Bengali Parsing System at ICON NLP Tool Contest 2010

Aniruddha Ghosh† Amitava Das* Pinaki Bhaskar† Sivaji Bandyopadhyay¥
Department of Computer Science and Engineering
Jadavpur University
Jadavpur, Kolkata 700032, India
arghyaonline@gmail.com† amitava.santu@gmail.com* pinaki.bhaskar@gmail.com† sivaji_cse_ju@yahoo.com¥

Abstract

This paper reports about the development of a Bengali Dependency Parser as a part in the ICON 2010 NLP TOOLS CONTEST: IL Dependency Parsing. Due to syntactic richness of Bengali language, hybrid architecture has been proposed for the problem domain. A statistical data driven parsing system (maltparser) has been used followed by a rule-based post-processing technique. The system has been trained on the ICON 2010 NLP TOOLS CONTEST: IL Dependency Parsing at ICON 2010 datasets. The system demonstrated an accuracy of 83.87% 64.31% 69.3% respectively over fine-grained tagset.

1 Introduction

Bengali language is characterized by a rich system of inflections (VIBHAKTI), derivation, and compound formation (Saha et al., 2004; Chakroborty, 2003) and karakas, which is why the Natural Language Engineering (NLE) for Bengali is a very challenging task. Naturally these language specific peculiarities involve parsing natural language sentences considerable problems. Therefore, developing a computational grammar for a natural language can be a complicated endeavor. Grammar development, also known as grammar engineering, is a formidable challenging task.

Previous research proposed two different approaches context of parsing of a sentence. These techniques are known as grammar driven parsing and data driven parsing. Most of the previous grammar driven research attempts was for detection and formation of the proper rule set to identify characteristics of relations among inter-chunk relations.

Natural languages are very diverse in nature, no matter how many sentences a person has heard and taking in consideration during the rule making, new ones can always be produced. Even when people speak natural language incorrectly i.e., not strictly in accordance with rules of grammar and syntax, anyone can still make sense out of it. Hence developing a parsing rule book always will remain inadequate. Most of the modern grammar-driven dependency parsers (Karlsson et al.; 1995, Bharati et al., 2008) parse by eliminating the parses which do not satisfy the given set of constraints. They require rules to be developed for each layer. It is the reason to start with a data driven parsing system (Maltparser1 ver.1.3.1) as a baseline. The data driven parser requires a large set of manually annotated corpus. But the available dataset is not large in size and thus only data driven cannot outperform in this experimental setup. We propose a hybrid technique and the output of the baseline system then filtered by a rule-based post-processing system.

2 Dataset

The ICON 2010 NLP TOOLS CONTEST: IL Dependency Parsing at ICON 2010 datasets was provided with fine-grain and coarse-grain tagset. The corpus statistics is reported in the Table 1. A few detailed statistics about the distribution of sentence type is reported in table 2.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>s#</th>
<th>#</th>
<th>t#/s#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>960</td>
<td>7269</td>
<td>7.57</td>
</tr>
<tr>
<td>Development</td>
<td>150</td>
<td>812</td>
<td>5.41</td>
</tr>
<tr>
<td>Testing</td>
<td>150</td>
<td>962</td>
<td>6.4</td>
</tr>
</tbody>
</table>

Table 1: Corpus statistics; s# = number of sentence; t# = number of tokens; t#/s# = number of tokens per sentence.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>simple</th>
<th>compound</th>
<th>complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>223</td>
<td>188</td>
<td>589</td>
</tr>
<tr>
<td>Development</td>
<td>31</td>
<td>11</td>
<td>108</td>
</tr>
<tr>
<td>Testing</td>
<td>26</td>
<td>7</td>
<td>117</td>
</tr>
</tbody>
</table>

Table 2: Corpus sentence statistics

1 http://maltparser.org/download.html
3 The Dependency parsing system

3.1 The Maltparser

Malt is a classifier based Shift/Reduce parsing methodology. It uses arc-eager, arc-standard, covington projective and covington non-projective algorithms for parsing (Nivre, 2006). History-based feature models are used for predicting the next parser action (Black et al., 1992). Support vector machines are used for mapping histories to parser actions (Kudo and Matsumoto, 2002). It uses graph transformation to handle non-projective trees (Nivre and Nilsson, 2005).

Maltparser accepts CoNLL format as the input. The FEATS column in the training and developing corpus set, there are 10 attribute fields in a node. Among the attributes, six morphological features namely lexicon, category, gender, number, person, vibhakti or Tense-Aspect-modality (TAM) markers of the node are considered in the experiment. After experimentation with set of different combination of these features, the vibhakti, TAM and the morphological category produce better results as these morphological informations contains most crucial information to identify dependency relations for Indian languages. The Bengali datasets consists of 7% non-projective sentences. Among the four parsing algorithm provided with maltparser, we found that nivreeager works best for the Bengali corpus with fine-grained tagset. Analyzing output of parsing with default setting, we found the complex and compound sentences are the most error prone. We tried to separate compound sentences and compound sentences into simple sentences. We calculated the average number of tokens per sentence is around 6. Thus the max sentence length was set to 6. Malt parser provides root handling which specifies how dependents of the special root node are handled. After experimentation with root handling, the relaxed root handling yields better results. As the learning method maltparser uses SVM with polynomial kernel to map feature vector representation of a parser configuration. While tuning the learning method parameter, we changed the cost parameter from default value 1 to .65, which controls the tradeoff between minimizing training error and maximizing margin. From corpus statistics we revealed that the average number of instances i.e. tokens per sentence is near about 6 and the number of attribute per node i.e. chunk is 9. Thus, we also experimented with the liblinear classifier. Due to small set of dataset, many of the dependency relations are sparsely distributed, which leads to low LAS value. The comparative study of accuracies of different maltparser configurations are shown below.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>UAS²</th>
<th>LAS³</th>
<th>LS⁴</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nivreager+Liblinear</td>
<td>81.64%</td>
<td>54.58%</td>
<td>50.62%</td>
</tr>
<tr>
<td>Convington non-projective+LIBSVM</td>
<td>78.22%</td>
<td>51.02%</td>
<td>48.43%</td>
</tr>
<tr>
<td>Convington non-projective+Liblinear</td>
<td>79.33%</td>
<td>52.47%</td>
<td>50.52%</td>
</tr>
</tbody>
</table>

Table 3: comparison of maltparser output with different settings

During the development process confusion matrix helped to detect errors. Most prominent errors on the development set are shown in Table 4.

<table>
<thead>
<tr>
<th></th>
<th>k1</th>
<th>k2</th>
<th>k7p</th>
<th>k7t</th>
<th>pof</th>
<th>vmod</th>
</tr>
</thead>
<tbody>
<tr>
<td>k1</td>
<td>0</td>
<td>29</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>k2</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>k7p</td>
<td>12</td>
<td>6</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>k7t</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>pof</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>vmod</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4: Confusion Matrix on Development Set1

3.2 Post-Processing

With the help of confusion matrix, depending on the nature of errors we have devised a set of parsing rules. A few rules have been devised to produce more dependency relation variations.

Vibhakti plays a crucial role in dependency relation identification. In corpus, as the vibhakti information is missing in some cases, a suffix analyzer is applied to the word.

r6: According to dependency tagset, r6 denotes genitive relation. It takes ‘ra’, ‘era’, ‘xera’ genitive markers. For the marker ‘ra’, it can appear at the end of many words, e.g. ‘AbiskAra’. We have used a dictionary based approach to exclude these words. When chunks with these genitive markers have any indirect relation with the main verb, then it is marked with r6 relation.

² UAS – Unlabeled Attachment Score
³ LAS – Labeled Attachment Score
⁴ LS - Labeled Score
and following NP chunks is marked the related chunks.

**k7t:** Chunks with suffix ‘kAle’ denotes a time temporal. We have also, from the training corpus, developed list of time temporal. After a manual check of the generated list, we used the list to extract time temporal and marked with k7t.

**k7p:** Chunks with suffix ‘pur’ denotes a space temporal. We have also, from the training corpus, developed list of space temporal. After a manual check of the generated list, we used the list to identify space temporal and marked with k7p.

After an in depth study of the errors made by maltparser, a rule based system has been developed with the help of linguistic knowledge. Depending of specific attributes of the chunk like vibhakti/case markers and/or word information, the rule based system derives the dependency relation of the chunk. For each dependency relation tag depending on specific linguistic features, syntactic cues are derived identify the dependency relations. As example:

1. A NP chunk with 0 vibhakti and NNP or PRP postag will be marked with k1 relation with the nearest verb chunk.
2. A chunk with “era” vibhakti will be marked with ‘r6’ relation with next noun chunk.
3. A NP chunk with 0 vibhakti and NN postag will be marked with k2 nearest verb chunk.
4. In co-ordinate type sentences, the verb chunk will be marked with ‘ccof’ relation with the nearest CCP chunk. If CCP chunk is surrounded by two NP chunk then NP chunk will be marked ‘ccof’ with the CCP.
5. Sub-ordinate sentences are identified using the presence of keywords like ‘ye’ etc. In a sub-ordinate type sentences, the verb chunk of sub-ordinate clause will be marked with “nmmod__relc” with that chunk of main clause, which the clause is modifying.
6. If a chunk with “0_weke” vibhakti, k5 relation will be indentified. Both of these relation is pre-dependent i.e. a dependent that precedes its head.
7. If a chunk has “0_prawi” vibhakti, dependency relation will be ‘rd’.
8. After carefully analyzing training corpus, we found certain vibhakti with semantic meanings like ‘0_pakRe’, ‘0_hisAbe’ etc. can be treated as cue to mark vmod relation.
9. Verb like ‘kar’ or ‘ha’ often takes another argument to form compound verbs. The argument is marked with part-of relation (pof). The preceding noun or verb chunk, if it has no suffix, is marked with ‘pof’ relation.
10. If a chunk marked with “ke”, ‘k2’ relation will be indentified.
11. Noun chunks with rootwords like “amii” with “NN” postag or “wumi” with “PRP” postag will be marked with ‘k1’.
12. If the rootword is ‘ye’ and the word is ‘ya’, ‘wa’, ‘ye’, then the chunk will be marked with ‘k2’ relation.

The ambiguity comes when for a certain vibhakti, multiple possible relations are identified. As example, when a chunk with “0” vibhakti two possible dependency output relation is ‘k1’ & ‘k2’. Then this ambiguity is resolved using postag. If postag is ‘NNP’ then dependency relation will be ‘k1’ and if ‘NN’ then it will be ‘k2’. If ambiguity is not resolved with this rule then position of the chunk in the sentence is considered. If there are two chunks with “0” vibhakti, the distant chunk from the verb chunk will be marked with ‘k1’ relation and nearer ones will be marked with ‘k2’ relation.

After a study of co-occurrence of ‘k1’ & ‘k2’, we found that single occurrence of noun chunk with ‘0’ vibhakti is marked with ‘k1’ relation.

The output of the maltparser and the output of the rule based system are compared. The rule based system is given the higher priority as it is based on syntacto-semantic cues. If there is any mismatch between the two results then if rule based system derive an output then output of rule based system has been taken.

### 4 Error Analysis

During the development stage of the system we have gone through the error prone areas of the system. The system works best with simple sentences.

In the example sentence in Figure 1, CCP chunk is identified as root and it split the sentence and treats ((rAwa jege AsA)) and ((xedIte snAna-KAoyZAra animeRera Alasemi boXa hachila)) as two individual sentences. But semantically it is not correct. Here both the preceding and succeeding VGNF chunk of CCP should be connected with ccof relation. Among the dependency relation tags of the VGNF chunks, one of them will be attached with the CCP chunk. System faces ambiguity to choose proper dependency relation for CCP among the ‘rh’ and ‘vmod’
Figure 1: Example Parsing

4.1 Experimental Results

We have trained maltparser with the training set data with fine-grained tagset only. A brief statistics of the datasets are presented in Table 1 & 2. The maltparser with nivreeager parsing algorithm, yielded unlabelled attachment score of 81.64%, Labeled attachment score of 54.58% and Labeled score of 50.62% on the development set. After the suffix analyzer, we achieved 3% improvement of UAS and 9% improvement of LAS. The LS score yields 69%. Depending upon the nature of errors involved in the results, we have devised a set of rules for different dependency tags. The rule based system yields UAS, LAS, LS 84.02%, 66.63% & 70.82% respectively on the development set.

In the final evaluation, the system demonstrated UAS (Unlabelled Accuracy Score) is 83.87%, LAS (Labeled Accuracy Score) is 64.31% and LS (Labeled Score) is 69.3% respectively.

5 Conclusion

This paper reports on our works as part of the NLP Tool Contest at ICON 2010. We have used the data driven maltparser along with rule based post-processing. Using maltpaser, we have obtained UAS 80%, LAS 54% and LS 59% with the nivreeager algorithm with relaxed root handling. We have used suffix analyzer and a rule-based post-processing and produce a better accuracy value. The rule based system is given the higher priority as it is based on syntacto-semantic cues and using bootstrapping method, performance of data driven parser can be accuracy can be further enhanced.

A properly designed NLP platform for Indian languages must come with an efficient morpho-syntactic unit for parsing words into their constituent morphemes where lexical projections of words can be obtained from projections of individual morphemes. Phrasal order could be vary depending on the corpus. In future task our aim is to develop a more proficient statistical system, can produce more than one possible parsed output. A more concise rule set should be generated with morpho-syntactic and lexico-syntactic variations.
References


