Unsupervised Morphological Analysis of Hindi

Aditi Krishn
Rabi Shanker Guha
Prof. Amitabh Mukherjee

CS365: Introduction to Artificial Intelligence, Course Project
Department of Computer Science and Engineering, IIT Kanpur

Morphological Analysis has been an active area of research in Natural Language Processing. Morphology aims at deriving the structure of a language by exploring word structure. A major breakthrough in this field was Goldsmith's paper 'Linguistica: An Automatic Morphological Analyser' in 2000 which takes an unsupervised approach to learning morphology and hence has widespread application because of its language independency. Since this paper was published, there has been active research in this field and attempts have been made to include phonological rules in the language structure [1]. In this project we have explored Linguistica and extended it to include those cases where even the root morpheme changes in morphological variants.

Introduction

Under this project we are trying to develop an unsupervised program that learns the structure of words in any human language on the basis of the data fed to it, taking as input a raw untagged Hindi corpus without any text preparation (but the corpus must contain spaces or separating symbols since the program analyses the data word by word). The output is in two parts, one is the output of a software Linguistica, when run on Hindi text, which is of the form (Stems)X(Signature) and the other we get by further collapsing words, which are morphological derivations or inflections of the same root word but have dissimilar spelling, into a single stem along with rules of the form:

(Stem) X (rule1: signature), (rule2: signature)

Here, stem is a root form of any word to which affixes can be attached. For example *'friend'* in *'friendship'* or *'chal'* in *'chalta'* are the stems. A Signature is a list of all suffixes (or prefixes) appearing in the given corpus associated with a given stem. Every stem in a corpus has a unique signature; similarly every signature has a unique set of stems associated with it.

In the first output we get:

O_1 = ({दौड, उड}×{null, आन , ना, आना})

O_3 = ({मार}×{ null, आ, ना})

O_4 = ({मर}×{ null, आ, ना, वाया})

In the second part we'll try to merge 'मार' and ' मर ' into a single stem, say मर and attach rules to it. rule1: -- & rule2: ◌ा - , then their signatures, where the i^th^ character in the rule is the *'matra'* associated with the i^th^ *'vyanjan'* in the root.

This kind of setup is based on the attempt to reduce the Minimum Description Length of the grammar.

Morphology

Morphology, in the field of linguistics, can be defined as the identification, analysis and description of a given language's structure like morphemes and other linguistic units such as parts of speech, affixes, words, etc.

The smallest semantically meaningful unit in a language is known as morpheme. Morpheme may or may not stand out as a complete word, in terms of a freestanding unit of meaning. Thus every word comprises of one or more morphemes.

e.g. *'unforgettable'* consists of:

un - a bound morpheme negating the root

forget - a free morpheme, the root in this case

-able – again, a bound morpheme signifying 'do-able'

Classification of Morpheme:

<u>Free and Bound Morpheme</u>

1. Free Morpheme: Stand-alone words that can function independently are called Free Morphemes. e.g. *break*

2. Bound Morpheme: They provide meaning only when associated with other words. e.g. *Un-* and *-able*

<u>Derivational and Inflectional Morpheme</u>

1. Derivational morphemes: They affect the semantic meaning or part of speech of the affected word, when used in conjunction with a root word.

E.g. *Sad (adjective)* to *sadness (noun),* here we attach *-ness.*

2. Inflectional morphemes: They change a noun's number, gender, grammatical mood, aspect, person, etc. Without affecting the word's meaning or class.

E.g. *dog (singular)* to *dogs (plural).*Here we attach as *–s*

<u>Examples of morphological changes in Hindi:</u>

चल *(verb),* चलें *(number),* {चला, चल रहा, चलेगा} *(tense),* चलकर *(adverb),* चलाऊ *(adjective),* {चली, चला} *(gender)*

Previous Work

1. UNSUPERVISED LEARNING OF MORPHOLOGY: (*John Goldsmith*)

The papers discuss the techniques and results of developing an algorithm, which accepts raw linguistic data as input and produces as their output an analysis of data or a grammar, for the purpose of learning morphology on the basis of essentially no prior knowledge except the data. The basic aim described is the determination of location of the breaks between morphemes inside any word.

BASIC ALGORITHM of LINGUISTICA: John Goldsmith proposes to divide the process of morphological analysis into 'a set of heuristics' and 'Minimum Description Length (MDL)' evaluation process. The heuristics then is divided into 'initial bootstrapping heuristics' which determines the first analysis of stems and suffixes, and 'incremental heuristics' which modifies this analysis. The MDL then decides whether the modifications made by the incremental heuristics should be adopted or dropped.

The first part of output in this project is obtained by running Linguistica Program on a large untagged Hindi Corpus.

2. PRIORS IN BAYESIAN LEARNING OF PHONOLOGICAL RULES:
(Sharon Goldwater and Mark Johnson)

The paper describes a Bayesian procedure for unsupervised learning of phonological rules from an unlabelled corpus of training data. Goldsmith's "Linguistica" program's output is taken as the input and it returns a grammar that consists of stems and set of signatures along with a set of phonological rules i.e. insertion, substitution and deletion rules which allows the grammar to collapse many more words into the signature as compared to Linguistica.

Linguistica Output:
O_1= ({work, roll} X {null, ed, ing, er})
O_2= ({carr} X {y, ied, ier})
O_3= ({carry} X {null, ing})
O_4= ({din} X {e, ed, ing, er})

After adding phonological rules:
O_1= ({work, roll, dine, carry} X {null, ed, er, ing})
rule$_1$=if CeeC then e->null
rule$_2$=if CeiC then e->null
rule$_3$=if CyeC then y->i
(Where, C is a consonant)

The second part of output in our project is inspired by this but this method cannot be directly applied to Hindi text since in Hindi generally such transformations do not occur. In Hindi other than suffixes and prefixes changes occur within the morpheme like ‘*shikshak*’ & *‘shaikshikata’*, ‘*maarna’* & *‘marvana’*. We have dealt with this in our algorithm for dissimilar morphological forms with an unsupervised approach.

Linguistica

The following observations will help us understand the approach taken by Goldsmith[1].
A word which has a complex composition in terms of morpheme boundaries is characterised by substrings that have a relatively high frequently in the corpus.
Analysis of such a morpheme structure results in a more compact definition of the grammar. For example the verb '*jump*' is followed by *~s, ~ed, ~ing, ~er, ~*. Such a description would enable us to describe the set *{jumps, jumped, jumping, jumper, jump}* in a more compact way. Hence a measure of the compactness of the analysed grammar can be a measure of the success of the morphological analysis.
At the same time, morphological analysis is not merely breaking up words at boundaries of high frequency substrings. For example not all words ending in *'ing'* are word boundaries like *'sing', 'string', etc*. Also for morphemes like *'ity'* a simple high frequency heuristic would not suffice because the frequency of *'ty'* would always be greater than or equal to *'ity' {dirty, treaty}* and the frequency of *'y*' would be even higher *{sky, fly, etc.}*. At this point the overall context also matters.

Based on these observations the following algorithm for analysing the corpus is proposed:

```
function Analysis(Corpus)

   A := Boot-Strap(Corpus);
   New_A := Increment (A);

   while ( MDL(New_A) < MDL(A) ) do
              A:= New_A;
              New_A := Increment(A);
   end
   return  A;
end
```

Boot-strapping Heuristic:

The bootstrap heuristic is an initial approximation that produces a morphological analysis of a corpus of languages. The approximation is based on the first observation (described above).

The first step is to generate a set of candidate suffixes and stems for further analysis. For this the word is broken up at points of high successor frequency.
For example the word government can be broken up in the following manner:

Stem	**Followed By**	**Count**
gover	n	1
govern	e,i,m,o,s,#	6
governm	e	1

Thus *'govern'* emerges as a clear cut candidate for stem. However such a heuristic sometimes leads to erroneous results. For example, let us take the word *'conservatives'*

String	c	o	n	s	e	r	v	a	t	i	v	e
Successor Frequency	9	18	11	6	4	1	2	1	1	2	1	1

Thus, *'co', 'conse', 'conserv'* and *'conservati'* emerge as possible candidates. This can be eliminated by imposing the following conditions:
Accept cuts with at least 5 letters in stem.
Demand that the successor frequency must be a clear peak 1...N...1
For each stem accept only signatures with 5 stems.
On applying these conditions, *'conserv'* emerges as the most probable stem.

Minimum Description Length:

The minimum description length is the quantitative measure which helps us determine which of the analysis is better in a set of the same. The MDL is independent of the vocabulary and is a mathematical model to determine how well the analysis fits the data. For this we compute the probability (p) assigned to the model and interpret it as compression $-log_2\ (p)$ bits.

Thus, the compressed length of the corpus is:

Frequency of analysed word:

$$-1* \sum_{i=1}^{\substack{Length\\ of\ corpus}} \log prob(word_i)$$

$$=-1* \sum_{i=1}^{\substack{Length\\ of\ word\\ list}} Count(word_i)prob(word_i)$$

W is a word analysed as belonging to stem t and suffix f, in signature σ

$$\text{Freq}(T+F)$$
$$= \text{Freq}(\sigma) * \text{Freq}(T \mid \sigma) * \text{Freq}(F \mid \sigma)$$
$$= \frac{[\sigma]}{[W]} * \frac{[T]}{[\sigma]} * \frac{[F\text{in}\sigma]}{[\sigma]}$$

where, [W] is the total number of words.

Length of Morphology:
The complexity of morphology is more complex, but can be summarized as follows. Almost all structures - structure of a morphology, structure of a grammar— can be understood and presented as a set of lists, in which each item in the list is a "pointer": a connection either to another list, or to a primitive item (such as a letter or phoneme). In the analysis the morphology contains:

- List of suffixes
- List of stems.
- A list of signatures with the associated stems.

The complexity of a list with N items is calculated as follows:
It consists of the sum of the length of each of the pointers on the list where the length of a pointer to an a item *i* on a list is of length *-log₂ prob(i)*, where again the units in which this length is measured is bits.
The length of the statement which makes it explicit that there are exactly N items ~ $\log_2$ N bits to formulate.
And, the total length of the bits used to represent the list.

For example, consider a list of suffixes:
{ed, s, NULL, ing}
The complexity of the list would be:

1. Sum of length of pointers to *'ed', 's', 'NULL', 'ing'* like the length of pointer to 'ed' is 3 because *p(ed) = 1/8.*
2. $log_2\ N = log_2\ 4$ to specify the length of the list.
3. 6*B bytes, where B is the length of each character.

Thus the length of a list of suffixes and a list of stems can be calculated as described above.
The length of the signatures is computed as follows:

$$\sum\nolimits_{\sigma \in \text{Signatures}} \log \frac{[W]}{[\sigma]}$$
$$+ \sum\nolimits_{\sigma \in \text{Signatures}} \log < \text{stems} > + \log < \text{suffixes}(\sigma) >$$

$$+ \sum\nolimits_{\sigma \in \text{Sigs}} (\sum\nolimits_{t \in \text{Stems}(\sigma)} \log \frac{[W]}{[t]}$$
$$+ \sum\nolimits_{f \in \text{Suffixes}(\sigma)} \log \frac{[\sigma]}{[f\text{in}\sigma]})$$

When all of this is summed up, a clear measurement of the complexity of an analysis is produced, and an automated process can determine which of two analyses is to be preferred.

Incremental Heuristics

1. The incremental procedure accepts an analysis and applies the following procedures to come up with a new morphology.
2. Accept any analysis of word if it contains a known stem and a known suffix.
3. Collect any string that precedes a known suffix. This way find all apparent suffixed and use MDL to decide whether it is an improvement or not.
4. Slide stem suffix boundary to the left and use MDL to decide.

Our Algorithm

The output of the above described process consists of a set of stem-suffix pairs. However morphological variants of the same stem like मार & मर cannot be identified by this program. Thus we propose the following approach to take care of such cases:

Step 1: Pre-process the Linguistica Output to filter out the root and suffix pair only
Step 2: Remove the 'matra' from the stems to obtain the base form of it.
Step 3: Sort the list alphabetically
Step 4: Compare consecutive words and club accordingly

This allows concise grouping of morphological forms of words.

```
#Set of matra
#ा ौ ृ ी ो े ् ि ु ॉ ँ ं ै

root := array();
while ( line in File ) do
  foreach word in line
          root[] := findRoot(word)
  end
end
#Sort the list of roots
Sort(root)
Loop through root
  If ( root[i] == root[i-1] )
          Group ( root[i], root[i-1])
end

#Subtracts the root from the word
def findRoot(word):
    s =
DeleteMatraFromWord(word)
    return s
```

Results

The following results have been obtained by running Linguistica on CFLIT corpus of about 53,000 sentences containing more than 60,000 unique words. We get approximately 14,000 stems and 308 suffixes.

The output of Linguistica can be found here.
The output of running our algorithm can be found here.

Result Analysis

The analysis also gives words which do not seem to exist in Hindi grammar. Mistakenly such words may be present in the corpus so Linguistica is detecting them.
E.g.
है {ँ और एड वी}
के {इस क का काल ल हर}

Also, at some places the corpus misses space between words, which must otherwise be present. In these cases Linguistica gives an error. E.g.
को {और}
नहीं {किया तो थी है}

False Positive (suffix-wise examples)
E.g.
-*na* : 1/7 stems
-*ta:* 0/~50 stems
-ik: 0/26 stems

False Negative
Some words whose word count is not enough in the corpus are not displayed. Also, Complex structural morphemes like *'ga utha'* are not displayed because they are separated by word boundaries.

Running our Algorithm on Linguistica Output, we get almost no false negative outputs but many false positive outputs.
E.g.
(correct output)

```
कव       -ि       NULL,ता
         ा-       ्य,

(Incorrect output)
अनज      -ा-      NULL,से,ों,
         ---      ान,
```

Linguistica works well on almost all languages with word boundaries but, if the program has to be run on languages like Thai or Chinese (which do not have word boundaries), then certain word separators have to be introduced.

References

[1] GOLDSMITH, J. 2000. *Linguistica: An Automatic Morphological Analyser.*

[2] GOLDSMITH, J. 2001. *Unsupervised Learning Of The Morphology Of A Natural Language.* Computat. Linguis. 27, 2, 153–198.

[3] GOLDSMITH, J.2004. *An algorithm for the unsupervised learning of morphology.*

[4] GOLDWATER, S. AND JOHNSON, M.2004.*Priors in Bayesian Learning of Phonological Rules.* http://linguistica.uchicago.edu/

[5] GOLDSMITH, J. 2005. *An Algorithm For The Unsupervised Learning Of Morphology.* Tech. rep. TR- 2005-06, Department of Computer Science, University of Chicago. http://humfs1.uchicago.edu/~jagoldsm/Papers/Algorithm.pdf.

[6] *'Hindi Morphology'* by Rajendra Singh and R.K. Agnihotri.
Motilal Banarsidass Publ., 1997

[7] http://www.uknow.gse.harvard.edu/teaching/TC102-407.html