ANEC: An Amharic Named Entity Corpus and Transformer Based Recognizer

EBRAHIM CHEKOL JIBRIL AND A. CÜNEYD TANTUĞ

Faculty of Computer Engineering and Informatics, Istanbul Technical University, 34469 Istanbul, Turkey

Corresponding author: Ebrahim Chekol Jibril (jibril@itu.edu.tr)

The work of Ebrahim Chekol Jibril was supported by the Turkcell-Istanbul Technical University Researcher Funding Program.

ABSTRACT

Named Entity Recognition is an information extraction task that serves as a pre-processing step for other natural language processing tasks, such as machine translation, information retrieval, and question answering. Named entity recognition enables the identification of proper names as well as temporal and numeric expressions in an open domain text. For Semitic languages such as Arabic, Amharic, and Hebrew, the named entity recognition task is more challenging due to the heavily inflected structure of these languages. In this study, we annotate a new comparatively large Amharic named entity recognition dataset and make it publicly available. Using this new dataset, we build multiple Amharic named entity recognition systems based on recent deep learning approaches including transfer learning (RoBERTa), and bidirectional long short-term memory coupled with a conditional random fields layer. By applying the Synthetic Minority Over-sampling Technique to mitigate the imbalanced classification problem, our best performing RoBERTa based named entity recognition system achieves an f1-score of 93%, which is the new state-of-the-art result for Amharic named entity recognition.

INDEX TERMS

Amharic, named entity recognition, synthetic minority over-sampling technique, deep learning, BiLSTM-CRF, transfer learning.

ACRONYMS

ANERS Amharic Named Entity Recognition System.
BERT Bidirectional Encoder Representations from Transformers.
BiLSTM Bidirectional Long Short-Term Memory.
BiLSTM-CRF Bidirectional Long Short-Term Memory-Conditional Random Fields.
CBOW Continuous Bag of Words Model.
CRF Conditional Random Fields.
ELRC Ethiopian Languages Research Center.
HMM Hidden Markov Models.
IOB Inside-Outside-Beginning.
LOC Location.
LSTM Long Short-Term Memory.
ML Machine Learning.
NE Named Entity.
NER Named Entity Recognition System.
NLP Natural Language Processing.
ORG Organization.
OOV Out of Vocabulary.
PER Person.
POS Part-of-Speech.
QA Question Answering.
RNN Recurrent Neural Networks.
RoBERTa Robustly Optimized BERT Pretraining Approach.
SVM Support Vector Machine.
SMOTE Synthetic Minority Over-sampling Technique.
SERA System for Ethiopic Representation in ASCII.
TTL Title.

The associate editor coordinating the review of this manuscript and approving it for publication was Sergio Consoli.
I. INTRODUCTION

Named Entity Recognition (NER) is a subtask of information extraction, which extracts and classifies specific predefined types of entities, which may be proper names, numerical and temporal expressions. Usually, person, location, and organization names are considered proper names in most of the studies. Numerical expressions usually cover numeral, money, and percentage expressions, while date and time expressions are classified as temporal expressions [32]. The NER task is a non-trivial task since using simple lookup lists to capture these expressions in the running text is insufficient due to the infinite number of elements in these types. It is not feasible to construct a large set involving all possible people names that can occur in a text [55]. The same fact is also valid for the location and organization names. For this Named Entity (NE) classes, we need a proper way of identifying NEs according to the context, usually by machine learning techniques. On the other hand, rule-based systems generally perform quite well in the temporal and numerical expressions identification tasks.

A NER system is usually a pre-processing step within a large Natural Language Processing (NLP) system. The quality of the NER system has a direct impact on the performance of the overall NLP system. Many research efforts have been conducted to prove the importance of NER to other natural language processing tasks such as knowledge graph, decision-making support system and question answering (QA) [31], [75]. The majority of the considered questions expect a named entity or a list of named entities as answers. Greenwood and Gaizauskas [31] use a NER system to improve the performance of the answer extraction module based on a pattern matching approach. The authors use the NER system to capture the answers, where pattern matching methods are not viable. They report that the NER system improved the accuracy of answering the questions.

In addition to the aforementioned NLP tasks, using NER as a pre-processing step also improves the performance of the search result clustering task. In search result clustering, using a NER system before comparing the contents of the documents has been proven to be fruitful [73]. In machine translation, Babych and Hartley [6] show that using NER systems as a pre-processing task improves the quality of the translation result.

Traditional sequence labeling models are linear statistical models, including Hidden Markov Models (HMM) and Conditional Random Fields (CRF) [62], [56], [45], which rely heavily on hand-crafted features and task-specific resources. For example, English NER benefits from carefully designed word spelling features; orthographic features, and external resources such as gazetteers. However, such task-specific knowledge is usually expensive to develop, making sequence labeling models difficult to adapt to new tasks or domains.

In the past years, non-linear neural networks with word embeddings have been broadly applied to NLP problems with great success. Collobert et al. [14] proposed a simple but effective feed-forward neural network that independently classifies labels for each word by using contexts within a fixed-size window. Recurrent neural networks together with their variants such as long-short term memory [34] have shown great success in modeling sequential data. Several recurrent neural networks (RNN) based neural network models have been proposed to solve sequence labeling tasks like speech recognition [30], part-of-speech (POS) tagging [36], and NER [13], [36]. A transformer is one of the most commonly used neural network architectures in natural language processing. The transformer architecture consists of stacked transformer layers, each of which takes a sequence of vectors as input and outputs a new sequence of vectors with the same shape [74]. Recently, transformer-based architectures are shown to achieve better performance against traditional models for NER and question answering tasks [19], [44], [75].

Most NER studies conducted for widely spoken languages such as English, German, Chinese and Arabic benefit from the large datasets to acquire high performance. However, the NER studies for low resource languages such as Amharic are still using small datasets. One of the major contributions of this work is constructing a relatively large Amharic NE dataset by manually annotating entities and making it publicly available for researchers. Our new Amharic NER dataset is almost twice the only publicly available SAY dataset, in terms of both sentence and token counts. Additionally, another main motivation of this study is building a state-of-the-art Amharic NER system by employing recent deep learning models trained on this new dataset.

The rest of the paper is structured as follows. The second section of the paper will give brief information about Amharic and an overview of the Amharic language peculiarities. Section three describes previous works related to the Amharic NER task. Section four is dedicated to giving a brief description of the model. Details about the evaluation data used in the experiments are given in Section five. The results of the experiments with RoBERTa and Bidirectional Long Short-Term Memory- Conditional Random Fields (BiLSTM-CRF) and a comparison with previous work results are presented in Section six. Finally, in Section seven, we draw some conclusions and discuss future works.

II. THE AMHARIC LANGUAGE

Amharic is a member of the Semitic language family, such as Arabic, Syriac, and Hebrew [54]. The language is spoken by more than 50 million people as their mother language and over 100 million as a second language in Ethiopia [50], [71]. Amharic is the second most spoken Semitic language, after Arabic [23]. It is also the second most spoken language in Ethiopia and one of the five most widely spoken languages on the African continent. The Amharic language is mainly spoken in Ethiopia and Eritrea. It is also the working language of the federal government of Ethiopia and also non-government organizations as well as private institutions.

Amharic uses an abugida writing system that evolved from the Ge’ez scripts known as Fidäl, which is now only used in the Ethiopian Orthodox Church. The Amharic language
writing system is composed of a total of 238 characters, among which are combinations of the 33 basic characters and the special character (ǐ) (“V”), which is a total of 34 core characters. Each of these core characters occurs in 7 forms (orders); one basic form and six non–basic forms representing syllable combinations consisting of a consonant and following vowel. The non-basic forms are derived from the basic forms by generally regular modifications (see Figure 1). For example, the second-order characters are formed mostly by attaching strokes to the right of the character [5].

![Character formation.](image)

In addition to the set of these 238 characters, there are 50 labialized characters, 9 punctuation marks, and 20 numerals. These bring the total number of characters in the script to 317 [7], [8].

Not all the letters of the Amharic script are strictly necessary for the pronunciation patterns of the spoken language; some were simply inherited from Ge’ez without having any semantic or phonetic distinction in modern Amharic. Most of the labialized consonants, which are simply inherited from Ge’ez, are redundant. The language also has its own unique set of punctuation marks (“/word separator”, “/full stop”, “/comma”, “/colon”, “/semicolon”, “/preface-colon”, “/question mark” (It is not used anymore), “/section marker”, “/paragraph separator”). Amharic, unlike other Semitic languages, is written from left to right.

A. THE CHALLENGES OF NER IN THE AMHARIC LANGUAGE

Conducting a NER study in Amharic necessitates dealing with some challenges, mostly due to its rich morphological structure and unique orthography [9].

1) LACK OF CAPITALIZATION

In the Amharic language, there is no capitalization when writing proper nouns. As in English and most European languages, proper nouns are written by making the initial letter of the word uppercase, which provides strong evidence for the identification and classification of named entities.

From a general viewpoint, the NER task can be considered as a composition of two sub-tasks: First, the detection of the existing NEs in a text, which is an easy sub-task if we can use the capital letters as indicators to determine where the NEs start and end. However, this is trivial only when the target language supports capital letters, which is not the case for the Semitic language family (Amharic, Arabic, and others). Figure 2 shows the example of two words where only one of them is a NE but both of them start with the same character (example sentence along with its English translation).

In the example sentence 1 ለም Portland is a named entity, whereas in example sentence 2 it is not. The absence of capital letters in the Amharic language seems to be a major obstacle to obtain high performance in Amharic NER. The same problem is also exhibited even in languages that support capital letters for social media and informal texts where capitalization is generally ignored [27].

2) THE AGGLUTINATIVE MORPHOLOGY

The Amharic language, like Arabic, has a highly agglutinative morphology in which a word may have prefixes, lemma, and suffixes in different combinations resulting in a very complicated morphology. For instance, a person name like “አስ-

Similar to Arabic, Turkish and other languages, Amharic is a highly inflectional language, where a surface word is constructed with the prefix(es), a lemma, and suffix(es), though prefix(es) and suffix(es) are optional. A prefix can be an article, a preposition, or a conjunction, whereas a suffix is generally an object or a personal/possessive anaphora. Both prefixes and suffixes are allowed to be combinations, and thus a word can have zero or more affixes. Compared to texts written in other languages whose morphologies are not complex, this inflectional and derivational characteristic of the language makes Amharic texts sparser, and thus most of the Amharic NLP tasks are harder and more challenging. However, concerning the classification subtask of NER, we can say that the classification of NEs relies mainly on the word form and the context in which they appeared in the text to decide the class they belong to.

3) ORTHOGRAPHIC VARIATION

Sometimes, an Amharic word may have different orthographies with the same pronunciation, still referring to the same word. For example, “አማር/find at -year” and “አማር/amat -year” and also “አማር/is-maquí -mosque” and “አማር/is-maquí -mosque” have the same meaning and pronunciation but spelled differently. Orthographic variation increases the number of out-of-vocabulary words, which
deteriorates the quality of word embeddings. In addition to this, it increases the amount of unseen data in the training corpus.

### III. RELATED WORKS

In this section, we present recent and important studies on NER with a focus on works that concentrate on Semitic languages. Most of the NER studies on Semitic languages are mainly in Arabic and partly in Hebrew. Early studies on Arabic NER are mainly based on handcrafted rules [47], [67]. Machine learning techniques, however, have played an important role in moving NER research forward by providing different learning methods such as the Hidden Markov Model (HMM), conditional random fields, maximum entropy, support vector machines, and deep neural networks.

Most of the current named entity recognition research utilizes machine learning approaches that mainly depend on the availability of large-scale training data [13], [36], [46], which is usually very hard to access for low-resource languages. In particular, most NER efforts have focused on a few European and Asian languages, while African languages have been given little attention. Only seven studies of NER on Amharic have been found in the literature [1], [3], [9], [17], [49], [69], [68]. In these Amharic NER studies, two NER datasets compiled from different sub-sets of the Walta Information Center Corpus [16] are used. In addition to the Walta Information Center corpus, there is also the Adelani [1] dataset and Sikdar and Gambäck [68] New Mexico State University Computing Research Laboratory dataset, which is annotated for the SAY project. The data is annotated with 6 classes (PER, LOC, ORG, TIME, TTL, and O-other) and it is available on GitHub.1 Sikdar and Gambäck [68] employed a stack-based deep learning approach incorporating various semantic information sources that were built using an unsupervised learning algorithm with word2Vec, feature vector, and one-hot vector extracted by using a CRF classifier. They have reported that the stack-based approach outperformed other deep learning algorithms. Sintayehu and Lehal [69] applied a graph-based label propagation algorithm for 6 classes (PER, LOC, ORG, DATE, MONEY, and O-other). The researchers compared expectation maximization with semi-supervised learning approaches. The experiment reveals that label propagation-based NER achieves superior performance compared to expected maximization using a few labeled training datasets. Table 1 presents the methods and performances of the previous Amharic NER studies.

All of the aforementioned Amharic NER studies extracted subsets of randomly selected sentences containing at least one named entity from the Walta Information Center Corpus, and these extracted sentences are manually annotated with 4-class named entity tags for persons, organizations, locations, and others (non-NE). As a result, each study trains the machine learning models on a different training dataset and evaluates their performance on a separate dataset. Consequently, the lack of a standard benchmark test dataset limits a comparable evaluation of these previous studies.

Both Mehamed [48] and Alemu [3] used conditional random fields [40] classifiers trained on different word and context features (word prefixes and suffixes, and the NE and part-of-speech tags of the word), using about 90% of their data for training and the remaining 10% for testing (without cross-validation). Mehamed [48] achieved recall, precision and f1-score values of 75.0%, 74.2%, and 74.6%, respectively, for the NER task on a Walta Information Center corpus subset consisting of 10,405 words, of which 961 were used for testing. Within the test data, only 96 named entities exist.

| Author         | Method      | Corpus                  | Entities                  | f1-score |
|---------------|-------------|-------------------------|---------------------------|----------|
| Mehamed [48]  | Conditional Random Field | Custom Corpus (18,299 tokens) | Person, Organization, Location | 75.0%    |
| Alemu [3]     | Conditional Random Field | Alemu[3] (13,538 tokens)   | Person, Organization, Location | 80.7%    |
| Belay [9]     | BiLSTM      | Alemu[3] (13,538 tokens) | Person, Organization, Location | 96.1%    |
| Demissie [17] | Semi-Supervised | Custom Corpus (82,870 tokens) | Person, Organization, Location | 92.6%    |
| Sintayehu and Lehal [69] | Stack-based LSTM | SAY Corpus (109,676 tokens) | Person, Organization, Location, Date, Money | 74.26%    |
| Adelani [1]   | XLM-R-base | Own corpus (59,531 tokens) | Person, Organization, Location, Date | 70.96%    |

Alemu [3] stated that the data used by [48] is not accessible, so the researcher experimented on another 13,538-word subset of the Walta Information Center corpus, of which 1,242 words were used for testing. Using context windows of up to two words before and after the current token, Alemu [3] achieved a recall of 84.9% and precision of 76.8% and 80.7% to f1-score. The small size of the test datasets makes the evaluations in both pieces of researches unreliable, as stated by the authors in their publications. Mehamed [48] found that part-of-speech tagging improved the results while using word prefixes did not. On the other hand, Alemu [3] claimed that word prefixes contributed positively, while part-of-speech tagging did not. However, they both agreed that context and word suffixes were useful features.

Belay [9] included all features previously used and utilized a combination of decision trees, support vector machines, and hand-crafted rules on the dataset created by [3], and artificially expanded it to get a better balance the classes by using

1 https://github.com/geezorg/data/tree/master/amharic/dictionaries/nmsu-say/
the synthetic minority over-sampling technique (SMOTE). They use two rules that classify words into NE types based on the occurrence of trigger words in a given sentence. Using a list of trigger words that appear before and after NEs, searching at the beginning of the sentence like personal names and searching at the end of the sentence like organization names. To balance the datasets, this work expanded the other classes (person, location, and organization) so that the actual training dataset contained 31,347 tokens and then apparently tested on 40% of the same data as used for training. Demissie [17] used word vectors as features instead of manually designed features and tested the automatically extracted word vector features with different classifiers (SVM, J48, LSTM, and BiLSTM). To generate the word vectors, the Walta Information Center website archive, which contains 611,294 untagged tokens, is used. Demissie [17] trained the classifier with the same datasets as [9]. Both Belay [9] and Demissie [17] applied SMOTE to balance classes. Demissie [17] split the datasets into 80% for training and 20% for testing purposes. Their experiments showed that word vector features could substitute manually designed features while maintaining high performance for Amharic NER. Compared to the SVM, the score obtained by BiLSTM is 2.9% lower. The justification for these results is the small number of datasets used for BiLSTM (deep neural networks require large datasets), testing datasets used for BiLSTM are two-fold of SVM and the last reason was the parameters used for the network may not be optimized properly.

Recently it is common to use a hybrid NER architecture by using two or more different algorithms like rule-based and SVM [9], rule-based and decision tree Belay, LSTM and CRF [21], [36], [53], [76]. These models achieved state-of-the-art for different languages like English, Arabic, Turkish, etc. However, such kind of a hybrid model is not tested for Amharic.

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based model that depends on pre-training to learn from the raw corpus, and fine-tuning on downstream tasks including NER task [19]. RoBERTa is an improved version of BERT, which is trained better, longer, and with more data. It removes the next sentence prediction task during the pre-training stage, compared with BERT [44]. These transformer-based model achieved the current state of the art for different languages such as English [19], [44], German [39], Arabic [4], [10], Chinese [15], [43], etc. Therefore, developing ROBERTa model and testing with relatively large dataset will improve the performance of the Amharic NER system.

A. IMBALANCED DATASETS
Predictive classifiers generally suffer from imbalanced training datasets where the number of examples in the dataset for each class label is not balanced. An imbalance on the order of 102 to 1 is prevalent in fraud detection and an imbalance of up to 105 to 1 has been reported in other applications [58]. In some real-world classification tasks like the one above, the unusual or interesting class is rare among the general population, and the class distribution is imbalanced [38], [25]. An imbalanced training dataset is challenging for predictive classifiers since the model cannot be trained on the proper number of examples to learn the discriminative characteristics of the examples in the minority class. A feed-forward neural network trained on an imbalanced dataset may not be able to learn to discriminate enough between classes [61].

There have been attempts to deal with imbalanced datasets in domains such as fraudulent telephone calls, telecommunications management, text classification, and detection of oil spills in satellite images. Many researchers have addressed the issue of class imbalance in two ways. One is to assign distinct costs to training examples, and the other is to re-sample the original dataset, either by under-sampling the majority class and/or oversampling the minority class. The problems of unequal error costs and uneven class distributions are related. It has been suggested that, for training, high-cost instances can be compensated for by increasing the dataset [24].

A naive approach to oversampling is to duplicate examples in the minority class. However, these additional examples are not capable of introducing new information to the model. As an alternative oversampling approach, new examples can be synthesized from the existing examples. This type of data augmentation for the minority class is named SMOTE. SMOTE is a means of increasing the sensitivity of a classifier to the minority class by balancing the number of instances between the classes. SMOTE generates synthetic examples by operating in feature space rather than data space. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples using nearest neighbors algorithms. Depending upon the amount of oversampling required, neighbors from the k nearest neighbors are randomly chosen. The SMOTE implementation used five nearest neighbors. Synthetic samples are generated in the following way: Take the difference between the feature vector (sample) under consideration and its nearest neighbor. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general [11]. The algorithm of SMOTE, taken from [11], is presented in Appendix A.

B. EVALUATION METRICS
To evaluate the performance of NER task, different types of named entity evaluation metrics are used in the literature. The frequently used metrics are usually proposed in NER-related conferences and these metrics are usually referred to by the name of the corresponding conference. The MUC, CoNLL and SemEval metrics are the most frequently used metrics and they are introduced shortly in the subsequent subsections.
1) CoNLL METRIC
The Language-Independent Named Entity Recognition task introduced at CoNLL-2003 [64] measures the performance of the systems in terms of precision, recall and f1-score, where: Precision is the percentage of named entities correctly found by the system. Recall is the percentage of named entities present in the corpus that are found by the system. A named entity is correct only if it is an exact match of the corresponding entity in the data file [72].

\[
\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}} \quad (1)
\]

\[
\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}} \quad (2)
\]

\[
F_1 = \frac{2(\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \quad (3)
\]

2) MESSAGE UNDERSTANDING CONFERENCE (MUC) METRIC
MUC considered different categories of errors. It is defined in terms of comparing the response of a system against the golden annotation [12]:

Correct (COR): both are the same.
Incorrect (INC): the output of a system and the golden annotation do not match.
Partial (PAR): system and the golden annotation are somewhat “similar” but not the same.
Missing (MIS): a golden annotation is not captured by a system.
Spurious (SPU): system produces a response, which doesn’t exist in the golden annotation.

\[
\text{Recall} = \frac{\text{correct} + (0.5 \times \text{partial})}{\text{Possible}} \quad (4)
\]

\[
\text{Precision} = \frac{\text{Correct} + (0.5 \times \text{partial})}{\text{Actual}} \quad (5)
\]

\[
f1 = \frac{2(\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \quad (6)
\]

Possible variable contains the tallies of the number of slot fillers that should be generated. It is the sum of the correct, partial, incorrect, and missing. Actual is the number of fillers that the system under evaluation actually generated, which is the sum of the correct, partial, incorrect, and spurious. Based on this they calculated Precision, Recall and f1-score;

3) INTERNATIONAL WORKSHOP ON SEMANTIC EVALUATION (SemEval)
The evaluation metrics should score if a system can identify the exact entity (regardless of type) and it can assign the correct NE class regardless of the boundaries [66]. The SemEval’13 introduced four different ways to measure precision, recall, and f1-score results based on the metrics defined by MUC.

Strict: exact boundary surface string match and NE class
Exact: exact boundary match over the surface string, regardless of the type

Partial: partial boundary match over the surface string, regardless of the type
Type: some overlap between the system-tagged entity and the gold annotation is required.

To calculate precision and recall, it used the scoring categories proposed by MUC (COR, INC, PAR, MIS, and SPU) in different ways. For both the boundaries and the type, the following measure is calculated:

\[
\text{COR}: \text{The number of correct answers}
\]

\[
\text{Possible(POS)} = \text{COR} + \text{INC} + \text{PAR} + \text{MIS} = \text{TP} + \text{FN} \quad (7)
\]

The total number of annotations produced by the system:

\[
\text{Actual(ACT)} = \text{COR} + \text{INC} + \text{PAR} + \text{SPU} = \text{TP} + \text{FP} \quad (8)
\]

After that, it computed precision, recall, and f1-score. The computation is made in two different ways depending on whether we want an exact match (i.e., strict and exact) or a partial match (i.e., partial and type) scenario. Exact Match (i.e., strict and exact):

\[
\text{Precision} = \frac{\text{COR}}{\text{ACT}} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (9)
\]

\[
\text{Recall} = \frac{\text{COR}}{\text{POS}} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (10)
\]

Partial Match (i.e., Partial and Type)

\[
\text{Precision} = \frac{\text{COR} + (0.5 \times \text{PAR})}{\text{ACT}} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (11)
\]

\[
\text{Recall} = \frac{\text{COR} + (0.5 \times \text{PAR})}{\text{POS}} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (12)
\]

Table 2 shows an example calculation of both CoNLL and MUC metrics along with the MUC error categories. Exact Match (CoNLL):

\[
\text{Precision} = \frac{\text{COR}}{\text{ACT}} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{3}{6} = 0.5
\]

\[
\text{Recall} = \frac{\text{COR}}{\text{POS}} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{3}{6} = 0.5
\]

\[
f1\text{-score} = 0.5
\]

Partial Match (MUC):

\[
\text{Precision} = \frac{\text{COR} + (0.5 \times \text{PAR})}{\text{ACT}} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{4}{6} = 0.66
\]

\[
\text{Recall} = \frac{\text{COR} + (0.5 \times \text{PAR})}{\text{POS}} = \frac{\text{COR}}{\text{ACT}} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{4}{6} = 0.66
\]

\[
f1\text{-score} = 0.66
\]
IV. MODEL

In this section, we describe the architecture of our Amharic NER models (BiLSTM-CRF and Amharic RoBERTa). In the BiLSTM-CRF model: - first define the input/output representation. Next, we provide information on LSTM and BiLSTM. Finally, we explain how the CRF-based decoder model is built. In Section B, we explain the Amharic RoBERTa model.

A. BiLSTM-CRF MODEL

1) INPUT REPRESENTATION

This section describes character embeddings and word embeddings.

a: CHARACTER EMBEDDING

Sub-word units play a substantial role in word representations, specifically in languages with complex morphological structures. Character-level representations have been found useful for morphologically rich languages and to handle the out-of-vocabulary problem. Learning character-level embeddings has the advantage of learning representations specific to the task and domain [41]. Extracting the prefix and suffix information from a surface requires a morphological analysis step. Embeddings are used instead of hand-engineering prefixes and suffixes information about words. Instead, we opt to rely on the word vector representation of the word formed by a BiLSTM which accepts the characters of the word as input. A randomly initialized character look-up table contains an embedding for every character. The character embeddings belonging to every character in a word are processed in both left-to-right and right-to-left directions by the BiLSTM Layer (Figure 3). The final character-level representation of a word is the concatenation of its forward and backward representations from the BiLSTM Layer. This character-level representation is also concatenated with a word-level representation, which is described in the next subsection.

b: WORD EMBEDDING

Word embeddings are vector representations of words that allow words with similar meanings to be represented as vectors close to each other in the vector space [52]. Since learning too many parameters from limited data is difficult, obtaining word embeddings from a large unlabeled corpus is a widely used technique. Instead of randomly initializing embeddings, using pre-trained embeddings has improved the performance of neural network architecture [40].

Demissie [17] obtained their word embeddings from scratch by training on an unlabeled dataset using the continuous bag of words model as described in [52]. In addition to training word embeddings from scratch, we also investigated the effect of using pre-trained word embeddings in our study. We opt for utilizing fastText2 Amharic word embedding model which has an embedding dimension of 300. Unfortunately, the word embeddings model used in [17] is not publicly available and not accessible, so, we have trained a new word embedding model from scratch for a fair comparison with the fastText pre-trained embedding model.

2) LSTM

Recurrent neural networks are shown to be able to achieve high performance in many natural language processing tasks, such as language modeling [51], parsing [22], and machine translation [70]. One major problem with simple RNNs is that they are difficult to train for long-distance dependencies due to the vanishing and exploding gradient problems. Hochreiter and Schmidhuber [34] proposed long short-term memory to overcome the long-term dependency problem. They introduced a special memory cell, which is controlled by input, output, and forget gates. The input gate controls how much new information should be added to the current cell. The forget gate controls what old information should be deleted. The output gate controls the information flow from the cell to the output. Long short-term memory has emerged as an effective and scalable model for several learning problems related to sequential data. LSTMs are general and effective at capturing long-term temporal dependencies. They do not suffer from the optimization difficulties that plague simple recurrent networks [33] and have been used to advance the state-of-the-art for many difficult problems.

The LSTM memory cell is defined by the following equations:

\[
f_t = \sigma(W_f x_t + W_{fh} h_{t-1} + w_f c_{t-1} + b_f) \quad (13)
\]

\[
i_t = \sigma(W_i x_t + W_{ih} h_{t-1} + w_i c_{t-1} + b_i) \quad (14)
\]

\[
c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + W_{ch} h_{t-1} + b_c) \quad (15)
\]

\[
o_t = \sigma(W_o x_t + W_{oh} h_{t-1} + w_o c_t + b_o) \quad (16)
\]

\[
h_t = o_t \odot \tanh(c_t) \quad (17)
\]

where \(\sigma\) is an element-wise logistic sigmoid function, \(\tanh\) is the hyperbolic tangent function and \(\odot\) is an element-wise product, \(W_s\)’s are weight matrices, and \(b_s\)’s are biases. \(f, i\) and \(c\) are the forget, input and output gates respectively, \(c\) denotes the cell vector, and \(h\) is the hidden state vector. All gate vectors and the cell vector have the same dimensionality as the hidden state vector.

It is a common approach to use both preceding and following tokens to derive features for the current tokens in natural language processing tasks. When we look at the LSTM equations, the current output depends only on previous inputs, the initial cell value and the hidden state. Graves and Schmidhuber [29] proposed BiLSTM to gain information from future inputs. In a BiLSTM, two LSTM components are present, namely the forward LSTM and the backward LSTM. The forward LSTM traverses the sequence in the forward direction and the backward LSTM traverses the same sequence in the reverse order using \(h_{t+1}\) and \(c_{t+1}\) are used instead of \(h_{t-1}\) and \(c_{t-1}\) for the gate calculations. In a bidirectional model, the output at time \(t\) depends on both the forward hidden state \(h_t\) and the backward hidden state \(h_t\). The outputs of character and word embeddings are concatenated as input to the BiLSTM.

---

2https://fasttext.cc/docs/en/crawl-vectors.html
3) CRF (CONDITIONAL RANDOM FIELD)
For sequence labeling tasks, it is important to consider the correlations between labels in neighborhoods and jointly decode the best sequence of output labels for a given input sentence. For example, in POS tagging an adjective is more likely to be followed by a noun than a verb, and in NER with standard IOB2 annotation I-LOC cannot follow I-ORG. The IOB2 annotation used a B-tag for all base noun phrase initial words [65]. Therefore, instead of decoding each label independently, we model the label sequence jointly using

![Diagram](image-url)
a conditional random field [40]. Conditional random fields are a family of conditionally trained undirected graphical models used to calculate the conditional probability of values on designated output nodes given values assigned to other designated input nodes [40]. The CRF layer in our model was designed to select the best tag sequence from all possible tag sequences by considering the outputs from BiLSTM and the correlation between adjacent tags.

B. AMHARIC RoBERTa
Researchers have proposed pre-training language models to improve the performance of the NLP tasks through a large number of unlabeled data because the pre-trained model generate more enriched character embeddings for other layers. To utilize pre-training language model, we designed a RoBERTa model as our architecture. RoBERTa is similar to BERT, except that it changes the masking strategy and removes the next sentence prediction.

Implementations of BERT prepare a mask during data preprocessing to obtain a static mask. However, RoBERTa uses a dynamic mask: whenever a new sequence is fed into the model, the model generates a new mask pattern. The dynamic mask will help the model adapt to different masking strategies during the processing of massive quantities of data, and consequently learn more diverse language representations [44]. Figure 4 shows the pre-training and fine-tuning procedures in BERT. RoBERTa has the same structure as BERT, with 12 Transformer layers, 768 hidden units and 12 self-attention heads. Our Amharic RoBERTa model is publicly available for testing purpose.3

V. DATA PREPARATION
A. ANNOTATION STANDARDS
The currently available NER datasets are mostly annotated according to Stanford, IOB1 or IOB2 standards. Although all of these annotation standards aim to mark the tags of NEs, there exist subtle differences among these standards. These differences are depicted in Table 3 and the details of these standards are briefly introduced in the following subsections.

1) STANFORD STANDARD
In the Stanford annotation standard, a tag (among ORG, LOC and PER tags) representing the NE type should be assigned to each token. The main disadvantage of this standard is its inability to address the consecutive NE tokens in the sentence which leads to confusion about entities.

| Amharic Words | English Translation | Stanford Tag | IOB1 Tag | IOB2 Tag |
|---------------|---------------------|--------------|----------|----------|
| ከወን ቍልት | the artist          | O            | O        | O        |
| ከወን ቍልት | yesterday           | O            | O        | O        |
| ከወን ቍልት | to walta            | ORG          | I-ORG    | B-ORG    |
| ከወን ቍልት | information         | ORG          | I-ORG    | I-ORG    |
| ከወን ቍልት | center              | ORG          | I-ORG    | I-ORG    |
| ከወን ቍልት | as expressed        | O            | O        | O        |
| ማሰራ | of Awasa            | LOC          | I-LOC    | B-LOC    |
| ክሎስ | Arba                | LOC          | I-LOC    | B-LOC    |
| ማስረጃ | Minch               | LOC          | I-LOC    | I-LOC    |
| ማሰራ | a city              | LOC          | I-LOC    | B-LOC    |

English: The artist told Walta Information Center yesterday that a project to clean up the towns of Awassa, Arba Minch, Nazareth and Mekelle has been launched.

2) INSIDE-OUTSIDE-BEGINNING 1 (IOB1)
Similarly, the Stanford Annotation Standard, IOB1 also assigns a tag for each token [60]. However, unlike the Stanford Annotation Standard, IOB1 tags consist of two parts: a position marker (I, O or B) and the type of NE. The position markers I, O and B represent Inside, Outside and Beginning respectively. A NE, whether it be a single-word or multi-word noun phrase, between two non-NEs are denoted by the I position marker. B position is only used when the first word of an NE phrase immediately follows another NE phrase. This allows discrimination of separate consecutive entities in the text. Words marked as O for the non-NE tokens.

3) INSIDE-OUTSIDE-BEGINNING 2 (IOB2)
In IOB2 annotations, a B tag is used to mark all base noun phrases with initial words [63]. Figure 7 shows an example of the IOB2 annotation standards.

B. CURRENT AMHARIC NER DATASET
Most of the state-of-the-art studies using a corpus-based approach by applying different machine learning algorithms necessitate large amounts of data. However, the Amharic language is a low-resource language and suffers from the
lack of a large annotated corpus. The only available Amharic NER corpus has 13,000 tokens [3], which is not efficient for training deep learning models.

Additionally, the annotation scheme of this NER dataset is based on the Stanford standard, which causes ambiguous representations in entities constructed by consecutive tokens. Converting this annotation format to IOB format is a non-trivial task for the Amharic language because two consecutive person, location or organization names can appear together in a sentence without a separation character such as a comma (see Figure 5). IOB2 standard assigns these consecutive names as B-xxx (as the beginning of an NE type) and I-xxx (as inside of an NE type). However, the correct tags should be B-xxx and B-xxx. Therefore, building this IOB2-style annotated corpus is important and useful for the success and standardization of Amharic NER research.

Since named entity recognition is proposed as a word-level tagging problem, all of the proposed data sets use word-level tags to denote named entity phrases. A named entity phrase may span multiple words; hence the NE tag is composed of a concatenation of a position indicator (B—Beginning, I—Inside) and a NE type (PER, ORG, LOC, etc.). In addition, an O tag indicates that a token is not inside a NE phrase. A named entity phrase starts with a B- tag and if it consists of multiple words, the following word tags are prefixed with I-prefix.

![FIGURE 5. Sample consecutive names appear for the sentence: “አ الاخبار የአሮካ Rotterdam እና ያለው እስከ እንደ ለማስታወቂያ እና ለማስታወቂያው እና ለማስታወቂያው በእንደ ለማስታወቂያ እና ለማስታወቂያው እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስታወቂያ እና ለማስት

We started our work by defining the NER task for Amharic. Short guidelines for tagging the Amharic corpus are defined based on the unique properties of the Amharic language. Using the IOB2 coding scheme, the first annotator labels 182K tokens and around 780 sentences which is 20K tokens randomly selected and re-annotated by a second annotator for measuring inter-annotator agreement. The inter-annotator agreement is a measure of how well multiple annotators can make the same annotation decision for a certain category. Supervised machine learning algorithms use a labeled dataset, mostly annotated by human beings. Two commonly used annotation agreement standards are Cohen’s Kappa and Fleiss’s Kappa coefficients. The Cohen’s Kappa is a pairwise reliability measure between two annotators while considering the possibility of an agreement by chance [48]:

Inter-Annotator Agreement is a measure of how well multiple annotators can make the same annotation decision for a certain category. Supervised machine learning algorithms use a labeled dataset, mostly annotated by human beings. There are two common annotation agreement standards:
Addis Ababa University in a project called “The Annotation of Amharic News Documents”. The project was to manually tag each Amharic word in its context with the most appropriate POS.

This corpus is prepared in two forms i.e. in the Amharic version (using Ge’ez Fidäl) and in a transliterated format called system for Ethiopic representation in ASCII (SERA) using Latin characters. However, we have chosen to use both the Amharic version and the transcription one separately to measure the efficiency of the model on spelling variation in the Amharic script. The corpus has 210,000 words collected from 1,065 Amharic news documents of Walta Information Center, a government news and information service located in Addis Ababa, Ethiopia.

D. BUILDING DATASET

The previous researchers [3] tagged only 13,538 tokens based on Stanford notation. These datasets are not enough to build a deep neural network model and the standard used is different from IOB2. Therefore, we decided to prepare our dataset. From the total Walta Information Center dataset, we tagged 8,070 sentences, which have 182,691 tokens. Figure 6 shows a sample Ethiopian Languages Research Center (ELRC) document format:

Before the annotation process, we conduct a couple of cleaning operations on the corpus. The Walta Information Center Corpus is in XML format. The corpus contains some XML tags (e.g. `<document>`, `<filename>`, `<title>`), which are unnecessary for our purposes. Since we have used the body part of the document, all other parts and tags of the document are removed.

Manual tagging on the cleaned dataset is conducted by employing the IOB2 tagging scheme. Annotators tagged each word/token with other (O) or one of three NE classes: person, location, or organization. A single named entity could contain several tokens within a sentence. We use the IOB2 annotation standard for Amharic NER tasks. Figure 7 shows the tagging scheme in the IOB2 of the Amharic corpus with an example sentence along with its English translation.

Since the corpus is in the Amharic alphabet, it may be confusing for those who are unfamiliar with the symbols. To help non-native users, text Romanization was applied to the raw dataset. Text Romanization is the process of converting non-Latin letters into Latin letters. Amharic NLP researchers use the system for SERA transcription system [26]. SERA transcription system uses the same character for different orthographic (for example, the SERA equivalent of “”) is “ha”). Using transliterated corpus reduced spelling variations, which are common in Amharic script (Fidäl) writing. The spelling variation problem is also common for other NLP tasks like information retrieval systems. We have two varieties of a corpus, which are the Amharic script and the Latin script. It is also possible to reduce the spelling variation by using normalization.

They calculated an attention factor from the proportion of samples presented to the neural network for training. The learning rate of the network elements was adjusted based on the attention factor. In the case of imbalanced datasets, the classifier mostly classifies entities as majority class members [2]. Since the dataset is imbalanced, we applied SMOTE like Belay [9] and Demissie [17] on the dataset and tested our model based on the new datasets.

SMOTE is a means of increasing the sensitivity of a classifier to the minority class by balancing the number of instances between the classes. The algorithm obtains new samples by random linear interpolation between a few samples and their neighboring. The data imbalance ratio is increased by generating a certain number of artificial minority samples,
TABLE 6. NE type distribution in our amharic NER dataset after smote oversampling.

| NE Class | Token Count | Token Percentage |
|----------|-------------|------------------|
| Person   | 107,744     | 18.73%           |
| Location | 153,745     | 26.73%           |
| Organization | 149,584 | 26.01%           |
| O        | 164,087     | 28.53%           |
| Total    | 575,160     | 100.00%          |

FIGURE 8. The Learning curve shows the f1-score over the size of dataset.

so that the classification effect of the imbalanced data set is improved. Table 6 shows the number of instances for each class after applying SMOTE.

VI. EXPERIMENTAL RESULTS AND EVALUATION

A. EXPERIMENTAL SETUP

The hyperparameters we used are proposed by Lample et al. [40]. The researcher experimented with the different hyperparameters and proposed the best ones. In addition to this, we have also experimented with epoch size and found a maximum of 50. We used the size of word embeddings as 300, the size of the character embeddings as 25, and the batch size as 20. We use Adam [37] with a learning rate of 0.001 for optimization. Adam is an optimization algorithm that can be used instead of the stochastic gradient descent procedure to update network weights iteratively based on training data. To mitigate overfitting, we apply the dropout method to regularize our model by applying dropout on both the input and output vectors of BiLSTM. The researcher fixed the dropout rate at 0.5 for all dropout layers throughout all the experiments. Our transformer model trained using a using 4 GPUs (Quadro RTX 6000 with 24GB RAM) and it has taken 6 days to complete. We have used a training batch size of 8 for each gpu, block size of 512 and epoch size of 5.

B. EVALUATION METRICS

There are different types of named entity evaluation metrics described in related literature sections. For our experiments, we used the CoNLL evaluation metrics of precision, recall, and f1-score [72]. Among the evaluation metrics defined above, the CoNLL metric is the harshest one. The CoNLL evaluation metric is aggressive since partially matched NE tokens cannot contribute to the overall score. An NE has to be identified as a whole and its type must be correctly classified to gain credit. So, to evaluate our system aggressively, we opt to use the CoNLL metric in our evaluations.

C. EXPERIMENTS

We have compared different models such as BiLSTM, BiLSTM-CRF, and RoBERTa and tested them using our dataset and the SAY dataset. The size of the SAY dataset is almost half of our dataset. Details of the datasets are presented in the dataset section. For these models, we used two-thirds of the dataset for training and one-third of the dataset for testing. To develop a pre-training language model, we have collected datasets from different sources like news, Twitter, and the web. The dataset has more than 6 million sentences. Based on this dataset we have developed fastText and RoBERTa language models.

To compare the two datasets using the BiLSTM-CRF model, we change the Title and Time classes to O because our dataset does not have these classes. Table 7 shows that increasing the size of the dataset increases the performance of the system. The experimental result shows that there is an increase of 0.47% in the f1-score from BiLSTM to BiLSTM-CRF. In this experiment, we have also compared our fastText word vector with Facebook fastText for the BiLSTM-CRF model. The result shows that our new fastText word vector is better than Facebook fastText. This shows that increasing the size of word vectors improves the performance of the system. In addition to this, we have also compared pre-defined word vectors with the transformer learning (Roberta) model. As we can see from Table 7 RoBERTa is much better than fastText embeddings for Amharic. The RoBERTa model improves the Amharic NER system by 1.19%.

We measured the performance of our model by using different hyper-parameters: randomly initialized word vector, pre-defined word vector (fastText), using SMOTE experiments. For these experiments, we have used 10-fold cross-validation for the original dataset and for SMOTE experiments 80% for training and the remaining 20% for testing. For each experiment, we tested the model and presented it in Table 8.

In the first experiment, the character-based BiLSTM-CRF model based on random initial word vectors was used. The random initial word vector uses the random initial value for the word vector through the training process. Taking character embedding of 25 and word embedding size of 300. This Amharic NER system achieved Precision, Recall, and f1-score values of 66.41%, 74.72%, and 70.18% respectively as shown in Table 8.

In the second experiment, a character-based BiLSTM-CRF model was built based on predefined word vectors (fastText). Language model pre-training is effective in improving
TABLE 7. Comparison of tagging performance in Precision, Recall and f1-scores.

| Model                        | Dataset Type | Precision | Recall | f1-score |
|------------------------------|--------------|-----------|--------|----------|
| Pre-defined Word Vectors (Facebook) | BiLSTM-CRF   | SAY       | 57.99  | 70.32    | 63.56    |
| Pre-defined Word vectors (Our)           | BiLSTM-CRF   | SAY       | 60.76  | 67.18    | 63.81    |
| Pre-defined Word Vectors (Our)           | BiLSTM       | Ours      | 66.98  | 77.70    | 71.94    |
| Pre-defined Word Vectors (Facebook)       | BiLSTM-CRF   | Ours      | 68.71  | 76.28    | 72.30    |
| Pre-defined Word Vectors (Our)           | BiLSTM-CRF   | Ours      | 70.10  | 74.87    | 72.41    |
| Amharic RoBERTa                  | RoBERTa     | Ours      | 71.44  | 75.87    | 73.60    |

TABLE 8. Experimental results in Precision, Recall and F1-Scores.

| Model                        | Precision | Recall | f1-score |
|------------------------------|-----------|--------|----------|
| Randomly Initialized Word Vectors | 66.41±0.65 | 74.72±4.12 | 70.18±3.67 |
| Pre-defined Word Vectors (fastText) | 72.92±2.75 | 75.37±0.00 | 74.12±3.50 |
| Applying Cross Validation SMOTE | 86.01±2.75 | 84.66±5.77 | 85.33±3.66 |
| Applying SMOTE Only for Training Datasets | 91.42 | 95.01 | 93.18 |
| SMOTE for All Datasets       | 98.75     | 99.15   | 98.95    |

We also analyze to measure the effect of the dataset size on the NER performance. We trained our best-performing setup with datasets of different sizes and measured the performance. The results are depicted in the learning curve presented in Figure 8. As expected, the more data we have, the better f1-score we get. The positive incline on the right-hand side of the learning curve represents we have not converged yet, in other terms, an even higher f1-score may be achieved with more training datasets.

The Amharic fastText word vectors have 304,649 words, increasing the size of the word vector decreases out of vocabulary (OOV) and increases the performance of the model [42]. Similar to Belay [9] and Demissie [17], using balanced classes in the dataset increased the performance of the Amharic NER system. The final version of our Amharic NER dataset is made public.

VII. CONCLUSION AND FURTHER WORK

In this study, we built a publicly available and relatively large Amharic NER dataset and achieved state-of-the-art NER results by using RoBERTa deep learning models.

One of the obstacles to improving the Amharic NER studies was the lack of high-quality and publicly available NER datasets. We address this issue in our study by manually tagging a relatively large Amharic NER dataset which is composed of 8,070 sentences and 182K tokens taken from the Walta Information Center dataset. To assess the quality of the annotation process, we also employ inter-annotation

4 https://github.com/Ebrahimc/ANEC-An-Amharic-Named-Entity-Corpus-
Algorithm 1 Algorithms SMOTE (T, N, k)

Input: Number of minority class sample $T$; Amount of SMOTE N%; Number of nearest neighbors k

Output: $\left(\frac{N}{100}\right) \times T$ synthetic minority class samples

(* if N is less than 100%, randomize the minority class sample as only a random percent of them will be SMOTE. *)

If $N < 100$
then Randomize the T minority class samples

$T = \left(\frac{N}{100}\right) \times T, \quad N = 100$
end if

$k = \text{Number of nearest neighbors},$

numattrs = \text{Number of attributors}

sample []]: array for original minority class samples

newindex: keeps a count of number of synthetic samples generated, initialized to 0

Synthetic[][]: array for synthetic samples (* Compute k nearest neighbors for each minority class sample only*)

for $i \leftarrow -1$ to $T$

Compute k nearest neighbors for $i$, and save the indices in the nnarray

Populate(N, i, nnarray)
endfor

Populate(N, i, nnarray)(* Function to generate the synthetic samples. *)

while $N \neq 0$

Choose a random number between 1 and $l$, call it nn. This step chooses one of the k nearest neighbors of $i$.

for attr $\leftarrow -1$ to numattrs

Compute: $\text{dif} = \text{Sample}[[\text{nnarray}[\text{nn}]]][\text{attr}] - \text{Sample}[i][\text{attr}], \quad \text{Compute: gap} = \text{random number between 0 and 1 Syn-}$

thetic[newindex][attr] = Sample[i][attr] + gap$^*\text{dif}$
endfor

newindex++,

$N = N-1$
endwhile

return (*End of Populate.*)

End of Pseudo-Code.

agreement measurements where we got a 0.7321 Kappa score denoting there is a substantial agreement between annotators. The final version of this dataset is made publicly available to serve as a baseline for further NER and other NLP-related studies on Amharic. Also, we would like to extend our efforts to increase the size of this corpus as well as to increase the variety of the corpus by acquiring textual content from sources other than the Walta Information Center dataset.

Another contribution of our study is an Amharic NER system developed by deep learning models employing a CRF classifier on top of a BiLSTM layer and a RoBERTa model. One of the findings from our experimental studies is that utilizing pre-trained word vectors trained on a comparatively large corpus performs better instead of randomly initialized word vectors. To overcome the problem of imbalanced class distribution of the dataset, we evaluate the effects of the SMOTE oversampling method. Our best performing configuration achieves to get a new Amharic NER state-of-the-art score of 93.18% as an f1-score, which is significantly outperforming the baseline.

Recent studies on the NER task for other languages take the advantage of transfer learning by using transformer architectures like BERT [19]. Transformer architectures need to optimize a large number of weights, which necessitates huge amounts of text. We have collected around 6 million datasets from news, Twitter, and web corpus for developing the RoBERTa model. The RoBERTa model improves the Amharic NER system by 1.19%.

Having morphological features as input is shown to be beneficial in many of the previous studies. Although an Amharic morphological analyzer is available [28], we did not augment our word vectors with morphological features since no morphological disambiguation tool is accessible. Since the outputs of the morphological analysis phase are ambiguous, morphological disambiguation is inevitable for Amharic. To improve our NER results and support future NLP-related studies in Amharic, we also started to work on the Amharic morphological disambiguation task. We expect to improve our current state-of-the-art Amharic NER result with the contribution of these morphological features.

APPENDIX A
THE ALGORITHM OF SMOTE
See Algorithm 1.

APPENDIX B
AMHARIC NER ANNOTATION GUIDELINE
We have prepared annotation guideline for Person, Location and Organization entities.
A. PERSON

When the entity refers to an individual or collective person (more than one individual) including fictitious persons. Even in the case of a collective person annotation, there must be the presence of a proper name. Titles/prefixes such as “Mr./Mrs.”, “Prof.”, “Ms.”, “Dr.”, “Imam”, “President” are not considered part of a person name.

Example:

"Mrs. E. C. Jibril, A. C. Tantuğ: ANEC: An Amharic Named Entity Corpus and Transformer Based Recognizer"

B. ORGANIZATIONS

A company which sells products or provides services that are not only administrative. It includes both private and public companies, as well as hospitals, schools, universities, political parties, trade unions, police, gendarmerie, mosques, churches, sportive clubs, etc.

An organization which plays a mainly administrative role. It is often an administrative and/or geographical division. This includes town halls, city council, regional council, state council, federal council, named government, ministry parliament, prefectures, ministries dioceses, tribunal, court, government treasury, public treasury, international org.

Corporate or organization designators such as “Co.” are considered part of an organization name.

Example:

"Mr. E. C. Jibril, A. C. Tantuğ: ANEC: An Amharic Named Entity Corpus and Transformer Based Recognizer"

Proper names that are to be tagged as ORGANIZATION include stock exchanges, multinational organizations, businesses, TV or radio stations, political parties, religions or religious groups, orchestras, bands, or musical groups, unions, non-generic governmental entity names such as “Congress” or “Chamber of Deputies,” sports teams and armies (unless designated only by country names, which are tagged as LOCATION), as well as fictional organizations (to ensure consistency with marking other fictional entities).

Example:

"Mr. E. C. Jibril, A. C. Tantuğ: ANEC: An Amharic Named Entity Corpus and Transformer Based Recognizer"

C. LOCATIONS

The TYPE LOCATION applies to entities representing either geographical, political, or astronomical locations. Examples of strings that are tagged as LOCATION include: continents, countries, provinces, counties, cities, regions, districts, towns, villages, neighbourhoods, airports, military bases, railways, railroads, highways, bridges, street names, street addresses, oceans, seas, straits, bays, channels, sounds, rivers, islands, lakes, national parks, mountains, fictional or mythical locations.

The following are examples of locations:

"Mr. E. C. Jibril, A. C. Tantuğ: ANEC: An Amharic Named Entity Corpus and Transformer Based Recognizer"
• Compound expressions in which place names are listed are to be tagged as separate instances of LOCATION.

```
<ENAMEX TYPE = "LOCATION"> Адада Абаба
<ENAMEX>
```

• Designators that are integrally associated with a place name are to be tagged as part of the name. For example, include in the tagged string the word ‘‘River’’ in the name of a river, ‘‘Mountain’’ in the name of a mountain, ‘‘City’’ in the name of a city, etc., if such words are contained in the string.

```
<ENAMEX TYPE = "LOCATION"> Адада Абаба
```

• A place name can be included in the name of organization

```
<ENAMEX TYPE = "ORGANIZATION">
```

• Street/neighborhood names will be tagged as a single expression that includes the noun Street.

```
<ENAMEX TYPE = "LOCATION">
```

• Directional expressions (north/south, etc.) will be tagged only when they are part of the official name

```
<ENAMEX TYPE = "LOCATION">
```

REFERENCES

[1] D. I. J. Adelani Abbott, G. Neubig, D. D’ouza, J. Kreutzer, C. Lignos, and S. Osei, “MasakhaNER: Named entity recognition for African languages,” Trans. Assoc. Comput. Linguistics, vol. 9, pp. 1116–1131, Dec. 2021.

[2] R. Akbani, S. Kwek, and N. Japkowicz, “Applying support vector machines to imbalanced datasets,” in Proc. Eur. Conf. Mach. Learn. Berlin, Germany: Springer, Sep. 2004, pp. 39–50.

[3] B. Alemu, “A Named entity recognition for Amharic,” M.S. thesis, Comput. Sci., Addis Ababa Univ., Addis Ababa, Ethiopia, 2013.

[4] W. Antoun, F. Baly, and H. Haji, “AraBERT: Transformer-based model for Arabic language understanding,” 2020, arXiv:2003.00104.

[5] A. A. Argaw and L. Asher, “An amharic stemmer: Reducing words to their citation forms,” in Proc. Workshop Comput. Approaches Semitic Lang. Common Issues Resour. Semitic, 2007, pp. 104–110.

[6] B. Babych and A. Hartley, “Improving machine translation quality with automatic named entity recognition,” in Proc. 7th Int. EAMT Workshop MT Other Lang. Technol. Tools, Improving MT Through Other Lang. Technol. Tools Resour. Tools Building MT (EAMT), 2007, pp. 1–8.

[7] B. Belay, T. Habtegebrial, M. Meshesha, M. Liwicki, G. Belay, and D. Stricker, “Factored convolutional neural network for amharic character image recognition,” in Proc. IEEE Int. Conf. Image Process. (ICIP), Sep. 2019, pp. 2906–2910.

[8] B. Belay, T. Habtegebrial, M. Meshesha, M. Liwicki, G. Belay, and D. Stricker, “Amharic OCR: An end-to-end learning,” Appl. Sci., vol. 10, no. 3, p. 1117, Feb. 2020.

[9] M. T. Belay, “Amharic named entity recognition using a hybrid approach,” M.S. thesis, Comput. Sci., Addis Ababa Univ., Addis Ababa, Ethiopia, 2014.

[10] A. Chaimae, E. Y. Yacine, M. Rybinski, and J. F. A. Montes, “BERT for Arabic named entity recognition,” in Proc. Int. Symp. Adv. Electr. Commun. Technol. (ISAECT), Nov. 2020, pp. 1–6.

[11] N. V. Chawla, K. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique.” J. Artif. Intell. Res., vol. 16, no. 1, pp. 321–357, Jan. 2002.

[12] N. Chinchor and B. Sundheim, “MUC-5 evaluation metrics,” in Proc. 5th Conf. Message Understand. (MUC), 1993, pp. 22–29.

[13] J. P. C. Chiou and D. Nichols, “Named entity recognition with bidirectional LSTM-CNNs,” Trans. Assoc. Comput. Linguistics, vol. 4, pp. 357–370, Dec. 2016.

[14] R. Collinbert, J. Westen, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” J. Mach. Learn. Res., vol. 12, pp. 2493–2537, Aug. 2011.

[15] Y. Cui and C. Zhu, “Fine-grained Chinese named entity recognition with RoBERTa and convolutional attention networks,” in Proc. IEEE 6th Int. Conf. Comput. Commun. (ICCC), Dec. 2020, pp. 2104–2109.

[16] G. A. Demeke and M. Getachew, “Manual annotation of Amharic news items with part-of-speech tags and its challenges,” in Proc. Ethiopian Languages Res. Center Working Papers, vol. 2, 2006, pp. 1–16.

[17] D. Demissie, “Amharic named entity recognition using neural word embedding as a feature,” M.S. thesis, Comput. Sci., Addis Ababa Univ., Addis Ababa, Ethiopia, 2007.

[18] E. DeRouin, J. Brown, L. Fausett, and M. Schneider, “Neural network training on unequally represented classes,” in Intelligent Engineering Systems Through Artificial Neural Networks. New York, NY, USA: ASME Press, 1991, pp. 135–141.

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” 2018, arXiv:1810.04805.

[20] A. M. Dai and Q. V. Le, “Semi-supervised sequence learning,” in Proc. Adv. Neural Inf. Process. Syst., vol. 28, 2015, pp. 3079–3087.

[21] C. Dong, “Character-based LSTM-CRF with radical-level features for Chinese named entity recognition,” in Natural Language Understanding and Intelligent Applications. Cham, Switzerland: Springer, 2016, pp. 239–250.

[22] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, “Transition-based dependency parsing with stack long short-term memory,” 2015, arXiv:1505.08075.

[23] D. M. Eberhard, G. F. Simons, and C. D. Fennig, Eds., Amharic. Ethnologue: Languages of the World, 18th ed. Dallas, TX, USA: SIL International, Mar. 2021.

[24] S. Ertekin, J. Huang, L. Bottou, and L. Giles, “Learning on the border: Active learning in imbalanced data classification,” in Proc. 16th ACM Conf. Conf. Knowl. Manage., Nov. 2007, pp. 127–136.

[25] T. Fawcett and F. Provost, “Combining data mining and machine learning for effective user profile,” in Proc. 2nd Int. Conf. Knowl. Discovery Data Mining, Portland, OR, USA: AAAI, 1996, pp. 8–13.

[26] Y. Firdiyiwek and D. Yaqob, “The system for Ethiopic representation in ASCII,” Indiana Univ., Univ. Virginia. Charlottesville, VA, USA, Tech. Rep., 1997.

[27] O. Cohan, B. Oz, and G. T. Oz, “Sentiment analysis for Turkish Twitter feeds,” in Proc. 23rd Signal Process. Commun. Appl. Conf. (SPC), May 2015, pp. 33–52.

[28] M. Gasser, “HornMorpho: A system for morphological processing of Amharic, Oromo, and Tigrinya,” in Proc. Conf. Hum. Language Technol. Proc. Conf. Amh. Tech., Alexandria, Greece, May 2015, pp. 33–52.

[29] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional LSTM and other neural network architectures,” Neural Netw., vol. 18, nos. 5–6, pp. 602–610, Jul. 2005.

[30] A. Graves, A.-R. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process., May 2013, pp. 6645–6649.

[31] M. A. Greenwood and R. Gaiauskas, “Using a named entity tagger to guide the surface matching text patterns for question answering,” in Proc. Workshop Natural Language Process. Question Answering (EACL), 2003, pp. 29–34.

[32] R. Grishman and B. Sundheim, “Message understanding conference-6: A brief history,” in Proc. 16th Conf. Comput. Linguistics, 1996, pp. 466–471.

[33] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,” in A Field Guide to Dynamical Recurrent Neural Networks. Piscataway, NJ, USA: IEEE Press, 2001.
