Addis Ababa University

College of Natural Sciences

Amharic Textual Entailment Classification Model Using Deep Neural Network

Helina Mesfin

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

Addis Ababa, Ethiopia
December, 2020

Addis Ababa University
College of Natural Sciences

Helina Mesfin

Advisor: *Fekade Getahun (PhD)*

This is to certify that the thesis prepared by *Helina Mesfin*, titled: *Amharic textual entailment classifier model using Deep Neural Network* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

	Name	Signature	Date
Advisor:	**Dr. Fekade Getahun**	_____	_____
Examiner:	**Dr. Mulugta Libse**	_____	_____
Examiner:	**Dr. Yaregal Assabie**	_____	_____

Abstract

Textual entailment can provide a semantic inference to natural language expression tasks that have meaning variability and it solves language ambiguity problems. Textual entailment classification task is to predict the given pairs of natural language expressions such that a human who reads the first element of a pair would most likely infer that the other element is entailment, neutral, or contradiction.

In this work, we developed a textual entailment classification by using deep learning approach. Amharic is one of the most under resourced language, and has challenges in its linguistic levels. We developed an end to end solution as the approaches using deep Neural networks that can eliminate the underlying extensive feature engineering that shown traditional approaches for the development of Amharic Textual entailment classifier.

The reverse side sentence matching model for Amharic textual entailment classification has five main layers. Which are word embedding, sentence embedding, and sentence matching, aggregation and prediction layers. Specifically, for the first word embedding layer, applied word vectors by attaching its sub word information that can represent the important features of Amharic words. Second In sentence embedding layer by leveraging bi-directional long short term memory network we are able to remember long sentence sequences of the dataset we prepared (pair of 8700 sentences) for Amharic Inference task. Third in matching layer we matched each time step of the hypothesis sentence against all time step of the premise by applying matching functions followed by a matching approaches, Our model produced encouraging result in contrast to the base line models by scoring 77% training and 72% of test accuracy,

Keywords: Sub word embedding, word vectors, skip gram, fasttext, Sentence embedding, Sentence Matching, mean and maxpooling

Dedication

To one of the nicest individual in the world Beza Negash (bezi) RIP

Acknowledgments

First and for most Dr. Fekade Getahun, I wanted thank you for being my advisor. Thank you for pointing out strengths I didn't even know I had, thank you for unlocking opportunities for me. Each time we meet you boost my confidence. I reached out to you because I heard and saw absolutely tremendous things about you in hopes of you becoming my advisor. You could have easily said no because saying yes meant adding another thing to your very busy schedule. But, you said yes and I'm forever grateful. Without you, I would not be able to go an inch of this thesis. Having you backing me up 100% allows me peacetime. Lastly, thank you for being such a wonderful person!

I would like to thank Samuel Bahiru very much, for your loyal and honest traits you have showed on tiresome task of dataset preparation. Even if you can do it carelessly you arrange a time in your maximum effort to provide undivided attention till last day of the dataset translation completed.

My next appreciation goes to Bethlehem Berhanu, in helping me evaluating and cleaning up the dataset line by line without boredom.

It will be a great lapse not to thank my parents (Enateye and Mesfine) for giving me time that I needed the most, while they craved spend it with me. Finally, I wanted to thank my little geek brother (Mike) for inspiring me to peruse the thesis with his technical thoughts.

Table of Contents

List of Tables

List of Figures

Acronyms

AL	Amharic Language
ANLI	Amharic Natural Language Inference
ANN	Artificial Neural Network
ATEC	Amharic Textual Entailment Classification
BiLSTM	Bidirectional Long short Memory
BMPM	Bilateral Multi perspective Matching
CAA	Compare and Aggregate
CBOW	Continuous Bag of words
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
GloVe	Global Vector
LSTM	Long Short Memory Network
ML	Matching Layer
NLI	Natural Language Inference
NLP	Natural Language Processing
NN	Neural Network
OOV	Out-of-vocabulary
PCA	Principal Component Analysis
POS	Part of speech
RNN	Recurrent Neural Network
RSSM	Reverse Side Sentence Matching
SE	Sentence Embedding
SEL	sentence Embedding Layer
SG	Skip Gram
SGM	Skip Gram Embedding
SNLI	Stanford Natural Language Inference
SVM	Support Vector Machine
SW	sub-word
SWE	Sub Word Embedding
TEC	Textual Entailment Classification
t-SNE	t-Distributed Stochastic Neighbor Embedding
WBWA	Word By Word Attention
WE	Word Embedding
WEL	Word Embedding Layer
LEA	Language Expert Answer

Chapter 1: Introduction

1.1 Background

Natural Language Processing (NLP) is one of the most important technologies of the information age. Application of NLP is practically pervasive, as the use of language in human to human, and human to machine interfacing is paramount. Natural language understanding must deal with meaning variability and language ambiguity problems. Text processing applications such as Information Retrieval, Information Extraction, and Text Summarization are the areas that require semantic inference for resolving variability and ambiguity problems.

Textual entailment can provide a semantic inference to natural language expression tasks that have meaning variability and it solves language ambiguity problems [1, 2] Textual entailment methods, at the same time, recognize, generate, or extract pairs of natural language expressions that go hand in hand, such that a human who reads the first element of a pair would most likely infer that the other element is also true.

The goal of textual entailment classification (TEC) is to detect whether the hypothesis can be inferred from the premises or not. TEC is categorized into three based on the relationship in between [3]. The first category is positive entailment that occurs within two sentences, premises to prove the hypothesis is true; the second category is negative entailment, is the inverse of positive entailment, the hypothesis inferred to disprove the premises, and when the two sentences have no correlation that is considered to be neutral.

Nowadays, as shown in several recent related works [4, 5, 6] deep learning in NLP outperformed the traditional way of implementing TEC. Traditional methods of NLP gives a tougher time for processing textual entailment classification. As the methods are predominance of classifiers [4] involving hand engineered features, they heavily rely on natural language processing pipelines and external resources. Even though there are attempt in the formal reasoning methods [7] for natural inference tasks, the approach is not widely used, because of its complexity and domain dependency. Deep learning approaches come up with a solution for most of NLP tasks ranging from capability representation learning to automatically learn good features by itself, and avoiding complexity shown at all linguistic levels [8].

The purpose of this thesis is to develop Amharic textual entailment classification using deep learning approaches. This can be developed with a single end to end model, and do not require traditional task specific feature engineering, which could exhibit a better performance and easy implementation than the traditional way using natural language expressions.

1.2 Motivation

For textual entailment come viable estimating semantic similarities between two sentences is crucial. Semantic analysis is about understanding the meaning of a sentence which relies on the relationship between words, sentences and phrases. Amharic language has challenges in its linguistic levels, and the most under resourced, which leads the process of implementing textual entailment classification to be complicated and underperformed.

Amharic words being rich in morphology has its downside in word representation, results in huge vocabularies, and leads to very sparse vectors. Amharic word is ambiguous and has different meanings expressed in one word (polysemy). Infliction shown in word morphemes can represented in phrase and sentence level, there is a need for morphological analyzer [9]; and to capture the structure and meaning of a sentence, there is a need for part of speech tagger, syntactic parser or dependency parser [10]. However, such tools do not exist for Amharic.

A need for an approach eliminates the underline language assumption mentioned above. We motivated by the Deep Neural network approach in which gives a solution by dealing with the complexity in variant behavior shown in all linguistic levels by designing its own features which enable us in developing end to end model [8].

1.3 Statement of the Problem

Currently there is one related work on recognizing textual entailment [11], which follow traditional machine learning approach and requires extensive analysis in every linguistic step and additional resources for feature extraction.

Traditional way of TE classification relied on extensive manual creation of features in every level of linguistic. Starting from linguistic features multiple levels of lexical pre-processing, syntactic parsing, semantic parsing, annotating semantic phenomena, and representation performed on bag of words, n-grams and logical representations are performed [1] . On knowledge source, various external resources, and specialized subcomponents, such as negation classification, syntactic mapping rules, lexical resources, semantic phenomena, and specific knowledge modules are needed. But these traditional way of implementing TEC is known to be complicated [4] and time consuming, especially with challenges Amharic language has.

Recently Deep learning in Natural language processing [8] showed a promising result for the feature designing. Thus, in this work deep neural network shall be used to gain improved semantic understanding of Amharic language inferences for the purpose of undertaking automatic TEC.

1.4 Objectives

1.4.1 General Objectives

The general objective of this work is to develop Amharic textual entailment classifier model using Deep Neural Network.

1.4.2 Specific Objectives

In order to achieve the mentioned general objective, the following specific objectives are formulated:

- ✓ Reviewing related works for semantic inference in Deep Neural network (DNN).
- ✓ Adopt word representation models to for representing Amharic words.
- ✓ Prepare a standardized sentence pair dataset for balanced language inference classification.
- ✓ Applying deep neural layers better at representing long sentence sequence meaning.

- ✓ Adopt matching function and approaches.
- ✓ Develop Amharic TEC model using deep neural network
- ✓ Develop a prototype TEC system,
- ✓ Evaluate the performance the system using test dataset.

1.5 Methods

To accomplish the general and specific objectives of this study, different methodologies will be employed, and majors are:

1.5.1 Literature review

Literature review is the key methodology use to investigate and find out the state of art in related works and identify gaps. In this methodology variety of TEC approaches and algorithms will be analyzed. Especially with deep Neural TEC approaches, and comparison with traditional ways, preprocessing, and vector representation for foreign language, methods to follow of textual entailment in which neural network layers can optimize the meaning sentence will be reviewed. Reviewing and criticizing related works help to gain knowledge on the area and also skill in the choice and use of the appropriate algorithms, and tools.

1.5.2 Data collection

In this work, data collection will be a significant task of all, it influences the performance of accurate results, as deep learning works well with larger corpus. A text Amharic corpora data will be collected and categorized as training and test datasets. The first one is the unstructured Amharic texts, which used for representing a meaning of word, and the other will be manually prepared Amharic language expression sentence pairs labeled for balanced classification, used as dataset applied in training the network layers for learned representation, and garbing vectorization of sentences inferences and for later classification.

1.5.3 Tools

Python an open source language and specifically TensorFlow, keras, genism in python will be used. TensorFlow is a scalable and multiplatform programming interface for implementing and running machine learning algorithms, including convenience packages in deep learning.

1.6 Scope and Limitation of the Study

The scope of this work is limited to textual entailment classification. The application of TE for other purpose is out of scope, only sentence level entailment within the three class labels (entailment, contradiction, and neutral) will be applied, the general architecture to be developed is generic but the testing and all the experiments will be restricted to Amharic language. The main source of data will be Amharic text only and hence image structure text will not be considered as input.

1.7 Applications of Results

Textual Entailment is a core Natural Language Understanding task (NLU). While it poses as a classification task, it is uniquely well-positioned to serve as a benchmark task for research on Amharic NLU as question answering, information extraction, summarization, multi-document summarization, and evaluation of machine translation systems, need to recognize that a particular target meaning can be inferred from different text variants.

1.8 Organization of the Rest of the Thesis

The remaining part of the thesis is organized as follows. Chapter 2 is basis to this work, it explains the general approaches in DNN which are foundational in representation of words and sentences, comparisons between sentences, and challenges of Amharic language insinuated. In Chapter 3, related and the state of art work to this thesis are discussed and examined. Chapter 4, is the core of the research, it explains in details our approach for Amharic TEC. Chapter 5 explains the experiments and the archived results. Finally, Chapter 6 provides conclusion and our future research direction.

Chapter 2: Literature Review

2.1 Overview

In this Chapter, a vast amount of reviews are conducted, in the general concepts and components (layers) of textual entailment classification based in deep learning approaches. Methods discussed are from capturing the meaning of word entity, until matching the important units between sentence sequences, as these are key elements to solve problems of TEC. The review is state of the art to decide on the approach that shall be adopted in each component. The Chapter is divided into the sub sections of word representation, sentence embedding, sentence matching and language specific analysis or element section along with Amharic.

2.2 Word Representation

Word representation is embedding each word into a high dimensional vector space, and these vectors of numbers that encodes the meaning of a word. Word embeddings showed great success in many downstream NLP tasks in recent years., they are close to fully replacing more traditional distributional representations such as LSA [12] features and Brown clusters [13].

Vector space models have been used in distributional semantics since the 1990s. Since then, we have seen the development of a number of models used for estimating continuous representations (count based) of words, Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA) can be taken as examples.

The term word embeddings originally developed in [14] trained in a neural language model together with the model's parameters. However, [15] were arguably the first to demonstrate the power of pre-trained word embeddings. A unified architecture for natural language processing, in which [15] establish word embeddings as a highly effective tool when used in downstream tasks, while also announcing a neural network architecture that many of today's approaches were built upon. It was [16], brought word embedding to the creation of word2vec, a year later, introduced GloVe [17], a competitive set of pre-trained embeddings, suggesting that word embeddings is among the mainstream in NLP tasks.

In the next sub sections, we will discuss word representation approaches in two, traditional distributional semantics and the modern neural network approaches of word embeddings.

2.3 Distributional Semantics (Count based Method)

Latent Semantic Analysis (LSA) [18] is a method, in distributional semantics, used to analyze relationships between documents and the words inside them by making a set of concepts related to the documents and words. This method 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 an approach called Singular Value Decomposition (SVD) to reduction of the number of rows in the matrix while making the linguistic features complete. 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.

LSA distributional semantic method faces basically two problems [18]. First, there is no clear way to count each word, whether using frequency method or based on the word presence. In this first point, if frequency of words is considered, in most of the corpus, the least important words like stop words, punctuation marks etc. are the most frequent ones. Second 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. This poses another problem.

2.3.1 Word Embedding Models

Every feed-forward neural network takes words from a vocabulary as input and embeds them as vectors into a lower dimensional space, which it then fine-tunes through back-propagation, necessarily yields word embeddings as the weights of the first layer. It is usually referred to as word Embedding Layer. In the next sections we will explain different models for the implementation of word embeddings.

a) Neural probabilistic language model

The classic neural language as [14] consists of a one-hidden layer feed-forward neural network that predicts the next word in a sequence as shown in Figure 2-1.

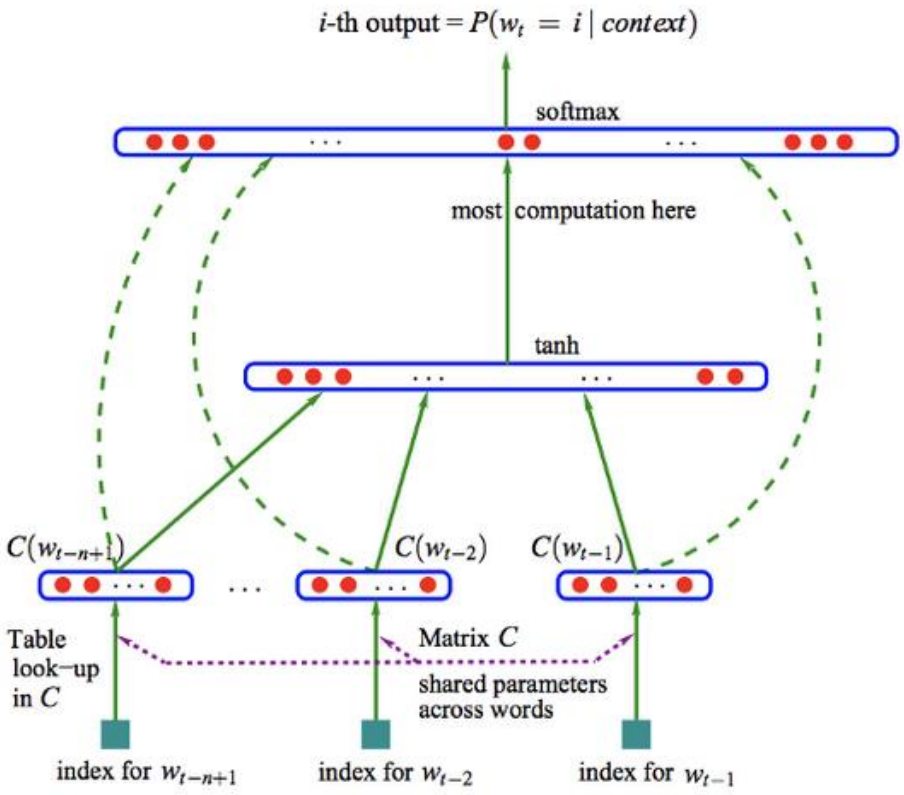

Figure 2-1: *Classic neural language model*

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log f(w_t, w_{t-1}, \dots w_t - n + 1) \quad (1)$$

From eq. (1) the model maximizes objective functions of J_θ the output of the model, i.e. the probability $p(w_t|w_{t-1}, \dots w_t - n + 1)$ calculated by the softmax, where n is the number of previous words fed into the model.

A real-valued word feature vector in R. The foundations of this can still be found in today's neural language and word embedding models [14]. These are the following:

- **Embedding Layer**: This layer generates word embedding's by multiplying an index vector with a word embedding matrix.

- **Intermediate Layer(s):** One or more layers that produce an intermediate representation of the input, e.g. a fully-connected layer that applies a non-linearity to the concatenation of word embeddings of n previous words.
- **Softmax Layer**: The final layer that produces a probability distribution over words in V.

In this approach two issues determined [14]. The first is that Layer two or intermediate layer is replaceable with an LSTM. Also identified the final SoftMax as the network's main bottleneck, as the cost of computing the softmax is proportional to the number of words in V, which is typically on the order of hundreds of thousands or millions.

Discovering methods that ease the computational cost related to computing the softmax over a large vocabulary is therefore one of the main challenges in both neural language and word embedding approaches.

b) A Unified Architecture for NLP.

In order to avoid computing the expensive Softmax [15] employ an alternative objective function, which maximizes the probability of the next word given the previous words. The architecture trains a network to output a higher score $\boldsymbol{f_\theta}$ for a correct word sequence (a probable word sequence in [14] model) than for an incorrect one. For this purpose, the architecture uses a pairwise ranking criterion, shown in eq. (2):

$$\boldsymbol{J_\theta = \sum_{x \in X} \sum_{w \in V} \max\{0, 1 - f_\theta(x) + f_\theta(x^{(w)})\}} \quad (2)$$

The sample windows x containing n words from the set of all possible windows x in the corpus. In eq. (2) for each window x, then produce a corrupted, incorrect version $\boldsymbol{x^{(w)}}$ by replacing x's center word with another word w from vocabulary. The objective j maximizes the distance between the scores output by the model for the correct, and the incorrect window with a margin of 1.

The resulting language model produces embeddings that already possess many of the relations word embeddings have become known for example countries are clustered close together, and syntactically similar words occupy similar locations in the vector space.

So, the benefits are, high quality embeddings can be learned efficiently in [15], especially when comparing against neural probabilistic models [14]. That means low space and low time

complexity to generate a rich representation. Additionally, the larger the dimensionality, the more features can have in representation. And still, can keep the dimensionality a lot lower than some other methods as bag of words.

c) Word2vec

Word2Vec is the most popular of the word embedding models [19]. There are two architectures which are computationally less expensive by abandoning the costly hidden layer and allowing the language model to take additional context into account.

- **Continuous bag-of-words (CBOW**): contrasting to language models [14, 15] that can only base its predictions on previous words. CBOW method Use both the n words before and after the target word (wt) to predict the next word as shown in Figure 2-3 [19]. This is known as a continuous bag of words (CBOW), owing to the fact that it uses continuous representations.

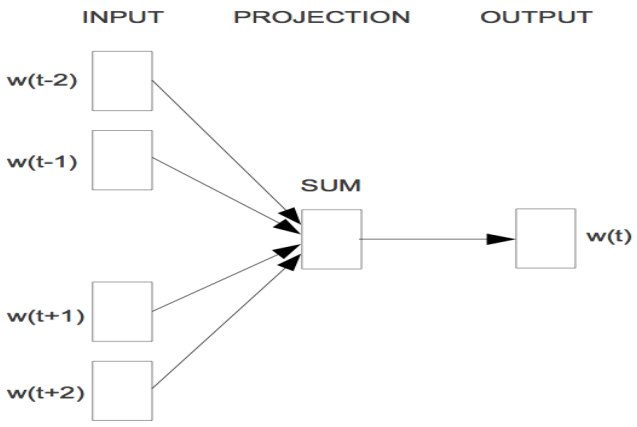

Figure 2-2: *Continuous bag-of-words architecture*

- **Skip-gram:** While CBOW can be seen as a precognitive language model, skip-gram approach does rather than using the surrounding words to predict the center word as with CBOW, skip-gram uses the Centre word to predict the surrounding words as can be seen in Figure 2-4 [19].

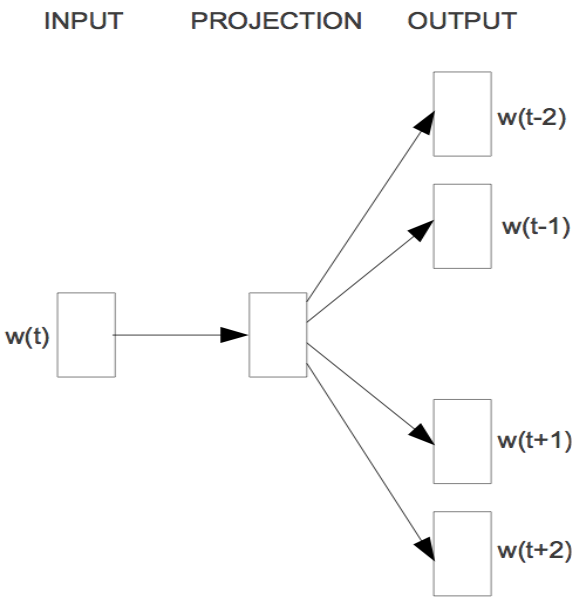

Figure 2-3: *Skip-gram Architecture*

The skip-gram objective thus sums the log probabilities of the surrounding n words to the left and to the right of the target word w_t to produce the following objective:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T}\sum_{-n\leq j\leq n,\neq 0} \log p(w_{t+j}|w_t) \qquad (3)$$

Instead of computing the probability of the target word wt given its previous words, eq. (3) calculates the probability of the surrounding word w (t+j) given wt.

d) Glove (Global log bilinear regression method)

This method combines the advantages of the two major model families, count based methods and local context window methods. Global matrix factorization [17] is the process of using matrix factorization methods from linear algebra to reduce large term frequency matrices. These matrices usually represent the occurrence or absence of words in a document.

The glove model efficiently leverages statistical information by training only on the nonzero elements in a word-word co-occurrence matrix, rather than on the entire sparse matrix or on individual context windows in a large corpus. The model produces a vector space with meaningful

substructure. The goal of Glove is to enforce the word vectors to capture sub-linear relationships in the vector space [17]. Thus, it proves to perform better than Word2vec in the word analogy tasks. It adds some more practical meaning into word vectors by considering the relationships between word pair and word pair rather than word. Glove gives lower weight for highly frequent word pairs to prevent the meaningless stop words like "the", "an" will not dominate the training progress. The appropriate starting point for word vector learning should be with ratios of co-occurrence probabilities rather than the probabilities themselves.

The weighted least squares objective J as mentioned in eq. (4) That directly aims to reduce the difference between the dot product of the vectors of two words and the logarithm of their number of co-occurrences:

$$J = \sum_{i\ j=1}^{v} f(X_{ij})(w_i^T w_j + b_i + b_j - logX_{ij})^2 \quad (4)$$

where w_i and b_i are the word vector and bias respectively of word i, w_j and b_j in eq. (4) are the context word vector and bias respectively of word j, X_{ij} is the number of times word i occurs in the context of word j, and f is a weighting function that assigns relatively lower weight to rare and frequent co-occurrences. As co-occurrence counts can be directly encoded in a word-context co-occurrence matrix, GloVe takes such a matrix rather than the entire corpus as input.

Fast training, scalable to huge corpora, good performance can be gained by even with small corpus, and small vectors. The disadvantage of this model is when trained on the co-occurrence matrix of words, which takes a lot of memory for storage. Especially, if change the fine-tuning parameters related to the co-occurrence matrix, a need to reconstruct the matrix again, which is very time consuming.

e) Sub- word level embedding (FastText)

The three main problems of using the previous word embeddings methods are: 1) inability to deal with out-of-vocabulary (OOV) words, i.e. words that have not been seen during training. Typically, such words are set to the unknown token and are assigned to no vector, which is an ineffective choice if the number of OOV words is large. 2) For morphologically rich languages finding embedding is hard because they are rarely distributed, 3) the use of minimal dataset couldn't produce better embedding results.

Sub word embedding [20] is extension of word2vec model (skip gram) that represents each word as an n-gram of characters and the word itself. So, for example, the representation of the word, "artificial" with n=3, are <ar, art, rti, tif, ifi, fic, ici, ial, al, artificial >, where the angular brackets indicate the beginning and end of the word.

This helps capturing the meaning of shorter words and allows the embeddings to understand suffixes and prefixes. Once the word has been represented using character n-grams, a skip-gram model is trained to learn the embeddings. This model is considered to be a bag of words model with a sliding window over a word as no internal structure of the word is taken into account

The following are the advantage of using sub word embedding over word2vec and GloVe [17, 16].

- Generate better word embeddings for morphologically inflicted or rare words (even if words are rare their character n-grams are still shared with other words – hence the embeddings can still be good). This is because, in word2vec a rare word (e.g. 10 occurrences) has fewer neighbors to be pulled by, in comparison to a word that occurs 100 times – the latter has more neighbor context words and hence is pulled more often resulting in better word vectors.
- Out of vocabulary words – it can construct the vector for a word from its character n-grams even if a word doesn't appear in training corpus.
- Can be produced better embedding results with small size of corpus.

In conclusion of word embedding layer within the advantages seen in sub word embedding over the other models, we choose to use this model, since it indicated from above, ways which leads to overcome challenges Amharic language structure has, as mentioned in section 4.2.

2.4 Sentence Embedding Methods

In deep neural network vectors (numerical representation) play an important role, as every word is transforming to these vectors which encodes its semantic meaning, then these word vectors transformed into sentence vectors by concatenating the component of each words that's found in the sentence. In this section, sentence embedding methods that transforms input words vector to sentence representation will be presented categorized on the bases of the approach used Convolutional Network based Recurrent Neural Network, LSTMs and GRUs.

2.4.1 Convolutional neural network (CNN)

CNNs are a specialized kind of neural network for processing data that has a known, grid-like topology. In a traditional feed forward neural network each input neuron is connected to each output neuron in the next layer and forms a fully connected layer [21]. In CNNs the network employs a mathematical operation called convolution region of the input and is connected to a neuron in the output.

CNN architecture for text processing is shown in Figure 2-5 [22]. It starts with an input sentence divided into word chunks or word embeddings low-dimensional representations generated by models like word2vec or GloVe. Words chunked into features and are fed into a convolutional layer. The results of the convolution are "pooled" or aggregated to a representative number. This number is fed to a fully connected neural structure, which makes a classification decision based on the weights assigned to each feature within the text.

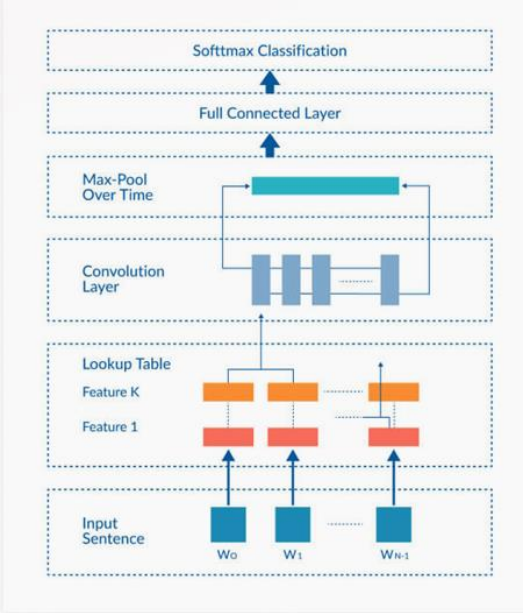

Figure 2-5: *CNN layers for text Classification*

a) Convolutional process on text vectors

In a CNN, text is organized into a matrix, with each row representing a word embedding, a word, or a character. The CNN's convolutional layer "scans" the text, breaks it down into features, and judges whether each feature matches the relevant label or not.

Figure 2-6 [23] Illustrates how the convolutional "filter" slides over a sentence, two words at a time. It computes an element-wise product of the weights of each word, multiplied by the weights assigned to the convolutional filter.

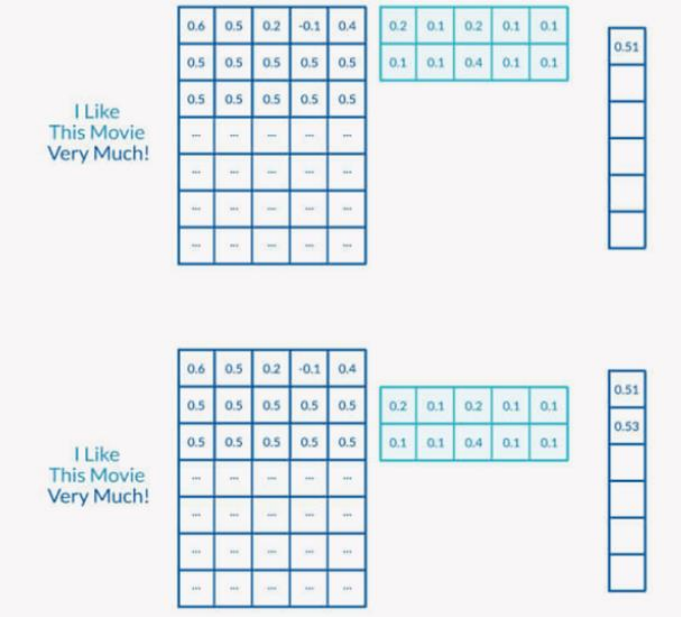

Figure 2-6: *Convolutional process on text vectors*

The sum of the products is taken as a representation of the current textual feature – 0.51 and 0.53 in the example of the Figure 2-6. This is the "pooling" stage, reducing the dimensionality of the word features and retaining only a simple probability score that reflects how likely they are to match a label.

At the final stage, these scores are the inputs to a fully connected neural layer. The "fully connected" part of the CNN network goes through its own backpropagation process, to determine the most accurate weights. Each neuron receives weights that prioritize the most appropriate label for example, "positive sentiment" or "negative sentiment". Finally, the neurons "vote" on each of the labels, and the winner of that vote is the classification decision.

A main advantage for CNNs is that they are fast in computation time. Convolutions are a central part of computer graphics and implemented on a hardware level on GPUs. CNNs are also efficient in terms of representation. With a large vocabulary, computing anything more than 3-grams can quickly become expensive. Convolutional Filters learn good representations automatically, without needing to represent the whole vocabulary.

Location Invariance and local Compositionality made intuitive sense for images, but not so much for NLP. CNN Do not care a lot where in the sentence a word appears (word order). In CNN Pixels close to each other are likely to be semantically related (part of the same object), but the same isn't always true for sentences [24]. In a sentence, parts of phrases could be separated by several other words.

The compositional aspect of sentences isn't obvious either. Clearly, words compose in some ways, like an adjective modifying a noun, but CNN needs further model optimization to be applied with higher level representations [24] on the bases of these facts, CNNs wouldn't be a good fit for NLP tasks.

2.4.2 Recurrent Neural Networks

The idea behind recurrent neural networks (RNNs) is to make use of sequential information. In a traditional neural network, all inputs (and outputs) are independent of each other RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Unlike a traditional neural network [25], which uses different parameters at each layer, and shares the same parameters across all steps. This greatly reduces the total number of parameters needed for learning. The main feature of an RNN is its hidden state, which captures some information about a sequence. RNNs have a memory which captures information about what has been calculated so far. But are limited to looking back only a few steps

RNN hidden state as the memory captures information about what happened in all the previous time steps. The output at step *t* is calculated solely based on the memory at time t, but the final output typically can't capture information from too many time steps ago faces vanishing gradient decent problem. With that regard RNN are not capable at representing sentence as sentences have

long term dependences. Depending on the task this standard RNN may not be necessary good in representation of sentence. For example, in tasks like textual entailment or text similarity, a need to capture input starting from the first step and capture inputs all the way through time steps.

2.4.3 Long Short-Term Memory Networks.

LSTM (Long Short-Term Memory networks) [26]. LSTMs are among the most widely used models in Deep Learning for NLP. LSTMs were designed to combat vanishing gradients in RNN through a gating mechanism. LSTMs don't have a fundamentally different architecture from RNNs but use a different function to compute the hidden state. Memory in LSTMs are called cells that take as input the previous state (t-1) and current input (x). Internally, these cells decide what to keep in (and what to erase from) memory. They then combine the previous state, the current memory, and the input. It turns out that these types of units are very efficient at capturing long-term dependencies.

LSTM [26] have three gates input forget and output gates. The input gate defines how much of the newly computed state for the current input you want to let through. The forget gate defines how much of the previous state to let through and to get rid of. Finally, the output gate defines how much of the internal state need to be exposed to the external network (hidden layers and the next time step).

All LSTM gates have the same dimensions which is, the size of the hidden state. Due to its ability to capture the long-term memory, the LSTM-RNN accumulates increasingly richer information as it goes through the sentence, and when it reaches the last word, the hidden layer of the network provides a semantic representation of the whole sentence.

2.4.4 Gated Recurrent Unit (GRU)

GRUs, first used in 2014, are a simpler variant of LSTMs [26] that share many of the same properties. To solve the vanishing gradient problem of a standard RNN, GRU uses, two gates, a reset gate, and update gate. Intuitively, the reset gate determines how to combine the new input with the previous memory, and the update gate defines how much of the previous memory to keep around. If we set the reset gate to all 1's and update gate to all 0's, we again arrive at our plain

RNN model. The basic idea of using a gating mechanism to learn long-term dependencies as in a LSTM, but there are a few key differences between the two architectures:

First GRUs don't possess internal memory and second, don't have the output gate that is present in LSTMs, and third, the input and forget gates are coupled by an update gate, and the reset gate to decide how much past information to get rid of and acts as the same as forget gate in LSTM. Which computed on previous hidden state.

According to comparative study of [27] GRUs train faster and perform better than LSTMs on less training data, and are simpler and thus easier to modify, for example adding new gates in case of additional input to the network. Uses less algorithms in general. LSTMs remember longer sequences than GRUs and outperform them in tasks requiring modeling long-distance relations. Since in textual entailment there involves two sentences embedding for making classification, we will be using LSTM family embedding BiLSTM for the sentence vector to keep remember long distance relationships between the sentence.

2.5 Sentence matching

Many natural language processing problems involve matching two or more sequences to make a decision. Textual entailment classification is one of the most complicated NLP task. One needs to determine whether a hypothesis sentence can be inferred from a premise sentence in three distinct classes. With recent advancements of deep neural network models in natural language processing, many downstream tasks as TE classification can achieve a good result if the quality and quantity of data is intact. The main approach followed is to encode a sequence of text as an embedding vector using models such as LSTM-RNN and CNN, and to match two sequences there exists three general approaches for matching textual entailments. In this section, we review these general architectures for matching sequences namely Siamese network, attentive network and compare-aggregate network.

2.5.1 Siamense Network

The Siamese neural network consists of two identical sub-networks that join together at their output levels [28]. During training, the two sub-networks extract semantic features from two disparate text chunks (e.g., documents), while the joining neurons measure the semantic distance

between the two hidden vectors of text chunks using the cosine similarity or other metrics. It has exhibited better results for several NLP-sequence similarity related tasks. .

In this framework, the same neural network encoder (e.g., a CNN or RNN) is applied to the two input sentences, so that each input is encoded into sentence vector independently using the same embedding space. Then, a matching decision is made solely based on the two sentence vectors [29] . The Siamese LSTM network aims to facilitate the measure of the semantic distance between the two different text documents, by learning distributed vector representations of the documents which reflect the semantic relatedness between any pair of documents. This model uses an LSTM to read in word-vectors representing each input sentence and employs its final hidden state as a vector representation for each sentence. Subsequently, the similarity between these representations is used as a predictor of semantic similarity.

The advantage of this framework is that sharing parameters makes the approach smaller and easier to train, and the sentence vectors can be used for visualization, sentence clustering and many other purposes [28] However, a disadvantage is that there is no explicit interaction between the two sentences during the encoding procedure, which may lose some important information. However, it has been found that using a single vector to encode an entire sequence is not sufficient to capture all important information. As a result, advanced techniques such as attention mechanisms and memory networks have been applied for sequence matching problems.

2.5.2 Attentive Network

As we mention in the above section the problem in Siamese neural network architecture it uses LSTM is that it encodes the input sequence to a fixed length internal representation. This imposes limits on the length of input sequences that can be reasonably learned and results in worse performance for very long input sequences. To overcome this limitation [30] attention mechanism is added to the LSTM architecture, Attention is the idea of freeing the LSTM cell architecture from the fixed-length internal representation. This is achieved by keeping the intermediate outputs from LSTM each step of the input sequence, and training the model to learn to pay selective attention to these inputs and relate them to items in the output sequence.

More precisely an LSTM with attention for TEC does not need to capture the whole semantics of the premise in its cell state [30]. Instead, it is sufficient to output vectors while reading the premise and accumulating a representation in the cell state that informs the second LSTM (which is the hypothesis) model use an attention mechanism, where the attention weights are computed between the last output of the premises LSTM and all the outputs of the hypothesis LTSM. Since sentences are padded with dummy words, a mask vector is computed for each hypothesis sentence to indicate which words truly belong to the sentence and thus can be used when computing attention weights.

2.5.3 Compare-Aggregate Network

In this framework, each sequence is represented using word representation format and word-level matching performed by using different comparison functions as cosine similarity, Euclidean distance and dot product followed by aggregation using Convolutional Neural Networks and LSTMS is done to make the final decision. This framework captures more interactive features between the two sentences; therefore, it can possess significant improvements.

In compare aggregate matching is done in basic two approaches to compare vector representations of smaller units such as words, or the entire representation of the sentence, then aggregate these comparison results to make the final decision.

First the match-LSTM for textual entailment compares each word in the hypothesis with an attention-weighted version of the premise [3]. The comparison results are then aggregated through an LSTM. Pairwise word interaction [31] that first takes each pair of words from two sequences and applies a comparison unit on the two words. It then combines the results of these word interactions using a similarity focus layer followed by a multi-layer CNN.

Second comparison of two sequences is done by comparing two vectors each representing an entire sequence. Proposed “compare-aggregate” [32] framework that performs word-level matching followed by matching result aggregation and feed into another Convolutional Neural Networks and finally make a classification. Decomposable attention [33] for textual entailment, in which words from each sequence are compared with an attention-weighted version of the other sequence to produce a series of comparison vectors. The comparison vectors are then aggregated and fed into a feed forward network for final classification.

2.5.4 Bilateral Multi-Perspective Matching.

Bilateral multi perspective matching, matches premises and hypothesis in two directions [34]. From the premises follow the hypothesis and from the hypothesis follow the premises. In each matching direction, each time step of one sentence is matched against all time steps of the other sentence from multiple perspectives the framework has three basic layers which described as follows.

- **Matching Layer**, the goal of this layer is to compare each contextual embedding (time-step) of one sentence against all contextual embeddings (time-steps) of the other sentence. They will match the two sentences P and Q in two directions: match each time-step of P against all time-steps of Q, and match each time-step of Q against all time-steps of P. To match one time-step of a sentence against all time-steps of the other sentence, used a multi-perspective matching operation and matching approaches.
- **Aggregation Layer.** This layer is employed to aggregate the two sequences of matching vectors into a fixed-length matching vector. By utilizing another BiLSTM model, and apply it to the two sequences of matching vectors individually. Then, construct the fixed-length matching vector by concatenating vectors from the last time-step of BiLSTM.
- **Prediction Layer**. The purpose of this layer is to evaluate the probability distribution $pr(y = j \mid P, Q)$to this end, here employed a two-layer feed-forward neural network to consume the fixed-length matching vector, and apply the softmax function in the output layer.

From the above Reviews on the sentence matching layer, we based our architecture on bilateral multi perspective [34]. With some change we applied and will be explained in chapter four. Selected this model because we found that from these approaches the task can highly benefit from the match performed from multiple perspective and direction. Also, Amharic writing system and sentence formation is different from languages as English (which is explained in the section 4.2). We can get the important feature of the sentences that contribute the most for the classification task if we match them in multi perspective.

2.6 Amharic language and its challenges

In this section, overview of the Amharic language and its challenge in natural language processing will be discussed in the following sections.

2.6.1 Overview of Amharic language

Amharic (Amharic: አማርኛ) is the working language of Federal Democratic Republic of Ethiopia and has its own alphabets. It is the second most spoken Semitic language in the world next to Arabic.

The alphabet of the Amharic language consists of 33 core symbols or Fidel (ፊደል). Each of these core symbols occur in seven different orders (basic character plus six different symbols or orders formed from the basic character). In total, there are 231 distinct symbols in the language [35] . Unlike English language, a symbol or character represents both a consonant and a vowel. The six orders are formed by attaching accent markings to the basic symbol to combine consonants with vowels.

Amharic is spoken by 21.8. Million native speakers according to [36] 4 million secondary speakers in Ethiopia. The majority of monolingual Amharic speakers are the Amhara people. Additionally, a great number of monolingual Amharic speakers live in bigger town and administrative centers all over the country [37]. As more Ethiopians are living outside their home town, the Amharic language speakers in the world are also growing. In Washington DC, Amharic has got the status to be one of the six non-English languages [38].

2.6.2 Amharic Sentence Structure

Amharic writing system is from left to right, the distinctive clause order noun + object + verb [39]. This order is different from the noun + verb+ object order of the English language [40]. Amharic has inflectional morphological structures, which requires a complex morpheme analyzer for morpheme generation and word formation as well.

In Amharic language, a word can be a sentence by itself implicitly combining object, subject and verb together [41]. For example, the word *ተሰናበታት* is a sentence; the subject is *"እሱ"*, the object is *"እሷ"* and the verb is *"ተሰናበተ"*. Therefore, identifying morphemes from a word is not easy.

Besides, the plural nouns plural verb formation in Amharic language is comparatively different from English language. For instance, *ሰጡት* is a word and in English is represented as a sentence having three grammatical components “they gave him”.

2.6.3 Challenges in Amharic Language

The main challenges in the Amharic language are categorized into three as:

- **Same word and Pronunciation with different character**. For example, the *ውሃ* and *ውኃ* can be written as the same as it pronounced similarly with one another not to mention words start with first order characters *ውሀ* and *ውኅ*. In word representation these can be treated as different vectors since it might not see these kinds of examples in the corpus, hence there should be a mechanism in handling out of vocabulary (OOV) words.
- **Diverse ways of writing the same word:** It is very common to see different people writing the same word in a different way. This spelling variation happens mostly when one writes the word with the closest or corresponding character without translation which is referred as transliteration. For example, the words Ethiopia can be written in two different ways as *ኢትዮጵያ* and *ኢትዮጲያ* and notice the characters at 5th position. When developing any Amharic word embedding these spelling variations should be handled.
- **Formation of Amharic words:** Amharic is one of the most highly inflected languages as one can generate several words from an Amharic term. For example: from the stem word of *ፍልግ* one can generate *ፈለገ፣ አሰፈለገ፣ፈለጎች፣ ለመፈለግ፣ በመፈለግ*, etc. This shows the vastness of the Amharic vocabulary. Amharic also has a lot of compound words formed by joining words with or without modification as *አንድላይ፣ቤተክርስቲያን፣ ህገመንግስት፣ ቤተመንግስት*, and etc. Most of NLP models learn such representations ignoring the morphology of words and compound words assigning a distinct vector to each word [20]. This is the main limitation, especially for languages with large vocabularies and many rare words as Amharic. Thus, there is a need to develop a concise word representation mechanism that is aware of word morphology.

Chapter 3: Related Work

In this Chapter, related researches regarding English textual entailment classifications are reviewed on the approaches based in deep learning. Meanwhile there is no work for Amharic entailment classification to the best of our knowledge, discussion have been performed in related works as Amharic text classification.

3.1 Deep learning Approaches

In Deep learning approaches within large and high-quality datasets end-to-end differentiable solution to TEC can be applicable, since it avoids specific assumptions about the underlying language. In particular, there is no need to manually engineer language features as part-of-speech tags or dependency parses. Furthermore, a generic sequence-to-sequence solution as LSTM and CNN allows to extend the concept of capturing entailment across any sequential data [42]. In the following sections we will assess deep neural network approaches for the development of TEC.

The first one who tested deep neural networks for natural language inference task are [43], developed a manually annotated 570K pairs of English sentences for learning NLI with class label, and tested it in straightforward architecture of deep neural networks on LSTM. These lexicalized classifiers outperform some sophisticated existing entailment models, and it allows a neural network-based model to perform competitively on natural language inference benchmarks for the first time. In the proposed architecture, the premise and the hypothesis are each represented by a dense fixed-length vectors in LSTM. The two vectors concatenation is subsequently used in a multi-layer perceptron (MLP) for classification. They got classification accuracy of 77.6%. The limitation of LSTM architecture is that it encodes the input sequence to a fixed length internal representation. This imposes limits on the length of input sequences that can be reasonably learned and results in worse performance for very long input sequences

Later that year [42], developed two models to addresses the difficulty that LSTM have [43] in retaining the meaning of words across long sentences. First by using LSTM with attention, it is possible to incorporate a weighted version of the word embeddings in the premise when computing hidden states. The attention model produces output vectors summarizing contextual information of the premise that is useful to attend over later when reading the hypothesis. Therefore, when reading the premise, the model does not have to build up a semantic representation of the whole

premise, but instead a representation that helps attending over the right output vectors when processing the hypothesis. In contrast, the output vectors of the premise are not used by the straightforward LSTMs. This model pushes the accuracy by 2.7% which is 80.9%.

Second another approach used by the same authors [42] adding a word by word attention of determining whether one sentence entails another it can be a good strategy to check for entailment or contradiction of individual word- and phrase-pairs. Here they do not use attention to generate words, but to obtain a sentence-pair encoding from fine-grained reasoning via soft-alignment of words and phrases in the premise and hypothesis. In their case, this amounts to attending over the first LSTM's output vectors of the premise while the second LSTM processes the hypothesis one word at a time, thus generating attention weight-vectors overall output vectors of the premise for every word x with x in the hypothesis. By using word by word attention, they pushed the accuracy to 83.5%.

A limitation of the mentioned two models of [42], is that they reduce both the premise and the hypothesis to a single embedding vector before matching them; i.e., in the end, they use two embedding vectors to perform sentence-level matching. However, not all word or phrase-level matching results are equally important. For example, the matching between stop words in the two sentences is not likely to contribute much to the final prediction. But, for hypothesis to contradict a premise, a single word or phrase-level mismatch (e.g., a mismatch of the subjects of the two sentences) may be sufficient [3], and other matching results are less important, but this intuition is hard to be captured if directly match two sentence embeddings

In to tackle the above limitation [3], proposed a layer called match -LSTM for NLI. The model builds on top of the neural attention model for NLI of [42]. Instead of deriving sentence embeddings cited in [43] to be used for classification, their model uses a match-LSTM to perform word by word matching of the hypothesis with the premise. This LSTM is able to place more emphasis on important word-level matching results. In particular, LSTM remembers important mismatches that are critical for predicting the contradiction or the neutral relationship label, and push the accuracy to 86.1%. A limitation of this model is on sequence matching problems is the use of a compare-aggregate matching approaches. In [3] comparison of two sequences is not done by comparing two vectors each representing an entire sequence. Instead, these models first

compare vector representations of smaller units such as words from these sequences and then aggregate these comparison results to make the final decision.

Many NLP tasks including machine comprehension, answer selection and text entailment require the comparison between sequences. Matching the important units between sequences is a key to solve these problems. Proposed a general "compare-aggregate" [44], approach that performs word-level matching followed by aggregation using Convolutional Neural Networks. They particularly focus on the different comparison functions, can use to match two vectors. They used four different datasets to evaluate the model on MovieQA, InsuranceQA, WikiQA and SNLI the first three are question answering, and the last on is textual entailment. More importantly, they systematically present and test six different comparison functions. They found that overall comparison function based on element-wise subtraction and multiplication works the best on the four datasets, and pushed the accuracy into 86.8%.

The limitation of the above models [44, 3] matching is only performed in a single direction (e.g., matching P(Premises) against Q (hypothesis)), but neglected the reverse direction (e.g., matching Q against P) to overcome the above limitation, proposes a bilateral multi-perspective matching (BiMPM) model [34], Natural language sentence matching fundamental technology for a variety of tasks. Given two sentences P and Q, the model first encodes them with a bidirectional long short-term memory (BiLSTM) encoder. Next, match the two encoded sentences in two directions P against Q and Q against P. In each matching direction, each time-step of one sentence is matched against all time-steps of the other sentence from multiple perspectives. Then, another BiLSTM layer is utilized to aggregate the matching results into a fixed-length matching vector. Finally, based on the matching vector, a decision is made through a fully connected layer. They evaluated their model on three tasks paraphrase identification, natural language inference and answer sentence selection. Experimental results on standard benchmark datasets show that their model achieves best performance on all tasks. Specifically, in entailment classification with SNLI dataset they push the accuracy to 88.8%.

3.2 Amharic Text Classification

As far as we know there is no work on Amharic entailment classification but there are some works on Amharic Text classification using different methodologies and approaches as presented hereafter.

Tigrinya, a very close Semitic language to Amharic, spoken in Ethiopia and Eritrea was explored for its word embeddings to improve POS tagger in [45]. They constructed a new text corpus and examined the optimal hyper parameters for generating word vectors. The authors indicated that the dimension of context affects the performance of semantic and syntactic relatedness of words. While the larger context window gives better semantic relatedness, shorter context extracts syntactic relatedness.

In [46] used Amharic word embedding analysis in capturing different linguistic characteristics, in intrinsic, such as word analogy, word similarity, and out-of-vocabulary words and odd-word out operation [46]. Besides intrinsic evaluations, the word embedding was evaluated on multiclass Amharic text classification task as an extrinsic evaluation. The resulting embedding 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. Out-of-vocabulary words were also entertained. Multiclass text classification on the model attained 97.8% F1-score; result being varied based on parameters. The author recommended [46] preparing a huge Amharic dataset to investigate the effect of morphology on word embeddings and the role of Amharic word embeddings in various NLP tasks.

Applied machine learning for Amharic text classification in [47], and observed the effect of preprocessing tasks such as stemming and POS tagging on the performance of text classification. The authors used a medium-sized, hand-tagged Amharic corpus and found out that stemming has no significant effect on the result of Amharic text classification, as without stemming 68% almost the same accuracy has showed with stemming which is 69%. In their work, bag-of-words approach used for word representation and applied it for text classification experiment. Their method suffers a huge sparsity problem which is the very nature of the way bag-of-words performs in word representations. Also their model faces the challenge of hitching very little information as the

vector-space is too huge. Using this approach also has another consequence on the probability of finding vectors for out-of-vocabulary words and the absence of context.

In [48] used 11,024 news articles and employed Naive Bayes and KNN and found out The best result obtained by the classifiers is on three categories data (95.80% vs. 89.61%) and the least performance is shown on the 16 categories (78.48% vs. 64.50%) respectively classification accuracy using Naive Bayes and KNN decreases if fewer documents are used in training. Further noticed that classifiers work well if news items are evenly distributed in the categories.

Context of Amharic documents similarity, the authors in [50] developed concept based Amharic approach that takes into account word level ambiguity in the documents. The main idea of the work is by building AmhWordNet which is used as input during concept tree extraction and to implement WSD. The extracted concept tree together with WSD helps to find the semantic similarity between words. This word similarity is used to compute sentence similarity.

3.3 Summary

All the related works in section 3.1 have developed for English language, if we used the architecture as is it doesn't provide the expected result for Amharic. We based our model on [42], [34]. And proposed some modifications on the architecture that consider Amharic language challenge, and the architecture that improves model performance as mentioned below

- Sub word information are added into the word vectors, in comparison of only used word2vec [16], glove, and character composed embedding using LSTM as an input layer [34] .
- In matching layer, our approach uses different granular matching (such as phase by sentence than word-by-word matching documented in [42]. The Multi-granularity is advantageous because entailment classification benefits from analyzing sentences at multiple levels. In addition, we used mean pooling than only max pooling as [34].
- In the matching direction, than matching Premises(P)-> Hypostasis(Q)and Q->P both ways as [34] or matching only in left direction as (matching *P* against *Q*) [42] we only matched in a reverse way Q->P (matching *Q* against *P*) , since it has not showed any improvement matching bilateral or in left direction.

TEC is vital in tasks ranging from information retrieval to semantic parsing to commonsense reasoning. In recent years, it has become an important testing ground for approaches in deep learning in which employing word representation and sentence representations followed by sentence matching techniques and this chapter we seen by leveraging deep neural networks used for TEC is also improved overtime.

Chapter 4: Design and Implementation of Amharic Entailment Classification

4.1 Overview

In this Chapter, the design and implementation of the proposed Amharic textual entailment recognition is presented in detail. The proposed architecture, depicted in Figure 4-1, is composed of five major layers: the input- word representation (word embedding), context representation or sentence embedding, matching, aggregation, and prediction layers.

The input is the bottom layer which is a word embedding constructed from unstructured Amharic text, and consists of words represented as vectors in high dimensional space. From this layer we get words meaning encoded as vectors. These embedded vectors are used for contextual embedding layer as an input of fragmented sentence sequence.

The second layer is the context embedding (sentence representation). This layer transforms each sentence from the sequence of word embedding into sentence embedding vectors using BiLSTM (bidirectional LSTMs) encoder to encode the meaning of the given sentences.

The third layer is Matching Layer that compares the output of each contextual embedding (time-step) of one sentence (hypothesis) against all contextual embedding's (time-steps) of the other sentences (premises) using Matching function followed by two matching approaches. The layer matches the two sentences in the reverse (right) direction which is hypothesis against the premise.

Aggregation layer, the fourth layer, aggregate the premise and the hypothesis matched vector into a fixed length matching vector though BiLSTM encoding model. It then passes the vector to MLP (multilayer perception) layers which is a densely connected layer that uses sigmoid as activation function.

Prediction layer, the last layer, is a MLP layer with SoftMax activation function. It has output of three classes. It's also known by a 3-way softmax layer. The result reflects on the relationship of matching value between premise and the hypothesis output sentences.

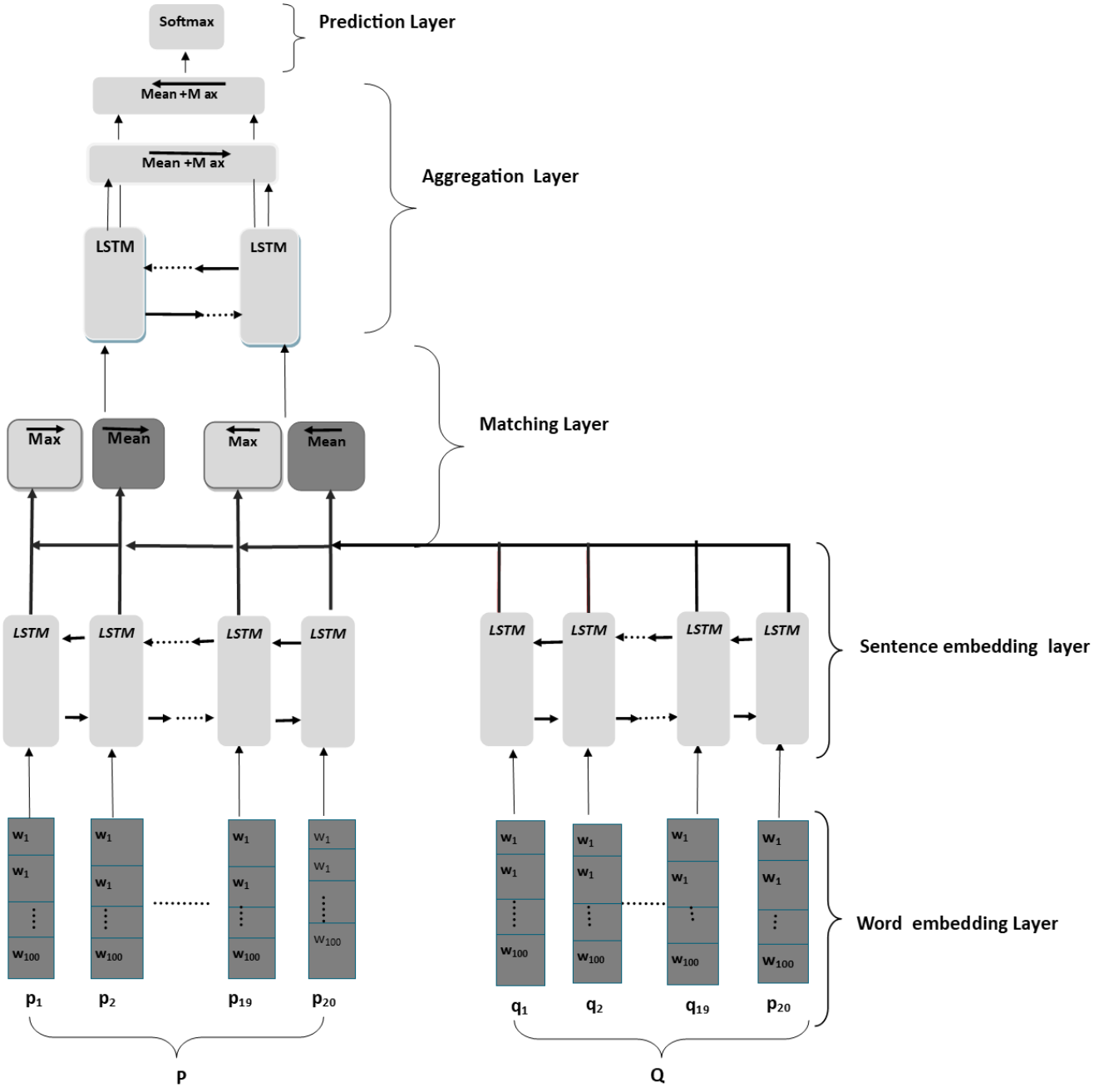

Figure 4-1: *Architecture for reverse side sentence matching (RSSM)*

The proposed architecture for reverse side sentence matching composed of five layer the first one is starts with the bottom word embedding ends with the top which is a prediction layer. The component we added are the first one, shaded in dark in word embedding layer, and mean pooling in matching layer. Also, in matching layer pointed in dark matching reverse side direction which is the hypothesis (Q) against the premise (P).

4.2 Word Embedding Layer

The goal of this layer is to represent each word in high dimensional space vector to encode its meaning. The word embedding has two components which will be trained jointly, one is its word and its sub word. We use sub word embedding the extension of word vector model of skip gram (SG) in which each word calculated by feeding the n-gram characters within the word, and also the word itself. We choose this structure of embedding since sub word information perform much better within morphologically complex language such as Amharic [20].

In this section, we explain how the model learn word representations while considering morphology of Amharic words. We will begin by presenting the component of sub word (SW) embedding model skip gram (SG), then present how sub word model handle the dictionary of character n-grams.

4.2.1 Skip-Gram Model (Word2Vec)

Word2Vec (skip-gram) [16] uses the logic in machine learning. Trained in a simple neural network with a single hidden layer to perform a certain task, the goal is actually to learn the weights of the hidden layer mentioned in chapter 2.

In one of the word vectors model skip gram approach trained the neural network to do the following. Given a specific word in the middle of a sentence (the input word), look at the words nearby and pick one at random. The network is going to tell us the probability for every word in our vocabulary of being the nearby word that we chose. When we say nearby, there is actually a window size parameter to the algorithm. For instance, consider the document as shown in Figure 4-2, "ሁለቱ ሴቶች ጓዛቸውን ይዘው እየተቃቀፉ ነው", (The two women were embracing while holding their bags) and window size in 2, meaning 2 words behind and 2 words ahead

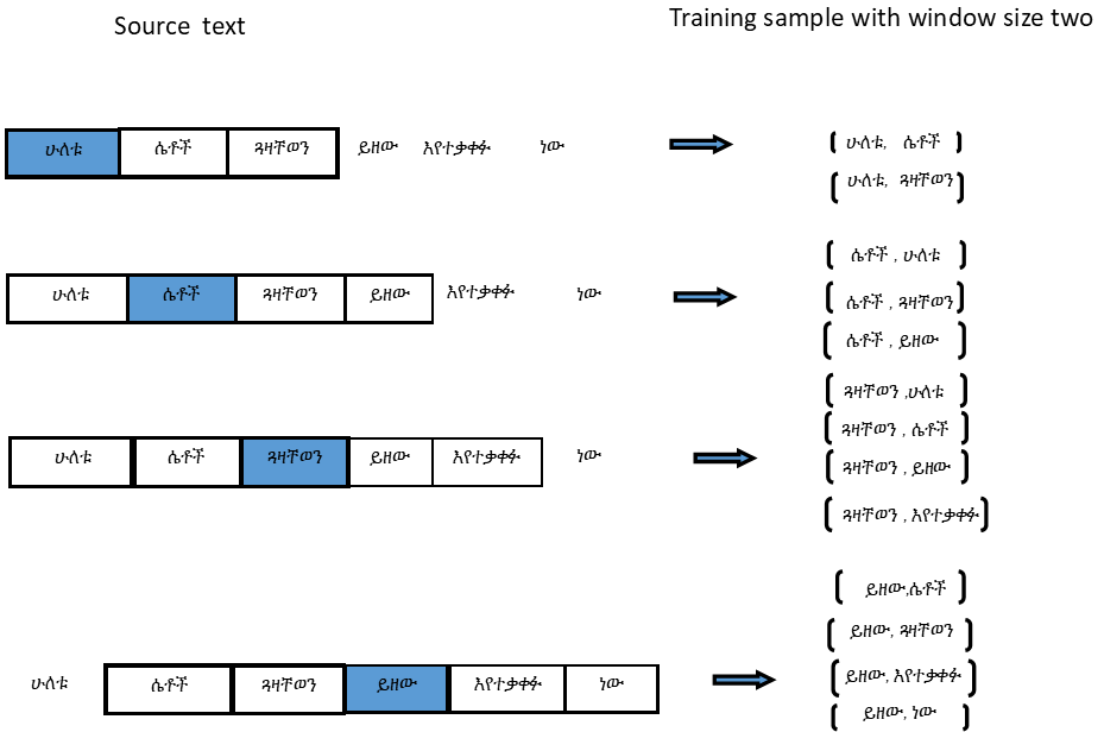

Figure 4-2: *Window size of two*

The neural network will learn the statistics from the number of times each pairing shown up. So, for example, in figure 4-2, it going to get many more training samples of ("ጓዛቸውን", "ይዘው") than it is of ("ጓዛቸውን", "እየተቃቀፉ"). if feeds the network a word "ጓዛቸውን" as input, after the training is done, then it will output a much higher probability for "ሴቶች" or "ይዘው" than it will for "እየተቃቀፉ").

If two different words have very similar "contexts" (that is, what words are likely to appear around them), then skip gram model will output very similar results for these two words. And one way for the network to output similar context predictions for these two words is if *the* word vectors are similar. So, if two words have similar contexts, then our network is motivated to learn similar word vectors for these two words.

And when we talk about for two words to have similar contexts, one could expect that synonyms like "ነፃነት" (freedom) and "ድል"(victory) or related words like "አየሱስ" (Jesus) and "ክርስቶስ" (Christ), would have very similar contexts.

4.2.2 Skip Gram with Sub Word Information

One huge limitation of SG model was that it ignored the morphological structure of the words and only assigned the features based on the semantic context of the word, and no assigned vectors for OOVs(out of vocabulary words) [20]. The major source of learning for SG model was the external neighbors of the word. This also limited it to learn the word vectors for only the words present in the vocabulary as stated in chapter two.

As, in languages like Amharic where the internal morphological structures of the words hold certain importance, the SG model failed to capture all the features of the available text data.

The mentioned limitation of word vectors can be addressed by adding sub word information into skip gram model [20] using fasttext library, for handling the morphological structures, rare words, and out of vocabulary words of Amharic linguistics.

The following issues are addressed by adding sub-word information onto skip gram model in Amharic word embeddings.

1. For a vocabulary size of 'W', vector representation for every *word* 'w' learned.
2. The word vector representation considers the context of the surrounding words.
3. The word vector representation considers the internal structure of the word. (morphology of words)
4. The word vector representation considers the word vectors for also out of vocabulary (OOV).

As we can see the first two lists can be achieved by applying only the skip gram model until there the algorithm is the same. But for the last two we need to append sub word model from that of word2vec algorithm.

4.2.3 Sub Word Embedding.

To achieve the 3rd goal in the above section, we represent the word "w" as *a* bag of n-grams in the word. The word is padded by a set of unique symbols such as star: *WORD*. This helps in distinguishing the suffixes and prefixes from the rest of the character sequences, also add the

complete word as a sequence into this bag of n-grams. And the 4th goal is achieved (for out-of-vocabulary (OOV) words), by summing up vectors for its component char-n grams, provided at least one of the char-n grams was present in the training data.

EXAMPLE: For n = 3, the word "እየተቃቀፉ" will give us the following n-grams bag represented as $\boldsymbol{r_w}$

$\boldsymbol{r_w}$ = ['∗እየ', 'እየተ', 'የተቃ', 'ተቃቀ', 'ቃቀፉ', 'ቀፉ∗', '∗እየተ', 'እየተቃ', 'የተቃቀ', 'ተቃቀፉ', 'ቃቀፉ∗', '∗እየተቃ', 'እየተቃቀ', 'የተቃቀፉ', 'ተቃቀፉ∗', 'እየተቃቀፉ']'>

Also because, we first created a dictionary of words and then a dictionary of the char n-grams, the word 'ተቃቀፉ' and the tri-gram 'ተቃቀፉ' in the word "እየተቃቀፉ" are assigned to different vectors and Amharic morphemes are identified as well: 'እየ', 'እየተ', ቀፉ.

Now, every character sequence in the n-grams bag is denoted by a vector <z> and the word vector <w> in eq. (5) [20], is given by the sum of the vectors of all the n-grams in that word. The context score function will be changed to:

$$\mathbf{s}(\mathbf{w}, \mathbf{c}) = \sum_{r \in r_w} \boldsymbol{z_r^T V_c} \qquad (5)$$

This allows the sharing of representations across words, thus allowing to learn reliable representation for rare words. This way, the sub word information is utilized in the learning process of calculating word embeddings.

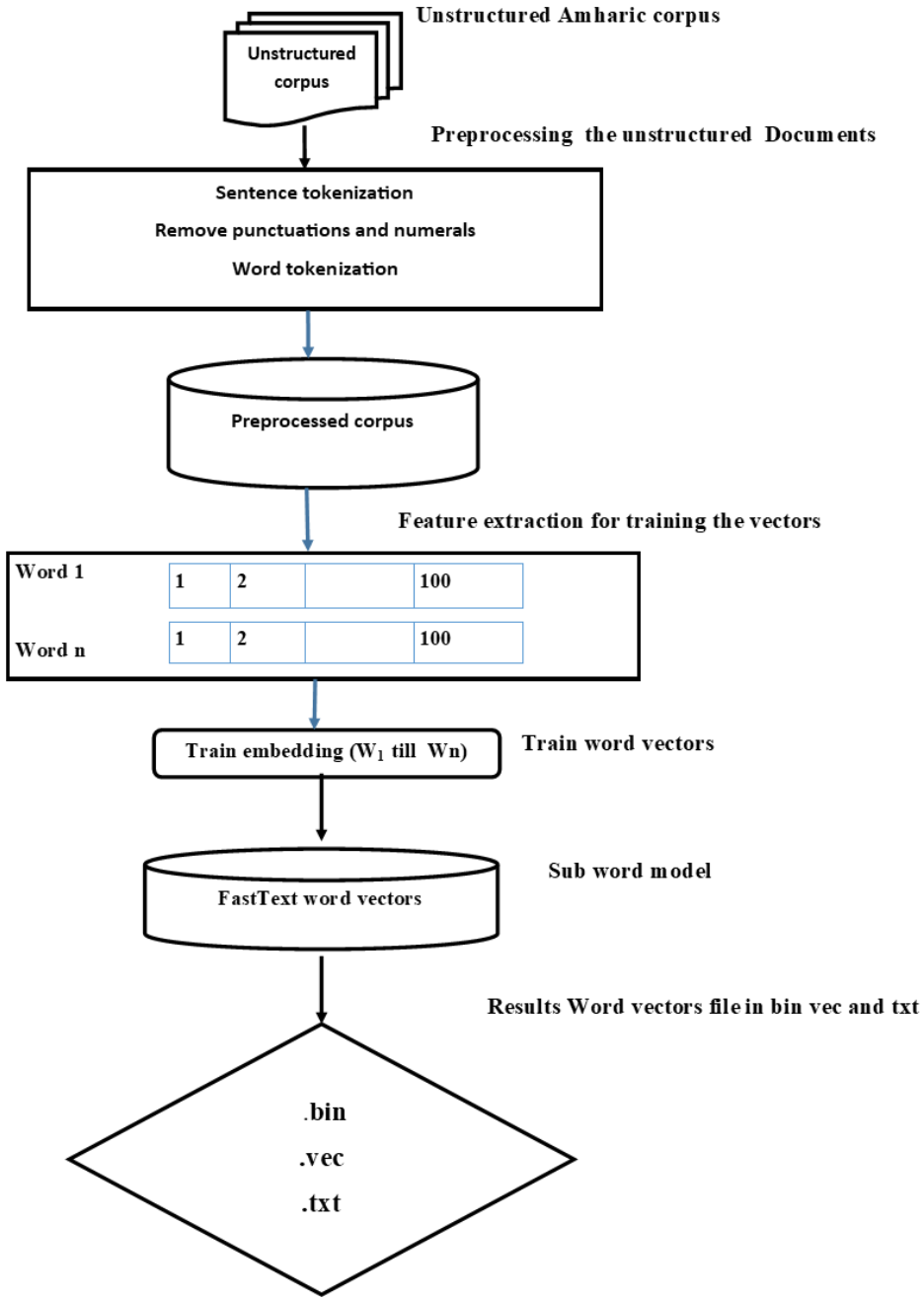

Figure 4-3: *Architecture of Sub-Word embedding.*

4.2.4 Implementation of Sub Word Embedding

The sub-word embedding is based on the architecture presented in Figure 4-3, in which it accepts unstructured text and preprocessed, it then performs feature extractions as fine tuning parameters

for training, by using genism library of fasttext we train model and finally returns the last output the trained word vectors in different file formats. Blow here the detail implementation is explained.

Algorithm 4.1: *Preprocessing Corpus*

```
Input: filename: unstructured file
Variable: Sent = []
Output:
Text = preprocessed text file
Begin:
1.  Sent = []
2.  for f in filename:
3.              Corpus_row =open(f)
4.              If corpus_row > 0
5.                 Rawsentence.readline(corpus_raw)// read line
6.              End if
7.  End for
8.  raw = sent_tokenize(Rawsentence)
    //sentences will be delimited by
9.  For each snt in raw
10.     Sclean = clean (snt)    /* using regular expression if the
             corpus has Amharic numerals OR punctuation marks
             */r"[፩፪፫፬፭፮፯፰፱።፡፣፤፥፦፧-.-/\_0-9a-zA-
             Z(;"'''")!…=.''''!()-{}”“‘’»«‹ ›;:'<>/?@#$%^&*~''']", "",

11.     if len(Sclean)>1
12.         split = clean_split(Sclean)
13.         Sent.append(split)
14.     End if
15. End for
16. Text = sentence_to_wordlist (Sent)
End
```

The algorithm 4.1 performs,

- ✓ **Loading Amharic corpus:** lines (1-7) load the unstructured Amharic corpus which intended of getting the knowledge all Amharic vocabularies (words) the corpus includes different themes.
- ✓ **Preprocess**: In lines 8-10 after loading the data the next step is preprocessing as separate punctuation, numbers and non-related characters from the corpus.
- ✓ **Read files to a wordlist**: In line (11 -13) read file into a list, so that we can pass this on to the sub word model applying document segmentation, or tokenizing a corpus using spaces and commas as white space.

For training the model we set the parameter by using genism fast text library we fine-tuned the parameters values as follows in Table 4.1.

Table 4.1: *Parameters for training word embeddings using fasttext library*

Parameters	**Description**	**values**
Min-count	minimal number of word occurrences	1
wordNgrams	Generate n-grams for the given word	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
Size	size of word vectors (dimension)	100
Window	size of the context window (context window)	5
iter	number of epochs	5
Neg	number of negatives sampled	5

The fine-tuned parameters values:

- ✓ **Min-count**: For min-count parameter set shown in table 4.1 for minimal word co-occurrence in the corpus. Because Amharic language is one of the most morphologically inflected, in which leads too many rarest words that might occurred once. And to overcome the problem of misspelling which the misspelled words that might occur once we set the min-count to the minimum which is 1.
- ✓ **Size**: The other parameter that needs fine-tuning is size, because we observed a variation as experimented in the size of dimension comparison between three dimensions: 100, 200 and 300 shows that a little progress is observed as dimension increases. However, the difference in the output between the two dimensions, 200 and 300, is found to be inexplicit, as shown in Table 4.2. Meanwhile 100 and 300 didn't show any big variation so we took 100 for conserving the RAM performance.

Table 4.2: *Variation in size parameter*

Dimensions	Terms	Most 5 similarity
100	እምነት በላ	በእምነት, እምነትም, ለእምነት, ከእምነት, እምነትና ሊበላ , አከራይቶ, በላች, ለጠጣ, ከጠጣ
200	እምነት በላ	ለእምነት, እምነትም, ከእምነት, በእምነት, እምነትና በላች, በላህ,በላው, በላሁ, ጠጉሩም
300	እምነት በላ	ለእምነት, በእምነት, ከእምነት, እምነትም, የእምነት ገፈራ, በላች, በላህ, ሊበላ, ከጠጣ

- ✓ **Window**: On the parameter of window size or context size is short, the result gets better. However, semantic relationship in different window size 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. Table 4.3, compares the results of analogical reasoning related to semantics like "man-woman" with two values of window size: 2 and 5. So for keeping the syntactic and semantic meaning intact preferred window size 5.

Table 4.3: *Variation in size parameter*

Semantic relatedness	Windows=2 with the top 5 similarity	Windows =5 with the top 5 similarity
If ወንድ is to ንጉሥ then ንግሥት is to what?	ንግሥት ለወንድ የወንድ ወንድም ወንድና ወንድምሽ ወንድምና ሴት	ንግሥት ለወንድ ሚስት ወንድና ሴት ከወንድ ወንድምዋ የወንድ

The feature extraction in sub-word embedding is done using modified word-vector modeling as presented in Algorithm 4.2. The algorithm accepts textual corpus, preprocesses it using algorithm 4.2 (line 1), generates n-gram representation using n-gram generator (line 2).

Algorithm 4.2: *Word Vector Modeling*

```
Input:
       Text : corpus   // corpus
       min_count : minimum co-occurrence of words
       Size : size of word vectors
       iter: number of epochs
       sg: train with skipgram
       window: size of the context window
       wordNgrams: if set to one it will generate n-grams for
                   each    wordlibrary
       FastText: train the model in subword embedding and
skipgram
Intermediate
       total_example: int
Output: SWE= trained word vectors model
Begin:
  1. Sentence =  preprocessingcorpus(Text)
     // result of calling algo 4.1
  2. FT = FastText(min_count, size, window, sg, wordNgram, iter
        wordNgrams)
  3. FT.build_vocab (Sentence)
  4. total_example = len(FT.wv.vocab)
  5. SWE = FT.train(Sentence , totalexamples, FT.iter)
  6. SaveModel(SWE)
End
```

- ✓ **N gram generator** (wordNgrams): In Algorithm 4.2, line 1 if wordNgrams is set to one it generates corresponding n grams aside to the word. N is the length of n-grams can be controlled using the -minn and -maxn flags of minimum and maximum range of values words in training. In this work, minn: 3 and maxn: 6 are set shown in Table 4.2 and given as parameters Algorithm 4.2.line 1. For instance, the generated word for "እየተቃቀፉ" and "ጓዛቸውን" Are

′እየተቃቀፉ′ = [′ ∗ እየ′,′እየተ′,′የተቃ′,′ተቃቀ′,′ቃቀፉ′,′ቀፉ ∗ ′,′ ∗ እየተ′,′እየተቃ′,′የተቃቀ′,′ተቃቀፉ′,′ቃቀፉ ∗ ′,′ ∗ እየተቃ′,′እየተቃቀ′,′የተቃቀፉ′,′ተቃቀፉ ∗ ′,′እየተቃቀፉ′]'>

′ጓዛቸውን = [′ ∗ ጓዛ′,′ጓዛቸ′,′ዛቸው′,′ቸውን′,′ውን ∗ ′,′ ∗ ጓዛቸ′,′ጓዛቸው′,′ዛቸውን′,′ቸውን ∗ ′,′ ∗ ጓዛቸው′,′ጓዛቸውን′,′ዛቸውን ∗ ′,′ጓዛቸውን′]'>

- ✓ **Build vocabulary**: In Algorithm 4.2, is done in Line 2 from a sequence of sentences and thus initialized the model. The model requires us to build the vocabulary table (digesting all the words and filtering out the unique words, and doing some basic counts on them).
- ✓ **Train the model**: the final step is to train the model to populate the word vectors which trained on Fasttext in line 4. by the total example which represent the count of the sentence, within the parameters given on line 1, and iterating in size of epochs as mentioned in Table 4.1.

The final output of word embedding layer is the trained word vectors by adding sub word information into the skip-gram model as mentioned in the above sections. This vector will be used as input in the next layer of sentence embedding.

4.3 Sentence Embedding Layer

This layer is responsible to incorporate contextual information into the representation of each time step of the sentence pair of the premise and hypothesis (*P* and *Q)* to form context2vec, by using BiLSTM. Bidirectional LSTMs train in two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in remembering long term sequences faster and even fuller learning on the problem.

The result of word embedding layer is the pertained word vectors, and this vector will be masked to the corresponding sequence of words found in the sentence pair of ANLI (Amharic Natural Language Inference) dataset, and will be trained together to embed the meaning of the sentences P and Q. The ANLI dataset used for entailment classification is the main input to the sentences embedding layer (BiLSTM), and then be trained on within the attached class label.

Table 4.4*: ANLI Example of entailment classification.*

Sentence 1(Premise)	Sentence 2(hypothesis)	Label
ሁለቱ ሴቶች ጓዛቸውን ይዘው እየተቃቀፉ ነው	እህትማማቾቹ ምሳ ለመብላት ከሄዱ በኋላ ጓዛቸወን ይዘው እየተሰነባበቱ ነበር	neutral
ሁለቱ ሴቶች ጓዛቸውን ይዘው እየተቃቀፉ ነው	ሁለቱ ሴቶች ጓዛቸውን ይዘዋል	entailment
ሁለቱ ሴቶች ጓዛቸውን ይዘው እየተቃቀፉ ነው	ወንዶቹ ከግቢጉ ውጭ እየተጣሉ ናቸው	contradiction

- **Sentence 1**: is the premise or the text sentence of the pair as shown in table 4.4.
- **Sentence 2:** is the hypothesis sentence for the pair, as a companion sentence for sentence1.
- **Label:** The label to be used for classification. In the above example which inferred as label which include the three kind of classification labels entailment contradiction and neutral.

Table 4.4, shows the three-label a of the dataset *sentence1* (Premises, P), *sentence 2* (Hypotheses, Q) and their pair label. Those are main component that we used to train model.

Formally, represented ANLI task as a triple (P; Q; y), where $\boldsymbol{p} = (\boldsymbol{p_1}, \dots \boldsymbol{p_i}, \dots . \boldsymbol{p_m})$ is a premise sentence with a length M, $\boldsymbol{Q} = (\boldsymbol{q_1}, \dots \boldsymbol{q_j}, \dots . \boldsymbol{q_n})$ is the hypothesis sentence with a length N, $\boldsymbol{y \epsilon Y}$ is the label representing the relationship between P and Q, and Y is either entailment; contradiction; neutral, where entailment indicates Q is inferred from P and Q is a sentence with similar meaning as the P; Contradiction indicates Q is a sentence with contradictory meaning in P and neutral means Q of the sentence with mostly the same lexical items as P but It neither proves nor disproves P.

The purpose of sentence embedding layer is to combine contextual meaning (encoding) into vector of each time step of P and Q in eq.(6) for the sequence modeling tasks. It is beneficial to have access to the past context as well as the future context [51], therefore we utilize BiLSTM layer for sentence embedding.

The premise P and hypothesis Q, are a fragmented word vectors as their notation is defined as follows.

$$\begin{aligned} \mathbf{P} &: [\mathbf{p_1}, \mathbf{p_2}, \dots \dots \ \mathbf{p_m}] \\ \mathbf{Q} &: [\mathbf{q_1}, \mathbf{q_2}, \dots \dots \ \mathbf{q_n}] \end{aligned} \tag{6}$$

Therefore, the premise P will be encoded in BILSTM as it goes with each-time step of P be as follows.

$$\mathbf{h_i^{p\rightarrow}} = \overrightarrow{\mathbf{LSTM}}(\mathbf{h_{i-1}^{p\rightarrow}}, \mathbf{p_i}) \qquad \mathbf{i = 1}, \dots, \mathbf{M}$$

$$\mathbf{h_i^{p\leftarrow}} = \overleftarrow{\mathbf{LSTM}}\left(\mathbf{h_{i+1}^{p\leftarrow}}, \mathbf{p_i}\right) \quad \mathbf{i = M, \dots, 1} \tag{7}$$

And same BiLSTM be implemented to encode hypothesis Q as it goes with each-time step of Q be as follows

$$\begin{aligned} \mathbf{h_j^{q\rightarrow}} &= \overrightarrow{\mathbf{LSTM}}\left(\mathbf{h_{j-1}^{q\rightarrow}}, \mathbf{q_i}\right) \quad \mathbf{j = 1, \dots, N} \\ \mathbf{h_j^{q\leftarrow}} &= \overleftarrow{\mathbf{LSTM}}\left(\mathbf{h_{j+1}^{q\leftarrow}}, \mathbf{q_i}\right) \quad \mathbf{j = N, \dots, 1} \end{aligned} \tag{8}$$

*where **h** is the hidden layer units of P and Q*

BiLSTM can be trained using similar algorithms to LSTMs, as the two directional neurons do not have any interactions. However, when back-propagation is applied, additional processes are needed because updating input and output layers cannot be done at once in eq. (7) and (8) [51]. General procedures for training are as follows:

- ✓ For forward pass, forward states and backward states are passed first, then output neurons are passed in both $\mathbf{h_i^{p\rightarrow}}$ and $\mathbf{h_j^{q\rightarrow}}$ of P and Q
- ✓ For backward pass, output neurons are passed first, then forward states and backward states are passed next after forward and backward passes are done, the weights are updated in both $\mathbf{h_i^{p\leftarrow}}$ and $\mathbf{h_j^{q\leftarrow}}$ of P and Q

4.3.1 Preprocessing for Sentence Embedding

The objective in sentence preprocessing is to make ready the sentences and combine contextual meaning (encoding) into vector. Each sentence (P and Q) found in the dataset is a sequence of words which will be changed to word index and padded with the specific number and masked with the equivalent word vectors. In the following section, we illustrate how these sentence sequences are preprocessed and mapped to word vectors.

Algorithm 4.3: *Preprocessing for sentence embedding*

`Input: TF: train file name`
`Variable: K = []`
`Intermediate:`

```
      SWE: pertained word embedding


Output:
WEM: word embedding matrix
PSP: Padded sequence sentence
PSQ: Padded sequence hypothesis
Begin:
  1.  If exists(TF):
  2.    Train_row =open(TF)
  3.    TS.exract(Train_row)
  4.  End if
  5.  TS,TS1, TS2, TL = loadDataset(TS)
      // the train sentence has three fields the premises , hypothesis
      an label
  6.  embedding_dim = size(SWE)
  7.  PTE = load(SWE)  // load the result of the model generated using
                          the word
                       // embedding algorithm 4.1 word+embedding.model()
  8.  For line on PTE:
  9.      Values = split(line,'')
  10.     Word = values[0]
  11.     Embedding.add(word)
  12. End for
  13. NUM_WORD= len(Embedding)
  14. TS= TS1+TS2
  15. tokenizer =Tokenizer(NUM_WORDS)
  16. tokenizerfitontexts(TS)
  17. SWS1 = tokenizer.texts_to_sequences(TS1)
  18. SWS2 = tokenizer.texts_to_sequences(TS2)
  19. word_index = tokenizer.word_index
  20. words_len = min(NUM_WORDS, length(word_index))
  21. K=[]
  22. for word, i in word_index.items():
  23.    if i >= NUM_WORDS:
  24.      continue
  25.    embedding_vector = Embedding.get(word)
  26.    if embedding_vector is not None:
  27.        word_embedding_matrix = embedding_vector
  28.        K.add( word_embedding_matrix )
  29.    Else
  30.        WM = assignrandom((words_len + 1, embedding_dim))
  31.        End if
  32.    End if
  33. End for
  34. WEM = getembedding(K)
  35. Return WEM
  36. PSP = pad_sequences(SWS1, maxlen)
  37. PSQ = pad_sequences(SWS2, maxlen)
End
```

As shown in Algorithm 4.3, Lines (1 - 3) read, extract and load the training dataset of ANLI with three fields. In lines (15-20) to find the word vectors for a certain context or paragraph, tokenize the sentences, and create a vocabulary that maps strings into unique integers, then after can be identified with word vectors. In this process, if a word is not identified, it is assigned as unknown token.

For instance, the sentence: *ሁለቱ ሴቶች ጓዛቸውን ይዘው እየተቃቀፉ ነው* is encoded to the word index list [143, 24, 1716, 271, 2552, 3].

In Algorithm 4.3 lines (37 -38) shows padding the sentence word sequences. When processing sequence data, it is very common for individual samples to have different lengths, and hence the shorter sentence need to be padded with 0 value, for example from ANLI dataset the maximum of the premise and hypothesis length is shown as below.

P (MAX) ሁለት ሰዎች ስለ ሰው ልጅ አመጣጥ ሲከራከሩ አንድ ሰው ለማስማማት ብሎ ስለሰነ ፍጥረት እና ስለ ዘረ መል ለማብራራት በቅርብ ካነበባቸው መጽሐፍት ውስጥ የገረመውን ነገር ለመናገር ሲሞክር ጆሮ ሊሰጡት ስላልቻሉ ወደ መጣበት ቦታ ሲመለስ አንዱ ልጅ አንተ አራስህ ባላረጋገጥከው ነገር እርግጠኛ ሆነህ ምንም ነገር ልትነግረን አትችልም አለው

[667, 429, 227, 377, 779, 818, 893, 606, 827, 75, 700, 273, 145, 36, 280, 169, 391, 27, 48, 30, 138, 30, 2, 3, 7, 8, 1, 4, 16, 10, 5, 2, 6, 3, 10, 1, 6, 9, 2, 1]

H (MAX) አንድ የአራት ወንዶች የሙዚቃ ቡድን እንዲሁም ሁለት ሳክስፎን ተጫዋቾች አንድ መለከት ነፊ እና አንድ የትራምፔት ተጫዋች መድረክ ላይ እየተጫወቱ ነው ሌሎች ሁለት ወንዶች ደግሞ ከቡድኑ ፊት ለፊት እየጨፈሩ ነው

1717, 1369, 375, 1505, 775, 1004, 616, 210, 56, 138, 122, 94, 17, 4, 33, 2, 12, 8, 1, 1, 1, 1]

Individual samples in the P and Q have different length across the dataset. Since the input data for a deep learning model must be a single tensor (batch_size, sequence_length, vocab_size (20,100)). We used pre -padding, so we make sure final hidden state of LSTM wouldn't get flushed out as mostly it will be 0, and make sure that the hidden states returning output in every time step.

As most of the sentence pair has a length below 20, we do not consider to pad with the maximum length in neither of the sentences, as because most of the sequence will be padded with zero so to eliminate that we pad P and Q with 20 as the following examples.

P = ሁለቱ ሴቶች ጓዛቸውን ይዘው እየተቃቀፉ ነው

Sequence of word list: [143, 24, 1716, 271, 2552, 3]

Padded sequence of word list P: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 143 24 1716 271 2552 3].

Each padded sequences word list will be masked with the word vectors of the P and Q to the word embedding matrix as shown in Algorithm 4.3 line (21-34), with the size of embedding dimension.

Note that word embedding has size of 100 dimension, Input shape of sequence length is 20 this so each word matrix has a shape of (20,100).

4.3.2 BiLSTM Sentence Encoding

After preprocessing performed on the pair of sentences, the next step is to feed these sentences to the BILSTM layer. The padded sentence sequence in Algorithm 4.3 Line (13- 16) this sentence sequences will be masked with the corresponding word vectors as shown in the Algorithm 4.4 (Line 4-10).sequences will be masked with the corresponding word vectors as shown in the Algorithm 4.4 (Line 4-10).

Algorithm 4.4: *Sentence Embedding in BiLSTM*

```
Input:
      inputs: sentence sequence1 and sentence sequence2,
   units: layers of BiLSTM(hidden layers)
   SL:  maximum sequence length
   SEM: pre-trained word vector
   WEM: word embedding matrix
   PSP: Padded sequence sentence
   PSQ: Padded sequence hypothesis

Output:
   BE: backward sentence Representation in BILSTM
   FE: forward sentence Representation in BILSTM
Begin:
1.   embedding_dim = size (SEM)
2.   WC = word_context(inputs)
3.   (text_embedding, hypo_embedding ) = word_input ( PSP,   PSQ )
     // sentence pre-processing – algo 4.3
4.   For input in  text_embedding, hypo_embedding
5.        Embedding = word_embedding(word_len+1, embedding_dim, SL,
     WEM)(input)
     //  WEM word embedding matrix calling of algorithm 4.3
6.        Embed.mask(Embedding)
7.        For each word in Embed
8.          If word > 1
9.              Context = BiLSTM(units,Seq_return)(word)
10.             Return Context
11.             End if
12.      End for
13.  End for
14.  Forward_context=[]
15.  Forward_context.extend(text_embedding)
16.  Forward_context.extend( hypo_embedding)
17.  FE= Forward_context.add(context)
18.  Backward_context []
19.  Backward_context.extend( hypo_embedding)
20.  Backward_context.extend( text_embedding)
21.  BE = Backward_context.add(context)
End
```

In Algorithm 4.4. when the sentences sequences masking is done, it then passed to BILSTM layer for representing the contextual meaning of the sentence, this embedding returns each time steps in each hidden layers of eq. (7) and (8) of BiLSTM as in line (8-11) (This will be explained in the following section of BiLSTM hidden units and directions) sentences embedded both forward and backward directions a line (14-21). Finally the returned sentence in both forward and backward

will have the shape of the sentence length times the hidden layers units 40 and sequence length 40 as (20, 40) of both P and Q.

4.3.3 BiLSTM Hidden Units and Time Steps

Figure 4-4 shows how BiLSTM represents the sentence P with the output intermediate vector result generated in each time step.

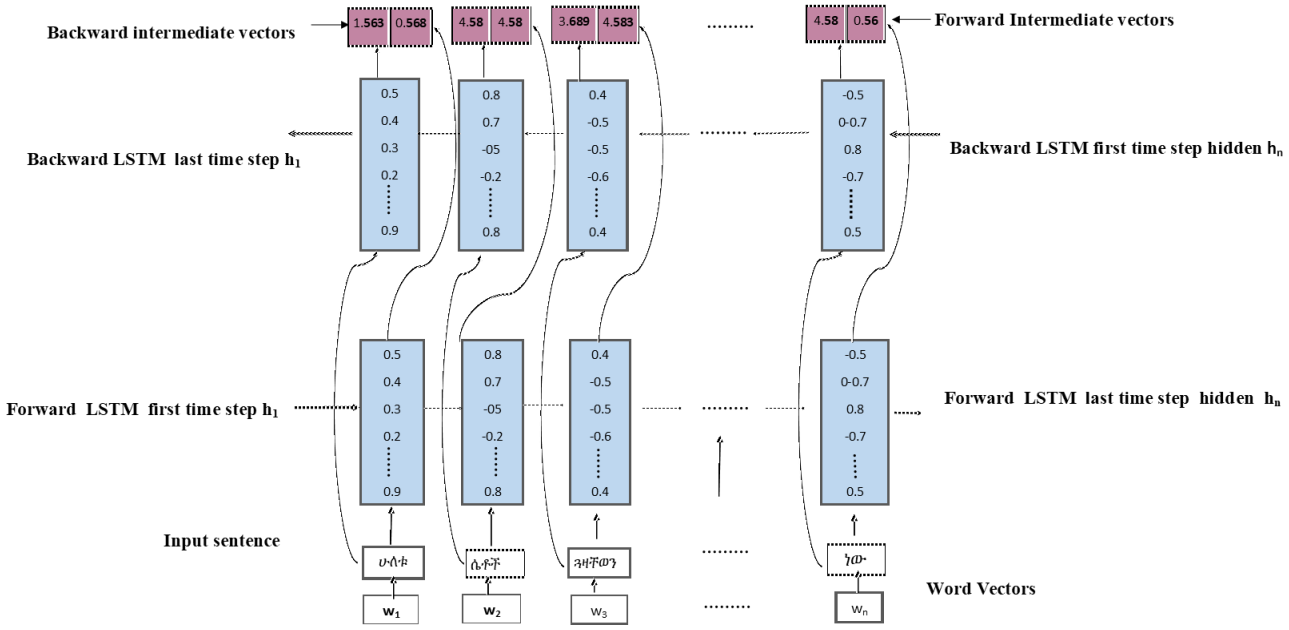

Figure 4-4: *BiLSTM Sentence Encoding*

a) BiLSTM Forward Hidden Steps

Once the P and Q sequences masked with the corresponding words vector, it will then passed to a forward LSTM block, it goes through all time step of hidden state until it generates the final intermediate representation as shown in Figure 4-4. When the first token of the premise passed to the forward LSTM, hidden representation is generated at time step 1 as shown in eq. (9) this hidden state is concatenated with the word vector of second word, and in the next step h_1 to get modified h_2. If there are 'n' words in the sentence then h_n will be the last hidden state. We have an option to either pass h_n or all the states h_1 to h_n (also called states at different time steps) in the subsequent layer.

Let's consider when the premises represented as follows:

1. Premises: **ሁለቱ ሴቶች ጓዛቸውን ይዘው እየተቃቀፉ ነው**

2. Then this premises will be masked with the corresponding embedding vector. Embedding input shape dimension will have an output of 20 words i.e. P (w_1 - w_n):
   [0.94292888 0.12460776 0.97114451 ... 0.12968908 0.01652156 0.88459217]
3. Thus, LSTM of the forward direction: $\overrightarrow{\boldsymbol{P}} = \boldsymbol{w_1, w_2, \cdots w_n}$ is generated
4. Premises for the forward direction has hidden step in each time step and are represented as eq. (9):

$$\begin{aligned} \overrightarrow{\boldsymbol{h_1}} &= \boldsymbol{h_0} + \boldsymbol{w_1} \\ \overrightarrow{\boldsymbol{h_2}} &= \boldsymbol{h_1} + \boldsymbol{w_2} \\ &\vdots \\ \overrightarrow{\boldsymbol{h_n}} &= \boldsymbol{h_{n-1}} + \boldsymbol{w_n} \end{aligned} \qquad (9)$$

b) BiLSTM Backward Hidden Steps

Once the P and Q sequences masked with the corresponding words vector, this will be passed to a backward LSTM block, it goes through all time step of hidden state until it generate the final intermediate representation as shown in figure 4-4. When the last token of the premises passed, it generate the last step of the premises that is h_n. This hidden representation is generated at time step 1. This hidden state is concatenated with the word vector of the last second word and in the next step h_n now gets modified to h_{n-1}. If there are 'n' words in the sentence then h_1 will be last hidden state as illustrated in eq. (10).

Let's say the premises represented as follows:

1. Premises: ሁለቱ ሴቶች ጓዛቸውን ይዘው እየተቃቀፉ ነው
2. And the sequence of word embedding of the sentence of the premises: Wn - W_1
   [0.12968908 0.01652156 0.88459217…0.94292888 0.12460776 0.97114451 ...]
3. So LSTM of the backward direction: $\overleftarrow{\boldsymbol{P^b}} = \boldsymbol{w_n, w_{n-1}, \cdots w_1}$
4. Premises for the backward direction representation of hidden step in each time step

$$\begin{aligned} \overleftarrow{\boldsymbol{h_n}} &= \boldsymbol{h_0} + \boldsymbol{w_n} \\ \overleftarrow{\boldsymbol{h_{n-1}}} &= \boldsymbol{h_n} + \boldsymbol{w_{n-2}} \\ &\vdots \\ \overleftarrow{\boldsymbol{h_n}} &= \boldsymbol{h_1} + \boldsymbol{w_1} \end{aligned} \tag{10}$$

When a new sentence is feed into BiLSTM layer, hidden state in eq. (10) h_n resets, and assigned at the initial step i.e. step 0. Hidden state was randomly initialized during time step. Then it start embedding from the last word of the premise by concatenating with each time step of the hidden layer concat (word vector, hidden state) until it goes through all hidden layers for resulting the final intermediate backward representation as shown in Figure 4-4.

Sentence embedding layer with ANLI dataset has 8700 pair of sentences contains 20 words sentence sequences and the BiLSTM layer has 40 units (hidden layers), and the output would be in shape of (8700, 20, 40). Therefore, each sentence word fragments (sequences) will go through 40 LSTM units, and then generate an array of 40 float numbers by performing backpropagations. The same process will be done for all 20 words in the sentence. So, at the end, get an output of shape (batch_size, 20, 40) per sentence in both directions (forward and backward).

4.4 Matching Layer

This is the fundamental layer for the classification model. The goal of this layer is to compare each contextual embedding (time-step) of one sentence against all contextual embedding (time-steps) of the other sentence. We followed approaches to match P and Q using matching function followed by two matching approaches of mean and max polling in Q's perspective.

In the matching layer, the inputs are the sentence embedding of each forward and backward vector which mentioned in section 4.3.2. And these vector sequence represent the semantic meaning in granular (word, phrase, sentence) level as P and Q represented in every Timestep. And to implement a matching approach, first, we calculate the cosine similarities between each backward and forward contextual embedding. To match each backward and forward contextual embedding one time-step of a sentence against all time-steps of the other sentence, we implemented a cosine (element wise) matching operation detailed in the next sub-sections.

4.4.1 Matching Function

The element wise matching operation works in two steps. The first step, define a cosine matching function $\boldsymbol{c_m}$ to compare two vectors

$$\boldsymbol{m = c_m(v_1, v_2; W)} \tag{10}$$

Where : $\boldsymbol{v_{1,}}$ and $\boldsymbol{v_{2,}}$ are two d-dimensional vectors, $\boldsymbol{W \in R^{l*d}}$ Is a trainable parameter with the shape $\boldsymbol{l * d}$, $\boldsymbol{l}$ Is the number of time steps we want to calculate, *d is* the dimension of the hidden state and. And the returned value *m* is an *l*-dimensional vector $\boldsymbol{m = m_1 \ldots m_k \ldots m_l}$, each element $\boldsymbol{m_k \in m}$ is a matching value from the $\boldsymbol{k^{th}}$ time step, and it is calculated using the cosine similarity of the two weighted vectors.

$$\boldsymbol{m_k = cosine(W_k \circ v_{1,}, \quad W_k \circ v_{2,})} \tag{11}$$

Where ∘ is the element-wise multiplication, and $\boldsymbol{W_{k,}}$ is the $\boldsymbol{k^{th}}$ row of *W*, which controls the k^{th} time step and assigns different weights to different dimensions of the *d* dimensional space.

4.4.2 Matching Approaches

In the second step, based on $\boldsymbol{C_m}$, on eq. (11) we applied two matching approaches to compare each time-step of one sentence against all time-steps of the other sentence, and apply two matching approach mean and max pooling over it. The first approach preserves only the maximum value of each dimension as shown in eq. (12), and the second preserves the average value of each dimensions of $\boldsymbol{m_k}$ of eq. (13)

Performed the matching in reverse direction the hypothesis against the premises $\boldsymbol{Q \leftrightarrow P}$ for each H-hidden hypothesis vector $\boldsymbol{h_j^q}$ we will compute a vector $\boldsymbol{m_i \epsilon R^l}$ of perceptions using the premises -hidden vector $\boldsymbol{h_i^p}$ as follows: For $\boldsymbol{m \ \epsilon [0, l)}$.

Max−pool Matching $$\boldsymbol{m_j^k = max_{i \in (1 \ldots M)} \frac{(W_k \circ h_j^q)^T (W_k \circ h_i^p)}{||W_k \circ h_j^q|| \, ||W_k \circ h_i^p||}} \tag{12}$$

Mean −pool Matching $$\boldsymbol{m_j^k = \frac{1}{m} \sum_{i=1}^{m} \frac{\left(W_k \circ h_j^q\right)^T (W_k \circ h_i^p)}{\left\|W_k \circ h_j^q\right\| \left\|W_k \circ h_i^p\right\|}} \tag{13}$$

Let's take two sentences in premises and hypothesis as v1 and v2 respectively.

v1= ["ሁለቱ ሴቶች ጓዛቸውን ይዘው እየተ፡ቃቀፉ ነው"]

v2= ["እህትማማቾቹ ምሳ ለመብላት ከሄዱ በኋላ ጓዛቸውን ይዘው እየትነሰባበቱ ነበር"]

As the matching performed in Q's side the sentence input will be the two embedding vectors of $\boldsymbol{Q} \leftrightarrow \boldsymbol{P}$ which are:

$$\begin{aligned} \boldsymbol{Q} \rightarrow \boldsymbol{P} &= \boldsymbol{LSTM}(\boldsymbol{h}\binom{\boldsymbol{j}}{\boldsymbol{q}}^{\rightarrow}, \boldsymbol{h}\binom{\boldsymbol{i}}{\boldsymbol{p}}^{\rightarrow} \\ \boldsymbol{Q} \leftarrow \boldsymbol{P} &= \boldsymbol{LSTM}(\boldsymbol{h}\binom{\boldsymbol{j}}{\boldsymbol{q}}^{\leftarrow}, \boldsymbol{h}\binom{\boldsymbol{i}}{\boldsymbol{p}}^{\leftarrow} \end{aligned} \tag{14}$$

Now the BiLSTM return the result of the forward and backward sentence embedding as eq. (14) [6] goes from Q to P

V1= ["እህትማማቾቹ ምሳ ለመብላት ከሄዱ በኋላ ጓዛቸውን ይዘው እየትነሰባበቱ ነበር"], V2 ["ሁለቱ ሴቶች ጓዛቸወን ይዘው እየተ፡ቃቀፉ ነው"]

V1= ["ነበር እየትነሰባበቱ ይዘው ጓዛቸውን በኋላ ከሄዱ ለመብላት ምሳ እህትማማቾቹ"], V2 ["ነው እየተ፡ቃቀፉ ይዘው ጓዛቸውን ሴቶች ሁለቱ "]

$$\boldsymbol{Q} \leftrightarrow \boldsymbol{P} = \boldsymbol{BiLSTM}(\boldsymbol{v_1}, \boldsymbol{v_2}) \tag{15}$$

From the above example of the sentence representation both output of forward and backward will be returned to the matching layer, to calculate $\boldsymbol{m_k}$, as eq. (11) first we compare each time step of one the hypothesis Q against all time step of the premises P ($\boldsymbol{Q} \leftrightarrow \boldsymbol{P}$). The output results of as the embedding goes to $\boldsymbol{Q} \rightarrow \boldsymbol{P}$ and $\boldsymbol{Q} \leftarrow \boldsymbol{P}$ will have a shape of (20, 40), transpose the hypothesis matrices and keep the premises matrices as it is and we perform the matching as below.

From the eq. (15) as. In making the cosine similarities we using the hidden vector of two sentences as $\left(\boldsymbol{W_k} \circ \boldsymbol{h_i^p}\right)\left(\boldsymbol{W_k} \circ \boldsymbol{h_j^q}\right) = (\boldsymbol{20} * \boldsymbol{40}), (\boldsymbol{20} * \boldsymbol{40})$ and transposing one of the sentence hidden layer in this case the hypothesis hidden vector would be as

$$\left(\boldsymbol{W_k} \circ \boldsymbol{h_i^p}\right)\left(\boldsymbol{W_k} \circ \boldsymbol{h_j^q}\right)^{\boldsymbol{T}} = (\boldsymbol{20} * \boldsymbol{40}), (\boldsymbol{40} * \boldsymbol{20})$$

Let's say $\left(\boldsymbol{W_k} \circ \boldsymbol{h_i^p}\right) = \begin{bmatrix} x_{11} & x_{12} & x_{13}, & x_{14} & x_{15} & x_{16} & x_{17} & \dots\dots\dots\dots\dots & x_{140}, \\ x_{21} & x_{22} & x_{23}, & x_{24} & x_{25} & x_{26} & x_{27} & \dots\dots\dots\dots\dots & x_{240}, \\ x_{31} & x_{32} & x_{33}, & x_{34} & x_{35} & x_{36} & x_{37} & \dots\dots\dots\dots\dots & x_{340}, \\ & & & & \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots & & & & \\ x_{201} & x_{202} & x_{203}, & x_{204}, & x_{205} & x_{206} & \dots\dots & \dots\dots\dots\dots & x_{2040} \end{bmatrix}$

And $\left(\boldsymbol{W_k} \circ \boldsymbol{h_j^q}\right)^{\boldsymbol{T}} = \begin{bmatrix} y_{11} & y_{12} & y_{13}, & \dots\dots\dots\dots\dots & y_{120}, \\ y_{21} & y_{22} & y_{23}, & \dots\dots\dots\dots\dots & y_{220}, \\ y_{31} & y_{32} & y_{33}, & \dots\dots\dots\dots\dots & y_{320}, \\ y_{41} & y_{42} & y_{43}, & \dots\dots\dots\dots\dots & y_{420}, \\ y_{51} & y_{52} & y_{53}, & \dots\dots\dots\dots\dots & y_{520}, \\ y_{61} & y_{62} & y_{63}, & \dots\dots\dots\dots\dots & y_{620}, \\ y_{71} & y_{72} & y_{73}, & \dots\dots\dots\dots\dots & y_{720}, \\ & & \dots\dots\dots\dots\dots\dots\dots\dots\dots & & \\ y_{401} & y_{402} & y_{403}, & \dots\dots\dots\dots\dots & y_{4020}, \end{bmatrix}$

As element wise multiplication of each time step of Q's hidden layer which is $\boldsymbol{h_j^q}$ against all time step of P's hidden layer $\boldsymbol{h_i^p}$ in the range of $\boldsymbol{i} \in (\boldsymbol{1} \dots . \boldsymbol{M})$ will generates:

$$\left(\boldsymbol{W_k} \circ \boldsymbol{h_i^p}\right)\left(\boldsymbol{W_k} \circ \boldsymbol{h_j^q}\right)^{\boldsymbol{T}}$$

$$\begin{bmatrix} x_{11}y_{11} + x_{12}y_{21} + x_{13}y_{31} \dots + x_{140}y_{401}, & x_{11}y_{12} + x_{12}y_{22} + x_{13}y_{32} \dots + x_{140}y_{402}, & \dots\dots & x_{11}y_{120} + x_{12}y_{220} + x_{13}y_{320} \dots + x_{140}y_{4020} \\ x_{21}y_{11} + x_{22}y_{21} + x_{23}y_{31} \dots + x_{240}y_{401}, & x_{21}y_{12} + x_{22}y_{22} + x_{23}y_{32} \dots + x_{240}y_{402}, & \dots\dots & x_{21}y_{120} + x_{22}y_{220} + x_{23}y_{320} \dots + x_{240}y_{4020} \\ x_{31}y_{11} + x_{32}y_{21} + x_{33}y_{31} \dots + x_{340}y_{401}, & x_{31}y_{12} + x_{32}y_{22} + x_{33}y_{32} \dots + x_{340}y_{402}, & \dots & x_{31}y_{120} + x_{32}y_{220} + x_{33}y_{320} \dots + x_{340}y_{4020} \\ & \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots & & \\ x_{201}y_{11} + x_{202}y_{21} + x_{203}y_{31} \dots + x_{2040}y_{401}, & x_{201}y_{12} + x_{202}y_{22} + x_{203}y_{32} \dots + x_{240}y_{402}, & \dots & x_{201}y_{120} + x_{202}y_{220} + x_{203}y_{320} \dots + x_{2040}y_{4020} \end{bmatrix}$$

Then after making the matrix multiplication and cosine similarity made with the two sentences hidden layer in a shape of (20*40) and (40*20) matrix we will have a result of cosine matrix (20*20) as follows:

$$\boldsymbol{m_k} = \begin{bmatrix} w_{11} & w_{11} & w_{12} & \dots & w_{120} \\ w_{21} & w_{22} & w_{23} & \dots & w_{220} \\ w_{31} & w_{32} & w_{33} & \dots & w_{320} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{201} & w_{202} & w_{203} & \dots & w_{2020} \end{bmatrix}$$

So we will make a mean and max pooling in each row of w and get W which is a matching column vector as (20*1) at the end, as we can see from the first row

w_{11} w_{12} w_{13}, $\cdots$ w_{120} we will take w_1 as a filter vector with respect to the function we have performed.

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & \cdots & w_{120} \\ w_{21} & w_{22} & w_{23} & \cdots & w_{220} \\ w_{31} & w_{32} & w_{33} & \cdots & w_{320} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{201} & w_{202} & w_{203} & \cdots & w_{2020} \end{bmatrix} = \overrightarrow{mean}(W^1) \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{20} \end{bmatrix} \text{ and } \overrightarrow{max}(W^2) \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{20} \end{bmatrix}$$

To sum up based on the result of C_m, applied two matching approaches to compare each time-step of one sentence against all time-steps of the other sentence. As we explained in the above section, each approach, first calculates m_k cosine similarities between each forward and backward contextual embedding i.e. $Q \rightarrow P$ and $Q \leftarrow P$.

Used two matching strategies which are mean pooling and max pooling, this pooling helps in filtering (picking selected vectors) against the similarity index based on the given pair of sentences vectors.

Algorithm 4.5: *Mean-Pooling-Reverse (hypothesis against premises) matching*

```
Input:
    BE: sentence embedding from hypothesis to premise
    FE: sentence embedding from premise to hypothesis

Output: right: Mean Pooled Vectors in reverse direction.
Begin:
1.  Right_match= []
2.  Left_match =[]
3.  CRMR (TV, HV) = Forward_context(FE):// algo 4.4.
4.  CRML (HV, TV) = Backward_context(BE)// algo 4.5.
    //for filtering embedding vector most similar for the input sentence
5.  For each text_vector, hypo_vector in CRMR(TV, HV)
6.        text_vector_tmp= tf.expand_dims(text_vector, 1)
7.        hypo _vector_tmp= tf.expand_dims(hypo _vector, 2)
8.        FM=cosine_distance(text_vector_tmp,hypo_vector_tmp)
9.         Forward_relevancy_matrix.Add(RM)
10. End For
11. For each text_vector, hypo_vector in CRMR(HV, TV)
12.       hypo_vector_tmp= tf.expand_dims(hypo _vector, 1)
13.       text _vector_tmp= tf.expand_dims(text _vector, 2)
14.       BM=cosine_distance(hypo_vector_tmp,text_vector_tmp)
15.         Backward_relevancy_matrix.Add(FM)
16. End For
```

```
17. right_side = matchinglayerR(Backward_relevancy_matrix)
18. left_side =  matchinglayerR(Forward_relevancy_matrix)
19. For RM in  right_side :
20.     BRM = load(RM)
        // this loads the call relevancy matrix of right context(Q,P)
21.     representation = []
22.     If (representation ≠ isinstance) // if the specified input shape is
    none
23.          MP = representation.append(tf.reduce_mean(cosine_matrix))
                 /* tf.reduce_mean is  tf library for calculating mean value
                 over the cosine similarity of the right sentence embedding
                 */
24.          RMP= RMP(representation,  BRM )
25.          right_match.add(RMP)
26.     End if
27. For LM in  left_side :
28.     FRM =load (LM)
29.     representation = []
30.     If (representation ≠ isinstance)// if the specified input shape is
    none
31.           MP = representation.append(tf.reduce_mean(cosine_matrix))
                           // tf.reduce_mean is  tf library for calculating
                 mean          value over the cosine similarity of left
                 sentence embedding
32.            LMP = MP (representation,  FRM)
33.            left_match.add(LMP)
34.      End if
35. End for
36. right_representation= (right_match)
37. right_representation.extend(left_match)
38. right =concatenate (right_representation)
39. Return right
    //returns the matching vector concatenated
End
```

a) Mean-pooling Reverse Matching

The mean pooling preserves the average features needed to find the weaker alignment in the sentences, by calculating the average result in all time steps of the sentence using the weighted summing all the contextual embedding's cosine similarity result of the entire sentence. This will be best in recognizing the contradiction and neutral label of the two sentences P and Q.

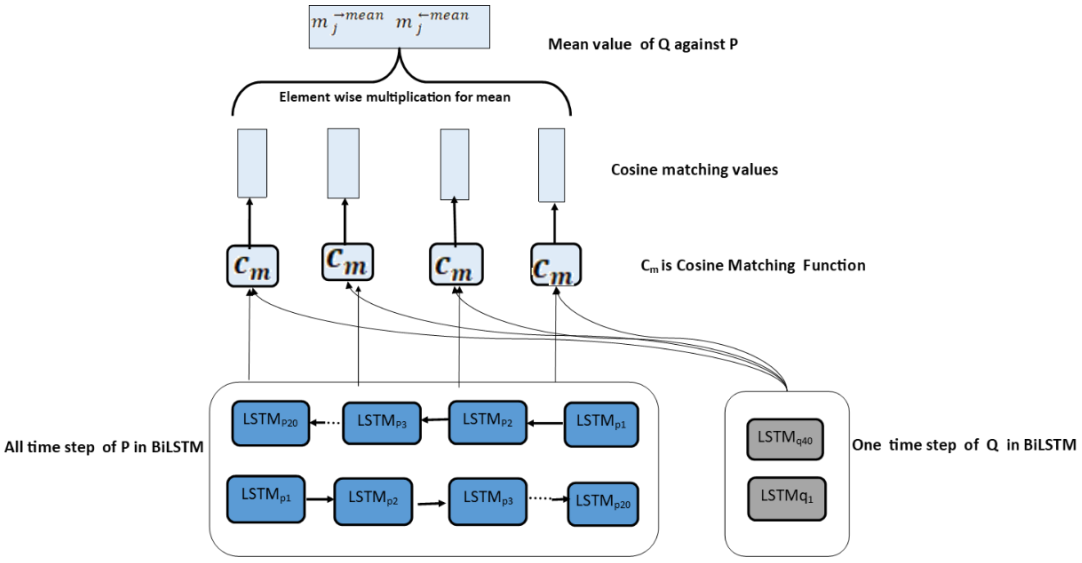

Figure 4-5: *Mean polling approach*

In Figure 4-5, the mean pooling matching strategy or approach which matches each time step of the hypothesis Q (right block) against all time step of the premise P (left block) sentence using $\boldsymbol{Cm}$ in eq. (10), and returns the forward and backward mean pooling matching result.

As in algorithm 4.5 depicts from Lines (5-19) compute the cosine similarities between each forward (or backward) contextual embedding of $\boldsymbol{Cm}$ in eq. (10) of $\boldsymbol{Q} \rightarrow \boldsymbol{P}$ and $\boldsymbol{Q} \leftarrow \boldsymbol{P}$ As we going to reverse direction $\boldsymbol{Q} \leftrightarrow \boldsymbol{P}$ each contextual embedding $\boldsymbol{h_q^{j\rightarrow}}$ and $\boldsymbol{h_q^{i\leftarrow}}$ matched with every contextual embedding of P $\boldsymbol{h_p^{j\rightarrow}}$ and $\boldsymbol{h_p^{j\leftarrow}}$ will return two vectors of the cosine matrix as follows.

$$\overrightarrow{\boldsymbol{Cm}(\boldsymbol{Q},\boldsymbol{P})} = \boldsymbol{LSTM}(\boldsymbol{h}\begin{pmatrix}\boldsymbol{j}\\ \boldsymbol{q}\end{pmatrix}^{\rightarrow}, \boldsymbol{h}\begin{pmatrix}\boldsymbol{i}\\ \boldsymbol{p}\end{pmatrix}^{\rightarrow})$$
$$\overleftarrow{\boldsymbol{Cm}(\boldsymbol{Q},\boldsymbol{P})} = \mathbf{LSTM}(\boldsymbol{h}\begin{pmatrix}j\\ q\end{pmatrix}^{\leftarrow}, \boldsymbol{h}\begin{pmatrix}i\\ p\end{pmatrix}^{\leftarrow}) \quad (16)$$

After taking the result of cosine similarity, in line (20, 36). Will preserve the average or the mean value of in each dimension as indicated in the below formula which is W, and resulted two vectors $\boldsymbol{w^1}, \boldsymbol{w^2}$.

$$m_j^{\rightarrow mean} = \frac{1}{m}\sum_{i=1}^{m} c_m \left(h\binom{j}{q}^{\rightarrow}, h\binom{i}{p}^{\rightarrow}; w^1\right)$$
$$m_j^{\leftarrow mean} = \frac{1}{m}\sum_{i=1}^{m} c_m \left(h\binom{j}{q}^{\leftarrow}, h\binom{i}{p}^{\leftarrow}; w^2\right) \quad (17)$$

b) Max-pooling Reverse Matching

This approach shown in Figure 4-6 considers only maximum feature for the contextual embedding of the sentence P and Q, and retain the highest cosine similarity vectors as the pooled vector. This max pooling will gather the strong alignment between vectors which the positive labeled entailment advantages from.

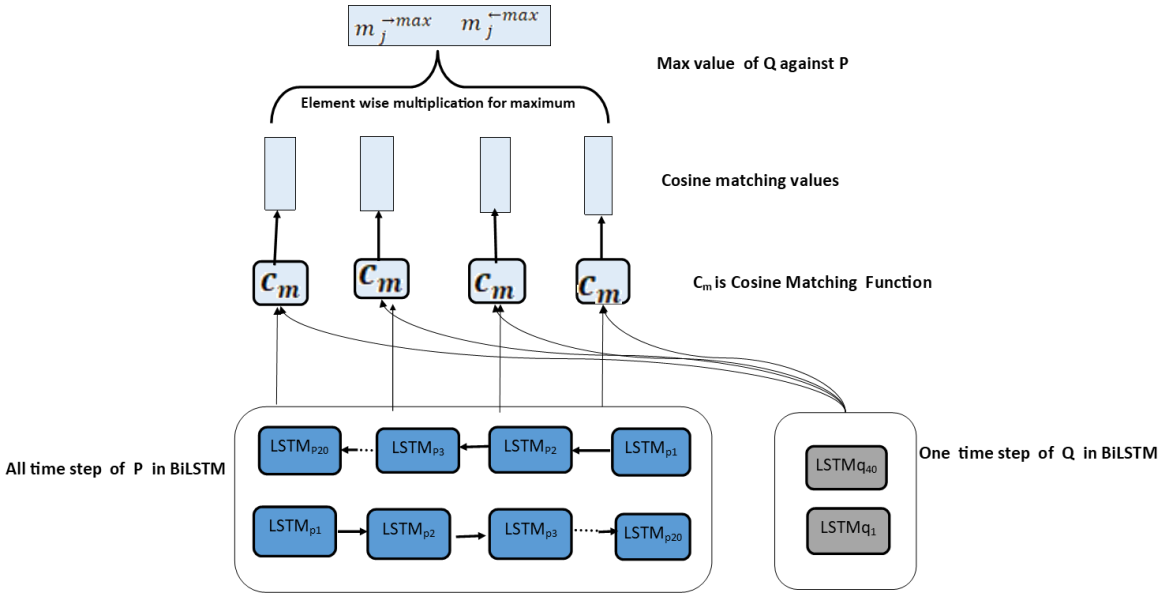

Figure 4-6: *Max polling approach*

Figure 4-6 shows the max pooling matching strategy or approach which matches one time step of the hypothesis Q (right block) against all time step of the premise P(left block) sentence using $\boldsymbol{Cm}$ in eq. (10) , and returns the forward and backward max pooling matching result.

This approach is similar with the above approach in how operate on calculating the cosine similarity and matching process. We first calculate the cosine similarities between each forward (or backward) contextual embedding of $\boldsymbol{Cm}$ in eq. (10) which is $\boldsymbol{Q} \leftrightarrow \boldsymbol{P}$ where each forward

$\boldsymbol{h_q^{j\rightarrow}}$ and backward $\boldsymbol{h_q^{j\leftarrow}}$ contextual embedding of Q matched with every forward (or backward) contextual embedding of the other sentence P as $\boldsymbol{h_p^{i\rightarrow}}$ or $\boldsymbol{h_p^{i\leftarrow}}$

$$\overrightarrow{\boldsymbol{Cm}(\boldsymbol{Q},\boldsymbol{P})} = \boldsymbol{LSTM}(\boldsymbol{h}\begin{pmatrix}\boldsymbol{j}\\ \boldsymbol{q}\end{pmatrix}^{\rightarrow}, \boldsymbol{h}\begin{pmatrix}\boldsymbol{i}\\ \boldsymbol{p}\end{pmatrix}^{\rightarrow})$$
$$\overleftarrow{\boldsymbol{Cm}(\boldsymbol{Q},\boldsymbol{P})} = \mathbf{LSTM}(\boldsymbol{h}\begin{pmatrix}j\\ q\end{pmatrix}^{\leftarrow}, \boldsymbol{h}\begin{pmatrix}i\\ p\end{pmatrix}^{\leftarrow}) \quad (18)$$

After calculating the cosine similarity, it takes the highest (maximum) cosine similarity vector and matching will be performed between each max vector to the forward direction $\boldsymbol{Q} \rightarrow \boldsymbol{P}$ and its corresponding vector to the backward direction $\boldsymbol{Q} \leftarrow \boldsymbol{P}$ and result two vectors of $\boldsymbol{w^3}, \boldsymbol{w^4}$.

$$\boldsymbol{m}_{\boldsymbol{j}}^{\rightarrow\mathbf{max}} = \boldsymbol{max}_{\boldsymbol{j}\in(\boldsymbol{M}\ldots\mathbf{1})}\boldsymbol{c_m}(\boldsymbol{h}\begin{pmatrix}\boldsymbol{j}\\ \boldsymbol{q}\end{pmatrix}^{\rightarrow}, \boldsymbol{h}\begin{pmatrix}\boldsymbol{i}\\ \boldsymbol{p}\end{pmatrix}^{\rightarrow}; \boldsymbol{w^3})$$
$$\boldsymbol{m}_{\boldsymbol{j}}^{\leftarrow\mathbf{max}} = \boldsymbol{max}_{\boldsymbol{j}\in(\boldsymbol{M}\ldots\boldsymbol{i})}\boldsymbol{c_m}(\boldsymbol{h}\begin{pmatrix}\boldsymbol{j}\\ \boldsymbol{q}\end{pmatrix}^{\leftarrow}, \boldsymbol{h}\begin{pmatrix}\boldsymbol{i}\\ \boldsymbol{p}\end{pmatrix}^{\leftarrow}; \boldsymbol{w^4}) \quad (19)$$

4.5 Aggregation Layer

This layer is designed to aggregate the two sequences of matching vectors into a fixed-length matching vector. We utilize BiLSTM model, and apply it to the two sequences of matching vectors. Then, construct the fixed-length matching vector as mentioned in Section 4.4.1.a and b by concatenating the output vectors in the architecture 4-5 and 4.5 from the last time-step of the BiLSTM models.

The mean pooling of contextual matching result of $\boldsymbol{Q} \rightarrow \boldsymbol{P}$ of both in forward and backward embedding at each time step at Q as indicated in eq. (17): $\boldsymbol{v1} = \boldsymbol{concat}(\boldsymbol{w^1}, \boldsymbol{w^2})$

The max pooling of contextual matching result of $\boldsymbol{Q} \leftarrow \boldsymbol{P}$ of both in forward and backward embedding at each time step at Q as (19): $\boldsymbol{v2} = \boldsymbol{concat}(\boldsymbol{w^3}, \boldsymbol{w^4})$

$$\boldsymbol{aggregattion}_{\boldsymbol{mean}} = \boldsymbol{BiLSTM}(\boldsymbol{v^1}) \quad (20)$$
$$\boldsymbol{aggregattion}_{\boldsymbol{max}} = \boldsymbol{BiLSTM}(\boldsymbol{v^2}) \quad (21)$$

$$\boldsymbol{aggregattion_{last} = concat(aggregattion_{mean}\, aggregattion_{max})} \quad (22)$$

The last time step of BiLSTM result will be returned from the two layers as shown in eq. (20) and (21) then the final result of the two vectors will be aggregated and return to a single vector as indicated in eq. (22), and this fixed vector will be sent to the prediction layer for the final classification.

4.6 Prediction Layer

Prediction layer is applied for producing entailment classification probabilities. The purpose of this layer is to evaluate the probability distribution. $\boldsymbol{p_r(y|P,Q)}$ Used accuracy for evaluation matrices in matched and mismatched dataset.

$$\boldsymbol{accracy} = \frac{\boldsymbol{correct\ labels} * \mathbf{100}}{\boldsymbol{total\ numers\ of\ example}} \quad (23)$$

To this end, we employ two layers feed-forward neural network to consume the fixed-length matching vector, using aggregation BiLSTM layer as explained in eq. (22), by using this vector as an input to this layer, we then used softmax(as explained in below section) function to generate the output classification probabilities. The number output class of this layer is three which are entailment, contradiction, and neutral.

4.6.1 Softmax Function

Softmax function calculates the probabilities distribution of the class label determined by greater than two class labels. In general, this function calculates the probabilities of class label over all possible target class labels.

The main advantage of using Softmax is the output probabilities range. The range will be between 0 and 1, and the sum of all the probabilities is one. Softmax function used for multi-layer classification model it returns the probabilities of each class, and the target class will be the one with the high probability.

$$\boldsymbol{p_r}(y = j\,|\theta^i) = \frac{e^{\theta^i}}{\sum_{j=0}^{k} e^{\theta_k^{(i)}}} \tag{24}$$

where y is the classification label of (P, Q)

The eq. (24) [52] computes the exponential (e-power) of the given input value, and the sum of exponential values of all the values in the inputs. Then the ratio of the exponential of the aggregated input value in BiLSTM as mentioned in section 4.5. And the sum of exponential values is the output of the softmax function.

Chapter 5: Experiment and Result

5.1 Overview

In this Chapter, the set of experiments conducted to validate the proposed deep learning based entailment classifications model are presented. The evaluation focuses on the two main components of the architecture, 1) word embedding layer - comparing two embedding layers the sub-word and skip gram, and 2) matching function. We have also compared the performance of the proposed model against baseline models.

5.2 Dataset and experimental setup for word vectors

The corpus used for training the word embedding layer is unstructured Amharic text file collected from different sources. Those text files includes, Wikipedia (wikidumps), fictions, news, history, bible with the total corpus size around 30000 KB.

FastText and Genism libraries used for training the models SW and SG experimentation and value visualization using PCA (Principal Component Analysis). While most of the Parameters of fasttext and skipgram are used, some parameters are fine-tuned (as mentioned in chapter 4 section 4.2) with the same initialization to differentiate between the two models. Experimental result between the two models presented in the following sections.

5.3 Prototype

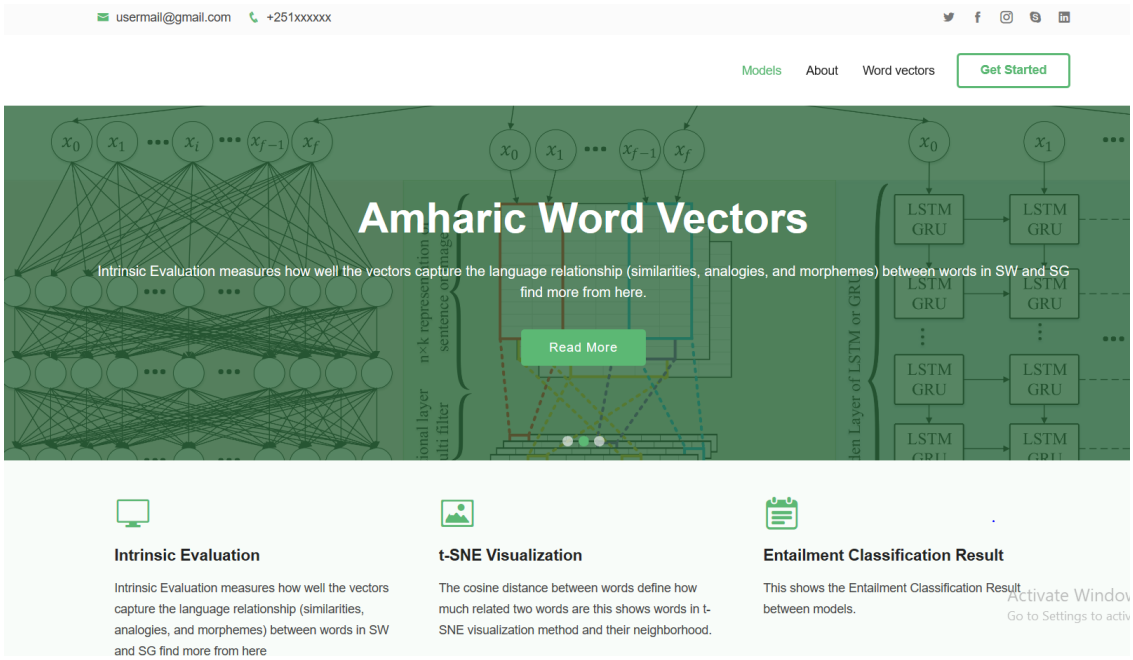

Figure 5-1*: Home Page for ATEC*

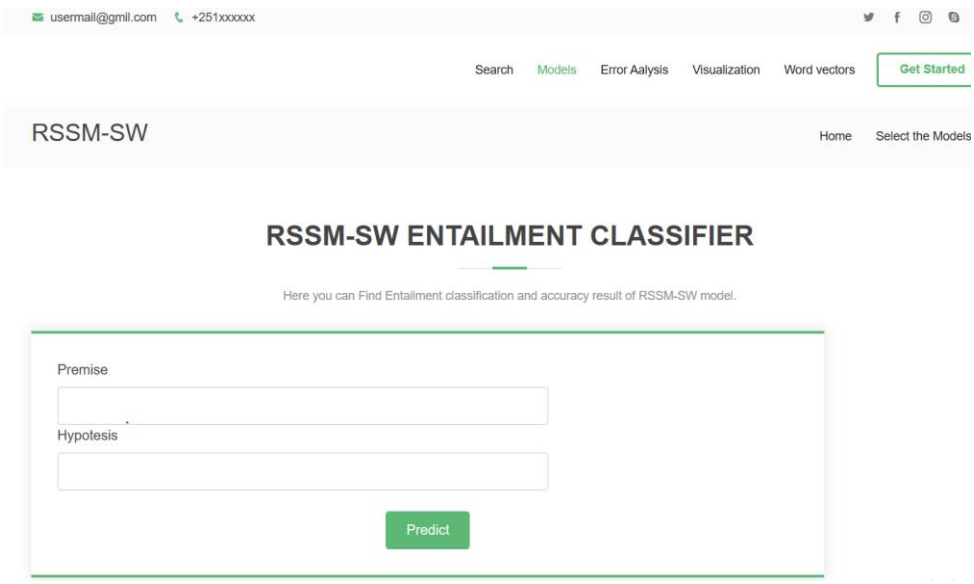

Figure 5-2: *Posting box for the entailment classification from the selected model RSSM*

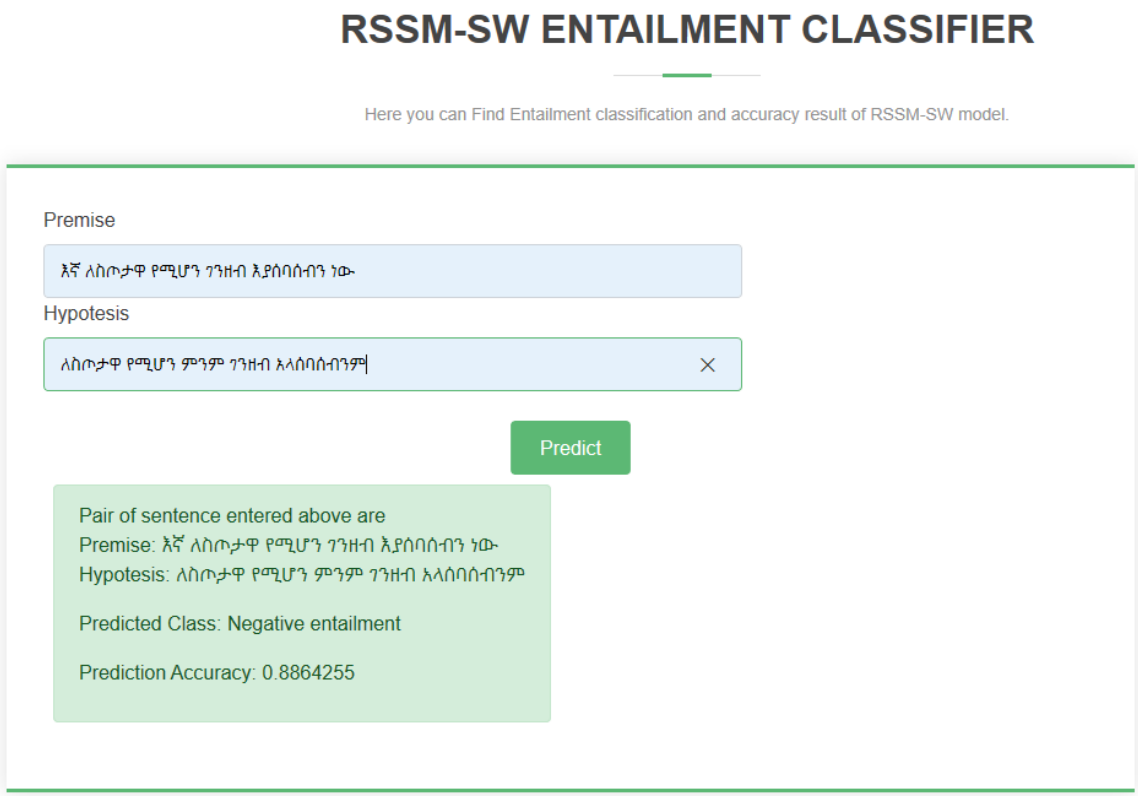

Figure 5-3: *Result of the RSSM classifier*

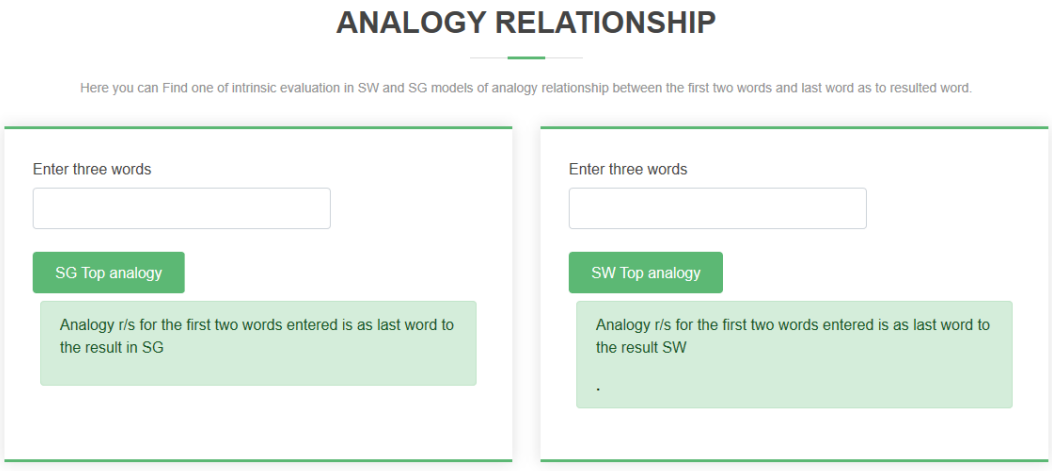

Figure 5-4: *Intrinsic evaluation input box of analogy relationship of words in SWE and SGE*

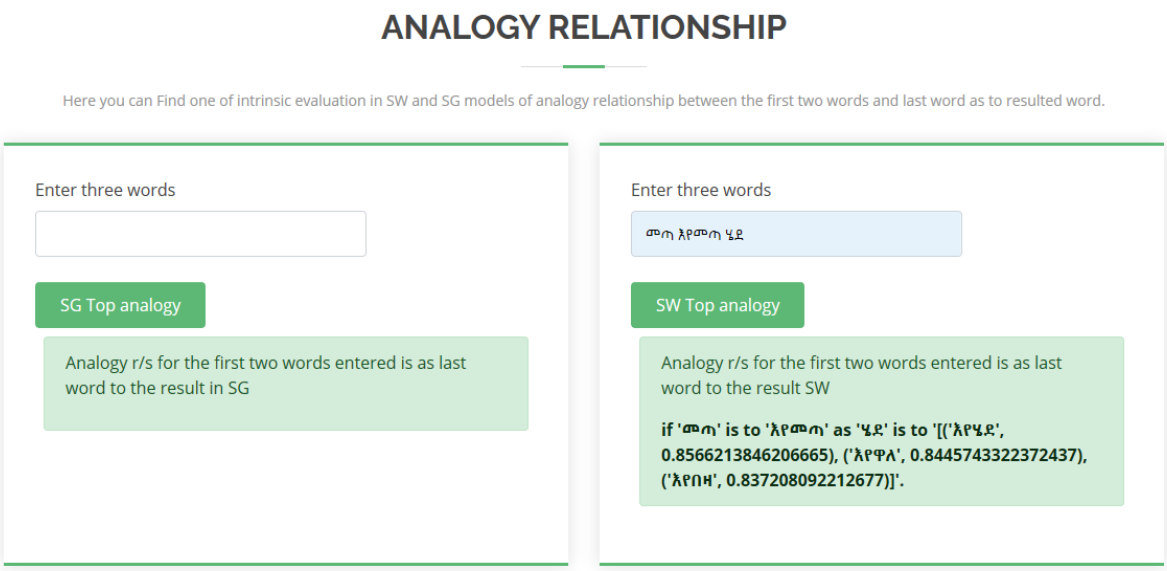

Figure 5-5: *Intrinsic evaluation result analogy relationship of words in SWE*

In the above figures depicted some of the interfaces from the many. Figure 5-1 shows the home page, which one can find further in the page as the entailment classifier input box, comparison of models, and word vector intrinsic evaluation. Figure 5-2. Shows the input box for posting the premise and hypothesis and make the prediction in RSSM model. Figure 5-3 displays the predicted class of the entered sentences. Figure 5-4. Displays one of the intrinsic evaluation of word vectors analogy relationships between SWE and SGE model, and the result of the analogy relationship of SWE showed in figure 5-5

5.4 Word Embedding Evaluation Metrics

The input (the first layer) for the classification task is the pre trained word vectors which are trained on sub word (SW) embeddings and skip gram (SG) models. To identify which embedding contributes more for Amharic language features, we evaluated the two models with Intrinsic and Extrinsic evaluation methods [19]. 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, 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 for supervised machine learning algorithms used in other downstream NLP tasks, in our case entailment Classification.

5.4.1 Intrinsic Evaluation

The purpose is to understand to what extent our approach is capable of learning from the structures of Amharic language. It measures how well the vectors capture the language relationship (similarities, analogies, and morphemes) between words. This task is used to measure the quality of a word vector directly using different measurements. We followed the of knowledge of language experts judgment we prepared 4 questioners attached in annexes using Amharic language features as key criterion presented in the next sub section of intrinsic evaluation of SG and SW model.

a) Word Similarities and Relatedness.

This task involves finding list of related words to a given query word. Most of Amharic words are morphologically inflicted or compound words, and these words must be treated the same as atomic. We evaluated with the two embeddings models from the expert judgment answers in the questionnaire that attached in annex A to find out which word vector model represents Amharic words syntactic feature better. And we selected the top 7 out of 19 words and the retrieved related words of the SW and SG vectors as shown in Table 5.1. And Table 5.2.

Table 5.1: *Similarity result for SW*

Terms	***SW Model Generated top 5 similar vectors with the terms given***				
መንፈስ	መንፈስስ	መንፈስህ	መንፈስም	መንፈስን	መንፈሱ
ኢየሱስ	በኢየሱስ	ስለኢየሱስ	ከኢየሱስ	ኢየሱስስ	ኢየሱስን
ድካም	ለድካም	ድካምና	በድካም	ድካሜ	ከድካም
ልዑል	በልዑልም	ለግርማዊ	ከልዑል	የልዑል	ልዑልስ
ዘላለም	ዘላለማዊ	ዘላለሙን	ለዘላለም	ዘላለሜ	ለዘላለሙ
ህይወት	ለሕይወት	ከሕይወት	ሕይወትና	ሕይወት	የሕይወት
እርግማን	ለእርግማን	እርግማንና	የእርግማን	ከእርግማኑ	እርግማንስ

Table 5.2: *Similarity result for SG*

Terms	***SG Generated Top 5 similar vectors with the terms given***				
መንፈስ	በመንፈስ	ደስታውን	የመፍራት	አሠቃየው	ቅዱስን
ኢየሱስ	ክርስቶስ	እያስተማራችኋቸው	እያጠመቃችኋቸው	በኢየሱስ	ከኢየሱስ
ድካም	ህልም	ለመገመት	ለወደፊቱ	ብስጭት	የሚቀር
ልዑል	ግርማዊ	ክቡር	ሃይማኖትን	ወራሽ	ለግርማዊ
ዘላለም	ከዘላለም	ከትውልድ	የጸና	ዘለዓለም	ይኖራልና
ህይወት	ህይወትማ	ከምትጣል	ቁርጠህ	ገሃነመ	ኖሮህ
ሰው	ሰውን	ሰውም	ይገደል	ተለይቶ	ወንድሙን

To find out how the two models retrieved words matched with the language expert's judgment in annex B we calculated Precision and recall using eq. (25) and (26) as shown in table 5.3 and table 5.4,. From the results we found out SW embedding are better in grasping the syntactic (morpheme) feature of Amharic words than SG as both the precision and recall is score higher in SW model than SG.

We measure the completeness of the query set *R* (recall) and how accurate is the answer set *P* (precision). The expression of recall and precision are as follow:

$$R = \frac{number\ of\ extracted\ word\ equvalent\ with\ experts\ answer}{number\ of\ expert\ answer\ for\ a\ given\ word} \quad (25)$$

$$P = \frac{number\ of\ extracted\ word\ equvalent\ with\ experts\ answer}{number\ of\ all\ answers\ retrived\ from\ our\ sysetm} \quad (26)$$

The detail computation of precision and recall is presented in the Tables 5.3 and 5.4. The table header consists of abbreviation, the description of those abbreviations is:

- LEA: Total number of Language Expert Answer (taken as relevant)
- RSWE : Total number of word retrieved by sub word Embedding
- RSGE : Total number of word retrieved by Skip Gram Embedding
- RSWMEA: Retrieved sub word model words which match with expert answer

- RSGMEA: Retrieved skip gram model words which match with expert answer

Table 5.3: *Precision and recall of SWE model*

Query Words	LEA	RSWE	RSWMEA	R	P
መንፈስ	3	5	3	1	0.6
ኢየሱስ	3	5	3	1	0.6
ድካም	3	5	1	0.33	0.2
ልዑል	3	5	2	0.66	0.4
ዘላለም	3	5	3	1	0.6
ህይወት	3	5	3	1	0.6
እርግማን	3	5	3	1	0.6
Average SWM				**0.86**	**0.51**

Table 5.4: *Precision and recall of SGE model*

Query Words	LEA	RSGE	RSGMEA	R	P
መንፈስ	3	5	1	0.33	0.2
ኢየሱስ	3	5	2	0.6	0.4
ድካም	3	5	0	0	0.2
ልዑል	3	5	1	0.33	0.2
ዘላለም	3	5	1	0.33	0.2
ህይወት	3	5	1	0.33	0.2
ሰው	3	5	2	0.6	0.4
Average SGM				**0.36**	**0.3**

b) Similarity score

Similarity score of the two models can be measured using cosine distance between words in margin between 0 and 1. The similarity score close to 1 shows higher similarity whereas when it is nears 0 it shows the dissimilarity of words. For evaluating the similarity score that the two model results we provide a questioner to the language experts as attached in annex C with 20 words. Table 5.5, shows the top sample from the questioner that shows the SW and SG models measure the similarity score.

Table 5.5*: Similarity score result using SG and SW approach*

Word pair		Similarity score	
W_1	**W_2**	**SW**	**SG**
ክርስቶስ	ኢየሱስ	0.80	0.80
በኢየሱስ	በክርስቶስ	0.86	0.85
እግዚአብሔር	አምላክ	0.71	0.57
ሐዋርያ	ጳውሎስ	0.75	0.62
ጳውሎስ	ኢየሱስ	0.80	0.80
ህይወት	ሞት	0.34	0.30
ጸጋ	ሰላም	0.75	0.68
ወንጌል	የምስራች	0.62	0.56
የክርስቶስም	የጌታም	0.80	0.79
አዳም	ኢየሱስ	0.10	0.32

For evaluation we calculated (Spearman or Pearson) to say well related. The correlation calculation is done using Spearman's correlation coefficient. The Spearman rank correlation coefficient, was proposed as a measure of the strength of the associations (how they relate) between two variables [49, 52]. We computed the Spearman's correlation coefficients between the two embedding model and language expert judgment mean score using IBM SPSS statistics software (Version 23). The computation result is shown in table 5.6. And 5.7

Table 5.6: *Correlation result between language expert and SW model*

Correlations

		LEA	RSW
LEA	Pearson Correlation	1	.907**
	Sig. (2-tailed)		.000
	N	20	20
RSW	Pearson Correlation	.907**	1
	Sig. (2-tailed)	.000	
	N	20	20

**. Correlation is significant at the 0.01 level (2-tailed).

Table 5.7: *Correlation result between language expert and SG model*

Correlations

		LEA	RSG
LEA	Pearson Correlation	1	.829**
	Sig. (2-tailed)		.000
	N	20	20
RSG	Pearson Correlation	.829**	1
	Sig. (2-tailed)	.000	
	N	20	20

**. Correlation is significant at the 0.01 level (2-tailed).

As of the answer found from the language experts attached in annex D the Spearman correlation coefficient shown in table 5.6, and 5.7. Rs, can take values from +1 to -1. In this computation, there appears to be a positive correlation Rs value (+0.90) and (+0.82) in both models. Rs of +1 indicates a perfect association of ranks. Rs value 0 of indicates no association between given compression, and Rs of -1 indicates a perfect negative association. The closer Rs is to zero, the weaker the association between the ranks. Since we achieved Rs of (+0.90) and (+0.82) it indicates there is a positive correlation of our relatedness assessment (the expert's judgment score and the two word vector models value are much more related) but with the higher positive correlation in SW model as it score (0.90) than SG (+82).

c) One Odd Word out Detection

The other experiment conducted is identifying the model that is capable enough to filter odd word out or the word that doesn't belong to given word chunks cluster. In the table 5.6 we took 8 out of 14 of the samples from questionnaire attached in annex E. this questioner is given to the language experts to filter one odd out from the word chunks.

Table 5.6: *Odd word out for SW and SG example.*

Words chunks	SW odd one out	SG odd one out
ሙሴ ኢየሱስ ቆሮንጦስ ዮሐንስ ጳውሎስ	ሙሴ	ዮሐንስ
ዮሐንስ ይሁዳ ጴጥሮስ ኢየሱስ	ይሁዳ	ማርያም
አክስት ኣጎት አባት	አባት	አባት
እህት ወንድም አናት	አናት	አናት
ንጉሥ ልዑል ንግሥት መምህር	መምህር	ንጉሥ
እሱ የእሱን እነርሱ	እነርሱ	እሱ
ጋዜጣ ራዲዮ ቴሌቪዥን ዜና	ዜና	ዜና
ፊልም ድራማ ተማሪ	ተማሪ	ተማሪ

As the language judgments result that attached in annex F we evaluated in what extent the two models retrieved the correct result by calculating Precision using eq. (26) in table 5.7, as can be seen in the result of precision SW has higher precision than SG. SW can easily pick words that do not belong to a list (either semantically, syntactically, or morphologically)

The table header consists of abbreviation, the description of those abbreviations is:

- PSW: Precision result of Sub word Model
- PSG: Precision result of Skip gram model

Table 5.7: *The OWO Precision result SWE and SGE.*

Intrinsic evaluation	**Given Sample**	**LEA**	**RSWMEA**	**RSWMEA**	**PSW**	**PSG**
One odd word out	14	14	12	9	**0.86**	**0.64**

Even if SG does filter out the odd word in some label as table 5.6, shows, it didn't identify dissimilarity of the given chunks as SW do. For example, ሙሴ is found mostly on Old Testament compared to the New Testament, መምህር semantics different from the royalty names, and እነርሱ different in semantic of the given words እሱ የእሱን in which SG failed selecting in this specific query as an odd.

d) Word analogy relation.

Word analogy relation is a way of examining and evaluating the quality and goodness of a word embedding. For example, according to [19], the association “if man is to king, woman is to what?” is an analogy that a word embedding to solve. The answer is queen which is logically correct. In this example, a tuple like ወንድ : ንጉሥ :: ሴት : ንግሥት is generated the embedding model should produce correct results. In word analogies, the task is to find a vector v such that vector (v) is closest to vector (ወንድ) - vector (ንጉሥ) + vector (ሴት) using cosine similarity.

For evaluating the Analogical relationship task of the two models we prepared 14 relationship type questioner for language expert as attached in annex G to give their judgment and compare with the two word vectors model SW and SG.. The given Analogical reasoning task has two categories: the semantic and syntactic analogies with respect of relatedness criteria. In the Table 5.8 and Table 5.9, we selected 4 for each semantically and syntactically related word of the two models. As showed in the tables word pair 1 column gives an example in making the semantic relationship, and when the first term in Word pair2 column is given it finds the second term having same relatedness type with Words pair 1 column. As ወንድ is to ንጉሥ ሴት is to what? And returns the word ንግሥት

Table 5.8: *Semantic analogy of SW embedding*

Relatedness type	**Words pair 1**	**Word pair 2**
Title relation	ወንድ is to ንጉሥ	ሴት is to ንግሥት
Opposite relation	አጭር is to ረዥም	ጥቁር' is to ነጭ
compound words	ቤት is to ቤተመንግስት	ህግ is to ሕገመንግስት
Sex relation	ሴት is to እናት	ወንድ is to አባት

Table 5.9: *Semantic analogy of in SG embedding*

Relatedness	Words pair 1	Word pair 2
Title relation	ወንድ is to ንጉሥ	ሴት is to በይሁዳ
Opposite relation	አጭር is to ረዥም	ጥቁር' is to ጠይም
compound words	ቤት is to ቤተመንግስት	ህግ is to ለገንዘብና
Sex relation	ሴት is to እናት	ወንድ is to ወንድም

In addition, we have tested syntactic analogy as shown in Table 5.10 and Table 5.11. Again, SW in Table 5.10 finds the analogies as the example in the given relatedness criteria better compared to SG shown in Table 5.11.

Table 5.10: *Syntactic analogy of in SW embedding*

Relatedness	**Words pair 1**	**Words pair 2**
Possessives in sex	'እሱ' is to 'የእሱን'	እሷ' is to የስዋን
Plural affixes	አባት is to አባቶች	እናት is to እናቶች
Passive voice affixes	ገደለ' is to 'ተገደለ'	'ሰማ' is to ያሰማ
Pronoun/verb affixes	'መጣ' is to 'እየመጣ	ሄደ is to እየሄደ

Table 5.11: *Syntactic analogy of in SG embedding*

Relatedness	**Words pair 1**	**Words pair 2**
Possessives in sex	'እሱ' is to 'የእሱን'	እሷ' is to አልቻልኩም
Plural affixes	አባት is to አባቶች	እናት is to ውርስ
Passive voice affixes	ገደለ' is to 'ተገደለ'	'ሰማ' is to እንዲናገር
Pronoun/verb affixes	'መጣ' is to 'እየመጣ	ሄደ is to ርቆ

As the language expert judgments result that attached in annex H we evaluated how the two models retrieved the correct semantic and syntactic analogies calculated the Precision using eq.(26) as shown in the table 5.12 the precision value find out the in what level Expert judgment answer are matched with the retrieved words of the two models.

Table 5.12: *Word analogy relationship Precision result SWE and SGE.*

Intrinsic Evaluation	**Given sample**	**LEA**	**RSWMEA**	**RSWMEA**	**PSW**	**PSG**
Analogy semantic	8	8	5	2	**0.62**	**0.25**
Analogy syntactic	8	8	6	1	**0.75**	**0.125**

The result of the analogy relationship precision indicates SW embedding much better than the SG model. In addition comparing the content of Table 5.8 and 5.9 one can see that SW (Table 5.8) provide more intuitive analogy detection compared to the result in Table 5.9. Which are the wrong in terms of meaning.

And in Syntactic relationship as Amharic language is highly influenced by the word morphology and SG works poorly on the morphology of the words, and couldn't find any of the corresponding analogy relationship in the given related criteria terms as SW does.

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. Within the following words in table 5.13, let's see the visualization of PCA (Principal component Analysis) for syntactic and semantic relatedness. In table 5.13 the word analogy relationship written in bold is the example, based on that we gave the third word written in italic for both models SW and SG and retrieved the related word.

Table 5.13: *Word analogy for PCA visualization*

Word analogy SW model	***Word analogy SG model***
If 'እሱ' is to 'የእሱን'	**'እሱ' is to 'የእሱን'**
አንቺ is to አንቺን	*አንቺ* is to *ነሽ*
እኛ is to የራሳችንን	*እኛ* is to *ራሳችን*
እሷ is to የሰዋን	*እሷ* is to *አልቻልኩም*
እሱ is to የእርሱን	*እሱ* is to *የማዳን*
If 'ሴት' is to 'እናት	**If 'ሴት' is to 'እናት**
ወንድ is to *አባት*	*ወንድ* is to *ወንድም*
ሴት is to *እናትና*	*ሴት* is to *እህት*
ሕገመንግስት is to *ቤተመንግስት*	*ሕገመንግስት* is to *ሕገመንግሥት*
If 'መጣ' is to 'እየመጣ'	**If 'መጣ' is to 'እየመጣ'**
ሄደ is to *እየሄደ*	*ሄደ* is to *ርቆ*
ፈለገ is to *እየፈለገ*	*ፈለገ* is to *መቆም*
መጣ is to *እየመጣሀ*	*መጣ* is to የሚያልፍ
ፈራ is to *እየፈራ*	*ፈራ* is to *መንፈሱ*
ገባ is to *እየገባ*	*ገባ* is to *ሰከሮ*

Figure 5-6 and 5-7 Illustrate PCA visualization of analogy relationship using both SW and SG showed in table 5.13, respectively. The distance between terms in the example as 'መጣ' to 'እየመጣ' and 'ሴት' to 'እናት results to be in the same distance, as these terms have the same affixes or sematic relationship, and hence have the same distance in visualization of SW; while in SG loses the cosine distance as the model couldn't grab the relationship terms.

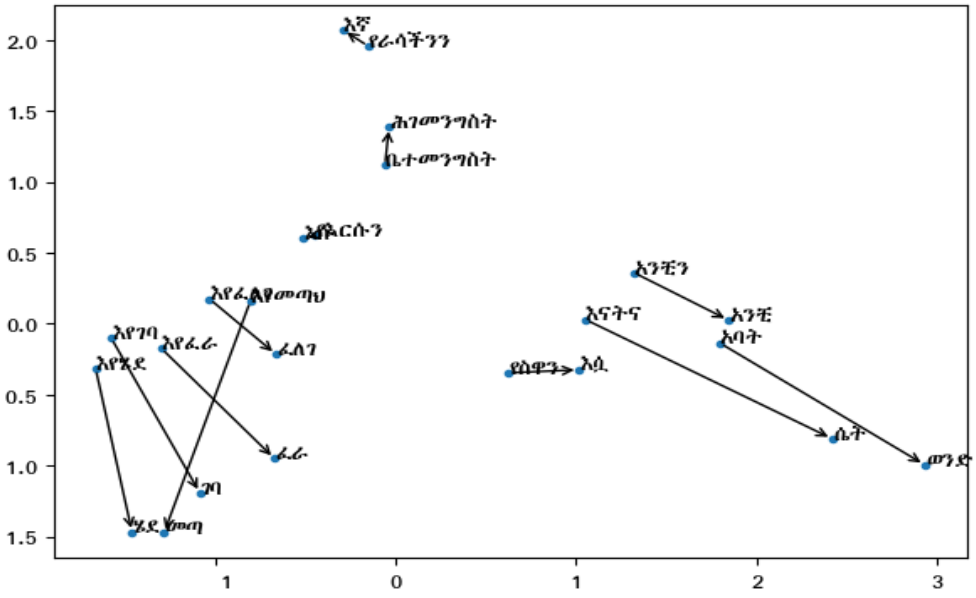

Figure 5-6: *PCA Visualization semantic and syntactic analogy relationship for SW*

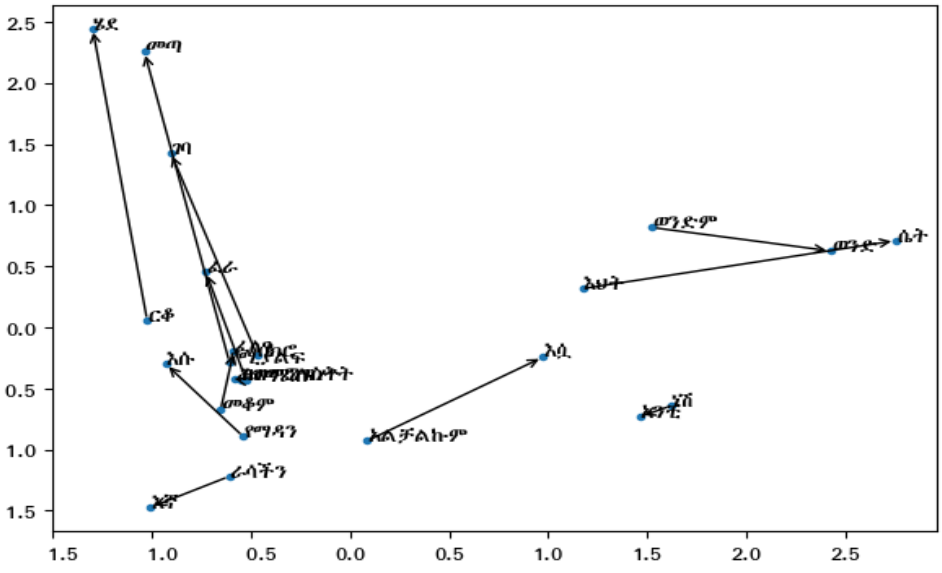

Figure 5-7: *PCA Visualization semantic and syntactic relatedness for SG*

The result of intrinsic evaluation such as word analogy is extremely biased by the quality and quantity of the corpus used during the training [19]. Though in our word vectors in SW model with minimal corpus used for training exhibited better intrinsic evaluation results in perspective of language expert judgment as discussed above (semantic and syntactic or analogy, similarity and relatedness of words , similarity score, odd word out evaluations). Contrasting from SG model, specifically performs poorly in similarity relatedness and analogy relationship evaluation.

5.4.2 Extrinsic Evaluation of Word Vectors

To analyze the word vectors in performance for the downstream entailment classification task, we experimented the two embedding models (SG and SW) by adding only LSTM sentence embedding layer on the top of the word embeddings then feed it to the classifier. We used accuracy as evaluation measurement to find out the correct label for the given examples as follows.

$$accracy = \frac{correct\ labels * 100}{total\ numers\ of\ example}$$

Table 5.14: *Training and test accuracy of SG and SW*

Model	**Training accuracy**	**Test accuracy**
SGE	**67%**	**65%**
SWE	**70%**	**68%**

As Amharic language is rich in word morphology, the downstream task of entailment classification can be advantageous by leveraging the extension sub word information to the word vectors. Compared to SG, SW showed an improvement of 3% during training and 2% in test accuracy. Thus, we choose SW embedding for the classifier.

5.4.3 Dataset and Experimental setup for RSSM model

In the sentence embedding layer structured dataset annotated with three class labels of entailment classification are used. The dataset we took from SNLI is (Stanford natural language inferences training dataset) 8700 of the development pair of sentences written in English and translated into Amharic. This translated sentences pairs manually labeled for balanced classification with the entailment, contradiction, and neutral, supporting the task of Amharic natural language inference

(NLI). The SNLI dataset annotated to serve both as a benchmark for evaluating representational systems for text [1] as well as a resource for developing NLP models of any kind.
For the purpose of validating the sentence classification task, among the 8700 pair of translated sentences we used 80% of the dataset to train model and the 10% to Validate and the remaining 10% to test. We initialize word embeddings randomly which is pertained in fast text. We do not update the pre-trained word embeddings during training.

During training, both the training and the validation sets are loaded into the RAM. Dataset is shuffled and batches are loaded into GPU memory for processing. The objective of the training is to minimize the total loss and increase the accuracy of model. The model is trained in epochs, where the model sees all the input data at least once. During each epoch, batches of size 10 are loaded into the model and evaluated, calculating a batch loss for each sample. The gradients from the batch are back propagated into the previous layers to improve them. We set size 100 for sentence embedding and aggregation BiLSTM layers. Apply dropout to every layer and set dropout allocation as 0.2. To train the model, we minimize the Cross entropy of training set, and use the ADAM optimizer to update parameters. Set the learning rate as 0.01. The training done on laptop NIVIDIA GEFORCE 960GTX with GPU processor for making the training fast. To train the model in 8 epochs, with sentence length of 20 it took an average of 30 minute of total time. We ablated and extend the model for compression other models and also to demonstrate the performance the added approaches. These model experiments will be presented in the next sections.

5.4.4 Evaluation Metrics

Our evaluation consists of three features. First, we tested training accuracy, second standard in-domain evaluation in which the training and test data are drawn from the same sources (validation accuracy), and third a cross-domain evaluation in which the training and test data differ substantially (test Accuracy).

The training and validation accuracy are shown in the Figure 5-8. It can be seen that the training accuracy keeps increasing until 78%, while the validation accuracy rises to 71% in 6-th epoch and nearly stops increasing after that. We stopped the training on the 7-th epoch as the validation accuracy stops increasing, which is also a way to prevent from overfitting.

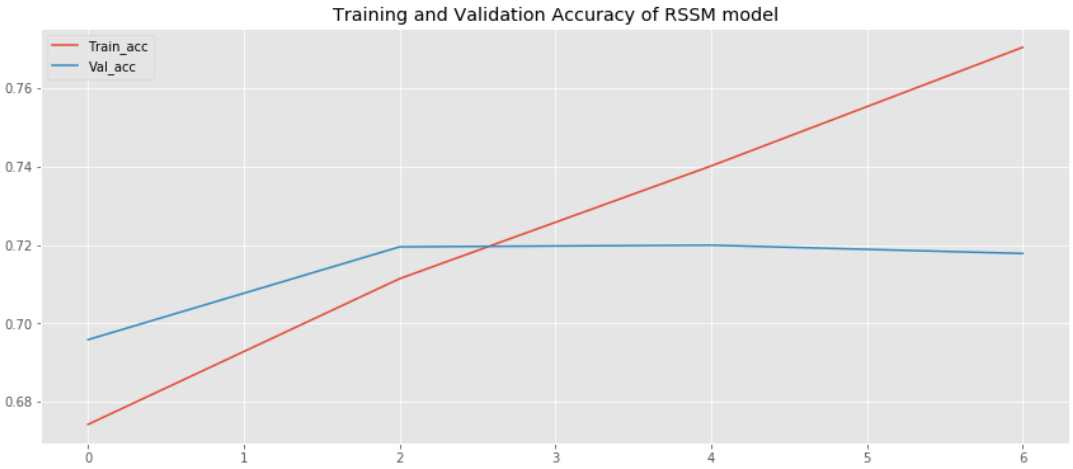

Figure 5-8: *Training and validation accuracy during training*

5.4.5 Ablations of RSSM model

As mentioned in above section to demonstrate the effectiveness RSSM model ablated the full model (the full model constructed with SW embedding followed by BiLSTM context embedding and on the top of matching function adding matching approach of mean and max pooling in the reverse directions as Q->P) in using SW and SG on word embedding layer, with the same approach we follow to build the full model. Second with only matching function of cosine similarity. Third with matching function followed by only mean, only max, and both of pooling approaches. The third one is to the matching direction of only (P->Q), both (P->Q and Q->P) and with that of our model (Q->P) direction (which is the full model) as depicted on Table 5.13.

Table 5.15: *Model ablation result*

Model	**Training accuracy**	**Matched(val) accuracy**	**Mismatched(test) Accuracy**
Without matching approach (mean and max)	74	70.3	68.0
Only max pooling	74	70.3	69.6
Only mean pooling	75	70.6	70.4
Only P->Q	77	71.8	69.5
P->Q and Q->P	77	72.9	70.6
Q->P	**77**	**71.9**	**72.1**

From the above Table 5.16, first, without using one of a matching approach scores minimum among all models, but adding one of the approaches as max pooling increases the accuracy on mismatched dataset by +1.6%. Second, only using max pooling do not perform better as compared to only mean pooling approach. From this, we can comprehend that the mean approach has a better effect in the model performance learning useful alignment during sentence comparison. Third matching only in the left direction P->Q scores 69% as shown in test accuracy of the evaluation results. In contrary to that matching in the reverse direction which matches the hypothesis against premises helps the classifier to extract the correct result by pushing the mismatched dataset accuracy to 72. With the extended RSSM model matching direction as bidirectional P->Q and P<-Q (left and right) which scores 70.6 in the test accuracy. This indicates that matching sentence bilaterally doesn't help the model to improve the model performance as much as matching Q->P, but it is better than P->Q only.

5.4.6 Model Compassions

RSSM model follows an approach of compare and aggregate to identify the model performance. In this part, several experiments were conduct to compare our model against some of the models that discussed in the related works chapter 3.2 by using the same dataset for training our classifier model. The RSSM and the comparisons model followed approach in each layer is summarized in table 5.16.

Table 5.16: *Model Compassions approach*

Model name	**Word embedding**	**Sentence embedding**	**Matching approach**	**Matching Direction**
RSSM (SW) (Our model with SW word embedding)	SWE	BiLSTM	max and mean pooling MA	P<-Q
RSSM (SG) Our model with SG word embedding)	SGE	BiLSTM	max and mean pooling MA	P<-Q
WBWAM (Word by word attention matching of **[42]) [27]**	SGE	BiLSTM	W/O MA	P->Q

CAM (compare and aggregate matching of **[32]**) **[41]**	SGE	BiLSTM with attention	W/O MA	P->Q
BIMPM (Bilateral multi perspective matching of) **[50]**	SGE	BiLSTM with attention	W/O MP	P->Q and P<-Q

Table 5.17: *Compared models and RSSM result.*

No	Models	Train accuracy	Validation accuracy	Test accuracy
1	WBWAM	74	69	68
2	CAA	74	70	69
3	BIMPM	75	70.8	70
4	RSSM(SG)	76	70.7	69.8
5	**RSSM(SW)**	**77**	**71.9**	**72.2**

Table 5.17 shows the result of the evaluating the different models. The first framework word by word attention (WBWAM) in [27] only matches vector representation of smaller units first (words) and matching result are aggregated by LSTM or CNN to make a final decision. This kind of approaches ignores granular matching which is phrase against sentence, this is the reason why it scores minimum accuracy in all evaluation dataset as shown in Table 5.17

The second model in table 5.17 [41] multi -perspective context matching works by comparing two vectors each representing an entire sequence. Then the matching results are aggregated (by a CNN or a LSTM) into a vector to make the final decision. The matching direction performed is P->Q from this model we can observe that not using a matching approaches and matching sentence in eft direction doesn't help the model to improve, which leads the results the test accuracy down to 3% as compared to our RSSM model.

The Third approach in table 5.17, bilateral multi perspective matching [52], is based on compare and aggregate model which uses matching function followed by matching approach max pooling and perform a bilateral matching as P->Q and Q->P. Even though the model brought a comparable

result, on the training and validation (matched) accuracy, encountered 2% less on test (mismatched) accuracy with that of RSSM model.

All the above three models use a word vector in CWOB and SG in word embedding layer. However, those models of embedding layer didn't help much in representing Amharic words.

The forth in table 5.17, RSSM model with SG model for word embedding layer decreases the model performance by 2 %. Hence, adding sub-word information to the word vectors benefits the Amharic language in learning better word representation. Using matching approach as max pooling only didn't get the necessary alignment between sentences that indicates the improvement when we added a mean pooling.

Matching bilateral the premise against hypothesis and the hypothesis against the premise didn't progresses the model. However, matching only in reverse (which shown in No five in table 5.17) the hypothesis against the premises improves the model, which confirms with our initial claim for Amharic natural language inference, matching the hypothesis against the premise is more effective than the other way around or both ways around.

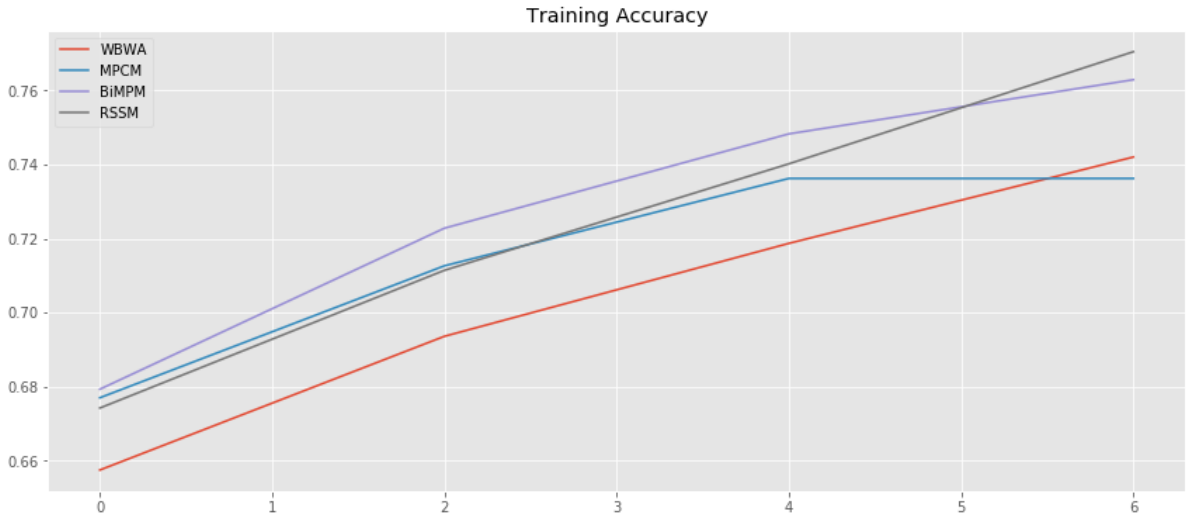

Figure 5-9: *Training Accuracy comparison between models*

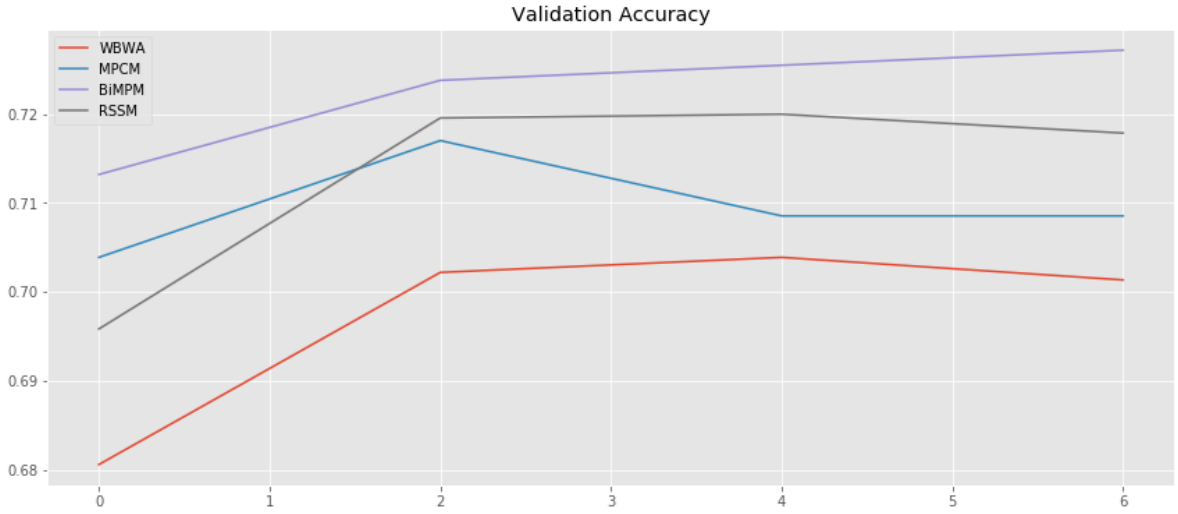

Figure 5-10: *Validation Accuracy comparison between models*

Figure 5-9 & Figure 5-10 shows the result of training and validation accuracies as we evaluating the comparison model specified in above.

5.4.7 Error Analysis between Models

One characteristics of natural language is the different ways to communicate a notion/ concept. A text may have several meanings and different texts may have the same meaning. For example,

given the two the sentences "*ኢትዮጵያ ከዚህ የተሻለ ማድረግ ትችላለች*" and "ኢትዮጵያ ከዚህ የተሻለ ማድረግ አትችልም", there is a need to say that though sentences have similar syntactic structure they are opposite/ contradictory in meaning.

In order to do that, we have conducted experiments for error analysis based on Amharic sentence-types by considering the test (Mismatched) dataset. As presented in Table 5.16. The RSSM model made correct result for the majority of the samples (consider the gray color – correct whereas no color shows wrong result) within the challenges Amharic sentence, words, and the limitation the dataset we have.

Table 5.18: *Error Analysis Result*

Description	Model	Predicted label	Actual label
CONDITIONAL Sample: (mismatched-neutral) Premises = ["የቤተሰብ ቀን በባሕሩ ዳርቻ ላይ"] Hypothesis = ["ምንም እንኳን የቤተሰብ ቀን ቢሆንም ብዙ ሰዎች በባሕሩ ዳርቻ ላይ አሉ"]	BIMPM	Neutral	Neutral
	RSSM(SG)	Neutral	
	RSSM(SW)	Neutral	
ACTIVE/PASSIVE Sample: (mismatched-Contradiction) Premises = ["*ፖ*ሊሱ ሰውየውን *መ*ታው"] Hypothesis = ["ሴትየዋ የተመታችው በፖሊሱ ነው"]	BIMPM	Entailment	Contradiction
	RSSM(SG)	Entailment	
	RSSM(SW)	Contradiction	
PARAPHRASE Sample: (mismatched-entailment) Premises =["አንድ ሰው ስልኩን እያየ ነው"] Hypothesis =["ሰውየው ስልኩን እየተመለከተ ነው"]	BIMPM	Neutral	Entailment
	RSSM(SG)	Neutral	
	RSSM(SW)	Entailment	
COREF Sample: (mismatched-entailment) Premises =["እኔ እና አንቺ የበነ አድራጎት አባል ነን"] Hypothesis =["እኛ የበነ አድራጎት አባል እንደሆንን ይታወቃል"]	BIMPM	Neutral	Entailment
	RSSM(SG)	Neutral	
	RSSM(SW)	Neutral	
QUANTIFIER Sample: (mismatched-contradiction) Premises =["እኛ ለስጦታዋ የሚሆን ገንዘብ እያሰባሰብን ነው"] Hypothesis =["ለስጦታዋ የሚሆን ምንም ገንዘብ አላሰባሰብንም"]	BIMPM	Neutral	Contradiction
	RSSM(SG)	Neutral	
	RSSM(SW)	Contradiction	
NEGATION Sample: (mismatched-contradiction) Premises = ["ሁለት ሴቶች በአንድ ምግብ ቤት ውስጥ አይደሉም"] Hypothesis = ["ሁለት ሴቶች ምግብ ቤት ውስጥ አየበሉ ናቸው"]	BIMPM	Entailment	Contradiction
	RSSM(SG)	Entailment	
	RSSM(SW)	Contradiction	
ANTONYMS	BIMPM	Neutral	Contradiction

Description	Model	Predicted label	Actual label
Sample: (mismatched-contradiction) Premises = ["ወንድ እና ሴት ጠረጴዛ አጠገብ ተቀምጠው ወይን እየጠጡ ነው"] Hypothesis = ["ሁለት አልኮል ማይጠጡ ሰዎች ለእራት እይተዘጋጁ ነው"]	RSSM(SG)	Contradiction	
	RSSM(SW)	Contradiction	
TENSE_DIFFERENCE Sample: (mismatched-entailment) Premises = ["የተወሰኑ ህጻናት የደህንነት ጠባቂውን እየሰሙ ነው"] Hypothesis = ["ልጆቹ ጠባቂውን ይሰማሉ"]	BIMPM	Neutral	Entailment
	RSSM(SG)	Neutral	
	RSSM(SW)	Neutral	
	RSSM(SG)	entailment	
	RSSM(SW)	entailment	
LONG SENTENCE Sample: (mismatched- neutral) Premises = ["ሰማያዊ ሸሚዝና መነጽር ያደረገ አንድ ወንድ ከአንዲት ልጃ ገረድ እና ከተደረደረ መጽሃፍ ፊት ለፊት ሆኖ አንድ መጽሃፍ ላይ ወደሚታይ ስእል እያመለከተ ነው"] Hypothesis = ["ልጅቷ በሥዕሉ ላይ ፍላጎት አላት"]	BIMPM	Contradiction	Neutral
	RSSM(SG)	Neutral	
	RSSM(SW)	Neutral	

Table 5.18, shows error analysis result as performed using RSSM (SW) model against using SG for word embedding layer, and with that of BIMPM. The value of the last two columns are value of the correct class and the predicted class for our model (RSSM (SW)) for the samples of CONDITIONAL, WORD_OVERLAP, LONG_SENTENCE, ACTIVE/PASSIVE, VERB SYNONYMY and PARAPHRASE have revealed a correct result than the other two model. Based on the preliminary result, one can deduce that our model capability to learn words or phrases syntactic features from the sub embedding layer as compared to the model RSSM (SG). The matching approaches and direction also help to predict the correct class. For LONG_SENTENCE the ability for BiLSTM with attentions improved the result of the classifier for predicting the correct class. As of the sentences types of CORF, NEGATION, TENSE_DIFFERENCE all the models predicted a vague result, this indicates the models have not found enough examples in the training.

Chapter 6: Conclusions and Future Works

This Chapter presents conclusions that discuss the activities done in this work with how the problems are addressed and achieved the intended objectives. Additionally, future works are also discussed.

6.1 Conclusion

In this work, we developed a textual entailment classification RSSM by using deep leering approach, generally by adding sub word information into the word vectors as input layer, adding mean pooling matching approach, and matching sentences in reverse direction. To build ATEC we performed discussions for producing findings, methodologies, and architectures on main components of the classifier in the approaches followed using DNN. And came up with an assumption to develop an end to end model for textual entailment classification that can eliminate the traditional approach of the underlying feature engineering challenges identified in Amharic language.

As the approaches followed in the in deep learning for entailment classification, first we used word vector as input layer, for producing the word vectors in best at learning the Amharic linguistic elements, we collected the corpus for the training from different source. And produce a word vectors for Amharic words, by adding sub word information to it using fasttext library. From the trained word vectors, we have found an interesting result of word vectors that eliminate the Amharic language barriers by manipulating its sub word level components. As this layer contribute the most for the classification task, we evaluated the resulted word vectors ability of capturing meaningful representations into intrinsic and extrinsic.

In the intrinsic evaluation, the question of how well the word vectors in the embedding capture linguistic relationships between words, in judgments gathered from language experts. The linguistic relationships under consideration are measurement were word similarity, word analogy, and odd word out. On those method, we observed that words that are similar or analogous to each other happen together or closer in the vector space. Related Amharic words are found contiguous to each other in the vector space. 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, In extrinsic evaluation we tested the trained word vectors both SW and SG model on the entailment classifier, and the resulted accuracy of SW model showed an improvement by 3% in train and of 2% in test dataset accuracy.

Second for evaluating entailment classifier performance with in the limited data have prepared 8, 700 of pair of sentences and embed with BiLSTM layer. We ablated RSSM model, by reducing the approaches followed to develop the model, and also compared it with three other models of BIMPM, CAM, and WBWAM in the same dataset we trained for our model. By far RSSM model performance showed an improved accuracy in train dataset accuracy by 77% and test of 72%, as the detail explained in chapter 5 section 6. Unlike the models used for comparison, adding the mean pooling approach in the matching layer, and matching the sentences reverse side (right directions Q<-P) improves our model mismatched performance. Even if we didn't include sentence genre in the training datasets, we performed error analysis on Amharic sentence types to show how the models handle the structure of Amharic sentences. And classifies mostly classifies as correct on sentence types, CONDITIONAL, WORD_OVERLAP, LONG_SENTENCE, ACTIVE/PASSIVE, VERB SYNONYMY and PARAPHRASE.

The contributions of this work are summarized as follows:

- Adopted the baseline word embedding model for Amharic language, by adding sub word information for learning meaningful representation of syntactic and semantic feature of Amharic words.
- Sentence embedding with attention to capture the meaning of long sequence sentence in Bidirectional embedding each time steps.
- Adding mean pooling matching approach in each time step of the sentences on the top of cosine matching of function, preserves the alignment between sentences.
- Reverse sided sentence matching (Right to left), matching hypothesis against the premise is more effective as it improves the mismatched accuracy of the classifier than the other way around or bilateral.
- Textual Entailment is a core Natural Language Understanding task (NLU). While it poses as a classification task, it is uniquely well-positioned to serve as a benchmark task for research on Amharic NLU as question answering, information extraction, summarization,

multi-document summarization, and evaluation of machine translation systems, need to recognize that a particular target meaning can be inferred from different text variants.

6.2 Future Works

In this research, we have made an attempt to explore deep learning approaches to develop a textual entailment classifier. In the future, to improve the performance of RSSM model the following directions are identified to be undertaken.

- Gathering a big and rich corpus to experiment with Amharic word embeddings for producing enhanced word vectors.
- Annotating a huge amount pair of sentences to increase the mismatch dataset accuracy of the classifier in sentence structures as CORF, NEGATION, and TENSE_DIFFERENCE, since the larger the dataset the more positive results produced in any DL based NLP task.
- Improving the accuracy of the mismatched texts, by using part of speech (POS) tagging. Word embedding's vector can be extended with additional POS embedding's vector.
- As the matching function contributes to different types of sentence compression experimenting on further matching function as SUB, MULT and SUBMULT+NN
- Adding compression strategy in DNN is always the best approach followed experimenting on the matching function, we can extend by adding a matching strategy as Max-Attentive-Matching. Mean -Attentive-Matching full max-matching and full mean matching and aggregating this all matching vectors to get a best classifier.

References

[1] J. Jiang and S. Wang, "Learning Natural Language Inference with LSTM," in *Computer Science Computation and Language*, 2016.

[2] L. Yang, S. Chengjie and l. lie, "Learning Natural Language Inference using Bidirectional LSTM model and inner Inner-Attention," *Computer Science Computation and Language*, 30 May 2016.

[3] J. Bos and K. Markert, "Recognising textual entailment with logical inference," *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, vol. 05, p. 628–635, 2005.

[4] R. Socher and C. D. Manning, "Natural language processing in deep learning," Stanford, 2015.

[5] Mesfin Abate and Yaregal Assabe, "Development of Amharic Morphological Analyzer Using Memory-Based Learning," *Advances in Natural Language Processing*, vol. 8686, 2014.

[6] Ababa, Ibrahim and Yaregal Assabe, "Amharic Sentence Parsing Using Base Phrase Chunking," 2014.

[7] Getenesh Teshome, "Machine Learning Approch for Recognizing Textual Entailment in Amaharic," in *Unpublished Master's Thesis*, 2018.

[8] R. Ido and d. Dan, "Recognizing Textual Entailment," *Textual Entailment*, vol. 15, no. 4, 2009.

[9] C. Aswani Kumar and S. Srinivas, "Latent Semantic Indexing using eigenvalue analysis for efficient information retrieval," *Int. J. Appl. Math. Comput. Sci*, vol. 16, pp. 551-558, 2006.

[10] B. Peter, D. P. Vincent, d. Peter, L. Jenifer and M. Robert, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, p. 467–479, 1992.

[11] Y. Bengio, R. Ducharme, P. Vincent and C. Janvin, "A Neural Probabilistic Language Model," vol. 3, p. 1137–1155, 2003.

[12] R. Collobert and J. Weston, "A Unified Architecture for Natural Language Processing," *Proceedings of the 25th international conference on Machine learning*, vol. 08, 2008.

[13] T. Mikolov, K. Chen, G. Corrado and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” in *International Conference on Learning Representations*, 2013.

[14] P. Jeffrey, R. Socher and D. C. Manning, "Glove: Global vectors for word representation.," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, p. 1532–1543, October 2014.

[15] D. Scott, S. T. Dumais, G. W. Furnas, T. K. Landauer and H. Richard, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, 1990.

[16] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[17] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, "Enriching Word Vectors with Subword Information," *Computer Science Computation and Language*, vol. v2, 19 jul 2016.

[18] Maysam, A. Chenaglu, "Convolutional Neural Netweork for NLP," 2018.

[19] J. Alon, S. Oren Sar and G. Yoav, "Understanding Convolutional Neural Networks for Text Classification," *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for* , p. 56–65, November 2018.

[20] Y. Li, Y. Hu and T. Yang, "Short Text Classification With A Convolutional Neural Networks Based Method," in *15th International Conference on Control, Automation, Robotics and Vision* , Singapore, 2018.

[21] Y. Zhang and W. Byron, "2015," *A Sensitivity Analysis of Convolutional Neural Networks for Sentence Classification.*, 13 oct 2015.

[22] T. Mikolov, M. Karafia´t, L. Burget, J. JČernocký and S. Khudanpur, "Recurrent neural network based language model," in *11th Annual Conference of the International Speech Communication Association*, Makuhari, 2010.

[23] J. Schmidhuber and H. Sepp, "Long short-term memory," *Neural computation*, vol. 9, p. 1735–1780, 1997.

[24] Y. Wenpeng, K. Katharina, Y. Mo and S. Hinrich, "Comparative Study of CNN and RNN for Natural Language Processing," *Computation and Language*, 7 feb 2017.

[25] P. Neculoiu, M. Versteegh and R. Mihai, "Learning Text Similarity with Siamese Recurrent Networks," *Proceeding of the 1st workshop on Representation Learning for NLP*, vol. 7, no. 4, pp. 148-157, August 2016.

[26] R. S. Bowman, G. Angeli, C. Potts and C. D. Manning, "A Large Annotated Corpus for Learning Natural Language Inference," *Procedding of the 2015 confrence on Emperical Methods in Natural Language Processing,* pp. 632-642, Septemeber 2015.

[27] T. Rocktaschel, E. Grefenstette, K. Moritz, H. K. Tomas and B. Phil, "Reasoning about entailment with neural attention," *Proceedings of the International Conference on Learning Representations,* 22 September 2016.

[28] L. Hua and H. Jimmy, "Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. I," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,*, 2016.

[29] S. Wang and J. Jiang, "A Compare and Aggregate Model for Sentence Matching," 6 Nov 2016.

[30] A. Parikh, O. Tackstrom, D. Das and J. Uszkoreit, "A Decomposable Attention Model for Natural Language Inference," in *Conference on Empirical Methods in Natural Language Processing*, 2016.

[31] Z. Wang, W. Hamza and R. Florian, "Bilateral Multi-Perspective Matching for Natural Language Sentences," *Computer Science Computation and Language,* 14 Jul 2017.

[32] R. Wynn and W. Amanda, "Amharic Language and Culture Manual," texas, 2011.

[33] L. Lewis, S. G. F. M and C. D. e. Fennig, "Ethnologue: Languages of the World," Dallas, Texas:, 2015.

[34] M. Ronny, "Amharic as lingua franca in Ethiopia," *Lissan Journal of African Languages and Linguistics,* Vols. 20,, no. 1/2:, pp. 117-131, 2006.

[35] B. Hamutal, G. Julia, H. Devlin and M. William, "Ten Years of Language Access in Washington, DC," vol. 43, April 2014.

[36] Amare Getahun, "ዘመናዊ የአማርኛ ሰዋሰው," Addis Ababa,, 1989.

[37] Baye Yimam, "The Ethiopian Writing," Addis Ababa University, Addis Ababa, Ethiopia, 1992.

[38] Baye Yimam, "ፋጭርና ቀላል የአማርኛ ሰዋሰው," Addis Ababa, 2002.

[39] T. Rocktaschel, E. Grefenstette and K. M. Hermann, "Reasoning about Entailment with Neural Attention," 2016.

[40] C. Manning and S. Bowman, "Annotated corpus for learning natural language inference," *Proceedings on Empirical Methods in Natural Language Processing,* p. 632–642, September 2015.

[41] S. Wang and J. Jiang, "A Compare-Aggregate Model for Matching Text Sequences," *Computer Science Computation and Language*, 6 Nov 2016.

[42] Yemane Tedlla and Kazuhide Yamamoto, "Analyzing word embeddings and improving POS tagger of tigrinya," in *International Conference on Asian Language Processing*, Singapore, Singapore , 2017.

[43] Aklilu Yared Yenalem, "Exploring Nural Word Embeddings For Amharic Language," in *Unpublished Master's Thesis*, Nicosia, 2019.

[44] Atelach Alemu Argaw, Lars Asker, Bj¨orn Gamb¨ack and Magnus Sahlgren, "Applying machine learning to Amharic text classification," Academia.edu, 2014.

[45] Surafeal Teklu Weldeselasse, "Automatic Categorization of Amharic news text: a machine learning approach.," in *Addis Ababa University,*, Addis Ababa, Ethopia., 2003.

[46] Addisalem Abera, "Concept-based Amharic Documents Similarity (CADS)," 2013.

[47] M. Schuster and K. Paliwal, "Bidirectional Recurrent Neural Networks," *Transactions On SIgnal Processing,* vol. 45, no. 11, pp. 2673-2681, 1997.

[48] Z. Wang, H. Mi, W. Hamza and R. Florian, "Multi-Perspective Context Matching for Machine Comprehension," *Computer Science Computation and Language,* vol. v1, no. 1612.04211, 13 Dec 2016.

[49] Steve Renals, "Multi-Layer Neural Networks," 2014.

[50] W. Zhiguo, H. Wael and F. Radu, "Bilateral Multi-Perspective Matching for Natural Language Sentences," 2017.

[51] I. Androutsopoulos and P. Malakasiotis, "A Survey of Paraphrasing and Textual Entailment Methods," *Journal of Artificial Intelligence Research*, vol. 38, pp. 135-187, 2010.

[52] B. Alexander and G. Hirst, "Evaluating wordnet-based measures of lexical," *Computational Lingustics*, vol. 32, no. 1, pp. 13-17, 2006.

Annexes

Annex A: Questionnaire form offered to Language Experts to List at most 3 morphologically similar words for a given query word.

We kindly ask you to assist us in a linguistic judgment, please list at most top 3 morphologically related words for a specified relation type for 19 query words of Amharic language. The aim is measure how much our models retuned answer and much with the experts listed out answer. Below there is a list words with relation type. For each word, please list at least 1 at most five word

Specific instructions:

- The questionnaire starts on the next page.
- Please fill in your full name at the beginning of the questionnaire.
- Please fill in your answer in the appropriate box of the table.

Full name: _____

No	Terms	Answer1	Answer2	Answer3
1	መንፈስ			
2	ኢየሱስ			
3	ግብፅ			
4	ወንጌል			
5	ህይወት			
6	ልዑል			
7	ሰው			
8	ነገሠ			
9	ተጠራ			
10	ፈራ			
11	ምድር			
12	ጽድቅ			
13	ፈጠረ			
14	ሰነፈ			
15	እርግማን			
16	በሽታ			
17	ድካም			
18	ብርሃን			
19	ዘላለም			

Annex B: Top 3 syntactic relation given by three experts and models retrieved results

No	Terms	SWE			SGM			Answer 1			Answer 1			Answer 3		
1	መንፈስ	መንፈስስ	መንፈስህ	መንፈስም	በመንፈስ	ደስታውን	የመፍራት	መንፈስም	መንፈስህ	መንፈስሽ	መንፈስን	መንፈሱ	መንፈስስ	መንፈሴ	መንፈስህስ	መንፈስስ
2	ኢየሱስ	በኢየሱስ	ስለኢየሱስ	ከኢየሱስ	ክርስቶስ	እየስተማራችኋቸው	እየጠመቃችኋቸው	ኢየሱስና	ከኢየሱስ	ከኢየሱስም	ኢየሱስን	የኢየሱስም	ለኢየሱስ	በኢየሱስ	ስለኢየሱስ	ከኢየሱስ
3	ግብፅ	ግብፅም	ግብፅን	ለግብፅ	ኤፍሬም	ተራራማው	አምናውያን	ግብፅን	ለግብፅ	ግብፅም	የግብፅ	ከግብፅም	ግብፅስ	ከግብፅ	ግብፅና	ግብፅም
4	ወንጌል	ወንጌልን	ወንጌልና	ወንጌልም	ጌታችን	ለአሕዛብ	ወንጌልን	ወንጌሌ	በወንጌል	ከወንጌል	የወንጌልም	ወንጌልና	ወንጌልን	ወንጌልም	ወንጌልስ	ስለወንጌል
5	ህይወት	ለህይወት	ህይወትዬ	በህይወት	ህይወትማ	ከምትጣል	ቁርጠህ	ህይወትማ	ከሕይወት	ሕይወትና	ሕይወትስ	የሕይወት	በህይወት	ህይወቷ	ህይወትዬ	ለሕይወት
6	ልዑል	በልዑልም	ለግርማዊ	ከልዑል	ግርማዊ	ክቡር	ሃይማኖትን	የልዑል	በልዑል	በልዑልም	ልዑልነታቸው	ከልዑል	በልዑል	ልዑልስ	በልዑል	ልዑልና
7	ሰው	ሰውና	ሰውም	ሰውሬ	ሰውን	ሰውም	ይገደል	ከሰው	ሰውዬ	ሰውና	ሰውስ	ሰውማ	ሰውየሽ	ስለሰው	ሰውም	ሰውም
8	ነገሠ	ነገሥች	በነገሠ	አነገሠ	በነገሠ	ኢዮራም	አካዝያስ	ከነገሠ	ነገሥች	ንጉስ	በነገሠ	ነገሥች	የነገሠ	አነገሠ	ከነገሠ	አነገሠ
9	ተጠራ	ቢጠራ	ተጠራን	ሊጠራ	ይጠራል	ሲጠራ	ተነገረው	የተጠራ	ተጠርቶ	ተጠራና	ቢጠራ	ተጠራን	ተጠራና	ሲጠራ	ሲጠራ	ተጠርቶ
10	ፈራ	ስፈራ	ልቅሶ	ልቅሶ	ተባ	ፈጥኖ	ለመነ	ፈራች	የፈራ	ስፈራ	ብፈራ	አስፈራ	ሲፈራ	ስፈራ	ፈራን	ተፈራ
11	ምድር	ምድርህ	ምድርና	ምድሬ	በግብፅ	ርስት	ወተትና	ምድራዊ	ለምድር	የምድር	ምድርህ	ምድርም	ምድርስ	ምድርና	ምድራዊ	ምድርስ
12	ጽድቅ	በጽድቅ	ጽድቅን	ለጽድቅ	ዘሪሁንን	ዮሀንስና	በሸር	ጽድቅስ	ጽድቅም	ጽድቅና	ጽድቅ	ጽድቅን	ለጽድቅ	በጽድቅ	ጽድቅህ	ለጽድቅ
13	ፈጠረ	በፈጠረ	የፈጠረ	ያሳሳል	የሚፈራው	የሚያድን	ሥራውም	ፈጠረን	ፈጠራት	ፈጠረና	ፈጠረና	ሲፈጥራቸው	ፈጠራቸው	የፈጠረን	ፈጠረና	ፈጠራ
14	ሰነፈ	ሰነፋቹ	የሰንፋ	መሰነፍ	OOV	OOV	OOV	ሰነፈና	ሰነፈች	ሰነፍና	ሰነፋቹ	ሰነፈና	መሰነፍ	ሰነፍና	የሰነፈ	መሰነፍ
15	እርግማን	ለእርግማን	እርግማንና	የእርግማን	OOV	OOV	OOV	እርግማኑ	ከእርግማኑ	እርግማንስ	እርግማኑን	እርግማኔ	እርግማኑ	የእርግማን	ላስረገመ	ለእርግማን
16	በሽታ	በሽታህ	በሽታህ	በሽታው	ሕመም	ጭንቀት	ጠፍቶ	በሽታና	በበሽታ	የበሽታ	በሽታዬ	የበሽታ	በበሽታና	በሽታው	የበሽታ	ለበሽታ
17	ድካም	ለድካም	ድካምና	በድካም	ህልም	ለመገመት	ለወደፊቱ	ድካምም	ድካምስ	የድካም	የድካም	ድካሙ	የድካም	ድካምም	ከድካም	ድካምም
18	ብርሃን	ብርሃንዋ	ለብርሃን	ቡብርሃን	ፀሐይ	ጨለማ	ጨረቃ	ብርሃንስ	ብርሃንም	ብርሃናማ	ከብርሃን	ብርሃኑስ	ብርሃንስ	ብርሃንስ	ብርሃንስ	ለብርሃን
19	ዘላለም	ዘላለማዊ	ዘላለሙን	ለዘላለም	ከዘላለም	ከትውልድ	የጸና	የዘላለም	ዘላለማዊ	የዘላለም	ዘላለማዊ	የዘላለም	ዘላለምና	ዘላለማዊ	ዘላለሙን	ለዘላለም

Annex C: Questionnaire form for Language Experts to Estimate Semantic Relatedness between Pairs of Words for a Given Relation Type

We kindly ask you to provide us your expert judgment, aimed you to estimate the relatedness score for 10 pair of Amharic language word for a specified relation. Based on the estimation word semantic relatedness evaluator measures how well human perceived relatedness captured by our two word vector models that we performed the evaluation; it correlate the distance between word vectors and human perceived semantic relatedness.

Below is a list of pairs of words. For each pair, please assign your rating score in the margin of 0 and 10 (0 = words are totally unrelated, 10 = words are very closely related).

Instructions

- The questionnaire starts on the next page
- Please fill in your full name at the beginning of the questionnaire.
- Please fill in the scores in the appropriate column of the table.

Full name: _____

Word pair		Answer 1	Answer 2	Answer 3	Answer 4	Answer 5
ክርስቶስ	ኢየሱስ	8	8	8	8	8
በኢየሱስ	በክርስቶስ					
እግዚአብሔር	አምላክ					
ሐዋርያ	ጳውሎስ					
ጳውሎስ	ኢየሱስ					
ህይወት	ሞት					
ጸጋ	ሰላም					
ወንጌል	የምስራች					
የክርስቶስም	የጌታም					
አዳም	ኢየሱስ					
አዲስ_	ፊሊጵስዩስ					
ብሉይ	ማርቆስ					
አገልጋይ	ሐዋርያ					
አገልጋይ	ሰባኪ					
ቅዱስ	እርኩስ					
ምርቃት	እርግማን					
ወረርሽኝ	በሽታ					
ፍቅር	ጥላቻ					
ነፍስ	ህይወት					
ጽድቅ	ኩነኔ					

Annex D: Language expert answers for semantic relation score out of 10 for the given word pair and relation type, and models retrieved results

Relationship type	Word pair		SWM	SGM	Answer 1	Answer 2	Answer 3	Answer 4	Answer 5	Average LEA
similar	ክርስቶስ	ኢየሱስ	0.80	0.80	8	8	9	9	8	8.4
similar	በኢየሱስ	በክርስቶስ	0.86	0.85	9	9	8	8	9	8.6
similar	እግዚአብሔር	አምላክ	0.71	0.58	9	9	8	8	9	8.6
similar	ሐዋርያ	ጳውሎስ	0.75	0.63	7	7	8	8	8	7.6
similar	ጳውሎስ	ኢየሱስ	0.80	0.80	7	6	6	5	7	6.2
similar	ጸጋ	ሰላም	0.75	0.69	8	8	7	7	7	7.4
similar	ወንጌል	የምስራች	0.62	0.57	9	9	9	8	8	8.6
similar	የክርስቶስም	የጌታም	0.79	0.79	8	8	8	8	8	8
similar	ጸም	ጸሎት	0.68	0.70	8	8	8	8	8	8
similar	አገልጋይ	ሐዋርያ	0.73	0.80	8	8	8	8	8	8
similar	አገልጋይ	ሰባኪ	0.66	0.65	8	8	7	7	7	7.4
similar	ወረርሽኝ	በሽታ	0.66	0.64	8	8	8	8	8	8
similar	ነፍስ	ህይወት	0.41	0.29	7	6	2	2	2	3.8
dissimilar	ጽድቅ	ኩነኔ	0.42	0.50	1	1	1	1	1	1
dissimilar	ፍቅር	ጥላቻ	0.21	0.41	1	1	1	1	1	1
dissimilar	አዳም	ኢየሱስ	0.10	0.32	2	2	3	3	1	2.2
dissimilar	ህይወት	ሞት	0.34	0.31	1	1	1	1	1	1
dissimilar	ብሉይ	ማርቆስ	0.10	0.41	1	1	1	2	2	1.4
dissimilar	ቅዱስ	እርኩስ	0.28	0.21	1	1	1	2	2	1.4
dissimilar	ምርቃት	እርግማን	0.30	0.34	1	1	1	1	1	1

Annex E: Questionnaire form offered to Language Experts to give their judgment in listing one odd word out from the given word chunk cluster.

We kindly ask you to provide us your expert judgment, aimed you to estimate from the given word chunks to select one word out that doesn't belong to the given word cluster. Give your answers to the specified box. Based on the estimation filtering one odd out words the evaluator measures how human judge the unrelated words is captured well by our two word vector models that we performed the evaluation;

Specific instructions:

- The questionnaire starts on the next page.
- Please fill in your full name at the beginning of the questionnaire.
- Please fill in your answer in the appropriate box of the table.

Full name: _____

Words chunks	Answer 1	Answer 2	Answer 3	Answer 4	Answer 5
ሙሴ ኢየሱስ ቆሮንጦስ ዮሐንስ ጳውሎስ					
ዮሐንስ ይሁዳ ጴጥሮስ ኢየሱስ					
አክስት ኣጎት አባት					
እህት ወንድም አናት					
ንጉሥ ልዑል ንግሥት መምህር					
እሱ የእሱን እነርሱ					
ጋዜጣ ራዲዮ ቴሌቪዥን ዜና					
ፊልም ድራማ ተማሪ					
ተጓዘ ሮጠ መንገድ ሽንት እምነት					
እግዚአብሔር አምላክ አዳም ኢየሱስ					
ወንጌል የምስራች ብስራት ሰላም					
አናት አባት ልጅ					
አናት አባት ልጅ ጎረቤት					
በላ አኘከ አላመጠ ተኛ					

Annex F: One odd word out answers given by 5 experts and models retrieved results

Words chunks	SWM	SGM	Answer 1	Answer 2	Answer 3	Answer 4	Answer 5
ሙሴ ኢየሱስ ቆሮንጦስ ዮሐንስ ጳውሎስ	ሙሴ	ዮሐንስ	ሙሴ	ሙሴ	ሙሴ	ሙሴ	ሙሴ
ዮሐንስ ማርያም ይሁዳ ኢየሱስ	ይሁዳ	ማርያም	ይሁዳ	ይሁዳ	ይሁዳ	ይሁዳ	ይሁዳ
አክስት ኣጎት አባት	አባት	አባት	አባት	አባት	አባት	አባት	አባት
እህት ወንድም አናት	አናት	አናት	አናት	አናት	አናት	አናት	አናት
ንጉሥ ልዑል ንግሥት መምህር	መምህር	ንጉሥ	መምህር	መምህር	መምህር	መምህር	መምህር
እሱ የእሱን እነርሱ	እነርሱ	እሱ	እነርሱ	እነርሱ	እነርሱ	እነርሱ	እነርሱ
ጋዜጣ ራዲዮ ቴሌቪዥን ዜና	ዜና	ዜና	ዜና	ዜና	ዜና	ዜና	ዜና
ፊልም ድራማ ተማሪ	ተማሪ	ተማሪ	ተማሪ	ተማሪ	ተማሪ	ተማሪ	ተማሪ
ተጓዘ ሮጠ መንገድ እምነት	እምነት	እምነት	እምነት	እምነት	እምነት	እምነት	እምነት
እግዚአብሔር አምላክ አዳም ኢየሱስ	ኢየሱስ	አዳም	አዳም	አዳም	አዳም	አዳም	አዳም
ወንጌል የምስራች ሰላም	ሰላም	የምስራች	ሰላም	ሰላም	ሰላም	ሰላም	ሰላም
አናት አባት ልጅ	አናት	አናት	ልጅ	ልጅ	ልጅ	ልጅ	ልጅ
አናት አባት ልጅ ጎረቤት	ጎረቤት	አናት	ጎረቤት	ጎረቤት	ጎረቤት	ጎረቤት	ጎረቤት
በላ አኘከ አላመጠ ተኛ	ተኛ	በላ	ተኛ	ተኛ	ተኛ	ተኛ	ተኛ

Annex G: Questionnaire form offered to Language Experts based on the analogy relationship criteria to list the most analogically related word

We kindly ask you to provide us your expert judgment, intended you to estimate based on the given analogy relationship criteria that seen in the second column of the questionnaire, to list the most related word of the third word. The aim is based on the your analogy relationship judgment the evaluator measures how human perceived produced analogically related word well by our two word vector models that we performed the evaluation.

Specific instructions:

- The questionnaire starts on the next page.
- Please fill in your full name at the beginning of the questionnaire.
- The analogy relationship criteria has syntactic and semantic relatedness criteria.
- Please fill in your answer in the appropriate box of the table.

Full name: _____

1. Fill in the word that semantically most related word based on the relatedness criteria

Relatedness type	**Given analogy**	**Answer 1**	**Answer 2**	**Answer 3**	**Answer 4**	**Answer 5**
title relation	ወንድ:ንጉሥ::ሴት፡?					
Opposite relation	አጭር :ረዥም::ጥቁር:?					
Opposite relation	አጭር :ረዥም።ትንሽ:?					
compound words	ቤት:ቤተመንግስት::ህግ:?					
Sex relation	ሴት :እናት::ወንድ:?					
Sex relation	ሴት :እህት::ወንድ:?					
Capital-country	ፓሪስ: ፈረንሳይ።አዲስአበባ :					
Country-continent	ኢትዮጵያ: አፍሪቃ።ቱርክ					

2. Fill in the word that syntactically most related word based on the relatedness criteria

Relatedness	**Given analogy**	**Answer 1**	**Answer 2**	**Answer 3**	**Answer 4**	**Answer 5**
Possessives in sex	እሱ:የእሱን::እኛ፡?					
Possessives in sex	እሱ:የእሱን።እሷ፡?					
Plural affixes	አባት:አባቶች። እናት:?					
Plural affixes	አባት:አባቶች። ልጅ:?					
Passive voice affixes	ገደለ: ተገደለ::ሰማ:?					
Passive voice affixes	ገደለ: ተገደለ::ሰበረ:?					
Pronoun/verb affixes	መጣ : እየመጣ።ሄደ:?					
Pronoun/verb affixes	መስማት፣መስማቱ።መናገር:?					

Annex H: Analogy relationship answers given by 5 experts and models retrieved results

1. Semantic analogy relationship Expert answer and the two word vector models

Relatedness type	Given analogy	SWM	SGM	Answer 1	Answer 2	Answer 3	Answer 4	Answer 5
title relation	ወንድ:ንጉሥ::ሴት:?	ንግስት	ንጉሥሽ	ንግስት	ንግስት	ንግስት	ንግስት	ንግስት
Opposite relation	አጭር :ረዥም::ጥቁር:?	ነጭ	ነጭ	ነጭ	ነጭ	ነጭ	ነጭ	ነጭ
Opposite relation	አጭር :ረዥም::ትንሽ:?	ትንሽና	ትንሽና	ትልቅ	ትልቅ	ትልቅ	ትልቅ	ትልቅ
compound words	ቤት:ቤተመንግስት::ህግ:?	ሕገመንግስት	ለገንዘብና	ሕገመንግስት	ሕገመንግስት	ሕገመንግስት	ሕገመንግስት	ሕገመንግስት
Sex relation	ሴት :እናት::ወንድ:?	አባት	ወንድም	አባት	አባት	አባት	አባት	አባት
Sex relation	ሴት :እህት::ወንድ:?	ወንድም	ወንድም	ወንድም	ወንድም	ወንድም	ወንድም	ወንድም
Capital-country	ፓሪስ: ፈረንሳይ::አዲስአበባ :	ላዲስ	ላዲስ	ኢትዮጵያ	ኢትዮጵያ	ኢትዮጵያ	ኢትዮጵያ	ኢትዮጵያ
Country-continent	ኢትዮጵያ: አፍሪቃ::ቱርክ	ቮልታና	ቮልታና	እስያ	እስያ	እስያ	እስያ	እስያ

2. Syntactic analogy relationship Expert answer and the two word vector models

Relatedness	Given analogy	SWM	SGM	Answer 1	Answer 2	Answer 3	Answer 4	Answer 5
Possessives in sex	እሱ:የእሱን::እንቺ:?	የእንቺን	ነሽ	የእንቺን	የእንቺን	የእንቺን	የእንቺን	የእንቺን
Possessives in sex	እሱ:የእሱን::እሷ:?	የእሷን	አልቻልኩም	የእሷን	የእሷን	የእሷን	የእሷን	የእሷን
Plural affixes	አባት:አባቶች:: እናት:?	እናቶች	አባቶች	እናቶች	እናቶች	እናቶች	እናቶች	እናቶች
Plural affixes	አባት:አባቶች:: ልጅ:?	ልጆች	በየወገናቸውም	ልጆች	ልጆች	ልጆች	ልጆች	ልጆች
Passive voice affixes	ገደለ: ተገደለ::ሰማ:?	ያሰማ	ያሰማ	ተሰማ	ተሰማ	ተሰማ	ተሰማ	ተሰማ
Passive voice affixes	ገደለ: ተገደለ::ሰበረ:?	ተሰበረ	አሳስባለሁ	ተሰበረ	ተሰበረ	ተሰበረ	ተሰበረ	ተሰበረ
Pronoun/verb affixes	መጣ : እየመጣ::ሄደ:?	እየሄደ	ርቆ	እየሄደ	እየሄደ	እየሄደ	እየሄደ	እየሄደ
Pronoun/verb affixes	መስማት:መስማቱ::መናገር:?	መደናገር	ገልጦ	መናገሩ	መናገሩ	መናገሩ	መናገሩ	መናገሩ

Declaration Sheet

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

Declared by:

Name: Helina Mesfin .

Signature: _____

Date: _____

Confirmed by advisor:

Name: Fekade Getahun (PhD)

.

Signature: _____

Date: _____