# Implementation of Graph Based Stemmer in Gujarati Language


| Kishan Vaishnani | Dhyanil Mehta | Darshil Patel | Tarang Ranpara | Sonal Nathwani |
|---|---|---|---|---|
| 202011004 | 202011032 | 202011034 | 202011057 | 202018034 |
| DAIICT-382421 | DAIICT-382421 | DAIICT-382421 | DAIICT-382421 | DAIICT-382421 |



***Abstract*—In this project we have implemented Graph Based Stemmer(GRAS) for the Gujarati language and performed the retrieval task. This is a language independent, statistical, lexicon analysis based stemming algorithm. The implementation is performed on FIRE 2011 document collection and queries. //Some analysis or evaluation statements//**



***Index Terms*—GRAS, stemmer, root words, graph, edges**


## I. Introduction

Stemming is the process of combining related words to a common root word usually refered to as stem. This process is done by removing the inflectional(-s, -ed, -ing, -er, -est) and derivational(un-, in-, dis-, non-, -ive,-al) affixes for the actual word. Stemming plays a very import role in Information Retrieval tasks as this step reduces the index size and simultaneously improves the recall as it successfully retrieves various possible forms of a word which is present in the query. This phenomenon is of utmost importance when working with any morphologically rich language where one word could take multiple forms in the language. Here, the final aim is to make sure that the words that are related to each other do fall into some similar group and map to the common stem, irrespective of whether the stem is a meaningful word in that language.[**1**]

There are various types of Stemmers that have been developed in this domain that prove to be better based on different situations. There are Rule based Stemmers, Statistical Stemmers as well as Hybrid Stemmers that make use of both these techniques. It is comparatively easy to stem an English word as compared to languages such as Hindi, Marathi, Gujarati etc. The reason for this is its morphological richness. Here, the grammatical changes are indicated by making changes to word itself by having various forms of the same word. This increases the complexity of the word. Also, if we focus more on the Gujarati language, the Devanagari script in which Gujarati is written makes it even more difficult to perform stemming.

Based on the above arguments, it is difficult to get good performance of a stemmer for a morphological rich language like Gujarati. However, based on the current state of the art models in this domain, GRAS is one such lexicon analysis based stemmer which performs well on morphological rich languages such as Bengali, Marathi, Bulgarian, French, Kurdish. This was tested by a few researchers and this is a statement made out of their experimental result.[**2**]

On the observation of the good performance of GRAS for few morphological rich languages, here we implement this Stemmer for Gujarati language and further perform retrieval task on it to check its performance.

## II. Methodology

### A. Preprocessing

We used NLTK library's[**4**] RegexTokenizer for removing the unnecessary components like english words, digits and punctuation marks. After tokenizing, we removed stop words from our tokens. We found 212 stop words for gujarati language.Some of the stop words include અથવા,અને,અમને,એમ,કઈ,કયું,કેમ,છું,જોઈએ,તમે,તું,તેવું, તો,ત્યારે,ત્યાં,થઈ,નહી,ના,મેં,શું,હતા, હુ and more.[**5**] Also we have made use the cbow word embedding text representation technique for our implementation.

### B. GRAS

**Algorithm 1:** Identify Suffix

```
Let C={W_1, W_2, ..., W_m} set of the unique tokens
for i = 1 to n do
    for any two words W_1, W_2 ∈ C do
        Output the suffix pair (s_1, s_2) such that
          W_j = r s_1. W_k = r s_2, and r is the longest
          common prefix of (W_j, W_k).
        if r ≥ l then
            Output the suffix pair
            Update the frequency of suffix pair
        end
    end
end
```

From the entire corpus, a set of unique tokens is formulated. Now, for every two words in that set, a suffix pair is generated such that the root word is common with two different suffixes. An example for this could be સમાજનો−સમાજમાં, નિયમોનો−નિયમોમાં Here નો=s1 and માં =s2 hence they are the suffix pair based on the condition r>l. r is

the longest common prefix which is સમાજ in case one and નિયમો in case two.

Here in this Algorithm 1 there is one user defined parameter l that is used while accepting or rejecting the longest common subsequence while creating the graph. The condition that is taken into consideration is : Longest common subsequence > l then it should be accepted while constructing graph.

**Algorithm 2:** Graph Processing

```
for suffix pairs in S do
    if frequency of pair ≤ alpha then
        Remove that suffix pair
    end
end
```

The Algorithm 2 explains that if the frequency of the of the suffix pair is less than or equal to a user defined parameter alpha then it is not to be considered. Logically, if a suffix pair is not occurring for more than a set threshold(alpha) number of times than it is not considered as the suffix pair and is removed from the storage as a valid suffix pair. An example to explain this could be: કારણ−કાર , તરફેણ−તરફે. Here the suffix pair is ણ−$\phi$ This pair may not occur frequently and is not a valid suffix pair.

**Algorithm 3:** Class formation

```
while G ≠ φ do
    Let u be the vertex of G with maximum degree.
    Let w(u,v) be the weight/frequency of edge
      between the vertex u and v
    S = u
    for all v ∈ Adjacent(u) taken in decreasing
      order of w(u, v) do
        if cohesion(u, v) ≥ δ then
            S = S ∪ {v}
        end
        else
            Delete the edge (u,v)
        end
    end
    Output the class S
    From G remove the vertices in S and their
      incident edges.
    Let G' be the new graph.
    G'=G
end
```

Here in the Class formation Algorithm, a word u with the maximum degree is chosen. In our algorithm we have implemented this in the form of adjacency list. The values of the list here are the edge weights which are the frequency of those suffix pairs. The class begins with the assumption that there is only one initial word u present in it. Now for all its adjacent/neighbour words it checks

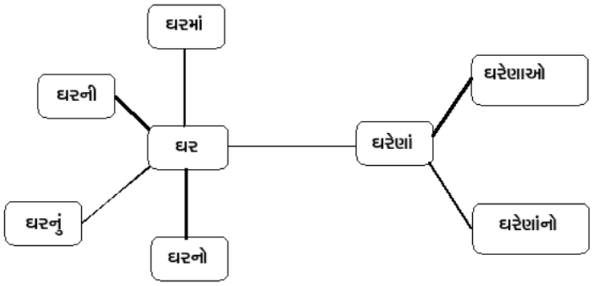


Figure 1. Representation of Class Formation Algorithm

for the value of cohesion, the formula of which is given below:

$$\text{cohesion(p,v)} = \frac{1+|Adjacent(p) \cap Adjacent(v))|}{|Adjacent(v)|}$$

There is a user defined parameter delta(which is the threshold value) against which the value of cohesion is measure. If it is greater than delta, the word is added to the class otherwise that particular edge is deleted. This process goes on untill all the words are a part of some class. And later the word u the root word or the stem for that class. The visual representation of this is given in the Figure 1.

## III. Experimental Results

### A. Dataset

The dataset that has been used for the creation of GRAS is a corpus from the FIRE(Forum for Information Retrieval and Evaluation) 2011.[**3**] The following are the features of the dataset:

1] This corpus has a total of 313,163 documents which have been curated from the Gujarat Samachar newspaper that is published in the Gujarati script.

2] There are 50 queries in the TREC format (title + desc + narr).

3] The relevance assessment document(qrels) has relevant documents for each query ranging from at least 4 to maximum 97 relevant documents.

### B. Evaluation metrics

The Evaluation metrics that we have used to check the performance is the MAP(Mean Average Precision) value. We have calculated this value with no stemming and five different variations of the user defined values of l, alpha and delta. The values for the experiments are shown in the table1.

## IV. Conclusion

In our experiment, we implemented the GRAS(Graph Based Stemmer) for Gujarati language and it definitely showed better results as compared to no stemming. However there was limitation for us in terms of computing resources which did not let us choose lower values of

Table I
MAP VALUES FOR DIFFERENT CASES

| l | alpha | delta | MAP |
|---|---|---|---|
| No Stemming | | | 0.308 |
| 6 | 4 | 0.5 | 0.356 |
| 6 | 4 | 0.9 | 0.348 |
| 6 | 6 | 0.5 | 0.347 |
| 7 | 4 | 0.5 | 0.324 |
| 7 | 4 | 0.8 | 0.333 |

'l' to perform stemming and check the results as this would result in such amount of data which would lead to exhaustion of the resources. Also, as mentioned earlier, cbow word embedding method of text representation was used for this experiment and we could not use TFIDF representation but this can be further checked with the availability of higher computational resources and the limitations of cbow can be avaoided.

## REFERENCES


[1] Patel, Pratikkumar, Kashyap Popat, and Pushpak Bhattacharyya. "Hybrid stemmer for Gujarati." In Proceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing, pp. 51-55. 2010.

[2] Singh, Jasmeet, and Vishal Gupta. "Text stemming: Approaches, applications, and challenges." ACM Computing Surveys (CSUR) 49, no. 3 (2016): 1-46.

[3] http://fire.irsi.res.in/fire/static/data

[4] https://www.nltk.org/

[5] https://github.com/quanteda/stopwords/files/2719155/A.List.of.210.Gujarati.Stop.Words.txt

[6] Paik, Jiaul H., Mandar Mitra, Swapan K. Parui, and Kalervo Järvelin. "GRAS: An effective and efficient stemming algorithm for information retrieval." ACM Transactions on Information Systems (TOIS) 29, no. 4 (2011): 1-24.