Software requirement Specifications for Sanskrit-Hindi Anusaaraka cum Machine Translation System

Amba Kulkarni

Department of Sanskrit Studies,
University of Hyderabad,
Hyderabad
apksh@uohyd.ernet.in / ambapradeep@gmail.com

1 Introduction

This document describes the architecture and functional specifications of the Sanskrit-Hindi Anusaaraka cum Machine Translation System being developed under the Consortium project funded by the MIT, Delhi.
Overall system is developed as a web service. The basic system requirements are:

- Linux operating system (The system has been tested on Ubuntu 10.04)
- g++ compiler
- perl (5.0 or above)
- Python
- flex (A lexical analyser)
- Apache with public_html enabled
- gdbm library (for c as well as perl)
- graphviz (A tool to draw and display graphics)
- lt-toolbox

System uses two more packages CLIPS and Minion which have been included with the distribution. Distribution contains an INSTALL script that installs the packages in user's public_html area.
Programs are written in C, Perl, Flex and Python.

2 Input Output Specifications of Anusaaraka cum MT system

The input is provided through the html form.
The form has three fields:

- Input Encoding
  Input can be given in any of the following encodings.

  - Unicode-Devanagari
  - WX-alphabetic
  - Itrans 5.3
  - Velthuis (VH)
  - Harward Kyoto (KH)
  - Sanskrit Library Project (SLP)

  A link to the encoding table is available online.
- Output Script
  Output can be displayed in either Devanagari script or in Roman Diacritical Notation.
- Text type
  Input text can be either sandhi split or sandhi unsplit.

3 CGI interface

The CGI interface does the following:

- Reads input parameters provided by the user in the HTML form
- Cleans and normalises the text
- Converts the input into WX notation
- Calls Anusaaraka cum MT system with given parameters
- Displays the output in HTML using frames

The input text is stored on the disk in /tmp area. To ensure that the file names are unique and do not clash with other processes invoked parallelly, the filenames are appended with the user process id.

4 Description of Anusaaraka MT system

4.1 Pipeline architecture

Each of the modules either modifies the output of the earlier module by changing the parameter values, or adds the output in a separate column. The main reason behind preserving the intermediary outputs is to view the spectrum of changes taking place after each module at a glance.

4.2 Module 1: Format Saver and Tokeniser

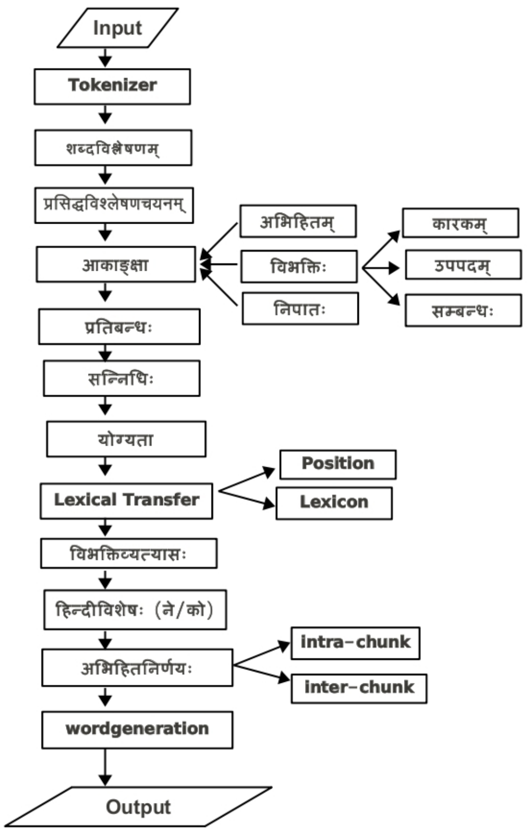
This module preserves the formatting information such as tags associated with the words, paragraph markers, display properties such as bold, italic etc.

It also adds a sentence tag where the sentence is terminated by any of these punctuation marks viz. [.?!].

Here is a sample input file:

संस्कृत-हिन्दी अनुसारक संरचना

Sanskrit-Hindi Anusaaraka Architecture



Sample Input in WX notation:
<TITLE> rAma rAma </TITLE>
<p> aham gqham gawvA pATam paTiRyAmi. </p>
<p> rAmaH gqhe nAswi. </p>

Output:
Output contains 4 fields separated by 'tabs':
1st Field (ID): Identifier with 3 fields separated by '.' indicating paragraph_number, sentence_number and word_number respectively.
2nd Field (LTAG): The punctuation marks and the tags to the left of the word.
3rd Field (TOKEN): The word.
4th Field (RTAG): The punctuation marks and the tags to the right of the word.

The output corresponding to the above file is:

```
1.1.1   <TITLE> rAma
1.1.2           rAma    </TITLE>

2.1.1   <p><s>  aham
2.1.2           gqham
2.1.3           gawvA
2.1.4           pATam
2.1.5           paTiRyAmi       .</s></p>

3.1.1   <p><s>  xaSaraWapuwraH
3.1.2           rAmaH
3.1.3           gqhe
3.1.4           nAswi   .</s></p>
```

A blank line is a terminator for a sentence.

4.3 Module 2: Sandhi splitter

This module is invoked only when the input text type is 'sandhied'.
Sandhi splitter splits the sandhied-words as well as compounds. A compound is split with '-' between the components. For example, in the previous example, the word 'nAswi' is in sandhied form. So this is split into two words as 'na aswi' as shown below. Similarly the compound word 'xaSaraWapuwraH' is split into components as 'xaSaraWa-puwraH'.

```
1.1.1   <TITLE> rAma
1.1.2           rAma    </TITLE>

2.1.1   <p><s>  aham
2.1.2           gqham
```

```
2.1.3           gawvA
2.1.4           pATam
2.1.5           paTiRyAmi       .</s></p>

3.1.1   <p><s>  xaSaraWa-puwraH
3.1.2           rAmaH
3.1.3           gqhe
3.1.4           na
3.1.5           aswi   .</s></p>
```

At this stage if a sandhied word is split into two or more, then the word numbering gets changed.

4.4 Module 3: Morph Analyser

Morph analyser performs 4 tasks:
a) Produce inflectional analysis
b) Prune the answers
c) Use local morph analysis to handle unrecognised words
d) Produce derivational analysis of the derived roots

Input:
Output of the sandhi splitter (or the formatter, if sandhi splitter is not called) is an input to this module. Thus the input is in the form of a table with 4 fields, each field separated by a tab. The output of a sentence is terminated by a blank line.
The four fields are:
1st Field (ID): Identifier with 3 fields separated by '.' indicating paragraph_number, sentence_number and word_number respectively.
2nd Field (LTAG): The punctuation marks and the tags to the left of the word.
3rd Field (TOKEN): The word.
4th Field (RTAG): The punctuation marks and the tags to the right of the word.

Output:
Output consists of 7 fields – the first four fields are the same as those in the input. Morph analyser adds three more fields corresponding to the inflectional morph analysis (MO_MW), inflectional morph analysis after pruning (MO_Apte), and morph analysis after adding the derivational information (MO_Deri).
The MO_MW has all possible answers, the prātipadikas are from the Monier Williams Dictionary. MO_Apte prunes out answers corresponding to rare prātipadikas (prātipadikas not found in Apte's dictionary). MO_Deri supplies the analysis of words after adding the derivational morph output if the pratipadika is derivational. These outputs are stored as columns 5, 6 and 7 respectively.
Since the output of later modules depends on the output of the morphological analyser heavily, a provision is made to supply analysis of unrecognised words manually. This anaysis is provided by the user, and then is used by later modules.

This analysis, when available, is copied to the 6th column, and also carried to the 7th column.
asmax < vargaH:sarva >< lifgam:a >< viBakwiH:1 >< vacanam:1 >< level:1>
gqha < vargaH:nA >< lifgam:napuM >< viBakwiH:1 >< vacanam:1 >< level:1 > / gqha < vargaH:nA >< lifgam:napuM >< viBakwiH:2 >< vacanam:1 >< level:1 >
gam1 < vargaH:avy >< kqw_prawyayaH:kwvA >< XAwuH:gamLz >< gaNaH:BvAxiH >< level:1 >
pATa < vargaH:nA >< lifgam:puM >< viBakwiH:2 >< vacanam:1 >< level:1 >
paT1 < prayogaH:karwari >< lakAraH:lqt >< puruRaH:u >< vacanam:1 >< paxI:parasmEpaxI >< XAwuH:paTaz >< gaNaH:BvAxiH >< level:1 >

In case of derivational morphology, the kṛt_prātipadikas and taddhita_prātipadikas are analysed further for kṛt/taddhita analysis.

Thus, for example for a word 'KAxan' the output is as follows:
KAx1 < kqw_prawyayaH:Sawq >< XAwuH:KAxqz >< gaNaH:BvAxiH >< level:0 >< kqw_pratipadika:KAxaw >< vargaH:nA >< lifgam:puM >< viBakwiH:1 >< vacanam:1 > / KAx1 < kqw_prawyayaH:Sawq >< XAwuH:KAxqz >< gaNaH:BvAxiH >< level:0 >< kqw_pratipadika:KAxaw >< vargaH:nA >< lifgam:puM >< viBakwiH:8 >< vacanam:1 >

Features corresponding to various parts of speech and the allowed range of values are specified in the morph manual and also available here in appendix (A).

4.5 Module 4: Parser

Parser takes the input of the morph layer, and produces the correct morph analysis in the context along with the kāraka analysis in fields 8 (Morph_in_context) and 9 (kāraka_analysis) respectively.
Thus the output of the parser for the previous sentences is shown below. (Only column nos. 3, 8 and 9 are shown)

```
aham asmax<vargaH:sarva><lifgam:a><viBakwiH:1><vacanam:1><level:1>   karwA,5
gqham gqha<vargaH:nA><lifgam:napuM><viBakwiH:2><vacanam:1><level:1>   karma,3
gawvA gam1<vargaH:avy><kqw\_prawyayaH:kwvA><XAwuH:gamLz>
<gaNaH:BvAxiH><level:0>   pUrvakAlaH,5
pATam pATa<vargaH:nA><lifgam:puM><viBakwiH:2><vacanam:1><level:1>     karma,5
paTiRyAmi paT1<prayogaH:karwari><lakAraH:lqt><puruRaH:u>
<vacanam:1><paxI:parasmEpaxI><XAwuH:paTaz>
<gaNaH:BvAxiH><level:1>         aBihiwa_karwA,1

xaSaraWa-puwraH xaSaraWa-puwra<vargaH:nA><lifgam:puM><viBakwiH:1>
<vacanam:1><level:1> viSeRaNam,2
```

```
rAmaH           rAma<vargaH:nA><lifgam:puM><viBakwiH:1><vacanam:1>
<level:1>     karwA,5
gqhe            gqha<vargaH:nA><lifgam:napuM><viBakwiH:7><vacanam:1>
<level:1>   aXikaraNam,5
na              na<vargaH:avy><level:1> sambanXaH,5
aswi as2<prayogaH:karwari><lakAraH:lat><puruRaH:pra>
<vacanam:1><paxI:parasmEpaxI><XAwuH:asaz>
<gaNaH:axAxiH><level:1>aBihiwa_karwA,1
```

4.6 Module 5: Shallow Parser

If a parser fails on any input, then shallow parser does minimum parsing of the sentence, and produces pruned morph output to the next layer.

4.7 Module 6: Word Sense Disambiguation

This module does the sense disambiguation of root, vibhakti and lakāra. Sense tags such as '_1', '_2' etc. are added to the disambiguated entities. This output is in the same form as of morph. The output is stored in the 10th column.

4.8 Module 7: POS

This layer in fact is not needed. But is being used just for evaluating the POS tagger. The POS tagger output is stored in the 11th column. At this layer color code is also added. This is needed to display proper color in the interface. Color code information is added to the 12th column.
Sample output after this layer is shown below. (Only columns 3, 11 and 12)

```
xaSaraWasya nAma-saM N6
puwraH nAma N1
rAmaH nAma-saM N1
nagare nAma N7
koSAw nAma N5
haswena nAma N3
brAhmaNAya nAma N4
gAM nAma N2
xaxAwi kriyA KP
```

4.9 Module 8: Chunker

In this layer some minimum grouping of the words such as 'gacCawi sma' etc. is done. The output is stored in the 13th column. We show below only the relevant columns viz. 10th column in the input and the 13th column in the output.
input:

```
rAma<vargaH:nA><lifgam:puM><viBakwiH:1><vacanam:1><level:1>
    <rel_nm:karwA><relata_pos:3>
vana<vargaH:nA><lifgam:napuM><viBakwiH:2><vacanam:1><level:1>
    <rel_nm:karma><relata_pos:3>
gam1<prayogaH:karwari><lakAraH:lat><puruRaH:pra><vacanam:1>
    <paxI:parasmEpaxI><XAwuH:gamLz><gaNaH:BvAxiH><level:1>
    <rel_nm:aBihiwa_karwA><relata_pos:1>
sma<vargaH:avy><level:1><rel_nm:sambanXaH><relata_pos:3>
```

Output:

```
rAma<vargaH:nA><lifgam:puM><viBakwiH:1><vacanam:1><level:1>
    <rel_nm:karwA><relata_pos:3>
vana<vargaH:nA><lifgam:napuM><viBakwiH:2><vacanam:1><level:1>
    <rel_nm:karma><relata_pos:3>
gam1<prayogaH:karwari><lakAraH:lat_sma><puruRaH:pra><vacanam:1>
    <paxI:parasmEpaxI><XAwuH:gamLz><gaNaH:BvAxiH><level:1>
    <rel_nm:aBihiwa_karwA><relata_pos:1>
-
```

4.10 Module 9: Hindi Lexical Transfer

In this module, the Sanskrit lexicon is mapped to the corresponding Hindi lexicon. The output now is in the format as required by the Hindi generator. The output is stored in column number 14. Here are the column numbers 13 and 14 as input and output.
input:

```
rAma<vargaH:nA><lifgam:puM><viBakwiH:1><vacanam:1><level:1>
    <rel_nm:karwA><relata_pos:3>
vana<vargaH:nA><lifgam:napuM><viBakwiH:2><vacanam:1><level:1>
    <rel_nm:karma><relata_pos:3>
gam1<prayogaH:karwari><lakAraH:lat_sma><puruRaH:pra><vacanam:1>
    <paxI:parasmEpaxI><XAwuH:gamLz><gaNaH:BvAxiH><level:1>
    <rel_nm:aBihiwa_karwA><relata_pos:1>
```

output:

```
rAma n m s a 0
vana n m s a ko
jA v m s a wA_WA
-
```

4.11 Module 10: Hindi Generator

Hindi generator has two main tasks:
a) Sentence level generation: This involves agreement between noun and adjectives, adding 'ne', dropping 'ko' at unnecessary places, handling kriyāmūla vibhakti

and agreement, handling agreement for ṣaṣṭī vibhaki, kartā and kartā_samānādhikaraṇa, and finally agreement between the noun and the verb.

b) Word level generation: At this level, proper form of the word is generated.
Sample input (Only column 14 is shown)

```
sIwA n f s a 0
vana n m s a ko
jA v m s a wA_hE

sIwA n f s a 0
rAma n m s a ko
anusaraNa_kara v m s a wA_hE
```

output after a) (Only column 15 is shown)

```
sIwA n f s a 0
vana n m s a ko
jA v f s a wA_hE

sIwA n f s a 0
rAma n m s a kA
anusaraNa_kara v f s a wA_hE
```

output after b) (Only column 16 is shown)

```
sIwA
vana
jAwI_hE

sIwA
rAma_kA
anusaraNa_karawI_hE
```

4.12 Module 11: Interface display

The interface display takes the output in various columns, and produces an xml document that is converted to an html file later. The output is converted to Roman Diacritic / Devanagari, as per the user's choice. Further the morph o/p and the kāraka relations are transferred to human readable form at this stage.

5 Evaluation

The subjective evaluation of the anusaaraka system will be carried out for comprehensibility and the quality, on a scale of 1 to 5. Appendix describes the scale and evaluation metric.

A File Names: Input, Output and Intermediary

The input text in the HTML form is stored in /tmp area. The file is named after the process_id with ‘in’ as a prefix. Corresponding to each input it generates the following files, where PID is the process ID.

- inPID – This contains the original text.
- inPID.wx – This contains the text converted to WX notation.
- inPID.html – This contains the anusaaraka layered output.
- inPID_trnsltn.html – This contains the final Hindi translation.
- inPID_frame.html – This is the frame invoking various source files.
- inPID_src.html – This contains the original text in html format.

In addition, there are css and js files which describe the style and contain java script for various effects in the output. These files are copied to the user-id/public_html/scl/SHMT/DEMO. All the intermediate files are stored in tmp_inPID directory which can later be removed.

B Morph output specifications

The morph analysis is produced as root followed by feature structure. Feature Structure is an attribute/value pair. Multiple feature structures are separated by ’—’.

Inflectional Morphology According to Pāṇini there are only two basic categories at the level of inflectional morphology. However, for the sake of computational purpose, we also consider avyaya as one of the categories. Later, when we would deal with the Vedic Sanskrit, we may require an additional category, upasarga.

The basic categories for morphological analysis of Sanskrit, therefore, are

- nāmapada (noun)
- kriyāpada (verb)
- avyaya (indeclinable)
- upasarga (pre-position?)

Inflectional morphology
sup: rt,vargaH,lifgam,viBakwiH,vacanam,level
wif: rt,XAwuH,lakAraH,prayogaH,puruRaH,vacanam,paxI,gaNaH,sanAxiH,level
avy: rt,level
upasarga: rt,level (Note: This category is required only for Vedic Sanskrit literature.)

Derivational morphology
avywaxXiwa: rt,waxXiwa_prawyayaH,lifgam,level
avykqw: rt,kqw_prawyayaH,XAwuH,gaNaH,level,sanAxiH
kqw: rt,kqw_prawyayaH,lifgam,viBakwiH,vacanam,XAwuH,gaNaH,kqw_vb_rt,level

waxXiwa: rt,waxXiwa_prawyayaH,waxXiwa_rt,lifgam,viBakwiH,vacanam,level

The values of each of these features for Sanskrit is given below.

- vargaH
  - nA (a nāmapada)
  - sarva (a sarvanāma)
  - saMKyeyam (a cardinal number as an adjective)
  - saMKyA (a cardinal number)
  - pUraNam (an ordinal number)
  - avy (An avyaya)

  In case of derivational morphology, we indicate finer category information. Following are the subcategories of noun.
  - sa-pU-pa (samāsa pūrva pada)
  - sa-u-pa-puM (samāsa uttarapada in pulliṅga)
  - sa-u-pa-napuM (samāsa uttarapada in napunsakaliṅga)
  - sa-u-pa-swrI (samāsa uttarapada in strīliṅga)
  - nA_mawup (derived taddhita)
  - nA_wva (derived taddhita)
  - nA_wamap (derived taddhita)
  - nA_warap (derived taddhita)
  - nA_mayat (derived taddhita)
  - nA_wal (derived taddhita)
  - nA_kAra (derived compound)
  - nA_wqc (derived kṛdanta)
  - nA_Sawq (derived kṛdanta)
  - nA_kwavawu (derived kṛdanta)
- liṅgam
  - puM
  - swrI
  - napuM
  - a (to indicate any possible lifgam, e.g. in case of sarvanāma asmad)
- vacanam
  - 1 (ekavacanam)
  - 2 (dvivacanam)
  - 3 (bahuvacanam)
- puruṣaḥ
  - u (uttama)
  - ma (madhyama)
  - pra (prathama)
- vibhaktiḥ
  - 1 (prathamā)
  - 2 (dvitīyā)
  - 3 (tṛtiyā)
  - 4 (caturthī)
  - 5 (pañcamī)

    - 6 (ṣaṣṭhī)
    - 7 (saptamī)
    - 8 (sambodhana)
  - lakāra
    - lat
    - lit
    - lut
    - lqt
    - lot
    - laf
    - viXilif
    - ASIrlif
    - luf
    - lqf
  - padī
    - AwmanepaxI
    - parasmEpaxI
  - prayogaH
    - karwari
    - karmaNi
    - BAve
  - gaNa
    - 1 BvAxiH
    - 2 axAxiH
    - 3 juhowyAxiH
    - 4 xivAxiH
    - 5 svAxiH
    - 6 wuxAxiH
    - 7 ruXAxiH
    - 8 wanAxiH
    - 9 kryAxiH
    - 10 curAxiH

    List of dhātu_with_it will be given in an appendix.

  - kqw_prawyayaH
    - wqc
    - wumun
    - wavyaw
    - yak
    - Sawq
    - SAnac
    - GaF
    - Namul
    - Nvul
    - Nyaw
    - lyut

    - yaw
    - kwvA
    - lyap
    - kwa
    - kwavawu
    - anIyar
- waxXiwa_prawyayaH
    - wal
    - mawup
    - warap
    - wamap
    - wva
    - vaw
    - wasil
    - karam
    - arWam
    - pUrvaka
    - mayat
    - vAram
    - kqwvasuc
    - xA
    - Sas

Derivational Morphology Here are a few examples of the derivational morphology.

- pATayawi paT1_Nic < prayogaH:karwari >< lakAraH:lat >< puruRaH:pra> < vacanam:1 >< paxI:parasmEpaxI >< XAwuH:paTaz > < gaNaH:BvAxiH >< level:1 >
- Agawya Af_gam1 < vargaH:avy >< kqw_prawyayaH:lyap >< XAwuH:gamLz > < gaNaH:BvAxiH >< level:1 >
- XanavawA Xana < vargaH:nA >< waxXiwa_prawyayaH:mawup >< lifgam:puM¿ < viBakwiH:3 >< vacanam:1 >< level:3 > / Xana < vargaH:nA >< waxXiwa_prawyayaH:mawup > < lifgam:napuM >< viBakwiH:3 >< vacanam:1 >< level:3 >
- gacCawi gam1 < lifgam:puM >< kqw_prawyayaH:Sawq >< XAwuH:gamLz > < gaNaH:BvAxiH >< viBakwiH:7 >< vacanam:1 >< level:2 > / gam1 < lifgam:napuM >< kqw_prawyayaH:Sawq >< XAwuH:gamLz > < gaNaH:BvAxiH >< viBakwiH:7 >< vacanam:1 >< level:2 > / gam1 < prayogaH:karwari >< lakAraH:lat >< puruRaH:pra > < vacanam:1 >< paxI:parasmEpaxI >< XAwuH:gamLz > < gaNaH:BvAxiH >< level:1 >