Hindi Derivational Morphological Analyzer

Nikhil Kanuparthi
LTRC
IIIT-Hyderabad
India
{nikhil.kvs,abhilashi}@research.iiit.ac.in

Abhilash Inumella
LTRC
IIIT-Hyderabad
India

Dipti Misra Sharma
LTRC
IIIT-Hyderabad
India
dipti@iiit.ac.in

Abstract

Hindi is an Indian language which is relatively rich in morphology. A few morphological analyzers of this language have been developed. However, they give only inflectional analysis of the language. In this paper, we present our Hindi derivational morphological analyzer. Our algorithm upgrades an existing inflectional analyzer to a derivational analyzer and primarily achieves two goals. First, it successfully incorporates derivational analysis in the inflectional analyzer. Second, it also increases the coverage of the inflectional analysis of the existing inflectional analyzer.

1 Introduction

Morphology is the study of processes of word formation and also the linguistic units such as morphemes, affixes in a given language. It consists of two branches: derivational morphology and inflectional morphology. Derivational morphology is the study of those processes of word formation where new words are formed from the existing stems through the addition of morphemes. The meaning of the resultant new word is different from the original word and it often belongs to a different syntactic category. Example: happiness (noun) = happy (adjective) + ness. Inflectional morphology is the study of those processes of word formation where various inflectional forms are formed from the existing stems. Number is an example of inflectional morphology. Example: cars = car + plural affix 's'.

The main objective of our work is to develop a tool which executes the derivational morphological analysis of Hindi. Morphological analysis is an important step for any linguistically informed natural language processing task. Most morphological analyzers perform only inflectional analysis. However, derivational analysis is also crucial for better performance of several systems. They are used to improve the efficiency of machine translators (C Gdaniec et al., 2001). They are also used in search engines to improve the information extraction (J Vilares et al., 2001). Since derivational processes can often be productive in a language, the development of an effective derivational analyzer will prove beneficial in several aspects.

We developed a derivational analyzer for Hindi over an already existing inflectional analyzer developed at IIIT Hyderabad. In this approach, first, the derived words in Hindi were studied to obtain the derivational suffixes of the language. Then the rules were designed by understanding the properties of the suffixes. The Hindi Wikipedia was also utilized to collect the required background data. Finally, an algorithm was developed based on the above findings. This algorithm has been used to upgrade the inflectional analyzer to a derivational analyzer.

In the sections that follow, we describe the approach we followed to develop our derivational analyzer and the experiments that we conducted using our system.

2 Related Work

There is no derivational morphological analyzer for Hindi to the best of our knowledge. However, a few inflectional morphological analyzers (IIIT; Vishal and G. Singh, 2008; Niraj and Robert, 2010)

Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology (SIGMORPHON2012), pages 10–16,
Montréal, Canada, June 7, 2012.

of this language have been developed. There are derivational analyzers for other Indian languages like Marathi (Ashwini Vaidya, 2009) and Kannada (Bhuvaneshwari C Melinamath et al., 2011). The Marathi morphological analyzer was built using a Paradigm based approach whereas the Kannada analyzer was built using an FST based approach. As far as English is concerned, there are some important works (Woods, 2000; Hoeppner, 1982) pertaining to the area of derivational morphological analysis. However, both of these are lexicon based works.

For our work, we employed a set of suffix replacement rules and a dictionary in our derivational analyzer, having taken insights from the Porter's stemmer (Porter, 1980) and the K-stemmer (R. Krovetz. 1993). They are amongst the most cited stemmers in the literature. The primary goal of Porter's stemmer is suffix stripping. So when a word is given as input, the stemmer strips all the suffixes in the word to produce a stem. It achieves the task in five steps applying rules at each step. Given a word as input, the Krovetz stemmer removes inflectional suffixes present in the word in three steps. First it converts the plural form of the word into a singular form, then it converts past tense to present tense, and finally removes -ing. As the last step, the stemmer checks the dictionary for any recoding and returns the stem. Our algorithm uses the main principles of both the Porters stemmer and Krovetz stemmer. The suffix replacement rules of our algorithm resemble that of the Porters and a segment of the algorithm is analogous to the dictionary based approach of the Krovetzs stemmer.

3 Existing Inflectional Hindi Morphological Analyzers

A derivational morph analyzer can be developed from an existing morph analyzer instead of building one from scratch. So three inflectional analyzers were considered for the purpose. The morphological analyzer developed by Vishal and Gurpreet stores all the commonly used word forms for all Hindi root words in its database. Thus, space is a constraint for this analyzer but the search time is quite low. The morph analyzer developed by Niraj and Robert extracts a set of suffix replacement rules from a corpus and a dictionary. The rules are applied to an inflected word to obtain the root word. They show that the process of developing such rulessets is simple and it can be applied to develop morphological analyzers of other Indian languages.

However, our derivational analyzer is an extension of an existing inflectional morphological analyzer developed at IIIT Hyderabad (Bharati Akshar et al, 1995). The inflectional analyzer is based on the paradigm model. It uses the combination of paradigms and a root word dictionary to provide inflectional analysis. Given an inflected Hindi word, this inflectional analyzer returns its root form and other grammatical features such as gender, number, person, etc. For example: if the input word to the morphological analyzer is *bAgabAnoM*[1] (gardeners), the output will be *bAgabAna* (gardener), noun, m, pl, etc. Here the word *bAgabAna* is the root word of the input word. 'Noun' is the category of the input word, 'm' means masculine and 'pl' means that the input word is plural in number.

The analyzer uses a root word dictionary for the purpose. If a word is present in the root word dictionary, the analyzer handles all the inflections pertaining to that word. For example: *xe* (give) is a root word present in the dictionary of the analyzer. *xewA* (gives), *xenA* (to give), *xiyA* (gave) and other inflectional forms of the root word xe are handled by the analyzer. There are 34407 words in the root word dictionary.

The analyzer handles inflected words using the paradigm tables. Every entry (word) in the dictionary has values like lexical category, paradigm class, etc. For example: there is a word *pulisavAlA* (policeman) in the dictionary. Its paradigm class is *ladakA*. Table 1 shows the paradigm forms of *ladakA*. Since the paradigm value of *pulisavAlA* is *ladakA*, its four inflections will be similar to the four paradigms of *ladakA* (root paradigm). The four inflections of *pulisavAlA* are *pulisavAlA, pulisavAle, pulisavAle, pulisavAloM*. Only the root form (word) *pulisavAlA* is present in the dictionary. In this way every root word present in the dictionary belongs to a paradigm class and this paradigm class has a structured paradigm table containing all the inflections of the main paradigm. This paradigm table is used by

[1]The Hindi words are in wx-format (sanskrit.inria.fr/DATA/wx.html) followed by IIIT-Hyderabad.

Table 1: Paradigm table of *ladakA*

Case	Singular form	Plural form
Direct	*ladakA* (boy)	*ladake* (boys)
Oblique	*ladake* (boy)	*ladakoM* (boys)

the analyzer to reconstruct all the inflections of the root words belonging to this paradigm class. Therefore the analyzer can analyze a word only if its root word is present in the dictionary.

This inflectional morphological analyzer works as a platform for our derivational morphological analyzer. So our tool gives derivational analysis of all the words whose root forms are present in the root word dictionary. Our tool also tackles certain words whose root forms are not present in the root word dictionary of the IIIT morphological analyzer.

4 Approach

We pursued the following five step approach for building our derivational analyzer.

4.1 Studying Hindi Derivations

To build the derivational morphological analyzer, we first conducted a study to identify the derivational suffixes and the related morphological changes. After identifying the suffixes, the rules pertaining to these suffixes were obtained.

First, the nouns present in the Hindi vocabulary were studied. The study of nouns helped us in identifying some of the most productive derivational suffixes present in the language. For example, let us consider the word *maxaxagAra (helper)*. This word is derived from the word *maxaxa* (*maxaxagAra* = *maxaxa (help)* + *gAra*). But gAra cannot be confirmed as a suffix because of just one instance. In order to confirm *gAra* as a suffix, even other words ending with *gAra* must be examined. The more the number of words we find, the greater is the productivity of the suffix. Words like *yAxagAra* (derived from *yAxa*) and *gunAhagAra (criminal)* (derived from *gunAha (crime)*) prove that *gAra* is a derivational suffix. However, every word ending with *gAra* need not be a derived word. For example: the word *aMgAra* is not a derived word. Therefore only relevant words were studied and the suffixes were obtained only from them.

Table 2: Example derivations of some suffixes

Suffix	Root	Derivation
Ana	*laganA*	*lagAna*
bAna	*bAga*	*bAgabAna*
gAra	*yAxa*	*yAxagAra*
xAra	*xukAna*	*xukAnaxAra*
ika	*aXikAra*	*aXikArika*
I	*KuSa*	*KuSI*
AI	*acCA*	*acCAI*

Table 3: Rules of few suffixes

Suffix	First set rules
bAna	noun = noun/adj + *bAna*
gAra	noun = noun/adj + *gAra*
xAra	noun = noun/adj + *xAra*
ika	adj = noun - *a* + *ika*

The entire process of obtaining the derivational suffixes was done manually and was a time consuming process. This process was repeated for adjectives as well. Only those suffixes that participate in the formation of nouns and adjectives were found. A total of 22 productive derivational suffixes were procured. Table 2 shows a few suffixes and their derivations.

4.2 Derivational Rules

After finding the derivational suffixes, two sets of derivational rules were developed for each suffix. The first set explains the formation of the derived words from their root words. Let us consider the suffix *gAra*. This suffix generates nouns from nouns and adjectives. The rule of this suffix explains the formation of derivations like *yAxagAra* (*yAxagAra* = *yAxa* (noun) + *gAra*) and *maxaxagAra* (*maxaxagAra* = *maxaxa* + *gAra*). The second set consists of reverse rules of the first set. The reverse rule for the previous example is noun/adj = noun - suffix. In this way, rules were developed for all the 22 derivational suffixes. These rules form a vital component of our algorithm. Table 3 contains the derivational rules of a few suffixes.

4.3 Finding Majority Properties

The majority properties (of derived words of a suffix) are the properties which most of the words ex-

hibit. Example: let us consider the derived words of the suffix *vAlA*. There are 36 derived words of the *vAlA* suffix in the root word dictionary. Some of these words are adjectives but the majority are nouns. Hence noun is fixed as the category (majority category) for derived words of this class. Similarly the majority paradigm class of these words is *ladakA*. The majority properties of derived words pertaining to all the 22 suffixes were acquired.

The majority properties of a suffix help us in the derivational analysis of the unknown derived words of that suffix. For example: consider the word *GaravAlA (housekeeper)*. Let us assume that it is not present in the root word dictionary. Therefore the lexical category, paradigm value and other important features of this word are not known. But let us assume that this word is a genuine derived word of the suffix *vAlA*. So the tool must handle this case. The majority properties of the *vAlA* suffix are assigned to this word. So noun and *ladakA* are fixed as the category and paradigm of this word. Thus the genuine derived words which are unknown to the analyzer will be analyzed using the majority properties.

The majority properties of derived words were obtained in two main steps. First, a suffix was considered. Then all the derived words pertaining to that suffix were acquired. Only genuine derived words were taken into consideration. Genuine derivations were found out using the suffix derivational rules. Example: let us take the word *maxaxagAra* (ending with *gAra*). First, the root word of this word is retrieved using the *gAra* derivational rule. The root word according to the rule is *maxaxa*. This word is present in the dictionary and it also satisfies the category condition of the rule. The word *maxaxa* is a noun. Hence the word *maxaxagAra* is accepted as a derived word. If the word *maxaxa* is not found in the dictionary or if its category is not a noun/adjective, the word *maxaxagAra* will be rejected. In this way all the valid derivations of the suffix were acquired. This process was repeated for other suffixes as well. In the second step, the majority properties of the derived words were directly retrieved.

Finally, a suffix table was built using the majority properties of the derived words. The suffix table contains all the suffixes and their inflectional forms. Table 4 contains few suffixes and their inflectional forms. For example: the majority paradigm of derived words of *vAlA* suffix is *ladakA*. This implies that the derived words of this suffix end with *vAlA*, *vAle* and *vAloM*. Thus the possible inflections of a suffix can be derived from its majority properties. This information was stored in a table. The majority properties and the suffix table play an important role in the analysis of the unknown words. Their usage in our algorithm will be described in the later sections.

Table 4: Few suffixes and their forms

Suffix	Suffix-forms
Ana	*Ana*
bAna	*bAna*, *bAnoM*
gAra	*gAra*, *gAroM*
xAra	*xAra*, *xAroM*
ika	*ika*
I	*I*
AI	*AI*
anI	*anI*, *aniyAz*, *aniyoM*

4.4 Using Wikipedia Data for Confirming Genuineness

If an invalid word is not analyzed by the inflectional analyzer, there is no need for proceeding to the derivational analysis of that word. Therefore the genuineness of a word must be tested before going for the derivational analysis. The Hindi Wikipedia was chosen as a resource that enables us to test the genuineness of a word.

A total of 400k words were extracted from the Hindi Wikipedia. This data contains many words which do not exist in Hindi vocabulary. So 220k proper Hindi words were selected (on the basis of frequency) from the data and a list containing those 220k words was created. A word will be treated as a genuine word only when it is present in that list. This assumption is used by our algorithm. The Wiki data is used as a standard corpus.

4.5 Algorithm for Derivational Analysis

An algorithm was developed to make use of the existing inflectional morphological analyzer for derivational analysis. This algorithm enabled us to bypass the construction of a derivational analyzer from the scratch. The majority properties of the derivations, the Wikipedia data and the suffix-table are also employed by the algorithm for analyzing un-

known derivations.

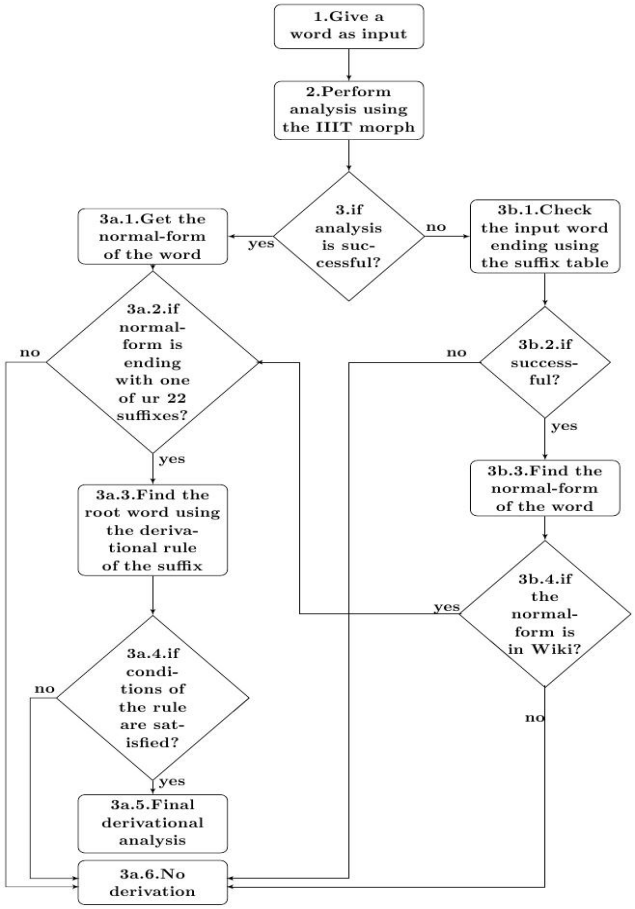

Figure 1: Algorithm

The input to the algorithm is a word. The output is a combination of the inflectional analysis and the derivational analysis of the input word. For example: if the input word is *bAgabAnoM* (gardeners). First, the algorithm gives the inflectional analysis of the input word. In this case the word *bAgabAnoM* is a noun, plural in number, etc. Then it gives the information (category, gender) of the root word (*bAga* (garden)) from which the input word is derived (derivational analysis). So a dual analysis of the input word is provided.

4.6 Examples

The following 4 examples explain the working of the algorithm in 4 different cases. These examples are provided to give a clear picture of the complete algorithm.

a) Example 1

Input word: ***pulisavAle* (Policemen)**

In the step-2, the word is analyzed by the IIIT inflectional analyzer. In the step 3a.1, the word *pulisavAlA* (Policeman) is the normal-form of the input word. The normal-form is ending (*vAlA*) with one of our 22 suffixes. The rule of the suffix is noun = noun/verb + *vAlA*. So the root word is *pulisa* because *pulisavAlA* = *pulisa* + *vAlA*. The word *pulisa* should be a noun or a verb in order to satisfy the rule. All the conditions are met and the step 3a.5 becomes the vital final step. This step gives the information that the final root word *pulisa* is a masculine noun and the input word is also a masculine noun and it is plural in number. Here the information about the final root word and the input word is again given using the inflectional morphological analyzer.

b) Example 2

Input word: ***kirAexAroM* (Tenants)**

The IIIT inflectional analyzer cannot analyze this word. The word *kirAexAroM* is ending with one of the forms (*xAroM*) present in the suffix table. The normal-form of the input word is obtained by replacing the suffix form in the input word with the suffix. Hence the normal-form of the input word *kirAexAroM* is *kirAexAra*. In this way, the normal-form of the input word is acquired without the inflectional analyzer. The word *kirAexAra* is present in Wiki data and it is ending with one of the 22 suffixes. The rule of the suffix is noun = noun/adj + *xAra*. So the root word is *kirAe* because *kirAexAra* = *kirAe* + *xAra*.

c) Example 3

Input word: ***ladake* (Boys)**

In the step-2, the word is analyzed by the IIIT inflectional analyzer. The normal form of the word is *ladakA* (boy). The normal-form of the word is not ending with any of our 22 suffixes. So there is no derivational analysis of this particular case.

d) Example 4

Input word: ***ppppwA* (invalid word)**

The IIIT inflectional analyzer cannot analyze this word. The word *ppppwA* is ending with one of the forms (*wA*) present in the suffix table. But the

normal-form (*ppppwA*) is not present in Wikipedia. So there is no derivational analysis for this particular case.

4.7 Expanding Inflectional Analysis

The algorithm for derivational analysis was also used for expanding the inflectional analysis of the analyzer. Consider the second example in the previous section. The word *kirAexAroM* is analyzed by the derivational analyzer even though its root form (*kirAexAra*) is not present in the root word dictionary. Words like *kirAexAra* are genuine derivations and can be added to the root word dictionary. The addition of such kind of words will extend the inflectional analysis of the analyzer. For example. if the word *kirAexAra* is added, its forms *kirAexAroM* and *kirAexAra* will be automatically analyzed. This is because the word *kirAexAra* would be added along with its features/values like category, paradigm class, etc.

Therefore all the words which fall into the example-2 category of the previous section can be added to the dictionary. All such words must be obtained in order to expand our dictionary. For this purpose, a Wiki data consisting of 220k Wiki words was extracted from Wikipedia. Out of these 220k words, 40k words are ending with our 22 suffixes and their forms. So the derived words which can be analyzed by our system are part of this sub-dataset. Out of 40k words, the derivational analyzer analyzed 5579 words. The inflectional analyzer analyzed only 2362 words out of 40000. So the derivational analyzer analyzed 3217 derived words more than the inflectional analyzer. So these words were added to the root word dictionary for expanding the inflectional analysis of the analyzer. The algorithm which was designed to perform derivational analysis also inflated the inflectional analysis of the analyzer.

5 Experiments and Results

The performance of our derivational analyzer must be compared with an existing derivational analyzer. Since there is no such derivational analyzer, we compared the performance of our tool with the existing IIIT inflectional analyzer (or the old morphological analyzer). The two tools must be tested on a gold-data (data that does not contain any errors). For example: let us assume that we have a data of 100 words and their morphological analysis. The analysis of these 100 words does not contain any errors and it is a gold-data. Now we must get the analysis of these 100 words from both the derivational analyzer and the old morphological analyzer. Then their analyses must be compared against the gold-data. This is nothing but directly comparing the outputs of the derivational analyzer and the old morphological analyzer. This will help in evaluating the derivational analyzer. This method of evaluation will also tell the improvement the derivational analyzer achieved.

Type	*Output / Gold*	*Description*
Type 1	ABCD / ABCD	All the analyses present in the reference present in the output + No wrong analyses present in the output
Type 2	ABCDE / ABCD	All the analyses present in the reference present in the output + Some wrong analyses present in the output
Type 3	ABC / ABCD	Some analyses present in the reference present in the output + No wrong analyses present in the output
Type 4	ABCE / ABCD	Some analyses present in the reference present in the output + Some wrong analyses present in the output
Type 5	EFG / ABCD	No analyses present in the reference present in the output + Some wrong analyses present in the output
Type 6	No Output / ABCD	No output given by the morph

Figure 2: Evaluation Methodology for Morph Analyzers

The figure 2 (Amba P Kulkarni, 2010) explains our evaluation methodology for morphological analyzers. Let us continue with the example mentioned in the previous paragraph. First, we find the analysis of the 100 words by the old morph analyzer. We compare its output with the gold output/analysis. Let there be 50 words which belong to Type-1. It means the gold analysis and morphological analysis (by old morph) of 50 words is perfectly equal. Let there be 10 words which belong to Type-6. It means

Table 5: Output analysis of old morph analyzer

Type	Number of instances	% of Type
Type1	*2361*	*47.2*
Type2	*763*	*15.2*
Type3	*419*	*8.4*
Type4	*575*	*11.5*
Type5	*599*	*11.9*
Type6	*288*	*5.8*

Table 6: Output analysis of derivational analyzer

Type	Number of instances	% of Type
Type1	*2600*	*51.9*
Type2	*771*	*15.4*
Type3	*418*	*8.4*
Type4	*576*	*11.5*
Type5	*609*	*12.2*
Type6	*31*	*0.6*

that the old morphological analyzer could not analyze 10 words but there is gold analysis of those words. In this way, each type forms an important part of the evaluation process. Similarly we evaluate the analysis of the 100 words by the derivational analyzer. Finally we compare the evaluations of the old morphological analyzer and our derivational analyzer. This is our evaluation methodology.

So a gold-data consisting of the analysis of 5000 words was taken. The linguistic experts of IIIT Hyderabad have built this data and it was acquired from that institution. The 5000 words were tested on both the derivational analyzer and the inflectional analyzer.

Both the analyzers were tested on the gold-data containing 5000 words. The table 6 proves that the performance of the new derivational analyzer is better than the old morphological analyzer. The old analyzer could not provide any output of 288 words (Type-6) whereas that number is only 31 incase of the derivational analyzer. As a result of this improvement, the overall Type-1 (Perfect output which is completely matching with the gold output) of derivational analyzer is nearly 5% more than that of the old morphological analyzer. The data size is small (only 5000). A testing on a larger gold-data will show an even better picture of the improvement that can be achieved by the derivational analyzer.

6 Conclusions

We presented an algorithm which uses an existing inflectional analyzer for performing derivational analysis. The algorithm uses the main principles of both the Porters stemmer and Krovetz stemmer for achieving the task. The algorithm achieves decent precision and recall. It also expands the coverage of the inflectional analyzer. But it must be incorporated in applications like machine translators which use derivational analysis for understanding its real strengths and limitations.

References

Claudia Gdaniec, Esm Manandise, Michael C. McCord. 2001. *Derivational morphology to the rescue: how it can help resolve unfound words in MT*, pp.129–131. Summit VIII: Machine Translation in the Information Age, Proceedings, Santiago de Compostela, Spain.

Jesus Vilares, David Cabrero and Miguel A. Alonso. 2001. *Applying Productive Derivational Morphology to Term Indexing of Spanish Texts*. In Proceedings of CICLing.

Vishal Goyal, Gurpreet Singh Lehal. 2008. *Hindi Morphological Analyzer and Generator*, pp. 1156–1159. IEEE Computer Society Press, California, USA.

Niraj Aswani, Robert Gaizauskas. 2010. *Developing Morphological Analysers for South Asian Languages: Experimenting with the Hindi and Gujarati Languages*. In Proceedings of LREC.

Ashwini Vaidya. 2009. *Using paradigms for certain morphological phenomena in Marathi*. In Proceedings of ICON.

Bhuvaneshwari C Melinamath, Shubhagini D. 2011. *A robust Morphological analyzer to capture Kannada noun Morphology*, VOL 13. IPCSIT.

William A. Woods. 2000. *Aggressive Morphology for Robust Lexical Coverage*. In Proceedings of ANLC.

Wolfgang Hoeppner. 1982. *A multilayered approach to the handling of word formation*. In Proceedings of COLING.

R. Krovetz. 1993. *Viewing morphology as an inference process*. In Proceedings of COLING.

M. F. Porter. 1980. *An algorithm for suffix stripping*. Originally published in Program, 14 no. 3, pp 130-137.

Bharati Akshar, Vineet Chaitanya, Rajeev Sangal. 1995. *Natural Language Processing: A Paninian Perspective*. Prentice-Hall of India.

Amba P Kulkarni. 2010. *A Report on Evaluation of Sanskrit Tools*.