Hybrid Stemmer for Gujarati

Pratikkumar Patel **Kashyap Popat**
Department of Computer Engineering
Dharmsinh Desai University
pratikpat88@gmail.com
kan.pop@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 a lightweight stemmer for Gujarati using a hybrid approach. Instead of using a completely unsupervised approach, we have harnessed linguistic knowledge in the form of a hand-crafted Gujarati suffix list in order to improve the quality of the stems and suffixes learnt during the training phase. We used the EMILLE corpus for training and evaluating the stemmer's performance. The use of hand-crafted suffixes boosted the accuracy of our stemmer by about 17% and helped us achieve an accuracy of 67.86 %.

1 Introduction

Stemming is the process of conflating related words to a common stem by chopping off the inflectional and derivational endings. Stemming plays an important role in Information Retrieval (IR) systems by reducing the index size and increasing the recall by retrieving results containing any of the various possible forms of a word present in the query. This is especially true in case of a morphologically rich language like Gujarati, where a single word may take many forms. The aim is to ensure that related words map to common stem, irrespective of whether or not the stem is 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. Building a rule based stemmer for a morphologically rich language is an uphill task considering the different inflectional and morphological variations possible. Purely unsupervised approaches on the other hand fail to take advantage of some language phenomenon which can be easily expressed by simple rules. We thus follow a hybrid approach by enhancing an unsupervised system with a list of hand-crafted Gujarati suffixes.

The remainder of this paper is organized as follows. We describe related work in section 2. Section 3 explains the morphological structure of Gujarati. We describe our approach in section 4. The experiments and results are described in section 5. Section 6 concludes the paper highlighting the future work.

2 Background and Related Work

The earliest English stemmer was developed by Julie Beth Lovins in 1968. The Porter stemming algorithm (Martin Porter, 1980), which was published later, is perhaps the most widely used algorithm for English stemming. 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, 2006) proposed an unsupervised algorithm 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. Creutz (2005, 2007) uses probabilistic maximum a posteriori (MAP) formulation for unsupervised morpheme segmentation.

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. They concluded that stemming improves recall of IR systems for Indian languages like Bengali. Dasgupta and Ng (2007) worked on unsupervised morphological parsing for Bengali. Pandey and Siddiqui (2008) proposed an unsupervised stemming algorithm for Hindi based on Goldsmith's (2001) approach.

Unlike previous approaches for Indian languages which are either rule based or completely unsupervised, we propose a hybrid approach which harnesses linguistic knowledge in the form of a hand-crafted suffix list.

3 Gujarati Morphology

Gujarati has three genders (masculine, neuter and feminine), two numbers (singular and plural) and three cases (nominative, oblique/vocative and locative) for nouns. The gender of a noun is determined either by its meaning or by its termination. The nouns get inflected on the basis of the word ending, number and case.
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. The 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 upon 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. e.g. *-nũ* (નું : genitive marker), *-mā̃* (માં : in), *-e* (◌ે : ergative marker), etc. These postpositions get agglutinated to the nouns or verbs and not merely follow them.

We created a list of hand crafted Gujarati suffixes which contains the postpositions and the inflectional suffixes for nouns, adjectives and verbs for use in our approach.

4 Our Approach

Our approach is based on Goldsmith's (2001) take-all-splits method. Goldsmith's method was purely unsupervised, but we have used a list of hand crafted Gujarati suffixes in our approach to learn a better set of stems and suffixes during the training phase. In our approach, we make use of a list of Gujarati words extracted from EMILLE corpus for the purpose of learning the probable stems and suffixes for Gujarati during the training phase. This set of stems and suffixes will be used for stemming any word provided to the stemmer. We have described the details of our approach below.

4.1 Training Phase

During the training phase, we try to obtain the optimal split position for each word present in the Gujarati word list provided for training. We obtain the optimal split for any word by taking all possible splits of the word (see Figure 1) and choosing the split which maximizes the function given in Eqn 1 as the optimal split position. The suffix corresponding to the optimal split position is verified against the list of 59 Gujarati suffixes created by us. If it cannot be generated by agglutination of the hand crafted suffixes, then the length of the word is chosen as the optimal split position. i.e. the entire word is treated as a stem with no suffix.

$\{stem_1+suffix_1, stem_2+suffix_2, \ldots, stem_L+suffix_L\}$

ઘરના= {ઘ + રના, ઘર + ના, ઘરન + ◌ા,ઘરના + NULL}

Figure 1. All Possible Word Segmentations

$$f(i) = i*log(freq(stem_i)) + (L-i)*log(freq(suffix_i)) \qquad \text{(Eqn 1)}$$

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

The function used for finding the optimal split position reflects the probability of a particular split since the probability of any split is determined by the frequencies of the stem and suffix generated by that split. The frequency of shorter stems and suffixes is very high when compared to the slightly longer ones. Thus the multipliers *i* (length of stem$_i$) and *L-i* (length of suffix$_i$) have been introduced in the function in order to compensate for this disparity.

Once we obtain the optimal split of any word, we update the frequencies of the stem and suffix generated by that split. We iterate over the word list and re-compute the optimal split position until the optimal split positions of all the words remain unchanged. The training phase was observed to take three iterations typically.

4.2 Signatures

After the training phase, we have a list of stems and suffixes along with their frequencies. We use this list to create signatures. As shown in Figure 2, each signature contains a list of stems and a list of suffixes appearing with these stems.

Stems	Suffixes
પશુ (*pashu* - animal)	ના (*nā*)
જંગ (*jang* - war)	નો (*no*)
	ને (*ne*)
	નું (*nũ*)
	ની (*nī*)

Figure 2. Sample Signature

The signatures which contain very few stems or very few suffixes may not be useful in stemming of unknown words, thus we eliminate the signatures containing at most one stem or at most one suffix. The stems and suffixes in the remaining signatures will be used to stem new words. An overview of the training algorithm is shown in Figure 3.

```
Step 1: Obtain the optimal split position for each
        word in the word list provided for training
        using Eqn 1 and the list of hand crafted suf-
        fixes

Step 2: Repeat Step 1 until the optimal split posi-
        tions of all the words remain unchanged

Step 3: Generate signatures using the stems and
        suffixes generated from the training phase

Step 4: Discard the signatures which contain either
        only one stem or only one suffix
```

Figure 3. Overview of training algorithm

4.3 Stemming of any unknown word

For stemming of any word given to the stemmer, we evaluate the function in Eqn 1 for each possible split using the frequencies of stems and suffixes obtained from the training process. The word is stemmed at the position for which the value of the function is maximum.

5 Experiments and Result

We performed various experiments to evaluate the performance of the stemmer using EMILLE Corpus for Gujarati. We extracted around ten million words from the corpus. These words also contained Gujarati transliterations of English words. We tried to filter out these words by using a Gujarati to English transliteration engine and an English dictionary. We obtained 8,525,649 words after this filtering process.

We have used five-fold cross validation for evaluating the performance. We divided the extracted words into five equal parts of which four were used for training and one for testing. In order to create gold standard data, we extracted thousand words from the corpus randomly and tagged the ideal stem for these words manually.

For each of the five test sets, we measured the accuracy of stemming the words which are present in the test set as well as gold standard data. Accuracy is defined as the percentage of words stemmed correctly.

The experiments were aimed at studying the impact of (i) using a hand-crafted suffix list, (ii) fixing the minimum permissible stem size and (iii) provide unequal weightage to the stem and suffix for deciding the optimal split position. Various results based on these experiments are described in the following subsections.

5.1 Varying Minimum Stem Size

We varied the minimum stem size from one to six and observed its impact on the system performance. We performed the experiment with and without using the hand-crafted suffix list. The results of this experiment are shown in Table 1 and Figure 4.

The results of this experiment clearly indicate that there is a large improvement in the performance of the stemmer with the use of hand-crafted suffixes and the performance degrades if we keep a restriction on the minimum stem size. For higher values of minimum stem size, all the valid stems which are shorter than the minimum stem size do not get generated leading to a decline in accuracy.

Min Stem Size	Accuracy	
	With hand-crafted suffixes	Without hand-crafted suffix-es
1	**67.86 %**	50.04 %
2	67.70 %	49.80 %
3	66.43 %	49.60 %
4	59.46 %	46.35 %
5	51.65 %	41.22 %
6	43.81 %	36.89 %

Table 1. Effect of use of hand-crafted suffixes and fixing min. stem size on stemmer's performance

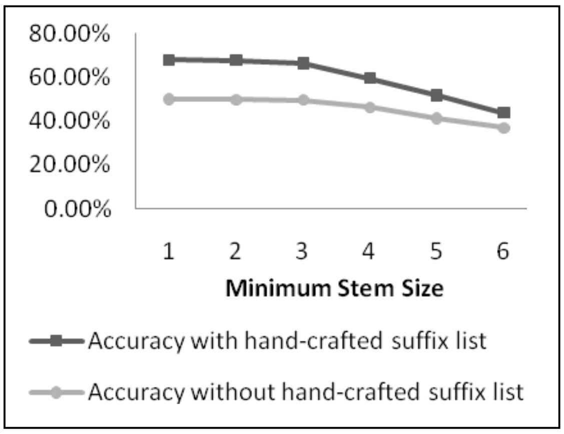

Figure 4. Variation stemmer's accuracy with the variation in min. stem size

There are several spurious suffixes which get generated during the training phase and degrade the performance of the stemmer when we don't use the hand-crafted suffix list. e.g. 'ક' is not a valid inflectional Gujarati suffix but it does get generated if we don't use the hand-crafted suffix list due to words such as 'અનેક' (*anek* - many) and 'અને' (*ane* - and). A simple validation of the suffixes generated during training against the hand-crafted suffix list leads to learning of better suffixes and in turn better stems during the training phase thereby improving the system's performance.

Thus we decided to make use of the hand-crafted suffix list during training phase and not to put any restriction on the minimum stem size.

5.2 Providing unequal weightage to stem and suffix

We have provided equal weightage to stem and suffix in Eqn 1 which is responsible for determining the optimal split position of any word. We obtained Eqn 2 from Eqn 1 by introducing a parameter 'α' in order to provide unequal weightage to the stem and suffix and observe its effect on system performance. We used Eqn 2 instead of Eqn 1 and varied α from 0.1 to 0.9 in this experiment. The results of this experiment are shown in Table 2.

$$f(i) = \alpha * i * \log(freq(stem_i)) + (1-\alpha) * (L-i) * \log(freq(suffix_i)) \qquad \text{(Eqn 2)}$$

α	Accuracy
0.1	53.52 %
0.2	61.71 %
0.3	65.43 %
0.4	67.30 %
0.5	67.86 %
0.6	67.48 %
0.7	67.49 %
0.8	67.72 %
0.9	66.45 %

Table 2. Effect of α on the stemmer's performance

The accuracy was found to be maximum when value of α was fixed to 0.5 i.e. stem and suffix were given equal weightage for determining the optimal split of any word.

6 Conclusion and Future Work

We developed a lightweight stemmer for Gujarati using a hybrid approach which has an accuracy of 67.86 %. We observed that use of a hand-crafted Gujarati suffix list boosts the accuracy by about 17 %. We also found that fixing the minimum stem size and providing unequal weightage to stem and suffix degrades the performance of the system.

Our stemmer is lightweight and removes only the inflectional endings as we have developed it for use in IR system. The list of hand-crafted suffixes can be extended to include derivational suffixes for performing full fledged stemming which may be required in applications such as displaying words in a user interface.

We have measured the performance of the stemmer in terms of accuracy as of now. We plan to evaluate the stemmer in terms of the index compression achieved and the impact on precision and recall of Gujarati IR system.

References

Creutz, Mathis, and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0.* Technical Report A81, Publications in Computer and Information Science, Helsinki University of Technology.

Creutz, Mathis, and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *Association for Computing Machinery Transactions on Speech and Language Processing,* 4(1):1-34.

Dasgupta, Sajib, and Vincent Ng. 2006. Unsupervised Morphological Parsing of Bengali. *Language Resources and Evaluation*, 40(3-4):311-330.

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

Goldsmith, John A. 2006. An algorithm for the unsupervised learning of morphology. *Natural Language Engineering*, 12(4):353-371

Jurafsky, Daniel, 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.

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

Majumder, Prasenjit, 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.

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

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

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

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

The EMILLE Corpus,
http://www.lancs.ac.uk/fass/projects/corpus/emille/