Sādhanā Vol. 34, Part 5, October 2009, pp. 703–715. © Indian Academy of Sciences

Improving word coverage using unsupervised morphological analyser

K V N SUNITHA and N KALYANI

Computer Science Engineering Department, G Narayanamma Institute of
Technology and Science, Hyderabad 500 008
e-mail: k.v.n.sunitha@gmail.com; narakalyani3@gmail.com

MS received 3 June 2008; revised 25 February 2009

Abstract. Powerful computers are needed for processing tasks related to human languages these days. Human languages, also called natural languages, are highly versatile systems of encoding information and can capture information of various domains. To enable a computer to process information in human languages, the language needs to be appropriately 'described' to the computer, i.e. the language needs to be 'modelled'. In this work, we present an approach for acquisition of morphology of inflectional language like Hindi. It is an unsupervised learning approach, suitable for languages with a rich concatenative morphology. Broadly, our work is carried out in three steps: 1. Acquire the morphology of Hindi from a raw (un annotated) Central Institute of Indian Languages (CIIL), Mysore text corpus, 2. prepare clusters and prepare stem bag and suffix bag, 3. use the morphological knowledge to decompose given word as stems and suffixes according to their morphological behaviour and add new words. A prime motivation behind this work is to eventually develop an unsupervised morphological analyser which is language-independent (used for Hindi). Second motivation is to develop a Morphological segmentation which is language-independent as it is shown that study of morphology would benefit to a range of NLP tasks such as speech recognition, speech synthesis, machine translation and information retrieval.

Though Hindi is an important and a national language in India, little computational work has been done so far in this direction. Our work is one of the first efforts in this regard and can be considered pioneering. There are many such languages for which it is very important to have a suitable but inexpensive computational acquisition process. Languages receive very little attention of computational linguistic research both in terms of availability of funds and number of researchers. We however do not claim that our approach is a solution for all such languages. Different languages have characteristics that require individual research attention.

Keywords. Human languages; unsupervised morphological analyser; clustering; morphological segmentation; linguistic research.

*For correspondence

1. Introduction

Morphological analysis (MA) is an integral part of language processing projects such as text-to-speech synthesis, information extraction, syllable identification or machine translation. Tasks include the separation of stems and affixes (prefixes, suffixes, infixes, crucifixes) and the identification of inflectional and derivational processes. These processes may be productive (i.e. apply to new words entering a language) and may also be combined (especially in agglutinative languages like Turkish) making an enumeration of morphological forms unfeasible as described in Jurafsky *et al* (2000).

Unsupervised approaches to MA are important for less studied (and corpus-poor) languages, where we have small or no machine-readable dictionaries and tools. Ideally, an unsupervised morphological analyser (UMA) would learn how to analyse a language just by looking at a large text in that language, without any additional resources, not even mentioning an expert or speaker of the language. The advantages of unsupervised approach for morphological analysis have been stressed by Hammarstrom *et al* (2006) and Goldsmith (2001). These advantages include elegance, economy, time and money, no additional resources, employment of same technology for new languages, appropriateness for languages with fewer resources (Hammarstrom *et al* 2006).

There is a body of related work that grows faster and faster as briefed in Déjean (1998) first induces a list of 100 most frequent morphemes and then uses those morphemes for word segmentation. His approach is thus not fully unsupervised. Keshava & Pitler (2006) combine the ideas of Déjean (1998). On the Morpho Challenge 2005 datasets, they achieved the best result for English, but they did remarkably worse for Finnish and Turkish. Other UMA learning algorithms exploit the Minimum Description Length (MDL) principle (Mathias Creutz & Krista Lagus 2002). Specifically, EM is used to iteratively segment a list of words using some pre-defined heuristics until the length of the morphological grammar converges to a minimum. Brent *et al* (1995) were the first to introduce an information theoretic notion of compression to represent the MDL framework. Goldsmith (2001) also used an MDL-based approach but applied a new compression system with different measuring of the length of the grammar. Rajeev Sangal *et al* (2001) uses a concept of 'observable paradigm' which considers the frequency of occurrences of word forms in a raw corpus. It generates its feature-structure set to obtain a suitable vector and compares it with reference vector to find most likely stem paradigm. There is a chance of misclassification for low frequency words and rare word forms. Creutz (2003) uses probabilistic distribution of morpheme length and frequency to rank-induced morphemes. He outperforms Goldsmith for Finnish but gets worse results for English.

Various approaches for unsupervised extraction of stems and suffixes have been reported for English, Assamese (Utpal Sharma 2006), Dutch, German and Finnish among others. For the best of our knowledge, this is the first time such approach has been employed on extracting Hindi stems and suffixes. As our approach presents a simple algorithm for unsupervised learning of morphological analysis for inflectionally rich languages like Hindi, given a low coverage morph and a corpus of raw text. It assumes no particular theoretical model of morph, but can used for any language.

We present a brief survey on acquisition of morphology from text corpus, our methodology for extracting valid stems and suffixes from a given text corpus. Technique used for generating new words based on probabilistic models as in Jhon Chen *et al* (2000). Apart from the assumptions given by Hammarstrom, our methodology further assumes words constitute only of stems and suffixes, maximum length of a suffix is 8 and word can consist of only a single suffix. The experiments were conducted on CIIL, Mysore Hindi corpus and the corresponding results are briefed in the next section 3 includes testing and results. The last section consists of conclusion and future scope.

2. Processing of raw text

The details of the complete system is shown in the figure 1.

2.1 *Step 1: Acquisition of morphology from a raw text corpus*

For acquiring morphology of a language from an unannotated text corpus we perform a surface level analysis of the corpus. An unannotated text corpus presents two kinds of information about the language – first, the lexical space of the language, i.e. of the infinite possible letter sequences, the ones that form valid words in the language, and second, the morphological phenomena, i.e. the noticeable similarity in the structure of groups of words. In case of inflectional languages like Assamese, Hindi the predominant morphological phenomenon is suffixation. We model the morphology acquired through analysis of the input training corpus, in the form of a collection of suffixes and the criteria for identifying the presence of these suffixes in different words. This knowledge of morphology is used in building a lexicon that is compact as well as provides more insight about words than a plain listing of the words encountered does. The morphological model and the lexicon can be subsequently used for morphological analysis of words in texts.

Our first task is to identify the underlying suffixes in the language. Suppose, S_C is the set of suffixes identified by the computational process, and S_L is the set of suffixes that are actually

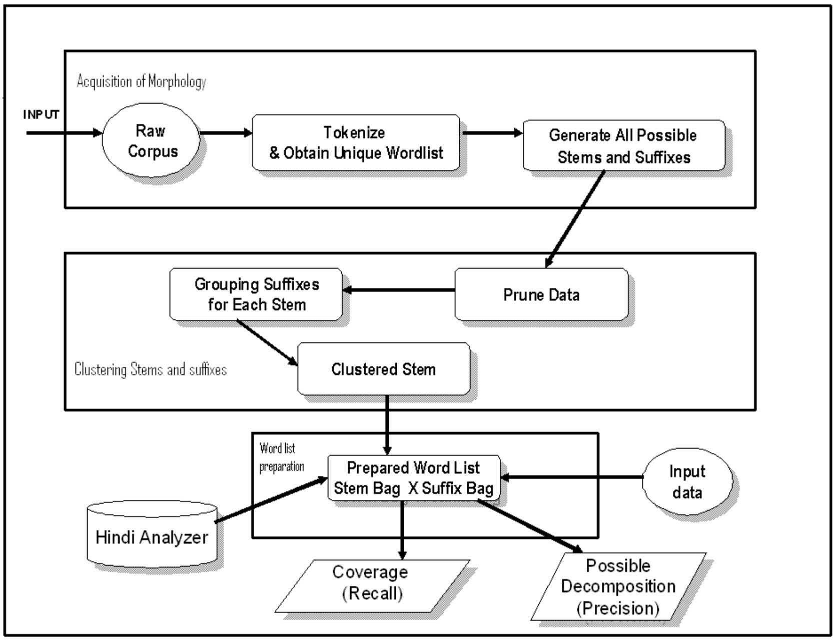
Acquisition of Morphology
INPUT
Raw Corpus
Tokenize & Obtain Unique Wordlist
Generate All Possible Stems and Suffixes
Grouping Suffixes for Each Stem
Prune Data
Clustering Stems and suffixes
Clustered Stem
Word list preparation
Prepared Word List Stem Bag X Suffix Bag
Input data
Hindi Analyzer
Coverage (Recall)
Possible Decomposition (Precision)

Figure 1. Proposed system.

there in the language. The ideal goal of the morphology acquisition process is to have S_C be the same as S_L. However, due to the constraints on the available evidence and the methods applied, S_L is usually not the same as S_C. Letter strings that are not really suffixes are identified as ones, while several valid suffixes are left unidentified. A morphology acquisition method is useful only if the S_C obtained is a close approximation of an underlying S_L. Similar issues arise in the next task of our approach.

The next task is to build a lexicon from a corpus – possibly, the same training corpus. We use the suffixes acquired to decompose the words in the corpus. Simply, looking for matching of the suffixes at the end portion of words leads to a large number of invalid decompositions. Methods discussed in John Goldsmith (2001) and Gaussier (1999) are representative of reported approaches to tackle the problem. In our approach, we apply heuristics based on statistics as well as other language-specific and script-specific aspects. We find that in case of Hindi, and possibly in other languages with similar features, our approach produces better results. We first briefly discuss the two approaches proposed by Gaussier and Goldsmith for identification of suffixes from a text corpus.

2.1a *Gaussier's approach:* Gaussier (1999) presents a method to acquire the suffixes used in a raw text corpus. In the method, a pair of decompositions using a common base is obtained for words W_i and W_j in the input corpus such that both are formed with common base or stem β_1 and suffixes α_1 and α_2, where β_1 is an accepted word in the corpus. The morphological extensions, α_1 and α_2, together referred to as a pseudo-suffix pair, are accepted as valid, if for some words W_k and W_l in the input, can be decomposed with valid stem β_2 and suffixes α_1 and α_2.

To ensure the valid morphological extensions the criteria used are:

(i) Minimum base length, $p = 5$.
(ii) Minimum morphological extension co-occurrence, $c = 2$, i.e. two suffixes occurring with the same base are considered as a pair. In other words, each base must have occurred with at least two morphological extensions 3.
(iii) Minimum morphological extension occurrence, $f = 2$, i.e. each morphological extension must occur with at least two bases.

2.1b *Goldsmith's approach:* Some unsupervised morphology acquisition methods Matthew *et al* (2002) are based on probabilistic models. A particularly interesting approach which can be seen as a special case of probabilistic idea is presented by John Goldsmith (2001). It is based on the Minimum Description Length (MDL) concept. The main intuition in this concept is that if all the morphemes, which are the basic elements of all words, involved in an input are assigned distinct numeric values in the smallest possible number space, then the input can be represented as a sequence of these numbers. Identification of morphemes can be guided by the goal of minimizing the length of the representation of the input corpus, which depends on the total number of morphemes as well as the representation lengths of the individual morphemes in number of bits. The implementation of this approach is available as a free downloadable software called Linguistica.

2.1c *Our approach for morphology acquisition:* To discover the set of suffixes in Hindi from a raw text corpus, our first step is somewhat similar to Gaussier's approach We obtain all the decompositions in each of which a word, $\mathbf{W_1}$: of the corpus is obtained by appending a string of letters $-\alpha$ to another word, $\mathbf{W_2}$, of the corpus. That is

$$\mathbf{W}_1 = \mathbf{W}_2 + \boldsymbol{\alpha}.$$

We refer to this exercise as initial decomposition. The idea is that since $\boldsymbol{\alpha}$ occurs as a suffix for other word in the corpus. Hence, it can be the valid suffix for some decomposition. Since word $\mathbf{W_1}$ has $\mathbf{W_2}$ as its leading portion, it is likely that $\mathbf{W_1}$ is derived from $\mathbf{W_2}$ with $\boldsymbol{\alpha}$ as a morphological extension. The method of Gaussier (1999) detects a morphological extension, $\mathbf{x_s}$, if at least two bases that occur with $\mathbf{x_s}$ also occur with some other, possibly NULL, suffix $\mathbf{y_s}$. i.e. if the words $\mathbf{ax_s}$, $\mathbf{ay_s}$, $\mathbf{bx_s}$ and $\mathbf{by_s}$ occur in the corpus, we get the pseudo-suffix pair $\boldsymbol{(x, y)_s}$. In our initial decomposition we miss a suffix $\mathbf{x_s}$ even if it occurs adequate number of times unless the corresponding bases also occur independently. In Hindi this does not adversely affect the recall since given a good corpus size, stems do occur without the suffixes, too.

Suppose, the set of words in the input corpus is w. We find the set of initial decompositions, D, as

$$D : \{[w = b + x] | w = bx, \text{ and } w, b \in W\}.$$

The set of morphological extensions obtained is

$$E : \{x | [w = b + x] \in D\}.$$

A few sample decompositions are shown below and these are samples of Hindi transcribed text.

acCa = acC + a (□□□□ = □□□□□ + □)
acCA = acC + A(□□□□□ = □□□□□ + □)
acCA = acCA + @(□□□□□ = □□□□□ + @)
acCAaba acC + Aaba(□□□□□□□ = □□□□□ + □□□)

2.1d *Algorithm for implementation:*

(i) Read text corpus and clean the data
ex: In Hindi transcription, some English text is there under header < > Remove that.
(ii) Create unique word list (L) from Corpus
(iii) Generate all possible suffixes and stems using following procedure:
For each unique word (w) in List (L)
(a) Decompose (w) into possible stems and suffixes assuming maximum length of a suffix is maxSuffixLength

stemList	suffixList	stemList	suffixList
printing		walking	
printin	g	walkin	g
printi	ng	walki	ng
print	ing	walk	ing
prin	ting	wal	king
pri	nting	wa	ylking
pr	inting		

Note: In this algorithm, max SuffixLength was taken as 8 and min. length of stem is 2. Examples are quoted in English.

(iv) For each stem, corresponding suffixes were identified ie. bag of suffixes for each stem using the following procedure.

```
for each generated stem (S)
do
  if (S) is not in the (stemList) append(S) into (stemList)
  if (S) is in the (stemList)
    if corresponding (suffix) is not in the (suffixesList) of particular stem (S) append
        (suffix) in to the (suffixList) of particular stem (S)
next.
```

2.2 *Step 2: Clustering stems and suffixes*

Clustering algorithms divide data into meaningful or useful groups, called clusters, such that the intra-cluster similarity is maximized and the inter-cluster similarity is minimized. These discovered clusters can be used to explain the characteristics of the underlying data distribution and thus serve as the foundation for various data mining and analysis techniques. The applications of clustering include characterization of different customer groups based upon purchasing patterns, categorization of documents on the World Wide Web, grouping of genes and proteins that have similar functionality, grouping of spatial locations prone to earth quakes from seismological data, etc.

CLUTO is a software package for clustering low and high-dimensional datasets and for analysing the characteristics of the various clusters. CLUTO provides three different classes of clustering algorithms that operate either directly in the object's feature space or in the object's similarity space. These algorithms are based on the partition, agglomerative, and graph partitioning paradigms. A key feature in most of CLUTO's clustering algorithms is that they treat the clustering problem as an optimization process which seeks to maximize or minimize a particular clustering criterion function defined either globally or locally over the entire clustering solution space. CLUTO provides a total of seven different criterion functions that can be used to drive both partitioned and agglomerative clustering algorithms, that are described and analysed in Zhao & Karypis (2001, 2002). Most of these criterion functions have been shown to produce high quality clustering solutions in high-dimensional datasets, especially those arising in document clustering. In addition to these criterion functions, CLUTO provides some of the more traditional local criteria (e.g. single-link, complete-link, and UPGMA) that can be used in the context of agglomerative clustering. Furthermore, CLUTO provides graph-partitioning-based clustering algorithms that are well-suited for finding clusters that form contiguous regions that span different dimensions of the underlying feature space.

Our approach uses partition algorithm where the number of partitions or clusters are specified. In our experiments we choose number of clusters as 25, 50, 60, 70, 75, and 100 for small corpus to make a study on how language generation gets effected. For corpus of size 40K we used 1000 clusters to generate large vocabulary.

To use CLUTO tool the data should be converted to matrix form which is done by using doc2mat tool is used.

2.2a *Algorithm for step two:*

(i) Drop all the stems that occur below a certain frequency in the entire corpus.
(ii) Drop all the suffixes that occur below a certain frequency in the entire corpus.

Table 1. Statistical details of corpus of size 2000 words and 1–2 million words.

Description	Corpus of size 2000 words	Corpus 1·2 million words
Number of input lines	186	89292
Number of input words	2352	572371
Number of distinct input words (including hyphenated words)	940	40119
Number of stems identified	2504	87291
Number of unique stems identified	1982	22613
Number of suffixes identified	2504	87287
Number of unique suffixes identified	765	9418
Number used for clustering	50	1000
Number of stems clustered	1642	22520
Number of stems not clustered	340	93

(iii) Generate vector matrix file for the pruned data.
* Note: DOC2MAT tool was modified to retain the cases of suffixes unchanged (usually DOC2MAT changes all words in to small case).

(iv) Clustering is carried out by using CLUTO tool, with matrix file as input. By analysing the output of CLUTO tool, different clusters of stems are identified (results are in table 1).

(v) Stem bag and suffix bag is prepared as shown in the figure 2, where the input words are printing, walking, walked, prints, wished watched, watches.

2.3 *Step 3: Technique to improve word list*

Words are generated by combining the stems belonging to a particular stem bag with the corresponding suffix bag. These words are treated as the training data for the next set of new samples.

The main advantage of generating the new words is that as the languages keep growing in the vocabulary size this would help in identifying the new set of words that get generated (figure 3).

Disadvantage is that it generates many words that are not valid. This can be controlled by having the control on the word generation. The best method is while forming the new word

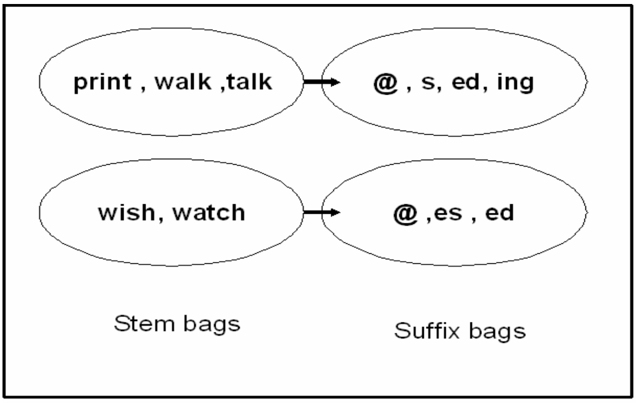

Figure 2. Example of words represented as stems and suffixes.

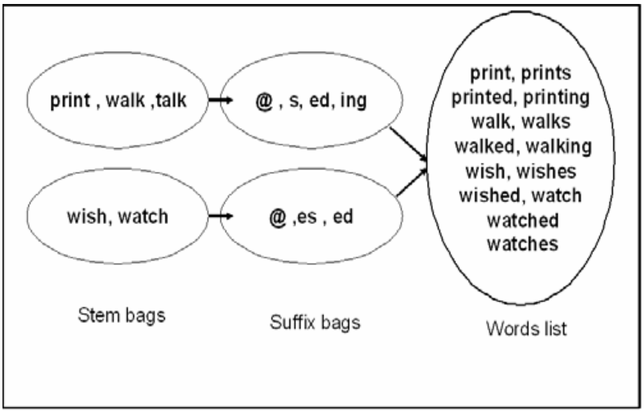

Figure 3. Example of English words generated.

selecting a stem from stem bag and a suffix from corresponding suffix bag choose only those stems and suffixes that have a frequency count that is more than a threshold value. The best threshold value to be chosen is 4 or 5.

2.3a *Algorithm for language generation:*

(i) Group the suffixes of the same cluster.
(ii) Generate wordlist.
(iii) Comparison of unsupervised method with the rule based Hindi Analyser tool is performed.

3. Testing and results

The experiments were conducted on raw CIIL MYSORE Hindi Corpus, collected from Indian Institute of Information Technology (IIIT), Hyderabad. The study was made in various directions to compare the results our system developed with the out put of Hindi Analyser developed by IIIT, Hyderabad.

3.1 *Comparison of rule-based morphological analyser with unsupervised morphological analyser*

3.1a *Calculation of recall precision and F measure:* For two class classification problem differential features are used which are the actual values of the different features of both the classes. In case of decision variables for the two classes are chosen symmetrically around zero and the parameters are estimated from training data. A test sample can then be classified as belonging to class c1 or c2 depending on whether the value of the decision variable is positive or negative.

μ - Number of words in the given test data.
α - Number of stems identified by the rule-based Hindi Analyser.
β - Number of stems identified by the Unsupervised Hindi Analyser, which are also identified by rule-based HA.
γ - Number of stems not identified by the Unsupervised Hindi Analyser, which are identified by rule-based HA.
δ - Number of stems identified by Hindi Analyser that are in modified form.

η - Number of stems identified by Hindi Analyser which are not in modified form

$$\eta = \alpha - \delta$$

$\text{recall}(R) = \beta/\eta * 100$
$\text{precision}(P) = \beta/(\alpha - \gamma) * 100$
$F - \text{measure} = 2 * P * R/(P + R).$

The results are as follows.

Description (2000 word Corpus)	Quantity	Percentage
Total no of words in input after cleaning	940	-
Number of stems that are identified by HA	708	75·32%
Number of stems not identified by Hindi analyser	232	24·68%
Number of stems identified by UMA	474	66·95%
Number of stems that were not identified by UMA which are identified by HA	38	5·37%
Number of stems in modified form	196	27·68%

Description (1·2 million word corpus)	Quantity	Percentage
Total no of words in input after cleaning	40119	-
Number of stems that are identified by HA	18980	47·31%
Number of stems not identified by Hindi analyser	21139	52·69%
Number of stems identified by UMA	12336	64·99%
Number of stems that were not identified by UMA which are identified by HA	1219	6·42%
Number of stems in modified form	5425	13·52%

Treating only those words that are recognized by the rule-based Hindi Analyser

corpus	μ	α	β	δ	η	γ	recall	precision	F-measure
small data	940	708	474	196	512	38	92·58%	100%	96·15%
large data	40119	18980	12336	5425	13555	1203	91·01%	99·87%	95·23%

3.2 *Generation of new words with limited words set*

Words are generated by combining the stems belonging to a particular stem bag with the corresponding suffix bag. These words are treated as the training data for the next set of new samples.

3.2a *Results:*

Total number of words used by UMA $(A_1) = 940$
Total number of words generated by UMA $(A_2) = 43343$

Test data size $(A_3) = 40119$
Test data words that are present in the $A_2 = 1191$
Test data that has common words with small set $(A_3) = 741$
New words that are present in the generated set $= 1191 - 741 = 450$.

Comparisons of different cluster numbers

A_0 - Clustering number used.
A_1 - Number of words analysed by UMA.
A_2 - Number of words generated by UMA.
A_3 - Test data size.
A_4 - Recall corresponding to cluster number.
A_5 - Total words that are covered in the test data.
A_6 - Test data that has common words with small set.
A_7 - New words that are present in the generated set.
$A_7 = A_5 - A_6$

A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7
25	940	63623	40119	92·58	1877	741	1136
50	940	43343	40119	92·58	1692	741	951
60	940	33844	40119	92·58	1585	741	844
70	940	31830	40119	92·58	1287	741	546

On observing the results it is clear that recognition of stems does not depend on the number of clusters used. The word coverage depends on the cluster number used. If the low number is used then it results in generating large number of new words which would improve the range of coverage. If cluster number id high, then it results in generating less number of new words ultimately affecting the coverage of new words.

3.3 *Extracting the word decomposition with the possible weight*

It is observed that very few words had multiple decompositions indicating the ambiguity. To resolve this we have assigned the weight, to decomposition based on the following criteria.

Probability of decomposition is calculated as

$$W_1 * \log(P(\text{stem})) + w2 * \log(p(\text{suffix})) + w_3 * \log(p(\text{stem,suffix}))$$

where $W_1 W_2 W_3$ are assumed to be 1 initially.

P(stem)=frequency of stem/total number of stems.
P(suff)=frequency of suffix /total number of suffixes
p(stem,suff)=frequency of the given stem for the given suffix/total number of stems and suffixes.

It is observed that the decomposition that has higher value is a valid decomposition.

3.3a *Results:*

Actual word	HARstem	Possibilities given by UMA	Probability Value
Alaga(□□□)	alaga(□□□)	**alaga + @ (□□□ + @)** **ala + ga (□□ +)**	−7·76% −26·80%
ammAz (□□□□□□)	ammAz (□□□□□□)	ammAz + @ ((□□□□□□+@)	−7·76%
AzsU(□□□□)	AzsU(□□□□)	AzsU + @ (□□□□ + @) **Azs + U (□□□□ + □)**	−7·76% −28·30%
barawAva (□□□□□)	barawAva (□□□□□)	barawAva + @ (□□□□□ + @) **bara + wAva (□□ + □□□)**	−7·76% −29·51%
kahIM (□□□□)	kahIM (□□□□)	kahIM + @ (□□□□ + @) **ka + hIM (□ + □□□)**	−7·76% −28·30%
Laga(□□)	laga(□□)	laga + @ (□□ +@) **lag + a (□□□ + □)**	−7·76% −18·30%

3.4 *Stem cluster and suffix cluster*

We have used CLUTO clustering tool for clustering the stems that have common suffixes.

Example.

stem bag	suffix bag	Words formed
bulAneh, xewA, WA, sUJA, rahe, nahIMhE, karUz, karoge, jagAyA, jAegA, hUz, hE	ariXana	bulAnehariXana, xewAariXana, WAariXana, sUJAariXana, raheariXana, nahIMhEariXana, karUzariXana, karogeariXana, jagAyAariXana, jAegAariXana, hUzariXana, hEariXana
aKwiy, yahAzwum, ya, xinoM, wum, wowum, wakana, wakagA, un, samaJowum, Ora, na, kI, ja, iwa, in, gAli, BIun, BIna, BE, bedZi, be, bA, Ba, apanI, am	ArA, ArI, Are, Az, AzKAne, Azjo, Azwuma, I, IM, IMAwI, IMhE, IMliyA, a, eM	naArA, naArI, naAre, naAz, naAzKAne, naAzjo, naAzwuma, naI, naIM, naIMAwI, naIMhE, naIMliyA, naa, name BaArA, BaArI, BaAre, BaAz, BaAzKAne, BaAzjo, BaAzwuma, BaI, BaIM, BaIMAwI, BaIMhE, BaIMliyA, Baa, BaeM . . . .
acCA, xinaus, xevaw, Xa, warahaus, us, un, sAs, sab, rUpayo, rAtiyo, rah, neun, me, logaus, lap, koIus, kis, KAw, kareus, jIwum, jis, jEseus, in, hi, hEus, hEab, ha, GudZ, cO, ba, bacc, Ap,	WA, jAwe, jI, kA, kAapanA, kI, kIsAsa, ke, kepAzva, kiyoM, ko, koTarI, logoM, ne, sahasA, se,	acCAWA, acCAjAwe, acCAjI, acCAkA, acCAkAapanA, acCAkI, acCAkIsAsa, acCAke, acCAkepAzva, acCAkiyoM, acCAko, acCAkoTarI, acCAlogoM, acCAne, acCAsahasA, acCAse, baccWA, baccjAwe, baccjI, vvbacckA, bacckAapanA, bacckI, bacckIsAsa, baccke, bacckepAzva, bacckiyoM, baccko, bacckoTarI, bacclogoM, baccne, baccsahasA, baccse,

3.5 *Failure of UMA with respect to HA*

This system could not recognize those stems which were in modified form because of *Sandhi* rules in the language.

3.5a *Results:*

Actual word	Stem by Hindi Analyser	Stem given by UMA	Probability Value
ladZakara □□□□	**Lada** □□	ladZakara + @ □□□□ + @	−7·76%
ladZake □□□□	**ladakA** □□	ladZake + @ □□□□ + @	−7·76%
Lambe □□□□	laMbA □□	**lambe + @** □□□□ **+ @**	−7·76%
Gaye □□□	**jA** □□	gaye + @ □□□ + @	−7·76%
gayI □□□	**jA** □□	gayI + @ □□□ + @	−7·76%

4. Conclusion

Morphological analysis is a very significant step of NLP for highly inflectional languages such as Hindi. Morphology and syntax are two complementary parts of the structural aspects of natural language expression. Because of the structural nature of morphology, simple computational methods can serve as the initial steps for acquisition of morphology of a language and morphological analysis. We have been successful in computational acquisition of the morphology of Hindi using unsupervised method.

We have identified and tackled several issues inherent in using a raw text corpus for acquisition of morphology of a language. We have also dealt with language-specific and script-specific issues in the process.

Our work is particularly significant because apart from acquisition of morphology we could focus on the natural language generation and also on the possible decompositions of the given word with probability of that occurrence.

The Unsupervised Morphological Analyser provided by our method can be directly used for processing of different languages of similar behaviour. This system can be improved by making it semi supervised. That is adding the learning algorithms which could learn the sandhi rules based on the output generated by the Rule based HA and applying these rules while decomposing. If such strategy is used then by having small data we can generate large vocabulary and also improve the performance of morphological analyser.

We are grateful to Mr Srinivas Bangalore AT & T Labs, Mr Sriram, Research Scholar from the LTRC group of IIIT and Prof. Rajeev Sangal, Director of Indian Institute of Information Technology (IIIT), Hyderabad who gave us good suggestions through Natural Language Processing (NLP) Winter school organized at IIIT, Hyderabad which was sponsored by Tata Consultancy Services (TCS). We also thank Central Institute of Indian Languages (CIIL), Mysore for providing text corpus.

References

Brent, Michael R, Sreerama K Murthy, Andrew Lundberg 1995 Discovering morphemic suffixes: A case study in MDL induction. *The Fifth International Workshop on Artificial Intelligence and Statistics*. Fort Lauderdale, Florida

Creutz M 2003 Unsupervised segmentation of words using prior distributions of morph length and frequency. In *Proceedings of the Association for Computations Languages* (ACL'03). Sapporo, Japan 280–287

D'ejean H 1998 Morphemes as necessary concept for structures discovery from untagged corpora. *Workshop on Paradigms and Grounding in Natural Language Learning*. Adelaide, Australia 295–299

Gaussier Eric 1999 Unsupervised learning of derivational morphology from inflectional lexicons. In ACL'99 Workshop Proceedings: Unsupervised Learning in Natural Language Processing, pages 24–30 ACL

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

Hammarstrom H, Wicentowski R, Kondrak G, SIGPHON 2006 A naive theory of morphology and an algorithm for extraction. In *Proceedings of the ACL Special Interest Group on Computational Phonology*, June 2006, New York City, USA, Association for Computational Linguistics (06) 79–88

Jhon Chen, Srinivas Bangalore, Owen Rambow, Marilyn A Walker 2000 Towards Automatic Generation of Natural Language Generation Systems. *In proceedings of the 18th International Conference on Computational Linguistics*

John Goldsmith 2001 Unsupervised learning of the morphology of a natural language. *Computational Linguistics* 27(2): 153–193

Jurafsky D and Schone Patrick 2000 Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 67–72, Lisbon

Keshava S and Pitler E 2006 A simpler, intuitive approach to morpheme induction. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 31–35, Venice, Italy

Matthew G Snover, Gaja E Jarosz, Michael R Brent 2002 Unsupervised learning of morphology using a novel directed search algorithm taking the first step. In *Workshop on Morphological and Phonological Learning, ACL-2002*, pages 11–20, Philadelphia

Mathias Creutz, Krista Lagus 2002 Unsupervised discovery of morphemes. In *proceedings of the 6th Workshop of the ACL Special Intrest Group in Computational Phonology* (SIGPHON), Philadelphia

Rajeev Sangal, Akshar Bharathi, Bendre S M, Pavan Kumar, Aishwarya 2001 *Unsurprised improvement of morphological analyser of inflectionally rich language*. In: Proceedings of NLPRS 2001. Tokyo

Utpal Sharma 2006 Unsupervised learning of morphology of a highly inflectional language. Ph. D thesis submitted to Department of Computer science and Information Technology, Tezpur University, Napaam – Assam India

Zhao Y, Karypis G 2002 Evaluation of hierarchical clustering algorithms for document datasets. In: *CIKM*

Ying Zhao, George Karypis 2001 Criterion functions for document clustering: Experiments and analysis. *Technical Report* TR #01–40, Department of Computer Science, University of Minnesota, Minneapolis, MN. Available on ht ttp://cs.umn.edu/˜karypis/publications.