Design And Implementation Of Morphology Based Spell Checker

Gaddisa Olani Ganfure, Dr. Dida Midekso

Abstract: Introducing texts to word processing tools may result in spelling errors. Hence, text processing application software's has spell checkers. Integrating spell checker into word processors reduces the amount of time and energy spent to find and correct the misspelled word. However, these tools are not available for **Afaan Oromo**, Cushitic language family spoken in Ethiopia. In this paper, we describe the design and implementation of a non-word Afaan Oromo spell checker. The system is designed based on a dictionary look-up with morphological analysis (i.e. morphology based spell checker). To develop morphology based spell checker, the knowledge of the language morphology is necessarily required. Accordingly, the morphological properties of Afaan Oromo have been studied. To the best of our knowledge, this work is the first of its kind for Afaan Oromo. The methodology delineated in the paper can be replicated for other languages showing similar morphology with Afaan Oromo.

Index Terms: Spell checker, non-word error, Error detection, Error correction, Morphology, Morphological Analyzer, Morphological generator, Afaan Oromo, typographic errors, cognitive errors

1 INTRODUCTION

A spell checker is a tool that enables us to check the spellings of the words in a text file, validates them i.e. checks whether they are rightly or wrongly spelled and in case the spell checker has doubts about the spelling of the word, suggests possible alternatives. The two core functionalities provided by a spell checkers are: spelling error detection and spelling error correction. 'Error Detection' is to verify the validity of a word in the language while 'Error Correction' is to suggest corrections for the misspelled word. Spell checker may be stand-alone capable of operating on a block of text, or as part of a larger application, such as a word processor, email client, electronic dictionary, or search engine [1]. Several researches have been done for the languages like English, Arabic, Chinese and few researches have been done for Amharic language, but none for Afaan Oromo. Afaan Oromo (when translated it means Oromo Language) is one of the major African languages that is widely spoken and used in most parts of Ethiopia and some parts of other neighbor countries like Kenya and Somalia. Afaan Oromo belongs to the Lowland East Cushitic sub-family of the Afro-asiatic super-phylum. Among the Cushitic language families to which it belongs, Afaan Oromo ranks first by the number of its speakers [2]. Currently, it is an official language of Oromiya regional state. Despite of its popularity and its status as a regional language, Afaan Oromo language processing is still in its infancy. According to Damerau [3] and Peterson [4] spelling errors are generally divided into two types, typographic errors and cognitive errors.

- *Gaddisa Olani Ganfure is a lecturer of Computer Science Department, Dire Dawa University, Ethiopia. Email: gaddisaolex@gmail.com*
- *Dr. Dida Midekso is an Associate Professor of Computer Science Department, Addis Ababa University, Ethiopia. Email: mideksod@yahoo.com*

Typographic errors occur when writer knows the correct spelling of the word but mistypes the word by mistake. Cognitive errors occur when a writer does not know or has forgotten the correct spelling of a word. A study by Damerau reports that 80% of the misspelled words in English are non-word errors and caused by single error misspellings [3]. We did a simple study to analyze spelling error pattern of Afaan Oromo before implementation. For this purpose, module prepared for teaching Afaan Oromo courses was selected. We used text analysis data gathering technique for this purpose. The finding of study depicts the existence of spelling errors. When analyzed, it was found that 1,342 words were misspelled. Out of this 1,287 words were in the category of non-word errors. Though a comprehensive study is required to come to a clear opinion, it was enough to realize that non-word error detection is the first step towards a truly professional spellchecker. The paper is organized in to the following sections. Section 2 discusses the challenges in building a spell checker for Afaan Oromo and the work done so far. Section 3 discusses the design of the system. Discussion and results are discussed in Section 4. Finally the paper ends with some concluding remarks.

2 Challenges and Related work

As stated in [5] like a number of other African languages, Afaan Oromo has a very rich morphology. In agglutinative languages most of the grammatical information is conveyed through affixes and other structures. Therefore, the grammatical information of the language is described in relation to its morphology. As Afaan Oromo is an agglutinative and morphologically rich language, each root word can combine with multiple morphemes to generate huge number of word forms. For the purpose of supporting such inflectionally rich languages, the structure of each word has to be identified. Afaan Oromo has compound, derived and simple nouns, verbs, and adjectives. It also has first person, second person, and derived pronouns. Nouns get inflected for number. Gender, number, tense, voice, aspect and mood cause inflections to verbs. Many times it is context which decides whether a word is a noun or adjective or adverb or post position. This increases the complexity of parsing Afaan Oromo. Because of all these reasons development of a spell checker for Afaan Oromo is a challenging task.

3 Design

Taking what we obtained from the review of the literatures and the morphological complexity and resource scarceness of Afaan Oromo language, we proposed a morphology based spell checker (i.e. a dictionary look-up with morphological rules). A morphology based spell checker has advantages such as its ability to reduce the dictionary size drastically and the ability to recognize new words that are not included in the dictionary. Morphological rules address word categories and their possible inflections, derivation and compounding. Further, the approach can be drawn upon in building grammar checkers. A morphological rule developed for the spell checker is also a stepping-stone for other NLP applications. The architecture of the system is shown in the Figure 1. The architecture has eight components: Tokenizer, Knowledge base, Error detection, Morphological analyzer, Error correction, Morphological generator, Suggestion ranker and Word assembler.

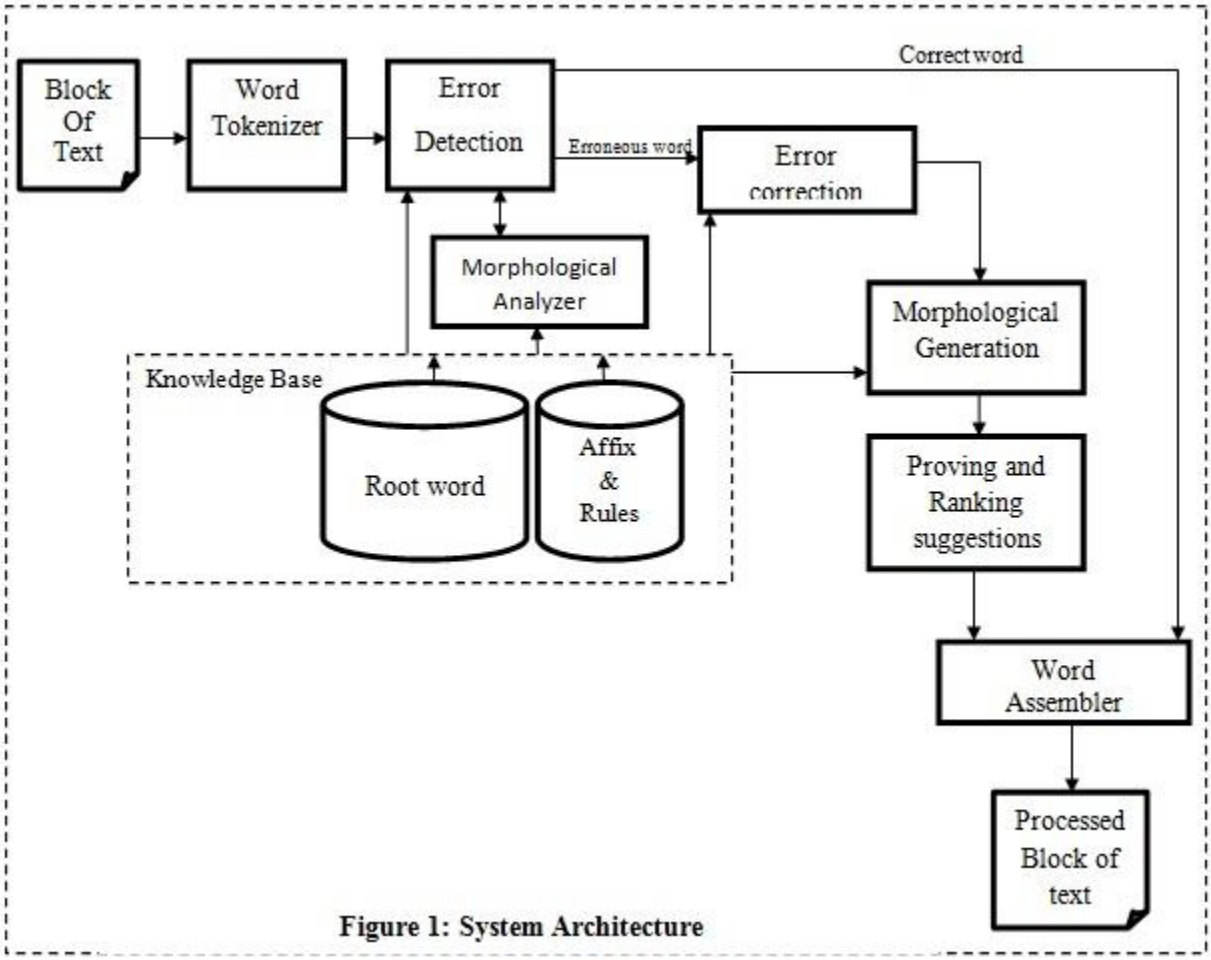

Figure 1: System Architecture

a. Tokenizer component

This component split a block of text into individual words, digits and punctuation marks. In Afaan Oromo, like in English languages, the blank space shows the end of one word. Moreover, parenthesis, brackets, quotes, etc are being used to show a word boundary. Furthermore, sentence boundaries and punctuations are almost similar to English language (i.e. a sentence may end with a period (.), line break, a question mark (?), or an exclamation point). Thus, space marks are used as the explicit delimiters or token separator. Every time a space is encountered, the word after the space becomes a token. The output of this component (i.e. list of tokens) becomes an input to error detection module.

b. Knowledge base component

Knowledge of the language plays an important role in order to design a morphology based spell checker. Such knowledge can be obtained in various ways. In this study, the knowledge for morphological rule and lexicon design was obtained from the analysis of Afaan Oromo text books, published papers, and discussion with Language professionals. In order to build a spell checker for Afaan Oromo, one would require a powerful dictionary for reference in the word error detection and suggestions prediction phases. Creating a dictionary having all the words of the language is a laborious task and infeasible considering the large variation of affix combinations in Afaan Oromo. To handle this problem, we used a morphological analyzer and a dictionary of root word. For instance, in Afaan Oromo for a single verb root word "***beek-***" 'go', over 800 valid word forms can be formed. We adopt the Hunspell dictionary and affix file format to design a lexicon (i.e. the knowledge base component). Hunspell is an open source spell checker [6]. It has been designed especially for languages with rich morphology and complex

system of word compounding, originally for Hungarian [6]. Even though our spell checker is a standalone application, storing both the root words and affix in this format will help us to directly integrate our system into Apache OpenOffice. All information (or rules) required for spelling error detection, morphological analysis and error correction are stored in this module. It contains a root word and affixes for different word classes.

c. Error Detection component

Error detection component is responsible for checking whether the input word is misspelled or not. The error detection component works first by looking the input word in the root word dictionary. If the input word exists in the root word dictionary, the spell checker does nothing. Otherwise (i.e. if this component returns 'not exist'), the input word will be sent to the morphological analyzer component for further processing. The morphological analyzer component decomposes the input word into the possible roots and affixes (based on their signature) and then passes them to the error detection component. Then, the error detection component once again lookups the knowledge base to check whether the returned root word and affix exists in the knowledge base or not. If they do not exist, the error detection component will recognize that it's a misspelled word and then passes them into the error correction component. If both the root word and affix are found in the knowledge base, the system cannot automatically say this word is valid, for the root word may not be inflected or derivated for this affix. Finally, to determine if this word is an acceptable word, the class of this root word is checked in the root word dictionary. If it has this affix flag, the system will recognize it as a valid word (i.e. no further processing is needed), otherwise the error detection component will recognize it as misspelled word. We adopt the Hunspell [7] hashing algorithm for lookup. Hashing is a well-known and efficient lookup strategy. If the word stored at the hash address is the same as the input string, there is a match. However, if the input word and the retrieved word are not the same or the word stored at the hash address is null, the input word is indicated as a misspelling. The random-access nature of hash tables eliminates the large number of comparisons required for lookups.

d. Morphological Analyzer component

The task of this component is to accept a list of words from error detection component and then decompose each word into root words and affixes and then pass them to the error detection component. Since we are using a dictionary lookup with morphological rules to develop the system, we develop knowledge based morphological analyzer algorithm. The proposed morphological analyzer algorithm is depicted in Figure 2.

Input: word **I_Word** from Error detection component
Output: list of affix and root words
Start

1. **Scan** input word from right to left and left to right to look for valid suffix and prefix
   For each valid suffix in I_Word strip them and store result in a buffer
   For each valid prefix in I_Word strip them and store result in a buffer
   //pass list of affix and stems to the error detection module
   Return root and affix
   //If there is no valid suffix and prefix
2. **Scan** input word from left to right and right to left and then look for a possible roots
   For each valid roots in I_Word strip the root and store result in a buffer
   //pass list of possible roots to the error detection module
   Return root and affix //valid roots and invalid affix

END

Fig 2: *Algorithm for Morphological Analysis*

This algorithm (Figure 2) makes use of rules stored in the knowledge base to strip a given word into its root words and affix. In this process, each individual word is scanned from right to left (and right to left) in the affix file and root word dictionary. Upon finding a valid affix, it is stripped from the word. However, the exact affix stripping is possible only for a correctly spelled word. In the case of misspelled word, as there is an ambiguity as to whether the error exists in the root or affix, only probable affix stripping can be done. To illustrate how the error detection and morphological analyzer works, consider the unknown word "***bishaaniin"*** 'by water'. The error detection module will first check if ***bishaaniin*** is found in the root word dictionary. Since it is not found in the dictionary, the system cannot automatically say that it is misspelled, for it may be inflected, derivated or compounded. Then the error detection component will call the morphological analyzer. To determine if this word is acceptable, morphological analysis will be done. The morphological analysis algorithm will first scan from the right to left and left to right to search for a valid suffix and prefix. Since /***-iin***/ is a valid suffix in Afaan Oromo, the morphological analyzer component will strip /-iin/ from ***bishaaniin*** and returns suffix */-iin/* and unknown morpheme /***bishaan-I***. Now the error detection component checks the unknown morpheme ***bishaan*** for its presence in the root word dictionary. Since it is found in the root word dictionary, the system cannot automatically say ***bishaaniin*** is a valid word, for the root ***bishaan*** may not be inflected or derivated for the suffix */-iin/*. Finally, to determine if the unknown word ***bishaaniin*** is an acceptable word, all the rules required to append suffix */-iin/* will be checked in the affix file (e.g. any nouns ending with consonant 'n' can append suffix /*-iin*/).

Now the error detection component will recognize *bishaaniin* as a valid word, and no further processing is needed. The same process will be done for a misspelled word.

e. Error Correction component

This component performs two tasks. First it accepts the pairs obtained after affix stripping in morphological analyzer from error detection component to classify the errors into one of the following classes:

- ✓ Valid prefix, invalid root, and invalid suffix
- ✓ Valid prefix, invalid root, and valid suffix
- ✓ Valid prefix, valid root, and invalid suffix
- ✓ Invalid prefix, valid root, and valid suffix
- ✓ Invalid prefix, valid root, and invalid suffix
- ✓ Invalid prefix, invalid root, and valid suffix
- ✓ Invalid prefix, invalid root, and invalid suffix
- ✓ Valid prefix, valid root, and valid suffix // incorrect combination

After classification, this component will make a probable correction to the misspelled morphemes based on their error classes. A single erroneous word may be classified under two or more classes and each of the error is handled separately. For instance, in the case of a valid affixes (prefix and suffix) and invalid root, the valid affix will give us the specific category of the possible root words. Similarly on finding that the suffix is invalid and root is valid, valid root will give us the specific category of the possible words. Then using the replacement rule and Levenshtein Edit Distance (LED) possible correction will be done. Rules in the knowledge base and Levenshtein Edit Distance techniques have been used for error correction. Edit distance is the number of simple edit operations required to transform one string to another [8]. The different operations allowed are substitution of a letter, deletion of a letter, insertion of a letter, and transposition of two adjacent letters. The less the edit distance between two strings is, the more similar are the strings to each other.

f. Morphological Generator component

This component will be called to put together the corrected parts (i.e. morphemes) from the error correction component to re-build the complete word form. In addition, based on the rules in the knowledge base it determines the various words that can be generated from a given valid morphemes. This component is again used to correct and generate suggestion for those errors resulted from valid root and valid affix (invalid combination of roots and affixes). If no candidates can be found after error correction and morphological generator, no candidates will be generated. The input to our morphological generator component can be a suffix, a prefix, a root word or combination of them. Taking this into account, we proposed knowledge based morphological generator algorithm for Afaan Oromo, which takes any combination of morpheme (s) as the input. This word form generation algorithm takes root (or stem) or affix as an input, then it identifies the class of each root words and retrieves corresponding affixes to generate a valid word form.

g. Providing and Ranking Suggestions

Once the process of error correction and morphological generation was done, the next step is to provide and rank suggestions for the detected error. Hence upon detecting the error, the user will be provided with a list of probable correct words which the user can select to update the misspelled word. It may also be possible that the correct word expected by the user may not be listed in the suggestions; if there is no possible root word and affix. Keyboard layout (i.e. character distance), replacement rule in the affix file, and Levenshtein Edit Distance has been used to rank the suggestion. The character distance method uses a Pythagorean-type metric to measure the distance between a misspelled word and a possible correction, based on the **QWERTY** keyboard layout [9]. The QWERTY keyboard is represented as two-dimensional arrays as shown in Figure 3. The suggested word with the shortest distance to the misspelled word is considered as the best suggestion.

i/j	0	1	2	3	4	5	6	7	8	9	10	11	12
0	Q	W	E	R	T	Y	U	I	O	P	[	]	\
1	A	S	D	F	G	H	J	K	L	;	‘		
2	Z	X	C	V	B	N	M	,	.	/			

Fig 3: *Two-dimensional representation of QWERTY Keyboard*

According to the Euclidean distance formula, the distance between two points in the plane with coordinates (x, y) and (a, b) is given by:

dist ((x, y), (a, b)) = $\sqrt{(x-a)^2+(y-b)^2}$ (1)

For example, based on the character distance chart in Figure 3, the distance between **w** located at (0, 1) and **n** located at (2, 5) is 4.47 and the distance between **n** located at (2, 5) and **m** located at (2, 6) is 1. This indicates that there is highest probability to mistype **n** as **m** than **w**. If the erroneous word is corrected by rules in the replacement table, word formed by this rule will take the top rank in the suggestion list. If two or more possible words are generated using the replacement rule, the top rank would be given to the one with the smaller LED. Character distance is considered only if there are words having the same LED. To illustrate how this component works, consider the misspelled word “***hindeemi”***. This word can be analyzed into the following pairs: valid prefix /**hin-/** ‘don’t’ and valid root **/deem-/** ‘go’ and valid suffix /**-i/**. We have the following rules in our knowledge base:

1. Any verb stem ending with consonant **m** can append suffixes /**–e/**, **/-a/**, **/-i/**, **/-u/**, **/-ti/** and the like without any criteria. This indicates that "***deemi***" is a correct word form.
2. After appending suffix like /**–e**/, /-**a**/, /-**u**/, /-**ti**/ etc, any verb stem ending with consonant **m** can also append prefix ***/hin-/*.** This indicates that "*hindeemi"* is incorrect combination.

In this case, the morphological generator will generate too many suggestions. By considering the error is because of prefix, words like ***deemi, deeme, deema, deemte, deemteetti*** and the like may be generated for suggestion. Again by considering the error is because of suffix ***/-i/***, words like ***hindeeme, hindeemti, hindeeme, hindeemu*** and the like may be generated for suggestion. The same process is done for the root word. Listing and displaying all the possible suggestions to a user makes confusion, hence ranking and trimming of suggestion is needed. Table 1 shows the number of edit operation required to convert misspelled word *hindeemi* to candidate word.

Table 1
LED required for converting "*hindeemi"* to candidate words

Candidate word	Insertion	Deletion	Substitution	Transposition	Total
deemi		3			3
deeme		3	1		4
deemte	1	3	1		5
hindeeme			1		1
hindeemti	1				1
hindeema			1		1
hindeemu			1		1

As shown in Table 1, the LED of the first three words is higher, thus they are trimmed and the last four candidates will be selected for ranking. Since all of them have the same LED, another criterion is needed to rank them. As observed from Afaan Oromo spelling error pattern analysis, the probability of making insertion and omission errors are higher than the other types. Thus, if two or more words have the same LED we give the top rank for insertion and omission errors, for this reason candidate word ***hindeemti*** will rank number 1. Now it's time to consider keyboard layout to rank the three words left. They differ from the erroneous word by one character, so we need to find the distance between the last characters of misspelled word ***hindeemi*** '***-i***' and the last character of all the other three words. Based on their score in Table 2, the one with the smallest character distance will be ranked first. Finally for the misspelled word ***hindeemi***, suggestion provider and ranker will return **hindeemti**, **hindeemu**, **hindeeme** and **hindeema** as the top four suggestions.

Table 2
Ranking suggestion for *hindeemi* with their character edit distance

Candidate word	Last character	Distance from i (0, 7)	Rank
hindeeme	e (0, 2)	5	2
hindeema	a (1, 0)	7.07	3
hindeemu	u (0, 6)	1	1

h. Word Assembler component

The task of this component is to combine correct word with the one flagged as misspelled word.

4 RESULTS AND DISCUSSION

The prototype of Afaan Oromo spell checker is developed using Microsoft Visual C# 2010. The snapshots of AOSC are depicted in Figure 4. The input for the prototype is a text file. The text file can be browsed or typed directly into the textbox. The system will check the spelling automatically after the space bar is pressed or automatically after the saved text file was exported. A word that the system believes to be misspelled is flagged with a wavy red underline and suggested corrections are available in a pop-up menu after right clicking on the erroneous word.

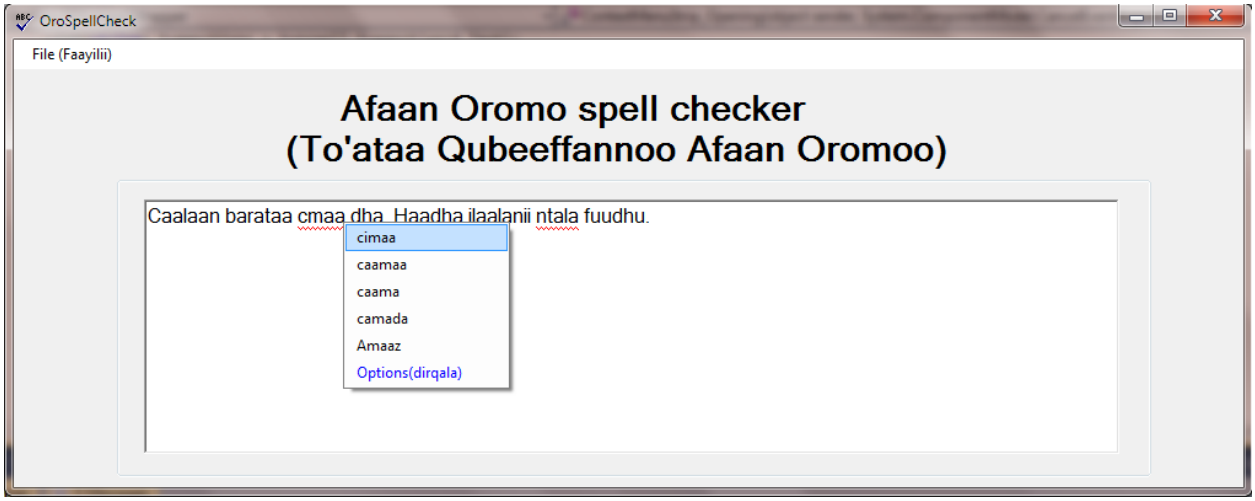

Fig 4: *Snapshot of Afaan Oromo Spell checker*

Besides our main objectives, we have also integrated our system (i.e. the knowledge base) into Apache OpenOffice 3.4.1 windows version for demonstration purpose. Figure 5 shows screen shots taken from OpenOffice word after integrating our system; while spell checking a sample text.

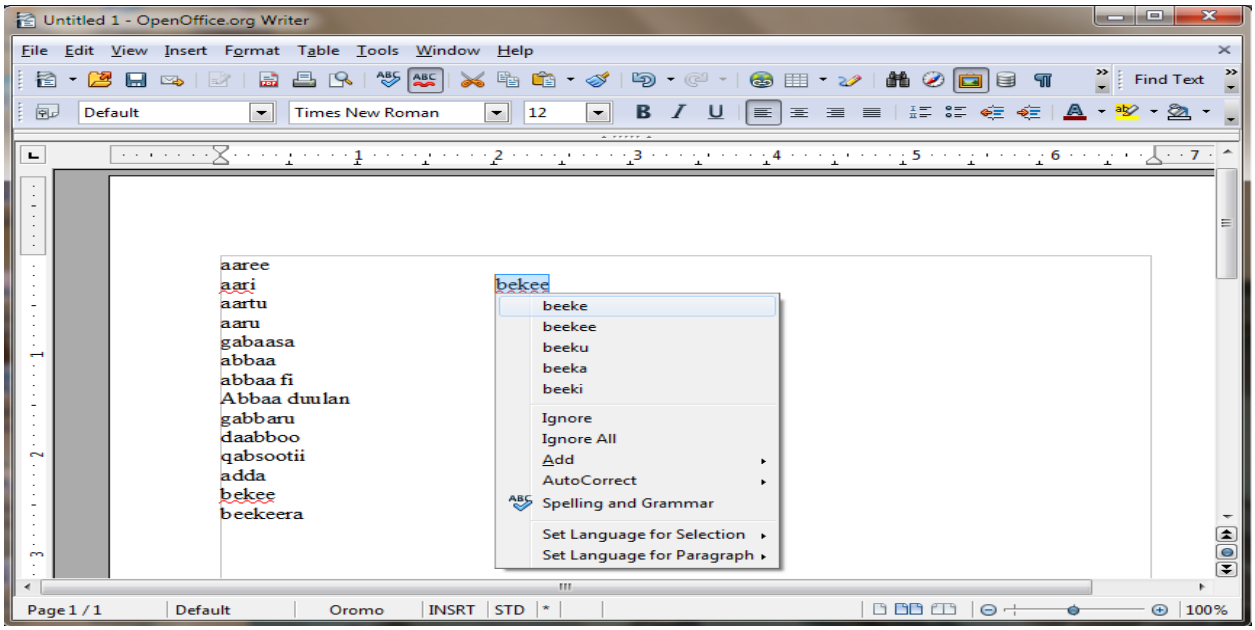

Fig 5: *Screen shot of OpenOffice*

The designed system must be evaluated to test its effectiveness. In the literature, several methods for evaluating spell checker system have been proposed. A work done by Kukich [8] proposed lexicon size, test set size, correction accuracy for single and multi error misspellings, and type of errors as evaluation criteria for a spell checker tool. A research work by Paggio et al. [10] recommend error recall, precision recall, interface and suggestion adequacy for the evaluation of a spell checker algorithm. Some of the measurements are subjective and difficult to evaluate. In this research work, more directly measurable parameters such as lexical recall, error recall, and precision are used to evaluate the developed system. We will follow Paggio et al. [10] In their definitions of the metrics. Valid words are words that are part of the language, or which are sanctioned by the language system, in contrast to invalid words, which are not part of the lexicon or language system. When the spelling checker claims a word is invalid, it is flagging that word, while accepting a word means treating it as valid. Accordingly, a flag is an indication that a word has been tagged as invalid. Suggestions are alternative valid words that are offered to the user to replace a flagged word with. With this information, it becomes intuitively clear what the metrics actually measure. Lexical recall indicates the percentage of valid words correctly accepted by the spelling checker, Error recall gives the percentage of invalid words correctly flagged and Precision gives the percentage of correct flags (correctly found invalid words) over all flags by the spelling

checker. Description about how these measurements are calculated is given in Table 3.

Table 3
Evaluation metrics

Metric	Measurement Method
Lexical Recall	# of valid words accepted / # of valid words
Error Recall	#of invalid words flagged / # of invalid words
Precision	#of correctly flagged invalid words / # of words flagged

It should be noted that the evaluation methods presented in Table 3 works best in a controlled environment (i.e. where the test data and the knowledge base are from the same or a similar source), but this work was evaluated on the dataset that was randomly taken from different sources. The test dataset was prepared to evaluate the number of valid words correctly accepted by the system and the number of invalid words correctly flagged by the system. Initially, we randomly selected sentences (and paragraphs) from stories and papers belonging to several domains which produced 6521 words. To trim repeated words and select derivated, inflected and compounded words Alchemist 2.0 has been used. This reduced the 6521 to 1464 unique words including compound words, words with derivational morphemes, inflected words, and functional words. The word lists are printed out and then manually spell checked by three postgraduate linguistic students. We found that, the data set consists of 1385 correctly spelled words and 79 misspelled words. In addition, we manually generated and added some inflection and derivation variants for the root word ***deem***- 'go', ***qab***-'hold' and beek-'know', a total of 347 unique words to the test data; making the total words 1811. In this sample, out of 1,732 valid Afaan Oromo words, 1,535 were accepted as a valid word; 197 words were flagged as misspelled words by the system. Out of the 197 incorrectly flagged words 186 words are due to the absence of root word in the lexicon, whereas the remaining 11 words have root words in the lexicon, but the spelling checker did not recognize the inflection, derivation and compounding rules. On the other hand, all the 79 misspelled words were flagged. The result of the test data set is shown in Table 4.

Table 4
Evaluation result

Description	Value
Lexical recall	(1,535/1,732)*100 = 88.62%
Error recall	(79/79)*100 = 100%
Precision	(79/276)*100 = 28.62%

In addition to the three metrics discussed in Table 3, we could also test how the spell checker generates a preferred suggestion. Ideally, the spell checker should only suggest the preferred suggestions. However, in practice it is sometimes unclear what the preferred suggestion really is. For example, what should be the suggestion for the invalid word: "***deem***"? It could be "***deemi***", "***deeme***", or "***deemu***", among possible others. It depends on the context. However, we only looked at each misspelled word and then check whether a right suggestion is generated or not based on our algorithm (i.e. without considering the context). The test shows that for all the flagged words a possible suggestion was generated. Generally, from the result and the feedbacks and suggestion of the linguists who evaluated the system, we observed that enhancement in the knowledge base will improve the accuracy of the system. Considering our knowledge base lexicon size, the obtained result is satisfactory. Moreover, we have observed that morphology in Afaan Oromo is complex, so it needs to be studied linguistically to promote the computational aspect.

5 Conclusion

Since most computers work in English and other few languages, people who do not speak such languages are either forced to access computers in those languages or will not use them at all. This has its own impact on the socio-economic development of that country. In order to increase the usability of computer devices and let people express their ideas using their native languages, these devices need to be localized into native languages. Hence, it is necessary to do research to alleviate these problems. Spell checker is one potential candidate to this. Use of computers for document preparation is one of those many tasks undertaken by different organizations. Introducing texts to word processing tools may result in spelling errors. Hence, text processing application software has spell checkers. Integrating spell checker into word processors reduces the amount of time and energy spent to find and correct the misspelled word. This study is the first work for Afaan Oromo, and we hope that this research paves a way for a full fledged Afaan Oromo spell checker for those who want to pursue conducting research in natural language processing for Afaan Oromo language.

Reference

[1] Rajashekara Murthy, Vadiraj Madi , Sachin D, Ramakanth Kumar, "A NON-WORD Kannada Spell Checker Using Morphological Analyzer And Dictionary Lookup Method", IJESET, 2(2), 43-52, 2012

[2] Tesfaye Tolessa, "Early History of Written Oromo Language up to 1900", Star Journal, 1(2):76-80, 2012

[3] F.J. Damerau, "A technique for computer detection and correction of spelling errors", Communication of ACM, 7(3), 171-176, 1964.

[4] Peterson, L.J, A Note on Undetected Typing Errors. In Communications of ACM, 29(7): 633-637, 1986

[5] G. Q. A. Oromoo, "Caasluga Afaan Oromo Jildi I, Komishinii Aadaaf Turizmii Oromiyaa", Finfinnee, Ethiopia, Pp: 105-220, 1995.

[6] Peter Halacsy, “Open language resources for Hungarian”, in proceedings of LREC, European Language Resources Association, 2004.

[7] Hsuan Liang. “Spell checker correctors: A unified Treatment.” Master’s Thesis University of Pretoria, South Africa, 2008.

[8] Karen Kukich, “Techniques for Automatically Correcting Words in Text”, ACM Computing Survey, 24(4), 377–439, 1992

[9] Min, K., Wilson, W., and Moon. Y, “Syntactic and Semantic Disambiguation of Numeral strings using an n-gram Method”, Advances in Artificial Intelligence, Springer, Berlin. Pp: 82-91, 2005.

[10] Paggio P., Music B., “Evaluation in the SCARRIE project”, In the proceedings of the first International conference on language resources and evaluation, 1998.