The Development of Pre-processing Tools and Pre-trained Embedding Models for Amharic

Tadesse Destaw Belay
College of Informatics
Wollo University
Kombolcha, Ethiopia
tadesseit@gmail.com

Abinew Ali Ayele
ICT4D Research Center
Bahir Dar University
Bahir dar, Ethiopia
abinewaliayele@gmail.com

Seid Muhie Yimam
Dept. of Informatics
Universität Hamburg
Hamburg, Germany
seid.muhie.yimam@uni-hamburg.de

Abstract

Amharic is the second most spoken Semitic language after Arabic and serves as the official working language of Ethiopia. While Amharic NLP research is getting wider attentions recently, the main bottleneck is that the resources and related tools are not publicly released, which makes it still a low-resource language. Due to this reason, we observe that different researchers try to repeat the same NLP research again and again. In this work, we investigate the existing approach in Amharic NLP and take the first step to publicly release tools, datasets, and models to advance Amharic NLP research. We build Python-based preprocessing tools for Amharic (tokenizer, sentence segmenter, and text cleaner) that can easily be used and integrated for the development of NLP applications. Furthermore, we compiled the first moderately large-scale Amharic text corpus (6.8m sentences) along with the word2Vec, fastText, RoBERTa, and FLAIR embeddings models. Finally, we compile benchmark datasets and build classification models for the named entity recognition task.

1 Introduction

Amharic is the second most widely spoken Semitic language after Arabic (Eshetu et al., 2020). It serves as the official working language of Ethiopia. It is written from left-to-right in Ge'ez alphabets called ፊደል (Fidäl). While there are a lot of attempts at building natural language processing (NLP) components for Amharic, it is a low-resource language without well-developed NLP applications and resources. However, there are some Amharic NLP research, for example Amharic to English or other languages machine translation systems (Melese et al., 2017; Abate et al., 2019), Amharic speech recognition (Abate and Menzel, 2007), Amharic Part-of-speech tagging (PoS) (Gashaw and Shashirekha, 2020; Ibrahim and Assabie, 2014), Amharic named entity recognition (NER) (Woldemariam and Dahlgren, 2020; Gambäck and Sikdar, 2017), and word embeddings (Grave et al., 2018).

While the emergence of NLP research for Amharic is encouraging, there is one chronic problem that endures almost in the last decade in Amharic NLP research, lack of publicly available resources and tools. Researchers start conducting NLP experiments from scratch to collect data, pre-processing, and train models, every time from scratch. We argue that, if Amharic NLP is required to bring meaningful impacts in the already booming artificial intelligent applications development, resources should be released publicly. The online publication of such NLP components has a plethora of advantages, 1) it will lay the groundwork for other researchers, 2) comparing benchmark results will be convenient, 3) adaptation makes the resources grow in size and quality, and 4) both commercial and open source applications can share resources and allow further contributions.

In this work, we mainly target the development of simple Amharic NLP pre-processing components and tools. Furthermore, we also build embedding models which are the main components

in the development of modern NLP applications. The embeddings are particularly required to develop end-to-end deep learning-based NLP applications.

2 Pre-processing Tools

The development of every NLP component starts with data cleaning, language identification, tokenization, segmentation, and normalization in the pipeline (Camacho-Collados and Pilehvar, 2018). This is particularly essential when the NLP component, such as sentiment analysis, is build based on a dataset collected from the social media platforms (Twitter, YouTube, Facebook, and so on). The absence of well-developed tokenization and segmentation tools makes Amharic text pre-processing difficult. Essentially, the tokenizers and segmenters for other languages will easily fail, as the Amharic language has its unique script(Fidäl), syntax, and writing styles.

In the pre-processing of Amharic text, we remove none Amharic tokens from the text. This includes removing URLs, HTML tags and scripts, and different boilerplate content. The Python Compact Language Detection Library (CLD2)[1] library is used to filter Amharic texts from other languages including languages written in Fidäl script, such as Ge'ez and Tigrinya.

Text tokenization in Amharic is challenging, which can not be achieved using the default 'WhiteSpaceTokenizer' available in many frameworks such as NLTK[2] and spaCy[3]. Consider the following phrases, which are retrieved from an online news portal.

Example 2.1 *Amharic tokenization examples*
1): **"አዋጅ ቁጥር 3/2012 አንቀጽ(5)"** *[Proclamation 3/2012 Article (5).*

2): **‹‹ዳግም ትኩረት ለኮቪድ-19›› 2014 ዓ.ም፡ ፡››** *["Attention again for Covid-19" 2021.].*

In the First example, a white space-based splitter will result in 4 tokens while there are 9 Amharic tokens in reality including the punctuation marks. The second example is more complex, where we should consider abbreviations (**ዓ.ም**), quotation marks (**‹‹**), years (1996), and Ethiopic full stop (**፡ ፡**).

Regarding segmentation, in formal Amharic writing, each sentence has to be separated using a unique punctuation mark, namely, the Ethiopic full stop, **አራት ነጥብ**[4] **(።)**. Nowadays, people also concatenate two Ethiopic commas **(፡)** or double Latin commas (:) to terminate Amharic sentences. As far as we know, there is no proper tool to tokenize words and segment sentences in Amharic. In this work, we have build and release the first publicly available Amharic tokenizer and segmenter.

Another highly overlooked pre-processing task is homophone normalization, which aims to normalize certain characters that have different forms in the same word into one form. There are characters in Amharic that currently have the same sound (homophones). The current trend regarding homophone normalization is, in most NLP research, homophone characters are represented into a single representation and in some others are not applied. We build the pre-trained embeddings models using regular (unnormalized) and normalized texts.

3 Pre-trained Embeddings Models

The availability of pre-trained word embedding models facilitated the development of many NLP tasks. In this work, we build word2Vec (Mikolov et al., 2013a; Mikolov et al., 2013b), fastText (Bojanowski et al., 2017), RoBERTa (Liu et al., 2019), and FLAIR (Akbik et al., 2018) pre-trained models to generate embedding representation of words. We have collected moderately large text corpus from social media (Twitter 2m sentences), news (2.5m sentences), and web corpus (2.3m sentences), a total of 106m words from 6.8m sentences. The word2Vec embeddings

[1] https://pypi.org/project/pycld2
[2] http://www.nltk.org/book_1ed/ch03.html
[3] https://spacy.io/
[4] literally mean 'four points'

Models	Regular models			Normalized models		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
word2Vec_CBOW_200D	79.4	67.5	73.0	81.0	68.9	74.5
word2Vec_CBOW_300D	80.7	62.5	70.5	79.6	67.9	73.3
word2Vec_Skip-Gram_200D	83.6	61.3	70.7	82.6	60.1	69.6
word2Vec_Skip-Gram_300D	82.4	62.7	71.2	83.5	61.6	70.9
fastText_CBOW_200D	80.8	63.7	71.3	81.6	63.0	71.1
fastText_CBOW_300D	81.8	66.5	73.4	81.8	67.7	74.1
fastText_Skip-Gram_200D	84.9	59.7	70.1	85.0	64.1	73.1
fastText_Skip-Gram_300D	85.3	64.6	73.5	85.5	65.5	74.1
RoBERTa	70.1	68.4	69.2	72.3	67.9	70.0
FLAIR	81.5	77.4	**79.4**	80.2	75.9	77.8

Table 1: Experimental result for NER classification using different embedding models.

are trained based on the corpus we have collected using the Gensim Python Library (Řehůřek and Sojka, 2011). Both the word2Vec and fastText models are trained using 200 and 300-dimensional vectors, with their CBOW and SkipGram techniques and 10 epochs, while a window size of 5 and 10 are used for word2Vec and fastText respectively. We used a GPU (NVIDIA Quadro RTX 6000 with 24GB RAM) to build both the RoBERTa and FLAIR contextual pre-trained embedding models. FLAIR embedding has been built with parameters: "epochs" of 50, "mini-batch size" of 32, and "block size" of 256. For the RoBERTa embedding, the parameters are: "epochs" of 10, "mini-batch size" of 32, and "block size" of 512.

4 Natural Language Processing Task

In this section, we have discussed the NLP task for Amharic in light of the basic NLP components and the utilization of embedding models we have discussed in the previous section (Section 3). **Named Entity Recognition (NER)**: NER is a process of extracting and classifying a specific predefined list of entities from text. For this experiment, we have used the dataset released by the New Mexico State University Computing Research Laboratory, which is annotated for the SAY project. The data is annotated with 6 classes (PER, LOC, ORG, TIME, TTL, and O-other) and it is available in GitHub[5]. We build sequence tagger-based NER models (BiLSTM with CRF) (Akbik et al., 2018) using our pre-trained static and contextual embedding models. Table 1 shows the performance of different NER models, and we have seen that the model trained with regular contextual FLAIR embeddings performs better than other models.

5 Conclusion and Future Work

In this work, we studied existing text preprocessing approaches and built the first publicly available tools for Amharic. The main components of the preprocessing tool included text cleaning, language identification, word tokenization, and sentence segmentation. Moreover, as more and more NLP applications have relied on deep learning approaches, we have collected around 6.8m sentences and built word2Vec, fastText, RoBERTa, and FLAIR embedding models. The corpus and the embedding models will be used to further develop advanced embedding models. Finally, we compiled benchmark dataset for NER and released baseline models that can be readily used in downstream NLP applications[6]. In the future, we plan to improve our text processing component, for example adding a text normalization component based on expert decisions. We also planned to build other contextualized transformer-based models.

[5] `https://github.com/geezorg/data/tree/master/amharic/tagged/nmsu-say`

[6] Dataset, models, and source codes are available here: `https://github.com/Tadesse-Destaw/Imacts-of-homopnone-normalization`

References

Solomon Teferra Abate and Wolfgang Menzel. 2007. Syllable-based speech recognition for Amharic. In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, pages 33–40, Prague, Czech Republic, June. Association for Computational Linguistics.

Solomon Teferra Abate, Michael Melese, Martha Yifiru Tachbelie, Million Meshesha, Solomon Atinafu, Wondwossen Mulugeta, Yaregal Assabie, Hafte Abera, Biniyam Ephrem, Tewodros Gebreselassie, Wondimagegnhue Tsegaye Tufa, Amanuel Lemma, Tsegaye Andargie, and Seifedin Shifaw. 2019. English-Ethiopian languages statistical machine translation. In *Proceedings of the 2019 Workshop on Widening NLP*, pages 27–30, Florence, Italy, August. Association for Computational Linguistics.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics*, pages 1638–1649.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jose Camacho-Collados and Mohammad Taher Pilehvar. 2018. On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 40–46, Brussels, Belgium, November. Association for Computational Linguistics.

Abebawu Eshetu, Getenesh Teshome, and Tewodros Abebe. 2020. Learning word and sub-word vectors for amharic (less resourced language). *International Journal of Advanced Engineering Research and Science (IJAERS)*, 7:358–366.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Named entity recognition for amharic using deep learning. In *2017 IST-Africa Week Conference (IST-Africa)*, pages 1–8. IEEE.

Ibrahim Gashaw and H L Shashirekha. 2020. Machine learning approaches for amharic parts-of-speech tagging. *arXiv preprint arXiv:2001.03324*.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. *arXiv preprint arXiv:1802.06893*.

Abeba Ibrahim and Yaregal Assabie. 2014. Amharic sentence parsing using base phrase chunking. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 297–306. Springer.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Michael Melese, Laurent Besacier, and Million Meshesha. 2017. Amharic-english speech translation in tourism domain. In *Proceedings of the Workshop on Speech-Centric Natural Language Processing*, pages 59–66.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Yonas Woldemariam and Adam Dahlgren. 2020. Adapting language specific components of cross-media analysis frameworks to less-resourced languages: the case of Amharic. In *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, pages 298–305, Marseille, France, May. European Language Resources association.

Radim Řehůřek and Petr Sojka. 2011. Gensim – Statistical Semantics in Python. In *EuroScipy 2011*, Paris, France.