ADDIS ABABA UNIVERSITY
COLLEGE OF NATURAL SCIENCES

Afaan Oromo WordNet Construction Using Sense Embedding

Henok Desalegn Alemayehu

A Thesis Submitted to the Department of Computer Science
in Partial Fulfillment of the Degree of Master Science in Computer Science
(Software Engineering)

Addis Ababa, Ethiopia
October 1, 2021

Addis Ababa University

College of Natural Science

Henok Desalegn Alemayehu

Advisor: ***Yaregal Assabie (PhD)***

This is to certify that the thesis prepared by ***Henok Desalegn Alemayehu*** titled: *Afaan Oromo WordNet Construction Using Sense Embedding* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science (in Software Engineering) complies with the regulations of the university and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

Advisor: Yaregal Assabie (PhD) signature_____date_____

Examiner: _____

Examiner: _____

Dedication

All that I am, or ever hope to be, I owe to my Angle Mother: Tadelech Getahun W/Selassie.

Abstract

One of the primary goals of the field of natural language processing is to create very high-quality WordNet which can be used in many domains. The main area which WordNet methods typically fall short is in handling polysemy representation. A word is polysemous when it has multiple meanings (e.g., the word bank when used in a financial context versus an ecological context). Current WordNet methods fail to handle this at all when using word embedding models for automatic WordNet construction that train just one embedding for all meanings of a word. Words have different meanings (i.e., senses) depending on the context. Disambiguating the correct sense is important and a challenging task for natural language processing. Contextualized models represent the meanings of words in context. This enables them to capture some of the vast array of linguistic phenomena that occur above the word level.

In this study, we propose automatic Afaan Oromo WordNet construction using sense embedding. The proposed model includes different tasks. We perform text pre-processing in Afaan Oromo text document and train the document using a sense embedding spacy library (sense2vec) and Facebook fastText library to generate sense embedding model. The embedding result provides a contextually similar word for every word in the training set. The trained sense vector model captures different patterns. After training the data we take the trained model as input and discover different patterns that used to extract WordNet relations.

We use POS tagged Afaan Oromo corpus to model WordNet. The resulting WordNet using fastText and sense2vec showed that words that are similar or analogous to each other happen together or closer in space. Related Afaan Oromo words were found closer to each other in the vector space. Morphological relatedness took the highest stake. The sense embedding has also learned the vector representation, “moti (king) - dhira(man) + dubara(woman)” resulting in a vector closer to the word “gifti(queen)”. Out-of-vocabulary words were also entertained. We got Spearman's correlation score of Rs=0.74 for each relation type, multi class text classification on the model attained 92.6% F1-score; result being fluctuated based on parameters.

Keywords: *Afaan Oromo WordNet, Word Sense Induction, polysemy, Word Analogy, Hypernym, Hyponym, Synonym, Antonym, Sense Embeddings, Word Embedding, Sense2Vec, fastText*.

ACKNOWLEDGMENT

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

It will be a great omission not to thank my families, friends, and fellows.

It would be against the will and conscience of my mind not to render my heartfelt and warmest gratitude and thanks to my supportive fiancé Yanet Zewdagegnehu (Emukoo), thank you for all the nagging to work on my thesis, couldn’t have done it without you

Table of Contents

List of Tables

List of Figures

List of Algorithm

Acronyms and Abbreviations

BERT	Bidirectional Encoder Representation from Transformers
CBOW	Continuous Bag-of-words
LSA	Latent Semantic Analysis
ML	Machine Learning
NLP	Natural Language Processing
POS	Part of Speech
PWN	Princeton WordNet
Sense2vec	Sense to Vector
SG	Skip-Gram
WN	WordNet
Word2vec	Word to Vector
WSD	Word Sense Disambiguation

CHAPTER ONE: INTRODUCTION

1.1 Background

Language is the primary method of human communication, but there are also other ways to communicate without the use of language. Language in its most complex form is unique to humans, although some animals have been found to have basic communication patterns. Language category falls into two i.e., natural language and formal language. Natural language is a language that is developed in humans through use and repetition without any planning [1]. This natural language existed for thousands of years; the designer of this language is not defined. In logic, mathematics, computer science, and linguistics, a formal language consists of words whose letters are taken from an alphabet and are well-formed according to a specific set of rules. It is believed that this language is created after induction of computers. Both languages may have some separate syntax and semantics rules.

The idea of computers being able to understand ordinary languages and hold conversations with human beings has been a staple of science fiction since the first half of the twentieth century and was envisaged in a classic paper by Turing [2] as a hallmark of computational intelligence. Since the start of the twenty-first century this vision has been starting to look more plausible. Artificial intelligence techniques allied with the scientific study of language have emerged from universities and research laboratories to inform a variety of industrial and commercial applications. Many websites now offer automatic translation; mobile phones can appear to understand spoken questions and commands; search engines like Google use basic linguistic techniques for automatically completing or ‘correcting’ our queries and for finding relevant results that are closely matched to our search terms.

Natural language processing (NLP) is a sub field of artificial intelligence that is focused on enabling computers to understand and process human languages [3]. It will take text in spoken languages or written format as an input, understand the content and interpret it after applying different rules or theories accordingly. To achieve better result out of NLP, the computer

should understand the language's structure deeply. As humans use natural language for communication, NLP will also help to ease communication with computer using natural way.

In general term, NLP tasks break down languages into shorter elemental pieces, understand and explore relationships between pieces to give a meaning [4]. The higher capability of NLP includes context categorization, topic discovery & modeling, contextual extraction, sentiment analysis, speech-to-text & text-to-speech conversion, document summarization and machine translation [4].

WordNet is a lexical database for natural languages. It groups words of a given language into different sets called synsets, which hold synonym as element. It provides little and brief definitions with examples; it also registers different relations (different implications of the word) among synsets. WordNet for English was introduced in 1985 at Princeton University in the cognitive science laboratory under the direction of psychology professor Fellbaum [5].

WordNet resource is the backbone of many research works of natural language processing applications [6]. WordNetsare widely used in various applications of NLP such as information retrieval, machine translation, multi-language information retrieval, automatic text summarization, word-sense disambiguation, automatic text classification [5] etc. It improves the accuracy of many NLP applications.

WordNet construction can be achieved using manual [5], automated [7] and semi-automated [8] methods. Manual construction of WordNet is done through the collection of words with their meaning in a different context (hyponym, hypernym, and synonym) manually. Most automated and semi-automated methods use machine translation (MT) and Princeton WordNet (PAWN) to construct synsets of a word automatically and use sense embedding approach to extract the context of words within its sense in addition to MT and English WordNet.

Morphological sense embeddings help improve the quality of WordNetssense embeddings for less frequent morphological variants, which is important for morphologically rich and low-resource languages like Afaan Oromo.

Nowadays a new method that can extract synonym of a word is introduced using neural sense embedding. Neural sense embedding includes conserve, word2vec, sentence2vec, and

doc2vec. From the list, sense2vec is used to generate synsets of a word. Sense2vec is a two-layer neural network which is used to process any text document. For a given input text corpus with its corresponding sense, it outputs a set of vectors. One of its models is the so-called skip-gram model that represents any words in a text in a real number called a vector. The vector consists of features of a word in N dimensions. Those features are contextually similar words. Sense2vec is also used to produce sense embeddings. Fatemeh, Torabi. [9] proposed a model that reconstructs the linguistic contexts of words using two-layer neural networks.

1.2 Motivation

Ethiopia has more than 80 languages spoken within the country. Afaan Oromo is spoken by about 33.80% people in Ethiopia [10]. Afaan Oromo is an Afro-Asiatic language belonging to the Cushitic branch. It is native to the Ethiopian state of Oromia and spoken predominantly by the Oromo people and neighboring ethnic groups in the Horn of Africa [10]. Afaan Oromo is the official language for Oromia region. The region uses it in education system, mass media broadcasts, proclamations, official documents, and newspapers.

As a result, many documents are produced using Afaan Oromo. However, to access and effectively process such documents, we need Afaan Oromo NLP applications. Afaan Oromo WordNet is an important component to enhance the performance of Afaan Oromo NLP applications. This has motivated us to develop Afaan Oromo WordNet.

1.3 Statement of the problem

Word embedding can capture many linguistic regularities and patterns [11]. However, they treat words holistically with no attention to their internal structure [12, 7] ignoring the fact that, morphologically, meaning is a multi-faceted concept with multiple axes along which two words can be similar. For languages known to be morphologically simple, like English, i.e., languages with a low morpheme-per-word ratio, this is often not a problem, but for processing inflectional languages with rich morphology, like Amharic and Afaan Oromo, exploiting word internal structure is mandatory. For example, applying word embedding to Afaan Oromo cause's different surface realizations of a word, like conjugations and inflections of a verb, to be the most semantically similar words and this strongly demotes other semantically similar

words that have different forms [12]. In addition, these kinds of similarities lead to sparsity problems and deficiency in exploiting shared semantics.

To our best knowledge, apart from a few sample resources [13], [14] there is no extensive study carried out to construct Afaan Oromo WordNet. Thus, the aim of this work is to study the effect of incorporating morphological senses to Afaan Oromo sense embedding in a way that enhances both semantic and morphological similarity in constructing Afaan Oromo WordNet.

1.4 Objective

General objective

The general objective of this research work is to construct Afaan Oromo WordNet using sense embedding.

Specific objective

- Review Afaan Oromo linguistic structure and WordNet construction approaches for morphologically complex languages.
- Collect Afaan Oromo POS Tagged corpus.
- Design general architecture of Afaan Oromo WordNet model.
- Develop prototype of the system.
- Test and evaluate performance of the model.

1.5 Methods

To achieve the objectives of this research, the following methodologies will be used.

Literature Review

We will make thorough review of different related works on WordNet construction and Afaan Oromo language. This helps to understand the morphological properties of Afaan Oromo, WordNet construction techniques in general and sense embedding.

Data Collection

The data needed for this research will be collected from different online and offline data sources. We will use the corpus for extraction of word contexts in different domains and to extract the synsets.

Development of a Prototype

We will develop a prototype of the system used to construct Afaan Oromo WordNet.

Experimental Evaluation

The accuracy of the constructed Afaan Oromo WordNet will be evaluated using different measuring metrics like precision and recall.

1.6 Scope and Limitation

The scope of this work is constructing Afaan Oromo WordNet. Among various relations between words, we will consider relations like synonym, antonym, hypernym, and hyponym. The limitation of this thesis work is that it does not include works on dialectal Afaan Oromo corpora to cover all variations of Afaan Oromo words. Embedding regeneration using morphological rules from lemma-based embeddings to represent inflected words are also not included in this work. It does not also include the other relations that exist in WordNet like meronym, homonym, toponym etc.

1.7 Application of Results

Wordnet, one of automatically compiled electronic dictionaries, is restricted to no specific domain, and covers most Afaan Oromo nouns, adjectives, verbs, and adverbs. WordNets are important resources in the NLP community. It's because WordNets are crucial for various downstream tasks such as POS tagging, sentiment analysis, NER, text classification, syntax parsing and so on. Therefore, in order to study, design, analyze and improve those tasks, WordNets should be thoroughly explored.

Therefore, this work will be an input to Afaan Oromo NLP research areas such as sentiment analysis. It will pave the path for improved text classification for many domains such as text classification, spam detection, document classification.

1.8 Organization of this Thesis

The rest of this thesis is organized as follows. Chapter Two provides the detail of different approaches to extract synonym, antonym, hyponym, and hypernym. In Chapter Three, we discuss the existing mechanism of WordNet construction for different language. Chapter Four details the system design and the proposed system architecture. In Chapter Five we discuss the system prototype and evaluation metrics and Conclusion of the study presented in chapter six. Possible future extension and contribution of the work also will be presented in Chapter Six.

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction

In this chapter, a brief overview of Afaan Oromo language and different WordNet construction approaches are discussed. The major Afaan Oromo word classes, which are nouns, verbs, adjectives, and adverbs are also discussed in this chapter.

2.2 A Brief Overview of Afaan Oromo Language

Afaan Oromo is the most widely spoken language in the family of Cushitic branch, and the most populous language of Ethiopia [15]. It is the third most widely spoken language in Africa, after Hausa and Arabic. Afaan Oromo is widely used as both written and spoken language in Ethiopia and some neighboring countries, including Kenya and Somalia. Besides being an official language of regional government of Oromia, Afaan Oromo is the instructional medium for primary and junior secondary schools throughout the region. Moreover, several literatures, newspapers, magazines, educational resources, official documents, and religious writings are written and published in Afaan Oromo [18, 19].

There are also television and radio programs on which information in Afaan Oromo is being broadcasted in Ethiopia. These include Ethiopian broadcasting service (EBC), Oromia broadcasting network (OBN), Oromia broadcasting service (OBS), Ethiopian radio, and radio Fana.

Before 1991, Ge'ez script was used for writing Afaan Oromo documents. A Latin-based alphabet called ***Qubee*** was adopted and became the official script of Afaan Oromo after 1991. There are about twenty-six consonants and ten vowels (five short and five long) in the language [18, 20].

2.2.1 Afaan Oromo writing system

Afaan Oromo uses *Qubee* (Latin based alphabet) that consists of thirty three basic letters, of which five are vowels, twenty-four are consonants, out of which seven are paired letters and fall together (a combination of two consonant characters such as '*Ch'* , Dh, Ny, Ph and Sh) and the rest consonants are represented by the letters: B, C, D, F, G, H, J, K, L, M, N, P, Q, R, S, T, V, W, X, Y and Z [16]. The Afaan Oromo alphabet characterized by capital and small

letters as in the case of the English alphabet. In Afaan Oromo language, as in English language, the vowels are sound makers and are sound by themselves. Vowels in Afaan Oromo are characterized as short and long vowels, represented by the letters, a, e, o, u and i, or long vowels: aa, ee, oo, uu and ii. The length of the vowel makes a difference in word meaning [17]. For Example:

Sena [come in] and Seenaa [history]

Fala [solution] and Faalaa [opposite]

The complete list of the Afaan Oromo alphabets is found on the manuscript by [18]. The basic alphabet in Afaan Oromo does not contain 'p', 'v' and 'z', because there are no native words in Afaan Oromo that are formed from these characters. However, in writing Afaan Oromo language, they are used to refer to foreign words such as "*vayirasii*" ("virus").

2.2.2 Word Categories of Afaan Oromo

Afaan Oromo words can be placed into five grammatical categories: nouns, verbs, adverbs, adjectives, and ad position [19]. According to Mandefro Legese [20] pronouns are included under the noun category, and conjunctions and interjections under ad position.

Nouns

A noun is a class of word that denotes person, animal, place, thing, or idea. Nouns are expressed by gender, number, and definiteness. In Afaan Oromo, a noun (maqaa) mainly occurs at the beginning of a sentence.

For example: *Lencoon farda dima yabate. (Lencho hopped on a red horse).*

Lencoon (Lencho) 'name of person' is a noun, comes at the beginning of the sentence.

A word that is categorized as a noun in a sentence can be a subject or an object [21]. In Afaan Oromo, a subject mostly comes at the beginning whereas an object mostly comes after subject and before verbs in a sentence.

For example: in the sentence, Gamachuun kophee bitate. (Gemechu bought a shoe).

The noun 'Gamachu' (name of person) is the subject and the noun 'kophee' (shoe) is the object.

In Afan Oromo nouns gender is marked by suffix like, **-eessa** and –**eettii** denote masculine and feminine respectively [19].

Qalbeessa (m)	Qalbeettii (F)	[thoughtful]
Og**eessa** (m)	Og**eetti** (F)	[Expert]

There are notable exceptions where nouns are derived from verbs, the masculine noun adds an **- aa** suffix and the feminine noun adds a **-tuu** suffix to the verb root.

Teacher	Barsiisaa (m)	Barsiistuu (F)	Barsiisuu (Verb) – to teach

Afaan Oromo plural nouns are mainly formed by adding suffixes: '–oota', '–ota', '–wwan', '– een', '–lee' and '–yyi' [11, 12].

In Afaan Oromo plural forms of nouns are rarely used, plurality is shown by suffix like **-oota**, **-aan**, **- wan**, and –**en** [19].

Harre (sg.) [Donkey], Harroota (Pl.) [Donkeys]

Like other languages, Afaan Oromo language does not have special word to denote definiteness, it is marked by the suffix **-icha** (**-ticha**) for masculine and **-ittii** (**-tittii**) for feminine [19].

Oromoo	Orom**ticha** (m)	Orom**tittii** (F)	[An Oromo]

And in definitiveness of words in Afaan Oromo is denoted by numeral 'tokko' or 'takka' which gives meaning of 'a certain' [19].

Intalla	**[one/a** girl]

Verbs

Verbs are content words that denote an action, occurrence, or state of existence. Afaan Oromo has base (stem) verbs and four derived verbs from the stem. Moreover, verbs in Afaan Oromo are inflected for gender, person, number, and tenses. There are four derived stems, the formation of which is still productive, Auto benefactive, passive, causative and intensive [20].

*Olaanan kuba mila **taphate**. (Olana played football)*

*Billissen **demte.** (Bilise has gone)*

*Gaadisseen barattuu **dha.*** *(Gadise is a student)*

Adverbs

Afaan Oromo adverbs are used to modify verbs. Adverbs precede the verbs that they modify or describe. They have the function to express different adverbial relations such as relations of time, place, manner, measure cause, or degree and answers questions such as „how? ", „when? ", „where? ", and „how much? " [20].

Some examples of adverbs of time:

amma - now, refu – just now

Some examples of adverbs of place:

achi(tti) - there, kessa - inside

Some examples of adverbs of manner:

saffisaan - quickly, sirritti - correctly

Some examples of adverbs of measure:

Baay'ee, danuu - much, many, very, duwwaa - only, empty.

In the following examples, each of the bold words is an adverb:

*Obboleessi Koo **kaleessa** dhufe. [My Brother came yesterday.]*

*Amma hojiin qaba **booda** kotu. [Now I am busy, come back later.]*

Adjectives

An adjective describes or modifies nouns and pronouns in a sentence. It normally indicates quality, size, shape, duration, feelings, contents, and more about a noun or pronoun. Adjectives usually provide relevant information about the nouns pronouns they modify/describe by answering the questions: *What kind? How many? Which one? How much?* Adjectives enrich writing by adding precision and originality to it [22].The masculine form terminates in one of the following suffixes – *aa*, *-eessa*, or -(a)acha, and the feminine form terminates in one of the following suffixes –*oo*, *-tuu*, *-eettii*, or –*aattii*.

Example: *Tollasaan furdaa (m) dha. [Tolesa is fat]*

Hawwiin qal'o (f) dha. [Hawi is skinny]

The number form of adjectives in the plural occurred by reduplication of first syllable.

Example: *Guddaa (m) (s) Guddoo (f) (s)*

Guguddaa (m) (p) Guguddoo (f) (p)

Preposition

In Afaan Oromo Postpositions are more common in Afaan Oromo than postpositions [23]. T he most common are:

akka - according to, like, as. eega - since, from, after. eegasu - in that case, therefore

gara - in the direction, towards, side. haga – until. hamma - upto, until such that, as much as.

Example, *Akka isaa jabaan namu hinjiru. [There is no one as strong as he is]*

Inni gara manaa deema. [He goes (towards) home]

Postpositions

Unlike European languages, Afan Oromo language uses frequently postpositions [23]. The most common postpositions in Afaan oromo are:

ala - out, outside. bira - besides, booda - after jala – under

Postpositions can be grouped into suffixed and independent words.

a. Suffixed postpositions (**-tti** *in, at, to,* **-rra/irra** *on,* **-rraa/irraa** *out of, from*)
   Example: *Adaamaatti yoom deebina? - When shall we go back to Adama?*

   Gammachuun sireerra ciise. - Gemechu lay down on bed.

b. Post position as independent words

   (**Ala** *outside,* **wajjiin** *with, together with,* **bira** *besides,* **teellaa** *behind*)

   Example: *Namoota nu bira jiraniis hin jeeqnu. - We do not hurt people who are with us.*

Pronoun

A pronoun is a term that is used instead of a noun or noun phrase in Afaan Oromo, as it is in other languages. They are classified depending on their number and gender. Based on their presence in a phrase, the pronoun in Afaan Oromo might be autonomous or concealed with the verb. Independent pronouns are pronouns that occur in a sentence as a distinct word. In the following example, "*Inni*" is an independent personal pronoun.

Example: *Yohaannis yeroo dhufe homaa hin nyaatu homaas hin dhugu ture; Isaanis* ***Inni*** *dhukuba qaba jedhu.*

However, hidden pronouns are pronouns attached to the word in the sentence. In the following example, the word "*deemte*" indicates the pronoun "*Ishee*" hidden in it.

Example: *Toltuun kaleessa magaala Finfinnee* ***deemte*** *ture.*

2.2.3 Afaan Oromo Morphology

Morphology is a branch of linguistic that studies and describes the internal structure of words and how words are formed in a language [24]. Afaan Oromo, like many other African and Ethiopian languages, has an extremely complex and rich morphology. Jurafsky and Martin [24] defined morphology as the study of the way word is built up from smaller meaningful units called morphemes. Word is the most basic unit of linguistic structure. Like other Ethiopian languages, Afaan Oromo has complex and rich morphology.

Types of Morphology in Afaan Oromo

In Afaan Oromo language, we have two broad types of morphology namely derivational morphology and inflectional morphology [25].

I. Inflectional Morphology

Inflectional morphology deals with combination of a word stem with a grammatical morpheme in the same word class. In inflectional morphology, inflectional morphemes, morphemes that serve a purely grammatical function, which never create a new word but only a different form of the same word, are added in words. It occurs in the form of different word classes. Inflection of verbs, Inflection of Nouns, and Inflection of Adjectives [25].

Inflection of Nouns - Many of Afaan Oromo nouns end with a vowel except few which ends with consonants like *n, l, t* [25]. Inflectional categories under nouns exists mainly in the forms of marking number, definite and gender.

In Number, marking inflections distinguish plural and singular. Several types of suffixes are attached to nouns to make plural forms.

Example: The plural –(o)ota attached on Nouns

Barataa [Base form], baratt-oota [Inflected form] students

The plural –lee attached on Nouns

Jabbii [Base form], jabb-iilee [Inflected form] Calves

In definite marker or singulative marker shows noun is marked for being used as single form.

Example: *sooressa [Base form,], sorricha [Inflected form] The rich man*

In gender marker, we use inflection to identify masculine and feminine through gender suffixes.

Example, boonaa [base form], boont-aa [inflected form -m] / proud boy boont-uu [inflected form –f]/proud girl

Inflection of Verbs - Verbs are the most classes in which inflection are occurred. Mainly verb inflection happens in the form of inherent and agreement properties. Inherent properties are a verb inflection that triggers inflection on that word class includes aspect, mood, and voice. However, agreement properties indicate inflection of word class for properties out of its members include person, number, gender, and case. In the Afaan Oromo language, the roots or stems of verbs, usually end in consonant, take inflectional morphemes showing distinction between aspects or mood or gender or number.

Example, qab – [root form] qab-eenya [Inflected form] Property

Rak – [root form] rak-kina [inflected form] Problem

Falm – [root form] falm-ii [inflected form] Argument

Inflection of Adjectives - are the same with that of nouns. Adjectives are inflected for number, gender and singulatives like nouns. If adjectives occur within sentences, number is marked on both. Inflection for number of adjectives occurred in the form lexical, reduplication and–(o)ota.

Example: Inflection for Numbers

Lexical: Sooressa (s)	*Sooreeyyi (p)*
Reduplication: guddaa (s)	*gud-guddoo (p)*
-(o)ota: hamaa(s)	*ham-oota (p)*

In Afaan Oromo, the base forms of adjectives are normal to be used as masculine, but inflection occurs when we make them for feminine.

Example: *Hamaa (m)* *Hamtuu (f)*

In Afan Oromo, singulative markers are not used on both noun and adjective at the same time. Which means, when nouns are marked, adjective is not and vice versa.

Example: *Muk-ni(n)* *dheer-icha(adj.).* *The long stick*

II. Derivational Morphology

Derivational morphology deals with combination of a word stem with a grammatical morpheme that yields different word class. Thus, in derivational morphology, there are methods of forming new lexemes from already existing ones by affixing derivational morphemes, morphemes that change the meaning or lexical category of the words to which they are attached. Different derivational suffixes are attached to the root or stem of the word [25].

It occurs in the form of different word classes. Derivation of verbs, Derivation of Nouns and Derivation of Adjectives.

Derivation of verbs – is the creation of new verb word class from other given word class stem.

Example: *arguu [base form], / to see argachuu [derivate form] / to get, find*

Derivation of Nouns – is the creation of new noun word class from other given word class stem.

Example: bulchuu [base form] / to administer bulchiinsa [derivate form] / administration

Derivation of Adjectives – is the creation of new adjectives from nouns or from other adjectives or from verbs.

Example: jabaa [base form] / strong jabaacuu [derivate form] / to be strong

2.3 WordNet

A WordNet is a lexical database for languages based upon a structure introduced by the Princeton WordNet (PWN) for English in which sets of cognitive synonyms, or synsets, are interconnected with arcs standing for semantic and lexical relations between them [5]. The WordNet categorizes nouns, verbs, adjectives, and adverbs into synonym groups and analyzes the links between them. The most ambitious feature of WordNet, however, is its attempt to organize lexical information in terms of word meanings, rather than word forms. In that respect, WordNet resembles a thesaurus more than a dictionary.

The most obvious difference between WordNet and a standard dictionary is that WordNet divides the lexicon into five categories: nouns, verbs, adjectives, adverbs, and function words and standard dictionary organizes words based on lexical similarity between words [5]. In computational linguistics, information retrieval, and machine translation, WordNetsare frequently utilized. Building one manually is time-consuming and complex, prompting a quest for automated or semi-automatic ways.

WordNet serves as the foundation for numerous natural language processing (NLP) research projects [26]. In this part, we will go through the specifics of WordNet's structure and components, as well as word/sense embedding and its many techniques.

WordNet and Word Relationships

The goal with WordNet is to organize word senses by meaning, rather than alphabetically. This poses a problem since a word can have multiple meanings, and multiple words can have the same meaning. To resolve this, there is a many-to-many relationship between words senses

and the concepts they represent. Words are organized into synsets, which represent concepts. Each word from a synset is a synonym of another word. Polysemous words can have their different senses in different synsets. Four different parts of speech are recognized in WordNet: nouns, verbs, adjectives, and adverbs.

Synonym WordNet Relation

When two senses of two different words (lemmas) are identical, or nearly identical, we say the two senses are synonyms. Synonyms include such pairs as

couch/sofa, vomit/throw up, car/automobile

The word synonym is commonly used to describe a relationship of approximate or rough synonymy. But furthermore, synonymy is a relationship between senses rather than words. Considering the words big and large. These words may seem to be synonyms in the following sentences, since we could swap big and large in either sentence or retain the same meaning:

How big is that plane?

Would I be flying on a large or small plane?

But note the following sentence in which we cannot substitute large for big:

Miss Caaltu, for instance, became a kind of big sister to Toola.

Miss Caaltu, for instance, became a kind of large sister to Toola.

This is because the word big has a sense that means being older or grown up, while large lacks this sense. Thus, we say that some senses of big and large are (nearly) synonymous while other ones are not.

A synonym can be classified into lexical and propositional synonymy, or into lexical, phrasal, and propositional synonymy. The focus of this thesis is lexical synonymy and words that belong in the same class. Synonymy has a different scale of similarity. It might be near-synonymy or absolute synonymy, or it might be cognitive synonymy as in [27].

According to the Maja's [27] description, if a word is completely similar in all aspects of meanings of words in every context, then it is called an absolute synonymy. In cognitive synonym, most semanticists would regard as synonymy. Near-synonym are wording whose

meaning is relatively less or more similar which is associated with overlapping the sense and meaning. These types of synonyms are mostly found in dictionaries of synonym or thesauri.

Hypernym WordNet Relation

Another way word senses can be related is taxonomically. A word (or sense) is a hyponym of another word or sense if the first is more specific, denoting a subclass of the other. For example, car is a hyponym of vehicle, dog is a hyponym of animal, and mango is a hyponym of fruit. Conversely, we say that vehicle is a hypernym of car, and animal is a hypernym of dog.

The relationship between hyponym and hypernym is defined as a hypernym is a generic term whereas a specific instance of hypernym is called hyponym. Since hypernym and hyponym always represent a class and subclass respectively, a hyponym is a phrase or word, and its semantic field is more specific than the hypernym (its hypernym). The semantic hierarchy of a hypernym relation is always wider than a hyponym relation; this is also called as a hypernym is superordinate. Hypernym relation is considered as that consists of the relation hyponym.

We can define hypernymy more formally by saying that the class denoted by the hypernym extensionally includes the class denoted by the hyponym. Thus, the class of animals includes as members of dogs, and the class of moving actions includes all walking actions. Hypernymy can also be defined in terms of entailment. Under this definition, a sense A is a hyponym of a sense B if everything that is A is also B, and hence being an A entails being a B, or $\forall x\, A(x) \Rightarrow B(x)$. Hyponymy/hypernymy is usually a transitive relation; if A is a hyponym of B and B is a hyponym of C, then A is a hyponym of C. Another name for the hypernym/hyponym structure is the IS-A hierarchy, in which we say A IS-A B, or B subsumes A.

Hypernymy is useful for tasks like textual entailment or question answering, knowing that leukemia is a type of cancer, for example, would certainly be useful in answering questions about leukemia.

Antonym Wordnet Relation

Whereas synonyms are words with identical or similar meanings, antonyms are words with an opposite meaning, like:

long/short, big/little, fast/slow, cold/hot, dark/light

rise/fall, in/out

Two senses can be antonyms if they define a binary opposition or are at opposite ends of some scale. This is the case for long/short, fast/slow, or big/little, which are at opposite ends of the length or size scale. Another group of antonyms, reversive, describe change or movement in opposite directions, such as rise/fall or up/down.

Antonyms thus differ completely with respect to one aspect of their meaning their position on a scale or their direction but are otherwise very similar, sharing almost all other aspects of meaning. Thus, automatically distinguishing synonyms from antonyms can be difficult. The assumption regarding antonymy is that, although being stated to be opposites, antonymous terms are semantically extremely similar since they share the same dimension. Antonymy can be found in the same phrase, for example, in contrastive formulations like "either x or y." In most instances, straight antonyms (such as cold/hot, strong/weak) are interchangeable. An antonym is a pair of words that have opposing or opposed meanings in the dictionary's thesaurus. Cosine values separates synonym from antonym, with antonym in the lower-value and synonym in the higher-value partition.

2.4 Word Embedding

2.4.1 Neural Language Modelling

A language model is a statistical model which, given the beginning of a sequence of words (e.g., *"Why did the"*, if the full sequence is *"Why did the chicken cross the road?"*), outputs a probability distribution over the vocabulary corresponding to its estimate of the probability of each word in the vocabulary being the next to occur in the sequence. Formally, given words $w1, w2; \ldots., wt - 1 \epsilon\ V$, a language model outputs a probability distribution over V for wt. Language modelling can be viewed as a multiclass classification problem, where the examples are sequences of words, and the classes are the words in the vocabulary. However, language modelling is considered an unsupervised task because examples can be constructed from an unlabeled text corpus.

For a long time, the dominant approach to language modelling was the N-gram approach, introduced by Shannon [28]. An N-gram is a unique sequence of N words. The probability predicted by an N-gram model of a given word occurring next in a sequence is the probability observed in the corpus of the word occurring, conditioned on the previous $N - 1$ tokens. A serious drawback of N-gram models is that they do not generalize well to previously unseen sequences; they tend to assign high probability only to sequences that have occurred in the training corpus, but as the number of plausible N-grams is exponentially large, even a large corpus can only contain a small subset of them.

A competing approach, neural language modelling, was proposed in 2000 by Bengio *et al.* [29]. The word neural denotes the use of continuous representations of word sequences rather than discrete representations such as N-grams, and specifically the use of artificial neural networks.

Bengio *et al.* [29]'s model consists of two components: an embedding matrix $E \epsilon \mathbb{R}|V|x\ d$ where d is the embedding dimensionality, and a probability function f, a neural network which takes as input the embeddings of the previous words in the sequence and outputs an estimated probability distribution over next words.

In order to obtain a fixed-sized input to f, Bengio *et al.* [29]'s model considers only the n previous words $wt - n,\ wt - n + 1, \ldots . wt - 1$ when predicting wt. The input representation $x \epsilon \mathbb{R}nd$ is found by concatenating the embeddings of these words, i.e., $x = [\mathrm{e}wt - n;\ \mathrm{e}wt - n + 1;\ \ldots .;\ \mathrm{e}wt - 1]$, where *ei* is the ith row of E. f is a feed-forward neural network with a single hidden layer (with skip connection) and a SoftMax output layer yielding a vector of length $|V|$, the elements of which correspond to the words in the vocabulary

$$y = b + Wx + U \tanh\ (d + Hx) \qquad (1)$$

$$f = softmax(y) \qquad (2)$$

Where the parameters $\theta = (\mathrm{b}, \mathrm{d}, \mathrm{W}, \mathrm{U}, \mathrm{H}, \mathrm{E}))$ are adjusted through stochastic gradient ascent to maximize the mean log-likelihood J of the words contained in the corpus, i.e.

$$J(\theta) = \frac{1}{T}\sum_t \log[f(wt - n, \ldots, wt - 1\ ;\ \theta)]\ wt + R(\theta) \qquad (3)$$

where T is the total number of words in the corpus and $R(\theta)$ is a regularization term.

Bengio *et al.* [29]'s model achieved state-of-the-art performance by a significant margin over the previous N-gram models. The use of word embeddings enabled the model to generalize in ways the N-gram models could not. Bengio *et al.* [29] offer the following explanation for this:

If we knew that dog and cat played similar roles (semantically and syntactically), and similarly for (the, a), (bedroom, room), (is, was), (running, walking), we could naturally generalize (i.e., transfer probability mass) from

The cat is walking in the bedroom

to *A dog was running in a room*

and likewise, to *the cat is running in a room*

A dog is walking in a bedroom

The dog was walking in the room

…..

and many other combinations. In the proposed model, it will so generalize because "similar" words are expected to have a similar feature vector, and because the probability function is a smooth function of these feature values, a small change in the features will induce a small change in the probability. Therefore, the presence of only one of the above sentences in the training data will increase the probability, not only of that sentence, but also of its combinatorial number of "neighbors" in sentence space (as represented by sequences of feature vectors / embeddings).

2.4.2 Neural Word Embedding

Neural word embedding is a set of language modeling techniques in which words or phrases from the vocabulary are mapped to vectors of real numbers in a low-dimensional space. Traditional word embedding methods adopt dimension reduction methods (such as SVD and PCA) on the word co-occurrence matrix. Comparatively, recent methods use discriminative learning, such that the vector of the target word is updated by minimizing its distance from the vectors representing its context words and maximizing its distance from the vectors of

other words. In this Section, we will introduce several state-of-the-art predictions based embedding models (Skip-gram, CBOW and GloVe).

Word to vector

After latent semantic analysis (LSA) being popularized in the early 1990s, word embeddings have gained attention again, when word2vec was released in 2013 [30]. Word2vec is one of the words embedding approaches that use neural networks for producing a vector representation of words. In the modern days, an artificial neural network is mostly structured by three layers (input, hidden, output) containing elementary units called artificial neurons or nodes. Each node receives one or more inputs that are often weighted, and then the sum of the inputs is passed into a non-linear function known as activation function. The activation function defines the output of a node given an input or set of inputs. It also maps the resulting values into desired ranges such as between 0 to 1 or -1 to 1. Its output is then used as input for the next node and so on, until a desired solution to the original problem is found.

A subfield of artificial intelligence, machine learning (ML), aims to effectively perform a specific computational task with the use of models and inference instead of giving explicit instructions. These models rely on algorithms that are used to predict the solution for a problem based on a set of examples of problems (data entities) often together with their respective solutions (labels). This set of sample data is also known as training data. The most important difference between applications based on machine learning techniques and regular programming is that in ML, the system needs to know what must be solved and not explicitly how to solve the problem. Many ML algorithms use neural networks to train data to make predictions or decisions without the need of custom programming for that specific topic. Most of ML techniques apply feature extraction, which is a dimensionality reduction process, where an initial set of raw variables is reduced to more manageable groups (features) for processing, while still accurately and completely describing the original data set. This process results in a feature vector, what can be comparable to the word vectors generated by word2vec.

Basically, word2vec is used for finding semantic similarity among words. A list (corpus) of lists of words (documents) is given as input, and the output is a list of vectors of same dimension, each containing values between -2 and 2 for every dimension. Each of these vectors represent one unique word present in the input list. Two or more vectors can then

easily be compared via cosine similarity, whose result defines the semantic relatedness between the words assigned to these vectors [30]. Word2vec has a set of hyper-parameters that can be refined to train different kind of datasets.

Word2vec is a set of two independent shallow, two-layered neural networks: continuous bag of words (CBOW) and skip-gram. In these models, a window of a given size moves along the corpus, and in every step the language model is trained on the words within this context span. In essence, the CBOW algorithm predicts a word based on a given context while the skip-gram predicts a context based on a given word [30].

Skip-gram Model

The skip-gram model [11] learns word embeddings such that they are useful for predicting the context words. As shown in Figure 1, skip-gram model is a fully connected neural network with one hidden layer. The input and output layers are vocabulary-sized sparse vectors. The input layer is one-hot representation that only the position representing the target word is 1 and the other positions are all 0s. The output layer is a probability distribution of outputting each word in the vocabulary given the input word. The reference output layer is also one-hot representation, and the errors at this layer are calculated based on the difference between the actual output and the reference. The hidden layer is a N-dimensional dense vector which is a row vector of matrix $Wv \times N$ (by product the input layer with the matrix). All predicting words share the same matrix $Wn \times V$.

Given the input word to be the ith word in the vocabulary, the probability of outputting the jth word in the context (positive case) is given by Equation 4, where $vec(wi)$ is the ith row vector of $Wv \times N$ and $vec(wj)$ is the jth column vector of $Wn \times V$.

$$\mathrm{P}\big(D = 1 \big| vec(wi),\ vec(wj)\big) = \frac{1}{1 + e^{vec(wi)T\ vec(wj)}} \tag{4}$$

Similarly, the probability of outputting the jth word which is not in the context (negative case) is given by Equation 5

$$P(D = 0|vec(wi), vec(wj)) = 1 - P(D = 1|vec(wi), vec(wj)) \tag{5}$$

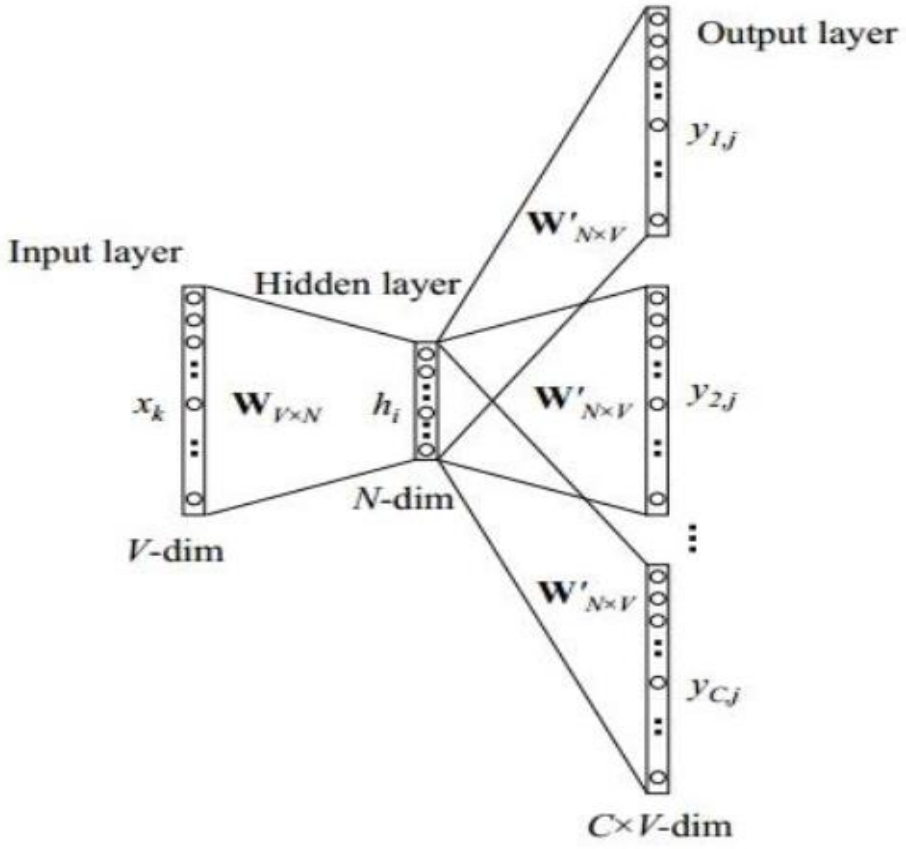


Figure 2.1: The Skip-Gram Model [11]

Given a training set containing the sequence of word tokens $w1, w2, \ldots, wT$, the word embeddings are learned by maximizing the following objective function:

$$J(W_{V\,X\,N}, W_{N\,X\,V}) = \sum_{i=1}^{T}[\sum_{c\epsilon Ci} P\big(D = 1\big|vec(wi), vec(c)\big) + \sum_{c\,\epsilon V-Ci} P(D = 0|vec(wi), (c))] \quad (6)$$

where $W_{V\,X\,N}$ and $W_{N\,X\,V}$ (mentioned before) are parameters, i is the position of the target word wi in the sequence, Ci represents the context of word wi, each c is a word appearing in the context of wi and each c' is a word not appearing in the context.

To make the training faster, Stochastic Gradient Descent (SGD) is adopted on each pair of target word and context. And the error on the output layer is backpropagated through the whole network. The adapted target function is shown in Equation 7, where wi is the target word, Ci is the set of the context words and V represents the whole vocabulary.

$$J(W_{V\,X\,N}, W_{N\,X\,V}) = \sum_{c \in Ci} P\big(D = 1 \big| vec(wi), vec(c)\big) + \sum_{c \in V - Ci} P(D = 0 | vec(wi), vec(\acute{c})) \tag{7}$$

In addition to the model, there are several techniques that have been proved helpful by later research [31]. One method is called "dynamic window size". For each training word wi, the set of context words includes Ni words to the left and right of wi. Here Ni, the window size for wi, is uniformly sampled from an integer list from 1 to N, where N is the manually set maximum context window size. This method highlights the influence of the surrounding words because the vector of the target word is updated more towards them comparing with the words that are relatively farther but are still within the range of N. Another method is called "sub-sampling" which is adopted to diminish the excessive influence of frequent words. Frequent words are usually function words and do not have much information for distinguishing different words. Here each word w has a chance to be dropped out with a probability as shown in Equation 8, where $punigram(w)$ is the unigram distribution of w and Z is the normalization factor.

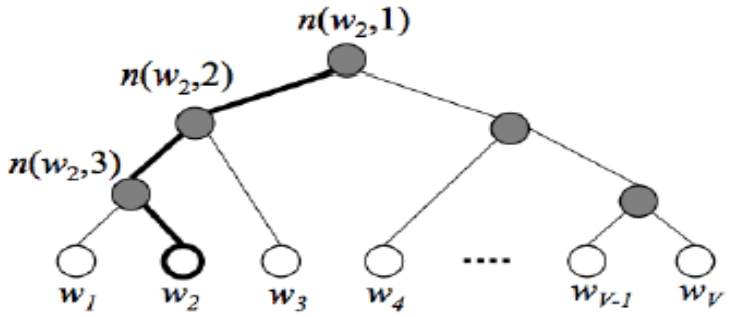


Figure 2:2 A binary tree for the hierarchical SoftMax [11]

$$P(w) = \frac{Punigram(w)^{3/4}}{Z} \quad (8)$$

Continues Bag of Word Model

Continuous Bag-of-Word (CBOW) Model is the reverse version of Skip-gram model that it takes a bag-of-word context as the input and outputs the target word. Shown in Figure 3, $x1k, x2k, \ldots, xCk$ are the input context words that each is fully connected with the hidden layer and shares the same matrix $W_{|V| \, X \, N}$. The hidden layer is fully connected with the output layer with parameter $W_{N \times |V|}$. Comparing with Skip-gram model which the hidden layer is just the vector of the target word, in this model the hidden layer is the average of vectors of all input words.

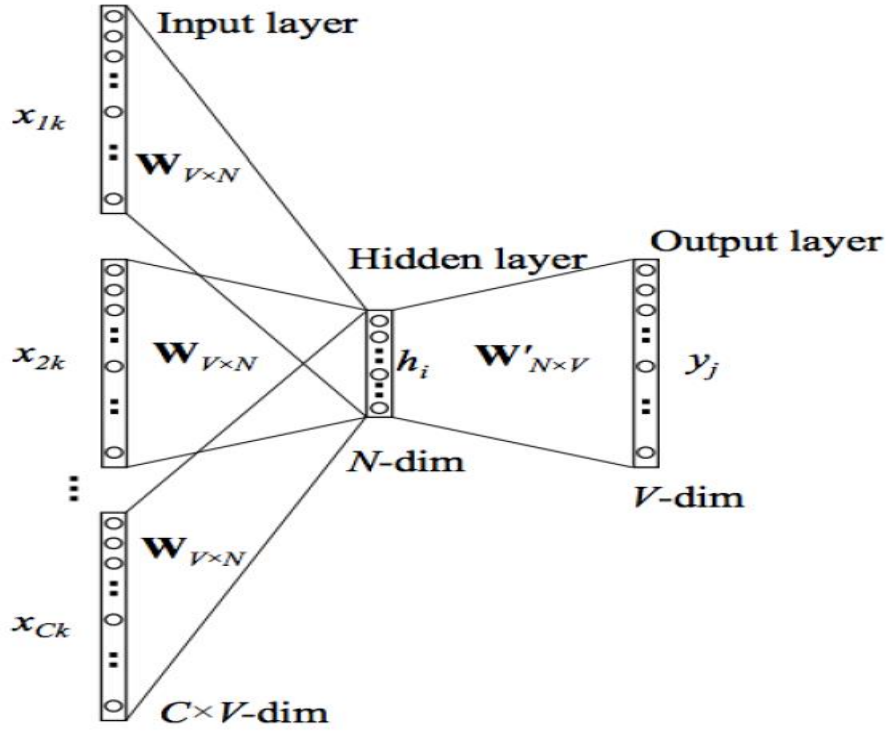


Figure 2:3 The Continuous Bag-of-Word (CBOW) Model [11]

GloVe: Global Vectors for Word Representation Model

Both Skip-gram and CBOW learn word embeddings from each local word and context pair. GloVe proposes another way to learn word embeddings directly from an aggregated global word-word co-occurrence matrix [32]. This work is based on the intuition that similar words should have similar co-occurrence numbers with other words. In this method, they first make a co-occurrence matrix basing on Equation 13 that sums over all co-occurrence instances that wj is within the N-word window of wj . Here $dist(\cdot,\cdot)$ is the distance function that returns the position difference between the two arguments.

$$X_{i,j} = \sum_{wi,wj} \frac{1}{dist(wi,wj)} \tag{9}$$

We directly show the objective of this model which is:

$$J = \sum_{i,j} f(Xij)(W_i^T wj - \log Xij)^2 \tag{10}$$

where $f(Xij)$ is the weight function for each pair of words, and it is defined as:

$$f(X_{ij}) = \begin{cases} \left(X_{ij}/X_{max}\right)^2 \; if \; X_{ij} < X_{max} \\ 1 \; otherwise \end{cases} \tag{11}$$

The training algorithm iterates through each element Xij of co-occurrence X and does update.

2.4.3 Frequency based Embedding

This is the very basic, easy, and fast method to word vectors. It works based on the count of a word in each document. It can be Count Vector, TF-IDF vector, or Co-Occurrence vector.

Count Vector

Count vector model learns a vocabulary from all the documents, then models each document by counting the number of times each word appears [33]. For example, consider we have ***D*** documents and ***T*** is the number of different words in our vocabulary then the size of count vector matrix will be given by $D * T$. Let's take the following two sentences:

Document 1: "The cat sat on the hat"

Document 2: "The dog ate the cat and the hat"

From these two documents, our vocabulary is as follows:

{the, cat, sat, on, hat, dog, ate, and}

so, D = 2, T = 8

Now, we count the number of times each word occurs in each document. In Document 1, "the" appears twice, and "cat", "sat", "on", and "hat" each appear once, so the feature vector for the documents is:

{the, cat, sat, on, hat, dog, ate, and}

So, the count vector matrix is: - is given in Table 2.1

Table 2.1 Count vector matrix

	The	Cat	Sat	On	Hat	Dog	Ate	and
Doc1	2	1	1	1	1	0	0	0
Doc2	3	1	0	0	1	1	1	1

Now, a column can also be understood as word vector for the corresponding word in the matrix M. For example, the word vector for 'cat' in the above matrix is [1,1] and so on. Here, the *rows* correspond to the *documents* in the corpus and the *columns* correspond to the *tokens* in the dictionary. The second row in the above matrix may be read as — Document 2 contains 'hat': once, 'dog': once and 'the' thrice and so on.

Since there is a problem related to dimensions of a matrix for a large corpus of text, we can use stop words (remove common words like ‘a, an, this, that’) or we can extract some top words from vocabulary based on frequency and use as a new vocabulary or we can use both methods.

TF-IDF Vector

In a large text corpus, some words will be very frequent (e.g., “the”, “a”, “is” in English). Hence carrying very little meaningful information about the actual contents of the document. If we were to feed the direct count data directly to a classifier, those very frequent terms would shadow the frequencies of rarer yet more interesting terms.

To re-weight the count features into floating point values suitable for usage by a classifier it is very common to use the TF-IDF transform. This method considers not just the occurrence of a word in a single document but in the entire corpus. Let’s take a business article that contain more business-related terms like stock-market, prices, shares etc. in comparison to any other article. but terms like “a, an, the” will come in each article with high frequency. So, this method will penalize these type of high frequency words.

Co-Occurrence Vector

The approach of words co-occurrence matrix describes how words occur together. Contextually related words tend to appear together, and this model reflects their connections. It computes the words co-occurrence matrix by counting the number of word co-occurrences in each text corpus. It initially retains the semantic connection between words to create more accurate word vector representations. The disadvantage is that storing the co-occurrence matrix requires a large amount of memory, and factorizing the matrix is not possible in the system.

2.4.4 Prediction-Based Word Embedding

Prediction-based word embedding models are those which, during training, use word embeddings to make local predictions within the corpus, such as predicting occurrences of words given their contexts, or predicting the context given a word contained within it. Count-based models on the other hand utilize global cooccurrence statistics [34]. For instance, the

popular GloVe method [35] operates on a matrix where the i, jth entry denotes the number of times words i and j occur within a certain distance of each other in the corpus.

2.4.5 Probability and Language

A fundamental concept in prediction-based word embedding models is that of word probability. Word probability is often thought of in relation to a corpus. The simplest notion of word probability is unigram probability. The unigram probability Pw of a word w is the probability that a word randomly sampled from the corpus is w. Formally,

$$Pw = \frac{fw}{\Sigma_v \varepsilon v f_v} \tag{12}$$

where V denotes the vocabulary and fv denotes the number of times token v occurs in the corpus.

The concept of word probability is more interesting when it is conditioned on a context. Consider the following question for example: if a sentence starts "Why did the...," then how likely is it that the next word will be "chicken"? This probability can be written informally as

$$\text{P(“chicken” | “Why did the…”).}$$

In general, the probability of a word w occurring in a context c can be written as

$$\text{P(w | c).} \tag{13}$$

The use of probabilities in linguistics has been dismissed by the pioneering linguist Chomsky [36] writing "*But it must be recognized that the notion 'probability of a sentence' is an entirely useless one, under any known interpretation of this term.*". Chomsky [37] provides evidence of this in the form of the following two sentences:

Colorless green ideas sleep furiously.

Furiously sleep ideas green colorless.

While both sentences are nonsensical, the first is a grammatical sentence of English and the second is not. Since it is almost certain that neither has occurred before in human discourse,

Chomsky argues that a statistical model ought to assign a probability of zero, failing to make any distinction between the sentences despite this fundamental difference.

Chomsky's argument is predicated on a frequentist view of probability, where the probability of a sequence of words is proportional to the number of times that sequence has occurred previously. However, the statistical models used in NLP are capable of generalizing to unseen sequences of words. As we will see, the use of word embeddings is one way to enable effective generalization of this type

2.5 Word Sense Embedding

A fundamental objection to the notion of word embeddings is that many words have more than one meaning, or sense. Homonymous words, such as bark, whose meanings are semantically unrelated, can be considered to consist of several, separate entries in the lexicon which happen to share the same spelling and phonological realization. Therefore, from a theoretical point of view it seems no more reasonable to represent with a single mathematical representation the meanings of the word bark (the sound made by a dog, and the outer layer of a tree) than it would to be to represent the words meow and leaf similarly, and yet this is what word embeddings do. This drawback is referred to as the meaning conflation deficiency [38]. A similar but less severe effect results from the existence of polysemous words, or those which have several meanings which share some semantic similarity. The polysemous word cell, for instance, can refer to a small room such as a prison cell, or a structural unit of a living organism. The two meanings both refer to enclosed, self-contained units that are part of a larger structure. We will take the term "polysemy" to encompass both homonymy and polysemy in its strict sense.

Intuitively, sense embedding is a fine-grain version of word embedding by clustering different contexts for each target word. For example, "windows" can appear either with "Microsoft", "Apple", "iPhone" or with "House", "Car". And the word embeddings of "Microsoft", "Apple" and "iPhone" are different from the word embeddings of "House" and "Car". For example, word "bank" generally has two meanings which are "a slope of land near the water" and "a finance institute", the sense embedding can be initialized as following:

$$v_s('bank'_1) = vg('slope') + vg('land') + vg('water')$$

$$v_s('bank')_2 = vg('finance') + vg('institution')$$

The meaning conflation deficiency results in distortion in word embeddings. It is common to estimate the degree of similarity between two words by comparing their vectors [38]. Suppose that d is a distance function, and that $d(left, right)$ denotes the distance between the embeddings of the words $left\ and\ right$. In Figure 2.4 the word $right$ is closely related to both the words $left$ and $wrong$, so we would expect both $d(right, left)$ and $d(right, wrong)$ to be small. But by the sub additive property of distance, we have

$$d(left, wrong) \leq d(right, left) + d(right, wrong) \qquad (14)$$

Therefore, we would expect a word embedding system to predict a relatively high degree of similarity between the words left and wrong, although they are in fact unrelated. This effect is illustrated in Figure 2.4.

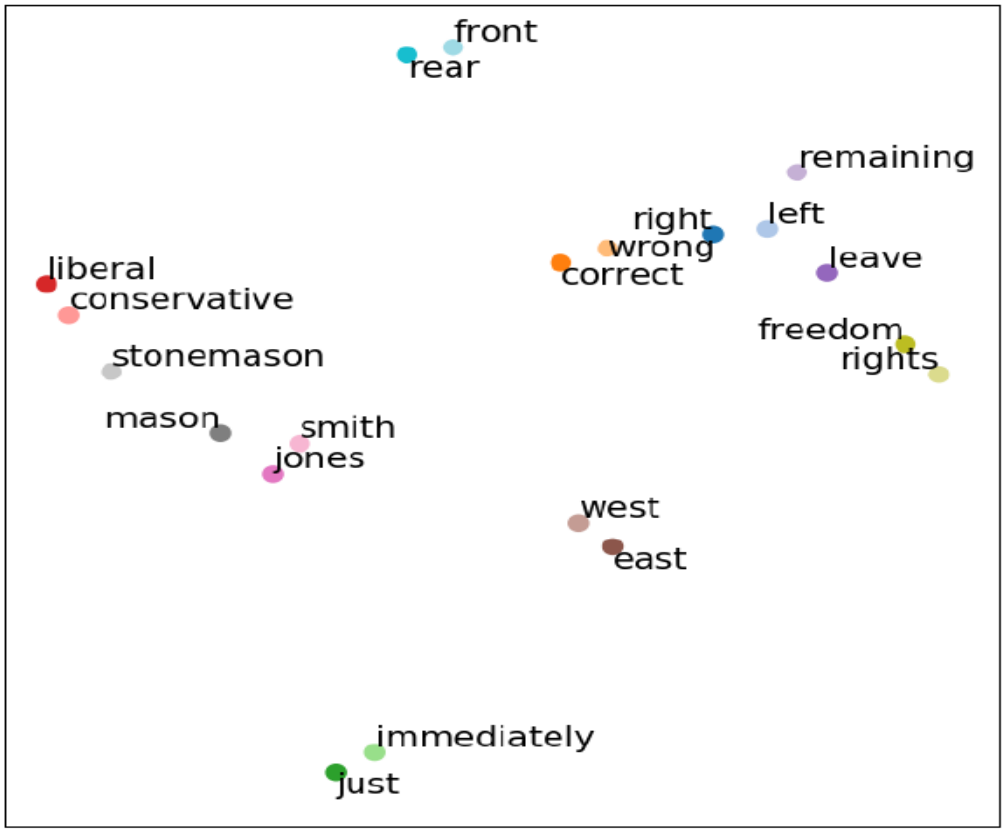


Figure 2.4 Word embeddings for several polysemous words, visualized using t-SNE and adjustText The occurrence of closely related polysemous words nearby in the word embedding space (i.e., left, and right) causes unrelated words to be closer together (e.g., left, and wrong) and related words to be further apart (e.g., right, and east) that they otherwise would be.

Unfortunately, there has been little work examining the extent to which the meaning conflation deficiency harms the performance of NLP systems which utilize word embeddings. Jurafsky [39] proposed a system for learning embeddings for individual word senses ("sense embeddings") and a pipeline for substituting these into NLP systems in place of word embeddings. They found that for some natural language understanding tasks, their sense embeddings outperformed word embeddings of equal dimensionality, but that similar performance gains could also be achieved by increasing the word embedding dimensionality.

Sense embedding methods learn embeddings for senses rather than words sense embedding methods learn embeddings for senses rather than words. According to the ways for learning senses, existing sense embedding methods can be roughly divided into clustering-based, non-parametric and ontology based. Clustering-based methods [40, 41] learns a fixed number of senses for each polysemous word. In addition, nonparametric [42, 43] dynamically decide the number of senses by a non-parametric process. Both methods treat a group of similar contexts of a word as a sense following the distributional hypothesis that the word meaning is reflected by a set of contexts where it appears. Finally, ontology-based methods [44, 45, 46] learn senses according to an existing sense inventory such as WordNet.

Clustering-base Methods

Reisinger and Mooney [40] made the first attempt to learn sense embedding. Based on the standard vector-space model of word, their approach represents each word token wi as a co-occurrence vector $vec(wi)$ that records all unigrams occurring within the windows around wi. To learn sense vectors, their approach first clusters all occurrences (such as wi) of word type w into K clusters, where K is a fixed constant. Then, the similarity between two-word types is calculated as a function of their cluster centroids.

Huang *et al* [41] proposed a neural network-based method. They first learn distributed word vectors by discriminating the next word given a word sequence with a recursive neural network. Then they cluster all occurrences of each word and re-label each occurrence according to the associated cluster. Finally, the sense embeddings are learnt by feeding the re-labeled corpus to the same neural network. In their recursive neural network, the scoring

components are computed by two sub neural networks, one capturing the local context and the other capturing the global context.

Non-parametric Sense Embedding Methods

Intuitively, it is problematic to assign a fixed number of senses for every word. Generally, some words like “beat” have several meanings, words like “people” may have only one meaning and functional words should have only one meaning. However, clustering-based methods such as MSSG assign an equal number of senses to each word regardless of the above situations. Non-parametric processes are naturally helpful for generating a different number of senses for each word.

Probably, the NP-MSSG model of Neelakantan *et al* [42] is the first work that addresses the task of learning a varying number of senses for different words. Generally, their method creates a new sense for a word type with the probability which is proportional to the distance from the context to the nearest sense. Initially, there is no sense for each word type, as the number of senses for each word type is unknown. Then the first sense for a word type is created from the first context in which it appears, and the first context is also assigned to the first sense.

Ontology-based Methods

Even though clustering-based and non-parametric sense embedding methods have demonstrated good performance, they are not applicable to the down-streaming NLP applications that use WordNet-based senses. Another motivation for ontology-based methods is that there are well developed sense inventories (such as WordNet) containing not only the definitions and glosses but also hyponym, hypernym, and synonym relations. Utilizing these resources may further improve the quality of sense embeddings.

Ontology-based methods are roughly divided into two categories: One kind of methods use a sense inventory for initialization that they initialize sense vectors by the definition, gloss, and relations between senses, then fine tune the vectors with a large plain corpus. The other kind of methods use the sense inventory as constraint that they transfer the sense embedding learning problem into an optimization problem under constraints.

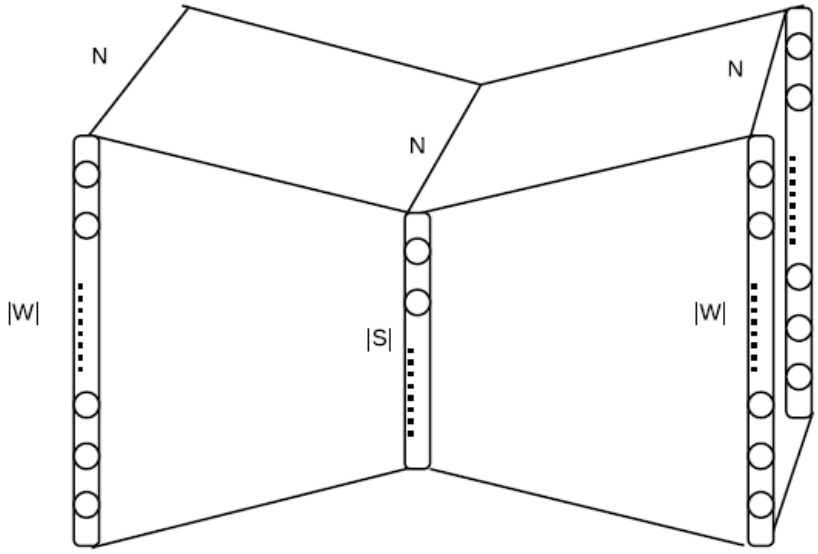


Figure 2.5 The Neural Network Structure for Sense Embedding Learning [79]

2.6 Evaluation of WordNet Construction

One of the biggest issues for the automatic construction of WordNets is how to evaluate their accuracy and/or precision properly and effectively. Across all the different lexical resource and word and/or sense embedding-based approaches to automatic synsets construction described in the previous sections, evaluation methods can be split across two types:

- Comparison against a reference WordNet
- Manual evaluations against fixed samples of automatically constructed synsets.

Focusing first on comparisons with reference WordNets, many of the research referenced in the preceding sections reports on problems with this kind of evaluation. The discrepancies between size and coverage of reference WordNetsand the original PWN can be viewed as an issue of granularity: PWN is large enough that its senses are fine-grained, and so several PWN synsets can generally mapped onto one synsets in a reference WordNet (such as FarsNet) while other PWN synsets will not be present at all. This makes it difficult, when comparing

automatically constructed WordNets to reference WordNets in a target language, to decide whether newly created synsets are correct or not.

The alternative to automatic evaluations of synsets correctness is of course manual evaluation, which is widely used both in isolation and in conjunction with automated evaluations in the works presented above. It is difficult to know how accurate manually evaluated synsets are without some common guidelines. Some works simply describe having manual annotators decide if an automatically extracted is or is not semantically similar to a reference synsets, while others much more in line with the idea of weighting accuracy according to a degree of correctness have used a Likert scale for conducting manual evaluations

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

2.7 Summary

In this chapter, we discussed an overview of Afaan Oromo language and the major Afaan Oromo word classes. We have also seen that, Afaan Oromo like many other African and Ethiopian languages, has an extremely complex and rich morphology.

We also discussed the WordNet itself and the different components of WordNet. We call the different relationships as components of WordNet. In the chapter, we discussed the different existing wordnet relationships like synonym, hyponym, antonym, and hypernym.

When two senses of two different words (lemmas) are identical, or nearly identical, we say the two senses are synonyms. The word synonym is commonly used to describe a relationship of approximate or rough synonymy. Another way word senses can be related is taxonomically. A word (or sense) is a hyponym of another word or sense if the first is more specific, denoting a subclass of the other. Two senses can be antonyms if they define a binary opposition or are at opposite ends of some scale.

In addition, we also discussed word sense embedding, a fundamental objection to the notion of word embeddings is that many words have more than one meaning, or sense. Sense embedding methods learn embeddings for senses rather than words, according to the ways for learning senses, existing sense embedding methods can be roughly divided into clustering-based, non-parametric and ontology based.

CHAPTER THREE: RELATED WORK

In this chapter, different related works on automatically constructing WordNets are reviewed. They are grouped as methods that use multilingual, methods that use word/sense embeddings and methods that use parallel corpora.

3.1 WordNet Construction Using Multilingual Resources

Lam *et al.* [43] use machine translation and publicly available WordNets in their method. They translate synsets of existing WordNets for different languages to the target language. They take lemmas for all the languages with WordNet and translate them to the target language. Translations are extracted in three ways. First, method is direct translation, where translations are extracted directly. Second, using intermediate WordNet's. Third translating other WordNet to English, that has translation to the target language and then translate that WordNet to the target language. The possible words for synsets in the target language are ranked in terms of a ranking score and highest scoring one is selected. The model is claimed to have obtained an average score of 3.78/5.00 (or informally and possibly incorrectly, 75.60% precision) which is believed it is better than 55.30% obtained by [44] and 43.20% obtained by [45]. In addition, the average coverage percentage of all Wordnet synsets the model creates is 44.85% which is better than 12% in [45].

Sagot and Fiser [46] use multilingual resources together with the Princeton WordNet [5] to build a French WordNet. They applied 2 different approaches, alignment, and translation. In alignment approach, they built a broad multilingual lexicon. All possible synsets of the words in different languages are generated and the synsets that takes place in all the languages is mapped to the French translation. Then translation approach is applied to all the monogenous words. Using bilingual dictionaries monogenous words are translated to French. They claim to have gotten 93.0% precision compared to FREWN.

In another method Taghizadeh and Faili [47] built a WordNet for Persian applying cross lingual WSD. They use only a bilingual dictionary and a monolingual corpus. The WSD problem is modeled as a probabilistic model. They take words from a Persian corpus and translate them to English. Then they extract all possible synsets of these words. Then they use

an expectation maximization approach to get the probability of each word synsets pair and select the synsets above the threshold score. 90% precision and 35% recall.

Patanakul and Charnyote [48] presented a semi-automatic expanding approach is presented to construct a Thai WordNet. Links between Thai words and WordNet synsets are derived from WordNet and its translations. To rank the links, 13 criteria are used. These criteria are categorized into three groups: monosemic, polysemic, and structural criteria. monosemic criteria includes, monosemic one-to-one, monosemic one-to-many, monosemic many-to-one and monosemic many-to-many criterions. polysemous criteria includes, polysemic one-to-one, polysemic one-to-many, polysemic many-to-one and polysemic many-to-many. Structural criteria include, variant, intersection, parent, brother, distance hyponym criteria. They claim to have obtained 80% coverage with 76% accuracy.

De Melo and Weikum [49] use previously known WordNets and dictionaries to build a Universal WordNet (UWN). Their method starts by building an initial graph using existing WordNets and dictionaries. In that graph, words and senses are the nodes and edges are their relations, such as translation or WordNet relations. An SVM classifier is trained to map the words to their synsets. They claim to have attained 92% accuracy.

Bond and Foster [50] use open license WordNets and Wiktionary for building a multilingual WordNet. It starts by taking those WordNets and linking them to the Princeton WordNet [5]. Then it is extended by linking Wiktionary senses to PWN synsets. They claim to have achieved 90% accuracy.

In another method Ercan and Haziyev [51] use Wiktionary and available WordNets to build WordNets for any language automatically. They used one greedy unsupervised method and a supervised method. In the greedy method, they build a graph using the Wiktionary translations and the current WordNet's. Nodes are the words and edges are the translations. At first clusters are formed from nodes that has synsets available for them. Then looking at their similarities to these clusters new nodes are added to the clusters and they are iteratively expanded to new words. In the supervised method, a binary classification method is trained using the words from available WordNet's. The binary classification task is defined for each word-synsets pair and classifies if the word should be a member of the synsets or not. 10 different features are used, that are taken from the graph relations.

3.2 WordNet Construction Using Sense and Word Embeddings Resources

Tarouti and Kalita [52] used the method of Lam *et al.* [43] to generate synsets. However, that method does not generate semantic links between the synsets. This method tries to add the links and remove some words from the synsets that seems to be irrelevant. To do this, they use word embeddings. The method works in two parts. First, they remove irrelevant words from synsets. To do this, they calculate the cosine similarities between the embeddings of the words in the synsets. If the highest similarity is below the threshold t, then that synset is removed, otherwise they look at all the words and if a word's highest similarity is below that threshold then that word is removed from the synsets. If a synset does not have any word left, then it is also removed. Second, they look at the similarity between the words of two semantically related synsets. If the highest similarity is below the threshold tp , then that relation is removed. They claim to have achieved 84.4% precision.

Biru Abera Bacha [13] used word embedding to construct automatic Afaan Oromo WordNet. They used merged and distributional semantics approaches. These approaches are automatic approach. To achieve this, they start by creating the word embedding model to generate WordNet terms by collecting the terms and their relations from the Afaan Oromo documents. They first preprocess, tokenize, then apply a lexical syntactic pattern, co-occurrence manipulation, and similarity. In addition to this, they do text operation (i.e., lexical analysis, and stop word elimination), extracting synonyms relations by word embedding, and extracting hyponym relations by lexical syntactic patterns with and without combination of word embedding components. They accept document collections and applied text operation on the documents to generate the nearest neighbors of the term. They also accept document collections and applied text operation on the documents to retrieve the hyponym relations exist in texts which can match the created patterns These nearest neighbors have automatically generated WordNet terms for that term. In their study they combine LSP with Word2Vec to improve the performance of the LSP, because the words which have the same hyponym relation are arranged with the same neighbors. Those words are considered the same words by Word2Vec because of their neighbor's words. The Word2Vec and LSP models do not adjust the WordNet relations which are extracted by them. The Word2Vec extracts the synonym relations with the weight of their similarity given to them by the system. The

WordNet doesn't include the weight of similarity of the terms i.e., it needs only the more similar words. So, they remove the given number before getting the similar words to save it in the Afaan Oromo WordNet. Applying word embedding (Word2Vec) algorithms for Afaan Oromo texts has been registered 80.09% accuracy and 85.04% accuracy for CBOW and Skip Gram respectively. Lexicon-syntactic patterns are applied to Afaan Oromo texts without Word2Vec, and it has been registered 66.73%, 72%, and 69.26% for precision, recall and F-Measure respectively and applying Word2Vec on the result of LSP has registered 81.14%, 80.8%, and 81.1% for precision, recall and F-Measure respectively.

Birehane Tesfaye [14] use WordNet to develop a WSD system that identifies a sense of an Afaan Oromo ambiguous word. The system identifies the sense by checking different types of sense relationships between words that will help to identify the sense of a word, in contrast to the structure of conventional WordNet, they used a clue word based model of WordNet. The related words for each sense of a polysemy word are referred to as the clue words. These clue words are used to disambiguate the correct meaning of the polysemy word in the given context using knowledge-based Word Sense Disambiguation (WSD) algorithms. The clue word can be a noun, verb, adjective or adverb which can solve limitation of English WordNet which has limited number of cross pos relation (relation not between single part of speech). The performance of the system is tested using 50 polysemy Afaan Oromo ambiguous words which are selected randomly. The performance of the WSD based on clue word-based WordNet achieved 92%.

Mulat Getaneh [12] uses word embedding to construct automatic Amharic WordNet. they implemented using the word2vec model. they first collected documents containing Amharic text. Then after, they applied the different natural language preprocessing tasks. By using preprocessed text data as input, they implemented the Word2vec model. Word2vec model captures different features of word (consists of a distributional representation of words). Based on those text features, they extracted the different relations of Amharic WordNet. Word2vec returns top-N conceptually related words. Those words are the different relations of the word, but it needs additional implementation to know which type of relationship it is. For instance, hypernym is one of the relation types that exist in WordNet. they extract this relation based on mutual information concept. Using agglomerative hierarchical clustering combined with

mutual information concept, they find the hypernym word for any given word. If a word shares information among different clusters, that word is a common word for those clusters. Since hypernym is a more general word and shares its information for its hypernym, they consider the common word as a hypernym of retrieved clusters. Then based on the cosine value of word vector, they compute and get the hypernym word within the rest hyponym words that exist in the given cluster. Since the representation includes a different feature of the text, they can also extract antonym relation based on word analogy. Word analogy captures different patterns of words, from those patterns' antonym pattern is one. A synonym is also another relation of WordNet among the different relations. For this relation, distance supervision method was used in addition to word embedding (word2vec model), which means it was implemented via both approaches. For that reason, the relationship was extracted based on both approaches, and the result is aggregated together. Both approaches used distributional semantics concept (words occur in similar context have similar meaning). But distance supervision is based on a supervised learning mechanism (use some examples of synonym as a seed word). It captures a feature of synonym from the given example and extracts a new list of synonyms based on extracted features. The experimental result in this study showed a considerable performance on Afaan Oromo specified WordNet relations (Hypernym, Hyponym, Synonym, and Antonym) extraction. They claim to have achieved a recall measure of 74.3% recorded during testing proves the high performance of their model and the evaluation result shows false positive below 50 %.

In another method Khodak *et al*. [53] use a bilingual dictionary and word embeddings. They apply different strategies for calculating similarity scores between words and synsets. First, they use a naive approach and take the average cosine similarity of the target word and all the lemmas of the synsets. Second, they improve this by changing synsets representation. They use relations, definitions and example sentences of the synsets. They calculate a vector from these data, using the sentence embedding formula of Arora *et al*. [54]. Then they apply a word sense induction method to solve problems caused by polysemous words. They produce French and Russian WordNet with F.5-Score 50.3%, F1-Score 59.3%, Precision 46.1%, Recall 100.0 % & F.5-Score, F1-Score 50.2%, Precision 59.6%, 45.9%, Recall 100.0% respectively.

Sand *et al*. [55] uses word embeddings to add synsets and hypernymy relations to existing WordNet. Word embeddings are trained on a large news text data. They identify candidate hypernyms by selecting ancestors of their nearest neighbors on WordNet. Then those hypernyms are scored according to distributional similarity and distance in the WordNet graph.

Yang Xu and Jiawei Liu [56] proposed a novel method to implicitly incorporate morphological information into word embedding. In their models, the morphemes' meanings are added on the input layer of CBOW rather than morphemes themselves. Based on this strategy, morpheme-similar words will not only gather but also have a trend to group near their morphemes' meanings. Based on the different strategies to add the morphemes' meanings, three models named MWE-A, MWE-S and MWE-M were built. To test the performance of their embeddings, they utilized three comparative baselines and tested them on the word similarity and syntactic analogy tasks. The results show that their models outperform the baselines on five-word similarity datasets. On the golden standard Wordsim-353 and RG-65, their models even approximately outperform CBOW for 5% and 7%, respectively. On syntactic analogy task, their models surpass the baselines to a great extent. Compared with CBOW, MWE-A approximately outperforms it for 7%.

3.3 WordNet Construction Using Parallel Corpora Resources

There are also methods that use parallel corpora for WordNet construction. One of them is the work of Oliver and Climent [57] where they use parallel corpora. If no parallel corpora exist for the language pair, exists or they create one using a machine translation system. Their algorithm starts with sense-tagging the English part of the parallel corpora using Freeling and UKB [58]. Then using a POS tagger, they tag the target language part of the parallel corpora. Then, they use an alignment algorithm to align the sense tagged English part to the target language.

Another method using parallel corpora for WordNet development was applied by Saveski and Trajkovski [59]. They have built a Macedonian WordNet, consisting of 17,553 words and 33,276 synsets. As there were no parallel corpora available between Macedonian and English, they used Google's translation tool. Their method should also solve the WSD problem, and it is like other methods using WSD for WordNet development. They start with Princeton

WordNet [5], translate it to Macedonian and apply WSD algorithm to decide which word maps to the synsets. For WSD, they used gloss of the synsets and translated it to Macedonian. Candidate words are compared with the gloss using Google similarity distance score [60]. Ones higher than the threshold are selected.

Lee *et al.* [61] use bilingual dictionary and Princeton WordNet [5] to build a Korean WordNet. They start with sense of a Korean word and select the WordNet synsets of the translations from English as candidate synsets. To correctly map the synsets to the word, they state that one method is not enough and calculate 6 different heuristic scores and then use those scores as features in a Decision tree for WSD.

Mousavi and Faili [62] use a Persian corpus, Persian WordNet (FarsNet), Princeton WordNet [5] and bilingual dictionary. Using a bilingual dictionary, Persian words are translated to English and using the synsets of those words in Princeton WordNet [5], initial links are created. As their corpus is a POS tagged corpus, they were able to remove the links between words and the synsets, when the word does not have the given POS tag. Then for all the words, context vectors are created. These contexts are created using100 words that occur most with the given word. Then using those context vectors 7 different features are calculated. Then those features are used to construct a classification system where FarsNet is used as the training data.

Chen *et al.* [63] proposed building a Chinese-English WordNet. To create a Chinese-English WordNet, they use five linguistic resources: ASBC corpus, Cilin, Semcor, Chinese-English dictionary, and WordNet. First, during construction, they looked up unambiguous words in a Chinese-English dictionary and entered them into WordNet. Although there is no ambiguity in the translation of these words, the corresponding English translation may include irrelevant senses in addition to the correct one. They look for the most similar synsets in each English translation. When they translate a word, if the English translation only corresponds to one synsets, that synsets is the solution. Otherwise, POS information is considered if the English translation is like more than one synsets.

Chinese-English dictionary lookup finds all the English translations. WordNet search collects the synsets candidates for the translations. Some synsets are selected. Here the problems of translation ambiguity and target polysemy must be faced. In other words, not all English

translations cover all the Cilin senses. Because the Chinese-English dictionary does not distinguish Chinese senses and English translations. It shows a problem when translation ambiguity and target polysemy are integrated together. Select a synsets from synsets candidates for each sense of the Chinese word by using the similar way as mapping unambiguous words. Finally, the result is tested in Chinese-English information retrieval. The CLIR model in monolingual information retrieval with smart information retrieval system, TREC topics 301-350 and TREC-6 text collection achieve 69.23% performance. It also gains 10.02% rise in comparison to a model that resolves together target polysemy and translation ambiguity by a Chinese-English dictionary.

3.4 Summary

Several studies have been done to construct WordNet resources. In this chapter, we discussed the different mechanisms of WordNet construction for different languages. They are grouped as methods that use multilingual, methods that use sense and word embedding and methods that use parallel corpora. The first English WordNet was carefully constructed manually over many decades.

The above methods that we discussed need a resource like multilingual lexical resources or monolingual text corpus, bilingual dictionary or effective machine translator, thesaurus etc. The manual construction of WordNet is also time taking and tiresome. The Afaan Oromo language has no effective machine translator and resource listed above as well. Using word2vec word embedding would treat words holistically with no attention to their internal structure, ignoring the fact that, morphologically, meaning is a multi-faceted concept with multiple axes along which two words can be similar. A fundamental objection to the notion of word embeddings is that many words have more than one meaning, or sense. Homonymous words, such as bark, whose meanings are semantically unrelated, can be considered to consist of several, separate entries in the lexicon which happen to share the same spelling and phonological realization. Therefore, from a theoretical point of view, it seems no more reasonable to represent with a single mathematical representation the meanings of the word bark (the sound made by a dog, and the outer layer of a tree) than it would to be to represent the words meow and leaf similarly, and yet this is what word embeddings do. A similar but less severe effect results from the existence of polysemous words, or those which have several

meanings which share some semantic similarity. The polysemous word cell, for instance, can refer to a small room such as a prison cell, or a structural unit of a living organism. The two meanings both refer to enclosed, self-contained units that are part of a larger structure. So, we propose to construct Afaan Oromo WordNet from sense tagged text data by using the advantage of sense embedding, which captures the conceptually similar word for a given word and sense.

CHAPTER FOUR: DESIGN OF AFAAN OROMO WORDNET

4.1 Introduction

In this chapter, we will discuss design part of the proposed automatic Afaan Oromo WordNet construction model using morphological sense embedding.

4.2 System Architecture

Figure 4.1 depicts the architecture of automatic construction of Afaan Oromo WordNet from supervised word-to-sense pairs. The system first accepts text as an input and then preprocesses the text. The texts are preprocessed to make suitable for further processing. The WordNet architecture has two components training and relation extraction. The training component has text preprocessing and sense embedding components. The text preprocessing subcomponent deals with text cleaning, tokenization, and stop-word removal. The sense embedding subcomponent uses morphological sense embedding to generate WordNet for Afaan Oromo in unsupervised manner. The relation extraction component does hypernym/hyponym extraction, antonym extraction, and synonym extraction using similarity measurement. The next section presents the details of each component.

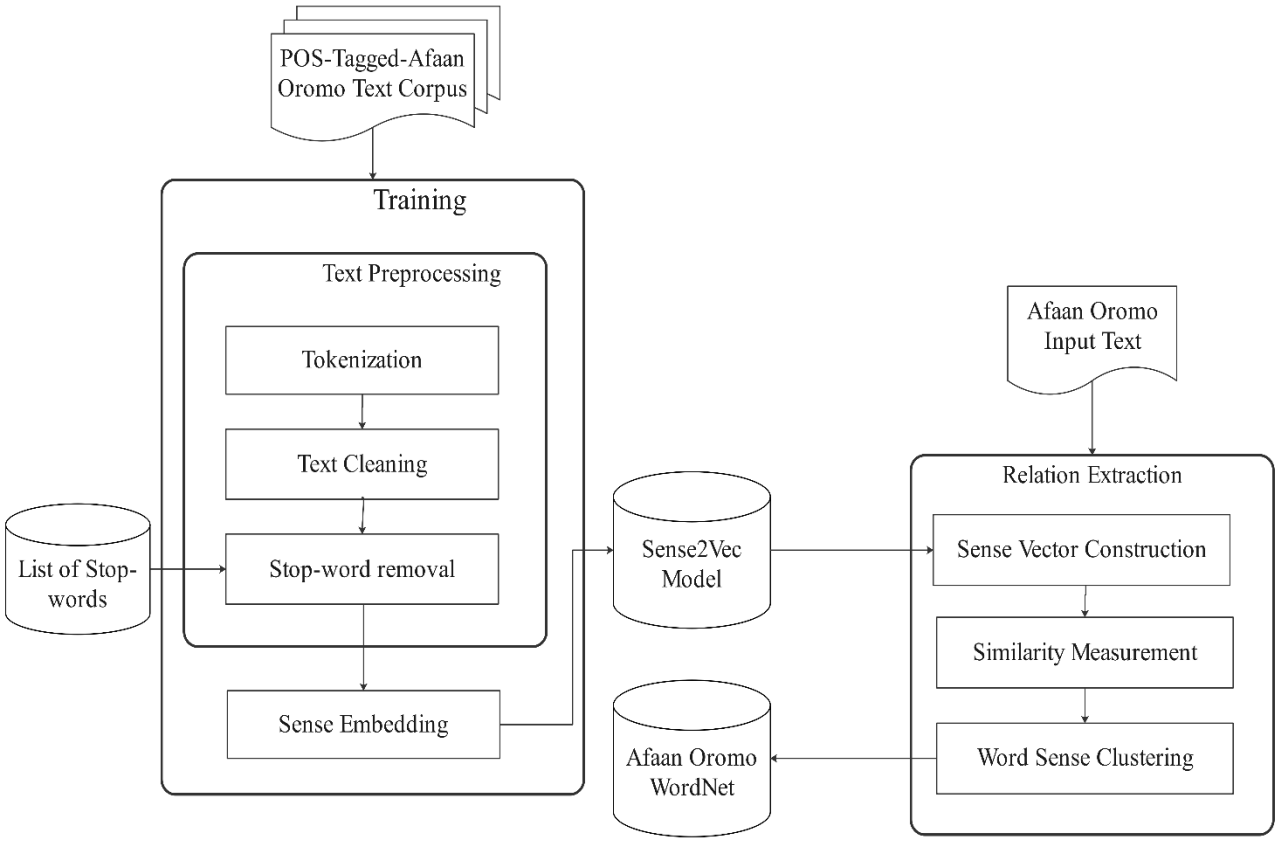


Figure 4.1 Design of System Architecture

4.3 Training

4.3.1 Text Preprocessing

Preprocessing is a significant task in natural language processing and mainly represents the data preparation step. Especially in NLP, this task does have tremendous impact on the success of text analysis and language models. This is mostly caused by the unstructured and arbitrary nature of text data. Furthermore, machines need structure and numerical data. Preprocessing is required to remove raw data that contains undesirable punctuation marks, stop words, numerical values, and special characters, as well as to substitute one alphabet in other

representations, among other things. Stop word removal, word segmentation (tokenization), and non-alphabet character removal are all part of the pre-processing step.

Tokenization

The first step in data pre-processing stage that is used, as input for our WordNet process is tokenization of texts. Tokenization is the process of breaking up texts into different number of levels, paragraphs, sentences, or words. Tokenizing of a given text depends on the characteristics of the language of the text in which it is written. Word demarcation in Afaan Oromo is handled following white space. Thus, Afaan Oromo tokenizer parses text into its constituent words usually by considering the white space and punctuation mark. Punctuation mark usage in Afaan Oromo is like that of English which include semicolon (;), comma (,), full stop (.), question mark (?) and exclamation mark (!). These punctuation marks are removed from the text because they don't have any relevance in building the WordNet but, in Afaan Oromo language apostrophe mark is considered as a part of a word. For example, in the word "sa'a" (cow), the apostrophe is used to show that the vowels are produced independently. Thus, the word "sa'a" must be treated as a single token in the tokenization process. For example, if the original document is "Oromiyaan, qabeenya uumamaan badhaatuu dha" the tokens will be 'Oromiyaan', 'qabeenya', 'uumamaan', 'badhaatuu', 'dha'. Thus, we can define token as an instance of a sequence of characters. In our case, POS tagged text will be broken up into sequence of sentences based on new line and word level by white space. The algorithm used for splitting of part of speech tagged dataset to sentences and then to individual words is presented in Algorithm 4.1.

Algorithm 4.1 Sentence splitting and tokenization

```
Input: POS Tagged Dataset
Output: index dictionary of individual tokens
```

```
Read POS Tagged file F
Define Sentence delimiters: period, question marks and white space
For sentence in F:
        Check sentence boundary
```

```
          If true:
              Put each sentence to separate line and append to
an empty list
For sentence in list:
       Check whitespace between words in a sentence
       Append each word in a sentence to an empty list
```

Text Cleaning

Text cleaning is about removing characters digits and pieces of text that can interfere with text analysis. All texts are noisy to a greater or lesser extent, and even when post-edited will contain varying amounts of typographical errors. A useful assumption when processing such texts is that the errors are isolated and surrounded by clean context that can be used to correct the errors. We define noise in text as any kind of difference between the surface form of a coded representation of the text and the correct text. Text cleaning is one of the most essential text preprocessing steps. It is also highly domain dependent.

Stop-word Removal

Stop word removal is used to remove stop words from the input text. Every language has its own list of stop words: words that have no significant discriminating powers in the meaning of ambiguous words like “yommuu, booddee, isaa, keetii, saniif’’. Stop words mainly consist of prepositions (irra, irraa, itti, and jala), conjunctions (fi, garuu, immoo, yookin, moo, kanaaf, and kanaafu). These words need to be removed during preprocessing phase. There are various techniques used to remove stop words. Among these IDF (inverse document frequency) value and dictionary lookup are the common one. The IDF approach assumes words that appear in many documents as stop words. However, most of the existing stop words removal techniques are based on a dictionary lookup that contains a list of stop words. This technique is much easier for well-studied languages that have standard list of such words. As a result of this, dictionary lookup was employed for this study. For this research work, list of around 430 stop words that is compiled from Afaan Oromo books and research papers were collected and used.

Algorithm 4.2 Text pre-processing

```
Input: Afaan Oromo POS-Tagged text document
Output: clean documents/corpus

Start
Open Afaan Oromo text document from a directory
Read tokenizers input from disk
                  Apply tokenization
                  Go to line 3
                  Check word noise
                     If true:
                          Apply text cleaning
                      Else:
                          Jump
    Read stop-words from disk
           Go to line 3
                 Check words
                     if word in stop-word list:
                          Remove it
                     Else:
                         Jump
    Else:
        End of file
     Write the file into the disk
    End
```

4.3.2 Sense Embedding

After preprocessing tasks of a corpus is done, the next step is to train the mode with sense2vec (machine learning algorithms) using the preprocessed text as input to generate dense vector representation of input word relation.

As we have discussed in Chapter Two, a solution to addressing the meaning conflation deficiency of word embeddings is to represent individual meanings of words, i.e., word senses, as independent representations. Such representations are generally referred to as sense representations. Unlike word2vec model sense2vec model can capture multiple meanings or senses of a word for any word that exists in the training corpus. It can create relatedness between concepts. This can be done by leveraging supervised NLP labels instead of unsupervised clusters to determine a particular word instance's sense. This eliminates the need to train embeddings multiple times, eliminates the need for a clustering step, and creates an efficient method by which a supervised classifier may consume the appropriate word-sense embedding. Figure 4.2 depicts how sense embedding tackles meaning conflation deficiency in Afaan Oromo text.

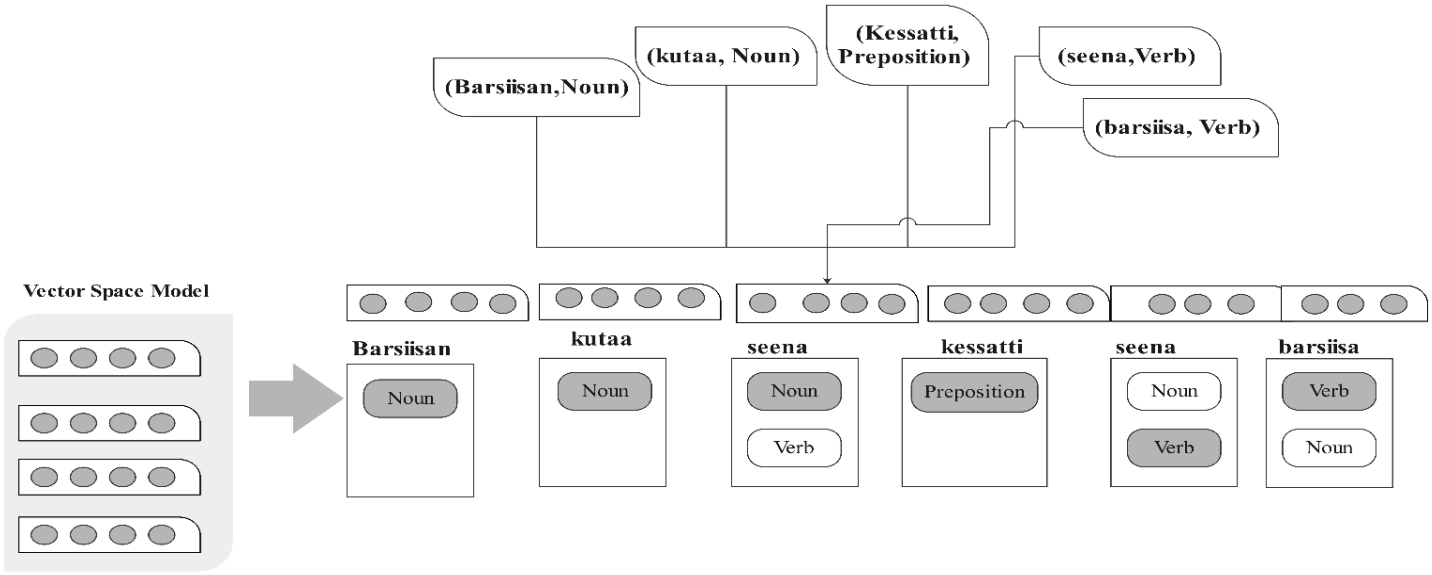


Figure 4.2 A graphical representation of sense2vec model

Given a labeled corpus (either by hand or by a model) with one or more labels per word, the sense2vec model first counts the number of uses (where a unique word map set of one or more labels/uses) of each word and generates a random "sense embedding" for each use. A model is then trained using either the CBOW, Skip-gram, or Structured Skip-gram model configurations. Instead of predicting a token given surrounding tokens, this model predicts a

word sense given surrounding senses. In the training, the input is each word combined with its sense which is its part of speech tag, along with configurable windows (most of the time 5 to 7 words) in addition to other parameters. Most of sense2vec model parameters are used as it is, only some of them need configuration to get better result.

For instance, in Table 4.1, we demonstrate how the training is performed partially. It takes the target word with or without its sense as input and finds its context words. Let us, consider the demonstration with a window size of 2:

Barsiisan|Noun kutaa|Noun seena|Noun kessatti|Preposition seena|verb barsiisa|verb.

The above text after preprocessing becomes: Barsiisan|Noun kutaa|Noun seena|Noun seena|verb barsiisa|verb.

Table 4.1 Sample Target and context words training with window size of 2

Barsiisan|Noun	**kutaa|Noun**	**seena|Noun**	**seena|verb**	**barsiisa|verb**
Barsiisan|Noun	kutaa|Noun	seena|Noun	seena|verb	barsiisa|verb
Barsiisan|Noun	kutaa|Noun	seena|Noun	seena|verb	barsiisa|verb
Barsiisan|Noun	kutaa|Noun	seena|Noun	seena|verb	barsiisa|verb
Barsiisan|Noun	kutaa|Noun	seena|Noun	seena|verb	barsiisa|verb

The sense to vector algorithm creates a dimension with the number of unique words that exist in the text and trains. For instance, for the above example after training the sense vector model the above result in Table 4:1 is obtained. It predicts context words for each target word explicitly. The sample in the table shows how we train the rest of our data. The interpretation is as follow: - for instance, for the target word “seena|Noun” it has the following context words ((“seena|Noun”, “Barsiisan|Noun”), (“seena|Noun”, “kutaa|Noun”), (“seena|Noun”, “barsiisa|verb”) and (“seena|Noun”, “seena|verb”). The red color is the target word whereas the black color in the left and right of the target word is context words.

Algorithm 4.3 Shows training of sense vector Model

```
Input: preprocessed text file
Output: N-dimensional feature vector trained in Sense2vec
model.
Begin
     Read preprocessed text file F
     Call tokenizer Function ()
{
          SenseEmbeddingConfig ()
               setHyperParametrs ()
     startTrain ()
          if Training_Architecture is 0
               model_type 'CBOW'
          else
               model_type 'Skip-Gram'
          Sense2Vec ()
               get Hyper Parameters ()
               Train ()
}
End
Save trained Model
Stop
```

4.4 Relation Extraction

WordNet is made up of several word connections. In this thesis, we will look at several word relationships such as synonym, antonym, hypernym, and hyponym. The mechanisms of WordNet relations extraction are presented in the sub-sections below.

4.4.1 Sense Vector Construction

The Sense2vec algorithm feeds the pre-processed training text corpus as an input and produces the word vectors (embedding) as output. The algorithm first creates a vocabulary from the training text data and then learns vector representations of the words. It is a mapping of words into vectors of floating numbers using the neural network. The vector representation that is formed using the Sense2vec algorithm is considered as a feature. Those are represented in semantic vector space, in the form of floating numbers between 0-1. Each unique word in the

sample corpus being assigned a corresponding vector in the space called dimensions. A vector space includes N-dimensions. In this space, words that share similar contexts in the corpus are placed close to one another. This is based on the cosine similarity of words. The vector representation result is a model. This model is saved as .bin or .txt or .mod etc. and is used as a feature to our task.

4.4.2 Similarity Measurement

The most popular similarity metric in distributional semantic measures is the vector cosine. Compared to Euclidean distances, vector cosine scores are normalized on each dimension and hence are robust to the scaling effect.

4.4.3 Word Sense Clustering

After sense vectors are constructed using the sense2vec embedding algorithm we measure the distance between the target word and words in the word vector space in order to get the different types of word relations, after the discovery of each word relations in the vector space these relations are clustered into facets of a defined dimension, the formed clusters of each word relation are then used as a WordNet.

CHAPTER FIVE: EXPERIMENT AND EVALUATION

5.1 Introduction

In this chapter, we present the experimental aspects of our study in order to validate the proposed automatic Afaan Oromo WordNet construction model. Word vector representation for Afaan Oromo language, the evaluation of the sense embeddings using intrinsic and extrinsic evaluation methods are discussed. While linguistic relationships: word similarities, nearest neighbors, and word analogies are chosen for intrinsic evaluation, binary text classification on Afaan Oromo dataset is tested as an extrinsic evaluation.

5.2 Experimental Setup

5.2.1 Hardware and Software Requirements

In this study, an experiment for the proposed model is performed on DELL Laptop with Intel core i7-10750H CPU, 16 GB RAM and Nvidia GEFORCE 2070 GPU (graphical processing unit) with 3845 cuda cores running on Windows 11 Operating system with Keras, Tensorflow backend support and sense2vec module. To train sense2vec model with sense tagged corpus.

5.2.2 Data collection

We have found a readymade POS tagged Afaan Oromo corpus on the internet that was prepared by Suchomel, *et al* [64]. From the above-mentioned source, we have collected around 117038 POS tagged Afaan Oromo sentences and 300,811 tagged words with nine senses verb, preposition, determinant, conjunction, adverb, pronoun, noun, auxiliary, adjective, see Table 5.1, sample annotated data prepared for training has been attached as Annex A.

5.3 Implementation

In this experiment, we use Streamlit, Anaconda software tools for the purpose of programming. Anaconda is a free open-source distribution of programming languages like Python and R, which is used for predictive analytics, machine learning applications, and data science etc. Anaconda simplifies package management and deployment. From Anaconda supported programming languages and IDEs, we used python and Spider respectively.

We performed our training using sense2vec (a library that is installed in anaconda) open-source library which provides the sense2vec class for working with a sense2vec model.

Natural language Toolkit (NLTK), spacy and pandas are the libraires' we used throughout the experiment.

Streamlit is an open-source python framework for building web apps for machine learning and data science. We can instantly develop web apps and deploy them easily using Streamlit. Streamlit allows us to write an app the same way we write a python code. We use Streamlit to develop the final Afaan Oromo WordNet application.

Table 5.1 POS Tags for Afaan Oromo WordNet construction

Sno	Tag Name	Training set (tagged words)
1	NOUN	114399
2	VERB	21070
3	PRONOUN	37827
4	PREPOSITION	40548
5	DETERMINANT	7169
6	CONJUNCTION	21531
7	ADVERB	17181
8	AUXILIARY	16911
9	ADJECTIVE	24175
		Total No tagged Words = 300,811

5.3.1 Sense embedding

Sense2vec is a twist on the word2vec family of algorithms that lets us learn more interesting word vectors. Before training the model, the text is preprocessed with linguistic annotations, to let us learn vectors for more precise concepts. Part-of-speech tags are particularly helpful: many words have very different senses depending on their part of speech, so it's useful to be able to query for the synonyms of duck|VERB and duck|NOUN separately.

We trained the sense tagged data using one of sense embedding models which is sense2vec model. We first use spacy library to parse the raw text and output binary collection of doc objects, The DocBin class in spacy lets us efficiently serialize the information from a

collection of Doc objects. We can control which information is serialized by passing a list of attribute IDs, and optionally also specify whether the user data is serialized.

Then we load a collection of parsed Doc objects produced in the previous step and output text files in the sense2vec format (one sentence per line and merged phrases with senses).

We used FastText to train vectors, which is another word embedding method that is an extension of the word2vec model. Instead of learning vectors for words directly, fastText represents each word as an n-gram of characters. So, for example, take the word, "artificial" with n=3, the fastText representation of this word is <ar, art, rti, tif, ifi, fic, ici, ial, al>, where the angular brackets indicate the beginning and end of the word.

This helps capture 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 because no internal structure of the word is taken into account. As long as the characters are within this window, the order of the n-grams doesn't matter.

FastText works well with rare words. So, even if a word wasn't seen during training, it can be broken down into n-grams to get its embeddings.

Word2vec and GloVe both fail to provide any vector representation for words that are not in the model dictionary. This is a huge advantage of this method.

The final step is to load the vectors and frequencies and output a sense2vec component that can be loaded via Sense2Vec.from_disk. There are different sense2vec parameters that we specify in our training. We reconfigured some of them and use the rest default parameters. The parameters that are reconfigured are listed below.

- **Vector Size:** It is the number of dimensions of the embedding, which is the length of the dense vector to represent each word. We implemented with 300 dimensions.
- **Window:** The maximum distance between a target word and context word. For our implementation, we set 5 as the window size.

- **Min_count:** It is the minimum count of words to include the word in a training vocabulary. Words with an occurrence of fewer than the frequency specified will be ignored. In our implementation, we set it 2.
- **Training architecture:** The training algorithm, either CBOW (0) or skip-gram (1). We used skip-gram (1).

Finally, the training result is returned in the file type that we want. It may be text file, or. model or .dat. This file consists of the number of unique vocabulary and dimension number (i.e., 300). In addition, there is a vocabulary with its context in real number representation. For instance, a vocabulary "akka|PREP" is represented as follow:

akka|PREP 0.1000922 -0.012499348 -0.081561066 -0.14848009 -0.1668173 -0.08248361 0.0073444587 -0.025839416 0.14838251 0.076605745 -0.031787332 -0.04722698 0.08855839 -0.04103339 -0.071934275 0.12529114 -0.079138294 0.022774953 0.1590681 -0.124558 0.074320264 0.014100378 0.008108465 -0.062185835 0.103301354 -0.090958335 … up to 300 dimension.

We interpret this real number result using sense2vec built in function. To get top-N most similar words for a given query word we use most_similar() function , sense vectors created using sense embedding has been attached as Annex B. Table 5.2 show top-10 most similar words for a given word. The real number value shows the cosine similarity between the query word and context word and frequency of the words throughout the training corpus respectively.

Table 5.2 Contextually related words

seenaa (NOUN)

Synonym	Similarity	Frequency
afaanii NOUN	0.668	60
jechoota NOUN	0.635	71
seena NOUN	0.635	77
aadaa NOUN	0.627	524
eenyummaa NOUN	0.624	156
oromootti NOUN	0.601	111
qubee NOUN	0.590	138
oromootiif NOUN	0.588	50
mallattoo NOUN	0.586	66
argama NOUN	0.577	98

dha (DET)

Synonym	Similarity	Frequency
miti	0.578	798
baayyee	0.542	92
hundee	0.498	107
dhaan	0.498	185
ifaa	0.481	66
ilaaluun	0.481	57
isaati	0.470	66
dhugaan	0.468	102
dalagaa	0.467	54
ta'e	0.466	999

afaan (NOUN)

Synonym	Similarity	Frequency
qubee	0.705	138
aadaa	0.632	524
barreeffama	0.632	71
mallattoo	0.598	66
kitaaba	0.586	107
qooqa	0.584	77
afaanii	0.578	60
amaaraa	0.576	156
safuu	0.571	53
seenaa	0.553	658

5.3.2 Wordnet prototype

Based on the designed model, the prototype model is developed to show the performance of the proposed solution model. Thus, we developed a prototype as shown in Figure 5.1 to demonstrate and test the proposed Afaan Oromo WordNet model and to see if it can perform the word relationship identification process based on the designed and trained model.

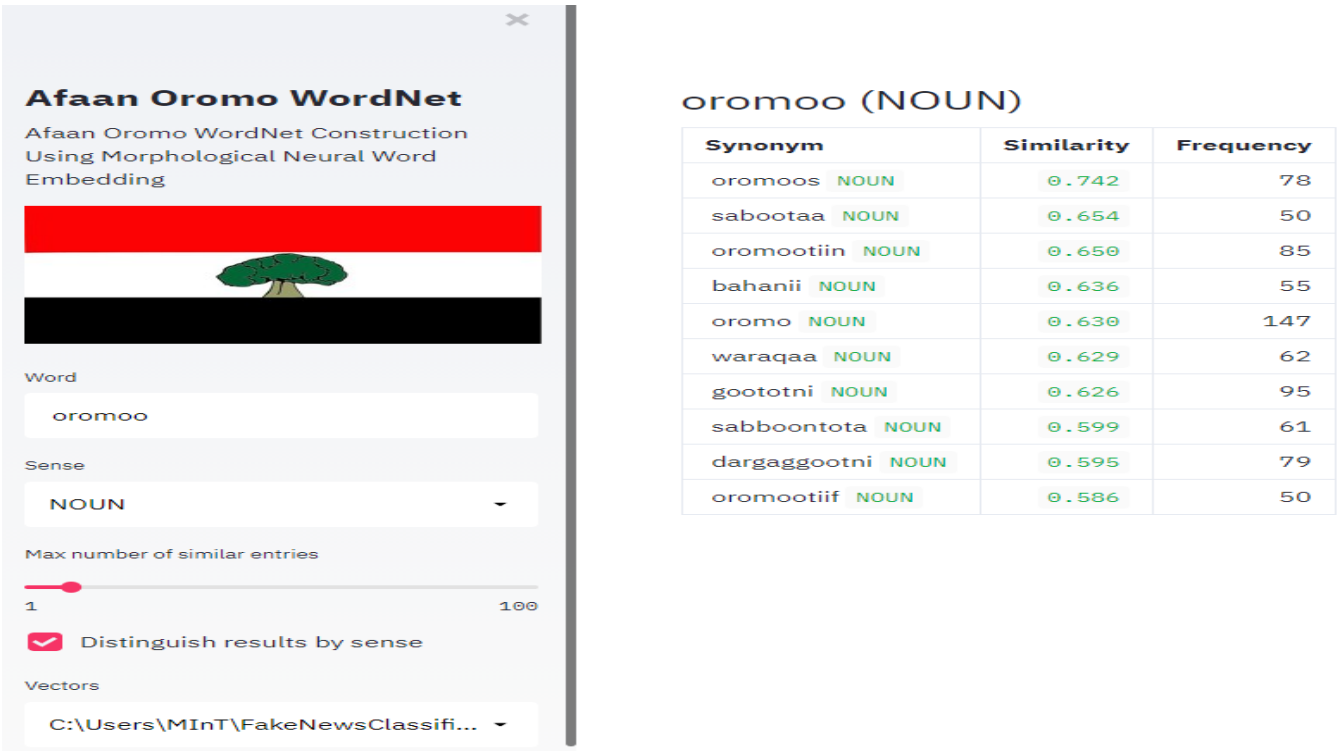


Figure 5.1 User interface of the proposed system prototype

5.4 Test Result

For evaluation of our WordNet, we used two evaluation metrics. These evaluation metrics are intrinsic and extrinsic and are presented in the next section. We explored both evaluation methods using the sense2vec, fastText and TensorFlow library and the tagged corpus prepared.

5.3.1 Intrinsic Evaluation

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

5.3.1.1 Word Similarities and Relatedness

This task involves finding related words with the query word in meaning or syntax. We used skip gram model to find near matches between terms. The similarity between say term1 and term2 is calculated using cosine similarity. Table 5.3 shows how similar two words are as generated from our embedding and their similarity score.

Table 5.3 Similarity scores between term1 and term2

Term1	Term2	Similarity score
Yesus (NOUN)	Kiristos (NOUN)	0.801
Yesus (NOUN)	ilaali (VERB)	0.737
Irraa (ADVERB)	Irra (ADVERB)	0.532

From the above table, let's see the three terms Yesus|NOUN (Jesus), Kiristos|NOUN (Christ) and ilaali|Verb (look) and compute their similarity scores using the cosine distance measure in the vector space.

Sim(Yesus|Noun, Kiristos|NOUN) = cos_dis(vec(Yesus|Noun), vec(Kiristos|NOUN)) == 0.801

Sim(Yesus|NOUN,ilaali|Verb)=cos_dis(vec(Yesus|NOUN),vec(Ilaali|Verb)) == 0.737

Therefore, as the results show the words Yesus|NOUN (Jesus), Kiristos|NOUN (Christ) are semantically closer than Yesus|NOUN (Jesus) and ilaali|Verb (look). The similarity score of a word with itself is a unit.

The various Afaan Oromo WordNet relations extracted from the sense embedding model are evaluated. The evaluation is based on a collection of 50 term pairs, pre-collected human-annotated scores of relatedness between term pairs of various relationships. It computes the cosine similarity of the sense embedding for two-word pairs with human mean semantic similarity scores. It should also have a high correlation (Spearman or Pearson) to be considered well related [65]. To evaluate our system, we first created a questionnaire and a dataset comprised of the relatedness scores of five language experts. The correlation between the human mean semantic relation score provided by the five language experts and the cosine distance calculated by our models was then determined. The questionnaire form we created for language experts to collect relatedness scores to pairs of words, as well as the collected experts' mean score and the cosine distance obtained from the sense embedding models, are attached as Annex C and Annex D, respectively. The cosine similarity of two words from sense embedding should have a strong correlation Spearman with human relatedness scores.

Table 5.4 Spearman's correlation coefficient result

Correlations				
			Sense vector model cosine value	Human mean relatedness score
Spearman's rho	SVM cosine value	Correlation coefficient	1.000	0.740**
		Sig. (2-tailed)		0.000
		N	50	50
	Human mean relatedness score	Correlation coefficient	0.740**	1.000
		Sig. (2-tailed)	0.000	
		N	50	50
** Correlation	is significant at	the 0.01 level	(2-tailed)	

Table 5.6 displays the computation result, which includes a statistical significance value (=0.0) and Spearman's correlation score (Rs=0.74). The calculated probability value is a measure of how probable or likely the observed correlation is. The value of is between 0 and

1. (zero means 0 percent and 1 means 100 percent). If the value of is close to zero, it indicates that there is a stronger relationship between the human mean sematic relatedness evaluation and the cosine values returned by the system. If the p-value is close to one, it indicates that there is no correlation between the two variables being correlated. As a result of the findings, we rejected the null hypothesis of no correlation and accepted the alternative hypothesis of a moderate positive correlation between word vector cosine value and human mean semantic relatedness score.

The cosine distance between words defines how much related two words are. It describes the similarity level and relatedness between words and how they go and found together in the corpus. Words with their morphological variations and their variances based on the POS tags are put as related words. As Afaan Oromo has more inflected forms, the retrieved results for similarity quest are more of inflected forms of the words in question.

5.3.1.2 Word Analogy Relation

When a large dataset is used in training for sense embeddings to represent words in vectors, the resulting vectors can learn subtle relations between the words and corresponding senses. Word analogy relation is a good way of examining and evaluating the quality and goodness of a sense embedding. This linguistic property of words is manifested in sense embeddings. For example, according to Mikolov *et al*. [11], the correlation "if man is to king, woman is to what?" is an analogy that sense embeddings can solve. The answer is queen which is logically correct. Say, a tuple like "suudan(sudan): kaartum(khartum) :: itoophiyaa(Ethiopia) : Finfinnee(addis ababa)", the embedding model should produce correct results if the nearest vector representation to vector Vector(suudan) – Vector(kaartum) + Vector(Itoophiyaa) is Vector(Finfinnee). In word analogies the task is to find a vector V such that vector(V) is closest to Vector(suudan) – Vector(kaartum) + Vector (Itoophiyaa) according to the cosine distance.

The relationships for the analogy might vary due to various reasons. One of the reasons is the size of the dataset or the corpus. The result of word analogy is heavily influenced by the quality and quantity of the corpus used during the training. A minimal corpus might not give the desired analogy. As the main factor is the distances between vectors representing the words in competition, semantics and logic does not involve in the output. How word analogy in sense

embedding works is based on the experimented truth that two groups of words that have similar relationships should be located similar distances apart in the vector space.

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

Table 5.4 Semantic analogy

Relatedness	Word pair 1	Word pair 2
Capital-country	Paaris (Paris) – faaraansayi(France)	Kaartum(khartoum) – suudan (Sudan)
Country – continent	Itoophiya (Ethiopia) – afriikaa(Africa)	Ispeeni (Spain) – awuroopaa(Europe)
Opposite	Gabbabaa(short)-dheera(tall)	Adii(white) – guracha(black)

Table 5.5 Syntactic analogy

Relatedness	Word pair 1	Word pair 2
Plural suffixes	Hadha(mother) – hadhotta(mothers)	Abbaa(father) – abbotta(fathers)
Passive voice suffixes	Nyaate (he ate) – nyaachisan(they late him ate)	Dhagahe (he listend) – dhagesisaan(they let it be listend)
Pronoun/verb suffixes	Keene (he gave) – Keenan(they gave)	Hate (he stole) – hataan(they stole)

5.3.1.3 observation from intrinsic evaluations

We experimented with the WordNet by tuning fastText and sense2vec parameters to evaluate the embedding obtained from the corpus. Several test words for similarities (nearest neighborhood), analogy and OOV were selected. As Table 5.6 show, the results were more concentrated on inflected forms of Afaan Oromo words. This happens due to the richness of morphology in Afaan Oromo. Afaan Oromo is a morphologically rich language and there is a high rate of morphological production in it. Since there are plenty of inflected forms for a

given word in Afaan Oromo, the possibility of having inflected forms of a single word in a single cluster or contiguously is high.

We noticed that syntactic and morphological relatedness gets improved with shorter window size. Table 5.6 compares the top 5 nearest neighbors of the word biyya|Pronoun (country) both using window size (ws) 2 and 5. Note that the underlined words in the table are words that do not have correlation to the task.

Table 5.6 Top neighbors for a word: biyya|pronoun

Ws	Word: biyya|pronoun (country) Top 5 nearest neighbors in the vector space
2	Biyyi (the country), biyyoota (the counties), ameerika (America), lafaa (land), bulchaa (governer)
5	Harki (the hand), seeraan (by the law), biyyaa (the country), dachee (the soil/land), saboota (nationalities')

As the window size or context size is shorter, the result gets finer. However, a careful investigation of semantic relationship in different parameters vary accordingly. For example, semantic relatedness, in contrast with the analogical variations related to syntax and morphology, gets improved when the context is longer.

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

Table 5.7 Word relatedness with the two models: CBOW and SG

Model parameters	Word: Oromoo (oromo). Top 5 Nearest neighbors
Skip gram	Oromoos (the Oromo's), sabootaa(nationalities), oromootiin(by the oromo), bahanii(went out), oromo(oromo)
CBOW	Waraqaa(revolution), goototni (the hero's), sabboontota(the prideful), dargaggootni(the youth), oromootiif(for oromo)

I. Effect of corpus size on the WordNet construction

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

II. Dimensionality

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

To summarize our observation on intrinsic evaluations, fastText and sense2vec Afaan Oromo sense embedding for WordNet construction perform better on morphological similarities due to the language's richness in morphemes, also due to the fact that fastText utilizes subword embedding and fine-tuning hyper-parameters further improves word similarity results.

5.3.2 Extrinsic Evaluation

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

5.3.2.1 Text classification

We used our preprocessed news dataset for multi class text classification purpose. Our dataset has three labels: Fake, enticing Violence and Normal. Each line of the news file contains a label at its start so that the model can recognize what is a word or what is a label.

Table 5.8 Number of datasets in each group and ratio

LABEL	**70/30**		**80/20**		**90/10**	
	Number of news items in train group	Number of news items in validation group	Number of news items in train group	Number of news items in validation group	Number of news items in train group	Number of news items in validation group
FAKE	1287	520	1460	320	1423	230
VIOLENCE	1100	210	1189	153	1252	100
NORMAL	1000	322	1120	210	1200	122

Before training our classifier, the dataset has been split into train and validation. Several experiments were carried out both in 90/10, 80/20 and 70/30 proportions of the dataset, where the proportion is the ratio of the train and validation sets (For example: 80/20 means 80% for training and 20% for validation/testing data). The train and validation datasets are stored in text format. The number of news items in each group and in each ratio of the split are presented in Table 5.9. We used different hyper-parameters on the dataset to train and evaluate the precision, recall and accuracy in predicting and classifying a new item. The following table shows the overall results of different tests conducted using default parameters but with alternating epoch size.

Table 5.9 Training accuracy and validation accuracy using different epochs.

EPOCH	**70/30**		**80/20**		**90/10**	
	Training accuracy	Validation accuracy	Training accuracy	Validation accuracy	Training accuracy	Validation accuracy
1	0.6708	0.8135	0.6969	0.8222	0.6610	0.8192
5	0.8952	0.8594	0.8935	0.8850	0.8942	0.8661
10	0.9255	0.8697	0.9259	0.8825	0.9264	0.8729

On average, as the tables show, the training and validation accuracy increases as the epoch increases. We see that on average the variables, training accuracy and validation accuracy gain an increment when the number of epochs raises up. Take the 80/20 as example, the training and validation accuracy of our classifier has become 0.9259 and 0.8825 respectively at epoch number 10. An increased value of epoch makes an improved classifier model. That means increasing the number of epochs steadily improved the classifier quality, model log while training the model has been attached as Annex F.

We tested the model with epoch number 10, other hyper-parameters being defaulted, and it took about 5 minutes to train on Intel® Core™ i7-64 core with NVIDIA GPU RTX 2070. This test produced 0.9259 F1-score on 80/20 sample of the dataset. Varying other parameters like window size and dimension made a marginal improvement on the quality of the classifier. Further parameter tuning gives us a significant boost on speed and quality. Overall, the Afaan Oromo sense embeddings can be used in text classification tasks with a better quality than other traditional methods.

CHAPTER SIX: CONCLUSION AND FUTURE WORK

6.1 Conclusion

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

We have seen that WordNet is the backbone of many natural language applications including word sense disambiguation, automatic text categorization, information retrieval, and question answering. Even though there are several resources like electronic dictionary and thesauri, WordNet is the largest and wildly used in natural language applications. Since WordNet, is a valuable resource for different applications, creating this resource is very essential. Even if it is possible to construct the resource manually, it is time-consuming and tiresome because the resource consists of different relations. Not only the complication created by the different relation it consists, but it also needs huge dataset. Constructing the resource manually from this large dataset is mind-boggling. To eliminate such issues, automatic WordNet construction based on the concept of distributional semantics approach is proposed. The distributional semantics approach is the current trend that captures the contextual meaning of words from unstructured text data.

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

The resulting WordNet/embedding was evaluated for quality and speed and their ability to capture meaningful representations using evaluation techniques. Two evaluation techniques were utilized: intrinsic and extrinsic. In the intrinsic evaluation, the question of how well the vectors in the embedding capture linguistic relationships between words. The linguistic

relationships under consideration were word similarity, word analogy and OOV words. In this method, we saw that words that are similar or analogous to each other happen together or closer in the space. Related Afaan Oromo words are found contiguous to each other in the vector space. The analogy relationships we found was quite congruent to the works of other researchers on other languages. The sense embedding has automatically learned the vector representations. It is also shown that words which were not part of the training or were rare or misspellings were entertained in the vector representation. The extrinsic evaluation method mainly focuses on the ability of sense embeddings to contribute to the downstream tasks. For this case, we used multiclass text classification. Experimental results vary as hyper-parameters are tuned in various sizes and amounts. The precision, recall and F1-score measures are also shown fluctuating based on parameters. However, as per the testing done on various ample ratios and parameters, the sense embedding can attain 92.59% F1-score in text classification.

6.2 Contribution of the Study

The major contributions of this study include:

- Designing a novel Afaan Oromo WordNet.
- A model is designed for unsupervised WordNet relations extraction.
- Algorithm which can extract hypernym/hyponym from sense embedding result by taking the advantage of mutual information concept and agglomerative hierarchical clustering concept.

6.3 Future Work

The system designed in this study is Afaan Oromo WordNet Construction using morphological sense embedding. WordNet has several relation types. However, this study includes only four main relations that are commonly used in different natural language processing applications. Since sense embeddings have multiple usages, in the future exploring the ways these embeddings can be used in various tasks is one area to listed in the “what to do in the future” list. So, we recommend further research in this study:

- Using BERT language model for it makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text, so that we can narrow down on polysemous words.

- Expanding WordNet relations by adding additional relations.
- Building and utilizing a big and rich corpus to experiment with Afaan Oromo Sense embeddings.
- Studying character level sense embedding and bilingual sense embeddings with other Semitic languages.

References

[1] J. Lyons, Natural Language and Universal Grammar, New York: Cambridge University Press, 1991, pp. 68 - 70.

[2] I. B. A. TURING, Computing machinery and intelligence-AM Turing, vol. 59, Mind, 1950, p. 433.

[3] wikipedia.org, "Natural Language processing," wikipedia.org, 15 June 2013. [Online]. Available: https://en.wikipedia.org/wiki/Natural_language_processing. [Accessed 2021 08 07].

[4] SAS, "Natural Language Processing," 30 July 2012. [Online]. Available: https://www.sas.com/de_ch/insights/analytics/what-is-naturallanguage-processing-nlp.html.. [Accessed 05 October 2020].

[5] C. Fellbaum, WordNet: An Electronic Lexical Database, Cambridge, MA: MIT Press, 1972.

[6] T. K. T. Mindaye, "The Need for Amharic WordNet," *The 5th International Conference of the Global WordNet Association (GWC-2010),* 2010.

[7] A. M. Khodak, "Automated WordNet Construction Using Word Embeddings," *Workshop on Sense, Concept and Entity Representations and Their Applications,* 2017.

[8] P. C. Sathapornrungkij, "Construction of Thai WordNet lexical database from machine readable dictionaries," *the tenth Machine Translation Summit, Thailand,* pp. 87 - 92, 2005.

[9] Torabi, N. Fatemeh, "Querying Word Embeddings for Similarity and Relatedess," *Conference of the North American Chapter of the Association for Computational Linguistics:Human Language Technologies,* vol. 1.

[10] wikipedia.org, "Oromo Language," wikipedia.org, 6 September 2011. [Online]. Available: https://en.wikipedia.org/wiki/Oromo_language. [Accessed 19 July 2020].

[11] Dean, Tomas Mikolov and Kai Chen and G. Corrado and J., Efficient Estimation of Word Representations in Vector Space, ICLR, 2013.

[12] M. Getaneh, "Amharic Wordnet Construction Using Word Embedding," *Master's Thesis*, 2020.

[13] B. A. Bacha, "Building WordNet for Afaan Oromoo," *master's thesis*, 2020.

[14] B. T. TILAHUN, "Afaan Oromo Word Sense Disambiguation Using WordNet," *Master's Thesis*, 2017.

[15] R. TM, "Modern Afaan Oromo grammar: Qaanqee galma Afaan Oromo.," in *Authorhouse*, Bloomington, 2004.

[16] G. Gutema, "Afaan Oromo Text Retrieval System," *Unpublished Masters Thesis, School of Information Science, Addis Ababa University*, 2012.

[17] A. Mideksa, "Statistical Afaan Oromo grammar checker," *Unpublished Masters Thesis, School of Information Science*, 2015.

[18] T. Guya, "CaasLuga, Afaan Oromoo Jildii-1," *Gumii Qormaata Afaan Oromootiin Komishinii "Aadaa fi Turizimii Oromiyaa,* 2003.

[19] Zaborski, Mohammed and Andrzej, "Handbook of the Afan Oromo Language," in *Polish Academy of Sciences*, Warsaw, Poland, 1990.

[20] M. Legesse, "Named Entity Recognition for Afaan Oromo," *Masters Thesis*, 2012.

[21] Meshesha, Getachew Mamo and Million, "Parts of Speech Tagging for Afaan Oromo," in *International Journal of Advanced Computer Science and Applications, Special Issue on Artificial Intelligence*, 2011.

[22] LearnGrammar.Net, "learnGrammar.Net," [Online]. Available: https://www.learngrammar.net/english-grammar/adjective. [Accessed 19 June 2021].

[23] Zaborski, Mohammed and Andrzej, "Handbook of the Afan Oromo Language," *Polish Academy of Sciences,* 1990.

[24] J. Martin, An introduction to natural language processing computational linguistics, and speech recognition, New Jersey, USA: Prentice-Hall Inc, 2007.

[25] W. Olani, "Research: Inflectional Morphology in Oromo [online]," Academia researchs, 10 May 2017. [Online]. Available: https://www.academia.edu/. [Accessed 21 May 2021].

[26] Mikhail Khodak, Andrej Risteski, Christiane Fellbaum and Sanjeev Arora, "Automated WordNet Construction Using Word Embeddings," *Proc. Workshop on Sense, Concepts and Entity Representations and their Applications,* 2017.

[27] S. Maja, "Cognitive synonymy: a general overview," *Facta Universitatis, Linguistics and Literature series,* vol. 2, no. 7, pp. 193-200, 2009.

[28] C. E. Shannon., "Prediction and entropy of printed english," *The Bell System Technical Journal,* vol. 1, no. 30, pp. 50-64, 1951.

[29] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin, A neural probabilistic language model, J. Mach. Learn. Res., 2000.

[30] K. C. G. C. a. J. D. Tomas Mikolov, Efficient Estimation of Word Representations in Vector Space, arXiv e-prints, 2013.

[31] Levy, Omer and Goldberg, Yoav and Dagan, Ido, "Improving Distributional Similarity with Lessons Learned from Word Embeddings," *Transactions of the Association for Computational Linguistics,* vol. 3, pp. 211--225, 2015.

[32] Jeffrey Pennington, Richard Socher, Christopher D. Manning, "GloVe: Global Vectors forWord Representation," *In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP),* pp. 1532-1543, 2014.

[33] Lebret, Rémi, and Ronan Collobert, "Rehabilitation of count-based models for word vector representations," *In International Conference on Intelligent Text Processing and Computational Linguistics,* pp. 417-429, 2015.

[34] Xex´eo, Felipe Almeida Geraldo, "Word Embeddings: A Survey," *Computer and Systems Engineering Program (PESC-COPPE),* 2019.

[35] Jeffrey Pennington, Richard Socher, and Christopher Manning, "Glove: Global vectors for word representation.," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),* pp. 1532-1543, 2014.

[36] N. Chomsky, "chapter Quine's Empirical Assumptions," in *Words and Objections Essays on theWork ofW.V. Quine,* D. Reidel, Dordrecht, 1969.

[37] N. Chomsky, Syntactic structures, Janua linguarum., 1957.

[38] Pilehvar, Jose Camacho-Collados and Mohammad Taher, "From Word to Sense Embeddings: A Survey on Vector Representations of Meaning," *Journal of Artificial Intelligenece Research,* 2018.

[39] Jurafsky, Jiwei Li and Dan, "Do multi-sense embeddings improve natural language understanding," *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* pp. 1722-1732, 2015.

[40] Reisinger, Joseph and Mooney, Raymond J., "Multi-Prototype Vector-Space Models of Word Meaning," *Human Language Technologies: The 2010 Annual Conference of the North {A}merican Chapter of the Association for Computational Linguistics,* pp. 109--117, 2010.

[41] Huang, Eric and Manning, Christopher and Ng, Andrew, "Improving Word Representations via Global Context and Multiple Word Prototypes," *Proceedings of*

the 50th Annual Meeting of the Association for Computational Linguistics, vol. I, pp. 873--882, July 2012.

[42] Neelakantan, Arvind and Shankar, Jeevan and Passos, Alexandre and McCallum, Andrew, "Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing ({EMNLP})*, pp. 1059--1069, October 2014.

[43] Khang Nhut Lam, Feras Al Tarouti, and Jugal Kalita, "Automatically constructing wordnet synsets," *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, vol. 2, pp. 106--111, 2014.

[44] F. B. a. R. Foster, "Linking and extending an open multilingual wordnet," *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1352-1362, August 2013.

[45] Thatsanee Charoenporn, Virach Sornlertlamvanich, Chumpol Mokarat and Hitoshi Isahara, "Semiautomatic compilation of Asian Wordnet," *In Proceedings of the 14th Annual Meeting of the Association for Natural Language Processing*, pp. 1041-1044, 2008.

[46] Fiˇser, Benoˆıt Sagot and Darja, "Building a free french wordnet from multilingual resources," *OntoLex*, 2008.

[47] Faili, Nasrin Taghizadeh and Hesham, "Automatic wordnet development for lowresource languages using cross-lingual wsd," *Journal of Artificial Intelligence Research*, no. 56, pp. 61--87, 2016.

[48] Pluempitiwiriyawej, Patanakul Sathapornrungkij and Charnyote, "Construction of thai wordnet lexical database from machine readable dictionaries," *Proceeding 10th Machine Translation Summit*, 2005.

[49] Weikum, Gerard De Melo and Gerhard, "Towards a universal wordnet by learning from combined evidence," *Proceedings of the 18th ACM conference on Information and knowledge management*, pp. 513--522, 2009.

[50] Foster R. Bond, F, "Linking and extending an open multilingual wordnet," *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pp. 1352--1362, 2013.

[51] Haziyev, Gonenc Ercan and Farid, "Synset expansion on translation graph for automatic wordnet construction," *Information Processing & Management*, vol. I, no. 56, pp. 130--150, 2019.

[52] J. K. e. al., "Enhancing automatic wordnet construction using word embeddings.," *Proceedings of the Workshop on Multilingual and Cross-lingual Methods in NLP*, pp. 30--34, 2016.

[53] Mikhail Khodak, Andrej Risteski, Christiane Fellbaum, and Sanjeev Arora, "Automated wordnet construction using word embeddings," *Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications*, pp. 12--23, 2017.

[54] Sanjeev Arora, Yingyu Liang, and Tengyu Ma, "A simple but tough-to-beat baseline for sentence embedding," 2016.

[55] Heidi Sand, Erik Velldal, and Lilja Øvrelid, "Wordnet extension via word embeddings: Experiments on the norwegian wordnet.," *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pp. 298--302, 2017.

[56] Yang Xu, Jiawei Liu, "Implicitly Incorporating Morphological Information into Word Embedding," 2017.

[57] Climent, Antoni Oliver and Salvador, "Automatic creation of wordnets from parallel corpora," *LREC*, pp. 1112--1116, 2014.

[58] Lluis Padr´o, Samuel Reese, Eneko Agirre, and Aitor Soroa, "Semantic services in feeling 2.1: Wordnet and ukb," *5th GlobalWordNet Conference,* pp. 99--105, 2010.

[59] Trajkovski, Martin Saveski and Igor, "Automatic construction of wordnets by using machine translation and language modeling," *13th Multiconference Information Society,* 2010.

[60] Vitanyi, Rudi L Cilibrasi and Paul MB, "The google similarity distance," *IEEE Transactions on knowledge and data engineering,* vol. III, no. 19, pp. 370--383, 2007.

[61] Bikel, Daniel M, "Automatic wordnet mapping using word sense disambiguation," *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language processing and very large corpora,* 2000.

[62] Faili, Zahra Mousavi and Heshaam, "Persian wordnet construction using supervised learning," *arXiv preprint arXiv,* 2017.

[63] Hsin-Hsi Chen, Chi-Ching Lin and Wen-Cheng Lin, "Construction of a Chinese-English WordNet and Its Application to CLIR".

[64] V. a. R. P. Suchomel, "Oromo web corpus," *LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL),* January 2016.

[65] Schnabel, Tobias and Labutov, Igor and Mimno, David and Joachims, Thorsten, "Evaluation methods for unsupervised word embeddings," *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* pp. 298-307, 2015.

[66] "Oromo Language," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Oromo_language. [Accessed 27 December 2020].

[67] W. Mekonnen, "Development of a Stemming Algorithm for Afaan Oromoo Text," *Master's thesis, Addis Ababa University, School of Information Studies,* 2000.

[68] A. Sani, "Afaan Oromo Named Entity Recognition Using Hybrid Approach," *Unpublished Masters Thesis,* 2015.

[69] W. Temesgen, "Effect of morphological information in Afaan Oromo word sequence prediction," *Unpublished Masters Thesis, School of Information Science, Addis Ababa University, Ethiopia,* 2017.

[70] D. Tesfaye, "A rule-based Afaan Oromo Grammar Checker," in *International Journal of Advanced Computer Science and Applications,* 2011.

[72] M. Getaneh, "automatic wordnet embedding using word embedding," 2020.

[73] X. Rong, "word2vec parameter learning explained," *arXiv preprint arXiv:1411.2738,* 2014.

[74] L. v. d. M. a. G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research,* no. 9, pp. 2579-2605, 2008.

[75] I. Flyamer, 2017.

[76] Kula Kekeba Tune, Vasudeva Varma and Prasad Pingali, "Evaluation of Oromo-English Cross-Language Information Retrieval," in *IJCAI 2007 Workshop on CLIA*, Hyderabad, India, 2007.

[77] Scheible, Silke, Sabine Schulte Im Walde, and Sylvia Springorum, "Uncovering distributional differences between synonym and antonym in a word space model," *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, 2013.

[78] Li, Jiwei and Jurafsky, Dan, "Do Multi-Sense Embeddings Improve Natural Language Understanding?," *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* pp. 1722--1732, September 2015.

[79] Chen, Xinxiong and Liu, Zhiyuan and Sun, Maosong, "A Unified Model for Word Sense Representation and Disambiguation," *Proceedings of the 2014 Conference on*

Empirical Methods in Natural Language Processing ({EMNLP}), pp. 1025--1035, October 2014.

[80] Jauhar, Sujay Kumar and Dyer, Chris and Hovy, Eduard, "Ontologically Grounded Multi-sense Representation Learning for Semantic Vector Space Models," *Proceedings of the 2015 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. I, pp. 683--693, 2015.

[81] Rothe, Sascha andSch"utze, Hinrich, "AutoExtend: Extending Word Embeddings to Embeddings for Synsets and Lexemes," *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 1793--1803, July 2015.

Appendix A: Sample Annotated Dataset

afaan|NOUN fi|CONJ aadaa|NOUN ufii|NOUN guddisuun|NOUN akkasuma|NOUN seenaa|NOUN

kan|PREP fiilmii|NOUN kun|DET seenaa|NOUN dhugaa|ADJ kan|PREP jiruu|NOUN fi|CONJ jireenna|NOUN

akka|PREP ta'e|VERB yaadatti|NOUN qabachuun|NOUN baay'ee|PREP barbaachisaa|NOUN dha|DET

haa|ADV ta'u|VERB iyyuu|NOUN malee|PREP waa'ee|NOUN dhoksaa|NOUN sadan|NOUN

lafa|PREP irra|ADV munyuuqu|NOUN bundumaa|NOUN irrattis|NOUN mootummaa|NOUN haa|ADV

akka|PREP bifa|NOUN keenyaa|ADJ fi|CONJ fakkenya|NOUN keenyaatti|NOUN nama|ADJ in|AUX

bra|NOUN namoota|NOUN in|AUX bittimse|NOUN waaqayyo|NOUN yeroo|ADJ hojjiif|NOUN waamicha|NOUN godhu|NOUN kana|PRON kottaa|NOUN

yasuus|NOUN ittiin|ADJ waamamu|NOUN ta'ee|VERB argama|NOUN faar|NOUN

irraa|ADV adda|ADJ hafuurri|NOUN qulqullunis|NOUN qaamaan|NOUN abbaa|NOUN fi|CONJ

waaqayyo|NOUN jedhamanii|NOUN waamamaniiru|NOUN sadan|NOUN isaanii|PRON iyyuu|NOUN

oromoo|NOUN keessatti|PREP araarri|NOUN bu'ee|NOUN qabsa'onni|NOUN oromoo|NOUN

jechuu|VERB dha|DET ha|NOUN ta'u|NOUN malee|PREP hundee|NOUN rakkoolee|NOUN

fi|CONJ kan|PREP akeekin|NOUN qbo|NOUN abo|NOUN n|PRON geggeeffamuu|NOUN maali|NOUN

Appendix B: Sample Word Vector Feature

ummata|NOUN -0.040394954 0.060424123 -0.124297194 0.13992646 0.045923907 -0.09016912 0.18311858 -0.050156165 -0.027644556 0.042845566 0.11520819 0.1388145 -0.037241194 -0.017839653 0.06950769 -0.027075741 0.11316303 0.13039842 -0.09964588 -0.0447149 -0.1494098 0.15591249 -0.026790645 -0.1 5386707 0.014340768 0.060429327 0.052300066 0.07245018 -0.15022843 0.05934904 -0.013964825 0.10478667 -0.08708937 0.044521775 -0.04082579 -0.15579318 0.1699447 0.18156102 0.0025632682 -0.07604045 0.17325974 0.04922623 -0.010099594 0.21696746

adda|ADJ 0.08313631 0.03347858 0.063889146 0.07826803 -0.20290412 -0.08067429 -0.097378075 0.10112306 -0.21434315 -0.24900676 -0.15522541 -0.17352213 -0.007823046 -0.08303921 -0.17125343 0.049759418 -0.045203183 0.07935451 -0.10224941 -0.091100916 0.012494216 0.3096167 0.1272842 -0.019113917 -0.3932186 0.007839557 0.14241678 0.14656432 -0.23742339 -0.093703926 -0.23095466 0.15055232 -0.03443354 0.018265137

ykn|CONJ -0.115338705 -0.053703226 -0.012020056 0.1201403 0.037173517 -0.17133732 -0.14144774 -0.024048561 -0.050376095 0.2767342 -0.017090881 0.08631202 0.05212917 0.20387286 -0.044445463 0.12442831 0.012133938 -0.21686447 -0.036973987 -0.08400637 -0.13389759 0.19427541 -0.08340016 -0.048438948 -0.07152782 -0.086177394 0.117780924 -0.15002587 -0.14917494 -0.26329005 -0.13755779 0.04941062 -0.028469104 0.07541304 -0.16853677 0.15294996 -0.08829405 -0.0074083605 -0.037964195 -0.07886606 0.13360849 -0.0041536447 0.08466732 0.14354351 0.12379973 -0.012278206 -0.19272783 -0.0625781 0.15352328 0.060355127 -0.049911644 0.096454054 -0.08157802 -0.040911436 0.014626689 -0.3027396 -0.18347986 -0.006451017 0.013167729 0.083903156 0.023982137

0.03321822 0.098862424 0.06474422 0.11803415 -0.033046857 0.02095195 0.16346169 -0.06564032 -0.11720504 -0.012829923

isin|PRON -0.06577052 0.086091176 -0.16554697 -0.06402683 -0.10347663 -0.024241563 0.14683884 0.32275483 -0.15922898 0.18434846 0.24017738 -0.09622125 -0.26798108 -0.32108954 0.13673429 -0.009323732 -0.04127555 -0.045163363 -0.076815635 0.12626038 0.008982815 0.23788455 0.12416666 0.045454703 -0.10290928 -0.11550672 0.022333728 0.12379405

qofa|ADV -0.024454178 0.013341446 0.046886627 -0.054200113 0.1633345 0.080367014 0.08486498 -0.0978314 -0.037963435 0.07553025 0.038667075 -0.07548233 0.10106443 0.15289526 -0.13212843 0.14802115 -0.01925004 -0.30746302 0.04645416 0.18611622 -0.035282407 0.12856628 -0.09234425 0.045959555 0.02576872 -0.0012397602 0.05796778

deebii|VERB -0.004721703 0.037819497 -0.012968581 -0.21805812 -0.31382853 0.23875107 0.025293913 -0.014030851 -0.038412888 -0.27340767 0.10906598 0.38769722 -0.27301338 -0.41330522 0.03783834 -0.027871143 0.3446821 0.003959489 0.03641573 0.16333055 -0.2766714 0.11573193 -0.23584087 -0.40669575 0.002528265

daandii|NOUN -0.13757889 -0.0538916 -0.03323544 0.12704276 -0.054046456 0.125855 -0.206609 0.12928028 -0.13260877 0.037770588 0.044479433 -0.020066606 -0.13411552 -0.06476408 0.042882524 0.11056357 0.010339909 -0.13951923 -0.023590006 0.12219517 -0.10906043 0.15606616 -0.16669491 0.004449558

Appendix C: Questionnaire form given to Language Experts to Estimate Semantic Relatedness of a Word

Semantic Relatedness between Pairs of Words for a Given Relation Type

We would like to ask you to participate in a psycholinguistic experiment in which you will estimate the relatedness score for 50 pairs of Amharic language words for a given relation. Based on the estimation, the word semantic relatedness evaluator measures how well our WordNet model captures human perceived relatedness; it correlates the distance between word vectors and human perceived semantic relatedness.

A list of word pairs is provided below. Please assign a numerical relatedness score between 0 and 10 to each pair (0 = words are completely unrelated, 10 = words are very closely related). The relation type Hyper/hypo refers to the Hypernym/hyponym relationship between two words.

Specific instructions:

- Please fill in your full name at the beginning of the questionnaire.
- Please fill in the scores in the appropriate column of the table.
- Please do not consult anyone while answering.

Full Name: _____

Semantic relatedness score between word term1 and term2 out of 5.

No	Relation Type	Term pair1	Term pair2
1.	Synonym	waggaa	Waggaa
2.	Synonym	waggaa	wagga
3.	Synonym	waggaa	waggaan
4.	Synonym	waggaa	dhibba
5.	Synonym	waggaa	waggaalee
6.	Synonym	Oromoo	oromootti
7.	Synonym	Oromoo	oromoott
8.	Synonym	Oromoo	oromootu
9.	Synonym	Oromoo	oromootif
10.	Synonym	Oromoo	oromoo
11.	Synonym	lafaa	lafaafi
12.	Synonym	lafaa	israa’el
13.	Synonym	lafaa	lafaarraa
14.	Synonym	lafaa	waaqayyoofi
15.	Synonym	lafaa	israa’eloota
16.	Synonym	Afaan	aadaa
17.	Synonym	Afaan	Afaan
18.	Synonym	Afaan	oromootiin
19.	Synonym	Afaan	afaanota
20.	Synonym	Afaan	adaadaa
21.	Hyper/hypo	afriikaa	kibba
22.	Hyper/hypo	afriikaa	itoophiiyaa

23. Hyper/hypo	afriikaa	keenyaa
24. Hyper/hypo	afriikaa	afriikaa kibbaa
25. Hyper/hypo	afriikaa	afriikaa
26. Hyper/hypo	biyya	biyyaa
27. Hyper/hypo	biyya	biyyattii
28. Hyper/hypo	biyya	biyyooti
29. Hyper/hypo	biyya	biyyummaa
30. Hyper/hypo	biyya	itoophiyaa
31. Hyper/hypo	amantaa_kiristaana	amanamummaa
32. Hyper/hypo	amantaa_kiristaana	kiristiyaanumaaa
33. Hyper/hypo	amantaa_kiristaana	kiristaanaa
34. Hyper/hypo	amantaa_kiristaana	hafuura
35. Hyper/hypo	amantaa_kiristaana	amantaa
36. Hyper/hypo	itoophiyaa	oromiyya
37. Hyper/hypo	itoophiyaa	finfinne
38. Hyper/hypo	itoophiyyaa	dimookiraasiin
39. Hyper/hypo	itoophiyaa	amaarraa
40. Hyper/hypo	itoophiyaa	eprdf
41. Antonym	amantaa	fayyadamummaa
42. Antonym	amantaa	siyaasaa
43. Antonym	amantaa	ergamtootaa
44. Antonym	amantaa	dimookraasii
45. Antonym	amantaa	soba
46. Antonym	dhugaa	fili
47. Antonym	dhugaa	rakkina
48. Antonym	dhugaa	qabaachuufi
49. Antonym	dhugaa	soba
50. Antonym	dhugaa	ta’uusaa

Appendix D: Human semantic relation scores out of 10 for the given target word and relation type

No	Similarity type	Term pair 1	Term pair 2	SVM Result	Expert1	Expert2	Expert3	Expert4	Expert5	Human mean semantic relation score
1	Synonym	Waggaa	Waggaa	0.99	8	8	7	8	8	78.254
2	Synonym	Waggaa	Wagga	0.99	8	7	8	8	8	78.254
3	Synonym	Waggaa	Waggaan	0.99	6	7	8	7	7	70.142
4	Synonym	Waggaa	Dhibba	0.98	8	8	8	8	8	80.432
5	Synonym	waggaa	Waggaalee	0.98	7	6	6	6	7	68.325
6	Synonym	Oromoo	oromootti	0.99	8	8	8	8	8	80.432
7	Synonym	Oromoo	oromoott	0.99	8	6	6	7	6	69.215
8	Synonym	Oromoo	oromootu	0.99	7	7	6	6	6	68.325
9	Synonym	Oromoo	oromootif	0.99	7	8	8	6	6	71.235
10	Synonym	Oromoo	oromoo	0.99	8	8	9	8	8	81.235
11	Synonym	Lafaa	lafaafi	0.99	8	7	6	7	7	78.456
12	Synonym	Lafaa	israa’el	0.95	9	8	9	8	8	82.356
13	Synonym	Lafaa	lafaarraa	0.97	7	7	7	7	7	70.541
14	Synonym	Lafaa	waaqayyoofi	0.94	6	6	7	6	7	68.325
15	Synonym	Lafaa	israa’eloota	0.95	9	9	9	8	9	88.247
16	Synonym	Afaan	aadaa	0.99	8	8	9	9	9	87.362
17	Synonym	Afaan	afaan	0.99	8	8	8	8	8	80.432
18	Synonym	Afaan	oromootiin	0.92	8	7	7	8	8	76.235
19	Synonym	Afaan	afaanota	0.99	8	8	7	7	7	74.623
20	Synonym	Afaan	adaadaa	0.98	8	8	7	7	7	74.625
21	Hyper/hypo	Afriikaa	kibba	0.95	8	8	8	7	7	76.123
22	Hyper/hypo	Afriikaa	itoophiiyaa	0.98	9	9	9	9	9	90.385
23	Hyper/hypo	Afriikaa	keenyaa	0.96	9	9	9	9	9	90.385
24	Hyper/hypo	Afriikaa	afriikaa kibbaa	0.98	8	8	8	8	8	80.432
25	Hyper/hypo	Afriikaa	afriikaa	0.99	8	8	8	8	8	80.432
26	Hyper/hypo	Biyya	biyyaa	0.99	8	7	7	8	7	74.213
27	Hyper/hypo	Biyya	biyyattii	0.99	7	7	7	7	7	70.541
28	Hyper/hypo	Biyya	biyyooti	0.99	8	8	8	7	7	76.235
29	Hyper/hypo	Biyya	biyyummaa	0.99	7	7	8	7	7	68.211
30	Hyper/hypo	biyya	itoophiiyaa	0.98	8	9	8	9	9	86.212
31	Hyper/hypo	amantaa_kiristaana	amanamummaa	0.97	7	7	6	7	7	68.223

32	Hyper/hypo	amantaa_kiristaana	kiristiyaanumaaa	0.98	8	8	8	7	8	78.322
33	Hyper/hypo	amantaa_kiristaana	kiristaanaa	0.98	8	8	8	8	8	80.012
34	Hyper/hypo	amantaa_kiristaana	hafuura	0.97	8	8	7	7	6	72.632
35	Hyper/hypo	amantaa_kiristaana	amantaa	0.97	8	8	6	8	7	74.235
36	Hyper/hypo	Itoophiyaa	oromiyya	0.98	8	9	8	8	9	84.012
37	Hyper/hypo	Itoophiyaa	finfinne	0.98	9	8	9	9	9	88.233
38	Hyper/hypo	Itoophiyaa	dimookiraasiin	0.97	7	6	7	7	7	68.232
39	Hyper/hypo	Itoophiyaa	amaarraa	0.98	8	8	8	8	8	80.012
40	Hyper/hypo	itoophiyaa	eprdf	0.95	7	6	6	7	6	64.255
41	Antonym	Amantaa	fayyadamummaa	0.92	6	6	6	6	6	60.221
42	Antonym	Amantaa	siyaasaa	0.93	8	7	5	6	7	66.425
43	Antonym	Amantaa	ergamtootaa	0.92	7	6	8	8	7	72.386
44	Antonym	Amantaa	dimookraasii	0.92	8	8	7	8	8	78.652
45	Antonym	amantaa	dimookraasii	0.92	8	8	7	8	8	78.652
46	Antonym	Dhugaa	soba	0.93	8	8	8	8	8	80.012
47	Antonym	Dhugaa	fili	0.92	7	6	7	6	6	64.332
48	Antonym	Dhugaa	rakkina	0.96	7	6	6	6	6	62.052
49	Antonym	Dhugaa	qabaachuufi	0.92	7	7	7	6	6	66.223
50	Antonym	dhugaa	ta’uusaa	0.91	8	7	7	6	6	68.254

Appendix E: Sample Screenshot of Word Relation of Sense Embeddings

oromoo (NOUN)

Synonym	Similarity	Frequency
oromoos	0.742	78
sabootaa	0.654	50
oromootiin	0.650	85
bahanii	0.636	55
oromo	0.630	147
waraqaa	0.629	62
goototni	0.626	95
sabboontota	0.599	61
dargaggootni	0.595	79
oromootiif	0.586	50
hidhaman	0.586	86
oromoof	0.576	262
baratoota	0.574	76
oromootti	0.572	111
kunneen	0.571	78
lammiilee	0.570	57
addatti	0.569	123
oromooti	0.564	127
marti	0.564	84
uumata	0.563	55

dha (DET)

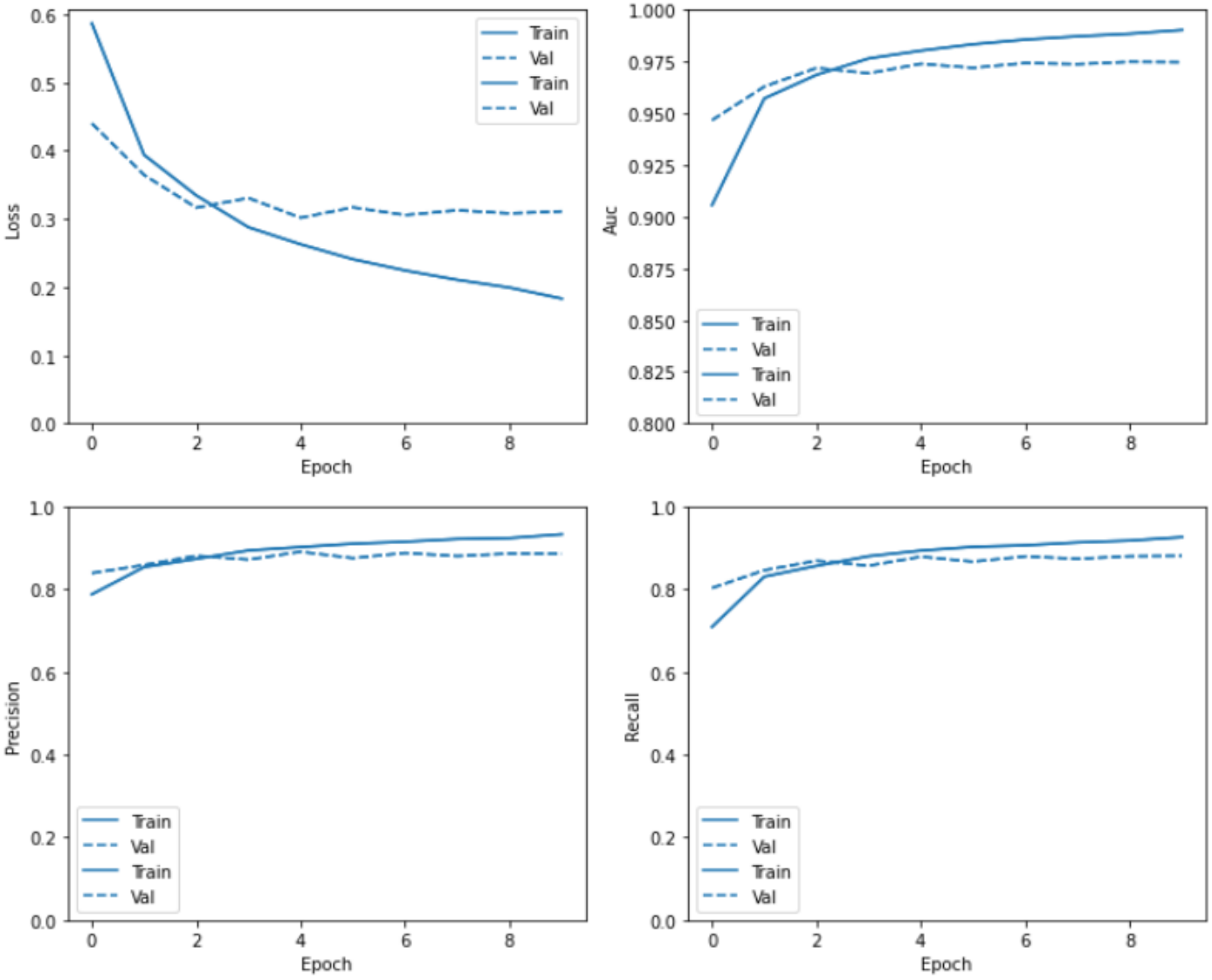
Synonym	Similarity	Frequency
miti	0.578	798
baayyee	0.542	92
hundee	0.498	107
dhaan	0.498	185
ifaa	0.481	66
ilaaluun	0.481	57
isaati	0.470	66
dhugaan	0.468	102
dalagaa	0.467	54
ta'e	0.466	999
taha	0.464	103

Appendix F: Extrinsic Evaluation of Afaan Oromo WordNet Model

Training and evaluation of Afaan Oromo news text classification using pretrained WordNet as an embedding source, history for 10 epochs:

```
Epoch 1/10
111/111 [==============================] - 24s 120ms/step - loss: 0.7406 - tp: 4527.4911 - fp: 1441.2054 - tn: 13017.9375 - fn:
2702.0804 - acc: 0.6610 - precision: 0.7226 - recall: 0.5561 - auc: 0.8385 - val_loss: 0.4471 - val_tp: 2001.0000 - val_fp: 40
3.0000 - val_tn: 4587.0000 - val_fn: 494.0000 - val_acc: 0.8192 - val_precision: 0.8324 - val_recall: 0.8020 - val_auc: 0.9439
Epoch 2/10
111/111 [==============================] - 9s 82ms/step - loss: 0.3945 - tp: 6006.2500 - fp: 1046.4286 - tn: 13412.7143 - fn: 1
223.3214 - acc: 0.8411 - precision: 0.8511 - recall: 0.8297 - auc: 0.9569 - val_loss: 0.3665 - val_tp: 2097.0000 - val_fp: 350.
0000 - val_tn: 4640.0000 - val_fn: 398.0000 - val_acc: 0.8501 - val_precision: 0.8570 - val_recall: 0.8405 - val_auc: 0.9621
Epoch 3/10
111/111 [==============================] - 9s 85ms/step - loss: 0.3123 - tp: 6276.0357 - fp: 842.3125 - tn: 13616.8304 - fn: 95
3.5357 - acc: 0.8776 - precision: 0.8831 - recall: 0.8700 - auc: 0.9722 - val_loss: 0.3537 - val_tp: 2123.0000 - val_fp: 338.00
00 - val_tn: 4652.0000 - val_fn: 372.0000 - val_acc: 0.8577 - val_precision: 0.8627 - val_recall: 0.8509 - val_auc: 0.9653
Epoch 4/10
111/111 [==============================] - 9s 79ms/step - loss: 0.2867 - tp: 6362.0625 - fp: 775.3125 - tn: 13683.8304 - fn: 86
7.5089 - acc: 0.8869 - precision: 0.8914 - recall: 0.8805 - auc: 0.9766 - val_loss: 0.3515 - val_tp: 2119.0000 - val_fp: 345.00
00 - val_tn: 4645.0000 - val_fn: 376.0000 - val_acc: 0.8561 - val_precision: 0.8600 - val_recall: 0.8493 - val_auc: 0.9657
Epoch 5/10
111/111 [==============================] - 9s 84ms/step - loss: 0.2606 - tp: 6436.1607 - fp: 717.3929 - tn: 13741.7500 - fn: 79
3.4107 - acc: 0.8942 - precision: 0.8992 - recall: 0.8896 - auc: 0.9805 - val_loss: 0.3305 - val_tp: 2147.0000 - val_fp: 319.00
00 - val_tn: 4671.0000 - val_fn: 348.0000 - val_acc: 0.8661 - val_precision: 0.8706 - val_recall: 0.8605 - val_auc: 0.9696
Epoch 6/10
111/111 [==============================] - 9s 84ms/step - loss: 0.2231 - tp: 6584.0000 - fp: 603.0982 - tn: 13856.0446 - fn: 64
5.5714 - acc: 0.9163 - precision: 0.9184 - recall: 0.9125 - auc: 0.9855 - val_loss: 0.3289 - val_tp: 2166.0000 - val_fp: 303.00
00 - val_tn: 4687.0000 - val_fn: 329.0000 - val_acc: 0.8717 - val_precision: 0.8773 - val_recall: 0.8681 - val_auc: 0.9702
Epoch 7/10
111/111 [==============================] - 9s 81ms/step - loss: 0.2177 - tp: 6580.1429 - fp: 590.2232 - tn: 13868.9196 - fn: 64
9.4286 - acc: 0.9133 - precision: 0.9166 - recall: 0.9095 - auc: 0.9863 - val_loss: 0.3639 - val_tp: 2149.0000 - val_fp: 324.00
00 - val_tn: 4666.0000 - val_fn: 346.0000 - val_acc: 0.8673 - val_precision: 0.8690 - val_recall: 0.8613 - val_auc: 0.9676
Epoch 8/10
111/111 [==============================] - 9s 83ms/step - loss: 0.2077 - tp: 6619.0179 - fp: 557.1875 - tn: 13901.9554 - fn: 61
0.5536 - acc: 0.9201 - precision: 0.9232 - recall: 0.9158 - auc: 0.9875 - val_loss: 0.3371 - val_tp: 2174.0000 - val_fp: 299.00
00 - val_tn: 4691.0000 - val_fn: 321.0000 - val_acc: 0.8741 - val_precision: 0.8791 - val_recall: 0.8713 - val_auc: 0.9700
Epoch 9/10
111/111 [==============================] - 9s 81ms/step - loss: 0.1983 - tp: 6687.0357 - fp: 499.1875 - tn: 13959.9554 - fn: 54
2.5357 - acc: 0.9273 - precision: 0.9303 - recall: 0.9249 - auc: 0.9883 - val_loss: 0.3412 - val_tp: 2175.0000 - val_fp: 300.00
00 - val_tn: 4690.0000 - val_fn: 320.0000 - val_acc: 0.8749 - val_precision: 0.8788 - val_recall: 0.8717 - val_auc: 0.9700
Epoch 10/10
111/111 [==============================] - 9s 81ms/step - loss: 0.1922 - tp: 6680.3125 - fp: 510.1964 - tn: 13948.9464 - fn: 54
9.2589 - acc: 0.9264 - precision: 0.9285 - recall: 0.9230 - auc: 0.9890 - val_loss: 0.3532 - val_tp: 2172.0000 - val_fp: 308.00
00 - val_tn: 4682.0000 - val_fn: 323.0000 - val_acc: 0.8729 - val_precision: 0.8758 - val_recall: 0.8705 - val_auc: 0.9696
```

Model convergence graph, loss, precision, recall and Area under the curve of the model are visualized.



Precision, recall and f1-score for the three text classification classes.

	precision	recall	f1-score	support
0	0.91	0.87	0.89	1237
1	0.86	0.86	0.86	864
2	0.86	0.93	0.89	671
accuracy			0.88	2772
macro avg	0.88	0.89	0.88	2772
weighted avg	0.88	0.88	0.88	2772

Declaration

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: Henok Desalegn Alemayehu

Signature: _____

Date: _____

Confirmed by advisor:

Name: Yaregal Assabie (PhD)

Signature: _____

Date: _____