YARED YENEALEM AKLILU
EXPLORING NEURAL WORD EMBEDDINGS FOR AMHARIC LANGUAGE
NEU 2019

EXPLORING NEURAL WORD EMBEDDINGS FOR AMHARIC LANGUAGE

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES OF NEAR EAST UNIVERSITY

By
YARED YENEALEM AKLILU

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Software Engineering

NICOSIA, 2019

EXPLORING NEURAL WORD EMBEDDINGS FOR AMHARIC LANGUAGE

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES OF NEAR EAST UNIVERSITY

By
YARED YENEALEM AKLILU

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Software Engineering

NICOSIA, 2019

Yared Yenealem AKLILU: EXPLORING NEURAL WORD EMBEDDINGS FOR AMHARIC LANGUAGE

Approval of Director of Graduate School of Applied Sciences

Prof. Dr. Nadire Çavuş

We certify that this thesis is satisfactory for the award of the degree of Master of Science in Software Engineering

Examining Committee in Charge:

Asst. Prof. Dr. Boran Şekeroğlu	Department of Information Systems Engineering, NEU
Assoc. Prof. Dr. Yöney Kırsal Ever	Department of Software Engineering, NEU
Assoc. Prof. Dr. Kamil Dimililer	Supervisor, Department of Automotive Engineering, NEU

I hereby declare that this thesis has been composed solely by myself and all the information in it has been obtained and presented in accordance with the academic rules and ethical conducts. I also declare that, as required by these rules and conducts, I have fully cited and referenced all materials and results that are not original to this work. I further declare that this work has not been submitted, or will not be concurrently be submitted, in whole or in part, for the award of any other degree in this institute or any other institute.

Name, Surname: Yared Yenealem, Aklilu

Signature:

Date:

To my beloved late elder brother…

ACKNOWLEDGEMENT

Foremost, let the Most High be praised and honored as he had led me beside the still waters. My supervisor Assoc. Prof. Dr. Kamil Dimililier: Your guidance, continuous support and the patience and love you showed towards me had helped me a lot to carry out this work. You have been fully supportive from day one till the end. I thank you very much.

My special respect and gratitude should go to Asst. Prof. Dr. Boran Şekeroğlu for his unreserved support starting from the very first day of my graduate class till this work. You were very helping, caring and inspiring during my stay in Near East University.

I also very much grateful to the Ethiopian Ministry of Science and Technology (now rebranded as: Ministry of Science and Higher Education) for giving me the opportunity to pursue my masters here in this institute.

It will be a great omission not to thank my families (Sister Mastewal and Brother Dejazmach), friends and fellows.

It would be against the will and conscience of my mind not to render my heartfelt and warmest gratitude and thanks to my late elder brother, Habtamu Yenealem. May the Almighty God grant you the surest mercies of David.

ABSTRACT

Word embeddings are recent developments in natural language processing where words are mapped to real numbers for ease of operations on characters, words, subwords and sentences. Word embeddings for many world languages have been generated and a study is underway. Though Amharic is one of the most widely spoken language in Ethiopia, it is lagging behind in computational analysis including word embeddings.

Word embeddings capture different linguistic characteristics, which are intrinsic, such as word analogy, word similarity, out-of-vocabulary words and odd-word out operations. In this thesis, these characteristics and operations were explored and analyzed on Amharic language. Besides these intrinsic evaluations, the word embedding was evaluated on multiclass Amharic text classification task as an extrinsic evaluation.

FastText, a recent method to generate and evaluate word embeddings was utilized. This was used because of the morphologically richness of Amharic and the features of fastText in capturing sub-word information.

The resulting embedding using fastText showed that words that are similar or analogous to each other happen together or closer in space. Related Amharic words were found closer to each other in the vector space. Morphological relatedness took the highest stake. The word embedding has also learned the vector representation, "ንጉሥ(King) - ወንድ(man) + ሴት(woman)" resulting in a vector closer to the word "ንግሥት(queen)". Out-of-vocabulary words were also entertained. Multiclass text classification on the model attained 97.8% F1-score; result being fluctuated based on parameters.

Keywords: Word embedding; text classification; word relatedness; word analogy; Amharic language; fastText

ÖZET

Kelime gömme işlemleri, doğal dil işlemede, karakterlerin, kelimelerin, alt kelimelerin ve cümlelerin kullanım kolaylığı için kelimelerin gerçek sayılarla eşleştirildiği son gelişmelerdir. Birçok dünya dili için kelime yerleştirmeleri yapıldı ve bir çalışma devam ediyor. Amharca Etiyopya'da en çok konuşulan dilden biri olmasına rağmen, kelime gömme işlemleri de dahil olmak üzere hesaplama analizlerinde geride kalmaktadır.

Sözcük yerleştirmeleri, sözcük analojisi, sözcük benzerliği, sözcük dışı sözcükler ve garip sözcük çıkarma işlemleri gibi kendine özgü farklı dil karakteristiklerini yakalar. Bu tez çalışmasında, bu özellikler ve işlemler Amharca dilinde araştırılmış ve analiz edilmiştir. Bu içsel değerlendirmelerin yanı sıra, gömme kelimesi çok sınıflı Amharca metin sınıflandırma görevinde dışsal bir değerlendirme olarak değerlendirilmiştir.

FastText, kelime gömme işlemlerini üretmek ve değerlendirmek için yeni bir yöntem kullanıldı. Amharic'in morfolojik olarak zengin olması ve alt-kelime bilgisinin yakalanmasında fastText'in özellikleri nedeniyle kullanılmıştır.

FastText kullanılarak elde edilen sonuç gömme, birbirine benzer veya birbirine benzeyen kelimelerin bir arada veya uzayda daha yakın olduğunu gösterdi. İlgili Amharca kelimeler vektör uzayında birbirlerine daha yakın bulundu. Morfolojik ilişki en yüksek tehlikeyi aldı. Gömme kelimesi aynı zamanda "ንጉሥ(Kral) - ወንድ (erkek) + ሴት (kadın)" vektör gösterimini de "ንግሥት (kraliçe)" kelimesine daha yakın bir vektörle sonuçlamıştır. Kelime dışı kelimeler de ağırlandı. Model üzerindeki çoklu sınıf metin sınıflaması% 97.8 F1 puanına ulaşmıştır; Sonuç parametrelere göre dalgalanma.

Anahtar Kelimeler: Sözcük gömme; metin sınıflandırması; kelime ilişkililiği; kelime benzetmesi; Amharca dili; Fasttext

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN: Artificial Neural Network

CBOW: Continuous Bag of words

CNN: Convolutional Neural Network

GloVe: Global Vector

IDF: Inverse Document Frequency

NER: Named Entity Recognition

NLP: Natural Language Processing

NN: Neural Network

OOV: Out-of-vocabulary

PCA: Principal Component Analysis

POS: Part of speech

RNN: Recurrent Neural Network

SG: Skip-gram

SVM: Support Vector Machine

TF: Term-frequency

t-SNE: t-Distributed Stochastic Neighbor Embedding

WE: Word Embedding

XML: Extensible Markup Language

CHAPTER 1
INTRODUCTION

The distributional representation of words plays a crucial role in many natural language processing approaches. Words of a certain language, to be processed and understood by machines, need to be represented or converted into real numbers. As numbers are easier for operations by machines, words are mapped to real numbers. Those number representations are formed from different complex mathematical operations with a given dimension. This representation is called word embeddings or word vectors.

Vector representation of words will create a link between the two prominent fields of studies: mathematics and linguistics. This linkage and relationship between the two studies enables analysis of words and other linguistic features easier using algebraic methodologies.

Word embeddings, as a distributional representation of words in a variable sized dimension, capture different linguistic characteristics such as word similarity and word analogy.

The fact that related words will have a related representation vector gives us the chance to find similar words. That is, semantically similar, related words are mapped in the vector space very closer to each other.

The other characteristics that would be caught in using word embeddings is word analogy. Words that are represented as vectors are easier for mathematical operations. Amongst the mathematical operations that are usefully employed in this case are addition and subtraction of vectors of words. The famous relation: KING – MAN + WOMAN ==QUEEN is pulled from the beauty of vectors to lend themselves for operations. The analogy goes like this: As a *KING is to MAN, QUEEN is to WOMAN*. As studied by (Mikolov et al., 2013a) proportional analogies can be drawn from hypothetical vector operations. In this thesis, intrinsic and extrinsic evaluation of word embeddings for Amharic are explored thoroughly.

1.1 Statement of the Problems

Amharic is one of the morphologically rich Semitic language. Although it's the most widely spoken language, in terms of computational linguistic, it's lagging behind.

Word representation of almost all languages in the globe have been proposed by (Mikolov et al., 2016) through the Facebook's AI Research (FAIR) lab. This lab released an open-source library and a model called fastText, which is dedicated to the task of word representation and text classification. It uses neural network for word embedding and the lab makes available pre-trained models for 294 languages, among these languages Amharic being one of them.

fastText can be used to make word vectors using either CBOW or skip-gram (SG) models, plus it is also an efficient method for text classification. Because of the complex nature of Semitic languages in terms of morphology, a number of inflected forms, that often cause unknown words to appear (Tedla & Yamamoto, 2017) in the word representation, are generated. This case is even worse for low resource languages with no or little support of annotated resources like Amharic. For this reason, fastText algorithm is chosen for Amharic word embeddings.

Word embedding analysis involves both qualitative analysis and non-qualitative analysis. In qualitative analysis, the linguistic properties of languages like word similarity, word analogies and nearest neighborhood etc. are studied. On the other hand, other downstream tasks and non-qualitative factors such as text classification and NER are part of the analysis in the NLP arena.

Works on analysis and exploration of word embeddings on different languages exist but as Amharic is a low-resource language, in regards to digitization, there is little attempt on this topic.

Therefore, in this work the performance of fastText word embedding algorithms on Amharic language is analyzed and explored. Qualitative analysis using Word Similarities, Word analogies and nearest neighbor features, and non-qualitative analysis using multiclass Amharic text classification on Amharic Word Embeddings are the focus of the paper.

1.2 Thesis Objectives

1.2.1 General objectives

The general objective of this study is to investigate, analyze and explore Word Embeddings for Amharic language.

1.2.2 Specific objectives

The specific objectives of this study to achieve the overall objective are:

- Analyzing how different hyper-parameters on Word Embeddings can achieve different accuracy levels in relation to non-qualitative tasks
- To explore the morphological linguistic feature of Amharic on Word Embeddings such as: - Word similarity, Word analogy and Nearest neighbors
- Study Amharic Sentence embedding as a sideline on the given model
- Collection and Preprocessing of unlabeled Amharic dataset.
- Train multiclass Amharic texts
- Experiment and Review on different hyper-parameters on Amharic multiclass classification
- Investigating problematic cases such as how embeddings reflect cultural bias and stereotype
- Show how word embeddings act as a window onto history.

1.3 Methods and Techniques

In this paper work, fastText model is chosen for training word vector representation and Amharic multiclass text classification. The following methods will be applied in the progress of the research.

1.3.1 Literature review

Extensive literature review has been conducted on concepts, tools, models, architectures and algorithms related to word embeddings and text classification. Related works on the subject, focus being on word representation and multiclass classification, and a brief overview and introduction about the Amharic language is taken into consideration.

1.3.2 Tool selection

In this work fastText, a library for efficient learning for distributed word representations and text classification from Facebook AI Research (FAIR) lab, Word2Vec by (Mikolov et al., 2013a), GloVe by (Pennington et al., 2014), PCA and t-SNE for dimension reduction and word embedding visualization, Gensim which is a powerful NLP toolkit, matplot for plotting have been used. Other Python libraries are deployed in the backend such as Tensorflow, Keras, NumPy, and Scikit-learn.

1.3.3 Data collection and preparation

An Amharic news dataset is collected from the web from different websites including Amharic Wikipedia and Amhara Mass Media Agency, a local media in Ethiopia, which mainly serves in Amharic. The dataset will be manually annotated, preprocessed and cleaned.

1.3.4 Models

In this work latest and popular models have been used for generating and experimenting with word embeddings. Neural network based models CBOW and SG (Mikolov et al., 2013a) from fastText tool are used. CBOW (Continuous Bag-of-words) tries to predict the target word according to the context window size from the surrounding words. While SG (Skip-gram) tries to do the reverse - predicting the surrounding words known as context based on the target word (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013).

1.3.5 Evaluation and analysis

For evaluation and analysis, two types of dimensionality reduction techniques are used. The first technique is called PCA (Principal Component Analysis), which focuses on capturing the component and dimension of data during visualization. It's a linear deterministic algorithm (Feng, Xu, & Yan, 2012). The second technique is t-SNE (t-distributed Stochastic Neighbor Embedding) which is another popular dimensionality reduction technique that focuses on preserving local neighborhoods in the data. It's non-linear nondeterministic algorithm.

1.4 Scope and Limitation

A manually annotated Amharic news dataset, prepared by the Author, has been used for the multiclass text classification task. This dataset is small in size due to the absence of pre-annotated Amharic corpus. For the rest of the tasks such as qualitative analysis, both pre-trained word vectors and newly trained vectors from an Amharic Wikipedia and Amharic books are used.

The work focuses on how models perform on Amharic language datasets. One extrinsic evaluation on downstream tasks is chosen for this work, i.e., text classification.

1.5 Significance of the Study

Word embeddings are recent research areas in the NLP community. It's because word embeddings are crucial for various downstream tasks such as POS tagging, Sentiment Analysis, NER, Text Classification, Syntax Parsing and so on. Therefore, in order to study, design, analyze and improve those tasks, word embeddings should be thoroughly explored.

Therefore, this work will be an eye-opening for Amharic NLP research areas such as Sentiment Analysis. It will pave the path for improved text classification for many domains such as customer service, Spam detection, document classification.

1.6 Thesis Outline

In this outline, a cursory glance of the topics presented in different chapters of this thesis is provided.

Chapter 1 presents the introductory and overall background of the thesis. It begins with a background study to give a brief glance about the what of the works presented. The motivation behind the work, the objectives to be met and the questions to be raised and addressed along with the methodologies and tools required are discussed in this chapter. The theoretical and brief historical background of word embeddings and related topics, works of other researchers on different languages and the state-of-the-art methodologies, architectures, tools are reviewed, analyzed and presented in Chapter 2. The types, models, application and usage of word embeddings in various NLP tasks are also given a space in

this second chapter. A very brief overview about Amharic language and works on the language related to NLP is conducted as well.

The third chapter focuses on methodologies, architectures and approaches that are planned to be utilized in carrying out this work. The ways to represent words in vectors, depth analysis of models chosen, corpus preparation and training, and methods of evaluation are the core concepts covered here.

After making ready all the tools, techniques, models and training requirements, experimentation and result analysis is the work presented in Chapter 4. Here experimental setups, evaluation metrics and the analysis of the results obtained are discussed.

Finally, Chapter 5 concludes the thesis by providing insightful recommendations and future works.

CHAPTER 2
LITERATURE REVIEW AND RELATED WORKS

In this section, relevant concepts on and related literatures about word embeddings are reviewed. Characteristics of languages in distributional representation, the focus being on Amharic language, are widely recalled.

2.1 Overview

The advent of deep learning in the scientific arena has had a significant effect on natural language processing (NLP). NLP enables machines draw meaning from natural languages (Farzindar & Inkpen, 2015). It's a multidisciplinary area where software that analyze, interpret, understand and generate useful information from natural languages used by humans are designed and built. Though the area has been around for long time as trending topics for research and studies, important developments and impressive milestones in NLP have been observed in recent years. Amongst the many milestones that are capturing attention are Named Entity Recognition, Sentiment Analysis, Text Classification. These areas are briefly introduced below.

2.1.1 Named entity recognition

Named Entity Recognition (NER) is an Information Extraction task where important entities of a certain text, sentence, document or corpora are identified. Phrases that indicate person, organization, quantities, times, location are identified for the purpose of data mining and machine translation (Fu, 2009). The task of NER is, in short, to identify entities like person, location and organization.

The process in identifying those entities starts by identifying relevant nouns such as names and locations, and important facts such as dates and numbers that are mentioned in the given document. For example, given a statement S:

"Asrat Woldeyes was an Ethiopian surgeon, a professor of medicine at Addis Ababa University, and the founder and leader of the All-Amhara People's Organization." In this statement, named entities are:

[Asrat Woldeyes]*person*, [Addis Ababa University]*location*, and [All-Amhara People's Organization]*organization*.

The role of word vectors in NER emanates from the fact that NER requires the task of annotation which requires human time, cost and time (Sienc̆nik, 2015) and word vectors do not require pre-annotated dataset. Word vectors can be obtained from a training on large unannotated corpora, which can aid in augmenting the training of small annotated data in downstream tasks such as NER. This in turn reduces the amount of annotated dataset necessary and enhances the classification accuracy.

2.1.2 Sentiment analysis

The explosive nature of social medias, and the advance of technology brought the interaction of people on the Internet viable and inevitable. While playing with social medias and other streaming sites, people leave their opinions, reviews, tags, ratings etc. People give their opinions and sentiments for different reasons. They give opinions for products they use, about groups they are fan for, about institutions they are part of, their governments and social organizations, and others. People on social medias also reviews products and services etc. Companies and organizations always push their users to react and give feedbacks on the services and products they offer to know the opinion of their users. Reviews and opinions have tremendous effect on individuals for taking decision, say in choosing political candidates, buying branded items, and other every day activities. These reviews and opinions can, nowadays, be found in commentaries, blogs, micro-blogs, social media comments, reactions and postings.

The reviews, opinions and other activities of the users have polarities, either negative, positive, or neutral. Every word written, every utterance spoken holds sentiment information along with the context. The question now is how can those polarities are identified and analyzed for different usages.

Sentiment analysis, also called opinion mining, solved this problem by collecting, identifying, analyzing, synthesizing contextual polarity of texts, reviews, tags, and other activities of users. As the name clearly dictates, sentiment analysis is the process of analyzing intentions and sentiments in a given text, document or word. It might be to classify negative

and positive senses (binary sentiment analysis) or might include neutral sentiments. This has tremendously been used for opinion mining, customer reviews, product reviews (Turney, 2002), document classification (Pang et al., 2002) and so on.

Before the advent of deep learning models, sentiment analysis approaches were using traditional classification models such as Naive Bayes (Narayanan, Arora, & Bhatia, 2013) and Support Vector Machines (Cortes & Vapnik, 1995). The later model, Support Vector Machines, is also used in pattern recognition (Kirsal Ever & Dimililer, 2018). But now deep learning models are doing well for NLP tasks.

Now, all of the above tasks and others not listed here already are using linguistic elements like words. Be it sentiment analysis, or NER or any other natural language processing tasks always strive to manipulate words. Humans can understand raw formatted words and texts quite intuitively. Words, when spoken or written, are easier for humans but difficult for machines to understand, analyze and operate with them. These words should somehow be converted into machine-readable formats for ease of manipulation and calculation. The words, texts or documents should be represented, without altering their semantic, syntactic and contexts, by machine-readable representation so that machines can handle operations such as classification, analysis, recognition. According to (Firth, 1935) and (Harris, 1954), contexts of a word are essential to infer its meaning. Plus, the contexts in which two similar words are used is also observed to be very similar. To exploit concepts, properties and other features of texts, words, documents, and corpora at large, word embedding is the most widely used natural language tool (Bengio et al., 2003; Mikolov et al., 2010; Mikolov et al., 2013c).

2.1.3 Text classification

Text classification dates back to the early 60's, but got popular in the early 90's (Sebastiani, 2002). With the fast growth of online data, text classification is becoming one of the task of NLP. Information that is flowing over the social medias, or through different medias such as books, videos, and so on should be handled and organized. For efficient usage of data, for classifying news stories either by author or topic, to classify support tickets by urgency, to tag products by categories, to ease search in storages, and other related tasks are tackled using text classification.

Text classification is among the fundamental tasks in NLP to areas like sentiment analysis, intent detection and smart replies. The goal of text classification is to classify documents (such as review, opinions, news, messages, posts, replies, emails, etc...) to specified categories. It involves assigning predefined tags to free-text documents (Zhang et al., 2015). The tags or categories can vary from two (binary-classification) to n (multi-label or multi-class classification).

We can find unstructured and unlabeled raw data in the form of text anywhere in social networking sites and media, chat conversations, email messages, web pages and more. Due to its unstructured nature, however, extracting insights and useful information from those raw data takes time and energy. These days, text classification is used by businesses for structuring, automatic labelling and extraction, balancing documents and texts in a cost-efficient way for automation processes and enhancement of decision-making.

For example, given a text *t*, a classifier can take the content of the text *t*, analyze its content and then automatically assigns relevant categories.

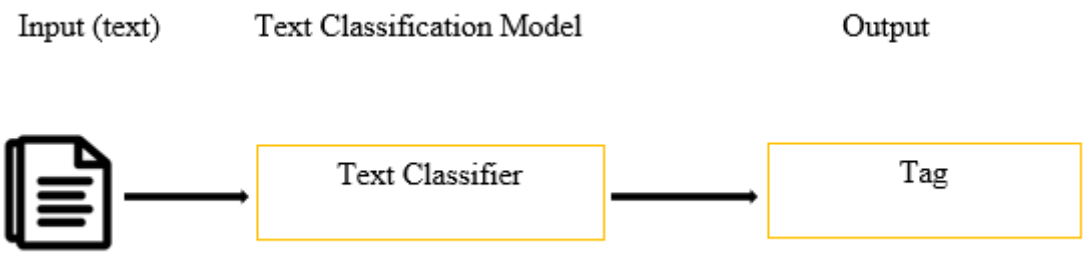

Figure 2.1: Text classification flow

2.1.3.1 General definition of classification

The general text classification problem can formally be defined as the process of predicting a new category assignment function $F : D \times C \rightarrow \{0,1\}$, where D is the set of all possible data and C is the set of predefined categories. The value of $F(d, c)$ is 1 if the text or document or data d belongs to the category c and 0 otherwise (Feldman & Sanger, 2007). The predicting function $F: D \times C \rightarrow \{0,1\}$is a classifier, that produces results as "close" as possible to the actual category assignment function F.

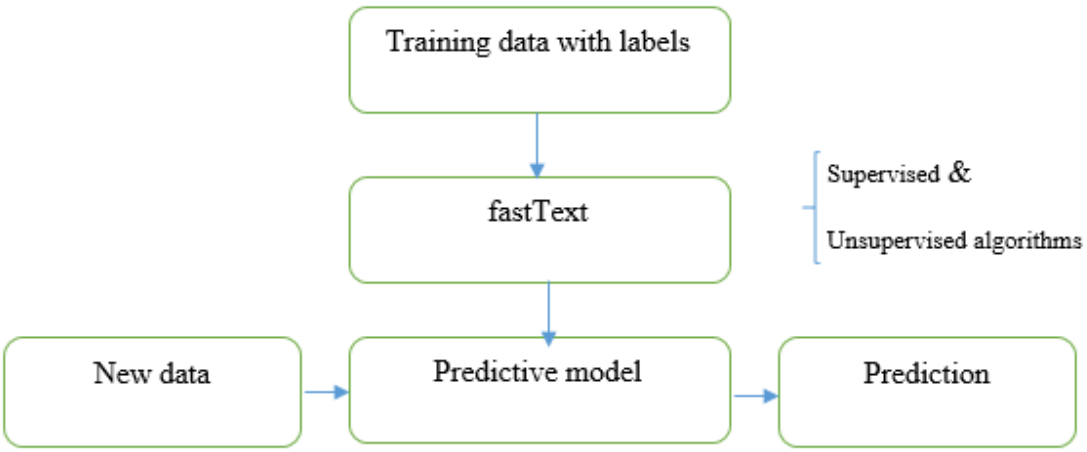

Figure 2.2: Classification Block Diagram

2.1.3.2 Multilabel versus multiclass classification

Based on the properties of function F in Section 2.1.3.1, classifications can be distinguished as multilabel and multiclass classification. In multilabel classification, labels might overlap and data may belong to any number of labels. It assigns to each sample a set of target labels. It is like predicting properties of a data-point which are mutually exclusive such as topics that are relevant for a sample. A text might be about any of football, athletics, baseball or chess at the same time or none of these. In general, multilabel classification assigns a text, data, or sample to one or more than one, or no label at all.

However, if the text, data, or sample or document belongs to exactly one class or label, it is known as multiclass classification. Here each sample belongs to exactly one category as the classes or labels are mutually exclusive. It assigns each sample to one and only one label.

In this work, the second type, i.e., multiclass classification is chosen for evaluation technique.

2.1.3.3 Approaches to text classification

In text classification (sometimes called text categorization) different approaches were evolved through the ages in the field. Before automation day-to-day tasks in the life of man were manual. Text classification was not an exception. The first successful approach used

for text classification was to manually build classifiers based on knowledge engineering (KE) techniques (Krabben, 2010). This technique requires manual annotation, parsing, syntax check and syntactic rules or patterns. The drawback of this approach is, however, that it depends on knowledge of the expert to hand-craft rules. This will hinder portability and maintenance of the system.

According to (Krabben, 2010), Machine Learning techniques became increasingly popular in text classification task in the 90's. This technique automates the task of classification by automatically building a classifier which learns the characteristics of each category from a set of labeled datasets. This approach is also not without drawbacks. Machine learning approaches do need to be trained on predefined categories and their efficiency depends on the quality of the training datasets.

In general text classification can be done either manually or automatically. In the former, it's a human that annotates, interprets and categorizes the text. This gives quality results with time trade-off. It's expensive, time-consuming and laborious. The speed, diligence and efficiency of humans affect the result. The latter applies NLP, deep learning and other methods to automate classification in a faster and more cost-effective approach. In this second ways, there are different approaches to classify text automatically. They range from rule-based systems to machine learning based systems. Some are even in between, called hybrid systems.

In rule based approaches, handcrafted linguistic rules articulated by linguists are used to organize text. The system uses those rule to semantically or syntactically identify relevant tags based on contents. As the rules are articulated by men, these types of approaches can be improved over time. The problem with these models are they are dependent on the skill of the linguist and the systems depend on knowledge of the domain. Most of the time experts and knowledgeable persons are required to use rule-based approaches.

With the machines learning ability, the need of manually crafted rules is questioned. machines can learn and classify texts based on previous observations. Machines are given pre-labeled training data so that machine learning algorithms can learn various correlations in texts and that a particular tag is expected for a particular input.

In machine learning based systems, before training a classifier, feature extraction is conducted. Feature extraction is the process of transforming the data into a numerical representation or into vectors. Different approaches such as bag of words and n-gram model can be used for transformation. Then the machine learning algorithm is fed with the vector and the tags to produce a classification model.

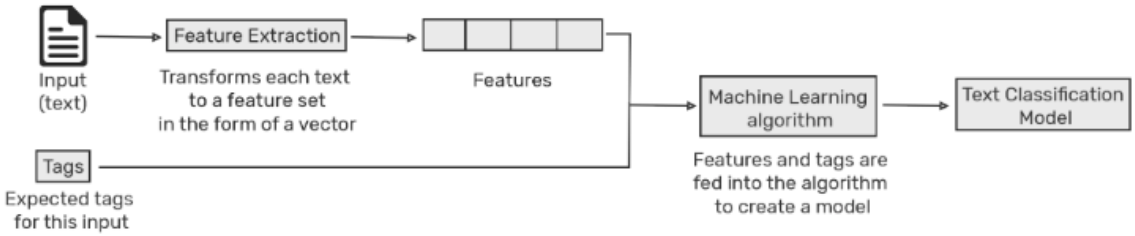

Figure 2.3: Steps in machine learning based Classification

2.1.3.4 Text classification algorithms

There are various machine learning algorithms for text classification modeling. The popular ones are: Naïve Bayes, SVM and deep learning.

a) Naïve Bayes

Naïve Bayes is a statistical algorithm based on Bayes' theorem. In this model, each feature is considered independent and the conditional probabilities of occurrence of words are computed. In text classification, the Bayes' Theorem calculates the probability of each label for a given text and then output the label with the highest one (Pawar & Gawande, 2012).

Among the Naïve Bayes family of algorithms, Multinomial Naïve Bayes (MNB) is the one which focuses on a multinomial distribution of features. MNB is a probabilistic model that computes class probabilities for a given dataset using Bayes' rule. Assume there are *N* vocabularies and *C* set of classes. Then MNB assigns a test sample di to the class that has the highest probability $P(c|d_i)$, which is given by:

$$P(c|d_i) = \frac{P(C) \times P(d_i|c)}{P(d_i)}, c \in C \quad (2.1)$$

where,

$$P(d_i) = \sum_{k=1}^{|c|} P(k)P(d_i|k) \tag{2.2}$$

The class prior $P(c)$ is the number of samples belonging to class to the total number of samples ratio. The probability of obtaining a sample like d_i in class *c* is represented as:

$$P(d_i|c) = \left(\sum_n f_{ni}\right)! \prod_n \frac{P(w_n|c)^{f_{ni}}}{f_{ni}!} \tag{2.3}$$

where f_{ni} is the number of word *n* in our test sample d_i and $P(w_n|c)$is the probability of word *n* given class *c* and $P(w_n|c)$ can be computed using

$$P(w_n|c) = \frac{1 + F_{nc}}{N + \sum_{x=1}^{N} F_{xc}} \tag{2.4}$$

where F_{xc} is the number of word *x* in all the training samples belonging to class *c*, and the Laplace smoothing technique is used to prime each word's count with one to avoid the zero-frequency problem (Kibriya et al., 2004). The final normalized computationally inexpensive equation would be:

$$P(d_i|c) = \alpha \prod_n P\,(w_n|c)^{f_{ni}} \tag{2.5}$$

where α is a constant diminished due to the normalization using Laplace estimator.

b) Support vector machines

Support Vector Machines are algorithms that divides a space into subspaces and its objective is to find the line or the hyperplane that has the maximum margin in an N-dimensional space that uniquely classifies the data points.

SVM determines the optimal decision boundary between vectors that belong to a given tag and vectors that do not belong to it. This algorithm draws the best "line" or hyperplane that

divides the vector space into two subspaces: one for the vectors belonging to the given tag and the other which do not belong to it.

SVMs try to maximally position a separating line in a high-dimensional feature space such that it divides the data points belonging to various classes, projected into the space from input very well using kernel functions (Cortes & Vapnik, 1995).

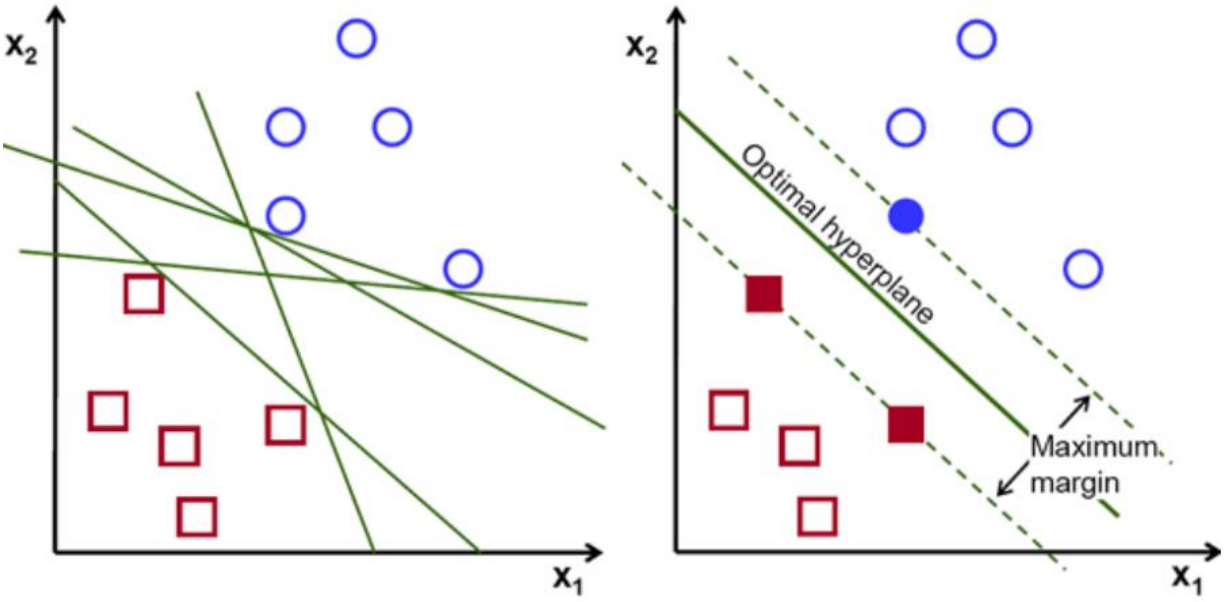

Figure 2.4: Support vector machines

In Figure 2.4 there are data represented by circles and squares. New, unclassified data can be assigned to either circles or squares (both used as labels or tags) using SVMs. To do this, SVMs use a separating line (hyperplane if multi-dimensional) to split the space into a circle zone and a square zone (see the second figure in Figure 2.4). The distance to the nearest point on either side of the separating line is known as the margin, and SVM tries to maximize the margin. The separating line or the hyperplane needs to satisfy two requirements to be optimal: (1) Cleanly separating the data, with circles to one side of the line and squares on the other side, and (2) maximize the margin. The first constraint, i.e., clean separation of data, is not easy in the real world. Therefore, SVM deals with this problem by softening the definition of "separate". This is done by allowing a few mistakes, hence loss function by adding a cost for misclassification. The other way is by increasing the number of dimensions to create a non-linear classifier.

SVM is a binary classifier developed by (Cortes & Vapnik, 1995). The algorithm maps input vectors to a very high-dimensional feature space, where the data can be optimally separated by a single hyperplane. By optimal it means widest possible margin is selected for the separating hyperplane to any of the training datasets. The two most important issues SVM takes into consideration are high dimensional input space and linearly separable classification problems.

c) Deep learning

Different deep learning models such as Convolutional Neural Network(CNN), Recurrent Neural Networks (RNN) and Hierarchical Attention Networks (HAN) are found very effective in the work of text classification.

Deep learning models have gained popularity in computer vision (Krizhevsky et al., 2012) and speech recognition (Graves et al., 2013) in recent years. Within NLP, much of the work with deep learning methods has involved learning word vector representations through neural language models (Bengio et al., 2003; Mikolov et al., 2013b) and performing composition over the learned word vectors for classification (Collobert et al., 2011).

CNN is a class of deep learning, feedforward ANNs that uses a variation of multilayer perceptron designed to require minimal preprocessing. ANNs are nowadays vastly used not only in text processing but also in image processing applications. Implementing ANNs in image processing applications is now trending (Khashman & Dimililer, 2008). This network exploits the spatial structure of data to learn about it so that useful output can be obtained. CNN models, originally invented for computer vision, utilize layers in word vectors to extract local features. In NLP, the features as an input usually take the form of word vectors. The input to a CNN, given a tokenized text $T = \{t_1, \dots t_N\}$, is a text matrix A where the i^{th} row is the word vector representations of the i^{th} token in T. The matrix A can be denoted as $A \in R^{N \times d}$ where d is the dimensionality of the word vectors.

CNNs use convolutional layers which are like a sliding window over a matrix. CNNs are many layers of convolutions with non-linear activation functions. In CNNs the output is computed over the input layer from local connections and then each layer applies different kernels, usually thousands of filters, to then combine their results.

According to (Kim, 2014) CNNs perform remarkably well for classification by using different tuning of hyperparameters. CNNs utilized the distributed representation of words after converting the tokens comprising each sentence into a vector which forms a matrix to be an input. However, these models require setting hyperparameters and regularization of parameters (Zhang & Wallace, 2015). Other issues like the higher training time, expensive configuration cost, vast space of possible model architectures and hyperparameter settings etc. are counted as downside of CNNs (Zhang & Wallace, 2015).

The other family of deep learning methods is the Recurrent Neural Network (RNN). RNN is a class of ANN where connections form a recurrent node (or a directed graph) along a sequence. They are networks with loops that aids in persistence of information. RNNs improved the traditional Neural Networks which considers all inputs as independent to each other by gaining memory and capturing information in arbitrary long sequences and predicting the previous and next sequences in the networks. They are deep in temporal dimension and used in time sequence modeling. The role of RNNs in text classification is to recurrently and sequentially process words in a sentence and map a dense and low-dimensional representation of words into a low-dimensional vector.

One of the feature of RNNs is their capability to improve time complexity and analyze texts word by word there by preserving the context of texts. This ability arises from their way of capturing the statistics of a long text. In this perspective RNNs has fall short of balancing the role of both earlier words and recent words. This issue can be overcome by introducing long short-term memory(LSTM) model.

Hierarchical Attention Network(HAN) was designed to capture document hierarchies (words→ sentences→ paragraphs→ articles→ document) and context of the words and sentences in a document. But the whole words in a document are not treated equal; as the word “attention” says it all, and since all words do not equally contribute to the representation of sentences meaning, the importance of words should be weighed by introducing an attention mechanism. The attention mechanism is effected to reward those sentences that hold strong features so as to properly classify a document.

Generally deep learning approaches start with sequence of word as an input in which the words are presented as a 1-hot vector. The words in the sequence are then projected into a

contentious vector space after multiplied by a weight matrix which forms a sequence of real value. The sequences are then fed into a deep NN, which processes the word sequence in multiple layers resulting in a prediction probability. “This whole network is tuned jointly to maximize the classification accuracy on a training set. However, one-hot-vector makes no assumption about the similarity of words, and is also a very high dimensional” (Hassan & Mahmood, 2017, p. 1108).

The above mentioned models such as by (Kim, 2014) achieve a good performance in practice, but they are slow at training and testing time (Joulin et al., 2016). To alleviate this limitation (Joulin et al., 2016) came up with another approach called fastText. This approach can be used both for sentence, document or text classification and word representation.

2.2 Word Embedding

The idea of word embeddings and representations has its roots in linguistics and language philosophy, especially in the works of (Harris, 1954) and (Firth, 1935) in 1950s. For example, (Osgood, 1964) used feature representations to quantify semantic similarity using hand-crafted features. In the early 1950s scholars used the semantic differentials technique to measure the meaning of concepts.

Methods for using contextual features were later devised in 1990s in different thematic study areas. The most known one was Latent Semantic Analysis(LSA). LSA is a technique, in NLP in general and in distributional semantics in particular, used to analyze relationships between documents and the words inside them by making a set of concepts related to the documents and words. This technique assumes the distributional hypothesis which states that related words occur in similar pieces of text and constructs a matrix that has word counts per paragraph from a corpus. It utilizes a technique called Singular Value Decomposition(SVD) to reduction of the number of rows in the matrix while making the linguistic features intact. The linguistic features such as the contextual-usage meaning of words are extracted and represented by statistical computations applied to a corpus of text. This helps to estimate the continuous representations of words.

At roughly the same time, Latent Dirichlet Allocation(LDA) was proposed by (Pritchard et al., 2000) in the context of population genetics. This scheme was rediscovered in 2003 in the

context of machine learning by (Blei, 2003). LDA is a generative probabilistic model of a collection of documents and words and/or phrases. According to the authors, LDA can be used for collections of any discrete data, be it DNA and nucleotides, molecules and atoms or keyboards and crumbs.

Another well-known models developed on neural networks that used contextual representations are Self Organizing Maps(SOM) and Simple Recurrent Networks(SRN). The former, developed by (Kohonen, 1982), uses unsupervised, competitive learning to produce low-dimensional representation of high-dimensional data, while keeping, at the same time, similarity relations between data items intact. The later was conceived and used by (Elman, 1990). This is a version of the backpropagation NN that processes sequential input and output. It is a 3-layer NN where the hidden layer activations are potentially used as input. First the copy of the hidden layer functions is prepared and saved. Their results and their copy is used as input to the hidden layer in the next time step. In this case, the previous hidden layer from which its copy is saved and its results are transferred to the next is fully connected to the layer next to it. Since the network has only the copy, backpropagation algorithm is used for training. SRN "can be trained to read a sequence of inputs into a target output pattern, to generate a sequence of outputs from a given input pattern, or to map an input sequence to an output sequence" (Miikkulainen, 2010).

Though the idea behind word embeddings were found in the early works of (Harris, 1954) and (Firth, 1935), the appearance of automatically generated contextual features, and deep learning methods for NLP gives word embeddings the chance to be the most popular research areas in the early 2010's (Mandelbaum & Shalev, 2016). Since then various developments and different embedding models were evolved.

Latent Semantic Analysis(LSA) for information retrieval, Self-Organizing Maps (SOM) - a distributional way to map words to 2-dimensions, such that similar words are closer to each other (Ritter & Kohonen, 1989) - for competitive learning and visualization, Simple Recurrent Networks (SRN) for contextual representations, Hyperspace Analogue to Language (HAL) for inducing word representations (Lund & Burgess, 1996) etc. were developed both in computational linguistics and ANNs. Later developments use these

models as a basis. The various refined models vary based on the type of contextual information they employ. Some use documents as contexts, others use words etc.

Later (Collobert & Weston, 2008) show the power of pre-trained word vectors as a tool for downstream tasks ranging from structural linguistic features, such as POS tagging to meanings and logic behind languages, such as word-sense disambiguation. The authors also introduced a single CNN architecture that defies older systems.

However, it was (Mikolov, Chen, et al., 2013) who made the eventual popularization of word embeddings after they released word2vec. Following the release of the word2vec toolkit, word embeddings became the latest in natural language processing. This sparked a huge amount of interest in the topic.

In 2014, Pennington et al. released GloVe, another model for unsupervised learning of word representations, which brought word embeddings to the mainstream NLP. This model develops a co-occurrence matrix using the global statistics of word-word co-occurrence. GloVe (stands for *Global Vectors*) uses the strengths of word2vec skip-gram model for word analogy tasks and matrix factorization methods for global statistical information

(Bengio et al., 2003) was the first person to coin the term Word Embeddings. As per the terminology, word embedding has different names like distributional semantic model, distributed representation, semantic vector space and so on. On this paper, the popular term - word embedding- will be used.

Word embedding is the task of converting words, strings, or characters into machine-readable formats specifically vectors. It is a means of representing a word as a low dimensional vector which preserves the contextual property of words (Mikolov, Sutskever, et al., 2013).

A word embedding as the name indicates embeds words into a vector space. It associates each word with a vector in a manner that relationship between words are preserved. Relatedness between words are reflected through the relations between vectors. In this vector representation, “similar words are associated with similar vectors” (Collobert & Weston, 2008; Mikolov et al., 2013c). The vectors associated with words are called word

embeddings, also known as word vectors (Basirat, 2018). Words are converted into real numbers. Therefore, word embedding can be described as vector representation of a word.

It's used in various tasks in deep learning and natural language processing (NLP), such as sentiment analysis, caption generation (Devlin et al., 2015), named entity relationship (Turian, Bengio, Ratinov, & Roth, 2010), machine translation (Bahdanau et al., 2014).

2.2.1 Types of word embedding

Word embeddings are classified into two broad categories: -

a) Frequency based embedding and; b) Prediction based embedding. These types are discussed as follows.

a) Frequency based embedding

In frequency-based embedding, various vectorization methods are employed. Amongst the methods the widely known are count vector, TF-IDF (Term Frequency-Inverse Document Frequency) vector and co-occurrence vector.

In count vector method, the number of times each word appears in each document is assessed. For example, suppose there are **D** documents, **T** number of different words from all documents (called vocabulary). Then the size of the count vector matrix will be $D \times T$.

This method faces primarily two problems. First, the size of the vocabulary and the dimension (after multiplication) would be very huge for bigger corpus. For big data, with millions of documents, hundreds of millions of unique words can be extracted. Therefore, the matrix would be very sparse and inefficient for computation. Second, there is no clear way to count each words, whether using frequency method or just based the words presence. In this second point, if frequency of words is considered, in real life corpus, the least important words like stop words, punctuation marks etc. are the most frequent ones. This poses another problem. For this case, TF-IDF vectorization is the solution.

TF-IDF stands for Term-frequency - Inverse Document Frequency. Term frequency(TF) shows the number of times a term or a word occurs or just frequency of occurrence (Salton & Buckley, 1988) in a document. Term frequency, still, cannot the problem that count vector faces due to most frequent-least relevant terms in documents. IDF just diminishes the

occurrence of most frequent terms in a document and increases the weight of terms that are rare. IDF takes into account the totality of a word by measuring how importantly rich a word is or whether the word is a common word or not. It is the logarithm of the quotient of the ratio of the total number of documents to the number of documents where the term t appears, as shown in the following equation.

$$idf(t,d) = \log \frac{N}{|\{d\ \varepsilon\ D : t\ \varepsilon\ d\}|} \tag{2.6}$$

where N is number of documents in the corpus N = |D|

$|\{d\ \varepsilon\ D : t\ \varepsilon\ d\}|\ \rightarrow$ number of documents where term t is part of ($tf(t,d) \neq 0$).

If term t is not part of the corpus, the denominator will be adjusted to 1 + $|\{d\ \varepsilon\ D : t\ \varepsilon\ d\}|$to avoid division by zero.

Then

$$tf - idf(t,d,D) = tf(t,d)\ .\ idf(t,D) \tag{2.7}$$

The other method worthy to discuss is the co-occurrence matrix. This captures the extent words occur together so that relationships between words are also captured. This is done simply by counting how words occur or found together in a corpus.

b) Prediction based embedding

Frequency based methods have been used for many natural language tasks such as sentiment analysis and text classification. However, after the introduction of word2vec (Mikolov, et al., 2013b) to the NLP community, the frequency based methods are proven to have limitations. Vo & Zhang (2016) have, in their work about learning sentiment lexicons, pledged not to count, but to predict. Therefore, prediction based methods are becoming the state of the art for tasks performed using word embeddings such as word similarities and word analogies (Mikolov et al., 2013c).

These methods are associated with the advent of neural network architectures. In this perspective, (Mikolov et al., 2013c) introduced two neural network architectures for word

vector computation. The authors aim was to introduce methods for learning high-quality word vectors from big data sets. According to the Authors, these model architectures are very effective in minimizing computational complexity by optimizing the hidden-layer in the model. The model, to boost performance, has been designed with a simple projection layer instead of the hidden layer.

One of the architectures proposed is a feed-forward NN like the language model of (Bengio et al., 2003) by removing the non-linear hidden layers. This model is known as a continuous bag-of-words (CBOW) model. This method learns an embedding by predicting the target word based on nearby words. The nearby words are surrounding words which determine the context. Basically, in CBOW model, the average of the vectors of the surrounding words is given to the neural network for predicting the target word, which appears in the output layer. The architecture is dubbed a bag-of-words model as the order in which words occur does not influence the prediction. Words before the target, history and words after the target, future, are evaluated (their vectors are averaged). The model architecture is shown below at Figure 2.5.

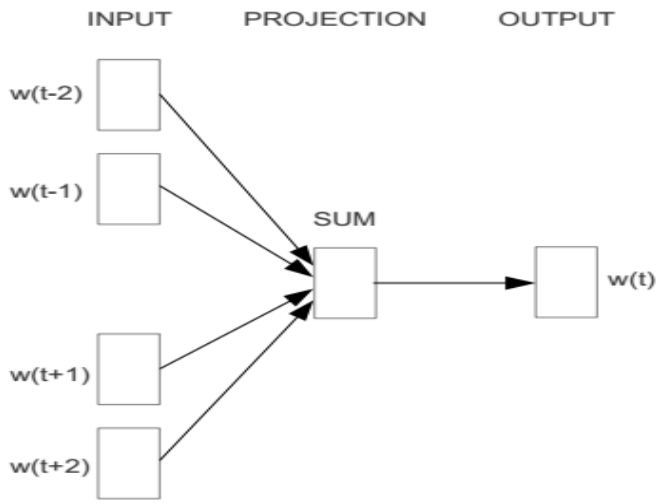

Figure 2.5: The CBOW architecture

The second architecture introduced by (Mikolov et al., 2013c) is the continuous SG model. This method is almost the inverse of the above method, but instead of predicting the target word, it predicts the surrounding words. This method, given the target word, tries to predict

the context, or nearby words. For a sequence of words, the continuous SG model takes the word in the middle of the sequence as input and predicts the words within a window size range before or after the input word (Basirat, 2018). The architecture of continuous SG model is depicted at Figure 2.6.

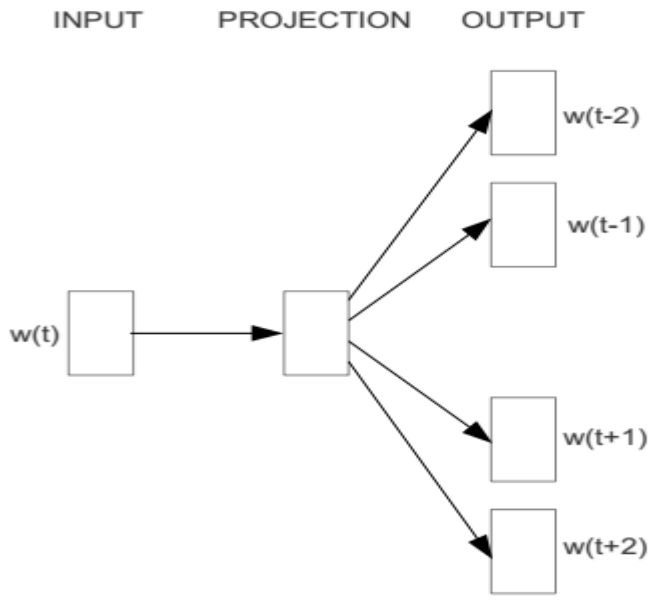

Figure 2.6: Continuous skip-gram architecture

Amongst the two architectures, CBOW model performs better in tasks involving small datasets because the model treats the entire context as one observation and it smooths over the distributional data at the averaging stage. On the other hand, for huge dataset, SG model is better and fine-grained and essentially outperforms every other method. Mikolov et al. (2013c) showed that skip-gram works better on semantics and worse on syntactic tasks.

2.3 Word Embedding Based Models

In literatures, several techniques are proposed to build word embedding models (Basirat, 2018). The most popular are word2vec, GloVe and fastText. Each embedding models are discussed thoroughly in Chapter 3 in Section 3.2. Here a brief overview is provided.

a) Word2vec

Word2vec, a shallow model developed by (Mikolov, Chen, et al., 2013), was one of the first neural model for efficient training of word embeddings. It learns low dimensional vectors

and predicts words based on their context using the two famous neural models: CBOW and SG. The introduction of these two simple log-linear models drastically reduces the time complexity, increases scalability and reliability of training word embeddings. Word2vec starts with a set of word vectors that are random. It scans the dataset in orderly fashion, always keeping a context window around each word it is neighboring with. Word2vec uses target words and context very tightly to observe how they behave throughout the corpus. The algorithm computes the dot product between the target word and the context words and tries to minimize this metric performing Stochastic Gradient Descent (SGD). Each time two words are encountered in in a similar context, their link, or spacial distance, is reinforced. The more evidence is found while scanning the corpus that two words are similar, the closer they will be.

The problem here is that the model only provides positive reinforcement to make vectors closer. This leads, with a huge corpus at minimum state, to the state that all vectors would be concentrated in the same position. To address this issue Word2Vec initially proposed a Hierarchical Softmax regulator at first then a Negative Sampling later. The latter is simpler and has been shown to be more effective. The basic premise is that each time the distance between to vectors is minimized, a few random words are sampled and their distance to the target vector is maximized. This way, it is ensured that nonsimilar words stay far from each other.

b) GloVe

Amongst word embedding models, like word2vec, GloVe is a well-known algorithm. The aim of GloVe is basically creating word vectors that capture meaning in vector space and taking advantage of the global count statistics instead of only local information. Glove learns embedding through a co-occurrence matrix and weights loss based on word frequency.

b) fastText

fastText is one of the text classification and word representation models that utilizes unsupervised learning techniques to make word representations. It is an extension of word2vec which views word representation from a different angle. The problem of predicting context words by Skip-gram can be tackled with a classification tasks. Thereby,

fastText model takes into account the real essence of words, such as literals and characters from which a word is composed of and introduced the idea of modular embeddings. fastText represents sentences as bag of words and train a classifier (Joulin et al., 2016). One of the peculiar feature of fastText is its ability to generate vectors for out-of-vocabulary words including unknown words and tokens. fastText considers not only the word itself but also groups of characters from that word and the subword information such as character unigram, bigram, trigrams etc. during learning word representations

2.4 Amharic and Amharic Word Embeddings

In this section, the Amharic language with respect to word embeddings and natural language in general will be discussed. Some aspects of the language, history and background of Amharic is also included.

2.4.1 Overview of Amharic

Amharic (Amharic: አማርኛ, Amarəññа) is the official language of Ethiopia. It is the Semitic language that is second most spoken Semitic language in the world next to Arabic. In Ethiopia, it's the first largest language, with a rich literature history, and has its own alphabets.

The alphabet of Amharic is called Fidel, which is, unlike Arabic, run from left to right and consists of 34 basic characters each having seven forms for each consonant-vowel combination. It has 4 characters (though variants of other characters are also used nowadays) with labiovelars. Other labialized consonants that are extended from basic characters are about 20. In total Amharic has more than 270 characters. Each character represents a consonant+vowel sequence.

Amharic is spoken by more than 90 million (Negga, 2000) people as their first and the rest population as their second language. The majority of monolingual Amharic speakers are the Amhara people (the name Amharic or Amarəññа is derived from the name of the people of Amhara -ʾÄməḥära[1]) of the Ethiopia. Furthermore, a great number of monolingual Amharic speakers live in bigger town and administrative centers all over the country (Appleyard: cited

[1] Amhara means free or independent people.

in Meyer, 2006). As more Ethiopians are living outside their home, the Amharic language speakers in different countries of the world is also growing. In Washington DC, Amharic has got the status to be one of the six non-English languages (Bernstein et al., 2014).

Amharic became the royal language of Ethiopia and was made the national (vernacular) language of the state during the reign of Emperor Yekuno Amlak, c., 1270 AD. Amharic is influenced by both Ge'ez (the language liturgy of the Ethiopian Orthodox Tewahedo Church-an ancient Christian Church established in 34 AD.) and the Cushitic languages such as the Agew.

Ethio-Semitic languages are Semitic languages spoken mainly in Ethiopia and modern day Eritrea. It includes Amharic, Geez, Tigre, Tigrinya, Argobba (closely related to Amharic), Harari (or Adare, spoken in Harar), Gurage (a cluster of at least twelve dialects) and Gafat (almost extinct).

According to (Woodard, 2008), most of the languages now spoken in Ethiopia as Semitic language are considered sister (or rather niece) languages. The Author put Amharic, Geez and Tigrinya and other Semitic languages under an Ethio-Semitic family. More clearly, according to (Armbruster, 1908), Amharic is niece to Geez (sometimes called Ethiopic). The Author further expressed that Proto-Ethiopic-Semitic evolved and split into Southern Semitic (Amharic) and Norther Semitic (Geez) or their intermediaries. Amharic and Geez come from the same root and are offspring of a common Ethiosemitic proto-language.

Though the language has tremendous resources in terms of resources, literatures, novels and other linguistic features and the abundance of both electronic and non-electronic documents, it’s considered one of the low-resource languages for natural language processing tasks. It has very low computational linguistic resources.

Amharic is under-resourced and has very few computational linguistic tools or corpora. According to (Gambäck et al., 2009), Amharic is spoken nation-wide and is the lingua-franca of Ethiopians (Weninger, 2011).

2.4.2 Amharic word embeddings

Word embeddings for different languages have been done. Experimenting on the word embeddings are mainly done by different techniques. Tripodi & Pira (2017) analyzed the

performance of skip-gram and continuous bag of words on Italian language, training being taken from Italian Wikipedia. They adopted a word analogy test and evaluated the generated word embeddings. The experiment is conducted by fine-tuning different hyper-parameters such as the size of the vectors, the window size of the words context, the minimum number word occurrences and the number of negative samples. They found out that due to the rich morphological complexity of Italian language, increasing the number of dimensions and negative examples improve performance of the two models in terms of semantic relationships and on the contrary the syntactical relationship is negatively affected by the low frequency of number of terms. Their work investigated major ideas like how different hyper-parameters can achieve different accuracy levels in relation recovery tasks; morpho-syntactic and semantic analysis and qualitative analysis to investigate problematic issues.

A work on Croatian language, one of the morphologically rich language spoken in the Republic of Croatia, by (Vasic & Brajkovic, 2018) showed that pre-trained fastText model results best output and fine tuning the parameters can even provide greater results. The authors used the Croatian Wikipedia and other corpuses for training and an experiment on latest models like word2vec and fastText showed that the results are bad for morphology rich languages such as Croatia. They showed that fastText (pretrained CBOW) approach produced better results, may be because the subword information used by fastText takes care of the morphology in the words (Vasic & Brajkovic, 2018). On this same language (Svoboda & Beliga, 2018) performed an evaluation but now adding specific linguistic aspects of the Croatian language. They did a comparative study of word embeddings for Croatian and English languages. The comparison in their experiment showed that the models for Croatian does not render a good result as for English.

The word embeddings of Polish, a highly inflectional language spoken primarily in Poland, was tested by (Mykowiecka et al., 2017) using word2vec tool by adjusting various parameters for tasks like synonym and analogy identification. They reported that word embeddings can be used for linguistic analysis such as similarity and analogy for Polish words and that the efficiency of the method highly depends on dataset and parameter tuning.

Arabic is one of the most spoken Semitic language in the world. It is highly related with Amharic language. Many researchers have tried to analyze and evaluate the Arabic word

embeddings. One is the work of (Elrazzaz et al., 2017) to perform a methodical evaluation of the Arabic word embeddings. They have performed both intrinsic and extrinsic evaluation on the embeddings. They described intrinsic evaluation as an evaluation that mostly relies on word similarity correlation datasets and analogy questions and describe the linguistic features in the low-dimensional embedding space while extrinsic evaluation is an evaluation that assesses the quality of the embeddings as features in models for downstream tasks like POS tagging and text classification. The authors compared CBOW and SG with another word embedding model called Polyglot, contributed by (Al-Rfou et al., 2013), in which the former models were found superior. (Soliman et al., 2017) also have crafted a set of Arabic word embedding models using data from twitter, the WWW and Wikipedia. They used Gensim to build the models and evaluate their models both qualitatively and quantitatively. Their models achieved well in both cases.

Tigrinya, a very close Semitic language to Amharic, spoken in Ethiopia and Eritrea was analyzed for its word embeddings to improve POS tagger by (Tedla & Yamamoto, 2017). The authors constructed a new text corpus, as Tigrinya has very little support of annotated resources, and investigated the optimal hyperparameters for generating word vectors for Tigrinya. They showed that the dimension of context affects the quality of semantic and syntactic relatedness of words. While wider context gives better semantic relatedness, shorter context renders syntactic relatedness.

As far as the author of the paper knows, there is no Amharic word embedding analysis so far. However, for downstream tasks such as NER, (Demissie, 2017) used Amharic neural vectors as a feature to design Amharic Named Entity Recognition system. Word vectors have just been used for classification and the author has left a recommendation for readers to investigate the impact of using word embeddings on Amharic Named Entity Recognition system.

2.4.3 Works on Amharic text classification

Researches on Amharic Text classification have been done by many researchers. Different methodologies and approaches have been utilized and experimented. Relevant works of those researchers are presented here.

(Gambäck et al., 2014) applied machine learning to Amharic text classification and examined the effect of operations like stemming and POS tagging on text classification performance for Amharic. They utilized a medium-sized, hand-tagged Amharic corpus and found out that stemming has no significant influence on the performance of Amharic text classification. In their work, bag-of-words approach was used for text classification experiment. The approach they utilized suffers a big sparsity which is resulted from the very nature of the way bag-of-words do word representations. The model faces the challenge of harnessing very little information as the vector-space is too huge. Using this approach also has another effect: the probability of finding vectors for out-of-vocabulary words and the absence of context.

Kelemework (2009) carried out a neural network approach on 9 categories of with a total size of 1,762 items in the dataset using LVQ (Learning Vector Quantization) for an automatic Amharic news classification. The issue with LVQ is that processing required for classification takes time as more hidden units are often required.

A paper by Habte (as cited in Gambäck et al., 2014) used ANN approach, called SOM (Self-Organizing Maps) on a corpus of 100 news items for document classification. Eyassu & Gambäck (2005) on the other hand utilized a set of queries for classifying news items taking the queries as class labels. They experimented on a 206 document corpus after converted to a term-document matrix and reduced using dimensionality reduction method called SVD (Singular Value Decomposition).

Zelalem (as cited in Kelemework, 2009) used statistical method and Cosine Similarity function as a matching technique for classification on a total of 1,481 news items, while (Weldesellassie, 2003) used 11,024 news articles and employed Naive Bayes and KNN and found out that classification accuracy using Naive Bayes and KNN decreases if fewer documents are used in training. Weldeselassie further noticed that classifiers work well if news items are evenly distributed in the categories.

Hierarchical classification on Amharic news was carried out by (Tegegnie, 2010) and experimented on a total of 16,075 news items. Tegegnie found out that the increase in the number of classes and documents or features has a reverse effect on the accuracy of flat classification and at the peak of features, the flat classifier accuracy also diminishes.

2.5 Word Embeddings Evaluation Methods

In the NLP community there are two evaluation methods that are often used: Intrinsic and Extrinsic evaluation methods. Intrinsic evaluations are experiments in which word embeddings are evaluated based on human judgements or their visual results on words relations. word semantic similarity and word analogy, are the most popular method of word embeddings evaluation. Extrinsic evaluation methods, on the other hand, are based on the ability of a certain word embeddings to be used as the feature vectors of supervised machine learning algorithms used in other downstream NLP tasks (Bakarov, 2018). The performance or efficiency of the supervised model measured on a dataset for a given NLP task functions as a measure of the quality of WEs. WEs can be used in natural language processing tasks such as Text Classification for extrinsic evaluation.

CHAPTER 3
METHODOLOGY AND APPROACH

In this chapter, the methodology, approaches, algorithms and tools that are utilized in this thesis are discussed. Since word embeddings require a dataset for training, the process of dataset preparation is presented in detail. Selected Techniques to evaluate the word embeddings in Amharic are presented. The general architecture of the work and the ways to visualize the expected results are among the issues raised.

3.1 Words and Word Vectors

Words are fundamental units of language formed from letters. They represent sounds using a sequence of characters that can be easily understood by humans. In linguistic syntax hierarchy, words are the atomic units of syntax which cannot be subdivided into smaller units (Basirat, 2018). For example, the Amharic sentence *የአማራ ክብሩ ክላይ ነው።።* is composed of the words *የአማራ, ክብሩ, ክላይ, ነው,* and the final punctuation ።. (Look Appendix XX for Amharic Punctuation marks). In most NLP tasks and computer systems, punctuation marks are taken as a word form.

During forming a sentence words are separated, in most languages including Amharic, by a space character. The boundary between words in a sentence are marked by the space character (exceptions are some languages such as Chinese with no word boundary).

3.1.1 Words and contexts

Context is a connection between elements of a paragraph or a dataset or a corpus. The surrounding words woven together to attain a certain meaning forms context. For a given corpus E as a set of elements (e), context is defined as a function C: E →P(E), where P(E) is the power set of E. This function is called a context function (Basirat, 2018). However, the context type that is customarily used in word embedding is the neighborhood context of words. This is referred to as a window-based context of a word which is formed by all words in a sequence (or window) of surrounding words. Let $E = \{1, \dots, e_T\}$ be a corpus of size T; then the neighborhood context of word say $e_t \in E$ with parameter $\tau \in Z$ is :

$$\delta_n(e_t;\ \tau) = \begin{cases} e1 & t+\tau < 1 \\ e_{t+\tau} & 1 \le t+\tau \le T \\ e_T & t+\ \tau\ > T \end{cases} \tag{3.1}$$

Depending on the sign of parameter τ, the neighborhood context returns the word at the t[th] position before or after e_t. The neighborhood context with $\tau < 0$ is called history or backward neighborhood context and the neighborhood context with $\tau > 0$ is called future or forward neighborhood context.

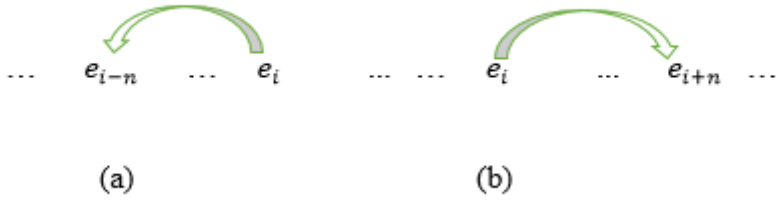

Figure 3.1: (a): a left neighborhood context with parameter τ=-n; (b): a right neighborhood context with parameter τ=+n, where n > 0.

3.1.2 Vectors and word vectors

Mathematically speaking, a vector is a quantity that determines the position of one point in space relative to another. They are characterized by magnitude and direction. As vectors are physical quantities, they can be compared with each other in different ways. The two famous methods are Euclidean distance and cosine of the angle. While Euclidean distance is the actual distance between vectors in N-dimensional space, the cosine distance is the angle between vectors in space. The Euclidean distance between two vectors $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_n)$ is computed as:

$$||a - b|| = \sqrt{\sum_{i=1}^{n} (ai - bi)^2} \tag{3.2}$$

The cosine of the angle between the above vectors is computed using dot and cross product of the two vectors as shown in the formula below.

$$\cos\theta = \frac{\sum_{i=1}^{n} a_i b_i}{||a||\,||b||} \tag{3.3}$$

Vectors can be formed from different objects such as images, videos, numbers and words. Before the advent of the introduction of vectors into NLP, the notions of phonemes, morphemes, syntax and semantics etc. were used to grasp the structure of words. Soon after the introduction of machine learning and deep learning, the need to represent words in a way that can enable machines understand words come into existence. The representation of words should retain meanings, semantic relationships and other linguistic features of words including context among words. The ideal proposed way to represent words is then using vectors. Word vectors represent words as multi-dimensional continuous floating point numbers where semantically related words are mapped to adjoining points in spacial space. This representation of words using vectors lends the capability to calculate distance, be it Euclidean or Cosine, as a measure of the relationship between words.

3.2 Word Representation

The relationships of words can be addressed either in morphological analyzer or in vector form. The morphological analyzer determines the morphological (structure and forms of words) relationships between words. In the vector form, words are mapped to a real-valued numbers called vectors. A word vector is a real value number where words are captured and represented in a way semantically related words will come closer in space and have similar vectors.

In order to represent words with vectors, there are many popular models and algorithms along with the approaches. Among them Word2Vec, GloVe and fastText have gained tremendous popularity and effectiveness in the NLP arena.

3.2.1 Word2Vec

Word2Vec is a neural network with single input, output and hidden layer. Mikolov et al., (2013c) proposed this shallow word embedding model. The model uses the neural models discussed in Section 2.3. It predicts words based on their context using CBOW and SG models. Hyper parameters such as embedding size and windows width are fine-tuned during creating word vectors using word2vec.The window width is just how many words should be

as history and future to the target word, while the embedding size specifies the number of neurons in the hidden lawyer. The embedding size is the dimensionality.

Word2vec, proposed by Mikolov et al. at Google, is a shallow, two-layered NN that is used in processing texts and produce word embeddings. It has a single hidden layer and a fully connected feedforward network. In word2vec, non-linearity of neural networks is removed.

Word2vec takes in large corpus and takes out a word embedding or a vector space. As it's used in representing words in distributional word embeddings, the representation keeps the semantic relationship among words. Words are represented in the form of vectors such that words with similar meaning appear closer. Word2vec is grouped under the predictive model (see Section 2.2.1).

According to (Mikolov et al., 2013a), this model for learning distributed representations of words lessen computational complexity caused by the non-linearity of hidden layers in traditional language models like Recurrent Neural Net Language Model (RNNLM). The authors preferred data efficiency, simplicity, and accuracy of representation of words that can help to guess about words linguistic relationship based on past appearances. It generates word embeddings by sliding a window over a large corpus of text.

Word2vec appears in 2 flavors, the CBOW and the SG model (highlight made on Section 2.3). These two model architectures pretty much are similar algorithmically except that their predicting method is opposite.

a) Continuous Bag-of-words Model

This model architecture learns an embedding by predicting the focus word based on nearby words, called context. The surrounding words determine the context. The context is represented by multiple words for a given probe word based on the size of the window. This model actually uses those surrounding words and try to predict the target words. Given words $w_1, w_2, \dots, w_i$, CBOW model learns to predict all words w_k from their nearby words (w_{k-1}, ..., w_{k-1}, w_{k+1},..., w_{k+1}).

For example, take the sentence: Amhara have engineered Ethiopia from South to North through its love and fist. As word2vec uses the history and future of the target word (that is words before and after the word based on their position) which is named as window, let the

window be 3 words. To predict, say the target word is *Ethiopia*, the surrounding words are then {Amhara, have, engineered, from}. What CBOW basically does is predicting the target word after the surrounding words. The idea, is given a context, to predict which words is most likely to appear along with the target.

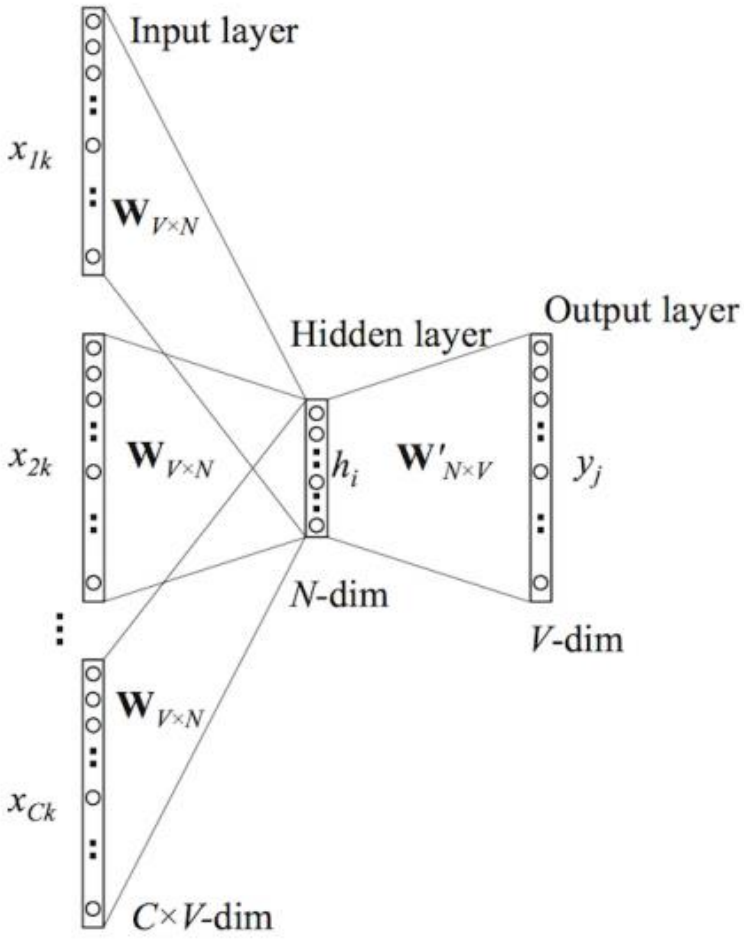

Figure 3.2: CBOW model diagram

The model diagram above (Figure 3.2) clearly depicts the CBOW architecture. Let C be size of window; V be size of vocabulary. The input is 1-hot encoded context words $\{X1, X2, \dots, Xc\}$ in an N-dimensional vector h, the output would be 1-hot encoded word y. The input vectors are fully connected to the hidden layer via a $V \times N$ weight matrix W and the hidden layer is connected to the output layer via a $N \times V$ weight matrix W'.

This model is dubbed continuous because it modifies the property of bag-of-words that the order of words does not matter. Now in CBOW, it matters. It matters because the order of

words is highly related with the context. And the context of words or sentences is the substance of every communication in NLP. On the other side, by continuous, it means the model is using continuous distributed representation of the context (Mikolov et al., 2013a).

b) Continuous Skip-gram Model

This architecture is similar to the first CBOW (Mikolov et al., 2013a) but in the reversed way. It predicts the nearby words given the target word. Given words w_1, w_2, ..., w_i, Skip-gram model learns to predict nearby words of the current target word w_k (Mikolov et al., 2013b). According to the authors, the model uses other adjacent words that are with the target word in the same sentence to maximize classification of words. Each current word being used as an input to the classifier with continuous projection layer, hence the name continuous skip-gram, then words are predicted within a range of window size.

In this architecture, the objective is to find word representations that are useful for predicting context given a target word. Context refers to words nearby the target word or the surrounding words.

Consider the sentence: *I want to teach about word vector representation*, and let the window be of size 1. The skip-gram model will break the sentence into (context, target) pairs, like:

([I, to], want), ([want, teach], to), ([to, about], teach), ([teach, word], about), ([about, vector], word), ([word, representation], vector) ....

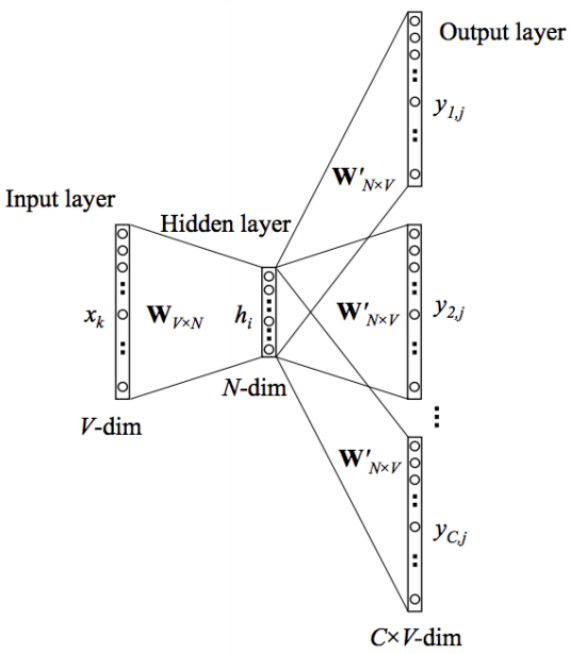

Figure 3.3: Skip-gram model

As shown in the above model (Figure 3.3), the input of the model is a word w_1 and the output is the adjacent words in w_1's context {wO,1,..., wO,C} where *c* is the window size. Taking the previous example once again, the potential training instance could be the word "vector" as an input and the words {"I", "want", "to", "teach", "representation"} as outputs. All these words are vectors of length *V*, which is the size of the vocabulary.

3.2.2 GloVe

Global Vectors is another popular method for learning word representations proposed by (Pennington et al., 2014). Global Vectors, GloVe for short, is an approach to perceive and seize the meaning of a word with the structure of the whole corpus. The model uses co-occurrence counts of words and do a statistic to minimize errors. GloVe constructs co-occurrence matrix that is used to calculate the probability the appearance of a word in the context of another word. That probability can be articulated as P(i/j), which shows relationships between words. GloVe then uses the ratio of co-occurrence probabilities between words to predict target word by training the word-word co-occurrence matrix.

GloVe takes advantages from two major models: global matrix factorization and local context windows methods (Pennington et al., 2014). Global matrix factorization is the

process of decomposing a matrix to the product of several matrices. This factorization method is a generalized coordinate matrix factor table. These matrices represent either the frequencies of a term in a document, or the co-occurrence of terms. The other major model the authors analyzed is local context environment methods. In this method the model learns by observing the contexts around target words or the vice versa. They scan context windows in the corpus.

Pennington et al., further noted that those major models suffer from disadvantages. While global matrix factorization methods efficiently leverage statistical information, they do not perform better on word analogy task though. On the other hand, context window methods act poorly on statistics of corpus and do better on analogy task.

So, the GloVe model takes the strengths of the two model families. From the global matrix factorization models, instead of learning raw co-occurrence probabilities or frequencies, the ratios of the co-occurrence probabilities are taken. GloVe gives numerical vectors for a huge corpus of words.

To illustrate how GloVe works the co-occurrence probability, suppose we want to study the relationship between two words, *i* and *j*.

Co-occurrence probability

Let word-word co-occurrence counts matrix be X, and Xij be the number of times word *j* appears in the context of word *i*.

Let k be a probe word.

$$Pij = P(j|i) = {}^{Xij}/_{Xi} \qquad (3.4)$$

P*ij* = probability that word j appears in the context of word *i*.

Xi= number of times any word occurs in the context of word *i*.

For words k related to *i* but not to *j*: ${}^{Pik}/_{Pjk}$ will be large.

For words k that are either related to both *i* and *j*, or to neither, the ratio ${}^{Pik}/_{Pjk} \approx 1$.

Therefore, co-occurrence probability is better in distinguishing relevant words from irrelevant ones.

In GloVe model, the idea is, given a corpus and words, to take data from that corpus in the form of global statistics and learn a function that gives information about the relationship between words. Now the authors have discovered that the ratios of co-occurrence probabilities render a good result in distinguishing relevant words, so it would be nice if context of the words is taken into account. So let the function the model is learning be F. A naive interpretation of the desired model is given by the authors as,

$$F(wi, wj, \varpi k) = \frac{Pik}{Pjk}, \tag{3.5}$$

where $w \in R^d$ are word vectors and $\varpi \in R^d$ are separate context word vectors. Note that the w's are real-valued word vectors.

The above two models rely on the substance of words, be it their occurrence or frequency or by their context. The essence of the word itself, its linguistic properties such as morphology of the words were not taken into account. The other recent technique that takes those issues into consideration to represent words to vectors is fastText (Joulin et al., 2016).

3.2.3 fastText

fastText combines the concepts of CBOW architectures and represents sentences using bag-of-words and bag-of-n-grams, as well as using subword information, and sharing information across classes via a hidden layer. fastText is not just for word representation, which is to be discussed soon, but also for text classification.

fastText is used as a classification model that can produce fast, efficient and accurate results which can put it comparable to other known deep NN classifiers and embedding models. The training and classification by fastText is very fast (Joulin et al., 2016). As a linear based and scalable model, fastText uses a hierarchical Softmax function to lessen computational complexity. The lesser the computational complexity, the faster the search to predict classes for text classification and producing word representation. fastText uses both bag of words and bag of n-grams, the latter for word ordering. To produce efficient outputs in using bag of n-grams, a hashing technique to map n-grams is utilized.

fastText for Word Representation

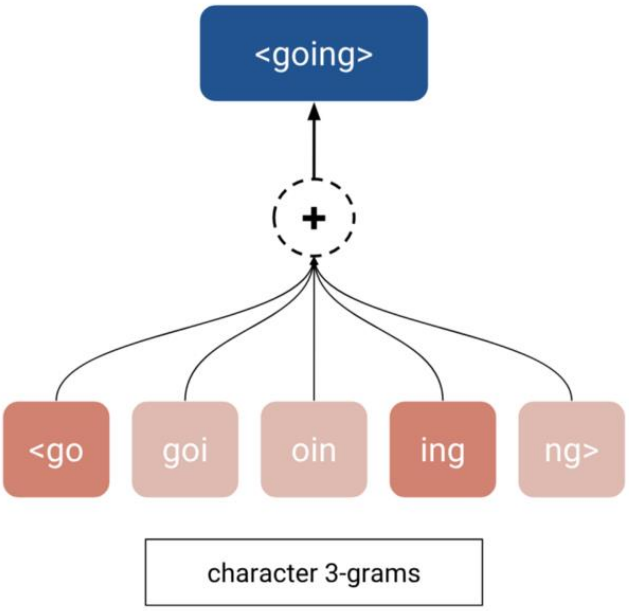

Figure 3.4: Character n-grams example using the word "going"

fastText, being an extension of the continuous SG model (Bojanowski et al, 2017; Mikolov et al., 2013) is a robust embeddings and text classifier using subword information. GloVe and Word2vec mainly focus on words for learning embeddings. As a result, words that are not incorporated with the vocabulary or new words that are created or formed anew are usually represented by a vector of zeros or simply ignored. According to the authors popular models that learn representation of words do not take into account the morphology of words and the parameter sharing. For morphologically rich languages such as Amharic and Turkish, ignoring the internal structure of words by just assigning a distinct vector to each word will have a limitation on the representation.

Therefore, fastText goes one level down from word to character n-gram level information. fastText takes each word as a sum of n-gram characters and words are represented in word embedding as the sum of those n-gram characters' vectors. In fastText, each word is broken down to an n-gram constituents in addition to the word itself. For example, take the n-gram

be 3 characters and take the following words: going, matter, apple, where and መጨቃጨቅ (Amharic: squabble).

Taking those words and n=3 as an example, the words in fastText embedding word representation will be represented by the character n-grams:

- going: <go, goi, oin,ing, ng> and the special sequence <going>
- matter: <ma, mat, att, tte, ter, er> and the special sequence <matter>
- apple: <ap, app, ppl, ple,le> and the special sequence <apple>
- መጨቃጨቅ: <መጨ, መጨቃ, ጨቃጨ,,ቃጨቅ, ጨቅ> and the special sequence <መጨቃጨቅ>

Each word *w* is represented as a bag of subword n-grams and the word *w* itself is also included in the set of n-grams (Bojanowski et al., 2017).For a given word, n-gram is taken from 3 to 6 grams and the word is represented by the sum of the vector representations of its n-grams. The n-grams are called subwords.

This fastText, because it considers subword information, fills the gap created by other models like word2vec and GloVe. It lets the reliable representation of new words, out-of-vocabulary words and rare words in the corpus.

The overall description about fastText and related concepts are discussed on Section 2.3. Here the text classification side of fastText is presented.

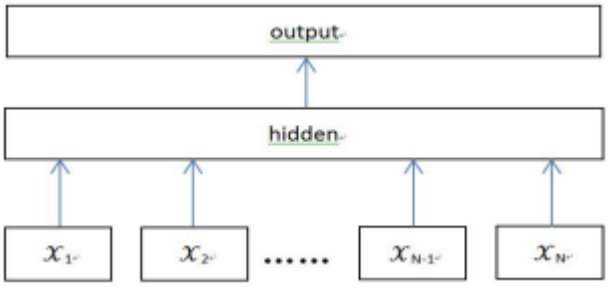

Figure 3.5: fastText model

fastText is a library for learning of word representation and text classification (Mikolov et al., 2016). As shown in the Figure 3.5 above, the model for fastText has three parts: the inputs (sequence of words, a piece of text or a sentence), the hidden layer(Softmax function

to compute the probability distribution) and the output (the probability that the word sequence belongs to a certain category). The probability distribution over prelabelled classes can be computed using a Softmax function:

$$Pd(w) = \sqrt{\frac{t}{f(w)}} + \frac{t}{f(w)} \; ; \tag{3.6}$$

Where,

$$f(w) = \frac{Count_w}{total\ no\ of\ tokens} \tag{3.7}$$

Each vocabulary in fastText is mapped to a real-valued vector, with new and out-of-vocabulary (OOV) words getting a special unique vocabulary ID. The structure of a more elaborated fastText classifier is shown in Figure 3.6. Text (or document) words w_i are represented with n-dimensional word vectors X_i. The vector for a text (document) y is computed as average of the vector of a linear bag of words of the document, as:

$$y = \frac{1}{N}\sum_{i=1}^{N} Xi \tag{3.8}$$

Here *N* is the number of words document Xi is a word occurrence in the document.

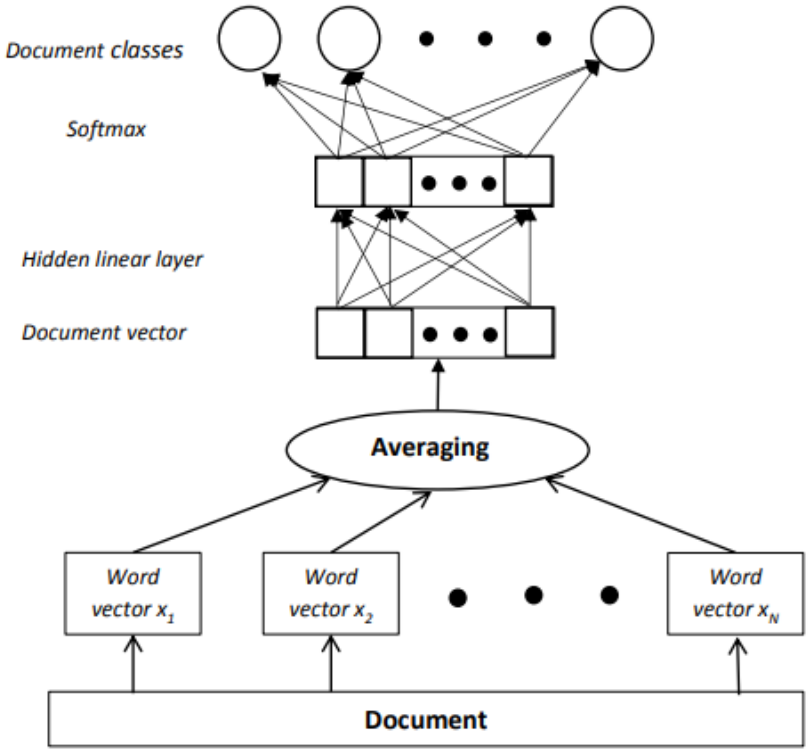

Figure 3.6: A more elaborated fastText classifier with hidden-layer

The vector y is input to the hidden layer, where it is multiplied by the matrix M of the hidden linear layer to get a classification vector Z:

$$Z = \begin{pmatrix} z_1 \\ \vdots \\ z_m \end{pmatrix} \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ \vdots & & & \vdots \\ m_{m,1} & m_{m,2} & \cdots & m_{m,n} \end{bmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = M.y \tag{3.9}$$

where Z is an m-directional vector, M is an $m \times n$ matrix, m is the number of labels and y is an n-dimensional vector. To do the classification the following Softmax function is used to compute class/label probabilities:

$$P_j = \frac{e^{z_j}}{\sum_{k=1}^{m} e^{z_k}} \tag{3.10}$$

where P_j is the predicted probability that the text or sentence belongs to the j[th] label; z_j and $z_k$are the components of the classification vector Z. fastText classifier calculates the sentence vector as the average of the word vectors (normalized by their length). The

described classification model computes the document vector y by averaging vectors of all word occurrences.

3.3 Word Representation for Amharic

In this section word representation of Amharic dataset, ways of evaluation of the word embeddings formed, the process of preparing the training corpus and the overall phases are discussed thoroughly. For training word representation, an Amharic corpus containing about 1.5 million words is prepared.

3.3.1 The corpus for word embedding

In NLP tasks, the role of Corpora is very immense. The success or failure of most NLP applications depends on the quality of appropriate data. The data used in NLP tasks usually take the form of corpora. Corpora can be annotated or unannotated. Raw data that is simple plain where the linguistic information is implicit is unannotated corpora. Annotated corpora, on the other hand, adds extra explicit information to the text such as categories, part-of-speech tags and so on.

The corpus utilized here is unannotated one. Part of it is taken from a multilingual parallel corpus created from translations of the Bible by (Christodouloupoulos & Steedman, 2015).The corpus is aligned (almost) at a sentence level and the document was formatted as an XML file, containing nested and <seg> elements. Each sentences were marked with an ID. The other content of the corpus was collected from different Amharic books. Figure 3.7 shows the sample corpus before pre-processing taken from the work of (Christodouloupoulos & Steedman, 2015).

```
▼<div id="b.GEN.1" type="chapter">
  <seg id="b.GEN.1.1" type="verse">በመጀመሪያ እግዚአብሔር ሰማይንና ምድርን ፈጠረ።</seg>
 ▼<seg id="b.GEN.1.2" type="verse">
    ምድርም ባዶ ነበረች፥ አንዳችም አልነበረባትም፤ ጨለማም በጥልቁ ላይ ነበረ፤ የእግዚአብሔርም መንፈስ በውኃ ላይ ሰፍሮ ነበር።
  </seg>
  <seg id="b.GEN.1.3" type="verse">እግዚአብሔርም። ብርሃን ይሁን አለ፤ ብርሃንም ሆነ።</seg>
 ▼<seg id="b.GEN.1.4" type="verse">
    እግዚአብሔርም ብርሃኑ መልካም አንደ ሆነ አየ፤ እግዚአብሔርም ብርሃንንና ጨለማን ለየ።
  </seg>
 ▼<seg id="b.GEN.1.5" type="verse">
    እግዚአብሔርም ብርሃኑን ቀን ብሎ ጠራው፥ ጨለማውንም ሌሊት አለው። ማታም ሆነ ጥዋትም ሆነ፥ አንድ ቀን።
  </seg>
 ▼<seg id="b.GEN.1.6" type="verse">
    እግዚአብሔርም። በውኖች መካከል ጠፈር ይሁን፥ በውኃና በውኃ መካከልም ይክፈል አለ።
  </seg>
 ▼<seg id="b.GEN.1.7" type="verse">
    እግዚአብሔርም ጠፈርን አደረገ፥ ከጠፈር በታችና ከጠፈር በላይ ያሉትንም ውኖች ለየ፤ እንዲሁም ሆነ።
  </seg>
  <seg id="b.GEN.1.8" type="verse">እግዚአብሔር ጠፈርን ሰማይ ብሎ ጠራው። ማታም ሆነ ጥዋትም ሆነ፥ ሁለተኛ ቀን።</seg>
 ▼<seg id="b.GEN.1.9" type="verse">
    እግዚአብሔርም። ከሰማይ በታች ያለው ውኃ በአንድ ስፍራ ይሰብሰብ፥ የብሱም ይገለጥ አለ እንዲሁም ሆነ።
  </seg>
 ▼<seg id="b.GEN.1.10" type="verse">
    እግዚአብሔርም የብሱን ምድር ብሎ ጠራው፤ የውኃ መከማቻውንም ባሕር አለው፤ እግዚአብሔርም ያ መልካም እንደ ሆነ አየ።
  </seg>
 ▼<seg id="b.GEN.1.11" type="verse">
    እግዚአብሔርም። ምድር ዘርን የሚሰጥ ''ሣርንና ቡቃያን በምድርም ላይ እንደ ወገኑ ዘሩ ያለበትን ፍሬን የሚያፈራ ዛፍን ታብቅል አለ፤ እንዲሁም ሆነ።
  </seg>
 ▼<seg id="b.GEN.1.12" type="verse">
    ምድርም ዘርን የሚሰጥ ''ሣርንና ቡቃያን እንደ ወገኑ ዘሩም ያለበትን ፍሬን የሚያፈራ ዛፍን እንደ ወገኑ እበቀለች። እግዚአብሔርም ያ መልካም እንደ ሆነ አየ።
  </seg>
  <seg id="b.GEN.1.13" type="verse">ማታም ሆነ ጥዋትም ሆነ፥ ሦስተኛ ቀን።</seg>
 ▼<seg id="b.GEN.1.14" type="verse">
    እግዚአብሔርም አለ። ቀንና ሌሊትን ይለዩ ዘንድ ብርሃናት በሰማይ ጠፈር ይሁኑ፤ ለምልክቶች ለዘመናት ለዕለታት ለዓመታትም ይሁኑ፤
  </seg>
  <seg id="b.GEN.1.15" type="verse">በምድር ላይ ያበሩ ዘንድ በሰማይ ጠፈር ብርሃናት ይሁኑ፤ እንዲሁም ሆነ።</seg>
 ▼<seg id="b.GEN.1.16" type="verse">
    እግዚአብሔርም ሁለት ታላላቆች ብርሃናትን አደረገ፤ ትልቁ ብርሃን በቀን እንዲሠለጥን፥ ትንሹም ብርሃን በሌሊት እንዲሰለጥን፤ ከዋክብትንም ደግሞ አደረገ።
  </seg>
```

Figure 3.7: Sample corpora before preprocessing

3.3.2 Pre-processing the corpus for word embedding

Pre-processing involves preparing the corpus or dataset into a format that is suitable for training and evaluation process. As shown in Figure 3.7, the Corpus is formatted in an XML format and it should be processed, cleaned and made ready for further work. fastText can take the corpus intact but the garbage characters that are part of the XML syntax and delimiters such as < and > might affect the result of the embedding. They are useless for the Amharic text in the first place.

The other characters that should be removed are the punctuation marks. Amharic has its own punctuation marks such as ፣ (ነጠላ ሰረዝ - Amharic comma), ፤ (አራት ነጥብ - Amharic semi-colon), ። (አራት ነጥብ - Amharic Full stop) (see Appendix III for full list of Amharic Punctuation Marks) and white space. White space between words is important but a space between paragraphs is of no value for the dataset. Numbers that were part of the corpus also

do not play a role in the training. These punctuation marks and numbers are also not important to our training dataset. Therefore, a prior pre-processing is required.

To preprocess and get a cleaned dataset from the corpus of (Christodouloupoulos & Steedman, 2015), a small script is crafted in python to remove the unwanted characters, punctuation marks including spaces between paragraphs and numbers in the corpus. The script supported by a library called EelementTree which is a library package to parse, explore, modify and populate XML files with python.

Preprocessing algorithm pseudo code

```
For each text in the corpus
Check for the presence of XML chars OR a numeral OR a punctuation mark
    If opening AND closing XML chars present
        Remove data between marks
    If numerals OR punctuation marks present
        Remove
        While size of removed chars <> 0
            Move words to the left
    If while space greater than one tab
        Move words to the left or remove
End
```

Figure 3.8: Preprocessing algorithm pseudo code

After preprocessing the dataset looks like the following script shown in Figure 3.9.

በመጀመሪያ እግዚአብሔር ሰማይንና ምድርን ፈጠረ ምድርም ባዶ ነበረች አንዳችም አልነበረባትም
ጨለማም በጥልቁ ላይ ነበረ የእግዚአብሔርም መንፈስ በውኃ ላይ ሰፍፎ ነበር እግዚአብሔርም ብርሃን
ይሁን ኣለ ብርሃንም ሆነ እግዚአብሔርም ብርሃኑ መልካም እንደ ሆነ አየ እግዚብሔርም ብርሃንንና
ጨለማን ለየ እግዚአብሔርም ብርሃኑን ቀን ብሎ ጠራው ጨለማውንም ሌሊት አለው ማታም ሆነ
ጥዋትም ሆነ አንድ ቀን እግዚአብሔርም በውኖች መካከል ጠፈር ይሁን በውኃና በውኃ መካከልም
ይክፈል አለ እግዚአብሔርም ጠፈርን አደረገ ከጠፈር በታችና ከጠፈር በላይ ያሉትንም ውኖች ለየ
እንዲሁም ሆነ እግዚአብሔር ጠፈርን ሰማይ ብሎ ጠራው ማታም ሆነ ጥዋትም ሆነ ሁለተኛ ቀን
እግዚአብሔርም ከሰማይ በታች ያለው ውኃ በአንድ ስፍራ ይሰበሰብ የብሱም ይገለጥ አለ እንዲሁም ሆነ
እግዚአብሔርም የብሱን ምድር ብሎ ጠራው የውኃ መከማቻውንም ባሕር አለው እግዚአብሔርም ያ
መልካም እንደ ሆነ አየ እግዚአብሔርም ምድር ዘርን የሚሰጥ ሣርንና ቡቃያን በምድርም ላይ እንደ ወገኑ
ዘሩ ያለበትን ፍሬን የሚያፈራ ዛፍን ታብቅል አለ እንዲሁም ሆነ ምድርም ዘርን የሚሰጥ ሣርንና ቡቃያን
እንደ ወገኑ ዘሩም ያለበትን ፍሬን የሚያፈራ ዛፍን እንደ ወገኑ አበቀለች እግዚአብሔርም ያ መልካም እንደ
ሆነ አየ ማታም ሆነ ጥዋትም ሆነ ሦስተኛ ቀን እግዚአብሔርም አለ ቀንና ሌሊትን ይለዩ ዘንድ ብርሃናት
በሰማይ ጠፈር ይሁኑ ለምልክቶች ለዘመኖች ለዕለታት ለዓመታትም ይሁኑ በምድር ላይ ያበሩ ዘንድ
በሰማይ ጠፈር ብርሃናት ይሁኑ እንዲሁም ሆነ እግዚአብሔርም ሁለት ታላላቆች ብርሃናትን አደረገ ትልቁ
ብርሃን በቀን እንዲሠለጥን ትንሹም ብርሃን በሌሊት እንዲሰለጥን ከዋክብትንም ደግሞ አደረገ

Figure 3.9: Preprocessed dataset sample

3.3.3 Amharic word embedding

After preprocessing the corpus, the next step is training. The library fastText is used to train the corpus. FastText provides two modes of computing word representations: CBOW and skipgram. Both ways of architectures were discussed in Section 2.3 and are employed here for training.

Both model architectures are used in fastText to lean a high-dimensional dense representation for each vocabulary term in the corpus. The representation is distributional and it tries to learn from surrounding context as well. In both model architectures, the network is a two-layer, shallow neural network.

In skipgram, as discussed in Sections 2.3 and 3.2.1, a context windows of K is considered and other parts are skipped. The relationship between the window or panel and the target word is explored. This is done by feeding the two-layer shallow neural network a 1-hot encoding of the target word. As the input is 1-hot encoded, the hidden layer consists of only

one row of input hidden weight matrix. Therefore, the task of the network is to predict the i[th] context given the target.

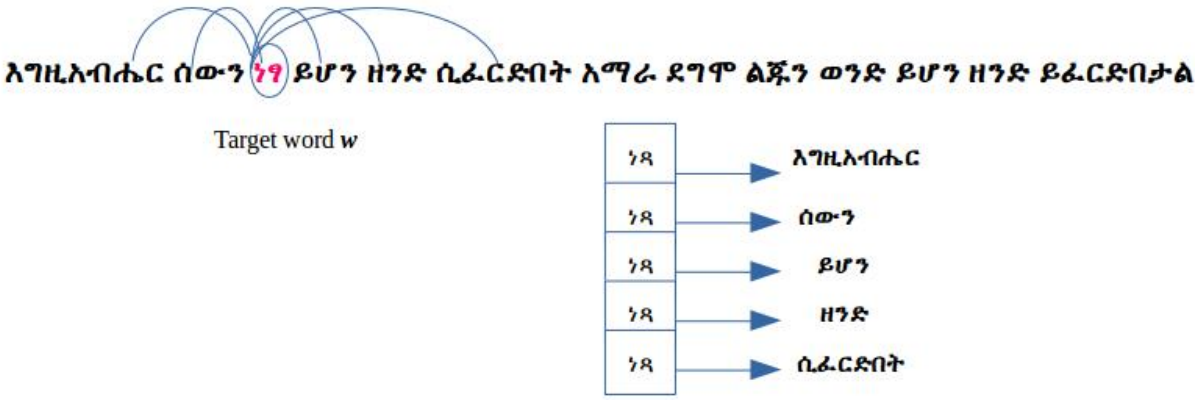

Figure 3.10: Model probability of a context word given a word w(colored red)

feature for word w: X_w

classifier for word c: v_c

The scores for each word are computed using the equation:

$$\upsilon = W'^{T} h \tag{3.11}$$

Here, h is a vector in the hidden layer and W is the hidden output weight matrix. After computing the score u, c multinomial weight distributions are computed, where c is the window size. In the Figure 3.10 above, the window size is 3 for example. The distributions are computed as:

$$P\left(w_{c,j} = wo_{,c} \left| wI\right.\right) = \frac{\exp \upsilon_{c,j}}{\sum_{j'=1}^{v} \exp \upsilon_{j'}} \tag{3.12}$$

where $w_{c,j}$is the j[th] word on the c[th] windows of the output layer, $wo_{,c}$is the actual c[th] word in the output context words, wI is the input word, and $\upsilon_{c,j}$ is the net input of the j[th] unit on the c[th] panel of the output layer.

The second model of architecture which have been discussed thoroughly in Section 3.2.1 is CBOW. CBOW is technically the opposite of skipgram, where the specific word is taken as the target given the context (Bhattacharjee, 2018). Therefore, in CBOW, given the previous sentence: "እግዚአብሔር ሰውን ነፃ ይሆን ዘንድ ሲፈርድበት አማራ ደግሞ ልጁን ወንድ ይሆን ዘንድ ይፈርድበታል!"

the word "ነፃ" can be generated given the context ["እግዚአብሔር", "ሰውን", "ይሆን", "ዘንድ", "ሲፈርድበት"]. CBOW takes the 1-hot vectors of all the words (context). The algorithm is pretty much the same as skipgram, but the hidden layer's output is generated using the following equation:

$$h = \frac{1}{C}\, W.(\sum_{i}^{c} \chi_i) \qquad (3.13)$$

The score in CBOW is generated with the same equation used in skipgram.

For training using fastText, there are different parameters to tune with. The parameters and their descriptions are detailed below in Table 3.1 as described in fastText library.

Parameters can be fine-tuned to adjust to more robust models. With different optimization and parameter tuning, accuracy of models can be adjusted. One of the features of fastText is its ability in capturing subword information. It takes into account not only the word itself, but also the constituent parts of the word or its sub-characters. The length of n-grams can be controlled using the -minn and -maxn flags for minimum and maximum number of characters during training. These parameters control the range of values to get n-grams for words.

fastText controls the size of the vocabulary using a -minCount parameter. It shows the minimum count for words that need to be part of the vocabulary. The windows size that words that are around a target word is taken is controlled by -ws.

The number of times fastText visits the dataset during training is controlled by -epoch parameter. By default, fastText takes a look at each data point 5 times. Another parameter that is used to control how fast the model updates during training is -lr. This parameter controls the size of the update that is applied to the parameters of the models.

Dim represented the dimension of the hidden layer in the training, and thus the dimension of the embeddings, and is set via the -dim flag. This is set to 100 by default.

Table 3.1: Parameters used for training fastText word embeddings

Parameter	Description	Default value
minCount	minimal number of word occurrences	5
wordNgrams	max length of word ngram	1
Minn	min length of char ngram (min char ngrams)	3
Maxn	max length of char ngram (maxx char ngrams)	6
Lr	learning rate	0.05
Dim	size of word vectors (dimension)	100
Ws	size of the context window (context window)	5
Epoch	number of epochs	5
Neg	number of negatives sampled	5
Loss	loss function {ns, hs, softmax}	ns

Loss function is the other key parameter that can help to compare the difference between the cost of the present model and the actual data distribution. In machine learning and related fields, error is computed by subtracting the predicted output from the actual one using a function called Loss Function. Choosing a loss function and an optimizing algorithm along with it is one of the key methods of machine learning (Bhattacharjee, 2018). The idea is that for specific loss function, optimizing algorithm pair, it would be possible to optimize the parameters of the model to make them mimic the real data as closely as possible. This function has three options that are currently supported by fastText: negative sampling (ns), softmax or hierarchical softmax(hs).

In general, a lot of hyperparameters can be used to optimized and find the right balance of models in fastText.

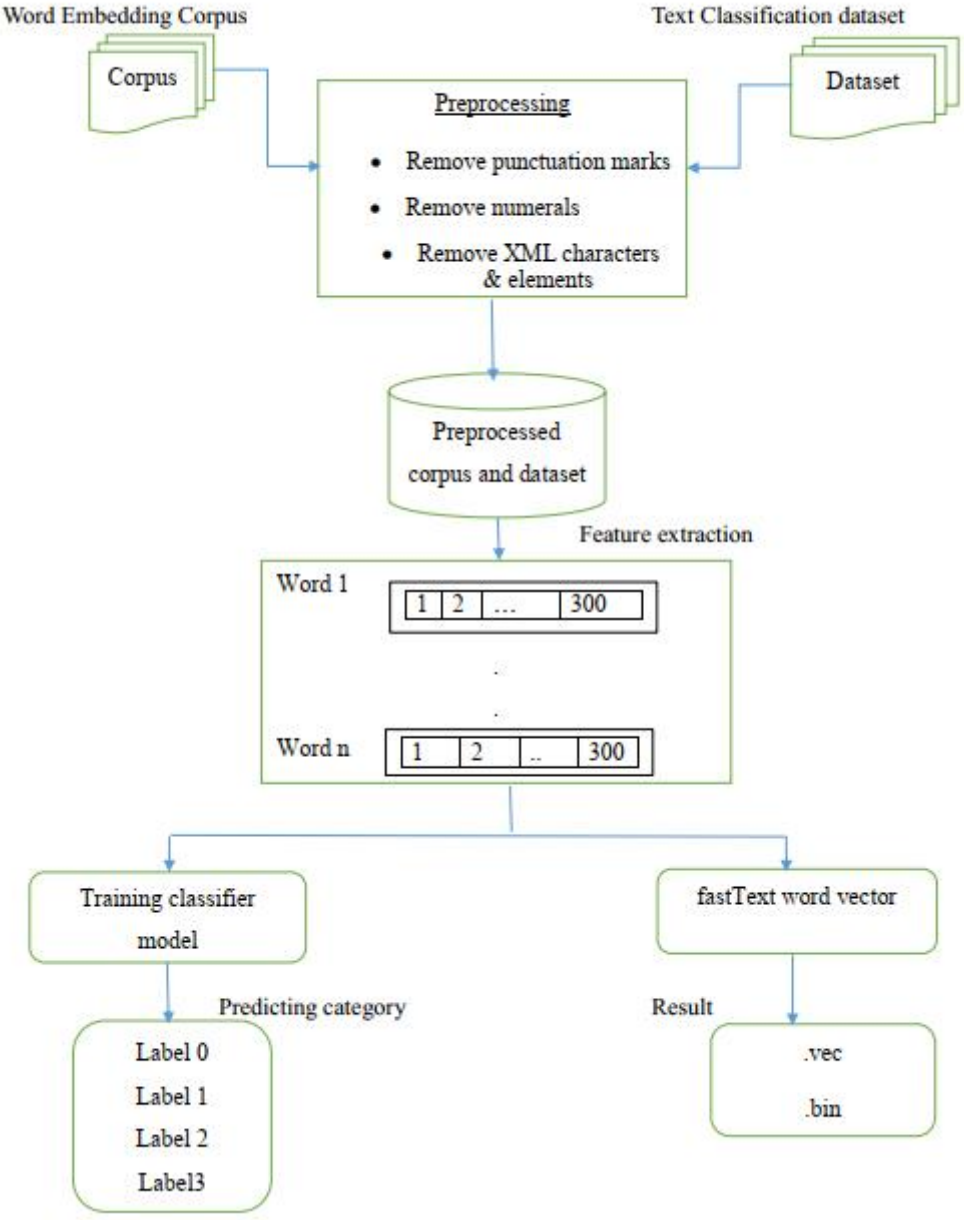

Figure 3.11: Proposed architecture and approach

3.3.4 Dataset for text classification

A manually prepared and labelled datasets is used to build the fastText classification model. A total of 900 news dataset is collected from the web is collected from Amhara Mass Media

Agency(AMMA[2]). The dataset contains a total of approximately 210.000 words and has been labelled manually after preprocessing. During data preprocessing, stop words, numerals, punctuations marks and symbols were eliminated and clean before the training and testing processes. Each line has a list of labels, followed by the corresponding data/document. All the labels start by the __label__ prefix, which is how fastText recognize what is a label or what is a text. The data format for FastText is as follows:

__label__<X> __label__<Y> ... <Text>

where X and Y represent the class labels. A sample from a training data file is given below.

```
__label__1 አትሌት ብርሃኔ አደሬ ጅኔቭ ላይ በተካሄደው የጎዳና ሩጫ አሸነፈች ...
__label__4 ስሎቭኒያ በተካሄደው የቼዝ ኦሎምፒያድ ላይ ከፍተኛ ውጤት ያመጡ ኢትዮጵያውያን...
__label__2 ጅጅጋና ጭናክሰን ከተሞችን የሚያገናኝ መንገድ እየተሰራ ነው መንገድ መንገድ ...
__label__3 በአዲስ አበባ ከተማ በሚገኙ ፀረ ኤድስ ማህበራት መካከል የሚካሄድ የእግር ኳስ ውድድር ተጀመረ ...
```

Figure 3.12: Sample Dataset with fastText labeling format

As shown in Figure 3.12, there are 4 classes, where __label__1 is news related to Athletics, __label__2 is about Chess, __label__3 is about Football and __label__4 is about others.

Each article has been manually classified as belonging to one of the four predefined classes. The four classes are presented in Table 3.2 below.

Table 3.2: The ten categories and the number of articles belonging to each category

Category	**Label**	**Count**
Athletics	1	154
Football	3	332
Chess	4	21
Others(agriculture, economics, etc…)	2	393

[2]Amhara Mass Media Agency is a government owned news and information service located in Bahir Dar, Ethiopia. At its web site www.amharaweb.com, it provides Ethiopia related news.

To get training data which are labeled four category -- Athletics, football, chess and others, the news site called AMMA is scraped. Each line of the text file contains a list of labels, followed by the corresponding content, as it is the default setup by fastText.

The preprocessing tasks in this dataset is more or less the same as described in Section 3.3.2. However, there are unique tasks that are taken place in preparing text classification datasets. The first one is labeling which is discussed above. As fastText expects a certain format for classification training, the dataset is prepared likewise. The other task is shuffling the data.

Shuffling the data before training the classifier is important. If the labels for the data are clustered, then the precision and recall, and hence the quality and performance of the model, will be poor and low. This is due to fastText's way of learning the model. fastText uses the stochastic gradient descent[3] based optimization. The training data from the training set is processed in order.

The other task which is obvious in text classification is dividing the dataset into training and testing sets. Model performance evaluation should always be done on independent data. Therefore, the whole dataset is separated out into training and testing sets.

To evaluate the performance of a classification model, the training dataset would be divided into test and train sets. Only the train set is used for model training. Once done, the test set is classified and comparison of the predictions with the actual ones and measuring the performance is performed. The portion of correctly classified sample to the portion of actual sample is called accuracy. Accuracy is the most natural performance measure and is the ratio of correctly predicted observation to the total observations. This measure is great but only when there are symmetric datasets where values of false positive and false negative are almost the same. Therefore, other performance measures are required to correctly quantify the performance.

One is the recall, which means the percentage of all the correct labels that are recalled as opposed to the labels that actually existed. This is the measure of what proportion of actual positives were identified correctly. The other measure is precision, which means answers the question: what proportion of all the predicted labels are the actual labels? It is the measure

[3]Gradient descent is basically an optimization algorithm that is meant for minimizing a function

of what proportion of positive identifications were actually correct. The weighted average of the above two measures: recall and precision is called F1 score. This score takes both false positives and false negatives into account.

$$F1\ score = 2 \times \frac{(Recall\ \times Precision)}{(Recall + Precision)} \tag{3.14}$$

3.4 Visualizing Word Embeddings

Visualizing the vectors and the embeddings in space is an effective way to understand and explore distributional properties of models. An embedding is a mapping from discrete objects, such as words, to vectors of real numbers. In its simplest way, word embeddings are matrices of XY coordinates. However, since the dimensions of the vectors are 300, which is quite high, a dimensionality reduction techniques are required so that the vectors can be visible in a 2-dimensional frame. Therefore, to visualize word embeddings there are two mainly used and popular algorithms: PCA (Principal Component Analysis) and t-SNE (t-distributed stochastic neighbor embedding).

The t-SNE and PCA are popular techniques for dimensionality reduction and are usually used for visualization of high-dimensional datasets. The idea in this case is to keep related words as close together as possible, while maximizing the distance between dissimilar words.

CHAPTER 4
EXPERIMENTATION AND RESULTS

4.1 Introduction

Natural Language Processing tasks are seriously taken in the research communities because of their role in solving real problems in our lives and the results they provide accordingly. Be it POS tagging, Sentiment Analysis, or text classification or the focus of this work -Word Embeddings, there are vital roles to play in addressing issues depending on the way we face them.

However, there is a little hardship here: the data-intensiveness of NLP tasks. NLP and other related fields like machine learning require big and rich data to produce the results we aspire. The availability of this data does not end the problem. The quality of the data is also another challenge in these fields.

When the data, in any useful form, quality and quantity, is available, as each NLP tasks are expected to throw an important output, experimenting with designed models and architectures are required to be carried out. By fine-tuning different parameters, levels and various features, different quality outputs and results are produced. Yet, the measure of accuracy of the results, the criteria of quality outputs in NLP are undergoing research areas.

In this chapter, word vector representation for Amharic language, the evaluation of the word embeddings using intrinsic and extrinsic evaluation methods are discussed. While linguistic relationships: word similarities, nearest neighbors, and word analogies are chosen for intrinsic evaluation, multi-class text classification on Amharic dataset is tested as an extrinsic evaluation.

4.2 Evaluation and Experimentation Setup

We used an Amharic corpus gathered from Wiki sources, local media sources and books as discussed in section 3.3. FastText library along with related models like word2vec and Gensim are employed both for experimentation and value visualization. While most of the parameters of fastText is used, some parameters are fine-tuned and used as follows. The embedding produced has a size of 100 or 300 (300 is default).

The experimental parameters are summarized in the previous section 3.3 in Table 3.1.

The machine used for training and experimentation has the following specifications: -

- Central Processing Unit(CPU): Intel(R) Core(TM) i3-2348M CPU @ 2.30GHz
- Random Access Memory(RAM): 4GB
- Operating System(OS): Ubuntu 16.04

For evaluation of our trained embeddings, we used two evaluation metrics. These evaluation metrics are intrinsic and extrinsic and are presented in the next section.

4.3 Evaluation Metrics

In word embeddings, there are two evaluation methods that are commonly employed: Intrinsic and Extrinsic evaluation methods. Intrinsic evaluations are experiments in which word embeddings are evaluated based on human judgments or their visual results on words relations. Word semantic similarity, nearest neighbors, and word analogy relations are the most popular method of word embeddings evaluation. Extrinsic evaluation methods measure on the ability of a word embedding to be used as the feature vectors of supervised machine learning algorithms used in other downstream NLP tasks such as Text Classification.

We explored both evaluation methods using the fastText library and the corpus prepared.

4.3.1 Intrinsic Evaluation

In neural word embeddings, as it is part of NLP tasks and it is highly related with language studies, taking the embeddings as tools to understand features of a certain language is common. It assesses how well the vectors capture the linguistic relationships (similarities, analogies) between words. This task is used to measure the quality of a word vector directly using different features. Among these features, the following three linguistic features are used to evaluate the word embeddings produced.

4.3.1.1 Word similarities and relatedness

This task involves finding related words with the query word in meaning or syntax. We used SG model to find near matches between terms. The similarity between say t1 and t2 is

calculated using cosine similarity. Table 4.1 below shows how similar two words are as generated form our embedding and their similarity score.

Table 4.1: Similarity scores between terms t1 and t2

Term1	Term2	Similarity score
ሠላም(Peace)	ሠላም(Peace)	1.0
መጣ(he comes)	ሄደ(he goes)	0.693
በላ(he eats)	ሰራ(he works)	0.519
መጣ(he comes)	ዩኒቨርስቲ(university)	0.199

From the above table, let's see the three terms መጣ (he comes), ሄደ (he goes) and ዩኒቨርስቲ (university), and computer their similarity scores suing the cosine distance measure.

sim(መጣ, ሄደ) = cos(vec(መጣ), vec(ሄደ)) == 0.693

sim(መጣ, ዩኒቨርስቲ) = cos(vec(መጣ), vec(ዩኒቨርስቲ)) ==0.199

Therefore, as the results show the words መጣ (he comes) and ሄደ (he goes) are semantically closer than ሄደ (he goes) and ዩኒቨርስቲ (university). The similarity score of a word with itself is obviously a unit as clearly put in the first row of Table 4.1.

The cosine distance between words defines how much related two words are. It describes the similarity level and relatedness between words and how they go and found together in the corpus. The Figure 4.1 below shows words in t-SNE visualization method and their neighborhood. The figure depicts places, languages, people names and foods based on their relatedness. The languages are closer to each other than others. This is separately put in Figure 4.2 as a magnified version focusing on the clustered languages names of Figure 4.1.

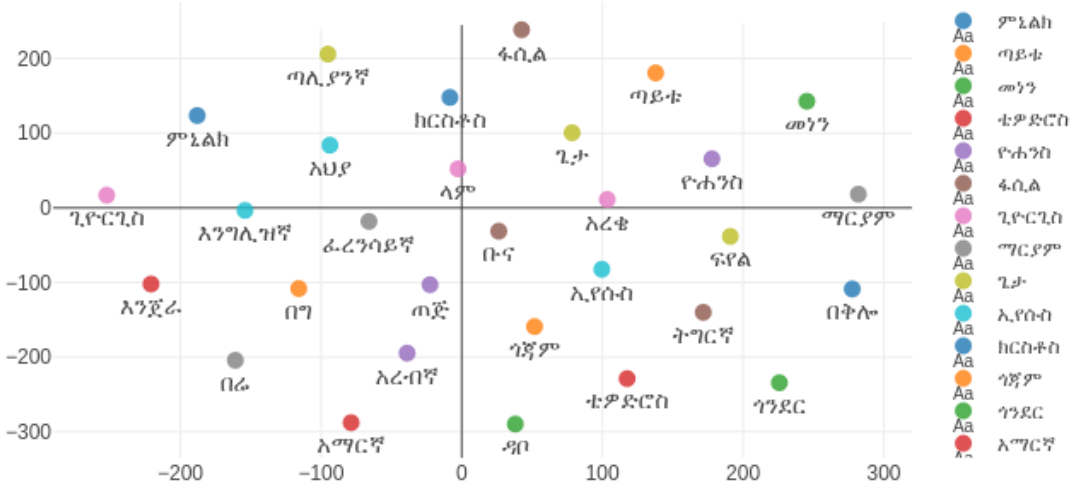

Figure 4.1: t-SNE cosine distance between words that are put in the right side of the picture. The words are names of people, languages, places, animals and foods.

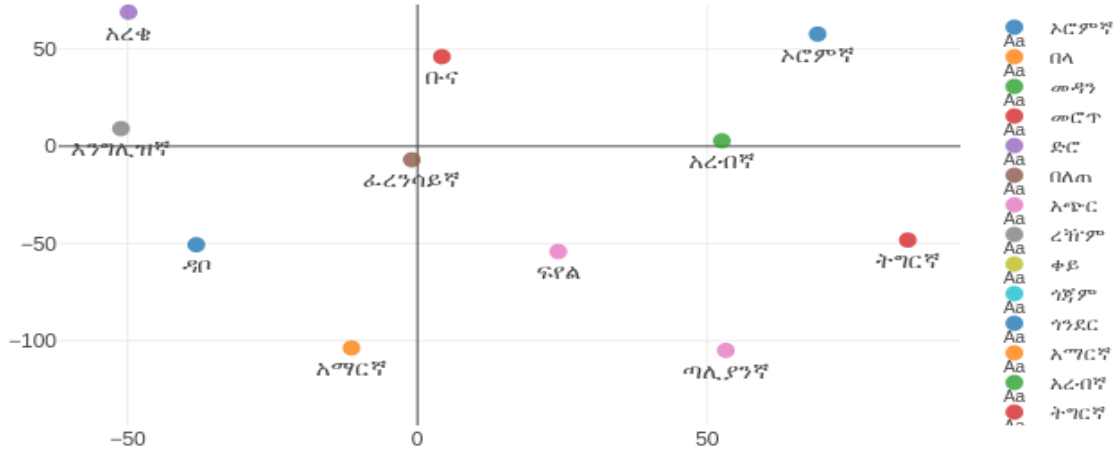

Figure 4.2: Magnified version of Figure 4.1 to show how names of languages are closer to each other

Word similarity also includes nearest neighbor searches to evaluate how well k-nearest neighbors are generated (more on this in Section 4.3.1.3), where k is an integer number. Embeddings capture proximity relationships between objects. In word embeddings, related words are put nearer to each other. Nearest neighborhood refers to closest neighbors to a

given word in an embedding space. This task helps to see if a word vector captures morphological, syntactic and/or semantic relations of words correctly.

Table 4.2: Most similar words for words: ልዑል (Prince), ሰው (Human) ነገሠ (Reign)

ልዑል(Prince)	**ሰው(Human)**	**ነገሠ (Reign)**
የልዑል (for prince)	ከሰው (from people)	ከነገሠ (if he reigned))
በልዑል (by prince)	ሰውዬ (you man)	ነገሠች (she reigned)
አልጋወራሽ (crown prince)	ሰውና (human and)	ከተከታዩ (from his follower/s)
ወራሽ (heir)	ሰውንም(and human)	ቀጥሎና (next and)
ልዕልት (princess)	ያለሰው (without human)	ንጉሥ (king)
	ሰውን (the people)	ንግሥት (queen)

As the above table shows words with their morphological variations and their variances based on POS tags are put as related words. As Amharic has more inflected forms, the retrieved results for similarity quest is more of inflected forms of the words in question. For example, the results in table above in second column to the word "ሰው Human" are all related to people/human. Moreover, all words in the response share the root word "ሰው" which can show that the result is a combination of both semantic and morphological relatedness.

4.3.1.2 Word analogy relation

When a large dataset is used in training for word embeddings to represent words in vectors, the resulting vectors have the ability to learn subtle relations between the words (Mikolov et al., 2013a). Figure 4.3 below illustrates this claim. The figure demonstrates city-country relationships

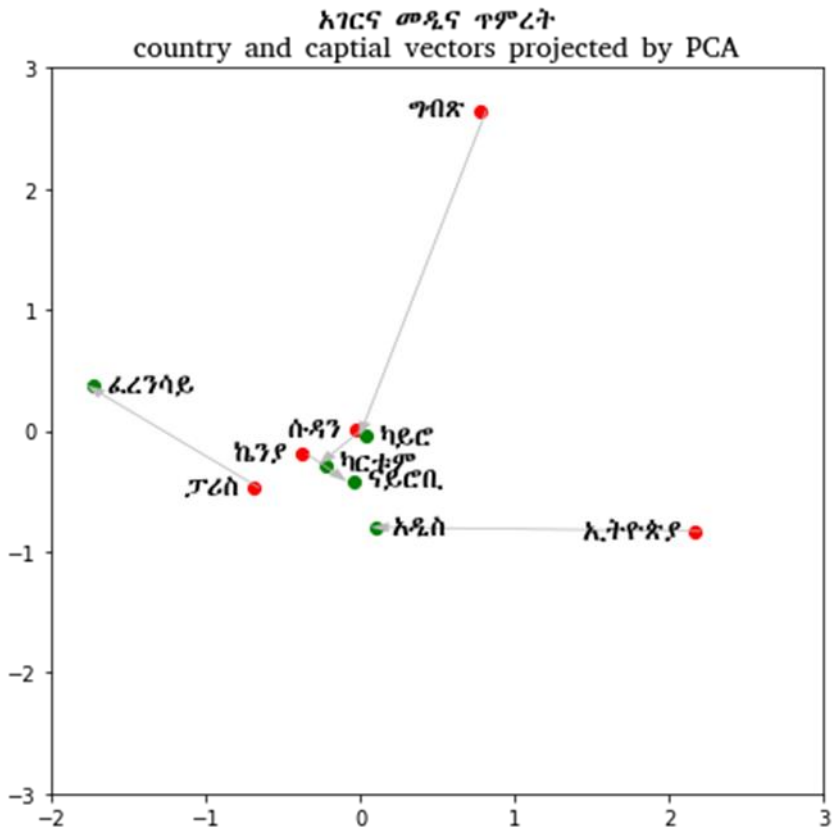

Figure 4.3: 2-D projection of 300-dimensional vectors of countries and cities

The figure shows the ability of the word embedding produced, though not supplied with any supervised information about what a city means to a country, to automatically organize concepts and catch relationships between them. In word analogy, the task is to find a word w1 for a given word w2 so that w1: w2 best resembles a sample relationship w3: w4.

Word analogy relation is a good way of examining and evaluating the quality and goodness of a word embedding. This linguistic property of words is manifested in word embeddings. For example, according to (Mikolov et al., year), the correlation "if man is to king, woman is to what?" is an analogy that word embeddings can solve. The answer is *queen* which is logically correct. Say, a tuple like "ግብጽ(Egypt) : ካይሮ(Cairo) :: ኢትዮጵያ(Ethiopia) : አዲስአበባ(Addis Ababa)", the embedding model should produce correct results if the nearest vector representations to vector(ግብጽ) - vector(ካይሮ) + vector(ኢትዮጵያ) is vector(አዲስአበባ). In word analogies the task is to find a vector v such that vector(v) is closest to vector(ግብጽ) - vector(ካይሮ) + vector(ኢትዮጵያ) according to the cosine distance.

The relationships for the analogy might vary due to various reasons. One of the reasons is the size of the dataset or the corpus. The result of word analogy is heavily influenced by the quality and quantity of the corpus used during the training. A minimal corpus might not give the desired analogy. As the main factor is the distances between vectors representing the words in competition, semantics and logic does not involve in the output. How word analogy in word embedding works is based on the experimented truth that two groups of words that have similar relationships should be located similar distances apart in the vector space.

This analogical reasoning task has two categories: the semantic and syntactic analogies.

Table 4.3: Semantic analogy

Relatedness	**Word pair 1**	**Word pair 2**
Capital-country	ፓሪስ(Paris) – ፈረንሳይ(France) ሱዳን(Sudan) – ካርቱም(Khartoum)	ካርቱም(Khartoum) – ሱዳን(Sudan) ግብጽ(Egypt) - ፈሮዖን (Pharaoh)
Country-continent	ኢትዮጵያ(Ethiopia) – አፍሪቃ(Africa)	ቱርክ(Turkey) - ኢሲያ(Asia)
Man-woman	ወንድ(man) – ንጉሥ(king)	ሴት(woman) – ንግሥት(queen)
Opposite	አጭር(short) – ረዥም(tall)	ነጭ(white) - ጥቁር(black)

Table 4.4: Syntactic analogy

Relatedness	**Word pair 1**	**Word pair 2**
Plural suffixes	እናት(mother) – እናቶች(mothers)	አባት(father) – አባቶች(fathers)
Passive voice suffixes	ገደለ(he killed) – ተገደለ(he was killed)	ሰማ(he listened) – ተሰማ(he was listened)
Pronoun/verb suffixes	ሰጠ(he gave) – ሰጡ(they gave)	ቀማ(he plundered) – ቀሙ(they plundered)
	መስማት(listening) – መስማቱ(him listening)	መናገር(speaking) – መናገሩ(him speaking)

4.3.1.3 Observations from intrinsic evaluations

We experimented with the word vector by tuning fastText parameters to evaluate the embedding obtained from the corpus. Several test words for similarities (nearest neighborhood), analogy and OOV were selected. As tables Table 4.2 and Table 4.5 show, the results were more concentrated on inflected forms of Amharic words. This happens due to the richness of morphology in Amharic. Amharic is a morphologically rich language and there is a high rate of morphological production in it. Since there are plenty of inflected forms for a given word in Amharic, the possibility of having inflected forms of a single word in a single cluster or contiguously is high.

We noticed that syntactic and morphological relatedness gets improved with shorter window size. Table 4.5 compares the top 5 nearest neighbors of the word በላ (he eats) both using window size (ws) 2 and 5. Note that the underlined words in the table are words that do not have correlation to the task.

Table 4.5: Top 5 nearest neighbors for a word: በላ

Ws	Word:በላ (he eats). Top 5 Nearest neighbors
5	ሊበላ(to eat), <u>አከራይቶ</u>(hire out), በላች(she eats), ለጠጣ (he who drinks), ከጠጣ(if he drinks)
2	በላች(she eats), በላሁ(I ate), ሆድ(stomach), <u>በላዩ</u>(over him/it), በላህ(you ate)

As the window size or context size is shorter, the result gets finer. However, a careful investigation of semantic relationship in different parameters vary accordingly. For example, semantic relatedness, in contrast with the analogical variations related to syntax and morphology, gets improved when the context is longer. The following table compares the results of analogical reasoning related to semantics like “city-country” and “man-woman” with two values of window size: 2 and 5.

Table 4.6: Analogical reasoning with varying window size

Semantic relatedness	Ws=2	Ws=5
ወንድ(man)-ንጉሥ(king)	ንግሥት(queen) -	ንግሥት(queen) -
	ለወንድ(for male)	ለወንድ(for male)
	የወንድ(to male)	ሚስት (wife)
	ወንድም(and male)	ወንድና(male and)
	ወንድና (male and)	ሴት (woman)
	ወንድምሽ (your brother)	ከወንድ(from male)
	ወንድምና(brother and)	ወንድምዋ(her brother)
	ሴት(woman)	የወንድ(to male)
ፓሪስ(paris)-ፈረንሳይ(France)	ሱዳን(Sudan) -	ሱዳን(Sudan) -
	ምስራቃዊ(eastern)	ናይሮቢ (Nairobi)
	ካርታ(map)	በናይሮቢ(by Nairobi)
	በሱዳን(by Sudan)	በአስመራ (by Asmara)
	በምስራቃዊ(in eastern)	ካርቱም (Khartoum)
	ምስራቅን(the east)	በረሩ (they flew)
	አረቢያ(Arabia)	ተከፈተ(opened)
	ኪሜ(km)	ከሱዳን (from Sudan)

In this table the male-female relationship is better learned when context is 2 than when context is 5. Therefore, the vector operation man(ወንድ)-King(ንጉሥ)+ queen(ንግሥት) results in a vector close to woman(ሴት) given a wider window size. Wider context or larger window size improves semantic relatedness while narrow context or small window size generates more inflected forms and is better for morphological relatedness.

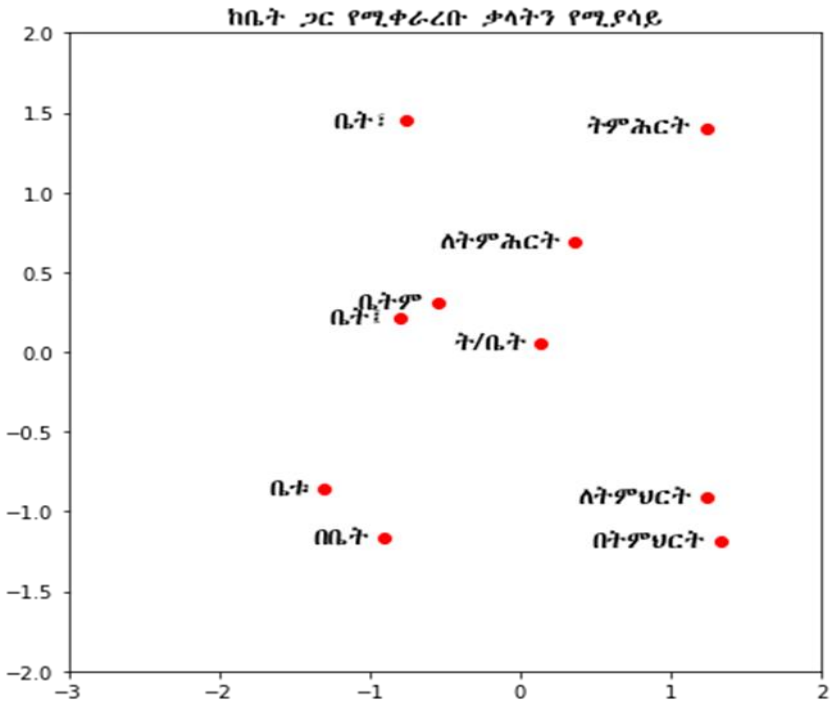

Figure 4.4: PCA visualization showing both morphological and semantic relatedness

Semantic and morphological relatedness do not happen exclusively. These relationships occur together as shown in Figure 4.4. Both semantic and morphologically related words with the word: ቤት (House) is displayed and visualized using a dimensional reduction method PCA.

The other case is model types: CBOW vs SG. FastText uses both models for production of word vectors. Using the same parameters in both models, the results produced in the embedding do not vary a lot. As the table below can show, skipgram model performs a little better and gives a result which is reasonable. Therefore, skipgram and CBOW gives results that are nearly equivalent or closer. Note that the underlined words in the table are words that do not have correlation to the task.

Table 4.7: Word relatedness with the two models: CBOW and SG

Model	Word:በላ (he eats). Top 5 Nearest neighbors
SG	ሊበላ(he-to eat), *ኣከራይቶ*(hire out), በላች(she eats), ለጠጣ (he who drinks), ከጠጣ(if he drinks)
CBOW	በላች(she ate), ቁስል(wound), በላሁ(I ate) , ዓይኑም(and his eye), በላዩ(above him)

When we visualize our embedding using PCA and t-SNE, an interesting result are displayed. To check the quality of the Amharic word embedding produced, the word vectors are projected on a 2-dimensional space using a dimension reduction technique called t-SNE.

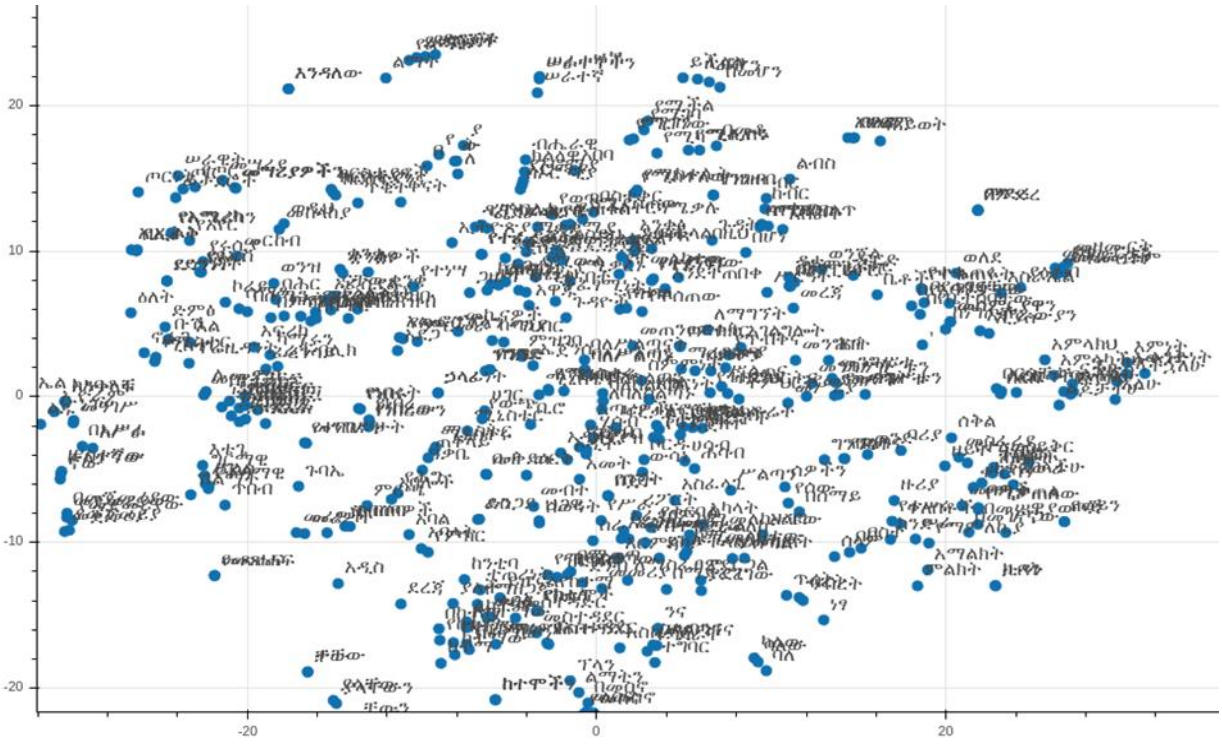

Figure 4.5: t-SNE embedding of top 500 words (using default parameters)

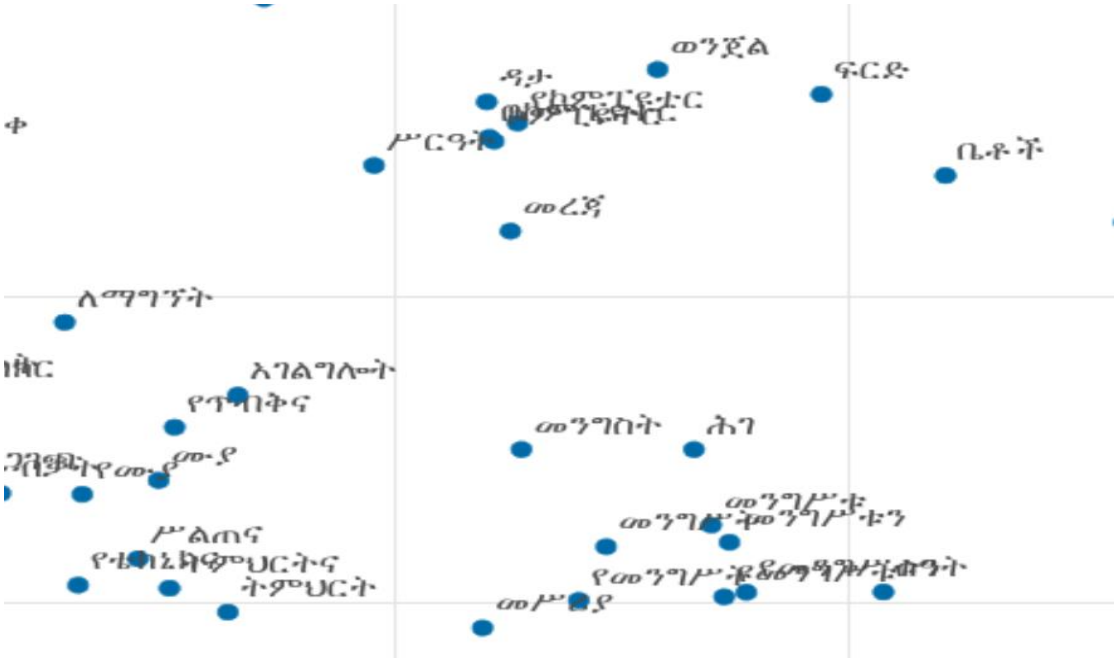

Figure 4.6: Magnified clusters clipped from Figure 4.5

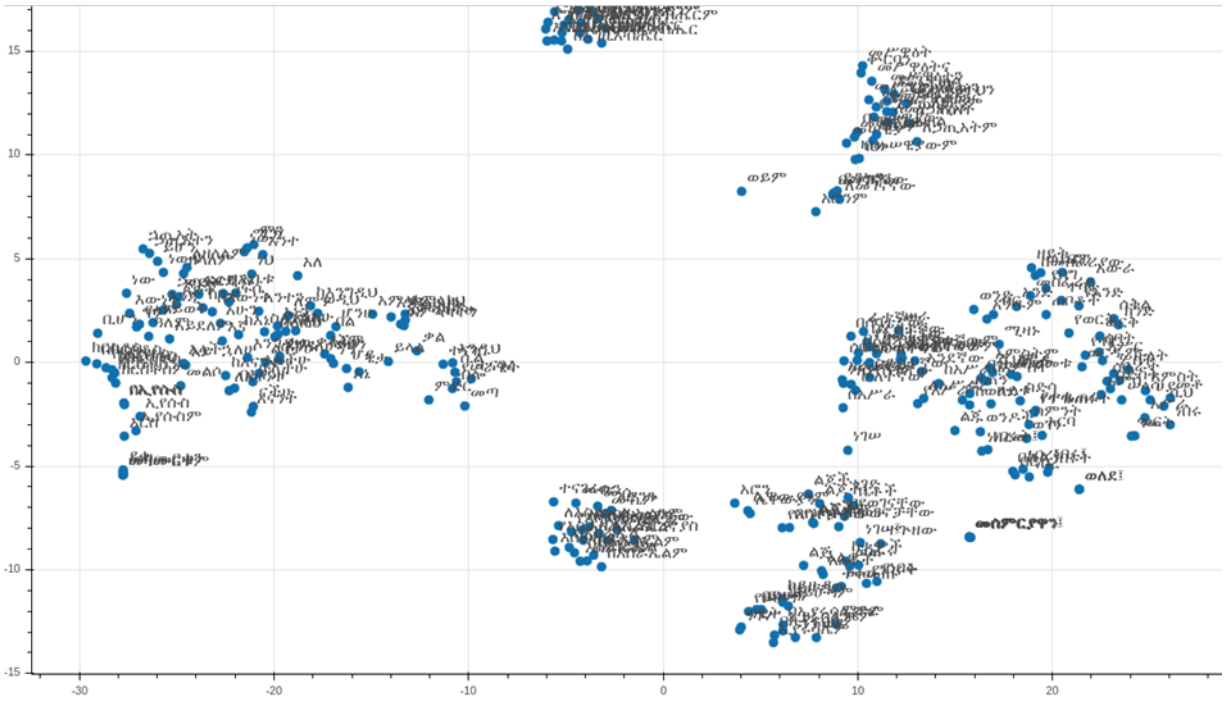

Figure 4.7: t-SNE embedding of top 300 words

The subset of words from the dataset plotted using t-SNE is visualized in the above figure. As expected, words with high similarity are clustered.

When Figure 4.5 is analyzed closely, we realize that semantically and morphologically related words are clustered together. This can be observed in Figure 4.6 where the magnified clusters are separately clipped. Semantically similar words like ዳታ (data) and መረጃ (information) got clustered together and morphologically related ones like መንግሥት (government) and መንግሥቱን (the government) are put closer to each other.

i) Effect of corpus size

To analyze the effect of data size, we trained a corpus (defaulted for the rest of the work in this paper) and part of the corpus (segmented only for this issue) using the default fastText parameters. We can see from Table 4.8 that when trained on a large dataset, the result tends to move more toward morphological similarity vectors of the query words.

Table 4.8: Corpus size and word relatedness

Corpus:412K size	**Corpus: >1M size**	**Pretrained size by fastText**
በላ(He ate)	**በላ(He ate)**	**በላ(He ate)**
ትበላ (you eat)	ሊበላ (he-to eat)	ሊበላ (he-to eat)
በላዩ (over him)	አከራይቶ (hire out)	ይበላ (he would eat)
ጥዋትም (and morning)	በላች (she ate)	ስበላ (while I eat)
መብልም (and food)	ሊጠጣ (he-to drink)	ከበላ (if he ate)
ሊበላ (he-to eat)	ከጠጣ (if he drank)	ቆዳው(the skin)

ii) Dimension

The other parameter that needs attention is the size of dimension during training. In this case, a comparison between three dimensions: 50,100, 200 and 300 shows that a little improvement is observed as dimension increases. However, the difference in the output between the two dimensions, 200 and 300, is found vague, as the table below shows.

Table 4.9: Dimension and word relatedness

dim	Words	k-5 nearest neighbors of the word
50	በላ(he eats)	.ሊበላ(he-to eat), ቂጣና(bread and), አከራይቶ(hire out), ገፈራ(?), ይጠጣ(he can drink)
	እምነት(creed)	.ለእምነት(for creed), ከእምነት(from creed), እምነትና(creed and), በእምነት(by creed and), እምነትም (and creed)
100	በላ	ሊበላ , አከራይቶ, በላች(she ate), ለጠጣ(for who drinks), ከጠጣ(if he drinks)
	እምነት	በእምነት, እምነትም, ለእምነት, ከእምነት, እምነትና
200	በላ	በላች, በላህ (you ate),በላው(he ate it), በላሁ(I ate), ጠጉሩም(and his hair)
	እምነት	ለእምነት, እምነትም, ከእምነት, በእምነት, እምነትና
300	በላ	ገፈራ(?), በላች, በላህ, ሊበላ, ከጠጣ
	እምነት	ለእምነት, በእምነት, ከእምነት, እምነትም, የእምነት(to creed)

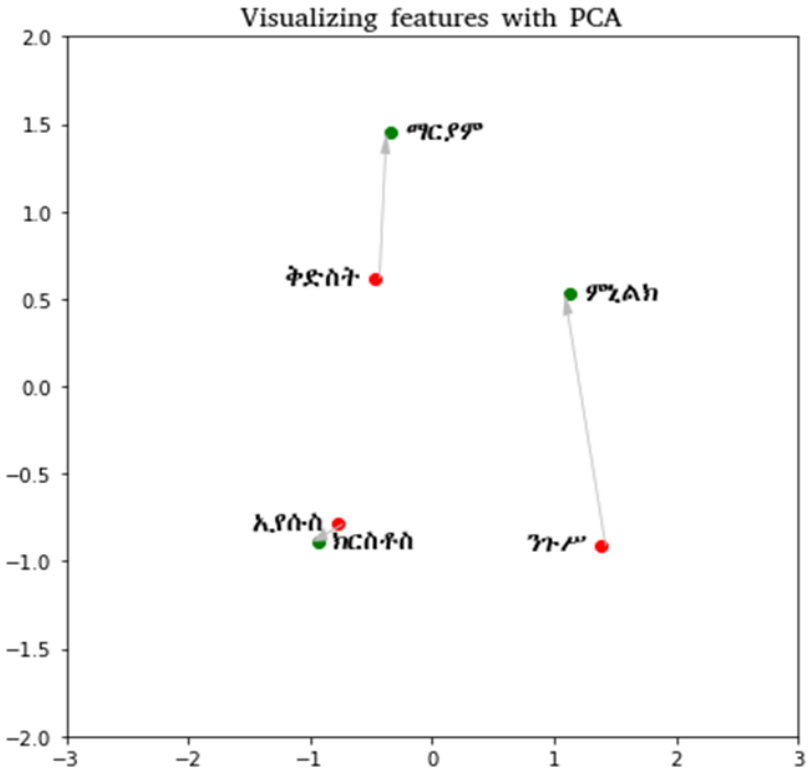

Figure 4.8: PCA visualization showing word relationship

To summarize our observation on intrinsic evaluations, fastText Amharic word embedding perform better on morphological similarities due to the language's richness in morphemes and fine-tuning hyper-parameters further improves word similarity results.

4.3.1.4 Out of vocabulary words and odd-word out

In fastText, one of the peculiar features compared to other models like GloVe and Word2vec is its ability to handle new, unknown, rare, misspellings, and out-of-vocabulary words or words that are not element of the training set. Given a word w, where the vocabulary being V and $w \notin V$is true, fastText enables us to harness a vector representation of w. Since the constituent fragments of a word w in the range between the two parameters -minn and -maxn are taken into account. The vector representation of w is then computed from the vectors representations of those fragments. Summing up the n-gram vectors or subword vectors would result in the vector representation of an OOV word w.

Figure 4.9 and Figure 4.10 below show a sample OOV words w1 =" ቅድስታምረት" and w2=" ሰበርታምዕራ". These words are not part of the training set. They are not common Amharic

words either. But their vector representation is computed from the constituent parts of each word's character n-grams from -minn to -maxn.

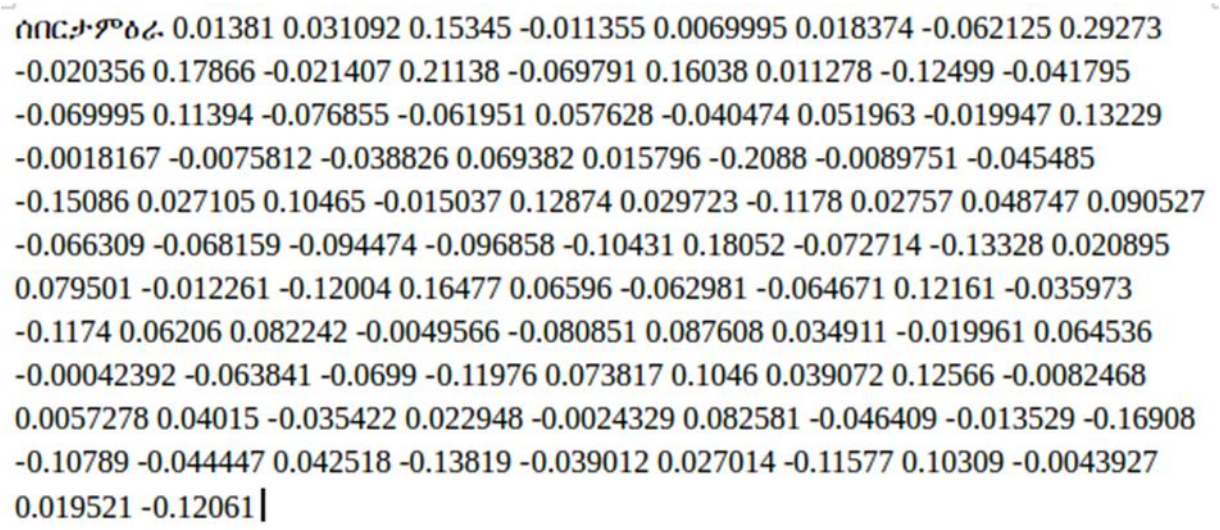

Figure 4.9: 100-dimensional WE of OOV word-ሰበርታምዕራ

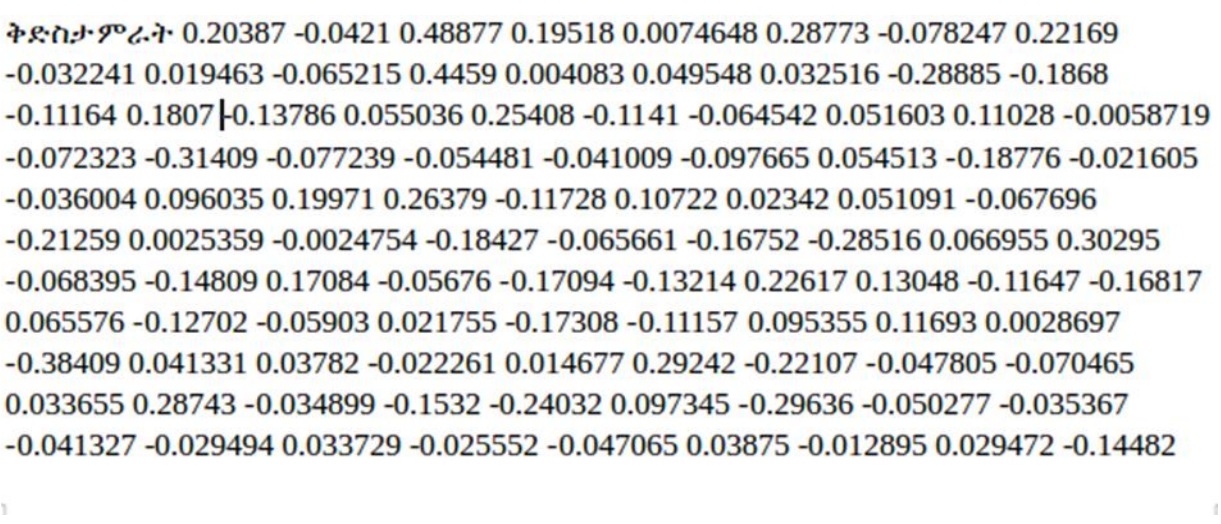

Figure 4.10: 100-dimensional WE of OOV word-ቅድስታምረት

Since these unknown words are represented in vector forms, the linguistic properties of other words are also their characteristics. They have neighbors that are both nearest or farthest, similarities records, analogical relationships and so on.

Table 4.10 shows the nearest neighbors of those OOV words and the cosine distances between the neighbors and the OOV words.

Table 4.10: Nearest neighbors for OOV words and their cosine distance

W1: ቅድስታምረት	nn	ዘረ	ቁጤማ	ቅሪት	ድፍረት	ጥንተ
	cosdis	0.857	0.841	0.840	0.838	0.833
W2: ሰበርታምዕራ	nn	በምዕራባዊ	ወደምዕራባዊ	ምዕራባዊ	በስተምስራቅ	ደቡብና
	Cosdis	0.894	0.893	0.882	0.856	0.852

As the table shows, the nearest words for the OOV words, w1, and w2 can be found. nn (nearest neighbor) shows words that are closer to w1 and w2. *Cosdis* is the cosine distance between each neighbor words and the two words w1, and w2.

Concepts and words that are related to w1 and w2 have closer cosine distance score to one another.

Table 4.11: Odd word out results

I	II	III	IV
ንጉሥ(king)	ልዑል(prince)	ንግሥት(queen)	**መምህር(teacher)**
ታሪክ(story)	ፊልም(film)	ድራማ(drama)	ተማሪ(student)
ጋዜጣ(gazette)	ራዲዮ(radio)	ቴሌቪዥን(TV)	**ዜና(news)**
ተጓዘ(walk)	ሮጠ(run)	መንገድ(road)	**ሽንት(urine)**

Odd word out is not related to OOV but it is a task that can show the other useful feature of word embeddings. Testing our word embeddings to identify words from a list which does not go with the others gives promising results. It can easily pick words that do not belong to a list (either semantically, syntactically, or morphologically). In Table 4.11 (above), words at column IV are odd words in relation to the other three words in the same row. The words in bold are results returned as odd by the model. The odd words are selected and isolated in

most part of the test excepting some instances like words in row II. In this row, as ታሪክ(story) is more related to ፊልም(film) and ድራማ(drama) than ተማሪ(student) is, the result should have been ተማሪ(student) as odd word instead of ታሪክ(story).

4.3.2 Extrinsic evaluation

This is another evaluation method to investigate and analyze the contribution of word embeddings in tackling downstream NLP problems such as text classification and POS tagging. We used multi-class text classification for this evaluation task.

4.3.2.1 Text classification

We used our preprocessed news dataset for text classification purpose. Our dataset has four labels: Athletics, Economy, Football and Chess. Each line of the news file contains __label__prefix at its start so that fastText can recognize what is a word or what is a label.

Table 4.12: Number of datasets in each group and ratio

Label	**70/30**		**80/20**		**90/10**	
	Number of news items in train group	Number of news items in validation group	Number of news items in train group	Number of news items in validation group	Number of news items in train group	Number of news items in validation group
Football	213	119	249	83	293	39
Athletics	134	20	152	2	152	2
Chess	12	9	12	9	12	9
Economy	271	122	307	86	353	40

Before training our classifier, the dataset has been split into train and validation. A number of experiments were carried out both in 90/10, 80/20 and 70/30 proportions of the dataset, where the proportion is the ratio of the train and validation sets (For example: 80/20 means 80% for training and 20% for validation/testing data). The train and validation datasets are

stored in text format. The number of news items in each group and in each ratio of the split are presented in Table 4.13. We used different hyper-parameters on the dataset to train and evaluate the precision, recall and accuracy in predicting and classifying a new item. The following table shows the overall results of different tests conducted using default parameters but with alternating epoch size.

Table 4.13: Precision and Recall at K=2, and F1-score using different epochs

Epoch	**70/30**			**80/20**			**90/10**		
	P@2	R@2	F1	P@2	R@2	F1	P@2	R@2	F1
5	0.474	0.948	0.632	0.472	0.944	0.629	0.444	0.889	0.592
25	0.481	0.963	0.641	0.475	0.950	0.633	0.450	0.900	0.600
50	0.493	0.985	0.657	0.483	0.967	0.644	0.467	0.933	0.622
100	0.498	0.996	0.640	0.500	1.000	0.666	0.500	1.000	0.666

Table 4.14: F1-score at K=1 using different epochs

Epoch	**70/30 F1**	**80/20 F1**	**90/10 F1**
5	0.722	0.689	0.833
25	0.952	0.944	0.889
50	0.948	0.944	0.889
100	0.959	0.961	0.922

On average, as the tables show, the precision increases as the epoch increases. We see that on average the variables, precision, recall and F1-score gain an increment when the number of epochs rise up. Take the 80/20 as example, the F1-score of our classifier has become 0.961

at epoch number 100. An increased value of epoch makes an improved classifier model. That means increasing the number of epochs steadily improved the classifier quality.

Table 4.15: Example from a validation set obtained with 100,000 epochs on 80/20 sample, label predictions included.

Input	Prediction	Evaluation
አንድ አሜሪክ ዶላር ብር ሳንቲም ተሸጠ የውጭ ምንዛሪ የውጭ ምንዛሪለኢንተርኔት ባንክ መካከል ዛሬ ተካሄድ የውጭ ምንዛሪ ገበያ አንድ አሜሪክ ዶላር አማካይ ብር ሳንቲም መሸጡን ኢትዮጵይ ብሄራዊ ባንክ አስታወቅ ባንኩ ዛሬ ኢዜእ ገለፀ በእለቱ ባንክ መካከል ተካሄድ የውጭ ምንዛሪ ገበያ ለአንድ አሜሪክ ዶላር ቀረብ ከፍተኛ ዋጋ ብር ሳንቲም ዝቅተኛው ዋጋ ደግሞ ብር ሳንቲም ሆኗል በሌላ በኩል ደግሞ በኢድስ አበባ ማእከላው ገበያ ዛሬ ሽህ ኩንታል ደረቅና ሽህ ኩንታል ታጠብ ቡና መቅረብ ቡን ሻይ ባለስልጣን አስታውቋል በእለቱ ሽህ ኩንታል ደረቅና ሽህ ኩንታል ታጠብ ቡና ተሸጧል አንድ ፈረሱላ ታጠብ ቡና ተሸጥ የእለቱ አማካይ ዋጋ ነ92ብር ሳንቲም ሲሆን ደረቅ ቡና ደግሞ ብር ሆኗል	Economy	Correct
አትሌት ቀነኒሳ አሸነፈ	Athletics	Correct
የኢትዮጵያ ፕሪምየር ሊግ ተስተካካይ ጨዋታዎች በተለያዩ ከተሞች ተከናውነዋል በዛሬው ተስተካካይ ጨዋታ አዳማ፣ መቀሌና አዲስ አበባ ላይ ሦስት የፕሪምየር ሊጉ ጨዋታዎች ተካሂደዋል በዚህም አዳማ ላይ በአበበ ቢቂላ ስታዲየም ቅዱስ ጊዮርጊስን ያስተናገደው አዳማ ከተማ ዐ ለ ዐ በሆነ ውጤት ጨዋታውን አጠናቋል የሊጉ መሪ ቅዱስ ጊዮርጊስ ከአዳማ ከተማ ጋር ያደረገው ጨዋታ ከተከታዮቹ ጋር ያለውን ነጥብ ማስፋት የሚችልበት ዕድል ነበር በሊጉ የመጨረሻ ደረጃ ላይ የሚገኘው ደደቢት መቀሌ ላይ መከላከያን አስተናግዶ 3 ለ ዐ በሆነ ውጤት ተሸንፏል ሦስተኛው የፕሪምየር ሊግ ተስተካካይ ጨዋታ በወላይታ ድቻና ሲዳማ ቡና መካከል ማምሻውን በአዲስ አበባ ስታዲየም ተካሂዷል በዚህ በሁለተኛ ሳምንት ተስተካካይ ጨዋታቸው ሲዳማ ቡና 2 ለ ነ በሆነ ውጤት ወላይታ ድቻን አሸንፏል	Football	Correct
ሮናልዶ በአንድ ጨዋታ ሶስት ጎል አስገባ	Football	Correct
ቼዝ ጨዋታ በአማራና ደቡብ ክልሎች እየተካሄደ መሆኑ ተነገረ	Football	Incorrect
መቀሌ መጋቢት 6/2011 በትግራይ ለመጀመሪያ ጊዜ የተዘጋጀ ክልል አቀፍ የቼዝ ውድድር ዛሬ በመቀሌ ከተማ ተጀመረ በውድድሩ በአገር አቀፍ ደረጃ ከሚያዝያ 5 እስከ 16 ቀን 2011 ዓ ም አዲስ አበባ ላይ ለሚካሄደው የክለቦችና የግል ሻምፒዮና ውድድር ክልሉን ወክለው የሚሳተፉ ስፖርተኞች እንደሚመረጡ ተነግሯል የትግራይ ክልል ቼዝ ፌዴሬሽን ከፍተኛ ባለሙያ አቶ ካህሳይ ፍሰሃ ለኢትዮጵያ ዜና አገልግሎት እንደገለፁት በክልሉ ከዚህ ቀደም የቼዝ ስፖርት ትኩረት የተሰጠው አልነበረም ውድድሩም ከግል ደረጃ ያለፈ ባለመሆኑ የዘርፉ ስፖርት ሳያድግ መቆየቱን ጠቁመዋል “ዘንድሮ ለዘርፉ በተሰጠው ትኩረት በክልል ደረጃ 12 ክለቦች ተደራጅተው ውድድሩ	Chess	Correct

ተጀምሯል" ብለዋል ከክለቦቹ መካከል አራቱ የሴቶች መሆናቸውን የገለፁት ባለሙያው በግልና በክለቦች መካከል ለአራት ቀን በሚካሄደው ውድድር 100 ስፖርተኞች ተሳታፊ መሆናቸውን ተናግረዋል እንደ ባለሙያው ገለጻ በውድድሩ እስከ ሦስተኛ ደረጃ በመያዝ ለሚያጠናቅቁ አሸናፊዎች የሜዳሊያ የዋንጫና የገንዘብ ሽልማት ይበረከታል ከተወዳዳሪዎች መካከል ወጣት ያሬድ ኃላፎም በሰጠው አስተያየት " ስፖርቱ በመንግስትና በስፖርት ማህበረሰቡ ትኩረት ከተሰጠው በክልልና በሀገር ደረጃ የሚወከሉ ተወዳዳሪዎችን ማፍራት ይቻላል" ብለዋል ወጣት ሳምራዊት ተክሉ በበኩሏ "ቼዝ የአካል ጥንካሬን ሳይሆን የአዕምሮ ብቃት የሚጠይቅ የስፖርት አይነት ነው" ብላለች ለስፖርት አይነቱ ትኩረት ሊሰጥ እንደሚገባም ወጪቷ አመልክታለች		
ድሬዳዋ ሀምሌ 2/2010 በድሬዳዋ ለአንድ ሳምንት ሲካሄድ የቆየው የክልሎች ከ20 ዓመት በታች የቼዝ ሻምፒዮና ውድድር በቡድን የትግራይ ክልል አሸናፊነት ተጠናቀቀ ድሬዳዋና አማራ በግል የተካሄደውን በአሸናፊነት አጠናቀው የዋንጫና የወርቅ መዳሊያ ተሸላሚ ሆነዋል ዛሬ ማምሻውን በተካሄደው የመዝጊያ ሥነ-ሥርዓት ላይ በበላይነት ውድድሩን ያጠናቀቁ ስፖርተኞች በቀጣይ ሀገራቸውን በስፖርቱ ብርቱ ተፎካካሪ ለማድረግ እንደሚሰሩ አስተያየታቸውን ለኢዜአ ሰጥተዋል የኢትዮጵያ ቼዝ ፌደሬሽን ከድሬዳዋ አስተዳደር ጋር በመተባበር ባዘጋጀው ከ20 ዓመት በታች ሀገር አቀፍ የቼዝ ሻምፒዮና ውድድር ላይ ድሬዳዋና አዲስ አበባን ጨምሮ የትግራይ የአማራ የደቡብና የኦሮሚያ ክልሎች ተካፍለዋል ከሰኔ 25 እስከ ሐምሌ 2 ቀን 2010 በተካሄደው በዚሁ ውድድር የትግራይ ቡድን በሴትና በወንድ የቡድን ውድድሮች በአንደኝነት በማጠናቀቅ የዋንጫና የወርቅ መዳሊያ ተሸላሚ በመሆን ውድድሩን በበላይነት አጠናቋል በሴቶች የቡድን ውድድር ሁለተኛ አማራ ሶስተኛ ደግሞ ኦሮሚያ ሲሆኑ በወንዶቹ ምድብ ደግሞ ኦሮሚያ ሁለተኛ ድሬዳዋ ሶስተኛ ሆነዋል በግል የወንዶች ውድድር የድሬዳዋ ማራኪ እንድሪያስ የበላይ ሆኖ የወርቅ መዳሊያና የዋንጫ ባለቤት ሲሆን በሴቶች የግል አሸናፊ የሆነችው የአማራ ክልል ስፖርተኛ መቅደስ አማረ ናት በሽልማት ሥነ-ሥርዓት ላይ የትግራይ ስፖርተኛ ክብሮም በርሔ በሰጠው አስተያየት "ክልላችን ለስፖርቱ ልዩ ትኩረት ሰጥቶት በመሰራቱ ውጤታማ ሆነናል" ብሏል "በቀጣይ ሀገሬን ልከ እንደአትሌቲክሱ በአለም አደባባይ የማስጠራት ዓላማዬን ለማሳካት ጠንክሬ እሰራለሁ" ብሏል "ውድድሩ ችሎታዬን ይበልጥ እንዳሳይ አድርጎኛል አንድ ቀን ድሬደዋንም ሆነ ሀገሬን በስፖርቱ ስማቸው እንዲነሳ አደርጋሁ" ያለው ደግሞ በወንዶች የግል ውድድር የበላይ ሆኖ የወርቅና የዋንጫ ተሸላሚ የሆነው ማራኪ እንድሪያስ ነው የአማራ ክልልን ውጤታማ ያደረገችው መቅደስ አማረ ውድድሩ ጥሩና ጠንካራ ፉክክር የታየበት እንደነበር ጠቅሳ ስፖርቱ ይበልጥ መሰረቱ እንዲጠናከር ከ20 ዓመት በታች ለሆኑት ወጣቶች ተለይቶ የተዘጋጀው ውድድር ሊበረታታ እንደሚገባ ተናግራለች የኢትዮጵያ የቼዝ ፌደሬሽን ጽህፈት ቤት ኃላፊ አቶ ሰይፉ በላይነህ ለአንድ ሳምንት ከተካሄደው የታዳጊ ወጣቶች ውድድር ለበሔራዊ ቡድን የሚመጥኑ ምርጥ ስፖርተኞች እንደተገኙበት ጠቅሰው እነዚህ ስፖርኞች ይበልጥ ውጤታማ እንዲሆኑ በትኩረት እንደሚሰራ ገልፀዋል ስፖርቱ ይበልጥ እንዲጠናከር በየዕድሜ	Chess	Correct

እርከኑ በርከታ ውድድሮች እየተዘጋጁ መሆኑንና እድሜያቸው ከ13 ዓመት ጀምሮ ያሉ ወጣቶች በፕሮጀክት እንዲታቀፉ እየተደረገ መሆኑን ተናግረዋል በመዝጊያው ሥነ-ሥርዓት ላይ ለአሸናፊዎቹ የተዘጋውን የዋንጫ የወርቅ የብርና የነሐስ መዳሊያ የሸለሙት የድሬዳዋ አስተዳደር ወጣቶችና ስፖርት ኮሚሽን ኮሚሽነር ከድር ጁሃርና ሌሎች እንግዶች ናቸው

We tested the model with epoch number 100,000, other hyper-parameters being defaulted, and it took about 2 hours to train on Intel® Core™ i3-64 core. This test produced 0.978 F1-score on 80/20 sample of the dataset. Varying other parameters like window size and dimension made a marginal improvement on the quality of the classifier. Further parameter tuning gives us a significant boost on speed and quality. Overall, the Amharic word embeddings can be used in text classification tasks with a better quality than other traditional methods.

4.4 Summary

Word embeddings for Amharic is found to be a useful tool to analyze the language's linguistic properties and bring the concept of mathematics to the field. It captures properties like context, semantics, syntactic, analogy and so on. The embedding is also found to be helpful for downstream tasks such as Text classification. The embedding of words in a vector form in space helps to learn features and patterns natural languages form.

The word embeddings capture relationships not only for words that are part of the vocabulary, but also for words that are rare, misspelled, typos, out-of-vocabulary etc… The results obtained are very promising in learning language features, relationships and patterns in Amharic language. The richness of Amharic inn morpheme has found influencing the results, however.

CHAPTER 5
CONCLUSION, RECOMMENDATION AND FUTURE WORKS

5.1 Conclusion

In this work we explored word embeddings for Amharic language. A thorough discussion on word embeddings in general, starting from their history to the current state-of-the-art findings, methodologies and architectures were conducted. Word embeddings are analyzed based on the properties of languages. Language properties like similarity, analogy and relatedness based on syntax and morphology are issues considered.

As is common in NLP, datasets or corpus were prepared for use in preparing word embedding for Amharic. The steps needed to make the corpus ready, called preprocessing, were done and the training was conducted. In the preprocessing phase we omitted punctuation marks, extra empty spaces, and numerals. Two datasets, one for extrinsic evaluation and the other for intrinsic evaluation of the word embedding were utilized. For the former, the format was prepared in a way to suit the library we used, which is fastText.

The resulting embedding was evaluated for quality and speed and their ability to capture meaningful representations using evaluation techniques. Two evaluation techniques were utilized: intrinsic and extrinsic. In the intrinsic evaluation, the question of how well the vectors in the embedding capture linguistic relationships between words. The linguistic relationships under consideration were word similarity, word analogy, OOV word and odd-word out. In this method, we saw that words that are similar or analogous to each other happen together or closer in the space. Related Amharic words are found contiguous to each other in the vector space. The analogy relationships we found was quite congruent to the works of other researchers on other languages. The word embedding has automatically learned the vector representation, "ንጉሥ - ወንድ + ሴት", resulting in a vector closer to the word "ንግሥት". It is also shown that words which were not part of the training or were rare or misspellings were entertained in the vector representation. The extrinsic evaluation method mainly focuses on the ability of word embeddings to contribute in the downstream tasks. For this case, we used multiclass text classification. Experimental results vary as hyper-parameters are tuned in various sizes and amounts. The precision, recall and F1-score

measures are also shown fluctuating based on parameters. However, as per the testing done on various ample ratios and parameters, the word embedding can attain 97.8% F1-score in text classification.

5.2 Recommendation

It's known that word embeddings can be used in various tasks. Sentiment Analysis, NER, co-reference resolution, semantic-role labeling (SRL), and other NLP tasks can utilize word embeddings as a tool to tackle speed and efficiency issues. Since this work only focuses on exploring how a word embedding in Amharic behaves in sample NLP tasks such as text classification, we recommend that other NLP tasks be thoroughly studied using word embeddings. Two challenges in this perspective are the scarcity of training data and the morphological richness of Amharic.

Therefore, researchers who have the courage to prepare a huge Amharic dataset can investigate morphological effect of Amharic on word embeddings and the role of Amharic word embeddings in various NLP tasks.

This work can be used as a starting point for NLP works related with word embedding in Amharic language.

5.3 Future Works

There are issues that are highly linked to this work that needs to be addressed and studied in the future. Since word embeddings have multiple usages, in the future exploring the ways these embeddings can be used in various tasks is one area to listed in the "what to do in the future" list.

Generally speaking, the following works are planned in the future:

- The impact of word embeddings on bias and stereotype
- The role of word embeddings in word sense disambiguation, NER, POS tagging and SRL.
- Utilizing a big and rich corpus to experiment with Amharic Word embeddings.

- Studying character level word embedding and bilingual word embeddings with Arabic or other Semitic languages.

REFERENCES

Al-Rfou, R., Perozzi, B., & Skiena, S. (2013). *Polyglot: Distributed Word Representations for Multilingual NLP*. 10.

Armbruster. (1908). *Initia Amharica, an introduction to spoken Amharic*. 432.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *ArXiv:1409.0473 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1409.0473

Bakarov, A. (2018). A Survey of Word Embeddings Evaluation Methods. *ArXiv:1801.09536 [Cs]*. Retrieved from http://arxiv.org/abs/1801.09536

Basirat, A. (2018). *Principal Word Vectors*. Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-353866

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). *A Neural Probabilistic Language Model*. 19.

Bernstein, H., Gelatt, J., Hanson, D., & Monson, W. (2014). *Ten Years of Language Access in Washington, DC*. 43.

Bhattacharjee, J. (2018). *fastText quick start guide get started with Facebook's library for text representation and classification*. Retrieved from http://proxy2.hec.ca/login?url=http://proquestcombo.safaribooksonline.com/?uiCode=hecmontreal&xmlId=9781789130997

Blei, D. M. (2003). *Latent Dirichlet Allocation*. 30.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146. https://doi.org/10.1162/tacl_a_00051

Christodouloupoulos, C., & Steedman, M. (2015). A massively parallel corpus: the Bible in 100 languages. *Language Resources and Evaluation*, *49*(2), 375–395. https://doi.org/10.1007/s10579-014-9287-y

Collobert, R., & Weston, J. (2008). *A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning*. 8.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (Almost) from Scratch. *NATURAL LANGUAGE PROCESSING*, 45.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Demissie, D. (2017). *Addis Ababa Institute of Technology School of Electrical and Computer Engineering*. 74.

Devlin, J., Cheng, H., Fang, H., Gupta, S., Deng, L., He, X., … Mitchell, M. (2015). Language Models for Image Captioning: The Quirks and What Works. *ArXiv:1505.01809 [Cs]*. Retrieved from http://arxiv.org/abs/1505.01809

Elman, J. L. (1990). *Finding Structure in Time*. 16.

Elrazzaz, M., Elbassuoni, S., Shaban, K., & Helwe, C. (2017). Methodical Evaluation of Arabic Word Embeddings. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 454–458. Retrieved from http://aclweb.org/anthology/P17-2072

Eyassu, S., & Gambäck, B. (2005). Classifying Amharic news text using self-organizing maps. *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages - Semitic '05*, 71. https://doi.org/10.3115/1621787.1621801

Farzindar, A., & Inkpen, D. (2015). Natural Language Processing for Social Media. *Synthesis Lectures on Human Language Technologies*, *8*(2), 1–166. https://doi.org/10.2200/S00659ED1V01Y201508HLT030

Feng, J., Xu, H., & Yan, S. (2012). *Robust PCA in High-dimension: A Deterministic Approach*. 8.

Firth, J. R. (1935). THE TECHNIQUE OF SEMANTICS. *Transactions of the Philological Society*, *34*(1), 36–73. https://doi.org/10.1111/j.1467-968X.1935.tb01254.x

Fu, G. (2009). Chinese Named Entity Recognition Using a Morpheme-Based Chunking Tagger. *2009 International Conference on Asian Language Processing*, 289–292. https://doi.org/10.1109/IALP.2009.68

Gambäck, B., Olsson, F., Alemu Argaw, A., & Asker, L. (2009). Methods for Amharic Part-of-Speech Tagging. *Proceedings of the First Workshop on Language Technologies for African Languages*, 104–111. Retrieved from http://www.aclweb.org/anthology/W09-0715

Gambäck, B., Sahlgren, M., Alemu, Atelach, & Asker, Lars. (2014). *Applying Machine Learning to Amharic Text Classification*.

Graves, A., Mohamed, A., & Hinton, G. (2013). Speech Recognition with Deep Recurrent Neural Networks. *ArXiv:1303.5778 [Cs]*. Retrieved from http://arxiv.org/abs/1303.5778

Harris, Z. S. (1954). Distributional Structure. *WORD*, *10*(2–3), 146–162. https://doi.org/10.1080/00437956.1954.11659520

Hassan, A., & Mahmood, A. (2017). Efficient Deep Learning Model for Text Classification Based on Recurrent and Convolutional Layers. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1108–1113. https://doi.org/10.1109/ICMLA.2017.00009

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of Tricks for Efficient Text Classification. *ArXiv:1607.01759 [Cs]*. Retrieved from http://arxiv.org/abs/1607.01759

Kelemework, W. (2009). *AUTOMATIC AMHARIC TEXT NEWS CLASSIFICATION: A NEURAL NETWORKS APPROACH*. 139.

Khashman, A., & Dimililer, K. (2008). *Image Compression using Neural Networks and Haar Wavelet*. *4*(5), 10.

Kibriya, A. M., Frank, E., Pfahringer, B., & Holmes, G. (2004). Multinomial Naive Bayes for Text Categorization Revisited. In G. I. Webb & X. Yu (Eds.), *AI 2004: Advances*

in Artificial Intelligence (Vol. 3339, pp. 488–499). https://doi.org/10.1007/978-3-540-30549-1_43

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746–1751. Retrieved from http://www.aclweb.org/anthology/D14-1181

Kirsal Ever, Y., & Dimililer, K. (2018). The effectiveness of a new classification system in higher education as a new e-learning tool. *Quality & Quantity*, *52*(S1), 573–582. https://doi.org/10.1007/s11135-017-0636-y

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*(1), 59–69. https://doi.org/10.1007/BF00337288

Krabben, K. (2010). *Machine Learning vs. Knowlegde Engineering in Classification of Sentences in Dutch Law*. 23.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–1105). Retrieved from http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Lund, K., & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, *28*(2), 203–208. https://doi.org/10.3758/BF03204766

Mandelbaum, A., & Shalev, A. (2016). Word Embeddings and Their Use In Sentence Classification Tasks. *ArXiv:1610.08229 [Cs]*. Retrieved from http://arxiv.org/abs/1610.08229

Meyer, R. (2006). Amharic as lingua franca in Ethiopia. *Lissan: Journal of African Languages and Linguistics 20,1/2: 117-131*. Retrieved from https://www.academia.edu/5514187/Amharic_as_lingua_franca_in_Ethiopia

Miikkulainen, R. (2010). Simple Recurrent Network. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 906–906). https://doi.org/10.1007/978-0-387-30164-8_762

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *ArXiv:1301.3781 [Cs]*. Retrieved from http://arxiv.org/abs/1301.3781

Mikolov, T., Karafiat, M., Burget, L., Cernocky, J., & Khudanpur, S. (2010). *Recurrent Neural Network Based Language Model*. 4.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *ArXiv:1310.4546 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1310.4546

Mikolov, T., Yih, W., & Zweig, G. (2013). *Linguistic Regularities in Continuous Space Word Representations*. 6.

Mykowiecka, A., Marciniak, M., & Rychlik, P. (2017). Testing word embeddings for Polish. *Cognitive Studies | Études Cognitives*, (17). https://doi.org/10.11649/cs.1468

Narayanan, V., Arora, I., & Bhatia, A. (2013). Fast and accurate sentiment classification using an enhanced Naive Bayes model. *ArXiv:1305.6143 [Cs]*, *8206*, 194–201. https://doi.org/10.1007/978-3-642-41278-3_24

Negga, W. (2000). *Wa zé ma*. Croydon: W. Negga.

Osgood, C. E. (1964). Semantic Differential Technique in the Comparative Study of Cultures. *American Anthropologist*, *66*(3), 171–200. Retrieved from JSTOR.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, 79–86. https://doi.org/10.3115/1118693.1118704

Pawar, P. Y., & Gawande, S. H. (2012). A Comparative Study on Different Types of Approaches to Text Categorization. *International Journal of Machine Learning and Computing*, 423–426. https://doi.org/10.7763/IJMLC.2012.V2.158

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. Retrieved from http://www.aclweb.org/anthology/D14-1162

Pritchard, J. K., Stephens, M., & Donnelly, P. (2000). *Inference of Population Structure Using Multilocus Genotype Data*. 15.

Ritter, H., & Kohonen, T. (1989). Self-organizing semantic maps. *Biological Cybernetics*, *61*(4), 241–254. https://doi.org/10.1007/BF00203171

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1–47. https://doi.org/10.1145/505282.505283

Sienc̆nik, S. K. (2015). *Adapting word2vec to Named Entity Recognition*. 5.

Soliman, A. B., Eissa, K., & El-Beltagy, S. R. (2017). AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP. *Procedia Computer Science*, *117*, 256–265. https://doi.org/10.1016/j.procs.2017.10.117

Svoboda, L., & Beliga, S. (2018). *Evaluation of Croatian Word Embeddings*. 7.

Tedla, Y., & Yamamoto, K. (2017). Analyzing word embeddings and improving POS tagger of tigrinya. *2017 International Conference on Asian Language Processing (IALP)*, 115–118. https://doi.org/10.1109/IALP.2017.8300559

Tegegnie, A. K. (2010). *HIERARCHICAL AMHARIC NEWS TEXT CLASSIFICATION HIERARCHICAL AMHARIC NEWS TEXT CLAS*. 119.

Tripodi, R., & Pira, S. L. (2017). Analysis of Italian Word Embeddings. *ArXiv:1707.08783 [Cs]*. Retrieved from http://arxiv.org/abs/1707.08783

Turian, J., Bengio, Y., Ratinov, L., & Roth, D. (2010). *A preliminary evaluation of word representations for named-entity recognition*. 8.

Turney, P. D. (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. *ArXiv:Cs/0212032*. Retrieved from http://arxiv.org/abs/cs/0212032

Vasic, D., & Brajkovic, E. (2018). Development and Evaluation of Word Embeddings for Morphologically Rich Languages. *2018 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 1–5. https://doi.org/10.23919/SOFTCOM.2018.8555822

Vo, D. T., & Zhang, Y. (2016). Don't Count, Predict! An Automatic Approach to Learning Sentiment Lexicons for Short Text. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 219–224. Retrieved from http://anthology.aclweb.org/P16-2036

Weldesellassie. (2003). *Automatic Categorization of Amharic news text: a machine learning approach*. Addis Ababa University, Addis Ababa, Ethopia.

Weninger. (2011). Stefan Weninger et al. (eds.): THE SEMITIC LANGUAGES: An International Handbook | Stefan Weninger - Academia.edu. Retrieved March 3, 2019, from https://www.academia.edu/6917762/Stefan_Weninger_et_al._eds._THE_SEMITIC_LANGUAGES_An_International_Handbook

Woodard, R. D. (Ed.). (2008). *The ancient languages of Mesopotamia, Egypt and Aksum*. Cambridge ; New York: Cambridge University Press.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level Convolutional Networks for Text Classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28* (pp. 649–657).

Retrieved from http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf

Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. *ArXiv:1510.03820 [Cs]*. Retrieved from http://arxiv.org/abs/1510.03820

APPENDIX 1

AMHARIC PUNCTUATION MARKS AND BASIC ETHIOPIC NUMBERS

Arabic Number	Ethiopic Number	Amharic Punctuation Marks and their description
1	፩	. (Period)
2	፪	: (Word space)
3	፫	… (Ellipsis)
4	፬	። (Full stop)
5	፭	፣ or ፡ (Comma)
6	፮	፤ (Semi-colon)
7	፯	፦ (Preface colon)
8	፰	፧ (Question mark)
9	፱	※ (Section mark)