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

Niraj Aswani and Robert Gaizauskas
n.aswani@dcs.shef.ac.uk r.gaizauskas@dcs.shef.ac.uk
University of Sheffield

Abstract

We present a method that given a corpus and a dictionary 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.

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. We propose an approach that both discovers rules from corpora and takes prefixes as well suffixes into account.

In this paper we present a rule-based morphological analyser where the rules are acquired semi-automatically from corpora. 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, to appear). In order to demonstrate our approach's portability to other similar languages, we present our experiments for both the Hindi and the Gujarati language. In future, we plan to carry out similar experiments on other similar languages (e.g. Bengali, Urdu and Punjabi).

2 Our Approach

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. We adapted[1] their ruleset and applied it on our test data[2] (DATASET1). We obtained P=0.74, R=0.68, F=0.71. We found that not all words' root forms were present in the RFList. Also, there were some inflected words which could not be matched with any of the rules in their ruleset. In Hindi, most base-form verbs end with suffix ना (*naa*) and base-form nouns with the vowel ा (*aa*) and ी (*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).

In Hindi, not only are nouns and verbs inflected but adjectives and adverbs are too (see table 1). We identify different suffix-replacement rules for each of 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

[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.

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

Table 1: Most Frequent Suffixes for Hindi Base Forms

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

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

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 five high frequency rules as obtained from our experiment.

Table 2: High Frequency Suffix-Replacement Rules for Hindi

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

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.

In our approach, rules are executed based on the following priorities: 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.

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

[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.

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).

Table 3: Most Frequent Prefixes

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)

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.

Table 4: Results of Experiments on the Hindi Language

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

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.

3 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[6]. 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. 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.

4 Conclusion

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

[6]Due to space contraints, the tables showing various statistics such as common base-form suffixes and suffix-replacement rules have been omitted from this abstract.

References

N. Aswani and R. Gaizauskas. 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 to appear.

D.A. Hull. Stemming algorithms: a case study for detailed evaluation. *J. Am. Soc. Inf. Sci.*, 47(1):70–84, 1996. ISSN 0002-8231.

R. Krovetz. 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, 1993. ACM. ISBN 0-89791-605-0.

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

P. Majumder, M. Mitra, S.K. Parui, G. Kole, P. Mitra, and K. Datta. Yass: Yet another suffix stripper. *ACM Transanctions and Information Systems*, 25(4):18, 2007. ISSN 1046-8188.

A. Pandey and T.J. Siddiqui. 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, 2008. ACM. ISBN 978-1-60558-196-5.

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

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

M. Shrivastava, N. Agrawal, B. Mohapatra, S. Singh, and P. Bhattacharya. 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 2005.