

RESEARCH ARTICLE

Semantic Role Labeling for Amharic Text Using Multiple Embeddings and Deep Neural Network

BEMNET MERESA HAILU¹, YAREGAL ASSABIE², AND YENEWONDIM BIADGIE SINSHAW³

¹Addis Ababa Institute of Technology, Addis Ababa University, Addis Ababa 1176, Ethiopia

²Department of Computer Science, Addis Ababa University, Addis Ababa 1176, Ethiopia

³Department of Software and Computer Engineering, College of Information Technology, Ajou University, Suwon 16499, South Korea

Corresponding author: Yaregal Assabie (yaregal.assabie@aau.edu.et)


This work was supported by Addis Ababa University.

ABSTRACT Amharic is morphologically complex and under-resourced language, posing difficulties in the development of natural language processing applications. This paper presents the development of semantic role labeler for Amharic text using end-to-end deep neural network architecture. The system implicitly captures morphological, semantic and contextual features of a word at different levels of the architecture, and incorporates the syntactic structure of an input sentence. The proposed neural network architecture has four core layers from bottom to top, namely non-contextual word embedding, contextual word embedding, fully connected and sequence decoding layers. The non-contextual word embedding layer is formed from the concatenation of character-based, word-based and sentence-based word embeddings. This layer captures the morphological and semantic features of a given word by making use of BiLSTM recurrent neural network. At the contextual word embedding layer, a context sensitive embedding of a word is generated by applying a new LSTM layer on the top of the non-contextual concatenated word embedding layer. A fully connected network layer is added on top of contextual word embedding layer to supplement it by extracting dependencies among training samples in the corpus. At the sequence decoding layer, a sequence of semantic role labels is predicted using a linear-chain conditional random field algorithm by capturing the dependency among semantic role labels. In addition to the four core layers, the architecture has dropout layers to prevent overfitting problem. The proposed system achieves 94.96% accuracy and 81.2% F1 score when it is tested using test data.

INDEX TERMS Semantic role labeling, Amharic text processing, multiple embeddings, deep neural network.

I. INTRODUCTION

Semantic roles are shallow semantic representations that express the roles that can be taken by arguments of a predicate describing an event. Computational systems use semantic roles as a shallow meaning representation to make simple inferences that are not possible from the pure surface string of words [1]. Semantic roles, thus, can help to generalize over different surface realizations of predicate arguments [1], [2]. Semantic role labeling (SRL) is a sentence-level semantic analysis of a text concerned with the characterization of

The associate editor coordinating the review of this manuscript and approving it for publication was Yiming Tang .

events, such as determining *who* did *what* to *whom*, *where*, *when*, and *how* [1]. Given a sentence, SRL consists of three basic tasks: predicate identification, argument identification, and argument classification [1], [2], [3]. This task of understanding how participants relate to events is a central question of natural language understanding [3] and, therefore, can potentially benefit many natural language processing (NLP) applications, such as question answering [4], information extraction [5], [6], text summarization [7], machine translation [8], [9], language understanding [10] and sentence similarity measurement [11].

Semantic role labeling can be feature-based, neural-based or a hybrid of the two. Feature-based algorithms estimate

a parse tree of a sentence and extract hand-designed features from its syntactic constituents in the parse tree, which includes part-of-speech tags, chunks and clauses [1], [2]. Neural algorithms for SRL are generally designed based on sequence processing neural networks in which each input word in a sentence is mapped to pre-trained embeddings and the concatenation of these embeddings is passed through multiple layers of the network [3]. Output from the last network can then be turned into a tag sequence by passing it through a single layer into a probability distribution over all the SRL roles (tags), and the most likely tag computed for each word is selected. Neural-based SRL systems are implemented using deep learning which has been proven to be particularly good at learning intermediate representations and providing an effective end-to-end joint system learning [12], [13]. SRL systems using deep learning can, therefore, easily learn important representations from just the raw input and can avoid the need to address the SRL problem in modular steps such as pruning, argument identification and argument classification. From the perspective of implementation, neural-based semantic role labeling systems can be modeled using dependency-based style [14], [15], span-based style [16], [17], or cross-based style [18]. Dependency-based and span-based styles annotate the syntactic heads of arguments and the entire argument span, respectively whereas cross-based style integrates both. From the perspective of the number of predicates, neural-based SRL systems can be first-order or second-order. When there are two predicates in a given sentence, first-order SRL systems detect each of predicate-argument pairs in isolation using local and short features [14], [15], [16], [17] whereas second-order SRL systems consider the detection of both pairs of predicate-arguments simultaneously [19]. To further improve the performance of pure neural-based SRL systems, syntactic information can be used as an additional input. Such types of SRL systems are considered as hybrid SRL systems, and they combine automatically extracted features from neural network model and syntactic features extracted by a syntactic parser from the featured-based approach [14], [15].

For languages like English having various resources, there are several successful research works on semantic role labeling by applying advanced machine learning algorithms on corpora that are annotated with semantic role labels like FrameNet [20], PropBank [21] and NomBank [22]. These corpora provide the training and test datasets to develop SRL systems using feature-based, neural-based and hybrid approaches to SRL systems. However, to our best knowledge, under-resourced languages like Amharic have no standard training and test data in Penn Treebank [23] and PropBank. As a result of this, research on semantic role labeling and other higher level NLP tasks for the language are still at their infancy stages. Thus, this work aims at developing neural-based SRL system for Amharic by considering its complex morphological structure and lack of lexical resources. In view of this, we proposed span-based first-order neural SRL

system for Amharic text. The major contributions of this work are summarized as follows.

- 1) An end-to-end deep learning based semantic role labeling framework is proposed for morphologically rich Semitic languages in general and for Amharic language in particular. Although we developed semantic role labeler for Amharic language, the framework can be customized easily to other Semitic languages.
- 2) The unique morphological and syntactical characteristics of the Amharic language that make it difficult to develop semantic role labeler are identified and a novel deep learning based multiple word embedding is proposed to represent the lexical semantics of each word at the input layer of the proposed neural based semantic role labeler.
- 3) We prepared a dataset, in consultation with domain experts, for training and testing Amharic semantic role labeling systems. As Amharic is a low-resource language, the dataset can serve as a baseline for the development of Amharic semantic role labeler.

The remainder of the paper is organized as follows. Section II describes the linguistic characteristics of Amharic semantic role labeling. Section III discusses related works employing various techniques and their limitations to apply directly for morphologically complex languages like Amharic. Section IV presents the proposed model that takes the characteristics of Amharic into consideration. Section V describes experimental works and the results obtained. Conclusion and future work are presented in Section VI.

II. AMHARIC LANGUAGE

A. AMHARIC MORPHOLOGY AND GRAMMAR

Amharic is the working language of the government of Ethiopia currently having an estimated population of 120 million. The language is written using a syllabic writing system having 34 base characters and 7 vowels (*ä, u, i, a, e, ə* and *o*). The vowels modify base characters to form a unique shape representing consonant-vowel combination. There are some homophone characters, along with their respective modifications, which are used interchangeably. These are *ሀ*, *ሐ*, *ኀ*, and *ኸ* /*hāl*; *ሠ* and *ሰ* /*säl*; *አ* and *ዐ* as /*ʔäl*; and *ጸ* and *ፀ* /*ṣäl*. Amharic uses its own punctuations such as *፥* (full stop), *፣* (comma), *፥* (semi-colon), and it also borrows some punctuation marks such as ‘?’ and ‘!’ from the Latin alphabet.

The language uses a set of numerals although there is nowadays a tendency of using the Hindu-Arabic numerals.

Amharic exhibits a typical behavior Semitic languages with regard to the pattern of inflectional and derivational morphology [24]. Among other word classes, the most morphologically complex ones are verbs, nouns and adjectives. The prominent phenomenon of the language is the reliance of verbs on root-and-pattern paradigms for word formation. The standard account of word formation process in the language describes words as combinations of two morphemes: a root and a pattern [25]. The root consists of consonants only,

Table 1. Examples of stem and word formation in Amharic.

Root	Stem	Words in surface form		
ሰ-ብ-ር /f-l-g/	ሰብር /səbär/	መሰብር /mäsäbär/	ለመሰብር /lämäsäbär/	ለመሰብሩ /lämäsäbäru/
	ሰብር /säbər/	ትሰብር /täsäbər/	ስትሰብር /sätäsäbər/	ስትሰብሩን /sätäsäbərən/
	ሰባበር /säbabär/	መሰባበር /mäsäbabär/	የመሰባበር /jämäsäbabär/	የመሰባበሪያ /jämäsäbabärija/
ፍ-ል-ግ /f-l-g/	ፈለግ /fäläg/	መፈለግ /mäfäläg/	ለመፈለግ /lämäfäläg/	ለመፈለጉ /lämäfälägu/
	ፈልግ /fäläg/	ትፈልግ /täfäläg/	ስትፈልግ /sätäfäläg/	ስትፈልጉን /sätäfälägän/
	ፈለለግ /fäläläg/	መፈለለግ /mäfäläläg/	የመፈለለግ /jämäfäläläg/	የመፈለለጊያ /jämäfälälägija/
ም-ር-ክ /m-r-k/	ማረክ /maräk/	መማረክ /mämaräk/	ለመማረክ /lämämaräk/	ለመማረኩ /lämämaräku/
	ማርክ /marək/	ትማርክ /tämarək/	ስትማርክ /sätämarək/	ስትማርኩን /sätämarəkän/
	መራረክ /märaräk/	መመራረክ /mämaräräk/	የመመራረክ /jämämaräräk/	የመመራረኪያ /jämämaräräkija/

called radicals. The pattern is a combination of vowels and, possibly, consonants too, with ‘slots’ into which the root consonants can be inserted. Stems are, therefore, formed by interdigitating the vowels among root consonants. Surface forms of words are then formed by combining stems with a particular set of prefixes or suffixes representing a single grammatical form. Table 1 presents examples of stem and surface word formation from roots. Amharic verbal roots can be marked for person, tense, aspect, mood and case. Furthermore, they can be combined with a set of prefixes and suffixes allowing the generation of thousands of words from a single root form. Nouns, adjectives and adverbs can be derived from a verbal root through the application of patterns. For example, the noun ፍጥነት /fätB-ənät ‘speed’/, the adjective ፈጣን /fätB-an ‘speedy’/ and the adverb በፍጥነት /bäfätB-ənät ‘speedily’/ are derived from the verbal root ፍጥ-ን /f-tB-n/. Once nouns and adjectives are derived, they can undergo the process of inflection and further derivation similar to that of non-derived ones. Nouns and adjectives can be marked for person, gender, objective and possessive cases, number and definiteness. Furthermore, prepositions, adverbs and conjunctions may come as prefixes and suffixes. Generally, the morphological structure of an Amharic word *W* takes the form [26]:

$$[p] * w[s]*$$

where *p* is a prefix morpheme, *w* is the root or stem of *W*, *s* is a suffix morpheme, [...] denotes optionality, and ‘*’ denotes the possibility of multiple occurrence. For example, ያለስምምነታቸው /jaläsəməmənätacäw ‘without their agreement’/ is constructed from the prefixes የ /jal/ and አለ /äläl/, the stem ስምም /səməm/, and the suffixes ነት /nät/ and እቸው /äcäw/.

The basic word order of Amharic language is subject-object-verb (SOV). Verbs agree with their subjects and definite objects in person, number and gender [24]. Figure 1 shows the agreement of a verb with the subject and object in the sentence ተማሪዎች መስኮቱን ሰበሩት /tämariwoc mäskotun säbärut ‘students broke the window’/. The verbal stem säbär /‘broke’/ is marked (by the suffix “u”) for the subject tämariwoc /‘students’/ which is third person plural.

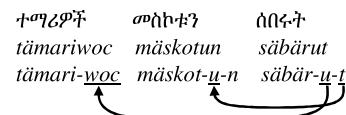


Figure 1. Agreement of a verb with its subject and object.

In addition, the verbal stem is marked (by the suffix “t”) for the definite object mäskotun /‘the window’/.

The complexity of grammatical agreements in Amharic sentences grow significantly with the increase in the number of words in the sentence due to proliferation of inflections. Consider the sentence መምህሩ ተማሪዎች እንዳይሰሩቱ እስቀድሞ በተደጋጋሚ እስጠንቅቋቸዋል /mäməħəru tämariwoc ?ändajəsasatu ?äsqädəmo bätädägagami äästB-änqəoacäwal ‘the teacher has previously warned students repeatedly to prevent them from making mistakes’/. The verb/action here is እስጠንቅቋቸዋል /?ästB-änqəoacäwal ‘[the teacher] has warned [students]’/ which inflects the stem ጠንቅቅ /tB-änqəq/ with the prefix እስ /?äs/ and the suffixes ኦ /?ol/, እቸው /?äcäw/ and አል /?äl/. The morphological and grammatical features representing agreements in the sentence are shown in Figure 2.

Pronoun subjects and pronoun objects are omitted unless they are emphasized [25]. In this case, verbs are still marked for omitted subjects and objects. Such complex word formation process may result in a single word representing a sentence constructed from subject, object, verb and other grammatical functions. Figure 3 shows the components of the word አልሰበርኳቸውም /?älsäbärkuacäwəm ‘I did not break them’/ which is a sentence.

B. SEMANTIC ROLE IDENTIFICATION IN AMHARIC

As Amharic verbs agree with their subjects and definite objects in person, number and gender, verb-agreement indicators are glued or inflected with the verb or other words in the sentence. Thus, the labeling of semantic roles in an Amharic sentence requires identifying verb-agreement indicators which are inflected with the words in that sentence. For example, as shown in Figure 1, the verb säbärut is

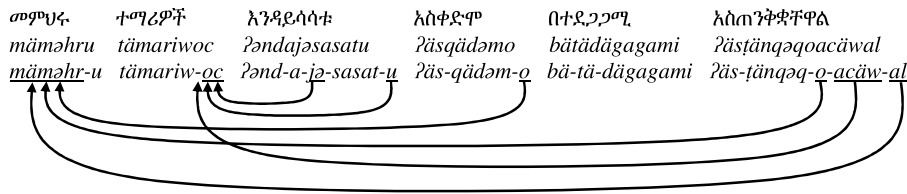


Figure 2. Agreement of words in a sentence.

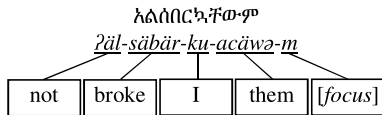


Figure 3. Morphological components of a word representing a sentence.

marked for the subject *tāmariwoc* (third person, plural) which answers the ‘who’ (AGENT) in the semantic role labeling (SRL) problem. The verb is also marked for the object *mäskotun* (third person, masculine, definite) which points to the recipient of the action and answers the ‘to whom’ (PATIENT/RECIPIENT) in the SRL problem. Accordingly, the sentence in Figure 1 is represented in PropBank semantic role label notation as:

[*tāmariwoc*]_{ARG0} [*mäskotun*]_{ARG1} [*säbärut*]_{PRED}

On the other hand, in Figure 2, the verb *ʔästB-änqəqoacäwal* is marked for the subject *māməhəru* (third person, masculine, singular) which answers the ‘who’ (AGENT) of the verb in SRL. It is also marked for the object *tāmariwoc* (third person, plural) which answers the ‘to whom’ (RECIPIENT) of the verb. The word *ʔəndajəsasatu* (derived from the verbal stem *sasat*) binds the verb to the plural object through the u suffix and answers the ‘why’ (PURPOSE). The prefix *bä* in the word *bätädägagami* represents the preposition ‘in’ and answers the ‘how’ (MANNER) for the verb. The prefix *ʔäs* in the word *ʔäsqädəmo* (derived from the verbal stem *qädə*) represents the specific time frame for the verb and answers the ‘when’ (TEMPORAL) for the verb *ʔästB-änqəqoacäwal* in the SRL problem. Thus, the sentence in Figure 2 is represented in PropBank semantic role label notation as:

[*māməhəru*]_{ARG0} [*tāmariwoc*]_{ARG1}
 [*ʔəndajəsasatu*]_{ARGM-PRP} [*ʔäsqädəmo*]_{ARGM-TMP}
 [*bätädägagami*]_{ARGM-MNR} [*ʔästB-änqəqoacäwal*]_{PRED}

The syntactic phenomena play a crucial role in the development of SRL system for the language. From a syntactic perspective, the following facts about Amharic are relevant for SRL.

1) PRO-DROP

Amharic tend to drop the subject and mark it morphologically on the verb. For example, ብርቱካን በሉ /*bərtukan bälu* ‘[they] ate orange’/ is pro-dropped where the pronoun ‘they’

is absent. The explicit rendering of the sentence would be እነሱ ብርቱካን በሉ /*ʔənäsu bərtukan bälu* ‘they ate orange’/. This is a significant phenomenon for SRL since one of the arguments is missing, and therefore, the system has to either be able to make it explicit or mark a trace with the argument class as well as identify the morpheme marker on the predicate.

2) RELATIVE FREE WORD ORDER

Amharic allows for verb-subject-object (VSO), OVS, SVO, etc. For example, the following sentence can all mean ‘Eyob ate his dinner’.

በላ ኢዮብ አራቱን /*bäla(V) ʔijob(S) ʔəratun(O)*/
 አራቱን በላ ኢዮብ /*ʔəratun(O) bäla(V) ʔijob(S)*/
 ኢዮብ አራቱን በላ /*ʔijob(S) ʔəratun(O) bäla(V)*/
 ኢዮብ በላ አራቱን /*ʔijob(S) bäla(V) ʔəratun(O)*/

This poses a significant challenge since it has implications in identifying the arguments and their roles, especially when the arguments are proper nouns with no explicit grammatical case marking due to diacritic under-specification especially in the singular cases.

3) POSSESSIVE CONSTRUCTS

Possessive constructs are especially tricky in Amharic as they allow for multiple recursive embeddings as well as allowing for multiple modifiers. These idafa constructs are quite difficult to parse and thus pose a significant challenge to boundary detection for the argument spans. For example, የአለቃዬ ማሰት ዐጎት የልጅ ልጅ ጓደኛጠፋኝ /*ʔjəʔäläqaye misət ʔägət ʔäləḅ ləḅ guadäña täfac*/ is literally interpreted as ‘my boss’s wife’s uncle’s grandchild’s friend disappeared’.

III. RELATED WORK

The task of automatic SRL was pioneered by Gildea and Jurafsky [2]. The system is based on the probability distribution statistical classifier trained on sentences that were hand-annotated with semantic roles. The probability estimation then determines how likely a constituent is to fill each possible role, given the features and the predicate, or a target word. The system achieved F-Score of 62.9% tested on FrameNet. Feature-based SRL was also proposed by Akbik and Li [27] to identify instances that share the most similar combination of atomic features. Comparative evaluations demonstrated that the approach significantly outperforms prior state-of-the-art SRL systems for verbal predicates and

their roles on both in-domain and out-of-domain data, reaching F1-Score of 89.28% and 79.91%, respectively using the scoring metric of the CoNLL-2009 shared task. On the other hand, Zhou and Xu [16] developed an end-to-end SRL system for English using neural embeddings to learn the important input features, without using any syntactic knowledge and avoiding human engineered feature templates. Experimental analysis shows that the model is better at handling longer sentences than feature-based models, and that latent variables of the model can also implicitly capture the syntactic structure of a sentence. The model achieved an F-score of 81.07% on CoNLL-2005 shared task and 81.27% on CoNLL-2012 shared task, both outperforming the previous systems based on parsing results and manual feature engineering.

Wang et al. [17] used deep learning for developing Chinese SRL. Prior feature-based Chinese SRL systems have failed to model the long range dependencies in a sentence. This work, however, uses bi-directional long short-term memory (BiLSTM) to capture both bi-directional and long range dependencies in a sentence with minimum feature engineering. Compared to previous works, the system achieved a significant improvement with experiments on Chinese PropBank resulting in an F-1 score of 77.09%. Syntax-aware long short-term memory (LSTM) was proposed by Qian et al. [15] to directly model complex tree structure of dependency relation. The model is based on BiLSTM, and directed connections between dependency related words in BiLSTM are added in order to model the whole dependency tree. Experimental results show that the model gives F1 of 79.92% on Chinese Proposition Bank and improves the F1 score of ordinary BiLSTM with feature engineered dependency relation information by 2.06%. On English CoNLL 2005 dataset, the model yields an improvement of 2.1% in comparison to BiLSTM model.

Diab et al. [28] developed SRL system for Arabic that exploits many aspects of the complex morphological features of the language. The hypothesis was that taking advantage of the interaction between morphology and syntax could improve the performance of SRL systems for morphologically rich languages. Kernel methods were used for feature engineering, and support vector machines algorithm was used to implement a two step classification approach. The system is trained and tested using Arabic Propbank, and experiments show F-score of 82.17%, which significantly improved previously reported results on the same task and dataset. Regarding the development of Amharic SRL, memory-based learning was proposed by Yirga [29] in which each word is tagged with part-of-speech and the input text is parsed with phrase structure schema to extract the features that capture the semantic role of words in a sentence. However, the lack of lexical resources for the language has made the system to be tested on only 240 manually annotated simple sentences, and evaluation result showed an F-Score of 79.77% with optimized parameter settings. The errors in both the morphological and syntactic parsers have also reduced the overall performance of the SRL system.

The review of related works show that the characteristics of languages has an impact on the overall performance of SRL systems. Amharic poses a significant challenge to SRL for several reasons. Amharic is known to be an under-resourced language where there are no publicly available annotated resources required to develop SRL. Furthermore, it is a morphologically complex language making the feature extraction task required in the feature-based SRL very complex and prone to errors as it requires morphological and syntactic parsers [24], [29]. Errors encountered in these parsers can propagate to the SRL prediction task. Considering the complex characteristics and unavailability of lexical resources for the language, we employ multiple word embeddings and deep neural network to develop SRL. The proposed model can avoid the possible errors in parsing that can possibly propagate to the SRL prediction. The model implicitly learns the semantic similarity of words, the morphological information about words and the syntactic structure using concatenation of multiple word embeddings, character embeddings and BiLSTM networks, respectively.

IV. THE PROPOSED SYSTEM

The proposed Amharic SRL system uses concatenations of multiple word embeddings and deep neural network. First, the raw input sentences in the corpus are preprocessed for data cleaning and character normalization. After preprocessing, words are represented by concatenating various types of word embeddings. Since context-sensitive word representation is critical for semantic role labeling, a sequence encoding recurrent neural network called BiLSTM is applied on the top of concatenated word embedding. By concatenating the forward and backward contextual representation of a given word in an input sentence, context-sensitive representation of the word is generated which captures long-range dependencies of the target word in an input sentence from its previous and next words. The neural architecture also has fully connected layer before the sequence encoder layer. Finally, at the sequence decoding layer, the linear-chain conditional random field (CRF) machine learning algorithm with Viterbi-decoding algorithm is used to predict the semantic role of each word in an input sentence by capturing the dependency among adjacent semantic role labels. In addition, the model has a dropout layer on the top of the concatenated word embedding layer to prevent overfitting problem. It also has another dropout layer above the contextual word embedding layer.

Mathematically, the entire model contains 18 trainable weight matrices that transform an input vector from the lower layer of the deep neural network into an output vector of the next layer of the network using matrix-vector product. In order to make the presentation simple, these trainable weight matrices are indexed with integer numbers starting from one, and they are denoted by $M_d \in R^{o_d \times i_d}$, where $1 \leq d \leq 18$. In this notation, i_1, i_2, \dots, i_{18} represent the dimension of input vectors from the lower layer of network. Similarly, o_1, o_2, \dots, o_{18} represent the dimension of the output vector at the next layer of the network. In order to show

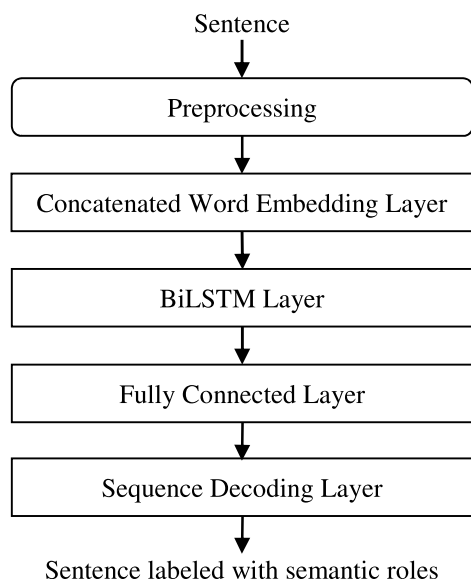


Figure 4. System architecture for Amharic SRL.

the sequence of operations that are executed by the proposed system, its architecture is presented hierarchically at three levels of abstraction.

- At the first level of abstraction, the proposed architecture has five main modules as shown in Figure 4.
- At the second level of abstraction, the Concatenated Word Embedding and BiLSTM modules are shown in Figure 5 and Figure 8, respectively.
- At the third level of abstraction, the two sub-modules of the Concatenated Word Embedding layer (Character-based Word Embedding and Sentence-based Word Embedding) are presented in Figure 6 and Figure 7, respectively.

A. PREPROCESSING

In the preprocessing module, each sentence in the corpus is cleaned by removing non-Amharic characters, double white spaces between words and all punctuation marks except the end of an Amharic sentence marker (፡፡). The text is also cleaned by separating hyphenated words into two separate words. For example, the hyphenated word ስነ-ጥበብ /sənä-ጥጥጥ/ is separated into two words by a single white space separator, namely, ስነ /sənäl and ጥጥጥ /ጥጥጥ/. After cleaning each sentence in a given corpus, each input sentence in the corpus is tokenized into words by using whitespace as a word separator. Semantic role labeling is implemented as a sequence labeling task for a given input sentence which is a sequence of a finite number words. However, the number of tokens in each input sentence is different. To address this problem, each sentence is padded if its length is shorter than the length of the longest sentence in the corpus.

B. CONCATENATED WORD EMBEDDING

Word embedding is a feature learning method where a word from the vocabulary is mapped to N dimensional vector.

As shown in Figure 5, we developed the prototype of the proposed neural SRL system stage by stage.

Since Amharic is a morphologically rich language, at first stage, we trained and tested the neural-based SRL model by using only character-based word embedding at the first core layer of the network model. Since the performance of the resulting model is promising, a sentence-based word embedding is added to complement the character-based word embedding because character-based word embedding captures local information of a given word whereas the sentence-based word embedding captures the global information of a word in a given input sentence. Thus, at the second stage, the network model is trained and tested by concatenating the character-based and sentence-based word embedding types. The performance of the system is improved when the above two embeddings are used jointly. This, in turn, motivates us to add word-based embedding of a word to supplement the above two-word embedding types by capturing the semantic feature of a word in a given vocabulary. In word-based word embedding, words with similar meanings have similar vector representation. Therefore, at the final stage, the network model is trained and tested using the concatenation of word-based, character-based and sentence-based word embedding types. The concatenated word embedding layer is not context-sensitive because its components are not context-sensitive word embeddings. Each of the components of the concatenated word embedding layer are described in detail in the following subsections.

1) WORD-BASED EMBEDDING

Word embedding layer is the first core layer of neural-based SRL models. This layer of the model can be formed from a pre-trained word embedding layer using unannotated corpus by applying existing neural word embedding algorithms such as word2vec [30], fastText [31] and GloVe [32]. However, the proposed model did not use a pre-trained neural word embedding algorithm to create the word-based embedding component of the concatenated word embedding layer. As a result of this, the word-based embedding component of the proposed network model is initialized randomly and trained with other layers of the network using the objective function of the network at its sequence decoding layer instead of using the objective function of the existing neural word embedding algorithms. The word-based embedding component of the proposed network model is computed at two steps. At the first step, a vocabulary of words is created by extracting a list of unique words from a list of Amharic sentences in a corpus. In the vocabulary, words are sorted and indexed with integer numbers starting from zero. The index number 0 is reserved for the padding of a given sentence. The index number 1 is also reserved for words that are not in the word vocabulary and considered as an unknown word. At the second step, the 1^{st} trainable weight matrix of the network model called word-based embedding matrix is created for the vector representation of each word in a given vocabulary. This matrix is denoted by $M_1 \in R^{o_1 * i_1}$, where i_1 is the dimension of a

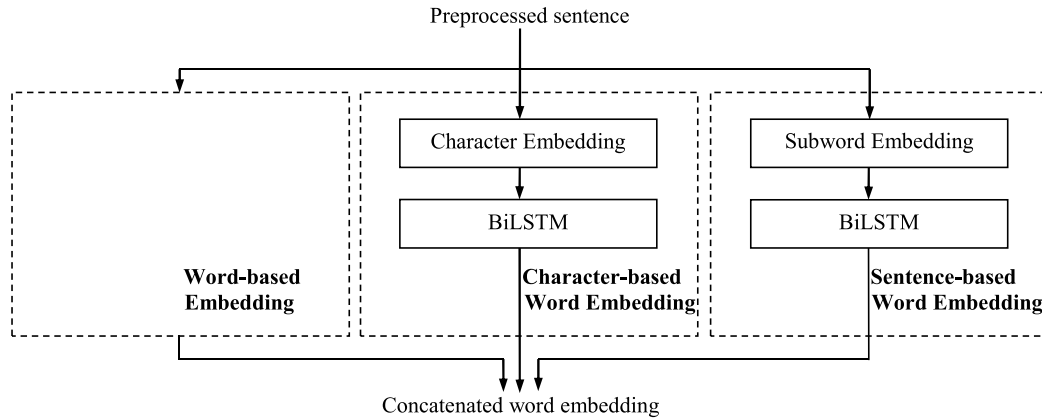


Figure 5. Concatenated word embedding layer.

one-hot encoding vector of a word which is the same as the number of words in the vocabulary, and o_1 is a hyperparameter that represents the dimension of a word-based embedding vector. This implies that the k^{th} column vector of this matrix, $m_1^k \in R^{o_1}$, corresponds to the word-based embedding of the word at the k^{th} index in the word vocabulary. Thus, the one-hot-encoding vector representation of the k^{th} word in the word vocabulary, $w_k \in R^{o_1}$, is transformed into its word-based embedding vector E_{word}^k by using matrix-vector product operation as shown below.

$$E_{word}^k = m_1^k = M_1 w_k, \quad \text{for } 1 \leq k \leq i_1 \quad (1)$$

This implies that when each word in an input sentence is processed by SRL system, the word-based embedding of the word is retrieved from this matrix by lookup table using the index of the word in a given vocabulary [30], [31], [32]. Different words which have the same meaning have similar word embedding vector representation. With the word-based embedding we can be able to capture the semantic similarity of words, such as መኪና /mäkina ‘car’/ and ተሻጃቅሪ /täškärkari ‘vehicle’/, and words which are written using different shapes (characters) but are the same, such as ሰው and ሠው /säw ‘man’/.

2) CHARACTER-BASED WORD EMBEDDING

Since Amharic language is a morphologically rich language, character-based word embedding is the core component of the concatenated word embedding layer of the proposed model. As shown in Figure 6, character-based embedding of a word is created from a sequence of its character embeddings by applying a bidirectional recurrent neural network on top of character embedding layer [33], [34]. The character embedding layer and BiLSTM layer on the top of this layer are the two main components of the character-based word embedding layer as described below.

a: CHARACTER EMBEDDING

At the character embedding layer of the proposed network model, an embedding vector is created for each character

of a given word. Similar to word embedding, character embedding is learned by training the layer with other layers of the network using the objective function of the network at its sequence decoding layer instead of using the objective function of the existing neural embedding algorithms [30], [31], [32]. This layer is created in two stages. At the first stage, a vocabulary of characters is created by extracting a list of unique Geez characters and Arabic numerals from a word vocabulary. Similar to word vocabulary, all characters in the character vocabulary are sorted and indexed with integer numbers starting from zero. The index number 0 is reserved for the padding of a given word because each word contains different number of characters. At the next stage, the 2^{nd} trainable weight matrix of the network model called character embedding matrix is created to generate the vector representation of each character in a given character vocabulary. This matrix is denoted by $M_2 \in R^{o_2 * i_2}$, where i_2 is the dimension of a one-hot encoding vector of a character which is the same as the number of characters in the character vocabulary and o_2 is a hyperparameter that represents the dimension of the character embedding vector. The j^{th} column vector of this matrix, $M_2^j \in R^{o_2}$, corresponds to the embedding of the character at the j^{th} index in the character vocabulary. The one-hot-encoding vector representation of the j^{th} character, $c_j \in R^{o_2}$, is transformed into its character embedding vector E_{char}^j by using a matrix-vector product operation as shown in Eq. 2. When each character in a word is processed by SRL system, the character embedding of each character is retrieved from the character embedding matrix $M_2^j \in R^{o_2 * i_2}$ by lookup table using the index of the character as a search key as shown in Figure 6. The character embedding layer in Eq. 2 is used as input to BiLSTM layer to generate character-based word embedding as shown in the following subsection.

$$E_{char}^j = m_2^j = M_2 c_j, \quad \text{for } 1 \leq j \leq i_2 \quad (2)$$

b: BiLSTM ON TOP OF CHARACTER EMBEDDING

To generate the character-based word embedding of a given word from its character embeddings, BiLSTM recurrent

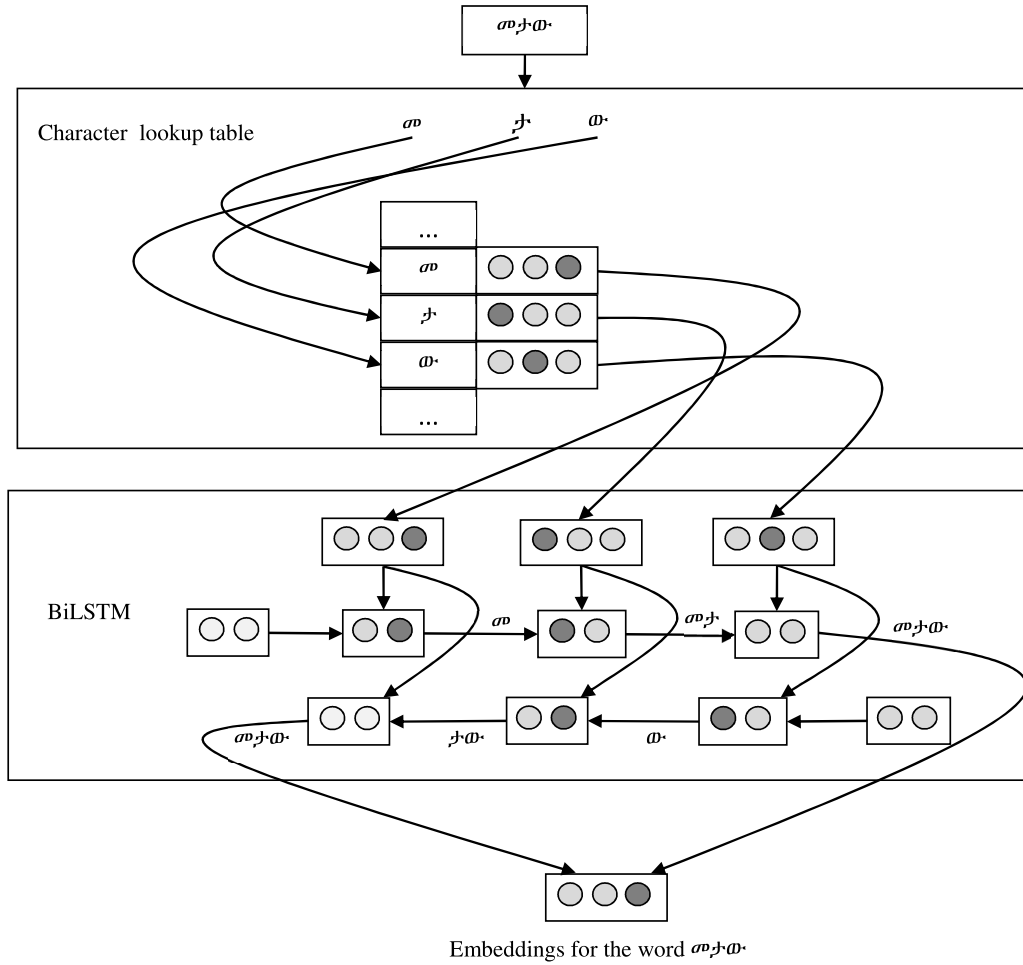


Figure 6. Character-based word embedding.

network is applied on the top the character embedding layer. The i^{th} word w_i in an input sentence is encoded into a single vector from a sequence of its character embeddings using Eq. 3.

$$h^i = BiLSTM(w_i) = BiLSTM(E_{char}^1, E_{char}^2, \dots, E_{char}^t, \dots, E_{char}^n) \quad (3)$$

where n is the number of characters in the i^{th} word, $w_i = (E_{char}^1, E_{char}^2, \dots, E_{char}^t, \dots, E_{char}^n)$ is the representation of the i^{th} word in terms of its character embeddings using Eq. 2. To generate the hidden vector of the i^{th} word h^i , the BiLSTM layer uses the 3rd - 6th trainable weight matrices of the network model [34]. These matrices are denoted by $M_3^f \in R^{o_3 \times i_3}$, $M_4^f \in R^{o_4 \times i_4}$, $M_5^b \in R^{o_5 \times i_5}$, and $M_6^b \in R^{o_6 \times i_6}$. The matrices M_3^f and M_4^f represent the hidden-to-hidden and input-to-hidden matrices of the forward LSTM layer, respectively. Similarly, the matrices M_5^b and M_6^b represent the hidden-to-hidden and input-to-hidden matrices of the backward LSTM layer, respectively. For these matrices, the dimension of the input vector is the same as the dimension of the character embedding vector, while the dimension of the output vector

is the same as the number of the hidden units of the LSTM layer. As shown in Eq. 3, if n is the total number of characters in a given word, the hidden vector of a target character at time t in a given word is generated using Eq. 4 and Eq. 5. Finally, if T is the number of words in an input sentence, the character-based embedding vector of the i^{th} word in an input sentence is generated from the concatenation of the hidden vectors of the first and last characters of the word as shown in Eq. 6. Note that the dimension of the hidden vector h^i is the sum of the number of hidden units of the forward LSTM and backward LSTM.

$$h_t^f = M_3^f h_{t-1}^f + M_4^f x_t, \quad \text{for } 1 \leq t \leq n \quad (4)$$

$$h_t^b = M_5^b h_{t+1}^b + M_6^b x_t, \quad \text{for } 1 \leq t \leq n \quad (5)$$

$$E_{char}^i = h^i = h_1^f \oplus h_n^b, \quad \text{for } 1 \leq i \leq T \quad (6)$$

where $x_t \in m_2^f$, $h_0 = 0$, and $h_{n+1} = 0$. In this regard, the character-based embedding for the character ከ /käl/ in the sentence ከበደና ለበበ ከለጎደጎ ሙጦ /käbädäna ጎäbäbä käländän mäṭB-u ‘Kebede and Abebe came from London’/ is treated differently. The character ከ /käl/ in the words ከበደና ለበበ /käbädäna ጎäbäbä ‘Kebede and Abebe’/ reflects an

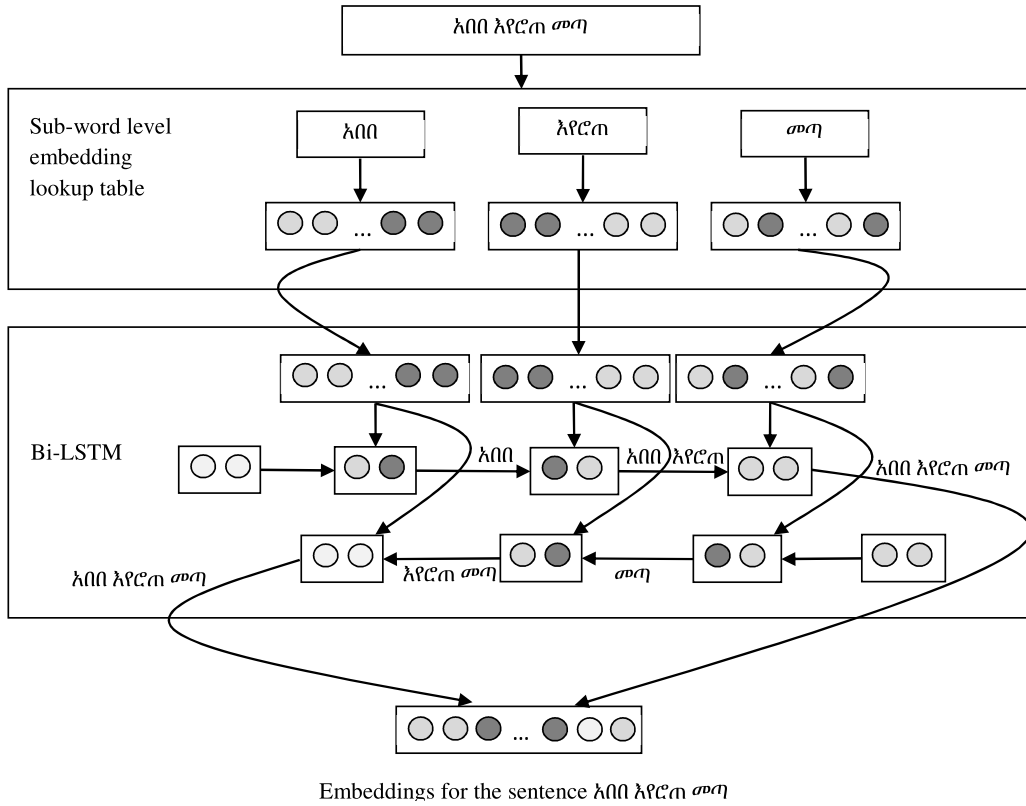


Figure 7. Sentence-based embedding from subword-based word embedding.

agent (ARG1), while the character ከ /käl/ in the word ከአጎደጎ /käländän ‘from London’/ reflects the direction (ARGM-DIR) the agents came from.

3) SENTENCE-BASED WORD EMBEDDING

The sentence-based word embedding component of the concatenated word embedding layer is learned from the embedding of all words in an input sentence by applying a bidirectional recurrent neural network layer on the top of pre-trained subword-based word embedding layer as shown in Figure 7. This representation extracts the global information of a word by capturing the morpho-syntactic features of the word in the input sentence [35], [36], [37], [38]. Hence, the three components of the sentence-based word embedding layer are subword embedding, subword-based word embedding and BiLSTM on the top of subword-based word embedding.

a: SUBWORD EMBEDDING

The subword embedding layer is pre-trained in three steps. At the first step, a vocabulary of subwords is created by extracting a list of unique subwords from a word vocabulary. All the subwords in the subword vocabulary are sorted and indexed with integer numbers starting from zero. At the second step, the 7^{th} trainable weight matrix of the model called subword embedding matrix is created and initialized

randomly. This matrix is denoted by $M_7 \in R^{o_7 * i_7}$, where i_7 is the dimension of a one-hot encoding vector of a subword which is the same as the number of subwords in the subword vocabulary, and o_7 is a hyperparameter that represents the dimension of a vector of a given subword. The r^{th} column vector of this matrix, $m_7^r \in R^{o_7}$, corresponds to the embedding of the r^{th} subword in the subword vocabulary. The one-hot-encoding vector representation of the r^{th} subword, $sub_r \in R^{i_7}$, is transformed into its subword embedding vector E_{sub}^r by using matrix-vector product as shown in Eq. 7.

$$E_{sub}^r = m_7^r = M_7 Sub_r, \quad \text{for } 1 \leq r \leq i_7 \quad (7)$$

When each subword in a given word is processed by SRL system, the subword embedding of each subword is retrieved from the subword embedding matrix $M_7 \in R^{o_7 * i_7}$ by lookup table using the index of the subword as a search key as shown in Figure 7. At the third step, the subword embedding matrix is trained using the training objective function of the existing neural embedding architecture called *fastText* [31].

b: SUBWORD-BASED WORD EMBEDDING

If the number of subwords of an i^{th} word is q , the subword-based embedding of the i^{th} word is computed from the average of its subword embedding vectors using Eq. 8.

$$E_{sub}^i = Average(E_{sub}^1, E_{sub}^2, \dots, E_{sub}^j, \dots, E_{sub}^q) \quad (8)$$

Since this representation captures the morphological structure of a word through character n-grams, it is appropriate for a morphologically rich language like Amharic. The pre-trained subword-based word embedding layer is used as input to BiLSTM layer to generate sentence-based word embedding as shown in the following subsection.

c: BiLSTM ON TOP OF THE SUBWORD-BASED EMBEDDING

BiLSTM layer is applied on the top the subword-based word embedding layer to encode a sentence into a single vector from a sequence of subword-based embeddings of words in an input sentence as shown in Eq. 9.

$$h_s = \text{BiLSTM}(S) \\ = \text{BiLSTM}(E_{sub}^1, E_{sub}^2, \dots, E_{sub}^t, \dots, E_{sub}^T) \quad (9)$$

where, T is the total number of words in an input sentence, and $S = (E_{sub}^1, E_{sub}^2, \dots, E_{sub}^t, \dots, E_{sub}^T)$ is the representation of an input sentence in terms of its subword-based word embeddings. To generate the hidden vector h_s , the 8th - 11th trainable weight matrices of the network model are created and trained by the BiLSTM layer. These matrices are denoted by $M_8^f \in R^{o_8 \times i_8}$, $M_9^f \in R^{o_9 \times i_9}$, $M_{10}^b \in R^{o_{10} \times i_{10}}$ and $M_{11}^b \in R^{o_{11} \times i_{11}}$. The matrices M_8^f and M_9^f represent the hidden-to-hidden and input-to-hidden matrices of the forward LSTM layer, respectively. Similarly, the matrices M_{10}^b and M_{11}^b represent the hidden-to-hidden and input-to-hidden matrices of the backward LSTM layer, respectively. For these matrices, the dimension of the input vector is the same as the dimension of the subword-based word embedding vector, while the dimension of the output vector is the same as the number of the hidden units of the LSTM layer.

Since T is the total number of words in a given sentence, the hidden vector of a target word at time t in a sequence of words of a given sentence is generated by the above matrices using Eq. 10 and Eq. 11. Finally, the sentence-based word embedding vector E_{sen}^t of the t^{th} word in an input sentence is generated from the concatenation of the hidden vectors of the first and last words of the t^{th} word Eq. 12. The dimension of this embedding vector is the sum of the number of hidden units of the forward LSTM and backward LSTM.

$$h_t^f = M_8^f h_{t-1}^f + M_9^f x_t, \quad \text{for } 1 \leq t \leq T \quad (10)$$

$$h_t^b = M_{10}^b h_{t+1}^b + M_{11}^b x_t, \quad \text{for } 1 \leq t \leq T \quad (11)$$

$$E_{sen}^t = h_s = h_1^f \oplus h_1^b \quad (12)$$

where $x_t \in E_{sub}^t$, $h_0 = 0$, and $h_{T+1} = 0$. With this sentence-based embedding of a word, since we are leaning a representation of an input sentence from a sequence of its words using BiLSTM layer, the character ን ከቤደ ን ከበበ ከአንደን ሠጡ *Ikäbädäna ?äbäbä käländän mä?B-u* ‘Kebede and Abebe came from London’/, which could have been a patient modifier (ARG1) or object marker in other word sequences, such as ኢዮብ አበበን ሠጡ */?ijob ?äbäbän mäta?w* ‘Eyob beat Abebe’/, is recognizing ከአንደን *Ikäländän* ‘from London’/ as the direction the agents

ከበበ *Ikäbädäna ?äbäbä* ‘Kebede and Abebe’/ came from. Finally, the concatenated word embedding of the t^{th} target word in an input sentence is formed from E_{word}^t , E_{char}^t and E_{sen}^t using Eq. 13. Since the concatenated embedding layer has character-based and subword-based word embedding components, it helps to deal with the out-of-vocabulary problem which is a common problem for data driven natural language processing applications.

$$w_{concat}^t = E_{word}^t \oplus E_{char}^t \oplus E_{sen}^t \quad (13)$$

C. CONTEXTUAL WORD EMBEDDING

The concatenated word embedding at the first core layer is not context sensitive. However, for higher label NLP applications like SRL, contextual word embedding is a crucial component. As a result of this, a context-sensitive embedding of a target word in an input sentence is generated at the second core layer of the network model. This is accomplished by applying BiLSTM sequence encoding layer on the top the concatenated word embedding layer as shown in Figure 8. The contextual embedding of a target word in an input sentence is created by concatenating the left context hidden vector from forward LSTM layer and the right context hidden vector from backward LSTM layer.

To accomplish this, the 12th - 15th trainable weight matrices of the model are created and trained. These matrices are denoted by $M_{12}^f \in R^{o_{12} \times i_{12}}$, $M_{13}^f \in R^{o_{13} \times i_{13}}$, $M_{14}^b \in R^{o_{14} \times i_{14}}$ and $M_{15}^b \in R^{o_{15} \times i_{15}}$. The matrices M_{12}^f and M_{13}^f represent the hidden-to-hidden and input-to-hidden matrices of the forward LSTM layer, respectively. Similarly, the matrices M_{14}^b and M_{15}^b represent the hidden-to-hidden and input-to-hidden matrices of the backward LSTM layer, respectively. For these matrices, the dimension of the input vector is the same as the dimension of the concatenated word embedding layer, while the dimension of the output vector is the same as the number of the hidden units of the LSTM layer. If T is the total number of words in a given sentence, the forward and backward context-sensitive representation of the target word at the t^{th} location in an input sentence is generated by the above matrices using Eq. 14 and Eq. 15. By concatenating the two unidirectional context-sensitive hidden vectors, the directional context-sensitive embedding of the t^{th} target word in an input sentence, E_{conxt}^t , is generated as shown in eq.(16).

$$h_t^f = M_{12}^f h_{t-1}^f + M_{13}^f x_t, \quad \text{for } 1 \leq t \leq T \quad (14)$$

$$h_t^b = M_{14}^b h_{t+1}^b + M_{15}^b x_t, \quad \text{for } 1 \leq t \leq T \quad (15)$$

$$E_{conxt}^t = h_t = h_t^f \oplus h_t^b \quad (16)$$

where $x_t \in E_{concat}^t$, $h_0 = 0$, and $h_{T+1} = 0$. The context-sensitive representation extracts the left and right contexts of the target word in an input sentence. This embedding can be used to identify the boundary of an argument, and argument boundary detection is very crucial for the performance of the SRL. For example, in the Amharic sentences ሰው ከፋ ነው */säw kəfu näw* ‘Man is bad’/ and ከፋ ሰው ነው

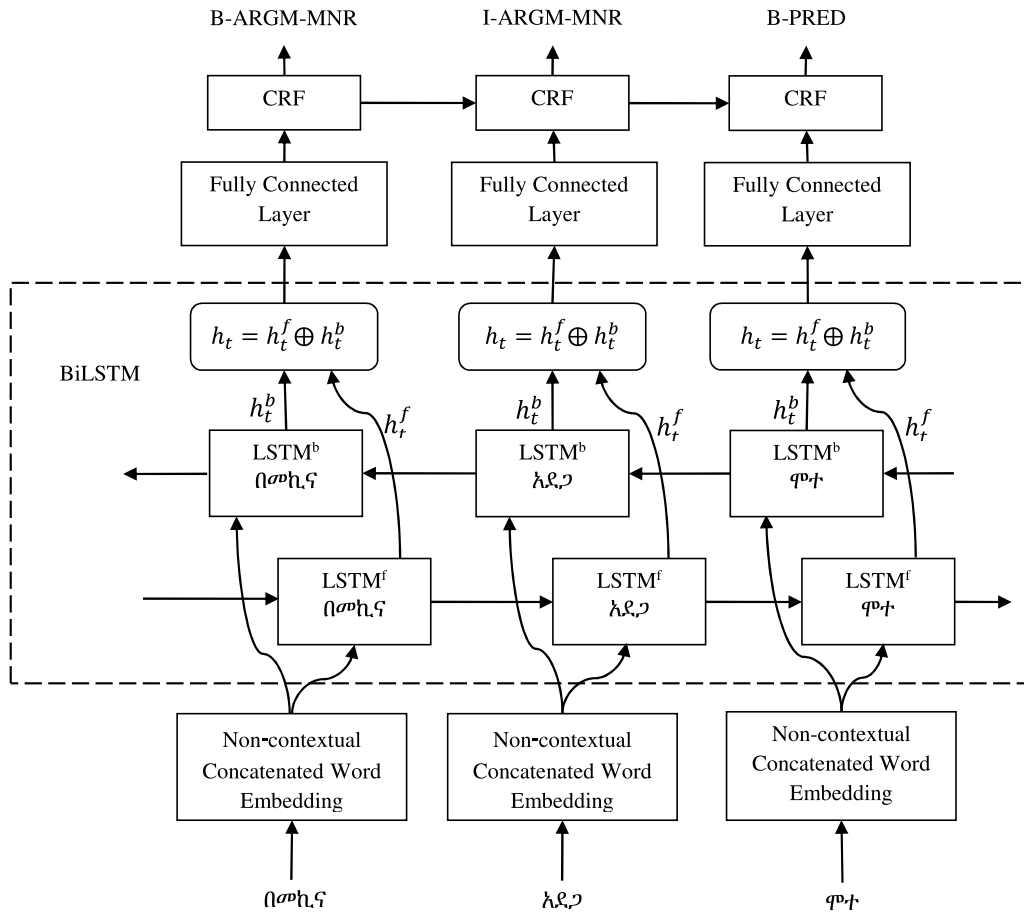


Figure 8. Context-sensitive embedding layer using BiLSTM.

/kəfu säw näw ‘He is a bad man’/, the semantic role label of the word *ጠዎ* /*säw* ‘man’/ depends on the availability of an argument to the right of it (boundary of the argument). The verb *ነዎ* /*inäw*/ in the second sentence is inflected for subject *አሱ* /*ʔəsu* ‘He’/. The semantic role label of this word is an AGENT in the first sentence and THEME in the second sentence as shown below.

[*säw*]_{ARG0} [*kəfu*]_{ARG1} [*näw*]_{PRED}
 [*kəfu säw*]_{ARG1} [*näw*]_{PRED}

Moreover, the BiLSTM network can capture long-range dependencies which is crucial task in the SRL task for a predicate’s argument detection. For example, in the Amharic sentence *አሱ ንጥናንት ጠዎኪና አደጋ ሞተ ʔəsu tənanət bämäkina ʔädäga motä* ‘He died in a car accident yesterday’/, the phrasal argument *ንጥናንት* /*tənanət* ‘yesterday’/ expresses the temporal context (WHEN) of the verbal predicate *ሞተ* /*motä* ‘[he] died’/.

[*ʔəsu*]_{ARG1} [*tənanət*]_{ARG-TMP}
 [*bämäkina ʔädäga*]_{ARG-MNR} [*motä*]_{PRED}

D. FULLY CONNECTED LAYER

To complement the BiLSTM layer which generates context-sensitive representation of words in the input

sentence, a fully connected layer is added on the top of BiLSTM layer. This is done to capture any dependency present among input sentences in the dataset. This layer transforms the context-sensitive representation of words in an input sentence into their higher-level representation by learning more complex features present in the input sentences of a training corpus. At this layer, the 16th trainable weight matrix, $M_{16}^f \in R^{o_{16} \times i_{16}}$, of the model is created and trained. We have accounted for the morphological, syntactic and semantic features with the concatenated word embeddings and the BiLSTM layer, but the Fully Connected Layer component can learn dependencies present between the semantic roles, themselves. For example, consider the sentence *አሱ ከሶስት ቀን በኋላ ተወለደ ʔəsu kəsosət qän bähala täwälädä* ‘he was born three days later’/ where *በኋላ* /*bähuala* ‘later’/ is misspelled as *በኋላ* /*bähahal*/. The semantic role of the word sequence *ከሶስት ቀን* /*kəsosət qän* ‘[from] three days’/, which could have been a temporal argument (ARGM-TMP) for the verb-predicate *ተወለደ* /*täwälädä* ‘[he] was born’/, seems to depend on the semantic role of the word *በኋላ* /*bähahal*/, as shown below.

[*ʔəsu*]_{ARG1} [*kəsosət qän*]_{ARG-DIR}
 [*bähala*]_{ARGM-MNR} [*täwälädä*]_{PRED}

Here, the Fully Connected Layer is learning the dependency between the semantic roles ARGM-DIR and ARGM-MNR from the dataset. However, if we correct the spelling error, the semantic role for word sequence ከሰሰጉ ቀን /*käsosət qän*/ along with the word ባሕላል /*bähualal*/ maintains its temporal nature in the sentence የሰሰ ከሰሰ ባሕላል ተሰሰ, as shown below.

[የሰሰ]_{ARG1} [ከሰሰ ባሕላል]_{ARGM-TMP}
[ተሰሰ]_{PRED}

E. SEQUENCE DECODING LAYER

A very simple sequence labeling method is to use the hidden vector h_c of the current word w_c in an input sentence to predict its label L_c without considering the semantic role label of the previous word L_{c-1} . However, this method has limitation because there is a strong dependency across the output semantic role labels. For example, in IOB sequence labeling format, I-ARG0 label must come after another I-ARG0 label or B-ARG0 label. Moreover, B-ARG1 label must not come after I-ARG1 label or another B-ARG1 label. When semantic role labels are predicted for each word in an input sequence, it is essential to decode the labels jointly by considering the correlations between neighboring labels. There are two methods to assign the semantic role label of the next word based on the semantic role of its previous word in an input sentence. The first method is to predict a distribution of semantic role labels for each time step and using beam decoding to find correct sequence of semantic role labels. Maximum entropy classifier and Maximum entropy Markov model fall in this category [39], [40]. The second method considers the entire sequence instead of individual positions in the input sentence. Conditional Random Field (CRF) belongs to this category. As a result of this, the proposed model uses a linear-chain version of CRF at the sequence decoding layer. The Sequence Decoding Layer is connected directly with Fully Connected Layer which, in turn, is connected to BiLSTM layer. Thus, the CRF uses past input features from a LSTM layer indirectly.

Before applying sequence decoding operation, the network model should be trained by using the objective function of the linear-chain CRF algorithm. To accomplish this, the 17th trainable state emission score matrix $M_{17} \in R^{o_{17} \times i_{17}}$ and the 18th trainable state transmission score matrix $M_{18} \in R^{o_{18} \times i_{18}}$ are created, where i_{17} is the dimension of the input vector which is the same as the number of hidden units of a fully connected layer, while o_{17} is the dimension of the output vector which is the same as the number of semantic role labels in the PropBank corpus. Since matrix M_{18} is a square matrix, the dimensions i_{18} and o_{18} are the same. These matrices have probabilistic interpretation, and the state emission score matrix M_{17} is used to compute the conditional probability distribution of the current word over the set of semantic role labels given the current word. The state transmission matrix M_{18} is used to compute the conditional probability distribution of the current word over the set of semantic role labels given the semantic role of the previous word. These two matrices are position independent across time.

Mathematically, let $Y = \{y_1, y_2, \dots, y_k\}$ be the set of semantic role labels and $V = \{v_1, v_2, \dots, v_m\}$ be the set of words in a given vocabulary. Let $W = \{w_1, \dots, w_c, \dots, w_n\}$ be a sequence of n words in an input sequence and $L = \{L_1, \dots, L_c, \dots, L_n\}$ be a sequence of predicted semantic role labels corresponding to a word sequence W where $w_c \in V$ and $L_c \in Y$. For a given sequence of words W , let $E = \{e_1, \dots, e_c, \dots, e_n\}$ be a sequence of state emission score vectors where $e_c = \{e_{(c,1)}, e_{(c,2)}, \dots, e_{(c,k)}\}$ is a vector of state emission scores corresponding to the current word w_c . The state emission matrix M_{17} assigns emission score to each possible semantic role labels for the current word w_c by transforming the vector representation of the current word w_c from the fully connected layer to sequence decoding layer of the network. Moreover, for a given sequence of labels L , let $T = \{t_1, \dots, t_n\}$ be a sequence of state transition score vectors, where $t_c = \{t_{(c,1)}, t_{(c,2)}, \dots, t_{(c,k)}\}$ is a vector of state transmission scores corresponding the current word w_c . This implies that the state transmission matrix M_{18} assign transition score to each possible semantic role labels for the current word w_c by transforming the vector representation of the previous semantic role label L_{c-1} at the sequence decoding layer of the network into the vector representation of the current semantic role label L_c at the sequence decoding layer of the network when the input sequence is scanned from left to right.

For an input sentence, we can consider E and T as score matrices with size $n \times k$ and $k \times k$, respectively. Hence, $E_{i,j}$ represents the score of the j^{th} semantic role label for the i^{th} word in an input sentence W . Similarly, $T_{i,j}$ represents the score of a transition from the i^{th} semantic role label y_i of the previous word w_{c-1} to the j^{th} semantic role label y_j of the current word w_c . Using matrix E and matrix T , the objective function of the entire network is derived by computing the score value of the sequence of predicted role labels L for a given sequence of words W as follows.

$$\begin{aligned} \text{Score}(W, L) &= S(W, L) \\ &= \sum_{c=1}^n T_{(L_{c-1}, L_c)} + \sum_{c=1}^n E_{(w_c, L_c)} \end{aligned} \quad (17)$$

where $L_{c-1} \in \{y_1, y_2, y_3, \dots, y_k\}$, $L_c \in \{y_1, y_2, y_3, \dots, y_k\}$ and $w_c \in \{v_1, v_2, v_3, \dots, v_k\}$ for $1 \leq c \leq n$. Note that $T_{i,j}$ represents the score of a transition from label y_i to label y_j and $E_{(i,j)}$ represents the score of the j^{th} label y_j of the i^{th} word w_i in an input sentence. Note that, y_0 is the starting label of the sequence L . Let A be the set of all possible sequence of semantic role labels called a path. Since n is the length of a sequence and there are k possible number of semantic roles labels, the total number of possible sequences or paths is k^n . Using a softmax function, the conditional probability of a specific sequence of semantic role labels $L \in A$ given a specific word sequence W is computed using Eq. 18. By applying log function on both sides of Eq. 18, the log-probability of the best tag sequence is computed using Eq. 19. During the training phase of the network model, the objective function in

Eq. 19 is optimized. During the testing phase of the network, the Viterbi sequence decoding algorithm is used to predict the sequence of semantic role labels L^* or a path that maximized the score function as in Eq. 20. When the value of k and n are large, it is not practical to compute the score of all the possible number of paths one by one. To address this problem, the label of the current word depends only on the label of the previous word instead of depending on the labels of the previous two or more words. In other words, the model uses only bigram relation among output labels. As a result of this, Eq. 19 and Eq. 20 are computed using dynamic programming method instead of computing all the k^n number of possible paths.

$$P(L|W) = \frac{e^{S(W,L)}}{\sum_{\hat{L} \in A} e^{S(W,\hat{L})}} \quad (18)$$

$$\log[P(L|W)] = e^{S(W,L)} - \log[\sum_{\hat{L} \in A} e^{S(W,\hat{L})}] \quad (19)$$

$$L^* = \operatorname{argmax}_{\hat{L} \in A} [S^{(W,\hat{L})}] \quad (20)$$

V. EXPERIMENT

A. DATASET COLLECTION

To increase the generalizing ability of the model for a generic data, we collected Amharic text from different data sources on the web such as the Amharic Bible, Addis Zemen Gazette, The Ethiopian Reporter Newspaper, Ethiopian Federal Negarit Gazette, and Amharic-English machine translation corpus, Amharic Part-of-speech tagging corpus, Amharic spell corrector corpus, and list of Ethiopic names in the GitHub. The collected corpus is encoded in UTF. From the collected text, 10,000 sentences were selected which follow the basic Amharic subject-object-verb (SOV) word-order and they have a single verb predicate. The maximum number of tokens in each sentence is 32 (31 words and the ‘ :: ’ character that indicates the end of an Amharic sentence). All sentences in the corpus are annotated semantically in IOB tagging schema and stored in a tab-separated text file. This is done by labeling each word of a given sentence with one of the predefined ProBank semantic role labels. The total number of predefined PropBank semantic roles considered is 23 including the predicate tag PRED and padding tag PAD.

A word dictionary is created by extracting 526,370 unique words. In addition to the unique words that are contained in 10,000 sentences (which amount to 20,562), the dictionary also includes unique words from other sources to address the out-of-vocabulary problem when the model has to label a sentence which contains words that are not part of the training sentences. A character dictionary is also created. This dictionary contains 365 unique Amharic characters and other characters that can be used in Amharic sentences. Most of the Amharic characters are included in this dictionary to avoid out-of-vocabulary problem because the embedding of out-of-vocabulary word is computed from the embedding of its characters. From the entire dataset, 80% is used as training data and 20% is used as testing data.

The number of samples in ARG3 and ARG4 classes in the entire dataset is extremely low in comparison to other classes.

Table 2. The entire dataset without IOB format.

N ^o	Label	Frequency		
		Total	Training	Test
1	ARG0	6850	5484	1366
2	ARG1	9939	7953	1986
3	ARG2	1511	1183	328
4	ARG3	6	5	1
5	ARG4	12	9	3
6	ARGM-DIR	817	647	170
7	ARGM-LOC	771	631	140
8	ARGM-MNR	2197	1780	416
9	ARGM-PRP	645	518	127
10	ARGM-TMP	623	488	135
11	PREDICATE	10000	8000	2000
12	<PAD>	429928	379816	50112

Unlike the case of English, there are no sufficient cases for ARG3 and ARG4 semantic role labels in Amharic sentences due to the intrinsic and unique characteristics of the language. These two labels form minority classes causing the training data to be unbalanced. For non-sequential traditional machine learning classifiers, an intrinsic class imbalance problem can be minimized by increasing the number of samples in the minority class using random up-sampling method by fixing the number of samples in the majority class. However, it is not applicable to make random up-sampling method for semantic role label which is a special type of sequence label. When a sample sentence constructed from words with rarely occurring semantic role labels is up-sampled using random up-sampling method, other words with frequently occurring semantic role labels are also up-sampled instead of remaining constant. Since the task of the proposed end-to-end semantic role labeling system is treated as predicate-argument-role triplets unlike traditional machine learning classifiers, it is impossible to address the imbalance data problem using random up-sampling method. Moreover, the primary purpose of semantic role labeler is to correctly predict the majority class instead of minority classes. Since ARG3 and ARG4 semantic role labels occur rarely, we can consider them as outliers or noise. Hence, we created a new dataset (consisting 9,987 sentences), which is a relatively balanced dataset, by removing the two outlier classes and their corresponding sentences from the original unbalanced dataset (consisting 10,000 sentences). Similar to the case of the entire dataset, we used 80% for training and 20% for testing in the relatively balanced dataset. Tables 2 and 3 show the statistics of the entire dataset and relatively balanced dataset without IOB format. Similarly, we created IOB-based datasets for the entire dataset and the relatively balanced dataset as presented in Tables 4 and 5. Thus, experiments were carried out with the following four datasets:

- The entire dataset without IOB format
- The relatively balanced dataset without IOB format
- The entire dataset with IOB format
- The relatively balanced dataset with IOB format.

Table 3. The relatively balanced dataset without IOB format.

N ^o	Label	Frequency		
		Total	Training	Test
1	ARG0	6846	5477	1369
2	ARG1	9926	7938	1988
3	ARG2	1511	1,175	336
4	ARGM-DIR	817	701	166
5	ARGM-LOC	771	939	158
6	ARGM-MNR	2197	1,787	410
7	ARGM-PRP	654	527	118
8	ARGM-TMP	623	489	134
9	PREDICATE	9987	7,989	1998
10	<PAD>	429389	379,237	50052

Table 4. The entire dataset with IOB format.

N ^o	Label	Frequency		
		Total	Training	Test
1	B-ARG0	6850	5484	1366
2	B-ARG1	9939	7953	1986
3	B-ARG2	1511	1183	328
4	B-ARG3	6	5	1
5	B-ARG4	12	9	3
6	I-ARG0	3319	2,639	680
7	I-ARG1	14898	12014	2884
8	I-ARG2	1416	1114	302
9	I-ARG3	3	2	1
10	I-ARG4	16	10	6
11	B-ARGM-DIR	817	647	170
12	B-ARGM-LOC	771	631	140
13	B-ARGM-MNR	2197	1780	416
14	B-ARGM-PRP	645	518	127
15	B-ARGM-TMP	623	488	135
16	I-ARGM-DIR	1148	906	242
17	I-ARGM-LOC	1097	884	213
18	I-ARGM-MNR	2502	2066	436
19	I-ARGM-PRP	1363	1108	255
20	I-ARGM-TMP	929	732	197
21	O	10010	8010	2000
22	PREDICATE	10000	8000	2000
23	<PAD>	429928	379816	50112

B. PARAMETER SETTING AND MODEL BUILDING

We trained many types of models by setting different values for different hyper-parameters of the model. We have used the Keras API with Tensorflow as the backend to implement the components of the model. In all models, the CRF objective function in Keras is used as the loss function [41]. This objective function is optimized with a variant of the Adam's stochastic gradient descent algorithm called Adamax optimizer by setting the learning rate to 0.001 and the constant (epsilon) is set to $\epsilon = 1e - 7$ for numerical stability. To avoid exploding gradients problem, the norm of gradients is clipped with a threshold of 1.0. All models were trained for 50 iterations (epochs) with mini-batches of size 4. However, the subword-based word embedding is pre-trained separately for 10 iterations. During training, the loss of the model was monitored at each iteration by setting the early stopping parameter to 5 as its patience. After doing an extensive experiment, the hyper-parameter settings of the best neural network model are summarized in Table 6. The

Table 5. The relatively balanced dataset with IOB format.

N ^o	Label	Frequency		
		Total	Training	Test
1	B-ARG0	6846	5477	1369
2	B-ARG1	9926	7938	1988
3	B-ARG2	1511	1,175	336
4	I-ARG0	3317	2,683	634
5	I-ARG1	14869	11,994	2875
6	I-ARG2	1416	1,084	332
7	B-ARGM-DIR	817	701	166
8	B-ARGM-LOC	771	939	158
9	B-ARGM-MNR	2197	1,787	410
10	B-ARGM-PRP	654	527	118
11	B-ARGM-TMP	623	489	134
12	I-ARGM-DIR	1148	909	239
13	I-ARGM-LOC	1097	854	243
14	I-ARGM-MNR	2502	2,079	423
25	I-ARGM-PRP	1363	1,116	247
26	I-ARGM-TMP	926	716	213
27	O	9997	7,996	2001
28	PREDICATE	9987	7,989	1998
19	<PAD>	429389	379,237	50052

values of the parameters are determined by evaluating their joint impact on the performance of the entire model. In the following subsections, a detailed description is given about the setting of each of the 18 trainable weight matrices of the network model that transforms the vector representation of a word at one layer of the network into its new vector representation at the next layer of the network.

C. SETTING PARAMETERS OF THE CONCATENATED WORD EMBEDDING LAYER

As described above, the Concatenated Word Embedding Layer contains the concatenation of three types of word embeddings, namely Word-based word embedding layer, Character-based word embedding layer and Sentence-based word embedding layer. The parameter setting of the trainable matrices associated to these embeddings are given in the following subsections.

1) SETTING PARAMETER OF WORD-BASED EMBEDDING MATRIX

The trainable word-based embedding matrix $M_1 \in R^{o_1 \times i_1}$ is created and initialized randomly by an "Embedding layer" of Keras. This API takes the dimension of the word-based embedding vector o_1 , the number of unique words in the vocabulary i_1 and the maximum number of words in an input sentence as its input hyper-parameters. The matrix is set to $M_1 \in R^{5 \times 526372}$ where:

- The number of unique words in the vocabulary is set to be 526,372.
- The maximum number of words in a sentence is set to be 32.
- The dimension of the word-based embedding vector is set to be 5.

When the dimension of the word-based embedding vector is more than 5, it is found that the similarity between words in vocabulary increases. This, in turn, results in model over-

Table 6. A summary of the hyper-parameter and training parameter settings of the best model.

Sequence Decoding Layer: 23 (number of semantic roles)		
Fully Connected Layer: 64		
Regular Dropout: 0.8(keeping rate)		
Contextual Embedding Layer: 64 (32+32)		
BiLSTM: 32		
Variational Dropout: 0.8 (keeping rate)		
Non-Contextual Concatenated Word Embedding Layer: 325 (5+256+64)		
Word-based Embedding Layer: 5	Character-based Word Embedding Layer: 56 (128+128)	Sentence-based Word Embedding: 64 (32+32)
	BiLSTM: 128 (hidden units)	BiLSTM: 32 (hidden units)
	Character Embedding Layer: 20	Subword-based Word Embedding Layer: 5 (Average of Subword Embeddings)
		Subword Embedding Layer: 5

fitting problem by losing the discriminating power of the embedding.

2) SETTING PARAMETERS OF CHARACTER EMBEDDING MATRIX

The trainable character embedding weight matrix $M_2 \in R^{o_2 \times i_2}$ is also created and initialized randomly by an ‘‘Embedding layer’’ of Keras API. This API takes the dimension of the character embedding vector o_2 , the number of unique characters in the character vocabulary i_2 and the maximum number of characters in a word as its input parameters. This implies that the trainable character embedding matrix is set to $M_2 \in R^{20 \times 367}$ by setting the parameters of the Embedding Layer of Keras as follows.

- The number of unique characters in the character dictionary is set to 367.
- The maximum number of characters in a word is set 15.
- The dimension of the character embedding vector is set to 20. When the dimension of the embedding vector increases above 20, it is found that the similarity between words increases. This, in turn, result in model overfitting problem.

3) SETTING PARAMETERS OF CHARACTER-BASED WORD EMBEDDING MATRIX

The character-based word embedding of a word is created and initialized by using BiLSTM layer of Keras on top of character embedding layer. The number of hidden units of this layer is set to 128 and its activation function is set to *tanh*. This implies that for forward LSTM layer, the trainable hidden-to-hidden matrix is set to $M_3^f \in R^{128 \times 128}$ and the trainable input-to-hidden matrix is set to $M_4^f \in R^{128 \times 20}$. Similarly, for the backward LSTM, the trainable hidden-to-hidden matrix is set to $M_5^b \in R^{128 \times 128}$ and the trainable input-to-hidden

matrix is set to $M_6^b \in R^{128 \times 20}$. When the number of hidden units is 128, all the possible morphological information of a word in a sentence can be captured by holding the discrimination power of the representations. Experimentally, it is found that a small number of hidden units resulted in overlapping representations. Moreover, increasing the number of hidden units above 128 resulted in poor representation by losing the discrimination power of the representations. Hence, the dimension of the character-based word embedding is 256 (128+128) by concatenating the hidden vectors of forward LSTM and backward LSTM. Empirical results show that *tanh* function performs well in comparison to other activation functions.

4) SETTING PARAMETERS OF THE SUBWORD WORD EMBEDDING MATRIX

The trainable subword-based word embedding matrix, $M_7 \in R^{o_7 \times i_7}$, is created and trained using Gensim’s *fastText* embedding API. This API takes the dimension of the subword-based embedding vector o_7 , the number of unique character n-grams in subword dictionary i_7 , the size of the local window of the target word, the minimum frequency of a word to be included in the character n-gram dictionary, the minimum value of n to construct character n -gram subwords from a given word, the maximum value of n to create character n -gram subwords from a given word as its input parameters [40]. The subword-based word embedding matrix is set to $M_7 \in R^{5 \times 70072}$ by setting the input parameters of the *fastText* word embedding algorithm as follows.

- The dimension of the subword-based embedding vector is set to 5.
- The number of unique character n -grams in the subword dictionary is set to 70072.

- The size of the local context window of the target word is set to 3 (3 words on the left and 3 words on the right of target word).
- The minimum frequency of a word to be included in the character n -gram dictionary is set to 1.
- The minimum value of n to construct character n -grams (subwords) from a given word is set to 1.
- The maximum value of n to create character n -grams (subwords) from given word is set to 6.

As the size of the local context window increases above 3, *fastText* word embedding algorithm suffers because the morphological features of a word cannot be captured when the size of the local context window increases. As a result of this, the model works well when the window size is set to 3. The subword-based word embedding matrix, $M7 \in R^{5*70072}$, is trained alone and it is frozen when all the remaining trainable weight matrices of the network model are updated jointly during the training phase of the entire model. Note that the subword-based word embedding of a given word is computed from the average of its subword embeddings.

5) SETTING PARAMETERS OF SENTENCE-BASED WORD EMBEDDING MATRIX

The sentence-based embedding of a word in an input sentence is computed from the embedding of its pre-trained subword-based embedding using BiLSTM layer of Keras. The number of hidden units of this layer is set to 32 and its activation function is set to tanh. Similar to the character-based word embedding, the morphological features of a word have already been captured by subword-based word embedding using *fastText* word embedding. As a result of this, a small number of hidden units (32) performs well by avoiding the model overfitting problem. This implies that for the forward LSTM layer, the trainable hidden-to-hidden matrix is set to $M_8^f \in R^{32*32}$ and the trainable input-to-hidden matrix is set to $M_9^f \in R^{32*5}$. Similarly, for the backward LSTM layer, the trainable hidden-to-hidden matrix is set to $M_{10}^b \in R^{32*32}$ and the trainable input-to-hidden matrix is set to $M_{11}^b \in R^{5*32}$. As a result of this, the dimension of the sentence-based word embedding is set to 64 (32+32) by concatenating the forward and backward hidden vectors.

6) SETTING CONCATENATED WORD EMBEDDING LAYER

The output of the word-based, character-based and sentence-based word embeddings of a given word in an input sentence are concatenated using the concatenate layer of Keras. The dimension of the concatenated vector is set to 325 which is the sum of the dimensions of the word-based word embedding (5), character-based word embedding (256) and sentence-based word embedding (64).

D. SETTING PARAMETERS OF THE CONTEXTUAL EMBEDDING LAYER

The context-sensitive representation of each word in an input sentence is trained by passing the concatenated word

embedding into BiLSTM layer of Keras. The number of hidden units of this layer is set to 32 to capture bidirectional and long-range context of a target word. The number of hidden units of this layer is small because of three reasons. First, the morphological information of a word has been already captured by character-based word embedding and subword-based word embedding. Second, the semantic similarity among words has been already considered by word-based word embedding. Third, the morphological and syntactic interaction of a word has been already counted by sentence-based embedding. As a result of this, it is found that increasing the number of hidden units above 32 would result in model overfitting. Thus, for the forward context LSTM layer, the trainable hidden-to-hidden matrix is set to $M_{12}^f \in R^{32*32}$ and the trainable input-to-hidden matrix is set to $M_{13}^f \in R^{32*325}$. Similarly, for the backward context LSTM layer, the trainable hidden-to-hidden matrix is set to $M_{14}^b \in R^{32*32}$ and the trainable input-to-hidden matrix is set to $M_{15}^b \in R^{32*325}$. Hence, the dimension of the contextual word embedding vector (output vector) is 64 (32+32).

E. SETTING PARAMETERS OF FULLY CONNECTED AND SEQUENCE DECODING LAYERS

In order to prevent the overfitting problem of the model, a variant of dropout layer is added on top of concatenated word embedding layer before passing this embedding into the BiLSTM layer. This is accomplished by implementing the dropout layer and setting the keep probability to 0.8. A regular dropout layer is also used on top of context-sensitive representation layer before passing this representation into the fully connected layer. This dropout layer is implemented using a Dropout layer of Keras by setting the keep probability to 0.8. Note that the two dropout layers have no trainable parameter. To capture any dependency present among input sentences in the dataset, a fully connected layer is added on the top of context-sensitive layer. This layer transforms the context-sensitive representation of words in an input sentence into their higher-level representations. The number of hidden units of this layer is set to 64 by maintaining the dimension of its context-sensitive input vector. This layer also maintains the linearity of the input vector by using linear activation function. For this layer, the 16th trainable weight matrix $M_{16}^f \in R^{64*64}$ of the proposed model is created and initialized using the Dense Layer of the Keras API. To incorporate the dependency among semantic role levels, sequence decoding layer is added on top of fully connected layer as a sequence decoder layer. The number of units in this layer is 23 (the number of semantic roles in in IOB schema). At this layer, the state transition trainable matrix is set to $M_{17}^f \in R^{23*64}$ and the state emission trainable matrix is set to $M_{18}^f \in R^{23*23}$. Both matrices are initialized randomly. This layer is implemented using the liner-chain CRF from Keras-contrib which is a repository for Keras contributed modules with additional layers, activations, loss functions and optimizers.

F. RESULTS AND DISCUSSION

In the standard evaluation criteria of a semantic role labeling system, the label of each phrasal argument of a verb (predicate) must be assigned to each word in an input sentence using IOB sequence labeling format. Based on this evaluation criteria, accuracy, precision, recall, and F-score are the major metrics that can be used to evaluate SRL systems. We developed the prototype of the proposed neural SRL system for Amharic language incrementally phase by phase by setting the parameters of the network model.

At the first phase, we assume that the character-based embedding of a word is the main word embedding type to develop the neural SRL system for Amharic language because Amharic is a morphologically rich language. To test our assumption, we compared the three types of word embedding methods, namely, character-based word embedding, subword-based word embedding and word-based word embedding. As shown in Table 7, character-based word embedding ranks first, subword-based word embedding ranks second and word-based word embedding ranks third. This result demonstrates that for morphologically rich languages like Amharic language, character-based embedding is appropriate to capture the morphological information of each word in a given sentence.

At the second phase, we assumed that the morpho-syntactic information of a word in an input sentence can be used as a supplementary information for character-based embedding of a word which captures only the morphological information of a word. In other words, the local information of the word which is captured by the character-based word embedding is complemented by the global information of the word which is captured by sentence-based word embedding. Our main motivation to augment character-based embedding with morpho-syntactic sentence-based embedding is that morpho-syntactic information is a necessary input for well know feature-based and hybrid SRL systems for other languages. In hybrid systems, the morpho-syntactic information augments the automatically extracted features. Basically, there are three options to generate the sentence-based embedding of a word. The first option is using character-based word embedding as an input for BiLSTM recurrent network as a sequence encoder. The second option is using the subword-based word embedding as input for BiLSTM. The third option is using word-based word embedding as input for BiLSTM. We assume that the subword-based word embedding is appropriate to generate a sentence-level word embedding instead of the character-based word embedding because the performance of character-based word embedding and subword-based word embedding is almost similar as indicated in the first phase of the experiment. Thus, if the character-based word embedding is used to generate sentence-based word embedding, the concatenation character-based word embedding and the sentence-based word embedding derived from character-based word embedding is redundant. Moreover, we assume that the subword-based word embedding is appropriate to generate a sentence-level word embedding in

comparison to the word-based word embedding because subword-based word embedding is appropriate for morphologically rich languages like Amharic as we proved in the first phase of the experiment. To test our assumption experimentally, three types of word embedding types are created by the concatenation of two types of word embedding types. The experimental result shows that the concatenation of character-level word embedding and sentence level word embedding (derived from subword-based word embedding) ranks first as shown in Table 7. The experimental result confirms our assumption. Thus, the sentence-based word embedding can capture the morpho-syntactic information of a word from a sequence of its subword-based word embeddings using BiLSTM recurrent network.

Finally, at the third phase, we assume that the semantic similarity among words in a given lexicon can be also used to complement the concatenation of character-level and sentence-level word embedding types because different words with identical meanings have similar word representations. As a result of this, word-based embedding is added to complement both the concatenation of character-based and sentence-based word embedding types. Thus, when the model is trained and tested using the concatenation of the three types of word embedding types, the performance of the model improved as shown in Table 7. This result shows that by capturing the semantic similarity among words in a given lexicon, the word-based embedding also contributes for the performance of the system jointly with character-based and sentence-based embedding types. Generally, experimental results show that the concatenation of character-based and word-based and sentence-based word embedding types give better performance by capturing the morphological, semantic and morpho-syntactic information about each word in given Amharic sentence, respectively. Test results also show that experiments with the relatively balanced dataset provides slightly better performance than the use of the entire dataset. This is attributed to lack of trainability of the system with ARG3 and ARG4 labels which have very few samples for training.

The precision, recall and F-score of the semantic role labels of all phrasal arguments of predicates (verbs) tested with in the test datasets of the entire dataset and the relatively balanced dataset without IOB labeling formats are presented in Tables 8 and 9, respectively. In Table 8, the performance of semantic role labels (classes) with low number of samples (ARG3 and ARG4) is low, with precision, recall and F-score results being 0. This is because the the number of samples for these labels in the training data, as presented in Table 2, is very low (5 and 9), which makes it difficult for the deep learning system to learn from examples. Since the number of these labels are still low in the test data (1 and 3), they have little significance on the overall system performance. On the other hand, the PREDICATE label has a very high rate of accuracy as it has a very high number of examples required for training deep learning systems. Moreover, the huge performance is probably due to the position of the

Table 7. Overall performance of the proposed model.

N ^o	Embedding Type	Accuracy	Precision	Recall	F-Score	
	Character-based Embedding	93.95	78.6	75.3	76.9	
1	Sub-word-based Embedding	92.44	74.6	66.3	70.2	
	Word-based Embedding	91.73	72.2	62.3	66.9	
	Character-based + Sentence-based (Derived from sub-word-based)	94.78	82.0	78.8	80.4	
2	Character-based + Word-based	94.67	81.6	78.7	80.1	
	Word-based + Sentence-based (Derived from sub-word-based)	92.79	76.4	69.2	72.6	
3	Character + Word + Sentence	With the entire dataset	94.96	82.8	79.7	81.2
		With the relatively balanced dataset	95.01	82.8	80.4	81.6

Table 8. Performance of the system for each label using the entire dataset without IOB format.

N ^o	Label	Precision	Recall	F-Score	Frequency in test data
1	ARG0	0.781	0.824	0.802	1366
2	ARG1	0.639	0.645	0.642	1986
3	ARG2	0.595	0.372	0.458	328
4	ARG3	0.000	0.000	0.000	1
5	ARG4	0.000	0.000	0.000	3
6	ARGM-DIR	0.760	0.447	0.563	170
7	ARGM-LOC	0.444	0.371	0.405	140
8	ARGM-MNR	0.554	0.435	0.487	416
9	ARGM-PRP	0.544	0.339	0.417	127
10	ARGM-TMP	0.414	0.215	0.283	135
11	PREDICATE	1.000	1.000	1.000	2000
12	<PAD>	1.000	1.000	1.000	2000
	Micro avg	0.828	0.797	0.812	8671
	Macro avg	0.561	0.471	0.505	8671
	Weighted avg	0.816	0.797	0.803	8671

Table 9. Performance of the system for each label using the relatively balanced dataset without IOB format.

N ^o	Label	Precision	Recall	F-Score	Frequency in test data
1	ARG0	0.799	0.807	0.803	1369
2	ARG1	0.650	0.654	0.652	1988
3	ARG2	0.605	0.396	0.478	336
4	ARGM-DIR	0.780	0.554	0.648	166
5	ARGM-LOC	0.478	0.418	0.446	158
6	ARGM-MNR	0.527	0.446	0.483	410
7	ARGM-PRP	0.445	0.517	0.478	118
8	ARGM-TMP	0.446	0.276	0.341	134
9	PREDICATE	0.999	0.999	0.999	1998
10	<PAD>	1.000	1.000	1.000	1998
	Micro avg	0.828	0.804	0.816	8675
	Macro avg	0.673	0.607	0.633	8675
	Weighted avg	0.820	0.804	0.810	8675

predicate in the subject-object-verb (SOV) word-order followed by Amharic sentences. Most of the time, the predicate (verb) in an input sentence is located at the last position of the sentence. As a result, identifying the location of a predicate in an input sentence is relatively easy for the proposed SRL model.

We also carried out similar experiments with the IOB-based formats of the entire dataset and the relatively balanced

dataset. The precision, recall and F-score of the semantic role labels of all phrasal arguments of predicates (verbs) in the test data of the entire dataset and the relatively balanced dataset with IOB labeling formats are presented in Tables 10 and 11, respectively. In Table 10, the performance of the system for semantic role labels with low number of samples (B-ARG3, B-Arg4, I-ARG3 and I-ARG4) is low whereas the performance for labels with high number of samples (e.g. PREDICATE) is high, similar to the case of non-IOB labeling formats. The system slightly improves the performance for each label when the labels with low samples are excluded from the dataset, as presented in Table 11. Generally, for both IOB and non-IOB labeling formats, test results show that the system performs well for labels having sufficient number of samples in the training data whereas it performs badly for labels with few samples. This is an expected result considering the amount of data required for deep learning systems.

The confusion matrix of the model with the concatenation of the three-word embedding methods is also presented in Figure 9. The diagonal and off-diagonal values represent the percentage of truly classified and misclassified instances of a given true semantic role label, respectively. Generally, there is a tendency that the system yields better accuracy for labels that have higher number of samples in the training data whereas misclassifications (confusions) are higher for labels whose number of samples are low. However, the linguistic characteristics associated with the labels has also impact on the overall performance. As observed in the diagonal values, an improvement in accuracy is obtained for most labels when experiments are carried out with relatively balanced dataset for both IOB and non-IOB formats. This is probably due to the fact that labels with insufficient samples are excluded in the dataset, leading the model to generalize better.

The learning curves of the model with three concatenated word embedding types are presented in Figure 10. The error value (loss) of the model is sufficiently decreasing on the training data to a point of stability where the gap between the final training and validation losses is minimal. The validation loss shows steady movements around the training loss with minimal gap. This implies that both the training and validation data are representative to each other. In other

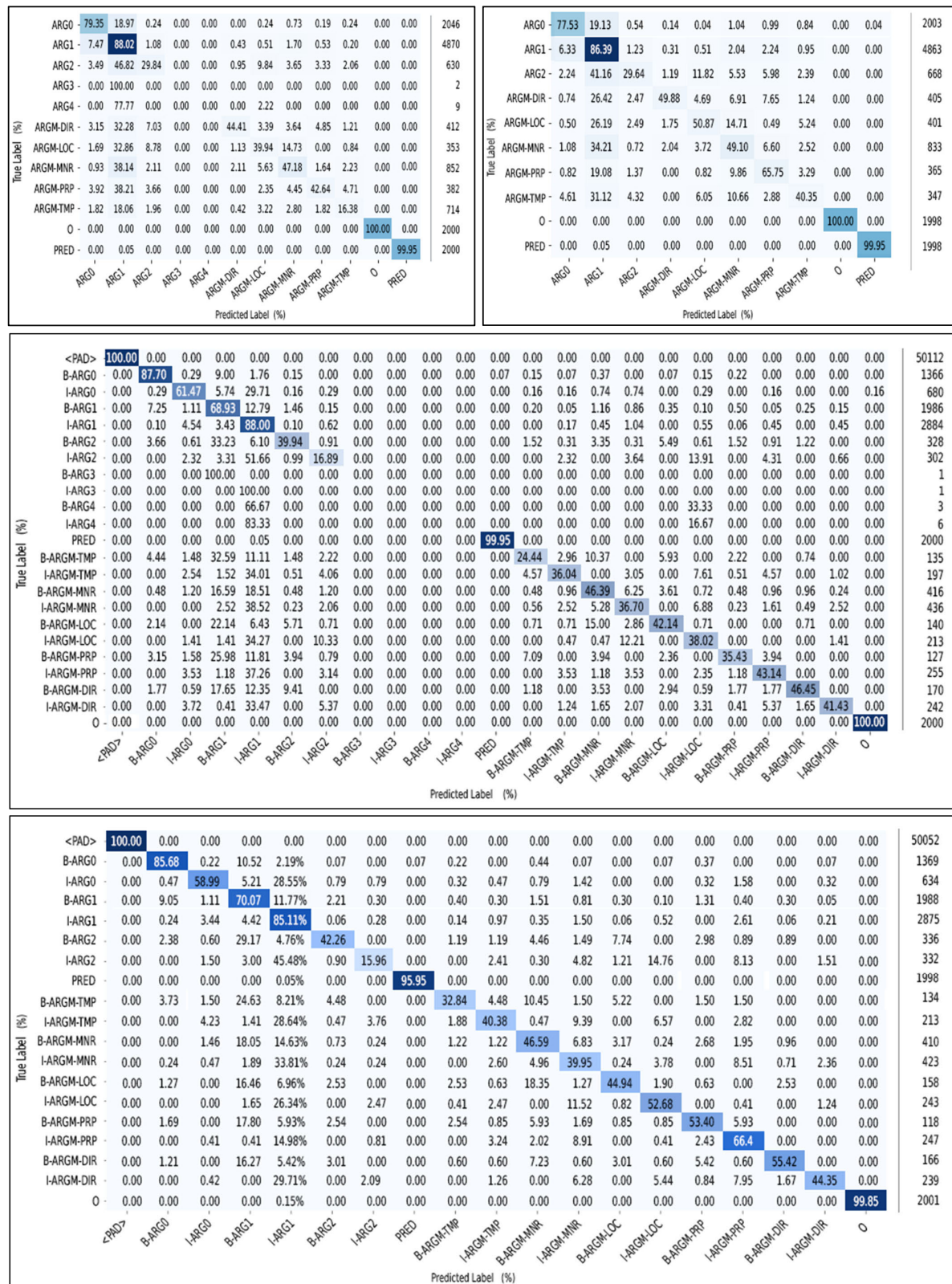


Figure 9. Confusion matrix for various tests; **Top left:** the entire dataset without IOB format, **Top right:** the relatively balanced dataset without IOB format, **Middle:** the entire dataset with IOB format, **Bottom:** the relatively balanced dataset with IOB format. The last column in each of the confusion matrices indicates the frequency of labels in test data.

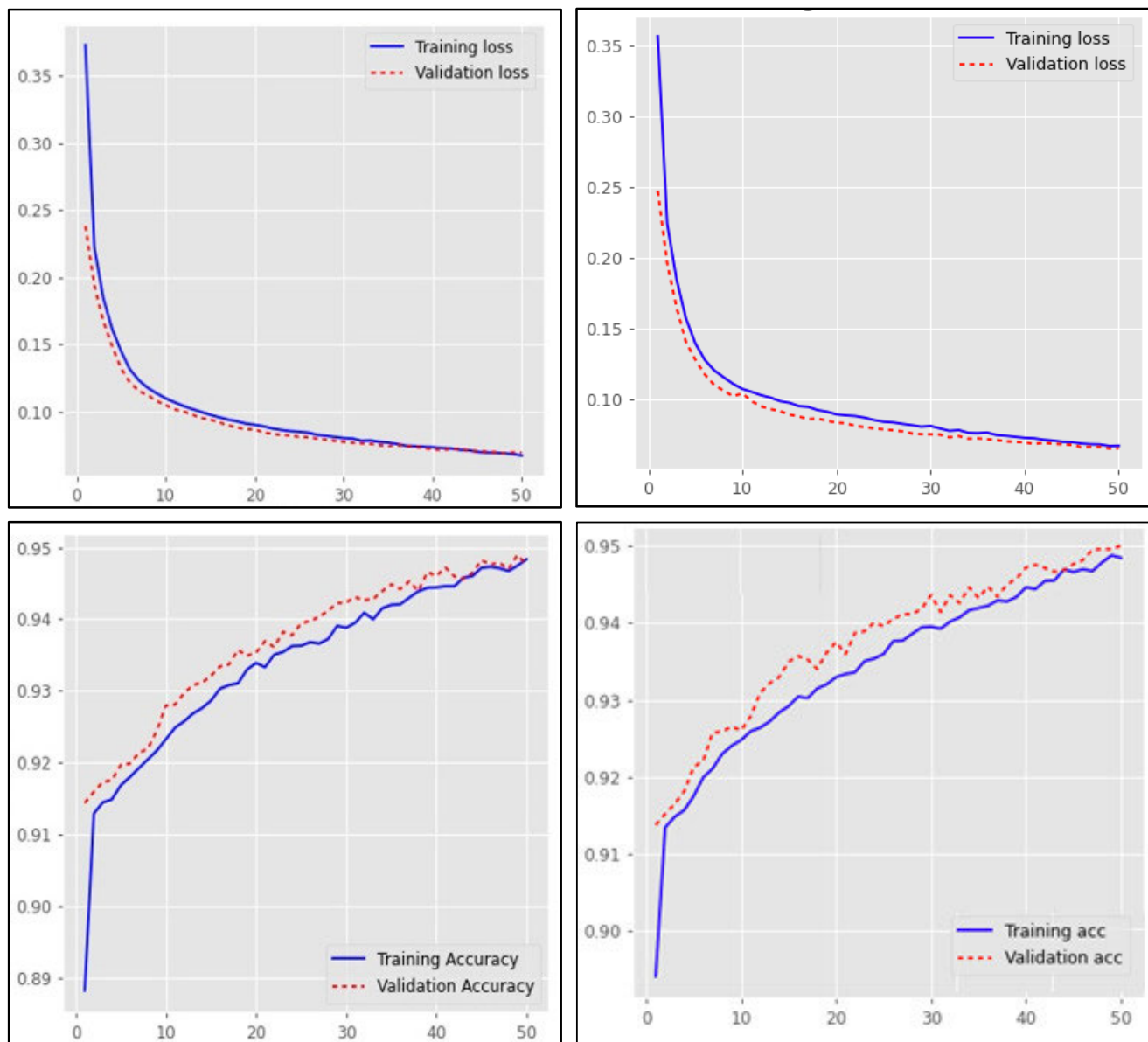


Figure 10. The learning curve of the model for training (blue color) and validation (red color) with the non-loB dataset; **Top left:** Training and validation loss for the entire dataset, **Top right:** Training and validation loss for the relatively balanced dataset, **Bottom left:** Training and validation accuracy for the entire dataset, **Bottom right:** Training and validation accuracy for the relatively balanced dataset.

words, the training data provides sufficient information to learn the problem and the validation data also provides sufficient information to evaluate the generalizing ability of the model. It is also observed that the training accuracy of the model is growing over time. This implies that the model is learning as it gets more experience. The validation accuracy of the model does not dip below the training accuracy. This implies that the model is a good-fit. Moreover, the training accuracy of the model grows rapidly at the beginning, and gradually grows over time before it reaches a plateau, which indicates that the model still requires more data to learn. When the outlier labels (ARG3 and ARG4) are removed,

the gap between training and validation loss curves increases by marginal value. Similarly, the gap between training and validation accuracy curves increases by marginal value.

To our best knowledge, there is only one previous work [29] on Amharic SRL developed using memory-based learning. The system was tested with 240 manually annotated simple sentences, and the result showed an accuracy of 89.29% and F-Score of 79.77%. Although results could not be directly compared due to variations in the size and type of test data, our system performed better with an accuracy of 94.96% and F-Score of 81.20% tested with the entire dataset.

Table 10. Performance of the system for each label using the entire dataset without IOB format.

Nº	Label	Precision	Recall	F-Score	Frequency in test data
1	B-ARG0	0.831	0.877	0.853	1366
2	B-ARG1	0.696	0.703	0.700	1986
3	B-ARG2	0.639	0.399	0.492	328
4	B-ARG3	0.000	0.000	0.000	1
5	B-ARG4	0.000	0.000	0.000	3
6	I-ARG0	0.672	0.615	0.642	680
7	I-ARG1	0.664	0.880	0.757	2884
8	I-ARG2	0.347	0.169	0.227	302
9	I-ARG3	0.000	0.000	0.000	1
10	I-ARG4	0.000	0.000	0.000	6
11	B-ARGM-DIR	0.790	0.465	0.585	170
12	B-ARGM-LOC	0.504	0.421	0.459	140
13	B-ARGM-MNR	0.590	0.464	0.520	416
14	B-ARGM-PRP	0.570	0.354	0.437	127
15	B-ARGM-TMP	0.471	0.244	0.322	135
16	I-ARGM-DIR	0.741	0.413	0.531	242
17	I-ARGM-LOC	0.382	0.380	0.381	213
18	I-ARGM-MNR	0.533	0.367	0.435	436
19	I-ARGM-PRP	0.604	0.431	0.503	255
20	I-ARGM-TMP	0.592	0.360	0.448	197
21	O	1.000	1.000	1.000	2000
22	PREDICATE	1.000	1.000	1.000	2000
23	PAD	1.000	1.000	1.000	50112
Accuracy		---	---	0.950	64000
Macro avg		0.549	0.458	0.491	64000
Weighted avg		0.948	0.959	0.947	64000

Table 11. Performance of the system for each label using the relatively balanced dataset without IOB format.

Nº	Label	Precision	Recall	F-Score	Frequency in test data
1	B-ARG0	0.847	0.852	0.849	1369
2	B-ARG1	0.696	0.699	0.697	1988
3	B-ARG2	0.672	0.396	0.498	336
4	I-ARG0	0.683	0.625	0.652	634
5	I-ARG1	0.702	0.832	0.762	2875
6	I-ARG2	0.588	0.120	0.200	332
7	B-ARGM-DIR	0.764	0.584	0.662	166
8	B-ARGM-LOC	0.486	0.449	0.467	158
9	B-ARGM-MNR	0.542	0.473	0.505	410
10	B-ARGM-PRP	0.465	0.559	0.508	118
11	B-ARGM-TMP	0.522	0.351	0.420	134
12	I-ARGM-DIR	0.801	0.473	0.595	239
13	I-ARGM-LOC	0.495	0.556	0.523	243
14	I-ARGM-MNR	0.430	0.404	0.417	423
15	I-ARGM-PRP	0.429	0.672	0.524	247
16	I-ARGM-TMP	0.465	0.432	0.448	213
17	O	1.000	0.999	0.999	2001
18	PREDICATE	0.999	0.999	0.999	1998
19	PAD	1.000	1.000	1.000	50052
Accuracy		---	---	0.950	63936
Macro avg		0.662	0.604	0.617	63936
Weighted avg		0.950	0.950	0.948	63936

VI. CONCLUSION AND FUTURE WORK

A neural-based semantic role labeling system is developed taking into account the morphological complexity of Amharic and unavailability of lexical resources for the language. The model uses the concatenation of character-based, sentence-based and word-based embedding types to capture the morphological, morpho-syntactic and semantic features of a given word in an input sentence, respectively. The

word embedding layer, which is the concatenation of the three-word embedding types, is a context-insensitive representation. Since a context-sensitive representation of a given word in an input sentence is necessary for the sequential semantic role labeling task, the context-insensitive concatenated word embedding is transformed into context-sensitive word representation. This is accomplished by capturing the bi-directional and long-range dependencies of each word in a sentence using BiLSTM recurrent network. Finally, a conditional random field with viterbi sequence decoding algorithm generates a sequence of semantic role labels for a given sequence of words in an input sentence. The system has achieved an accuracy of 94.96% and F-score of 81.2% on test data. The experimental result shows that without using manually designed features of words in an input sentence, it is possible to develop SRL system for Amharic language by extracting morphological, syntactic and semantic features of words automatically. Character embeddings are the most important word embedding type for Amharic language, and the sentence-based embedding, which is achieved by passing a sequence of subword-based embeddings in an input sentence into a BiLSTM recurrent network, is the second most important word embedding type for the language.

This work gives the first insight for the possibility of developing semantic role labeling system for Amharic text using deep learning algorithm. However, it has targeted simple Amharic sentences. In addition, the choice of the word embeddings used is constrained to the limited computational resources that were available to us. The size of the corpus is also small in comparison to high-resource languages like English and Chinese, and lacks balanced class distribution. Thus, future work is directed at improving the quality of the corpus (by increasing the corpus size and balancing the class distribution), and experimenting with other contextualized neural word embedding methods such as ELMo and BERT in a higher capacity computational environment.

References

- [1] L. Màrquez, X. R. Carreras, K. C. Litkowski, and S. Stevenson, "Semantic role labeling: An introduction to the special issue," *Comput. Linguistics*, vol. 34, no. 2, pp. 145–159, 2008.
- [2] D. Gildea and D. Jurafsky, "Automatic labeling of semantic roles," *Comput. Linguistics*, vol. 28, no. 3, pp. 245–288, 2002.
- [3] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2009.
- [4] S. Narayanan and S. Harabagiu, "Question answering based on semantic structures," in *Proc. 20th Int. Conf. Comput. Linguistics*, Geneva, Switzerland, 2004, pp. 693–701.
- [5] M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth, "Using predicate-argument structures for information extraction," in *Proc. 41st Annu. Meeting Assoc. Comput. Linguistics*, Sapporo, Japan, 2003, pp. 8–15.
- [6] J. Christensen, Mausam, S. Soderland, and O. Etzioni, "An analysis of open information extraction based on semantic role labeling," in *Proc. 6th Int. Conf. Knowl. Capture*, Banff, AB, Canada, 2011, pp. 113–120.
- [7] A. Khan, N. Salim, and Y. J. Kumar, "A framework for multi-document abstractive summarization based on semantic role labelling," *Appl. Soft Comput.*, vol. 30, pp. 737–747, 2015.
- [8] D. Liu and D. Gildea, "Semantic role features for machine translation," in *Proc. 23rd Int. Conf. Comput. Linguistics*, Beijing, China, 2010, pp. 716–724.

- [9] W. Aziz, M. Rios, and L. Specia, "Shallow semantic trees for SMT," in *Proc. 6th Workshop Stat. Mach. Transl.*, Edinburgh, Scotland, 2011, pp. 316–322.
- [10] Z. Zhang, Y. Wu, H. Zhao, Z. Li, S. Zhang, X. Zhou, and X. Zhou, "Semantics-aware BERT for language understanding," in *Proc. 34th AAAI Conf. Artif. Intell.*, New York, NY, USA, 2020, pp. 9628–9635.
- [11] Q. Zhou, Z. Jiang, and F. Yang, "Sentences similarity based on deep structured semantic model and semantic role labeling," in *Proc. Int. Conf. Asian Lang. Process.*, Kuala Lumpur, Malaysia, 2020, pp. 40–44.
- [12] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn.*, Helsinki, Finland, 2008, pp. 160–167.
- [13] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.
- [14] Z. Li, H. Zhao, S. He, and J. Cai, "Syntax role for neural semantic role labeling," *Comput. Linguistics*, vol. 47, no. 3, pp. 529–574, 2021.
- [15] F. Qian, L. Sha, B. Chang, L. Liu, and M. Zhang, "Syntax aware LSTM model for semantic role labeling," in *Proc. 2nd Workshop Structured Predict. Natural Lang. Process.*, Copenhagen, Denmark, 2017, pp. 27–32.
- [16] J. Zhou and W. Xu, "End-to-end learning of semantic role labeling using recurrent neural networks," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, Beijing, China, 2015, pp. 1127–1137.
- [17] Z. Wang, T. Jiang, B. Chang, and Z. Sui, "Chinese semantic role labeling with bidirectional recurrent neural networks," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Lisbon, Portugal, 2015, pp. 1626–1631.
- [18] Z. Li, H. Zhao, J. Zhou, K. Parnow, and S. He, "Dependency and span, cross-style semantic role labeling on PropBank and NomBank," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 21, no. 6, pp. 1–16, 2022.
- [19] H. Fei, S. Wu, Y. Ren, and D. Ji, "Second-order semantic role labeling with global structural refinement," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 29, pp. 1966–1976, 2021.
- [20] C. J. Fillmore, C. Wooters, and C. F. Baker, "Building a large lexical databank which provides deep semantics," in *Proc. 15th Pacific Asia Conf. Lang., Inf. Comput.*, Hong Kong, Feb. 2001, pp. 3–26.
- [21] M. Palmer, D. Gildea, and P. Kingsbury, "The proposition bank: An annotated corpus of semantic roles," *Comput. Linguistics*, vol. 31, no. 1, pp. 71–106, Mar. 2005.
- [22] A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman, "The NomBank project: An interim report," in *Proc. Workshop Frontiers Corpus Annotation HLT-NAACL*, Boston, MA, USA, 2004, pp. 24–31.
- [23] M. P. Marcus and M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of English: The Penn Treebank," *Comput. Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.
- [24] I. Zitouni, *Natural Language Processing of Semitic Languages*. Berlin, Germany: Springer, 2014.
- [25] R. Hetzron, *The Semitic Languages*. London, U.K.: Routledge, 1997.
- [26] T. Yeshambel, J. Mothe, and Y. Assabie, "Amharic *ad hoc* information retrieval system based on morphological features," *Appl. Sci.*, vol. 12, no. 3, p. 1294, 2022.
- [27] A. Akbik and Y. Li, "K-SRL: Instance-based learning for semantic role labeling," in *Proc. 26th Int. Conf. Comput. Linguistics*, Osaka, Japan, 2016, pp. 599–608.
- [28] M. Diab, A. Moschitti, and D. Pighin, "Semantic role labeling systems for Arabic using kernel methods," in *Proc. ACL HLT*, Columbus, Ohio, 2008, pp. 798–806.
- [29] E. Yirga, "Semantic role labeler for amharic text using memory-based learning," M.S. thesis, Dept. Comput. Sci., Addis Ababa Univ., Addis Ababa, Ethiopia, 2017.
- [30] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
- [31] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, Dec. 2017.
- [32] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.
- [33] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.
- [34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [35] C. Xia, C. Zhang, T. Yang, Y. Li, N. Du, X. Wu, W. Fan, F. Ma, and P. Yu, "Multi-grained named entity recognition," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, Florence, Italy, 2019, pp. 1430–1440.
- [36] Y. Zhang, Q. Liu, and L. Song, "Sentence-state LSTM for text representation," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, Melbourne, VIC, Australia, 2018, pp. 317–327.
- [37] Y. Liu, F. Meng, J. Zhang, J. Xu, Y. Chen, and J. Zhou, "GCDDT: A global context enhanced deep transition architecture for sequence labeling," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, Florence, Italy, 2019, pp. 2431–2441.
- [38] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 24, no. 4, pp. 694–707, Apr. 2016.
- [39] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, Berlin, Germany, 2016, pp. 1064–1074.
- [40] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," 2015, *arXiv:1508.01991*.
- [41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–10.



BEMNET MERESA HAILU received the B.Sc. degree in software engineering from Unity University, Addis Ababa, Ethiopia, in 2006, and the M.Sc. degree in computer science from Addis Ababa University, Ethiopia, in 2021. He is currently a Technical Assistant with the Addis Ababa Institute of Technology, Addis Ababa University. His research interests include language technology, machine learning, artificial intelligence, and data science.



YAREGAL ASSABIE received the B.Sc. degree in computer science and the M.Sc. degree in information science from Addis Ababa University, Ethiopia, in 1998 and 2002, respectively, and the Ph.D. degree in electrical engineering from the Chalmers University of Technology, Gothenburg, Sweden, in 2009. He is currently an Associate Professor with the Department of Computer Science, Addis Ababa University. He has authored more than 45 papers in reputable journals and conference proceedings. His research interests include computer vision, natural language processing, pattern recognition, and artificial intelligence. He has received best paper awards for three of his papers presented at international conferences.



YENEWONDIM BIADGIE SINSHAW received the B.Sc. degree in mathematics from Bahir Dar University, Ethiopia, in 2000, the M.Sc. degree in information science from Addis Ababa University, Ethiopia, in 2006, and the Ph.D. degree in computer engineering from Ajou University, South Korea, in 2012. He was a Lecturer with Addis Ababa University, from 2007 to 2008. He was also a Postdoctoral Research Fellow with the Visual Computing Research Laboratory, Ajou University, from March 2012 to August 2015, where he has been an Assistant Professor with the Department of Software and Computer Engineering, since September 2015. His research interests include pattern recognition, computer vision, natural language processing, machine learning, and deep learning.

...