https://dx.doi.org/10.22161/ijaers.78.39 ISSN: 2349-6495(P) | 2456-1908(O)

Learning Word and Sub-word Vectors for Amharic (Less Resourced Language)

Abebawu Eshetu[1], Getenesh Teshome[2], Tewodros Abebe[3]

[1,2]Department of Information Technology, Haramaya University, Ethiopia
[3]Department of Computer Science, Wolaita Sodo University, Ethiopia

Abstract**—** *The availability of pre-trained word embedding models (also known as word vectors) empowered many tasks in natural language processing, leading to state-of-the-art performance. A key ingredient to the successful application of these distributed word representations is the existence of large curated corpora to train them and use the pre-trained models in downstream tasks. In this paper, we describe how we trained such quality word representations for one of less-resourced Ethiopian languages, Amharic. We used several offline and online data sources and created 100, 200, and 300-dimension word2vec and FastText word vectors. We also introduced new word analogy dataset to evaluate word vectors for Amharic language. In addition, we created Amharic sentence piece model, which can be used to decode and encode words for subsequent NLP tasks. Using this SentencePiece model, we created Amharic sub-word word2vec embedding with 25, 50, 100, 200, and 300 dimensions trained over our large curated dataset. Finally, we evaluate our pre-trained word vectors on both intrinsic word analogy and extrinsic downstream natural language processing task. The result shows promising performance for both intrinsic and extrinsic evaluations as compared to previously released model.*

Keywords**— Amharic, word vectors, fasttext, word2vec.**

I. INTRODUCTION

The introduction of the paper should explain the nature of the problem, previous work, purpose, and the contribution of the paper. The contents of each section may be provided to understand easily about the paper.

Learning distributed word representations using neural models such as word2vec (Mikolov et al., 2013b) and FastText (Bojanowski et al., 2017) have been widely used in natural language processing and shown promising progress over conventional approaches. The existence of freely available Wikipedia data for most languages supported the ability of learning from large amounts of texts. Publicly available models for resourced languages, which are pre-trained on large amounts of data through unsupervised learning, have become a key component of many neural language understanding models. However, Amharic is not benefited from such an advancement due to lack of large curated data collection over web. While most interesting neural architectures are massively introduced for English, but lack of data and pure word vector barely represented Amharic language for this domain.

The large text collection from Wikipedia and common crawl are commonly used data source to train and learn word vectors (Al-Rfou et al., 2013; Bojanowski et al., 2017) for many languages. Unfortunately, the size of Wikipedia is relatively small and often not enough to learn high quality word vectors with wide coverage for many less-resourced languages including Amharic. While common crawl can be used as an alternative solution to get large text with a broader coverage, its data content is noisier than Wikipedia articles (Bojanowski et al., 2017).

As part of building 157 language word vectors project, Facebook released a pre-trained embedding for Amharic language. The Polyglot project (Al-Rfou et al., 2013) also provides pre-trained embedding trained on Wikipedia. While word vectors with about 3K vocabularies of Amharic language is included with Al-Rfou et al. (2013) and Grave et al. (2018) work, most of the words in the vocabulary are non-Amharic words. This is happened because of the dataset is not well prepared. Moreover, as mentioned above, Wikipedia is not rich to train word vectors for most less-resourced countries. In addition, none of the above authors evaluated Amharic word vectors. Thus, preparation of large and curated text collection from different available online and offline data sources and learning quality word vectors is required for empowering semantic word understanding of Amharic NLP tasks.

https://dx.doi.org/10.22161/ijaers.78.39 *ISSN: 2349-6495(P) | 2456-1908(O)*

We contributed a high-quality word vectors for Amharic language using different word embedding models trained over varieties of online and offline data sources. We used Wikipedia, government websites, news articles, Facebook page posts, blogs, explicitly published Amharic text corpora for different NLP tasks, legal documents, fictions, academic books, and spiritual offline documents as data source to train and learn word vectors of both FastText and word2vec models in 100, 200 and 300 dimensions. We evaluated our word vectors with intrinsic word analogy task and extrinsic neural part of speech tagger task. We have also introduced new word analogy dataset that contain both semantic and syntactic word formations for Amharic language.

While pre-trained word embeddings are publicly available, vectors for sub-word are commonly trained on a per-task basis or used with dummy vector values using one-hot-vectors. Heinzerling B. and Strube M. (2018), released pre-trained sub-word embeddings in 275 languages which is trained by Wikipedia dumps using BPE tokenizer and GloVe (Pennington J. et al., 2014) model. Amharic sub-word GloVe vector is also released as part of their work. However, less quality of Wikipedia dataset and standard evaluation made it difficult to use. We trained sub-word tokenizer over our raw dataset for using open source SentencePiece library (Kudo T. and Richardson J., 2018) that can be also used as sub-word segmenting model for other Amharic NLP downstream tasks. Using our SentencePiece model, we have also released Amharic sub-word word2vec embedding with 25, 50, 100, 200, and 300 dimension trained in our curated quality and large coverage dataset and evaluated in neural Amharic-English downstream task. All Amharic learned word vectors we trained can be accessed from https://github.com/Abe2G/AM-Vectors and the code used to normalize Amharic text, train and evaluate word vector embedding is publicly available at https://abe2g.github.io/.

II. RELATED WORK

Several monolingual pre-trained word vectors have been released for different languages together with open source implementation of word embedding models. English word vectors trained on a part of the Google News dataset (100B tokens) were published with word2vec (Mikolov et al., 2013b). Pennington et al. (2014) released GloVe models trained on Wikipedia, Gigaword and Common Crawl (840B tokens). Al-Rfou et al. (2013), in which word vectors have been trained for 100 languages using Wikipedia data. Grave et al. (2018) released FastText word vectors for 157 languages trained over Wikipedia and Common Crawl.

III. AMERHARIC LANGUAGE

Amharic (አማርኛ,ʾämarəña) is an official working language of Ethiopian government spoken by over 100M people all over the country. The Amharic script is known as Ge'ez or Ethiopic. The writing system is called Fidäl (ፊደል) in Ethiopian Semitic languages. Fidäl means "script", "alphabet", "letter", or "character" or abugida (አቡጊዳ), from the first four symbols. Fidäl is a syllabary writing system where the consonants and vowels co-exist within each graphic symbol (Hudson, G., 2009). Unlike majority of its Semitic scripts, such as Arabic and Hebrew, fidäl is written from left to right. The writing system consists of 33 consonants, each having seven "orders" or shapes depending on the vowel with which a given consonant is combined. These are arranged into seven houses (orders) according to the kind of each vowel that the consonants associate themselves with, i.e., the consonant-vowel (CV) combinations (Mekonnen A. et al., 2018). Consider for instance the following individual symbols:

ባ(bä) ቡ(bu) ቢ(bi) ባ(ba) ቤ(be) ብ(bə) ቦ(bo)

In Amharic, if string begins with a vowel then the vowel is written independently; which means, there are no symbols added to the individual vowels. In such cases as affixation, a vowel may come in contact with a consonant on its left side. The following are used as vowel (aka አናባቢ, ʾänababi) characters.

ኧ(ä) ኡ(u) ኢ(i) ኣ(a) ኤ(e) እ(ə) ኦ(o)

At the time of formation, vowel will no longer be considered independently but together with the immediately preceding consonant forming a new unique symbol. We can better clarify this by taking a case for the declension of a noun for number by adding -ኦች(ʾočə) at the end of the noun (Yimam B., 1997). As can be seen in the example below the combination of ት and ኦ results in ቶ.

ቤት(betə) + -ኦች(ʾočə) = ቤትኦች(betəʾočə)

Written as: ቤቶች (betočə)

ሰው(säwə) + -ኦች(ʾočə) = ሰዉኦች(säwuʾočə)

Written as: ሰዎች (säwočə)

Unlike majority of the languages in the GF library that construct words by linearly concatenating morphemes,

https://dx.doi.org/10.22161/ijaers.78.39 *ISSN: 2349-6495(P) | 2456-1908(O)*

Semitic languages have unique non-concatenative properties in addition to the 27 conventional concatenative modifications (Mekonnen A. et al., 2018).

In Amharic language, the end of a sentence is marked by pair of colon (። አራት ነጥብ, ʾäratə nät'əbə). Whereas exclamatory and interrogative sentence are ended by '!' and '?' punctuation marks respectively. The individual words in a sentence are separated by colon (: ሁለት ነጥብ, hulätə nät'əbə) or sometimes with whitespace character (Yimam, B., 1997). In Amharic there are 9 homophonic characters such as (ሀ, ኀ, ሐ, and ኸ), (ሰ and ሠ), (ጸ and ፀ), (ው and ዉ) and (አ and ዓ), which has the same pronunciation and meaning. Sometimes such characters are used interchangeably in a language and this also makes it challenging.

IV. DATASET PREPARATION

To ensure a high-quality product, diagrams and lettering MUST be either computer-drafted or drawn using India ink.

The quality of the word embeddings depends on frequency of a word, which is related to the corpora size. As the morphological richness of the language increases, the vocabulary size increases and the average frequency of every token decreases. Hence, it is important to collect and prepare a large collection of monolingual corpora for Amharic language to get high quality word embedding.

Table 1: Amharic Monolingual Dataset used to train word vectors.

Domain	#Tokens	#Status
Wikipedia	102,307,356	Clean
Fictions and Academic books (Offline)	55,879,654	Clean
Bible and other spiritual data (Offline)	34,102,452	Clean
Different Amharic Blogs	68,204,904	Noisy
Facebook Posts Scraped	88,254,789	Noisy
Amharic News Article (Walta, FanaBC, EBC, BBC-Amharic, VOA Amharic, Reporter)	122,768,827	Noisy
Other Legal and Official Documents (Offline)	115,948,336	Clean
Walta Information Center (WIC)	210,000	Noisy
JW300	68,204,904	Noisy
Habit-project (amWaC17)	30,692,207	Noisy
Total	686,573,429 tokens	

One of the big challenges for Amharic language is unavailability of curated corpora off-the-shelf. Even if Wikipedia is available for Amharic, the dumps are relatively small in size (compared to the English). However, for Amharic language, the proper data collection technique can get a large text collection from Wikipedia, social medias, newspapers, news portals, government websites, etc. which can be a source of high-quality, diverse monolingual corpora. Thus, crawling these sources was our primary task to create large monolingual corpora for Amharic. To get large coverage of word with domain inclusiveness, we trained our Amharic word vectors using the dataset composed over different offline and online sources. Government websites, news, Facebook page posts, blogs, Wikipedia, and publicly available Amharic text data from HabitProject[1], and OPSUS[2] Amharic machine translation dataset are used as online data source. Whereas legal documents (constitution and other guiding rules), fictions, academic books (grade 1st – 12th Amharic language text book and modules of higher education), spiritual offline documents, government office rules and guiding principles are used as offline data source. We have collected a dataset with more than 686.5 million token collections. The detailed dataset and token coverage is depicted at Table 1 above.

4.1 PREPROCESSING

As the data is collected from different source, we have noticed a number of irregularities between tokens. Thus, we performed series of preprocessing steps at token level to canonize all tokens to a standard format. Using simple regular expression, we tokenized each document into sentences and then each sentence is preprocessed at word level and stored to a file using one line per sentence format to enable memory efficient batch iterator based training.

After tokenization, in order to minimize irregularities in the dataset, we applied different forms of character level normalization to standardize the dataset.

In Amharic, there are several characters that have the same pronunciation and meaning with different structure. For example, the word "Habtamu" (ሀብታሙ) can also be written as ሐብታሙ, ሃብታሙ, ሓብታሙ ኻብታሙ and

[1] https://habit-project.eu/wiki/HabitSystemFinal
[2] http://opus.nlpl.eu/JW300.php

https://dx.doi.org/10.22161/ijaers.78.39 *ISSN: 2349-6495(P) | 2456-1908(O)*

ታብታሞ. In addition, Amharic words ends with a suffix such as ቷል can also be written as ቱዋል or ቱአል. For example, በልቷል can also be written as በልቱዋል or በልቱአል. Thus, all homophonic characters are replaced into their common forms: ሐ and ኀ are replaced with ሀ, ሠ with ሰ, ዐ with አ, and ፀ with ጸ. We normalize any character under such category to common canonical representation to avoid unnecessary representation of single word in different forms.

The other issue that needs attention is normalization of punctuation marks. Different styles of punctuation marks have been used in the data as most of the data sources are noisy. For instance, for double quotation mark two single quotation marks, “,”, ‹‹, ››, ``, '', « or » are used. Thus, normalization of punctuation marks is a nontrivial matter. We normalized all types of double quotes by '', all single quotes by ', question marks (e.g., ？ and ፧) by ?, word separators (e.g., : and ፡) by plain space, full stops (e.g., :: and ።) by ።, exclamation marks (e.g., ! and ！) by !, hyphens (e.g., :-, and ፡—) by ፡-, and commas (e.g., ፣ and ،) by ፣. All other punctuation marks and non-Amharic characters are also removed.

In addition to character and punctuation marks irregularities, we have also identified number notation inconsistencies in Amharic document that are created because of nature of the language. In Amharic numbers can be written using Arabic or Amharic (Geez) numeral notation. For example, 100 in Arabic can also be written as ፻ in Geez. To handle such inconsistency, we have transformed all Arabic notations into Geez notation. In addition, in writing shortened notation of long number is also used in different scripts. For example, One Million Five Hundred Thousand can be written as 1.5 ሚሊዮን (1.5 million). We expand numbers expressed in shortened form to their equivalent long form of representation before transforming the representation into Geez notation.

4.2 Models

We used both word2vec (Mikolov et al., 2013b) and FastText (Bojanowski et al., 2017) models to train our word vectors. In this section, we briefly describe the models that we compare to train our word vectors.

4.3 Word2Vec

Word2Vec is the name given to a class of neural network models with two layer that, given an unlabeled training corpus, produce a vector for each word in the corpus that encodes its semantic information. The architecture of word2vec has two model variants: continuous bag-of-words (CBOW) and SkipGram. (Mikolov et al., 2013). In the CBOW architecture, the model predicts the current word from a window of surrounding context words whereas the skip-gram architecture weighs nearby context words more heavily than more distant context words. In word2vec model, every word W in the dictionary V is mapped to a vector w(x), which is a column (vector) in the matrix W. The CBOW model predicts a word w_t using its context w_{t-n}, …, w_{t-1}, w_{t+1} ,…, w_{t+n}. A vector representation h of this context is obtained by averaging the corresponding word vectors:

$$\sum_{x=-n}^{n} u_{w(x)} \quad (1)$$

, where $u_{w(x)}$ are the word vectors.

Mikolov et al. (2013) also introduced the Skip-gram architecture built on a single hidden layer neural network to learn efficiently a vector representation for each word w of a vocabulary V from a large corpora of size C. Skip-gram iterates over all (target, context) pairs (w_t,w_c) from every window of the corpus and tries to predict w_c knowing w_t. The objective function is therefore to maximize the log likelihood:

$$\sum_{t=1}^{c} \sum_{k=-n}^{n} logp(w_{t+k} \vee w_t) \quad (2)$$

, where n represents the size of the window (composed of n words around the central word w_t.

4.4 FastText

Bojanowski et al. (2017), introduced another unsupervised word representation as an extension to word2vec model in order to take into account internal word structure with character n-grams and improve the representation of rare words. Similar to Mikolov et al. (2013), fasttext also has both cbow and skipgram architecture. But the author recommended using skipgram model than cbow. To this end, they train a Skipgram architecture to predict a word w from C given the central word w_t and all the n-grams G_{w_t} (subwords of 3 up to 6 charcters) of w_t. The objective function becomes:

$$\sum_{t=1}^{c} \sum_{k=-n}^{n} \sum_{w \in G_{w_t}} logp(w_{t+k} \vee w_t) \quad (3)$$

https://dx.doi.org/10.22161/ijaers.78.39 *ISSN: 2349-6495(P) | 2456-1908(O)*

where n is the size of window on which the model includes as context for the central word w_t. Here, along learning one vector per word as of word2vec, fastText also learns one vector per n-gram. FastText is able to extract more semantic relations between words that share common n-gram(s) or morphological variants of a word which can also help to provide good embedding for rare words since we can obtain a vector by summing vectors of its n-grams. Particularly, this subword consideration helps morphologically rich languages such as Amharic, Arabic, Hebrew, etc. to capture vector for word with similar semantic relation in to vocabulary word but differs with its formation.

4.6 Word Embedding Experimental Setups

Genism[3] python is used to train and evaluate analogy for both word2vec and FastText models. First, we used default hyper parameters based on recommendation of the original paper (Mikolov et al., 2013). Then, we also tested our model with different parameter settings to check whether increasing window size, epochs and sampling techniques has effect on learning or not following. We used 5 as minimum count to discard words that appear less than minimum count times with 1e-5 threshold to randomly down-sample highly frequent words. The hyper parameter settings we used in our work are described below:

Increasing window size: this parameter depicts the maximum distance between the current and predicted word within a sentence. We started with 5 window size and checked up to 10 window size. Our experiment has shown that increasing window size has no significant effect and we recommend using the size between 6 to 10 is enough for Amharic to include sentences with large word sequence length.

Increasing n-gram (for FastText) size: other than default 3 to 6 characters, we used 3 to 10 characters to make it more appropriate for morphologically rich Amharic language. The experiment shows that increasing subword size helps to improve the performance on syntactically rich languages. In our experiment, FastText model trained in 3 to 10 character range better performed on syntactic level word analogy.

Word embedding dimension: we used 100, 200, and 300 word embedding size. In our experiment, both word2vec and FastText model with 200 dimensions has comparatively improved performance for both intrinsic and extrinsic task evaluation as compared to 100 and 300 dimensions. The vector with 300 dimensions is the one with less performed model in our extrinsic task evaluation.

Increasing training epochs: We started with default parameter by FastText library trains models, which is 5 epochs and then used more epochs (iterations of training) to train the models. Here, we propose to use in range of 10 to 15 epochs.

Increasing negatives: By default, the FastText library samples 5 negative examples. We tested to different value in range of 5 and 15 and as proposed by (Bojanowski et al., 2017), we also propose to use 10.

One of contribution in this work is usage of Sub-word tokenization to train word2vec models. Word2vec models are criticized for ignoring the internal structures of words as compared to other character n-gram augmenting models such as FastText. Especially for languages with rich morphology such as Amharic, considering word as basic unit maximizes probability of OOV words. Thus, incorporating morphological variants of a word through systematic learning is required. Today, using sub-word units such as character, character n-grams or Byte Pair Encodings (BPE) to address the problem of out-of-vocabulary words in NLP is popular in word representation tasks (Sennrich et al., 2016; Bojanowski et al., 2017). However, global vector of such sub-word units is not further investigated other than using in specific downstream tasks. Following Kudo T. and Richardson J., 2018, we trained word2vec model in SentencePiece tokenized sub-words. To consider character level tasks, we released our embedding with 25, 50, 100, 200 and 300 dimensions.

4.7 Evaluation

We used both intrinsic (word analogy) and extrinsic (Amharic downstream task: Neural POS Tagger) evaluation. For all the evaluation we have used, Amharic embedding's from Al-Rfou et al. (2013) and Grave et al. (2018) as baseline.

4.8 Intrinsic Evaluation

4.8.1 Word Analogy Evaluation

One of commonly used word embedding evaluation approach for intrinsic evaluation is word analogy. This evaluation directly test for syntactic or semantic relationships between words. This approach was popularized by Mikolov et al. (2013a). Analogy datasets is used to evaluate semantic and syntactic ability of the word embedding. One of the contributions of this work is the introduction of word analogy datasets for Amharic language. Following Mikolov et al. (2013a), we prepared a dataset that contain a collection of semantic and syntactic

[3] https://radimrehurek.com/gensim/index.html

https://dx.doi.org/10.22161/ijaers.78.39 *ISSN: 2349-6495(P) | 2456-1908(O)*

relationship of words. The dataset is composed of four words separated with whitespace, of the form of በርሊን(berlin) : ጀርመን(Germany) :: ካይሮ(Cairo) : ግብጽ(Egypt). Total of 311 analogies are prepared by considering different morphological variants of a language and semantic analogies. The dataset contains opposite words, plural-nouns, future tense formations, past tense formations, past-participle formations, present-participle formations, actions-verbs, continent analogy, language analogy, nationality analogy, adjective-to-adverb mapping analogy, family analogy, currency and capital-common-countries classes. In general, given a triplet of words A : B :: C, the goal of word analogy is to find the word D for a given word C such that A : B and C : D share the same relation.

Table 2: Performance of the various word vectors on the word analogy task.

Models	Embedding Dimension	Word Analogy (Average Accuracy)
FastText Grave et al. (2018)	300D	11.02
Word2Vec-CBOW	100D	11.02
	200D	16.04
	300D	20.6
Word2Vec -SkipGram	100D	18.7
	200D	18.9
	300D	19.56
FastText-CBOW	100D	18.5
	200D	22.05
	300D	24.08
FastText-SkipGram	100D	16.9
	200D	17.45
	300D	16.07

The values of word analogy column are average accuracies in percentage. We only reported model trained with our proposed hyper-parameter settings.

According to the results, all models outperformed the baseline Grave et al. (2018). The one with best accuracy is FastText model trained with 300D, which improved +13.06 from the baseline FastText model. In all experiments for both word2vec and FastText models, our CBOW model outperformed skipgram.

4.9 Extrinsic Evaluation

Apart from the analogy task, we have also conducted extrinsic evaluation using neural Amharic part-of-speech tagger. The task is selected based on availability of public dataset. The following sections briefs the model architectures of Amharic neural POS tagger.

4.9.1 Amharic Neural Part of Speech Tagger

We adopted a dataset from (Girma A. et al., 2006) as a training corpus. We grouped the dataset into 20, 10, and 70 splits for testing, validation and training sets respectively. Since our aim is to evaluate the performance of our word vectors in Amharic part of speech tagger task, we have create simple two layer bidirectional LSTM based neural network architecture to encode sentence and predict tags. We trained the same architecture for all word vectors (excluding sentence piece trained model) and compared our word vectors with Amharic word vectors released by Grave et al. (2018).

Table 3: Performance of Sub-word tokenized word2vec model in POS tagger task.

Model	Dimension	POS Tagger (Test Accuracy)
BPEembed (Kudo T. and Richardson J., 2018)	50D	0.902
	100D	0.925
SentencePiece-Word2Vec	50D	0.947
	100D	0.951

To evaluate sub-word embedding's in POS tagging task, due to **SentencePiece** tokenizer we must take special care for the correct alignment of token to tag. We keep tags of original word to each tokenized sub-word units which can then be used to project labels to the tokenized representation. Table 3 above depicts our model performance on neural POS tagging task using sub-word tokenized word2vec model. From over all, extrinsic evaluation, our sub-word embedding outperformed the baseline BPEembed with +3 in 100D and +4 in 50D. This is because our tokenizer plus word embedding models are trained in wide data coverage.

https://dx.doi.org/10.22161/ijaers.78.39 *ISSN: 2349-6495(P) | 2456-1908(O)*

Table 4: Performance of the various word vectors on extrinsic POS tagger task

Models	Dimension	POS Tagger
FastText Grave et al. (2018)	300D	0.91
W2V-CBOW	100D	0.92
	200D	0.944
	300D	0.93
W2V-SkipGram	100D	0.925
	200D	0.931
	300D	0.90
FastText-CBOW	100D	0.944
	200D	0.952
	300D	0.935
FastText-SkipGram	100D	0.89
	200D	0.90
	300D	0.93

The values of word analogy and word detection column are average accuracies in percentage. We only reported model trained with our proposed hyper-parameter settings.

The experiment on word level word2vec and FastText models also performed promising accuracy for morphologically rich Amharic language part of speech tagging. Similar to word analogy, our model also performed interesting improvement in test set when compared to base line FastText model (Grave et al., 2018).

4.10Visualizing Word Embedding

To evaluate how our word vectors cluster syntactically and semantically related words, we visualized few neighbor word using Tensorboard TSNE[4]. We created python function called plot_vector that read our embedding vector binary file and extract vocabulary words from embedding file. Then we created non-trainable TensorFlow model and defined TensorFlow 2D tensor variable that holds our embedding. Finally, we associated metadata with our word embedding. Then we run Tensorboard by referring the log directory of metadata. Because of limited computational resource, we visualized only few word space.

[4] wwww.tensorflow.org/tensorboard

The visualization depicted interesting feature of our word to cluster words to their semantic space. As depicted in Figure 1, Amharic words such ወር(wärə), ቀን(qänə), ሰዓት(säʾatə) to one category using as time measurement as semantic class. We can also see that ሜትር(metərə) and ኪሎ(kilo) are clustered together. Number categories such as ሚሊየን(miliyänə), ቢሊዮን(biliyonə),

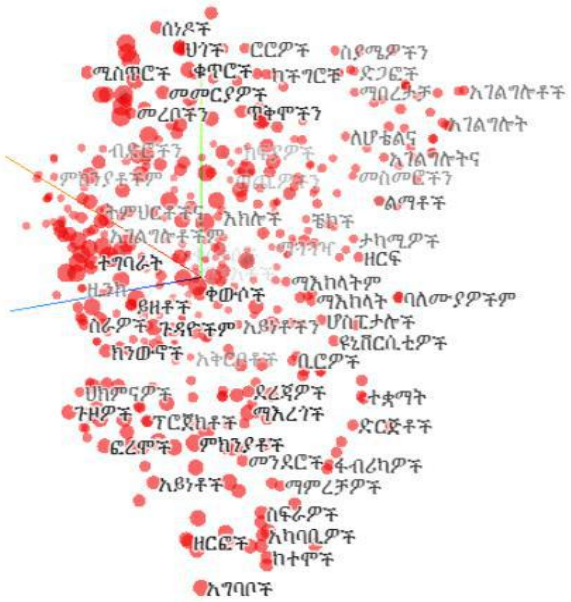

Fig.1: Visualizing word2vec 200D vectors to see how the model cluster semantically related words

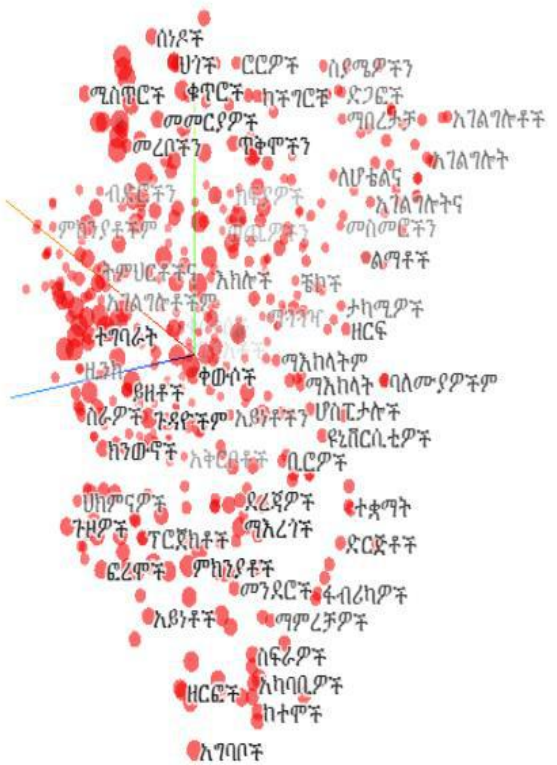

Fig.2: Visualizing FastText 200D vectors

https://dx.doi.org/10.22161/ijaers.78.39 *ISSN: 2349-6495(P) | 2456-1908(O)*

ትሪሊዮን(təriliyonə), and ሺህ(šihə) to one cluster; percentage indicators such as በአማካይ(bäʾämakayə) and በመቶ(bämäto); financial terms such as ክፍያ(kəfəya) ወጪ(wäč'i) and ገቢ(gäbi) at one category.

This show the performance of our word vectors in capturing semantic relationship between words. As depicted in Figure 2 below, our FastText model clustered words with different morphological variations and having the same meaning in to similar vector space.

The model also clustered synonym words to one category. For example, words like ድርጅት(dərəǧətə) and ተቋም(täqwamə), ከተማ(kätäma), ስፍራ(səfəra), መንደር(mänədärə), and አካባቢ(äkababi), መመሪያ(mämäriya), ህግ(həgə), and ሰነድ(sänädə) are grouped in similar space with the same derivational forms in a way that shows the quality of our word vectors to predict word analogy. In addition, visualization in Figure 3 also depicts our word2vec model can also capture word morphology.

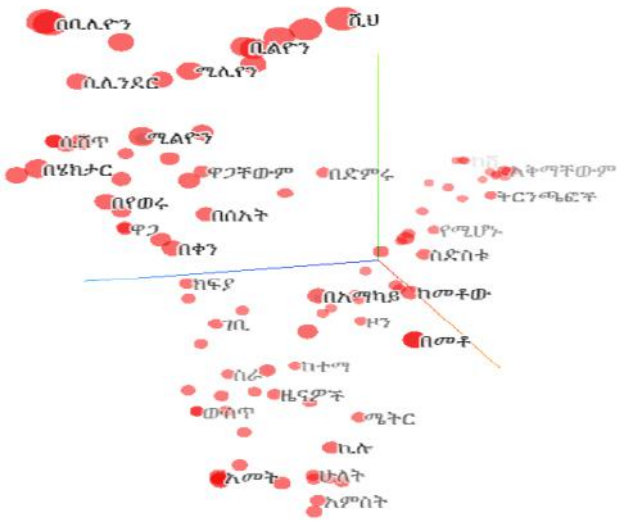

Fig.3: Visualizing word2vec 200D vectors for morphology

V. CONCLUSION

In this work, we contribute word vectors trained on varieties of online and offline sources, as well new analogy datasets to evaluate these models. We study the effect of various hyper parameters on the performance of the trained models, showing how to obtain high quality word vectors. In addition, we also created sub-word embedding using word2vec model. Here we get advantage of word2vec semantic similarity detection ability and capture syntactic features with sub-word embedding.

We also released Amharic SentencePiece tokenizer model that can be used for subsequent NLP tasks. Our evaluation shows that our models can generate quality vectors for Amharic words and proved the vectors are trained over large and curated data with wide coverage from online and offline sources. As future work, we would like to explore more techniques to improve the quality of models for Amharic and other Ethiopic languages including creation of contextualized transformer models.

REFERENCES

[1] Al-Rfou, R., Perozzi, B., and Skiena, S. (2013). Polyglot: Distributed word representations for multilingual nlp. Proc. CoNLL.

[2] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5.

[3] Girma A. Demeke G., Mesfin G.. (2006). Manual annotation of Amharic news items with part-of-speech tags and its challenges. Ethiopian Languages Research Center Working Papers, 2, pages: 1-16.

[4] Mikolov, T., Chen, K., Corrado, G. D., and Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[5] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In Adv. NIPS.

[6] Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. (2017). Advances in pre-training distributed word representations. arXiv preprint arXiv:1712.09405.

[7] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In Proc. EMNLP.

[8] Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany, August. Association for Computational Linguistics

[9] Yimam, B. (1997). *Yamariñña Säwasäw(Amharic Grammar)*. Journal of Ethiopian Studies, 28(2), pages 55-60, Institute of Ethiopian Studies.

[10] Hudson, G. (2009) "Amharic". The World's Major Language, pages. 594–617, Print. Ed. Comrie, Bernard. Oxon and New York: Routledge.

[11] Mekonnen A., Ephrem B., Gasser M., Nürnberger A., (2018) Contemporary Amharic Corpus: Automatically Morpho-Syntactically Tagged Amharic Corpus, Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing, pages 65–70 Santa Fe, New Mexico, USA

[12] Heinzerling B. and Strube M. (2018) BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages, Proceedings of the Eleventh International Conference on

https://dx.doi.org/10.22161/ijaers.78.39 *ISSN: 2349-6495(P) | 2456-1908(O)*

Language Resources and Evaluation (LREC), Miyazaki, Japan

[13] Kudo T. and Richardson J. (2018), SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (System Demonstrations), pages 66–71 Brussels, Belgium, August. Association for Computational Linguistics

[14] Grave1 E., Bojanowski1 P., Gupta P., Armand J. Mikolov T. (2018), Learning Word Vectors for 157 Languages, arXiv:1802.06893

[15] Pennington J., Socher R., and Christopher D. Manning. (2014). GloVe: Global Vectors for Word Representation.