Discovering suffixes: A Case Study for Marathi Language

Mudassar M. Majgaonker
Comviva Technologies Limited
Gurgaon, India

Tanveer J Siddiqui
J. K. Institute of Applied Physics and Technology
Department of Electronics and Communication
University of Allahabad, Allahabad, India

***Abstract*— Suffix stripping is a pre-processing step required in a number of natural language processing applications. Stemmer is a tool used to perform this step. This paper presents and evaluates a rule-based and an unsupervised Marathi stemmer. The rule-based stemmer uses a set of manually extracted suffix stripping rules whereas the unsupervised approach learns suffixes automatically from a set of words extracted from raw Marathi text. The performance of both the stemmers has been compared on a test dataset consisting of 1500 manually stemmed word.**

Keywords-component; Marathi morphology, Marathi stemmer, Unsupervised stemmer, Rule-based stemmer, Natural language processing

I. INTRODUCTION

Suffix stripping is a pre-processing step required in a number of natural language processing applications such as information retrieval, text summarization, document clustering, and word sense disambiguation. One of the quite widely used tool for this processing is stemmer which uses a suffix list to remove suffixes from words. The stem is not necessarily the linguistic root of the word. For example, words like भारताची(bharatachi), भारतासाठी(bharatsathi), भारतामध्ये (bharatmadhye), भारतानी (bharatani) after stemming may be mapped to common stem भारता(bharatbharata) whereas the root form is भारत. This paper presents the design of two Marathi stemmer – a rule-based and an unsupervised stemmer – and compares their performance. Rule-based stemmers require identification of suffix stripping rules for creating morphological variants. Obtaining such rules for a highly inflectional language like Marathi is difficult and time consuming besides being language specific. It uses a set of words extracted from online Marathi documents [1][2] to learn suffixes automatically and hence can be easily applied to other languages as well. Earlier work in this direction for Indian languages includes Hindi, Bengali, Tamil, and Oriya. But very little amount of work has been done for Western Indian languages like Marathi and Konkani. The rest of the paper is organized as follows:

Section 2 reviews the earlier work done in morphological analysis and stemming for Indian languages. Morphological characteristics of Marathi language has been discussed in section 3. Section 4 offers details on the rule-based and unsupervised approaches. Section 5 presents the details of the experiments and discusses the results. Conclusions have been made in section 6.

II. RELATED WORK

Most of the early work done for stemmers was rule based [3][4]. This requires formulation of linguistic rules for suffix stripping. Wicentowski [6] proposed a supervised approach using the WordFrame model which uses the set of inflection-root pairs for learning the set of suffixes for stripping. An information theoretic approach based on Minimum Description length (MDL) was proposed by Brent et al. [7]. Later a Bayesian Model for MDL was proposed by Snover and Brent [8] for English and French. The work by Goldsmith [9] focuses on the MDL while the work in [10] involves automatic clustering of words using co-occurrence information.

Earlier work in Indian morphology includes [11, 12, 13, and 14]. Larkey et al. [11] defined a light weight stemmer for Hindi using a manually formulated list of 27 most common suffixes for stemming. A similar approach was proposed by Ramanathan and Rao [12]. They used a manually extracted list of 65 most inflectional suffixes. In [13] a statistical Hindi stemmer was developed and used for evaluating the performance of the Hindi information retrieval system. Similar work has been done by Dasgupta and Ng [14] for Bengali morphological analyzer. In [5] an unsupervised Hindi stemmer has been discussed. An approach based on "observable paradigms" for Hindi morphological analyzer is proposed in [15]. A rule based approach TelMore was proposed by [16] for Telugu language. The western Indian languages like Marathi and Konkani have gained very less attention. Earlier reported work includes [17] which uses simple corpus based n-gram matching approach for stemming. In this approach, the classes of words which share a common prefix of given character length were extracted and each of them was replaced by the common prefix.

III. MORPHOLOGICAL CHARACTERISTICS OF MARATHI LANGUAGE

Marathi is morphologically very rich. A single root word may have different morphological variants, for example, words like भारताची , भारतानी , भारतासाठी , भारताकडून , भारतावर , भारताकडे

(*Bharatachi, Bharatani, Bharatasathi, Bharatakadun, Bharatavar, Bharatakade*) are morphological variants of the word भारत .

Like Hindi, the variants in Marathi are usually formed by adding suffixes to the stem or root word. We can categorize the suffixes found in Marathi into three types:

i. **Plain Suffixes**: Plain suffixes are also called dependent vowel Signs. ◌ा, ◌ि, ◌ी, ◌ु, ◌ू, ◌े,… are some of the suffixes which combine with the root word to produce its morphological variants.

For example, मुलगा , मुलगी , मुली , मुले

ii. **Join word suffixes**: Join word suffixes are those suffixes which are formed by merging two or more consonants and vowels. These join words are formed by merging any of the consonants with the morphological variant ◌्या of the consonant य . Variants like ळ्या , ल्या , ऱ्या , न्या , च्या are considered as the join word suffixes. For example, words like उद्या , देवासाराख्या , सलमानच्या consists of join word suffixes.

iii. **Complex suffixes**: Complex suffixes are formed by combining two or more consonants with the plain suffixes. For example, देवासाठी , ऐश्वार्यावर , तुझ्याकडून are words containing complex suffixes साठी , वर , कडून , मध्ये . The inflections forming prefix part are rarely found in Marathi. Prefix stripping results in change in the meaning of the word. So, it can be neglected.

iv. **Words follow a specific pattern**: Unlike other Indian languages it is found that words in Marathi language follow a specific pattern. The words in the Marathi can be expressed as:

<token>:= <stem/root word> + <inflection>

<inflection>:=<inflections>+<inflections>

IV. OUR APPROACH

The rule-based stemmer extracts suffix stripping rules based on the morphological characteristics of Marathi discussed in previous section. The unsupervised stemmer learns suffixes automatically from a set of Marathi words.

A. Rule-based Stemmers

The common morphological patterns found in Marathi are:

1. <original word>:= <stem/root words> + <plain suffixes>

   e.g., घेत + ◌ा , भारत + ◌ाची

2. <original word>:= <plain suffixes + <complex suffixes>

   e.g., सलमान + वर

3. <original word>:= <plain suffixes + join word + complex suffixes>

   e.g., दहशतव + ◌ा + यां + कडून

4. <original word>:= <plain suffixes + join word suffixes + complex suffixes + join word suffixes>

   e.g., घर + ◌ा + समोर + च्या

The suffix stripping rules for the rule-based stemmer are based on these patterns. Fig. 1 depicts steps in the algorithm and fig. 2 shows the trace of the algorithm on the word ऑफिसमधलेच.

Input: List of words

Output: Stem of words

Step 1: Eliminate all the complex suffixes.

e.g:- कडून , मुळे, साठी , प्रमाणे, वरून, वर ,…

Step 2: Eliminate the join word suffixes i.e. Eliminate the inflections of consonants like च, ल, ण, ळ, र, द… with या

e.g., द्या = <द> + <या>

Step 3A: Eliminate the inflections for consonant च

Step 3B: Eliminate the inflections for consonant ल

Step 3C : Eliminate the inflections involving plain suffixes

Fig. 1. Algorithm for Rule Based Marathi Stemmer

In some cases the algorithm results in under stemming, e.g., the output of the algorithm on the word ऑफिसमधलेच isre is < ऑफिसमधले > instead of ऑफिस. <मधले> is a valid suffix in Marathi. But the algorithm fails to eliminate the suffix <मधले>. One of the solutions to this problem is to call the algorithm recursively. But this results in over-stemming resulting in a drop in the performance.

B. Unsupervised Stemmer

The unsupervised stemmer is based on n-gram splitting approach as in [5] [19]. We have extracted words from the Marathi corpus [1] and split them using n-gram model to get the n (n=1,2,3,……,*l*) split suffixes, where *l* is the length of the word. Fig. 2 briefs the algorithm and the steps are discussed below.

1. Segment words using n-gram splitting
2. Generate stem classes using maximum common prefix
3. Generate suffix lists for stem classes identified in (2)
4. Generate suffix rules

Fig. 2. Algorithmic steps

Step 1. Word Segmentation

In this step the word W_j is split into n-grams using n-gram model to obtain corresponding stems and suffixes as:

W_j: = {($stem_{1j}$|$suffix_{1j}$); ($stem_{2j}$|$suffix_{2j}$); …….. ; ($stem_{ij}$|$suffix_{ij}$)}

where $stem_{ij}$ is the i^{th} stem of j^{th} word and $suffix_{ij}$ is the i^{th} suffix of j^{th} word.

For example, the word अंकात can be split up into following stems and suffixes:

अंकात := { (अंकात | NULL); (अंका | त); (अंक | ात); (अं | कात); (अ | ंकात); (NULL| अंकात) }

Step 2. Generation of Stem Classes

Next, we have used maximum common prefix method to find common stems and then grouped the words with common stems under a single common stem class.

For example, if W_m and W_n are two words segmented as:

W_m: = {($stem_{1m}$|$suffix_{1m}$); ($stem_{2m}$|$suffix_{2m}$); …($stem_{im}$|$suffix_{im}$); ($stem_{lm}$|$suffix_{l1m}$)}

W_n = {($stem_{1n}$|$suffix_{1n}$); ($stem_{2n}$|$suffix_{2n}$); …($stem_{in}$|$suffix_{in}$); ….. ; ($stem_{ln}$|$suffix_{l2n}$)}

Let R be the largest common prefix such that R = $stem_{im}$ = $stem_{jn}$ then the words are stored under the same equivalence class R , called stem class, as:

R = [W_m, W_n]

For example, the words चित्रपट, चित्रपटांतील, चित्रपटाचा, चित्रपटाचे, चित्रपटात, चित्रपटाने can be grouped under the stem class चित्रपट as:

चित्रपट := [चित्रपटांतील, चित्रपटाचा, चित्रपटाचे, चित्रपटात, चित्रपटाने]

Step 3. Generation of stem and suffixes

This step finds stems and suffixes of the word. For this, the longest common prefix from the stem equivalence class formed in the first stage is identified and stored as stem; the remaining part is stored as suffix.

For example, the words चित्रपटांतील, चित्रपटाचा, चित्रपटाचे, चित्रपटात, चित्रपटाने, चित्रपटाप्रमाणे, चित्रपटाच्या can be grouped as:

चित्रपट := [◌ांतील, ◌ाचा, ◌ाचे, ◌ात, ◌ाने, ◌ाप्रमाणे, ◌ाच्या]

Similarly,

प्रशासन:= [◌ाचा, ◌ाकडून, ◌ात, ◌ाने, ◌ावर]

Step 4. Generation of suffix rules

In this step the rules for suffix stripping are generated. Several simple rules such as s → ϵ, where s is the suffix and ϵ is any non-empty string, are introduced. Three approaches for suffix rules generation has been used in evaluation.

(a) Frequency based suffix stripping

This is the crudest method for suffix rule generation. The suffixes are sorted according to the descending order of their frequencies. A threshold is set manually and the frequencies lying above the particular threshold are considered as the valid candidates for suffix rules generation and those lying below the threshold are discarded. This is done in order to reduce the number of rules for suffix stripping. This method works for highly inflectional languages like Marathi up to some extent because the number of suffixes is very high in Marathi.

(b) Iterative suffix stripping

This is an optimization over the frequency based suffix stripping approach. It involves repeated application of suffix rules derived using frequency-based stripping approach in an attempt to handle under stemming. If the accuracy of the stemming increases after iteration the stripped part of the word is added as a rule.

(c) Statistical stripping

Statistical stripping approach is similar to unsupervised approach discussed in [5]. It assigns a score to individual stems and suffixes using frequency of the word in the corpus and total number of words. The split score is calculated by multiplying the scores of the stem and suffixes.

Stems and suffixes are given an initial score using the following expression:

$$P(stem_{ij}) = \frac{f_{W_j}}{T_w * len(W_j)}$$

$$P(suffix_{ij}) = \frac{f_{W_j}}{T_w * len(W_j)}$$

Where, f_{W_j}= Frequency of word W_j in corpus.

T_w = Total number of words in corpus.

$len(W_j)$ = Length of word W_j (number of character).

$$P(split_{ij}) = P(stem_{ij}) * P(suffix_{ij})$$

The scores are updated as follows:

$$P_{new}(stem_{ij}) = \sum_{k=0}^{m} \{P(stem_{ij})\}$$

$$P_{new}(suffix_{ij}) = \sum_{k=0}^{m} \{P(suffix_{ij})\}$$

$$P_{new}(split_{ij}) = P_{new}(stem_{ij}) * P_{new}(suffix_{ij})$$

Where m is the number of similar suffixes or stems.

The split having maximum score is used to generate suffix list.

V. EXPERIMENT AND RESULTS

A. Dataset

Two test dataset has been created. The TestDataset1 consists of 1500 words extracted from the Marathi Corpus [1]. These documents were not used in training. TestDataset2 consists of 1500 words extracted from the documents collected from internet [2]. The stem for these words have been defined manually. The training dataset consists of 1,32,895 words extracted from the Marathi corpus. Table 1 summarizes the statistics used for the dataset generation.

TABLE 1. DATA SET GENERATION

Dataset	Total Number of Words	Total Number of Unique Words	Minimum Length of the Word	Maximum Length of the Word
Train Dataset	1,32,895	27,613	4	12
Test Dataset 1	19,365	1500	4	10
Test Dataset 2	14,956	1500	4	12

B. The Experiment

In order to evaluate the performance we have conducted two test runs on test dataset 1 and test dataset 2 respectively. The accuracy is measured in terms of accuracy which is defined as fraction of words stemmed correctly.

C. Results and Discussions

Table 2 shows the results of comparisons of the three approaches used for suffix rule generation. As shown in table 2, we observed a maximum accuracy of 80.7% with rule-based stemmer and a maximum accuracy of 82.5% with the unsupervised stemmer using statistical stripping approach on test dataset 1. The accuracy observed with rule-based stemmer in test run 2 is 78.4%. In test run 2 the three different suffix generation methods of unsupervised approach results in an accuracy of 61.8%, 70.3% and 81.6% respectively. The frequency based suffix stripping approach has the lowest accuracy in both the test run. This is due to the problem of under stemming. For example, the word अंकात after applying frequency based suffix stripping approach is stemmed to अंका instead of correct stem अंक. The iterative suffix stripping approach is able to overcome this problem resulting in an improvement in the accuracy in both the test run. The maximum accuracy achieved using this approach is 72.8%. The statistical suffix stripping approach gave the maximum accuracy of 82.5%. This is even better than the accuracy observed with rule-based approach. All the approaches performed better on test dataset 1. The reason may be that the dataset 2 consists of words extracted from internet documents. The presence of noise, e.g. foreign language words, spelling variations, may be responsible for comparatively poor performance.

TABLE 2. RESULTS

Run	Approach	Accuracy
Run1	Rule-based stemmer	80.7%
	Frequency Based Suffix stripping	63.5%
	Iterative suffix Stripping	72.8%
	Statistical stripping	82.5%
Run 2	Rule-based stemmer	78.4%
	Frequency based suffix stripping	61.8%
	Iterative suffix stripping	70.3%
	Statistical stripping	81.6%

VI. CONCLUSION

An unsupervised approach to Marathi stemmer has been discussed. Three different approaches for suffix rules generation has been used in unsupervised stemmer. The maximum accuracy observed is 82.5% for the statistical suffix stripping approach. The approach is unsupervised and language independent. It uses a set of words to learn suffixes and does not require any linguistic input. Hence, it can be used for developing stemmer of other languages as well.

REFERENCES

[1] http://www.cfilt.iitb.ac.in/marathi_Corpus/

[2] http://www.esakal.com/

[3] M. Porter, "An algorithm for suffix stripping program," Vol. 14, pp. 130-137, 1980.

[4] Julie Beth Lovins, "Development of a stemming algorithm. Mechanical Translation and Computational Linguistics," 11:22–31, 1968.

[5] Amaresh Kumar Pandey, Tanveer J Siddiqui, "An unsupervised Hindi stemmer with heuristic improvements," In the Proceedings of the Second Workshop on Analytics for Noisy Unstructured Text Data, AND 2008, Singapore, July 24, pp. 99-105, ACM International Conference Proceeding Series, 2008.

[6] R. Wicentowski, Multilingual Noise-Robust Supervised Morphological Analysis using the WordFrame Model," In Proceedings of Seventh Meeting of the ACL Special Interest Group on Computational Phonology (SIGPHON), pp. 70-77, 2004.

[7] M. R. Brent, S. K. Murthy and A., Lundberg "Discovering morphemic suffixes: A case study in minimum description length induction," In Proceedings of the fifth international workshop on artificial intelligence and statistics, 1995.

[8] M. G. Snover, and M. R. Brent, "A Bayesian model for morpheme and paradigm identification" In Proceedings of the 39th annual meeting of the ACL, pp. 482–490, 2001.

[9] John Goldsmith, "Unsupervised Learning of the Morphology of a Natural Language," Computational Linguistics, Volume 27, No. 2, pp 153-198, 2001.

[10] D. Freitag, "Morphology induction from term clusters," In Proceedings of the ninth conference on computational natural language learning (CoNLL), pp. 128–135, 2005.

[11] Leah S. Larkey, Margaret E. Connell and Nasreen Abdul Jaleel, "Hindi CLIR in thirty days" ACM Transaction on Asian Language Information Processing, Vol. 2, No. 2, Pages No. 130-142, 2003.

[12] A. Ramanathan, and D. Rao, "A lightweight stemmer for Hindi," In Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL) on Computational Linguistics for South Asian Languages (Budapest, Apr.) Workshop, 2003.

[13] A. Chen and F. C. Gey, "Generating statistical Hindi stemmers from parallel texts," ACM Trans. Asian Language Inform. Process. Vol. 2(3), 2003.

[14] Sajib Dasgupta, Vincent Ng, "Unsupervised morphological parsing of Bengali," 2007.

[15] Akshar Bharat, Rajeev Sangal, S. M. Bendre, Pavan Kumar and Aishwarya, "Unsupervised improvement of morphological analyzer for inflectionally rich languages," Proceedings of the NLPRS, pp. 685-692, 2001.

[16] Madhavi Ganapathiraju and Levin Lori, TelMore: "Morphological Generator for Telugu Nouns and verbs," In the proceedings of Second International Conference on Universal Digital Library Alexandria, Egypt, November 17-19, 2006.

[17] Jiaul H. Paik and Swapan K. Parui, "A Simple Stemmer for Inflectional Languages ," (No date)

AUTHORS PROFILE

Mudassar J. Majgaonkar Tech. (Intelligent System) from IIIT Allahabad, India. Currently he works for Comviva Technologies Limited, Gurgaon, India.

Tanveer J. Siddiqui is a Senior Lecturer at the Allahabad University of India. She has more than 10 years of teaching and research. Her current rcscarch interests are Natural Language Processing and Information Retrieval.