Morphological and Phonological Learning: Proceedings of the 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON), Philadelphia, July 2002, pp. 1-10. Association for Computational Linguistics.

Unsupervised Learning of Morphology for Building Lexicon for a Highly Inflectional Language

Utpal Sharma* Jugal Kalita† Rajib Das‡

Abstract

Words play a crucial role in aspects of natural language understanding such as syntactic and semantic processing. Usually, a natural language understanding system either already knows the words that appear in the text, or is able to automatically learn relevant information about a word upon encountering it. Usually, a capable system—human or machine, knows a subset of the entire vocabulary of a language and morphological rules to determine attributes of words not seen before. Developing a knowledge base of legal words and morphological rules is an important task in computational linguistics. In this paper, we describe initial experiments following an approach based on unsupervised learning of morphology from a text corpus, especially developed for this purpose. It is a method for conveniently creating a dictionary and a morphology rule base, and is, especially suitable for highly inflectional languages like Assamese. Assamese is a major Indian language of the Indic branch of the Indo-European family of languages. It is used by around 15 million people.

1 Introduction

A child learns his or her native language by *carefully* analyzing sentences that he or she hears. He or she neither consults a dictionary nor gets any clear instructions on grammar or vocabulary. That he or she can relate the sentences to real world events and entities, is often very useful for this learning process. In the case of a computer program, the lack of such correlation must be compensated. The method that we describe here attempts to build a lexicon and learn morphological rules of a language by studying texts of the language and without any direct manual specification of the language.

A motivation for the particular approach taken in this paper for morphological analysis is that in Assamese formation of derivatives from the root or base forms of words is ubiquitous, it being very inflectional. We believe that for further computational processing of Assamese text, it will be very useful to analyze words to identify the root form and the exact nature of derivation used in each case. To handle such a task, unsupervised learning of morphology is useful. Our algorithm uses techniques distinct from those described in [14].

It must be noted that there is no known published computational linguistic work on Assamese at all. Ours is the first and only attempt to work computationally with Assamese, an important and a national language of India, but utterly neglected. Thus, the nature of the work reported here is pioneering.

*Department of Computer Science, Tezpur University, Tezpur - 784028, Assam, India, utpal@tezu.ernet.in
†Department of Computer Science, University of Colorado, Colorado Springs, CO 80933, kalita@pikespeak.uccs.edu
‡Department of Computer Science, Tezpur University, Tezpur - 784028, Assam, India, rkd@tezu.ernet.in

2 Assamese word derivation

Broadly there are two kinds of derivations. First, multiple root words can be combined to form *compounds*, e.g.,

$grantha + melA = granthamelA$ (গ্রন্থমেলা).

This often occurs with some change in spellings of the root words, e.g.,

$par + adhIn = parAdhIn$ (পৰাধীন).

Compound formation in the languages of the Indic branch is of two types namely, using *sandhi* and *samas* ([10], [11]). In this work, we do not target analysis of compounds. The second kind of derivation is by the use of *affixes* (prefixes and suffixes). Affixation is sometimes differentiated into two types, namely *inflectional* and *derivational* (such as in [3]), e.g.,

Inflectional : $mAnuh + e = mAnuhe$ (মানুহ, মানুহে)

Derivational : $driha + tA = drihatA$ (দৃঢ়, দৃঢ়তা, = *firm, firmness*)

Our system, however, treats them alike. Languages of the Indic branch inherit many affixes from Sanskrit, as well as have many affixes specific to each language. To begin with, we lay emphasis on processing suffixes since suffixes are more common in Assamese and embody significant linguistic information. Most common among the Assamese suffixes are inflectional suffixes called *bibhakti*s ([4], [5], [7]) and are distinct from *bibhakti*s of other Indic languages. In Indic languages, particularly in Assamese, in a large number of cases of application of suffixes, spelling changes do not occur. It is also important to note that a very common morphological phenomenon in Assamese is *sequences of suffixes* in a single word. For example, $l'rAkeiTAkeino = l'rA + keiTA + k + ei + no$ (ল'ৰাকেইটাকেইনো: boys+some+objective+emphasis+emphasis).

The frequent occurence of such sequences and the large number of suffixes in some of these sequences is a phenomenon that distinguishes Assamese from most other languages. In some cases Assamese also allows certain suffixes to be detached from the base part. i.e., write the suffix as a separate word following the root word.

Assamese inherits 20 prefixes (*upasarga*s) from Sanskrit ([4], [10], [11]). There are additional prefixes specific to Assamese. Prefixes in many cases change the meaning of words in such a way that the derived words may be treated as root words themselves. Few prefixes, viz., *e, du, Ca*, (these denote numbers one, two, and six of the object) and *na* (also its other forms - *ne, nA, nu, ni,* etc., all of which denote negation) only qualifies the basic meaning of the root word.

3 Decomposing words

From the set of words in the input text, we identify cases where one word can be derived from another word by addition of some suffix (or prefix) in that set. We refer to such a case as a "decomposition", the former word a "derivative", the latter a "base" (or root), and the suffix a "rule". i.e.,

Decomposition: derivative = base + rule

For example,

$karA = kar + A$	(কৰা = কৰ + আ)
$bahalAi = bahal + Ai$	(বহলাই= বহল + আই)

We may be able to find more than one derivative for a single word from the same base by adding different rules. A base may be found to be a derivative with respect to another base. For example, suppose we have the words, *a, ab, ac, abd* (*a,b,c* and *d* are strings possibly longer than a single character). We can represent these as the following.

Abstraction	**Example**	
ab = a + b	$kalamar = kalam + ar$	(কলমৰ)
ac = a + c	$kalamat = kalam + at$	(কলমত)
abd = a + bd,	$kalamarhe = kalam + arhe$	(কলমৰহে)

or, abd = ab + d $kalamarhe = kalamar + he$

In the case of *abd*, we would like to record the decomposition $ab + d$, since it reflects more decomposition information than the other alternative, when considered along with $ab = a + b$.
We summarize our algorithm below

Preprocessing

1. Take a input text T
2. Form a sorted list, L, of distinct words in T.

   Phase 1

3. For each word w in L, identify another word b in L such that w can be obtained by appending some (non-null) suffix s to b. If there are multiple candidates for b, select the longest among them and record the decomposition
   $$w = b + s$$
   If no b can be identified for a w, w is "undecomposed".
4. From the decompositions identified above,
   • for each base b count the number of decompositions where it is a base. Call it the base count of b, bc_b.
   • for each suffix (rule), s, count the number of decompositions where it is a suffix. Call it the rule frequency of s, rf_s.
5. Discard the decompositions in which the rule has a *very low* value α of rf_s. α can be adjusted experimentally. We have used $\alpha = 1$ in our experiments. Also, discard the decompositions in which the base has a *very high* value of bc_b.

   Phase 2

6. For each word that could not be decomposed in phase 1 (some of which may have come out as base in some decomposition), try each of the rules identified in that phase. That is, see if a rule (suffix) is the ending part of an undecomposed word. Record such decompositions, and the base forms.

   Phase 3

7. For each word w in L, identify the set of suffixes (rules) that appear with it and call it the *characteristic* of that word, C_w.
   Words are classified using their characteristics.

Example: [Numbered according to the steps outlined above.] Note that in our experiment the input corpus uses Roman orthography. The mapping between the Assamese letters and the Roman letters is chosen so as to facilitate lossless translation between the two representations. The following example uses this orthography (though it reduces readability to an unaccustomed reader).

2. Suppose the list of distinct words is,
   ..., asm, asmk, asmklE, asmlE, asmr, ..., brdlE, ..., bhArtk, bhArtr, bhArte, ..., kAlilE, k, klm, klmr, kr, ..., mAnuhe, ...
   (..., অসম, অসমক, অসমকলৈ, অসমলৈ, অসমৰ, ..., বৰদলৈ, ..., ভাৰতক, ভাৰতৰ, ভাৰতে, ..., কালিলৈ, ক, কলম, কলমৰ, কৰ ..., মানুহে, ...)
3. Decompositions
   $(asmk = asm + k)$, $(asmklE = asmk + lE)$, $(asmlE = asm + lE)$, $(asmr = asm + r)$, $(klm = k + lm)$*, $(klmr = klm + r)$, $(kr = k + r)$*.
   Undecomposed: *asm, brdlE, bhrtk, bhArtr, bhArte, kAlilE, klm, mAnuhe.*
4. Base counts
   $asm : 3$, $asmk : 1$, $k : 2$, $klm : 1$
   asmklE, asmlE, asmr, brdlE, bhArtk, bhArtr, bhArte, kAlilE, klmr, kr, mAnuhe : 0

   (The suffixes identified are k, lE, r and lm)

Suffix frequencies:
$k : 1$, $lE : 2$, $r : 3$, $lm : 1$

5. Discard the decompositions involving the suffixes k, and lm (since they have frequency $\alpha = 1$). The base count of asm becomes 2, and k becomes 1.

6. Decompositions
$(brdlE = brd + lE)$*,
$(bhArtr = bhArt + r)$,
$(kAlilE = kAli + lE)$

7. Characteristics
$asm : \{lE, r\}$, $asmk : \{lE\}$, $klm : \{r\}$,
$bhArt : \{r\}$, $brd : \{lE\}$, $kAli : \{lE\}$
asmklE, asmlE, asmr, brdlE, bhArtk, bhArtr, kAlilE, klmr : { }

(The decompositions marked * are linguistically incorrect.)

In the first phase, we extract rules by using base forms that are known, i.e., words that are present in the list of words. After this analysis, there may be words which could not be decomposed. It is possible that some of these words are actually root words and cannot be decomposed. But there may be words that could not be decomposed because no base form exists in the list of words provided. So in the second phase, we consider the "undecomposed" words and see if any "rule" identified so far can be applied to decompose such a word, i.e., if ab is word that could not be decomposed so far, is there a rule b so that we can decompose ab as $a + b$? In fact multiple rules might be applicable for a single word. In such a case, in a language where suffix sequences are a common phenomenon, we should give preference to rules that are longer and can be obtained as a suffix sequence (See Section 8). Otherwise suffixes with higher frequencies may be given preference. With more base forms known following the second phase, we can repeat the first phase analysis, then the second phase analysis, and theoretically, so on. In the example given above, if we were to iterate the steps after the first pass, due to the word $bhArt$ in the list now, we can decompose the hitherto undecomposed word $bhArtk$ as $bhArt + k$ and $bhArte$ as $bhArt + e$. This gives us the new suffix e. Assuming e does not get discarded, we can decompose the word $mAnuhe$ as $mAnuh + e$, obtaining the new word $mAnuh$. This process may be stopped when in a particular pass we do not detect any new rule or base. Phase 3 can then be undertaken to determine the *nature* of the words based on the suffixes that each takes. Reaching such a stage, however, does not imply that all possible decompositions have been detected. Since our method works by comparing related words, the presence or absence of certain words makes significant difference. For instance, if the input list of words in the example given above did not contain the word $bhArtr$, the base $bhArt$ would not be obtained, and consequently the suffix e would not be obtained from $bhArte$, and further $mAnuhe$ would not be decomposed. In general, providing a large number of words for analysis will cause more rules to be detected in phase 1, and with the larger number of rules so obtained, more base forms can be detected through phase 2. In practice, it is typical to expect texts in chunks in successive analysis runs. So, during implementation of the algorithm, we refine it so that while a new chunk of text is processed, the words in that text are analyzed along with the words existing in the lexicon. The focus is put on the new words and their effects.

An analysis based simply on detection of presence of common substrings may fail to detect decompositions where the spelling of the derived word is not simply a concatenation of the base word and the suffix but a modification of that. The proportion of such words in a corpus may vary across languages, and so will the effectiveness of the above algorithm. In a language such as Assamese (as also in other Indic languages), it is seen that the modification of spelling is generally related to the actual pronunciation of the words and the suffixes. More specifically, the sounds of the ending part of

the base word and the beginning of the suffix combine to form a sound which is represented in the result by another string of letters than the combination of the original letters. That is, a word ab may combine with a rule xy to form acy instead of $abxy$. We feel that some amount of phonological knowledge, either acquired by a learning method, or provided directly, may be used to detect and handle such cases.

4 Information schema

We accumulate the information extracted by the analysis process in two primary data structures—a lexicon, and a rule base. In the lexicon we keep an entry for each word that has been encountered so far in various runs. Each entry contains the

- the word, w
- the base, b ('-' if the word is not decomposed)
- the number of times it is encountered in text, wf_w.
- the number of times it participates as base in decompositions, bc_w
- other attributes

Similarly, the rule base contains entries each of which comprises

- the rule, s (i.e., suffix, or prefix)
- the number of times it participates in different decompositions, rf_s.
- other attributes

5 Effect of additional text

An effort to build a lexicon and enumerate morphological rules of a natural language, is unlikely to be complete in a single run of the algorithm. It is likely after processing a number of chunks of texts, no new rules are generated. However, base forms are likely to be discovered for a much longer time, and quite possibly, for ever, with new texts. But then, this is the truth about vocabulary! This expectation is borne out by the actual observations in our experiment with 111 chunks of texts depicted in Figure 1 and Figure 2. In Figure 1 we have considered the proportion of distinct unseen words (not base forms, however) to the total words in each text chunk, and in Figure 2 it is the proportion of distinct unseen rules to the total distinct rules in each text chunk.

6 Experimental results

In Table 1 we present the quantitative summary of observations in an experiment with a corpus of 111 news articles containing over 49,000 words in all. These results cover only *Phase 1* of the algorithm described.

With Phase 1 processing only, about 77.28% of the words are decomposed of which 65.4% are correct decompositions. Of the undecomposed words, about 71% are actually root words that should not be decomposed. Of the around 29% words that should have been decomposed but were not, about 3% are compounds and we are not keen on decomposing anyway. In other words we missed about 26% (29 - 3) of possible decompositions.

On applying two simple criteria to eliminate incorrect decompositions (point 5 in the algorithm), viz., decompositions that have root frequencies 20 or higher (because bases occuring with too many suffixes are very short words, usually one or two letters, and match the leading portion of longer words not related to them), and decompositions that have suffix frequencies only $\alpha = 1$, about 64% of the decompositions were retained. Of these about 90.39% were correct decompositions. In the 36% decompositions that were discarded, less than 21% were correct decompositions. It should also be noted that of the 90% correct decompositions, 12.5% are incomplete in that there were more than

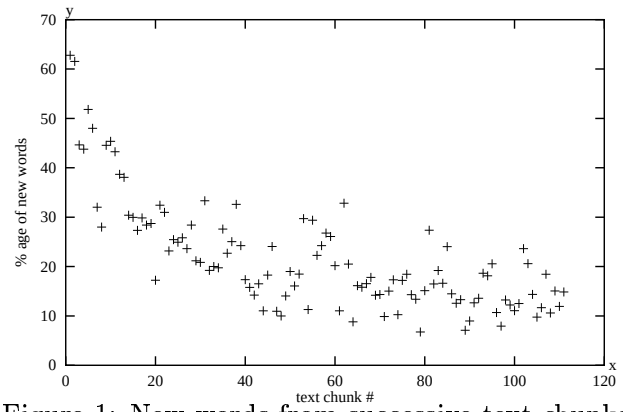

Figure 1: New words from successive text chunks

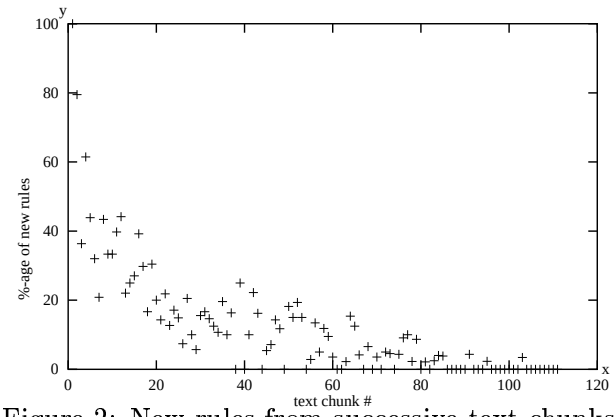

Figure 2: New rules from successive text chunks

Total number of distinct words : 11450

Head	Decomposition Filter			
	All	Root freq < 20 Suff freq > 1	Root freq ≥ 20 Suff freq > 1	Suff freq $= 1$
Decompositions*	8849	5672 = 64.1%	548 = 6.2%	2628 = 29.7%
Precision	65.40%	90.39%		
Recall	79.51%	70.45%		
Incorrect decompositions	34.8%	9.61%	79.20%	79.22%
Complete inflectional suffix decompositions	41.5%	63.54%	13.32%	0.42%
Incomplete inflectional suffix decompositions	7.35%	10.12%	1.28%	2.63%
Complete derivational suffix decompositions	4.35%	6.61%	0.73%	0.49%
Incomplete derivational suffix decompositions	0.57%	0.85%	0%	0.11%
Complete compound decompositions	7.76%	7.62%	5.29%	8.6%
Incomplete compound decompositions	3.55%	1.59%	0.18%	8.52%

* Percentages shown in this row are *wrt* total no. of decompositions

No of undecomposed entries: 2601 = 22.7% of total entries; Of these — Actual base words : 70.85%; Inflectional words : 23.95%; Derivational words : 2%; Compounds : 3.15%

Table 1: Quantitative results with corpus size of 49000 words

two morphemes in the words some of which could not be separated.

Intuitively the value of α should be low due to the "conservative" way of identifying suffixes in phase 1. In the experiment we have chosen $\alpha = 1$ since while higher values only marginally improved the *precision*, the *recall* was worse; for $\alpha = 2 : precision = 91.76\%, recall = 67.16\%$ and for $\alpha = 3 : precision = 92.83\%, recall = 65.11\%$. The *recall* is likely to improve for larger corpus.

The choice of upper limit for the root frequency is, on the other hand, less rigid. We kept it 19. For a value 18, $precision = 90.53\%$, $recall = 70.38\%$; for a value 20, $precision = 90.09\%$, $recall = 70.60\%$; and for a value 21, $precision = 89.97\%$, $recall = 70.60\%$.

6.1 Results from *Linguistica*

Upon running Linguistica ([14]) over the same corpus of 11450 distinct words, the output reported the analysis of only 6040 of the distinct input words, i.e., 52.75%. In these 6040 analysis the precision of decompositions is 92.05% (including about 24% correct but incomplete decompositions), and the recall is 40.62%. If we consider the fraction of the actual base words left rightly undecomposed together with the fraction of the derived words correctly decomposed, then this *extended* precision is 61.54%. In the results of our program, when no filtering is done then the extended precision is 67.5%, and when decompositions with suffix frequency 1 or base frequency above 19 are discarded, then the extended precision is 77.45%.

7 Quality of decompositions

Among the linguistically correct decompositions, our process leaves some decompositions incomplete, and also produces some "false" decompositions. An incomplete decomposition means detecting only some of the multiple suffixes occurring in sequence in a word. To reduce such cases the valid suffix sequences in the language can be identified and this knowledge can be used to (See Section 8), say, break up suffixes identified by the basic algorithm into valid suffix parts. The false decompositions, in general, may happen due to cases such as

- both a and ab are valid words but b is not a true "rule" in the language. We refer to this case as "false rule". An even more difficult case is that b is a true rule in the language, but not in the case of ab, i.e, ab is actually not derived from a. We refer to this case as "false decomposition".
  In the example presented earlier, the rule lm obtained from the decomposition $klm = k + lm$ is a false rule. The decomposition $kr = k + r$ is a false decomposition.

- ab is a word and b is a known rule, but a is not the "base" of ab. We shall refer to this case as "false base". Here, the difficult case occurs when a is a valid word, but it has nothing to do with the word ab. This too is a case of "false decomposition". In the example presented earlier, the base brd obtained from the decomposition $brdlE = brd + lE$ is a false base.

In short, we have the problem of unconfirmed decompositions (i.e., whether the base forms, rules and decompositions are correct). To tackle this problem of unconfirmed decompositions we compute the probabilities that various entities, i.e., (base form, rule and decomposition) tuples are correct. For this, we assign a numeric "confirmation" value to each entity. A higher confirmation value of an entity indicates that it is more likely to be correct, and it is updated as the system processes more and more input. Qualitatively, we may apply the following heuristics to obtain the confirmation values of the entities:

- The rules and decompositions discovered in phase 1 have a high confirmation value, since the base forms assumed in that phase are actually present in the text.

- In phase 2 (where we discover base forms and decompositions) the confirmation value is low. Among such base forms and decompositions, the confirmation value is higher for rules (suffixes) which are long, say at least two consonants. Also, base forms identified in phase 2 have higher confirmation values if it participates in more such decompositions.
- The confirmation value of a decomposition depends on the confirmation value of the base forms and the rules that are involved.

8 Identifying suffix sequences

A language such as Assamese allows certain suffixes to occur together in sequence in words. For example, suffixes x, y and z may occur with a word w as $wxzy$. These suffixes may or may not appear in other arrangements. Identification of such sequences may help extract more linguistic information about the words. To classify words using the suffix characteristics, identification of valid suffix sequences can help in obtaining a kind of *canonicalization* of the suffix characteristic of a word. For example, if it is known that the suffix xzy is actually a sequence x, z, y, due to the words w and $wxzy$ being in a corpus we associate only the suffix x with w and not xzy.
The following simple algorithm can be used to identify valid suffix sequences in the language using the decompositions performed on the words in the input corpus

1. Start from beginning of the lexicon sorted in reverse alphabetical order. Let a string variable sfx_seq contain a suffix sequence, and n_sfx contain the number of suffix components in sfx_seq. Initially sfx_seq contains the NULL string, n_sfx contains 0, and *next decomposition* means the first decomposition.

2. For the next decomposition encountered, say $wxzy = wxz + y$,
   $sfx\_seq \leftarrow y$,
   $n\_sfx \leftarrow 1$

3. Take the base of the decomposition, wxz, and identify the entry for it in the subsequent part of the lexicon.
   If the base appears as a decomposed entry, say, $wxz = wx + z$, then
   $sfx\_seq \leftarrow$ "z " $+$ sfx_seq,
   $n\_sfx \leftarrow n\_sfx + 1$
   repeat this step 3 for the word wx.
   Else (i.e., if the base appears as an undecomposed entry), then if $n\_sfx > 1$ then record the contents of sfx_seq as a valid sequence of suffixes.
   In our current example suppose we find that w is an undecomposed word in the lexicon. So we obtain the suffix sequence $x\ z\ y$.

4. Go to step 2, unless the end of the lexicon is reached.

In our experiment we have identified 488 sequences of suffixes.

9 Word classification using affix knowledge

We have carried out some experiments on classifying words based on the suffixes that have been encountered. These classes should resemble the various known linguistic categories of words. A brief account of our experiment is given below. Recall that we have called the set of suffixes for a word its *characteristic*. We consider suffixes that occur with a frequency of 10 or more. There were 81 such suffixes.

1. *Direct classification based on characteristics*: Form classes of words by exact matching of respective characteristics. This leads to too many classes of words, because in a corpus many words are likely to occur only with a subset of the set of linguistically valid suffixes

for it. Our analysis forms the characteristic for a word on the basis of the suffixes that have been found with the word in that corpus. So if different words of the same linguistic category have different subsets of suffixes in the corpus, they are classified into different categories.

2. *Identifying subsets of characteristics*: One attempt to overcome the drawback of the above method is to assume that at least some words from each true linguistic category will occur with all or almost all valid suffixes for that category. This implies that the characteristics of all words which are actually of the same linguistic category will be subset of the characteristic of the word which has occurred with all possible suffixes. For example, w occurs with all possible suffixes for its linguistic category, and p and q with two different subsets of the suffixes with w. Then the characteristics of p and q will be subsets of the characteristic of w. Thus we classify w, p and q into the same class. Also note that after a characteristic (e.g., of p) has been found to be a subset of another (e.g., of w), other characteristics are not tested for being its (i.e., characteristic of p) subset. It turns out that words that occur with very few suffixes falls in more than one class. This is because in Assamese there are some suffixes that occur with words of multiple linguistic categories.

3. *Classification based on Closures of characteristics*: The drawback of the idea of subsets described above is that for a linguistic category hardly any word occurs with all valid suffixes for that class. To overcome this we modified the idea to *synthesize* the master characteristic of each linguistic category by taking *union* of its tentative subsets that have occurred. We have called this synthesized master characteristic, a *closure*. To compute a closure we decide on a positive number k which we call the *degree of closure*. We start by selecting the largest of all characteristics, and assume that it is the closure. Then sequentially for each remaining characteristic, c, we determine if c has at least k elements common with the closure or c has less than k elements which are all common with the closure. If so, we update the closure by taking its union with c. If during one *pass* of such testing of characteristics the closure actually gets updated, we have to perform another pass considering the characteristics that failed the test in the previous pass(es). This continues till the closure is not updated in a particular pass. Then we proceed to generate another closure by starting with the largest characteristic from among the ones not included in the previous closures. Higher degree of closure leads to more categories to be identified. In our experiment, closure degree of 3 has resulted in 5 categories of words.

4. *Classification based on mutual exclusion of suffixes*: Another observation in this regard is that while there are certain suffixes that apply to words of multiple linguistic category, there are other suffixes which are specific to certain such categories. So we have tried another idea. In this, for each suffix s, we find out the set of suffixes which have occurred together with s by scanning through all the characteristics. This implies that for the remaining suffixes there is no evidence that they occur in the linguistic category where s applies. That is, they are *excluded by* s. Hence we partition the set of characteristics into two: one with characteristics which contain any suffix excluded by s, and another with the remaining characteristics. However, our experiment using this approach has so far produced too many categories of words. Though these categories do not very much map to the known linguistic categories, still it leads us to consider, what we might call

hierarchical classification of words.

Classification of words based on characteristics is likely to work better for linguistic categories that take *more* suffixes. Some of the problems faced in such classification is due to certain words being actually of multiple linguistic categories (*word sense ambiguity*). In Assamese this is comparatively rare, though not altogether absent. For example, *kar* in one sense means "tax" (a noun) and in another it means "do" (verb imperative).

10 Conclusion

We have presented results of on-going unsupervised morphology learning experiments in Assamese, an Indic language. There are no published computational linguistic work in Assamese. There is no available corpus, and we had to build one for our experiments. There is not even one electronic dictionary in Assamese. Our work is preliminary, but with sufficient potential for the future.

References

[1] Shwartz, Steven C., 1986. *Applied Natural Language Processing.* Petrocelli Books, Princeton, New Jersey

[2] Rich, Alaine and Knight, Kelvin, 1991. *Artificial Intelligence, 2e.* Tata McGraw-Hill Publishing Company Limited, New Delhi

[3] Allen, James, 1995. *Natural Language Understanding, 2e.* The Benjamin/Cummings Publishing Company Inc., Redwood City

[4] Bora, Satyanath, 1968. *bahal byaakaran.* Jnananath Bora, Guwahati

[5] Goswami, Golokchandra, 1990. *asamiyaa byaakaranar moulik bisaar.* Bina Library, Guwahati

[6] Choudhury, Bhupendranath, 18e, 1973. *asamiyaa bhaashaar byaakaran, pratham bhaag.* Lawyer's Book Stall, Guwahati

[7] Sarma, Durgashankar Dev, 1977. *sahaj byaakaran.* Assam State Textbook Production and Publication Corporation Ltd., Guwahati-1

[8] Baruah, Hemchandra, 1985 *Hem Kosha, 6e.* Hemkosh Prakashan, Guwahati

[9] Verma, Shyamji Gokul, 1981. *Maanak Hindi Byaakaran Tatha Rachnaa.* Arya Book Depot, New Delhi-5

[10] Whitney, William Dwight, 1977. *Sanskrit Grammar.* Motilal Banarasidass, Delhi.

[11] Whitney, William Dwight, 1979. *Roots, Verb Forms and Primary Derivatives of the Sanskrit Language.* Motilal Banarasidass, Delhi.

[12] Gabor Proszeky and Balazs Kis, "A Unification-based Approach to Morpho-syntactic Parsing of Agglutinative and Other (Highly) Inflectional Languages". *ACl'99 37th Annual Meeting of the Association of Computational Linguistics*

[13] Bharati, Akshar, Chaitanya, Vineet and Sangal, Rajeev, 1995 *Natural Language Processing - A Paninian Perspective.* Prentice-Hall of India Pvt Ltd., New Delhi

[14] Goldsmith, John, "Unsupervised Learning of the Morphology of a Natural Language" *Computational Linguistics*, 27:2 (2001), pp 153-193, Association of Computational Linguistics

[15] Kazakov, Dimitar, "Unsupervised Learning of Naive Morphology with Genetic Algorithms" *Workshop Notes of the ECML/MLnet Workshop on Empirical Learning of Natural Language Processing Tasks*, pp 105-112, April 26, 1997, Prague, Czech Republic