PAN/L10n/PhaseII/Reports
Madan Puraskar Pustakalaya
01-04-2008

Nepali grammar checker

Bal Krishna Bal, Bineeta Pandey, Laxmi Khatiwada, Prajwal Rupakheti

Madan Puraskar Pustakalaya, Lalitpur, Nepal

bal@mpp.org.np ,bineeta@mpp.org.np, lkhatiwada@mpp.org.np, prajwal@mpp.org.np

Abstract

This report summarizes the research and development works currently in progress for developing a grammar checker for Nepali. The approaches and the methodologies adopted are also discussed in the report.

1. Introduction

Research works in the development of a grammar checker for Nepali has been going on for quite sometime. The previous reports on the development of a grammar checker for Nepali involve some survey of the available NLP tools and other computational linguistic resources for Nepali. As revealed by the reports, the status of the tools and resources for Nepali is very minimal. Some works on morphology has started with the Stemmer and the Morphological Analyzer being developed. Our works in the later stages have been focussed on developing the modules required for the grammar checker. These include the Parts-Of-Speech(POS) Tagger, Chunker, Parser and the Agreement Module. In the later sections of this document, we briefly highlight about these modules.

2. System architecture

A brief description of the modules involved in the system architecture of the grammar checker is given below:

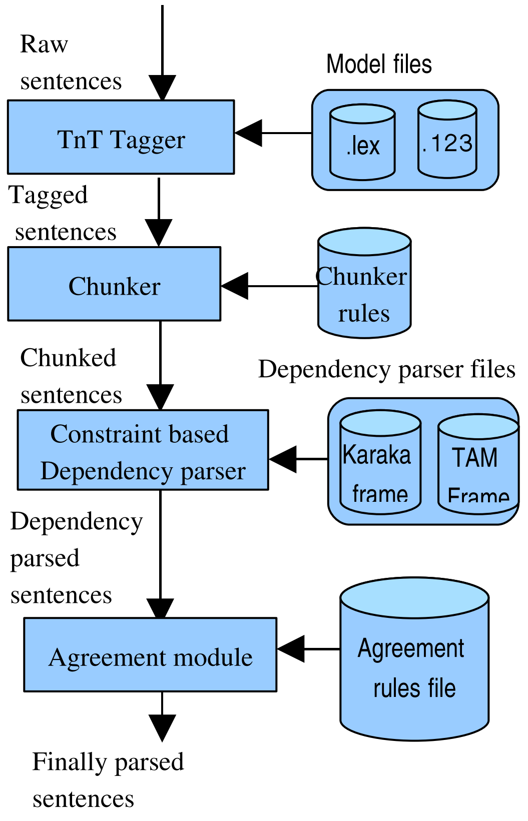


Fig. 1. System architecture

2.1. Parts-of-Speech(POS) Tagger

For the purpose of the POS Tagger for Nepali, we have adapted TnT[1], a free POS Tagger that follows a tri-gram statistical approach for POS Tagging. Besides the trigram approach, TnT has the smoothing technique incorporated into it. The smoothing technique basically implies the consideration of bi-gram, uni-gram as well as n-grams while tagging. The Nepali POS Tagset used by the tagger has 43 tags.

For training the POS Tagger, we have developed a training corpus of manually POS Tagged text of around 8,000 words. The corpus is preserved in XML format with the tag *<w>* for each word and its POS tag is marked by *attr* attribute in the attribute column. In order to use this corpus for training TNT, we have written a converter application in Java that can convert the corpus from the XML format to the required TNT format i.e. a single line containing a word and corresponding POS column.
The training model of TnT extracts useful information in the two model files, namely, .lex and .123 files, which basically store learnt patterns and other useful information from the training corpus.

1 TnT is a very efficient statistical part-of-speech tagger that is trainable on different languages and virtually any tagset.

2.2. Chunker

The chunker module takes the POS Tagged Text from the POS Tagger Module and identifies the chunks in the sentence. For identifying the chunks, the chunker module consults the chunk rule file. Currently we have about 13 chunk rules and 11 chunks identified for the Nepali chunkset. Linguistic rules for chunking is converted into a set of regular expressions and represented in the form of a context free grammar formalism. A sample of the rules in the chunk rule file is presented below:

```
NCH:(DET)?((NUM)(CL)?)?(INT)?(AD
J)*((NN)|(NP)|(PP))((PLE)|(PLAI))?
JOCH:(ADJ)*((NN)|(NP)|(PP))(PKO)
NCH:((VNE)|(VKO))((NN)|(NP)|
(POP))((PLE)|(PLAI))?
FVCH:(VOP)|(VKO)|((VI)(VF))|(VAUX)|
(VF)
NVCH:(VOP)|((VOP)(VKO))
GVCH:(VI)|(VNE)
AJCH:(INT)?(ADJ)
AVCH:(INT)*((ADV)+|(VOP)+)
PNCH:(YM)|(YF)|(YB)
CNCH:(CCON)
```

Fig. 2. Sample of chunk rules

While the non-terminal symbols on the right are the chunks, the terminal symbols to the left are the POS Tags taken from the Nepali POS Tagset. The nonterminals and the terminals are separated by the ":"

symbol.

For the implementation of the chunker module, we have devised a simple algorithm which we describe below:

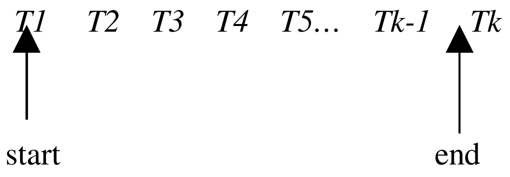
Let us say we have following *n* rules for defining *n* chunks:
Pattern 1-> chunk 1
Pattern 2- >chunk 2

Pattern n ->chunk n

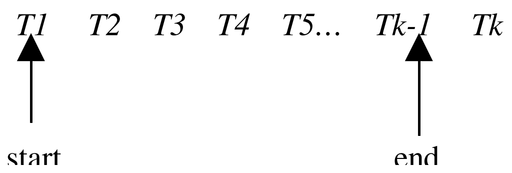
Now, if a given sentences has the following pattern of POS tags for its respective lexemes:

T1 T2 T3 T4 T5…Tk

Then, we proceed the chunking as shown below:



Initially, we mark the first token by the *start pointer* and the last token by the *end pointer*.
Now, if the pattern in between the *start* and *end* inclusive (i.e. *T1 T2 T3 T4 T5…Tk)* exists in the chunk rule file, than we will tag the whole pattern as a single chunk. If the rule does not exist, then we will decrease the *end pointer* to one token left and continue doing this until we locate the pattern between *start* and *end* in the chunker rule.



Similarly by following the same process, if the end *pointer* reaches the T3 token and we find the pattern between *start* and *end* (i.e. T1 T2 T3) in the chunker rule, we chunk the tokens between *start* and *end* as a single chunk and shift the *start pointer* to one token right of *end pointer* and *end pointer* to the rightmost end token as shown below.

T1 T2 T3 T4 T5… Tk-1 Tk

We will continue doing this until the *end pointer* reaches the last token i.e. *Tk*.

Following is the pseudo code for the chunking algorithm described above:

```
Flag=false;
Start = points to first token in the list;
End = points to the last token in the list;
While(start <=end ){
    While(flag !=true){
        pat = pattern between Start and End inclusive;
        For all patterns present in the rule{
            If( pat is found there in the rule){
                Flag=true;
                Start=end+1;
                End=points to the last token in the list;
                Mark the pattern between Start and End as a single chunk;
            }
            else{
                End=End-1;
            }
        }
    }
}
```

2.3. Parser Module

The parser Module is still in the research phase. We plan to develop a dependency based parser that takes into account the modifier and modified relationships, karakas and case markers, which are typical of the Nepali language and grammar.

2.4. Agreement Module

The agreement module is also still in the research phase. This module would check for subject verb and object verb agreements prevalent in Nepali taking into account the different agreements applicable for Gender, Number, Person, Tense,Aspect and Mood.

3. Results

The four modules discussed above would be integrated thus forming the larger system, i.e. the grammar checker for Nepali. So far the two modules , viz., the POS Tagger and the Chunker have been developed and in the process of being tested and refined for performance. While the other two modules are still in the research phase, the first results of both the POS Tagger and the Chunker have been quite promising. Currently, the POS Tagger performs with an accuracy of 80%. It is expected that the system would perform with a better result once the training corpus size gets bigger and represents text from wide domains. As far as the chunker module is concerned, it's performance measure largely depends upon the availability of rules. So far, random test of sample sentences show a 100% coverage of the chunk rules. However, testing is continuously being conducted to see if there are instances of non-coverage by the available rules.

4. Conclusion

The research and development of the Nepali grammar checker has followed a modular approach. In this regard, we have been consequently testing and refining the already developed modules thus trying to enhance the output of these modules that would serve as input to the modules to be developed. Many aspects related to the grammar checker being new to the team, they are evolving with time. Hence, the proposed system architecture might be subject to some minor changes in future.

Acknowledgement

"This work was carried out with the aid of a grant from the International Development Research Centre, Ottawa, Canada administered through the centre for Research in Urdu Language Processing (CRLUP),National University of Computing and Emerging Sciences, Lahore, Pakistan (NUCES)"

5. References

[1] Daniel Jurafsky and James H. Martin, University of Colorado, Boulder. Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Fifth Indian Reprint, 2005, Pearson Education (Singapore).

[2] Architectural and system design of the Nepali grammar checker. Bal Krishna Bal et. al., PAN Localization, Working Papers 2004-2007, pp. 397-400.

[3] TnT, a statistical part-of-speech tagger. Thorsten Brants, Saarland University, Computational Linguistics, Saarbrucken, Germany.

[4] Natural Language Processing, a Paninian Perspective. Akshar Bharati et.al.,Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, India.