. So, we knew these features were quite useful.

After that, we added the POS tags, and this CRF Model C gave us a very slight boost over Model B as we anticipated. There was a minor increase because the POS tagger we used is not trained on a large corpus and was not giving promising POS tags for our sentences.

Later, we added the Gazetteer. CRF Model D showed a drastic increase of 0.1274 F_1 in the dev data and a whopping 0.3621 F_1 increase in the test data. We can clearly see the significance of creating this gazetteer as it massively improved the overall scores. As expected, cooperating knowledge-based solutions like gazetteer have a monumental impact on the performance of the Named Entity Recognition task.

Next, we tried to incorporate some other basic features like isBangla and isStopword. This CRF Model E showed a little performance boost in the dev data; however, in

the test data, the F_1 dropped. It was mainly because the features were not very useful and overfitted the model. So, we dropped this feature for our later model strategies.

We added the K-Means of BanglaBERT Large's 24th layer Embedding the. This CRF Model G did mot improve the dev score; rather the score dropped. But it was the only time the model performed better in the test than the dev data. This clearly showed that this feature is very promising.

As the K-Means of BanglaBERT Large's 24th layer Embedding worked pretty fine, we tried to add another additional layer's K-means value, basically the K-Means of BanglaBERT 23rd layer Embedding; however, in CRF Model G, we saw a performance drop in the test data. So, we did not experiment with additional layers and dropped the K-Means of BanglaBERT 23rd layer embeddings for later model strategies.

Then in CRF Model H, we added the BanglaBERT Large Softmax Outputs. Even though it produced the best result in the dev data, the model performed poorly in the test data. It was mainly due to the performance issue of the BanglaBERT Large itself. As we discussed in section , the macro F_1 score of BanglaBERT Large was only 0.61 in the test data, so, the features gathered from this were not very useful and were accelerating toward bad predictions.

Finally, in our last strategy, we used all 1024 dimensions values of the embedding as features. Similar to CRF Model H, CRF Model I also faced the same problem, doing very well on dev but failing in test data. It was due to having too many diverse sets of words in the test sentences.

In conclusion, CRF Model F showed the best performance in our experiment on this dataset.

5.3.1 What CRF Classifier Learned

In CRF models, transition features represent the learned weights or parameters associated with the transitions between labels in the sequence. These features capture the dependencies between neighboring labels and help model the sequential structure of the data. As CRF Model F, mentioned in section 5.3, showed the best performance, we tried to understand what our model learned.

Top likely transitions:
B-PER -> I-PER 6.360221
B-CW -> I-CW 4.440876
B-LOC -> I-LOC 4.244385
B-PROD -> I-PROD 3.875165
I-LOC -> I-LOC 3.732626
B-GRP -> I-GRP 3.700956
B-CORP -> I-CORP 3.689070
I-CW -> I-CW 3.402518
I-GRP -> I-GRP 3.350253
I-CORP -> I-CORP 3.265144

From the top likely transitions, we can see it has learned pretty well about how the NER works in general. It was mostly trying to transition from a B-tagged entity to an I-tagged entity which is correct. From this, we can also see it was trying to predict the bigger length of LOC, CW, GRP, and CORP which is also correct as they are usually more than 2-3 words.

Top unlikely transitions:
I-GRP -> B-GRP -4.310401
I-PER -> B-PER -4.406159
I-CORP -> B-CORP -4.520391
B-GRP -> B-GRP -4.677539
I-CW -> B-CW -4.903207
I-LOC -> B-LOC -4.948305
I-PROD -> B-PROD -5.381020
O -> I-PROD -6.745918
O -> I-GRP -7.427158
O -> I-CW -7.869977

From the top unlikely transitions, it was trying its best not to predict a B-tagged entity after an I-tagged entity which was correct for NER tasks, as it is not possible. So, our CRF model was working properly. It's also showing there cant be Two staring of GRP and it was trying its best not to predict it. It's also trying not to predict I-tagged entitles after O which is also correct as they need to be either followed by a B-tagged entity or an I-tagged entity.

5.3.2 CRF Model State Features

In a CRF, state features represent the learned weights or parameters associated with individual labels or states in the sequence. These features capture the characteristics or properties of each label independently. As CRF Model F showed the best performance, we tried to understand our model's State Features.

Top positive:
5.176312 O BOS
4.167283 O word.ispunctuation
4.129853 O word[:2]:(
3.825457 B-LOC is_loc
3.812146 B-PROD is_prod
3.274901 O word[:2]:১৯
2.884351 B-CORP is_corp
2.702784 B-PER is_per
2.587418 B-CW word:চিত্রসংগীত
1.453978 B-GRP word:ন্যাটো
1.323634 B-GRP word[-3:]:সংঘ

As we can see, it was trying to memorize BOS as O which mostly is a good thing however it might have some problems with some sentences. We can also see a word being a punctuation is remembered as O which is helping the model so much. The gazetteer's information is also having a massive impact on the model. We can see if a word is is_loc, is_prod, is_corp, or is_per the model is trying to remember it as those represented Tags. This is why the gazetteer had a massive performance boost in the model. We saw it was trying to memorize some words for some entities, which here we can see as helpful however it may have led to some overfitting as well sometimes.

5.4 Result Comparison

We have conducted extensive experiments with different types of models with different strategies. The comprehensive summarized result in test data of every model that are developed by us is presented in table 5.4.

Model name	**Macro F_1 (Test: 133,199 data)**
Two-Layer BiLSTM Network	0.30
Single Transformer-Based Network	0.44
BanglaBERT Base (custom weight)	0.4858
BanglaBERT Large (custom weight)	0.5883
CRF Model F	**0.8267**

Table 5.4: Our Best Models Performances

Among all the models we conducted experiments with, CRF model F performed the best with 0.8267 macro F_1 score in test data. Undoubtedly, the implementation of a knowledge-based approach in corporations to state-of-the-art BERT model's embeddings makes the CRF model provide outstanding performance.

The MultiCoNER dataset was introduced in Task 11 of the SemEval 2022 Competition. A comparison of our best-performing model with other researchers is given in table 5.5.

Team/Model name (Rank in bn dataset)	**Macro F_1 (Test: 133,199 data)**
USTC-NELSLIP (Ranked 1)	0.8424
DAMO-NLP (Ranked 2)	0.8351
CRF model F (Ours)	**0.8267**
NetEase.AI (Ranked 3)	0.6628
BanglaBERT Large	0.61

Table 5.5: Performance Comparison with other state-of-the-art Models

From the table 5.5, we can see that our model outperformed the 3rd Rank in the competition by a huge margin. It performed really well and placed very near to the Rank 1 and Rank 2 teams.

A complete overview of all the models we experimented with and compared with is given in figure 5.1.

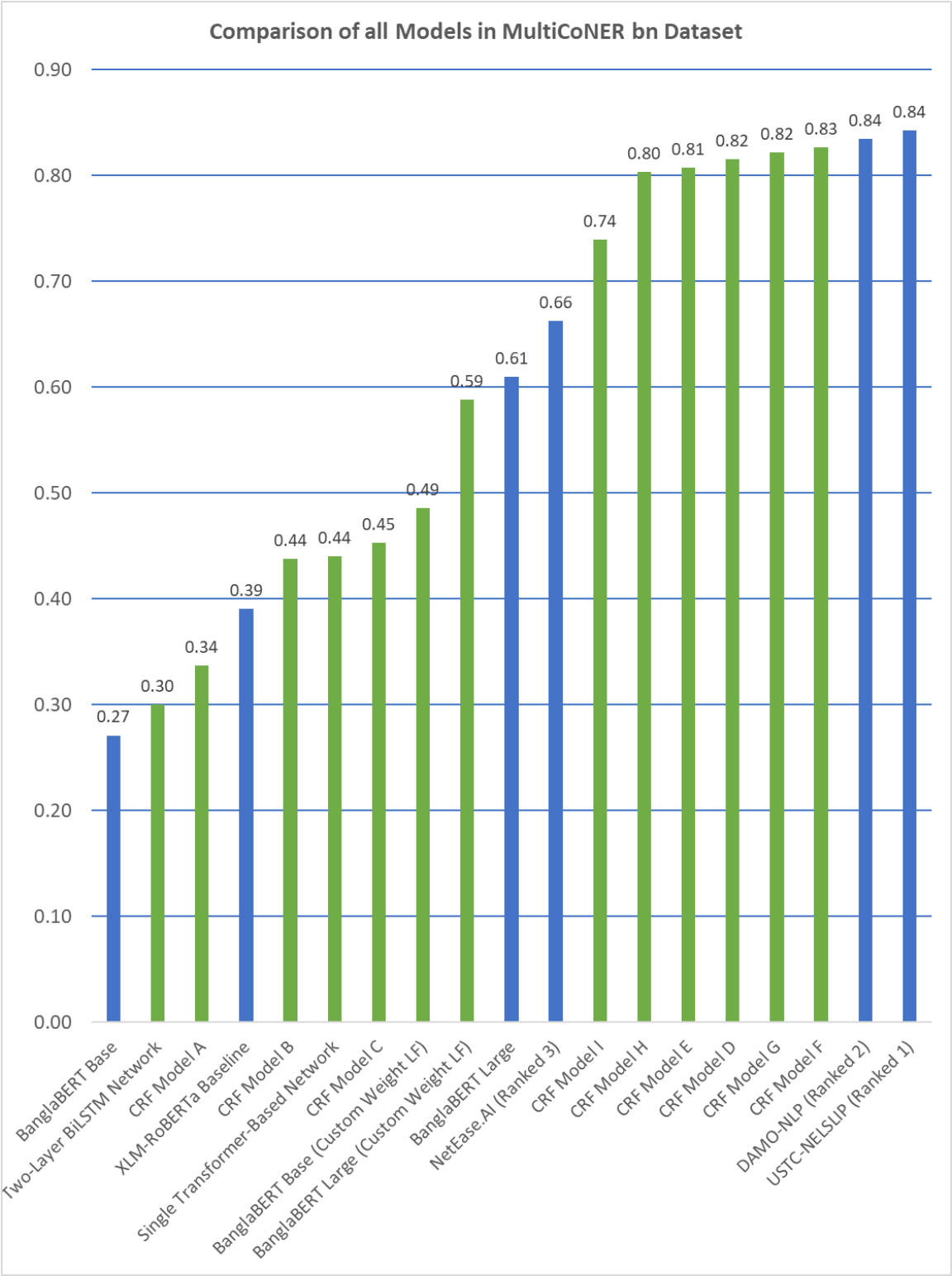


Figure 5.1: Comparion of all Models in MultiCoNER 1 Dataset

Chapter 6

Conclusion

In this research work, we explored the current trends of Bangla Named Entity Recognition tasks. As Bangla is a low-resource language, limited work was done. We had chosen MultiCoNER's Bangla dataset which is considered as one of the toughest. We conducted experiments with available models and tried to improve them. We proposed a custom loss function to improve BanglaBERT base model performance. Furthermore, we created a Gazetteer containing over 96 thousand entities. The implementation of the Gazetteer showed drastic improvement. Finally, we proposed a complex CRF model which uses the embeddings of the final layer in BanglaBERT with Gazetteer and other features which show promising state-of-the-art performance in Bangla NER tasks. For future work, we would like to explore the custom weights for categorical cross entropy loss with other BERT models in other datasets and more features for our proposed CRF model.

Bibliography

[1] Ben Goodey. *Machine learning vs rule-based NLP*. June 2021. URL: https://www.sentisum.com/success-article/machine-learning-nlp.

[2] Ethnologue. *The Ethnologue 200: What are the top 200 most spoken languages?* [Online]. Available: https://www.ethnologue.com/guides/ethnologue200. [Accessed: 18-Sep-2022]. June 2022. URL: https://www.ethnologue.com/guides/ethnologue200.

[3] 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/.

[4] 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.

[5] 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.

[6] 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.

[7] I. Ashrafi et al. "Banner: A cost-sensitive contextualized model for Bangla named entity recognition." In: *IEEE Access* 8 (2020), pp. 58206–58226.

[8] 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.

[9] S. A. Chowdhury, F. Alam, and N. Khan. "Towards Bangla named entity recognition." In: *2018 21st International Conference of Computer and Information Technology (ICCIT)*. 2018.

[10] M. Tanvir Alam and M. Mofijul Islam. "Bard: Bangla Article Classification using a new comprehensive dataset." In: *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*. 2018.

[11] Xinyu Wang et al. *DAMO-NLP at SemEval-2022 Task 11: A Knowledge-based System for Multilingual Named Entity Recognition*. 2022. arXiv: 2203.00545 [cs.CL].

[12] Abhik Bhattacharjee et al. “BanglaBERT: Language Model Pretraining and Benchmarks for Low-Resource Language Understanding Evaluation in Bangla.” In: *Findings of the Association for Computational Linguistics: NAACL 2022*. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 1318–1327. DOI: 10.18653/v1/2022.findings-naacl.98. URL: https://aclanthology.org/2022.findings-naacl.98.

[13] 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.

[14] Shervin Malmasi et al. “SemEval-2022 Task 11: Multilingual Complex Named Entity Recognition (MultiCoNER).” In: *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 1412–1437. DOI: 10.18653/v1/2022.semeval-1.196. URL: https://aclanthology.org/2022.semeval-1.196.

[15] Leon Derczynski et al. “Results of the WNUT2017 Shared Task on Novel and Emerging Entity Recognition.” In: *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 140–147. DOI: 10.18653/v1/W17-4418. URL: https://aclanthology.org/W17-4418.

[16] Tri Nguyen et al. “MS MARCO: A Human Generated MAchine Reading COmprehension Dataset.” In: *CoRR* abs/1611.09268 (2016). arXiv: 1611.09268. URL: http://arxiv.org/abs/1611.09268.

[17] Nick Craswell et al. “ORCAS: 18 Million Clicked Query-Document Pairs for Analyzing Search.” In: *CoRR* abs/2006.05324 (2020). arXiv: 2006.05324. URL: https://arxiv.org/abs/2006.05324.

[18] Beiduo Chen et al. “USTC-NELSLIP at SemEval-2022 Task 11: Gazetteer-Adapted Integration Network for Multilingual Complex Named Entity Recognition.” In: *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. Association for Computational Linguistics, 2022. DOI: 10.18653/v1/2022.semeval-1.223. URL: https://doi.org/10.18653%5C%2Fv1%5C%2F2022.semeval-1.223.

[19] 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.

[20] Ashish Vaswani et al. “Attention Is All You Need.” In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[21] Kevin Clark et al. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators*. 2020. arXiv: 2003.10555 [cs.CL].

[22] Sagor Sarker. “BNLP: Natural language processing toolkit for Bengali language.” In: *CoRR* abs/2102.00405 (2021). arXiv: 2102.00405. URL: https://arxiv.org/abs/2102.00405.