Vector distance based spelling checking system in Nepali with language-dependent heuristics

Birodh Rijal
birodh [dot] rijal [at] gmail [dot] com

Santa B. Basnet
sbasnet81 [at] gmail [dot] com

Integrated ICT, Jwagal, Lalitpur
Nepal

Abstract

In this paper, we present an n-gram based spelling correction system for Nepali. It utilizes the font unification engine, vector distance metric based on the tf-idf weighting of character n-grams and language dependent conjunct replacement list to generate candidate suggestions. These suggestions are then ranked using Levenshtein distance to provide the final list of suggestions. Finally, the system can be useful to make spelling suggestion among single byte and multi-byte encoded writing environment in Nepali.

Keywords: *n-gram, non-word error, term-weighting, heuristics, suggestions ranking*

1 Introduction

Spelling checkers detect spelling errors and suggest replacements. These are widely used in word processors, email clients and search engines. Current word processing applications check for non-word spelling errors and contextual word errors [1]. Search engines have a query processing module which uses spelling suggestions instantly and automatically to guide the user in forming correctly worded queries [2]. Automatic spelling error correction is crucial in the performance of an optical character recognition (OCR) and many natural language processing applications.

The word-level error in written text are mainly of two types, non-word errors and context dependent word errors which is sometimes called real-word errors. Non-word errors are misspellings that are invalid tokens in a particular language and can be detected by using the dictionary/lexical databases and word morphology. Non-word misspellings are typographical or cognitive errors. The misspelled token "*taht*" (correct: "*that*") is a typographic error and "*neumonia*" (correct: "*pneumonia*") is a cognitive error in English. The context dependent error occurs when the intended word is a valid lexicon but appear because of the mishandling of word in its context.

In case of Nepali language, some characters are in the conjunct (dependent) form and appears with another characters. [3] Half-form of characters constructed using viram or halanta have multiple orthography in the writing system. Misspellings in Nepali are mostly due to character conjuncts and their orthography in Devanagari writing. These errors are generated in a computer due to inconsistencies such as character orthography and character composition formed by different types coding system and multiple codes for the same character conjuncts [4]. Non-word error such as "निति", "नीती" and "निती" are invalid representations of the word "नीति" (meaning "regulation") characterized by wrong use of the dependent vowels i.e. *hraswa* and *deergha*. The character conjunct "क्त" is constructed from the characters [क + ् + त] and not from "त्त" which has a similar orthography. Most Nepali people use glyph based single byte encoding of characters that produces non-standard conjuncts which does not have relationship with the language representation. But the words "फुलहरू" and "फूलहरू" (meaning eggs and flowers respectively) can be easily misspelled. Corrections to such an error requires decision based on the context surrounding the misspelling. These examples show there are lots of possibilities to incorrectly use dependent and free forms of characters, special symbols to produce multiple sounds and shapes that brings error in Nepali Devanagari writing. Detection of context dependent spelling errors has to be approached from a natural language processing perspective. It is currently in the research phase.

2 Background of spelling checking system

Nepali is a highly inflectional language. The vocabulary consists of a large number of words

which are in derived and inflectional forms. In some cases, orthographic changes occur during morphological operations [5] on the head words. This produces spelling variations and generates considerably large set of word instances in the language. Usage of the vocabulary constructed in such a way can produce non-word as well as context dependent spelling errors. Bal et. al [6] developed a set of affix rules to generate word variations in order to be used by the morphological-based spellchecker. These rules are later integrated in OpenOffice.org[1] application as a Nepali spelling checker with Hunspell[2] in the background. Hunspell was originally developed for the Hungarian language. This system supports multi-byte Unicode encoding which makes seamless integration of Nepali dictionary and the affixes rules possible. The system is limited to non-word spelling errors and greatly suffers from the lack of word coverage and non-usable word generations. It also does not support spelling correction in a single-byte true type font encoding system which is the major source of text generation in Nepal. Microsoft's MS-Office[3] has its own Nepali spelling checking system implemented as a language pack. This is however only supported in early editions of the Office package. A team of enthusiasts have developed a few number of orthographic rules for Nepali spell checker[4].

Several approaches have been developed to address the non-word spelling errors. The vector distance on *n*-gram statistics provides the best result [1]. Similar technique is used by Sai K. et al [7] in the generation of list of candidate suggestions. Final ranking of the suggested words is done by a combined Longest Common Substring (LCS) similarity metric. The aim of LCS is to address the understanding that common misspellings are at the middle of the words.

Minimum edit distance measure is used to select a candidate to replace the misspelt word [8, 9]. The edit distances of two words measures the difference in terms of edit operations. Three single character operations are considered, viz., insertion, deletion and substitution. A dynamic programming technique is implemented to speed-up the calculation. Chaudhuri [10] maps phonetically similar character symbols to a single symbol which generates a near-phonetic dictionary. This helps to detect and correct phonetic errors in Bangla writing. Sasu [11] proposes a spelling correction method using two components. The first one is the edit distance function. The second component relies on the Bayesian decision theory that utilizes prior probability of a candidate word, that is to say, it favors the most frequent words.

Our work is on spelling checking for non-word errors for Nepali. The system relies on two types of ranking measures. First, the language specific heuristics which reports the errors among special symbols, independent and dependents forms of characters made during Nepali text writing. Second, the vector distance between *n*-grams is employed to list out the promising replacements. These candidates are then ranked using the Levenshtein distance[5] measure. Finally, the top ranked words are given as the replacement for a misspelling.

3 Nepali spelling checking system

Nepali spelling system analyzes Nepali Devanagari text from the font unification system. [4] It supports both single byte and multi-byte fonts encoding system. It focuses on non-word error detection in Nepali. The word "*विद्युतीय*" (meaning: electric) in Nepali is found with various misspellings in a corpus such as "*विद्युतिय*", "*बिद्युतीय*", " *बिद्युतिय*", "*बिधुतीय*", "*विधुतीय*" and "*विधुतिय*". Here, the variation of the non-words captures major types of writing error in Nepali. Especially, the character "*ब*" and "*व*", "*द्य*" and "*ध*" are similar in orthography and pronunciation. Another source of confusion comes from the construction of character conjunct using dependent character like "*ती*" and "*ति*", known as *hraswa* and *deergha*. A large source of errors is in the use of symbols like halanta (◌्), candrabindu (◌ँ), cirabindu (◌ं) and visarga (◌ः). For example, the character "र" can join with halanta (◌्) and then attach with other free form characters such as "*र्य*", "*र्च*", "*र्ट्र*", "*र्ग्र*", "*र्श्र*". It can also exist as the half form "*र्*". Hence, Nepali writing has a distinct behavior and rich in character conjuncts formation rules as

[1] https://www.openoffice.org/
[2] https://en.wikipedia.org/wiki/Hunspell
[3] https://en.wikipedia.org/wiki/Microsoft_Office
[4] http://www.nepalilanguage.org/spellcheck/
5 https://en.wikipedia.org/wiki/Levenshtein_distance

compared to English (roman script). This work captures the non-word Nepali spelling errors using the following components.

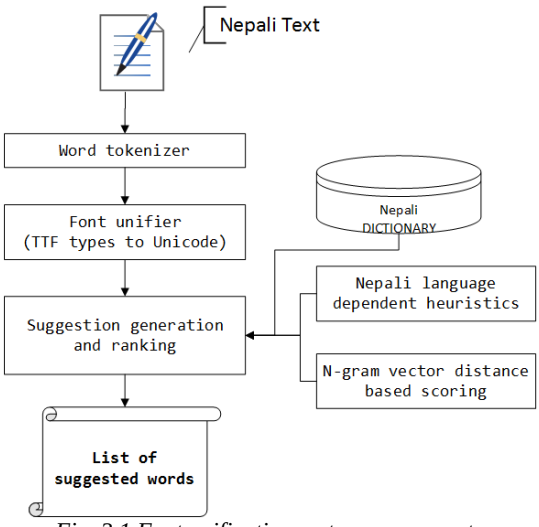

Fig. 3.1 Font unification system components.

3.1 Word tokenizer

Generally, the spelling system uses white-space characters like blank space, tab space and carriage returns to identify word boundary. This simple strategy suffers from the word boundary discovery problem across languages. Nepali words like "*खानेहरू*" could be written as "*खाने हरू*" which is an error. The word "*किनभने*" could also be written as "*किन भने*" and is of the context dependent error type. Our system is therefore unable to detect multi-word spelling errors or split of words errors.

3.2 Font unifier

We embed the font unification system [4] to preserve the writing integrity in the single byte and multi-byte encoded Nepali text. The unification system uses TTF and Unicode characters groupings to form minimal unification units. These units are then converted either from TTF to Unicode or from Unicode to TTF for spell checking. The conversation module uses font mapping tables, glyphs re-arrangement rules and implements run-time cache to preserve the integrity of written Nepali text. The run-time cache is used to hold many to one mapping of characters that are possible from the font map table. This cache exists for the whole conversion cycle i.e. TTF → Unicode → TTF while making a list of suggestions. The conversion accuracy is almost *100* percent which benefits the Natural Language Processing (NLP) applications like the spell checking. Our system integrates font unifier as an integral component so that many publications and text editor users can use spell checker in their own writing environments.

3.3 Spelling suggestion generations

We detect non-word spelling errors by using dictionary lookup. If the test word is absent from the dictionary, it constitutes an error. It is flagged by using red colored underlines. The system then generates spelling suggestions for the flagged words. The suggestions are generated from the words in the dictionary with Nepali language dependent heuristics and the word similarity measures.

3.3.1 Nepali spelling dictionary

Building a word level dictionary takes tremendous amount of manual effort. Often, the dictionaries themselves are inadequate for compiling the lexicons. An appropriately sized word collection is necessary to make the spelling system usable in practice. Words with few characters can create a lot of false word errors. Nepali has a rich availability of inflections and derivations of words. The attachment of Case Marker (CM), Post Positions (PP) and the composition of them makes the dictionary size significantly large which has been addressed by Prasain [5]. During dictionary construction, we use handcrafted rules [5, 12] to generate word variations. The head words are taken from the Nepali Brihat Sabdakosh published by the Nepal Academy[6].

The rules suffer from over-generation of non-words i.e. rarely used words. We filter out these unused words with the help of a Nepali text corpus of size one million documents. The documents have been collected from different news sources and online platforms over the last decades. Let the word collections in dictionary D is filtered from Nepali Corpus N and generated words G is given by the set difference:

$$D = N - G \quad \ldots (3.1)$$

[6] http://nepalacademy.org.np

Significant amount of human effort is used to correct all the words produced by equation (3.1). We again add the non-words in the Nepali Corpus by manually correcting them aiming to cover the word compounding which are in regular use. We realize that we are still undersized in word coverage due to the post positions attachments. To handle this problem, we devise a count based heuristics to identify whether the given attachment of a particular post position is valid or not. If a word appears with attached PP then we check if the same word can have other PP attached to it or not. For our purpose, we enforce a limit that minimum of *two* post positions should be attached with the same word in the corpus and that the word should be of minimum length *three*. This thresholding helps to filter out the over-generation of non-word during spelling check. The accepted non-word PP attachment $P_{attached}$ is given by the rule:

$$P_{attached} = \begin{cases} Accept\ if\ C(W') \geq 2\ and\ Len(W) \geq 3 \\ Reject, Otherwise \end{cases} \quad \ldots (3.2)$$

where *W'* are the valid word instances with other PPs attached in the dictionary and the *W* is the stem obtained by detaching the current PP. Finally, we build the dictionary of *750K* word variations appeared in around *52K* entries from the Nepali Brihat Sabdakosh. Besides this, we have added some name entities to make richer word collections.

3.3.2 Heuristics based generations

In the generation of relevant suggestions, we prepare a list of possible character conjuncts and special symbols that are frequently the source of confusion while writing Nepali text. We reflect this observation on the basis of the corpus collected from the news and online sources. For example, people frequently makes spelling mistakes using ["ं"] in places of ["ँ", "न्", "ङ्", "म्"], ["ाे", "ेा"] in place of ["ो"] and ["ए"] in places of ["ऐ", "य"]. We managed to prepare such *40* conjunct replacement entries for our purpose. We generate the suggestion candidates using the conjunct replacement entries in the first pass. These are then verified by looking up in the spelling dictionary. Validated words receive higher score and occupy upper positions in the list.

3.3.3 Vector distance *n*-gram based generation

We assume that people tend to make more spelling mistakes in longer words than the frequent shorter words, i.e. the possibility of non-word errors can happen more in a word containing high number of conjuncts. Our system considers a candidate suggestion if it is an advisable or not from the dictionary words such that it should have at least of similarity threshold of *0.65* based on the Levenstein Distance measure to the wrong word. Initially, we store the *n*-gram representation of all word variations from the dictionary entries. The following figure *3.1* shows the *n*-gram entries for the word "*विद्युतीय*".

[*ि्द्, द्य, तीय, युत, ्यु, ुती, विद, विद्, ुतीय, द्यु, ्युत, ि्द्य, युती, विद्युतीय*]

Fig. 3.1 n-gram representation of a word "विद्युतीय" in the Nepali spelling system.

We chose word length based *n*-gram generation criteria for effective retrieval of relevant words. For example, lengthy word could easily get huge numbers [$O(n^2)$] of gram segments. Single length grams make no-sense during suggestion generation because they tend to occur in many word instances and adds noise during statistical calculation. In our case, the following *table 3.1* represents the word-length based strategy of *n*-gram generation during indexing.

Word Length	***=5***	***>5***	***<5***
Min-Gram Size	2	3	1
Max-Gram Size	3	4	2

Table 3.1: n-gram generation criteria for a word

All the grams of the dictionary entries are indexed using inverted-index[7] posting which gives better search performance upon the calculation of candidate suggestion. We ignore words of length *1* and *2* during indexing. We measure the word similarity for candidate suggestions using vector space model (VSM) [13]. The cosine similarity which is based on the deviation angle between the error word and the dictionary suggestion is calculated using the following relation *3.3*.

[7] https://en.wikipedia.org/wiki/Inverted_index

$$\theta = \cos^{-1} \frac{w_k \,.\, w_q}{|w_k| \,.\, |w_q|} \ldots (3.3)$$

Here, w_k is the vector for k^{th} word entry in the dictionary and w_q is the query vector constructed from *n*-gram. The value of $w_k \,.\, w_q$ is the dot product of two vectors. The $|w_k|$ and $|w_q|$ are the norms of the vectors which is calculated by the following relation *3.4*.

$$|w_q| = \sqrt{\sum_{i=1}^{n} (w_{q_i})^2} \ldots (3.4)$$

The geometric interpretation of the vector space model measures the angle between the collections of vectors; here $\mathbf{w_1}$ and $\mathbf{w_2}$ to the query vector $\mathbf{w_q}$. This is demonstrated by the figure *3.2*.

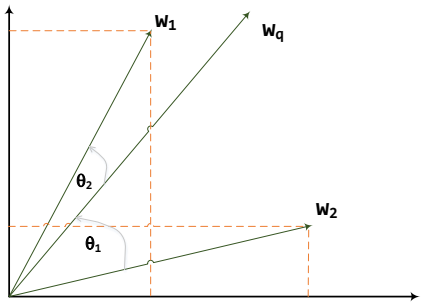

Fig. 3.2: Vector space model representation of words $\boldsymbol{w_1}$, $\boldsymbol{w_2}$ *and the query word* $\boldsymbol{w_q}$

Now, we consider all the components of the vectors that have non-negative value estimated by the *tf-idf* weights. This quantity estimates significance of a gram with respect to the words in the data dictionary. The term frequency (*tf*) is the number of occurrences of the gram segment through-out the word-dictionary entries. The inverse document frequency (*idf*) is the number of words that contains the gram segment. In the *tf-idf* weighting scheme, the weighted vector for a word w_k is given by,

$$\overrightarrow{w_k} = [g_{1,k}, g_{2,k}, g_{3,k}, \ldots, g_{N,k}]^T \ldots (3.5)$$

where $g_{i,w} = tf_{i,w} \times log \frac{|W|}{|\{w' \in D | g \in w'\}|}$.

Finally, the similarity of the word w_j with the given error word query w_q having $[1, n]$ grams g is calculated using the following relation.

$$Similarity\,(w_j, w_q) = \frac{\sum_{i=1}^{n} g_{i,j} g_{i,q}}{\sqrt{\sum_{i=1}^{n} g_{i,j}^2} \sqrt{\sum_{i=1}^{n} g_{i,q}^2}} \ldots (3.6)$$

3.4 Suggestions ranking

We rank the suggested words from candidates in the priority queue using two steps. The first step is based on human intuition which is language dependent and the score is assigned manually. The valid words obtained by the conjunct replacements of section *3.3.2* are of this kind and are assigned high score, in our case the default value *1.0*. In the second step, we score suggested words using Levenshtein distance measure and words with high scores appears at the top. Finally, we select the first *k* number of words from the ranked queue. In our spelling suggestion system, the *k* is a configurable parameter which we set to *12* by default.

4 Performance analysis

We create a list of erroneous words by randomly selecting them from the collected Nepali texts. The respective suggested word list is labelled manually for the performance evaluation of the Nepali spelling system.

4.1 Test data preparation and criteria

We prepared the dataset based on the text gathered from news article published in Nepali online sites and divided them into two categories. The first one is the set of correctly spelled words and the second is the misspelled words. We have chosen *2552* words uniformly from the set of *500K* isolated misspelled words for the purpose of system evaluation. The length of test words ranges from *2* to *23* characters. These words covers all the characters and conjuncts that can appear in Nepali writing in Devanagari.

4.2 System evaluation

At first, we checked the spelling suggestion using our system for each of the misspelled words in the test dataset. We nominated the correctly spelled replacement word from the list of suggestions. We then evaluated our system against the number of relevant suggestions with single and multiple edit errors. We measure the accuracy in percentage as is given by the following relation.

$$Accuray = \frac{Correctely\ Suggested\ Words}{Total\ Test\ Words} \ldots (4.1)$$

The evaluation of non-relevant suggestions of the system is given in table 4.1.

Evaluation model	*Accuracy*
Baseline model (without language dependent heuristics)	51.74 %
Improved with heuristics	**92.32** %

Table 4.1 Summary result of accuracy measure

The output of our baseline system, with the vector distance based measure for suggesting candidates, achieves an accuracy of 51.74%. Addition of language dependent heuristics improves the accuracy by 41%. It is frequently observed that many special characters have multiple edit values and are dominated by the suggestions with single edit value. There are three causes behind not getting the correct suggestions. First, the words of count = 184 have non-word errors which are poorly spelled, that is, the dictionary does not contain any of the relevant words which match the threshold of 0.65 for the Levenshtein distance measure. Second, some words with count = 10 appear as the English language words written in Devanagari. Finally, few words with count = 2 do not have a correctly matched entry in the dictionary.

5 Conclusion and Future works

In this paper we presented a working vector distance based spelling checking system for Nepali language. The system achieved significantly large number of word coverage to check for non-word errors. The language dependent heuristics improves the quality of relevant suggestions greatly during error correction. To our knowledge this is the first Nepali spell checker to use count based statistics of characters n-grams. The system achieves an impressive accuracy of *92.32%*. Furthermore, the word coverage has a considerable advantage over systems with word generation using hand crafted rules. To our knowledge, it is the first system to use the statistics of character's *n*-gram that combines with Nepali language dependent heuristics to build a spelling checking system. The spelling system has support for both TTF and Unicode writing system in Nepali which is missing from existing tools.

There are limitations and immediate extensions. The system does not address the case of word split which generates expressions such as "झापा तिरको" and "झापातिर को". The two tokens in each expression has to be combined for a correct spelling. A solution is to generate information about word bi-grams and tri-grams. We also need to extend the system to correct context based spelling errors such as "कुखुराले फूलहरू कोरल्यो।" (meaning: hen hatched the eggs.) We are working to develop a machine learning tool to learn word's concept from the text corpus. These two problems have been set as our future work.

6 References

[1] Kukich K., Techniques for Automatically Correcting Words in Text, *ACM Computing Surveys*, Volume 24 (1992).

[2] F. Ahmed, E. William De Luca, A. Nürnberger, Revised *N*-Gram based Automatic Spelling Correction Tool to Improve Retrieval Effectiveness, *Polytechnic Open Library International Bulletin of Information Technology and Science*, ISSN 2395-8618 (2009).

[3] Bal B. K., Structure of Nepali Grammar, *Working papers*, Madan Puraskar Pustakalaya, Nepal (2007)

[4] Singh T., Basnet S., Unification of fonts encoding system of Devanagari writing in Nepali, *Submitted to 37th annual conference of Linguistic Society of Nepal*, Nepalese Linguistics (2017)

[5] Prasain B., A computational analysis of Nepali Morphology, *A dissertation work of Doctor of Philosophy in Linguistics*, Tribhuvan University, Nepal (2011)

[6] Bal B. K., Pandey B., Khatiwada L., Rupakheti P., Nepali Spell Checker, *Research Report on the Nepali Spell Checker*, PAN/L10n/PhaseII/Reports (2008)

[7] Sai K., Pingali P., Varma V., An information retrieval approach to spelling suggestion, *WWW 2010 Poster*, Raleigh, NC, USA (2010)

[8] Damerau F.J., A technique for computer detection and correction of spelling errors, *Communications of ACM*, Vol. 7, No. 3, pp. 171-176 (1964)

[9] Levenshtein V.I., Binary codes capable of correcting deletions, insertions and reversal, Soviet Physics Doklady, Vol. 10, pp.707, (1966)

[10] Chaudhuri B. B., Reversed word dictionary and phonetically similar word grouping based spell-checker to Bangla text, *Computer Vision and Pattern Recognition Unit*, Indian Statistical Institute, 203, Kolkata 700035, India

[11] Sasu L., A probabilistic model for spelling correction, *Bulletin of the Transilvania*, University of Brasov, Vol. 4(53) pp. 141-146 (2011)

[12] Basnet S., Pandey S., Morphological analysis of verbs in Nepali, *30th Annual conference of Linguistic Society of Nepal*, Vol. 24, pp. 21-30 (2009)

[13] Salton G., Buckley C., Term-weighting approaches in automatic text retrieval, *Information Processing & Management*, Vol. 24, Issue 5, pp.513-523, (1988)