A high recall error identification tool for Hindi Treebank Validation

Bharat Ram Ambati, Mridul Gupta, Samar Husain and Dipti Misra Sharma
Language Technologies Research Centre, International Institute of Information Technology
Hyderabad, INDIA – 500032
E-mail: ambati@research.iiit.ac.in, mridulgupta@students.iiit.ac.in, {samar, dipti}@mail.iiit.ac.in

Abstract

This paper describes the development of a hybrid tool for a semi-automated process for validation of treebank annotation at various levels. The tool is developed for error detection at the part-of-speech, chunk and dependency levels of a Hindi treebank, currently under development. The tool aims to identify as many errors as possible at these levels to achieve consistency in the task of annotation. Consistency in treebank annotation is a must for making data as error-free as possible and for providing quality assurance. The tool is aimed at ensuring consistency and to make manual validation cost effective. We discuss a rule based and a hybrid approach (statistical methods combined with rule-based methods) by which a high-recall system can be developed and used to identify errors in the treebank. We report some results of using the tool on a sample of data extracted from the Hindi treebank. We also argue how the tool can prove useful in improving the annotation guidelines which would in turn, better the quality of annotation in subsequent iterations.

1. Introduction

For effective processing of text, tools at different conceptual levels, say from letter/syllable level to discourse level are needed. Output of these tools can then be used in different NLP applications beginning with spell checkers to machine translation. These tools could be completely rule-based, completely statistical or a combination of both, i.e., hybrid systems. In quite a few cases, manually annotated gold standard data is required to build such tools. The annotated data, as one would expect, should be error free. Hence, the importance of validation of data and error correction cannot be undermined. On the other hand, annotation in itself is a time-consuming task. Thus, it is only desirable that the task of validation of data is fast without compromising quality. But doing validation of data completely manually would again be time consuming, as the validators have to look at each word in the annotated corpus. To make the task of validation easy and cost effective, we need tools that can supplement validators' task with a view of making the overall task fast without compromising reliability. With the help of such tools, validator can directly go to error instances and correct them. Therefore we need the tool to have high recall. It is easy to see that a human validator can directly reject un-intuitive errors (false positives) without much effort; one can therefore compromise on precision.

The proposed tool has been used for validating the dependency representation of a multi-layered and multi-representational treebank for Hindi (Bhatt et al., 2009).

The tool identifies errors in the Hindi annotated data at POS, chunk and dependency levels. Additionally, the identification of errors can help resolve ambiguous cases and thus improve the guidelines for annotation. Improved guidelines will directly make the task of annotation more consistent.

The paper is divided as follows. The first section is about introducing the work. Section 2 gives a brief overview of the Hindi dependency treebank. A survey of some of the previous efforts on automated validation is done in section 3. In section 4, we describe our approach in detail with examples. Results are reported in section 5. General discussion and directions for future work follow in section 6. We conclude our paper in section 7.

2. Hindi Dependency Treebank

A multi-layered and multi-representational treebank for Hindi (Bhatt et al., 2009; Xia et al., 2009) is being developed. The treebank will have dependency, verb-argument (PropBank, Palmer et al., 2005) and phrase structure (PS) representation. Automatic conversion from dependency structure (DS) to phrase structure (PS) is being worked out. Hence, it is important to have a high quality version of the dependency treebank to ensure efficient conversion from DS to PS representation. The focus of the current paper is to describe the methodology employed to detect errors in the DS representation. The dependency treebank contains information encoded at the morpho-syntactic (morphological, part-of-speech and chunk information) and syntactico-semantic (dependency) levels. Each sentence is represented in SSF format (Bharati et al., 2007). POS and chunk information is encoded following a set of guidelines (Bharati et al., 2006). The guidelines for the dependency framework (Bharati et al., 2009) have been adapted from computational Paninian grammar (CPG) (Bharati et al., 1995). For Indian languages, like Hindi, Paninian dependency scheme has been shown to be effective by Begum et al. (2008).

3. Related Work

Validation and correction tools are an important part for making treebanks error-free and consistent. Significant efforts have been made in this direction to develop such tools. One such approach for treebank error detection was employed by Dickinson and Meurers (2003; 2005) where they find out 'variations' in syntactic annotation. They use certain statistical patterns (*n*-grams) derived from large annotated corpora such as the Penn treebank (Marcus et al., 1993) to detect anomalies in treebanks. Their work includes anomaly detection in continuous and

discontinuous structural annotation. Adapting from a generalized approach on discontinuous structural annotation, this work was extended to detect errors at the dependency level in treebanks (Boyd et al., 2008). Some other earlier noteworthy methods employed for error detection in syntactic annotation (mainly POS and chunk), are by Eskin (2000) and van Halteren (2000). Other examples of detection of annotation errors in treebanks include (Kaljurand, 2004; Kordoni, 2003).

4. Approach

Our aim is to identify errors in POS, chunk and dependency annotated data. To identify the errors at each level of annotation we use both rules and statistics. We take 40k words manually annotated and validated data as development data. We used this development data to frame rules as well as to take decisions based on statistics. We followed a two-fold approach. The first part of approach involves detection of errors purely by rule-based methods. In the second part of the approach we use frequency-based measure to determine the possible errors and then prune out the false positives to improve precision by using some rules.

4.1 Rule-Based approach

In this approach, we use generic rules to identify the errors. Particular tags (POS/chunk/dependency) demand some particular patterns and vice-versa. This is the main idea in framing the generic rules. For example, if the **POS-tag is "SYM**[1]**"** then the *lexical item should not contain any character in the unicode range of Hindi or digits.* Similarly, if the **lexical item is a digit,** then the *POS tag should be QC* (POS tag for cardinals). Similar rules can be framed at chunk and dependency levels also. We used the annotation guidelines (Bharati et al., 2006, 2009) as an initial step to frame the rules. The guidelines, apart from providing description of the tags, give many pointers for annotators, in the form of linguistic cues to identify the tags, exceptional cases, common confusing and error-prone cases. More rules were later formulated using the development data. Further, we extracted mismatches in the annotated and validated sets of the development data. These mismatches are basically errors made by annotators which were corrected by validators. Analysis of these mismatches helped in framing additional rules. The nature of the rules varies for different type of annotation, as the context required is different for different types of annotation. For example, POS tagging rules are based on current lexical item, POS tags of previous words etc., whereas in case of dependency tags, rules are framed on features of current node, its parent, siblings, children and sometimes even a complete tree/sub-tree.

Figure 1, shows a sample output of the tool identifying the POS tag errors. In the example sentence depicted in the figure, "Ram gave three books to Sita", the rule that, a number should have its POS tag either a 'QC[2]', or a 'QF[3]' (refer, Bharati et al., 2006) comes in handy while detecting the error. Therefore, the word "3" which had been erroneously tagged as a demonstrative (DEM) in the sentence, is identified as an error which can be then promptly corrected by the human validator.

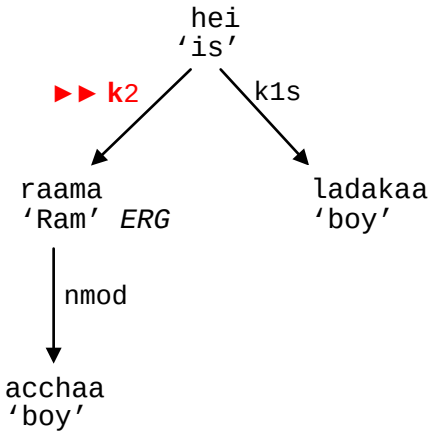
Identification of Errors using Rules

1	raama	*'Ram'*	NNP
2	ne	*'ERG'*	PSP
3	sitaa	*'Sita'*	NNP
4	ko	*'DAT'*	PSP
5	***3***	***'3'***	***DEM*** ◄◄
6	kitaabein	*'books'*	NN
7	diiM	*'gave'*	VM

"Ram gave three books to Sita."

Figure 1: Error detection by rule-based approach at POS level. The erroneous case is shown by the pointer '◄◄' in the sentence above.

Error detection at the dependency level is illustrated with the help of example sentence in Figure 2 below. The sentence is "Ram is a good boy."



"Ram is a good boy"

Figure 2: Error detection by rule-based approach at the dependency level. The erroneous case is shown by the pointer '◄◄' in the tree.

There are some dependency labels that are dependent on the presence of particular labels in a sentence. Following from this rule in the sentence above, a 'k1s' should be marked only when a 'k1' is present in the same sentential clause. Hence, an error is detected in the dependency tree. The number of actual errors detected using such rules is high on precision but low on recall value. In order to detect a wider coverage of errors we need to employ other techniques. These measures are described in the following subsection.

[1] SYM: POS tag for a punctuation marker, see Bharati et al., 2006 for complete details.

[2] QC: POS tag for words denoting a cardinal number

[3] QF: POS tag for words denoting quantifiers

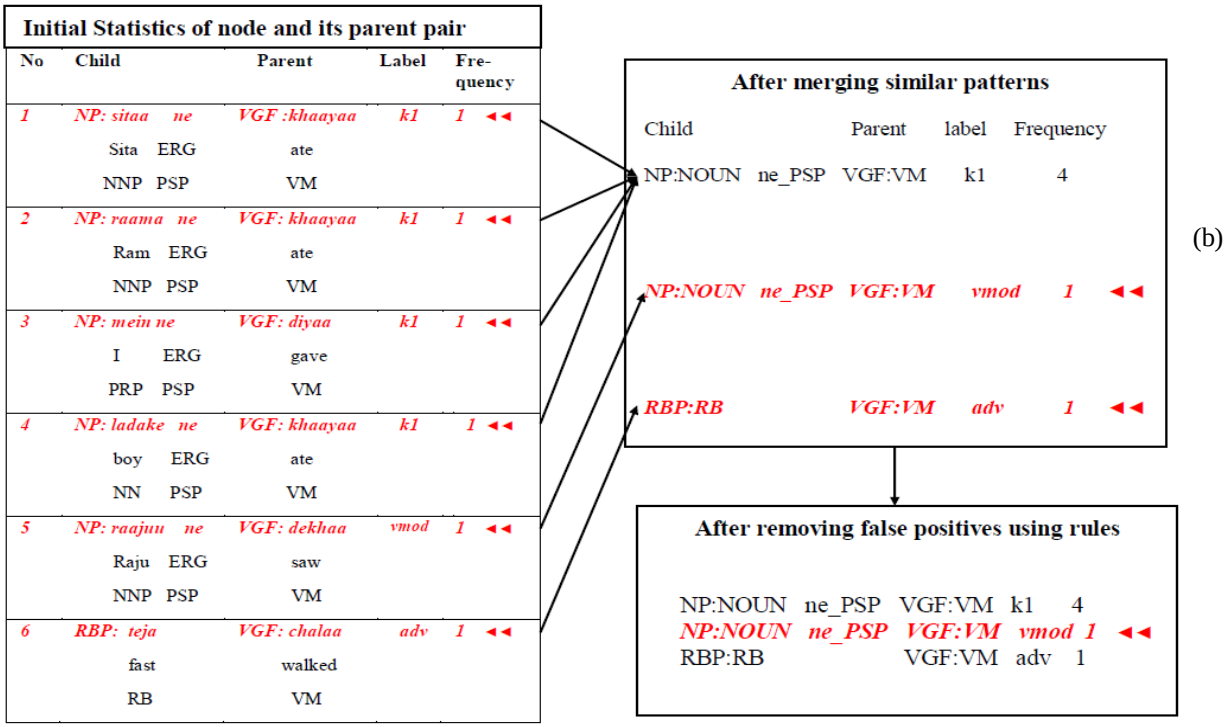


Figure 3: Error detection in inter-chunk dependencies by hybrid approach.

4.2 Hybrid Approach

Hybrid approach comprises of two modules. (1) the statistical module and, (2) Rule-based post-processing module. Statistical module tries to identify as many errors as possible. The goal of the statistical module is to achieve a high recall. Following this, we run a rule-based post-processing module on the output of the statistical module. The aim of this module is to increase precision of the system. With this approach we intend to detect the errors with high recall and reasonable precision.

4.2.1. Statistical Module

Statistically, low frequency is a sign of possible error. We calculate the frequencies of pattern and tag pairs, where tag can be either of POS or chunk or dependency. These patterns are annotation specific. For POS, word level patterns are considered. For chunks, lexical items and the POS tags of the sequence of words within the chunk are considered. For inter-chunk dependencies, chunk tag, lexical item and POS tag sequence within the chunk of child and parent are considered as the pattern. As both label and attachment are important for dependency analysis, our patterns contain child as well as parent features.

Once we get the frequencies at each level, we keep some threshold on the frequency and all the pairs less than that threshold are considered as possible errors. This threshold is decided after experiments with the development data and it can vary with annotation level. For all the pairs greater than the threshold, if a pattern has multiple tags, then there might be a possibility of error. So, for such pairs, if the frequency of a pair is less than certain percentage of the total instances of that pattern, then it is considered as a possible error.

The above approach is fine at POS level. But, when it comes to chunk and dependency levels, sparsity creates problems. Probability of occurrence of the same pattern is very low due to which a lot of valid instances get identified as errors. To resolve this, instead of original patterns, we find similarity between patterns and merge similar patterns. Again, the measure of similarity varies with annotation type. On these merged patterns, we apply the above approach to detect the errors.

4.2.2. Rule-based post-processing Module

The approach explained above about finding similarity patterns reduces the instances of correct patterns being identified as errors but not completely remove it. To further reduce the negative effect of sparsity on these merged patterns, we use certain robust rules to remove correct patterns from the errors list. So, a robust rule is capable of overriding a low frequency based pattern induction and can remove such pattern from the final selection.

4.2.3. Description of hybrid approach

Figure 3, shows the complete approach taking inter-chunk dependency as an example.

There are 6 pairs (pattern + tag) where all the patterns are different as shown in 3(a). As the frequency is low, all the 6 patterns are identified as errors. After finding similarity between patterns and merging similar patterns, 6 pairs get reduced to 3. This is shown in 3(b). The arrows connecting the patterns in (a) with (b) show the merging process. Similarity criterion used here is as follows:

For both child and parent chunks, consider POS type of the head of the chunk and lexical item and POS tags of the

functional words.
Out of 3 pairs in (b), 2 pairs are identified as errors based on statistics. After applying the following rule,
If the child is an adverbial chunk (RBP) and the parent is a verbal chunk (VGF), then the dependency label can be "adv".
the number of errors reduced from two to one.

5. Results and Analysis

We evaluated the performance of our system using a 65k-token (2694 sentences) manually annotated and validated sample of data derived from the Hindi dependency treebank. We divided the data into 40k, 10k and 15k for training, development and testing respectively. For the rule-based system, training and development data was used to frame the rules. In the case of hybrid approach, we used training data to train the models and development data to tune the parameters like threshold values. Rules meant for pruning false positives were also framed using this data.
We ran the rule-based tool on the test data. Details of the type and number of errors identified by the rule based system are presented in Table 1. Using our rule-based system we detected 75%, 62.5% and 25.86% of errors at POS, chunk and dependency levels respectively. Currently in the treebank, dependency annotation is done at inter-chunk level only. So, dependency errors only represent inter-chunk dependency errors.

Type of Error	**Total instances**	**Total Errors**	**Recall of the tool**
POS Errors	13922	16	12/16 = 75%
Chunk Errors	7113	24	15/24 = 62.5%
Dependency Errors	7113	843	218/843 = 25.86%

Table 1: Error Detection using rule-based system at different levels.

At POS and chunk levels, as the number of errors is low which can be identified based on some standard rules, rule-based system performs quite well. We also tried the hybrid approach, but the number of false positives is so high that the hybrid approach is practically of no use at POS and chunk levels.
But at dependency level, as more complex linguistic information is being annotated, the chance of making errors is more. As the number of errors is large we need tools to detect the errors so that the validation process becomes faster. With the rule based system we were able to identify only 25.86% of the dependency errors. We then tried out the hybrid based approach. Using this approach, we were able to identify 18.74% of the dependency errors. When we combined the outputs of both the rule-based and hybrid approaches, we could identify 40.33% of the errors at the dependency level. Results are shown in Table 2.

Approach	**Total Errors**	**System output**	**Correct Errors**	**Recall**
Rule Based Approach	843	218	218	25.86%
Hybrid Approach	843	2546	158	18.74%
Combining both the Approaches	843	2728	340	40.33%

Table 2: Recall of error detection using different approaches.

6. Discussion and Future Work

One basic difference between our approach and the other previous approaches is that we use a combination of a rule-based system and a hybrid system to detect errors. Most of the previous approaches work well with large corpora in which the frequency of occurrence of words is very high. Hence, none of them account for data sparsity. Our work is focused on detecting errors during the process of annotation. This means that the size that we worked on is not very large and hence we need to take care of the problems that accrue from sparsity. We employ a combination of a rule-based approach with a hybrid approach for error detection. Moreover, unlike earlier efforts, our work focuses on reduction of validation time and effort during treebank construction. So, our focus is on high recall with reasonable precision.
The tool is constantly being improved. We are planning to improve the rules of both the rule-based error detection system and the rule-based post-processing module of the hybrid approach. We also plan to experiment with different similarity criteria to improve the recall.
One limitation of our hybrid approach is that we can't give richer context due to the problem of sparsity. To find whether the dependency label is correct or not, apart from node and its parent information, sibling and child information is also helpful. Current state-of-the-art dependency parsers use these features for dependency labeling (McDonald et al., 2006; Ambati et al., 2009). Finding similarity between patterns and merging similar patterns would not help when we wish to take a much richer context. For this purpose, we also plan to explore a probability based hybrid approach. Instead of counts, we plan to use probabilities to detect the errors. We hope to achieve much better recall with the probability based hybrid approach.
This tool can also help in improving the guidelines which subsequently improves the annotation. While correcting the errors if the validator comes across some ambiguous decisions or some common errors or comes up with new decisions, guidelines can be modified accordingly to reflect the changes. Data annotated based on new guidelines will reduce the occurrence of these errors and eventually the quality of annotation of individual as well as entire data will improve. Figure 4, shows the complete cycle of this process.

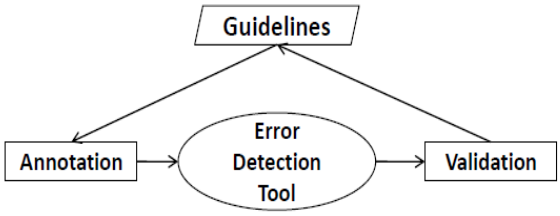


Figure 4: Cycle for improving guidelines for annotation.

7. Conclusion

In this paper, we proposed a new tool which uses both rule-based and hybrid systems to detect errors during the process of treebank annotation. We tested it on Hindi dependency treebank and were able to detect 75%, 62.5% and 40.33% of errors in POS, chunk and dependency annotation respectively. For detecting POS and chunk errors, we used the rule-based system. For dependency errors, we used the combination of both rule-based and hybrid systems. The proposed approach works reasonably well for relatively smaller annotated datasets.

8. Acknowledgements

We would like to thank Rafiya Begum for helping us during the validation process. The work reported in this paper is supported by the NSF grant (Award Number: CNS 0751202; CFDA Number: 47.070).

9. References

Ambati, B.R., Gadde, P., Jindal, K. (2009). Experiments in Indian Language Dependency Parsing. In *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, pp. 32-37.

Begum, R., Husain, S., Dhwaj, A., Sharma, D.M., Bai, L., Sangal. R. (2008). Dependency annotation scheme for Indian languages. In *Proceedings of IJCNLP-2008*.

Bharati, A., Chaitanya, V., Sangal, R. (1995). *Natural Language Processing: A Paninian Perspective*, Prentice-Hall of India, New Delhi, pp. 65-106.

Bharati, A., Sangal, R., Sharma, D.M., Bai, L. (2006). AnnCorra: Annotating Corpora Guidelines for POS and Chunk Annotation for Indian Languages. *Technical Report (TR-LTRC-31)*, Language Technologies Research Centre, IIIT-Hyderabad. http://ltrc.iiit.ac.in/MachineTrans/publications/technicalReports/tr031/posguidelines.pdf

Bharati, A., Sangal, R., Sharma, D.M., Bai, L. (2009). AnnCorra: TreeBanks for Indian Languages, Guidelines for Annotating Hindi TreeBank. http://ltrc.iiit.ac.in/MachineTrans/research/tb/DS-guidelines/DS-guidelines-ver2-28-05-09.pdf

Bharati, A., Sangal, R., Sharma, D.M. (2007). SSF: Shakti Standard Format Guide. *Technical Report, TR-LTRC-33*, Language Technologies Research Centre, IIIT-Hyderabad, India. http://ltrc.iiit.ac.in/MachineTrans/publications/technicalReports/tr033/SSF.pdf

Bhatt, R., Narasimhan, B., Palmer, M., Rambow, O., Sharma, D.M., Xia, F. (2009). Multi-Representational and Multi-Layered Treebank for Hindi/Urdu. In *Proc. of the Third Linguistic Annotation Workshop at 47th ACL and 4th IJCNLP.*

Boyd, A., Dickinson, M., Meurers, D. (2008). On Detecting Errors in Dependency Treebanks. *Research on Language and Computation* 6(2), pp. 113-137.

Dickinson, M., Meurers, W.D. (2003). Detecting Inconsistencies in Treebank. In *Proc. of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*.

Dickinson, M., Meurers, W.D. (2005). Detecting Errors in Discontinuous Structural Annotation. In *Proc. of the 43rd Annual Meeting of the ACL*, pp. 322–329.

Eskin, E. (2000). Automatic Corpus Correction with Anomaly Detection. In *Proceedings of the First Conference of the North American Chapter of theAssociation for Computational Linguistics (NAACL-00)*. Seattle, Washington.

van Halteren, H. (2000). The Detection of Inconsistency in Manually Tagged Text. In *Proceedings of the 2ndWorkshop on Linguistically Interpreted Corpora*. Luxembourg.

Kordoni, V. (2003). Strategies for annotation of large corpora of multilingual spontaneous speech data. In *Proc. of Workshop on Multilingual Corpora: Linguistic Requirements and Technical Perspectives held at Corpus Linguistics 2003.*

Kaljurand, K. (2004). Checking treebank consistency to find annotation errors.
http://math.ut.ee/~kaarel/NLP/Programs/Treebank/ConsistencyChecking/.

Marcus, M.P., Marcinkiewicz, M.A., Santorini, B. (1993). Building a large annotated corpus of English: the Penn treebank. *Computational Linguistics*, Volume 19, Issue 2, pp. 313 – 330.

McDonald, R., Lerman, K., Pereira, F. (2006). Multilingual dependency analysis with a two-stage discriminative parser. In *Proc. of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pp. 216–220.

Palmer, M., Gildea, D., Kingsbury, P. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics, 31(1):*71-106.

Xia, F., Rambow, O., Bhatt R., Palmer, M., Sharma, D.M. (2009). Towards a Multi-Representational Treebank. In *Proc. of the 7th International Workshop on Treebanks and Linguistic Theories (TLT 2009)*, Groningen, Netherlands.