© June 2016 | IJIRT | Volume 3 Issue 1 | ISSN: 2349-6002

Unsupervised Stemmer for Kannada Language

Mounika S[1], B H Manjunath Kumar[2]
[1]*M.Tech Scholar, Computer Science and Engineering*
[2]*Asst. Professor, Dept. Computer Science and Engineering*
Sri Siddhartha Institute of Technology, Tumkur
Sri Siddhartha Academy of Higher Education, Tumkur, India

***Abstract*— The Information storage in the internet is increasing day by day. Information Retrieval (IR) is the process which is used to get the information through internet. Stemmer is the basic Natural Language Processing (NLP) tool for achieving IR in a simple manner. Even though Kannada is a language rich in literature, its resources are poor when viewed through the prism of computational linguistics. The development of NLP in Kannada language is not explored much and is in the beginning stage compared to other Indian languages.This paper presents an unsupervised approach for the development of a stemmer for Kannada Language. The proposed algorithm is very simple which can be used with other Indian languages.**

***Index Terms*-- Stemming, Kannada Language, Information Retrieval, Over-stemming, Under-stemming**

I. INTRODUCTION

The information such as news, weather report, annual report, technical and scientific books in the Internet are available in Kannada language. Kannada is a language rich in literature; its resources are poor when viewed through the prism of computational linguistics. In non-Indian languages like English, lot of work has been done in Natural Language Processing (NLP). But in Kannada it is not that much explored and is in the beginning stage. There is lack of resources for NLP in Kannada. Now a day the amount of information stored is increasing. The advent of computers increased the amount of electronic data that can be stored. IR is essentially the requirement of documents in a collection that should be retrieved to satisfy a user's need for information.

The user's information requirement is represented by a query or profile, and contains one or more search terms. The search terms may take different morphological variations. This is because in a natural language a word can assume a variety of morphological variations which can be attributed solely to the process of inflation or derivation. For instance, *reduce* may be used as *reduce, reduced,* or *reduction.* Hence the need for some kind of natural language processing in order to recognize their equivalent form & reduce of the variants to a common base word or stem which improves the performance of IR. The processing can exploit stemming or lemmatization as its tool.

Stemming reduces the words to the common stem merely by truncation i.e. chopping off the unnecessary morpheme (suffix). They are not considerate about the grammatical rules of the language. For example, the collection of words viz. '*produce'*, *'produced'*, '*producing'* and '*production'* are stemmed to a common word '*produc-'*. While above is the case with stemming, Lemmatization removes the inflectional endings and returns the base or dictionary form of the word. The lemma for the above example is *'produce'.* It is closely related to stemming with a difference that the stemmer operates on a single word without the knowledge of the context, and therefore does not consider the parts of speech.

The stemming is normally exposed to two problems. *Over-stemming* and *under-stemming*. *Over-stemming* is said to have occurred when words that are not morphological variants are conflated. Alternatively, it occurs in case of conflation of semantically distant words. An example in English would be, *compile* and *compute* getting stemmed to *comp*. Under-stemming occurs when two semantically exact words which may be differently inflected should be stemmed. Evidently, under stemming occurs when words that are morphological variants are not conflated. For instance, *compile compiling* gets stemmed to "*compil"*. Though under stemming limits the reduction in the size of the data index for IR, it is not an alarming issue compared to over-stemming.

Over stemming exhibits the potential of misleading the user by producing irrelevant stems & there by retrieving inaccurate & probably wrong results. This paper presents simple and unsupervised algorithm for stemming. Also solves the problem of over stemming and under stemming.

II. RELATED WORK

Ramanathan and Rao (2003) [1] - developed a stemming algorithm for Hindi which was lightweight stemming approach .They used the striping method with suffix matching. This Hindi stemmer contained list of suffixes which were created manually. It was tested on text documents taken from different fields like entertainment, politics, business, health and sports. The number of unique terms were thirty five thousand nine hundred seventy seven. This algorithm also discussed about the over stemming and under stemming. It has not reported any recall or precision for performing stemming.

Deepa Gupta,Rahul Kumar Yadav, Nidhi sajan(2012) [2] - In their work, They propose a stemming with partial lemmatization for Hindi. The stemmer proposed is unique. They proposed the different grouping criteria. Aimed to overcome over stemming. Later is tackled by the introduction of lemma. They incorporated lemmatization based on data heuristics obtained from the corpus, without the use of word class information. Application of this concept to unsupervised stemming yielded significant improvements in the desired results when compared to other prevailing approaches of its genre.

Dalwadi Bijal, Suthar Sanket,[3] - discussed different stemming algorithm for non-Indian and Indian language, methods of stemming, accuracy and errors.

Vishal Gupta (2014)[4] – designed a simple algorithm for stemmer. They proposed the different suffix elimination method by using frequently occurring suffix list. It is a rule based stemmer closer to the morphology.

III. PROPOSED ALOGORITHM

We have developed an unsupervised stemmer, which can be used with any language.

The following lines describe the steps for algorithm in detail.

Step1: decompose the word list into stems and suffixes

Each word is decomposed. The first part is taken as the stem and the second is suffix. Taking minimum length of the stem is 2.

Step2: Collect suffixes for each stem

The data we get in the step1 is stored such that each stem is having all its suffixes without duplication of stems. To get this following algorithm must be used.

```
While [stem!]
        {
        If [stem] not available in [stem list]
        Then push [stem] into [stem list]
        If [stem] available in [stem list]
        If [suffix] not present in
        [Suffix bag] of particular stem
        Then push [suffix] into [suffix bag]
        }
```

Step3: Generation of output

The output we get after applying the previous steps is of the form,

Stem + Suffixes.

Figure1 shows the semantic diagram of stemmer.

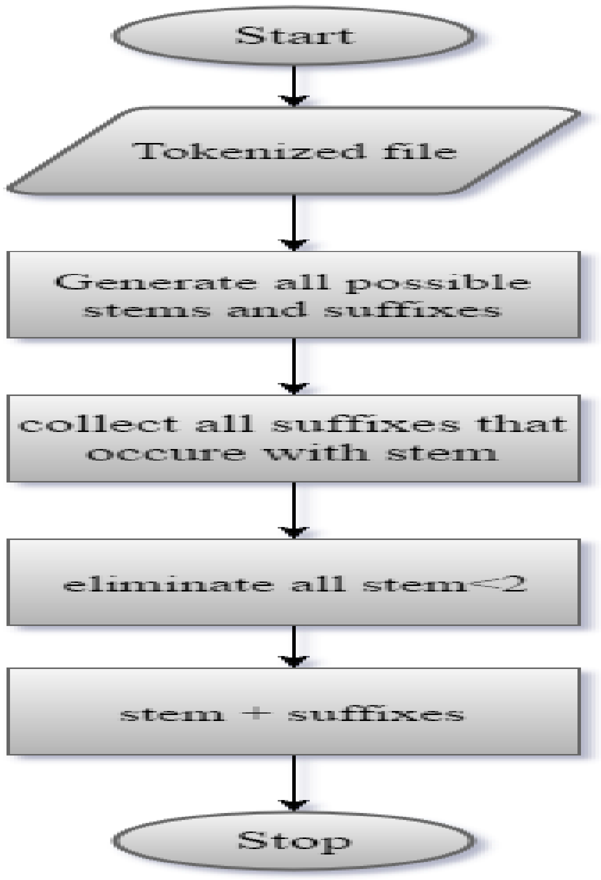
Start
Tokenized file
Generate all possible stems and suffixes
collect all suffixes that occure with stem
eliminate all stem<2
stem + suffixes
Stop

Figure1: Flow Chart of proposed Algorithm

To easily understand the algorithm, we can consider some illustrations.

IV ILLUSTRATION

The sample data set containing Kannada words is taken as shown in Figure2. The output we got when we apply the proposed algorithm is as shown in the figure3.

ಜೀವನದಲ್ಲಿ

ಜೀವನ

ಜೀವನಕ್ಕೆ

ಜೀವನಕ್ರಮ

Figure2: The sample input

ಜೀವನದಲ್: ಲಿ, ಜೀವನಕ: ್ಕೆ ್ರಮ,
ಜೀವನದಲ: ್ಲಿ

ಜೀವ: ನದಲ್ಲಿ ನಕ್ಕೆ ನಕ್ರಮ, ಜೀವನಕ್: ಕೆ ರಮ

ಜೀ: ವನದಲ್ಲಿ ವನ ವನಕ್ಕೆ ವನಕ್ರಮ, ಜೀವನ:
ದಲ್ಲಿ ಕ್ಕೆ ಕ್ರಮ

ಜೀವನದ: ಲ್ಲಿ

Figure3: Output of Stemmer for sample input

V CONCLUSION

We have proposed an unsupervised stemmer for Kannada language. Result of which can be used for POS tagging, Information Retrieval process. This stemmer can be used to any Indian language.

In future one can use Dictionary of Kannada Language to increase the accuracy of the stemmer.

REFERENCES

[1] Ananthakrishnan Ramanathan, Durgesh D Rao "A Light Weight Stemmer for Hindi". (2003)

[2] Deepa Gupta,Rahul Kumar Yadav, Nidhi sajan(2012) " Improving Unsupervised Stemming by using Partial Lemmatization Coupled with Data-based Heuristics for Hindi". International Journal of Computer Applications (0975 – 8887)

[3] Dalwadi Bijal Suthar Sanket "Overview of Stemming Algorithms for Indian and Non-Indian Languages " (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (2)

[4] Vishal Gupta "Hindi Rule Based Stemmer for Nouns" International Journal of Advanced Research in Computer Science and Software Engineering. Volume 4, Issue 1, January 2014

[5] Mohd. Shahid Husain, "An unsupervised approach to develop stemmer". International Journal on Natural Language Computing (IJNLC) Vol. 1, No.2, August 2012.