

## **MORPHOLOGICAL ANALYSIS**

### **Problem statement**

The goal of the project was to understand the method behind generating and recognizing the various forms of morphological variants of a given word and then to design files by ourselves to implement the method.

The project assigned to us involved the following sequence of tasks-

- Understand how a morphological analyzer like PC kimmo works.
- Identify and analyze the different types of files that PC kimmo works with.
- Develop rule, lexicon, and grammar files for Bengali words.
- Run the files in PC Kimmo and Jkimmo and show generation and recognition of morphological variants.

### **List of files handled by PC Kimmo**

- The rule file.
- The lexicon file.
- The grammar file.
- The generation comparison file.
- The recognition comparison file.
- The pairs comparison file.
- The synthesis comparison file.
- The synthesis file.

### **Structure of the rule file**

COMMENT <character>  
ALPHABET <symbol list>  
NULL <character>  
ANY <character>  
BOUNDARY <character>  
SUBSET <subset name> <symbol list>  
. (more subsets)

```

.
.
RULE <rule name> <number of states> <number of columns>
  <lexical symbol list>
  <surface symbol list>
  <state number>{: | .} <state number list>
  . (more states)
.
.
. (more rules)
.
.
END

```

The design shown above is the skeleton of the rule file. It is the file where we specify the alphabets to use, the word boundary, the most important component is a set of rules according to which morphological variants are generated from a root word, and a few other constituents.

It is not possible to write a rule file directly. First a file is written in “.txt” format according to a particular syntax which specifies the information shown above. Then using a tool like Kgen the file is converted into a “.rul” file which consists of the necessary finite state transducers to perform the generation of the morphed forms of a word. The file has the structure shown above.

### **The rule file for Bangla**

The format that we followed for our experiment is the format handled by PC Kimmo. Sample information from the first file created:

List of alphabet used-

```

SUBSET Cons k K g G c C j J N t T
SUBSET Voweldep a i I u U R e E o O

```

Written rules-

- RULE +:0 <= liK \_ C e
- RULE +:0 <= kr \_ e C e n
- RULE a:e <= Ka \_ e C e n



```

+ + l i K e @
0 @ l i K e @
1: 1 1 2 1 1 1 1
2: 1 1 2 3 1 1 1
3: 1 1 2 1 4 1 1
4: 1 5 2 1 1 1 1
5: 1 1 2 1 1 0 1

```

RULE "+:0 <= liK\_C e" 6 8

```

___+ + l i K C e @
0 @ l i K C e @
1: 1 1 2 1 1 1 1 1
2: 1 1 2 3 1 1 1 1
3: 1 1 2 1 4 1 1 1
4: 1 5 2 1 1 1 1 1
5: 1 1 2 1 1 6 1 1
6: 1 1 2 1 1 1 0 1

```

The matrix structure specifies information related to finite state transducers of how these machines will be able to generate words like “liKeCi” given its corresponding input.

### **Structure of the lexicon file**

It should be mentioned that the lexicon file is actually subdivided into several parts. There is one Main lexicon, an Initial lexicon, an End lexicon and a few others.

Structure of the main lexicon-

ALTERNATION <alternation name> <sublexicon name list>

. (more ALTERNATIONS)

.

.

FEATURES <feature abbreviation list>

FIELD CODE <lexical item code> U

```
FIELDPCODE <sublexicon code> L
FIELDPCODE <alternation code> A
FIELDPCODE <features code> F
FIELDPCODE <gloss code> G
```

```
INCLUDE <filespec>
. (more INCLUDED files)
.
.
END
```

A sample lexical entry:

```
\lf `knives
\lx N
\alt Infl
\fea pl irreg
\gl N(`knife)+PL
```

The above information is present in a lexicon file that is included in the main lexicon. This structure and the flow of information concerned will become clear after specifying such a file for Bangla:

Now, in order to build the lexicon necessary the following lexicon files were created:

1. The main lexicon
2. The Initial lexicon
3. The Initial and the End lexicon together
4. The verb root lexicon, and
5. The verb suffix lexicon

Our main concern was with the last two lexicon files. In the verb root lexicon we specify a set of verb roots and in the verb suffix file we specify the suffixes related to the root words. The specification is such that when recognizing a morphological variant a flow of information in and through the files results. This situation is shown below with an example:

## The lexicon file for Bangla

Let us say that we want to recognize the word “krCil”. Now the corresponding information for the word in the verb root lexicon is as follows:

```
\lf kr
\lx VROOT
\alt Aspect
\gl kr
```

The above information means that the word kr is a root word, the next place that you have to go is the Aspect and the lexical entry for the word is kr. From here we reach the verb suffix file. The place we hit is the Aspect entry. This entry specifies the tense of a word in relation to its suffix.

```
\lf 0
\lx SADHARON
\alt Time
\gl +saDarN
```

```
\lf +C
\lx GHOTOMAN
\alt Time
\gl +Gvman
```

The information provided is two of the entries for aspect. In the “\lf” field we specify that if this suffix is encountered then the aspect is that written beside “\lx”, the next target is Time and the lexical entry for this aspect is specified as “\gl”. In our example the match is the second entry so the tense is ghotoman. Now we go to time and find a set of choices, for example-

```
\lf +il
\lx ATIT
\alt Person
\gl +AtIt
```

```
\lf +b
\lx BHOBISHSHOT
\alt Person
```

\gl +bhobishat

We find a match for the suffix “+il” and the time is found to be Atit, that is past tense. The last step of the journey is to find the personification of the word. We come across a list of entries like-

\lf 0  
\lx TRITIO  
\alt End  
\gl +tRtIY

\lf +i  
\lx PROTHOM  
\alt End  
\gl +p^rTm

Since we have used up all our suffix components we find a match for 0 or NULL and we say that the personification is Third person. Hence, the final result is the root word id kr and Ghotoman+Atit+Tritio is the sequence for tense, time and person.

### **Problems Faced while working**

The main problem was to identify what suffix entries to give to the lexicon in order to properly identify the aspect, the time and the person. We had to try different combinations for the same suffix in order to make the recognition. Problem arises when, for example, a situation comes such that we need a NULL entry to specify second person, or Ditio purush. Then we have a problem because we already have a NULL entry for third person. Currently, our work has this problem for some suffixes. The solution to problems like this is the use of Feature in the lexicon and the grammar file. We hope to work on these issues soon.

### **Future Work by our group**

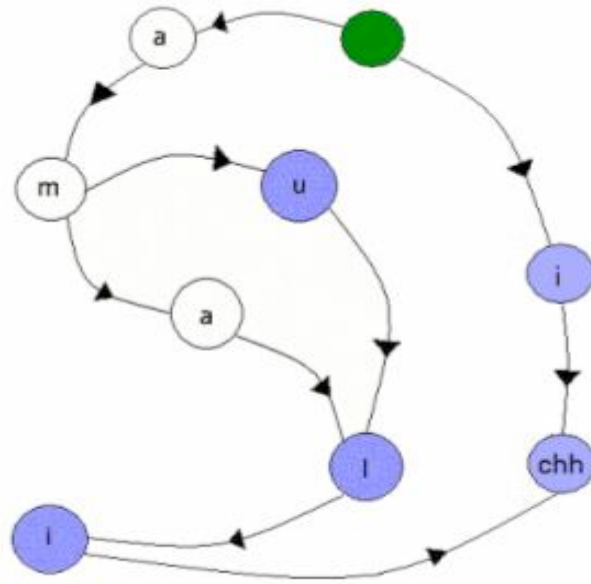
We hope to work on the feature part of the lexicon and also quickly catch up on the grammar file. Another interesting work that we want to do regarding morphological analysis is the analysis for noun roots and prefixes in Bangla.

## **LITERATURE REVIEW**

The Indian Institute of technology of Kharagpur has developed a morphological analyzer for Bangla. They have also provided a web-based version of their software. The analyzer can identify the tense, aspect, modality and person of an inflected verb form. For nouns, the task of the morphological analyzer is to determine its vibhakti (inflection), suffixes and prefixes. The morphological analyzer analyzes the lexical word groups corresponding to the noun and determines the 'karakā' or semantic role.

In the future they hope to perform the decompositions for sandhi and samaasa (conjugating and compounding words) so as to have a powerful vocabulary for the system, and a generalized prefix and suffix handler.

They have developed a Dam structure that can identify possible inflections (changes to the form of the word) by scanning the word backward from the end, one step at a time. In their design there is a start node, which refers to the end of the word. As one progresses backward, we move along a directed acyclic graph depending on the letter that is encountered. There are "possible" terminal nodes, and the information about the morphology can be obtained from the path traversed on the DAG. A given word can have more than one decomposition, for e.g.  $jAnA = jA (to\ go) + nA$  or  $jAnA = jAn (to\ know) + A$ .



A subpart of the DAM structure

### Reference

- [www.sill.org](http://www.sill.org)
- <http://www.mla.iitkgp.ernet.in/>