# Developing Morphological Analysers for South Asian Languages: Experimenting with the Hindi and Gujarati Languages

**Niraj Aswani, Robert Gaizauskas**

Department of Computer Science
University of Sheffield
Regent Court, Sheffield, S1 4DP, UK
n.aswani@dcs.shef.ac.uk, r.gaizauskas@dcs.shef.ac.uk

**Abstract**

A considerable amount of work has been put into development of stemmers and morphological analysers. The majority of these approaches use hand-crafted suffix-replacement rules but a few try to discover such rules from corpora. While most of the approaches remove or replace suffixes, there are examples of derivational stemmers which are based on prefixes as well. In this paper we present a rule-based morphological analyser. We propose an approach that takes both prefixes as well as suffixes into account. Given a corpus and a dictionary, our method can be used to obtain a set of suffix-replacement rules for deriving an inflected word's root form. We developed an approach for the Hindi language but show that the approach is portable, at least to related languages, by adapting it to the Gujarati language. Given that the entire process of developing such a ruleset is simple and fast, our approach can be used for rapid development of morphological analysers and yet it can obtain competitive results with analysers built relying on human authored rules.

## 1. Introduction

A considerable amount of work has been put into development of stemmers and morphological analysers (Lovins, 1968; Porter, 1980; Majumder et al., 2007; Krovetz, 1993; Hull, 1996; Shrivastava et al., 2005; Ramanathan and Rao, 2003; Pandey and Siddiqui, 2008). The majority of these approaches use hand-crafted suffix-replacement rules but a few try to discover such rules from corpora. While all these approaches remove or replace suffixes, approaches such as Krovetz (1993) and Hull (1996) propose derivational stemmers which are based on prefixes as well.

In this paper we present a rule-based morphological analyser where the rules are acquired semi-automatically from corpora. We propose an approach that takes both prefixes as well suffixes into account. Given an inflected Hindi word, our system returns its root form. It uses a dictionary, and a monolingual corpus to obtain suffix-replacement rules. Most rules in our approach are learnt automatically. However, a few of them need to be verified manually. Our algorithm is a part of our efforts on developing a general framework for text alignment (Aswani and Gaizauskas, 2009).

In the following sections, we first look at some of the well-known methods used for stemming and morphological analysis. Then, we describe our morphological analyser for the Hindi language. Finally, in order to demonstrate our approach's portability to other similar languages, we present our experiments for the Gujarati language.

## 2. Related Work

Porter's stemmer (Porter, 1980) is amongst the most cited stemmers in the literature. His stemmer is based on predefined hand crafted suffix-replacement rules. His algorithm proceeds through a fixed succession of five stages, with a different set of rules defined under each section. The stemmer uses an explicit list of suffixes, and, with each suffix, the criterion under which it may be removed from a word to form a valid stem.

Majumder et al. (2007) present a system called YASS (*Yet Another Suffix Stripper*) that uses a corpus to learn suffix stripping rules. They define a set of string distance measures for clustering the related words. Their main intuition is to reward long matching prefixes and to penalize early mismatch. Using these metrics they decide whether the words under consideration belong to the same cluster or not. They match the longest common prefix to cluster similar words and stem them to the central word in that cluster to infer suffix-replacement rules.

Unlike other approaches, Hull (1996) proposes a derivational stemmer which not only removes suffixes but prefixes as well. For example, `superconduction` is stemmed to the word `conduct`. From their experiments with derivational process, Hull (1996) shows that it is a bad idea to remove prefix from words which in most cases resulted in over-stemming. On the other hand, Krovetz (1993) reports that the derivational stemmers perform slightly better than inflectional stemmer.

Xu and Croft (1998) suggest that the rules of stemming algorithms should be modified prior to its use on any corpus and the corpus itself should be used for this purpose. They suggest that this step is necessary as the rules learnt from one corpus might not reflect the language used in other corpora. They propose to use an aggressive rule based stemmer or a language independent (n-gram) stemmer to create equivalent classes and refine them by using corpus co-occurrence statistics. They assume that the word variants that should be conflated occur in the same text windows. They propose to use this method for refining suffix-replacement rules.

Ramanathan and Rao (2003) are said to have reported the first work on Indian language stemming by developing a lightweight stemmer for the Hindi language. Their stemmer is based on some hand-crafted rules which they claimed to have derived after careful observations of the Hindi language. Based on their analysis of verbs, nouns, adjectives

| Verbs | %* | Nouns | %* | Adjectives | %* | Adverbs | %* |
|---|---|---|---|---|---|---|---|
| ा(*aa*) | 99.81 | ा(*aa*) | 19.99 | त (*ta*) | 17.06 | र (*ra*) | 14.29 |
| ना (*naa*) | 99.71 | ी(*ee*) | 16.62 | ी(*ee*) | 14.89 | ा(*aa*) | 13.21 |
| ाना (*aanaa*) | 44.21 | र (*ra*) | 9.15 | ा(*aa*) | 10.76 | : | 11.52 |
| वाना (*vaanaa*) | 12.19 | न (*na*) | 8.06 | य (*ya*) | 9.20 | त :(*taha*) | 10.09 |
| कना (*kanaa*) | 8.03 | - | - | क (*ka*) | 7.87 | क (*ka*) | 8.13 |
| राना (*raanaa*) | 7.98 | - | - | ित (*ita*) | 8.74 | त (*ta*) | 6.43 |
| लना (*lanaa*) | 5.03 | - | - | न (*na*) | 6.50 | - | - |
| - | - | - | - | रि (*ira*) | 6.90 | - | - |

∗ percentage of number of words that have the listed suffix in that category.

Table 1: Most Frequent Suffixes for Hindi Base Forms

and adverbs they produced a list of suffixes for each of these categories. Their system conflates terms by stripping off word endings from the suffix list on a *longest match* basis.

Pandey and Siddiqui (2008) use some heuristic rules which are derived using a split-all method. From their experiments with the Hindi language, they automatically obtained a list of 51 suffixes from a set of Hindi documents from the EMILLE corpus. They stripe off the longest suffixes to perform stemming. In their method, words are split to give n-gram `(n=1..l)` suffix where `l` is the length of a word under consideration. Having obtained a suffix and stem for each word, they calculate suffix and stem probabilities over the corpus and multiply them to obtain split probability. They use the highest split probability to define segments and group words accordingly. For example, the words such as लडका (*ladakaa, a boy*), लडकों (*ladakon, boys*), and लडके (*ladake, boys*) with common stem लडक (*ladak*) are grouped together with suffixes ा (*aa*), ों (*on*) and े (*e*). They also apply heuristic rules that attempt to concatenate stems with common prefix of suffixes. For example, आवश् (*aavash*) is a stem with three different suffixes यक (*yak*), यक्ता (*yaktaa*) and यक्तानुसार (*yaktaanusaar*). They obtain a common prefix string from these suffixes (i.e. यक (*yak*) and concatenate it with the stem to make it आवश्यक (*aavashyak, necessary*). They performed experiments on two different datasets and report accuracies as 85.685 for the first run and 89.979 for the second run.

To our knowledge, the morphological analyser presented by Shrivastava et al. (2005) is the best available for the Hindi language. They use a hand-crafted set of 86 suffix-replacement rules. These suffix-replacement rules are applied to an inflected word and the resulting candidate root form is checked against a list of root forms (RFList) obtained from Hindi Wordnet. If it is found, it is deemed to be the correct root form. They report 100% precision for their system.

## 3. Our Approach

We adapted[1] the ruleset presented by Shrivastava et al. (2005) and applied it on our test data[2] (DATASET1) and obtained P=0.74, R=0.68, F=0.71. The system could not find any applicable rule for 644 words. One of the reasons (as highlighted by the authors) is that not all words in the dataset were present in the RFList. Secondly, our dataset includes some words which are not exactly the proper Hindi words. For example, ऑथोरिटियाँ (*authoritiyaan, authorities*)). Finally, there were some inflected words which could not be matched with any of the rules in their ruleset.

The approach presented in this paper is a combination of various methods described under the related work (Pandey and Siddiqui, 2008; Hull, 1996; Majumder et al., 2007; Shrivastava et al., 2005). In Hindi, most base-form verbs end with suffix ना (*naa*). Every noun in Hindi can be categorized as a masculine or a feminine noun. While, most masculine nouns end with the vowel ा (*aa*), most feminine nouns end with the vowel ी (*ee*). Usually when using these words in sentences, these base-form suffixes are removed and the words are inflected based on varying criteria. Thus, finding a root form for a given Hindi word involves removing inflections and attaching relevant base-form suffixes. We used a Hindi dictionary and added all missing root forms in the RFList (called the "extended RFList" hereafter).

As all the entries in the RFList are in their base forms, the RFList can be used for obtaining common base form endings. In the RFList, words are divided in four different grammatical categories: noun, verb, adjective and adverb. For the words in each category, we obtained a list of common suffixes (see table 1). We used this list to identify missing words which appear in the corpus but not in the RFList. Considering one suffix at a time, we obtained words from a subset of the EMILLE corpus that end with this suffix. Each of these words was automatically checked against the RFList. If it did not exist, it was manually checked and added to the RFList. The words which made to the RFList were excluded from the candidates list for the next suffix. This resulted in a discovery of 442 new base forms (mostly verbs and adjectives). Although, this is a time consuming process, the method discussed above, certainly helps in reducing the numbers of candidates. On the other hand, words ending with suffixes with very high frequency (such as the suffix ना (*naa* for verbs), could be added automatically without any verification.

We identify different suffix-replacement rules for each of

[1]We use a program called GATE Morphological Analyser (http://gate.ac.uk/userguide/sec:misc-creole:morpher) that given a set of suffix-replacement rules and an inflected word, returns its root form.

[2]We obtained a list of unique words from the EMILLE corpus which did not exist in the RFList. Assuming that they are inflected words, we randomly picked 2500 words from the list for our evaluation.

| Suffix | Replacement | Count | Example | Category |
|---|---|---|---|---|
| े (*e*) | ा (*aa*) | 1561 | लडके (*ladake, boys*) - लडका (*ladakaa, a boy*) | noun |
| े (*e*) | ा (*aa*) | 1561 | रोने (*rone, for crying*) - रोना (*ronaa, to cry*) | verb |
| ो (*o*) | ी (*ee*) | 492 | नौकरों (*naukaron, servants*) - नौकरी (*nakaree, a job*) | noun |
| े (*e*) | ी(*ee*) | 483 | कटोरे (*katore, bowls*) - कटोरी (*katoee, a bow*) | noun |
| ों (*on*) | - | 476 | परिवारों (*parivaaron, families*) - परिवार (*parivaar, a family*) | noun |
| - | ना (*naa*) | 474 | फिसल (*phisal, to slip*) - फिसलना (*phisalanaa, to slip*) | verb |
| ों (*on*) | ा | 466 | नमूनों (*namoonon, samples*) - नमूना (*namoonaa, a sample*) | noun |
| ते (*te*) | ना (*naa*) | 461 | जाते (*jaate, while going*) - जाना (*jaanaa, to go*) | verb |
| यिों (*iyon*) | ी(*ee*) | 432 | नदियों (*nadiyon, rivers*) - नदी (*nadi, a river*) | noun |
| ता (*taa*) | ना (*naa*) | 371 | देता (*dete, while giving*) - देना (*denaa, to give*) | verb |

Table 2: High Frequency Suffix-Replacement Rules for Hindi

| Prefix | Count | Example | Decomposition |
|---|---|---|---|
| अ (*a*) | 1616 | अपरिचित (*aparichit, unknown*) | अ (*a*) + परिचित (*parichit, known*) |
| वि (*vi*) | 307 | विदेशी (*videshee, foreign*) | वि (*vi*) + देशी (*deshee, local*) |
| अन् (*an*) | 239 | अनावश्यक (*anaavashyak, unnecessary*) | अन् (*an*) + आवश्यक (*aavashyak, necessary*) |
| सु | 197 | सुपात्र (good character) | सु + पात्र (character) |

Table 3: Most Frequent Prefixes

the four grammatical categories[3] using the following procedure. We obtained all unique words from the EMILLE corpus which did not exist in the RFList. Assuming that these words are inflected, for each word in this list, we remove one character from the end of the string and replace it with the first suffix in the table 1. If the resulting word can not be found in the RFList, we use the next suffix. This procedure is repeated until all the replacements are tried. Even then, if the word can not be located, another character from the end of the string is removed and the entire procedure is repeated. This continues until only one character is left. We replace the removed characters with the base-form suffixes and check if the resulting word is found in the RFList. If the resulting word exists in the RFList, a rule is formed using the deleted characters from the original string as suffix, and the appended characters as replacement. Also, both the original word (inflected word) and the root form as identified by the rule are recorded for rule verification (if needed) at the later stage. Table 2 lists the top ten high frequency rules as obtained from our experiment.

As expected, not all rules produce correct results. For example the third rule converts a masculine plural (नौकरों (*naukaron, servants*)) into a feminine singular (नौकरी (*naukaree, a job*)). On the other hand, according to one of the rules if a word ends with ाकर (*aakar*), the suffix is replaced with ी (*ee*). Given a word कमाकर (*kamaakar, after earning*), it gives a base form कमी (*kamee, lacking*), which, in itself is a valid base form but not a correct base form for the word in the question. The order in which these rules are applied is also very important. For example, the third and the fifth rules. Given a word नौकरों (*naukaron, servants*), if the third rule is applied first, the system returns an incorrect baseform नौकरी (*naukaree, job*). However, if the fifth rule is applied first, the system would return a correct base form नौकर (*naukar, servant*). Therefore, we order the rules in the following order:

1. Given two rules, the rule with the longer suffix is executed first. This is to make sure that more specific rules are given priority over generalized rules.

2. If they have suffixes of the same length the rule with the higher frequency in our list (table 2) is executed first.

3. If they have suffixes of the same length and their frequencies (table 1) are same, the frequencies of the rules are looked up (table 2) and the rule with higher frequency is executed first.

4. Finally, if the frequencies of replacements are the same, the one with the smallest replacement string is executed first. This makes sure that the minimum change is applied to the word. For inflected words whose base-forms do not exist in the RFList, we use the output of the first rule that matches the inflected word.

Since every resulting base form is validated using the RFList, this approach restricts us from processing inflected strings whose base forms do not exist in the RFList. For such words, we use the output of the most likely rule (i.e. the first rule that matched the inflected word).

In order to filter the ruleset, we verified it against a set of 2500 words (DATASET2). These words were randomly obtained from the EMILLE corpus and had no word in common with DATASET1. For each word in this set we obtained its root form[4]. We started with an empty ruleset. One rule at a time from the table 2 (in order of top-to-bottom) was added to the ruleset and its performance on the DATASET2 (F-measure) was recorded. If the addition of a rule resulted in a lower score, the rule was removed from the ruleset. If it did not bring any change[5], the user was prompted for a decision to include or exclude the rule from the ruleset. Along with the rule, a list of word pairs as collected earlier for this rule was shown to the user.

[3]The RFList has 2118 verbs, 48563 nouns, 16173 adjectives and 1151 adverbs.

[4]We used high-frequency rules (see table 2) to obtain root forms which were manually verified.

[5]This happens when there is no word in the dataset to which the rule can be applied.

| RFList | (Shrivastava et al., 2005) | | | | | | Extended Ruleset | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Non-derivational | | | Derivational | | | Non-derivational | | | Derivational | | |
| | P | R | F | P | R | F | P | R | F | P | R | F |
| **Wordnet** | 0.736 | 0.683 | 0.708 | 0.735 | 0.683 | 0.708 | 0.743 | 0.717 | 0.730 | 0.743 | 0.717 | 0.730 |
| **Extended** | 0.817 | 0.768 | 0.792 | 0.817 | 0.768 | 0.792 | **0.821** | **0.803** | **0.812** | 0.820 | 0.803 | 0.812 |

Table 4: Results of Experiments on the Hindi Language

| Verbs | % | Nouns | % | Adjectives | % | Adverbs | % |
|---|---|---|---|---|---|---|---|
| વું (*vun*) | 92.57 | ી(*ee*) | 12.27 | ં (*n*) | 22.36 | ં (*n*) | 22.82 |
| વવું (*vavun*) | 17.91 | ા(*aa*) | 11.27 | ુ ં (*un*) | 22.95 | ુ ં (*un*) | 17.41 |
| ાવવું (*aavavun*) | 12.95 | ં (*n*) | 8.86 | ક (*ka*) | 14.22 | ા(*aa*) | 12.00 |
| ાવું (*aavun*) | 9.90 | ર (*ra*) | 8.62 | ી(*ee*) | 9.50 | ે(*e*) | 11.52 |
| ડવું (*davun*) | 9.68 | ો (*o*) | 8.05 | િત (*ita*) | 7.48 | ર (*ra*) | 7.64 |
| રવું (*ravun*) | 9.23 | ુ ં (*un*) | 7.99 | િક (*ika*) | 7.46 | ક (*ka*) | 7.29 |
| વવું (*lavun*) | 6.41 | તા (*taa*) | 6.04 | ર (*ra*) | 7.20 | ત (*ta*) | 7.29 |
| કવું (*kavun*) | 5.29 | ું (*nun*) | 0.14 | ુ (*lu*) | 6.91 | ી (*thee*) | 7.05 |
| - | - | ાલ (*aal*) | 0.14 | વાળુ (*vaalu*) | 6.09 | ાં (*aan*) | 6.17 |
| - | - | વાદ (*vaad*) | 0.13 | ાળુ (*alu*) | 6.09 | - | - |

Table 5: Most Frequent Suffixes for Gujarati Base Forms

Although, one could use DATASET2, in the first place, to learn such rules, it is important to mention that the size of such a dataset is not very big but to manually prepare larger one could be very expensive. On the other hand, obtaining a true distribution of suffixes just based on a small dataset could lead to incorrect results. However, the rules, as shown here, are based on automatically collected high-frequency suffixes and replacements from the corpus. In order to evaluate the final ruleset, we applied it to DATASET1 and obtained P=0.78, R=0.61, F=0.69. Although the results are lower than that obtained with Shrivastava's ruleset, it is important to mention that their ruleset was derived manually whereas our ruleset consists of rules where majority of them are learnt automatically without prior knowledge of the language. Even when the user is asked to verify a rule, he or she is shown a set of word pairs which are specific to the rule and were collected during the discovery phase of these rules. Such a list of word pairs can help users to foresee the outcome of including this rule into the ruleset.

We merged our ruleset with Shrivastava's ruleset (referred to as the "extended ruleset" below) and applied it to DATASET1. Table 4 shows the results of our experiments.

We also experimented with a derivational process whereby the most common prefixes were obtained from the RFList. Given a word from the RFList, characters from the start of the inflected word were removed one by one until the remaining string was found as another individual word in the RFList (see table 3).

For example, consider the inflected word असुवीधाऐं (*asuveedhaein, lack of facilities*). Based on the rules derived earlier, it is possible to obtain a candidate baseform असुविधा (*asuvidhaa, lack of facility*); however, if this form is not present in the RFList, we cannot verify if it is a correct linguistic root. By looking in the prefix list and removing the prefix अ (*a*), the remaining word सुविधा (*suvidhaa, facility*) is found in the RFList. In such cases, the original resulting base-form (i.e. असुविधा (*asuvidhaa, lack of facility*)) is considered a valid base form.

As can be seen in the table 4, we compare the results of Shrivastava's ruleset with the extended ruleset. It also shows the effect of adding more words to the RFList. The table also shows the results that were obtained after the addition of the derivational process. While the derivational process does not appear to make much difference this can be because very few pairs appear in the dataset to test the process.

## 4. A Ruleset for the Gujarati Language

In order to test if the methodology discussed above can be used for a similar language, an experiment was carried out on the Gujarati language. We used a Gujarati dictionary as a base-form list (GRFList from now) and obtained the most common base-form suffixes (see table 5) using the same method as described earlier for the Hindi language.

Using the list of base-form suffixes we were able to locate 576 words in their base-forms in the corpus which did not exist in the GRFList. We obtained a set of unique and inflected words (84,489) from the corpus that did not exist in the GRFList and obtained suffix-replacement rules (see table 6).

The training data we used for our experiments contains 2000 inflected words (GUJDATASET1). Words in this training data were collected by random selection from the EMILLE corpus. Similar to the experiment with the Hindi language, we used GUJDATASET1 to filter the ruleset to one that yielded the highest f-measure on the dataset. In order to test the performance of the final ruleset we prepared another dataset containing 2000 inflected words (GUJDATASET2). The final ruleset was tested on the GUJDATASET2. We obtained P=0.83, R=0.70, F=0.76. Observing the GRFList revealed that there are many incorrect entries in the GRFList, which, if improved, could improve overall results of the system.

## 5. Conclusion

We have presented a method that given a corpus and a dictionary can be used to obtain a set of suffix-replacement

| Suffix | Replacement | Count | Example | Category |
|---|---|---|---|---|
| ીળ(*ee*) | ું (*vun*) | 392 | પીગળી (*peegalee, melting*) - પીગળવું (*pigalavu, to melt*) | verb |
| ાર (*naa*) | - | 358 | સલાહકારના (*salaahakaaranaa, advisor's*) - સલાહકાર (*salaahakaar, advisor*) | noun |
| ની (*nee*) | - | 347 | સલાહકારની (*salaahakaaranee, advisor's*) - સલાહકાર (*salaahakaar, advisor*) | noun |
| ને (*ne*) | - | 318 | સલાહકારને (*salaahakaarane, to advisor*) - સલાહકાર (*salaahakaar, advisor*) | noun |
| ાવ(*aa*) | ું (*un*) | 278 | બતાવવા (*bataavavaa, for showing*) - બતાવવું (*bataavavun, to show*) | verb |
| ાયે(*aa*) | ું (*un*) | 275 | જોડાયેલા (*jodaayelaa, connected(p)*) - જોડાયેલું (*jodaayelu, connected(s)*) | adj |
| માં (*maan*) | - | 273 | વિચારમાં (*vichaarmaan, in thought*) - વિચાર (*vichaar, thought*) | noun |
| ે(*e*) | ું (*vun*) | 269 | જાળવે (*jaalave, preserve*) - જાળવવું (*jaalavavun, preserve*) | verb |
| તા (*taa*) | ું (*vun*) | 258 | બતાવતા (*bataavataa, showing*) - બતાવવું (*bataavavun, to show*) | verb |
| નો (*no*) | - | 258 | કટોકટીનો (*katokateeno, of urgency*) - કટોકટી (*katokatee, urgent*) | noun |

Table 6: High Frequency Suffix-Replacement Rules for Gujarati

rules for deriving an inflected word's root form. We presented a Hindi morphological analyser and showed that it is possible to adapt the same approach to similar languages by developing a morphological analyser for the Gujarati language as well. Given that the entire process of developing such a ruleset is simple and fast, our approach can be used for rapid development of morphological analysers and yet can obtain competitive results with analysers built relying on human authored rules.

## Acknowledgement

This work is partially supported by the EU-funded MUSING project (IST-2004-027097).

## 6. References

N. Aswani and R. Gaizauskas. 2009. Evolving a general framework for text alignment: Case studies with two south asian languages. In *Proceedings of the International Conference on Machine Translation: Twenty-Five Years On*, Cranfield, Bedfordshire, UK, November.

D.A. Hull. 1996. Stemming algorithms: a case study for detailed evaluation. *J. Am. Soc. Inf. Sci.*, 47(1):70–84.

R. Krovetz. 1993. Viewing morphology as an inference process. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 191–202, New York, NY, USA. ACM.

J. Lovins. 1968. Development of a stemming algorithm. *Mechanical Transactions - Computational Linguistics*, 11:22–31.

P. Majumder, M. Mitra, S.K. Parui, G. Kole, P. Mitra, and K. Datta. 2007. Yass: Yet another suffix stripper. *ACM Transanctions and Information Systems*, 25(4):18.

A. Pandey and T.J. Siddiqui. 2008. An unsupervised hindi stemmer with heuristic improvements. In *Proceedings of the second workshop on Analytics for noisy unstructured text data*, pages 99–105, New York, NY, USA. ACM.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–0137.

A. Ramanathan and D. Rao. 2003. A lightweight stemmer for hindi. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(2):130–142.

M. Shrivastava, N. Agrawal, B. Mohapatra, S. Singh, and P. Bhattacharya. 2005. Morphology based natural language processing tools for indian languages. In *Proceedings of the 4th Annual Inter Research Institute Student Seminar in Computer Science*, IIT, Kanpur, India, April.

J. Xu and W. B. Croft. 1998. Corpus-based stemming using cooccurrence of word variants. *ACM Trans. Inf. Syst.*, 16(1):61–81.