



VYTAUTAS MAGNUS UNIVERSITY

FACULTY OF INFORMATICS

APPLIED INFORMATICS DEPARTMENT

Senait Gebremichael Tesfagergish

**DEEP LEARNING-BASED PART-OF-SPEECH TAGGING OF THE  
ETHIOPIC LANGUAGE**

Master final thesis

Applied informatics study programme, state code 6211BX012

Study field Informatics

**Supervisor** Dr. Jurgita Kapociute Dzikiene

\_\_\_\_\_  
(signature)

\_\_\_\_\_  
(date)

**Defended**

prof. dr. Daiva Vitkutė-Adžgauskienė

(Dean of Faculty)

\_\_\_\_\_  
(signature)

\_\_\_\_\_  
(date)

Kaunas, 2020

# CONTENT

LIST OF FIGURES.....	3
LIST OF TABLES .....	4
LIST OF ABBREVIATIONS .....	5
SANTRAUKA .....	7
ABSTRACT .....	8
1 INTRODUCTION .....	9
2 RELATED WORKS .....	11
3 MORPHOLOGY OF THE TIGRINYA LANGUAGE.....	14
4 THE CORPORA.....	18
5 VECTORIZATION .....	23
5.1 One-hot encoding .....	23
5.2 Word2Vec [33] .....	24
6 Dnn classifiers for tigrinya pos tagging .....	29
6.1 Feed Forward Neural Network .....	31
6.2 Recurrent Neural Network (LSTM and BiLSTM) .....	33
6.3 Convolutional Neural Network (CNN) .....	36
7 Experiments and results .....	39
7.1 Manually tuned DNN architectures and hyper-parameters .....	41
7.2 Automatic DNN Hyperparameter Optimization.....	49
8 Discussion .....	50
9. CONCLUSION AND FUTURE WORK.....	51
References .....	52
APPENDIX .....	56

# LIST OF FIGURES

Figure 1. HornMorpho Output.....	12
Figure 2. Ge'ez-Script.....	15
Figure 3. Morphology and POS in Tigrinya .....	17
Figure 4. The snippet from the Nagaoka corpus used for the Tigrinya POS tagging.....	20
Figure 5. A distribution of POS tags(classes) for the Tigrinya language in the Nagaoka corpus.....	21
Figure 6. . Label Encoding and One Hot Encoding.....	24
Figure 7. One-Word Context.....	25
Figure 8. Multi- word Context.....	25
Figure 9. Skip- Gram Model.....	27
Figure 10. Word2Vec Training Models.....	28
Figure 11. Representation of Biological Neuron.....	29
Figure 12. Representation of Artificial Neuron.....	30
Figure 13. Non-deep (AKA shallow neural network) and Deep neural network.....	30
Figure 14. FFNN with 1 hidden layer.....	31
Figure 15. Recurrent Neural Network.....	33
Figure 16. A schema of the unfolded RNN.....	34
Figure 17. Vanishing Gradient Problem of RNN.....	35
Figure 18. Bidirectional LSTM.....	35
Figure 19. LSTM interactive layers.....	36
Figure 20. Architecture of CNN.....	37
Figure 21. Illustration of CNN architecture.....	38
Figure 22. Instances of POS tags of NTC.1.0.....	40
Figure 23. Architecture of the best determined model of FFNN.....	43
Figure 24. Accuracies with the different activation functions using FFNN.....	44
Figure 25. Evaluation of the LSTM model.....	45
Figure 26. Accuracies with different activation functions using LSTM.....	45
Figure 27. Architecture of the best determined model of LSTM.....	46
Figure 28. Accuracies with different activation functions using BiLSTM.....	46
Figure 29. Architecture of the best determined model of BiLSTM.....	47
Figure 30. Accuracies with different activation functions using CNN.....	48
Figure 31. Architecture of the best determined model of CNN.....	48

## LIST OF TABLES

Table 1. Some patterns of gerundive verbs (V_GER).....	15
Table 2. Semitic Language Corpus available in Sketch Engine .....	19
Table 3. The classes of Nagaoka Corpus.....	19
Table 4. Training, testing and validation splits of the dataset used in Tigrinya POS tagging .....	21
Table 5. Random and Majority Baseline.....	40
Table 6. Activation Functions.....	41
Table 7. Hyperparameter optimization result.....	49

## LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ADJ	Adjective
ADV	Adverb
BLSTM	Bidirectional Long Short -Term Memory
CON	Conjunction
CNN	Convolutional Neural Network
CRF	Conditional Random Field
FNN	Feed Forward Neural Network
FW	Foreign Word
HMM	Hidden Markov Model
INT	Interjection
LSTM	Long Short-Term Memory
ML	Machine Learning
N	Noun
N_PRP	Noun Proper
N_V	Noun Verbal
NTC	Nagaoka Tigrinya Corpus
NLP	Natural Language Processing
NUM	Numeral
POS	Part of Speech tagging
PRE	Preposition
PRO	Pronoun
PUN	Punctuation
RNN	Recurrent Neural Network
SVM	Support Vector Machine

UNC	Unclassified
V	Verb
V_AUX	Verb Auxiliary
V_IMF	Verb Imperfective
V_IMV	Verb Imperative
V_GER	Verb Gerundive
V_PRF	Verb Perfective
V_REL	Verb Relative

# SANTRAUKA

Autorius	Senait Gebremichael Tesfagergish
Pavadinimas	Giliųjų neuroninių tinklų taikymas kalbos dalių nustatymui etiopų kalbos tekstuose
Vadovas	Jurgita Kapociute Dzikiene
Darbas pristatytas	Vytauto Didžiojo Universitetas, Informatikos fakultetas, Kaunas 17/06/2020
Puslapių skaičius	69

## ABSTRACT

Author	Senait Gebremichael Tesfagergish
Title	Deep Learning- Based Part-of-speech Tagging of the Ethiopic- Language
Supervisor	Jurgita Kapociute Dzikiene
Presented at	Vytautas Magnus University, Faculty of Informatics, Kaunas 17/06/2020
Number of pages	69

Deep Neural Networks have demonstrated the great efficiency in many NLP task for various languages. Unfortunately, some resource-scarce languages as, e.g., Tigrinya still receive too little attention, therefore many NLP applications as part-of-speech tagging are still in their early stages. Consequently, the main objective of this research is to offer the effective part-of-speech tagging solutions for the Tigrinya language having rather small training corpus.

In this paper the Deep Neural Network classifier, (i.e., Feed Forward Neural Network Long Short Term Memory, Bidirectional LSTM and Convolutional Neural Network) are investigated by applying them on a top of separately trained distributional neural Word2Vec embeddings. Seeking for the most accurate solutions, DNN models are optimized manually and automatically. Despite automatic hyper- parameter optimization demonstrates a good performance with the Convolutional Neural Network, the manually tested Bidirectional Long Short – Term Memory method achieves the highest overall accuracy equal to 91%.



# 1 INTRODUCTION

Part-of-speech (POS) tagging is a well-known task in NLP: it represents a process of mapping words from sentences into their corresponding parts-of-speech, based on their context and the meaning. POS tagging is a prerequisite for many NLP tasks such as dependency parsing, machine translation, speech recognition and so many others. POS tagging seems as a basic and easy task, but it is actually considered an AI-hard problem, especially due to word disambiguation. Until nowadays there is no single smart POS tagging solution that could work for all languages in the world. The problem arises due to the ambiguities in each language and our human understanding of those categories. "Even trained human annotators do not agree on the words' POS categories 3-4% of the times"[27].

"Language expressions are ambiguous, and computers do not have the commonsense and the world knowledge that humans have when they communicate"[5]. Examples of lexical ambiguities in English are between verbs and nouns (e.g., rain, chair): singular and plural noun forms (e.g., fish, dice); present and simple past tense (e.g., put, hit), etc. Moreover, words may have different semantical meanings (e.g., *make* can mean cook or create) (however, semantics is not in a scope of this research). Despite catching a meaning of some word in a sentence (when a situation and conditions is clear) for human beings is a rather easy task, it sometimes goes beyond the capability of machine understanding. Therefore, the main objective of the POS tagging is to feed a computer program with the diverse and comprehensive data that can be used to train a tagger how to understand meanings of words in sentences correctly.

Consequently, to train an accurate POS tagger, a large annotated corpus is required. Unfortunately, for many resource-scarce languages such annotated corpora do not exist. Besides, there is no one right solution that could work for all languages: each language is different and difficult in its own way, therefore requires adaptation. The classical approaches applied to a POS tagging problem are rule-based and stochastic-methods, in which rule based taggers usually function according to the rule set manually prepared by linguists and stochastic taggers use probabilistic mathematical models acquired from the corpus. Hybrid approaches usually blend rules and statistics. Unsupervised solutions do not require annotated data and usually search for morphemes (stems and affixes). However, mechanisms based on the morphemes search work for the agglutinative languages only and still underperform supervised approaches; therefore, to expect better results the annotated data is crucial.

The resource-rich languages (such as English) have large corpora, well-developed language tools and therefore POS taggers can be further developed based on finding from previous studies. On the contrary, the Tigrinya language has only one corpus, available for the POS tagging, which is rather small.[9]. The previous Tigrinya POS tagging research (presented in [1] [31]) was based on training the traditional supervised machine learning techniques; in particular, the Conditional Random Fields (CRFs) classifier and Support Vector Machines (SVMs). Despite the authors managed to achieve good results they have not tested state-of-the art techniques yet that could boost the accuracy even further.

This Tigrinya POS tagging research is focused on the application of the state-of-the-art Deep Learning (DL)-based solutions. Seeking for the best model we test several Deep Neural Network (DNN) classifiers (Feed Forward Neural Network - FFNN, Long Short-Term Memory Network – LSTM, Bidirectional LSTM - BiLSTM and Convolutional Neural Network - CNN) by tuning hyperparameters manually and automatically. It resulted in finding an accurate model able to reach the accuracy of 91% with the BiLSTM method.

The main goal of this research is to solve the POS tagging task with innovative methods. In order to achieve this goal, the following tasks were completed.

1. Dataset Preparation
2. Experiment with different approaches (classifiers and vectorization)
3. Manual and automatic tuning of hyper-parameters

**Practical value:** to have POS tagger for the Tigrinya language

**Scientific value:** to try deep learning approaches in the Tigrinya POS tagging( and they have never been tested before)

This paper is organized as follows. Related Works are described in Section 2. Section 3 explains the detailed structure and morphology of the Tigrinya language. The corpora used for this research and vectorization methods are discussed in Section 4 and 5 respectively. Analysis of the Methods applied for solving the POS tagging task are analyzed in Section 6. Experiments and results are presented in Section 7. This paper ends with the Discussion and Conclusion about the overall objectives and achievements of this research and Future work.

## 2 RELATED WORKS

NLP research for Tigrinya is neglected due to the absence of linguistical resources of annotated and unannotated data in the electronic format. Tigrinya is spoken by 7 million of people and it is official language in Eritrea and Northern Ethiopia. Unlike most of the Semitic languages (such as Arabic, Hebrew and Amharic), Tigrinya has not benefited from the research on the language technologies yet. Amharic morphological patterns are the most similar to patterns in Tigrinya. Amharic is supported by Google, a lot of other search is done for this language; however, in this description the major focus will be on the recent achievements on the topic of the part-of-speech tagging.

NLP for Amharic language has begun in 2001. (presented in [40]). The most recent research done (presented in [5]), explore the part-of-speech tagging for Amharic with Conditional Random Fields (CRF++), Brill, and TnT of Python Implementation in NLTK. The best accuracy was achieved with CRL and equal to 90.95%. The paper describes, that the usage of partially cleaned corpus, selection of most informative features, morphological studies of the language, and application of parameter tuning and tagging algorithms helps to boost the accuracy. Corpus size for this experiment included 210,000 tokens with 31 tag set (11 basic). The morphology of the language is studied in detail and it helps to apply more features like vowel pattern, radicals, punctuation, alphanumeric and suffixes. These feature types were informative, which resulted in sufficiently good accuracy.

Mequanent Argaw(2019) (presented in[41]) trained the Amharic Part-of-Speech Tagger using the Neural word embeddings as the feature type and Neural Network as the classifier. By using the word embedding for vectorizing the text and extract the features from the given sentence, Neural Network Models had been tested using those embeddings. LSTM, FFNN and BILSTM were investigated and gave the accuracy of 92.8%, 88.88 % and 93.7% respectively.

**HornMorpho** [30], is a system for morphological processing of Amharic, Oromo and Tigrinya. Morphological analyzer segments words into morphemes and assigns grammatical categories to them(See Fig.1) Despite HornMorpho is also used for the POS tagging; it only can recognize nouns, verbs or copula, but other POS tags are ignored. Hence, the application options of such analyzer are very limited.

```

>>> l3.anal('ti', 'ብዙጋጥመኛ')
word: ብዙጋጥመኛ
POS: verb, root: <gTm>, citation: ኣጋጠመ
subject: 3, sing, masc
object: 1, plur
grammar: imperfective, reciprocal, transitive, relative

```

**Fig.1.** The example of the HornMorpho output (source: [30])

Teklay Gebregzabiher Abreha also presented the research about the POS tagging for Tigrinya language[31]. Using 26,000 words labelled with 36 tag types, he experimented with the Probabilistic Hidden-Markov Model (HMM) method combined with the rule-based tagger. “Vetebri- algorithm and Brill transformation-based Error driven learning is adapted for the HMM and Rule based taggers respectively” [31]. The applied HMM and rule-based approaches used alone achieved the accuracy of 89.13% and 91.8%, respectively. The hybrid approach (combining HMM and rule-based method) boosted the accuracy to 95.88%. This research is important for the Tigrinya language; however, it is done with the classical (HMM) and inflexible (rule-based) approach making it hardly transferable to the new domains (covering other topics) and language styles (fiction, legal texts, etc.) and types (spoken, non-normative, etc.).

In 2016, a new small-sized gold standard Nagaoka Tigrinya Corpus (NTC.1.0) released by Yemane Keleta, Kazuhide Yamamoto, Ashuboda Marasinghe[1]. Currently it is the only publicly available corpus for the Tigrinya. Using NTC 1.0 they have tested traditional ML methods, i.e., Conditional Random Fields (CRF) and Support Vector Machine (SVM) for the Part-of-speech tagging of Tigrinya. The original corpus contains 76 different tag types that were reduced to 20 classes for 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's affixes, comprising prefixes from one up to six characters in length, consonant-vowel patterns of the word (infixes), and suffix from one to five characters in length” [2]. The best accuracy equal to 89.92% and 90.89% was achieved with SVM and CRF,

respectively. Despite the results are promising they were achieved with the classical, but not state-of-the-art approaches (i.e., neural networks).

As it can be seen from the summarized related research, the POS tagging task for the Tigrinya language is solved only with classic approaches. The main contribution of this research is that the POS tagging task for the Tigrinya language is tackled with the innovative deep learning approaches and neural word vectorization.

### 3 MORPHOLOGY OF THE TIGRINYA LANGUAGE

Tigrinya belongs to the Semitic language branch of the Afro-asiatic family, along with Hebrew, Amharic, Maltese, Tigre and Arabic. Semitic languages are characterized with the rich derivational and influential morphology which results in the numerous variations of word forms. “The distinguishing feature of Semitic languages lies in the 'root-template' morphological pattern that is often composed of trilateral roots. The verb roots in Semitic languages comprise a sequence of consonants, whereas templates are patterns of vowels that are intercalated in between these consonants, forming various stems”[1].

The Ge'ez script is adapted to write other mostly Semitic languages, particularly Amharic in Ethiopia, and Tigrinya in both Eritrea and northern Ethiopia. Semitic languages are characterized by the rich inflectional and derivational morphology that generates numerous variations of word forms. Tigrinya (together with Amharic and Tigre) uses the ancient Ge'ez script as writing system. The Ge'ez script is one of the few native scripts that are still in active use in the African continent. According to the consonant-vowel syllable, Tigrinya identifies seven vowels sounds which are usually called 'orders'. There are also five-ordered alphabets which are variants of the basic 35 consonants. Altogether 275 symbols constitute the Tigrinya alphabet chart known as 'Fidel'. (see Fig.2)

Although resource-rich languages such as English have well-developed language tools, low-resource languages suffer from either the low grade or absence of electronic data support altogether to pursue NLP research. According to this, the Tigrinya language is one of such low-resourced languages. Unlike major Semitic languages (that enjoy relatively widespread NLP research and resources) the Tigrinya language has been ignored in the NLP community mainly due to the absence of annotated corpora.

In the linguistics field, morphology is an important aspect of any language (especially for morphologically complex), which studies an internal structure of words in sentences. In essence, morphology describes systematic relations between different word forms and their functions. Depending on the language, morphology can be divided into inflectional (changing a grammatical category, but rarely affecting its part-of-speech, e.g., *song*, *songs*) and derivational (changing a grammatical category).

Name		ä	u	i	a	e	ə	o	wa	yä
	IPA	æ	u:	i:	a:	e:	ə	o:	wə	jæ
hoy	h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ		
läwe	l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ	ሊ	
hāwt	h	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሐ	ሐ	
may	m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ	ሚ	ሚ
sāwt	s	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ	ሢ	
re's	r	ረ	ሩ	ሪ	ራ	ሪ	ር	ሮ	ሯ	ሯ
śat	s	ሰ	ሱ	ሲ	ሳ	ሴ	ሰ	ሰ	ሲ	
kaf	k	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቆ	
bet	b	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ	ቢ	
tāwe	t	ተ	ቱ	ቲ	ታ	ቲ	ት	ቶ	ቲ	
harm	h	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኆ	
nāhas	n	ነ	ኑ	ኒ	ና	ኔ	ነ	ኖ	ኗ	
'ālf	'	አ	ኡ	ኢ	ኣ	ኤ	አ	ኦ	ኦ	

kaf	k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኰ	
wāwe	w	ወ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ	ዖ	
'āyn	'	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ		
zāy	z	ዘ	ዙ	ዚ	ዛ	ዞ	ዟ	ዠ	ዡ	
yāmān	y	የ	ዩ	ዪ	ያ	ዬ	ይ	የ	ዩ	
dānt	d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ	ዷ	
gāml	g	ገ	ገ	ጊ	ጋ	ጌ	ግ	ገ	ጊ	
tāyt	t	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጠ	ጢ	
pait	p	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጸ	ጺ	
ṣādāy	s	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጸ	ጺ	
ḡāppā	ḡ	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ	ፇ	
āf	f	ፈ	ፋ	ፊ	ፋ	ፌ	ፍ	ፎ	ፋ	ፈ
psa	p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ	ፒ	

Fig.2. Ge'ez- Script ( source [42])

Tigrinya has a rich derivational and inflectional morphology. It has a root and pattern morphological system. The roots denotes the consonants in a particular word and patters are in the form of suffix, prefix and infix added to the roots and makes different inflectional and derivational word forms. Nouns in Tigrinya are inflected for gender, number, case and definiteness. "For example, ሃገራት (hagerat) - countries, ተማሃራይ (temaharay) - male student, ተማሃራት (temaharit) - female student. Tigrinya adjectives are inflected for gender and number. For example, ጸሊም (Tselim), ጸሊምቲ (Tselemti) meaning 'black' (masculine), 'blacks' respectively"[2]. Similarly, with the other Semitic languages Tigrinya has rich verb morphology. The modification of the verbs is also similar to the Nouns and the arrangement of the consonant and vowel patters results in different morphological structure of the basic verb form." For example, the root 'sbr' /to break/ of patter (CCC) has forms such as 'sebere' ( CVCVCV) in Active, 'te-sebere'(te- CVCCV) in Passive"[2]. Table.1 shows the most informative patters according to the distribution of the gerundive verb (V\_GER) throughout the NTC 1.0 corpus.

**Table 1.** Some patterns of gerundive verbs (V\_GER) ( source [2])

V_GER patterns	V_GER %	Examples
CeCiCu	31.1	feliTu 'he knew'
CeCeCiCu	17.8	tefeliTu 'it was known'
CeCiCiCu	15.9	tefaliTu ' to know each other'
CeCiCiCa	14.6	feliTika ' you knew'
aCiCiCu	11.8	afliTu 'made something known'
CeCiCoCI	11.3	feliTomI 'they knew'
CeCiCa	10.5	feliTa 'she knew'



Token: እንተዘይሓተትካዮ ‘IntezeyHatetIkayo’						
Gloss: if you did not ask him						
Word order:	if	not	asked	you	him	
Morphology:	InIte	zI	ayI	HatetI	ka	yo
Category:			VERB			
Subcategory:		REL	NEG	TAM: perf.		
Clitics:	CON					
Attribute:			SUBJ.PRO: 1 <sup>st</sup> ,Sg,M	OBJ. PRO: 3rd,Sg,M		
Stem: HatetI (CaCeCI)			Root: Htt			
POS: V_PRF_C (Perfective Verb with Conjunction)						

**Fig.3.** Morphology and POS in Tigrinya ( source [2])

## 4 THE CORPORA

A text corpus is a large collection of raw or annotated texts (or speech transcriptions) constructed specifically for language processing tasks [24]. In order to tackle a supervised POS tagging task, a gold-standard corpus containing words with annotated part-of-speech tags is needed. The function (so-called model) mapping words to their appropriate POS tags can be learned from examples, i.e., words (given as input) and their POS tags (as output) present in the corpus. The model is trained to adapt to the knowledge in the training corpus and later it can be used to POS tag unseen sentences. Thus, the accuracy of such model depends on the diversity and completeness of the annotated corpus. Besides, the more detailed tags it contains, the wider uses it has. In addition to the coarse-grained POS tags (as noun, verb), annotated corpora often contain more specific morphological information (i.e., fine-grained tags) (as gender, number, tense, etc.).

POS tagging research for the Semitic languages( i.e., Arabic, Amharic, Maltese and Hebrew) in contrast to Tigrinya language is done on a large annotated corpus with detailed morphological information covering:

- Transliteration of a word
- Lemmas
- Prefix, Suffix, Base strings
- Gender
- Status
- Polarity
- Tense
- Number
- Suffix number (Plural, Singular)

The statistics about an existing corpus for the Semitic Languages from the Sketch Engine Website is presented in Table 2 [25].

**Table 2.** The Semitic Language Corpus available in the Sketch Engine [25]

Languages	Number of tag types	Size in Words
Arabic	32	7,475,624,779
Amharic	29	25,975,846
Hebrew	23	895,876,116
Maltese	43	110,714,844

The annotated corpus for the Tigrinya POS tagging is taken from the *Nagaoka University of Japan* (NTC.1.0) [9]- To our knowledge it is the only corpus publicly available for the Tigrinya language. The corpus contains newspaper articles. The newspaper texts assure diversity of texts in different topics, domains and stylometry. The content covers news, editorial reports, reportages, commentaries, interviews, stories and biographies. [2]. NTC 1.0 was released in 2016 along with the POS tagging research based on the traditional machine learning approaches (Conditional Random Fields – CRF and Support Vector Machine – SVM). This corpus is a gold-standard corpus that initially has 73 tag types, consists of 72,080 tokens from 4656 sentences (the snippet from the corpus can be found in Fig. 4) [9]. Words in the corpus are labelled with 20 POS tag types(See Table.3). {**V\_PRF**, **UNC**, **V\_AU6X**, **V\_IMV**, **N**, **PUN**, **V\_REL**, **ADV**, **INT**, **N\_V**, **ADJ**, **NUM**, **N\_PRP**, **FW**, **V**, **V\_GER**, **CON**, **V\_IMF**, **PRE**, **PRO**} (the statistics about the distribution of tags can be found in Fig. 5) [2], making this problem a supervised multi-class classification problem. It is prepared in Ge'ez and English alphabets, but in our experiments only an English alphabet is used.

**Table 3.** The classes of Nagaoka Corpus

Category	Type	Label
Noun		N
	Verbal	N_V
	Proper	N_PRP
Pronoun		PRO
Verb		V
	Perfective	V_PRF
	Imperfective	V_IMF
	Imperative	V_IMV
	Gerundive	V_GER
	Auxiliary	V_AUX
	Relative	V_REL
Adjective		ADJ
Adverb		ADV

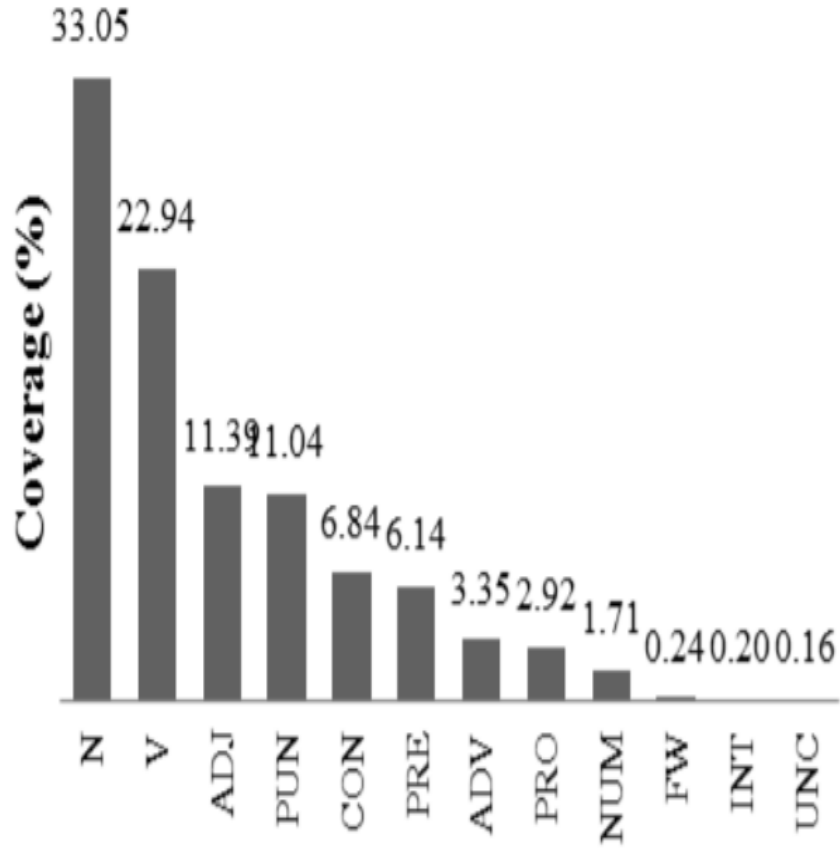
Category	Type	Label
Preposition		PRE
Conjunction		CON
Interjection		INT
Numeral		NUM
Punctuation		PUN
Foreign Word		FW
Unclassified		UNC

```

42 </s>
43 <s n="2">
44 <w type="PRE">ቅድሚያ</w>
45 <w type="ADJ">ብዙሃን</w>
46 <w type="N">ዓመታት</w>
47 <c type="PUN">"</c>
48 <w type="ADJ">እኛም</w>
49 <w type="N">ሰዓታት</w>
50 <c type="PUN">"</c>
51 <w type="ADJ">እኛም</w>
52 <w type="N">ሰዓታት</w>
53 <w type="CON">ወይ</w>
54 <w type="ADJ">እኛም</w>
55 <w type="N">ሰዓታት</w>
56 <w type="V_AUX">አይ</w>
57 <w type="V_REL">ዝመጽእ</w>
58 <c type="PUN">"</c>
59 <w type="V_REL">ዝብል</w>
60 <w type="ADJ">ግንደ</w>
61 <w type="N">እመላክታት</w>
62 <w type="V_GER">ነይሩ</w>
63 </s>
42 </s>
43 <s n="2">
44 <w type="PRE">qIdImi</w>
45 <w type="ADJ">bIzuHI</w>
46 <w type="N">Oametati</w>
47 <c type="PUN">"</c>
48 <w type="ADJ">aImIroawi</w>
49 <w type="N">sInIkilIna</w>
50 <w type="N">bIganEnI</w>
51 <w type="CON">weyI</w>
52 <w type="ADJ">IkeyI</w>
53 <w type="N">menafIsIti</w>
54 <w type="V_AUX">iyu</w>
55 <w type="V_REL">zImexII</w>
56 <c type="PUN">"</c>
57 <w type="V_REL">zIbIlI</w>
58 <w type="ADJ">gIguyI</w>
59 <w type="N">ameleKaKIta</w>
60 <w type="V_GER">neyIru</w>
61 <c type="PUN">::</c>
62 </s>
63 <s n="3">

```

Fig.4. The snippet from the Nagaoka corpus used for the Tigrinya POS tagging[9]



**Fig. 5.** A distribution of POS tags(classes) for the Tigrinya language in the Nagaoka corpus(source [2])

The corpus was split into training, validation and testing sets. The statistics can be found in Table 4.

**Table 4.** Training, testing and validation sets of the dataset used in Tigrinya POS tagging

	Percentage points	Number of tokens	Number of sentences
Training	60%	43248	2792
Validation	20%	14416	931
Testing	20%	14416	933

"Tigrinya words possess rich morphological information embedded in the form of prefixes, infixes and suffixes" [1]. For the POS tagging task, the overall processes contain the following steps:

1. Tokenization
2. Feature Extraction
3. Tagging

The tokenization task for languages as, e.g., English is straightforward; . but it requires a special adaptation to some other languages. For Semitic languages as Amharic, Tigre, Tigrinya words are usually represented as concatenation of morphemes each capable of having its own POS tag (e.g., NP for Noun attached with a Preposition). Feature extraction is needed to shape the problem by giving some property of the data. Thus, the Tigrinya language specifics makes it necessary for us to consider the following features:

- If a word is the first word in a sentence
- If a word is the last word in a sentence
- If a word is the last word in a sentence
- POS tags of 2 words before a target word
- A POS tag of 1 word before a target word
- POS tags of 2 words after a target word

## 5 VECTORIZATION

“Machine Learning (ML) is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed” [29]. Mathematically it performs calculations of Linear algebra (matrix, and vector operations) Calculus, statistics and probability theory. Accordingly, the features of ML are numerical values with which mathematical operations can be performed. However, it is different in NLP. NLP analyze text and enable human-computer interactions with the real-world natural languages. Since the NLP dataset is text, both input and output are also textual values. Therefore, word vectorization is implemented to bridge the gap between the datasets and the mathematical operations of the ML networks. "In language processing, the vectors  $x$  are derived from textual data, in order to reflect various linguistic properties of the text." [26]

"Word embedding is a collective name for a set of language modelling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers" [44].

There are different ways of representing a text as a vector. Such as:

- One-hot encoding
- N-gram models
- Word2Vec
- GloVe and etc.

In this paper, we used One-hot encoding and Word2Vec vectorization methods. The One-hot encoding is selected because it is the fastest and the simplest way to encode text, therefore often selected as the baseline approach. The neural Word2Vec vectorization is typically used for the input into the deep neural network classifiers

### 5.1 One-hot encoding

One-hot encoding is a distributed word representation method. The length of One-hot encoding vector is equal to the size of the vocabulary (i.e., number of distinct words from the text). Each word is represented with a unique vector having one value equal to 1 and all the rest are 0-oes.

This type of representation is always used in the output of the categorical data. The vector is equal to the number of classes and each class gets unique vector filled with 0-oes and one value equal to 1. (See Fig.6)

	<i>data</i>	<i>AI</i>	<i>book</i>	<i>Algorithms</i>
<i>data</i>	1	0	0	0
<i>AI</i>	0	1	0	0
<i>book</i>	0	0	1	0
<i>Algorithms</i>	0	0	0	1

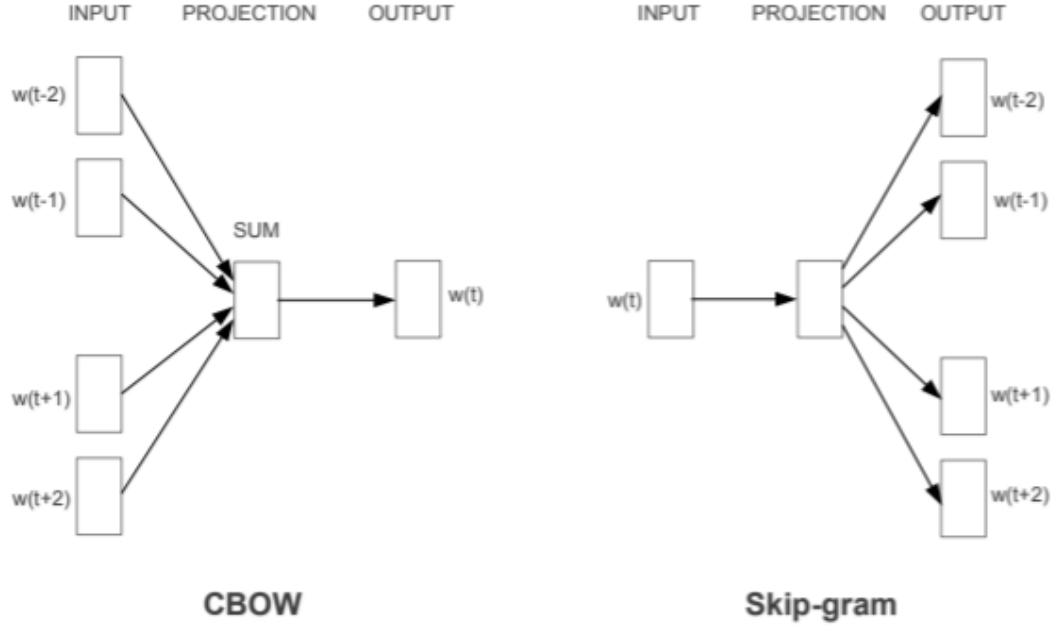
**Fig.6.** One-hot encoding.(source: [45])

Although the One-hot encoding is the straightforward way to encode the class data; it has limitations encoding the textual data when the vocabularies are large. It is especially complicated working with the highly inflective and derivationally complex languages, as Tigrinya. Thus, One-hot encoding vectors not only become very large (due to the large vocabularies), but very sparse as well (especially when the vectorized texts are small). In addition, there is no contextual information embedded within the representation.

## 5.2 Word2Vec [33]

Word2Vec is an embedding model trained with a one hidden layer neural network (see Fig. 6). Inputs  $(x_1, \dots, x_V)$  and outputs  $(y_1, \dots, y_V)$  of this two-layer neural network are One-hot vectors of distinct words. A hidden layer size  $N$  is arbitrary selected and represents the dimensionality space into which words are projected during training. The learned word embedding for word  $x_i$  is a vector of  $N$  values, i.e., weights between  $x_i$  and all  $N$  neurons  $(h_1, \dots, h_N)$  in the hidden layer. The input  $(x_1, \dots, x_V)$  and the output  $(y_1, \dots, y_V)$  relates words in the close context, therefore learned word embeddings carry syntactic and semantic word relationships. Depending on the input and the output, Word2Vec can be trained by using CBOW (Continuous Bag-of-Words) or Skip-gram algorithms. In the CBOW case model predicts a word from the surrounding context words while in the Skip-gram case it is vice versa, i.e., the model predicts a context of the input word.(See Fig.7)

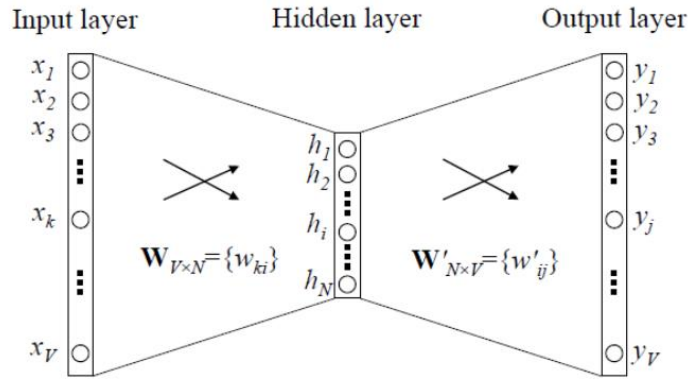




**Fig .7.** Word2Vec Training Models: CBOW vs. Skip-gram (source: [22])

The formal description of the word embedding training process is given below.

**One-word Context:** this model assumes only one word per context(See Fig.8). The architecture of this model takes One-hot encoding vector as input. Weights are multiplied with every input and transferred to the output using the softmax activation function. The goal of this network is to calculate the probability of the distribution of the vector representation of the words in the given context.



**Fig.8.** A scheme of Word2Vec model training (source: [33]).

$$P(W_j|W_i) \quad (1)$$

That is the hidden layer is calculated as:

$$h = W^T x \quad (2)$$

there is another weight matrix associated from the hidden layer to the output layer which is  $N \times V$  matrix. Using these weights, we can compute a score  $u_j$  for each word in the vocabulary,

$$U_j = v'_{w_j}{}^T h \quad (3)$$

Using the softmax activation function, we can calculate the posterior distribution of words, which is a multinomial distribution.

$$P(W_j|W_i) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^V \exp(u_{j'})} \quad (4)$$

**Multi-word Context (CBOW):** similar model with the one-word context, the probability distribution and the type of hidden layer makes it different. In here it predicts multinomial probability distribution in multi-word context word and many context words stores information about relation to the target word to other context words from the given corpus.(See Fig.9). The probability distribution calculated in the multi-word context is:

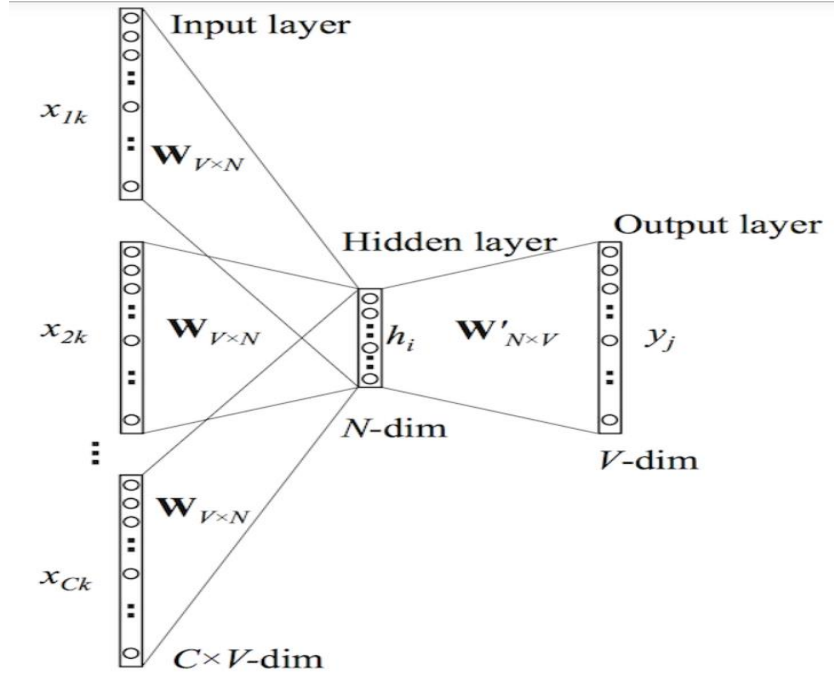
$$P(W_j|W_{1,1}, \dots, W_{i,c}) \quad (5)$$

This model takes the average of the vectors of the input context words and hidden layer is calculated as product of the average of the vectors of the input context and hidden weight matrix for every word in the context:

$$h = \frac{1}{c} W^T (x_1 + x_2 + \dots + x_c) \quad (6)$$

The loss function is:

$$E = -\log P(w_0|w_{1,1}, \dots, w_{1,c}) \quad (7)$$



**Fig.9.** Multi- word (CBOW) Context: predicts a word  $y_j$  form it's surronound context of words  $x_{1k}, ..., x_{Ck}$   
(source:[33])

**Skip-gram Model:** the Skip-gram model is introduced in Mikilov[43]. It is the inverted situation to the common bag of words model. It predicts the context of the wording given one target word in the input(See Fig.10). Hidden layer is calculated the same as *eq.(2)* that is :

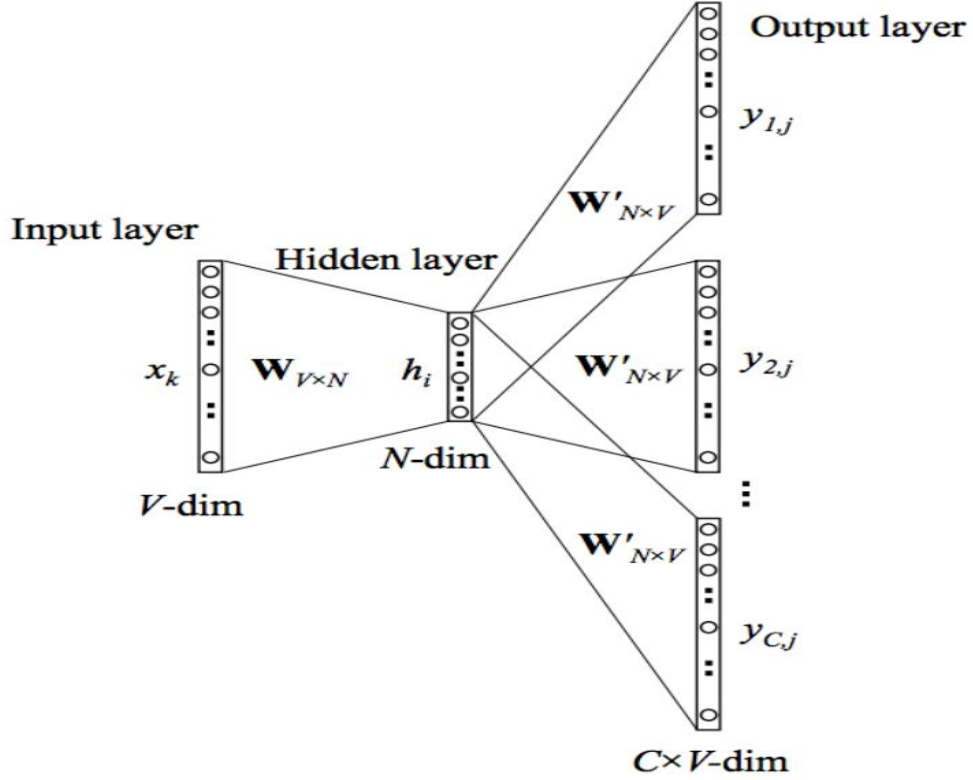
$$h = W^T x \quad (8)$$

hidden→output layer, it gives  $C$  miltinomial distribution. It is calculated as follows:

$$P(W_{c,j} = W_{o,c} | W_i) = y_{c,j} = \frac{\exp(u_j)}{\sum_{j'=1}^V \exp(u_{j'})} \quad (9)$$

Output layer is calculated as:

$$u_{c,j} = u_j = v'_{w_j}{}^T h \quad (10)$$



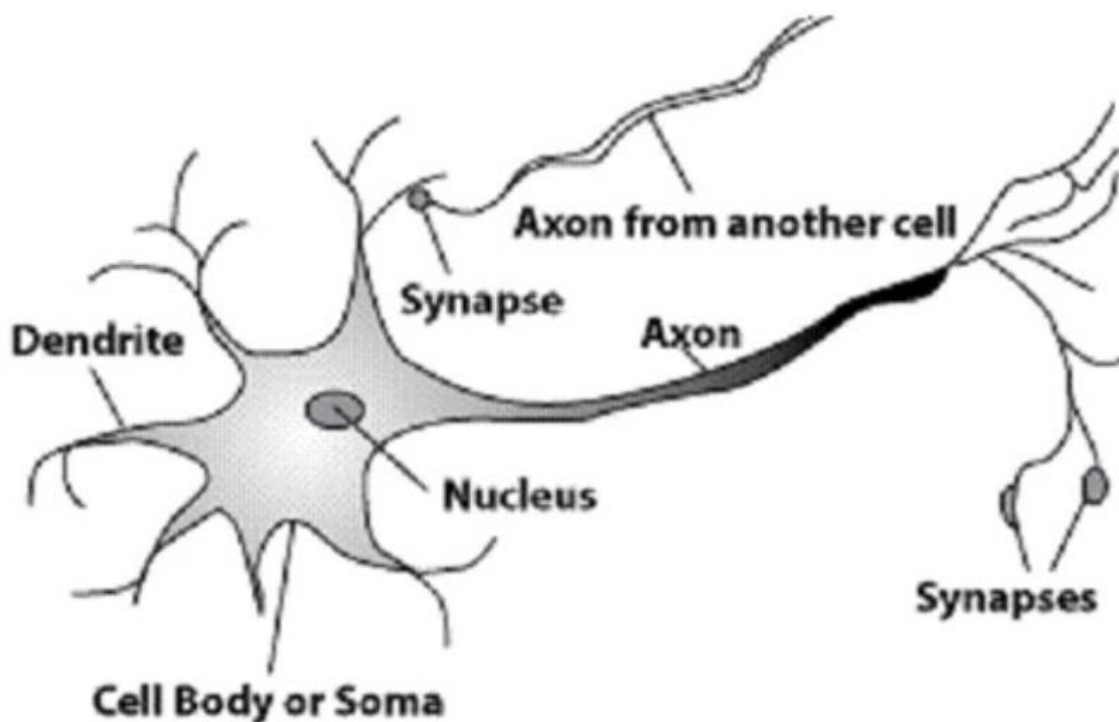
**Fig.10.** Skip- Gram Model: predicts a surrounding context of words  $y_{1j}, \dots, y_{Cj}$  for one inputted word  $x_k$   
(source: [33])

In this experiment the distributional Word2Vec Skip-gram approach is used. Before training, the text corpus was split into windows containing a focus word and its context: this information was used as the input and the output, respectively. Since similar focus words appear in the similar context; their Word2Vec vectors are also closer in the vector space. Given enough training data, the semantic meaning of each word is learned correctly.

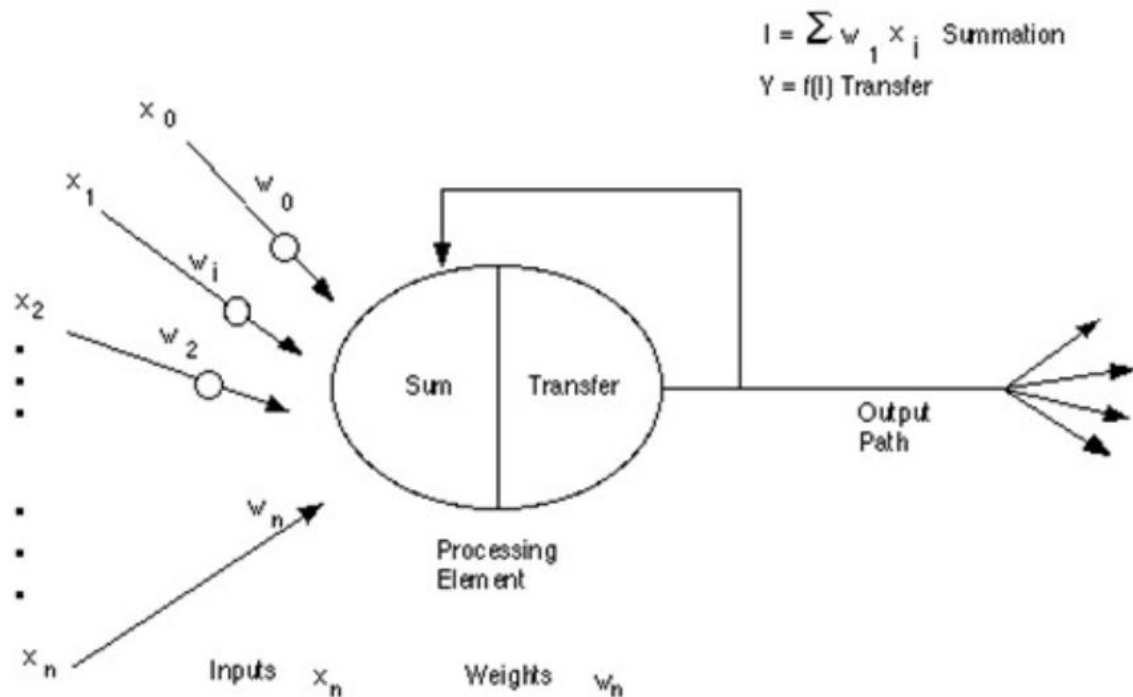
Since pre-trained publicly available word embeddings do not exist for Tigrinya language, they have been trained during this research from 4656 sentences of the Nagaoka corpus. These Word2Vec embeddings were trained with 100 dimensions and the context window size equal to 3 (which means that a context of 3 words before and 3 words after a target word were considered). All the rest parameters were set to their default values. To train the word embeddings, the Python programming language with genism [20] was used. The pre-trained word embeddings were saved and used afterwards in all experiments.

## 6 DNN CLASSIFIERS FOR TIGRINYA POS TAGGING

"Deep learning is a subfield of ML concerned with algorithms inspired by the structure and function of the brain called artificial neural networks [35]." In Human brain, neuron is a neural cell that has its nucleus, axon and many dendrites(See Fig.11). Basically, a neuron may be in activated or inactivated state. Activation can happen to neurons if the sum of the incoming excitatory stimuli is over a specified threshold. Synapses are the gaps between the axon of one neuron and dendrites of the other ones. Interaction between neurons are accomplished through the synapses. In fact, synaptic connections can have different strengths, that their impact upon the other neuron differs by the strength of the connection. Similarly, Artificial neuronal networks are modeled in that way. Inputs required to activate the model and the sum of the weighted inputs pass through hidden layer and mapped (converge) to a particular output by different activation functions.(See Fig.12)

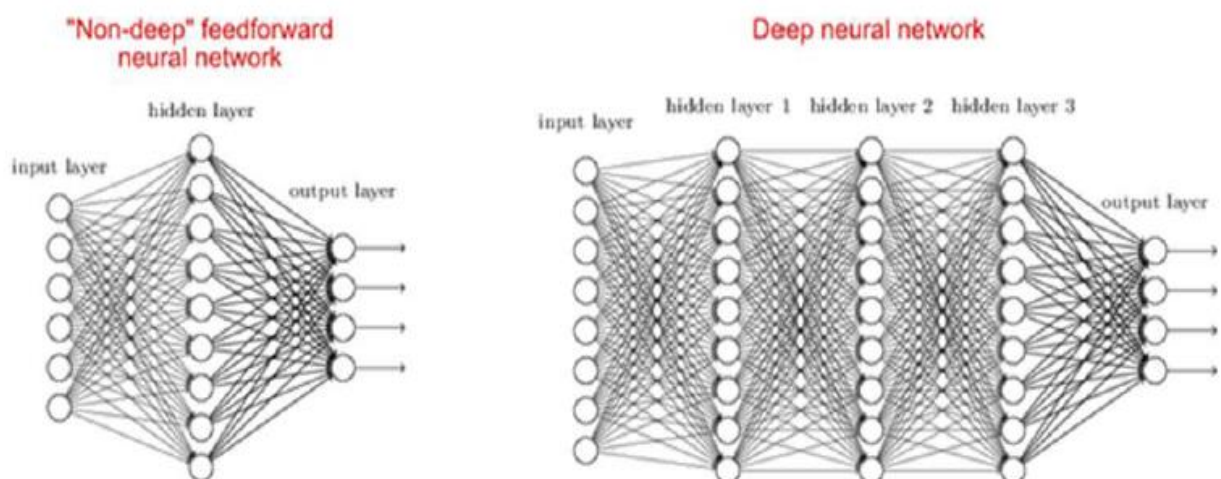


**Fig .11.** Representation of Biological Neuron (source: [36])



**Fig .12.** Representation of Artificial Neuron (source:[22])

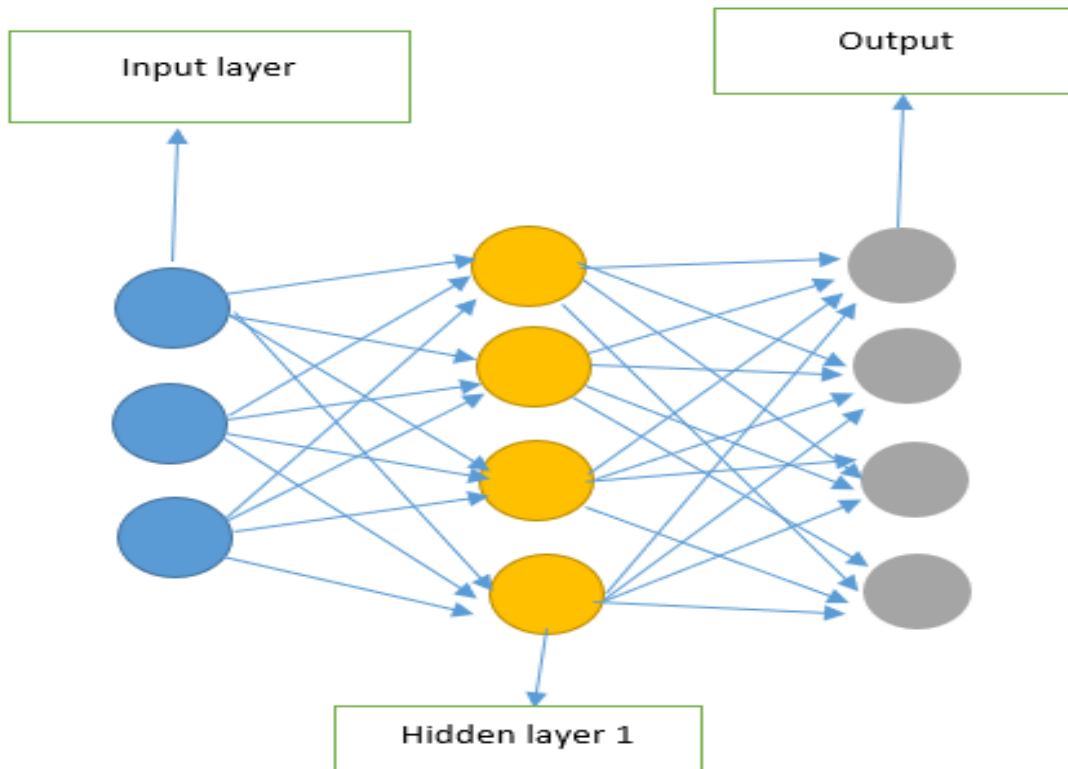
Components of the artificial neural networks are the Input, Hidden layer, weights, activation function and Output. Non-deep neural networks differ from the Deep neural network by the number of hidden layers, weights associated to each layer and activation functions.(See Fig.13)



**Fig.13.** Non-deep (AKA shallow neural network) and Deep neural network.(source:[36])

## 6.1 Feed Forward Neural Network

Deep feedforward networks also often called Feed Forward Neural Networks (FFNN), or multilayer perceptron (MLPs) are the quintessential deep learning models. The goal of feedforward network is to approximate some functions  $F^*$ .



**Fig .14 .** FFNN with 1 hidden layer

$$Y = F * x \quad (11)$$

Where,  $Y$  is output class

$x$  are input values

$F^*$  are weights

In FFNN there are no feedback connections in which an output of a model is fed back into itself. This DNN-type is the simplest network of all the DNN types. For the POS tagging problem, FFNN plays a role of classifier; besides, due to its simplicity FFNN can perform faster compared to the other types of DNNs. FFNN input is vectorized words passed to the deeper hidden layers. For each unit  $j$ , the function sums all the weights associated with every input and bias parameter  $O_j$  is added.

$$S = \sum W_{ij} X_{ij} + O_j \quad (12)$$

Where, W is the weight

X is the input

O is the bias parameter

The resulting network input S is passed through an activation function(See eq.13. Sigmoid function) in order to restrict the value range of the resulting activation  $X_i$  to a range of an interval.

$$f(S) = \frac{1}{1+e^{-S}} \quad (13)$$

- The network learns by adapting the weights of the connections between units, until the correct output is produced. Backward propagation adjusts the weights in FNN by searching the gradient descent on the error surface. Giving the output it modifies all the associated weights to reduce the overall error. The weight update  $\Delta W_i$ , i.e., the difference between the first and adjusted value of  $W_{ij}$  is here defined, Output Y of hidden and output neurons are calculated

$$Y_{1,j} = f[\sum W_{1,i,j} X_i] \quad (14)$$

$$Y_{2,k} = f[\sum W_{2,i,k} Y_{1,j}] \quad (15)$$

- Errors of the output and hidden neurons are calculated

$$\delta_{2,k} = Y_{2,k}(1 - Y_{2,k})(T_k - Y_{2,k}) \quad (16)$$

$$\delta_{1,j} = Y_{1,j}(1 - Y_{1,j}) \sum \delta_{2,k} W_{2,j,k}, \text{ where} \quad (17)$$

$T_k$  is Target vector

- Weights linked to output and hidden neurons are updated

$$W_{2,j,k}(t+1) \leftarrow W_{2,j,k}(t) + c \delta_{2,k} Y_{1,j} \quad (18)$$

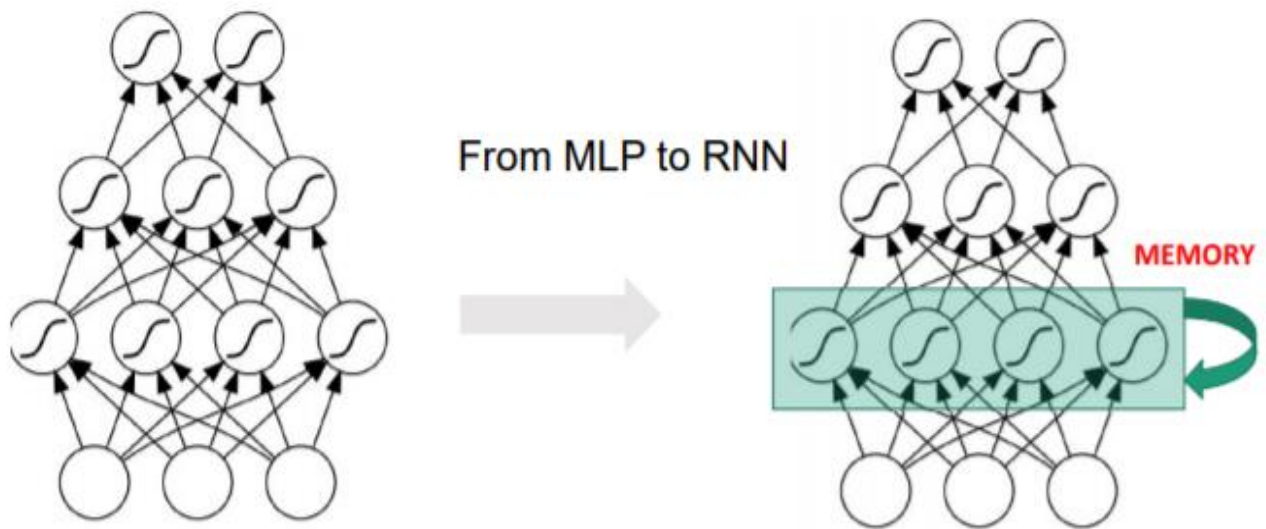
$$W_{1,i,j}(t+1) \leftarrow W_{1,i,j}(t) + c \delta_{1,j} X_i \quad (19)$$

Despite FFNN is fast and simple, it has limitations working with smaller amounts of training data. FFNN is incapable of working with sequential data( i.e., words in sentences), that is especially important in POS tagging; therefore, the sequential information (the context of the target word) was entered directly to the FFNN in the form of 2 succeeding and 2 preceding words are not enough for the method to demonstrate very good result in the POS tagging task. However, the FFNN method is still selected as the baseline approach to see how far the accuracy can go with such naïve measures.



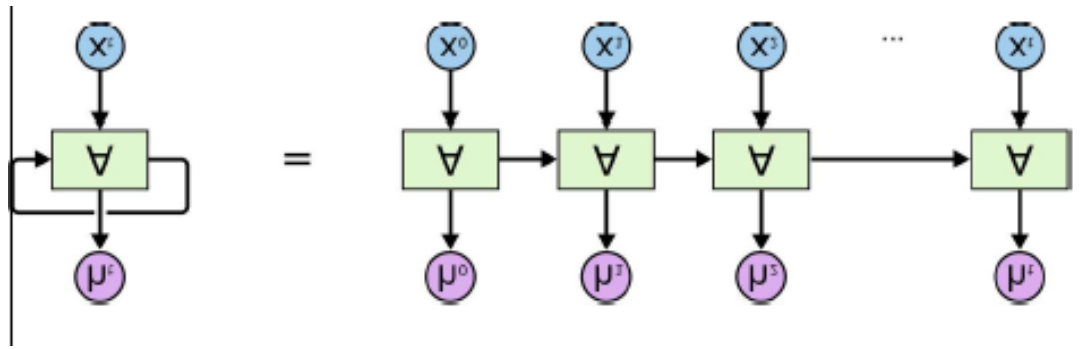
## 6.2 Recurrent Neural Network (LSTM and BiLSTM)

However, working with the sequential data (as text) the word order cannot be ignored. Especially the word order is informative for the Tigrinya language, because when changing it in a sentence, the meaning of a sentence also changes.



**Fig.15.** Recurrent Neural Network(source:[37])

For this reason, Recurrent Neural Networks (RNNs) should be a good choice for our POS tagging task. RNNs are methods having memory units and therefore adjusted to cope with the sequential data(See Fig.15). An input of the current time step is an input for the next time step, thus it has two inputs of which the first one is an actual input (i.e., incoming word from the sentence) and the second one is the output of the previous time steps (i.e., generalized information about previous words)(See Fig.16). Despite RNNs have the memory, they suffer from the vanishing gradient problem. RNNs remember only recent information and therefore are accurate when working with short-term dependencies. However, for the Tigrinya language longer dependencies matter in longer sentences and can have importance for the POS tagging.

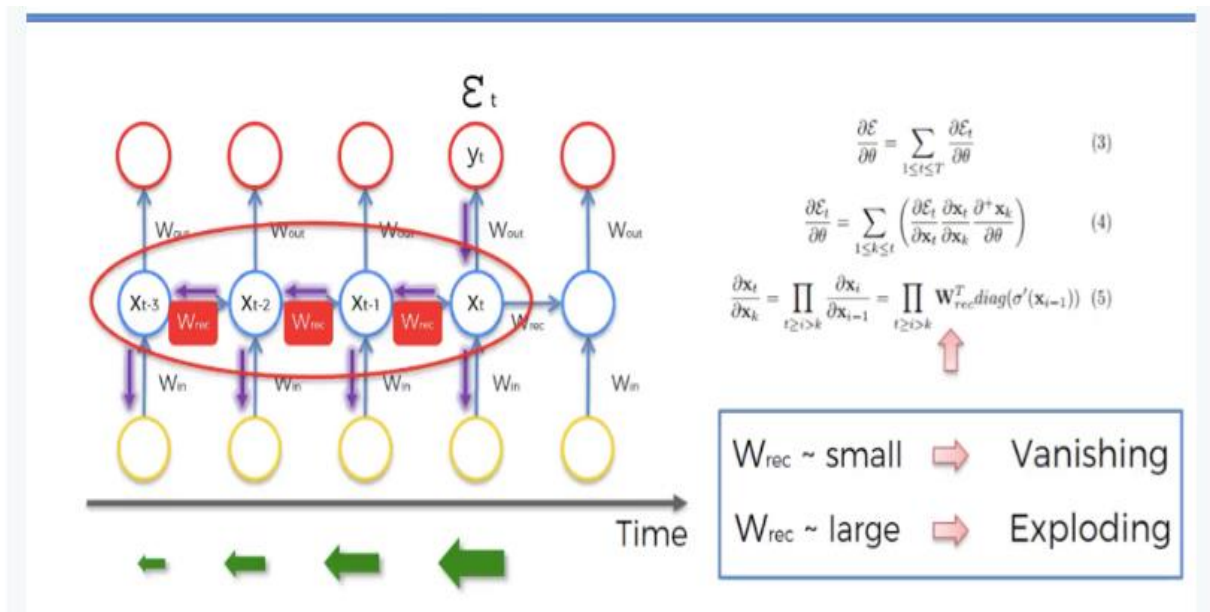


**Fig.16.** A schema of the unfolded RNN(source: [39])

Gradient descent algorithm finds the global minimum of the error for optimizing the accuracy of the network. In FFNN( as explained in Section 6.1) information travels directly from the input layer to the output layer through the hidden layers, while the weights are adjusted by back propagation. It is also similar for RNNs but there is more going on through the loop inside the network. That is:

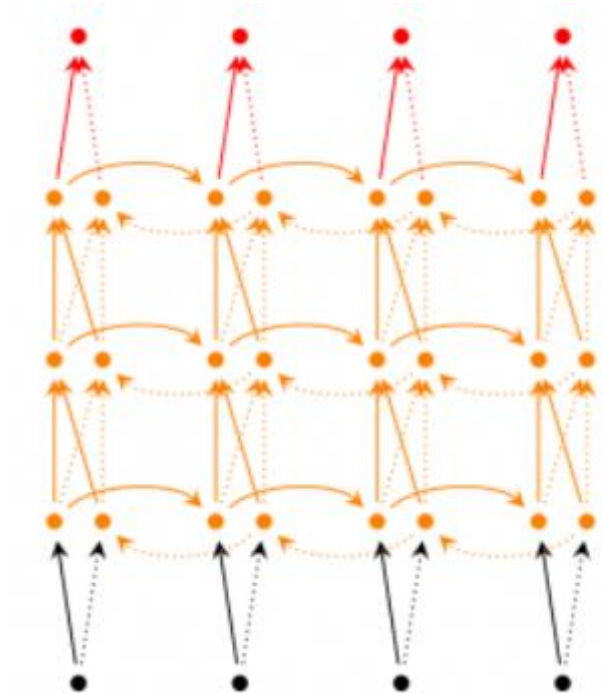
- Outputs from the previous time step are fed into the network as additional input together with the next time step
- Errors are searched by means of backpropagation and at each time step the loss function is calculated

Initially, during training weights are updated by comparing the output and the target. While calculating the loss function for the output, not only one neuron contributed to the outcome but also the previous time step. Therefore, every neuron in the network should have its weight adjusted in order to optimize the overall performance of the network. So, we need to propagate all the way back through time to the neurons. "The problem relates to updating the *wrec*(weight recurring)-the weight that is used to connect the hidden layer to themselves in the unrolled temporal loop"[37].Thus, weight values are assigned randomly and close to zero at the beginning of the network, and it happens to multiply the weight recurring to each time step multiple times and the value decreases very quickly.(See Fig.17)

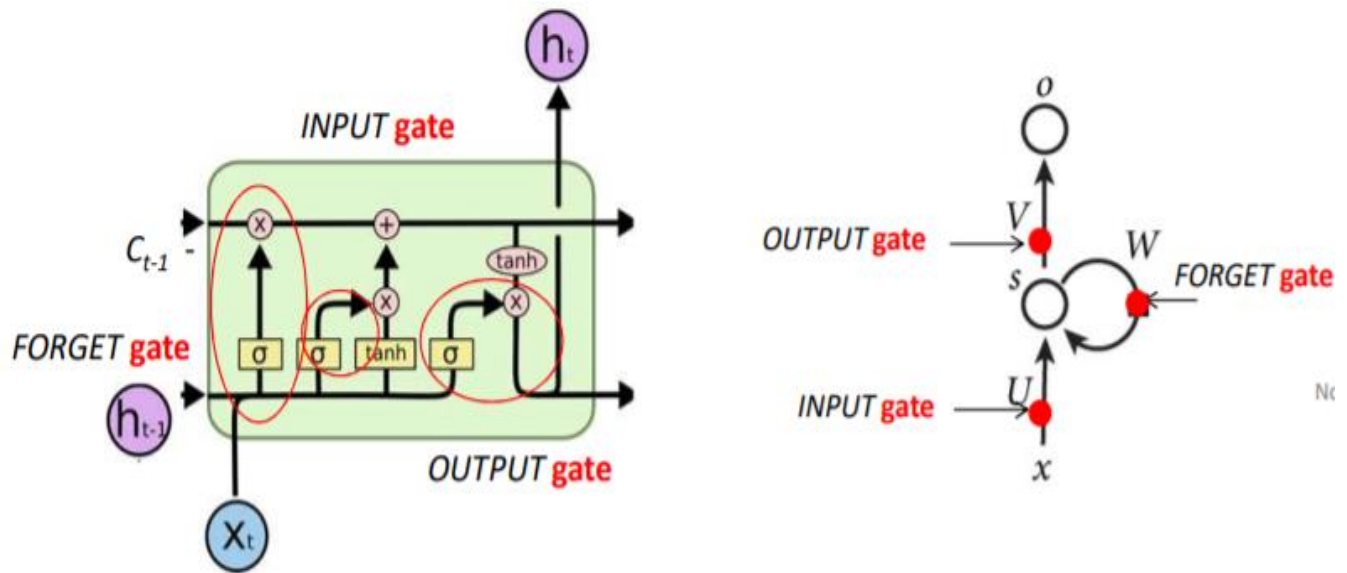


**Fig.17.** Vanishing Gradient Problem of RNN(source:[37])

For longer sequences LSTM and BiLSTM are used instead, because they are refined to remember longer sequences.



**Fig.18.** Bidirectional LSTM(source:[38])

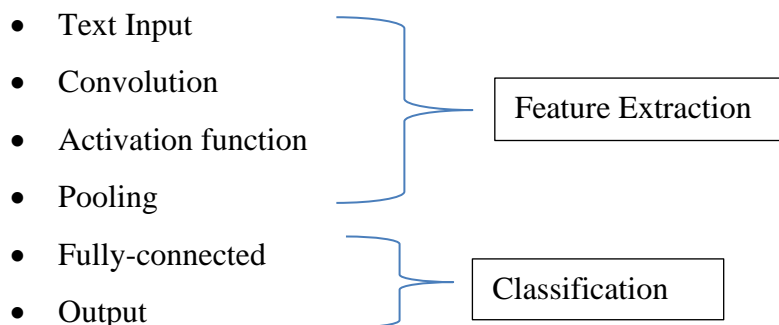


**Fig.19.** LSTM interactive layers (source: [39])

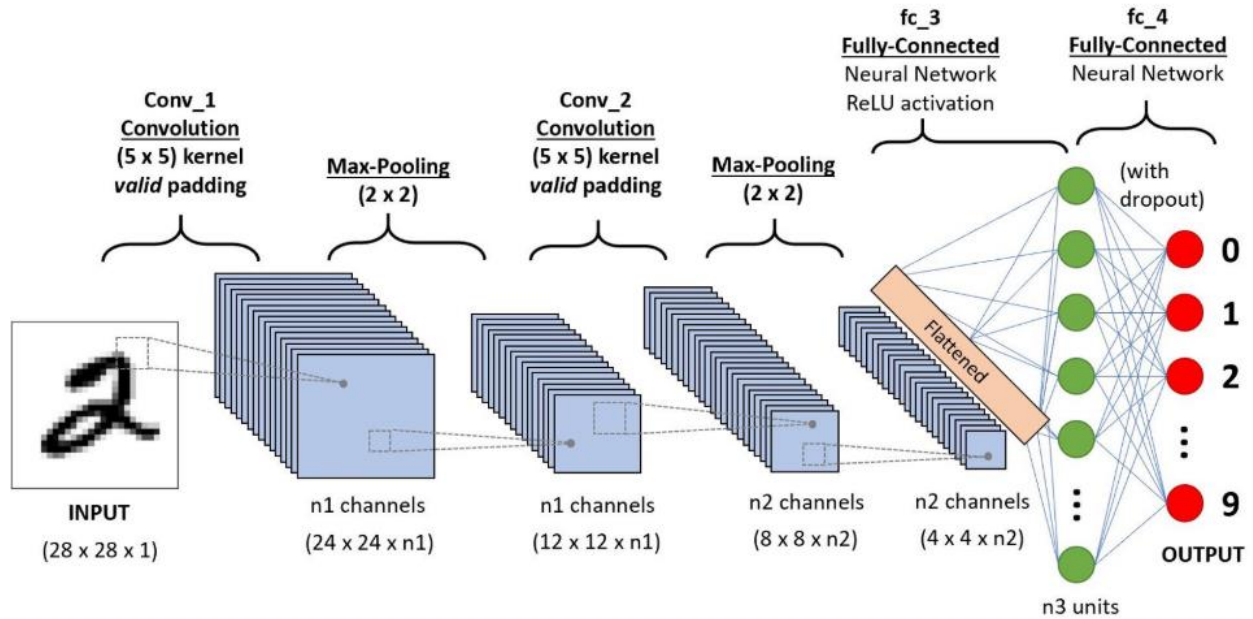
LSTM memory cells has 3 weighted gates adjusted during training the input, forget and output gates are used to memorize what information to input, forget and output, respectively.(See Fig.19) LSTM considers information going in only one direction: i.e., from the beginning to the future. However, Bidirectional LSTM considers sequences going in both directions: form-the-past-to-the-future and from-the-future-to-the-past.(See Fig.18) In the POS tagging task, some later words may give important details about the morphological form of the current word and it is especially evident with the Tigrinya language verbs carrying a lot of information about nouns and pronouns, mostly appearing at the end of sentences.

### 6.3 Convolutional Neural Network (CNN)

The Convolutional Neural Network consists of two sections : Feature Extraction and Classification. In with those two sections possesses several layers.



The above layers of CNN are, the inputs of pixel values, they are connected to the Convolutional layer neurons of their local regions. Filters or kernel are weights that were randomly generated during initialization. Output of a neuron is a dot product between the filters (weights) and local region of the input. Through the activation function applied to every neuron in the network, down sampling along the width and height dimensions is performed in the pooling stage. Then, it is connected to a feed forward neural network using the values from pooling layer as its input. For adjusting the weights backpropagation learning algorithm is applied.(See Fig.20)



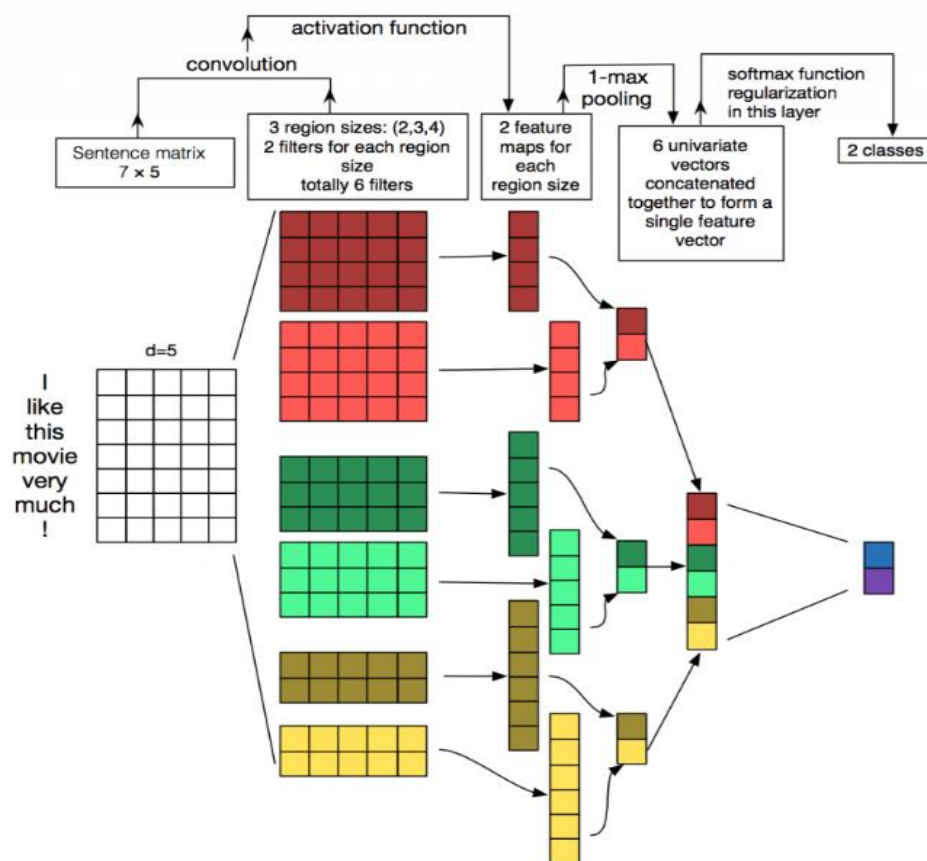
**Fig.20.** Architecture of CNN(source: [46])

CNN is used when classes are stable and if the network is not depending on the previous input or timesteps. In case of Natural Language Processing, CNN recently adapted and achieved a good result in classification task. Only one dimension (1D) is possible as 2D convolutions are possible with pictures. Number of neurons on the input is equal to the product of length of the word embedding vector and  $k$  number of words.

Despite RNNs(LSTM, BiLSTM) are adjusted to cope with the text, sometimes they underperform other conventional machine learning approaches. Recurrent methods consider the whole context including context unrelated to the target word. For this reason, CNNs are considered as a good solution. Initially these methods were used for image data, however, recently they are successfully adapted for the text. Instead of sequences of words, CNNs consider only patterns of sequential words (called as  $n$ -gram) and the width  $n$  of some filter determines how long patterns are worth of taking into account.(See Fig.21)

Instead of image pixels, word vectors of the pretrained word embeddings are used as the input of the network. That is each row in the input matrix denotes one token(word). Unlike in the pictures, where the filter slides over the local values of the matrix in both directions, in NLP the width of the filter is usually the same as the size of the input matrix. It only slides over the words(rows). The filters height or region size, usually is between 2-5 words. This is the window size that our network considered. Filter values are adjusted during training( when training samples with input/output values are fed into the neural network). The calculated cost function helps to adjust the filter weight values during the backpropagation. Filter values converge to the values for which the cost function is minimized.

CNNs could be a good option for solving this task, because for the Tigrinya language not all the information in a sentence matters, but only some context (approximately up to 3-grams) close to the target word.



**Fig.21.** Illustration of CNN architecture(source: [23])

## 7 EXPERIMENTS AND RESULTS

In this research experiments with the dataset (described in Section 2), using vectorization (described in Section 3) and the DNN methods (presented in Section 4) were performed. For the method implementation the Python programming language with TensorFlow [12] and Keras [13] was used.

For the evaluation the *accuracy* (presented in eq. 20) and *loss* (eq. 21) metrics were used.

$$accuracy = \frac{tp+tn}{tp+tn+fp+fn}, \text{ where} \quad (20)$$

*tp* (true positives) represent + instances with, determined class +;

*fn* (false negatives) represent + instances, with determined class -;

*fp* (false positives) represents – instances, with determined class +;

*tn* ( true negatives) represent – instances, with determined class - .

$$loss = - \sum_{c=1}^M Y_{o,c} \log \log(p_{o,c}), \text{ where} \quad (21)$$

$M$  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 observation  $o$  of class  $c$ .

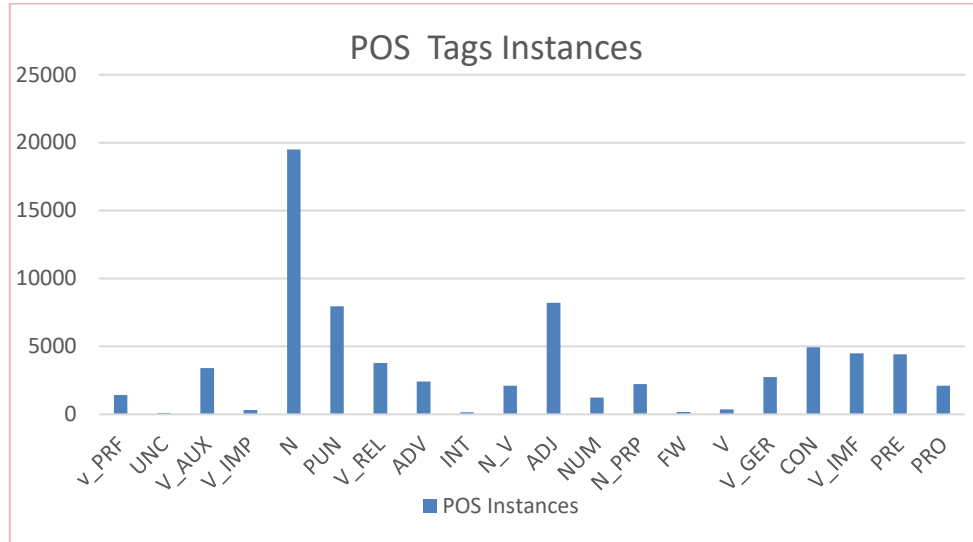
For any classification task it is necessary to have baseline values. The calculated accuracy values must exceed baselines for the method to be considered reasonable. Hence, the POS tagging result will be considered reasonable and appropriate if the calculated accuracy ( in eq. 20) is above random (see eq.22) and majority (eq. 23) baselines.

$$Random\ baseline = \sum P(c_i)^2 \quad (22)$$

$$Majority\ baseline = \max P(c_i), \text{ where} \quad (23)$$

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

Fig.22 shows distribution of the tokens in the NTC.1.0 corpus to each POS tags.[9]



**Fig.22.** Instances of POS tags of NTC.1.0 (number of instances taken from [9])

For evaluation of the experiments, the baselines are calculated using both Random (eq.22) and Majority(eq.23) baselines( See Table.5). The calculated accuracy of the experiment must be above 0.27.

**Table 5.** Random and Majority Baselines.

POS tags	Number of Instances	P( C)
V_PRF	1437	0.019937
UNC	113	0.078636
V_AUX	3409	0.047297
V_IMP	316	0.004384
N	19495	0.270475
PUN	7960	0.110437
V_REL	3787	0.052541
ADV	2415	0.033506
INT	145	0.002012
N_V	2104	0.29191
ADJ	8210	0.113906
NUM	1235	0.017134
N_PRP	2220	0.0308
FW	176	0.002442
V	357	0.004953
V_GER	2734	0.037932
CON	4933	0.068441
V_IMF	4501	0.062447

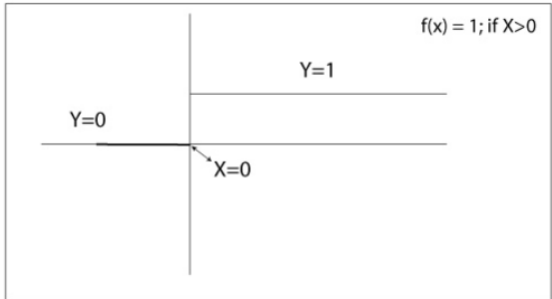
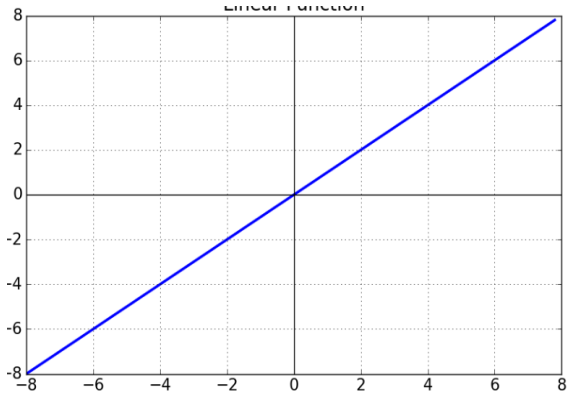


PRE	4424	0.061379
PRO	2106	0.029219
<b>Random Baseline</b>		0.127821
<b>Majority Baseline</b>		0.270475

## 7.1 Manually tuned DNN architectures and hyper-parameters

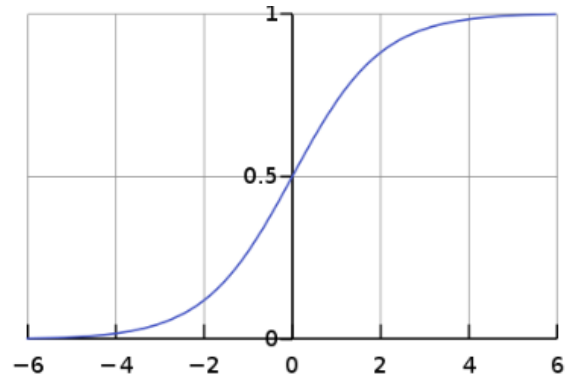
**Activation Functions:** Activation functions are one of the most important DNN hyper-parameters. They determine the output of DNNs deciding whether it should be activated or not based on each neurons' input and relevancy in prediction. Apart from the linear functions there are many non-linear activation functions used for DNN.(See Table.6.) In this paper we have explored 3 types of activation functions (*relu*[17], *softmax*[19], and *tanh*[18]) to determine the best one. They differ in speed and the way they converge. Each activation function has their own advantages, e.g., with *relu* a network converges very quickly, *softmax* effectively handles multiple classes, *tanh* is suitable to model inputs that have strong values. The most frequently used activation functions are: Binary step, Linear, ReLU, LeakyReLU, Sigmoid, Tanh and Softmax(See Table.6).

**Table.6.** Activation Functions (Summarized from [18])

Function	Activation Functions	Graphs
Binary Step	$f(x) = 1 \text{ if } x > 0 \text{ else ,}$ $0 \text{ if } x < 0$	
Linear	$f(x) = cx$	

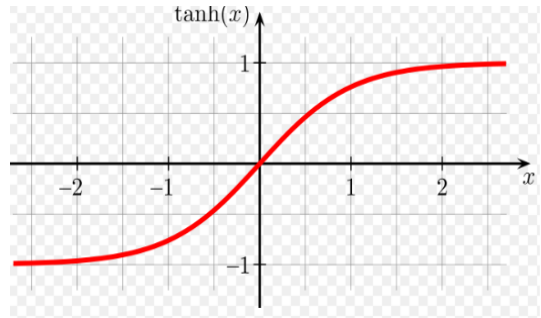
Sigmoid

$$f(x) = \left( \frac{1}{1 + \exp(-x)} \right)$$



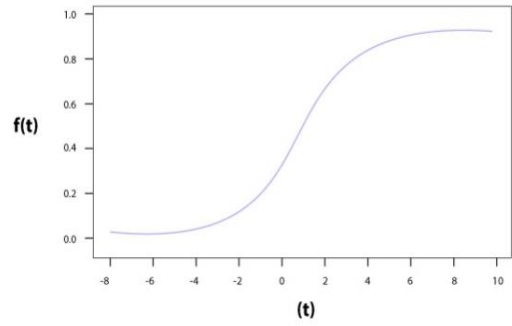
Tanh

$$f(x) = \left( \frac{e^x - e^{-x}}{e^x + e^{-x}} \right)$$



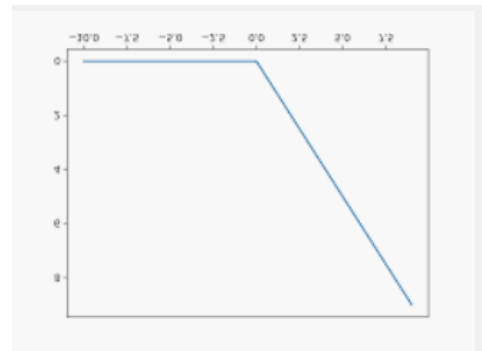
Softmax

$$f(x) = \left( \frac{\exp(x_i)}{\sum_j \exp(x_j)} \right)$$



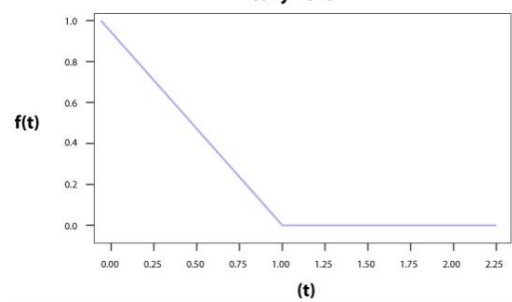
ReLU

$$f(x) = \max(0, x) = \begin{cases} x_i, & \text{if } x_i \geq 0 \\ 0, & \text{if } x_i < 0 \end{cases}$$



LeakyReLU

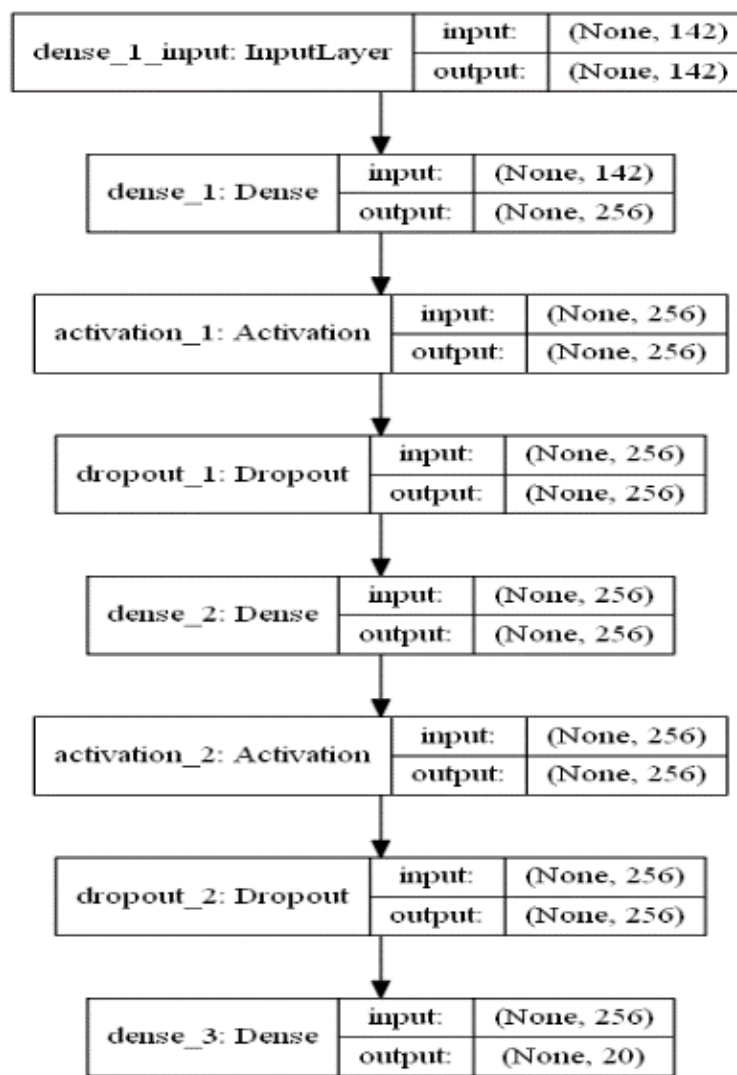
$$f(x) = ax + x = \begin{cases} x, & \text{if } x > 0 \\ ax, & \text{if } x \leq 0 \end{cases}$$



**FFNN.** Experiments were performed with the following parameters:

- One-hot encoding
- Up to 3 hidden layers and neurons of 256,512,1024
- 100 epochs
- 256 batch size.

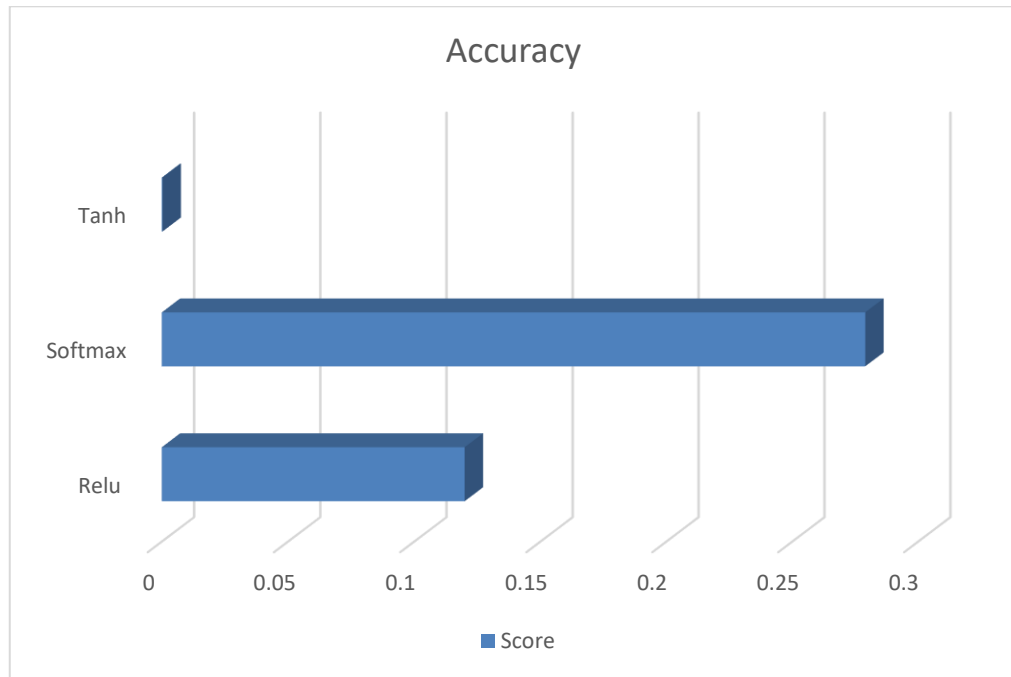
The highest accuracy (equal to 28%) was achieved with the *softmax* activation function (See Fig.24) and the architecture (presented in Fig.23<sup>1</sup>). However, different numbers of neurons and hidden layers didn't show any significant impact on the accuracy. The best determined FFNN architecture is presented in Fig.23<sup>1</sup>.



1

**Fig.23.** Architecture of the best determined model of FFNN

<sup>1</sup> For representing this and further models plot\_model implementation in Keras was used



**Fig.24.** Accuracies with the different activation functions using FFNN

**LSTM :** For the LSTM and BiLSTM experiments the pretrained word embeddings (explained in Section 3) are applied on a top of these models. More than one hidden layer was tested and with only one hidden layer and 64 neurons, the best accuracy of the model was achieved. Model training included several steps such as:

1. Vectorize all words and their respective part-of-speech tags.
2. Loading the pretrained word embeddings
3. Train the model in 100 *epochs* and with the *batch size* equal to 32. *Tanh*, *softmax* and *relu* activation functions are tested for the model and achieve their result as shown in Fig.26.
4. Evaluation of the model: a given sequence of words is tokenized, padded and vectorized with the word embeddings. Using the pre-trained model (in 2), the POS tags are generated (See Fig.25.)

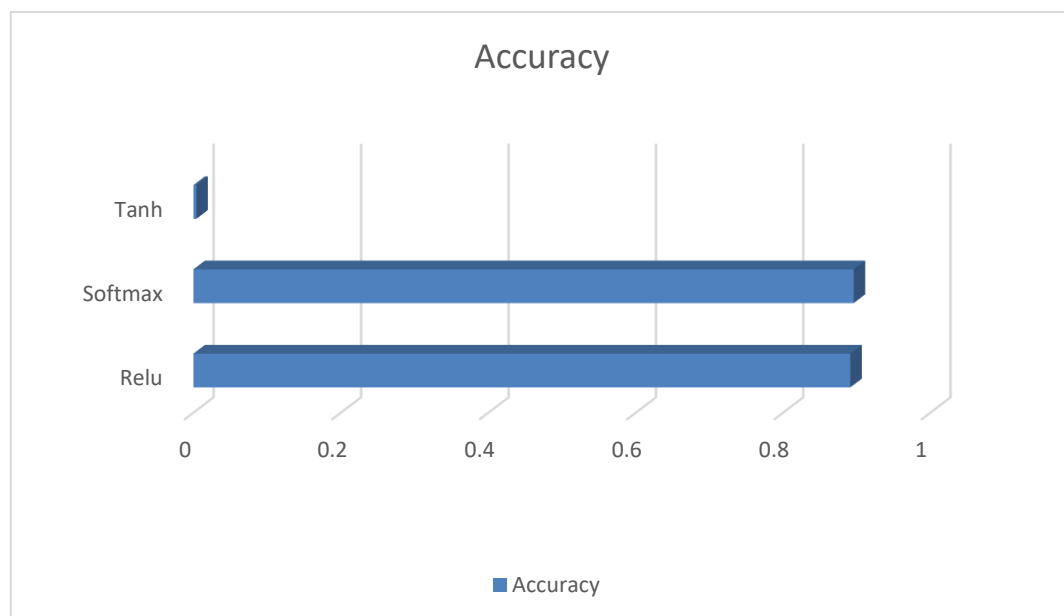
```

The sentence is ['sInowIdenI', 'abI', 'rusIya', 'OIQuba', 'yIHatItI', '::']
The tokenized sentence is [[ 350  521 3284 3444 1432 3710]]
The padded tokenized sentence is [[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1432 3710]]
2019-12-16 11:02:40.414033: I tensorflow/core/platform/profile_utils/cpu_utils.c
c:94] CPU Frequency: 2600000000 Hz
2019-12-16 11:02:40.414886: I tensorflow/compiler/xla/service/service.cc:168] XL
A service 0x55cfe94683c0 executing computations on platform Host. Devices:
2019-12-16 11:02:40.414990: I tensorflow/compiler/xla/service/service.cc:175]
StreamExecutor device (0): Host, Default Version
(1, 100, 21)
100
6
sInowIdenI sInowIdenI V_AUX
abI abI V_AUX
rusIya rusIya V_AUX
OIQuba OIQuba V_AUX
yIHatItI yIHatItI N_V
:: :: V_GER
(base) mif18011@mif18011:~$

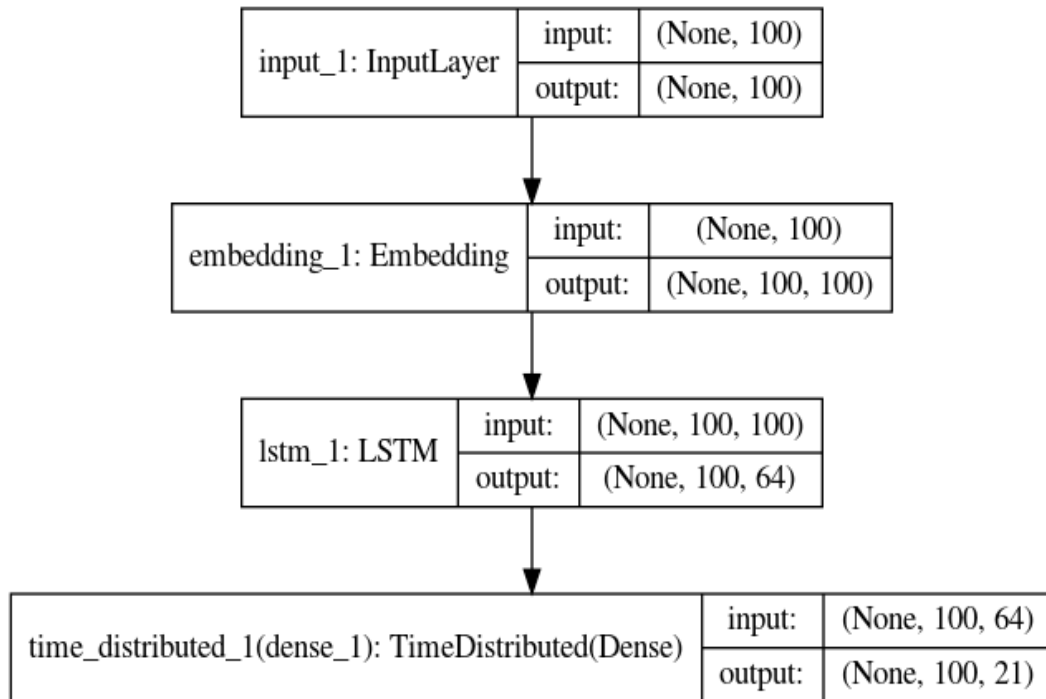
```

**Fig.25.** Evaluation of the LSTM model.

The best accuracy is achieved using the Softmax activation function (See Fig.26) and the architecture of the model is presented in Fig.27.



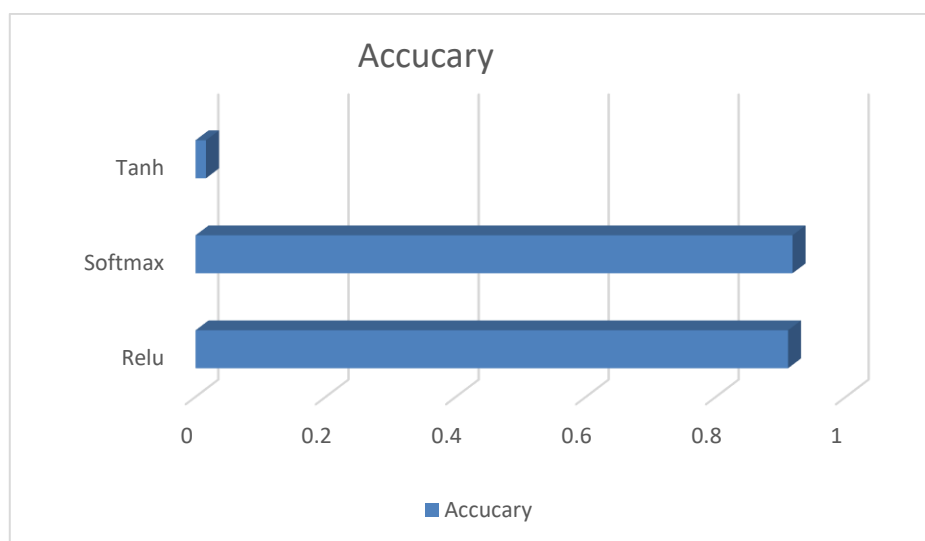
**Fig.26.** Accuracies with different activation functions using LSTM



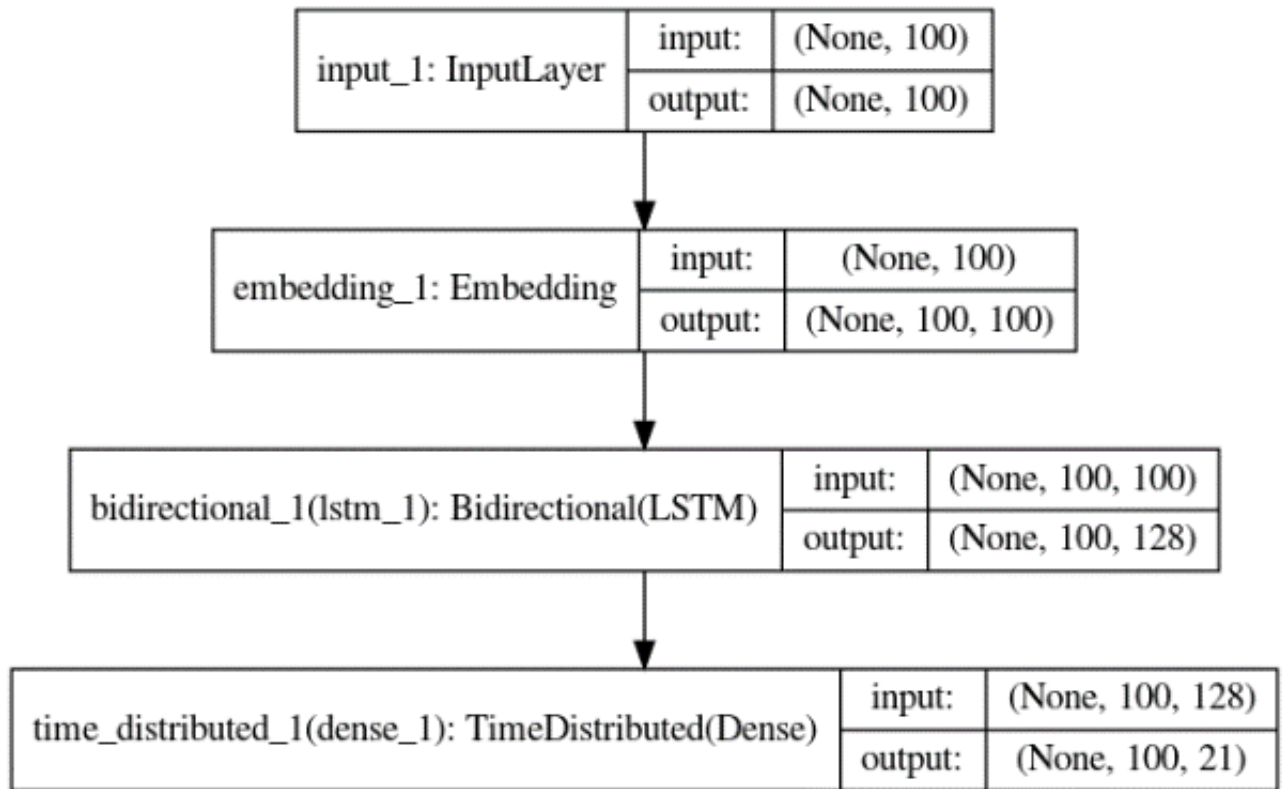
**Fig.27.** Architecture of the best determined model of LSTM

### Bidirectional LSTM

For the parameters see LSTM description. The highest accuracy using the Bidirectional LSTM model is achieved using the Softmax activation function (See Fig.28) and the architecture is presented in Fig.29.



**Fig.28.** Accuracies with different activation functions using BiLSTM



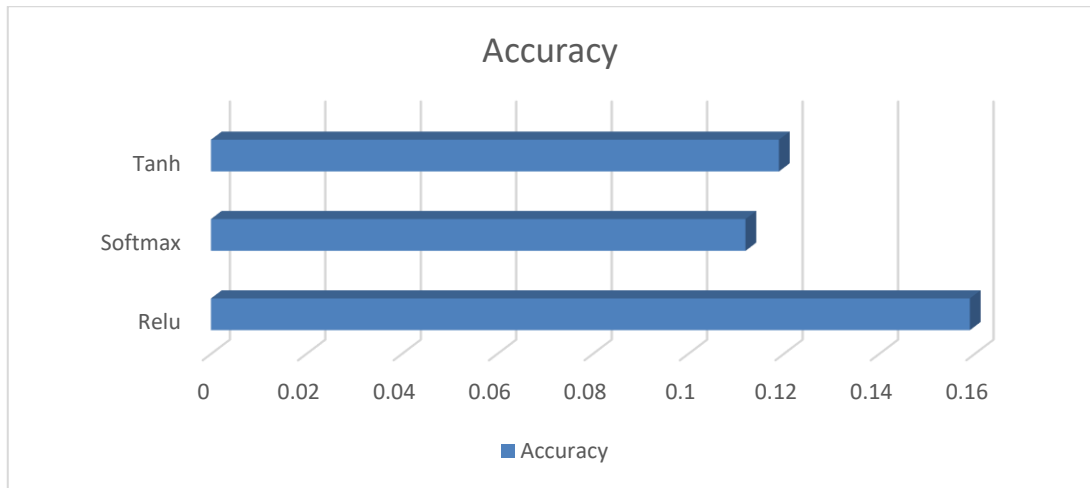
**Fig.29.** Architecture of the best determined model of BiLSTM

**CNN:** The pretrained word vectors were applied on a top of 1D convolution architecture.

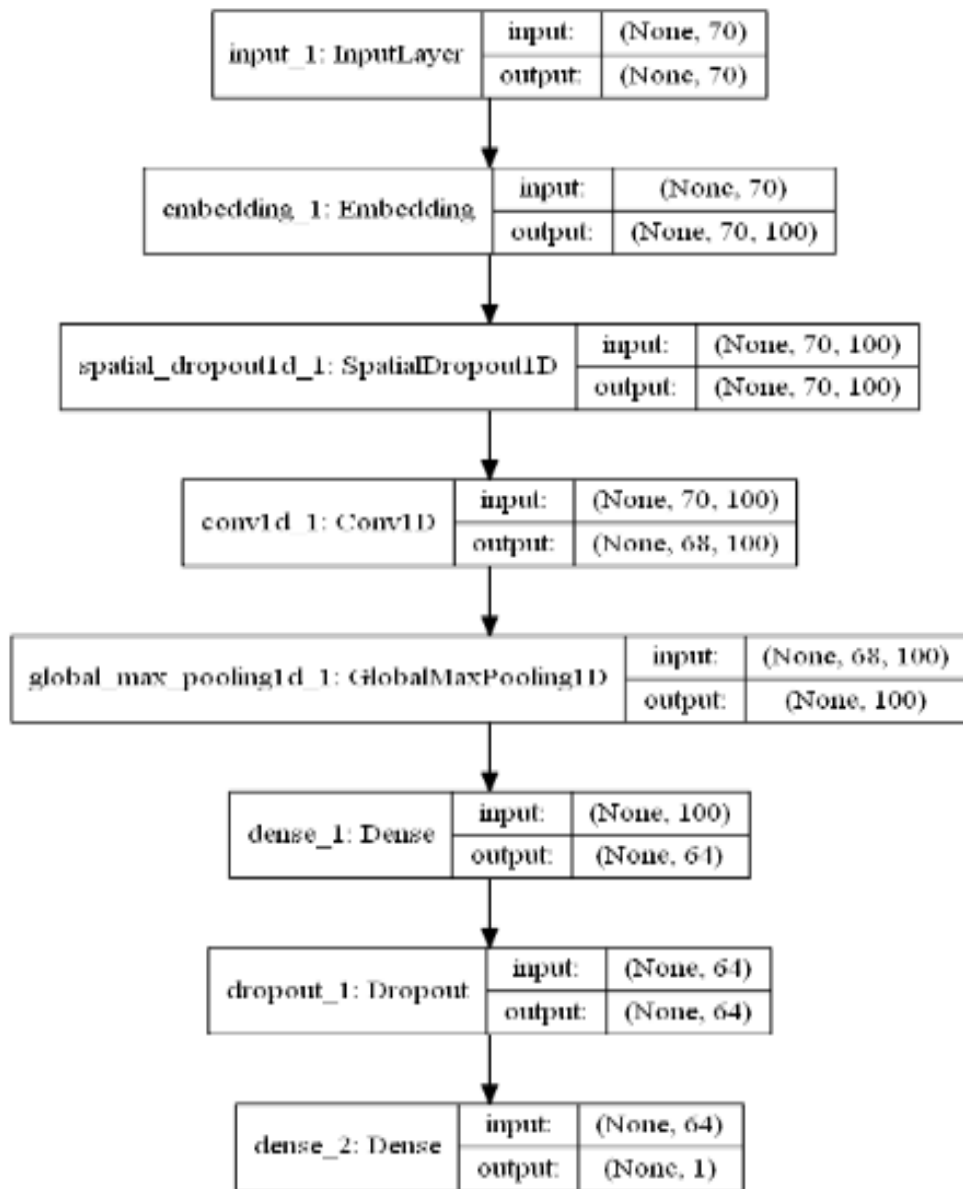
The parameters of the CNN model:

- Dimension : 1D
- Filters : 100
- Kernel size: 3
- Output layers: 2 layers
- Neurons: 64 and 1 neurons

The explored CNN architecture is shown in Fig. 31. Relu activation function gives the best accuracy of the model that is 0.15.(See Fig.30)



**Fig.30.** Accuracies with different activation functions using CNN



**Fig.31.** Architecture of the best determined model of CNN



## 7.2 Automatic DNN Hyperparameter Optimization

Tuning hyper-parameters of DNN models manually is very time consuming. For this reason, the automatic optimization of the Keras model hyper-parameters has been performed using Python's library Hyperas[10]. The discrete and real hyper-parameter values can be tuned automatically searching for best options giving the highest accuracy on the validation dataset. The following options of hyper-parameter values were tested in our experiments:

- *Activation* : sigmoid, softmax, tanh, relu, swish, selu
- *Optimizer* : adam, sgd, rmsprop
- *Batch size* : 16, 32, 64, 128
- *Layers*: up to 3 layers

We have tuned the hyper-parameters for LSTM, BiLSTM and CNN methods. The optimization was performed in 20 iterations tuning hyper-parameters in the directed manner using *tpe.suggest* strategy.

The best determined hyper-parameters and the best obtained results for different DNN classifiers are presented in Table 7.

**Table.7.** Hyperparameter optimization result

Method	Activation	Layers	Neurons	Batch_size	Optimizer	Accuracy
LSTM	Sigmoid	1	32	32	rmsprop	0.89
BiLSTM	Sigmoid	1	64	32	rmsprop	0.91
CNN	Sigmoid, Softmax	1	32	32	adam	0.61

## 8 DISCUSSION

Zooming into the results allow us to make the following statements. The results (especially the best achieved) are much above random and majority baselines, therefore are considered appropriate and reasonable.

After testing the DNN methods (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 solving our task. Even though the best results were achieved when tuning hyper-parameters automatically with the CNN model but still this didn't beat the achievement of the manually tuned hyper-parameter in BiLSTM model. Hyper-parameter optimization didn't make any improvement in LSTM and BiLSTM models, however it helped to get the higher accuracy in CNN model, that is 61%. Since the accuracy of FFNN and CNN is not satisfactory, it can yield a good result by the help of training with large amount of data or more features of the Tigrinya language. working with small amount of data requires rule-based method and with the Nagaoka Tigrinya corpus more feature extraction is needed for better accuracy with other methods.

The overall best accuracy of 91% was achieved with the BiLSTM method, which considered sequences of words in both direction (from-the-past-to-the-future and from-the-future-to-the-past) that is important for the Tigrinya language as in morphologically rich language every words in the sequence has strong connection to each other. Calculations was done to see if differences between those results are statistically significant using McNemar test with the significance level of 95% [47] and the calculated p value is equal  $= 3.83E-33 < 0.05$  which means that these results are statistically significant. In the previous work for Tigrinya POS tagging using the traditional methods of CRFs and SVMs 90% accuracy was achieved by enriching contextual features with morphological and affix features. In this research the DNN BiLSTM outperformed previous work and it is influential and important for the Tigrinya language. In addition, our experiment didn't use the morphological, affix or any other special features that the previous work of POS tagging used. This experiment is also influential and important, because it tests the efficiency of the state-of-the-art DNN methods that are the best in many NLP tasks.

Since the experiment is done with rather small training data, it can be expected that the accuracy of the model will increase when adding more representative and diverse sentences to the corpus.

## 9. CONCLUSION AND FUTURE WORK

The contribution of this research is that the part-of-speech tagging task for the Tigrinya language and the rather small corpus for the first time is solved by using state-of-the-art DNN methods (in particular, Feed Forward Neural Network, Long Short-Term Memory, Bidirectional LSTM and Convolutional Neural Network).

Besides, the Tigrinya neural word embeddings were also trained during this research and it is a novel resource, which has never been created before for this resource-scarce language. Seeking for the best model in this part-of-speech tagging task, different options of hyper-parameter values for different types of DNNs were also investigated. It was done manually and automatically by tuning sets of hyper-parameters. The best result reaching 91% of accuracy was achieved with Bidirectional LSTM (the architecture is presented in Fig. 29) and the *softmax* activation function. The result is considered promising (being much above random and majority baselines); however, to improve the accuracy even further probably more training data is needed. This research is especially significant for the resource-scarce Tigrinya language; however, other similar languages (e.g., Tigre, Saho, Afar) that are resource-scarce could also benefit from this research: the conclusions about the methods (and hyper-parameters) should be similar.

## REFERENCES

- [1] Y. Keleta, K. Yamamoto ir A. Marasinghe, „Tigrinya Part-of-Speech Tagging with Morphological Patterns and the New Nagaoka Tigrinya Corpus,“ *International Journal of Computer Applications*, t. 146, pp. 33-41, 2016.
- [2] O. Omer ir M. Yoshiki, „Stemming Tigrinya Words for Information Retrieval,“ *itrakta COLING*, 2012.
- [3] W. Peilu, Q. Yao, S. Frank K, H. Lei ir Z. Hai, „Part-of-Speech Tagging with Bidirectional Long Short -Term Memory Recurrent Neural Network,“ arXiv :1510.06168 ,[cs.CL], 2015.
- [4] D. Gibbon, „Finite State Morphology of Amharic,“ *itrakta Recent Advances in Natural Language Processing- RANLP*, 2000.
- [5] B. Gebrekidan, „Part of speech tagging for Amharic,“ Research Institute in Information and Language Processing. Wolverhampton University, UK , 2010.
- [6] B. Gambäck, F. Olsson, A. Argaw ir L. Asker, „Methods for Amharic Part-of-Speech Tagging,“ 10.3115/1564508.1564527, 2009.
- [7] Helmut Schmid.(1994). Part-of-Speech tagging with neural networks.*In Proceedings of the 15th Conference on Computational linguistics*, Volume 1 (COLING '94). Association for Computational linguistics, USA 172-176
- [8] Stefan Block. (2009) A Comparison of Three Part-of-Speech taggers
- [9] Keleta, Yemane (2016). Nagaoka Tigrinya Corpus
- [10] "Hyperas : Keras + Hyperopt : A very simple wrapper for convenient hyperparameter optimization". Accessed on: May 26, 2020. [Online]. Available: <http://maxpumperla.com/hyperas/>
- [11] Francois Chollet. Keras: Deep learning library for theano and tensorflow. Accessed on: May 26, 2020. [Online]. Available : <https://keras.io/>
- [12] "Tensorflow". [Online]. Aailable: <https://www.tensorflow.org/>
- [13] Zell, Andreas (1994). Simulation Neuroner Netze [Simulation of Neural Networks] (in German) (1st ed.). Addison-Wesley. p. 73. ISBN 3-89319-554-8.
- [14] Limsopatham, Nut and Nigel Collier. “Bidirectional LSTM for Named Entity Recognition in Twitter Messages.” *NUT@COLING* (2016).

- [15] Kim J. , Understanding how Convolutional Neural Network (CNN) perform text classification with word embeddings, December 2017. Accessed on: May 26,2020. [Online]. Available: <https://joshuakyh.wordpress.com/2017/12/02/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-embeddings/>
- [16] Machine Learning Mastery, A Gentle Introduction to the Rectified Linear Unit (ReLU), January 9, 2019. [Online]. Available : <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/>
- [17] Dustin E. Stansbury, A Gentle Introduction to Artificial Neural Networks. Accessed on: May 26, 2020. [Online]. Available: <https://theclevermachine.wordpress.com/>
- [18] Chris, How does the Softmax activation function work?, Accessed on: May 26, 2020. [Online]. Available : <https://www.machinecurve.com/index.php/2020/01/08/how-does-the-softmax-activation-function-work/>
- [19] Radim \v Reh\r u\v rek, et al. "Software Framework for Topic Modelling with Large Corpora." *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA,
- [20] Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput.*1997;9(8):1735-1780. doi: 10.1162/neco.1997.9.8.1735
- [21] Varior R.R., Haloi M., Wang G. (2016) Gated Siamese Convolutional Neural Network Architecture for Human Re-identification. In: Leibe B., Matas J., Sebe N., Welling M. (eds) *Computer Vision – ECCV 2016*. ECCV 2016. Lecture Notes in Computer Science, vol 9912. Springer, Cham
- [22] Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space. arXiv preprint [arXiv:1301.3781v3](https://arxiv.org/abs/1301.3781) [cs.CL]
- [23] Zhang Y, Wallace B. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:151003820. 2015; PMID: 463165
- [24] K. N. Björkenstam, Computational linguistics, [accessed 05-May-2018], Stockholm University, 2018. [Online]. Available: [https://nordiskateckensprak.files.wordpress.com/2014/01/knb\\\_whatisacorpus\\\_cph-2013\\\_outline.pdf](https://nordiskateckensprak.files.wordpress.com/2014/01/knb\_whatisacorpus\_cph-2013\_outline.pdf).
- [25] Sketch Engine. Accessed on May 26, [Online]. Available: <https://www.sketchengine.eu/>
- [26] Yoav Goldberg. 2017. *Neural Network Methods for Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers
- [27] MARCUS, M.P., MARCINKIEWICZ, M.A. and SANTORININ, B(1993), "Building a large annotated corpus of English: the Penn Treebank", *Comput. Linguist.* 19(2), 313-330.

- [28] JURAFSKY, D., MARTIN, J. and KEHLER, A. (2000), Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, MIT Press.
- [29] Expert System, What is Machine Learning? A definition, May 6, 2020. Accessed on: May 26, 2020. [Online]. Available: <https://expertsystem.com/machine-learning-definition/>
- [30] Gasser, Michael. (2011). HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya.
- [31] Teklay Gebregzabiher, "Part of Speech Tagger for Tigrigna Language," Unpublished Master's Thesis, Department of Computer Science, Addis Ababa University, 2010.
- [32] DelSole M, What is One Hot Encoding and How to Do It, April 25, 2018. Accessed on: May 26, 2020.[Online]. Available: <https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179>
- [33] Rong X, Word2Vec Parameter Learning Explained. arXiv preprint [arXiv:1411.2738v4](https://arxiv.org/abs/1411.2738v4) [cs.CL]
- [34] Oliinyk Halyna, Word embeddings: exploration, explanation, and exploitation (with code in Python), Dec 3, 2017. Accessed on May 26, 2020.[Online]. Available: <https://towardsdatascience.com/word-embeddings-exploration-explanation-and-exploitation-with-code-in-python-5dac99d5d795>
- [35] Brownlee J, What is Deep Learning?, August 16, 2019. Accessed on May 26, 2020.[Online]. Available: <https://machinelearningmastery.com/what-is-deep-learning/>
- [36] Hernandez C, Deep Learning Neural Networks for Self-Driving Cars(2017).
- [37] Pascanu R, Mikolov T, Bengio Y, On the difficulty of training Recurrent Nerual Networks, [arXiv:1211.5063v2](https://arxiv.org/abs/1211.5063v2) [cs.LG]
- [38] Britz D, Recurrent Neural Networks Tutorial, Part 1 - Introduction to RNNs, September 17, 2015. Accessed on May 26, 2020.[Online]. Available: <http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/>
- [39] Colah's blog, Understanding LSTM Networks, August 27, 2015. Accessed on May 26,2020.[Online]. Available:<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

- [40] GETACHEW, M. (2001) Automatic part of speech tagging for Amharic: An experiment using stochastic hidden markov(hmm) approach, Master's thesis, Addis Ababa University
- [41] Mequanent, Argaw. " Amharic Part-of-Speech Tagger using Neural Word Embeddings as Features."(2019).
- [42] Omniglot, Accessed on May 26,2020.[Online]. Available: <https://omniglot.com/writing/ethiopic.htm>
- [43] Tomas Mikolov, Wen-tau Yih and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representation. In Proceedings of NAACL HLT, 2013
- [44] Wikipedia contributors,. "Word embedding" Wikipedia, The Free Encyclopedia., 18 Apr. 2020. Web. 26 May 2020.
- [45] Kumar M, Deshpande A., One hot encoding, Accessed on May 26, 2020.[Online]. Available:<https://www.oreilly.com/library/view/artificial-intelligence-for/9781788472173/5fe6418e-c487-4aea-a940-57ef3c628290.xhtml>
- [46] Medium, Accessed on May 26,2020.[Online]. Available: [https://cdn-images-1.medium.com/max/1600/1\\*uAeANQIOQPqWZnnuH-VEyw.jpeg](https://cdn-images-1.medium.com/max/1600/1*uAeANQIOQPqWZnnuH-VEyw.jpeg)
- [47] McNemar test: McNemar, Quinn (June 18, 1947). "Note on the sampling error of the difference between correlated proportions or percentages". *Psychometrika*. 12 (2): 153–157. doi:10.1007/BF02295996. PMID 20254758.

# APPENDIX

## 1. RESEARCH PROJECT No.1

Author	Senait Gebremichael Tesfagergish
Title	Machine learning for Part-of-Speech Tagging in Tigrinya Language
Supervisor	Jurgita Kapociute Dzikiene
Presented at	Vytautas Magnus University, Faculty of Informatics, Kaunas 29/01/2019
Number of pages	13

Natural Language is the central role of communication in human beings. Natural Language Processing (NLP) is the branch of artificial intelligence that enables machines to understand and process human language. NLP products has turned to very popular tools and it has a vast applications in our daily activities including language translation, grammar correction, spam filtering. Noteworthy NLP applications have been implemented for resourceful languages such as English. However many languages with scarce sources are not beneficiary of this application..Today we have around 6500 spoken languages globally. However about 2000 of this languages have fewer than 1000 speakers. Thus Google translate supports about 100 languages so far. Tigriyna is one of the languages with very limited language resources. It is a morphologically rich Semitic language spoken by over seven million people in Eritrea and northern Ethiopia.

The main objective of my research project is to know more about Natural Language Processing techniques and algorithms.as well as implementing those NLP application in my native language, Tigrigna. There is very little NLP study done for Tigrinya language, I want to do my research on this area and contribute to the development of NLP resources for Tigrinya.

The first phase of the research is dedicated for two parts. First part is literature review and understand the current status of NLP studies in Tigrinya. The second part is collecting sample data and tagging them for farther analysis. The data consists of 100 sentences tagged with eight different parts of speeches namely: Noun, Pronoun, Verb, Adjective, Adverb,Preposition, Conjunction, Numeral,Determiner and Punctuation).



## 2. RESEARCH PROJECT No.2

Author	Senait Gebremichael Tesfagergish
Title	Machine learning for Part-of-Speech Tagging in Tigrinya Language
Supervisor	Jurgita Kapociute Dzikiene
Presented at	Vytautas Magnus University, Faculty of Informatics, Kaunas 05/09/2019
Number of pages	15

In this paper, I propose a deep learning special neural network architecture for Part of Speech tagger in Tigriyna language. Part of speech tagging is not a new concept in the field of Natural language processing and has been studied using the traditional probabilistic methods like Hidden Markov Models (HMM) and Conditional Random Fields (CRF). Nowadays neural network models are implemented in different areas and yield a great accuracy and applications. Recently Recurrent Neural Networks (RNN), long short-term memory (LSTM), Convolutional Neural Network (CNN) are developed for part of speech tagger and obtain a good result in many languages.

The second research project is dedicated for two parts. First part is literature review for the deep learning neural network for NLP. And the second part is the data preparation and experiments done for this research project.

### 3. RESEARCH PROJECT No.3

Author	Senait Gebremichael Tesfagergish
Title	Deep Learning for the Part-of-speech tagging for Tigrinya language
Supervisor	Jurgita Kapociute Dzikiene
Presented at	Vytautas Magnus University, Faculty of Informatics, Kaunas 19/12/2019
Number of pages	15

Deep learning neural network architecture has been shown a great efficiency in the classification of texts for many languages. Tigrinya part-of-speech tagging is in its early age and not many experiments are done with in this field. The only experiment done is by using the traditional probabilistic method as SVM and CRF's. The main objective of this research project is to offer part-of-speech tagger for the Tigrinya language trained using deep learning approaches on the small corpora and few morphological features.

This research is composed of the three parts. The first part presents data preparation and method analysis for better understanding the solving task. The second part describes and explains the deep learning approaches. The third part is for experimental setup explained in detailed, the results and the conclusion. The report is organized as follows:- 1) it will introduce about the used corpora and its classes distribution. 2) it will explain the concept of the word embeddings. 3) it will then go into detail about the experiments with different methods for training the model and obtained results. In the conclusion the most accurate method will be recommended for the solving task.

## 4. DEEP LEARNING-BASED PART-OF-SPEECH TAGGING OF THE ETHIOPIC-LANGUAGE

Authors	Senait Gebremichael Tesfagergish and Jurgita Kapociute Dzikiene
Title	Deep learning-based part-of-speech tagging of the Ethiopic-language
Submitted	International conference on information and software technologies(ICIST 2020) Kaunas, Lithuania 01/05/2020
Accepted	ICIST 2020( International Conference on Information and Software Technologies) 10/06/2020
Number of pages	12

**Abstract.** Deep Neural Networks have demonstrated the great efficiency in many NLP task for various languages. Unfortunately, some resource-scarce languages as, e.g., Tigrinya still receive too little attention, therefore many NLP applications as part-of-speech tagging still in their early stages. Consequently, the main objective of this research is to offer the effective part-of-speech tagging solutions for the Tigrinya language having rather small training corpus.

In this paper the Deep Neural Network classifiers,(i.e., Feed Forward Neural Network, Long Short Term Memory, Bidirectional LSTM and Convolutional Neural Network) are investigated by applying them on a top of separately trained distributional neural word2vec embeddings. Seeking for the most accurate solutions, DNN models are optimized manually and automatically. Despite automatic hyper-parameter optimization demonstrates a good performance with the Convolutional Neural Network, the manually tested Bidirectional Long Short – Term Memory method achieves the highest overall accuracy equal to 91 %.

**Keywords:** Deep Neural Networks, -; FFNN, CNN, LSTM, BiLSTM methods; word2vec embeddings; Nagaoka Corpus, Tigrinya part-of-speech tagging

### 1 Introduction

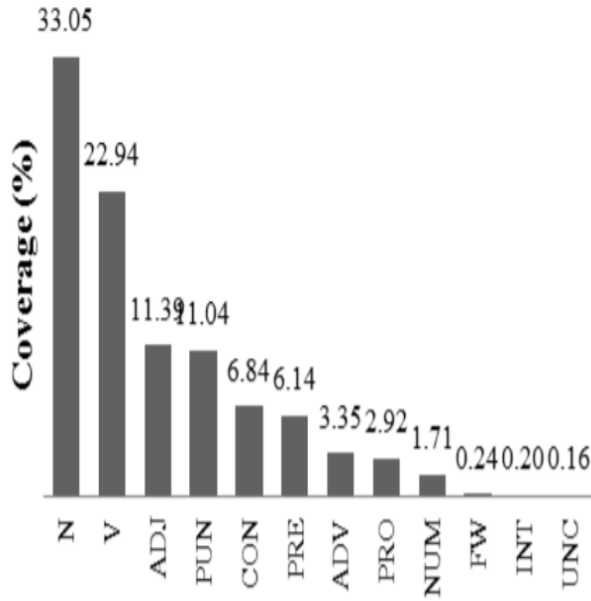
Part-of-speech(POS) tagging is a well-known task in NLP: it represents a process of mapping words in sentences into their corresponding parts-of-speech, based on their context and the meaning. POS tagging is a prerequisite for many NLP tasks such as dependency parsing, machine translation, speech recognition and so many others. To train the accurate POS tagger, a large annotated corpus is required. Unfortunately, for many resource-scarce languages such annotated corpora do not exist. Unsupervised solutions still underperform supervised, therefore, to expect better results the annotated data is crucial. Besides, there is no one right solution that could work for each language: each language is different and difficult in its own way, therefore requires adaptation.[1].

Tigrinya belongs to the Semitic language branch of the Afro-asiatic family, along with Hebrew, Amharic, Maltese, Tigre and Arabic. Semitic languages are characterized with the rich derivational and influential morphology which results in the numerous variations of word forms. The distinguishing feature of Semitic languages lies in the ‘root-template’ morphological pattern that is often composed of trilateral roots. The verb roots in Semitic languages comprise a sequence of consonants, whereas the templates are the patterns of vowels that are intercalated in between these consonants, forming various stems.

## 2 The Corpora

42	</S>
43	<S n="2">
44	<W type="PRE">qIdImi</W>
45	<W type="ADJ">bIzuHI</W>
46	<W type="N">Oametati</W>
47	<C type="PUN">"</C>
48	<W type="ADJ">aImIroawi</W>
49	<W type="N">sInIkIIna</W>
50	<W type="N">bIganEnI</W>
51	<W type="CON">weyI</W>
52	<W type="ADJ">IkeyI</W>
53	<W type="N">menafisIti</W>
54	<W type="V_AUX">iyu</W>
55	<W type="V_REL">ziMexII</W>
56	<C type="PUN">"</C>
57	<W type="V_REL">ziBili</W>
58	<W type="V_REL">giGuyI</W>
59	<W type="N">ameleKaKaIta</W>
60	<W type="V_GER">neyIru</W>
61	<C type="PUN">":</C>
62	</S>
63	<S n="3">

**Fig.1.** The snippet from the Nagaoka corpus used for the Tigrinya POS tagging[10]



**Fig. 2.** A distribution of POS tags(classes) for the Tigrinya language in the Nagaoka corpus

The corpus was split into training, validation and testing. The statistics can be found in Table 1.

**Table 1.** Training, testing and validation splits of the dataset used in Tigrinya POS tagging

	Percentage points	Number of tokens	Number of sentences
Training	60 %	43248	2792
Validation	20 %	14416	931
Testing	20 %	14416	933

Tigrinya words possess rich morphological information embedded in the form of prefixes, infixes and suffixes [4]. Thus, the Tigrinya language specifics makes us to consider the following features:

- If a word is the first word in a sentence
- If a word is the last word in a sentence
- POS tags of 2 words before a target word
- A POS tag of 1 word before a target word
- POS tags of 2 words after a target word
- A POS tag of 1 word after a target word

### 3 Vectorization

DNNs can be applied on a top of the vectorized words. For this reason, we have selected the novel distributional word embeddings approach, where each word is represented with a real values vector and vectors of the semantically similar words are projected closer in a vector space.

In our experiments we have used word2Vec approach, which is an embedding model trained with the neural network. Before training, the text corpus was split into windows containing a focus word and its context: this information was used as the input and the output. Since similar focus words appear in the similar context; their word2vec vectors are closer in the vector space. Given enough training data, the semantic meaning of each word is learned correctly.

Since pre-trained publicly available word embeddings do not exist for Tigrinya language, we have trained them ourselves from 4656 sentences of the Nagaoka corpus. Our word2vec embeddings were trained- with 100 dimensions. We have set the context window size equal to 3, which means that a context of 3 words before and 3 words after a target word were considered. All the rest parameters were set to their default values. For training the word embeddings we have used python programming language with `gensim`([reference](#)). The pre-trained word embeddings were saved and afterwards used in all our experiments.

## 4 DNN Classifiers for Tigrinya POS tagging

### 4.1 Feed Forward Neural Network [14]

Deep feedforward networks also often called Feed Forward Neural Networks(FFNN), or multilayer perceptron (MLPs) are the quintessential deep learning models. The goal of a feedforward network is to approximate some function  $f^*$ .

In FFNN there are no feedback connections in which an output of a model is fed back into itself. This DNN-type is the simplest network of all the DNN types. For the POS tagging problem, FFNN plays a role of a classifier; besides, due to its simplicity FFNN can perform faster compared to the other types of DNNs. FNN input is vectorized words passed to the deeper hidden layers. The function sums all the weights associated with every input and maps it to 20 POS tags class. Weights in FFNN are adjusted by backward propagation.: giving the output it modifies all associated the weights to reduce the overall error. Despite FFNN is fast and simple, it has limitations working with a smaller amounts of training data. FFNN does not have solutions to work with the sequential data (i.e., words in sentences), that is especially important in POS tagging; therefore, the sequential information (a context of the target word) to the FFNN was inputted indirectly: in a form of 2 succeeding and 2 preceding words around a target word. Despite contextual information was fed into FFNN in such a fake way, we do not expect the method to demonstrate very good results in the POS tagging task. However, the FNN method is still selected as the baseline approach to see how far the accuracy can go with such naïve measures.

### 4.2 Recurrent Neural Network (LSTM and BiLSTM)

However, working with the sequential data (as the text) a word order cannot be ignored. Especially a word order is informative for the Tigrinya language, because changing it in a sentence, the meaning of a sentence also changes.

For this reason, Recurrent Neural Networks (RNNs) (see a schema in Figure 4) should be a good choice for our POS tagging task. RNNs are methods having memory units and therefore adjusted to cope with the sequential data. An input of the current time step is an input for the next time step, thus it has two inputs of which the first one is an actual input (i.e., incoming word from a sentence) and the second one is an output of the previous time steps (i.e., generalized information about previous words). Despite RNNs have the memory, they suffer from the vanishing gradient problem. RNNs remember only the recent information and therefore are accurate working with the short-term dependencies. However, for the Tigrinya language longer dependencies doesn't have more importance for the POS tagging.

For the longer sequence's LSTMs and BiLSTMs are used instead, because they are refined to remember longer sequences.

LSTM memory cell has 3 weighted gates adjusted during training the input, forget and output gates are used to memorize what information to input, forget and output, respectively

LSTM considers information going in only one direction: i.e., from the beginning to the future. However, Bidirectional LSTM considers sequences going in both directions: from-the-past-to-the-future and from-the-future-to-the-past. In the POS tagging task, some later words may give important details about the morphological form of the current word and it is especially evident with the Tigrinya language verbs carrying a lot of information about nouns and pronouns, mostly appearing at the end of sentences.

### 4.3 Convolutional Neural Network (CNN)

Despite RNNs (LSTMs, BiLSTM) are adjusted to cope with the text, sometimes they underperform the other conventional machine learning approaches. Recurrent methods consider the whole context including unrelated to the target word. For this reason, CNNs are considered as a good solution. Initially these methods were used for the image data, however, recently they are successfully adjusted for the text. Instead of sequences of words, CNNs consider only patterns of sequential words (called as n-grams) and a width  $n$  of some filter determines how long patterns are worth to take into account.

The filters width usually varies in between 2-5 words. Filter values are adjusted during training (when training samples with input/output values are fed into the neural network). The calculated cost function helps to adjust the filter weight values during the backpropagation. Filter values converge to values of which the cost function is minimized.

CNNs could be a good option for our solving task, because the Tigrinya language not all the information in a sentence matters, but only some context (in terms of n-grams) close to the target word.

## 5 Experiments and Results

In this research we have experimented with the dataset (described in Section 2), using vectorization (described in Section 3) and the DNN methods (presented in Section 4). For the method implementations we have used python programming language with TensorFlow [12] and Keras[13].

As the evaluation metrics we have used the accuracy and the loss presented in eq. 1 and eq. 2, respective.

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

tp (true positives) represent + instances with, determined class +;

fn (false negatives) represent + instances, with determined class -;

fp (false positives) represent - instances, with determined class +;

tn (true negatives) represent - instances, with determined class -.

For the POS tagging results to be considered reasonable and appropriate the accuracy (in eq. 1) must be above calculated random (see eq. 3) and majority (eq. 2) baselines.

$$\begin{aligned} \text{Random baseline} &= \sum P(c_i)^2 \\ &= 0.127 \end{aligned} \quad (3)$$

$$\begin{aligned} \text{Majority baseline} &= \max P(c_i), \text{ where} \\ &= 0.27 \end{aligned} \quad (4)$$

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

Activation functions determine the output of DNNs deciding whether it should be activated or not based on each neurons' input and relevancy in prediction. In this paper we have explored 3 types of activation functions (relu [17], softmax [19] and tanh [18]) to determine the best. Each of activation functions have their own advantages, e.g., with *relu* a network converges very quickly, *softmax* effectively handles multiple classes, *tanh* is suitable to model inputs that have strong values.

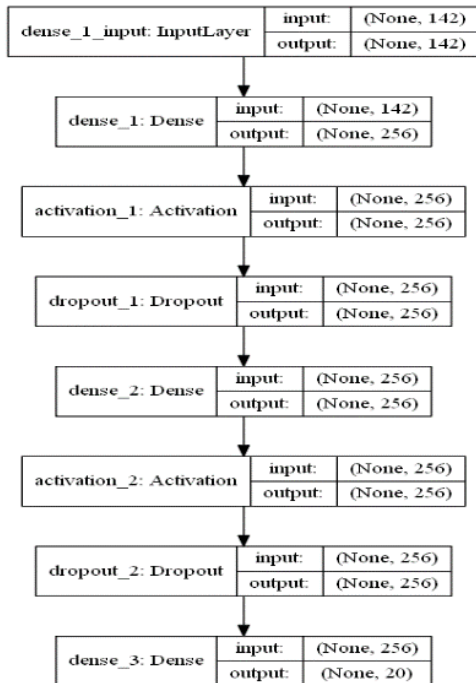
Manually selected DNN architectures and parameters

### 5.1 Manually tuned DNN architectures and hyper-parameters

**FFNN.** Experiments were performed with the following parameters:

- One-hot encoding
- Up to 3 hidden layers and neurons of 256,512,1024
- 100 epochs
- 256 batch size.

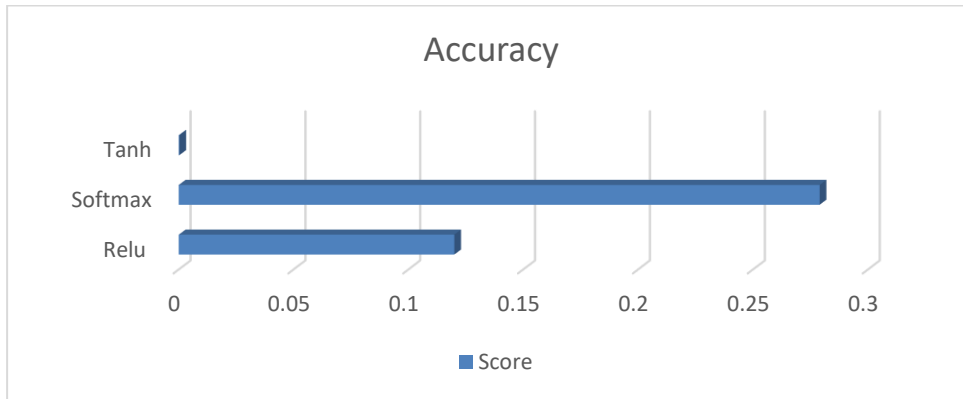
The highest accuracy (equal to 28% ) was achieved with the softmax activation function (see Figure 7) and the architecture presented in Figure 6<sup>2</sup>. Different numbers of neurons, hidden layers didn't show any significant impact on the accuracy.



**Fig.6.** Architecture of the best determined model of FFNN

<sup>2</sup> For representing this and further models plot\_model implementation in Keras was used.

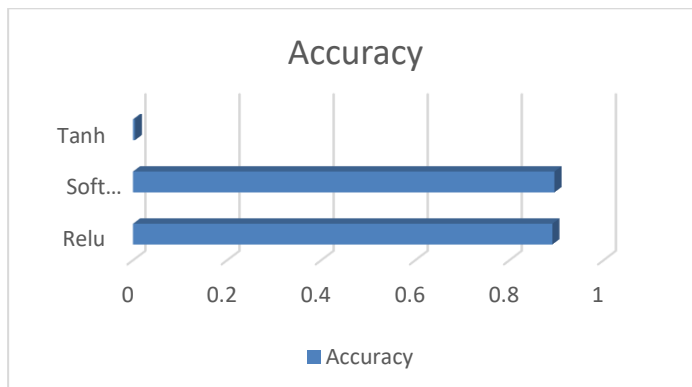




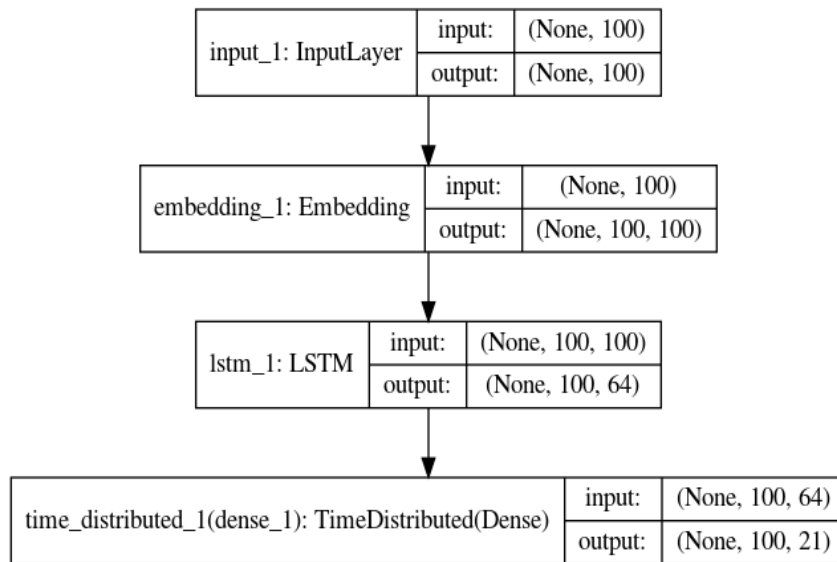
**Fig.7.** Accuracies with different activation functions using FFNN

**LSTM :** For the LSTM and BiLSTM experiment the pretrained word embeddings (explained in Section 3) is loaded into the model. More than one hidden layer is tested and with only one hidden layer and 64 neurons, the best accuracy of the model is achieved. Training of the model possessed several steps such as:

1. Index all the words and its respective part-of-speech tags.
2. Loading the pretrained word embeddings
3. Train the model using vectors from 2. 100 epochs and Batch\_size 32. Tanh, Softmax and Relu activation functions are tested for the model and achieve their results as shown in Fig 8.

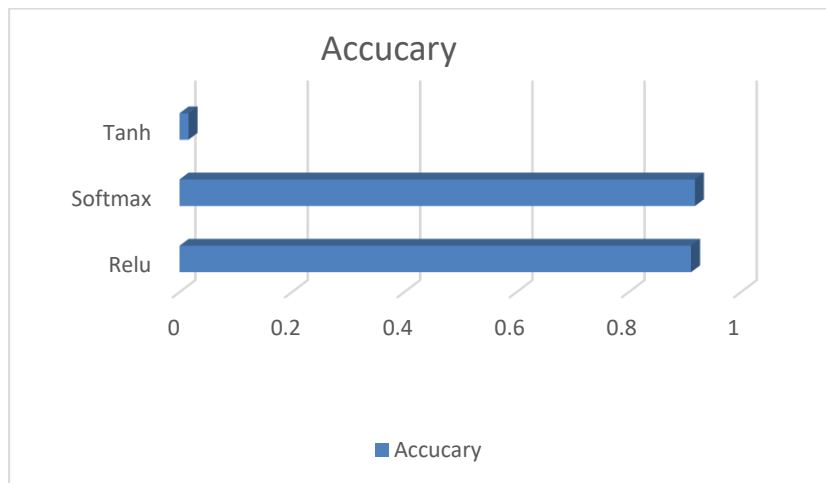


**Fig.8.** Accuracies with different activation functions using LSTM

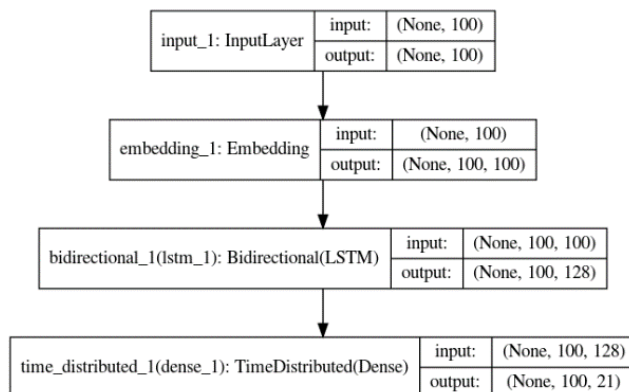


**Fig.9.** Architecture of the best determined model of LSTM

## Bidirectional LSTM



**Fig.10.** Accuracies with different activation functions using BiLSTM

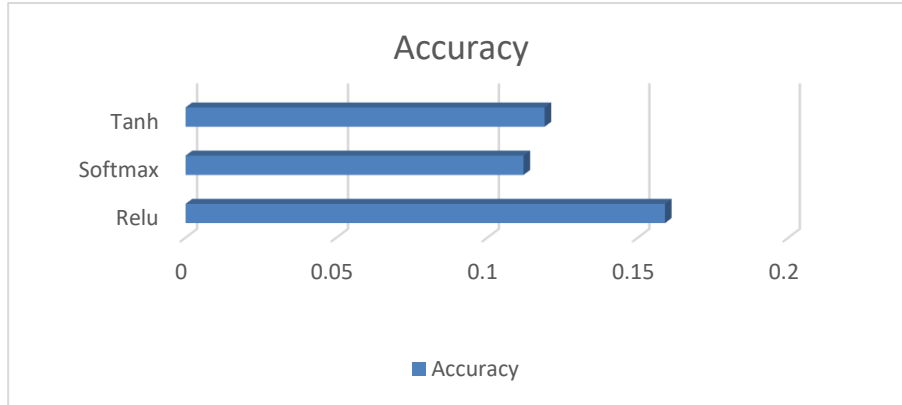


**Fig.11.** Architecture of the best determined model of BiLSTM

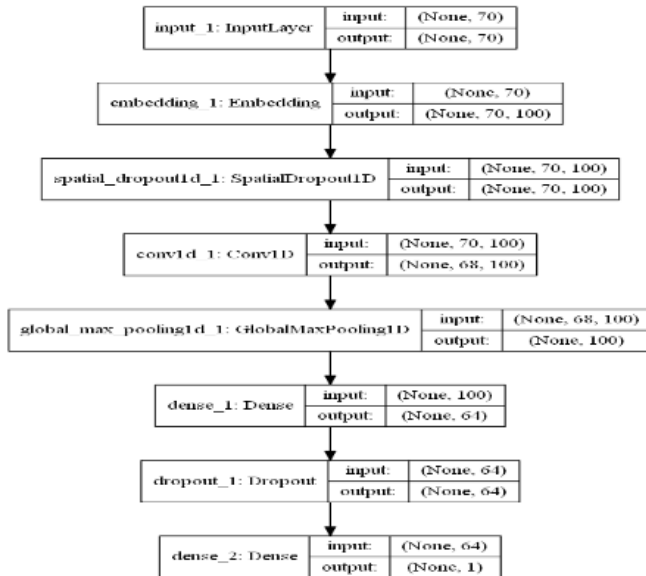
**CNN:** The pretrained word vectors are used for the classification task using convolutional neural network. Data organize as the labels and texts. Where the labels indicate the classes or part-of-speech tags. As the texts are the target word with its two succeeding and preceding words.

The convolutional layer has 1 dimension, 100 filters and 3 kernel size.

Output layers: 2 layers with 64 and 1 neurons respectively.



**Fig.12.** Accuracies with different activation functions using CNN



**Fig.13 .** Architecture of the best determined model of CNN

## 5.2 Automatic DNN Hyperparameter Optimization

Tuning hyper-parameters of deep learning models manually is time consuming. For this reason, we have performed an automatic optimization of the hyper-parameters with the Keras model by using python's library Hyperas[11]. The discrete and real hyper-parameter values can be tuned automatically searching for the best their options giving the

highest accuracy on the validation dataset. The following options of hyper-parameter values were tested in our experiments:

- *Activation*: sigmoid, softmax, tanh, relu, swish, selu
- *Optimizer*: adam, sgd, rmsprop
- *Batch size*: 32, 64, 128
- *Neurons*: 16, 32, 64, 128
- *Layers*: up to 3 layers

In this paper Hyperparameter Optimization is also tested with the Long-Short term memory (LSTM) , Bidirectional LSTM (BiLSTM) and CNN. The parameters tested are

*Activation* : sigmoid, softmax, tanh , relu, swish, selu

*Optimizer* : adam, sgd, rmsprop

*Batch size*: 32, 64, 128

*Neurons* : 16, 32, 64, 128

*Layers* : up to 3 layers

We have tuned the hyper-parameters for LSTM, BiLSTM and CNN methods. The optimization was performed in 20 iterations tuning hyper-parameters in the directed manner using *tpe.suggest* strategy.

The best obtained results are presented in Table 2.

**Table 2.** Hyperparameter optimization result

Method	Activation	Layers	Neurons	Batch_size	Optimizer	Accuracy
LSTM	Sigmoid	1	32	32	rmsprop	0.89
BiLSTM	Sigmoid	1	64	32	rmsprop	0.91
CNN	Sigmoid, Softmax	1	32	32	adam	0.61

## 6 Discussion

Zooming into the results allow us to make the following statements.

The results (especially the best achieved) are much above random and majority baselines, therefore are considered appropriate and reasonable.

Using the limited amount of data the DNN methods, FFNN, LSTM, BiLSTM and CNN are tested. The manual tuning of hyper-parameters and different DNNs revealed that for our solving task the best option is the BiLSTM. When tuning hyper-parameters automatically, the best results were achieved with the CNN model, yet didn't beat the achievement of the manually tuned hyper-parameter in BiLSTM model.

Hyper-parameter optimization didn't make any improvement in LSTM and BiLSTM models, however, it gets higher accuracy in CNN more than the manually tested CNN model achieves 61% accuracy. Since the accuracy of Feed forward Neural network and CNN is not satisfactory yet it can yield a good result by the help of training in large amount of data or more features of the Tigrinya language. working with small amount of data requires rule-based method and with the Nagaoka Tigrinya corpus more feature extraction is needed for better accuracy with the other methods

The overall best accuracy reaching 91% was achieved with the BiLSTM method, which consider sequences of words in both direction (from-the-past-to-the-future and from-the-future-to-the-past) that is important for the

Tigrinya language as morphologically rich language every words in the sequence has a strong connection to each other. . In the previous work for Tigrinya POS tagging using the traditional methods of CRFs and SVMs 90% accuracy was achieved by enriching contextual features with morphological and affix features. In this paper the DNN BiLSTM outperformed previous work and it is an influential and important for the Tigrinya language. In addition, our experiment didn't use the morphological, affix or any other special features that the previous work of POS tagging used. Our experiment is still influential and important, because it tests the effectiveness of the state-of-the-art DNN methods that are the best in many NLP tasks.

Since the experiment is done with rather small training data, we anticipate that the accuracy of the model will increase adding more representative and diverse sentences to the corpus.

## 7 Conclusion

In this paper we are solving the part-of-speech tagging problem for the morphologically complex Tigrinya language. The contribution of this research is that we test state-of-the-art DNN methods (in particular, Feed Forward Neural Network, Long Short- Term Memory, Bidirectional LSTM and Convolutional Neural Network) for the Tigrinya language in the part-of-speech tagging tasks for the first time using the rather small corpus. Besides, we train and use Tigrinya word embeddings, which is a novel resource, that is not publicly available for the resource-scarce Tigrinya language. Seeking for the best model for our solving task, we also test different options of hyper-parameter values for different types of DNNs. It is done manually and automatically tuning sets of hyper-parameters. The best result reaching 91% of accuracy is achieved with Bidirectional LSTM and the softmax activation function. The result is considered promising (because much above random and majority baselines); however, to improve the accuracy even more we probably need more training data. This research is important for the Tigrinya language, however, other similar languages (e.g., Tigre ) that are resource-scarce could also benefit from this research: we expect that the conclusions about the methods (and hyper-parameters) would be similar.

## References

1. Keleta, Yemane & Yamamoto, Kazuhide & Marasinghe, Ashuboda. (2016). Tigrinya Part-of-Speech Tagging with Morphological Patterns and the New Nagaoka Tigrinya Corpus. *International Journal of Computer Applications*. 146. 33-41. 10.5120/ijca2016910943.
2. Osman, Omer & Mikami, Yoshiki. (2012). Stemming Tigrinya Words for Information Retrieval. 345-352.
3. Wang, Peilu & Qian, Yao & Soong, Frank & He, Lei & Zhao, Hai. (2015). Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network.
4. Gibbon, Dafydd. "Finite State Morphology of Amharic." *Recent Advances in Natural Language Processing - RANLP*, 2000.
5. Kapočiūtė-Dzikiene, Jurgita & Rimkutė, Erika & Boizou, Loic. (2017). A Comparison of Lithuanian Morphological Analyzers. 47-56. 10.1007/978-3-319-64206-2\_6.
6. Binyam. "Gebrekidan Gebre Part of Speech Tagging for Amharic." (2010).
7. Gambäck, Björn & Olsson, Fredrik & Argaw, Atelach & Asker, Lars. (2009). Methods for Amharic Part-of-Speech Tagging. 10.3115/1564508.1564527.
8. Helmut Schmid.(1994). Part-of-speech tagging with neural networks. In *Proceedings of the 15<sup>th</sup> conference on Computational linguistics, -Volume 1 (COLING '94)*. Association for Computational linguistics, USA 172 -176
9. Stefan Block . (2009). A comparison of three part-of-speech taggers
10. Keleta, Yemane. (2016). Nagaoka Tigrinya Corpus
11. GitHub Homepage, <https://github.com/maxpumperla/hyperas>, last accessed 2020/04/30.
12. Keras Homepage, <https://keras.io/>, last accessed 2020/04/30
13. TensorFlow Homepage, <https://www.tensorflow.org/>, last accessed 2020/04/30

14. Zhang Z., Zhang S., Chen E., Wang X., Cheng H. (2005) TextCC: New Feed Forward Neural Network for Classifying Documents Instantly. In: Wang J., Liao XF., Yi Z. (eds) Advances in Neural Networks – ISNN 2005. ISNN 2005. Lecture Notes in Computer Science, vol 3497. Springer, Berlin, Heidelberg
15. Limsopatham, Nut and Nigel Collier. “Bidirectional LSTM for Named Entity Recognition in Twitter Messages.” *NUT@COLING* (2016).
16. Joshuakyh.wordpress.com, <https://joshuakyh.wordpress.com/2017/12/02/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-embeddings/>, last accessed 2020/04/30.
17. Machine Learning Mastery, <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/>, last accessed 2020/04/30
18. The Clever Machine Home Page, <https://theclevermachine.wordpress.com/>, last accessed 2020/04/30.
19. Machine Curve, <https://www.machinecurve.com/index.php/2020/01/08/how-does-the-softmax-activation-function-work/#>, last accessed 2020/04/30.