JOURNAL OF MODERN LANGUAGES

FACULTY OF LANGUAGES AND LINGUISTICS
UNIVERSITY OF MALAYA

VOLUME 22 ISSN NO. 1675-526X 2012

FACULTY OF LANGUAGES AND LINGUISTICS
UNIVERSITY OF MALAYA

Journal of Modern Languages

Jurnal Bahasa Moden

Faculty of Languages and Linguistics
University of Malaya

Journal of Modern Languages
Vol. 22, (2012)

Building Morphological Analyzer for Nepali

Shahid Mushtaq Bhat
Rupesh Rai
Linguistic Data Consortium for Indian Languages
Central Institute of Indian Languages

Abstract
Morphological analyzer is a fundamental tool in Natural Language Processing (NLP) that generates the morphological analyses of a given word-form. It can be used in enhancing the accuracy of POS-Tagging, Chunking, Syntactic Parsing, Word Sense Disambiguation (WSD), Information Retrieval (IR) & Machine Translation (MT) Systems. This paper describes an ongoing effort to develop Nepali morphological analyzer, using an open source platform-Apertium (LT-Toolbox). Since, it is the initial stage of this project; we have confined our work to inflectional morphology. So far, we have covered all the possible categories, as per LDC-IL[1] POS tag-set of Nepali. Currently, the coverage of Nepali Morph-Analyzer is 20,000 words, classified into 219 paradigms.

Keywords: Morphological analyzer, Word and paradigm model, Apertium, LT-Tool Box, Paradigm, Concatenative Morphology, Machine Translation, Devnagri, Transliteration

1. Introduction

Automatic morphological analysis is the basic level of analysis in NLP. The primary requirement in the development of an automatic morphological analyzer is the choice of an appropriate theoretical model and the associated tool that can handle all morphological operations of a language. Owing to the limitations of Item and Arrangement (IA) & Item and Process (IP) models, it was decided that Word & Paradigm-WP (Hockett, 1954) is one such model that can capture all the morphological information of all Indian Languages including Nepali. Since, Apertium is an open-source tool that uses finite-state transducers & operates on the basis of Word and Paradigm (WP) model; we use it for developing Nepali-Morphological Analyzer.

Given the fact that most of natural languages construct words by concatenating morphemes together in strict orders, concatenative morphotactics is highly productive in Nepali and other Indian Languages, particularly, in agglutinative languages like Manipuri, Tamil, Kannada, etc. However, there are also languages like Arabic to which we can’t extend the principle of concatenation as their main morphological operation is infixation. It constitutes

[1] Linguistic data consortium for Indian languages is an 11th plan project of Govt. of India set in Central Institute of Indian Languages (CIIL), Mysore, on the lines of LDC at University of Pennsylvania, USA. Its primary goal is to create annotated quality data of 22 Scheduled Languages of Indian Union.

what is called Non-Concatenative morphology also known as Templatic or Root & Pattern morphology (MacCarthy, 1981). Beyond these concatenative & Non-concatenative operations, some Indian Languages like Urdu, Kashmiri, etc show interplay of both types of morphological operations.

But, Nepali morphology is essentially concatenative in nature. Our experience with in handling different types of morphological patterns confirms our assumption that Apertium is adequate to capture all morphological operations of Nepali. This paper mainly focuses on Nepali inflectional morphology and its implementation in Apertium.

2. Motivation

Since, the primary requirement in the development of morphological analyzer is the choice of an appropriate theoretical model and the associated tool/algorithm that can handle all morphological operations of a language. Our choice of the same was motivated partly by economy of procedures and partly by the reasons given below:

i. Theoretical Model

Instead of positing combination of morphemes into word-forms or generation of such word-forms from the stem by the application of rules, WP (word-based) approach proposes generalizations about the relationships that hold between the word-forms of an inflectional paradigm.

Both IA (morpheme-based-approach) and IP (lexeme-based-approach) fail miserably in explaining the irregular morphology. For example: They assume there are two different morphemes (or two different rules) involved in the formation of 3rd person singular verbal form and plural nominal form (-s) in English but the distinction between them turns out to be artificial. WP model treats such forms as whole words that are related to each other by some analogical rules. Words are categorized on the basis of pattern they fit into.

ii. Tool or Algorithm

Since, using an open source tool is a better choice and an economic way (both in terms of procedures and finances) of developing computational resources for resource poor languages, decision to choose Apertium is a credible one. Further, Apertium is designed to implement WP (by means of Finite State Transducers) which motivates us even more to use it.

3. Nepali in the Context of Indian Languages

Nepali is National language of Nepal and also one of the 22 schedule languages of the Indian Union. This makes it very significant Indo-Aryan language like Urdu. It is mainly spoken in Nepal but also in its neighboring countries like India, Bhutan & Myanmar, by approximately 45 million speakers (Hardie *et al.*, 2011)2. In India it is spoken in Darjeeling district of West Bengal, some parts of Jalpaiguri District, Sikkim, Assam, Manipur, Mizoram, Nagaland and Bhaksu

[2] Himalayan Linguistics, Vol. 10(1). © Himalayan Linguistics 2011 ISSN 1544-7502

Dehradun of Himachal Pradesh. It serves as the lingua franca for Nepali speech community present in this part of South Asia which is highly multi-lingual in nature. It is written in Devnagrai script like many Indian Languages (Hindi, Marathi, Gujarati and Dogri).

Like other Indian languages, Nepali is a resource poor language. However, some computational resources have been developed for Nepali under Project- NeLRaLEC (Nepali Language Resources & Localization for Education and Communication) also known as the Bhasha Sanchar Project3 (Yadava *et al*, 2008). This project created POS annotated corpus resources for Nepali & proved instrumental in NLP awareness in Nepal but more thorough research in Nepali Language Processing was started in the Grammar Checker Development under PAN Localization Project4.

This led to the study of the structure of Nepali grammar and language as well as the components required for development of Grammar Checker. Hence, the work on Morph-Analyzer and Stemmer started in the same project (Bal Krishna Bal & Prajol Shrestha, 2004-2007). However, in India such work started more than a decade back but for some languages only but need was felt to promote the technological development in all Indian languages. So, a serious effort for developing computational resources for all the 22 scheduled languages under a single umbrella-project started with the inception of Linguistic Data Consortium for Indian Languages (LDC-IL), set in CIIL. Therefore, the work on Nepali resource creation & language processing was also undertaken.

Like other Indian languages, Nepali is morphologically rich with highly productive inflectional and derivational morphology. Nepali nouns forms show agreement variants for number, gender and case. They also show diminutive variant forms. Marked adjectives show similar agreement changes for number, gender and case. Nepali verbs inflect to show agreement for number, gender, and case. In addition to these features, Nepali also has different inflections for infinitive, past, non-past, habitual and imperative forms. All these forms for a regular verb are duplicated for transitive and causative forms, thus producing a huge number of inflected variations.

4. Nepali Morphology

Nepali morphology is essentially concatenative or linear in nature with abundance of affixes attached to the roots. It is clearly visible in the feature strings. Dionysius Thrax's Techne (C.100 B.C) – a grammatical sketch of Greek – is not only a role model for contemporary POS descriptions in European languages; it has also become a role model for POS descriptions in Indian languages. It includes a scheme of 8 POS-categories (noun, verb, pronoun, preposition, adverb, conjunction, particle, and article) but the present scheme for capturing inflectional morphology of Nepali includes 9 POS categories & 29 subcategories.

[3] http://www.bhashasanchar.org
[4] http://www.PANl10n.net

i. Noun (N)

A noun is a word that refers to people, animals, objects, substances, ideas, concepts and feelings etc. In Nepali, a noun is usually inflected for gender, number, and case. It is of four types: Common Noun (NC), Proper Noun (NP), Verbal Noun (NV), Spatio-temporal Noun (NST).

CATEGORY: N
TYPE: NC, NP, NV, NST

ATTRIBUTES:	Number (0/sg/pl)[5]
	Gender (0/mas/fem)
	Case (0/dir/obl)
	Case marker
	Particle
	Quantitative

[6]FEATURE STRING: ghara (घर) \ [N.sg.mas.dir.0.0.0]

ii. Verb (V)

A verb refers to events, actions and states. In Nepali, verb is inflected for tense, aspect, mood, and honorificity. It also shows agreement for gender, number and person. It is of two types; Main verb (VM) & Auxiliary verb (VA).

CATEGORY: V
TYPE: VM, VA

ATTRIBUTES:	VM=	Person, Number, Gender, Tense, Aspect, Mood Finite, Voice, Negation, Particle, Honorific
	VA=	Person, Number, Gender, Tense, Aspect, Mood, Finiteness, Negation, Particle

FEATURE STRING: jA(जा)\[VM.2.sg.0.npst.0.imp.fn.act.0.0.0],
cha(छ)\[VA.3.sg.mas.npst.0.imp.fn.0.0.0]

iii. Pronoun (P)

Pronoun is a word which inflects & functions like a noun and substitutes a noun or a noun phrase. It is of four five types; Pronominal (PPR), Reflexive Pronoun (PRF), Relative Pronoun (PRL), Reciprocal Pronoun (PRC), Wh-Pronoun (PWH).

CATEGORY: P
TYPE: PPR, PRF, PRL, PRC, PWH

ATTRIBUTES:	PPR=	Person, Number, Gender, Case, Case marker, Particle, Definiteness, Dimension, Quantitative
	PRF=	Number, Gender, Case, Case marker, Particle, Quantitative
	PRL=	Number, Gender, Case, Case marker, Definiteness, Particle, Quantitative, Dimension
	PRC=	Number, Gender, Case, Case marker, Particle
	PWH=	Number, Gender, Case, Case marker, Particle

[5] Every attribute has a set of values e.g. [Number has (0.sg.pl), Gender has (0.mas.fem), Tense has (0.pst.npst), etc]

[6] Feature strings are actually morphological features [e.g. (N.sg.mas.dir.0.0.0)] associated with a given word-form [e.g. ghara (घर)] without any reference to the actual division of morphemes within the word-form which is not necessary for the current work.

FEATURE STRING: ma(म)\ [PPR.1.sg.0.dir.0.0.0.y.prx.0],
AphnO (own)\[PRF.sg.mas.obl.gen.0.0],
jaskO(जसको)\[PRL.sg.mas.obl.gen.0.0.0.0],
EkaApasa(एकआपस)\[PRC.0.0.dir.0.0],
kO(को)\[PWH.0.0.dir.0.0]

iv. Nominal Modifier (J)

Nominal modifier modifies nominals (i.e. nouns as well as pronouns).

a. *Adjective (JJ)*
An adjective is a word that modifies nouns. Morphologically, adjectives inflect for gender, number, and case. Adjectives have comparative and superlative forms.

b. *Quantifier (JQ)*
A quantifier refers to the quantity of the noun described. They also show Gender, number, and case inflections like Adjectives. A quantifier can be numeral (cardinal, ordinal) or non-numeral.

CATEGORY: J
TYPE: JJ, JQ, JINT
ATTRIBUTES: JJ=Number, Gender, Case, Particle, Degree,
JQ = Gender, Case, Numeral, Particle
FEATURE STRING: rAmrO (राम्रो)\ [J. sg.mas.obl.0.pos]
EuTA(एउटा)\[JQ.0.pl.crd.0]

c. *Demonstrative (D)*
Demonstrative is like pronoun but has a deictic function, i.e. pointing out. So, it will always be followed by a noun, a pronoun or an adjective.

CATEGORY: D
TYPE: DAB, DRL, DWH.
ATTRIBUTES: Person, Number, Gender, Case, Case marker, Particle, Definiteness, Dimension

d. *Quantitative*

DRL= Number, Gender, Case, Case marker, Particle, Quantitative
DWH= Gender, Number, Case, Case marker, Particle, Definiteness, Quantitative
FEATURE STRING: tyO(त्यो)\[DAB.3.sg.mas.dir.0.0.y.dist.0]
jasalE(जसले)\[DRL.0.0.obl.erg.0.0]
kasakO(कसको)\[DWH.sg.mas.obl.gen.0.0.dist]

e. *Adverb (A)*
An adverb is a word that modifies or qualifies the verb. Nepali has two kinds of adverbs, basic and derived. Some basic adverbs are phutta

(फुत्त), besari (बेसरी), chiTO (छिटो), etc. Derived adverbs are from pronouns, and adjectives.

CATEGORY: A
TYPE: AMN
ATTRIBUTES: 000
FEATURE STRING: pratidina (daily)\[A.000]

f. *Postposition (PP)*
Postpositions denote a relationship between two entities. It is placed after the noun or pronoun. Functionally, a postposition and a case marker are same but when it is attached to the noun or pronoun it is called case marker, when it is separate it is called a postposition. In Nepali, with pronouns it is written together whereas with nouns it is written separately.

CATEGORY: PP
TYPE: - 0
ATTRIBUTES: Number, Gender, Case marker, Particle
FEATURE STRING: kO(को) [PP.sg.mas.gen.0]

g. *Participles (L)*
Participle is a non-finite verb form that has some of the characteristics and functions of both verbs and adjectives.

CATEGORY: L
TYPE: LPRV, LPRF, LIPFV
ATTRIBUTES: LPRV=Negation, Particle
LPRF=Number, Gender, Negation, Particle
FEATURE STRING: khAEra (खाएर)\[PFV.0.0]
khAEkO (खाएको)\[PF.sg.mas.0.0]

h. *Particle (C)*
A particle is a lexical item that is indeclinable and has some grammatical or pragmatic meaning.

CATEGORY:C
TYPE: CSIM, CLA, CINT, etc.
ATTRIBUTES: CSIM=Number, Gender, Case
CLA= Number, Gender, Case
FEATURE STRING: jastO(जस्तो)\[CSIM.sg.mas.dir]
vaTA(वटा)\[CLA.sg.0.0], E(ए)\[CINT.000]

5. Apertium or LT-Toolbox

Apertium an open-source platform (both LINUX & WINDOWS compatible) for creating rule-based machine translation (MT) system (Armentano Oller *et al*., 2006; Corbi Bellot *et al*, 2005). It was developed through a number of projects like Open-Source Machine Translation for the Languages of Spain and EurOpen Trad: Open-Source Advanced Machine Translation for the European Integration of the Languages of Spain by the Transducers Research Group. It was initially designed for closely-related Romance languages pairs (such as Spanish–Portuguese, Czech–Slovak, Swedish–Danish etc), but has also been adapted to work better for less related languages like the Indian Languages. It consists of pipeline of lexical processing modules: deformatter, morphological-analyzer, categorical disambiguater, structural and lexical transfer module, morphological-generator, post-generator and reformatter. It is now well established that morphological-analyzer cum generator module can be utilized to develop morphological analyzers and generators for all Indian languages. The only effort that one has to be made in this direction is to identify paradigms and add language specific data in the XML schema of the tool.

6. Transliteration Scheme TOOL

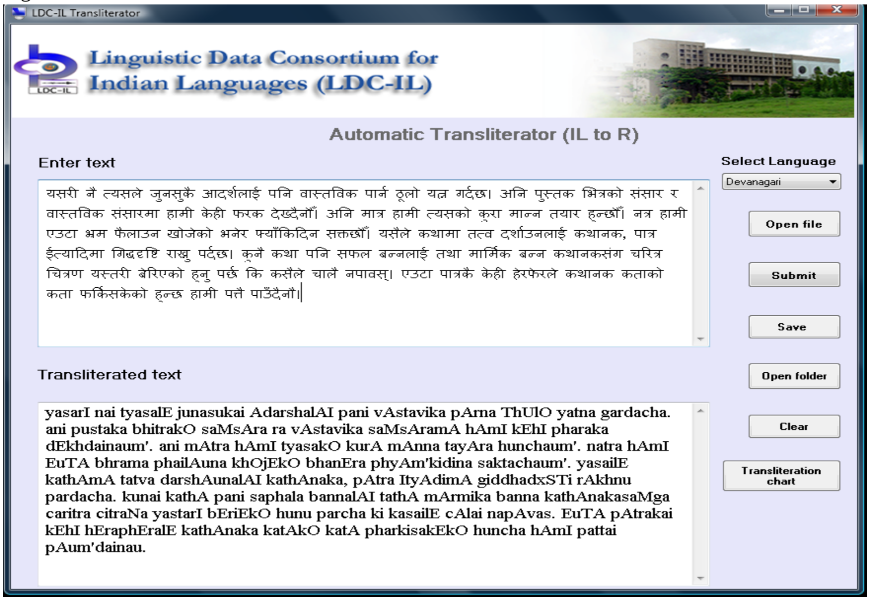
Since the tool, morphological analyzer-cum-generator, supports only roman characters properly, Indic or Persio-Arabic words need to be transliterated first before one starts the actual work on paradigm building. For this purpose, a well-defined transliteration module is required. There are many open source romanizing tools available based on different transliteration schemes. One such tool has been developed by LDC-IL for romanizing all Indian Scripts. Such tools vis-à-vis schemes are very essential to benefit from the computational tools developed across the world which mostly support roman character.

In the present work, we have used the LDC-IL transliteration tool which is predominantly based on different mapping schemes but script grammar rules have been also incorporated to set constraints on the mapping process for better results. The transliteration system has made the present work very easy. The Roman-Devanagari transliteration scheme for Nepali is as under:

[अ-a, आ-A, इ-I, ई-I, उ-u, ऊ-U, ऋ-x, ॠ-X, ए-E, ऐ-ai, ओ-O, औ-au, क-ka, ख-kha, ग-ga, घ-gha, ङ-ng’a, च-ca, छ-cha, ज-ja, झ-jha, ञ- nj’a, ट-Ta, ठ-Tha, ड-Da, ढ-Dha, ण-Na, त-ta, थ-tha, द-da, ध-dha, न-na, प-pa, फ-pha, ब-ba, भ-bha, म-ma, य-ya, र-ra, ल-la, व-va, श-sha, ष-Sa स-sa, ह-ha, ड़-D’a, ढ़-D’ha, क्ष-kSa, त्र-tra, ज्ञ- jnj’a]

So, a well-defined transliteration scheme as given above & the associated transliteration tool as shown in Figure 1 is prerequisite for computing morphology of Nepali, using LT-Toolbox.

Figure 1. Transliteration Tool interface



यसरी नै त्यसले जुनसुकै आदर्शलाई पनि वास्तविक पार्न ठूलो यत्न गर्दछ। अनि पुस्तक भित्रको संसार र वास्तविक संसारमा हामी केही फरक देख्दैनौँ। अनि मात्र हामी त्यसको कुरा मान्न तयार हुन्छौँ। नत्र हामी एउटा भ्रम फैलाउन खोजेको भनेर फ्याँकिदिन सक्तछौँ। यसैले कथामा तत्व दर्शाउनलाई कथानक, पात्र ईत्यादिमा गिद्धदृष्टि राख्नु पर्दछ। कुनै कथा पनि सफल बन्नलाई तथा मार्मिक बन्न कथानकसंग चरित्र चित्रण यस्तरी बेरिएको हुनु पर्छ कि कसैले चालै नपावस्। एउटा पात्रकै केही हेरफेरले कथानक कताको कता फर्किसकेको हुन्छ हामी पत्तै पाउँदैनौं।

yasarI nai tyasalE junasukai AdarshalAI pani vAstavika pArna ThUlO yatna gardacha. ani pustaka bhitrakO saMsAra ra vAstavika saMsAramA hAmI kEhI pharaka dEkhdainaum'. ani mAtra hAmI tyasakO kurA mAnna tayAra hunchaum'. natra hAmI EuTA bhrama phailAuna khOjEkO bhanEra phyAm'kidina saktachaum'. yasailE kathAmA tatva darshAunalAI kathAnaka, pAtra ItyAdimA giddhadxSTi rAkhnu pardacha. kunai kathA pani saphala bannalAI tathA mArmika banna kathAnakasaMga caritra citraNa yastarI bEriEkO hunu parcha ki kasailE cAlai napAvas. EuTA pAtrakai kEhI hEraphEralE kathAnaka katAkO katA pharkisakEkO huncha hAmI pattai pAum'dainau.

7. Computing Nepali Morphology

The process of computing morphology of Nepali, simply, involves the manipulation of strings or words forms (letter transducers), with or without much consideration to real morphemic division. For illustration, we will show the implementation of inflectional morphology which can be divided into the following steps:

Step 1. Defining Elements

All the characters and terms <sdefs> (Characters, Categories, Subcategories, Attributes and Attribute values) that are used in Transliteration scheme and Morphological analysis are defined in the XML code as shown below for Noun:

```
<?xml version="1.0"?
<dictionary>
<alphabet>abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ-
'</alphabet>
<sdefs>
<sdef n="cat:N" c="Noun"/>
<sdef n="subcat:NC" c="Common noun"/>
<sdef n="subcat:NP" c="Proper noun"/>
<sdef n="subcat:NV" c="Verbal noun"/>
<sdef n="subcat:NST" c="Spatio-Temporal noun"/>
<sdef n="gender:m" c="masculine"/>
<sdef n="gender:f" c="feminine"/>
<sdef n="gender:0" c="feminine"/>
<sdef n="number:sg" c="singular"/>
<sdef n="number:pl" c="plural"/>
<sdef n="case:d" c="direct"/>
```

```
<sdef n=“case:o” c=“oblique”/>
</sdefs>
```

Step 2. Developing Paradigms

All the possible paradigms (<pardefs>) of a language are first defined which includes splitting of a Romanized word-form (string of characters) into the left-unchangeable string which is considered as computational root (not necessarily a linguistic root) and the right- variable string which is considered as computational inflection (not necessarily a morpheme). Both are separated by a splitter (slash). By choosing a word-form (frequent one) for defining a paradigm, we are actually naming/tagging that very paradigm by the name of the word-form chosen (e.g. kETO). The variable part of string on the right side of the splitter gets substituted while generating the inflectional paradigm (all possible word-forms) for the same word as shown below:

Noun Paradigms

```
<pardefs>
<pardef n=“kET/O_Nm”>
<e><p><l>O</l><r>O<s n=“cat:N”/><s n=“subcat:NC”/><s n=“Gender:m”/><s
n=“Number:sg”/><s n=“Case:dir”/><s n=“Case_marker1:0”/><s
n=“Case_marker2:0”/><s n=“Case_marker3:0”/><s n=“Particle1:0”/><s
n=“Particle2:0”/><s n=“Particle3:0”/><s n=“Quantitative:0”/><s
n=“Distributive:0”/></r></p></e>
</pardef>
<pardef n=“AmA/_Nf”>
<e><p><l></l><r><s n=“cat:N”/><s n=“subcat:NC”/><s n=“Gender:f”/><s
n=“Number:sg”/><s n=“Case:dir”/><s n=“Case_marker1:0”/><s
n=“Case_marker2:0”/><s n=“Case_marker3:0”/><s n=“Particle1:0”/><s
n=“Particle2:0”/><s n=“Particle3:0”/><s n=“Quantitative:0”/><s
n=“Distributive:0”/></r></p></e>
</pardef>
```

Pronoun Paradigms

```
<pardef n=“ma/_P”>
<e><p><l></l><r><s n=“cat:P”/><s n=“subcat:PPR”/><s n=“Gender:0”/><s
n=“Number:0”/><s n=“Case:dir”/><s n=“Case_marker1:0”/><s
n=“Case_marker2:0”/><s n=“Case_marker3:0”/><s n=“Particle1:0”/><s
n=“Particle2:0”/><s n=“Particle3:0”/><s n=“Quantitative:0”/><s n=“Distributive:0”/><s
n=“Person:1”/><s n=“Definiteness:y”/><s n=“Dimension:0”/></r></p></e>
</pardef>
<pardef n=“tam’/_P”>
<e><p><l></l><r><s n=“cat:P”/><s n=“subcat:PPR”/><s n=“Gender:0”/><s
n=“Number:sg”/><s n=“Case:dir”/><s n=“Case_marker1:0”/><s
n=“Case_marker2:0”/><s n=“Case_marker3:0”/><s n=“Particle1:0”/><s
n=“Particle2:0”/><s n=“Particle3:0”/><s n=“Quantitative:0”/><s n=“Distributive:0”/><s
n=“Person:2”/><s n=“Definiteness:y”/><s n=“Dimension:0”/></r></p></e>
</pardef>
```

Nominal Modifier Paradigms
<pardef n=“catura/_J”>
<e><p><l></l><r><s n=“cat:J”/><s n=“subcat:JJ”/><s n=“Gender:0”/><s
n=“Number:0”/><s n=“Case:0”/><s n=“Particle1:0”/><s n=“Distributive:0”/><s
n=“Degree:0”/></r></p></e>
</pardef>
<pardef n=“budhdU/_J”>
<e><p><l></l><r><s n=“cat:J”/><s n=“subcat:JJ”/><s n=“Gender:0”/><s
n=“Number:0”/><s n=“Case:0”/><s n=“Particle1:0”/><s n=“Distributive:0”/><s
n=“Degree:0”/></r></p></e>
</pardef>

Verb Paradigms
<pardef n=“bas/_V”>
<e><p><l></l><r><s n=“cat:V”/><s n=“subcat:VM”/><s n=“Gender:0”/><s
n=“Number:sg”/><s n=“Particle1:0”/><s n=“Person:2”/><s n=“Negation:0”/><s
n=“Tense:npst”/><s n=“Aspect:0”/><s n=“Mood:imp”/><s n=“Voice:act”/><s
n=“Finiteness:fin”/></r></p></e>
</pardef>
<pardef n=“rU/_V”>
<e><p><l></l><r><s n=“cat:V”/><s n=“subcat:VM”/><s n=“Gender:0”/><s
n=“Number:sg”/><s n=“Particle1:0”/><s n=“Person:2”/><s n=“Negation:0”/><s
n=“Tense:0”/><s n=“Aspect:0”/><s n=“Mood:imp”/><s n=“Voice:act”/><s
n=“Finiteness:fin”/></r></p></e>
</pardef>

Adverb Paradigms
<pardef n=“acAnaka/_A”>
<e><p><l></l><r><s n=“cat:A”/><s n=“subcat:AMN”/><s
n=“Particle1:0”/></r></p></e>
</pardef>
<pardef n=“acAklI/_A”>
<e><p><l></l><r><s n=“cat:A”/><s n=“subcat:AMN”/><s
n=“Particle1:0”/></r></p></e>
</pardef>

Participle Paradigms
<pardef n=“khAikana/_L”>
<e><p><l></l><r><s n=“cat:L”/><s n=“subcat:LPRF”/><s n=“Particle1:0”/><s
n=“Particle2:0”/><s n=“Particle3:0”/><s n=“Negation:0”/></r></p></e>
</pardef>
<pardef n=“khAEra/_L”>
<e><p><l></l><r><s n=“cat:L”/><s n=“subcat:LPRV”/><s n=“Particle1:0”/><s
n=“Particle2:0”/><s n=“Particle3:0”/><s n=“Negation:0”/></r></p></e>
</pardef>

Postposition Paradigms
<pardef n=“lAgi/_PP”>
<e><p><l></l><r><s n=“cat:PP”/><s n=“subcat:PP”/><s n=“Gender:0”/><s
n=“Number:0”/><s n=“Case_marker1:bnf”/><s n=“Case_marker2:0”/><s
n=“Case_marker3:0”/><s n=“Particle1:0”/><s n=“Particle2:0”/><s n=“Particle3:0”/><s
n=“Distributive:0”/></r></p></e>
</pardef>
<pardef n=“sAtha/_PP”>
<e><p><l></l><r><s n=“cat:PP”/><s n=“subcat:PP”/><s n=“Gender:0”/><s
n=“Number:0”/><s n=“Case_marker1:0”/><s n=“Case_marker2:0”/><s
n=“Case_marker3:0”/><s n=“Particle1:0”/><s n=“Particle2:0”/><s n=“Particle3:0”/><s
n=“Distributive:0”/></r></p></e>

```
</pardef>
```

Particle Paradigms

```
<pardef n="ani/_C">
<e><p><l></l><r><s n="cat:C"/><s n="subcat:CCD"/></r></p></e>
</pardef>
<pardef n="EvaM/_C">
<e><p><l></l><r><s n="cat:C"/><s n="subcat:CCD"/></r></p></e>
</pardef>
```

Step 3. Dictionary Building

Lemmatization is done manually and computationally feasible lemmas are included in the dictionary along with the paradigm name, they follow (already defined). Here, compromising of linguistic feasibility can't be ruled out. Sufficient number of dictionary entries is to be made in order to assure wider coverage of the morphological analyzer/generator. Computationally feasible lemmas with their respective paradigm tag are show below in the XML encoded dictionary.

```
<section id="main" type="standard">
<elm="kET"><i>KET</i><par n="kET/O_Nm"/></e>
<elm="acAn"><i>acAn</i><par n="kET/O_Nm"/></e>
<elm="acAnak"><i>acAnak</i><par n="acAnaka/_A"/></e>
<e lm="garn"><i>garn</i><par n="garna/_V"/></e>
<e lm="aDn"><i>aDn</i><par n="garna/_V"/></e>
<e lm="ghar"><i>ghar</i><par n="ghara/_Nm"/></e>
<e lm="AkAr"><i>AkAr</i><par n="ghara/_Nm"/></e>
<elm="akAs"><i>akAs</i><par n="ghara/_Nm"/></e></section>
</dictionary>
```

Step 4. Compiling & Processing

After defining each paradigm and finishing the associated dictionary building, we can run the system to check if the given word form is properly analyzed (i-e assigned the analysis of the proper paradigm). For running purpose, we first compile dictionary into a sort of compact & efficient representation by a default command called lt-comp.

Syntax:

lt-comp lr dictionary.dix outputFile.bin ←

Then, we process the compiled data for getting the analysis of the word form, using another default command called *lt-proc*.

Syntax:

lt-proc –c outputFile.bin ←

8. Issues

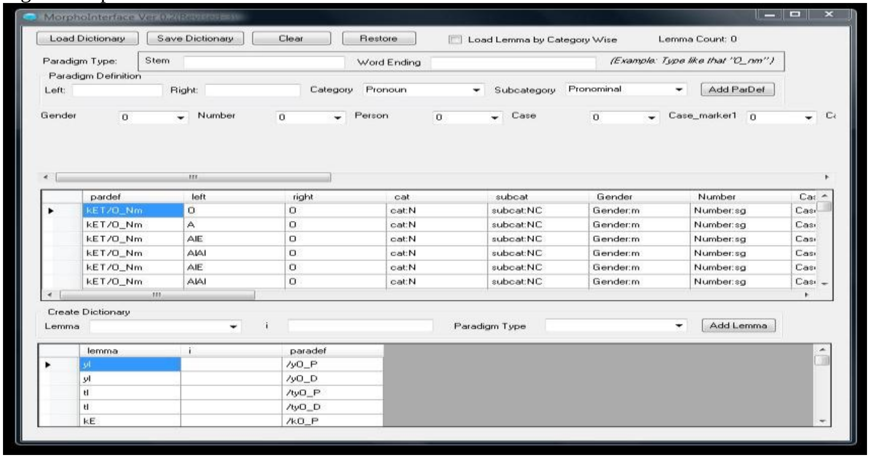
As such there are no issues for Nepali morph-analyzer like most of Indo-Aryan Languages but the behavior of genitive case marker is really problematic in Nepali. For illustration ko [gen.mas.sg] & kI [gen.fem.sg], markers are written together with the preceding nominals like "AmA-ko & AmA-ki" unlike Urdu-Hindi. Since, genitive markers or postpositions show object agreement in Indo-Aryan. It is problematic to determine the gender of "AmA-ko" as 'AmA' has (fem) gender but the inflection '-ko' has (mas) gender. It seems that splitting or segmentation is the only viable solution to avoid this gender paradox.

9. Interface

Since, the entire process of resource building (i.e. paradigm defining & dictionary building) is manual and labor intensive, involving direct manipulation of the XML code called Meta-Data. There are much chances of mishandling & thus, even a minute alteration in the XML schema renders the whole system non-functional. Moreover, the inputting is very prone to repetitions & redundancies. One can't remember the entire list of words that has been entered in the lexicon. So, to avoid such complexities, a more interactive interface has been developed to facilitate our work and reduce the time taken for defining a single paradigm. Besides, preventing the mishandling of Meta-data and repetition of lemma entries, the interface additionally reduced the burden of manual lemmatization as it lemmatizes automatically.

As shown in Figure 2, the window of the interface has the two parts; the upper paradigm defining part and the lower lexicon building part. We can now; either input the word directly into the text box or load it from the word list and get its lemma easily, just by assigning an appropriate paradigm from the drop down list.

Figure 2. Apertium interface



Lemma Count: 0

Paradigm Type: Stem | Word Ending | *(Example: Type like that "O_nm")*

Paradigm Definition

Left: | Right: | Category: Pronoun | Subcategory: Pronominal

Gender 0 | Number 0 | Person 0 | Case 0 | Case_marker1 0

pardef	left	right	cat	subcat	Gender	Number	Ca
kET/O_Nm	O	O	cat:N	subcat:NC	Gender:m	Number:sg	Cas
kET/O_Nm	A	O	cat:N	subcat:NC	Gender:m	Number:sg	Cas
kET/O_Nm	AIE	O	cat:N	subcat:NC	Gender:m	Number:sg	Cas
kET/O_Nm	AIAI	O	cat:N	subcat:NC	Gender:m	Number:sg	Cas
kET/O_Nm	AIE	O	cat:N	subcat:NC	Gender:m	Number:sg	Cas
kET/O_Nm	AIAI	O	cat:N	subcat:NC	Gender:m	Number:sg	Cas

Lemma | i | Paradigm Type

lemma	i	paradef
yl		/yO_P
yl		/yO_D
tl		/tyO_P
tl		/tyO_D
kE		/kO_P

10. Quantum and Coverage

Currently, the quantum of our work is very less, only 20,000 Ws. Since, the time consuming analysis part of the work is almost over. It is expected to show speedy growth in near future as our work will be mostly confined to dictionary building. So far, we have covered 9 categories and 29 Sub-categories in our analysis. The number of paradigms and lemma per category is shown in Table 1.

Table 1. Details of Nepali Morph-Analyzer

Categories	No. of Sub-Categories	No. of Paradigms	No. of Lemma (Ws)
Noun	04	045	-
Pronoun	05	016	-
Demonstrative	03	003	-
Nom-Modifier	03	031	-
Verb	02	023	-
Adverb	01	016	-
Postposition	01	008	-
Participles	01	006	-
Particle	09	071	-
Total	29	219	20,000

11. Conclusion

The above discussion reveals that developing a Finite-state morphological-analyzer for Nepali using LT-Toolbox is an efficient way given the elegance of the combinatory effect of word and paradigm model and the finite state transducers on speedy processing. Further, using an open source tool in creation of computational resource for resource poor Indian languages is need of hour, as it involves low cost and less time. Finally, for developing morphological analyzer with this approach one doesn’t require any programming skill but a little working knowledge of computers and morphology. So, every one irrespective of having programming background and strong linguistic skills can develop language specific resources and can contribute in the technological development of Indian language. In the current work we have finished the development of inflectional morphological analyzer for Nepali. Almost all inflectional paradigms have been captured and increasing the size of dictionary will continue until extracting and using unique words from 1 million text corpus of Nepali is over. Further, once inflectional morphology is done with sufficient coverage, derivational morphology will be taken care of in future to make a robust morph analyzer of Nepali.

References

Adhikari, H. R. (1993). *Samsamayik Nepali vyaakaran*. Kathmandu: Kunjal Prakashan.

Bharti, A., Chantanya, V. & Sanghal, R. (1995). *Natural language processing: A Paninian perspective*. New Delhi: Prentice Hall.

Baerman, M., Brown, D., & Corbett, G. G. (2005). *The syntax-morphology interface: A study of Syncretism*. Cambridge University Press.

Boye, G. (1999). Nepali verb morphophonology. In P. Yogendra, P. Yavada & W. Warren (eds.). *Topics in Nepalese linguistics* (pp. 118-169). Kathmandu: Royal Nepal Academy.

Forcada, M. L., Bonev, B. I., Rojas, S. O., Ortiz, J. A. P., Sanchez, G. R., Martinez, F. S., Armentano-Oller, C., Montava, M. A., & Tyers, F. (2010). Documentation of the open-source shallow-transfer machine translation platform Apertium. Retrieved on December 3, 2012 from http://xixona.dlsi.ua.es/~fran/apertium2-documentation.pdf

Hussain, S. (2004). *Finite-state Morphological analyzer for Urdu*. Unpublished MS thesis, Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, Pakistan.

Jurafsky, D. & Martin, J. (2005). *Speech and language processing: An introduction to natural language processing computational linguistics and speech recognition*. Boulder: University of Colorado Boulder.

LDC-IL Transliteration Chart For Indian Languages, Retrieved on December 3, 2012 from http://www.ldcil.org/download/Transliterationstandards.pdf

Stump, G. T. (2001). *Inflectional morphology. A theory of paradigm structure*. Cambridge: Cambridge University Press.

Uma Maheshwar Rao G. & Prameshwari, K. (2010). On the description of morphological data for morphological analyzers and generators: A case study of Telgu Tamil & Kannada. Knowledge Sharing Events. LDC-IL, CIIL, Mysore.

Vaidhya, Ashwini and Dipti Mishra.2009.Using Paradigms for Certain Morphological phenomenon in Marathi, 7th International Conference on NLP (ICON-2009. New Delhi: Macmilan.

Zwicky, A.M. (1985). How to describe inflection. *Linguistic Society* 11, 372-386.

About the Authors

Shahid Mushtaq Bhat is presently working as a Lecturer-cum-Resource Person in LDCIL, CIIL, Mysore, India. He is working in the area of NLP. He is also engaged as a part-time consultant in the Open Office Localization project, C-DAC, Pune. He is pursuing Ph.D. with the topic “Developing Dependency Treebank for Kashmiri” from the Department of Linguistics, Lucknow University, Lucknow.

E-mail: shahid.bhat3@gmail.com, ldc-shahid/ciil@ciil.stmpy.soft.net

Rupesh Rai is a Junior Research Assistant at the LDC-IL, Mysore, India.

E-mail: ldc-rupesh@ciil.stpmy.soft.net