

Article

Text Classification Based on Convolutional Neural Networks and Word Embedding for Low-Resource Languages: Tigrinya

Awet Fesseha ^{1,2} , Shengwu Xiong ^{1,*}, Eshete Derb Emiru ^{1,3} , Moussa Diallo ¹  and Abdelghani Dahou ¹

¹ School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, China; awet.fesseha@mu.edu.et (A.F.); eshetede@whut.edu.cn (E.D.E.); moussdiallo@whut.edu.cn (M.D.); dahou@whut.edu.cn (A.D.)

² College of Natural and Computational Sciences, Mekelle University, Mekelle 231, Ethiopia

³ School of Computing, DebreMarkos University, DebreMarkos 269, Ethiopia

* Correspondence: xiongsw@whut.edu.cn

Abstract: This article studies convolutional neural networks for Tigrinya (also referred to as Tigrigna), which is a family of Semitic languages spoken in Eritrea and northern Ethiopia. Tigrinya is a “low-resource” language and is notable in terms of the absence of comprehensive and free data. Furthermore, it is characterized as one of the most semantically and syntactically complex languages in the world, similar to other Semitic languages. To the best of our knowledge, no previous research has been conducted on the state-of-the-art embedding technique that is shown here. We investigate which word representation methods perform better in terms of learning for single-label text classification problems, which are common when dealing with morphologically rich and complex languages. Manually annotated datasets are used here, where one contains 30,000 Tigrinya news texts from various sources with six categories of “sport”, “agriculture”, “politics”, “religion”, “education”, and “health” and one unannotated corpus that contains more than six million words. In this paper, we explore pretrained word embedding architectures using various convolutional neural networks (CNNs) to predict class labels. We construct a CNN with a continuous bag-of-words (CBOW) method, a CNN with a skip-gram method, and CNNs with and without word2vec and FastText to evaluate Tigrinya news articles. We also compare the CNN results with traditional machine learning models and evaluate the results in terms of the accuracy, precision, recall, and F1 scoring techniques. The CBOW CNN with word2vec achieves the best accuracy with 93.41%, significantly improving the accuracy for Tigrinya news classification.

Keywords: text classification; CNN; low-resource language; machine learning; word embedding; natural language processing



Citation: Fesseha, A.; Xiong, S.; Emiru, E.D.; Diallo, M.; Dahou, A. Text Classification Based on Convolutional Neural Networks and Word Embedding for Low-Resource Languages: Tigrinya. *Information* **2021**, *12*, 52. <https://doi.org/10.3390/info12020052>

Academic Editor:

Yannis Korkontzelos

Received: 18 November 2020

Accepted: 19 January 2021

Published: 25 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The rise of Internet usage has led to the production of diverse text data that are provided by various social media platforms and websites in different languages. On the one hand, English and many other languages are regarded as affluent languages for the accessibility of the tools and data for numerous natural language processing tasks. On the other hand, many languages are also deemed to be low-resource languages [1]. Similarly, Negaish [2] and Osaman et al. [2,3] mentioned that Tigrinya is a “low-resource” language because of its underdeveloped data resources, few linguistic materials, and even fewer linguistic tools. Likewise, the lack of data resources for Tigrinya is manifested through the absence of a Tigrinya standard text dataset, and this is a significant barrier for Tigrinya text classification. Consequently, Tigrinya remains understudied from a natural language processing (NLP) perspective and this imposes challenges for the advancement of Tigrinya text classification research [3,4]. Tedla et al. [5] mentioned that unlike many other languages, the use of Tigrinya is rare in wiki pages. These pages are used as raw sources to construct unlabeled corpora (i.e., they are used for word embedding). Moreover, there are almost no

available Tigrinya datasets (i.e., datasets that are not freely available) [6,7]. Nevertheless, with the rise of Tigrinya textual data on the Internet and the need for an effective and robust automated classification system becomes necessary. Recent data show that the number of Internet users in Eritrea has increased by 5.66% (accessed September 2020, <https://www.internetworldstats.com/stats1.htm>). Similarly, in Ethiopia, the Internet growth rate is 204.72%. The two countries feature remarkable numbers of people that speak Tigrinya [8]. In addition to these dramatic growth trends, no significant research has been published for Tigrinya, unlike advanced studies on English, Arabic, and other languages. Furthermore, the unique nature of Tigrinya is also challenging from a NLP perspective due to its complex morphological structure, enormous number of synonyms, and rich auxiliary verb variations in terms of the subject, tense, aspect, and gender, besides the quantifiable availability of resources. Most of the characters in the language are inherited from a Semitic language background. Very few studies have reported results for Tigrinya NLP problems and these studies often report outcomes that have been found with small datasets. A few studies with small datasets have been conducted with the use of (SVM, decision tree, and others) machine learning techniques. Convolutional neural networks (CNNs) have recently gained popularity in various artificial intelligence areas, including image classification, face recognition, and other areas [9–11]. Kim et al. [12] demonstrated good performance of CNNs in the field of natural language processing.

Additionally, given the importance and utilization of news articles, the capability of the word embedding tool word2vec and associated CNNs for deep learning have been examined in several studies [10,12]. Kim et al. [12] proved that pretrained word vectors in sentence classification play vital roles by comparing word vectors with pretrained vectors. Earlier, Mikolov et al. [13,14] proposed several word embedding techniques that consider the meanings and contexts of words in a document based on two learning techniques, specifically, the continuous bag-of-words (CBOW) and skip-gram techniques. However, to the best of our knowledge, no previous research has been conducted with these state-of-the-art embedding techniques with Tigrinya news articles. Furthermore, for Tigrinya news articles, a comparison of the performance between skip-gram and CBOW techniques has also not been presented. We aim to investigate which word representation techniques perform better for learning in terms of solving Tigrinya single-label text classification problems using pretrained word vectors generated with both the CBOW and skip-gram techniques for FastText and word2vec. In this paper, we study Tigrinya text classification using CNNs. We evaluate the performance of the word2vec and FastText CNN classification models in terms of the training volumes and numbers of epochs and we contribute two Tigrinya datasets, i.e., a single-label dataset and a large unlabeled corpus. Furthermore, we explore the performances of various word embedding architectures [13] and CNNs [15] for classifying Tigrinya news articles. We also evaluate the performance in terms of the classification accuracies of the CNNs with pretrained word2vec and FastText models. The experimental results show that word2vec significantly improves the classification model accuracy by learning semantic relationships among the words. The results also show that the CBOW CNN model with word2vec and FastText performs better than the skip-gram CNN model.

The key contributions of this work are the following:

- We develop a dataset that contains 30,000 text documents labeled in six categories.
- We develop an unsupervised corpus that contains more than six million words to support CNN embedding.
- This work allows an immediate comparison of current state-of-the-art text classification techniques in the context of the Tigrinya language.
- Finally, we evaluate the CNN classification accuracy with word2vec and FastText models and compare classifier performance with various machine learning techniques.

It is expected that the results of this study will reveal how the use of a given word embedding model affects Tigrinya news article classification with CNNs. Furthermore, CNNs are used in many research approaches for natural language processing research

problems because of their ability to learn complex feature representations as compared with traditional machine learning approaches. We apply a CNN-based approach for categorization at the sentence level based on the semantics extracted from the corpus. We compare the FastText and word2vec pretrained vectors in terms of their impact on text classification. Our results indicate that the word2vec CNN approach outperforms the other approaches by 93.41% in terms of classification accuracy. The structure of this paper is as follows: In Section 2, we present the research background and related works; in Section 3, we present the research methodology; in Section 4, dataset construction and CNN architecture are described; in Section 5, we detail the evaluation techniques; in Section 6, we conclude the paper with a summary and discuss possible future work.

2. Background and Related Works

2.1. Previous Attempts for Tigrinya Natural Language Processing

We review existing works related to the proposed scheme, mainly considering previous attempts with the Tigrinya language. The majority of Tigrinya speakers live in Eritrea and the northern part of Ethiopia (Tigray Province) in Africa's horn, with an estimated population of more than 10 million [16]. Tigrinya is ranked third among the widely-spoken Semitic language families in the world, after Arabic and Amharic [17]. Despite Tigrinya sharing similarity with most Semitic languages in several ways, Tigrinya has different compound prepositions such as “ኣብልዕሉዓራት or ab leliarat” (on (top of) the bed), which has the preposition ኣብ/ab, preposition ልዕሉ/leli, and noun ዓራት/arar.

Furthermore, Abate et al. [18] mentioned that Tigrinya is a highly inflected language and features complex morphological characteristics due to basic word formations being based on sequences of consonants expressed by “roots” and “template patterns.” Littell et al. [19] stated that Tigrinya also shows both inflectional and derivational morphologies, where the former pertains to the tense, mood, gender, person, number, etc. Simultaneously, the latter produces different case patterns that include voice, causative, and frequentative forms. The presence of the two morphologies for the construction of enormous numbers of variants for a single word through the prefix, infix, and suffix affixations leads to the lack of data. Nonetheless, Tigrinya belongs to the set of low-resource languages that are conveyed with minimal data, linguistic materials, and tools [4,6].

Recently, a few researchers have attempted to challenge common corpora techniques, as shown in Table 1. Some of the researchers have attempted to develop volumes of words, tokens, or sentences. Furthermore, a few Tigrinya researchers have considered preprocessing by using word stemming techniques and their impact on result accuracy [3]. In their first attempt, Fisseha [20] developed a rule-based stemming algorithm with a dictionary-based stemming method that found better results for feature selection with Tigrinya information retrieval tasks, despite the fact that most stemming methods give good results. Overall, according to the Tigrinya NLP literature review, we have observed that none of the researchers have implemented neural network approaches for the text classification of the Tigrinya language.

Table 1. Literature review of previous Tigrinya natural language processing (NLP) research papers.

Author	Main Application	Sentences/Tokens	Year
Fisseha [20]	Stemming algorithm	690,000	2011
Reda et al. [21]	Unsupervised ML word sense disambiguation	190,000	2018
Osman et al. [3]	Stemming Tigrinya words	164,634	2012
Yemane et al. [6]	Post tagging for Tigrinya	72,000	2016

2.2. Text Classification

A conventional text classification framework consists of preprocessing, feature extraction, feature selection, and classification stages. These applications have to deal with several problems related to both the nature and structure of the underlying textual infor-

mation for languages by converting word variations into concise representations while preserving most of the linguistic features. Similarly, Uysal et al. [22] also studied the impact of preprocessing on language in both the text and language domains, where the preprocessing affected the accuracy. They further concluded that the preprocessing step in text classification is as essential as the feature extraction, feature selection, and classification steps. Specifically, conventional approaches for text analysis use typical features, such as bag-of-words [23], n-gram [24], and term frequency-inverse document frequency (TF-IDF) methods [25] as input methods for machine learning algorithms such as Naïve Bayes (NB) classifiers [26], K-nearest neighbor (KNN) algorithms [27], and support vector machines (SVMs) [28] for classification. Text classification is based on the statistical frequency of sentiment-related words extracted from tools such as lexicons [29]. Zhang et al. [29] provided an improved TF-IDF approach that used confidence, support, and characteristic words to enhance the recall and accuracy for text classification.

It is easy to see how machine learning has become a field of interest for text classification tasks, where machine-learning methods show great potential for obtaining linguistic knowledge. Although statistical machine learning-based representation models have achieved comparable performance, their shortcomings are apparent. First, these techniques only concentrate on word frequency features and completely neglect the contextual structure information in text, making it a challenge to capture text semantics. Second, the success of these statistical approaches in machine learning typically heavily depends on laborious engineering feats and the use of enormous linguistic resources.

In recent years, there has been a complete shift from statistical machine learning to state-of-the-art deep learning with text categorization models [30,31]. Zhang et al. [32] mentioned that natural language-based text classification has a wide range of applications, ranging from emotion classification to text classification. With their first design, Collobert and Weston [33] found that image preprocessing methods could also be used for natural language preprocessing. Moreover, many researchers have applied neural networks to text classification problems by using end-to-end deep neural networks to extract contextual features from raw text data. Kim [12] adopted a method to capture local features from different positions of words in sentences using various convolutional neural network architectures. Similarly, Zhang et al. [34] designed a powerful method using character-level information for text classification. Furthermore, Lai [35] also recommended recurrent neural network models for contextual information, together with a convolutional neural network. Most useful information from text is obtained through pooling technology and also CNNs in conjunction with unsupervised word vectors on top of single-layer convolutions and the use of relatively simple kernel convolutional kernels as fixed windows. Pennington et al. [36] devised an approach based on a corpus that considered linear substructures in word embedding space models such as word2vec via thorough training with global word co-occurrence data.

For multi-label classifications with short texts, Parwez [37] mentioned that CNN architectures introduce promising results by using domain-specific word embedding. Tang et al. [38] stated that sentiment-based word embedding models should be designed by encoding textual information along with word contexts, enabling the discernment of opposite word polarities in related contexts. On the basis of the improved word embedding methods where training is based on word-to-word co-occurrence in a corpus, a CNN is used here to extract features in order to obtain excellent high-level sentence representations. Pennington et al. [36] introduced the GloVe model, which is an unsupervised word log-bilinear regression model that is used for mastering the representations of relatively uncommon words. Additionally, Joulin et al. [39] tried to show learning model representations of vectors by mixing unsupervised and supervised techniques to research the vectors of words to capture semantic information. In [40], it was shown that combining a CNN with a RNN (recurrent neural networks) for the sentiment analysis of short texts provide good results. In [25], a CNN was used, and character-level information was considered, to support word-level embedding. One of the contributions of this work is using an

end-to-end network that comprises four main steps, namely, word vectorization, sentence vectorization, document vectorization, and then classification. We compare the results of the proposed method with other machine learning approaches.

2.3. Word Embeddings

Word embedding is foundational to natural language processing and represents the words in a text in an R-dimensional vector space, thereby enabling the capture of semantics, semantic similarity between words, and syntactic information for words. Word embedding approaches via word2vec have been proposed by Mikolov et al. [15]. Pennington et al. [36] and Arora et al. [36,41] introduced Word2vec's semantic similarity as a standard sequence embedding method that translates natural language into distributed representations of vectors; however, in order to overcome the inability of a predefined dictionary to learn rare word representations, FastText [42] is also used for character embedding. The word2vec and FastText models, which include two separate components (CBOW and skip-gram), can capture contextual word-to-word relationships in a multidimensional space as a preliminary step for predictive models used for semantics and data retrieval tasks [14,40]. Figure 1 shows that when the context words are given, the CBOW component infers the target word, while the skip-gram component infers the context words when the input word is provided [43]. In addition to that, the input, projection, and output layers are available for both learning algorithms, although their processes of output formulation are different. The input layer receives $W_n = \{W_{(c-2)}, W_{(c-1)}, \dots, W_{(c+1)}, W_{(c+2)}\}$ as arguments, where W_n denotes words. The projection layer corresponds to an array of multidimensional vectors and stores the sum of several vectors. The output layer corresponds to the layer that outputs the results of the vectorization.

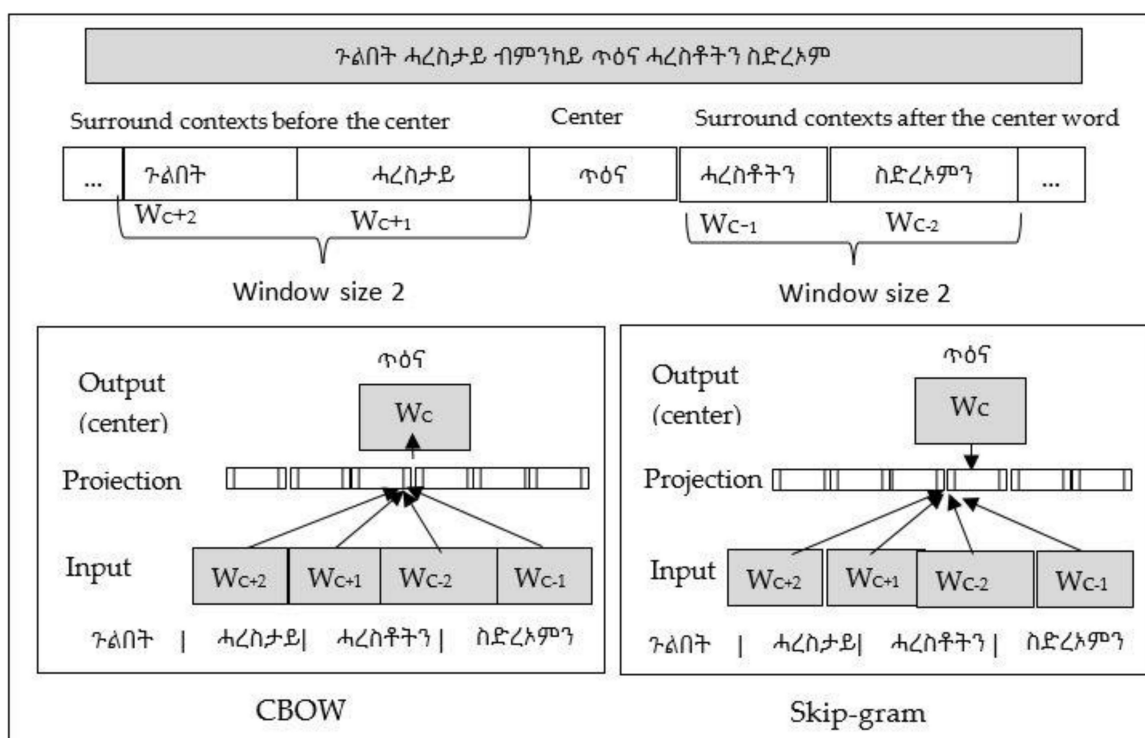


Figure 1. Continuous bag-of-words (CBOW) and skip-gram architectures.

3. Research Methods and Dataset Construction

Research Methods

Figure 2 shows the research methodology. Generally, the first process is data collection, where text data are collected from various sources. We created a single-label dataset (Dataset 1) that used 90% of data for training and 10% for testing. We also considered an unsupervised corpus (Dataset 2). Some text preprocessing steps were carried out before the data were passed to the model. These steps included removing extra white spaces, removing meaningless words, removing duplicate words, tokenization, cleaning, and the removal of stop words. These steps provided unique and meaningful sequences of words with unique identifications. With Dataset 2, we performed word embedding using FastText and word2vec with both the CBOW and skip-gram algorithms. These algorithms were trained with 100 dimensions and window sizes of 5 for both word embedding techniques to capture meaningful vectors that were able to learn from the nature of our data type, as well as from the morphological richness of the language. Using the preprocessed words, the embedding layer learned distributed representations for input tokens and these tokens had the same latent relationships. We applied a CNN-based approach to automatically learn and classify sentences into one of the six categories in evaluation Dataset 1. CNNs require inputs to have a static size and sentence lengths can vary greatly. Consequently, we used a maximum average word length of 235. Finally, considering the categorical news articles, based on the pretrained word vectors, we evaluated the accuracy, precision, recall, and F1 scores between methods. The methodology that we used is very close to that which was proposed in [12].

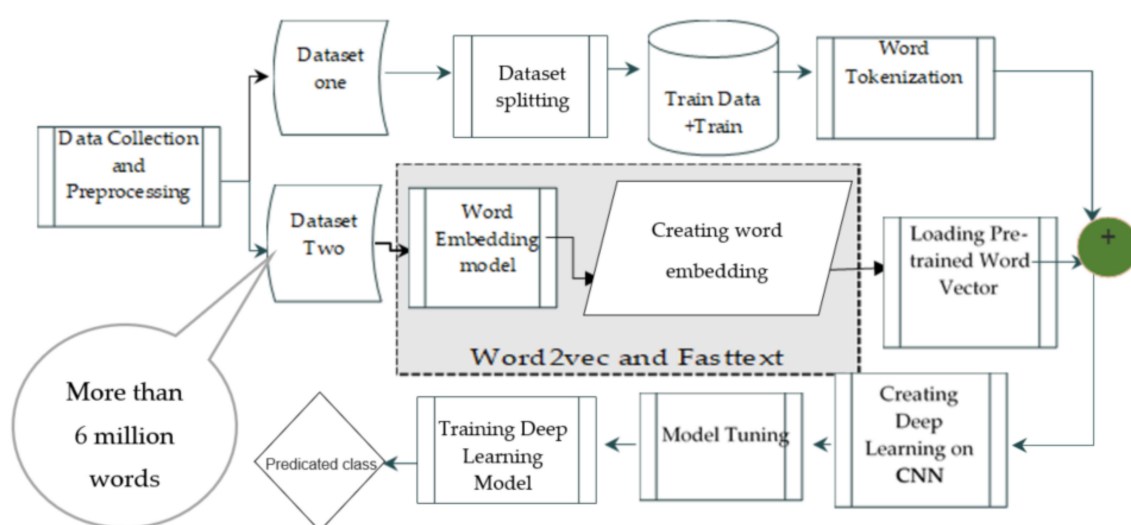


Figure 2. Block diagram of the overall architecture of our method. CNN, convolutional neural network.

4. Dataset Construction and CNN Architecture

4.1. Single-Label Tigrinya News Articles Dataset

We obtained news articles from popular news sources via web scraping and manual data collection techniques. We employed web-scraping tools (Selenium Python, requests, BeautifulSoup, and PowerShell) for news accessible sources on the Internet (Figure 3a) and the number of articles for each categories as stated in (Figure 3b). Furthermore, we also collected news articles in the form of word documents from the Tigray Mass Media Agency (a broadcast TV news agency) that transmitted on air from 2012 to 2018. The newly underlying corpus has six categories, i.e., ስፖርት ("sport"), ሃይማኖት ("religion"), ጥዕና ("health"), agriculture ("ኤኒሻ"), politics ("ፖለቲካ"), and education ("ትምህርት") (<https://github.com/canawet/Tigrigna-convolution-using-word2vec>) (<https://github.com/canawet/Tigrigna-convolution-using-word2vec>).

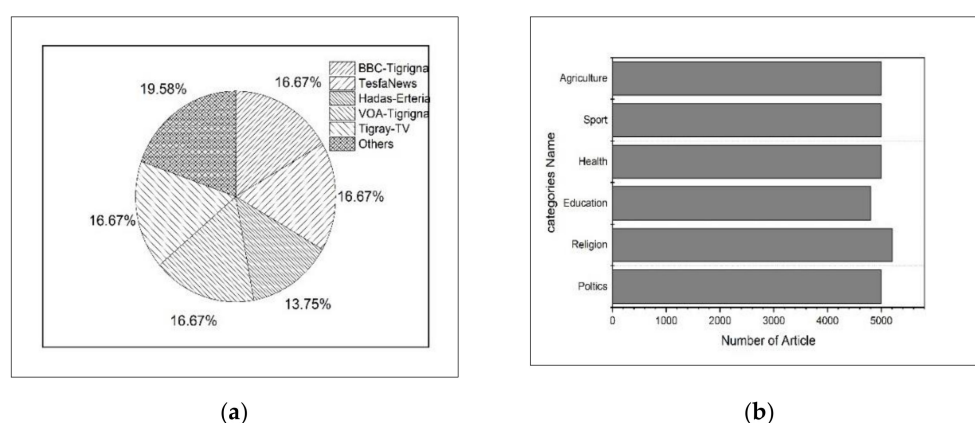


Figure 3. (a) Corpus website sources; (b) Number of articles per categories.

For the collected news articles, data were assigned to particular categories based on manual categorization.

Our dataset consisted of 30,000 articles, and we split the dataset into subsets, i.e., 24,000 articles for the training set and 6000 for the test set. We used 90% of articles in the training set (3600 for each category) and 10% for testing (1000 articles for each class). The training dataset was used to train the classifier and optimize the parameters, while the test dataset (unseen to the model) was reserved for testing the built model and determining the quality of the trained model. Statistics for the aforementioned dataset are given in Table 2.

Table 2. Dataset summary.

Training Instances	Validation Instances	Category (Class Labels)	Length (Maximum)	Words (Average)	Vocabulary Size
30,000	6000	6	1000	235	51,000

4.1.1. Tigrinya Corpus Collection and Preparation

In NLP applications, the use of conventional features such as term frequency-inverse document frequency (TF-IDF) has proven to be less efficient than word embedding [44]. Consequently, as a significant part of our contribution, we developed our own “Tigrinya multi-domain” corpus via data collected from various sources. Our second dataset was used to test the success of solving Tigrinya text classification problems and was used with the word2vec model using Genism [45]. Figure 4 show the steps for constructing our word embedding model Furthermore Figure 5 also shows similar model that apply to our methodology.

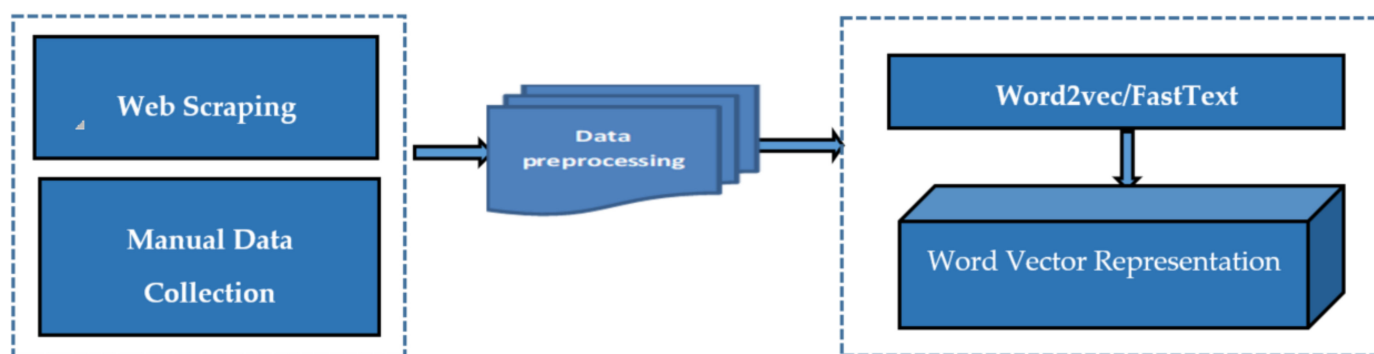


Figure 4. Summary for Tigrinya word embedding. The left section of the figure describes the data collection process. The middle section shows the data-preprocessing step. The right section describes the word embedding process.

4.1.3. Basic CNN Architecture

Sequence Embedding Layer

Similar to many other CNN models [30,46], as shown in Figure 6, which is based on [12], a sequential text vector is obtained by concatenating the embedded vectors of the component words. Equation (1) details our method of word embedding as:

$$\mathbf{X} = \mathbf{X}_1 \oplus \mathbf{X}_2 \oplus \mathbf{X}_3 \oplus \dots \oplus \mathbf{X}_n \quad (1)$$

We made the lengths of sentences equal by padding zero values to form a text matrix of $k \times n$ dimensions with k tokens and n length-embedding vectors. As shown in Equation (1), we can represent a concatenation operator to concatenate word vector \mathbf{X}_i corresponding to the i th word, and k represents the number of words/tokens present within the text. We consider k with a fixed length here ($k = 250$). To capture the discriminative features from low-level word embedding, the CNN model applies a series of transformations to the input sentence $\mathbf{X}_{i:n}$ using convolution, nonlinearity activation, and pooling operations in the following layers.

Convolutional Layer

Unique features in the convolutional layer are extracted as word vectors corresponding to each filter and feature map from a different width-embedding matrix. Discriminative word sequences are found during the training process. The extracted features have low-level semantic features as compared with the original text, thus, reducing the number of dimensions. The convolution word filter considers positions that are independent for every word and filters at higher layers capture syntactic or semantic associations between phrases that are far apart in a text.

A $\mathbf{W} \in \mathbb{R}^{m \times n}$ filter was applied to word sections to obtain high-level representations, where m shifts with stride s through the embedding matrix to produce feature map c_i and where each C_i is calculated using Equation (2). In this equation, “*” represents the convolution operation, which represents word vectors from \mathbf{X}_i to \mathbf{X}_{i+m-1} (i.e., m rows at a time) from \mathbf{X} covered by filter \mathbf{W} using strides. Additionally, “bi” represents the biased term and the activation function f is usually of a nonlinear form, such as a sigmoid or hyperbolic tangent form. In our case, f represents the rectified linear unit (ReLU) activation function as:

$$\mathbf{C}_i = f(\mathbf{W} * \mathbf{X}_{i:i+m-1} + \mathbf{b}_i) \quad (2)$$

The ReLU is applied to a layer to inject nonlinearity to the system by making all the negative values zero for any input x , as shown in Equation (3):

$$f = \max\{0, 1\} \quad (3)$$

This helps increase the model training speed without any significant difference in accuracy. Once the filter F iterates over the entire embedding matrix, we obtain a corresponding feature map as shown in Equation (4):

$$\mathbf{C}(f) = [\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_{n-m} + 1] \quad (4)$$

Pooling Layer

The convolutional layer outputs are, then, passed to the pooling layer, which aggregates the information and reduces the representation through common statistical methods, such as finding the mean, maximum, and L2-norm. The pooling layer can alleviate overfitting and produce vectors of sentences with fixed lengths. Suppose there are K

different filters, we aggregate the original information in feature maps by pooling as per Equation (5):

$$Z_{pooled} = \begin{bmatrix} \text{pool}(f(F_1 * W + b)) \\ \vdots \\ \text{pool}(f(F_z * W + b)) \end{bmatrix} \quad (5)$$

where F_i is the i th convolutional filter map with the bias vector $Z_{pooled} = (Z_1^{max}, Z_2^{max} \dots z_k^{max})$, which is a learned new distributed representation of the input sentence. In this paper, we utilize a max-over-time pooling operation, which selects global semantic features and attempts to capture the most important feature with the highest value for each feature map. Given a feature map z_i , the pooling operation returns the maximum value z^{max} in map z_i , i.e., $z^{max} = \max$, where Z_i is the resulting feature corresponding to the filter. Therefore, we can obtain vectors if the model employs K parallel filters with different window sizes.

Fully Connected Layer

In this layer, each neuron has full connections with all of the neurons in the previous layer. The connection structure is the same as with layers in classic neural network models. Dropout regularization is applied to the fully connected layer to avoid overfitting, and therefore improve the generalization performance. When training the model, neurons that are dropped have a probability of being temporarily removed from the network. Dropped neurons are ignored when calculating the input and output for both forward propagation and backward propagation. Therefore, the dropout technique prevents neurons from co-adapting too much by making the presence of any neuron unreliable.

Softmax Function

Finally, the vector representations in the fully connected layer are passed to the softmax function, and the output of the softmax function is the probability distribution over the labels. For example, the probability of label y_i is calculated as per Equation (6):

$$\begin{aligned} P(y = j | \hat{C}, W, b') &= \text{softmax}(W * (\hat{C} \text{ or}) + b') \\ &= \frac{\exp((W * (\hat{C} \text{ or}) + b')_j)}{\sum_k^n \exp((W * (\hat{C} \text{ or}) + b')_k)} \end{aligned} \quad (6)$$

A categorical cross-entropy loss function was used to train the classifier to categorize news articles into different categories by training the classifier via calculating gradients and using backward propagation. The loss was calculated using Equation (7), where x_i is the i th element of the dataset, y_i represents the predicted label of the element x_i , t represents the number of training samples, and θ describes the parameters:

$$J(\theta) = -\frac{1}{t} \sum_{i=1}^t \log(P(Y = y^i | x^i)) \quad (7)$$

5. Experiment and Discussion

In this section, we explore the performances of CNN models trained with word2vec (CBOW and skip-gram) and CNN models trained with FastText (CBOW and skip-gram) for news articles. We use the accuracy, recall, precision, and F1 score as performance metrics. The expressions of these metrics are given as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (9)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (10)$$

$$\text{F1 score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (11)$$

where the number of real positives among the predicted positives is delineated by true positive (TP) and true negative (TN) denotes the number of real negatives among the predicted negatives. Similarly, false negative (FN) denotes the number of real positives among the predicted negatives, and false positive (FP) denotes the number of real negatives among the predicted positives. Therefore, accuracy denotes the proportion of documents classified correctly by the CNN among all documents, and recall denotes the proportion of documents that are classified as positive by the CNN among all real positive documents. Precision denotes the percentage of documents that are real positives among documents classified as positive by the CNN, and the F1 score denotes the average of the weighted recall and precision scores.

5.1. CNN Parameter Values

The CNN parameters that were applied for the pretrained CBOW and skip-gram models were adapted from the literature [12,47]. A grid hyperparameter optimization search method was applied to find the optimum value, as shown in Tables 4 and 5 for both word embedding and CNN parameter. The word vector dimension d was equal to 100. Four different window sizes (filter widths) were used. The use of 100 filters resulted in 100 feature maps, and the convolution filter weights and softmax weights were taken uniformly from the interval $[-0.1, 0.1]$. The maximum pooling size was 2, which was used to pool high-level features from the feature maps, and the pooled values were concatenated to produce a single vector at the fully connected dense layer which was used to calculate class probabilities. The model was trained using the “Adam” learning rate method with 20 epochs to avoid overfitting using a callback function. We used the dropout parameter P at the embedding layer with $p = 0.15$ and the L2 regularization parameter value of 0.03 for the convolutional layer.

Table 4. Word embedding parameters.

Parameter	Value
Word embedding size	100
Window size (filter size)	2, 3, 4, and 5
Number of filters for each size	100
Dropout probability at the embedding layer	0.15

Table 5. Parameters for our CNN.

Parameter	Value
Epochs	20
Learning rate	0.001
Regularization rate	0.025
CNN dropout probability	0.2
Optimization	Adam

5.2. Comparison with Traditional Models

We compared our CNN models with traditional machine learning models. The CNN models were tested and compared with four of the most common machine learning models, i.e., SVM, Naïve Bayes, decision tree, and random forest models with BOW features, and considering unigrams where vector representation was carried out using TF-IDF vectors. For the SVM models, models from the “sklearn” package in the scikit-learn library were used, while for Naïve Bayes, we employed “MultinomialNB” from scikit-learn.

The Naïve Bayes model that we used was the one in the scikit-learn library. Similarly, for the decision tree and random forest models, we used the modules from scikit-learn. Furthermore, to deal with the sparsity of the feature matrix, all CNN algorithms were run five times with the same parameter settings. Table 6 presents accuracy and training values, where the experiment analysis showed that the CNN had featured better training accuracy, while, among the traditional machine learning models, the random forest model showed the best validation accuracy (SVM, naive Bayes, decision tree, and random forest).

Table 6. Training and validation accuracies between models.

Classifier	Training Accuracy	Validation Accuracy
Naïve Byes	0.9287	0.8658
Random Forest	0.9307	0.8356
SVM	0.9503	0.8452
Decision tree	0.9663	0.8123
CNN without embedding	0.9335	0.9141

5.3. Comparison of Pretrained Word2vec with CNN-Based Models

In this section, we compare the overall performance of word embedding, considering both the CBOW and skip-gram models and without pretrained vector CNN model based on the experiments. Figure 7 shows accuracy and F1 score comparisons for the generated vectors with word2vec embedding, when the number of training volume were static with 22k news articles for the CNN with skip-gram, the CNN with CBOW, and the CNN with pretrained vectors. The CBOW CNN showed the highest performance for all training volumes, with 0.9341 and 0.9274 as the values for the accuracy and F1 score, respectively. The CBOW CNN also delivered better performance in terms of accuracy in volume 16. The skip-gram CNN also showed better performance in volume 20, while the other CNNs trained without pretrained vector fluctuations at various volumes and ultimately decreased in accuracy.

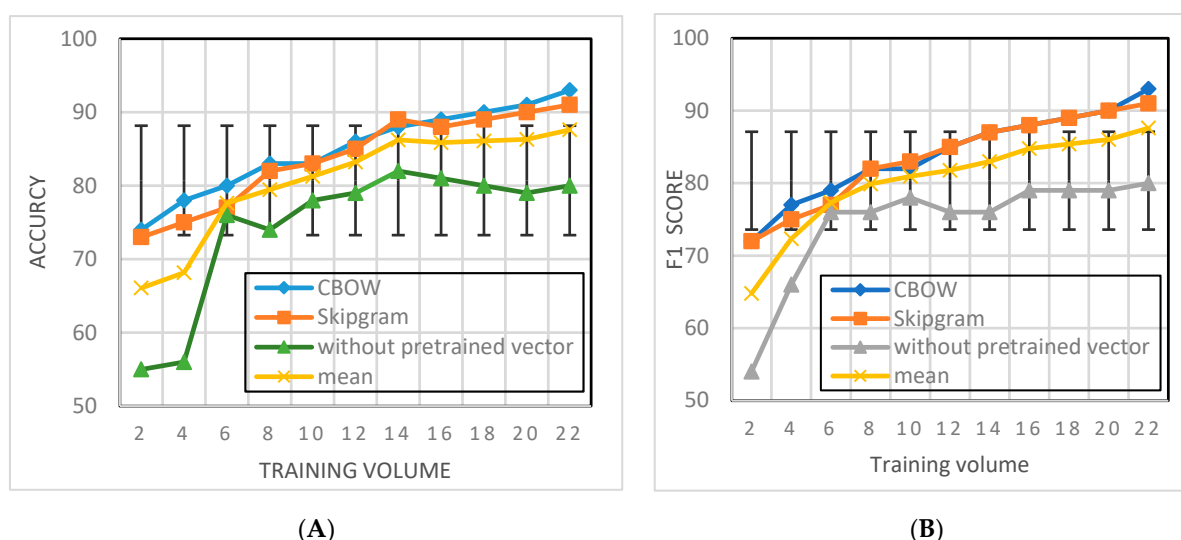


Figure 7. CNN with word2vec. (A) Accuracy; (B) F1 score.

5.4. Comparison of FastText Pretrained on CNN-Based Models

This section considers FastText word embedding vectors and compares the performances among the CNNs. Figure 8 clearly shows that the CBOW-FastText model scored better results than the skip-gram CNN even though it did not show significant differences for almost all training volumes. The CNN with CBOW obtained values of 0.9013 and 0.9012

for the accuracy and F1 scores, respectively. The CBOW and skip-gram CNNs showed higher performance at volume 14. The random CNN vectors subsequently decreased.

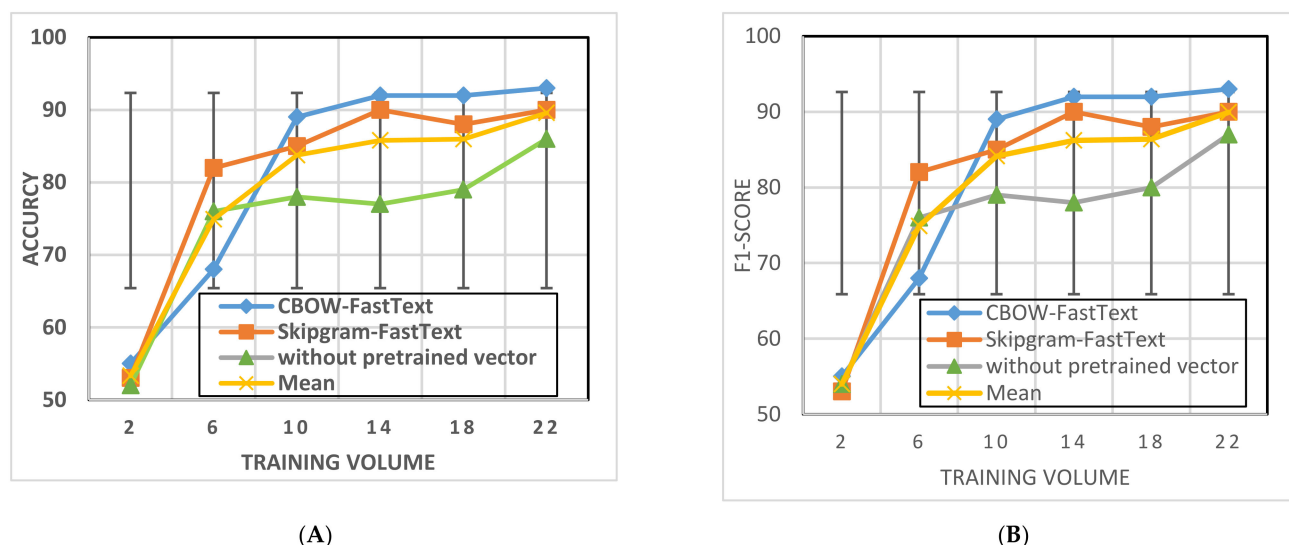


Figure 8. CNN results with FastText. (A) Accuracy; (B) F1 score.

We observed that the CBOW CNN trained with word2vec shows better performance than the CBOW CNN trained with the skip-gram model. Nevertheless, CNN without pretrained vector accuracy decreased when the number of volumes and epochs increased; however, all CBOW CNN models showed better performance when the volumes and epochs increased. The performance of the random CNN decreased with the absence of pretrained data, which implies that there are direct relationships between word embedding representations for text classification with CNNs. Table 7 shows the accuracy and F1 scores of a CNN with both word2vec and FastText. As compared with the CBOW CNN for the vector in news articles, this model was less stable and required more epochs to reach maximum performance. For example, when the training volume corresponded to 24, the CNN with the vector in the news article exhibited an accuracy of 0.93 or more when the epoch number corresponded to 17, although the CNN with the vector exhibited an accuracy of 0.91. However, the skip-gram CNN with the vector did not exhibit any significant difference in terms of the maximum values for the accuracy and F1 scores. FastText also showed similar results to the CBOW CNN when we compared the vectors in news articles when the training volume corresponded to 24, and the news articles exhibited an accuracy of 0.90 or more when the number of epochs was 20.

Table 7. Comparison of the experimental results.

Pretrained Vector	Model with Pretrained	Accuracy	F1 Score	Time for Training (S)	Training Volume	Epochs
Word2vec	CNN + CBOW	0.9341	0.9151	2000	24	20
	CNN + skip-gram	0.9147	0.9161	1865	24	17
	CNN + without pretrained	0.7905	0.7809	900	16	13
FastText	CNN + CBOW	0.9041	0.9054	1980	24	20
	CNN + skip-gram	0.8975	0.8909	1723	24	20
	CNN + without pretrained	0.7941	0.7951	868	16	14

5.5. Comparison of CBOW Results Word2vec and FastText

Similarly, in Table 8 we compared pretrained word embedding results for the CBOW models with the word2vec and FastText results. Subsequently, those algorithms showed better results. The best results for news categories were found for sport category with the

word2vec CBOW model, obtaining values of 0.9301, 0.9312, and 0.9306 for the precision, recall, and F1 scores, respectively. Meanwhile, the CBOW FastText model also showed that sport category achieved values of 0.9202, 0.9214, and 0.9206 for the precision, recall, and F1 scores, respectively, which indicated that the sport category provided good results.

Table 8. Word2vec and FastText results by news categories.

Category	Pretrained Word2vec (CBOW)			Pretrained FastText (CBOW)		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Agriculture	0.8807	0.8834	0.8820	0.8607	0.8624	0.8615
Sport	0.9301	0.9312	0.9306	0.9202	0.9214	0.9206
Health	0.9212	0.9234	0.9222	0.9201	0.9202	0.9202
Education	0.9231	0.9311	0.9270	0.9031	0.9311	0.9168
Politics	0.9105	0.9151	0.9127	0.9105	0.9151	0.9127
Religion	0.9201	0.9113	0.9156	0.9001	0.9013	0.9006
Average	0.9438	0.9425	0.9150	0.9438	0.9425	0.9054

Table 4 shows that the CNNs showed better results than traditional machine learning results due to the use of pretrained word vectors for enhancing the learning of word representations. Additionally, CNN performance was meaningfully reduced in the absence of pretrained word vectors (PWVs). This indicates that word relationships are significant factors for learning in this context. Moreover, in Figure 7, we compared two word embedding models in terms of the accuracy and F1 scores and revealed that the CBOW CNN-CBOW trained with word2vec and FastText shows higher performance than the skip-gram CNN model, even though the output results were close between some models. Moreover, as in Tables 6 and 7, this situation can be interpreted as the CBOW CNN model being better at representing common words in the corpus considered here, since the skip-gram CNN algorithm was better for learning rare words. In this paper, FastText and word2vec provided better results for Tigrinya news article classification than the CBOW skip-gram and CBOW word2vec models.

6. Conclusions

We have used word2vec and FastText techniques, and our results suggest that they are among the best word embedding techniques in the field of NLP research. In this study, we have evaluated word2vec and FastText in classification models by applying CNNs to Tigrinya news articles. We observed that word2vec improved the classification model performance by learning the relationships among words. We further compared and analyzed both of the word2vec models with machine learning models. The current study shows that the most successful word embedding method was the CBOW algorithm for the news articles considered here. This study is expected to aid future studies on deep learning in the field of Tigrinya text processing and natural language processing. The word vectors and datasets created in this study contribute to the current literature on Tigrinya text processing. In the future, we plan to address other embedding techniques, such as GloVe, BERT, XLNET, and others. We also intend to find optimal solution methods for large-scale word embedding problems, which is currently a time-consuming problem.

Author Contributions: Conceptualization, A.F.; data curation, A.F.; formal analysis, A.F., S.X., E.D.E. and A.D.; funding acquisition, S.X.; investigation, A.F., E.D.E. and M.D.; methodology, A.F., E.D.E. and M.D.; resources, S.X.; supervision, S.X.; validation, M.D.; writing—original draft, A.F.; writing—review & editing, A.D. All authors have read and agreed to the published version of the manuscript.

Funding: The National Natural Science Foundation of China supported this work under grant number 2017GF004.

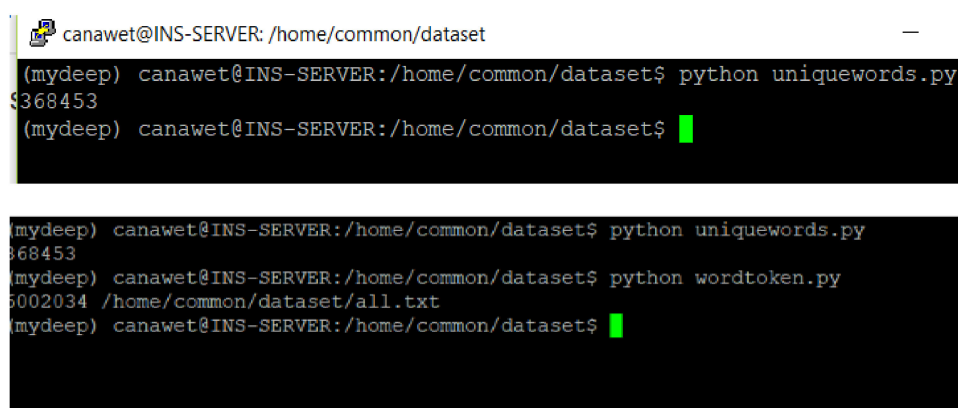
Acknowledgments: We are very grateful to the Chinese Scholarship Council (CSC) for providing financial and moral support and Mekelle University staff for their help during data collection.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

CNN	Convolutional neural network
TF-IDF	Term frequency-inverse document frequency
NLP	Natural language processing
ML	Machine learning
CBOW	Continuous bag of words
ReLU	Rectified linear unit
SVM	Support vector machine
NB	Naïve Bayes
KNN	K-nearest neighbor
BOW	Bag-of-words
PWV	Pretrained word vector

Appendix A



```

canawet@INS-SERVER: /home/common/dataset
(mydeep) canawet@INS-SERVER:/home/common/dataset$ python uniquewords.py
368453
(mydeep) canawet@INS-SERVER:/home/common/dataset$

(mydeep) canawet@INS-SERVER:/home/common/dataset$ python wordtoken.py
368453
(mydeep) canawet@INS-SERVER:/home/common/dataset$ python wordtoken.py
5002034 /home/common/dataset/all.txt
(mydeep) canawet@INS-SERVER:/home/common/dataset$

```

Figure A1. Further information.

References

1. Al-Ayyoub, M.; Khamaiseh, A.A.; Jararweh, Y.; Al-Kabi, M.N. A comprehensive survey of arabic sentiment analysis. *Inf. Process. Manag.* **2019**, *56*, 320–342. [CrossRef]
2. Negash, A. The Origin and Development of Tigrinya Language Publications (1886-1991) Volume One. 2016. Available online: <https://scholarcommons.scu.edu/cgi/viewcontent.cgi?article=1130&context=library> (accessed on 20 January 2021).
3. Osman, O.; Mikami, Y. Stemming Tigrinya words for information retrieval. In Proceedings of the COLING 2012: Demonstration Papers, Mumbai, India, 1 December 2012; pp. 345–352. Available online: <https://www.aclweb.org/anthology/C12-3043> (accessed on 20 January 2021).
4. Tedla, Y.K.; Yamamoto, K.; Marasinghe, A. Nagaoka Tigrinya Corpus: Design and Development of Part-of-speech Tagged Corpus. *Int. J. Comput. Appl.* **2016**, *146*, 33–41. [CrossRef]
5. Tedla, Y.K. *Tigrinya Morphological Segmentation with Bidirectional Long Short-Term Memory Neural Networks and its Effect on English-Tigrinya Machine Translation*; Nagaoka University of Technology: Niigata, Japan, 2018.
6. Tedla, Y.; Yamamoto, K. Analyzing word embeddings and improving POS tagger of tigrinya. In Proceedings of the 2017 International Conference on Asian Language Processing (IALP), Singapore, 5–7 December 2017; pp. 115–118. [CrossRef]
7. Tedla, Y.; Yamamoto, K. The effect of shallow segmentation on English-Tigrinya statistical machine translation. In Proceedings of the 2016 International Conference on Asian Language Processing (IALP), Tainan, Taiwan, 21–23 November 2016; pp. 79–82. [CrossRef]
8. Stats, I.W. Available online: <https://www.internetworldstats.com/> (accessed on 10 September 2020).
9. Kalchbrenner, N.; Blunsom, P.J. Recurrent convolutional neural networks for discourse compositionality. *arXiv* **2013**, arXiv:1306.3584.
10. Jiang, M.; Liang, Y.; Feng, X.; Fan, X.; Pei, Z.; Xue, Y.; Guan, R.J.N.C. Text classification based on deep belief network and softmax regression. *Neural. Comput. Appl.* **2018**, *29*, 61–70. [CrossRef]
11. Kalchbrenner, N.; Grefenstette, E.; Blunsom, P.J. A convolutional neural network for modelling sentences. *Neural. Comput.* **2014**. [CrossRef]
12. Kim, Y. Convolutional neural networks for sentence classification. *arXiv* **2014**, arXiv:1408.5882. [CrossRef]

13. Mikolov, T.; Sutskever, I. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 3111–3119.
14. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
15. Lai, S.; Liu, K.; He, S.; Zhao, J.J.I.S. How to generate a good word embedding. *IEEE Intell. Syst.* **2016**, *31*, 5–14. [CrossRef]
16. T. S. International, in Tigrinya at Ethnologue, Ethnologue. 2020. Available online: <http://www.ethnologue.com/18/language/tir/> (accessed on 3 March 2020).
17. Mebrahtu, M. Unsupervised Machine Learning Approach for Tigrigna Word Sense Disambiguation. Ph.D. Thesis, Assosa University, Assosa, Ethiopia, 2017.
18. Abate, S.T.; Tachbelie, M.Y.; Schultz, T. Deep Neural Networks Based Automatic Speech Recognition for Four Ethiopian Languages. In Proceedings of the ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 8274–8278.
19. Littell, P.; McCoy, T.; Han, N.-R.; Rijhwani, S.; Sheikh, Z.; Mortensen, D.R.; Mitamura, T.; Levin, L. Parser combinators for Tigrinya and Oromo morphology. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, 7–12 May 2018.
20. Fisseha, Y. *Development of Stemming Algorithm for Tigrigna Text*; Addis Ababa University: Addis Ababa, Ethiopia, 2011.
21. Reda, A. Unsupervised Machine Learning Approach for Tigrigna Word Sense Disambiguation. *Philosophy* **2018**, *9*.
22. Uysal, A.K.; Gunal, S. The impact of preprocessing on text classification. *Inf. Process. Manag.* **2014**, *50*, 104–112. [CrossRef]
23. Wallach, H.M. Topic modeling: Beyond bag-of-words. In Proceedings of the 23rd international conference on Machine learning, Haifa, Israel, 21–25 June 2010; pp. 977–984.
24. Gauging, D.M.J.S. Similarity with n-grams: Language-independent categorization of text. *Science* **1995**, *267*, 843–848.
25. Joachims, T.A. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. Available online: <https://apps.dtic.mil/docs/citations/ADA307731> (accessed on 20 January 2021).
26. McCallum, A.; Nigam, K. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*; Citeseer: Princeton, NJ, USA, 1998; Volume 752, pp. 41–48.
27. Trstenjak, B.; Mikac, S.; Donko, D.J.P.E. KNN with TF-IDF based framework for text categorization. *Procedia Eng.* **2014**, *69*, 1356–1364. [CrossRef]
28. Joachims, T. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 137–142.
29. Yun-tao, Z.; Ling, G.; Yong-cheng, W. An improved TF-IDF approach for text classification. *J. Zhejiang Univ. Sci. A* **2005**, *6*, 49–55.
30. Johnson, R.; Zhang, T. Semi-supervised convolutional neural networks for text categorization via region embedding. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 919–927.
31. Johnson, R.; Zhang, T. Supervised and semi-supervised text categorization using LSTM for region embeddings. *arXiv* **2016**, arXiv:1602.02373.
32. Zhang, L.; Wang, S.; Liu, B.; Discovery, K. Deep learning for sentiment analysis: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1253. [CrossRef]
33. Collobert, R.; Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; pp. 160–167.
34. Zhang, X.; Zhao, J.; LeCun, Y. Character-level convolutional networks for text classification. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 649–657. Available online: <https://arxiv.org/abs/1509.01626> (accessed on 20 January 2021).
35. Lai, S.; Xu, L.; Liu, K.; Zhao, J. Recurrent convolutional neural networks for text classification. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Menlo Park, CA, USA, 25–30 January 2015.
36. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543. [CrossRef]
37. Parwez, M.A.; Abulaish, M.J.I.A. Multi-label classification of microblogging texts using convolution neural network. *IEEE Access* **2019**, *7*, 68678–68691. [CrossRef]
38. Tang, D.; Wei, F.; Qin, B.; Yang, N.; Liu, T.; Zhou, M.; Engineering, D. Sentiment embeddings with applications to sentiment analysis. *IEEE Trans. Knowl. Data Eng.* **2015**, *28*, 496–509. [CrossRef]
39. Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of tricks for efficient text classification. *arXiv* **2016**, arXiv:1607.01759.
40. Wang, X.; Jiang, W.; Luo, Z. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, 11–17 December 2016; pp. 2428–2437.
41. Arora, S.; Liang, Y.; Ma, T. A simple but tough-to-beat baseline for sentence embeddings. 2016. Available online: <https://openreview.net/forum?id=SyK00v5xx> (accessed on 20 January 2021).
42. Joulin, A.; Grave, E.; Bojanowski, P.; Douze, M.; Jégou, H.; Mikolov, T. Fasttext. zip: Compressing text classification models. *arXiv* **2016**, arXiv:1612.03651.
43. Peng, H.; Song, Y.; Roth, D. Event detection and co-reference with minimal supervision. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–5 November 2016; pp. 392–402.

-
44. Kulkarni, A.; Shivananda, A. Converting text to features. In *Natural Language Processing Recipes*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 67–96. [[CrossRef](#)]
 45. Řehůřek, R. Models.Word2vec–Deep Learning with Word2vec. Available online: <https://radimrehurek.com/gensim/models/word2vec.html> (accessed on 16 February 2017).
 46. Liu, J.; Chang, W.-C.; Wu, Y.; Yang, Y. Deep learning for extreme multi-label text classification. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 115–124. [[PubMed](#)]
 47. Zhang, Y.; Wallace, B. A sensitivity analysis of (and practitioners’ guide to) convolutional neural networks for sentence classification. *arXiv* **2015**, arXiv:1510.03820.