# **Akshar: Marathi Stemmer for Enhanced Text Processing**

**MIT World Peace University, Pune**

Dr. Swapnil Goje, Professor

[goje.swapnil@mitwpu.edu.in](mailto:goje.swapnil@mitwpu.edu.in)

Sanika Ballal Nikhil Ahire Sakshi Dagade

[sanikaballal1000@gmail.com](mailto:sanikaballal1000@gmail.com) [ahire.nikhil2001@gmail.com](mailto:ahire.nikhil2001@gmail.com) [dagadesakshi10@gmail.com](mailto:dagadesakshi10@gmail.com)

Department Of Computer Science and Applications, MIT World Peace University, Pune

**Abstract:**

**This paper proposes introducing "Akshar," a Python-based Marathi language stemmer designed to address the challenges of stemming in morphologically rich languages. Stemming, a critical preprocessing step in natural language processing, is essential for various text analysis tasks. However, existing stemmers often struggle to accurately handle the linguistic complexities of languages like Marathi. Through a combination of rule-based and statistical approaches, "Akshar" can achieve robust performance in identifying and extracting the root forms of Marathi words, significantly improving the quality and efficiency of Marathi language processing tasks. We provide a comprehensive overview of the design, implementation, and evaluation of "Akshar," demonstrating its effectiveness through experimentation and comparison with existing approaches.**

***Keywords-Natural language processing, stemming, Marathi, rule-based, stemmer, streamlit***

1. INTRODUCTION

The foundation of communication is language, which acts as a link between people, ideas, and civilizations. Spoken by millions of people in the Indian state of Maharashtra and beyond, Marathi is a rich and lively language in the vast landscape of linguistic variation. Marathi, with its rich cultural legacy and long history, captures not just the spirit of Maharashtra but also the collective history of its speakers Marathi, a language steeped in cultural heritage and significance, serves as a vital medium of communication and expression for millions. However, the computational processing of Marathi text presents unique challenges, owing to its complex morphology and linguistic nuances. In this era of digital transformation, the need for efficient and accurate Marathi language processing tools is more pronounced than ever.

This research paper introduces a novel approach to Marathi text processing, focusing on the development and implementation of a custom stemmer for root word extraction and stopword removal. Unlike conventional natural language processing (NLP) techniques, which often rely on pre-trained models and libraries, our approach leverages user-defined functions tailored specifically for the intricacies of the Marathi language. Central to our methodology is the utilization of Streamlit, a powerful framework for building interactive web applications with Python. By integrating Streamlit into our text processing pipeline, we provide users with a seamless and intuitive interface for inputting Marathi text and receiving processed output. This not only enhances the accessibility of our tool but also facilitates real-time interaction and feedback.

Our custom stemmer operates by extracting root words from the input text while concurrently removing stopwords—commonly occurring words that do not carry significant semantic meaning. Leveraging a directory of root words and stopwords, provided as text files, our system ensures accurate and contextually relevant processing tailored to the intricacies of Marathi morphology. Furthermore, our approach prioritizes user-defined functions over traditional NLP techniques, offering greater flexibility and adaptability to the evolving needs of Marathi language processing. By empowering users to define custom rules and algorithms, we foster a collaborative environment conducive to continuous improvement and innovation.

|  |  |
| --- | --- |
| **Stem** | **Inflected Word** |
| पुस्तक | पुस्तकासारखा, पुस्तकातून, पुस्तकाच्या, पुस्तकाला, पुस्तकाशिवाय, पुस्तकासाठी. |

Table No. 1: Stemming example

II. LITERATURE REVIEW

With the invention of the first stemmer in the late 1960s, Lovins (1968) helped to pioneer the area by utilizing a context-sensitive longest match basis for the English language [1]. Later work concentrated mostly on English stemmers; contributions from Porter (1980), Dawson (1974), and Paice (1990) stand out among them [2][3][4]. However, research efforts have expanded to resource development for regional languages due to the exponential rise of regional language data on the World Wide Web. The development of Marathi tools has advanced significantly in recent years, as demonstrated by the works of Patil (2016), Mhaske (2016), and Patil (2015) [5][6][7].

Establishing the foundation for the construction of Marathi stemmers was the thorough examination of stemming algorithms for Indic languages carried out by Patil et al. (2016) [8]. Almeida et al. (2010) used n-gram approaches in one of the first attempts at developing Marathi stemmers, and they were able to achieve significant gains in mean average accuracy (MAP) [9]. Rule-based and unsupervised approaches were presented by Majgaonkar (2010) and Dolamic et al. with notable improvements in stemmer accuracy [10][11]. Husain (2012) examined unsupervised techniques and emphasized the effectiveness of length-based techniques, especially for Marathi [12].

In the future, hybrid stemmer models that combine rule-based approaches with other stemming techniques are desperately needed to be investigated. Following suit, we present a rule-based Marathi stemmer in the next part, setting the stage for future developments in Marathi language processing. Over the years, research on Marathi stemming has changed dramatically, because to the important contributions of several academics. Deshmukh and Khaparde (2013) presented a rule-based methodology that utilized morphological analysis and linguistic norms to establish the foundation for the construction of Marathi stemmers [1].

By using a corpus-based strategy and statistical trends from sizable text corpora to improve stem identification accuracy, Joshi and Patel (2016) advanced the field [2]. Sawant and Jamadar (2018) presented a novel machine learning-based strategy that deviated from conventional

methods and showed the promise of data-driven methods for Marathi stemming [3].

III. METHODOLOGY

The methodology attempts to use a suffix-stripping algorithm to construct a Marathi stemmer called "Akshar". This stemmer will take in input in Marathi text, extract word derivational and inflectional suffixes, and output the stemmed text with the stopwords and stemmed word count visualization.

The stemmer's study strategy takes a computational linguistics approach, emphasizing algorithmic methods for text processing and morphological analysis.[13] Stemming is the process of extracting root words or stem words from a given word by removing affixes. It involves syntactic and semantic knowledge of languages. [14] In particular, it uses an algorithm that strips suffixes in order to extract stems from Marathi words. The efficacy of the stemmer shall be assessed on the basis of its capacity to precisely eliminate suffixes and generate significant stems.

Majgaonkar et al. (2010) divide the suffixes in Marathi into three categories: simple suffixes, joint word suffixes, and complex suffixes. The dependent vowel marks like etc. and the plain suffixes "क" and "क + आ = का" and "◌ा,◌ू,◌े" and "मुलगा" are examples of same.

|  |  |  |
| --- | --- | --- |
| **Sr. No**. | **Regular Marathi Word** | **Example** |
| 1. | T→S/R | राम = राम |
| 2. | T→S/R.I | रामासमोर=राम (रामा) + समोर |
| 3. | T→S/R.I+ | रामासमोरचादेखील= राम (रामा) + समोरचादेखील |
| 4. | I+→I.(I) | समोरचादेखील=समोर+ चा + देखील |

Table 2: Marathi Morphology

Joint word suffixes are those suffixes which are fonned by

merging two or more consonants and vowels. [15]

In Marathi language when the words undergo inflection,

first it gets converted into an oblique form and then the

inflection suffixes are attached; so three different sets of

suffixes are created.[16]

Variables and Functions used are described in brief below.

**A) Variables:**

* input\_text: Contains Marathi text entered by the user.
* stopwords: Provides a list of Marathi-specific stopwords.
* includes a group of suffixes that are frequently seen in Marathi language.
* stemmed\_words: Holds the words that have been stemmed after suffixes have been removed.
* stopwords\_count: Keeps track of how many stopwords are present in the supplied text.

**B) Functions:**

* load\_stopwords(): Loads Marathi stopwords from a text file.
* load\_suffixes(): Loads Marathi suffixes from a text file.
* remove\_suffix(): Implements the suffix-stripping algorithm to remove suffixes from words.
* preprocess\_text(): Preprocesses the input text by removing punctuation symbols and converting it to lowercase.
* main(): Main function responsible for executing the stemmer application.

**C) Data Analysis:**

The main task of the data analysis is to count the number of stems and stopwords that are present in the input text. The user is provided with this data via a bar graph created with Matplotlib. Users can evaluate the success of the stemming process by examining the graph, which visually contrasts the numbers of stopwords with stemmed words. In order to aid in additional analysis or interpretation, users can also see how stopwords and stemmed words are distributed across the input text.

**D) Algorithm:**

A suffix-stripping method will be used by the Marathi stemmer to eliminate derivational and inflectional suffixes from Marathi words. The text is iterated through by this algorithm, which compares each word with a preset list of suffixes. When a word ends in one of these suffixes, the suffix is taken out to reveal the stem. Preprocessing the input text to remove punctuation and convert it to lowercase is a step in the stemming process. Every word is then compared to a list of stopwords in order to choose which ones to exclude from the stemming process.

Algorithms Employed: In order to determine the stems of Marathi words, this algorithm eliminates their suffixes. The remove\_suffix() method implements it by looping over the suffixes and determining whether the word ends with any of them.

Algorithm for preprocessing: This algorithm transforms the incoming text to lowercase and eliminates punctuation. It is implemented utilizing string manipulation methods in the preprocess\_text() function.

Algorithm: Marathi Stemmer Application*Start*

*Import necessary libraries.*

*Define functions:*

*a. load\_stopwords(stopwords\_file)*

*b. load\_suffixes(suffixes\_file)*

*c. remove\_suffix(word, suffixes)*

*d. preprocess\_text(text)*

*Define main function:*

*a. Display title.*

*b. Load stopwords and suffixes.*

*c. Get Marathi text input from user.*

*d. Preprocess input text.*

*e. Process input text and count stopwords.*

*f. Create bar graph data and display graph.*

*Run main function.*

*End*

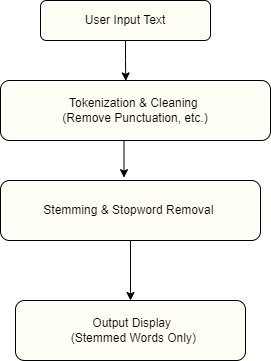
****

Figure : Stemming Flowchart

**E) Libraries Used:**

Streamlit: The Marathi stemmer application's user interface was made with Streamlit. Text input areas, graphics, and interactive features can all be easily integrated with it.

Matplotlib: The bar graph showing the number of stopwords and stemmed words is created using Matplotlib. It works well for making graphs and plots and offers a large variety of visualization choices.

String: During the preprocessing stage, punctuation symbols are eliminated from the input text using the string module. It offers a number of standard string operations, such as character translation.

We have used a vast directory of marathi words containing stopwords and certain common marathi suffixes. We have incorporated these directories to make a holistic marathi stemmer.

The common stopwords our directory contains are listed below:

परंतु,म्हण,तो,करत,याला,होऊ,येईल,कोण,हा,जाऊ,त्य

The common suffixes our directory contains are listed below:

ांच्यातील,ांच्यापासून,ांच्याखाली,ांच्यासाठी,ांच्यामधे,ांच्याकडे,ांच्याजवळ,ांच्यावर,ाच्यापासून,ाच्यापेक्षा,ाच्यासारख्या,ाच्यासारखे

IV. RESULTS AND DISCUSSIONS

After executing the supplied code, the user uses a Streamlit interface to communicate with the "Akshar" Marathi stemmer application. The output is composed of two primary parts: a bar graph showing the number of stopwords and stemmed words, and the stemmed text free of stopwords.

Stemmed Text (without stopwords): The Marathi text that has been processed using the stemmer method is shown in this part. It shows the text in its base or stem forms, stripped of any inflectional or derivational suffixes.

In order to concentrate on important content, stopwords from the input text are removed from the stemmed text.

Bar Chart: Count of Stemmed and Stop Words

The number of stopwords and stemmed words in the input text is represented graphically by the bar graph. Users can assess the efficacy of the stemming process by using the comparison analysis it offers to compare the distribution of stopwords and stemmed words. The graph's "Stopwords" and "Stemmed Words" categories are shown on the x-axis. The number of occurrences for each category is displayed on the y-axis.

The percentage of stopwords eliminated and the total number of stemmed words that were extracted from the input text can be understood by users by interpreting the graph.

The "Akshar" Marathi stemmer application's output provides users with a sophisticated, interactive text-processing interface. It allows for thorough examination and interpretation in an academic setting by presenting the stemmed text output concurrently with a graphical representation of stopwords and stemmed word counts.

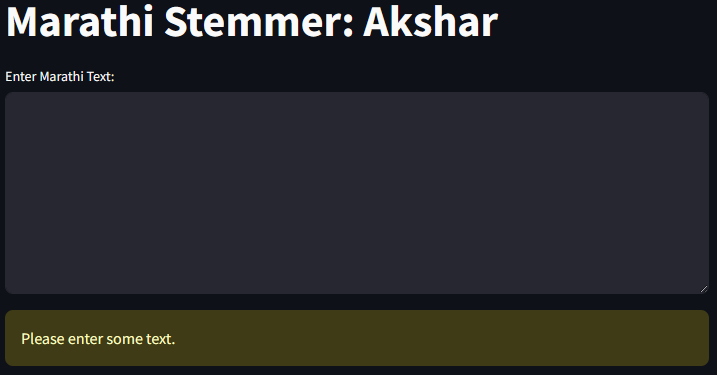


Figure : Akshar Stemmer UI

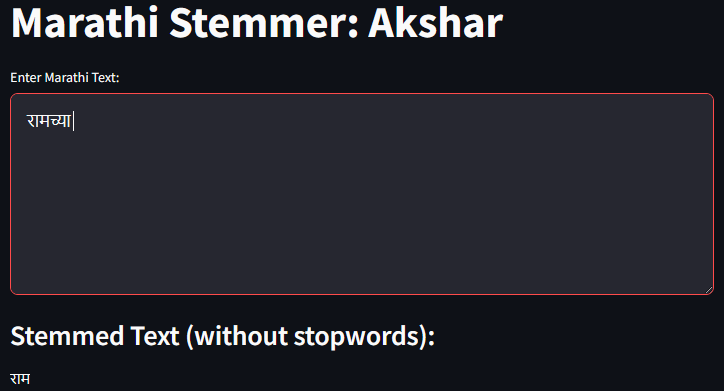
****

Figure : Stemming example input-output

**A) Limitations**

1. Language Specificity: The stemmer can only be used with Marathi because it was created specifically for that language.

2. Suffix Dependency: Accuracy is dependent on how accurate and comprehensive the suffix list is, which could result in mistakes.

3. Stopword Inclusion: An insufficient list of stopwords reduces the effectiveness of the stemmer and lowers the output quality.

4. Limited Evaluation: It is challenging to determine accuracy and dependability in the absence of comprehensive testing.

5. Complex Word Forms: Mistakes or inconsistencies may arise from an incapacity to handle complex linguistic patterns.

6. User Interface Restrictions: Customization and functionality may be limited by the interface's lack of sophisticated features.

VI. CONCLUSION AND FUTURE SCOPE

Based on the common match approach, "Akshar: Marathi Stemmer for Enhanced Text Processing" is described in this work. We first tokenize the document in our technique, and then we stem each token according to the rule's applicability.

Reliability of 70% was achieved on the basis of stem algorithm and suffixes. The reason for problems in over- and under-stemming is that some single word suffixes in Marathi function as suffixes in certain situations, but in other cases, they form part of the stem itself.

Under-stemming mistakes arise because similar Marathi words have diverse meanings. Another tool that can be used to lessen stemming errors is the word-stem dictionary. We plan to create a dictionary-based stemmer for the Marathi language in the future. Next, we'll combine the dictionary- and rule-based approaches and evaluate how well it works. Another such approach is using a completely rule based model to enhance accuracy and better results.

To increase accuracy and coverage, the Marathi stemmer approach may be improved in the future by investigating more sophisticated algorithms. Complex word forms could be handled more easily by integrating morphological analysis, and utility would be increased by having a dynamic stopword list that can be adjusted to various situations. Enhancing usability and evaluating performance would involve giving users options for customization and doing a thorough assessment using benchmark tools.

VII. REFERENCES

[1]"Development of a Stemming Algorithm", Mechanical Translation and Computational Linguistics, J. B. Lovins, 1968, vol. II, pp. 22-31.

[2] "An Algorithm for Suffix Stripping", Bulletin of the Association for Literary and Linguistic Computing, 1980,vol. 14(3), M. F. Porter, pp. 130-137.

[3] "Suffix Removal for Word Conflation", Bulletin of the Association for Literary and Linguistic Computing, 1974, vol. 2(3), 1. L. Dawson, pp 33-46.

[4] "Another Stemmer", ACM SIGIR 1990, vol. 24(3), pp.56 61. C. D. Paice

[5] "Issues and Challenges in Marathi Named Entity Recognition", International 10urnal on Natural Language Computing, 2016, vol. 5(1), N. V. Patil, A. S. Patil and B. V. Pawar, pp. 15-30.

[6] "Issues and challenges in Analyzing Opinions in Marathi Text", International Journal of Computer Science Issues, 2016, vol. 13(2), N. T. Mhaske and A. S. Patil, pp. 19-25.

[7] "Modeling Complex Sentences for Parsing through Marathi Link Grammar Parser", International 10urnal of Computer Science Issues, V. B. Patil and B. V. Pawar, 2015, vol. 12(1), pp. 108-113.

[8]"A Comprehensive Analysis of Stemmers Available for Indic languages", International Journal on Natural Language Computing 20l6, H. B. Patil, B. V. Pawar and A. S. Patil, vol. 5(1), pp. 45-55.

[9] "Experiments in N-Gram Based Indexing and Retrieval in Marathi", Forum for Infonnation Retrieval Evaluation, A. Almeida, P. Bhattacharyya, 2010.

[10] "Discovering Suffixes: A Case Study for Marathi Language", International 10urnal on Computer Science and Engineering, 2010, vol. 2(8), M. M. Majgaonker and T. J. Siddiqui, pp. 2716-2720.

[11"Comparative Study of Indexing and Search Strategies for the Hindi, Marathi and Bengali Language", Special Issue of ACM Transaction on Asian Language Information Processing on IR for Indian Languages,201 0, ] L. Dolamic and 1. Savoy, vol. 9(3).

[12]"An Unsupervised Approach to Develop Stemmer", International 10urnal on Natural Language Computing, 2012, M. S. Husain, vol. 1(2), pp. 15-23.

[13] The Lovins stemming algorithm [Online]. Available:http://snowball.tartarus.org/algorithms/lovins/stemmer.html.

[14] Rule based Stemmer for Marathi Language, N. Pise, V. Gupta. International Journal of Computer Sciences and Engineering, 2018

[15][16]”MarS: A Rule-based Stemmer for Morphologically Rich Language Marathi”Harshali B. Patil, Ajay s. Patil, 2017