A Light Weight Stemmer for Bengali and Its Use in Spelling Checker

Md. Zahurul Islam, Md. Nizam Uddin and Mumit Khan
Center for Research on Bangla Language Processing, BRAC University, Dhaka, Bangladesh
zahurul@bracu.ac.bd, nizam02201001@gmail.com, mumit@bracu.ac.bd

Abstract

Stemming is an operation that splits a word into the constituent root part and affix without doing complete morphological analysis. It is used to improve the performance of spelling checkers and information retrieval applications, where morphological analysi would be too computationally expensive. For spelling checkers specifically, using stemming may drastically reduce the dictionary size, often a bottleneck for mobile and embedded devices. This paper presents a computationally inexpensive stemming algorithm for Bengali, which handles suffix removal in a domain independent way. The evaluation of the proposed algorithm in a Bengali spelling checker indicates that it can be effectively used in information retrieval applications in general.

1. Introduction

Stemming is a process by which a word is split into its stem and affix [1]. Terms with common stems tend to have similar meaning, which makes stemming an attractive option to increase the performance of spelling checkers and other information retrieval applications. Another advantage of stemming is that it can drastically reduce the dictionary sized used in various NLP applications, especially for highly inflected languages.

The design of stemmers is language specific, and requires some to significant linguistic expertise in the language, as well as the understanding of the needs for a spelling checker for that language [2]. Consequently, a stemmer's performance and effectiveness in applications such as spelling checker vary across languages. A typical simple stemmer algorithm involves removing suffixes using a list of frequent suffixes, while a more complex one would use morphological knowledge to derive a stem from the words. The various stemming algorithms have been evaluated in various applications from spelling checker to information retrieval [1, 2], and the results show that stemming appears to be more effective in such applications for highly inflected languages [3, 4].

There has been no published effort to develop a stemming algorithm for Bengali. In this paper, we present a lightweight stemmer for Bengali that strips the suffixes using a predefined suffix list, on a "longest match" basis, using the algorithm similar to that for Hindi [19]. The proposed stemmer, as is the case with [19], is both computationally inexpensive and domain independent. We review the existing work in this area in Section 2; then we present the proposed stemming algorithm in Section 3, followed by its application and performance in a spelling checker in Sections 4-5 and evaluation in section 6. Finally, we conclude with a look at future research directions.

2. Related work

Martin Porter developed the "Porter Stemmer", which is a conflation stemmer, in 1980 at the University of Cambridge [5]. The Porter Stemmer uses the fact that English language suffixes are mostly a combination of smaller and simpler suffixes. Porter designed a rule-based stemmer with five steps, each of which applies a set of rules [6]. There are a number of other stemming algorithms for English such as Paice/Husk [7], Lovins Stemming [8], Dawson [9], and Krovetz [10]. Among these, the Porter Stemmer is the most prevalent one, it and has been applied to languages other than English. Stemming algorithms for spelling checkers and other information retrieval applications have been developed for a wide range of languages including Malay [11], Latin [12], Indonesian [13], Swedish [14], Dutch [15], German [16], French [17], Slovene [4], and Turkish [18]. The stemming work for Hindi, a sibling of Bengali, includes an evaluation of its performance by computing the under-stemming and the over-stemming statistics for corpus of documents [19]. To the best of the authors' knowledge, this work represents the first published effort to develop a stemmer for Bengali.

There are a few spelling checkers that are available for Bengali language. Puspa speller [20, 21, and 22] is a phonetic spelling checker, whereas Bspeller [23] is based on aspell [24], and bundled with Bengali Linux distribution. Additionally, there are a few more Bengali spelling checkers available that do not document the methodology and distributed as closed source application.

3. Stemming algorithm for Bengali

Bengali is a highly inflected language with relatively free or pragmatically free word order. All Bengali verbs are inflected forms of verb roots; in addition, a significant number of nouns and a few adjectives can be inflected as well.

3.1. Noun inflection

Bengali nouns are inflected for case, including nominative, objective, genitive (possessive), and locative. The case marking pattern for each noun being inflected depends on the noun's degree of animacy. When a definite article such as -□□ (singular) or -□□□□ (plural) is added, as in the Tables (1 and 2) below, nouns are also inflected for number. [30]

Table 1: Singular noun inflections

	Animate	*Inanimate*
Nominative	□□□□□□□	□□□□□□
Objective	□□□□□□□□ □	□□□□□□
Genitive	□□□□□□□□	□□□□□□র
Locative		□□□□□□□□

Table 2: Plural noun inflections

	Animate	*Inanimate*
Nominative	□□□□□□□	□□□□□□□ □
Objective	□□□□□□□□ (□□)	□□□□□□□ □
Genitive	□□□□□□□□	□□□□□□□ □□
Locative		□□□□□□□ □□□

3.2. Verb inflection

Bengali verbs are either finite or non-finite. Non-finite verbs are not inflected for tense or person; finite verbs are fully inflected for person (first, second, third), tense (present, past. future) tense, aspect (simple, perfect, progressive), and honor

Table 3: Verb inflections

	Present				**Past**				**Future**	
Verb Root	**Simple**	**Continuous**	**perfect**	**Subjunctive**	**Simple**	**Habitual**	**Continuous**	**Perfect**	**Simple**	**Subjunctive**
কর (1st person)	□□□	□□□□	□□□□ □		□□□ □□	□□□□ ম	□□□□ □□□	□□□□ □□□□	করব	
বল (2nd Person)	বল	বলছ	□□□□	□□□	□□□ □	□□□□	□□□□ □□	□□□□ □□□	□□□□	□□□
□□□□ □ (3rd person)	□□□ □□□	□□□□ □□□□ □	□□□□ □□□	□□□ □□□	□□□ □□□	□□□□ □□	□□□□ □□□□ □□	□□□□ □□□□	□□□□ □□□	□□□ □□□

(intimate, familiar, and formal), but not for number. Each inflection is indicated by a suffix. Additionally, the suffixes indicating tense and aspect can be replaced by conditional, imperative, and other special inflections [30]. The number of inflections on many verb roots can total more than 450. A few examples of Bengali verb inflexion are given below in Table 3.

3.3. Adjective inflection

In Bengali, adjectives are rarely inflected for the gender, number or person of the nouns or pronouns they qualify. A few adjectives can be inflected to denote the female gender (e.g., □□□□□□ -> □□□□□□□, □□□□□□□□□ -> □□□□□□□□□□) but these can be considered sanskritisms rater than general phenomena.

3.4. Algorithm

A word may contain suffixes or prefixes. We should note that our stemming algorithm only strips suffixes from words. In the algorithm, a suffix is not necessarily the shortest possible one as one would expect, but may contain other suffixes as well. For example, the word "□□□□□□□□□□" consists of four suffixes: □+□□+□□+□□ম . In this paper, however, we consider □□□□□□□□ as one suffix as done in the Hindi stemmer [19]; though grammatically not a suffix, it is computationally easier for stripping from the stems. We have found 72 suffixes for verbs, 22 for nouns, and just 8 for adjectives for Bengali language. We also order the suffixes according to the length so that the longest suffix will be at the top of the list. We are always finding which suffix from suffix list matches with given word from right. If matches found then remove the given word to stem and root. We have tried to strip the larger suffix first then smaller and so on. There are few cases we can't get actual root after stemming. For an example Bengali verb □□□□□□□□□ the actual root is □□ but we will get □□□□, this type can be solved after we get the stem. Table 4 shows a few results of the Bengali stemmer. The first four entries are Bengali nouns, the fifth entry is an adjective and the last five entries are verbs.

Table 4: Result of our stemming algorithm

Word	*Stem*	*Suffix*
□□□□□	কলম	□□
□□□□□□□	□□□□	□□□
□□□□□	□□□	সব
□□□□□□□□	□□□□□□	□□
□□□□□□□	□□□□□□	□
□□□□□□□	কর	□□□□□
□□□□□□□□□□	□□□□	□□□□□□
□□□□□	□□□□	ব
□□□□□□□□□	□□□□□	□□□□
□□□□□□□□	□□□□□	□□□

4. Spell check using stemming

A Bengali spelling checker is an essential component of many of the common desktop applications such as word processors as well as the more exotic applications, such as a machine language translator. One particular challenge facing the development of a usable spelling checker for Bengali is most Bengali words are inflected, and follow complex orthographic rules, in part a result of the large gap between the spelling and pronunciation of a word [2]. Though there are many spelling checkers available for Bengali but biggest trade of these spelling checkers is they can't handle inflection related word.

In the following sections we will describe the steps in the process of checking the spelling of a word on our spelling checker using stemming:

(a) Detect whether it is misspelled or not,

(b) Generate suggestions if it is misspelled, and

Lastly, we show the performance and evaluation of our stemmer in spelling checking.

4.1. Error types and detection

To give suggestions for a misspelled word, the first step for a spelling checker is to detect the misspelled word. But before detecting a misspelled word, we need to know what a misspelled word is. Misspelled words or errors can be of many types, such as typographical error (e.g., misspelling 'spell' as '*speel*'), cognitive error (e.g., misspelling 'separate' as '*seperate*'), etc. Damerau [14] finds that 80% [25, 26] of all misspelled words (non-word errors) in a sample of human keypunched text were caused by single error misspellings, i.e., any of the following errors for Bengali words:

a) Deletion. For example: mistyping □□□□□□□ as □□□□□□
b) Insertion. For example: mistyping □□□□□□□□ *as* □□□□□□□□□
c) Substitution. For example: mistyping □□□ as □□□
d) Transposition. For example: mistyping চমক as চকম

To detect an error first look up at root words dictionary. If not found then tries to find out the stem if not this is erroneous string.

4.2. Suggestion generation

We already discussed different types of error that may occur in Bengali word. But, in a spelling checker using stemming these errors may occur in stem and suffix portion. Here are some explanations with some examples of the operations:

4.2.1. Suggestion Generation of Deletion Errors. User may forget to type one character in a word. Suppose user made mistake and the given word is “পরিচাক” which is miss spelled. The spelling checker will make it “পরিচালক” by inserting character “ল”. For the word “পরিচাক” it will insert all the Bengali letters one by one in all possible position in the suffix and finds the best match and finally if it makes a valid word, it just added to the suggestion. In that case the correct word will be top of the suggestion list. We used edit-distance algorithm to find the best match. User may also make mistake in stem portion. We will get suffix and stem (misspelled) after stemming and then we can find the correct stem form stem dictionary and combine with suffix and add to suggestion list. In this case the suggestion is “পরিচালক”. It just added to the suggestion.

4.2.2. Suggestion Generation of Insertion Errors. User may type a word contains an extra character in any position. In case of error in stem portion (e.g.: □□□□□□□□□) we will get □□□□ as stem which is not available in stem dictionary. The spelling checker deletes one character in different position at a time and finds the stem in dictionary. Suppose error in suffix portion (e.g.: □□□□□□□□□). Then there will be no suffix after stemming. The spelling checker deletes the "ত ”and give "□□□□□□□□" which is a valid word.

4.2.3. Suggestion Generation of Substitution Errors. User may type wrong character in any position of a word. Suppose there is a word “□□□” which is miss spelled. But there are two valid word “□□□” and “□□□”. Here the spelling checker delete “ড ” and find those word by replacing “ড ” by “ড় ” and “ড় ”. To do this, the spelling checker deletes a character at a time, replace it by all the character in Bengali and try to match a valid word from the lexicon. If the word matches any valid word it just adds to the suggestion.

4.2.4 Suggestion Generation of Transposition Errors. The Interchange character takes place when the characters are right but not in it's own position. Suppose there is a word “চকম” which is miss spelled. The spelling checker makes it “চমক” by swapping “ক” and “ম”. It is done by swapping all possible pair of characters from their position.

5. The spelling checker algorithm using stemmer

First the spelling checker checks the given word with a lexicon containing only the root words. If the word is found, then it is a valid word, terminating the checking process. For example, if the given word is “ডাক”, the algorithm finds it in the lexicon, and thus terminates.

If the word is not found in the lexicon, we apply the stemming algorithm. There are two possible scenarios: the stemming algorithm finds and returns a stem, or it cannot find a possible suffix. Let us suppose that we find the stem. Now we check the stem from the lexicon. Process ends if stem found suppose the given word is “ডাকতাম”. We get the stem “ডাক" from the stemmer. Now after checking it from the lexicon we find that it is a valid stem. So “ডাকতাম ” is a valid word. If, on the other hand, the stem is not valid, then we try to produce a list of suggestions using the suggestion generation process. If we get some suggestion then output them with their suffix. Suppose the word is “গাকতাম”. The stemmer removes “তাম ”and gives the stem “গাক”. “গাক” is not a valid word and not found in the lexicon. So the spelling checker tries to generate some suggestion. The suggestions will be ডাক, গাড়, গাদ, গাল. So the outputs will be ডাকতাম, গাড়তাম, গাদতাম, গালতাম.

Now if we do not have any stem from the stemming method then we do not know if the given word contains error in the suffix portion or the stem portion. First we assume that it is a miss spelled root. So we try to get some suggestion from the suggestion generation process. If we get any suggestion then the given word is misspelled and we give it to output. Now if we do not have any suggestion then we try to get probable stem list with their suffixes from modified stemming method. Modified stemming process end if, no suffix found. Here the word is a miss spelled word and the spelling checker cannot provide any suggestion. Now if the modified stemming method can provide some suggestion (stem and suffix list). We check each stem from the lexicon. If valid then give output. Suppose the word is "ডাকতাব". The modified stemmer returns two pair of stem and suffix. (Stem: ডাকতা Suffix: ব) and (stem: ডাক Suffix: তাম). We the first pair we cannot find any thing. We output second pair because "ডাক" is a valid root.

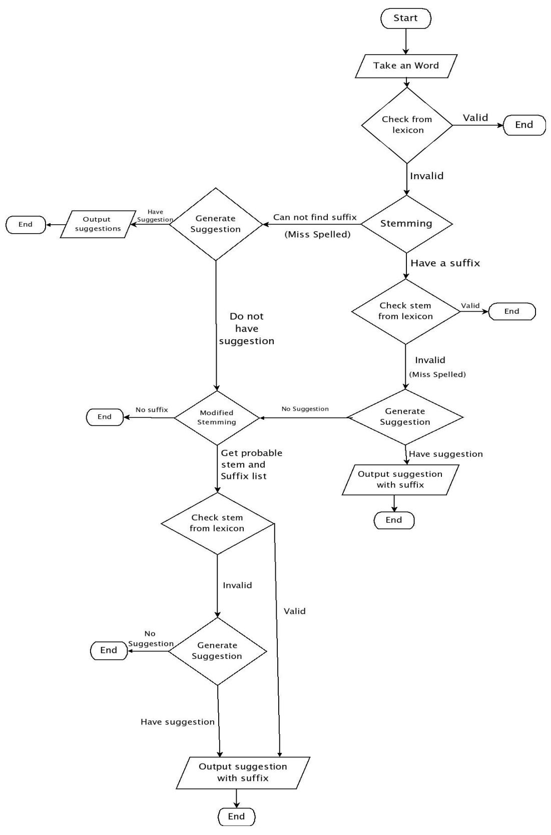

Fig 1: Flow chart of Spelling Check Algorithm

Otherwise we try to get suggestion from the suggestion generation process. If any suggestion is found then give it output with the corresponding stem.

The flow charts of Spelling checker are given on Figure 1.

6. Evaluation

We noted certain parameters that should be considered during the evaluation of spelling checkers for isolated-word error correction [28]. These are:

- lexicon size,
- test set size,
- correction accuracy for single error misspellings,
- correction accuracy for multi-error misspellings, and
- type of errors handled (phonetic, typographical, OCR generated etc.);

Another paper on Bengali spelling checker [29] also considers these parameters for evaluation of spelling checkers. We are also considering these parameters to evaluate our spelling checker.

Lexicon size: Our lexicon contains 600 root word and 100 suffixes.
Test set size: We tested our spelling checker with 13,000 words that list the most common single and multiple word mistakes in Bengali [19].
Correction accuracy for single error misspellings: 90.8%.
Correction accuracy for multi-error misspellings: Not good as single error correction, it's close to 67%. But it does over generate few suggestions.
Type of errors handled (phonetic, typographical, OCR generated etc.): we are not considering these.

7. Conclusion

We present a light weight stemmer for Bengali, and show its application and evaluation in a spelling checker. A modified stemmer customized for a spelling checker application showed significant improvement in the spelling checker's performance. This leads us to believe this stemming algorithm will prove to be beneficial for other information retrieval applications for Bengali. We should note that the proposed stemming algorithm is primarily for handling inflections – it does not handle derivational suffixes, for which one would need a proper morphological analyzer. Reducing derivationally related terms to the same stem would lead to over-conflation in some cases, potentially affecting the precision of information retrieval applications, other than spelling checkers.

8. Direction for Future Work

The proposed algorithm has been evaluated only using a spelling checker, but not with other information retrieval applications search as a search engine. More extensive evaluations will provide the statistical information needed to manage the suffix list, which in turn will determine the tradeoff between under-stemming and over-stemming. An obvious enhancement is to handle prefixes as well. Bengali has a small number of prefixes, which also happen to be frequently used. It is quite reasonable to extend the algorithm to support these prefixes. It would also be instructive to apply this algorithm to Bengali's sister languages such as Assamese and Oriya.

9. Acknowledgement

This work has been supported in part by the PAN Localization Project (www.panl10n.net), grant from the International Development Research Center, Ottawa, Canada, administrated through Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, Pakistan. We would also like to thank Arnab Zaheen, Kamrul Hayder, Naira Khan and other members of our research group.

10. References

[1] W. Frakes and R. Baeza-Yates, eds, "*Information Retrieval: Data Structures and Algorithms*", Prentice-Hall, 1992.

[2] W. Kraaij and R. Pohlman, *"Viewing Stemming as Recall Enhancement"*, In the *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996, pp. 40–48.

[3] A.Pirkola, "A. Morphological typology of languages for IR", *Journal of Documentation, 57 (3)*, 2001, pp. 330-348.

[4] M. Popovic and P.Willett, "The effectiveness of stemming for natural-language access to Slovene textual data", *JASIS, 43 (5)*, 1992, pp. 384-390.

[5] M.F. Porter, "*An algorithm for suffix stripping*", Program, 14(3) 1980, pp. 130–137.

[6] C.J. van Rijsbergen, S.E. Robertson and M.F. Porter, "New models in probabilistic information retrieval", *British Library Research and Development Report, no. 5587*, 1980.

[7] C.D. Paice, "An evaluation method for stemming algorithms", In the *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, 1990, pp. 42 – 50.

[8] J.B.Lovins, "Development of a stemming algorithm", *Mechanical Translation and Computational Linguistics 11*, 1968, pp. 22-31.
[9] J. Dawson, "Suffix removal and word conflation", *ALLCbulletin, 2(3)*, 1974, pp. 33–46.

[10] R. Krovetz, "Viewing morphology as an inference process", In Proceedings of the *16 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993, pp. 191-202.

[11] S.Y. Tai, C.S. Ong, and N.A. Abdullah, "On designing an automated Malaysian stemmer for the Malay language", (Poster) In the *Proceedings of the fifth international workshop on information retrieval with Asian languages*, Hong Kong, 2000, pp. 207-208.

[12] M. Greengrass, A.M. Robertson, S. Robyn, and Willett, "Processing morphological variants in searches of Latin text", *Information research news, 6 (4)*, 1996, pp. 2-5.

[13] V. Berlian, S.N. Vega, and S. Bressan, "Indexing the Indonesian web: Language identification and miscellaneous issues", In the *Tenth International World Wide Web Conference*, Hong Kong. 2001.

[14] J. Carlberger, H. Dalianis, M. Hassel, and O. Knutsson, "Improving precision in information retrieval for Swedish using stemming", In the *Proceedings of NODALIDA '01 - 13th Nordic conference on computational linguistics*, Uppsala, Sweden, 2001.

[15] W. Kraaij and R. Pohlmann, "Viewing stemming as recall enhancement", In the *Proceedings of ACM SIGIR96*, 1996, pp. 40-48.

[16] C. Monz and M.de Rijke, “Shallow morphological analysis in monolingual information retrieval for German and Italian in Cross-language information retrieval and evaluation”, In the *Proceedings of the CLEF 2001 workshop*, C. Peters, Ed., Springer Verlag, 2001.

[17] I. Moulinier, A. McCulloh and E. Lund, “Non-English monolingual retrieval in Cross-language information retrieval and evaluation”, In the Proceedings of the *CLEF 2000 workshop*, C. Peters, Ed.: Springer Verlag, 2001, pp. 176-187.

[18] F.C. Ekmekcioglu, M.F. Lynch and P.Willett, “Stemming and n-gram matching for term conflation in Turkish texts”, *Information Research News, 7 (1)*, 1996, pp. 2-6.

[19] A. Ramanathan and D.D. Rao, “A Lightweight Stemmer for Hindi”, In the Proceedings of *EACL*, 2003.

[20] N. UzZaman and M. Khan, “A Comprehensive Bengali Spelling Checker”, In the *Proceeding of the International Conference on Computer Processing on Bengali (ICCPB)*, Dhaka, Bengalidesh, 2006.

[21] N. UzZaman and M. Khan, “A Double Metaphone Encoding for Bengali and its Application in Spelling Checker”, In the *Proceeding of IEEE International Conference on Natural Language Processing and Knowledge Engineering*, Wuhan, China, 2005.

[22] N. UzZaman and M. Khan, “A Bengali Phonetic Encoding for Better Spelling Suggestions”, In the Proceeding of the *7th International Conference on Computer and Information Technology (ICCIT)*, Dhaka, Bengalidesh, 2004.

[23] Bspelling available at: www. sourceforge.net/project/showfiles.php?group_id=43331

[24] Aspell available at: www.aspell.sourceforge.net

[25] P. Kundu and B.B. Chaudhuri, “Error Pattern in Bengali Text”, *International Journal of Dravidian Linguistics*, 28(2) 1999.

[26] B.B. Chaudhuri, “Reversed word dictionary and phonetically similar word grouping based spell-checker to Bengali text”, In the *Proceedings of LESAL Workshop*, 2001.

[27] A.B.A. Abdullah and A. Rahman, “A Different Approach in Spell Checking for South Asian Languages”, In the *Proceedings of 2nd International Conference on Information Technology for Applications (ICITA)*, China, 2004.

[28] K. Kukich, “Techniques for automatically correcting words in text”, *ACM Computing Surveys, 24 (4)*, 1992. pp. 377 - 439.

[29] A. Bhatt, M. Choudhury, U. Sarkar and A. Basu, “Exploring the Limits of Spellcheckers: A comparative Study in Bengali and English”, In the *Proceedings of the Second Symposium on Indian Morphology, Phonology and Language Engineering (SIMPLE'05)*, Published by CIIL Mysore, 2005, pp. 60 – 65.

[30] Wikipedia, www.wikipedia.org.