Hybrid Inflectional Stemmer and Rule-based Derivational Stemmer for Gujarati

Kartik Suba **Dipti Jiandani**
Department of Computer Engineering
Dharmsinh Desai University
suba.kartik@gmail.com
jiandani.dipti@gmail.com

Pushpak Bhattacharyya
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
pb@cse.iitb.ac.in

Abstract

In this paper we present two stemmers for Gujarati- a lightweight inflectional stemmer based on a hybrid approach and a heavyweight derivational stemmer based on a rule-based approach. Besides using a module for unsupervised learning of stems and suffixes for lightweight stemming, we have also included a module performing POS (Part Of Speech) based stemming and a module using a set of substitution rules, in order to improve the quality of these stems and suffixes. The inclusion of these modules boosted the accuracy of the inflectional stemmer by 9.6% and 12.7% respectively, helping us achieve an accuracy of 90.7%. The maximum index compression obtained for the inflectional stemmer is about 95%. On the other hand, the derivational stemmer is completely rule-based, for which, we attained an accuracy of 70.7% with the help of suffix-stripping, substitution and orthographic rules. Both these systems were developed to be useful in applications such as Information Retrieval, corpus compression, dictionary search and as pre-processing modules in other NLP problems such as WSD.

1. Introduction

Stemming is a process of conflating related words to a common stem by chopping off the inflectional and derivational endings.

Stemming plays a vital role in Information Retrieval systems by reducing the index size and increasing the recall by retrieving results that contain any of the possible forms of a word present in the query (Harman, 1991). This is especially true in case of a morphologically rich language like Gujarati.

The aim is to ensure that all the related words map to common stem, wherein, the stem may or may not be a meaningful word in the vocabulary of the language.

Current state of the art approaches to stemming can be classified into three categories, viz., rule-based, unsupervised and hybrid (Smirnov, 2008). In case of inflectional stemmer, building a completely rule-based system is non-trivial for a language like Gujarati. On the other hand, adopting a purely unsupervised approach, such as *take-all-splits* discussed in section 4, may fail to take advantage of some language phenomena, such as, the suffixes in a language like Gujarati, are separable based on their parts of speech. For example, the suffix ી (*-ī*) should be stripped off for verbs (as in case of કરી *karī* 'did'), but not for nouns (as in case of ઈમાનદારી *īmāndārī* 'honesty'). Such characteristics can be easily represented in the form of substitution rules. So, we follow a hybrid approach for the inflectional stemmer taking advantage of both rule-based and unsupervised phenomena.

However, in case of derivational stemming, words that are derived, either by adding affixes to the stems or by performing changes at the morpheme boundary, are reduced to their stem forms. To accomplish this task of derivational stemming, we have adopted a completely rule-based approach.

The remainder of this paper is organized as follows. We describe the related work in section 2. Next, section 3 explains the morphological structure of Gujarati. We describe our approach to inflectional stemmer in section 4 and to derivational stemmer in section 5. Experiments and results are presented in section 6. Section 7 concludes the paper, pointing also to future work.

2. Background and Related Work

The earliest English stemmer was developed by Julie Beth Lovins (1968). The Porter stemming algorithm (Martin Porter, 1980), which was published later, is perhaps the most widely used algorithm for stemming in case of English language. Both of these stemmers are rule-based and are best suited for less inflectional languages like English.

A lot of work has been done in the field of unsupervised learning of morphology. Goldsmith (2001) proposed an unsupervised approach for learning the morphology of a language based on the Minimum Description Length (MDL) framework which focuses on representing the data in as compact manner as possible.

Not much work has been reported for stemming for Indian languages compared to English and other European languages. The earliest work reported by Ramanathan and Rao (2003) used a hand crafted suffix list and performed longest match stripping for building a Hindi stemmer. Majumder et al. (2007) developed YASS: Yet Another Suffix Stripper which uses a clustering-based approach based on string distance measures and requires no linguistic knowledge. Pandey and Siddiqui (2008) proposed an unsupervised stemming algorithm for Hindi based on Goldsmith's (2001) approach.

Work has also been done for Gujarati. Inspired by Goldsmith (2001), a lightweight statistical stemmer was built for Gujarati (Patel et al., 2010) which gave an accuracy of 68%. But no work was done so far in the area of derivational stemming for Gujarati.

3. Gujarati Morphology

The Gujarati phoneme set consists of eight vowels and twenty-four consonants. Gujarati is rich in its morphology, which means, grammatical information is encoded by the way of affixation rather than independent free-standing morphemes.

The Gujarati nouns inflect for number (singular, plural), gender (masculine, feminine, neuter), and declension class (absolute, oblique). The absolute form of a noun is its default or uninflected form. This form is used as the object of the verb, typically when inanimate as well as in measure or temporal construction. There are seven oblique forms in Gujarati corresponding more or less to the case forms- nominative, dative, instrumental, ablative, genitive, locative and vocative. All cases, except for the vocative, are distinguished by means of postpositions.

The Gujarati adjectives are of two types – declinable and indeclinable. The declinable adjectives have the termination *-ũ* (ુ઼ં) in neuter absolute. The masculine absolute of these adjectives ends in *-o* (ો) and the feminine absolute in *-ī* (ી). For example, the adjective સારું *sārũ* 'good' takes the form સારું *sārũ*, સારો *sāro* and સારી *sārī* when used for a neuter, masculine and feminine object respectively. These adjectives agree with the noun they qualify in gender, number and case. Adjectives that do not end in *-ũ* in neuter absolute singular are classified as indeclinable and remain unaltered when affixed to a noun.

The Gujarati verbs are inflected based on a combination of gender, number, person, aspect, tense and mood. There are several postpositions in Gujarati which get bound to the nouns or verbs which they postposition. For example, *-nũ* (નું : genitive marker), *-mɜ̃* (માં : in), *-e* (ે : ergative marker), etc. These postpositions get agglutinated to nouns or verbs and do not merely follow them. For example, the phrase 'in water' is expressed in Gujarati as a single word પાણીમાં *pāṇīmɜ̃*, wherein, માં *mɜ̃* is agglutinated to the noun પાણી *pāṇī*.

We created four lists of Gujarati suffixes which contain postpositions and inflectional suffixes respectively for nouns, verbs, adjectives and adverbs for use in our approach for the inflectional stemmer. Similar lists have been used for the derivational stemmer, in the form of orthographic, suffix-stripping and substitution rules.

4. Our Approach for Inflectional Stemmer

We have been inspired by Goldsmith (2001). Goldsmith's approach was based on unsupervised learning of stems and suffixes, and he proposed a *take-all-splits* method. Besides this, we have incorporated two more modules, one performing POS-based stemming and the other doing suffix-stripping based on linguistic rules. During the training phase of our approach, the Gujarati words

extracted from EMILLE corpus[1] are used in order to learn the probable stems and suffixes. This information is used in order to stem any unseen data. We describe the approach in detail below.

4.1 Training phase

As mentioned earlier, the input to the training phase is a list of Gujarati words. During this phase, the aim is to obtain optimal split position for each word in the corpus. The optimal split position for each word is obtained by systematic traversal of various modules.

In the first module, a check is performed to see if the input word is already in its stem form. This is accomplished by using a list of stems. Besides being used in training the stemmer, this list of stems is also updated with the new stems learnt correctly at the end of training phase. For the first time that the stemmer is trained, this list is empty. If the word exists in the above mentioned list, the optimal split position will be at the end of the word with suffix as NULL.

In the second module, POS-based stemming is performed. As Gujarati does not have a POS tagger, there had to be some method to determine the POS of a word. Since we had the files which shall be used in the development of the Gujarati WordNet and since they also contained POS information, we created a set of files (hereafter referred to as POS-based files), each containing words of a specific POS. We used these files to decide the POS of the word. Also, as mentioned in section 3, we made files (hereafter referred to as suffix files), each containing suffix list for a specific POS. Thus POS-based stemming i.e., stripping of the corresponding suffixes is performed if the word is found in any of the POS-based files.

In the third module, linguistic rules are applied in order to determine the optimal split position. Each such rule is expressed as a pair of precedent and antecedent, both of which are regular expressions. If any part of the word matches any of the precedents, that part is replaced by the corresponding antecedent and the split position is returned as the length of the new word.

[1] http://www.lancs.ac.uk/fass/projects/corpus/emille/

If all the previous module checks fail, as a final resort, take-all-splits of the word is performed (see Figure 1) considering all cuts of the word of length L into stem + suffix, i.e., $w_{1,i} + w_{i+1,L}$, where $1 \leq i < L$. The ranking function that can be used to decide the optimal split position can be derived from Eqn 1.

{$stem_1$+$suffix_1$, $stem_2$+$suffix_2$, …, $stem_L$+$suffix_L$}
પાણીમાં={પ + ◌ાણીમાં, પા + ણીમાં, પાણ + ◌ીમાં, પાણી + માં, પાણીમ + ◌ાં, પાણીમા + ◌ં, પાણીમાં + NULL}

Figure 1. All possible word segmentations for the word પાણીમાં *pāṇīmã̃* ‘in_water’ which has પાણી *pāṇī* ‘water’ as its stem and માં *mã̃* ‘in’ as its suffix

The function used for finding the optimal split position must reflect the probability of a particular split since the probability of any split is determined by frequencies of the stem and suffix generated by that split. Hence, probability of a split can be given by Eqn 1 below.

$$P(Split_i) = P(stem = w_{1,i}) * P(suffix = w_{i+1,L}) \quad \text{(Eqn 1)}$$

i: split position (varies from 1 to L)
L: length of the word

Taking *log* on both sides of Eqn 1 and ignoring the constant terms, we get,

$$log(P(Split_i)) = log(freq(stem)) + log(freq(suffix)) \quad \text{(Eqn 2)}$$

The frequency of shorter stems and suffixes is very high when compared to the slightly longer ones. Thus, Eqn 3 is obtained from Eqn 2 by introducing the multipliers i (length of stem) and L-i (length of suffix) in the function in order to compensate for this disparity.

$$f(i) = i * log(freq(stem)) + (L-i) * log(freq(suffix)) \quad \text{(Eqn 3)}$$

Finally, a split position which maximizes the ranking function given by Eqn 3 is chosen as the optimal split position. Once the optimal split of any word is obtained, the frequencies of the stem and the suffix generated by that

split are updated. The word list is then iterated and the optimal split position is recomputed until the optimal split positions of all the words do not change any more. The training phase was observed to take four iterations typically. At the end of the training phase, a list of stems and suffixes along with their frequencies is obtained. A list of signatures (see Figure 2) is also obtained, where a signature is a data-structure that provides a mapping between the stem and the suffixes with which that stem appears in the corpus. This list of signatures provides a compact representation of the corpus and can be used in case of a need to retrieve the original corpus.

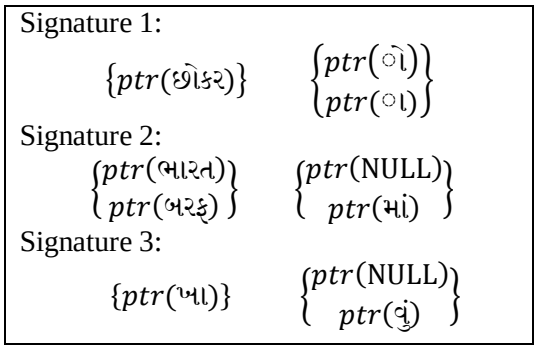

Figure 2. A sample signature-list for the words - છોકરો *chokro* 'boy', છોકરા *chokrā* 'boys', ભારત *bhārat* 'India', ભારતમાં *bhāratmã̃* 'in_India', બરફ *baraf* 'ice', બરફમાં *barafmã̃* 'in_ice', ખા *khā* 'eat', ખાવું *khāvũ* 'to_eat'

Based on the approach discussed above, an overview of the training algorithm is shown in Figure 3 below.

Step 1. Check if the word is already in its stem form, if yes, return it as it is, else proceed to Step 2.
Step 2. Check if the word is in any POS-based file, if yes, perform POS-based stemming and return, else proceed to Step 3.
Step 3. Check if a match occurs with any of the linguistic rules, if yes, apply the rule and return, else proceed to Step 4.
Step 4. Perform take-all-splits on the word and obtain the optimal split position based on Eqn 3.
Step 5. Perform Step 4 through several iterations until optimal split position of all the words remain unchanged.

Figure 3. Overview of training algorithm

4.2 Stemming of any unknown word

For the stemming of any unknown word, a similar set of steps is followed as in the training phase, with the only change in the take-all-splits module, wherein, for any given word, the function given by Eqn 3 is evaluated for each possible split using the frequencies of the stems and the suffixes learnt during the training phase.

Consider that the words કરવું *karvũ* 'to_do', કરીને *karīne* 'after_doing' and કરીશ *karīsh* 'will_do' existed in the training set, then the frequency of the stem કર *kar* 'do' will be high. Now if the unknown word કરવાથી *karvāthī* 'by_doing' appears in the test set, it will be stemmed as કર + વાથી due to the frequencies learnt during training. In contrast to this, if the training set contained the words પાણીમાં *pāṇīmã̃* 'in_water' and ઘરમાં *gharmã̃* 'in_house', the unknown word ટોપીમાં *ṭopīmã̃* 'in_hat' will be split as ટોપી + માં, due to the high frequency of the suffix માં *mã̃* 'in' learnt during training.

5. Our Approach for Derivational Stemmer

Derivation is a process of combining a word stem with grammatical morphemes usually resulting in a word of different class, not necessarily different POS. Derivational morphology deals with derivation of the words either by affixation (For e.g., જવાબદારી *javābdārī* 'responsibility' derived from જવાબદાર *javābdār* 'responsible') or by performing changes at the morpheme boundary (For e.g., ધાર્મિક *dhārmik* 'religious' derived from ધર્મ *dhārm* 'religion').

The task of derivational stemming is that of reducing the derived word to its derivational stem form. The approach for derivational stemming is inspired from the chapter on morphology by Jurafsky and Martin (2009).

Their approach consisted of the following components. However, only two of them were useful in our case.

1. Lexicon: It is a list of stems and suffixes together with some basic information such as POS. The importance of a lexicon is to determine whether the resultant stem is correct or not. But, as there is no

lexicon for Gujarati, the validation of the stem form cannot be accomplished.

2. Morph-tactics: It is a model that explains morpheme ordering i.e., it explains which class of morphemes can follow which other class of morphemes.
   E.g.: બારીમાંથી *bārīmā̃thī* 'from_window' indicates that થી *thī* can follow માં *mā̃* but the other way round is not possible.
   In order to model morph-tactics, Finite State Automata (FSA) accepting different transitions within words are usually used.
3. Orthographic or spelling rules: These are the rules used to handle changes in the words at the morpheme boundary.
   E.g.: ખવડાવવું *khavḍāvvũ* 'to_make_eat' has its stem as ખા *khā* 'eat', but there is no direct way to reflect this transition. So there is a need of spelling or orthographic rule for such words. Example of such a rule is: વડાવ → ◌ા. The way it is applicable in the system is discussed after the algorithm. We have 73 such hand-crafted rules.

The algorithm steps are shown in Figure 4.

Step 1.	Check if any of the orthographic rules match, if yes, apply the rule and proceed, else proceed to step 2.
Step 2.	Check if any substitution rule is matched, if yes, apply the rule and proceed, else proceed to step 3.
Step 3.	Check if any suffix-stripping rule is matched, if yes, apply the rule and proceed, else proceed to step 4.
Step 4.	Check if the resultant word gets accepted by any FSA, if yes, return the word as the stem, else return the word obtained from the previous module as the stem.

Figure 4. Derivational stemming algorithm

For example, the word ખવડાવવું *khavḍāvvũ* 'to_make_eat' is to be stemmed. In the first step, an orthographic rule matches, which specifies that, if ડાવ appears between વ and વું, વડાવ *vḍāv* should be replaced by ◌ા *ā*, resulting into the intermediate form ખાવું *khāvũ* 'to_eat'. Next, step 2 is not applicable. In step 3, the suffix વું *vũ* is a valid suffix for verbs; hence it is stripped off; resulting into ખા *khā* 'eat', which gets accepted by the FSA for verbs in the final step. Thus, ખા *khā* 'eat' is returned as the derivational stem of ખવડાવવું *khavḍāvvũ* 'to_make_eat'.

6. Experiments and Results

We performed various experiments to evaluate the performance of both the inflectional and derivational stemmer using EMILLE Corpus for Gujarati. We extracted around ten million words from the corpus. We obtained 8,525,649 words after filtering out the wrongly spelt words. In order to create the test set, each time we randomly extracted thousand words from the corpus.

6.1 Performance of the inflectional stemmer

The performance of the inflectional stemmer is evaluated based on three factors. The first factor is the accuracy based on the gold standard data, where the gold standard data contains the ideal stems of all the words in the test set manually tagged by us. Accuracy is defined as the percentage of words stemmed correctly. The second factor is the Index Compression Factor (Fox and Frakes, 2003) that shows the extent to which a collection of words is reduced by stemming. ICF is defined as the ratio of difference in number of unique words and number of unique stems to the number of unique words. Finally, the third factor is mean number of words per signature (MW_c) (Fox and Frakes, 2003) that indicates the strength of the stemmer. MW_c is defined as the ratio of the number of unique words to the number of unique stems.

The experiments were aimed at studying the impact of three heuristics: (i) fixing the minimum permissible stem size, (ii) provide unequal weightage to the stem and suffix and (iii) introduce a threshold as a restriction on the minimum number of stems and suffixes to qualify as a signature, known as the stem filter threshold and the suffix filter threshold respectively.

Various experiments were done to study the impact of different combination of these heuristics. This impact is studied in terms of comparison of various factors as discussed above. The results of such experiments are described in the following subsections.

Varying Minimum Stem Size:

Minimum stem size was varied from 1 to 7 and its impact was observed on performance of the lightweight stemmer. The results of this experiment are shown in Table 1.

Min Stem Size	Accuracy (%)	ICF	MW_c
1	**90.7**	0.53	2.11
2	89.9	0.53	2.11
3	84.8	0.52	2.00
4	74.2	0.49	1.90
5	63.5	0.47	1.92
6	52.1	0.49	1.96
7	44.6	0.55	2.22

Table 1. Effect of minimum stem size on performance of the inflectional stemmer

It can be observed that maximum accuracy of 90.7% is obtained by neglecting the restriction on the minimum stem size and the average index compression is 52% which is considerable as far as IR application is concerned.

The results also show that the performance degrades if a restriction is placed on the minimum stem size. The reason may be that when the minimum stem size is increased lots of genuine, but small stems are neglected, leading to a decline in accuracy.

Providing unequal weightage to stem and suffix along-with minimum stem size:

Initially an equal weightage was provided to stem and suffix in Eqn 3 which is responsible for determining the optimal split position of any word. Then Eqn 4 was obtained from Eqn 3 by introducing a parameter 'α' in order to provide unequal weightage to stem and suffix and its effect was observed on performance of the lightweight stemmer.

We used Eqn 4 and varied α along-with varying the minimum stem size. The results are shown in Table 2.

$$f(i) = \alpha * i * \log(freq(stem)) + (1 - \alpha) * (L-i) * \log(freq(suffix)) \quad \text{(Eqn 4)}$$

Min Stem Size	α	Accuracy (%)	ICF	MW_c
1	0.3	90.0	0.51	2.04
	0.5	**90.7**	0.53	2.11
	0.7	87.0	0.51	2.04
2	0.3	89.2	0.51	2.08
	0.5	89.9	0.53	2.11
	0.7	86.6	0.51	2.04
3	0.3	84.7	0.51	2.05
	0.5	84.8	0.52	2.00
	0.7	82.9	0.50	2.03
4	0.3	74.0	0.49	1.96
	0.5	74.2	0.49	1.90
	0.7	73.2	0.48	1.95
5	0.3	63.2	0.46	1.88
	0.5	63.5	0.47	1.92
	0.7	62.5	0.47	1.90

Table 2. Effect of α along with min. stem size on performance of the inflectional stemmer

It can be observed that the maximum accuracy of 90.7% is obtained by neglecting the restriction on the minimum stem size and providing equal weightage to stem and suffix by keeping α = 0.5. Even for this combination of heuristics, the average index compression of 52% is obtained.

Introducing restriction on the number of stems and suffixes to qualify as a signature:

A restriction was placed on the minimum number of stems and the minimum number of suffixes needed in a signature. These numbers are called stem filter threshold and suffix filter threshold respectively.

We varied all the parameters, viz., minimum stem size, α, stem filter threshold and suffix filter threshold. There were two important observations that will be stated below. The results of this experiment are shown in Table 3 below.

The results show how this combination of heuristics improves the quality of stems and suffixes, as well it brings big boost in the Index Compression Factor.

Min Stem Size	α	Thres-hold	Accu-racy (%)	ICF	MW_c
1	0.3	0	90.0	0.51	2.0
		1	85.8	0.88	9.0
		2	87.1	**0.95**	20.3
1	0.5	0	**90.7**	0.52	2.1
		1	88.3	0.89	9.9
		2	87.7	**0.95**	22.4
1	0.7	0	87.0	0.51	2.0
		1	84.9	0.95	22.2
		2	84.8	**0.95**	22.2
2	0.3	0	89.2	0.51	2.1
		1	85.1	0.88	9.0
		2	86.5	**0.95**	20.3
2	0.5	0	89.9	0.52	2.0
		1	87.6	0.89	9.9
		2	86.7	**0.95**	22.4
2	0.7	0	86.6	0.51	2.0
		1	87.6	0.94	19.2
		2	84.1	**0.95**	22.2

Table 3. Effect of varying all three parameters, viz., min. stem size, α and filter threshold on performance of the inflectional stemmer

It can be observed that the maximum accuracy of 90.7% is obtained by neglecting the restriction on the minimum stem size, providing equal weightage to stem and suffix by keeping $\alpha = 0.5$ and ignoring the restriction on the minimum number of stems and suffixes to form a signature.

Another important observation in this experiment was that by restricting the filter threshold to two, we obtain the highest index compression of 95% with a slight decrease in accuracy. This is an excellent result for applications like corpus compression.

6.2 Performance of the derivational stemmer

The performance of the derivational stemmer was evaluated by direct comparison of the stems generated by the system with the ideal stems present in the gold standard data which gave an accuracy of 70.7%.

7. Conclusions and Future Work

We developed two systems for Gujarati language, one performing inflectional stemming and the other performing derivational stemming.

The inflectional stemmer has an average accuracy of about 90.7% which is considerable as far as IR is concerned. Boost in accuracy due to POS based stemming was 9.6% and due to inclusion of the language characteristics it was further boosted by 12.7%. Heuristic with filter threshold set to 2 gives highest index compression of 95% which is extremely good for applications like compression of data.

The derivational stemmer has an average accuracy of 70.7% which can act as a good baseline and can be useful in tasks such as dictionary search or data compression.

The systems possess potential to be used as pre-processing modules for NLP problems other than IR, such as Word Sense Disambiguation, similarity measure, etc.

The limitations of inflectional stemmer can be easily overcome if modules like Named Entity Recognizer are integrated with the system.

In order to elevate the accuracy of the derivational stemmer, the list of substitution, orthographic or suffix-stripping rules can be improved further if needed.

References

Amaresh K. Pandey and Tanveer J. Siddiqui. 2008. An unsupervised Hindi stemmer with heuristic improvements. *Proceedings of the Second Workshop on Analytics for Noisy Unstructured Text Data*, 303:99-105.

Ananthakrishnan Ramanathan and Durgesh D. Rao. 2003. A Lightweight Stemmer for Hindi. *Workshop on Computational Linguistics for South-Asian Languages*, EACL.

Christopher J. Fox and William B. Frakes. 2003. Strength and Similarity of Affix Removal Stemming Algorithms. *Special Interest Group on Information Retrieval Forum*, 37(1):26-30.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. 2nd edition. Prentice-Hall, Englewood Cliffs, NJ.

Donna Harman. 1991. How effective is suffixing? *Journal of the American Society for Information Science*, 42(1):7-15.

Ilia Smirnov. 2008. Overview of Stemming Algorithms. *Mechanical Translation*.

John A. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153-198.

Julie B. Lovins. 1968. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22-31.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130-137.

Prasenjit Majumder, Mandar Mitra, Swapan K. Parui, Gobinda Kole, Pabitra Mitra, and Kalyankumar Datta. 2007. YASS: Yet another suffix stripper. *Association for Computing Machinery Transactions on Information Systems*, 25(4):18-38.

Pratikkumar Patel, Kashyap Popat and Pushpak Bhattacharyya. 2010. Hybrid Stemmer for Gujarati. *Proceedings of the 1st Workshop on South and Southeast Asian Natural Languages Processing (WSSANLP), the 23rd International Conference on Computational Linguistics (COLING), Beijing*, 51-55.

William St. Clair Tisdall. 1892. *A simplified grammar of the Gujarati language: together with A short reading book and vocabulary*. D. B. Taraporevala Sons & Company, Bombay