A Lightweight Stemmer for Hindi

Ananthakrishnan Ramanathan
National Centre for Software Technology
Rain Tree Marg, Sector 7, CBD Belapur
Navi Mumbai 400614, India
anand@ncst.ernet.in

Durgesh D Rao
DR Systems
S-27, Lane 1, Sector 9, CBD Belapur
Navi Mumbai 400614, India
drsystems@vsnl.net

Abstract

Stemming is an operation that conflates morphologically similar terms into a single term without doing complete morphological analysis. Stemming is used in information retrieval systems to improve performance. Additionally, this operation reduces the number of terms in the information retrieval system, thus decreasing the size of the index files. This paper presents a lightweight stemmer for Hindi, which conflates terms by suffix removal. The proposed stemmer is both computationally inexpensive and domain independent. The paper discusses the systematic manner in which the suffix list was developed, and provides the linguistic rationale behind including various suffixes in the list. Similar techniques can be used to build stemmers for other Indian languages such as Marathi, Gujarati, and Punjabi. The stemmer has been evaluated by computing understemming and overstemming figures for a corpus of documents. The results are favourable and indicate that the proposed stemmer can be used effectively in IR systems.

1 Introduction

Stemming is an operation that relates morphological variants of a word. The term ‘conflation’ is used to denote the act of mapping variants of a word to a single term or ‘stem’. Stemming is used in Information Retrieval systems (Frakes and Baeza-Yates, 1992; Korphage, 1997) to improve performance. For example, when a user enters the query word stemming, he most likely wants to retrieve documents containing the terms stemmer and stemmed as well. Thus, using a stemmer improves recall, i.e., the number of documents retrieved in response to a query. Also, since many terms are mapped to one, stemming serves to decrease the size of the index files in the IR system.

Many stemming algorithms have been proposed, and there have been many experimental evaluations of these (Frakes and Baeza-Yates, 1992; Hull and Grefenstette, 1996). But, no work on stemming has been reported for Indian languages. In this paper, we present a lightweight stemmer for Hindi, which conflates terms by stripping off word endings from a suffix list on a ‘longest match’ basis. The key advantages of this stemmer are: it is computationally inexpensive, and it is domain independent. We have evaluated the stemmer by computing understemming and overstemming statistics for a corpus of documents.

The paper is organised as follows: The next section looks at the different approaches to stemming possible, and related work. Section 3 discusses the stemmer that is proposed by this paper. Section 4 presents the results of evaluation. The last two sections contain some discussion and directions for future work.
2 Approaches to Stemming

One approach to stemming (Frakes and Baeza-Yates, 1992) is to store all possible index terms and their stems in a table, and stem terms via table lookup. Though this is accurate, and efficient in terms of speed, such data is usually not available. Even if they were, such an approach would restrict the stemmer to the words in the table, and render it domain dependent.

Other dynamic approaches that use statistical measures to conflate terms have been proposed. Two such approaches are successor variety stemmers and ngram stemmers. Successor variety stemmers (Hafer and Weiss, 1974) identify morpheme boundaries based on the distribution of morphemes in a large body of text. N-gram stemmers (Adamson and Boreham, 1974) conflate terms based on the number of n-grams that are shared by the terms.

Affix-removal stemmers perform stemming by removing word prefixes and suffixes. These stemmers iteratively remove the longest possible string of characters from a word according to a set of rules. Some affix-removal stemmers also transform the resultant stem in some cases. (Porter, 1980) and (Paice, 1974) are popular iterative longest match stemmers.

The stemmer proposed in this paper strips off word suffixes from a suffix list on a longest match basis. Our stemmer, though, is not iterative, which makes it a lightweight program. In the next section, we look at how the morphological features of Hindi allow such a simple suffix removal program to be used as a stemmer.

3 A Lightweight Stemmer for Hindi

Hindi is a (relatively) free word-order and highly inflectional language. In English, which is more fixed in its word order, the relations between the various components of the sentence are shown largely by their relative positions. In Hindi, these relations are shown by using postpositions, and accordingly inflecting nouns to express case information, and inflecting verbs to reflect gender, number, and person information. This is illustrated in Figure 1.

Hindi has been represented using an ASCII transliteration scheme to facilitate use of commonly available text processing tools. The appendix shows the transliteration scheme that was used.

Though Hindi is inflectionally a rich language, the rules governing inflections are fairly simple and few in number. Most inflected forms of a word can be reduced to a common stem by one suffix removal operation. The noun, adjective, and verb inflections are discussed below with a few examples.

3.1 Noun Inflections (McGregor, 1977)

Hindi nouns have two cases: the direct case and the oblique case. The direct case denotes sentence subjects or direct objects; the oblique occurs when the noun is followed by a postposition. All nouns are either masculine or feminine; Hindi does not possess a neuter gender. Nouns are inflected based on the case, the number, and the gender. Below, we explore the inflections possible for the two genders, for singular and plural nouns, for the direct and oblique cases.

a) For masculine nouns, the following rules govern most inflections.

(i) Ending in A
Singular Direct  
ladakA (boy)
Singular Oblique  
(boy)
Plural Direct  
ladake
Plural Oblique  
(ladakoM)

In some exceptional cases, the plural ending does not change, and the plural oblique ends in AoM.

E.g. For the singular direct rAjA (king), the singular oblique and the plural are the same, and the plural oblique is rAjAoM

(ii) Other endings
Singular Direct  
xina (day)
Singular Oblique  
xina
Plural Direct  
xina
Plural Oblique  
(xinoM)

(iii) Ending in AMh
These are inflected as in rule (i), except that the endings are nasalised.

E.g. The singular direct kuAMh (well) has oblique kueMh, and the plural direct kueMh has oblique kuoMh.

(iv) Masculines ending in I and U shorten these vowels before the oblique plural ending, and masculines in final I also use a y before the ending. E.g. AxamI (man) has plural oblique AxamiyoM, and hindU (Hindu) has plural oblique hinduoM.

(v) Vocatives
In the singular, these are expressed by using the oblique, and in the plural, a final o is used instead of the oM that is used for the oblique. E.g. ladakA becomes ladake, and in the plural, ladake becomes ladako.

Thus, the following suffix deletions (longest possible match) are required to reduce inflected forms of masculine nouns to a common stem:

a A i l u U e o AM eM oM AMh iyoM uAM uoM ueM AeM AoM

b) The rules for inflecting feminine nouns are:

(i) ending in I
Singular Direct  
ladakiI (girl)
Singular Oblique  
ladakiI
Plural Direct  
(ladaktiyAMh)
Plural Oblique  
(ladakiyoM)

(ii) Feminines ending in i, are inflected as in (i).
E.g. swiWi (position).

(iii) Ending in iyA
Singular Direct  
cidiyA (bird)
Singular Oblique  
cidiyA
Plural Direct  
cidiyAMh
Plural Oblique  
cidiyoM

(iv) Other endings
Singular Direct  
havA (air)
Singular Oblique  
havA
Plural Direct  
h ava eM
Plural Oblique  
havoM

(v) Feminine vocatives are formed in the same way as masculines.

Thus, the only additional suffix required to account for feminine nouns is iyAMh.

3.2 Adjective Inflections (McGregor, 1977)

Adjectives whose direct singular masculine form ends in A or AM agree with the noun in gender, number, and case. Other adjectives do not vary. E.g. The singular direct badA becomes bade in all other masculine cases, and badI in all feminine cases.

Thus, no new suffixes are added for adjectives.

3.3 Verb Inflections

Hindi verbs are inflected depending on the gender, number, person, tense, aspect, negation, and voice. A complete list of verb inflection rules can be found in (Rao, 1996). A couple of entries from this list are shown in Figure 2.

The first entry says that if the tense-aspect agreement is present simple, and the gender-number-person agreement is masculine 1st person, the root is inflected as root+wA hUM.
Note that the vowel a at the end of roots is removed, and hence awA is added to the suffix list instead of wA.

The full list of suffixes for verbs was generated by working through the entire list in a similar way.

Since the suffix list for verbs includes AI, awA and anI, the following suffixes had to be added to the list to handle nouns with these endings:

AiyAM, AiyOM, AiyAMh, awAeM, awAoM, anAeM, anAoM

<table>
<thead>
<tr>
<th>T</th>
<th>A</th>
<th>G</th>
<th>N</th>
<th>P</th>
<th>Inflection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pres</td>
<td>Simp</td>
<td>Mas</td>
<td>Sing</td>
<td>1</td>
<td>+wA hUM</td>
</tr>
<tr>
<td>Pres</td>
<td>Simp</td>
<td>Mas</td>
<td>Sing</td>
<td>2</td>
<td>+we hO</td>
</tr>
</tbody>
</table>

Figure 2: Verb Inflection Rules – Sample Entries

3.4 The Stemmer

The complete suffix list is shown in Figure 3. The stemmer is implemented by simply removing from each word the longest possible suffix from this list.

```
A AeM awA Ane egA
i AoM awI UmgA egI
l iyAM IM UmgI AegA
u iyoM awIM AUMgA AegI
U AiyAM awe AUMgI AyA
e AiyOM AwA eMge Ae
o AMh Awl eMgI Al
eM iyoAMh AwIM Aemge AIM
oM AiyAMh Awe AemgI ie
AM awAeM anA oge Ao
uAM awAoM anI ogI Aie
ueM anAeM ane Aoge akara
uoM anAoM AnA AogI Akara
```

Figure 3: Suffix List

4 Evaluation

Parameters that can be used for evaluating stemmers are: retrieval effectiveness achieved using the stemmer, the stemmer’s compression performance, and the correctness of the stems produced by it.

Correctness of a stem does not imply linguistic correctness, in the sense that the stem need not be the morphological root. For example, the morphological root of the word computing is compute, whereas a stemmer could remove the suffix ing and leave the stem comput. This would not be considered incorrect if the morphological variants of compute, such as computing, computed, computes, are all mapped to the same stem - comput. Thus, a stemmer can be viewed as an efficient approximation of a morphological analyser (Bharati et al., 1995). A stemmer is said to be correct if - a) words that are morphological variants are actually conflated to a single stem, and b) the words conflated to a single stem are indeed morphological variants.

“Overstemming” occurs when words that are not morphological variants are conflated. For example, in English, if the words compile and compute are both stemmed to comp, it is a case of overstemming. Another example of overstemming would be wander and wand being conflated to wand. The error here is that the ending er of wander is considered a suffix, whereas it is actually part of the stem.

“Understemming” occurs when words that are indeed morphological variants are not conflated. An example of understemming, in English, would be: compile being stemmed to comp, and compiling, to compil.

We have evaluated our stemmer by computing the number of understemming and overstemming errors for a corpus of documents. The corpus used for evaluation was a collection of documents from different sections of an online Hindi news magazine. Documents were chosen from varied domains such as Films, Health, Business, Sports, and Politics. The collection contained 35977 unique words.

For each unique word in the corpus, we obtained the root using a freely available morphological analyser (Morph, 2001). This program has a reported coverage of 88%. The words conflated by the morphological analyser were considered as variants.

The understemming and overstemming percentages were calculated using the following formulae:

\[
\text{Understemming} = \frac{\text{Understemming Errors}}{\text{Total Unique Words}}
\]

\[
\text{Overstemming} = \frac{\text{Overstemming Errors}}{\text{Total Unique Words}}
\]
<table>
<thead>
<tr>
<th>Variants</th>
<th>Case</th>
<th>Stem</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAIbahana</td>
<td>Direct</td>
<td>BAIbahan</td>
</tr>
<tr>
<td>(Brothers and Sisters)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAIbahanOM</td>
<td>Oblique</td>
<td></td>
</tr>
<tr>
<td>PLEtia</td>
<td>Direct</td>
<td>PLEt</td>
</tr>
<tr>
<td>(Transliteration of “Flat”)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLEtoM</td>
<td>Oblique</td>
<td></td>
</tr>
<tr>
<td>GusapETie</td>
<td>Direct</td>
<td>GusapET</td>
</tr>
<tr>
<td>(Infiltrators)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GusapETiyoM</td>
<td>Oblique</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4: A sample of variants that were not conflated by the morphological analyser

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of unique words</td>
<td>35977</td>
</tr>
<tr>
<td>Morphological variants</td>
<td>7750</td>
</tr>
<tr>
<td>Words understemmed</td>
<td>363</td>
</tr>
<tr>
<td></td>
<td>(error: 4.68%)</td>
</tr>
<tr>
<td>Words conflated by the stemmer</td>
<td>13710</td>
</tr>
<tr>
<td>Words overstemmed</td>
<td>1898</td>
</tr>
<tr>
<td></td>
<td>(error: 13.84%)</td>
</tr>
</tbody>
</table>

Figure 5: Evaluation Results

\[
\text{% understemming error} = \frac{\text{(Number of variants not conflated by the stemmer \times 100)}}{\text{(Total number of morphological variants)}}
\]

\[
\text{% overstemming error} = \frac{\text{(Number of nonvariants conflated by the stemmer \times 100)}}{\text{(Total number of words conflated by the stemmer)}}
\]

5 Discussion

The stemmer proposed in this paper largely handles inflectional morphology. It does not account for most of the derivational morphology of Hindi; that is, it does not conflate terms that belong to different word categories. A morphological analyser, on the other hand, would conflate such terms also. It is our contention that for categorization and information retrieval tasks reducing derivationally related terms to the same stem would lead to overconflation in some cases, thus balancing out the performance. For example, it is not entirely clear whether a query for baccA (child) should retrieve documents containing the word bacapanA (childhood).

Though the stemmer was developed with the intent of dealing with inflectional morphology alone, there are a few suffixes such as awA and AI whose removal causes some derivationally related words to be conflated as well. Examples of such conflations are: acCA (good) and acCAI (goodness), and BarawIya (Indian) and BarawIyawA (Indianness).

6 Directions for Future Work

The proposed stemmer needs to be further evaluated with Hindi information retrieval systems. Such statistical evaluation will suggest the best tradeoff between understemming and overstemming that can be achieved by dropping or adding a few suffixes in the list.

A more thorough error analysis is required to ascertain what improvement is possible by including iterative rules, and whether such rules will substantially increase the computational cost.

Most West and North Indian languages (Marathi, Gujarathi, Punjabi etc.) are similar to Hindi in morphology. It would be interesting to see whether similar techniques can be used to develop stemmers for these languages.

Acknowledgements

The initial work for this project was done at a workshop organised at the AU-KBC Research Centre, Chennai. We would like to thank the lead faculty, Dr. Srinivas Bangalore of AT&T Labs-Research, who motivated this study and provided
valuable insights and guidance during the workshop. We also thank Mr. Sasikumar, Mr. Vivek Mehta, Mr. Jayprasad Hegde, and other staff at the KBCS division of the National Centre for Software Technology for useful feedback and encouragement.

References


Appendix

Hindi Transliteration Scheme

<table>
<thead>
<tr>
<th>अ</th>
<th>आ</th>
<th>इ</th>
<th>ई</th>
<th>उ</th>
<th>ऊ</th>
<th>ऋ</th>
<th>ऌ</th>
<th>ऎ</th>
<th>ए</th>
<th>ओ</th>
<th>औ</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>A</td>
<td>i</td>
<td>I</td>
<td>u</td>
<td>U</td>
<td>o</td>
<td>O</td>
<td>e</td>
<td>E</td>
<td>ə</td>
<td>ɛ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ख</th>
<th>ग</th>
<th>घ</th>
<th>ङ</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>G</td>
<td>ɣ</td>
<td>ə</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>च</th>
<th>छ</th>
<th>ज</th>
<th>झ</th>
</tr>
</thead>
<tbody>
<tr>
<td>ȷ</td>
<td>g</td>
<td>j</td>
<td>ʃ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ट</th>
<th>ठ</th>
<th>ड</th>
<th>ढ</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>T</td>
<td>d</td>
<td>ʒ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ण</th>
<th>त</th>
<th>थ</th>
<th>द</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>t</td>
<td>θ</td>
<td>d</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>न</th>
<th>ब</th>
<th>भ</th>
<th>ध</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>b</td>
<td>ɬ</td>
<td>ɭ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>प</th>
<th>फ</th>
<th>ब</th>
<th>भ</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>ŋ</td>
<td>b</td>
<td>ɬ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>य</th>
<th>र</th>
<th>ल</th>
<th>व</th>
<th>स</th>
<th>ह</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>r</td>
<td>l</td>
<td>v</td>
<td>s</td>
<td>h</td>
</tr>
</tbody>
</table>