AMHARIC LIGHT STEMMER

Anonymous authors
Paper under double-blind review

ABSTRACT

Stemming is the process of removing affixes(i.e. prefixes, infixes and suffixes) that improve the accuracy and performance of information retrieval systems.This paper presents the reduction of Amharic words to corresponding stem where with the intention that it preserves semantic information. The proposed approach efficiently removes affixes from an Amharic word. The process of removing such affixes (prefixes, infixes and suffixes) from a word to its base form is called stemming. While many stemmers exist for dominant languages such as English, under resourced languages such as Amharic which lacks such powerful tool support. In this paper, we design a light Amharic stemmer relying on the rules that receives an Amharic word and then it finds a match to the beginning of a word to the possible prefixes and to its ending with the possible suffixes and finally it checks whether it has infix. The final result is the stem if there is any prefix, infix or/and suffix, otherwise it remains in one of the earlier states. The technique does not rely on any additional resource (e.g. dictionary) to verify the generated stem. The performance of the generated stemmer is evaluated using manually annotated Amharic words. The result is compared with current state-of-the-art stemmer for Amharic showing an increase of 7% in stemmer correctness.
Key Words: Amharic light Stemmer, Affixes, Amharic Sentiment Classification

1 INTRODUCTION

Amharic is a highly morphological rich language that adds more challenge for the stemmer performance. The main aim of stemming is to reduce the different morphological (e.g. inflectional or derivational) variations of word forms associated the linguistic information such as number, case, gender, tense, definitive, functional, etc. to its common base form or roots (Jivani et al., 2011). In this work, we follow a simple methodology that depends on the removal of affixes from Amharic words. The idea for designing Amharic stemmer is relying on stripping affixes of Amharic words. The algorithm in the stemmer searches a match of substring pattern expressions which represents affixes from Amharic input word to produce the remaining substring of Amharic word as base form if one of affixes is found. In other words, the rules in the patterns used in the stemmer help to reduce affixes from a word. The patterns representing the affixes of Amharic language are implemented using python library. In general, the existing stemmer algorithms have problems stated as follows: (i) Amharic is a morphologically rich language, that makes development of efficient stemmer very difficult. E.g. A given Amharic verb can have more than 80 different forms. (ii) It is difficult to handle infixes appropriately by a stemmer. (iii) It is challenging to identify and stem Amharic compound words as the language is complex in nature and thus it is not governed by specific rules. (iv) Handling loan words adds additional factors in degrading the performance of stemmer. (v) It is a universal problem to develop stemmer algorithms with minimal errors as it is caused by over-stemming or under- stemming and mis-stemming. Specifically, in this research, we try to answer the following research questions: (i) how do we handle the best possible prefixes lists, suffixes lists and set of conditions to enforce reduplicative words in Amharic texts? (ii) How can we design light stemmer for Amharic texts? (iii) How can we evaluate and enhance the performance of the stemmer? (iv)Does stemming improve performance of Amharic sentiment classification system? The rest of this paper is organized as follows: in the section 2, the related works are presented. The proposed Amharic Stemmer is described in section 3. In section 4, stemmer evaluations are presented. Conclusions and recommendations are drawn in section 5.

2 Related Works

In this section, we briefly present few of related works. The approaches of existing works related to stemmer are categorized into three types of stemmers: truncating (affix removal (Lovins, Porters, Dawsons, etc.), statistical (ngram, HMM, YASS) and mixed (corpus based, Xerox, context sensitive) (Jivani et al., 2011; Ali et al., 2017). Stemmer development was restricted in English. Nowadays, stemmers are adapted and constructed in other languages: Spanish, Urdu, Arabic, Amharic and so on. For stemming Amharic texts, there are few works carried out. The work in (Argaw & Asker, 2007), the authors developed rule based Amharic stemmer using dictionary look-up (machine readable dictionary). This stemmer also relies on corpus statistics to resolve ambiguities of citation forms. The approach in (Argaw & Asker, 2007) contains 65 rules to reduce an Amharic word to citation form for cross-lingual information retrieval applications. The accuracy of this stemmer is 60% and 75%, for old fashion fiction text and news text, respectively. The work in (Fikremariam, 2009) develops corpus based approach using successor variety algorithm with peak and plateau method to stem Amharic texts. This approach with peak and plateau method performs (accuracy of 71.8%) better than successor variety algorithm with entropy (accuracy of 63.95%) on news corpus with 6270 words. The recent work in (Gasser, 2017) builds a morphological tool for three languages (Amharic, Oromo and Tigreygna). The tool is both morphological analyzer and morphological generator relying on Finite State Transducer (FST) where its processes includes rules from surface level(alteration rules: phonological & orthographic) to lexical level(morphotactics). For Amharic and Tigreygna, the morphological analyzer part of the developed tool(HornMorpho) is evaluated. For 200 Amharic verbs, 200 Amharic nouns and 200 Tigregna verbs, morphological analyzer has accuracies of 99, 95.5 and 96%, respectively. This work tries to address morphological analysis and generation of open source tool for the aforementioned languages. Even though this tool has a lot of features, it has a number of associated constraints and limitations. The major one is lack of coverage of the languages and computationally expensive to return the linguistic features of a given Amharic input word. Specifically, it is slow and takes up more memory space while execution of the program. To reduce this problem, we try to develop efficient Amharic light stemmer for Amharic Sentiment classification. As Amharic is one of Semitic family, nouns and verbs are derived from limited roots. In other words, if surface words are reduced to roots, it loses semantic information of the original text. Thus, we design a light stemmer of Amharic language where it keeps semantic information by removing frequent prefixes and postfixes of the input word. Our approach is similar to the approach in (Alemayehu & Willett, 2002) in that it is affix removal. The authors in (Alemayehu & Willett, 2002) developed the first Amharic stemmer relying on an iterative approach for hoping to improve performance of Amharic text retrieval system. This stemmer tries to handle prefixes and suffixes and it is evaluated with accuracy of 95% on 1221 words from different domains. However, the stemmers in both (Alemayehu & Willett, 2002) and (Gasser, 2017) are reducing an Amharic surface word to a root word where as our proposed light stemmer reduces the surface word to a stem that preserves semantic information of the original word. This is one of the reasons why we inspired to develop light stemmer rather than root(heavy) stemmer. Our aim is to test how better our light stemmer for sentiment classification while comparing it with the available root stemmer(benchmark) (Gasser, 2017).

3 Proposed Amharic Stemmer Framework

Amharic light stemmer is proposed almost from scratch. This stemmer is built to remove prefix and suffix using manually compiled affix lists and a specific set of conditions is also set to enforce removal of infix from Amharic word. Prefixes and suffixes are removed using longest match in pattern with regular expression. In other words, affixes with largest length is removed first, and smallest affixes removed last. If any of the affixes has no match in string patterns of Amharic word, the stemmer algorithm terminates. Moreover, we set the minimum length of the stem/root to be at least two characters. The proposed framework contains different steps for developing efficient algorithm for Amharic light stemmer that preserves sentiment information. The framework for Amharic stemmer is shown in Figure 1 below.

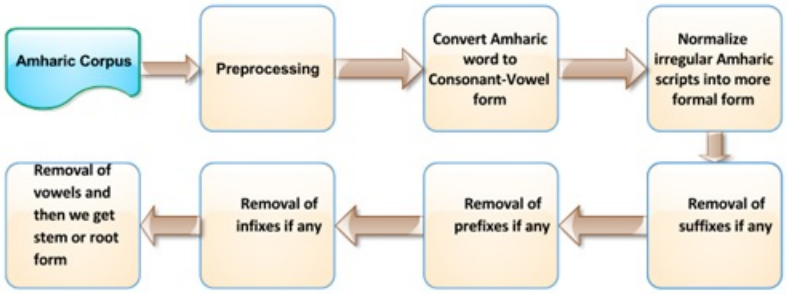

Figure 1: Amharic Light Stemmer Framework

Each of the major elements of the proposed framework are stated as follows:

3.1 Amharic Corpus

For evaluating the generated stems, we use the following corpora and lexicons.
i. Amharic News Corpus: For the Amharic words list in (Foundation, 2017), stems/roots for these words are generated for evaluation of the developed Amharic stemmer.
ii. Amharic Facebook Comments: The data sets for evaluating the performance of the Amharic stemmer for hoping to improve accuracy of Amharic sentiment classification at sentence level.
iii. Amharic Sentiment Lexicons: The Amharic sentiment lexicons includes manual (1000) (Gebremeskel, 2010), Amharic SentiWordNet(SWN)(13679) (Neshir Alemneh et al., 2019) and Amharic Semantic Orientation Calculator(SOCAL) (5683) (Neshir Alemneh et al., 2019). These lexicons are used to compute the sentiment score of the Amharic texts for testing and evaluating the performance of the generated Amharic light stemmer with respect to the state-of-the art(SOTA).

3.2 Preprocessing

We apply basic preprocessing steps on Amharic News Comments. Amharic script symbols are adapted from Geez alphabets. Each alphabet has 7 different syllabic forms (or orders) representing consonant-vowel (CV) pairs (called phonemes). The first order is the base form and the other are derived uniformly from it relying on the vowel sounds (ኧ ኡ ኢ ኣ ኤ እ ኦ) referring to the English vowels (ä, u, i, a, e, o), respectively. Authors in (Yimam, 2000) argued that the sixth alphabets represent the consonant scripts and the remaining orders are reflecting implicitly the corresponding vowel sounds as it is shown in Table 1. There are 33 base alphabets and 7 orders (33 syllables times 7 orders gives 231syllables plus 5 characters with 4 orders of labialized velars and plus 24 additional labialized consonants gives a total of 275 characters(called 'fidels')). As these syllables are directly taken from Geez, there are lists of redundant alphabets (ሀ፡ሐ፡ኀ)፡(ሰ፡ሠ)፡(ጸ፡ፀ)፡(ዐ፡አ) represents the same sound. Those in brackets are redundant base forms. The last pair (ዐ፡አ) represents the Amharic vowels and the remaining 31 represents consonants of Amharic. In this research, first we normalize these redundant symbols into a common symbol. Then, to handle the vowel features of the language explicitly, we require conversion of Amharic scripts to consonant-vowel form by developing mapping lookup table, similar to SERA (Yacob). Particularly, before performing stemming, the algorithm converts each Amharic word to its consonant vowel form.

3.3 Convert Amharic word to Consonant-Vowel form

For normalization purpose, we use the above-mentioned vowels in this research. Table 1 depicts the 7 orders of Amharic alphabets with corresponding CV in amharic and in SERA transliteration forms. The CV format replace and can alternatively used in place of SERA which uses romans and phonetics.

3.4 Handling Irregularities of Amharic Writing dialects into more Formal text form

We develop some conditions to enforce and normalize the variations of style of Amharic writing to formal form. Moreover, irregular nouns, name of places, name of person, irregular verbs, prepositions and irregular adjectives should be used identified to handle

Table 1: Amharic Consonants with 7 orders which are converted consonant vowel form shown in brackets. Left to the bar refers to the consonant vowel in Amharic and right to the bar denotes the consonant vowel transliterated into Romans using SERA.

-	1^{st} Order	2^{nd} Order	3^{rd} Order	4^{th} Order	5^{th} Order	6^{th} Order	7^{th} Order
1	ሀ/ሀኧ/Hä/	ሁ/ሀኡ/Hu/	ሂ/ሀኢ/Hi/	ሃ/ሀአ/Ha/	ሄ/ሀኤ/He/	ህ/ሀ/H/	ሆ/ሀኦ/Ho/
2	ለ/ልኧ /Lä/	ሉ/ልኡ/Lu/	ሊ/ልኢ/Li/	ላ/ልአ /La/	ሌ/ልኤ/Le/	ል/L/	ሎ/ልኦ/Lo/
3	መ/ምኧ/Mä/	ሙ/ምኡ/Mu/	ሚ/ምኢ/Mi/	ማ/ምአ/Ma/	ሜ/ምኤ/Me/	ም/M/	ሞ/ምኦ/Mo/
4	ረ /ርኧ/Rä/	ሩ/ርኡ/Ru/	ሪ/ርኢ/Ri/	ራ/ርአ/Ra/	ሬ/ርኤ/Re/	ር/R/	ሮ/ርኦ/Ro/
5	ሰ/ስኧ/Sä/	ሱ/ስኡ/Su/	ሲ/ስኢ/Si/	ሳ/ስአ/Sa/	ሴ/ስኤ/Se/	ስ/S/	ሶ/ስኦ/So/
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
33	ፈ/ፍኧ/Fä/	ፉ/ፍኡ/Fu/	ፊ/ፍኢ/Fi/	ፋ/ፍአ/Fa/	ፌ/ፍኤ/Fe/	ፍ/F/	ፎ/ፍኦ/Fo/
34	ፐ/ፐኧ/Pä/	ፑ/ፐኡ/Pu/	ፒ/ፐኢ/Pi/	ፓ/ፐአ/Pa/	ፔ/ፐኤ/Pe/	ፕ/P/	ፖ/ፐኦ/Po/

the exceptions to the rules of the regular expressions that represents list of prefixes and suffix rules.

3.5 Removal of Suffixes if any

We tried to identify suffix lists that represent morphological information such as gender, definitive, case, tense, number, part of speech and so on. We identify 97 suffix lists which are converted consonant vowel form as it is shown in appendix.

3.6 Removal of Infixes if any

We understand that if Amharic verb contains pattern CaC- at the middle of the word where C represents Amharic consonant and /a/ represents Amharic vowel. The infix tells us something is done repeatedly performed. E.g. **ፈላለገ**/'ፍኧልኣልኧግኧ'/'FeLaLege'/ means 'search something a number of times'. Its root becomes '**ፍ-ል-ግ**'/flg/. If Amharic noun is repeated and connected by vowel 'a'. This tells us something is big in number. E.g. **ቅጠላቅጠል**/'ቅጥኧልኣቅጥኧል'/'qTeLaqTeL' means 'a number of leaves'. Its root will be **ቅ-ጥ-ል**/qTL/. Thus, to remove such infixes, we enforce conditional rules.

3.7 Removal of Prefixes if any

We also identified 30 prefix lists representing Amharic morphological clues. Sample prefix lists are found in appendix.

3.8 Removal of Vowels and then we get stem or root form

At the end of the stemming process, we transform the Amharic consonant-vowel form to a root form that only contains Amharic consonants. It is shown in the example above section 3.6 (**ፍ-ል-ግ**/flg/, **ቅ-ጥ-ል**/qTl/). Our proposed stemmer is light weight stemmer as it considers affix removal to get the stem or root of Amharic word without considering any additional dictionary. The algorithm matches all possible affixes (suffixes, infixes and/or prefixes). And the process starts from longest match and then shorter match. For example, to stem the word **ቤታቸውን**, the longest match suffix is first searched in the pre-fix list and removed from the word. The longest suffix is -**አችኧውን** in **ቤታቸውን**. However, both -**አችኧውን** and -**ውን** are in the suffix list. Thus, to find the stem of **ቤታቸውን**, the suffix **ውን** is not removed first rather the longest match suffix -**አችኧውን** is removed first. Specifically, for the

proposed Amharic light stemmer, the stated steps are implemented in algorithm 1.

Algorithm 1: Light Stemmer Algorithm of Amharic Text

Input: $word_{orginal}$: input word
Output: $word\_stem$: output stemmed word

```
1  sufix← load suffix lists
2  prefix← load prefix lists
3  word_CV_form←convert_to_CV(word_original)
4  try: if 'ዕንድኦ' in word_CV_form and len(word_CV_form)>2: then
5  |  word_CV_form←word_CV_form.replace('ዕንድኦ','ዕኦ')
6  if 'ጭኦል' in word_CV_form and len(word_CV_form)>2: then
7  |  word_CV_form←word_CV_form.replace('ጭኦል','ጥኦል')
8  if 'እኽስ' in word_CV_form and 'ኽ'==word_CV_form[len(word)-1] and len(word)>2: then
9  |  try: word_CV_form←word_CV_form.replace('እኽስ','ስ')
   |    word_CV_form←word_CV_form[:len(word_CV_form)-1] except: pass
10 if 'ዕ'== word_CV_form[0] and 'ኧ'== word_CV_form[2] and len(word_CV_form)>2: then
11 |  word_CV_form←word_CV_form[1:]
12 if 'ዕኽልኦ' in word_CV_form and len(word_CV_form)>2: then
13 |  word_CV_form←word_CV_form.replace('ዕኽልኦ','ዕኦ') except: pass
14 if len(word_CV_form)>3: then
15 |  word_stem, suffix ← re.findall(suffix, word_CV_Form)[0]
16 |  try: word_stem ← re.findall(prefix, word_CV_Form)[0][1] except: pass
17 if len(word_CV_form)<3: then
18 |  word_stem←word_CV_Form
19 word_stem←remove_vowel(word_stem)
20 Return word_stem
```

Description: The algorithm in 1 takes preprocessed Amharic word which is not in the stop wordlist. In lines 1 and 2, the manually compiled suffix and prefix lists are loaded respectively. In line 3, the input word is converted into consonant -vowel(CV) form. From line 4 to line 13, the input word is normalized if it has change of consonants due to vowels. In line 14, if the input word length is greater than 3, then in line 15, the input word is tested whether it contains the one or more of suffixes in the suffix list. If it contains, the suffix is recursively removed from largest length to smallest length. Following this action, in line 16, the outputed word from the previous step is tested if it contains one or more prefixes from the prefix list. If it contains, the prefix is removed recursively from largest length to smallest length prefix. In lines (17-18), the outputed word from previous step is tested whether its length is not less than 3. Finally, in line 19, the output word is converted to consonant form by removing vowels.

3.9 Sentiment Score Calculation

The sentiment score of each stemmed word has a match with either of the Amharic Sentiment lexicons (Manual, SOCAL, SWN), then the sentiment score is added for each Amharic news comment. Finally, the score is inverted if the comment contains any negation clue. As our approach to sentiment score calculation is the simplest case, intensifier or negation scope is not considered. Finally, the sentiment class of the comment is decided based on the value of the computed sentiment score. If the score is greater than zero, then the sentiment of the comment is positive. If the score is less than zero, the sentiment of the comment is negative. Otherwise, the sentiment of the comment is unclassified or neutral or mixed. Consider the i^{th} Amharic ,a, news comment C_{ai} in Amharic News Comments da has preprocessed and finally, the comment is tokenized into lists. The Amharic Sentiment Lexicon is denoted by s_a . As part of preprocessing, we normalized not only all Amharic words in the Amharic News Comments but handling entries of Amharic Sentiment Lexicon by replacing varied alphabets of the same sound with identical symbols. Moreover, a stemmer is applied after negation identification is completed. As Amharic is morphologically rich, light stemmer is used to reduce the mismatch of Amharic words during string comparison

operation. Thus, the effect of light stemmer on the performance of Amharic Sentiment classification is investigated shortly.

4 EVALUATION OF THE STEMMER

In this section, we present the discussion related to the evaluation of the developed stemmer. The evaluation is made in two ways: including stemmers' performance metrics (e.g. stemmer strength and index compression factor) and present whether the proposed stemmer improves performance of Amharic sentiment classification.

4.1 STEMMER STRENGTH

This tells us the average size of the group of words converted to a par-ticular term regardless of whether they are correct or not. The stemmer strength is measured by computing the number of words per conflation class. Mean of Words per Conflation class (MWC) $= N/S$, where N is number of words before stemming and S is number of words after stemming. The higher the value of the MWC, the stemmer is heavy (or strong) stemmer where it has errors due to over stemming. On the other hand, the smaller the value of the MWC indicates that the stemmer is light (or weak) stemmer where there are errors due to under stemming. The value of MWC of our stemmer is presented in Table 2 on small and large corpus.

4.2 INDEX COMPRESSION FACTOR(ICF)

This is another way of evaluating stemmers' conflation rate. This specifies the extent to which the stemming operation reduces the input word collections to manageable size of index terms for efficient performance of information retrieval. In other words, the smaller the size of the index means that it requires smaller capacity of storage space necessary to store index terms. The value of this factor tells to what extent the collection of words are compressed or reduced by the stemmer. It is calculated as:

$$ICF = (N - S)/N \tag{1}$$

where N is number of words before stemming and S is number of words after stemming. Table 2 depicts the values of ICF of our stemmer on small and large corpus. This result indicates the extent of compression of index terms for IR applications. Table 2. The values of performance metrics of stemmers (Ours and HornMorpho)

Table 2: The values of performance metrics of stemmers (Ours and HornMorpho)

Word Size	**Metrics**	**Our Stemmer**	**HornMorpho**
13968 (LARGE)	No roots	3574	2138
	MWC*	3.91	6.53
	ICF*	0.744	0.846
248 (SMALL)	No roots	35	30
	MWC*	7.09	8.27
	ICF*	0.86	0.88

[1] Discussion:Table 2 presents the performance metrics result of our stemmer compared to performance of hornmorpho on small and large corpus. The result is depicted in terms of parameters including the number of roots generated mean of words per conflation class and in-dex compression factor. Based on the results of stemmers, the strength of our stemmer is weaker (or lighter) than Hornmorpho by nearly a factor of 0.5. That means, hornmorpho removes more affix related strings which could contain semantic information as it removes affixes to get root from input word. On the contrary, our stemmer removes fewer characters from input word to get its corresponding root/stem. For text classification (e.g. sentiment classification), our stemmer might be better than Hornmorpho. This is one of the factors we need to evaluate our stemmer's performance on sentiment classification. The index compression factor of our stemmer is almost close to hornmorpho. This indicates that our stemmer is compressing index terms to considerable level that could save storage space to use it for Information retrieval applications.

[1] *MWC stands for Mean of Words per Conflation class, and *ICF stands Index compression factor

4.3 Stemming Errors

In general, there are two major sources of errors. These include over-stemming where the stemmer removes too many of a term. This tends to make the recall of IR to be high. In other words, the different meanings of terms are diluted into stems. More affixes that contain more semantic information are cleared from the root. On the other hand, under- stemming where the stemmer removes too few of a term. This causes the recall lower. That means a single concept is distributed over a number of stems. In addition, the stemmer could also remove affixes which were part of the root. This leads to an error refers to mis-stemming. Both error types lead to poor performance of the stemmer in information retrieval applications. To this end, two or more words having actually the same stem could have different root as sub-strings of the affix is not stripped of from the root (=under stemming). A group of words having different stems could have the same root by the stemmer as the stemmer removes some affixes that are part of the roots(=over stemming). We can roughly see the number of roots generated by our stemmer compared to Hornmorpho in Table 2 for small and large corpora. On large corpus, our stemmer generates more roots/stems than Hornmorpho by a factor of 0.25. For small corpus, our stemmer generates almost the same size of roots to the size of roots by Hornmorpho. Thus, our stemmer leaves some space for holding semantic information in the generated roots than roots of Hornmorpho.

4.4 On spot Errors Analysis

Yet it is difficult to get a perfect stemmer, there a number of errors in our stemmer based on on-spot analysis. For the sake of simplicity in analysis and interpreting the nature of errors generated in small corpus, let us categorize the errors generated into common stemming error types: under-stemming, over-stemming, unchanged, spelling errors and other mis-stemming errors presented in Table 3.

Table 3: The categories of errors generated by stemmers (Ours and HornMorpho) on small corpus(300 words)

Error Types	Number of errors (in %)	
	Our Stemmer	HornMorpho
Under-Stemming	9%(28)	20%(60)
Over-Stemming	0%(1)	4%(11)
Spelling errors	0%	0%
Unchanged	3%(15)	1%(3)
Others	8%(24)	6%(17)
Accuracy	77%(231)	70%(208)

Discussion: For small corpus of 300 words, stemmer errors related to under-stemming, over-stemming, spelling errors, unchanged (input word is unchanged), accuracy and other error reports are presented in Table 3. The errors related to under-stemming and over-stemming of hornmorpho is higher than our stemmer. The word **ቅዱስም**/'kidus' means 'saint' /is stemmed **ቅድ_ኡስ** and **ቅድስ** by hornmorpho and our stemmer. But our stemmer is correctly generating the root word **ቅድስ**. Hornmorpho generates stem **ቅድ_ኡስ** which is not in root form and its stem size greater than the root size. This might be error related to under-stemming. On the other hand, our stemmer leaves greater number of input words than hornmorpho. Similarly, the numbers of input words without root are greater in our stemmer. However, the accuracy (77%) of our stemmer is greater than the accuracy of hornmorpho(70%). Our stemmer has got errors on removing affixes, mainly single letter suffix, prefix or detecting and removing reduplication is quite problematic to the accuracy of our stemmer. For example, the word **በቅዱስ** has prefix **በ**/b/ and has roots **ቅድ_ኡስ** and **ብቅድስ** by horn-morph and our stemmer respectively. To address this problem, it requires disambiguation model to recognize whether a single letter prefix or suffix is the right affix that should be removed from the root. Some-times, reduplication strings are part of root word. This also requires some heuristic to correctly identify the right infix that should be re-moved from the root.

4.5 Amharic Sentiment Classification

Prior to sentiment score calculation of Amharic facebook comments, we perform basic text preprocessing operations. The effect of stemming and negation detection technique on Amharic text is investigated to increase the accuracy of lexicon based Amharic sentiment classification. The performance of our stemmer is compared with performance of HornMorpho on Amharic sentiment classification as it is shown in Table 4.

Table 4: The Accuracy (in percent) of Lexicons for Sentiment Classification)

Amharic Senti.Lexicons	Accuracies (%)		
	NoStem	Our Stemmer	HornMorpho
Manual +SOCAL + SWN	53.7	86.2	67.9

Discussions:The effect of our stemmer on the performance of Amharic Sentiment Classification is evaluated in terms whether the accuracy of classifying sentiment of Amharic facebook news comments is increased or not. Table 4 presents the results of sentiment classification using our stemmer and HornMorpho. The result reveals the effect of stemming operations on Amharic texts improving the performance of Amharic sentiment classification. The accuracy of Amharic sentiment classification increases from 53.7% to 86.2% by our stemmer and 53.7% to 67.9% by hornmorpho. In conclusion, the results of our stemmer and Hornmorph shows that applying stemmer is necessary for efficient Amharic sentiment classification revealing that our stemmer preserves sentiment information than root stemmers (Alemayehu & Willett, 2002; Gasser, 2017). Besides, the performance of Amharic Sentiment classification is improved by employing stemmer.

5 Conclusions and Recommendations

This work presents design of Amharic light stemmer that removes affixes for hoping to efficiently improve performance of Amharic Sentiment Classification by preserving semantic information. Few of the contributions of this work are summarized as follows:

- The work reveals that Amharic stemmer improves performance of sentiment classification compared to SOTA(i.e. compared to hornmorpho).
- As Amharic alphabets are in the unicode, rather than transliterating into romans using SERA in (Yacob, 1997), we developed custom transliteration of Amharic texts into CV form using Amharic Alphabets
- The approach developed is generic enough that it can be adapted to develop stemmer to other resource limited languages.
- Apart from sentiment classification, our stemmer can also be used to other tasks of natural language processing including information extraction, multilingual semantic lexicons, question and answering, just to name a few.
- Our stemmer is light in the sense that it is efficient for IR applications in terms of processing time to generate root word for a particular Amharic input word with considerable accuracy.
- Related resources including the code will be accessible online for research communities.

Yet, the developed Amharic light stemmer may lack accuracy of correctly stemming Amharic input words. One of the reasons is that the affix list is not comprehensive enough to cover all the variant word forms of the same root. The other thing is that the approach we used is similar to the approach used in look-up table that lacks context information for the affixes specifically prefixes and suffixes of length one might be ambiguous with part of root consonant character. So to handle this, hybrid approach or corpus based methods should be incorporated.

References

Nega Alemayehu and Peter Willett. Stemming of amharic words for information retrieval. Literary and Linguistic computing, 17(1):1--17, 2002.

Mubashir Ali, Shehzad Khalid, and Muhammad Haseeb Aslam. Pattern based comprehensive urdu stemmer and short text classification. IEEE Access, 6:7374--7389, 2017.

Atelach Alemu Argaw and Lars Asker. An amharic stemmer: Reducing words to their citation forms. In Proceedings of the 2007 workshop on computational approaches to semitic languages: Common issues and resources, pp. 104--110. Association for Computational Linguistics, 2007.

Genet Mezemir Fikremariam. Automatic stemming for amharic text- an experiment using successor variety approach. Unpublished Masters Thesis and Department of Information Science and Addis Ababa University and Addis Ababa, 2009.

The Ge'ez Frontier Foundation. Lexical data repository of geez data. https://github.com/geezorg/data, 2017.

Michael Gasser. Horn morpho. http://www.cs.indiana.edu/ gasser/HLTD11/, 2017.

S. Gebremeskel. Sentiment mining model for opinionated amharic texts. Unpublished Masters Thesis and Department of Computer Science and Addis Ababa University and Addis Ababa, 2010.

Anjali Ganesh Jivani et al. A comparative study of stemming algorithms. Int. J. Comp. Tech. Appl, 2(6):1930--1938, 2011.

Girma Neshir Alemneh, Andreas Rauber, and Solomon Atnafu. Dictionary Based Amharic Sentiment Lexicon Generation, pp. 311--326. 08 2019. ISBN 978-3-030-26629-5. doi: 10.1007/978-3-030-26630-1_27.

Daniel Yacob. Localize or be localized.

Daniel Yacob. The system for ethiopic representation in ascii—1997 standard. Webpage: http://www. abyssiniacybergateway. net/fidel/sera-97. html, 1997.

Baye Yimam. (የአማርኛ-ሰዋሰዉ)yäamarIña säwasäw. Educational Materials Production and Distribution Enterprise(EMPDE), 2000.

6 Appendix

Amharic Affixes compiled in this research.

1. Suffix lists : ‘ኢዕኧልኧሽ|ኣውኢው|ኣችኧውኣል|ኧችኣት|ኧችኣችሀኡ|ኧችኣችኧው|ኣልኧሀኡ|ኣውኦች|ኣልኧህ|ኣልኧሽ|ኣልችሀኡ|ኣልኣልኧች|ብኣችኧውስ|ብኣችኧው|ኣችኧውን|ኣልኧች|ኣልኧን|ኣልኣችሀኡ|ኣችሀኡን|ኣችሀኡ||ኣችሀኡት|ውኦችንን|ውኦችን|ኣችኧው|ውኦችኡን|ውኦችኡ|ኧውንኣ|ኦችኡን|ኦውኦች|ኧኘኣንኧትም|ኧኘኣንኣ|ኧኘኣንኧት|ኧኘኣን|ኧኘኣውም|ኧኘኣው|ኘኣውኣ|ብኧትን|ኣችሀኡም|ኦውኣ|ኧችው|ኧችኡ|ኤችኡ|ንኧው|ንኧት|ኣልኡ|ኣችን|ክኡም|ክኡት|ክኧው|ኧችን|ኧችም|ኧችህ|ኧችሽ|ኧችን|ኧችው|ይኡሽን|ይኡሽ|ኧውኢ|ኣችንንኣ|ኣውኢ|ብኧት|ኦች|ኦችኡ|ውኦን|ኧኘኣ|ኘኣውን|ኘኣው|ኦችን|ኣል|ኧም|ሸው|ክም|ኧው|ትም|ውኦ|ውም|ውን|ንም|ሸን|ኣች|ኡት|ኢት|ክኡ|ኤ|ህ|ሽ|ኡ|ሽ|ክ|ኧ|ኧች|ኡን|ን|ም|ንኣ|ው’
   Converted to SERA : 'iI2IaLIaX|AWiW|AcIaWAL|IacAt|IacAchU|IacAcIaW|ALIahU|AWOc|
   ALIah|ALIaX|ALchU|ALALIac|BAcIaWs|BAcIaW|AcIaWn|ALIac|ALIan|
   ALAchU|AchUn|AchU||AchUt|WOcnnA|WOcn|AcIaW|WOcUn|WOcU|
   IaWnA|OcUn|OWOc|IaNAnIatM|IaNAnA|IaNA-nIat|IaNAn|IaNAWM|
   IaNAW|NAWA|BIatn|AchUM|OWA|IacW|IacU|EcU|nIaW|nIat|
   ALU|Acn|kUM|kUt|kIaW|Iacn|IacM|Iach|IacX|Iacn|IacW|YUXn|
   YUX|IaWi|OcnnA|AWi|BIat|Oc|OcU|WOn|IaNA|NAWn|NAW|
   Ocn|AL|IaM|XW|kM|IaW|tM|WO|WM|Wn|nM|Xn|Ac|Ut|it|
   kU|E|h|X|U|X|k|Ia|Iac|Un|n|M|nA|W'

2. Prefix lists: ‘ስልኽምኣይ|ይኽምኣት|ዕንድኽ|ይኽትኽ|ብኽምኣ|ብኽትኽ|ዕኽል|
ስልኽ|ምኽስ|ዕይኽ|ዕኽስ|ዕኽት|ዕኽን|ዕኽይ|ይኣል|ስኣት|ስኣን|ስኣይ|ስ ኣል|
ይኣስ|ይኽ|ልኽ|ክኽ|እን|ዕን|ዐል|ይ|ት|አ|እ’ Converted to SERA: 'sLIaMAY|YIa-MAt|I2ndIa|YIatIa|BIaMA|
BIatIa|I2IaL|sLIa|MIas|I2YIa|I2Ias|I2Iat|I2Ian|I2IaY
|YAL|sAt|sAn|sAY|sAL|YAs|YIa|LIa|kIa|In|I2n|
IeeL|Y|t|a|I'