

ITC 4/49 Information Technology and Control Vol. 49 / No. 4 / 2020 pp. 482-494 DOI 10.5755/j01.itc.49.4.26808	Part-of-Speech Tagging via Deep Neural Networks for Northern-Ethiopic Languages	
	Received 2020/06/18	Accepted after revision 2020/06/22
	 http://dx.doi.org/10.5755/j01.itc.49.4.26808	

HOW TO CITE: Tesfagergish, S. G., Kapočiūtė-Dzikienė, J. (2020). Part-of-Speech Tagging via Deep Neural Networks for Northern-Ethiopic Languages. *Information Technology and Control*, 49(4), 482-494. <https://doi.org/10.5755/j01.itc.49.4.26808>

Part-of-Speech Tagging via Deep Neural Networks for Northern-Ethiopic Languages

Senait Gebremichael Tesfagergish

Department of Software Engineering, Kaunas University of Technology; Studentų str. 50, 51368 Kaunas, Lithuania;
e-mail: senait.gbremichael@ktu.edu

Jurgita Kapočiūtė-Dzikienė

Faculty of Informatics, Vytautas Magnus University; Vileikos str. 8, 44404 Kaunas, Lithuania;
e-mail: jurgita.kapociute-dzikiene@vdu.lt

Corresponding author: senait.gbremichael@ktu.edu

Deep Neural Networks (DNNs) have proven to be especially successful in the area of Natural Language Processing (NLP) and Part-Of-Speech (POS) tagging—which is the process of mapping words to their corresponding POS labels depending on the context. Despite recent development of language technologies, low-resourced languages (such as an East African Tigrinya language), have received too little attention. We investigate the effectiveness of Deep Learning (DL) solutions for the low-resourced Tigrinya language of the Northern-Ethiopic branch. We have selected Tigrinya as the testbed example and have tested state-of-the-art DL approaches seeking to build the most accurate POS tagger. We have evaluated DNN classifiers (Feed Forward Neural Network – FFNN, Long Short-Term Memory method – LSTM, Bidirectional LSTM, and Convolutional Neural Network – CNN) on a top of neural word2vec word embeddings with a small training corpus known as Nagaoka Tigrinya Corpus [19]. To determine the best DNN classifier type, its architecture and hyper-parameter set both manual and automatic hyper-parameter tuning has been performed. BiLSTM method was proved to be the most suitable for our solving task: it achieved the highest accuracy equal to ~92% that is ~65% above the random baseline.

KEYWORDS: Deep Learning; word2vec embeddings; part-of-speech tagging; natural language processing; computational linguistics; Tigrinya language.

1. Introduction

POS tagging (also called grammatical tagging) is an important application of NLP and a core concept that many higher-level language technologies depend on. NLP applications as machine translation, speech recognition, dependency parsing and many more depend on POS tagging to be more accurate.

POS tagging is a mapping process of words from a sentence to their corresponding parts-of-speech, based on their context and the meaning. Despite from a human point-of-view the manual POS tagging looks a rather easy task, it is a challenging AI problem to solve, mainly due to words disambiguation. Languages are different by their nature and morphological complexity, therefore there is no single smart solution that could solve all POS tagging problems for all languages of the world. Fortunately, there are solutions that work for groups of languages (e.g., morphologically complex languages, agglutinative languages, etc.). The major POS tagging problems usually arises due to the ambiguities in each language and different POS tagging annotation schemes, when even trained human annotators sometimes cannot agree on the words' POS label [24]. The different annotation schema issue is not easily tackled, whereas disambiguation issues can be resolved by training Machine Learning (ML) methods with the enough manually POS tagged corpora (so-called gold-standard corpora).

In the recent years many traditional ML approaches are replaced with the DL and it is cross-cutting phenomenon. DL is used for various tasks as parameter prediction in the Internet-of-Things devices [30]; for skin marks analysis to detect melanoma [31]; for robot vision [6] and many more. The DL also has covered all areas of NLP, including POS tagging. Starting from 1994 [33] (with the seed work on the POS tagging with the neural networks), this problem is still relevant today [8] and can be solved with the high-accuracy DNN approaches. However, the higher the accuracy is expected, the more resources are required.

Unlike the rich-resource languages like English, most of the low-resourced Ethiopian languages do not have enough corpora that are needed for this task. Open resource corpus (such as a Crubadan Corpus Building for

Minority Languages¹) doesn't contain the morphological labels crucial for the POS tagging task. The only publicly available corpus for Tigrinya POS tagging is the Nagaoka corpus [19], which is a rather small resource (especially compared to what is available for English or Chinese languages). Besides, the Northern-Ethiopic languages have more complex morphology, therefore require more resources compared to English or Chinese to cover different inflection forms.

The previous research using the Nagaoka corpus is performed with the traditional supervised ML approaches (i.e., traditional feature types with traditional classifiers as Conditional Random Fields – CRFs and Support Vector Machines – SVMs. Although authors managed to get a sufficient accuracy with the traditional approaches, they have not explored state-of-the-art deep learning techniques, which are likely to improve the accuracy even more. Hence, in this research we focus on the DNN types, that could be the most promising for our solving task: FFNN, LSTM, BiLSTM and CNN. The highest possible POS tagging accuracy is our main goal and to achieve this we need the whole package (the correct DNN type + it's architecture + hyper-parameters) to perform in the best possible way. Due to it, we have tested different architectures and hyper-parameter values by tuning them manually and automatically.

DNN methods have to be applied on the vectorized text. Since Tigrinya language do not have pre-trained word embeddings that could be publicly available, we have trained them and describe this process in Section 4. Section 5 describes the DNN classifiers; Section 6 presents the technicality of the experiments and their results. Finally, we conclude with nominating the best approach for the Tigrinya POS tagging.

The main novelty and contribution of this work covers the comparative state-of-the-art DNN research (including architecture and hyper-parameter tuning) on the POS tagging task for the Tigrinya language sharing similar characteristics to other Northern-Ethiopic languages that could strongly benefit from this research.

1 Available at <http://crubadan.org/languages/ti> and word list compiled by Biniam Gebremichael's web crawler, available at <http://www.cs.ru.nl/~biniam/geez/crawl.php>

2. Related Works

The Ethiopian languages (Tigre, Ge'ez, Amharic, Tigrinya) are spoken by around 30 million people. Some Ethiopian languages as opposite to other Semitic languages (such as Arabic, Hebrew and Amharic) get rather little attention in the area of NLP research. This is mainly due: 1) to the absence of linguistic resources; 2) proprietary resources; 3) resources not in the electronic format. The most researched Ethiopian language is Amharic [1,2,9,11,13,23], which is also supported by Google. For the other languages as, e.g., Tigrinya, the research on different NLP topics are in their early stages (e.g., statistical machine translation from English to Tigrinya [3]).

If focusing on the POS tagging task, the pioneering POS tagging research on the Amharic language was done in 2001 [23]: the author applied stochastic Hidden Markov approach. In [13] authors use the Brill, and TnT of python Implementation in NLTK. Their most accurate POS tagger was based on CRF and achieved 90.95% of the accuracy. The paper describes how the usage of partially cleaned corpus, selection of most informative features (based on the linguistic information as vowel pattern, radicals, punctuation, alphanumeric and suffixes), and applications of parameter tuning and tagging algorithms help to boost the accuracy. Their experiments were carried on the corpus of 210,000 tokens with 31 tag labels (11 basic).

The most innovative work for Amharic POS tagging is presented in [2]. The author trained the POS tagger with neural word embeddings as the feature type and DNN methods as classifiers. The tested LSTM, FFNN and BiLSTM approaches achieved the accuracy equal to 92.8%, 88.88% and 93.7%, respectively.

Another POS tagging work for Amharic compared one traditional ML approach (in particular, CRF) with one neural-based approach (BiLSTM) and achieved 90% and 91%, respectively [5]. Such a high accuracy is achieved on the task which is less complicated to our solving task: the authors use 11 POS labels (whereas in our experiments we have 20).

However, both research works (i.e., [2] and [5]) provide too abstract conclusions lacking the important details about the DNN architectures and hyper-parameters. Different combinations of architectures and hyper-parameters can impact the accuracy in the broad range. We cannot repeat the previous exper-

iments, but at least we can benefit from the insights proving DNN superiority over the traditional ML approaches.

However, POS taggers for Amharic already exist and it opens the opportunity to continue the research on the other topics that cannot be initiated without POS tagging as e.g., dependency parsing [35], enhancement of MT and information retrieval tasks [10].

HornMorpho [11] is a system for morphological processing of three Ethiopian languages (i.e., Amharic, Oromo and Tigrinya). The morphological analyzer segments words into morphemes and has to assign grammatical labels to them. However, for Tigrinya it assigns only verbs. Hence, the application options of such analyzer are very limited.

POS tagging for the Tigrinya language is studied in [12]. The author used 26,000 words labelled with 36 POS labels and experimented with the probabilistic Hidden-Markov Model (HMM) method combined with the rule-based tagger (Viterbi algorithm and Brill) [12]. The HMM and rule-based approach achieved the accuracy of 89.13% and 91.8%, respectively, when applied separately. A hybrid HMM + rule-based approach boosted the accuracy to 95.88%. This research is important; despite done with the traditional ML method combined with the inflexible rule-based approach making it hardly transferable to the new domains, language styles (fiction, legal texts, etc.) and types (spoken language, non-normative, etc.). Another research for Tigrinya POS tagging was also done on the small-sized gold standard Nagaoka Tigrinya Corpus (NTC.1.0) [19], which is currently the only publicly available POS tagged corpus for the Tigrinya language. Using NTC 1.0 the authors tested traditional ML methods, i.e., CRF and SVM. The original corpus contains 76 different tag labels (some of them are weakly covered) that were reduced to 20 labels for the better distribution. For POS tagging of the target words the authors extracted contextual features covering two succeeding and two preceding words. In addition, lexical features were extracted from the focus word: affixes, comprising prefixes from one up to six characters length, consonant-vowel patterns (infixes), and suffixes from one to five characters length [20]. The best accuracy=89.92% and 90.89% was achieved with SVM and CRF, respectively. Another interesting approach on the Tigrinya POS tagging [36] use the traditional ML approaches

(presented in [20]), but explores different types of neural word embeddings to improve the POS tagging of the unrecognized words. However, this research is not focused on the improvement of the POS tagger: it presents extrinsic evaluation of the word embeddings and the POS tagging task is selected as a good example. When experimenting with the DNN classifiers, the neural word embeddings are recommended. Unfortunately, neural word embeddings (trained and presented in [36]) are not publicly available online, therefore for our POS tagging research we will need to train them ourselves.

To conclude, despite some Northern-Ethiopic languages (as Amharic) are more researched, different research works cover different: 1) languages, 2) datasets, 3) annotation schemas (resulting in various numbers of tag labels), 4) applied methods (traditional ML or DL-based). Due to different experimental conditions the results are hardly comparable and even interpretable. Moreover, the previous research works do not provide important recommendations on the different DNN classifiers, architectures, hyper-parameters that we could rely before starting our experiments. Northern-Ethiopic languages also lack of publicly available resources. All it makes our research on the Tigrinya POS tagging even more important; especially assuming that the other similar Northern-Ethiopic languages (sharing similar characteristics to Tigrinya) could benefit from it.

3. Ethiopic Languages Characteristics

Northern-Ethiopic languages are from the Afro-asiatic family and belongs to the South Semitic languages along with Amharic, Maltese, Tigre and Arabic. The Ge'ez script is adapted to write other Semitic languages. Mainly Amharic, Tigrinya, Ge'ez and Tigre languages are characterized with the rich derivational and influential morphology which results in the numerous variations of word forms. The *root-template* morphological pattern that is usually the distinguishing feature of Semitic languages is composed of trilateral roots.

For example, ሃገራት (*hagerat*) – *countries*, ተማሃራይ (*temaharay*) – *male student*, ተማሃራት (*temaharit*) – *female student* is based on the same noun, inflected for gender (examples in Tigrinya). Adjectives are inflected for gender and number: ጸሊም (*Xelim*), ጸሊምቲ

(*Xelemti*) – black (in masculine), blacks respectively [20]. Similarly, verbs (most the Ge'ez script using languages) have rich morphological structures. Different structures of the basic verb form are made in the arrangement of consonants and vowel patters, e.g., the root *sbr* – *to break* of pattern (CCC) has forms such as *sebere* (CVCVCV) in the active form, *te-sebere* (te-CVCCV) in passive form [20]. Table 1 shows the most informative patterns according to the distribution of the gerundive verb (V_GER).

As it can be seen from the previous examples, the morphology for the Northern-Ethiopic languages

Table 1

Some patterns of the gerundive verb (V_GER) [20]

V_GER pattern	V_GER %	Examples
CeCiCu	31.1	feliTu (<i>he knew</i>)
CeCeCiCu	17.8	tefeliTu (<i>it was known</i>)
CeCiCICu	15.9	tefaliTu (<i>to know each other</i>)
CeCiCICa	14.6	feliTIka (<i>you knew</i>)
aCICiCu	11.8	afliTu (<i>made something known</i>)
CeCiCoCI	11.3	feliTomI (<i>they knew</i>)
CeCiCa	10.5	feliTa (<i>she knew</i>)

is very specific and must be indirectly incorporated (it is usually done via word vectorization, described in Sub-Section 4.2) into the DNN approaches when seeking for the most accurate solutions.

4. Dataset Preparation

4.1. The Corpus

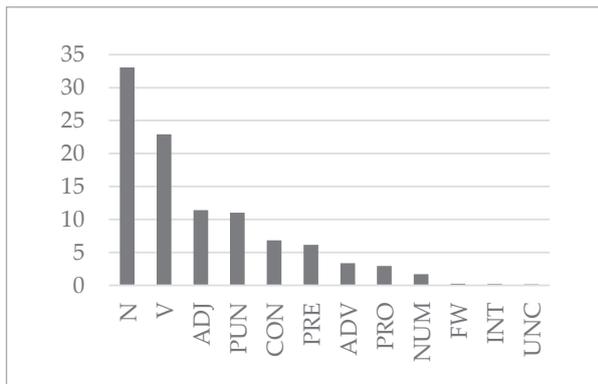
As we already mentioned before, the main problem for the Northern-Ethiopic languages are the lack of resources. However, in this research we are using the publicly available Nagaoka Tigrinya Corpus (NTC 1.0) [19], created specifically for the POS tagging task. This corpus contains gold POS labels that makes this corpus suitable for the supervised POS tagging tasks. The POS tagger can be trained in the supervised manner and afterwards be used to POS label of the new sentences. The trained model of the

POS tagger typically depends on: 1) the diversity and completeness of the annotated corpus; 2) on the used tag labels (coarse-grained as noun, verb, etc. or fine-grained containing more detailed morphological information as gender, number, tense, etc. that is much more difficult to predict).

The dataset used in our research consists of 72,080 tokens of 4,656 sentences gathered from the newspaper articles. The corpus is set up in Ge'ez and English alphabets, but for the simplify reasons only the English alphabet is used. Words in the corpus are tagged with 20 different types of POS labels (the statistics about the distribution of labels can be found in Figure 1). The solving POS tagging task is indeed the multi-class supervised classification problem [22,34] and the most promising classifiers for our task are represented in Section 5.

Figure 1

A distribution of POS labels in the Nagaoka corpus [20]



For our experiments the NTC 1.0 corpus was split into training, validation and testing sets. The distribution is presented in Table 2.

Table 2

Training, testing and validation splits of the NTC 1.0 corpus used in our experiments

Sets	Percentage	Number of tokens	Number of sentences
Training	60%	43,248	2,792
Testing	20%	14,416	931
Validation	20%	14,416	933

Before the POS tagging, the following pre-processing steps were necessary:

- 1 *Tokenization*. The tokenization task for languages like English is straightforward; however, it requires a special adjustment to some other languages, including Northern-Ethiopic. For instance, Amharic, Tigre, Tigrinya words are usually spoken to as connection of morphemes each having its own POS tag (e.g., nouns attached with a preposition has separate label, namely NP).
- 2 *Feature Extraction*. POS tags also depend on positions of words in a sentence. Due to this reason it is important to consider the following features: if a word is the first or last in a sentence; take 1-2 POS tags before and after the target word.

4.2. Vectorization

In the previous Sub-Section 4.1 we came up with the conclusion that supervised ML methods are the most suitable for our solving task. However, these methods cannot be applied directly on the textual data. The word vectorization is implemented to bridge the gap between texts and the mathematical operations of the supervised ML methods. Textual elements (usually words) are represented by vectors that are derived from textual input data and reflect contextual and linguistic properties [14].

There are two main directions to represent words (and texts) as vectors: by using discrete or distribution vectorization. The most famous example of the discrete vectorization is *one-hot* encoding. The length of the *one-hot* word vectors is equal to the size of the vocabulary, all values are zeros except for one set to 1. It is very fast and simple approach to vectorize texts and due to it *one-hot* encoding is often selected as the baseline approach. On the other hand, *one-hot* encoding does not incorporate contextual information between words. As the opposite to it, distributional word embeddings (as *word2vec*, *gloVe*, *BERT*, etc.) are trained with the DNNs and encode similarities between words. Distributional word embeddings are typically used as the input for the DNN classifiers.

In this paper, we are testing both *one-hot* encoding and *word2vec* vectorization (for more details see [26-28]) methods as representatives of discrete and distributional vectorization types, respectively.

Unfortunately, there are no pre-trained word embeddings for the Tigrinya language what would be publicly available and we could use them as the input for our neural classifiers. To overcome this problem, *word2vec* method with dimensions = 100 and window size = 3 (with other default parameter values) was used to train Tigrinya neural word embeddings. As the training corpus we have used 4,656 sentences from NTC 1.0 corpus. The training was performed with the Python programming language and the open-source library *Gensim* [32]. The pre-trained embeddings were saved and used afterwards in our experiments.

5. Deep Learning Classifiers

5.1. Feed Forward Neural Network

The Feed Forward Neural Network (FFNN) is the simplest DNN type of all existing. FFNNs have numerous applications where nonlinear mapping is done between inputs and outputs to predict future state. However, it cannot be used if the outputs dependent on the previous state of inputs. According to this definition, FFNN should not be the best option for the POS tagging task in which the order of words should play a key role.

On the other hand, the sequential information can be entered directly to the FFNN in the form of X succeeding and Y preceding words ($X=2$ and $Y=2$ in all our experiments). Although this FFNN method is selected as a baseline approach to see how far the accuracy can increase with naïve solutions, we have doubted this method could surpass other types of classifiers (e.g., LSTM or BiLSTM) adjusted to learn from the sequential data.

The results of FFNN are presented in Sub-Section 6.1.

5.2. Recurrent Neural Networks

In the Northern-Ethiopic languages the word order in a sentence is important (it can even change the meaning of a sentence), therefore cannot be disregarded. Thus, Recurrent Neural Networks (RNNs) ought to be a decent choice for our POS tagging task. These types of networks have memory cells and inputs from the previous states, therefore are adjusted to process

sequential data. RNN has two inputs at each time step, where the initial one is the real input (i.e., incoming word from the sentence) and the subsequent one is the output of the previous step. Despite of the fact that RNNs have a memory, they suffer from the vanishing gradient problem and only latest inputs are remembered and considered. In the Northern-Ethiopic languages (as well as in the Tigrinya language) this could have negative effect on learning and prediction of the POS tags that are influenced by words more distant from the target word.

To overcome the short memory problem of RNNs Long Short-Term Memory (LSTM) network [17] or Bidirectional LSTM (BiLSTM) [15] are used instead. LSTM has 3 weighted gates adjusted during training. The input, forget and output gates are used to decide what information to input, forget and output, respectively. While LSTM allows the data stream just one way (from the past to the future), BiLSTM takes both directions bearing data streams from the past to the future and from the future to the past. In the POS tagging tasks, some succeeding words can provide important information about the POS tags of the previous words. E.g., in the Tigrinya language verbs convey a lot of information about nouns and pronouns, but they mostly appear last, i.e., at the end of the sentence. Theoretically, by considering the language specifics and the nature of these RNN approaches, LSTM should be a good choice for our POS tagging task, but BiLSTM should be the best. However, to determine the best classifier type is not enough, it is necessary to choose its architecture and a right set of hyper-parameter values.

5.3. Convolutional Neural Network

The Convolutional Neural Network (CNN) [21] comprises of two sections: feature extraction (the input, convolution and activation) and classification (max pooling, fully-connected layer and the output).

The upper layers of CNN are inputs of word embedding values from the input sequence. These values are connected to the 1D convolutional layer in which the neurons of their local regions are attached to their weights (called filters or kernels). During initialization weights are randomly generated. Output of a neuron is a dot product between the filters (weights) and local region of the input. Through the activation func-

tion applied to every neuron in the network, down sampling along the width (length of the word vector) and height (length of the input text sequence) dimensions is performed in the max pooling stage that is where the classification task begins. Then, it is connected to a fully-connected layer using values from the max pooling layer (as extracted features) for this input. For adjusting the weights, the backpropagation learning algorithm is applied. Instead of working with the sequential data, the CNN is adjusted to search for the important patterns (usually n-grams/sequences of words) that impacts the target POS tag the most.

Despite our credit goes to LSTM and BiLSTM methods as the most adjusted to process sequential data; CNN sometimes outperforms those networks and it is mostly because the recurrent methods feed the whole context (also unrelated) into the network, but CNN performs the careful search over the related word n-grams instead.

CNNs could be a good option for our solving POS tagging problem, in light of the fact that there are cases in Northern-Ethiopic languages (including Tigrinya) where not all the information in the sentences are relevant, but only a small subset typically 3-grams close to the target word can impact its POS tag.

6. Experiments and Results

6.1. Evaluation Metrics

In this research we have experimented with the NTC 1.0 corpus (described in Sub-Section 4.1), using the vectorization (described in Sub-Section 4.2) and the DNN classifiers (presented in Section 5). Python programming language with the *TensorFlow* engine [37] and *Keras* library [7] was used for these methods' implementation.

In this paper we have adopted the following formulas to measure the *accuracy* (Equation (1)) and the *loss* (Equation (2)).

$$Accuracy = \frac{tp + tn}{tp + tn + fn + fp}, \quad (1)$$

where: *tp* (true positives); *fn* (false negatives); *fp* (false positives); *tn* (true negatives).

$$loss = - \sum_{c=1}^M y_{o,c} \log \log(P_{o,c}), \quad (2)$$

where: *M* is a number of classes (POS tags); *Y* is a binary indicator (0 or 1) if a class label *c* is the correct classification for observation *o*; *P* is a predicted probability of observation *o* in class *c*.

For any classification task it is necessary to have baseline values. The determined accuracy must surpass baselines for the method to be considered reasonable and suitable for the solving classification task. Only then the POS tagging result will be considered reasonable and appropriate if the calculated *accuracy* is above *random* (Equation (3)) and *majority* (Equation (4)) baselines.

$$Random\ baseline = \sum P(C_i)^2 \quad (3)$$

$$Majority\ baseline = \max P(C_i), \quad (4)$$

where: *c_i* is a probability of a class (where classes represent POS tags).

The calculated baselines are presented in Table 3. It shows that the calculated accuracy of reasonable method must be above 0.27 and, of course, our goal is going far beyond this boundary. To evaluate if differences between the accuracies are statistically significant, we have used the McNemar test with one degree of freedom and the significance level equal to 95% [25]. The differences were considered statistically significant if the calculated *p* value exceeded 0.05.

The most suitable classifier can poorly perform if it's architecture and a set of hyper-parameter values are not properly selected. It doesn't matter if those parameters are set manually or automatically, the most important is that they would be set in the way allowing to achieve the highest possible POS tagging *accuracy*.

The parameter optimization was performed on the training dataset and validated on the validation dataset. The most accurate model (achieving the highest accuracy on the validation dataset) was evaluated with the training dataset (the detailed information on the splits of training, validation and testing is presented in Table 2).

6.2. Manual Hyper-Parameter Tuning

The selection of the DNN architecture and hyper-parameter values based purely on the expert knowledge is not always the right decision. For this reason, we have investigated several DNN architectures (deeper and shallower) together with different hyper-parameter values (in particular, activation functions).

Table 3

Calculation of *random* and *majority* baselines

POS tags	Number of Instances	P(C)
V_PRF	1,437	0.020
UNC	113	0.079
V_AUX	3,409	0.047
V_IMP	316	0.004
N	19,495	0.270
PUN	7,960	0.110
V_REL	3,787	0.053
ADV	2,415	0.034
INT	145	0.002
N_V	2,104	0.292
ADJ	8,210	0.114
NUM	1,235	0.017
N_PRP	2,220	0.031
FW	176	0.002
V	357	0.005
V_GER	2,734	0.038
CON	4,933	0.068
V_IMF	4,501	0.062
PRE	4,424	0.061
PRO	2,106	0.029
Random Baseline		0.128
Majority Baseline		0.270

Activation functions are one of the most important DNN hyper-parameters. They determine if the output of DNNs has to be activated or not, based on each neurons' input and relevancy in prediction. In this paper we have explored 3 types of activation functions [29]: *relu*, *softmax*, and *tanh*. To determine the best one, we considered their speed and convergence. Each activation function has its own advantages: with *relu* a network converges very quickly, *softmax* effectively handles multiple classes, *tanh* is suitable to model inputs that have strong values.

Experiments with FFNN:

Vectorization: *one-hot* encoding

Hidden layers: 1, 2 and 3

Neurons: 256, 512 and 1024

Epochs: 100

Batch size: 256 (a number of training instances processed before a model is updated)

The experimental investigation proved the *softmax* activation function to be the most accurate. The model with the activation *softmax* achieved the highest accuracy equal to 28% (see Table 4) and it is above random and majority baselines. The best manually tuned FFNN architecture is presented in Figure 2 (this and the following architectures were plotted with the *plot_model* function in *Keras*). By the way, different numbers of neurons and hidden layers didn't show any significant impact on the accuracy.

Experiments with LSTM:

Vectorization: *word2vec* encoding (Section 4.2)

Architectures with simple and stacked LSTM ($1 \geq$ LSTM)

Neurons: 64, 128, 256, 512 neurons in the hidden

Epochs: 100

Batch size: 32

Activation function: *tanh*, *softmax*, *relu*

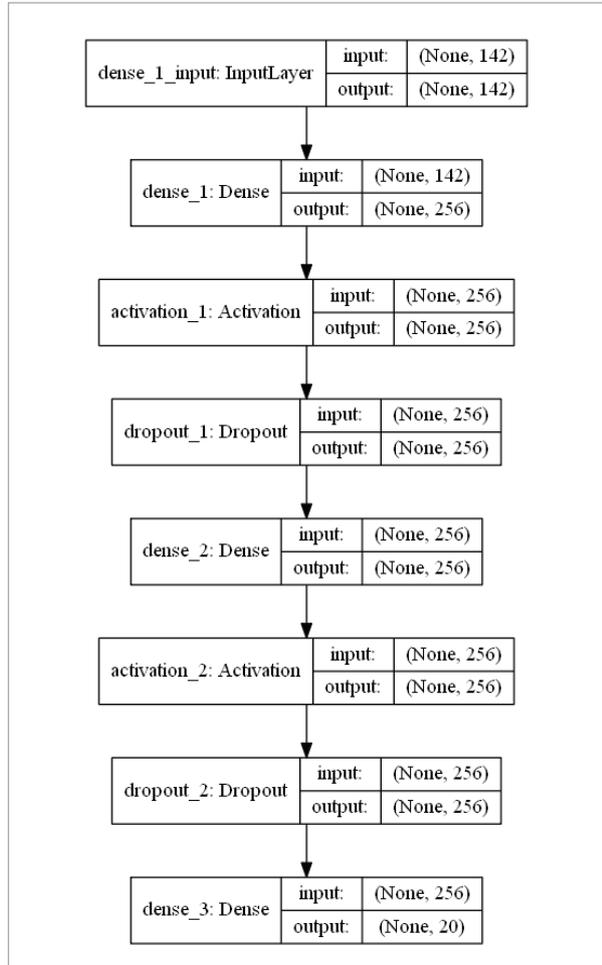
Other parameters were set to their default values.

The experimental investigation revealed that simple LSTM (1 = LSTM) with 64 neurons outperforms deeper architectures. The best accuracy is achieved using the *softmax* activation function (See Table 4) and the architecture presented in Figure 3. As presented in Table 4, *softmax* and *relu* activation func-

tions are both suitable (the difference between these results are not statistically significant). However, *softmax* still slightly outperformed *relu*.

Figure 2

Schematic representation of the best determined FFNN architecture



Experiments with BiLSTM:

Vectorization: *word2vec* encoding

Architectures with simple and stacked BiLSTM ($1 \geq$ BiLSTM)

Neurons: 64, 128, 256, 512 neurons in the hidden layers

Epochs: 100

Batch size: 32

Activation functions: *tanh*, *softmax*, *relu*

Other parameters were set to their default values.

The highest accuracy using BiLSTM model was achieved using the *softmax* activation function (See Table 4) and the architecture presented in Figure 4. The picture is very similar to LSTM: *softmax* and *relu* compete for the winner, however, *softmax* is slightly, but insignificantly better than *relu*.

Figure 3

Schematic representation of the best determined LSTM architecture

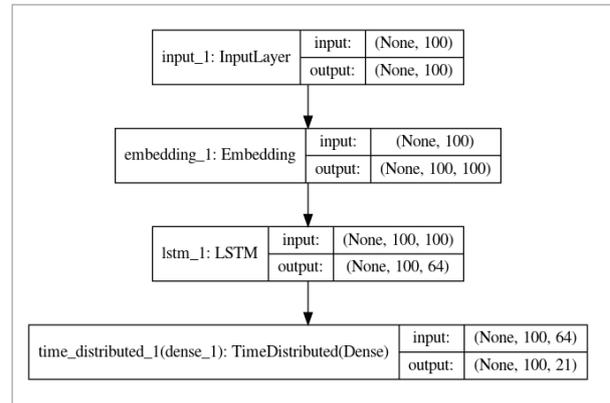
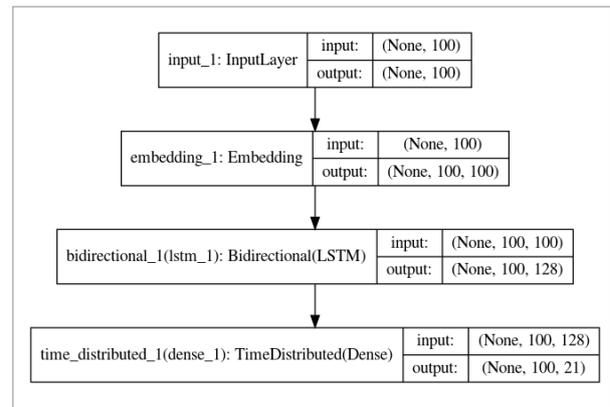


Figure 4

Schematic representation of the best determined BiLSTM architecture



Experiments with CNN:

Vectorization: *word2vec* encoding

Dimension: 1D dimension

Filters: 100

Kernel size: 3

Output layers: 2

Neurons: 64 and 1

Activation functions: *tanh, softmax, relu*

Other parameters were set to their default values

The best determined CNN architecture is in Figure 5. *Relu* activation function gives the best and significantly better accuracy compared to other activation functions (See Table 4). The result is not suitable, because it is under random and majority baselines.

Figure 5

Schematic representation of the best CNN architecture

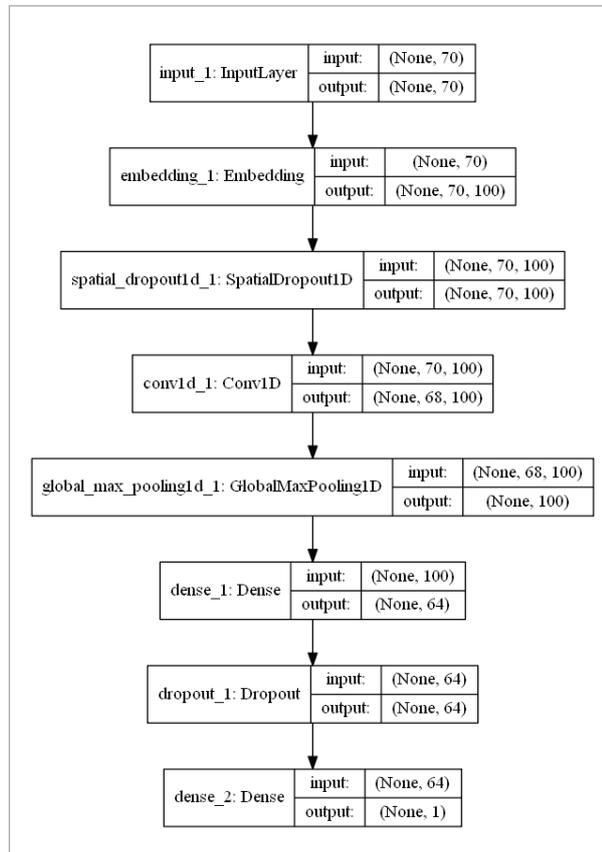


Table 4

Manually tuned hyper-parameter optimization results (in accuracies)

DNN	Tanh	Softmax	Relu
FFNN	0.00029	0.279	0.120
LSTM	0.004	0.896	0.891
BiLSTM	0.016	0.918	0.911
CNN	0.119	0.112	0.159

The best obtained accuracies of tuned architectures for different classifiers and activation functions are summarized in Table 4.

6.3. Automatic Hyper-Parameter Tuning

Manual optimization of hyper-parameters in DNN models is a tedious work. Besides, the expert knowledge may sometimes mislead, especially when unconventional solutions are needed. Due to this reason we have performed automatic hyper-parameter optimization. It was implemented with Python's library *Hyperas* [16]. It allows to tune discrete (by selecting from the list of determined values) and real (by selecting from the interval) hyper-parameter values when seeking for the highest accuracy on the validation dataset. The following options of hyper-parameter values were tested in our experiments:

Activation function: *sigmoid, softmax, tanh, relu, swish, selu* activation functions

Optimizers: *adam, sdg, rmsprop*

Batch sizes: 16, 32, 64, 128

Hidden layers: 1, 2 and 3

In this research, the hyper-parameter tuning was performed for LSTM, BiLSTM and CNN classifiers with the pre-trained word2vect embeddings for vectorization. The optimization was restricted to 20 iterations. The tuning of hyper-parameters was done in the directed manner using *tpe.suggest* strategy. *Tpe* stands for Tree-structured Parzen Estimator [4], which organizes hyper-parameters into a tree-like space. This Bayesian modelling approach decides which set of hyper-parameters to try in the next iteration based on the distribution of previous results.

The best determined hyper-parameter values and the best accuracies for different DNN classifiers with *word2vec* embeddings are presented in Table 5.

Table 5

Hyper-parameter optimization results

	LSTM	BiLSTM	CNN	CNN
Activation	Sigmoid	Sigmoid	Softmax	Sigmoid
Hidden layers	1	1	1	1
Neurons	32	64	32	32
Batch_size	32	32	32	32
Optimizer	rmsprop	rmsprop	rmsprop	rmsprop
Accuracy	0.890	0.918	0.610	0.610

7. Discussion

Our results allow us to make the following statements. The accuracy (particularly the best achieved) is much above random and majority baselines, therefore methods are considered appropriate and reasonable.

After testing different DNN classifiers (FFNN, LSTM BiLSTM and CNN) with a small amount of data, the manual tuning of hyper-parameters and different DNNs revealed that BiLSTM is the best option for our solving task. This RNN approach is adjusted to process the sequential data and able to consider the context (words before and after) when predicting the POS tag of the target word.

The best results with CNN were achieved when tuning hyper-parameter automatically and reached the accuracy of 61%. As we expected, CNN is not the worst option, thus it is not the most suitable choice for the sequential data: CNN could not beat the BiLSTM models (tuned manually and automatically) of which the accuracy is ~91.8%. Accuracies of FFNN and CNN are not satisfactory (often even below random and majority baselines). Besides, e.g., FFNN needs specific adjustments by feeding it with the context in an unnatural way: i.e., feature extraction and definitely more context needed is needed to compete with the other approaches.

The overall best accuracy of 91.8% was achieved with the BiLSTM method, which considers sequences of words in both direction (from the past to the future and from the future to the past). It seems that the sequential nature of the data for the POS tagging of the Northern-Ethiopic languages (and the Tigrinya language as the representative example of this group) is much more important than we expected at the beginning. Neither the particular keywords, nor the n-grams of words, but the sequential nature of the text (i.e., the order of words in sentences) is important for the Tigrinya POS tagging.

As the McNemar test proved, the difference between the closest achieved result (with BiLSTM + *relu* activation function in Table 4) from the best determined (BiLSTM + *softmax* in Table 4 or BiLSTM in Table 5) and equal to 91.8% is statically significant, because the calculated $p = 0.04$, which means $p < 0.05$. The p values for the 4 closest results to the best one achieved are summarized in Table 6 (here we demonstrate how

much differences are statistically significant between the best result and the closest results achieved by other approaches).

Table 6

Calculated p values to measure if differences to the best achieved accuracy = 91.8% are statistically significant

In comparison with:	Accuracy	p value
BiLSTM + <i>relu</i> (Table 4)	0.911	0.04
LSTM + <i>softmax</i> (Table 4)	0.896	1.04E-09
LSTM <i>relu</i> (Table 4)	0.891	1.42E-13
LSTM (Table 5)	0.890	1.86E-14

In the previous work done on the same dataset (presented in [20]) with the traditional ML methods of CRFs and SVMs the achieved accuracy was lower and equal to 90.89%. Our improvement on this traditional approach is statistically significant (with the calculated $p = 0.009 < 0.05$. Moreover, the traditional approaches have already reached their limits (in terms of discrete vectorization, parameters and sizes of the datasets); whereas the accuracy of DNNs is very sensitive to the sizes and variety of the training data. The accuracy of DNNs methods can be increased with more classification data (labeled with the POS tags) and more texts (for training word embeddings). This is the direction to go in the future.

The comparative experiments with the different DNN approaches are the significant achievement for the Tigrinya language, as well for the whole group of the Northern-Ethiopic languages sharing similar characteristics. The work is also influential from the practical perspective as well: the accurate POS tagger can boost the continues research in the other NLP tasks, especially for the resource-scarce Tigrinya.

8. Conclusion and Future Work

In this paper we are tackling the POS tagging problem for the Tigrinya language and achieve the best accuracy of ~92%. The best accuracy was achieved with the BiLSTM classifier applied on neural word embed-

dings, carefully selected DNN architecture and optimized hyper-parameter values. This achieved result is much better than random and majority baselines.

The paper also presents the comparative POS tagging experiments with various DNN classifiers (specifically, FFNN, LSTM, BiLSTM and CNN). To our knowledge comparative analysis of several DNN classifiers has never been performed before on any of the Northern-Ethiopic languages for the POS tagging task.

For various DNNs we have investigated their different architectures and sets of hyper-parameter values by tuning those parameters manually (via expert insights) and automatically (with automatic parameter optimization methods). Such comprehensive search (seeking for the best POS tagger solution manually

and automatically) has never been performed before for any of the North-Ethiopic languages.

The achieved results and especially recommendations on classifiers, their architectures and hyper-parameter values are important not only for the Tigrinya language, but for the whole group of Northern-Ethiopic languages sharing similar characteristics.

In the future we are planning to continue our research for the Tigrinya language in the other areas of NLP by solving tasks which could not even be initiated before (without the accurate POS tagger). It would be also interesting to perform experiments for the other Northern-Ethiopic languages (e.g., Tigre, Saho, Ge'ez) to see how much they can benefit from the recommendations in this research.

References

1. Amsalu, S., Gibbon, D. Finite State Morphology of Amharic. International Conference on Recent Advances in Natural Language Processing, 2005, 47-51.
2. Argaw, M. Amharic Part-of-Speech Tagger using Neural Word Embeddings as Features, Addis Ababa University, Addis Ababa Institute of Technology, Master Thesis, 2019.
3. Azath, M., Kiros, T. Statistical Machine Translator for English to Tigrigna Translation. International Journal of Scientific and Technology Research, 2020, 9(1), 2095-2099.
4. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B. Algorithms for Hyper-Parameter Optimization. NIPS, 2011.
5. Birhanie, W. K., Butt, M. Automatic Amharic Part of Speech Tagging (AAPOST): A Comparative Approach using Bidirectional LSTM and Conditional Random Fields (CRF) Methods. Advances of Science and Technology. 7th EAI International Conference, Bahir Dar, Ethiopia, 2020, 512-521. https://doi.org/10.1007/978-3-030-43690-2_37
6. Browne M., Ghidary S. S. Convolutional Neural Networks for Image Processing: An Application in Robot Vision. In: Gedeon T. D., Fung L. C. C. (eds) AI 2003: Advances in Artificial Intelligence. AI 2003. Lecture Notes in Computer Science, 2003, vol 2903. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24581-0_55
7. Chollet, F. Keras: Deep Learning Library for Theano and Tensorflow, 2015. Accessed on: March, 2020. Available: <https://keras.io/>.
8. Dhupal Deshmukh, R., Kiwelekar, A. Deep Learning Techniques for Part of Speech Tagging by Natural Language Processing. 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), IEEE Access, 2020, 76-81. <https://doi.org/10.1109/ICIMIA48430.2020.9074941>
9. Gambäck, B., Olsson, F., Argaw, A., Asker, L. Methods for Amharic Part-of-Speech Tagging. AfLaT '09: Proceedings of the First Workshop on Language Technologies for African Languages, 2009, 104-111. <https://doi.org/10.3115/1564508.1564527>
10. Gashaw, I., Shashirekha, H. L. Enhanced Amharic-Arabic Cross-Language Information Retrieval System Using Part of Speech Tagging. 6th IEEE International Conference on Advances in Computing, Communication and Control, ICAC3, Mumbai, India, 2019. <https://doi.org/10.1109/ICAC347590.2019.9036807>
11. Gasser, M. HornMorpho: A System for Morphological Processing of Amharic, Oromo and Tigrinya. Conference on Human Language Technology for Development, Alexandria, Egypt, 2011.
12. Gebregzabiher, T. Part of Speech Tagger for Tigrigna Language. Department of Computer Science, Addis Ababa University, Master Thesis, 2010.
13. Gebrekidan, B. Part of Speech Tagging for Amharic. Centre Tesnière. Université de Franche-Comté, France. Research Institute in Information and Language Processing, Master Thesis, 2010.
14. Golderberg, Y. Neural Network Methods for Natural Language Processing. Synthesis Lectures on Human

- Language Technologies, 2017. <https://doi.org/10.2200/S00762ED1V01Y201703HLLT037>
15. Graves, A., Schmidhuber, J. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks*, 2005, 18(5-6), 602-610. <https://doi.org/10.1016/j.neunet.2005.06.042>
 16. Hyperas: Keras + Hyperopt: A Very Simple Wrapper for Convenient Hyperparameter Optimization. Available: <https://github.com/maxpumperla/hyperas>. Accessed on: March, 2020.
 17. Hochreiter, S., Schmidhuber, J. Long Short-Term Memory. *Neural Computing*, 1997, 9(8), 1735-1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
 18. Jurafsky, D., Martin, J. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2008.
 19. Keleta, Y., Yamamoto, K., Marasinghe, A. Nagaoka Tigrinya Corpus: Design and Development of Part-of-speech Tagged Corpus. *The Association for Natural Language Processing*, 2016, 413-416.
 20. Keleta, Y., Yamamoto, K., Marasinghe, A. Tigrinya Part-of-Speech Tagging with Morphological Patterns and the New Nagaoka Tigrinya Corpus. *International Journal of Computer Applications*, 2016, 146(14) 33-41. <https://doi.org/10.5120/ijca2016910943>
 21. Kim, Y. Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, 1746-1751. <https://doi.org/10.3115/v1/D14-1181>
 22. Kotsiantis, S. B. *Supervised Machine Learning: A Review of Classification Techniques*. *Informatica* 2007, 31, 249-268.
 23. Mamo, G. Automatic Part of Speech Tagging for Amharic: An Experiment Using Stochastic Hidden Markov (HMM) Approach. Master's thesis, Addis Ababa University, 2001.
 24. Marcus, M. P., Santorini, B., Marcinkiewicz, M. A. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 1993, 19(2), 313-330. <https://doi.org/10.21236/ADA273556>
 25. McNemar, Q. Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. *Psychometrika*, 12(2), 153-157. <https://doi.org/10.1007/BF02295996>
 26. Mikolov, T., Chen, K., Corrado, G., Dean, J. Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781, 2013.
 27. Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A. Advances in Pre-Training Distributed Word Representations. *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.
 28. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*, 2013, 26, 3111-3119.
 29. Nwankpa, Ch., Ijomah, W., Gachagan, A. Marshall, S. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. 2018, arXiv:1811.03378v1.
 30. Plonis, D., Katkevičius, A., Gurskas, A., Urbanavičius, V., Maskeliūnas, R., Damaševičius, R. Prediction of Meander Delay System Parameters for Internet-of-Things Devices Using Pareto-Optimal Artificial Neural Network and Multiple Linear Regression. *IEEE Access*, 2020, 8, 39525-39535. <https://doi.org/10.1109/ACCESS.2020.2974184>
 31. Połap, D. Analysis of Skin Marks Through the Use of Intelligent Things. *IEEE Access*, 2019, 7, 149355-149363. <https://doi.org/10.1109/ACCESS.2019.2947354>
 32. Řehůřek R., Sojka, P. Software Framework for Topic Modelling with Large Corpora. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010, 45-50. doi: 10.13140/2.1.2393.1847.
 33. Schmid, H. Part-of-Speech Tagging with Neural Networks, *COLING'94: Proceedings of the 15th Conference on Computational Linguistics*, 1994, 1, 172-176. <https://doi.org/10.3115/991886.991915>
 34. Sebastiani, F. *Machine Learning in Automated Text Categorization*. *ACM Computing Surveys* 2002, 34, 1-47. <https://doi.org/10.1145/505282.505283>
 35. Seyoum, B. E., Miyao, Y., Mekonnen, B. Y. Universal Dependencies for Amharic. *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC, European Language Resources Association (ELRA)*, 2019, 2216-2222.
 36. Tedla, Y., Yamamoto, K. Analyzing Word Embeddings and Improving POS Tagger of Tigrinya. *Proceedings of the 2017 International Conference on Asian Language Processing, IALP 2017*, 2018, 115-118. <https://doi.org/10.1109/IALP.2017.8300559>
 37. Tensorflow. Available: <https://www.tensorflow.org/> Accessed on: March, 2020.

