Automatic spelling error detection and correction for Tigrigna information retrieval: a hybrid approach

Solomon Gebremariam Desta, Gurpreet Singh Lehal
Department of Computer Science, Faculty of Computing Sciences, Punjabi University, Patiala, India

Article Info

Article history:

Received Jun 6, 2022
Revised Sep 4, 2022
Accepted Sep 26, 2022

Keywords:

Effective hybrid approach
Information retrieval
Spell checker
Spelling corrector
Tigrigna language

ABSTRACT

This paper proposes a hybrid approach to design and implement query spelling error detection and correction (SEDC) for Tigrigna information retrieval (IR). Our approach, which is the main contribution to this work, is fast and robust to achieve better performance and also helps the users to easily insert their corrected queries to retrieve relevant information from the IR. This is achieved by combining the normalized measure of bigram overlap using the Jaccard coefficient (J.C) technique, a dynamic programming algorithm for edit distance, and probability of occurrence, which were used to make suggestions for the misspelt words. Our approach was evaluated on the SEDC subtasks separately. It achieved an F-measure of 98.85% on the spelling error detection subtask and an accuracy of 95.36% on the spelling error correction subtask. Thus, a comparison was conducted between our approach and the existing Tigrigna spell checker. It is found that our approach outperformed the existing spell checker and shows a 5.36% improvement in accuracy. This is by far the most promising result with regard to correcting the misspelt users' queries and improving the overall performance of the IR.

Corresponding Author:

Solomon Gebremariam Desta
Department of Computer Science, Faculty of Computing Sciences, Punjabi University
NH 64, Urban Estate Phase II, Patiala, Punjab, India
Email: solomong6@gmail.com

1. INTRODUCTION

How people share and look for information has been significantly impacted by the internet and the world wide web (WWW) [1]. Probably the most important phenomenon that has affected how information is handled and processed is the advent of the internet. Therefore, it is crucial to have access to any kind of information in a short time and it is so easy for information seekers [2]. Despite recent improvements in search quality made possible by using large search engines like Google and Yahoo. The rapid increase in the size of web information has introduced new challenges for search engines in the manner in which they retrieve search results. In reality, there are still many situations in which users are presented with inaccurate or very poor search results [3].

One of the main issues with search engine queries is misspelt queries in the user's query [4]. Li [5] reported that about 10 to 12% of all search phrases entered into search engines on the web are misspelt. Spelling errors may occur for different reasons. When users type quickly, they may add or delete letters in an unintentional manner. As mentioned in [6], typing an adjacent key on the keyboard by accident is very common, especially on mobile devices with small virtual keyboards. Besides the typographical errors, some of the errors come from the difficulty of spelling itself. Spelling provides a difficult barrier for foreign and native speakers equally, with varying spelling norms, uncertain word-breaking boundaries, and the constant introduction of new terms [7].

Query spelling correction is to automatically detect misspelt queries and find alternative forms in order to correct them [8]. The capability to automatically correct misspelt queries has become crucial in today's search engines. Users commonly make spelling mistakes. Search engine users are especially prone to making mistakes in their queries since they frequently explore novel content. The search engine can better grasp the users' intentions by using automatic spelling correction for queries, which can enhance the effectiveness of information retrieval (IR) [5].

There is local research work that has been done on a spell checker for the Tigrigna language on mobile-phone devices [9]. To find the input strings in a dictionary, they employed a dictionary look-up technique as well as the Levenshtein distance for the edit distance algorithm to suggest the candidate terms for the misspelt query. They also reduced the number of correction candidates by using statistical spelling suggestions based on frequency. Finally, they evaluated the performance of the developed spelling corrector, and the experimental result showed that the algorithm had achieved an accuracy of 92%. However, their approach is very expensive and slow because they have to do a large number of edit distance computations using a dynamic programming algorithm to calculate the distance between the query and every term in the dictionary.

This study has proposed an effective hybrid approach that is associated with the isolated-word error. It attempts to solve the problem of query misspellings by cutting down the number of candidate terms in the dictionary. Instead of computing the edit distances of the query term from every term in the dictionary, we selected a small-sized and good set of candidate terms from the dictionary, and then we computed the edit distances only to those selected terms, not to every term in the dictionary. Among the selected terms, we look at their probability of occurrence to decide which of them is the best candidate term for the misspelt query as a correction suggestion in order to improve the retrieval performance of the Tigrigna IR.

Tigrinya, often written as Tigrigna ("ትግርኛ", Tigrina), is a Semitic language that is part of the Afro-Asian language family [10]. It is primarily spoken in Eritrea and Ethiopia's regional state of Tigray. There are more than 6 million Tigrigna speakers on the globe. The regional state of Tigray has more than 4.3 million Tigrigna speakers, according to [11] and Eritrea has around 2.4 million Tigrigna speakers [12].

The language uses a script derived from the ancient Geez language for writing [13]. Each symbol in Tigrigna is made up of a combination of a consonant and a vowel, and the symbols are arranged in groups of related symbols. There is no noticeable symbol that represents a consonant followed by an inherent vowel for each consonant in each symbol [14]. There are 276 distinct symbols, 20 numerals, and eight punctuation marks in the writing system. Each of the 34 core consonants has seven different shapes or orders. Out of the 276 symbols, 238 (34*7) are distinct. Twenty labiovelars and 18 labialized consonants are among the remaining symbols [15].

Its form of writing system differs slightly from the so-called "Ethiopic" writing system that is used for Amharic and Geez languages. Ethiopian and Eritrean Orthodox Christians still use it as their liturgical language since it is a respected and often ancient language. The most visible graphical units in the "Ethiopic" writing system represent a consonant accompanied by a vowel. A similar form can be found in characters that contain the same consonant followed by distinct vowels [16]. As an example, Table 1 presents some of the character representations of Tigrigna scripts in Latin scripts.

Table 1. Character representation of Tigrigna scripts into Latin scripts [16]

1st order		2nd order		3rd order		4th order		5th order		6th order		7th order	
ሀ	he	ሁ	hu	ሂ	hi	ሃ	ha	ሄ	hie	ህ	h	ሆ	ho
ለ	le	ሉ	lu	ሊ	li	ላ	la	ሌ	lie	ል	l	ሎ	lo
ሐ	He	ሑ	Hu	ሒ	Hi	ሓ	Ha	ሔ	Hie	ሕ	H	ሖ	Ho

The writing system is typically represented as a two-dimensional matrix, with rows of units beginning with the same consonant and columns of units encoding vowels of the same kind. The columns are commonly referred to as "orders". The vowel /e/ is in the first order, the vowel /u/ is in the second order, the vowel /i/ is in the third order, the vowel /a/ is in the fourth order, the vowels /ie/ are in the fifth order, there is no vowel in the sixth order, and the vowel /o/ is in the seventh order in Tigrigna language.

2. METHOD

In this section, our proposed methods and algorithms, which are associated with the isolated-word error rather than the contextual aspect of the misspelt word, are discussed in detail. It combines the normalized measure of bigram overlap using the Jaccard coefficient (J.C) technique. A dynamic programming approach for editing distances, and then sorts them based on their probability of occurrence in order to provide the best candidate term as a suggestion.

2.1. Dataset description

Unfortunately, Ethiopian languages have no standard lexicon available, like Webster's English dictionary for the English language [17]. As a result, we have collected Tigrigna documents from different websites of the British broadcasting corporation [18], Tigrai media house [19], and voice of America [20]. The document collection comprises one thousand four hundred twenty (1,420) documents, and data preprocessing is performed in order to generate the fifty-five thousand three hundred twenty (55,320) unique Tigrigna words which are stored in the lexicon of the indexed corpus. As shown in Table 2, the document collection contains news articles about various topics like accidents, agriculture, defense, education, finance, health, law, politics, and social issues. These news articles are stored in a common folder in plain text format. Plain text is used for document representation because it saves memory and is supported by the majority of programming languages [21].

Table 2. Sources of the indexed corpus

No.	Type of news	#Documents in the corpus
1.	Accident	64
2.	Agriculture	184
3.	Defense	254
4.	Education	105
5.	Finance	94
6.	Health	111
7.	Law	123
8.	Politics	155
9.	Social	330
Total#of documents		1,420

2.2. Spelling error detection and correction tasks

This subsection presents the spelling error detection and correction (SEDC) that performs the following tasks in order to suggest the correct word. The following tasks are carried out as shown in: i) data preprocessing is done to generate valid terms; ii) the given misspelt query is decomposed into bigrams; iii) a small-sized but a good set of candidate dictionary terms are generated; iv) the minimum edit distance required between the query term and the set of candidate dictionary terms is calculated; and v) among the best candidate dictionary terms chosen, we look at the probability of occurrence to decide which of them is the most likely suggested candidate term.

2.2.1. Data preprocessing

User's query preprocessing is an essential procedure in the process of SEDC tasks. During the preprocessing, a series of procedures are applied to produce valid terms. First, tokenization is performed by removing all punctuation marks, control characters, digits, and special characters and replacing them with space. Secondly, normalizing the user's query is done. In this work, normalization is related to the expansion of Tigrigna short words abbreviated by either a period (.) or a slash (/).

In the Tigrigna writing system, there are many abbreviations (shortened words). These short words can be single or compound short words abbreviated by either a period (.) or a slash (/). Looking up in the lexicon of the indexed corpus is not necessary because the short words are expanded to their correct full compound form without searching for the short words in the indexed corpus.

2.2.2. Bigram overlap using J.C

Instead of calculating a series of edit distance computations of the query term from every term in the dictionary, it is important to cut down the number of candidate terms in the dictionary. We choose a small-sized but good set of candidate terms from the dictionary and then compute the edit distance only for those chosen terms, not for every term in the dictionary. This brings us to the measure of distance between two terms, which is called the n-gram overlap technique. An N-gram is a consecutive sequence of n-characters from a given term. In this work we use bigrams, which is an n-gram for n=2.

We enumerated all the bigrams in the query string as well as every term in the indexed corpus. We have built a bigram index which has its own bigram vocabulary and posting lists. We use this bigram index to retrieve all the terms in the standard inverted index that have a good overlap with the query bigrams. When most of the bigrams match between the query bigrams and a term in the lexicon, then we can define that particular term in the lexicon as having a good overlap with the query term. This is one of the ways we can decide that something has a good overlap with the query term and it is a good candidate to suggest as a spelling correction. The retrieval process for plausible definitions of multiple n-grams in common, according to [8], is simply a single scan of the postings for the n-grams in the query term *Q*. In order to generate the set of bigrams for both *Q* and *T*, we designed an algorithm as shown in algorithm.

```
Bigram algorithm
Begin:
      bigrams = []
      for word in [words]:
          count = 0
          for token in word[:len(word) - 1]:
              bigrams.append(word[count:count + 2])
              count = count+1
      return bigrams
End
```

We used a normalised measure of bigram overlap using the J.C technique. It is a commonly used measure of n-gram overlap. Satriady *et al.* [8] demonstrated how the linear scan intersection might be adapted when the measure of bigram overlap is the J.C for measuring the bigram overlap between two groups *Q* and *T*. It is given by (1):

$$J.C = \frac{Q \cap T}{Q \cup T} \tag{1}$$

where *Q* and *T*, respectively, are the two groups of bigrams in the query term *Q* and the group of bigrams in the dictionary term *T*. We move from one dictionary term *T* to the next as the scan progresses, computing the J.C between *Q* and *T* on the fly. We include *T* in the output if the coefficient is greater than the experiment's threshold value; otherwise, we will continue on to the next term in the postings. To compute the J.C., we need a group of bigrams in *Q* and *T*.

J.C always assigns a number between 0 and 1. If the J.C is close to 1, then it is a good overlap. We use J.C to decide whether the query and the lexicon term have a good overlap or not. So, the set Q is going to be the set of bigrams in the query term and the set T is going to be the set of bigrams in the lexicon terms, and then we compute the J.C between these two sets. If the J.C ends up being approximately close to 1, then we can say that the query and the lexicon term have a good overlap. In this work, we defined the threshold to be relatively close to 1, and then we chose the particular lexicon terms as good lists of candidate dictionary terms.

2.2.3. Dynamic programming for edit distance

The smallest edit distance required between the query term and the set of candidate dictionary terms is computed by SEDC to select words from the candidate suggestion lists, as stated below. The edit distance table, created using the application of a dynamic programming algorithm, shows how to transform a query term into a target word with the smallest number of insert, delete, or substitute operations. The tabulation method is the most efficient approach to solve this problem. As a result, the following procedure is used to create an edit distance table between the query term (*Q*) and the target word (*T*).

The edit distance between *Q* and *T* is referred to as ED (*Q*, *T*). The smallest number of steps required to transform *Q* to *T* is computed as follows [22]; i) case 1, if one of the arguments is zero, the base case for the edit distance function is (2):

$$ED\ (0,0) = 0 \tag{2}$$

where ED is the edit distance between 0 and 0, which means we don't need any edit distance operation to transform an empty string into another empty string (3):

$$ED\ (m, 0) = m \tag{3}$$

where ED is the edit distance between m and 0, which means we need to do m delete operations to transform the first m characters into an empty string (4):

$$ED\ (0, n) = n \tag{4}$$

where ED is the edit distance between 0 and n, which means we need to do n insert operations to transform the empty string to the first n characters; and ii) case 2, if both of the arguments are different from zero, the minimum number of steps required to transform from Q into T is (5):

$$ED\ (m,n) = \min ED\ (m, n-1) + 1 \tag{5}$$

where ED is the edit distance between m and n, which means the minimum number of steps required to transform Q into the first n-1 characters of T plus the cost of adding a letter to T, which costs a single operation or (6):

$$ED\ (m,n) = \min ED\ (m-1, n) + 1 \tag{6}$$

where ED is the edit distance between m and n, which means the minimum number of steps required to transform the first m-1 characters of Q into T followed by a delete operation on the very last character of Q or (7)

$$ED\ (m,n) = \min ED\ (m-1, n-1) + 1\ if\ Qm \neq Tn\ and\ 0\ if\ Qm = Tn \tag{7}$$

where ED is the edit distance between m and n, which means the minimum number of steps required to transform the first m-1 characters of Q into the first n-1 characters of T so that we are doing a substitution operation. This would count as one if the two are different, or it would be zero as we are changing the character into itself.

We used a bottom-up technique in order to choose terms with the smallest edit distance between the query term and the target word. We start by computing the edit distance for smaller sub-problems and then using the results of these smaller subproblems to compute the results for larger problems that come after. The outcomes are stored in the edit distance table. For example, the following edit distance table shows three substitution edit operations required to transform from the query term "*ተምህሬቲ*" to the target word "*ትምህርቲ*" as illustrated in Table 3.

Table 3. Edit distance table

	-	ተ	ም	ህ	ሬ	ቲ
-	0	1	2	3	4	5
ት	1	1	2	3	4	5
ም	2	2	1	2	3	4
ህ	3	3	2	2	3	4
ር	4	4	3	3	3	4
ቲ	5	5	4	4	4	**3**

2.2.4. Spelling suggestion using probability

SEDC recommended a list of candidate terms which have the least edit distance from the query term, and now we need to sort them according to their probability of occurrence and present the best one as a suggestion to the user. We used the indexed corpus as well as a list of words compiled from various sources on the internet [18]–[20]. Then the words are stored with their own collection frequencies in the descending order as shown in Table 4. The collection frequency (CF) indicates how often each word occurs in the indexed corpus. Finally, we compared the probability of occurrence of the suggested list of terms, and the one with the highest value of probability was given the highest ranking.

The SEDC detects the misspelt word and suggests the correct words with their probability of occurrence in descending order. It is assumed that the suggested correct words are lexically related to the misspelt words when SEDC predicts them. As demonstrated in [23], the probability of the proposed word alternatives is computed using (8):

$$P(w) = \frac{CF(w)}{TW(w)} \tag{8}$$

where P(w) is the probability of the suggested word, CF(w) is the CF of a given word in the document collection, and TW(w) is the total number of words in the document collection. For example, if the given query is "*ሁራ*" and the suggested words with their probability of occurrence in the document collection are computed as shown in Table 5.

Table 4. Sample list of words with their CF

No.	Word	CF	No.	Word	CF	No.	Word	CF
1.	ኣብ	15,855	11.	ምስ	1,903	21.	ዓም	1,163
2.	ናይ	7,432	12.	ልምዓት	1,827	22.	መንግስቲ	1,158
3.	ካብ	3,985	13.	ዘሎ	1,746	23.	ትምህርቲ	1,138
4.	እዩ	3,345	14.	ምግባር	1,738	24.	ከም	1,062
5.	እቲ	2,953	15.	ትልሚ	1,721	25.	ከይዲ	1,054
6.	ስራሕ	2,826	16.	ስራሕቲ	1,564	26.	ልዕሊ	1,041
7.	ናብ	2,649	17.	መሰረት	1,450	27.	መሬት	1,029
8.	ወረዳ	2,324	18.	እዚ	1,380	28.	ኣብዚ	1,019
9.	ድማ	2,151	19.	ዓመት	1,239	29.	መደብ	1,006
10.	ቤት	2,061	20.	ሰብ	1,180	30.	ግልጋሎት	990

Table 5. Suggested words and their probability values of occurrence

Rank	Suggested word	CF(w)	TW(w)	P(w)
1st	ህልቒ	24	55,320	0.000433839
2nd	ህልዉ	14	55,320	0.000253073
3rd	ህልኽ	2	55,320	0.000036153

3. RESULTS AND DISCUSSION

We have performed experiments on the SEDC subtasks in order to evaluate our proposed approach. Then we calculated the precision, recall, and F-measures as the performance evaluation metrics. Precision is defined as the number of correctly detected misspellings compared to all detected misspellings. Recall is defined as the ratio of the number of correctly detected misspellings divided by the total number of errors in the test dataset [24]. As we have mentioned in sub-section 2.1, we used 55,320 total words in the indexed corpus and then we determined the sample size using Slovin's formula for the purpose of testing [25]. It is calculated in (9):

$$n = \frac{N}{1+Ne^2} \tag{9}$$

where n, N, and e represent sample size, population size, and margin of error, respectively (we can use 5% if there is no given margin of error).

The sample size is calculated at a 95% confidence level, and the indexed corpus of this study contains a total of 55,320 words [18]–[20]. As a result, the sample size using the formula we got $n = \frac{55,320}{1+55,320(0.05)^2} = 397.13$. Since the sample size must be a whole number, then the approximate value of the sample size is 397. That is the sample size of commonly misspelt words that were used for testing purposes in this study.

We consulted two linguistic experts of the Tigrigna language in order to select the commonly misspelt words, which were inflected words, functional words, and words with derivational morphemes. The test dataset was prepared in order to evaluate the number of misspelt words correctly identified by the spelling error detection subtask. Hence, the experimental results of the technique used for the spelling error detection subtask are summarized in Table 6.

Table 6 shows that some experiments were conducted and SEDC detected 388 misspelt words, while 9 misspelt words were not detected. That is out of 397 sample total errors prepared in the dataset, which represents a recall rate of 97.73%. This indicates that the indexed corpus contains the misspellings as the words were collected on the web. Moreover, the correctly spelled words were never wrongly detected as misspellings, resulting in a precision rate of 100%. SEDC proves that it is highly reliable in detecting misspellings correctly by achieving a 98.85% F-measure. That means only nine words, or 1.15%, were not detected and corrected using our approach.

Table 6. Experimental results of the spelling error detection subtask

Error detection measurements	Outcomes
Total errors in the dataset	397
Detected misspellings (TP)	388
Not detected misspellings (FN)	9
Correctly spelled word–incorrectly detected (FP)	0
Recall rate (%)	388/397=97.73%
Precision rate (%)	388/(388+0)=100%
F-measure (%)	2(97.73)(100)/(97.73+100)=98.85%

In addition to the three performance evaluation metrics discussed in Table 6, we could also evaluate how the spelling error correction subtask generates the suggestion list of candidates. The quality of suggestions proposed by the spelling error correction subtask is measured by the relative positions of the correct word spellings in the suggestion list of candidates [23]. In the best case, the right word correction mostly comes out on the top ranked lists of candidates. Hence, the performance of the spelling error correction subtask is determined by selecting 388 words that were correctly detected misspellings by SEDC during the spelling error detection subtask from the test dataset. The test dataset, however, excludes misspelt words linked to people's names, places' names, and technological terminology. Here is an overview of the experimental outcomes as shown in Table 7.

Table 7. Experimental results of spelling error correction subtask

No.	Number of top suggestion(s)	Number of correct suggestions	Correct suggestions in %
1.	Correct suggestion in top one	332	332/388=85.56
2.	Correct suggestions in top three	344	344/388=88.65
3.	Correct suggestions in top five	357	357/388=92.01
4.	Correct suggestions in top ten	370	370/388=95.36

SEDC provided the correct candidate suggestion list as its top answer in 85.56% of the test dataset. The percent of correct candidate suggestions listed among the top three answers is 88.65%. In 92.01% of the test dataset, it is among the top five responses, and in 95.36% of the test dataset, the correct word is present in the top ten words of the suggestion lists of candidates.

In general, we evaluated our proposed hybrid approach separately on SEDC subtasks. Our approach achieved an F-measure of 98.85% on the subtask of spelling error detection. In contrast, the spelling error correction subtask achieved an accuracy of 95.36%. Hence, we compared our approach to the Tigrigna spell checker that is currently available [9]. In terms of accuracy, our approach outperformed the existing spell checker by 5.36%.

4. CONCLUSION

The aim of this research work was to design and implement an automatic query SEDC for Tigrigna IR using a hybrid approach. To solve the problem of query misspellings, we observed that the normalized measure of bigram overlap using the J.C technique improved the performance for looking up in the indexed corpus by cutting it down into a small and good set of candidate dictionary terms instead of computing the edit distances of the query term from every term in the dictionary. To evaluate our proposed approach, we calculated precision, recall, and F-measures as the performance evaluation metrics. It was evaluated separately on the SEDC subtasks. Our approach achieved an F-measure of 98.85% on the subtask of spelling error detection. The spelling error correction subtask, on the other hand, achieved an accuracy of 95.36%. As a result, we compared our approach to the existing Tigrigna spell checker. It is found that our approach outperformed the existing spell checker and shows a 5.36% improvement in accuracy. The findings of this study showed that our approach is fast and robust to achieve better performance and also helps the users to easily insert their corrected queries to retrieve relevant information from the Tigrigna IR. As an extension to this work, it would be interesting to identify and correct real-word errors to solve the contextual meaning of the user's query.

ACKNOWLEDGEMENTS

We would like to take this opportunity to thank the Department of Computer Science at Punjabi University Patiala for their support in accomplishing this research work.

REFERENCES

[1] C. Ziakis, M. Vlachopoulou, T. Kyrkoudis, and M. Karagkiozidou, "Important factors for improving Google search rank," *Future Internet*, vol. 11, no. 2, p. 32, Jan. 2019, doi: 10.3390/fi11020032.

[2] M. Alrwashdeh, O. L. Emeagwali, and H. Y. Aljuhmani, "The effect of electronic word of mouth communication on purchase intention and brand image: An applicant smartphone brands in North Cyprus," *Management Science Letters*, vol. 9, no. 4, pp. 505–518, 2019, doi: 10.5267/j.msl.2019.1.011.

[3] F. Cao, J. Zhang, X. Zha, K. Liu, and H. Yang, "A comparative analysis on digital libraries and academic search engines from the dual-route perspective," *The Electronic Library*, vol. 39, no. 2, pp. 354–372, Jul. 2021, doi: 10.1108/EL-09-2020-0265.

[4] K. Cao, C. Chen, S. Baltes, C. Treude, and X. Chen, "Automated query reformulation for efficient search based on query logs from stack overflow," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, May 2021, pp. 1273–1285, doi: 10.1109/ICSE43902.2021.00116.

[5] Y. Li, "Query spelling correction," in *Query Understanding for Search Engines*, vol. 46, Cham: Springer, 2020, pp. 103–127, doi: 10.1007/978-3-030-58334-7_5.

[6] D. Soyusiawaty and D. H. R. Wolley, “Hybrid spelling correction and query expansion for relevance document searching,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 8, pp. 332–339, 2021, doi: 10.14569/IJACSA.2021.0120838.
[7] J. Gupta, Z. Qin, M. Bendersky, and D. Metzler, “Personalized online spell correction for personal search,” in *The World Wide Web Conference*, May 2019, pp. 2785–2791, doi: 10.1145/3308558.3313706.
[8] W. Satriady, M. A. Bijaksana, and K. M. Lhaksmana, “Quranic latin query correction as a search suggestion,” *Procedia Computer Science*, vol. 157, pp. 183–190, 2019, doi: 10.1016/j.procs.2019.08.156.
[9] A. B. Kiros and P. U. Aray, “Tigrigna language spellchecker and correction system for mobile phone devices,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, pp. 2307–2314, Jun. 2021, doi: 10.11591/ijece.v11i3.pp2307-2314.
[10] A. Fesseha, S. Xiong, E. D. Emiru, M. Diallo, and A. Dahou, “Text classification based on convolutional neural networks and word embedding for low-resource languages: Tigrinya,” *Information*, vol. 12, no. 2, p. 52, Jan. 2021, doi: 10.3390/info12020052.
[11] M. A. G. Yohannes, “The Tigray region of Ethiopia,” in *Language Policy in Ethiopia*, vol. 24, Cham: Springer, 2021, pp. 29–51, doi: 10.1007/978-3-030-63904-4_2.
[12] M. A. Keletay and H. S. Worku, “Developing concatenative based text to speech synthesizer for Tigrigna language,” *Internet of Things and Cloud Computing*, vol. 8, no. 2, pp. 24–30, 2020, doi: 10.11648/j.iotcc.20200802.12.
[13] G. H. Gebremedhin and A. A. Mebrahtu, “Linguistic evolution of Ethiopic languages: A comparative discussion,” *International Journal of Intelligent Systems and Applications*, vol. 8, no. 1, pp. 1–9, 2020, [Online]. Available: https://www.researchgate.net/publication/338448158_Linguistic_Evolution_of_Ethiopic_Languages_A_Comparative_Discussion
[14] S. T. Abate, M. Y. Tachbelie, and T. Schultz, “Deep neural networks based automatic speech recognition for four Ethiopian languages,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020, pp. 8274–8278, doi: 10.1109/ICASSP40776.2020.9053883.
[15] E. D. Emiru, S. Xiong, Y. Li, A. Fesseha, and M. Diallo, “Improving Amharic speech recognition system using connectionist temporal classification with attention model and phoneme-based byte-pair-encodings,” *Information*, vol. 12, no. 2, p. 62, Feb. 2021, doi: 10.3390/info12020062.
[16] Z. Berihu, G. M. Assres, M. Atsibaha, and T.-M. Grønli, “Enhancing bi-directional English-Tigrigna machine translation using hybrid approach,” 2020. [Online]. Available: https://ojs.bibsys.no/index.php/NIK/article/view/835
[17] F. Gereme, W. Zhu, T. Ayall, and D. Alemu, “Combating fake news in ‘low-resource’ languages: Amharic fake news detection accompanied by resource crafting,” *Information*, vol. 12, no. 1, pp. 1–9, Jan. 2021, doi: 10.3390/info12010020.
[18] British Broadcasting Corporation, “BBC news ትግርኛ.,” *BBC News*. https://www.bbc.com/tigrinya (accessed Jun. 10, 2021).
[19] Tigrai Media House, “TMH topics and issues,” *TMHTV.org*. https://tmhtv.org/ፕሮግራም/ (accessed Mar. 10, 2021).
[20] Voice of America English News, “VOA daily news,” *VOA.com*. https://tigrigna.voanews.com/z/2914 (accessed Jan. 23, 2022).
[21] W. S. Pittard and S. Li, “The essential toolbox of data science: Python, R, Git, and Docker,” in *Computational Methods and Data Analysis for Metabolomics*, vol. 2104, New York: Springer, 2020, pp. 265–311, doi: 10.1007/978-1-0716-0239-3_15.
[22] G. Lancia and M. Dalpasso, “Speeding-up the dynamic programming procedure for the edit distance of two strings,” in *Database and Expert Systems Applications*, Cham: Springer, 2019, pp. 59–66, doi: 10.1007/978-3-030-27684-3_9.
[23] P. Kumar, A. Kannan, and N. Goel, “Design and implementation of NLP-based spell checker for the Tamil language,” in *Proceedings of 1st International Electronic Conference on Applied Sciences*, Nov. 2020, p. 7636, doi: 10.3390/ASEC2020-07636.
[24] J. Miao and W. Zhu, “Precision–recall curve (PRC) classification trees,” *Evolutionary Intelligence*, vol. 15, no. 3, pp. 1545–1569, Sep. 2022, doi: 10.1007/s12065-021-00565-2.
[25] I. Aryadinata and F. Samopa, “Analysis acceptance of use Internet banking and mobile banking, case study: Standart application in XYZ company,” *IPTEK Journal of Proceedings Series*, no. 5, pp. 465–472, Dec. 2019, doi: 10.12962/j23546026.y2019i5.6402.

BIOGRAPHIES OF AUTHORS

Solomon Gebremariam Desta received his first degree in Computer Science from Mekelle University, Ethiopia, in 2006 and an M.Sc. in Information Science from Addis Ababa University, Ethiopia, in 2013. Currently, he is a PhD student in Computer Science at Punjabi University, Patiala, India. He can be contacted at email: solomong6@gmail.com.

Gurpreet Singh Lehal holds a Ph.D in Computer Science from Punjabi University, Patiala, India. Currently, he is Dean of the College Development Council, Director of the Research Center for Punjabi Language Technology, and Professor in the Department of Computer Science at Punjabi University, Patiala, India. His research interests include natural language processing, multilingual computing, optical character recognition, Punjabi speech synthesis, and gurmukhi OCR. He can be contacted at email: gslehal@gmail.com.