Language Processing Applications Stemmer, Parts of Speech Tagger, Chunker, Grammar Analyzer

Bal Krishna Bal

Project Manager

PAN Localization Project

Madan Puraskar Pustakalaya, Nepal

URL : www.madanpuraskar.org

Email: bal@mpp.org.np

Contents

- Background
- Morphological Analyzer and Stemmer for Nepali
- Parts of Speech Tagger for Nepali
- Chunker for Nepali
- Grammar Analyzer for Nepali
- Grammar analyzer coverage and further work

Background

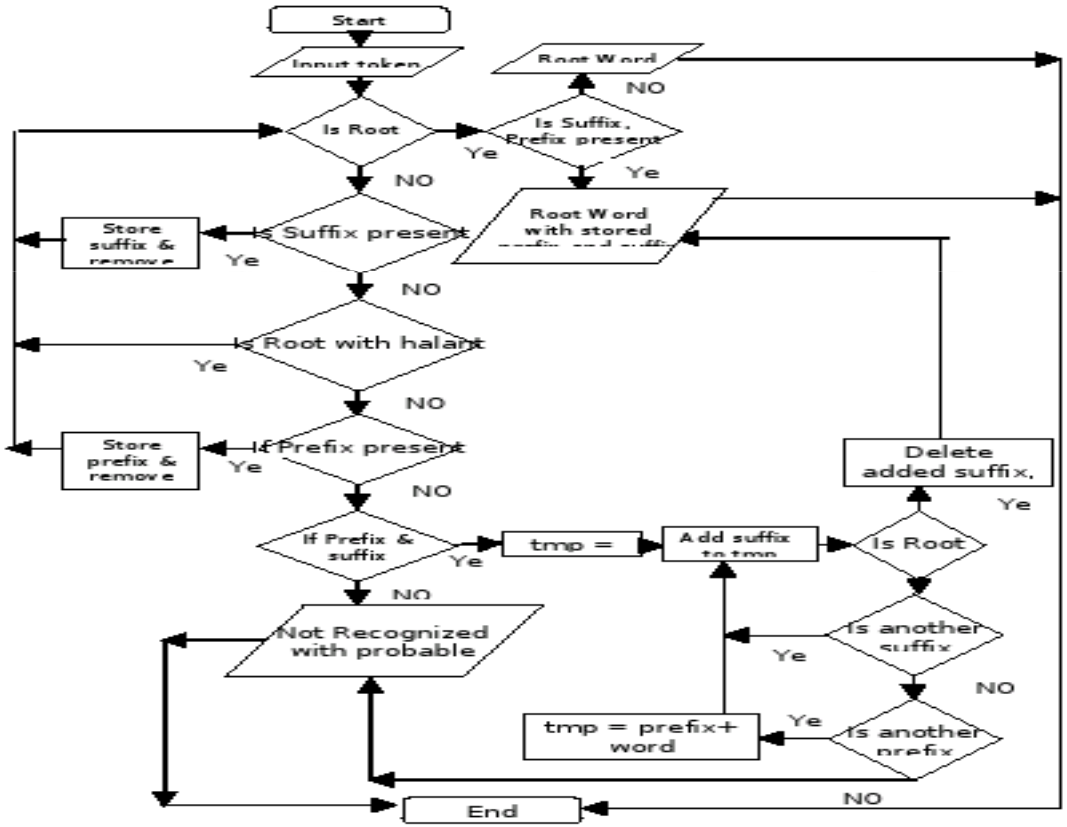
- Development of the Natural Language Processing (NLP) tools, very vital in developing Natural Language Processing Applications.
- In the year 2005, when NLP Research and Development (R&D) just started in Nepal, both resources and tools were at scarce.
- In this scenario, the idea of developing a basic prototype of a Nepali grammar checker/analyzer was conceived under the PAN Localization Project.
- This involved developing almost everything from scratch – for example, a machine readable and a multi-purpose Nepali lexicon, morphological analyzer, parts of speech tagger, chunker, parser etc.

Morphological Analyzer and Stemmer for Nepali

- A Morphological Analyzer (MA) is a program or algorithm which determines the morpheme(s) of a given inflected or derived word form including the analysis of the bound morphemes in it's grammatical form.
- A stemmer on the other hand returns the stem of a word which is a result of the stripping of one or more affixes from the word.
- We have developed a hybrid module which incorporates the functions of both the Morphological Analyzer and the Stemmer.
- The prerequisites of the module are:
  - POS tagset
  - Tokenizer
  - Free morpheme based lexicon
  - Two sets of affixes each for the suffix and the prefix
  - A database of word breaking grammatical rules

Morphological Analyzer and Stemmer …

Flowchart



Morphological Analyzer and Stemmer...

```
<s>
<token morph="मान्छे(NN)" pos="NN" type="w">मान्छे</token>
<token morph="संसार(NN)+मा(PPG)" pos="NN_PPG" type="w">संसारमा</token>
<token morph="पाउ(VF)+इ(i)+ने(NE)" pos="VF_i_NE" type="w">पाइने</token>
<token morph="प्राणी(NN)+हरू(HRU)+मा(PPG)" pos="NN_HRU_PPG"
type="w">प्राणीहरूमा</token>
<token morph="सब(ADJ)+भन्(VF)+दा(DA)" pos="ADJ_VF_DA"
type="w">सबभन्दा</token>
<token morph="चलाख(ADJ)" pos="ADJ" type="w">चलाख</token>
<token morph="र(CCON)" pos="CCON" type="w">र</token>
<token morph="बुद्धिमान(ADJ)" pos="ADJ" type="w">बुद्धिमान</token>
<token morph="प्राणी(NN)" pos="NN" type="w">प्राणी</token>
<token morph="हो(VAUX)" pos="VAUX" type="w">हो</token>
</s>
```

Output of the Morphological Analyzer and Stemmer

Parts of Speech Tagger for Nepali

• We have adopted the Trigrams 'n' Tags (TnT) for developing a Parts of Speech Tagger for Nepali. TnT is a statistical part-of-speech tagger trainable on different languages and virtually on any tagset.

• For Nepali, TnT is trained on a corpus of about 80,000 manually tagged words.

• Currently the accuracy of the trained TnT POS Tagger for Nepali is 56% for unknown words and 97% for known words.

राम/NNP ले/PLE हार/NN लाई/PLAI दियो/VBF

Output sample of the TnT trained Nepali POS Tagger

Chunker for Nepali

A chunk can be defined as a collection of contiguous words such that the words inside a chunk have dependency relations among them.

A chunker is a tool to identify such chunks in the given sentences.

The chunker for Nepali involves the following:

i) Set of linguistic rules for chunking Nepali phrases

ii) A chunking algorithm that makes use of the chunk rules

Chunker for Nepali...

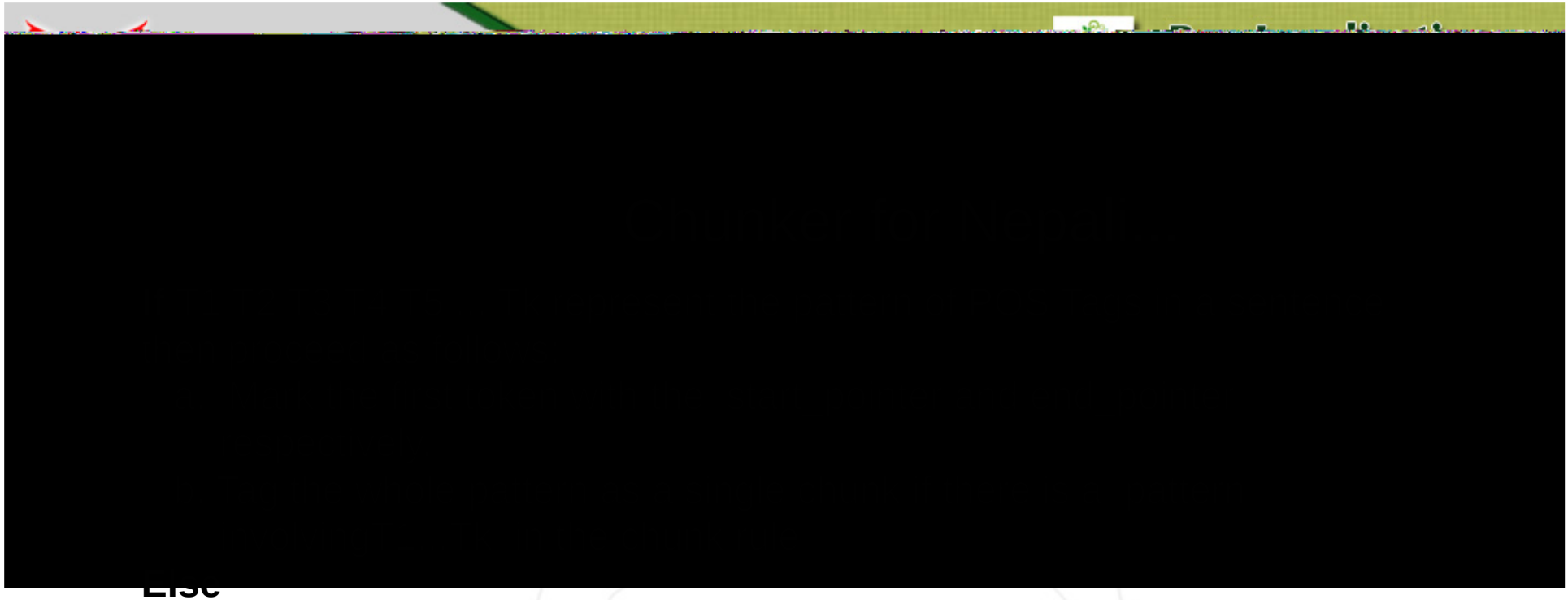
NCH: (DUM) (PLAI)˜

NCH: (NN)˜

NCH: (NNP)˜

VCH: (VBF) | (VOP) | (VKO) | ((VI) (VF)) | (VAUX)˜

Sample chunk rules for Nepali



Else

Decrease the end_pointer to one token left and continue until a pattern is detected between the start_point and end_point

As soon as a pattern is detected, increment the start_pointer by one and shift the pointer end_point to the right most position.

c. Continue a-c until the start_pointer reaches the last token i.e. Tk.

Chunker Algorithm

Chunker for Nepali...

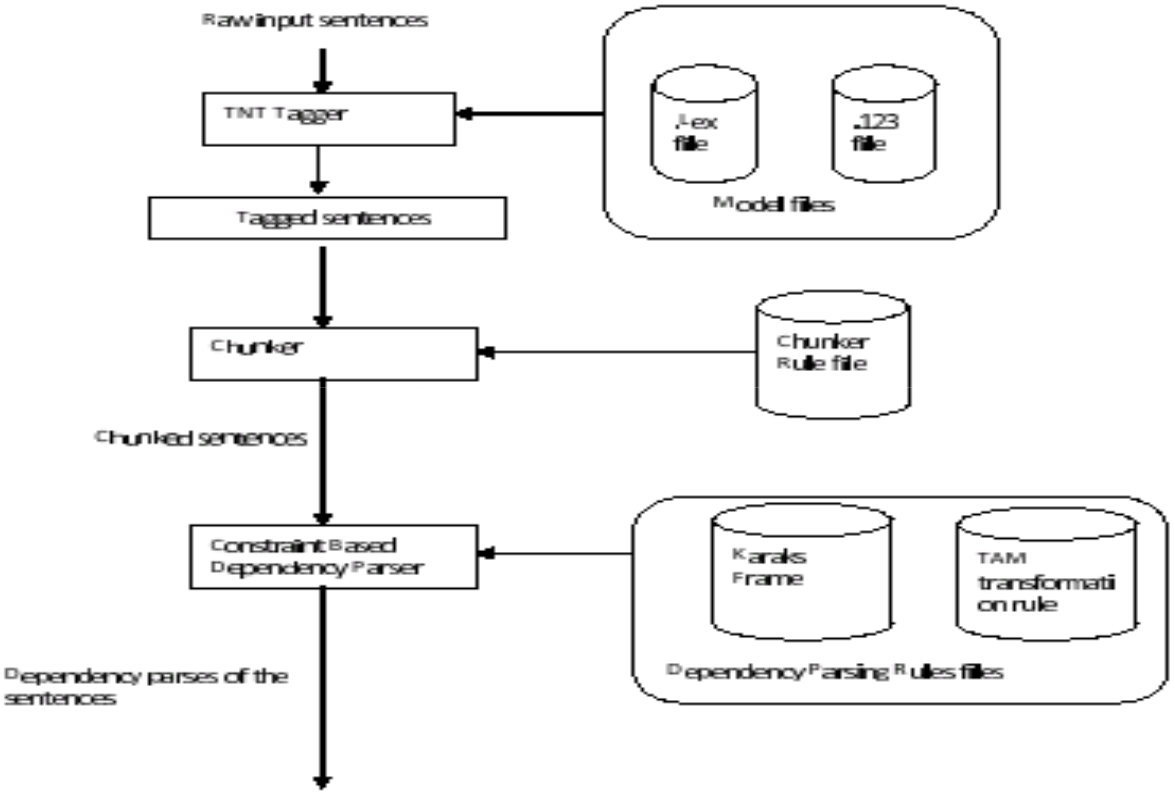
Currently, we have around 30 chunk rules, which has to be further optimized for better coverage and output.

The chunkset consists of 11 chunks at the moment.

{राम/NNP ले/PLE }/NCH {हरि/NN लाई/PLAI }/NCH {दियो/VBF }/VCH

Sample output of the chunker

Grammar Analyzer for Nepali



High Level System Architecture of the Grammar Analyzer for Nepali

Grammar Analyzer for Nepali

In addition to the modules discussed earlier, we would require a parser module for developing the Grammar Analyzer for Nepali.

The Nepali parser module follows the **dependency grammar formalism** which treats a sentence as a set of modifier-modified relations.

The dependency grammar formalism is believed to be best suited free word order languages like Nepali.

Generally, a verb is a primary modifier or the root of a sentence.

A **dependency parser** gives us a framework to identify these relations.

Relations between noun constituents and verb are called **karaka relations.**

Grammar Analyzer for Nepali...

The **Paninian grammar** defines six types of karaka relations as listed below:

Karta – agent/doer/force(k1)˜
Karma – object/patient(k2)˜
Karana – instrument (k3)˜
Sampradaan -beneficiary (k4)˜
Apadaan – sources (k5)˜
Adhikarana – location in place/ time/other (kx)˜

The **karaka frame** specifies what karakas are mandatory or optional for the verb and what vibhaktis (postpositions) they take respectively.
Each verb belongs to a specific verb class and each class has a basic karaka frame.
Each Tense, Aspect and Modality (TAM) of a verb specifies a transformation rule.
The **demand frame** for a verb indicates the demands that a verb makes. It depends on the verb and its TAM label.
Based on the TAM of a verb, a **transformation** is made on the verb frame taking reference of the TAM frame.

Grammar Analyzer for Nepali…

Arc label	Necessity	Vibhaktis	Lextype	Arc pos	Arc dir
K1	Mandatory	Null	‘N	L	O
K2	Mandatory	Null/को	N	L	O
K3	Optional	द्वारा	N	L	O
K4	Optional	लाई	N	L	O

Karaka frame for verb~“दिन्छ”~

Arc label	Necessity	Vibhaktis	Lextype	Arc pos	Arc dir
K1	Mandatory	**ले~(transformed)**	‘N	L	O
K2	Mandatory	Null/को	N	L	O
K3	Optional	द्वारा	N	L	O
K4	Optional	लाई	N	L	O

Transformed frame for “यो”

Grammar Analyzer for Nepali...

Steps of parsing:

1. Finding the verb candidate

The verb candidate(s) in the sentence should be identified. From the identified verb, its seed or root verb form and the Tense, Aspect and Modality (TAM) forms should be identified.

2. Identifying the verb frame and making the necessary transformation

On the basis of the seed verb, the respective verb and the TAM frames would be loaded.

3. Labeling of the arcs

Labeling of the arcs is performed on the basis of 1 and 2.

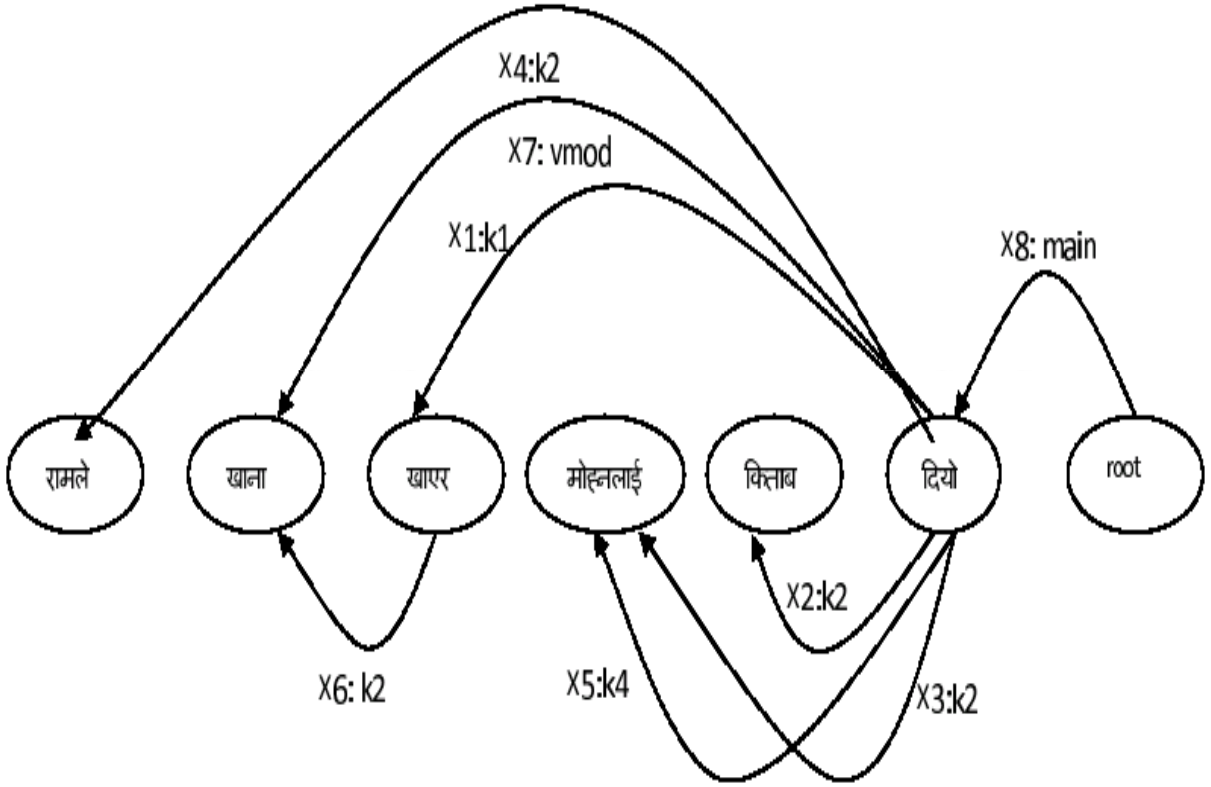
4. Imposing the constraints

By imposing constraints on the possibility of incoming and outgoing arcs from the word constituents,, we filter the inappropriate arcs.

Constraints:

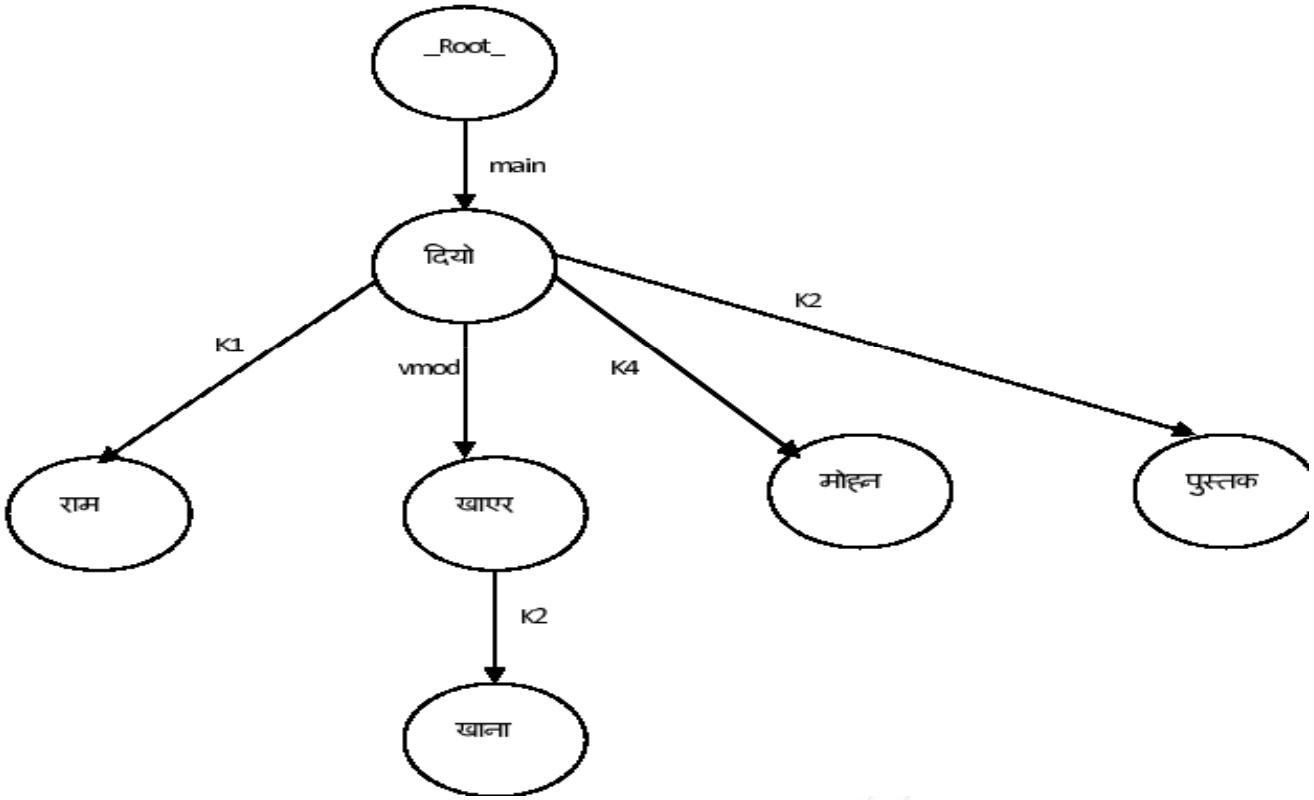
- For each of the mandatory demands in a demand frame for each demand group, there should exactly one outgoing edge labeled by the demand from the demand group.
- For each of the optional demands frame for each demand group, there should be at most one outgoing edge labeled by the demand from the demand group
- There should be exactly one incoming arc into each source group.

Grammar Analyzer for Nepali...



Graph formed for the transformed frame

Grammar Analyzer for Nepali...



Solution parse

Grammar Analyzer for Nepali…

For example,

If we have the Nepali sentence:- रामले हरिलाई दियो ।

Ram gave to Hari.

{दियो/VBF }/VCH -->k1<--{राम/NNP ले/PLE }/NCH
{दियो/VBF }/VCH -->k2<--{हरि/NN लाई/PLAI }/NCH

Interpreting the solution parse:

Between the first NCH and VCH, there is a valid K1 relation whereas between the second NCH and the VCH, we have a valid K2 relation.

Hence the sentence रामले हरिलाई दियो is a valid sentence.

Grammar analyzer coverage

- The Grammar Analyzer for Nepali currently parses and analyzes simple declarative sentences with just one verb candidate.
- We have developed the karaka frame for around 700 Nepali verbs.
- An agreement module if added to the analyzer could further filter the parses returned by the parser module, this time taking the feature agreements like gender, number, person, tense, aspect and modality.

Acknowledgment

This work was carried out with the aid of a grant from the Language Resource Association (GSK) of Japan and International Development Research Centre (IDRC), Ottawa, Canada, administered through the Centre for Research in Urdu Language Processing (CRULP), National University of Computer and Emerging Sciences (NUCES), Pakistan.

Thank You!!