International Journal of Software Engineering.
ISSN 0974-3162 Volume 8, Number 1 (2017), pp. 1-12

http://www.irphouse.com

Spell Checker for Khasi Language

Mark Sheridan Nonghuloo
Department of Computer Science
Christ University
Bengaluru- 560029, India

Karthik Krishnamurthi
Department of Computer Science
Christ University
Bengaluru- 560029, India

Abstract

Natural Language Processing (NLP) is a standout amongst the most vital research areas completed in the realm of Human dialect. For each dialect, spell checker is a basic part of a large number of the basic Desktop applications, Machine Translation framework and Office Automation framework. In Meghalaya, Khasi Language is utilized as an official dialect. Khasi Pronunciation and orthography has differences because spelling is often not an accurate reflection of pronunciation. In this paper, we created Khasi Spell Checker which can deal with Typographic Errors (Non-word Errors) of Khasi words. In the event that incorrectly spelled word contains in the sentence, this framework can rectify the incorrectly spelled Khasi words. We apply Khasi Corpus to check Khasi words. And after that we utilized String Cosine Similarity to produce redress word for mistyped Khasi words. The system can improve the quality of suggestion for misspelled Khasi words and users' efficiency when the users cannot figure out the correct spelling by themselves.

Keywords: Khasi spell checker, natural language processing, string cosine similarity, tokenization, text corpus, tokenization, error correction, error detection

I. INTRODUCTION

A software program or program feature designed to identify misspelled words and notify the user of the misspellings. Depending on the spell checker, the feature may either auto correct the word or allow the user to select from potential correct words of the misspelled word. The spell checker works by comparing every word typed with a list of thousands of correctly spelled words and then uses algorithms to determine the correct spellings, either a list of correct words is given to replace the erroneous word or the erroneous word is replaced automatically. Although spell checkers have become one of the most commonly used features in many programs, they have also become a hindrance. Some have become so dependent on spell checkers that their spelling and grammar skills have declined, and they have a difficult time writing anything correctly without the help of a computer.

II. KHASI LANGUAGE

Khasi is a language spoken primarily in the state of Meghalaya in India by the Khasi people. Khasi is part of the Austroasiatic language family, and is related to Cambodian, Vietnamese and Mon languages of Southeast Asia. The language uses 23 alphabets by removing the letters c, f, q, v, x and z from the basic Latin alphabet and adding the diacritic letters ï and ñ, and the digraph ng, which is treated as a letter in its own right.

Table 1: Khasi Alphabets

a (aah)	b (bee)	k (kay)	d (dee)	e (eh)
g (eg)	ng (eng)	h (esh)	i (ee)	ï (yee)
j (jay)	l (el)	m (emm)	n (enn)	ñ (eiñ)
o (oh)	p (pee)	r (arr)	s (ess)	t (tee)
u (oo)	w (double yuu)	y (why)		

Khasi has a pervasive gender system. There are four genders in this language: u – masculine, ka – feminine, i – diminutive, ki – plural. Khasi has a classifier system, apparently used only with numerals. Between the numeral and noun, the classifier *tylli* is used for non-humans, and the classifier *ngut* is used for humans.

In English language, we have postfixes to words (verbs and adjectives) where as in the Khasi language we have prefixes. The following prefixes, which may be formed with verbs and adjectives to make compound words, have been omitted, except in the case where the derivatives so formed bear a special meaning from that of the radicals, e.g., jingri – cattle, jingbriew – human, nongwei – a stranger, iaroh – to praise, sngewsih – to be sorry, pynhiar – export, etc. Prefixes of the above kind are:

i – (or ih--), ia -, iai -, jing -, kaba -, la -, myn -, nang -, nong-, pyn-, sngew-,

i-, ih-, to appear to look as, ibha – to take a fancy to; isynei – to have pity for; itlot – to appear weak; i-raikhoh – to appear lean or reduced or skinny.

ia-, denotes plurality of the subject agreeing with the verb, as, ki ia kren – they are talking; u ia shoh bad u dkhar – he is fighting with a dkhar.

iai-, nang-, expresses continuity or progression of action as, iai-sneng – to continue rebuke; iai-leit – to continue to go; nangthoh – to be writing, is writing; nangiaid – to be going on, is going on.

jing-, when prefixed a verb or an adjective, to form anstract noun, as, jingbha – goodness; jingshai – light; jingle -action, and so on.

kaba-, a prefix to form a verbal noun and the imperfect participle, as, *kaba sum kaba pynkhuid ia ka met* – bathing cleanse the body.

la-, a prefix denoting future time, as, lashai – tomorrow; lashisngi – day after tomorrow; lanymwei – next season or year.

myn-, a prefix to denote a doer, as, myntuh – a thief; mynpang – a patient. In adverbs of time it denotes past time, as, mynhynne – a short time ago; mynshisngi – day before yesterday; mynnymwei – last season or year.

nong-, denotes a doer or an inhabitant of, as, nongkren – a speaker; nong Sohra – an inhabitant from Cherrapunji village; nong Mylliem – an inhabitant from Mylliem village.

pyn-, a prefix to form a causative verb, pynthoh – to cause one to write; pynshong – to cause one to sit; pynsngew – to announce.

sngew-, a prefix which means to feel, as, sngewbha – to be pleased; sngewma – to be afraid of; sngewkwah – to have a desire; sngewtlot – to feel weak.

A few suffixes have also a significance of their own, such as -duh, -kai, -sa, -sah, -sat, etc., as-

Shong-duh – to settle permanently.

Iaid-kai – to ramble, to take a stroll.

Leh-sa – to be peevish.

Thiah-sah – to remain sleeping or in bed.

Shoh-sat – to beat in such a way as to reflect cruelty on the part of the person who beats.

III.LITERATURE REVIEW AND EXISTING TECHNIQUE

As we discuss there are two main issues related to spell checker i.e. error detection and error correction. Further there are two types of errors these are non-word errors and real – word errors. Many techniques are available for non-word errors. The error detection process usually consists of checking to see if an input string is a valid index or dictionary word. Efficient techniques have been devised for detecting such types of errors. The two most known techniques are n-gram analysis and dictionary lookup. Error correction means just to replace the incorrect with most likely corrected word. Techniques available for error correction are Edit distance, Similarity keys, Rule based technique n-gram based technique, neural technique, Probabilistic technique and neural network. [Gupta, Mathur, 2012]

A. ERROR DETECTION

In error detection, the system presently compares each input word to a master list of correct words and rejects those for which it can find no match. The rejections are spelling errors. These must be corrected and reinserted on a subsequent computer run. An inspection of those items rejected because of spelling errors showed that over 80 percent fell into one of four classes of single error- one letter was wrong, or one letter was missing, or an extra letter had been inserted, or two adjacent characters had been transposed. These are the errors one would expect as a result of misreading, hitting a key twice, or letting the eye move faster than the hand. These errors can be detected using various algorithms. [Kukich, 1992]

1. N GRAM ANALYSIS

N-gram analysis is described as a method to find incorrectly spelled words in text and used for non-word errors. Instead of comparing each entire word in a text to a dictionary, just n-grams are controlled. A check is done by using an n-dimensional matrix where real n-gram frequencies are stored.

If a non-existent or rare n-gram is found the word is flagged as a misspelling, otherwise not. An n-gram is a set of consecutive characters taken from a string with a length of whatever n is set to. If n is set to one then the term used is a unigram, if n is two then the term is a Bigram, if n is three then the term is trigram. N-gram tables can take on a variety of forms but the simplest is bi-gram array which is 2-D array of size 41*41 whose element represents all possible 2-D letter combination of alphabet. The n-grams algorithms have the major advantage that they require no knowledge of the language that it is used with and so it is often called language independent or a neutral string matching algorithm. Using n-grams to calculate for example the similarity between two strings is achieved by discovering the number of unique n-grams that they share and then calculating a similarity coefficient, which is the number of the n-grams in common (intersection), divided by the total number of n-grams in the two words (union). [Gupta, Mathur, 2012]

2. DICTIONARY LOOK UP

This technique simply lookup every word in the dictionary if the word is not there then it is said to be an error. Dictionaries have their own characteristics and storage requirements. It is a straightforward task. Large dictionary might be a dictionary with most common word combined with a set of additional dictionaries for specific topics such as computer science or economy. Big dictionary also uses more space and may take longer time to search. The non-word errors can be detected as mentioned above by checking each word against a dictionary. The drawbacks of this method are difficulties in keeping such a dictionary up to date. At the same time one should keep down system response time. Too small a dictionary can give the user too many false rejections of valid words. The most common technique used for gaining fast access in dictionary is Hash tables. In order to lookup a string, one has to compute its hash address and retrieve the word stored at that address in the pre-constructed hash table. If the word stored at the hash address is different from the Input string, a misspelling is flagged. Hash tables main advantage is their random-access nature that eliminated the large number of comparisons needed to search the dictionary. The main disadvantage is the need to devise a clever hash function that avoids collisions. To store a word in the dictionary we calculate each hash function for the word and set the vector entries corresponding to the calculated values to true. To find out if a word belongs to the dictionary, you calculate the hash values for that word and look in the vector. If all entries corresponding to the values are true, then the word belongs to the dictionary, otherwise it does not.

B. ERROR CORRECTION

1. EDIT DISTANCE

Edit Distance Edit distance is a simple technique. first edit distance spelling error correction algorithm was implemented by Damerau Simplest method is based on the assumption that the person usually makes few errors if ones, i.e. errors from keyboard input therefore for each dictionary word .The minimal number of the basic editing operations (insertion, deletions, substitutions) necessary to covert a dictionary word in to the non-word As edit distance is useful for correcting errors resulting from keyboard input, since these are often of the same kind as the allowed edit operations. It is not quite as good for correcting phonetic spelling errors.

2. N-GRAM TECHNIQUE

N-grams can be used in two ways, either without a dictionary or together with a dictionary. Letter N-gram including trigram, bi-gram and unigram have been used in variety of ways in text recognition and spelling correction techniques. Used without a dictionary, n-grams are employed to find in which position in the misspelled word the error occurs. If there is a unique way to change the misspelled word so that it contains only valid n-grams, this is taken as the correction. The performance of this method is limited. It is that it is simple and does not require any dictionary. Together with a

dictionary, n-grams are used to define the distance between words, but the words are always checked against the dictionary.

Using n-grams to calculate for example the similarity between two strings is achieved by discovering the number of unique n-grams that they share and then calculating a similarity coefficient, which is the number of the n-grams in common (intersection), divided by the total number of n-grams in the two words (union) for example:

String 1 – "statistics"

String 2 – "statistical"

If n is set to 2 (Bigrams are being extracted), then the similarity of the two strings is calculated as follows.

Initially, the two strings are split into n-grams:

Statistics - st ta at ti is st ti ic cs 9 Bigrams

Statistical - st ta at ti is st ti ic ca al 10 Bigrams

Then any n-grams that are replicated within the term are removed so that only unique Bigrams remain. Repeated bigrams are removed to prevent strings that are different but contain repeated bigrams from matching. Statistics - st ta at is ti ic cs 7 unique bigrams

Statistical - st ta at ti is ic ca al 8 unique bigrams

The n-grams that are left over are then arranged alphabetically.

Statistics - at cs ic is st ta ti

Statistical - al at ca ic is st ta ti

The number of unique n-grams that the two terms share is then calculated. The shared unique bigrams is: at ic is st ta ti. In total 6 shared unique bigrams. A similarity measure is then calculated using the above-mentioned Similarity coefficient using the following formula:

A = the sum of unique n-grams in term 1.

B = the sum of unique n-grams in term 2.

C = the sum of unique n-grams appearing in term 1 and term 2.

The above example would produce the result 0.80.

(2*6) / (7+8) = 12/15 = 0.80. [Gupta, Mathur, 2012]

3. SIMILARITY KEYS

In this technique, we map every string into a key such that the similarly spelled strings will have similar key. It is known as SOUNDEX system. In this it is not necessary to directly compare misspelled string to each word in dictionary. Here, a key is assigned to each dictionary word an only dictionary keys are compared with the key computed

for the non-word. The word for which the keys computed for the non-word. The word for which the keys are most similar are selected as suggestion. Such an approach is speed effective as only the words with similar keys have to be processed with a good transformation algorithm this method can handle keyboard errors.

Table 2: Soundex Table

b, p, f, v	1
c, s, k, g, j, q, x, z	2
d, t	3
l	4
m, n	5
r	6

The algorithm:

- Ignore all characters in the string being encoded except for the English letters, A to Z.
- The first letter of the Soundex code is the first letter of the string being encoded.
- After the first letter in the string, do not encode vowels or the letters H, W and Y. These letters may affect the code by being present but are not encoded directly.
- Assign a numeric digit between one and six to all letters after the first using the table above.
- Where adjacent digits are the same, remove all but one of those digits unless a vowel, H, W or Y was found between them in the original text.
- Force the code to be four characters in length by padding with zero characters or by truncation.

4. RULED BASED TECHNIQUE

Rule-based methods are interesting. They work by having a set of rules that capture common spelling and typographic errors and applying these rules to the misspelled word. Intuitively these rules are the “inverses” of common errors. Each correct word generated by this process is taken as a correction suggestion. The rules also have probabilities, making it possible to rank the suggestions by accumulating the

probabilities for the applied rules. Edit distance can be viewed as a special case of a rule-based method with limitation on the possible rules. [Randhawa, Saroa, 2014]

5. PROBABILISTIC TECHNIQUE

They are, simply put, based on some statistical features of the language. Two common methods are transition probabilities and confusion probabilities. Transition probabilities are similar to n-grams. They give us the probability that a given letter or sequence of letters is followed by another given letter. Transition probabilities are not very useful when we have access to a dictionary or index. Given a sentence to be corrected, the system decomposes Each string in the sentence into letter n-grams and retrieves word candidates from the lexicon by comparing string n-grams with lexicon-entry n-grams. The retrieved candidates are ranked by the conditional probability of matches with the string, given character confusion probabilities. Finally, a word-bigram model and a certain algorithm are used to determine the best scoring word sequence for the sentence. They claim that the system can correct non-word errors as well as real word errors and achieves a 60.2 % error reduction rate for real OCR text.

6. NEURAL NETWORK

This method works on small dictionaries. Back propagation algorithm is used in neural network.it consist of 3 layers input, hidden and output layer. It has potential to adapt specific error pattern. In this input information is represented by on- off pattern. A=1 indicates that node is turned on and A=0 means node is turned off. For e.g. in spell checking applications a misspelling represented as binary n-gram vector may be taken as input and output pattern might be vector of m elements means number of words in lexicon. The current methods are based on back-propagation networks, using one output node for each word in the dictionary and an input node for every possible n-gram in every position of the word, where n usually is one or two.

Normally only one of the outputs should be active, indicating which dictionary words the network suggests as a correction. This method works for small (< 1000 words) dictionaries, but it does not scale well. The time requirements are too big on traditional hardware, especially in the learning phase. [Randhawa, Saroa, 2014]

IV. METHODOLOGY AND IMPLEMENTATION

A. Tokenization

Tokenization is a pre-processing step for this system. It is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing such as parsing or text mining. Tokenization is useful both in linguistics and in computer science, where it forms part of lexical analysis. Typically, tokenization occurs at the word level. However, it is sometimes difficult to define all contiguous strings of alphabetic

characters and to define what is mean by a "word‖ . Tokens are separated by whitespace characters, such as a space or line break, or by punctuation characters. In languages, such as English where words are delimited by whitespace, this approach is straightforward. [Mon, 2012]

Tokenize: - This operator splits the text of a document into a sequence of tokens. There are several options to define the splitting points. The options are as follows:

- mode: -This selects the tokenization mode. Depending on the mode, split points are chosen differently. The Range is non-letters, specify characters, regular expression and the default value is non-letters
- characters: - The incoming document will be split into tokens on each of this characters. For example, enter a '.' for splitting into sentences. The Range is string and the default value is '.:'
- expression: - This regular expression defines the splitting point. The Range is string.

Stopword Elimination: - The most common words that unlikely to help text mining such as prepositions, articles, and pro-nouns can be considered as stop words. Since every text document deals with these words which are not necessary for application of text mining. All these words are eliminated. We can choose any group of word for this purpose. It also reduces the text data and helps to improve the system performance. For e.g., “a”, “is”, “you”, “an”.

Stemming: - Stemming also known as lemmatization is a technique for the reduction of words into their stems, base or root. Many words in the English language can be reduced to their base form or stem e.g. like, liking, likely, unlike belong to like. Moreover, names can be transformed into root by removing the “s”, for e.g., During the stemming process the variation “Stem’s” in a sentence is reduced to” Stem” and this removal may lead to an incorrect stem or root. However, if these words are not used for human interaction then, these stems do not have to be a problem for the stemming process.

B. Corpus Creation

Corpus is a large and structured set of texts. It is used to do spell checker, checking occurrences or validating linguistic rules on a specific universe. Besides it is a fundamental basis of many researches in Natural Language Processing (NLP). Building of the text corpus is very helpful for the development of spell checking. Our corpus was built manually. Khasi being a language with very limited resource, we combined many few documents available online.

C. Cosine Similarity

The word-error can belong to one of the two distinct types, namely, Non-word Error and Real-word Error. In this paper, we only focus on non-word Error. If the input

sentence of the isolate word has no meaning, it is a non-word error. By real word error we mean a valid but not the intended word in the sentence, thus making the sentence syntactically or semantically ill-formed or incorrect. In both cases, the problem is to detect the erroneous word and either suggests correct alternatives or automatically replace it by the appropriate word. Since there is no research done in identifying erroneous words in Khasi, it requires analyzing features of the typo words. We have to know several features of the erroneous word so it can be referred during the development of the approach in identifying erroneous words later. Non-word errors correction is an important task. In this paper, we will focus on correcting Non-word error. [Basri, Alfred, et al, 2012]

The process in this module starts when the words from the tokenization process are going through the process of checking the input word. The system will check if the input word exists or not in the dictionary. If the word does not exist, then the input word will go through the process of identifying the correct word. Although n-gram analysis may be useful for detecting machine-generated errors such as those produced by optical-character recognizes, it has proven to be less accurate for detecting human-generated errors. Hence, most current spelling correction techniques rely on dictionary lookup for error detection. [Damerau, 1964]

There are many kinds of String Similarity Algorithm for spelling checking such as Hamming Distance, Levenshtein Distance, longest common sequence, Rules based, N-grams, probability and neural nets. Among these algorithms, Levenshtein Distance (minimum edit distance) based approaches is the most popular that required to transform one text string into another. In this paper, we apply String Cosine Similarity and corpus Lookup approach.

Cosine Similarity is widely used in text/document matching. The standard way of quantifying the similarity between two documents A and B is to compute the cosine similarity of their vector representations and measure the cosine of the angle between vectors. We return the documents ranked by the closeness of their vectors to the query. [Mon, 2012]

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Figure 1: Cosine Similarity Equation

In general, String Cosine Similarity error detection technique works by examining each word in an input string and looking it up in a precompiled table of similarity statistics to ascertain either its existence or its frequency. String Cosine Similarity technique usually requires either a dictionary or a large corpus of text in order to precompiled string similarity. If an input string exists in the Corpus, i.e., a list of acceptable word, pass to next word. If not, the string is flagged as a misspelled word.

There are subtle problems involved in the compilation of a useful Corpus for a spelling correction application. When errors cross word boundaries resulting in run-on or split words problems arise.

D. Khasi Spell Checker Aproach

The process of the Khasi spell checker is as follow:

- Accept Input sentence.
- Example: Ka sngi ba itynnad
- Tokenize the input sentence.
- Example: “Ka” “sngi” “ba” “itynnad”
- Check the isolated word by word in the corpus, If the input word matches the word in the corpus, pass to next word until the end of input sentence, but if the input word is different from the words in the corpus or not exist in the corpus, compute distance to words in corpus.
- Replace the erroneous word with the word which has the highest cosine similarity.

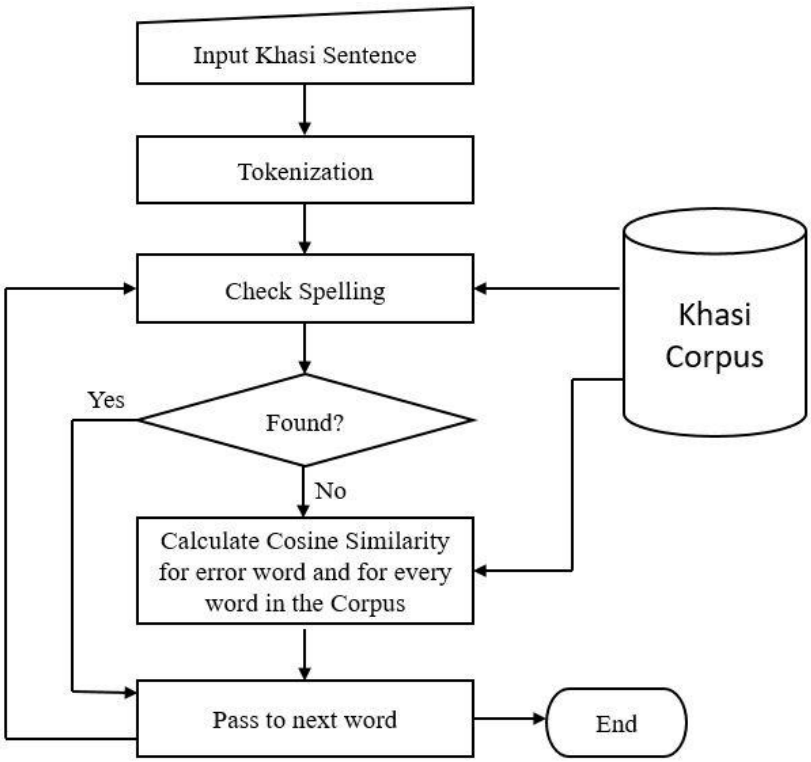

Figure 2: System Flow of the Khasi Spell Checker

V. CONCLUSION AND FUTURE WORKS

We have presented a String Cosine Similarity to improve spelling checker for targeted learners of Khasi. This work can be used to develop a spell checker, which can detect non-word errors from the input sentences. This system checks a word of Khasi Language and corrects the misspelled Khasi words. This system can correct misspelled Khasi sentences. This proposed system is very useful in applications that are needed to determine the two strings which are similar in pronunciation by typing error.

In this paper, we can only solve Typographic errors but not Content errors. We can also increase the efficiency of the spell corrector, and for this we need to create a proper dictionary with more words. We will solve Context errors and increase the efficiency of the spell corrector in the future work. In this case, we intend to employ Bayesian classifier.

REFERENCES

[1] Aye Myat Mon, "Spell Checker for Myanmar Language", IEEE, 2012.

[2] Bidyut Baran Chaudhuri, "Towards Indian Language Spell-checker Design", Proceedings of the Language Engineering Conference, 2002.

[3] Fred J. Damerau, "A Technique for Computer Detection and Correction of Spelling Errors", Communications of the ACM, Volume 7, Number 3, March, 1964.

[4] Karen Kukich, "Techniques for Automatically Correcting Words in Text", ACM Computing Surveys, Vol. 24, No. 4, December 1992

[5] Monisha Das, S. Borgohain, Juli Gogoi, "Design and Implementation of a Spell Checker for Assamese", Proceedings of the Language Engineering Conference, 2002.

[6] Neha Gupta, Pratistha Mathur, "Spell Checking Techniques in NLP: A Survey", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 12, December 2012.

[7] Sumreet Kaur Randhawa1, Charanjiv Singh Saroa, "Study of Spell Checking Techniques and Available Spell Checkers in Regional Languages: A Survey", International Journal For Technological Research In Engineering, Volume 2, Issue 3, November, 2014.

[8] Surayaini Binti Basri, Rayner Alfred, Chin Kim On, "Automatic Spell Checker for Malay Blog", IEEE International Conference on Control System, Computing and Engineering, November. 2012.