ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL SCIENCES

SEMANTIC-AWARE AMHARIC TEXT CLASSIFICATION USING DEEP LEARNING APPROACH

Girma Moges Misganew

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

(In Data and Web Engineering)

Addis Ababa, Ethiopia

October 2020

Addis Ababa University

College of Natural Sciences

Girma Moges Misganew

Advisor: Yaregal Assabie (PhD)

This is to certify that the thesis prepared by Girma Moges Misganew, titled: ***Semantic-aware Amharic Text classification using Deep learning Approach*** and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science (in Data and Web Engineering) complies with the regulations of the university and meets the accepted standards concerning originality and quality.

Signed by the Examining Committee:

Name	Signature	Date
Advisor: Yaregal Assabie (PhD)		10/23/2020
Examiner: Dida Midekso (PhD)		10/23/2020
Examiner: Ayalew Belay (PhD)		10/23/2020

DEDICATED TO:

My Mother (Emebet Aleben)

Abstract

Now we are at the age of information era, information is stored, extracted, and used in different formats. Text can be an extremely rich source of information, but extracting insights from it can be hard and time-consuming due to its unstructured nature. Text classification is one of the methods used to organize massively available textual information in a meaningful context to maximize the utilization of information. Amharic text classification is done using the classical and traditional machine learning approaches with a limitation of semantic representation and use of high engineered feature extraction. However, the newly emerged deep learning approach and the use of word embedding improves the performance of text classification through extracting features automatically and represent words semantically in sparse vector.

Thus, we develop the LSTM model to train our data and to make the classification process. The classification of the Amharic text documents using the LSTM pass through the process of; preprocessing, word-embedding, deep network building, output determination, training the model, and classification. The semantics of document is done using word2vec, to map similar words in to a single vector using neural network architecture. Thus, the vector representations of words are used as the input for the dep network building component. The model is evaluated using accuracy and loss by training, testing, and validation dataset and resulted 92.13 testing accuracy, and 86.71 validation accuracy.

Keywords: Text classification, Deep learning, RNN, LSTM, and word-embedding.

Acknowledgments

First of all, I would like to thank the Almighty God and His mother Saint virgin Mary, for their support, help, and guidance in my overall life and the accomplishment of this work. The state of the art of my life depends on him to do everything and I did every task of this work by trusting on his willingness and support. Next, I would like to express my deepest gratitude to my advisor Yaregal Assabie (PhD) for his guidance and supportive comment and advice.

I have great respect and gratitude to my family and friends who support me to be successful in my entire life. Especially My father, Moges Misganew, contributes the more and the better for my achievement of the situation that I am here and my deepest gratitude finds him well.

My deepest gratitude and appreciation also go to Gashaw Demlew (Msc) for his willingness to support data collection and his advice to do better and achieve more. Besides, I am very grateful to my classmates that we share more information and help with each other for our accomplishment of the course and final thesis.

Table of Contents

List of figures

List of Tables

List of Algorithms

List of Acronyms and Abbreviations

AE Auto Encoder

BOW Bag of Words

CNN Convolutional Neural Network

CBOW Continuous Bag of Words

CPU Central Processing Unit

DNN Deep Neural Network

DL Deep Learning

DBOW Distributed Bag of Words

Doc2vec Document to Vector

GPU Graphical Processing Unit

GAN Generative Adversarial Network

LIWC Linguistic Inquiry and Word Count

LVQ Learning Vector Quantization

LSTM Long Short-Term Memory

ML Machine learning

NLP Natural Language Processing

RAM Random Access Memory

RDF Resource Description Framework

PCA Principal Component Analysis

SG Skip Gram

SVM Support Vector Machine

SRNA Semantic-aware Recurrent deep Network Architecture

SRM Structural Risk Minimization

TC	Text Classification
TF	Term Frequency
TF-IDF	Term Frequency Inverse-Document Frequency
Word2vec	Word to vector

Chapter 1: Introduction

1.1 Background

Text classification is one of the methods used to organize massively available textual information in a meaningful context to maximize the utilization of information. It is a hard yet very useful operation frequently applied to assign subject categories to documents, to route and filter texts. As a part of natural language processing systems unstructured data in the form of text is everywhere: emails, web pages, social media, support tickets chats, survey responses, and more. Text can be a major and rich source of information, but extracting insights from it can be time-consuming and hard due to its unstructured nature. Businesses are turning to text classification for structuring text in a fast and cost-efficient way to enhance decision-making and automate processes [1].

In the past, several methods proposed for text categorization were typically based on the classical Bag-of-Words model where each term or term stem is an independent feature. More recently in text classification systems; in addition to developing an ontology for extracting semantic meaning or concept, there are also other approaches which are semantic vocabulary, word embedding (word2vec, doc2vec, FastText), and WordNet to capture the semantic similarity between words, phrases, sentences, paragraphs and documents [2]. Semantic similarity is a metric defined over a set of documents or terms, where the idea of the distance between them is based on the likeness of their meaning or semantic content as opposed to similarity which can be estimated based on their structure or syntactical representation.

Semantic-aware text classification is the method for accomplishing the classification of texts in large volumes of information using the concept or meaning that the text or the document infers rather than categorizing texts based on word and/or phrase analysis of the text that is extracted from it [1]. Research works on Semantic-aware classification is flourishing in the context of other languages; whereas research on automatic Amharic text classification is not well adapted, and very few attempts have been made till now.

Machine learning is used for accomplishing text classification tasks. There are several learning techniques. The common ones include probabilistic methods, regression methods, decision tree, neural networks, batch and incremental learners of linear classifiers, Example-based methods, and support vector machines. Instead of relying on manually crafted rules Text classification with machine learning learns to make classifications based on past observations especially using deep learning approach [2]. By using pre-labeled examples as training data, a machine learning algorithm can learn the different associations between pieces of text and that particular output is expected for a particular input [2, 3].

Deep learning is a set of algorithms and techniques inspired by how the human brain works. Text classification has benefited from the recent resurgence of deep learning architectures due to their potential to reach high accuracy with less need for engineered features [4]. The two main deep learning architectures used in text classification are convolutional neural networks and recurrent neural networks. Thus, deep learning models have achieved state-of-the-art results across many domains, including a wide variety of NLP applications [5]. It is a kind of representation learning in which there are multiple levels of features. These features are automatically extracted and they are composed together in the various levels to yield the output. Each level represents abstract features that are extracted from the features represented in the previous level. Thus, the level of abstraction increases with each level. This type of learning enables discovering and representing higher-level abstractions. In neural networks, the multiple layers correspond to multiple levels of features. These multiple layers compose the features to produce the output [4].

Thus, we propose semantic-aware text classification using a deep learning approach to achieve the good features of both the semantics meaning and the performance of the deep learning approaches.

1.2 Motivation

Amharic is the working language of the Federal Democratic Republic of Ethiopia (FDRE), which increases the production of Amharic documents. Text classification is also one of the Natural language processing applications that enables to classify texts documents in a defined class based on the feature that is extracted from the semantic representation of the document. It is also used for different applications of natural language processing like information retrieval, information extraction, named entity recognition and document validation, etc.

The other motivation issue to work on this problem is also the invention of deep learning that improves the accuracy as well as the performance of different natural language processing applications through creating a computational model. That enables to extract features automatically from the source or the input data as a training data and its flexibility with feature design. The need to process and analyze that information for different purposes and applications of the world inspires or motives to do an investigation on the Amharic text classification.

1.3 Statement of the problem

Many of the text classification techniques are based on word and/or phrase analysis of the text. Term frequency analysis results in the importance of a term within a document. Two terms within a document may have the same frequency, but one term may underwrite more to the meaning of the sentence compared to the other term. The limitations of this classical representation are [1]:

- The ignorance of any relation between words; as a result of which learning algorithms are restricted to detect patterns in the used terminology only while conceptual patterns remain ignored.
- The big dimensionality of the representation space.

Opposed to the traditional text classification methods that need a set of well-classified trained dataset to perform classification in efficient ways, the semantic-aware classifier provides more accuracy, and benefits from its semantic nature[6].

Lower performance in the traditional approach of machine learning like SVM on the larger dataset and a more stable growth trend with the gradually increasing amount of training samples in deep learning [2, 7] This results in the increase in the performance of the classifier and the accuracy of the result by improving the problems which are faced by the traditional machine learning and keyword-based text classifications. Deep learning approaches have achieved surpassing results in comparison to previous machine learning algorithms text classifications. The success of these deep learning algorithms relies on their capacity to model complex and non-linear relationships within data [2].

In general, the semantic-aware deep learning approaches of text classifications are developed for different languages like English [7, 8], Arabic [9] and Telugu [1] and the characteristic of Amharic

language (writing system, morphological structure, encoding style) are different from those languages. Thus, we propose semantic-aware Amharic text classification using deep learning approach to overcome the problem encountered on the classical and traditional machine learning approach. Amharic text classifications are mostly done using the keyword-based [10] and the traditional machine learning approaches [4, 11] So that, we proposed deep learning approach with the semantic representation of documents to improve the performance of the Amharic text classification system.

1.4 Objectives

General objective

The general objective of this study is to develop a semantic-aware Amharic text classifier using a deep learning approach.

Specific objectives

To accomplish the general objective, the following specific objectives are done in the entire of this work are devised.

- Review literatures on deep learning architectures and algorithms
- Collecting data for preparing a corpus.
- Develop a model based on deep learning
- Develop a prototype of the system
- Evaluate the performance of the model

1.5 Methodology

To accomplish the general and specific objectives of this study and to solve the problems that we mention, different methodologies will be applied.

Literature review

Extensive literature reviews will be conducted to acquire enough understanding of the various approaches to text classification and semantics. Specifically, literatures in the area of text classification methods, semantic similarity, Amharic language writing system, machine learning approaches on the application of natural language applications. Thus, Reading and criticizing all these literature and related works help to gain the required knowledge on a certain subject. Hence, different research papers, books, and web sites will be exploited the right methods and tools for implementing the different components of the system will be identified.

Data collection

Text document corpus will be collected from different offline and online sources like magazines, fictions, religious books, electronic medias and different news agencies. Documents collected will be manually categorized by experts that will be further used for evaluation. These data will be organized and pass through the preprocessing step of natural text data in a way that they are easy for experimentation and testing.

Prototype development

In order to accomplish the objectives of the research, experimentation using available tools and programming will be engaged in the process. Thus, Python and Java programming language with keras and TensorFlow deep learning libraries will be used to develop the proposed system. The prototype is developing to show the outcome of our approach in the visual description or we develop the blueprint of our study.

Testing and Evaluation

To evaluate the performance of the proposed semantic-aware text classifier, the classifier will be tested and some of the collected data is for testing and the remaining data is for training purposes. Here those collected data are used to test and evaluate the performance of the model and the prototype. The relevance of the classified documents will be checked by experts on the domain based on the users and the experts' feedback; the result of the test will be evaluated by applying accuracy, recall, precision, or F-measure techniques.

1.6 Scope and limitation

The scope of this study on the designing a concept-based Amharic text classification system using a deep learning approach. In this study, different deep learning classification algorithms, such as convolutional neural networks, deep neural networks, or recurrent neural networks are used. The study is limited only to the classification of Amharic text items. HTML documents, image documents, and others that are not considered in this study. The study only used Amharic text data sets for the experimental purpose. It also considers only Amharic textual documents that contain a sequence of Amharic alphabets without any figure, table, images, or pictorial representations.

1.7 Significance of the study

Text classification is the activity of labeling natural language text data with relevant categories from a predefined set of class. It is also a process of extracting generic tags from unstructured text. These generic tags and labels come from a set of pre-defined classes. Classifying your content and products that help users to easily search and navigate within a website or application.

It can basically be used whenever there are certain tags to map to a large amount of text data. Especially in marketing, as it has moved from search engines to social media platforms where real communication between brands and users takes place. As marketing is charming more targeted, marketers are using personalization to energy better engagements.

Text classification can be applied in different ways of managing information, semantic-aware Amharic document categorizer plays an important role in a wide variety of information management tasks and the findings of this work can be used:

- In search engines to improve search results in the area of information retrieval.
- For filtering and categorizing documents for any interest group,
- To organize and improve browsing Amharic web documents.
- To automatically categorize documents for better management.

1.8 Organization of the thesis

The remaining part of the thesis is organized as follows. Chapter Two includes a literature review in which different concepts and approaches related to our thesis are presented. Moreover, text classification techniques, neural word embedding, Amharic language, and text document similarity measurement techniques are described. Chapter Three is about works related to our study that are previously done by other researchers in different natural language texts. Chapter Four deals with the design of our system, i.e., Semantic-aware Amharic text classification using deep learning approach. It presents the general architecture of the system with its basic components; the discussion of the components and their interaction within the system. The algorithms we developed for achieving the goal are presented. Chapter Five focuses on the detailed testing and evaluation of the system. It discusses the details about the testing and the results obtained together with their explanations. Conclusions are drawn from the thesis result, the contributions of this research work, and possible future works are presented in the last chapter.

Chapter 2: A literature review

2.1 Introduction

In this chapter, extensive literature reviews are conducted to get general information about the semantic representation of text documents, the writing system of Amharic language, and the different approaches of text classification techniques; the traditional machine learning, statistical and deep learning techniques. The tools and applications used for the text classification model design and relevant tools and libraries used for natural language processing tasks are reviewed. The extensive literature has been reviewed to understand the problem associated with the realm of this thesis and also to identify an appropriate solution.

2.2 Semantics

According to Euzenat [12] semantics "provides the rules for interpreting the syntax which does not provide the meaning directly but constrains the possible interpretations of what is declared". The classical text representation approach aims to numerically represent the unstructured text data to make them mathematically computable. With the rapid growth of text data mining research and information retrieval, semantic text representation is the leading approach in the area of NLP [13]. The problem is how to represent the text data by implicit or explicit semantics instead of word occurrence in the document. The term semantics refers to the meaning of language constructs, as opposed to their form or syntax. In NLP semantics or semantic similarities between words, sentences, paragraphs, or documents are extracted for different applications like text classification, sentiment analysis, and information retrieval.

The goals of semantic text representation are to improve the text classification, information retrieval, clustering, and other text mining problems' performance. Semantic similarity is a metric defined over a set of documents or terms, where the idea of the distance between them is based on the likeness of their meaning or semantic content as opposed to similarity which can be estimated regarding their syntactical and semantical representation. Thus, there are different approaches to extracting semantic similarity in textual documents; which are, ontology [8], word2vec [14],

doc2vec [15], FastText [16,17] and word embedding [18], are the common ones that are conducted in different pieces of literature.

2.3 Text classification

Text classification also termed as text categorization or text tagging or is the process of categorizing text into organized groups by using natural language processing (NLP) techniques, Text classifiers can automatically analyze text and then assign a set of pre-defined tags or categories based on its content [5]. In general, the text classification system contains four different levels of scope that can be applied [2]:

- Document-level: the algorithm obtains the relevant categories of a full document.
- Paragraph level: the algorithm obtains the relevant categories of a single paragraph or a portion of a document.
- Sentence level: obtains the relevant categories of a single sentence or a portion of a paragraph.
- Sub-sentence level: the algorithm obtains the relevant categories of sub-expressions within a sentence or a portion of a sentence.

The text classification systems can be deconstructed into the following four phases: Feature extraction, dimension reductions, classifier selection, and evaluations [2]

2.3.1 Feature extraction

Feature extraction is the process of selecting a subset of the terms occurring in the training set and using only this subset as features in the purpose text classification. Feature selection helps two main purposes. First, feature extraction makes training and applying a classifier more efficient by reducing the size of the effective vocabulary. Second, feature selection often increases classification accuracy by removing noise features and stop words [4]. There are two common methods of text feature extraction that are commonly used for text classification, which is the weighted word and word embedding techniques.

2.3.1.1 Weighted word

The most common form of weighted word feature extraction is TF, TF_IDF, and BOW, where each word is mapped to a number corresponding to the number of occurrences of that word in the

whole document or corpora. In all weighted word methods, each document is translated to a vector (with length equal to that of the document) containing the frequency of the words in that document. Although the approach is intuitive, it is limited by the fact that certain words that are commonly used in the language may direct such representations.

In NLP, the term frequency-inverse document frequency (TF_IDF), is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of text classification, information retrieval, and user modeling. The TF_IDF value increases correspondingly to the number of times a word seems in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the case in which some words appear more frequently in general. While term frequency (TF) indicates the occurrences of a term in a given document or corpus, in which it used to indicate the importance of the word in the given document by its occurrence as the term that is frequently occurred may be considered as a relevant term for the given document but terms in the different document may occur frequently with the same frequency and extracting their semantics to determine which are the representatives of which document.

Bag of words (BOW) also one of the methods to extract features of textual documents it is a reduced and simplified representation of a text document from selected parts of the text, based on the specific criteria, such as frequency of words. In CBoW, a document or a sentence is thought of like a bag of words or a collection of list words. Lists of words are created in the CBoW process and these words in a matrix are not sentences with structure sentences and grammar, and the semantic relationship between these words is ignored in their collection and construction. It has a scalability problem in which the structure or the sequence of words in the sentence or the document is not considered; which means the same words in different structure or context may be recognized as a single bag of words, simply it counts the number of unique words in the document and organizes those words as a bag of words.

2.3.1.2 Word embedding

Word embedding is a feature learning technique in which each word or phrase from the vocabulary is mapped to the N dimension vector of real numbers. It is another way to improve the performance of text classification based on deep neural networks by considering the semantics of documents. It is a method used in deep learning for the automatic processing of natural languages, based on the

representation of words in a large corpus by projecting a set of words of size k into a vector space of dimension M such that M < K, to facilitate the semantic analysis of words and to improve learning performance [20]. Word2Vec, doc2vec, FastText and Global Vectors (GloVe) [14] are currently among the most accurate and usable word embedding methods that can convert words into meaningful vectors and each dimension of the vectors generates from the model encodes a different aspect of words. These methods have achieved a lot of attention in text categorization because of their abilities to capture the syntactic and semantic relations among words. The use of word embedding for text classification tasks has become a standard approach. Deep learning-based text classification gives a vectored value to a word using word embedding or statistical methods.

Word2vec

Word2vec is a network of artificial neurons of two layers that can learn how to represent each word with a real number vector with its semantic features, thus it allows grouping similar words into a single vector [20]. It is a type of mapping that allows words with similar meaning to have similar vector representation.

The idea behind Word2vec is rather simple: use the surrounding words to represent the target words with a neural network whose hidden layer encodes the word representation. Word2vec, like doc2vec, belongs to the text preprocessing phase. Specifically, to the part that transforms a text into a row of numbers. The same idea of word2vec can be extended to documents where instead of learning feature representations for words, it learns for sentences or documents [4].

To get a general idea of a word2vec, think of it as a mathematical average of the word vector representations of all the words in the document. Doc2Vec extends the idea of word2vec, however words can only capture so much, there are times when we need relationships between documents and not just words. The authors of [21] Presented "word to vector" representation as an improved word embedding architecture. The Word2Vec approach uses shallow neural networks with two hidden layers, continuous bag-of-words (CBOW), and the Skip-gram model to create a high dimension vector for each word, the feature vectors is part of a wider concept in machine learning. Such representations encapsulate different relations between words, like antonyms, synonyms, or analogies.

Doc2vec

A Document Vector [9] is an extension of the Word Vector, which represents the whole of a document in a single digital vector to easily identify the similarity between the documents. It can learn the representation of documents through two models, namely distributed bag of words, which is equivalent to Skip-Gram in Word2vec, and distributed memory, which is equivalent to a continuous bag of words [20]. Distributed Bag of Words (DBOW) randomly predicts a probability distribution of words in a document from the identifier of a document, to create its vector. It does not take into consideration the order of words. During training, the document vector and word weights are randomly initialized and updated using the stochastic gradient descent. Paragraph Vector-Distributed Memory unlike distributed bag of word, predicts a word from the context of the document. It takes a set of words of the implementation needs to create two classes, one to label the documents for training and the other one for the preprocessing [4].

The goal of doc2vec is to produce and create a numeric and vector representation of a document, regardless of its length. But unlike words, documents do not come in logical structures like words. A word vector is generated for each word, and a document vector is generated for each document. The model also trains weights for a SoftMax activation on the hidden layer. In the inference stage, a new document may be appeared, and all weights of the document are fixed to calculate the document vector.

FastText

FastText is one of the most important tools of word embedding that is used to represent words and sub-words of a document in the numerical form of the vector by considering the morphology of words. Other word embedding representations ignore the morphology of words by assigning a distinct vector to each word [18]. Facebook artificial intelligence research lab released a novel technique to solve this issue by introducing a new word embedding method called FastText. Each word is represented as a bag of character n-gram. In contrast, it creates a vector for each character n-gram. The word vector is then calculated as the average of its characters of n-gram vectors [16] Therefore, even an out-of-vocabulary word gets assigned a vector based on its sub-word units. This is even most important for inflected languages since some inflected forms of words are rare and may not even contained in the training data. Because of its simplicity and efficient realization,

training the FastText embedding is faster than most other alternatives. The simplicity of FastText makes it very conducive for formal analysis. Despite its simplicity, it combines several very important techniques: a bag of words (BoW), representation of words as vectors of linear space, and linear classification [18]

2.3.2 Dimensionality reduction

Dimensionality reduction is an important step in text mining. It improves the performance of clustering and classification techniques by reducing dimensions so that text mining procedures process data with a reduced number of terms. High-dimensional data present many mathematical challenges and computations as well as some opportunities and are bound to give rise to new theoretical developments. One of the problems with high-dimensional datasets is that, in many cases, non-importance of all measured variables for understanding the underlying phenomena of interest. While different computationally expensive novel methods can construct predictive models with high performance or accuracy from high-dimensional dataset, it is still of interest in many applications to reduce the dimension of the original data before any modeling of the data [17]. The dimension of the data is the number of variables that are measured on each observation and singular value decomposition is a technique used to reduce the dimension of a vector. Term-based vector models consist of many features of text sequences. Thus, memory consumption and time complexity are very expensive for these methods. To address this issue, many researchers use and propose many dimensionality reduction techniques to reduce the size of feature space [3]. Such techniques are Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and non-negative matrix factorization (NMF).

2.3.3 Classification Techniques

Classification techniques could be divided into machine learning (ML) and statistical approaches. Statistical techniques purely satisfy the proclaimed hypotheses manually, thus the need for algorithms is little, but machine learning techniques were specially invented for automation [22]. In general, there are different text classification techniques that are assessed under the two categories including, the two popular techniques in ensemble learning algorithms such as, Boosting and bagging. Some other methods, such as Naïve Bayes, logistic regression, and k-nearest neighbor, are more traditional but still commonly used in the scientific community. Tree-based classification algorithms, such as decision trees and random forests are fast and accurate for

document categorization. Support vector machines, especially kernel SVM, are also widely used as a classification technique. The deep learning; neural network-based algorithms such as deep neural networks (DNN), convolutional neural network (CNN), recurrent neural network (RNN), deep belief network (DBN), hierarchical attention networks (HAN), and combination techniques [2] are the major classification techniques in many natural language processing applications.

A. Statistical approaches

Statistical techniques are innocently mathematical processes, and they act as the mathematical foundation for all other text classifiers. It works similar to a computer program, executing the given instructions without any ability of its own [23]. Naive Bayes is a family of statistical algorithms that we can use for text classification. Among other statistical classifiers [24], the Bayesian classifier is the simplest, but it is still very effective and due to its simplicity, it is also the most researched classifier. Naïve Bayesian classifier assumes that features of the input feature vector usually word distributions are statistically independent.

B. Machine learning approaches

The ability to learn from past observations and the increase in data volume, variety, and velocity leads to automation in text processing techniques including text classification. In some situations, defining a set of logical rules using knowledge-engineering techniques and based on expert opinions to classify documents helps to automate the classification task [24]. In machine learning approaches of text classification, there are traditional machine learning algorithms; such as logistic regression, support vector machine, decision tree, random forest tree, rule induction, K-nearest neighbor, K-means and the Hebbian algorithms are the common one. Whereas the newly emerged and highly applied approaches, the deep learning approaches also include the CNN, DNN and the RNN are the basic and highly adapted techniques of text classification.

KNN: k-nearest neighbor: The most common and widely used distance function for the KNN classifier is the Euclidean distance formula and it is used to calculate the distance between the new unlabeled data point and the training data points. The main step in the classification stage of the KNN is to measure the distance to identify the nearest neighbors of the new input data point.

Naive Bayesian: it is a simple classification method based on the Bayes rule with the dimensionality of the inputs is high. In this Bayes rule applied to documents and classes. This

model gives class labels to problem instances, represented as vectors of feature values. The value of a particular feature is autonomous of the value of another feature. Prior probability is known and posterior probability is checked in naive Bayesian technique. The class with a higher posterior probability assigned to the document.

SVM: Support Vector Machine: is an impressive approach for classifying high-dimensional data with the use of the Structural Risk Minimization (SRM) principle. It has been conveyed as a discriminative classifier which is further accurate than most former classification prototypes. SVM acquires the optimal hyperplane that splits training data points from different classes by maximizing the classification margin. Similarly, SVM applicable to data points with nonlinear decision surfaces by engaging a system identified as the kernel method that designs the input data to a higher dimensional feature space, where a linear separating hyperplane can be launch.

Decision Tree: decision tree shapes classification mockups in the usage of a tree structure. The aim is to construct a model that predicts the value of a target variable based on input variables. In these tree structures, leaves characterize class labels and branches denote features of class labels. Its disturbances a dataset into smaller subsets while at the identical time a decision tree is incrementally established. The ultimate result is a tree with decision nodes and leaf nodes. It can handle both numerical data and categorical.

Neural Network: A neural network classifier is a network of units, where the input units show terms and output units speak for the category. Each unit receives a set of inputs. There are two types of separators according to classes that are single linear perception and multiple perceptions. In single-layer classes are not separated. Multiple layers are used to find non-linear classification boundaries. In the neural network approach deep learning that contains CNN, RNN and DNN is common for text classification.

2.3.4 Evaluation metrics

Evaluation metrics are measurements that used to measure the performance of the model or the algorithm. A number of performance metrics are used to evaluate the major machine learning algorithms. Such as [4] , Accuracy, precision, recall, and f-score, can be used for sorting algorithms primarily used by search engines and other classification techniques. Choosing the appropriate

metric to evaluate the model is very important. Choice of evaluation metrics influences how the performance of machine learning algorithms is compared and measured.

Confusion Matrix: The confusion matrix is one of the easiest and intuitive metrics used for finding the accuracy and correctness of the model and algorithms. It is used for the classification problem where the output can be of two or more types of classes.

The terms that are associated with confusion matric to determine the performance of the model are; True positive (TP), the cases when the actual class of the data point was true and the predicted is also True. False positive (FP), are the cases when predicted is true and the actual class of the data point was False. The False is because the model has predicted incorrectly and positive because the class predicted was a positive one. True negative (TN), are the cases when the predicted is also False and the actual class of the data point was False. False negative (FN), are the cases when the predicted is False and the actual class of the data point was True.

Accuracy: Accuracy in classification problems is the number of correct predictions made by the model over all kinds of predictions made. It can be determined as the ratio between the sum of TP and TN with the sum of all the terms that describe the confusion matrix, TP, TN, FP, FN. All of the evaluation metrics are not suitable for every type or aspect of classification. Thus, we use accuracy to measure the performance of classification models when the target variable classes in the data are nearly balanced. Accuracy should not be used as a measure when the target variable classes in the data are a majority of one class.

Precision: It is a measure that tells us what proportion of data's that we classify as being in a specific class and had in that class. The predicted positives (data predicted as a specific class are TP and FP) and the data being in a class are TP. Thus, precision is the ratio between the TP with the sum of TP and FP.

Recall: recall gives us information about a classifier's performance concerning false negatives (how many did we miss), it is determined us the ratio between the TP with the sum of TP and TN. while precision gives us information about its performance concerning false positives (how many did we caught).

F1 Score: We don’t want to carry both precision and recall in our pockets every time we make a model for solving a classification problem. So, it is best if we can get a single score that kind of represents both precision and Recall. It is also determined simply by taking the arithmetic mean of precision and recall.

2.4 Deep learning

Deep learning is part of machine learning that can utilize either unsupervised or supervised algorithms or both. While it's not essentially new, it has recently seen a surge in popularity as a way to improve the solution of certain types of difficult computer problems, most notably in the natural language processing and computer vision fields. It is based on the feature learning or representation learning branch of machine learning theory. By extracting high-level, complex abstractions as data representations through a hierarchical learning process, deep learning models yield results more quickly than standard machine learning approaches [25].

A deep learning model will learn the features that are important automatically by using less engineered feature, instead of requiring the data scientist to manually extract the important features. It is so popular today due to two main reasons [25]. First, it was discovered that CNNs run much faster on GPUs, such as NVidia's Tesla K80 processor. Secondly, data scientists realized that the huge stockpiles of data benign collected can serve as a massive training corpus and thereby supercharge the CNNs into yielding a substantial improvement in the accuracy of natural language processing and computer vision algorithms. Deep learning-based representation learning involves a hierarchy of features or concepts, where the high-level concepts can be defined from the low-level ones and low-level concepts can be defined from high-level ones.

In some articles, DL has been described as a universal learning approach that can solve almost all kinds of problems in different application domains. In other words, DL is not task-specific [26]. Thus, Text classification also one of the natural language processing applications that get advantages from deep learning through its ability to extract features automatically and the capacity to create non -linear and complex relationships within the data.

2.4.1 Types of Deep Learning Approach

Deep learning approaches can be categorized into supervised, semi-supervised, unsupervised, and reinforcement learning [26].

Supervised learning

It is a learning technique that uses labeled dataset. In the case of supervised DL approaches, the environment has a full set of labeled inputs and corresponding outputs are expected by following different learning algorithms [26]. Fully labeled means that each data in the training dataset is tagged with the answer, the algorithm should come up with its inference. Thus, it is best suited to problems where there is a set of available reference points or a ground truth with which to train the algorithm. There are a number of supervised learning approaches for deep leaning, including Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), including Long Short Term Memory (LSTM), and Gated Recurrent Units (GRU) [26] Thus, the use of those approaches may differ in the applications used and most of the time convolutional neural network and recurrent neural networks (LSTM) are used for text classification.

Semi-supervised Learning

Semi-supervised learning is a learning process that occurs based on partially labeled datasets. It is, for most of the applications in the case when we have a training dataset with both labeled and unlabeled data. This method is particularly useful when labeling examples is a time-intensive task for experts, and extracting relevant features from the data is difficult. It is especially useful for medical images, where a small amount of labeled data can lead to a significant improvement in accuracy [27]. In some cases, deep reinforcement learning and Generative Adversarial Networks (GAN) are used as semi-supervised learning techniques. Additionally, RNN, including LSTM and GRU, is used for semi-supervised learning as well.

Unsupervised learning

Unsupervised learning systems are ones that can operate without the presence of data labels. In this case, the agent learns the internal representation or important features to discover unknown relationships or structures within the input data. Often clustering, dimensionality reduction, and generative techniques are considered unsupervised learning approaches. There are different approaches to deep learning [26, 28] Auto-Encoders (AE), Restricted Boltzmann Machines (RBM), and RNNs, such as LSTM used for unsupervised learning in many application domains. In unsupervised learning, a deep learning model is supplied with a dataset without explicit instructions on what to do with it. The training dataset is a collection of inputs without a specific desired output or correct answer. The neural network then endeavors to automatically find structure in the data by extracting useful features and analyzing its structure

Reinforcement learning

It is a learning technique for use in unknown environments. In reinforcement learning, there is no a straight forward loss function, thus making learning harder compare to traditional supervised approaches [26]. Thus, the fundamentals are: First, you do not have full access to the function you are trying to optimize or use for training as well as testing so you must query them through interaction. Second, you are interacting with a state-based environment. In this deep learning, the agents are attempting to find the optimal way to accomplish a particular goal or improve performance on a specific task. As the agent takes action that yield toward the goal, it receives reward. The overall aim is to predict the best next step to take to earn the biggest final reward [27]. To make its choices, the agent relies on both learnings from past feedback and exploration of new tactics that may present a larger payoff. This take in a long-term strategy just as the best immediate move in a chess game may not help you win in the long run; the agent tries to increase the final reward.

2.4.2 Deep Learning architectures for Text Classification

Deep learning models and architectures have achieved state-of-the-art results across many domains, including a wide variety of NLP applications. Deep learning for text or document classification includes three basic architectures of deep learning in parallel [2].

Deep neural network (DNN)

Deep neural networks (DNN) are designed to learn by multi-connection of layers that every single layer only receives the connection from previous and provides connections only to the next layer in a hidden part [28]. The input consists of the connection of the input feature space with the first hidden layer of the DNN. The input layer may be constructed via word embedding, TF*IDF, or some other feature extraction method. The output layer is equal to one for binary classification or the number of classes for multi-class classification. In multi-class DNNs, each learning model is generated or the numbers of nodes in each layer as well as the number of layers are randomly assigned. The implementation of Deep Neural Network (DNN) is a discriminatively trained model that uses the standard back-propagation algorithm and sigmoid or ReLU as activation functions. The output layer for multi-class classification should use SoftMax [4]

Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNN) is another neural network architecture that is addressed by the researchers for text mining and classification [29]. It assigns more weights to the previous data points of sequence of processing. Therefore, it is a powerful method for text, string, and sequential data classification. In RNN, the neural net considers the information of previous nodes in a very sophisticated method which allows for better semantic analysis of the structures in the dataset. RNN mostly works by using LSTM or GRU for text classification [2].

Long Short-Term Memory (LSTM)

LSTM is a special type of RNN that addresses problems by preserving long term dependency more expertly in comparison to the basic RNN. It is particularly useful for overcoming the vanishing gradient problem [30]. Although it has a chain-like structure similar to RNN, LSTM uses multiple gates to carefully regulate the amount of information that is allowed into each node state. LSTM contains three different gates and one cell state that regulate the network and store long term and relevant information.

Forget gate

It is the first gate in the LSTM network that is used to decide the status of the information's to be kept or forgotten. Thus, information from the previous hidden state (h_{t-1}) and information from the current output (X_t) composed together and pass through the first activation function, and then the forget gate (f_t) is formed. When those information's are pass through the SoftMax activation the value lays between 0 and 1, values closer to 0 implies the information is not important to form the next hidden state and it becomes forgotten and values closer to 1 implies the information is relevant to form the next hidden state and it becomes kept.

Input gate

It is the next gate in LSTM that is used to update the cell state using the previous hidden state and the current input in advance with the tanh activation function that is used to regulate value pass through the network by squishing the value between -1 and 1. Thus, first, the previous hidden state and the current input pass through the SoftMax activation and the information kept to update is decided and then after the previous hidden state and the current input pass through the tanh activation to regulate the network and finally, the output of the SoftMax activation is multiplied by the output of the than activation. The SoftMax output determines which information from the tanh function needs to be kept and updated.

Cell state

It is the memory of the network that stores relevant and long-term information. thus, the previous cell state and the forget vector forms a pointwise multiplication to create the possibility of removing some values in the cell state if it is probably multiplied by zero. Then after the output of the input gate and apply pointwise addition to updates the cell state to new values that the neural network finds important.

Output gate

It is the final gate of LSTM that decides what the next hidden state, which contains information from the previous hidden state and used for prediction should be. Thus, first the previous hidden state and the current output pass through the SoftMax activation, and the updated cell state pass through the tanh activation function. Then after the pointwise multiplication is performed on the

SoftMax output and the tanh output to decide what information the new hidden state contains. the final output of the output gate is a new hidden state, the new hidden state, and the new cell state is carried over the next time step of the LSTM sequence.

Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) is a gating mechanism for RNN which was introduced by [31] it is a simplified variant of the LSTM architecture, but there are differences as follows: GRU contains two gates and does not possess any internal memory, and non-linearity is not applied. GRU is now popular in the community who is working with recurrent networks. The main reason for the popularity is the computation cost and simplicity of the model [2].

Convolutional neural network

A convolutional neural network (CNN) is a deep learning architecture that is commonly used for hierarchical document classification [2] [32]. These convolution layers can be stacked to provide multiple filters on the input, and are called feature maps. To reduce the computational complexity, CNN uses pooling to reduce the size of the output from one layer to the next in the network. Different pooling techniques are used to reduce outputs while preserving important features [3]. The most common pooling technique is max pooling where the maximum element in the pooling window is selected. To forage the pooled output from stacked featured maps to the next layer, the maps are firmed into one column. The final layers on CNN are typically fully connected. In general, during the back-propagation step of a convolutional neural network, both the weights and the feature detector filters are adjusted. Although it is originally built for image processing with architecture similar to the visual cortex, CNNs have also been effectively used for text classification [4].

The CNN architecture consists of a combination of three types of layers: Convolution, max-pooling, and classification layers [26]. There are also two types of layers in the low and middle-level of the network: Convolutional layers and max-pooling layers in which the even-numbered layers are for convolutions and the odd-numbered layers are for max-pooling operations. The output nodes of the max-pooling and convolution layers are grouped into a 2D plane called feature mapping. Each plane of a layer is usually resulting from the combination of one or more planes of

previous layers. The nodes of a plane are connected to a small region of each connected plane of the previous layer.

2.4.3 Development Tools of Deep Learning for Text Classification

There is a different development tool of deep learning for natural language processing that is open source and easily adapted for different applications. Those tools are implemented using different programming languages; Java, JavaScript, Python, R programming, C++, etc. Since Python programming language is one of the best suited for data processing, many tools and libraries are written for it. Solutions like Jupiter and other data visualization tools are written in Python, and many other software instruments provide native Python functionality or support through APIs or various wrappers [4]. This is the reason there are a number of NLP libraries out there, and many more come into service regularly.

TensorFlow: It is an end-to-end open-source platform for machine learning. It has community resources that lets researchers push the state-of-the-art in ML and a comprehensive, libraries, flexible ecosystem of tools, and developers easily build and deploy ML-powered applications. TensorFlow offers multiple levels of abstraction so we can choose the right one for our needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy. If we need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large machine learning training tasks, use the distribution strategy API for distributed training on different hardware configurations without changing the model definition.

Keras

It is a high-level neural networks API, capable of running on top of TensorFlow, CNTK, or Theano and written in Python programming. It was developed with a focus to accelerate the experimentation. The key to doing good research is being able to go from idea to result with the least possible delay is. Keras is used if we need a deep learning library that:

- Allows for easy and fast prototyping (through user-friendliness, modularity, and extensibility).
- Supports convolutional networks and recurrent networks, as well as combinations of the two.

- Runs seamlessly on CPU and GPU.

Keras has its guiding principles which are:

User-friendliness: It follows best practices for reducing cognitive load, it offers consistent and simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.

Modularity: A model is understood as a graph or a sequence of standalone, fully-configurable modules that can be plugged together with as few restrictions as possible. In particular, neural layers, optimizers, cost functions, activation functions, initialization schemes, and regularization schemes are all standalone modules that you can combine to create new models.

Easy extensibility: New modules are simple to add as new classes and functions, and existing modules provide ample examples. To create new modules that allows for total expressiveness, making it suitable for advanced research.

Work with Python: No separate models configuration files in a declarative format. Models are described in Python code, which is easier to debug, compact, and allows for ease of extensibility.

NLTK

NLTK stands for Natural Language Toolkit and it is the best solution for learning the chains of the NLP domain. Its modular structure helps to get the firsthand experience with composing appropriate models for solving certain tasks and to comprehend the dependencies between components. Since its release, NLTK has helped solve multiple problems in various aspects of Natural Language Processing. There are multiple guides, the most useful being this book and this tutorial that will help anybody master the NLTK. Truth be told, doing it otherwise is not advised, as this is quite a complicated solution with a harsh learning curve and a mess of internal limitations. However, NLTK can become the excellent playground of the text analysis researcher.

TextBlob

TextBlob is an interface for NLTK that turns text processing into a simple and quite enjoyable process, as it has rich functionality and a smooth learning curve due to detailed and understandable documentation. TextBlob allows simple addition of various components like sentiment analyzers

and other convenient tools. It can be used for the rapid prototyping of various NLP models and can easily grow into full-scale projects.

Genism

Genism is a library for document similarity analysis. While it can be not as ubiquitous and all-around capable as the previous components, there is an area where it shines. This area is the document similarity comparison, and the topic modeling, highly-specialized Genism library has no equals there. Offering the tools like LDA (or Latent Dirichlet Allocation), scalable and robust, Genism is a production-ready tool you can trust with several crucial components of your NLP projects, not to mention topic modeling being one of the most engaging and promising fields of the modern NLP science

2.5 Amharic Language

Amharic (also called as Amarigna, Amarinya) is a Semitic language spoken in Ethiopia. Amharic is rooted in the Ancient language Geez (Ge'ez). The official language of Ethiopia before Amharic was discovered from Geez. Although Geez is ceased to be spoken popularly sometime between 900 and 1200 AD, it continues as a language of the Ethiopian Church. Since the 13th century, Amharic has been the language of the court and the population in the Highlands of Ethiopia. Amharic is the second-most spoken Semitic language in the world, after Arabic [33].

Amharic is written using Fidel, ፊደል, which grew out of the Geez. Amharic is a syllabic language that uses a script that initiated from the Ge'ez alphabet. It has 33 basic characters, or groups with each having 7 forms, or families for each consonant-vowel combination. Unlike Hebrew, Arabic, or Syrian, the language is written from left to right. In the syllabic system, it is possible only to point out to the phonemes but not the vowels and consonants that comprise it. This is possible only in what is called the alphabetic system [34]. The typical clause order of Amharic language is noun + object + verb.

Nouns may denote gender, number, definiteness, case, and direct object status by affixes prefixes and suffixes, predominately suffixes. The noun form of Amharic may have a masculine or feminine gender. Suffixes are added to denote a feminine or masculine noun gender. Some nouns may have both masculine genders, while other nouns may only have one gender. The feminine

gender is used to indicate female and smallness. Plurals are indicated by the suffix. The order in which affixes are added: gender, number, definiteness, case, and direct object status.

Pronouns: Amharic is a pro-drop language. Sentences with no highlighted element do not require independent pronouns. The verb denotes the person, number, and gender. Object pronoun suffixes indicate number, person, and gender of the object of a verb, and are affixed to verbs. to indicate possession, possessive suffixes are affixed to nouns.

Verbs are derived from affixes and roots to inflect person, number, gender, aspect, mood, voice, and polarity are added. it agrees with their subjects. Verb agreement with objects is optional. it is placed at the end of the sentence. *Adjectives* are predominately derived from verbs, nouns, and other parts of speech.

Unlike English, Amharic is a morphologically complex language Although Amharic is morphologically complex, it could represent a text in fewer words than English and other languages. In contrast, geez is a suitable language to analyze its morphology and phonology [35] Unlike the English language, in Amharic one symbol or character represents both a consonant and a vowel. The six orders are formed by attaching diacritic markings to the basic symbol to combine consonants with vowels. For instance, by attaching the "å" marker to the basic symbol "ሀ" the second-order "ሁ" symbol is generated. The Ethiopic script is a syllabary rather than an alphabet because it doesn't have separate characters for vowels unlike alphabets in other languages but for the sake of convenience, it is named as an alphabet [35].

Amharic is written from left to right with its writing system, the typical clause order noun + object + verb [34]. This order is different from the noun + verb+ object order of the English language. Amharic has inflectional morphological structures, which require a complex morpheme analyzer for morpheme generation and word formation as well.

In the Amharic language, there are different graphemes in which users use in writing interchangeably. For instance, {ሀ, ሐ, ኀ} representing the {h} sound and {ጽ, ፅ} that represent the {ts} sound. Because of this, Amharic text data can contain features that affect processing. Among those, one is the orthographic variation (for one meaningful entity, the same word can be written differently in Amharic text data), like ኀይለ ሥላሴ, ሃይለ ስላሤ, ኀይለ ሥላሤ, ኃይለ ሥላሤ, ሀይለ ሥላሤ, ሃይለ ሥላሤ, ሐይለ ስላሴ, ሐይለ ስላሤ, ሐይለ ሥላሴ ኃይለ ስላሴ, ኀይለ ስላሴ, ኀይለ ስላሤ, ኃይለ ስላሤ, ሀይለ ስላሴ, ሀይለ ስላሤ, ሀይለ ሥላሴ, ሃይለ ስላሴ, ሃይለ ሥላሴ,, refers to emperor Haile Selassie (አፄ ኀይለ ሥላሴ); ማህደር, ማሕደር, ማኀደር for Mahedir; ድረ ገጽ, ድህረ ገጽ [36] , etc. Amharic texts which refer to the same meaningful element can be written in abbreviation or normal form, example ጠቅላይ ሚኒስትር, ጠ/ሚኒስትር, ጠ .ሚኒስትር for Prime Minster and መስርያ ቤት, መ/ቤት for work office, etc. Besides, for writing numbers in Amharic text data, users might use either the Geez format '፩','፪', '፫'or normal Arabic format '1','2','3', for example, መስከረም ፩ and መስከረም 1. Similarly hyphenated Amharic texts can be written like ወዝ-አደር, ወዝ አደር, ቤተ-እስራኤል, ቤተ እስራኤል.

Chapter 3: Related work

3.1 Introduction

In this chapter, we reviewed the literature which is published in journals and in conferences proceeding that focus on the area that we study specifically on the semantic meaning extraction; word2vec, doc2vec, FastText, and ontology. In addition to the semantic meaning, we review the deep learning architectures and approaches for the application of natural language processing which is text classification. The nature and the writing system of Amharic language are also reviewed because the target language that we make our experiment is Amharic. Finally, we summarize the review as describing the limitation as well as the strength of those papers and the general knowledge that we get from it and the methods that we adapt to follow are reviewed and stated. Those literature may or may not apply for Amharic languages. But our main target is that assessing and reviewing papers that are related to our domain knowledge methods that we use and the model that we develop for our final proposed solution.

3.2 Text Classification for non-Ethiopian Languages

Joulin, et al [37] explored ways to scale the baselines to a very large corpus with a large output space, in the context of text classification. Their experiments show that their text classifier, FastText, is often one part of deep learning classifiers in terms of accuracy, and many orders of magnitude faster for training and evaluation. They trained FastText on more than one billion words in less than ten minutes using a standard multicore CPU, and classify half a million sentences among 312K classes in less than a minute. A good understanding of simple linear classification algorithms such as FastText is a key to constructing good nonlinear text classifiers, that was the good motivation for analyzing the FastText classification algorithm. On the other hand, the simplicity of FastText makes it very conducive for formal analysis. Despite its simplicity, FastText combines several very important techniques: a bag of words (BoW), representation of words as vectors of linear space, and linear classification. In this paper, the ability to extract the nonlinear relationship between data and its programing simplicity as well as its good performance on both large and small amounts of training data is the good feature of using FastText.

Goto, et.al [38] proposed an approach that the use of Word2Vec can be used as an input for deep learning in categorizing web news. They have to search for target news in some categories since each news site has its categorization policy. They categorize web news in this research using machine learning. For the analysis, they used Japanese text data delivered by Japanese web sites. They used bag-of-words which is a method of vectored representation of the word and it is often used as an input for text classification. Although the method is good for categorization because of higher accuracy, it has a problem in computational complexity in using a neural network. Through the experiment, they found that it is practical to express words using Word2Vec as an input of deep learning for the categorization document of web news. The number of collected news data is 1,728,942 records, and the number of vocabularies in datasets is 218,295. They labeled 800 news data of the full datasets, and 600 of these are used to learn, the other data are used for the test. They used Pylearn2 to conduct deep learning. Pylearn2 is a tool for carrying out deep learning in the python. Using word2vec also has limitation which is like inability to handle unknown or out of vocabulary words and no shared representations at sub-word levels which is solved using the FastText approach.

Hayati, et.al [39] proposed an automatic system to assess learners' cognitive presence regarding their social interactions within asynchronous online discussions by combining the Doc2Vec document embedding method, natural language preprocessing, and machine learning techniques, They first made some pre-processing and transformations to the given transcripts, then they applied Doc2Vec method to represent each content as a vector that is concatenated with Linguistic Inquiry and Word Count (LIWC) and context features. The vectors are input data of the Naïve Bayes algorithm; a machine learning method; that aims to classify transcripts according to cognitive presence categories. They also conduct some comparisons to evaluate the semantic meaning of the extract using different approaches. Thus, they compare CBOW, word2vec, and doc2vec and finically they conclude as BOW-based meaning extraction ignores the semantic representation and words ordering and Word2Vec has proven its effectiveness in a word-level representation. Therefore, they adopted the Doc2Vec method since it can extract surrounding word vectors that are specific to a document, in their case the hall message, unit of analysis. The performance of their classifier is evaluated and the obtained classification accuracy is 77%.

Kralj, et.al [40] proposed semantic-aware recurrent deep neural architecture (SRNA) that enables the system to learn simultaneously from the raw text documents and semantic vectors. They tested the effectiveness of the approach on three text classification tasks: sentiment analysis, news topic categorization, and gender profiling. The experiments show that the proposed approach outperforms the approach without semantic knowledge, with the highest accuracy achieved on short document fragments. They compared the result of the classifier in different machine learning algorithms like the support vector machine, random forest tree, recurrent neural network, and convolutional neural network, The conclusion is that the convolutional neural network and recurrent neural network outperforms than support vector machine and random forest tree while the amount of training data increases. The semantic information of the document is extracted using taxonomy and ontology development for their experimentation. We will used recurrent neural network that outperform and the word embedding to extract semantic feature extraction. Finally, the result of their study indicates that recurrent neural network outperforms other approaches in text classification on their dataset and it can end by getting benefits from the addition of the semantic information. They also achieved 80-85% accuracy on average for those different learning algorithms.

Govardha [1] proposed a model that enables to capture the terms that indicates the concepts in the text and that identifies the topic of the document. They have introduced the new concept-based model which analyzes the terms on the documents and sentence level. This concept-based model effectively discriminates between terms that hold the concepts that represent the sentence meaning and non-important terms concerning sentence semantics. It is also done for Indian Telugu documents. Their classification model is based on an adaptation of the classic vector-space model. They use their ontology as a classifier rather than using any of the machine learning approaches and machine learning approaches like deep learning enables an improved model and improve the performance as well as the accuracy of the classifier while the amount of data or document is increased. Since the ontology is developed in a specific domain to make it generic and more suitable for huge amounts of data deep learning approach is advisable [2].

Allahyari and Kochut [8] proposed a method for the automatic classification of text documents into a dynamically defined set of topics of interest. The proposed approach requires only ontology and a set of user-defined classification topics, specified as contexts in the ontology. Their method

is based on measuring the semantic similarity of the thematic graph created from a text document and the ontology sub-graphs resulting from the projection of the defined contexts. The ontology effectively becomes the classifier, where classification topics are expressed using defined ontological contexts and the proposed method doesn't require a training set of documents. In their experiments, they used the English language Wikipedia converted to an RDF ontology to categorize a corpus of current Web news documents into the selection of topics of interest.

Wei, et al. [7] reported their preliminary studies using deep learning in legal document review. Specifically, they conducted experiments to compare deep learning results with results obtained using an SVM algorithm on the four datasets of real legal matters. Their results showed that CNN performed better with a larger volume of training datasets and should be a fit method in the text classification in the legal industry. They also investigated convolution neural network provides an effective way of text classification by learning the text in sequences compared to a bag of word methods that lacks the sequence information, thus better extract features in terms of sentences/phrases, the challenge lies in the actual identification of relevant sentence, as legal documents are often identified as relevant due to a few sentences or short passages. The CNN model outperforms SVM with a large volume of the training dataset.

Gawade et al. [12] proposed a text document classifier by using WordNet ontology and neural network. The main problem is to classify which documents are irrelevant and which are relevant. They proposed a method of automatic text classification using Convolutional Neural Network based on the disambiguation of the meaning of the word they used the WordNet ontology and word embedding algorithm to eliminate the ambiguity of words so that each word is replaced by its meaning in a suitable context. In their paper, an automatic text classification system has been proposed for classifying the bulk documents in the database based on the topic label.

Zhou and Gohary [13] proposed ontology-based multi-label text classification for enhanced information retrieval for supporting automated environmental compliance checking. To fully automate the environmental regulatory compliance checking process, they automatically extract the rules from applicable environmental regulatory textual documents, such as energy conservation codes. In their automated compliance checking approach, before rule extraction, they first classify the text into pre-defined categories to only retrieve relevant clauses and filter out irrelevant ones, thereby improving the efficiency and accuracy of rule extraction. Then they apply a deep learning

algorithm (word2vec) to learn the similarities between each clause and each topic for classifying each clause into zero or more topics and to capture ontological relationships like is-a relationship.

3.3 Text Classification for Ethiopian Languages

Meron Sahilemaryam [3] proposed a framework that automatically categorizes Amharic documents into predefined categories using knowledge represented in the news ontology. The heart of the classification system is the knowledge base that enables the representation of different domain concepts. They show that the use of concepts for the Amharic document categorizer results in 92.9% accuracy which is a promising outcome. During the classification process, all the documents pass through pre-processing stages. Then index terms are extracted from a given document which is mapped onto their corresponding concepts in the ontology. Finally, the selected document is classified into a predefined category, based on the weighted concept. Here, the ontology is limited only in news documents and they don't use any of the machine learning approaches that enable improved accuracy and performance for the classifier.

Alemu Kumlachew [10] proposed Automatic text classification that focuses on the flat classification, where each topic is treated as a separate class, which is inadequate in text classification where there are a huge number of relevant features and a large number of classes and needed to distinguish between them. They achieved the maximum accuracy of 90.37% at level-3(last level) of the category tree. Moreover, at an increasing number of top feature set, the accuracy of the hierarchical classifiers increases compared to the flat classifier. The peak accuracy was 89.06% using level three classifiers when the top 15 features were used. Moreover, at an increasing number of top feature set, the accuracy of the flat classifier decreases. So, the actual result of this paper shows that the proposed solution is not suitable for huge number of documents and a higher number of classes and features and it results in a decrease in accuracy and performance. They also used a support vector machine (SVM) to see the performance of the classifier and SVM has a limitation of lack of transparency in results caused by a high number of dimensions (especially for text data), choosing an efficient kernel function is difficult (susceptible to overfitting or training issues depending on the kernel) and Memory complexity [3].

Worku Kelemework [11] proposed an automatic Amharic news classification that works based on one of the neural networks learning methods called Learning Vector Quantization (LVQ), to see its suitability for Automatic Amharic text news classification. And also, they used two weighting schemes, Term Frequency (TF) and Term Frequency by Inverse Document Frequency (TF*IDF), which are used to weight the features in news data to construct news by features matrix, which is the input to the learning algorithm. For similar experiments, the application of the TF*IDF weight method resulted in 69.63%, 78.22%, and 68.03% accuracies with an average of 71.96% accuracy. The overall result of this paper shows that they try to use a neural network algorithm to classify the texts and document weighting to extract features from their data set for extracting features. So, they consider the term frequency and document frequency rather than the meaning or the concept that the document contains.

3.4 Summary

The aforementioned related works have their strength and weakness for text classification. Some of the works follow key-word based text classification with and without considering the conceptual meaning of the document and using different traditional machine learning algorithms to propose a model and train their dataset for classifying. From the reviews of the related works, we learned that deep learning performs better than traditional machine learning algorithms. Furthermore, we can also see that the use of semantics enhances the accuracy of text classification. Accordingly, we propose an Amharic text classification that uses semantics and a deep learning approach for better performance and accuracy. We also compare the performance of the text classification system using deep learning with other traditional machine learning algorithms in different pieces of literature.

Chapter 4: Design of Semantic-aware Amharic Text Classifier

4.1 Introduction

In this chapter, the discussion is on the designing components and the structure of the Semantic-aware Amharic text classification model using a deep learning approach in detail. The proposed model architecture which is depicted in *Figure 4.1, is* composed of five (5) major components.

The preprocessing component contains its subpart which is stop-word removal, normalization, tokenization and padding. To do those activities, unstructured Amharic text documents are collected from different sources and those data are preprocessed to be an input for the rest of the components. The model building and training component contains the word embedding, deep network building, output determination, compiling and classification component.

4.2 Architecture of the proposed solution

We propose Semantic-aware Amharic text classification using the deep learning approaches. It contains two phases the training phase and the testing phase. The flow and the order in which those components are organized and form the system are clearly shown in *Figure 4.1*.

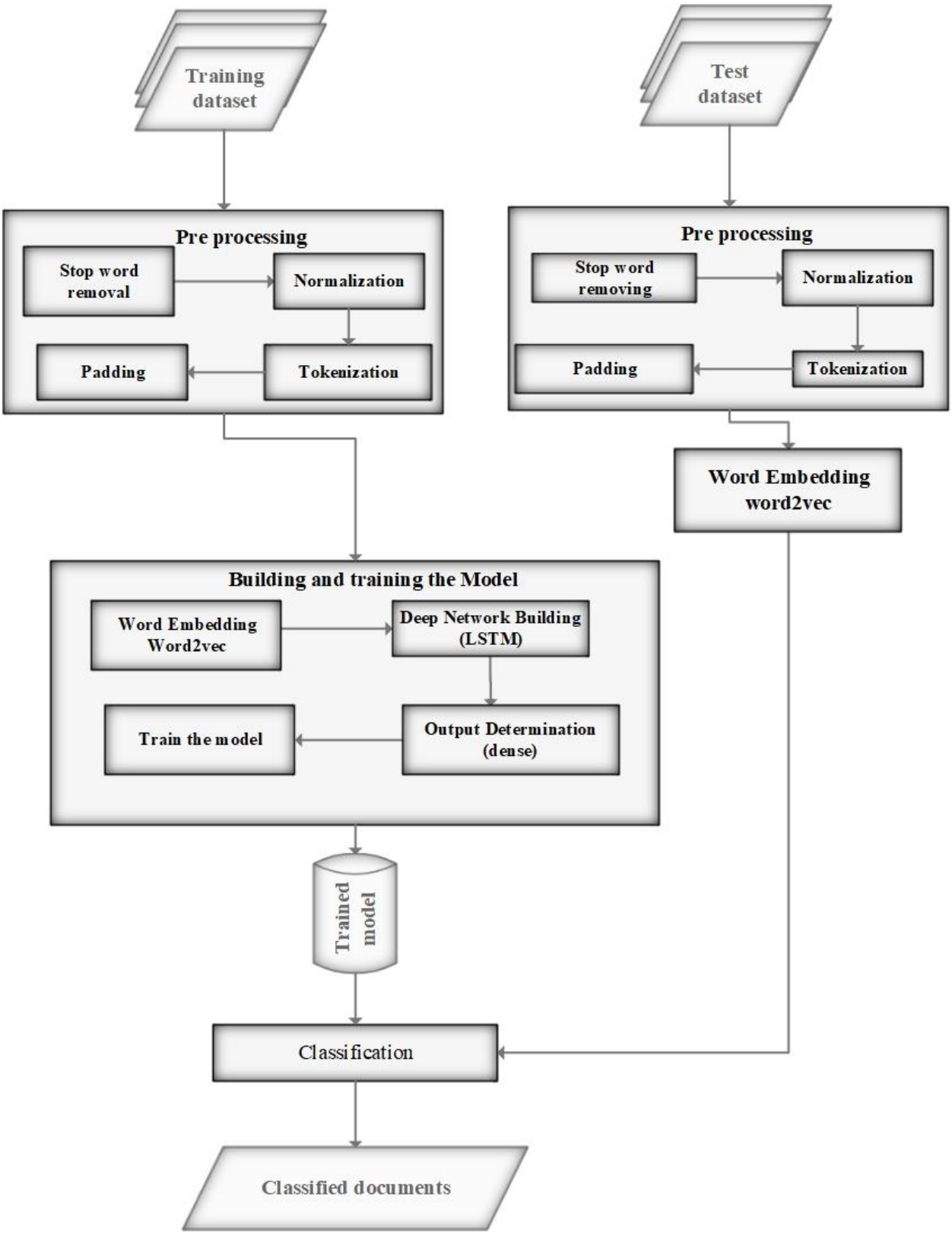

Figure 4.1 The architecture of the proposed LSTM text classifier

4.2.1 Training and testing dataset

Deep learning model requires large amount of data for training, testing and validation. The data that we collect are divided in to training testing and validation and all data are pass through preprocessing phase for training to be an input for the embedding component. Those data are collected from seven domains. As shown in *table 4.1*, The corpus is divided into training and testing data using split of 17% for training and 83% are for training and validation

Table 4. 1 Training and testing Dataset

No	Labe or class	Training	Testing	Format
1.	Art	919	197	Text
2.	Religion	1189	210	Text
3.	Business	1317	220	Text
4.	General	366	97	Text
5.	Health	272	70	Text
6.	Politics	1002	201	Text
7.	Sport	877	170	Text
Total		5942	1174	

4.2.2 Amharic text preprocessing

The first step in data processing and analysis is preprocessing, in which text data before using for the application purpose must be preprocessed and makes it ready for usage for further analysis and interpretation. Unstructured data are data in different formats and representations. So, such data has to be normalized, tokenized, padded and some words which have no contribution to the content building in the document have to be removed.

Stop word removal

There are many words in a given text document that are used for grammatical formation and connecting parts of a sentence rather than describing the intent of the text document. Like: ነው፣ ናቸው፣ ግን፣ ሆኖም፣ እና፣ ነበር፣ ና፣ ከ፣ ስለ etc. are words considered as common Amharic stop word. Stop-words are words that have not content building contributions and words that occur most frequently in many text data but are not relevant or have no impact to create classification among

text documents. Thus, those words are filtered out during the preprocessing step in the processing of natural language data. Amharic stop words are not known by the NLTK and keras preprocessor, we need to identify well known Amharic stop words and create list of stop words from our corpus.

We used sub-sampling technique to identify stop words that are most frequently occur in all documents of the corpus. Because, it destructively subsamples words whose frequency is greater than defined frequency *F* with a probability *p>defined probability* while preservative the ranking of the frequencies. Finally, commonly known stop-words in Amharic and those subsampled words are appended into one text file and remove those from our corpora. Algorithm 4.1 shows how stop words are remove from our dataset.

```
start
    Input: Text document corpus(F) and list of stop words(S)
    Output: Text documents free from stop words
    find stop words (s) from file(F)
      if S is in F
        remove S from the file F
      else
        navigate through all words in F
     save   F in new file as new file
stop
```

Algorithm 4. 1 Algorithms for Stop word removing

A. Normalization

Before further processing, the text needs to be normalized. Normalization is a process of converting a list of orthographically diverse words to a more uniform or common sequence. It generally refers to a series of related tasks meant to put all text on a level playing field: removing punctuation, converting numbers to their word equivalents. It puts all words on equal footing and allows processing to proceed uniformly. We identify homophones (those are characters that have different symbol but having the same pronunciation. The characters such as {ሠ፣ ሰ} ፣ {ጸ፣ ፀ}፣ {ሀ፣ ኃ፣ ሐ}፣ {አ፣ ዓ፣ ዐ} with their seven subcomponents are some of the characters which are used

interchangeably to build a orthographically variant word. After those characters are identified, we tend to normalized with the common representation through the total corpus. Algorithm 4.2 shows how normalization is done in our dataset.

```
start
Input: Text document corpus(F)and list of corresponding
normalizers(N), homophones(n) characters
Output: Normalized text document
    split words from file(F) in character level
    Find homophones (n) from F
      if n is in F
        replace it by the corresponding normalizer(N)
      else
        navigate through all words in F
     save F in new file as New file
stop
```

Algorithm 4. 2 Algorithms for normalization

B. **Tokenization**

Tokenization is a step that splits longer strings of text into smaller pieces or tokens. Larger chunks of text data can be tokenized into a list of sentences, sentences can be tokenized into list of words, etc. The tokenizer function contains the number of words in the document as a parameter and it fits with the training and the test data. Those tokens are also converted to vector representation to be understood by the deep learning algorithm. Thus, we have a number of text documents and those documents are tokenized in to a number of word tokens to be represented by a vector with unique numerical representation. Algorithm 4.3 shows the process of tokenization.

```
start
    Input: Normalized text document corpus(F)
    Output: Tokenized and numerically represented corpus
    split sentence's and phrases of file (F) into tokens
    assign a number to each unique token
stop
```

Algorithm 4. 3 Algorithms for tokenization

C. Padding

All the deep learning network requires input documents in uniform matric that are in the same shape and size. However, during preprocess, not all the text documents have the same length. In other words, naturally, some of the texts are longer or shorter in the number of tokens or words. Thus, documents are padded into the same size by adding zero at the end of the shorter document matric and removing some of the elements of the less frequent document to make it uniform. Algorithm 4.4 shows how tokenized documents are padded.

```
Start
    Input: Tokenized and numerically represented corpus
    Output: uniformly padded tokenized document
    Count number of tokens (t) from each file
    Find maximum and minimum tokens
    Determine the pad size n
    Pad the tokens (t)to defined size n
       if t<n
          add n-t zero pads to t
        else remove t-n tokens from t
Stop
```

Algorithm 4. 4 Algorithms for document padding

4.2.3 Word-embedding

Word-embedding is a feature learning technique in which each word or phrase from the vocabulary is mapped to the N dimension vector of real numbers Thus, we use word2vec as an embedding schema to form semantic meaning-extraction and representation of the data for the deep network.

Table 4.2 shows the demonstration of word2vec in the skip-gram model using a sample sentence from our corpus.

Sentence: ሀይማኖታዊ አለባበሱ ከየአካባቢው ባህል ጋር የሚፈጥረው የመቀላቀል ሁኔታ ይታያል

After preprocessing: ሀይማኖት አለባበስ ባህል ፈጠራ መቀላቀል

Table 4. 2 Demonstration of word2vec using Amharic sentences

Tokens of the sentence	Target word	Context word
[ሀይማኖት፣ አለባበስ፣ ባህል፣ ፈጠራ፣ መቀላቀል]	ባህል	[ሀይማኖት፣ አለባበስ፣ ፈጠራ፣ መቀላቀል]
[ሀይማኖት፣ አለባበስ፣ ባህል፣ ፈጠራ]	ባህል	[ሀይማኖት፣ አለባበስ፣ ባህል፣ ፈጠራ]
[አለባበስ፣ ባህል፣ ፈጠራ፣ መቀላቀል]	ባህል	[አለባበስ፣ፈጠራ፣ መቀላቀል]
[ሀይማኖት፣ አለባበስ፣ ባህል፣ ፈጠራ፣ መቀላቀል]	ሀይማኖት	[አለባበስ፣ ባህል፣ ፈጠራ፣ መቀላቀል]
[ሀይማኖት፣ አለባበስ፣ ባህል፣ መቀላቀል]	ሀይማኖት	[አለባበስ፣ ባህል፣ መቀላቀል]
[ሀይማኖት፣ አለባበስ፣ ባህል፣ ፈጠራ፣ መቀላቀል]	አለባበስ	[ሀይማኖት፣ ባህል፣ ፈጠራ፣ መቀላቀል]
[ሀይማኖት፣ አለባበስ፣ ባህል]	አለባበስ	[ሀይማኖት፣ ባህል]
[ሀይማኖት፣ አለባበስ፣ ባህል፣ ፈጠራ]	አለባበስ	[ሀይማኖት፣ ባህል፣ ፈጠራ]

Table 4.2 describes how Wordv2vec illustrates documents in context -to- target pair with their probability of similarity in the document. The word [ሀይማኖት፣ አለባበስ፣ ባህል] have the same context in the different arrangement in the given sentence and they considered as semantically similar words. Algorithm 4.5 shows the process of building word2vec.

```
Start
    Input: Text Document Corpus
    Output: Trained word2vec model feature vector
    Preprocess the text ()
    Add all text in one file F ()
    Train F (Word2Vec (F)) ()
   save the trained Model
Stop
```

Algorithm 4. 5 Algorithms for building Word2vec

4.2.4 Deep Network Modeling

After the data are preprocessed and the embedding component is built the next component is building a deep network for training the data. As we describe in the literature review, there are different deep network architectures that are used for different applications of NLP. RNN is a sequential model in the field of artificial intelligence and neural network. It processes input data that are represented in a sparse vector sequentially. But, while RNN process inputs sequentially, if the process is held on a large amount of data with a complex and large sequence it may lose important information, and the problems of long-term dependency is happened [45]. Thus, long term dependency problems happen because the RNN lacks memory that stores the content of the previous state of a sequence. For example, "ኢትዮጵያ ውስጥ የምኖር የ አዲስ አበባ ዩኒቨርሲቲ ተማሪ ነኝ" the given sentences are Amharic sentence that contains some sequence of words that a student who lives in Ethiopia and a student at Addis Ababa University, here RNN can easily understand the word አዲስ አበባ is the name of University found in Ethiopia. Thus, there is a strong relationship between ኢትዮጵያ, አዲስ አበባ, and ዩኒቨርሲቲ; Ethiopia has a university, Addis Ababa is A university, Addis Ababa is Found in Ethiopia those are the basic relationship and RNN can understand but if we want to narrow down to know the name, college, Department, and batch of the student RNN lacks memory to store those properties of the student and this problem is called

long term dependency. Thus, LSTM is come up with a memory unit called cell state to overcome this problem. It is an improved version of RNN that accepts the vector representation of the data as an input to learn features.

LSTM contains two main components the cell state and the gate which is the regulator of the network. The gate of LSTM can decide which information is allowed on the cell state and it can learn what information is relevant to be saved or forgotten during training. Thus, LSTM can store the universities found in Ethiopia (ኢትዮጵያ), name of colleges, number of students (ተማሪ) and name of students (ተማሪ) found in Addis Ababa University (አዲስ አበባ ዩኒቨርሲቲ), it enables to perform operations on the given data based on its past history and relationship between sequenced data.

4.2.5 Output Determination

The final output of the model is determined by the dense layer which is a fully connected layer that accepts the deep representation of the input data and transforms to the final output class using an activation function. In a dense layer, all nodes in the previous layer connect to the nodes in the current layer. As we discussed in the literature review there are two basic activation functions for classification which are the sigmoid, used for binary classification, and SoftMax, which is used for both binary and multi-class classification. As a result, we may use the SoftMax activation function to generate the final output that produces values in the range of 0 -to- 1. Thus, it is used as the final layer in classification models.

4.2.6 Training the model

The next process after the output is determined the model is trained using the training data and evaluated using the validation data which is split from the training data using validation split. To train the model the number of epochs and the amount of training data which is trained at a time (batch size) is determined. Training the model using all of the training data at a time is impossible and it need to subdivided to predefined number of batch size. The training capability of the model is evaluated at each batch size and it is evaluated at each epoch. Finally, the trained model is saved and tested by the testing dataset

```
Start
        Input: semantically represented words(word2vec)
        Output: Trained LSTM model
        Add the LSTM layer with defined number of neurons ()
        Add dropout ()
        Add recurrent dropout ()
        Determine the output ()
        Compile the model ()
        Summarize the model ()
        Fit the data with the model ()
        Train the model with training data ()
Stop
```

Algorithm 4. 6 algorithm to train the model

4.2.7 Classification

After the model is summarized and fit with the data to be evaluated with the training, testing, and validation data set separately by using the metric and the loss function. When the model is evaluated, the metrics are accuracy and loss; training accuracy, training loss, validation accuracy, validation loss, and testing accuracy. Thus, the higher the accuracy and the lower the loss indicates the better the model performs

The process of classification accomplished on the trained model using the test data that pass through the preprocessing phase and embedded with the same dimension of the training and validation data are embedded. After the model is tested the summery of the classification is organized. The classification process also accomplished with external datasets that are not labeled into measure how much the model learns the data and capable of predicting external unknown domain text documents to their appropriate class. Thus, the final output of the testing result is documents with its corresponding class or classified document.

Chapter 5: Experimentation and Evaluation

5.1 Introduction

In this chapter experimentations, procedures, and evaluation methods of the entire works are described in detail. Thus, to make experimentation, the tools and programming languages are described and the evaluation metrics and methods as well as the result of the evaluation are explained concerning the data that is used for experimentation and the metrics or approaches used for evaluation. The experimental setting such as the capacity of the computer, operating system, RAM, processor, etc. are described and the experimental procedures and the result obtained are documented. The actual result and performance of this study are also compared with other related researchers which are described in the literature review and related work. The comparison may be described through the values of the accuracy, the amount of data used and the approaches they used concerning their final result of performance measure.

5.2 Experimental procedure

In this section of the chapter, the activities and procedures that are used to develop and evaluate the Semantic-aware Amharic text classification using a deep learning approach are described in detail with some examples and graphical or tabular representation.

5.2.1 Dataset collection

A large amount of data is required to undertake the experimentation to achieve better result and well-performing model for the classification of Amharic texts. The deep learning approaches by its nature requires a large amount of data and it is compatible with large dataset. The data which are used to undertake this work is unstructured Amharic text documents collected from different sources such as the Amharic bible (the old and new testament), Ethiopian news agency, broadcasting media, and from different online newspapers and magazine. The classification is performed on seven (7) domains of Amharic text document and data are collected for each class of texts. After the data are collected those data are tagged or labeled with their corresponding class. Thus, the labeled documents are examined by the experts whether the data are labeled to the correct class or not. The experts are the ones that are familiar with the language implies that those who

conduct researches on the domain, who can understand the language in a good manner and whose native language is Amharic.

Table 5.1 Amount of collected text documents in each class

No	Name of the class	Number of text documents	Label name
1.	Religion	1399	bible
2.	Business	1537	Business
3.	Politics	1172	Politics
4.	Art	1116	Art
5.	Sport	1047	Sport
6.	Health	382	Health
7.	General	463	general
Total		7116	

5.2.2 Tools and programming language

A. Tools

There are different development tools of deep learning for natural language processing that is open source and easily adapted. Thus, the anaconda is a free and open-source distribution of R and python programming languages. It is popular because it brings many of the tools used in data science and deep learning with just one install, so it's great for having a short and simple setup. To conduct this work Keras, TensorFlow, and Genism are used as a tool. Keras is a high-level neural network, written in Python and capable of running on top of TensorFlow and it enables easy and fast prototyping, supports CNN and RNN and it runs seamlessly on CPU and GPU. TensorFlow is an end to end open-source platform for deep learning and it builds and trains deep learning models by using the high-level Keras API. Keras and TensorFlow are used for preprocessing and the construction of the model for training and validation. Genism is a library for document similarity analysis and it is a production-ready tool that you can trust with several crucial components of NLP applications. The semantic component of this work is done by the genism

library for developing the word embedding that analysis the similarity of words with in-text document. Word2vec is developed for semantic support for the model.

B. Programing Language

Python is an object-oriented, an interpreter, and high-level programming language with dynamic semantics. Python programming is used for the experimentation of the Semantic-aware Amharic text classification using deep learning approaches using Jupiter notebook editor.

C. Experimental setup

Deep learning experimentations require high processor and GPU supported computing. In this study, we used a computer with a memory capacity of 8 GB RAM, 2.41 GHz processor, and 64-bit Windows 10 operating system.

5.2.3 Text data cleaning or preprocessing

After data are collected and programming languages and tools are selected the next step for the experimentation of Semantic-aware Amharic text classification using deep learning approaches is text data preprocessing. The preprocessing tasks are adopted as it is described in chapter four, the collected data are processed to make it ready for training. During preprocessing 563 stop words (words which don't have any contribution for the context constriction of documents) are removed and words that have different representations but having the same meaning are normalized into canonical form.

Deep learning algorithms and architectures use numerical representation as an input for the model to be trained. Thus, the text data are tokenized and converted into a numerical representation which is called tokenization. Those tokens are padded into the uniform representation of matric since the input of the deep learning model is represented in uniform matric. During padding, the maximum number of words from the entire document is selected and the other documents are padded to the same dimension with it. The variation of the number of words in each text file affects the sparseness of the padding structure. While padding to make the small number of word documents uniform with the maximum on, the zero (0) padded is added. But the vector may be sparser, the larger the zero-padded element of the vector may lead the model less performed. Sparse vectors must be converted to dense vector or the sparseness of the data must be reduced by adjusting the

pad size during padding. The numerically represented and padded tokens converted into a vector representation, thus the representation of tokens into vectors is embedding components of the model.

5.2.4 Word Embedding

The embedding component is the representation of the words to feature learning in which each word or phrase from the vocabulary is mapped to 300 dimensions of vector. It improves the performance of text classification based on the deep neural network by considering the semantics of the document. word2vec is one of the embedding representations in the semantic and neural network field of study.

Table 5.2 parameters needed for training word2vec

No	Parameters to train word2vec	Values Of parameter	Descriptions of the parameter	Remark
1.	Windows	10	Maximum skip length window between words	
2.	Embedding dimension	300	Size of word vector in which each word is represented	
3.	Learning rate	0.05	Learning rate for SGD estimation	
4.	Number of epochs	30	Number of training epochs	
5.	Number of threads	12	Number of training thread	
6.	SG	1	Choosing Skip-gram model	
7.	Iteration	20	The training iteration	
8.	Minimum frequency	3	This may discard words appear minimum frequency in the document	
9.	Embedding directory	“E/MyWordVector”	The directory in which the word2vec is saved	

Table 5. 3 Top five similar words for the word ቋንቋ

Number	Similar word	Probability of similarity
1.	ስነዘደ	98.77
2.	መፍቻ	98.50
3.	ስነልሳን	98.13
4.	ማስተማር	97.86
5.	ሰዋሰው	97.49

Table 5. 4 Top 6 similar words for the word እስልምና

Number	Similar word	Probability of similarity
1.	አሸንዳ	99.57
2.	ሙስሊም	99.49
3.	ኦርቶዶክስ	99.48
4.	መውሊድ	99.42
5.	እስላማዊ	99.3
6.	ቤተክርስቲያን	99.27

Table 5. 5 Top five similar words for the word ዘገባ

Number	Similar word	Probability of similarity
1.	አንባቢ	98.45
2.	አቀራረብ	97.81
3.	ይዘት	97.54
4.	ጽሁፍ	97.44
5.	መጣጥፍ	96.68

5.2.5 Model construction (LSTM)

LSTM layer

After the data are preprocessed and the embedding component is built the next component is the LSTM layer that accepts the vector representation of the data as an input to learn features. The learning capability of this LSTM is depending on the performance of the embedding layer, the amount of input data, and the number of neurons used for training. The LSTM layer of the model contains the number of neurons, the dropout, and the recurrent-dropout as a parameter and the model may contain one or multiple LSTM layer that depends on the problem that we solve and the data that we have for training.

Most of the time single layer LSTM is used for simple and linear problems, whereas, multiple layer LSTM is used for nonlinear and convex problems. We are doing our experimentation both in single and multiple layer LSTM and finally, we select the one that performs well. The dropout component has a value between 0 and 1, which is a fraction of the units to drop for the linear transformation of the input data; we use 0.2 or 20% dropout for the experimentation. Whereas the recurrent-dropout is a fraction of values between 0 and 1 to drop for the linear transformation of the recurrent state; we use 0.2 or 20% recurrent dropout for the experimentation.

Dense layer

The dense layer is a fully connected layer that accepts the deep representation of the input data and transforms into the final output class using an activation function. The dense layer of the model contains two basic parameters, the number of class as a dimension of the output vector and the activation function. We use the SoftMax activation function to generate the final output that lays values in the range of 0 -to- 1. We have also seven (7) class such as, religious, politics, business, general, art, health and sport, in which the output layer maps the result in to 7 dimensions of vector. Therefore, it is used as the final layer in the text classification models.

Compiling the model

After the output layers are defined the model needs to be compiled. To compile the model, the loss function, the optimizer, and the metric that validate the model need to be specified. The

optimizer controls the learning rate. We used ‘Adam’ as an optimizer. The learning rate determines the computational of the optimal weights for the model to be calculated. We used the sparse categorical cross-entropy as a loss function since we have 7 classes to be predicted after compiling the model. The metric is used for plotting the result after the history of the model is saved and we used accuracy as a metric for our model. The metrics contain training loss, training accuracy, validation loss, validation accuracy, and testing accuracy.

Summarizing the model

After the model is compiled, it needs to be summarized to visualize and generate each layer of the model with their corresponding output shape and number of parameters. It generates the total parameter in the model by summarizing from each layer and identifies the number of the trainable and non-trainable parameter from the total parameter.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 300, 100)          22192100
_________________________________________________________________
lstm_5 (LSTM)                (None, 100)               80400
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 707
=================================================================
Total params: 22,273,207
Trainable params: 22,273,207
Non-trainable params: 0
_________________________________________________________________
```

Figure 5. 1 summary of the model with Single LSTM

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 300, 100)          22192100
_________________________________________________________________
lstm_4 (LSTM)                (None, 300, 100)          80400
_________________________________________________________________
lstm_5 (LSTM)                (None, 300, 100)          80400
_________________________________________________________________
lstm_6 (LSTM)                (None, 100)               80400
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 707
=================================================================
Total params: 22,434,007
Trainable params: 22,434,007
Non-trainable params: 0
_________________________________________________________________
```

Figure 5. 2 summaries of the model with three LSTM

Model fitting

After the model is summarized it should be fit with the data that are prepared as a training set with their corresponding label or category. The fitting model component includes the training set, the label or the annotation of training data(the class it belongs), the number of epochs (the number of times the model goes over the entire dataset), and the batch size (how may training examples are processed at a time), as well as the validation split (how much of the training data is taken each epoch to measure how well the model generalizes on unseen data), need to be specified. We used validation split as 20% of our training set and 50 and 100 alternative numbers of epochs with 32 batch sizes.

Evaluating the Model

After the model is fitted with the data, its performance should up to be evaluated with the training, testing, and validation dataset separately using the metric that is defined in the model fitting component. Those metrics are accuracy and loss; training accuracy, training loss, validation accuracy, validation loss, and testing accuracy. Thus, the higher the accuracy and the lower the loss indicates the better the model performs.

Using the loss and the accuracy values of the model we adjust the parameters which are the batch size, the embedding dimensions, and the number of neurons in the embedding, number of LSTM layers, and the value of dropout of the model to get a better-performed model.

Testing and Classification

After the model is built, fitted with data, and evaluated using the evaluation metrics, the last and the major task is making classification using the model that we trained. To make a classification, the model needs to be saved and loaded. Thus the model is saved using a method called *save.model()* using h5 extensions and finally load for testing using *models.load_model()* Keras library.

The performance of the model is tested in two major ways, using the test data and using external data in which the class it belongs or its domain is unknown. Thus, when we use the testing data, it generates the testing accuracy 92.13 percent of testing data is classified into the correct class.

When the model is tested using arbitrary unknown class or category data, the data need to be passed all the preprocessing steps done for the training and testing data. After it is preprocessed, the model compares its content with each of the class and computes the probability in which the given document belongs to each class, and finally, the one with high probability is predicted as the class in which the given document belongs. The prediction process is done as the labels or class are stored in a single one-dimensional array that contains the name of classes in the corresponding index with the index number appended for it during labeling and the probability of being in a specific class is stored on the order of its index.

Table 5. 6 Test result using external data

No	Doc-No	Probability of being in class in %							Actual Class	Predicted class
		Politics	Bible	Health	Art	General	Business	Sport		
1.	D1	0.07	1.6	6.5	99.3	0.0039	1.08	5.5	Health	Art
2.	D2	0.031	0.018	0.07	0.74	0.0036	99.9	8.8	Business	Business
3.	D3	1.4	0.15	79.38	20.4	0.26	0.175	9.6	Health	Health
4.	D4	10.5	0.15	52.6	32.15	0.5	6.4	89.7	Sport	Sport
5.	D5	97.1	0.07	22.1	0.47	0.7	3.96	1.08	Politics	Politics
6.	D6	98.1	0.0019	0.47	4.57	3.1	9.92	0.41	Politics	Politics
7.	D7	23.1	0.019	0.0017	0.028	1.73	97.6	2.10	Politics	Business

5.3 Evaluation

The performance of the system or the model is evaluated using different evaluation metrics. Accuracy is one of the major evaluation metrics to evaluate different deep learning applications. It is the degree to which the result of a system, model, or specification conforms to a standard or the correct value. Our LSTM text classification model is evaluated based on the nature of the LSTM structure. We have training, testing and validation dataset that pass though the preprocessing component. Thus, the model is trained using those datasets and the performance of the model is evaluated and its accuracy is recorded besides the training, testing, and validation dataset. The evaluation held on the nature of the LSTM structure is based on the number of neurons, the number of epochs, and the number of hidden layers with constant batch size in which the model is built-in. Thus, we use 50 and 100 number of neurons for single-layer and three-layer LSTM with arbitrary 50 and 100 epoch.

The evaluation processed is held on 4753 training,1174 testing, and 1189 validation dataset. Thus, the training accuracy and loss evaluates the training data in every learning rate in which the model learns the data once at a time (batch size =32), the model learns 32 training data at a time and it computes the training accuracy. But, validation accuracy and loss, evaluate the training performance of the model in each epoch.

Table 5. 7 Evaluation result for training, testing, and validation dataset

No	No of layers	No of neurons	No of epochs	Validation accuracy	Training loss	Validation loss
1.	1	50	50	86.88	0.32	0.42
2.	1	100	50	85.53	0.32	0.45
3.	3	50	50	75.02	0.54	0.62
4.	3	100	50	75.02	0.66	0.70
5.	1	50	100	85.62	0.25	0.49
6.	1	100	100	86.71	0.16	0.47
7.	3	100	100	82.67	0.46	0.55

Table 5.7, demonstrates the, validation accuracy, validation loss, and testing accuracy of the model trained on epoch (50,100), number of neurons (50,100) separately. For each experimentation the performance of the model is differ based on those parameters, from those the model with 100 number of neurons, 100 epochs on a single layer performs better which is testing accuracy = 92.12 and validation accuracy= 86.71.

5.4. Discussion

The proposed model is accomplished by 7116 Amharic text documents. Out of those documents, 4753 are training, 1189 validation, and 1174 for testing are used for testing and training the model. The performance of the model is determined by the amount of training data. A large amount of training data results in better results while the performance of the model reduced when using a small amount of data. During training all of the training data can’t be pass to the model at once, to

overcome this problem defining batch size to divide the data into smaller size divisions and those divisions are passed to the model one by one and the weight of the neural network that passed to the next node is updated at the end of each step.

The other factor that affects the performance of the model is the number of epochs and neuron, the number of epochs is the number of times the model trained using the entire dataset and at each epoch, the entire dataset is passed forward and backward through the neural network once. Learning repeatedly increases the capability of the model to learn the feature and content of the data. But, training a model with a large number of times may lead to overfitting, the overfitting problems need to be optimized to the normal learning state. Thus, dropout optimizes the overfitting through deactivating arbitrarily some amount on nodes from the given node or neurons.

The number of neurons affects the performance of the model. It is more important than the number of layers. *Table 5.7* clearly shows its effect on the performance of the model, using less neuron on the layer may lead to loss some basic information's because its learning capability determined on the neuron it contains. There is a great change of accuracy and loss when using 50 and 100 neurons for training, for fewer neurons there is an increase in the loss (the measure of how well the model generalized to unseen records) and a decrease the accuracy for both the training, testing and validation phase. When we use 100 neurons maximum of 92.13 and using 50 neurons 88.25 testing accuracy using 100 epochs are the result of the model after training.

The number of layers affect the performance of the model. Multiple layer network enables the network to more eager to recognize the certain aspect of the input data. Most of the time the number of layers in deep learning model determined by the nature of the problem; multiple layer networks are needed for more complex, non-linear, and convex problems whereas single layer networks are used for linear and simple problems. In our experimentation, we use a three-layer and single-layer LSTM network to analyze the change in it.

As shown in *Table 5.7 there* is a change in the testing accuracy, validation accuracy, and validation loss. We have got better performance when we use a single-layer network since the problem that we solve is not too complex and convex. It learns the content of the data and translates it to a single output; it has many to one relationship having a set of input documents, a sequence of words and

the corresponding output is a single class or label. When we use single-layer the maximum testing accuracy is 92.13 and when we use a Three-layer network the maximum testing accuracy is 83.65.

Comparison with related works

There are investigations which are done for the classification of Amharic documents using different approaches resulting in different result concerning their dataset and methods used. Most of those researches are conducted using the same data structure which is collected from the Ethiopian News Agency (ENA).

Table 5. 8 Comparison between related works and the proposed system

Title	Approaches	Result	Remark
1. Concept-based automatic Amharic Document categorization	• **Use domain ontology (news)** • **Use manually crafted index terms and mapped to the ontology**	**92%**	**Related work**
2. Hierarchical Amharic news text classification	• Flat text classification • Topics are treated as a class • SVM • Not suitable for large data	90%	Related work
3. Automatic Amharic text news classification: a neural networks approach	• Learning vector quantization (LVQ) • TF and TF-IDF • Features are extracted manually using TF	71.96%	Related work
4. Semantic-Aware Amharic Text Classification Using Deep Learning Approach	• LSTM • Word embedding (Word2vec)	92.13	Our proposed system

In general, our model is done by using LSTM, an improved version of RNN that is capable of learning long-term dependencies, especially in sequence prediction problems. LSTM has feedback connections, i.e., it is capable of processing the entire sequence of data, apart from single data points such as words.

Chapter 6: Conclusion and Recommendation

6.1 Conclusion

This research work has attempted to look into the approaches of deep learning-based Amharic text classification. Throughout this study, the basic elements of the model for text document classifications are clearly defined. The designing component consists of the document preprocessing module (stop-word removal, normalization, tokenization and padding), the model building module (word embedding, deep network, output determination, and model compiling), the summarization and fitting, and evaluation and testing are the basic modules in designing and implementation of the entire work.

The use of deep learning in the area of neural network requires less feature engineering in which the word embedding component is neural-based extraction of features from unstructured text data in vector representation with some degree of similarity between tokens throughout the entire tokens of the dataset. Thus, the vector representation of the document is used to compute the semantics of the word. Word2vec is used to represent words in vector form to generate the semantic similarity between them. We used Skip-Gram based word2vec, which is the representation of the semantics of the target word from the given sentences using its surrounding context words.

The major element of this study is the deep network in which it learns the features of the data and transfers it to the output layer to generate the expected output of the model. We used an improved version of RNN which is LSTM to train and build the Semantic-aware Amharic text classification model. LSTM is a sequential model that is used to process time sequence data and it improves some of the limitations of RNN. It reduces the problem of long-term dependency and vanishing gradient through its memory called cell state. The cell state of LSTM is used to store the content of the previous hidden state and the given processed input data to determine the next hidden state since it is a sequential process. It contains a gate that is used to determine the information to be kept, forgotten, and need special attention to the cell state during training.

In general, we develop Semantic-aware Amharic text classification using the LSTM approach. The model is developed using unstructured Amharic text documents. The overall result shows that

the model performs better when the number of epochs and neurons increase the performance of the model increases. The performance of the model is measured in terms of accuracy and loss in training, validation, and testing phase. The final result shows that the model performs 92.13 testing accuracy, and 86.71 validation accuracy using.

6.2 Contribution to the study

The main contributions of the study are summarized as:

- A generic model is designed for Semantic-aware Amharic text document classification that takes advantage of neural word embedding technology.
- The thesis contributes to the growth of semantic feature extraction to text data analysis using word embedding.
- The thesis contributes by developing a generic word2vec that is used for other applications of NLP.

6.3 Future work and recommendation

In this research, we have attempted to explore the use of a deep learning approach to build a Semantic-aware Amharic text classification model. However, some attempts may be adopted to the area of text classification to achieve better using different feature extraction and learning methodology. The main recommendation we want to recommend or future works to be done are:

- Acquiring good semantic representation of text document needs enriched or large amount of data to train the word2vec
- Using the hybrid of LSTM with other deep learning architectures to take the advantages of other models
- The performance of the machine used for training affects the training rate of the model and using high processing computer train the model with a large number of epochs and neurons is recommended
- To reduce the dimensionality of the data use lemmatization.

References

[1] Govardha, "ontology-based text classification -Telugu documents," *International Journal of Scientific and Engineering Research,* 2017.

[2] K. Kowsari , K. J. Meimandi, M. Heidarysafa, S. Mendu , L. Barnes and D. Brown, "Text Classification Algorithms: A Survey," *MDPI,* Vols. information 10,50, 2019.

[3] Meron Sahilemaryam, "Concept-based automatic Amharic document categorization," not published, Addis Ababa, 2009.

[4] Kowsari, "medium.com," [Online]. Available: https://medium.com/text-classification-algorithms/text-classification-algorithms-a-survey-a215b7ab7e2d. [Accessed 13 september 2019].

[5] Rimchala, "MonkyLearn," [Online]. Available: https://MonkyLearn.com/deep learning in text classification. [Accessed 23 september 2019].

[6] Nayat Sanchez-Pi, M. Luis, and B. Cristina "improving ontology-based text classification: An occupational health and security," *Journal of Applied Logic,* 2017.

[7] F. Wei, Han, Shi and Haozen "Empirical Study of Deep Learning for Text Classification in Legal Document Review," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018.

[8] M. Allahyari and Kyrs J. Kochut, "ontology-based text classification into dynamically defined topics," in *2014 IEEE international conference on semantic computing*, 2014.

[9] Ertam, "Deep learning-based text classification with Web Scraping methods”," in *2018 IEEE international conference on artificial intelligence and data processing*, 2018.

[10] Alemu. K, "Hierarchical Amharic News Text Classification," not published, Addis Ababa, 2010.

[11] Worku. K, "Automatic Amharic text news classification: a neural networks approach," *Ethiopian Journal of Science and Technology,* vol. volume 6, no. no 2, 2013.

[12] J. Euzenat, " Ontology Matching. Springer-Verlag Berlin Heidelberg," pp. , p. 36, 2007.

[13] “Natural Language Processing and Semantic”, "Spring.com," 2018. [Online]. Available: https://www.springest.nl/icm/nlp-in-3-dagen-2. [Accessed October 23 October 2019].

[14] M. Giatsoglou, M. Vosalis, K. Diamentares, A. Vakalil, G. Sargiannidis and K. chatzisavvas "Sentiment analysis leveraging emotions and word embeddings.," *Expert Systems with Applications,* vol. 69, pp. 214-224., 2017.

[15] M. Rhanoui, M. Mikram, Y. Sihan, and B. Soukanian"A CNN-BiLSTM Model for Document-Level Sentiment Analysis," *Machine learning and knowledge extraction,* vol. 1, pp. 832-847, 2019.

[16] Deksne, k. Balodis, and Daiga "FastText-Based Intent Detection forInflacted language," *MDPI,* vol. 1, 2019.

[17] Kumar and Chandrasekha, "Text Data Pre-processing and Dimensionality Reduction Techniques for Document Clustering," *International Journal of Engineering Research & Technology (IJERT),* vol. 1, no. 5, pp. 2278-0181, 2012.

[18] E. Bojanowski, "EnrichingWord Vectors with Subword Information," *Facebook AI Research,* vol. 2, 2017.

[19] Goldstine, "Summarizing Text Documents: Sentence Selection and Evaluation Metrics," *ACM SIGIR,* pp. 121-128, 1999.

[20] B. Rhanoui, "A CNN-BiLSTM Model for Document-Level sentiment Analysis," *Machine learning and Knowledge extraction,* pp. 832-848, 2019.

[21] Mikolov, Sutskevor, Chen, Corradol and Dean, "Distributed Representations of words and Phrases," *Advanced Neural information,* vol. 26, pp. 3111-3119, 2013.

[22] Sivakami, and Thangaraj, "Text classification techniques: a literature review," *Interdisciplinary Journal of information, knowledge, and management,* vol. 13, 2018.

[23] "Modeling the difference between Machin Learning, statistical analysis" Jan 14, 2013. [Online]. Available: https://www.analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/. [Accessed 3 Nov 2019].

[24] Baxter, and Roy, "Text classification method review," *Decision Engineering Report Series,* 2007.

[25] Hdwehle, "Machine Learning, deep learning, artificial intelligence: what is the difference," HD Wehle, USA, 2017.

[26] Alom,Taha,Yakopicic, Westberg and Sideke, "A State-of-the-Art Survey on Deep Learning Theory," *MDPI electronics*, 2019.

[27] I. Salian, "Difference Between Supervised, Unsupervised, & Reinforcement learning," Nividia, 2 August 2018. [Online]. Available: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/. [Accessed 12 October 2019].

[28] K. Kowsari, D. Brown and H. Heidarysafa, "Hierarchical Deep Learning for Text Classification. Machine Learning and Applications (ICMLA).," in *2017 16th IEEE International Conference on Machine Learning and Applications*, Mexico, 2017.

[29] Sutskever, Martens and Hinton, "Generating text with recurrent neural networks.," in *28th International Conference on Machine Learning (ICML-11)*, Bellevue, WA, USA, 2011.

[30] Pascanu, Mikolov and Bengio, "On the difficulty of training recurrent neural networks," *ICML*, vol. 28, p. 1310–1318, 2013.

[31] Chung, Gulcehere, Cho and Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *Deep Learning and Representation Learning Workshop*, vol. 1, 2014.

[32] Jaderberg, Simonya, Vadaldi, and Zisserman, "Reading text in the wild with the convolutional neural network," *Int. J. Comput.*, no. 116, pp. 1-20, 2016.

[33] "Ethiopian Government Open Data Portal," 13 Jun 2018. [Online]. Available: ” https://www.data.gov.et/. [Accessed 20 January 2012].

[34] ባየ ይማም, "የአማርኛ ሰዋስው፣ ት.መ.ማ.ማ.ድ," in *Ye- AmariGna Sewasew, T.M.M.M.D*, 1987.

[35] Mesfin. A, Yaregal. A, "Development of Amharic Morphological Analyzer," in *Springer International Publishing*, Switzerland, 2014.

[36] Yohannes. A, "Automatic Amharic Document Categorization: The Case of Ethiopian News Agency", Addis Ababa University, Addis Abab, 2013.

[37] A. Joulin, E. Grove, P. Bojanowski, and T. Mikolv, "Bag of Tricks for Efficient Text Classification," *Facebook AI Research*, vol. 3, 2016.

[38] Goto, R.Vato and Hiroyti, "Categorization of Web News Documents Using Word2Vec and deep learning," in *Proceedings of the 2016 International Conference on Industrial Engineering and Operations Management*, Kuala Lumpur, Malaysia, 2016.

[39] H. Hayati, A. Chanoa, M. Khalid, S. Bennani, "Doc2Vec &Naïve Bayes: Learners' Cognitive Presence Assessment through Asynchronous Online Discussion TQ Transcripts," *iJET*, vol. 14, no. 8, pp. 70-81, 2019.

[40] Kralj, B.Skrlj and Jan "Towards Robust Text Classification with Semantic-aware Recurrent Neural Architecture," *Machine learning and knowledge extraction,* vol. 1, no. 34, 2019.

[41] Mekonnen. A, "Automatic Thesaurus Construction for Amharic Text retrival," Addis Ababa, 2009.

[42] Baroni, Dinu, Kruszewski, "Don't count, Predict! A Systematic Comparison of Context-Counting vs. Context-Predicting Semantic vector," *ACL*, p. 238–247, 2014.

[43] Alemayehu, Nega, W. piter"Stemming of Amharic Words for Information Retrieval," *Literary and Linguistic Computing,* January 2002.

[44] D. Yohannes, "Unsupervised text document clustering using encyclopedic knowledge with word embedding," Addis Ababa, 2018.

[45] "Understanding LSTM Networks," colah's blog, 2015.

[46] M. Phi, "Towards data science," 4 Sep 2018. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[47] Ghahramani, Yarin, Zoubin, "A Theoretically Grounded Application of Dropout in Recurrent neural network," vol. 5, 2016.

[48] P. Jain, "medium.com," 12 Jun 2019. [Online]. Available: https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044.

[49] R. Khandelwal, "Medium.com," 3 Feb 2019. [Online]. Available: https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3.

[50] E. Allibhai, "Building A Deep Learning Model using Keras," *Towards data science,* 17 Sept 2018.

[51] Gawade, "Text Document Classification by using WordNet Ontology and Neural Network," *International Journal of Computer Applications (0975 □ 8887),* vol. volume 182, no. 33, pp. 33-41, 2018.

[52] Peng Zhou, Nora El-Gohary, "Ontology-Based, Multi-Label Text Classification for Enhanced Information Retrieval for Supporting Automated Environmental Compliance Checking," in *2014 International Conference on Computing in Civil and Building Engineering*, North Mathews Avenue, 2014.

Annexes

Annex one: list of Amharic stop word

አይገምቱም	አስገንዝበዋል	እያንዳንዳቸው	ተጠቅሷል	ብለዋልአል	ከትትል	በውኑ
ይኖረዋል	አብራርተዋል	አካሂደዋል	ያባብሱታል	ነበሩም	መሆኑም	አይዘነጋም
እላችኋለሁ	ትናገራል	ተደርገዋል	ይጭራል	በታች ተዳርገዋል	ሳይሆን	አስይዘዋል
አጠናቀዋል	ሰሞን	ይጠቅሳል	ተንትኗል	ስለዚህ	ማንኛውም	ተሠርተዋል
አልቀረበም	ሆናል	ወዘተ	ይገኙበታል	ይኖሩታል	ተሰጥቶበታል	ብአድ
ሆይ	ቆይተዋል	አይደለምን	ተሰምቷል	ተዘርዝሯል	ተደርጓል	ወደዚህ
አልታየም	አምኗል	ነበራች	ተነግሯል	ነን	ነህ	አቅርበዋቸዋል
አስገብተዋል	በዚያም	አመልከተዋል	ተበትኗል	አሳስቧል	ተቃጥሏል	ተረጋግጧል
አትቷል	ያመላክታል	አሳውቋል	እነዚህንም	ይመከራል	ሌላ	ሆነ
ይከፍታል	ይቀጥላል	ተከስተዋል	ቢል	ነኝ	አልቀረቡም	ብላችሁ
የለም	ነገሮች	አልተገለጸም	አስጠንቅቋል	ዳሰዋል	ይሁን	ሆኑ
ይሠራል	ናት	አስቀምጦታል	ተጠቆም	አድንቀዋል	ተሰጥተዋል	እያንዳንዷ
ይገርመኛል	አከለዋል	ይታወሳል	አድርጓቸዋል	በፊት	አለው እስከ	ሰሞኑ
ሆነህ ከኋላ	አይችሉም	እናንተ	በሆነ	ተቆጥረዋል	አግኝተዋል	አሉም
ተነስቷል	ተሰጥቷቸዋል	ያከናውናል	አይደሉም	አብቅቷል	አይቀበሉትም	ተሰንዝረዋል
ማንም	ይቋቋማል	ውስጥ	አስታውቋል	ተቋርጧል	ወጥቷል	ጠፍቷል
ይኖርበታል	ሆነም	ኖርን	ይቻላል	አላገኙም	ተቆጥበዋል	በዚያው
ያገለግላል	እባክዎ	በሆላ	እነዚያም	ወይዘሪት	ተዘግቧል	አስተባብለዋል
ብለውታል	አላደረገም	ተጠቅመዋል	አንችልም	ይሰጣቸዋል	ይገኝበታል	አውስተዋል
ይጠፋዋል	ተቆጥሯል	አሉ	ተይቀዋል	ሆኗል	አስተየት	ጋይተዋል
ተጠቁሟል	አላቆሙም	ቢሆን	ገብታል	ከላይ	ተለያዩ	ማናቸውንም
አድርጎታል	ይችላል	አስተላልፏል	ተወያይተዋል	ይገምታል	መግለጭ	ከዚያ
አስታውሰዋል	አስተላልፏል	ተሰጥታል	ባሉ ሰሞኑን	አቅርቧል	መርጧቸዋል	አስተባብሯል
ነች	አካቷል	አሳይተዋል	በውስጥ	አይቻልም	ዘርዝረዋል	አለብአወጥ
አሁንም	ቀጥለዋል	ተመልሰዋል	ተነግሯቸዋል	አራምደዋል	ተመላክቷል	እኔ
ታውቋል	አሳስበዋል	አድርጋዋል	ቢኖር	አግኝታል	ያሳስባል	አልችልም
አብራርተዋል	ይሰጣል	ሆነው	ያመለከታል	አደርጓል	አነጋግረዋል	አልችልም
	አንፈልግም	ከመካከል	ነበር		አልፏል	ቀበላቸዋል
			መካከል			

	ሆኖም ይጠይቃል ቆይቷል ያሳያል አለች	አንፀባርቀዋል ነበረች ይሰማል በሌላ እንደገለጹት እንደአስረዱት	አቅርበዋል ቢገለጹም እባኩዎ	ሆኗልም አልተካተተም ተገብቷል ትሆን እባክሽ አይችልም	እያንዳንዷ አለኝ በዚህ አሉት	አልጀመሩም አልቻሉም ተመሠረት አያስፈልግም አድርገናል

ሁሉ ሁሉም ኋላ ሁኔታ ሆነ ሆኑ ሆኖም ሁል ሁሉንም ላይ ሌላ ሌሎች ልዩ መሆኑ ማለት ማለቱ መካከል የሚገኙ የሚገኝ ማድረግ ማን ማንም ስሞኑን ሲሆን ሲል ሲሉ ስለ ቢሆን	በኩል በጣም ብቻ በተለይ በተመለከተ በተመሳሳይ የተለያየ የተለያዩ ተባለ ተገለጸ ተገልጿል ተጨማሪ ተከናውኗል ታች ትናንት ነበረች ነበሩ ነበረ ነው ነይ ነገር ነገሮች ናት ናቸው አሁን	እባክዎ አንድ አንጻር እስኪደርስ እንኳ እስከ እዚሁ እና እንደ እንደገና እንዲሁም እንጂ እዚህ እዚያ እያንዳንዱ ኋላ ከላይ ከመካከል ከሰሞኑ ከታች ከውስጥ ከጋራ ከፊት ወዘተ ወይም ወደ	ውጪ ያለ ያሉ ይገባል የኋላ የሰሞኑ የታች የውስጥ የጋራ ይታወሳል ይህ ደግሞ ድረስ ጋራ ግን ገልጿል ገልጸዋል ግዜ ደግሞ ዛሬ ጋር ተናግረዋል የገለጹት ይገልጻል ሲሉ ብለዋል ስለሆነ	አርስ ተችሏል አስይዟል ተናግሯል ያሳስበዋል ተረከበዋል ይወሳል ይባላል አይታወቅም ተለቋል አስቀምጧል ስጥታቸዋል አታደርግም አያስከትልም ይናገራል አፍርሰናል አልተሞከረም ሌሎችም ስጥተውበታል እንጂ አረጋግጠዋል ሁለቱም ቀድም ሆኖኛል ይኖርብናል አይኖርም እንኳ	አንስተዋል አብራርቷል ተዘዋውረዋል ሊሆን አልገቡም ነግረዋቸዋል ቀድሞውም አመለከት አልተፈጠረም ምክንያት ታልፏል አሞግሷል አንድም አሰምቷል አውግዟል ይታያል ይፈቀድላቸዋል ይኼንንም ያወሳል ደርሳል ይገኙባቸዋል አስተለልፈዋል ሆና ይታወቃል ተወስኗል	ኋላ ተችተዋቸዋል ተቀይረዋል ተጠቅሟል አልነበረም ተሞክሯል ይህንንም አላስብም እርሱም እዚያ ይገለጻል ቻሉት ይስተዋላል አናስታውስም ያቀርባል ተፈጥሯል ሰፍሯል ትናንት በርካታ እባክህ አስከትሏል አጠናቋል የጋራ ችሎት በእነዚህ

ብለዋል ብቻ ብዛት ብዙ ቦታ በርካታ አስፈላጊ ስገነዘቡ	አለ አስታወቀ አስታውቀዋል አስታውሰዋል እስካሁን አሳሰበ አሳስበዋል በጣም	ዋና ወደፊት ሲል በቀር በፊት ውስጥ እባክሽ	አቶ ሆኖም አመልክተዋል ይናገራሉ አበራርተው አስረድተዋል እስከ ከጋራ	አምነዋል ቀርባቸዋል አያውቅም አስቀምጠዋቸዋል ተወስተዋል ያትታል	አልተጠናቀቀም አይቀርብም አውጥተዋል ከፊት ተደግፏል ስላሉ ናቸው	ዋና ግን ወደ ተፈጽሚል ቋም ያምናል ቆይታል ይመሠርታል

Annex Two: Training result with 100 neuron and 100 epochs

Model: "sequential_3"

_____

Layer (type) Output Shape Param #

==========

embedding_3 (Embedding) (None, 300, 100) 22192100

_____

lstm_5 (LSTM) (None, 100) 80400

_____

dense_3 (Dense) (None, 7) 707

==========

Total params: 22,273,207

Trainable params: 22,273,207

Non-trainable params: 0

_____

Train on 4753 samples, validate on 1189 samples

Epoch 1/100

4753/4753 [==========] - 116s 24ms/step - loss: 1.8254 - accuracy: 0.2607 - val_loss: 1.8066 - val_accuracy: 0.2590

Epoch 2/100

4753/4753 [==========] - 122s 26ms/step - loss: 1.8009 - accuracy: 0.2712 - val_loss: 1.8080 - val_accuracy: 0.2574

Epoch 3/100

4753/4753 [==========] - 122s 26ms/step - loss: 1.7953 - accuracy: 0.2767 - val_loss: 1.7996 - val_accuracy: 0.2590

Epoch 4/100

4753/4753 [==========] - 123s 26ms/step - loss: 1.7850 - accuracy: 0.2819 - val_loss: 1.7852 - val_accuracy: 0.2767

Epoch 5/100

4753/4753 [==========] - 122s 26ms/step - loss: 1.7628 - accuracy: 0.2939 - val_loss: 1.7342 - val_accuracy: 0.3045

Epoch 6/100

4753/4753 [==========] - 122s 26ms/step - loss: 1.7212 -

4753/4753 [==========] - 121s 26ms/step - loss: 0.1829 - accuracy: 0.9360 - val_loss: 0.4559 - val_accuracy: 0.8654

.

.

.

.

.

.

.

.

.

.
.
.
.
.
.

Epoch 95/100
4753/4753 [==============================] - 121s 25ms/step - loss: 0.1823 - accuracy: 0.9333 - val_loss: 0.4605 - val_accuracy: 0.8696
Epoch 96/100
4753/4753 [==============================] - 122s 26ms/step - loss: 0.1799 - accuracy: 0.9400 - val_loss: 0.4633 - val_accuracy: 0.8612
Epoch 97/100
4753/4753 [==============================] - 122s 26ms/step - loss: 0.1806 - accuracy: 0.9415 - val_loss: 0.4557 - val_accuracy: 0.8663
Epoch 98/100
4753/4753 [==============================] - 121s 26ms/step - loss: 0.1682 - accuracy: 0.9411 - val_loss: 0.4722 - val_accuracy: 0.8621
Epoch 99/100
4753/4753 [==============================] - 121s 25ms/step - loss: 0.1643 - accuracy: 0.9455 - val_loss: 0.4657 - val_accuracy: 0.8671
Epoch 100/100
4753/4753 [==============================] - 121s 25ms/step - loss: 0.1685 - accuracy: 0.9430 - val_loss: 0.4716 - val_accuracy: 0.8671
testing Accuracy: 92.1308
training Accuracy: 94.3958

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a d egree in any other university, and that all source of materials used for the thesis have been duly a cknowledged.

Declared by:

Name: Girma Moges Misganew

Signature: _____

Date: 10/08/2020

Confirmed by advisor:

Name: Yaregal Assabie (PhD)

Signature: _____

Date: 10/08/2020