Morphological Parsing of Bangla Words Using PC-KIMMO

Sajib Dasgupta and Mumit Khan
BRAC University, Bangladesh.
sajib44new@bracuniversity.net, mumit@bracuniversity.net

Abstract

This paper describes Morphological Parsing of Bangla words using PC-KIMMO, based on Kimmo Koskenniemi's model of Two-level Morphology. There are three sections in the PC-KIMMO: rules section, lexicon section and grammar section. We explain here how to write these sections in PC-KIMMO to do morphological analysis for Bangla.

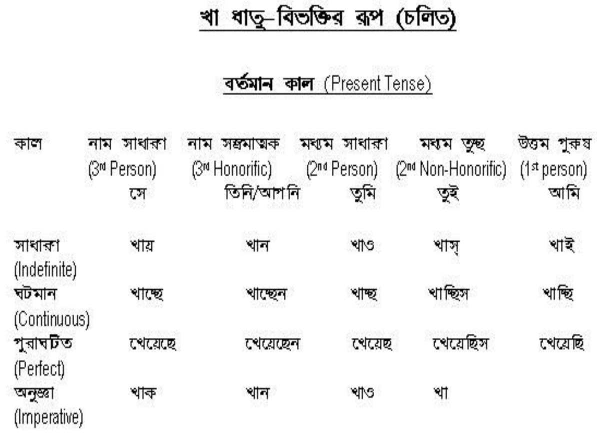
1. Introduction

Morphological analysis is one of the most important subjects in Natural Language Processing (NLP). From MT to Machine Readable Dictionary morphological parsing and generation is absolute requirement. For Bangla there is hardly any work to implement a complete morphological parser or generator. Our aim is to produce a Morphological Parser for Bangla using PC-KIMMO, which is widely used by linguistics around the world for morphological parsing and generation. PC-KIMMO is based on Kimmo Koskenniemi's famous model of Two-level Morphology in which a word is represented as a correspondence between its lexical level form and its surface level form. In this paper we discuss how to incorporate Bangla in PC-KIMMO. [1][3]

2. Why Do Morphological Parsing?

Bangla has a vast inflectional system, the number of inflected and derivational forms of a certain lexicon are huge. For example there are nearly (10*5) forms for a certain verb word in Bengali as there are 10 tenses and 5 persons and a root verb changes its form according to tense and person. For example here are 20 forms of verb root KA (খা).[1]

[1] Through out this paper we have used English alphabet to represent Bangla characters. For example "আ" is "a", "া" is "A", "ি" is "I", "ক" is "k", "খ" is "K", "য়" is "y", "্"(hasanta) is "~" etc.
we have also assumed that the words are given in Unicode Format (vowel comes after consonant). For example খেয়েছি is represented as KEyECI.


খা ধাতু-বিভক্তির রূপ (চলিত)

বর্তমান কাল (Present Tense)

কাল	নাম সাধারণ (3rd Person) সে	নাম সম্ভ্রমাত্মক (3rd Honorific) তিনি/আপনি	মধ্যম সাধারণ (2nd Person) তুমি	মধ্যম তুচ্ছ (2nd Non-Honorific) তুই	উত্তম পুরুষ (1st person) আমি
সাধারণ (Indefinite)	খায়	খান	খাও	খাস্	খাই
ঘটমান (Continuous)	খাচ্ছে	খাচ্ছেন	খাচ্ছ	খাচ্ছিস	খাচ্ছি
পুরাঘটিত (Perfect)	খেয়েছে	খেয়েছেন	খেয়েছ	খেয়েছিস	খেয়েছি
অনুজ্ঞা (Imperative)	খাক	খান	খাও	খা	

Figure 1: Different forms of verb root KA.

Other than this, there are lots of prefixes and suffixes, which can attach with a root word and form a new word. So having an exhaustive list of all these forms in the lexicon will make it huge and space consuming. So in the dictionary we will store only the root form (open morphemes) of a word and suffix, prefixes (closed morphemes) are stored in a known database. Using these we can generate other words via morphological parser.

3. Finite State Morphological Parsing

Here the goal is to divide a Bangla word into smaller subdivisions. For example if a word is given KAc~CE (খাচ্ছে) to the morphological parser it will generate the output

KA + PresentContinuous + ThirdPersonNormal
খা + ঘটমান বর্তমান + নাম পুরুষ

Here খা (KA) is the root morpheme and PresentContinuous, ThirdPersonNormal are morphological features. These features specify the additional information about the stem.

In order to build a morphological parser we need at least the following: (1) Lexicon (2) Morphotactics (3) Orthographic Rules. [4]

3.1. Lexicon

The list of stems and affixes, together with basic information about them (whether a stem is a Noun stem or a Verb stem, etc.). Every lexicon is of a certain class.
Example: Here are two examples

morpheme1:
hAt
Class: Verb_Stem or Root
Feature: Parts of Speech = Verb
morpheme2:
ECI
Class: Tense_Person_Affix
Feature: Tense = Present Perfect
, Person = 1st person

All the lexicons in a certain class are stored in a FSA.

3.2. Morphotactics

The model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes inside a word. For example, the rule that the Bengali Tense_Person_Affixes follow the Verbs, rather than precede it. Normally morphotactics is implemented using Finite State Automata (FSA).

3.3. Orthographic rules

These spelling rules are used to model the changes that occur in a word, usually when two morphemes combine. For example here A is turned into E: [8]
hAt + Present Perfect + 1st person
= hAt + ECI
= hEtECI (হেটেছি)

4. Two-Level Morphological Parsing

Morphological parsing took a new turn with Kimmo Koskenniemi ‘s 1983 dissertation on “Two-level morphology: A general computational model for word-form recognition and generation”. [3] The “Two Level” refers to the two levels of representation of a word – the lexical or underlying form and the surface form. There is a direct, letter-for-letter correspondence between the two levels, which is a natural application of a Finite State Transducer (FST). In the two-level model, a word is represented Koskenniemi's model is "two-level" in the sense that a word is represented as a direct, letter-for-letter correspondence between its lexical or underlying form and its surface form. The word hEtECIlAm for example has the following two-level representation be represented using the following two-levels, where + denotes a morpheme boundary and 0 denotes a null character. [1]

gloss:	hAt + PERFECT + PAST + FIRST_PERSON
lexical:	hAt + EC + Il + Am
surface:	hEt 0 EC 0 Il 0 Am
last:	hEtECIlAm

This is done by extending FSA (which is used to represent lexicon) to add an output for every transition. As the FSA traverses an edge from one state to another, it emits an output symbol. To do this we must add an output alphabet to our definition of Finite State Machine. This yields a Finite State Transducer (FST). This is called transducer because it can work both ways: if the lexical form is given it can generate the surface form (generation) and if the surface form is given it can parse it to lexical form (Parsing).

Normally there is not much difference in between lexical and surface representation. For the above example we can see that h is turned into h, t is always turned into t, + is always turned into 0, and so on. But A is turned into E. So what we have to do is to write a rule that specifies when A is turned into E. So there are two types of correspondence in two level morphological parsing: (1) Default correspondence: like h:h in the previous example (2) Special correspondence: like A:E in the previous example.

The two major components of the two-level model –rules and lexicon– are described below.

4.1. A rules component

It contains phonological rules which specify the special correspondences. The rule notation is of the following form: a->b / c _ d [rule notation of Chomsky and Halle(1968)]. It means a is changed into b if it occurs in between c and d in lexical form.

The rules are represented as finite state devices as are done in string matching algorithm. Representing rules like the above way made the rules work in both parsing and generation stage. The application of the two-level rules is a concurrent process, in contrast to the sequential processing of generative rules.

4.2. A lexical component, or lexicon

The lexicon contains the indivisible words and morphemes in their lexical forms, i.e., the lexical items, as well as the morphotactic constraints.

5. The PC-KIMMO Parser

The first implementation of the two-level model was PC-KIMMO version 1, developed by the Summer Institute of Linguistics in 1990. PC-KIMMO implemented the two components in two files: (i) the rules file (.RUL) which specifies all the orthographic rules, and (ii) the lexical file (.LEX) which specifies all the lexicons, classification of lexicons and morphotactic constraints of these classes. [1]

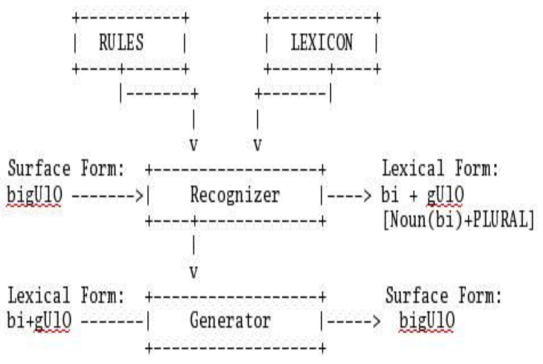


Figure 2: PC-KIMMO parser

The next version of PC-KIMMO, version 2, included a feature-structure unification based chart parser capable of producing parse trees. This feature requires specifying the word grammar, which was added as the third component. The word grammar is specified using a grammar (.GRM) file, so this version requires a total of 3 files for morphological processing. Now we will discuss in details how we wrote these files to implement morphological parser for Bangla. [2]

5.1. The rules section

Here we have to handle all the spelling changes of the root morpheme that are associated with Bangla morphological analysis. Here are two examples: [8]

Rule 1: a:e. If we are to generate the form of a verb for the tense PERFECT and there are a (আ) in the first char and consonant in the last char of the root verb then a (আ) is changed to e (এ).

For example “ak” is a root verb whose first char is a(আ) and last char is k(consonant). So

ak + PERFECT + PRESENT + FIRST_PERSON
= ak + EC+ 0 + I
= ek + EC+ 0 +I
= ekECI

(1)

Rule 2: A:E. If we are to generate the form of a verb for the tense PERFECT and if there are 3 chars in the root and middle char is A (া) then change A to E (ে).

For example “hAt” is a root verb which has 3 chars and middle char is A (া). So

hAt + PERFECT + PRESENT + FIRST_PERSON
= hAt + EC+ 0 + I
= hEt + EC+ 0 +I
= hEtECI

(2)

To incorporate these rules we have to write rules in PC-KIMMO format. Here we will show the contents of .RUL file that handles the above two rule:

Alphabet. Here we use all the English characters to represent the Bangla character. We use one to one mapping for each Bangla character (example “আ” is “a”, “া” is “A”, “ি” is “I”, “ে” is “E”, “ক” is “k”, “খ” is “K”, “য়” is “y”, “্” (hasanta) is “~” etc). NULL, ANY, morpheme boundary and BOUNDARY symbols are as usual 0, @, +, # respectively.

Section of .rul file:
ALPHABET
b c d f g h j k l m n p q r s t v w x y z B C D F G H J K L M N P Q R S T V W X Y Z a e i o u A E I O U + ~ '
NULL 0
ANY @
BOUNDARY #
Subset:
We defined 5 subsets

Section of .rul file:
SUBSET Cons b c d f g h j k l m n p q r s t v w x y z B C D F G H J K L M N P Q R S T V W X Y Z
SUBSET Vowel a e i o u
SUBSET ShortV A E I O U
SUBSET CnsVow b c d f g h j k l m n p q r s t v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Rules. The two special correspondences given above in (1) and (2) are shown below in PC-KIMMO format: [6]

Section of .rul file:
RULE " a:e <=> _ CnsVow* Cons +:0 E C" 9 8

	a	a	CnsVow	Cns	+	E	C	@
	e	@	CnsVow	Cns	0	E	C	@
1:	2	6	1	1	1	1	1	1
2.	0	0	2	3	0	2	3	0
3.	0	0	0	0	4	0	0	0
4.	0	0	0	0	0	5	0	0
5.	0	0	0	0	0	0	1	0
6:	2	6	6	7	1	6	7	1
7:	2	6	1	1	8	1	1	1
8:	2	6	1	1	1	9	1	1
9:	2	6	1	1	1	1	0	1

RULE " A:E <=> +:0 Cons _ Cons +:0 E C" 11 7

	A	A	+	Cons	E	C	@
	E	@	0	Cons	E	C	@
1:	0	1	2	1	1	1	1
2:	0	1	2	3	1	3	1
3:	8	4	2	1	1	1	1
4:	0	1	2	5	1	5	1
5:	0	1	6	1	1	1	1
6:	0	1	2	3	7	3	1
7:	0	1	2	1	1	0	1
8.	0	0	0	9	0	9	0
9.	0	0	10	0	0	0	0
10.	0	0	0	0	11	0	0
11.	0	0	0	0	0	1	0

5.2. The lexicon section

Its primary task is to decompose a word into its constituent morphemes using a simple positional analysis. The positional analysis need only go far enough to ensure that all correct parses are produced but not too many incorrect parses. Co-occurrence restrictions between morpheme positions are best handled in the word grammar, not the lexicon, because that will raise complexities of morphotactic analysis. [4, 5, 6]

Verb. When a verb word is given it is divided into four parts:

1. Verb Root: KA, jA, hAt, GUr
2. Tense: GHATAMAN , PURAGHATITA , SADARAN
3. Time: BARTAMAN, ATIT, VHABISWAT
4. Person: UTTAM, MADDAM, MADDAM_ASAMMAN, NAM NAM_SAMMAN.

For example if a word is given like this:

hEtECIlAm = hAt + EC + Il + Am
= hAt + Tense + Time + Person
= hAt +PURAGHATITA+ATIT+UTTAM

A normal FSA will be as is shown in Figure 3.

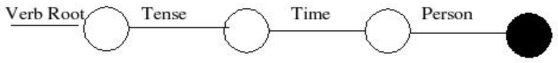


Figure 3: FSA for verb

This obviously is a rather coarse analysis of morphotactic structure, and as such greatly over-recognizes. For example it recognizes both hAtCIlAm (হাটছিলাম) and hAtc~CIlAm (হাটচ্ছিলাম). This incorrect parse can be filtered out in the word grammar component.

Noun. A Noun word consists of an obligatory root (or indivisible stem) preceded by zero or more prefixes and followed by zero or more suffixes. Here is the regular expression of a Noun word.

Noun= (Prefix)* + Nroot + (Suffix)*

For example
অনাধুনিকীকরণের
= অন + আধুনিক + করণ + এর
= Prefix+ Nroot + Suffix + Suffix

While it enforces the relative order of prefixes, roots, and suffixes, it does not enforce any order among prefixes or suffixes. For example,

adUnIktA = ROOT + SUFFIX
= adUnIk + tA
adrtA = ROOT + SUFFIX
= adr + tA

Both follow the same morpheme structure. But we know that the first one is correct and the second one is not [8]. However, this incorrect parse would be filtered out by the word grammar, which knows that the suffix +tA can only attach to an Adjective stem.

The morphotactic analysis shown above is implemented in PC-KIMMO using ALTERNATION declarations in the main lexicon file (.LEX) [6]. The alternation name stands for a positional slot, while the sub-lexicon names stand for the classes of lexical items that can fill that slot.

ALTERNATION Vroot	ROOT1 VROOT2
ALTERNATION Tense	SADARAN
	GHATMAN
	PURAGHATITA
ALTERNATION Time	BARTAMAN
	ATIT
	VHABISWAT
ALTERNATION Person	PERSON1
	PERSON2
	PERSON3
	PERSON4
ALTERNATION Nprefix	NPREFIX
ALTERNATION Nroot	NROOT
ALTERNATION Nsuffix	NSUFFIX
ALTERNATION End	End

Now we will discuss about the different sublexicons we used above. A PC-KIMMO lexicon must start with an INITIAL sublexicon. The INITIAL sublexicon in our implementation contains just two null entries, one showing path for the Verb and other for the Noun.

Some of the lexicons are shown below: (Note: Here \lf is the lexicon \lx is its category \alt is its next state \gl is the output when this lexicon is found, \fea is its features). The features we use here are **root:** v1, v2; **tense:** puraghatita(pg),ghataman(gh) or sadaran(sd); **time:** bartaman(b), atit(a), vhabiswat(v); **person:** uttam(ut), maddam sadaran(ms), maddam asamman(ma), nam sadaran(ns), nam samman(nm).

```
;LEXICON INITIAL
\lf 0
\lx INITIAL
\alt Vroot
\gl

\lf 0
\lx INITIAL
\alt Prefix
\gl

;LEXICON VROOT1(whose last char is consonant)
\lf hAt
\lx VROOT1
\alt Tense
\gl  hAt
\fea v1     ........ [more VROOT1 lexicons]

;LEXICON VROOT2(whose last char is vowel)
\lf  KA
\lx VROOT2
\alt Tense
\gl  hAt
\fea v2     ........ [more VROOT2 lexicons]

;LEXICON SADARAN
\lf   0
\lx  SADARAN
\alt Time
\gl  +SADARAN
\fea sd

;LEXICON GHATAMAN
\lf   C
\lx  GHATAMAN
\alt Time
\gl  +GHATAMAN
\fea gh v1

;LEXICON GHATAMAN
\lf   c~C
\lx  GHATAMAN
\alt Time
\gl  +GHATAMAN
\fea gh v2

;LEXICON BARTAMAN
\lf   0
\lx  BARTAMAN
\alt Person
\gl  +BARTAMAN
\fea b

;LEXICON ATIT
\lf   Il
\lx  ATIT
\alt Person
\gl  +ATIT
\fea a

;LEXICON VHABISWAT
\lf   b
\lx  VHABISWAT
\alt Person
\gl  +VHABISWAT
\fea v

;LEXICON PERSON1
\lf   I
\lx PERSON1
\alt End
\gl  +UTTAM
\fea ut b a
.........[other lexicons of PERSON1 specifying maddam, nam
,.....]

;LEXICON NPREFIX
\lf  bI
\lx NPREFIX
\alt NROOT
\gl an+
......[more prefixes]

;LEXICON NROOT
\lf adr
\lx NROOT
\alt NSUFFIX
```

\gl adr
......[more nouns]

;LEXICON NSUFFIX
\lf tA
\lx NSUFFIX
\alt End
\gl +tA
......[more suffixes]

;LEXICON End
\lf 0
\lx End
\alt #
\gl

5.3. The word grammar section

When the morphemes are given by the lexicon section they are combined using a word parser. Here co-relations between different morphemes are considered using feature unification [2, 7].

In the .GRM file first of all we have to do a feature and category definition. For example:

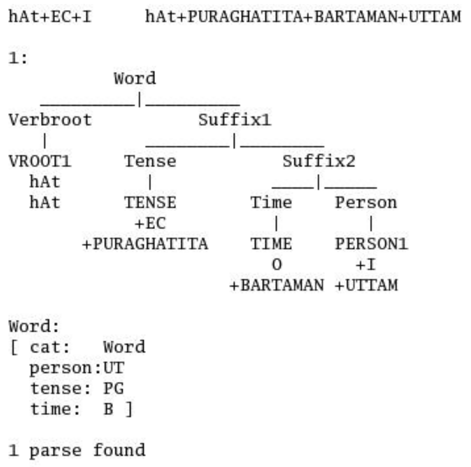
Let v1 be <root> = V1
Let v2 be <root> = V2 [Which assigns root features in root variable]

Let VROOT1 be <cat>=VROOT
Let VROOT2 be <cat>=VROOT
Let GHATAMAN be <cat>=TENSE
Let PURAGHATITA be <cat>=TENSE
Let SADARAN be <cat>=TENSE
Let BARTAMAN be <cat>=TIME
Let ATIT be <cat>=TIME
Let VHABISWAT be <cat>=TIME
Let PERSON1 be <cat>=PERSON
Let PERSON2 be <cat>=PERSON
Let PERSON3 be <cat>=PERSON

The word grammar we used for Bangla is:

```
RULE    Word -> Verbroot  Suffix1
RULE    Suffix1-> Tense Suffix2
RULE    Suffix2 -> Time Person
RULE    Verbroot -> VROOT
RULE    Tense -> TENSE
RULE    Time ->  TIME
RULE    Person->PERSON
RULE    Word -> Prefix NSUFFIX
RULE    Prefix-> NPREFIX NROOT
```

So when we send hEtECI to the recognizer we get the output shown in Figure 4.



```
hAt+EC+I     hAt+PURAGHATITA+BARTAMAN+UTTAM

1:
          Word
   _________|_________
Verbroot          Suffix1
   |         ________|________
VROOT1     Tense          Suffix2
  hAt        |         _____|_____
  hAt      TENSE       Time    Person
            +EC          |         |
       +PURAGHATITA    TIME    PERSON1
                         O       +I
                     +BARTAMAN +UTTAM

Word:
[ cat:   Word
  person:UT
  tense: PG
  time:  B ]

1 parse found
```

Figure 4: PC-KIMMO output

6. Acknowledgement

This work has been partially supported through PAN Localization Project (www.PANL10n.net) grant from the International Development Research Center, Ottawa, Canada, administered through Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, Pakistan. We would like to give them special thanks. We also acknowledge the constant advices and support of all the members of the BRAC University specially Ms. Nupur, Mr. Sheemam Monjel, Mr. Dewan Shariar Hossain Pavel, Mr. Asif Iqbal Sarkar and Mr. Kamrul Haider.

7. Conclusion

PC-KIMMO is an open-source software and its functions can be called from any other program like dictionary, MT, search and replace etc where morphological parsing needs to be incorporated. PC-KIMMO can also handle multiple senses, compound words etc. The analysis we did here for Bangla is quite elementary. Lots of rules still need to be added to make it a complete one. Hopefully this paper can work as a guide for future morphological analysis.

8. References

[1] E.L. Antworth, PC-KIMMO: a two-level processor for morphological analysis. *Occasional Publications in Academic Computing No. 16.* Dallas, TX: Summer Institute of Linguistics, 1990.

[2] E.L. Antworth, Morphological Parsing with Unifcation-based Word Grammar. A paper presented

at North Texas Natural Language Processing Workshop, May 23, 1994.

[3] K. Koskenniemi, Two-level morphology: a general computational model for word-form recognition and production. *Publication No. 11*. Helsinki: University of Helsinki Department of General Linguistics 1983.

[4] D. Jurafsky and J.H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall, 2000.

[5] Lab1abackround.pdf MIT, Department of Electrical Engineering and computer science, 6.863/9.611J spring 2004, NLP. "ai.mit.edu/courses/6.863"

[6] www.sil.org/pckimmo

[7] www.sil.org/pckimmo/v2/doc

[8] S. K. Chottapaday "Vasha Prokash Bangla Bakaran".