Automatic Enhancement of LTAG Treebanks

Farzaneh Zarei, Ali Basirat*, Heshaam Faili and Maryam Sadat Mirian
School of Electrical and Computer Engineering
College of Engineering
University of Tehran, Tehran, Iran
{zareeifarzaneh,hfaili,mmirian}@ut.ac.ir
*a.basirat@srbiau.ac.ir

Abstract

The Treebanks as the sets of syntactically annotated sentences, are the most widely used language resource in the application of Natural Language Processing. The occurrence of errors in the automatically created Treebanks is one of the main obstacles limiting the using of these resources in the real world applications. This paper aims to introduce an statistical method for diminishing the amount of errors occurred in a specific English LTAG-Treebank proposed in Basirat and Faili (2013). The problem has been formulated as a classification problem and has been tackled by using several classifiers. The experiments show that by using this approach, about 95% of the errors could be detected and more than 77% of them could successfully be corrected in the case of using Adaboost classifier. In addition, it has been shown that the new treebank could reach a high of 76% F-measure which is 8% higher than the original treebank.

1 Introduction

Treebanks, as special corpora annotated with syntactic structures, play a crucial role in the recent success of natural language processing applications like speech recognition, spoken language systems (Xue et al, 2005), parsing (Mirroshandel et al, 2012), and machine translation (Kotze et al, 2012).

Regarding the development methods of the treebanks, generally, they can be placed in either manually crafted or automatically extracted treebanks. Due to the large number of sentences, the manual creation of the treebanks can be very expensive and time consuming. For instance, Penn English treebank as one of the outstanding handmade ones took eight years (1989-1996) to be completed. The difficulties, raised in the manual creation of Treebanks, led the researchers to use automatic and semi-automatic methods of treebank development methods. On the other hand, the automatically extracted Treebanks are not as accurate as manual versions. In fact, these resources mostly suffer from the occurrence of error in the annotated sentences that in turn reduces the applicability of these resources in the real world applications.

A large number of researchers tried to improve the quality of the automatically extracted Treebanks in order to increase the applicability level of these resources in the NLP tasks (Xue et al, 2005).

For instance Dickinson and Meurers (2003) proposed an n-gram based approach for detecting Part-of-Speech errors in Penn English Treebank. In other works, Agarwal et al. (2012) proposed a hybrid approach to improve the mechanism of error detection introduced by Ambati (2011) for detecting the errors in a dependency treebank. Ule and Simov (2003) also could find unexpected tree productions by using a method called Directed Treebank Refinement (DTR).

In this paper, we try to correct the errors occurring in the treebank automatically generated from the approach proposed by Basirat and Faili (2013). This Treebank named *LTAG Treebank* is a corpus of supertag annotated sentences. A supertag is an abstract concept of the syntactic structures defined by the elementary structures of the lexicalized grammars like LTAG, HPSG, and CCG. This concept, as an extension of a simple part-of-speech tag, provides a rich and complex linguistically motivated description for the lexical items of language. A concrete instance of this concept is the elementary tree of a Lexicalized

Proceedings of Recent Advances in Natural Language Processing, pages 733–739,
Hissar, Bulgaria, 7-13 September 2013.

Tree-Adjoining Grammars (LTAG), which gives a comprehensive description of the syntactic environment on which a word can be appeared.

In the supertag annotated corpus, the supertags are considered as the elementary trees of a typical LTAG of English, called XTAG grammar. The main interesting point of this grammar is the linguistically motivated descriptions provided by the elementary trees of this grammar for the syntactic environments of the words. Each sentence in the LTAG Treebank is associated with a sequence of elementary trees of XTAG grammar that directly defines a set of parse trees for the sentence, regarding the standard tree attachment operations defined in the LTAG formalism called substitution and adjunction.

In order to correct the miss-annotated words in the LTAG Treebank, a discriminate based formulation of the problem working in two main steps: *error detection* and *error correction* have been proposed. The error detection phase is responsible for detecting the miss-annotated words by employing some contextual features of the words. The output of this phase beside the other contextual features of the word then would be used by the error correction phase in order to find the best candidate among all elementary trees that can be assigned to the word.

To do so, two main classes have been considered for the error detection phase, *correct* and *incorrect*. Regarding this fact that in LTAG Treebank the number of miss-annotated words is much less than correct ones, the classes are imbalanced.

To the purpose of error detection and correction, three different classification methods have been employed: Adaboost (Freund and Schapire, 1998), Multilayer perceptron (MLP) and C4.5 (Quinlan, 1993). These classifiers are chosen due to the following justifications:

- Adaboost is a strong classifier in handeling imbalanced data (Japkowicz and Stephen, 2002).
- MLP is a universal function approximator.
- C4.5 is a decision tree approach which facilitates visualization of the found rules.

To handle the aforementioned class imbalance problem, these classifiers have been suggested. Japkowicz and Stephen studied several resampling methods and established the relation among concept complexity, size of the training set and class imbalance level.

We selected C4.5 among decision tree classifiers as it is a typical classification approach and has some superiorities to ID3 such as handling missing values and different feature costs.

On the other hand, there have been a number of studies on extending Adaboost to imbalanced datasets(Japkowicz and Stephen, 2002).

By applying these classifiers on the LTAG Treebank in the best case the precision increased by 8% and reached 76%.

The rest of this paper would be as follows: Sec. 2 gives brief information about the LTAG Treebank used in this work. Sec. 3 deals with the feature selection of the classifiers. In the next section, Sec. 4, the classification methods is expressed in details. Finally, Sec. 5 elaborates the numerical results of the error detection and correction. It also represents the quality of the resultant LTAG-Treebank according to different evaluation criterion.

2 LTAG Treebank

An LTAG-Treebank can formally be defined as a set of sentences annotated with the elementary trees of a lexicalized tree-adjoining grammar each of which defines a set derived/derivation trees for the sentences. The LTAG on which this work is focused has been developed as a part of a grammar development system, called XTAG. The importance of this grammar can be seen in the linguistic notions and rich feature structures like semantic representations that are embedded in its elementary trees. Nevertheless, lack of enough statistical information of co-occurrence elementary trees of XTAG grammar has limited its usage in the powerful statistical and machine learning approaches proposed in the recent decades.

It is expected that the LTAG-Treebank, can significantly compensate this weakness of the XTAG grammar by providing the empirical probability distributions of the co occurring elementary trees.

The idea of automatic error detection and correction mentioned in this work, has been applyied on the LTAG treebank introduced in Basirat and Faili (2013). This treebank has been developed based on the hidden relationship between two LTAGs of English, XTAG grammar and an automatically extracted LTAG used by MICA parser (Bangalore et al, 2009).

We have applied this method on a subset of sentences of Wall Street Journal (WSJ) in order to annotate them with the elementary trees of the

XTAG grammar. The result was a set of English sentences and their related XTAG elementary tree sequences each of which could define a set of parse trees for their sentences.

One of the difficulties raised in using this approach is the occurrence of errors in the elementary trees assigned to the words. Regarding the standard tree attachment operations defined in TAG, the existence of these errors would lead to the following problems: i) The sequence of elementary trees that cannot attach to each other to create a parse tree for the sentence. ii) The sequences of elementary trees that can attach to each other but the resultant parse tree is not correct.

In principle, the occurrence of these errors is the direct consequence of the weakness of the classifier used by Basirat and Faili (2013) for assigning the XTAG elementary trees to the sentences. The assignment was based on a Hidden Markov Model (HMM) and due to the inherent weaknesses of the HMM, some miss-annotations in the generated Treebank have occurred.

As a specific example, the sentence "I believe in the system" is labeled by the approach proposed by Basirat and Faili (2013). Table 1 shows the output of this approach and its correct version.

word	Output of the HMM	Correct version
I	alphaNXN	alphaNXN
believe	alphanx0V	alphanx0V
in	betavxPnx	betavxPnx
the	alphaD	betaDnx
system	alphaDnx0V	alphaNXN

Table 1: the output of the HMM for the sentence "I believe in the system" and its corrected version

Figure 1 also shows the elementary trees resulted from the HMM proposed by Basirat and Faili (2013). As it can be seen, these elementary trees cannot attach to each other in order to create full parse tree. Because miss annotating occurred in two last word of the sentence. But after correcting these errors, we would have a full parsed tree.

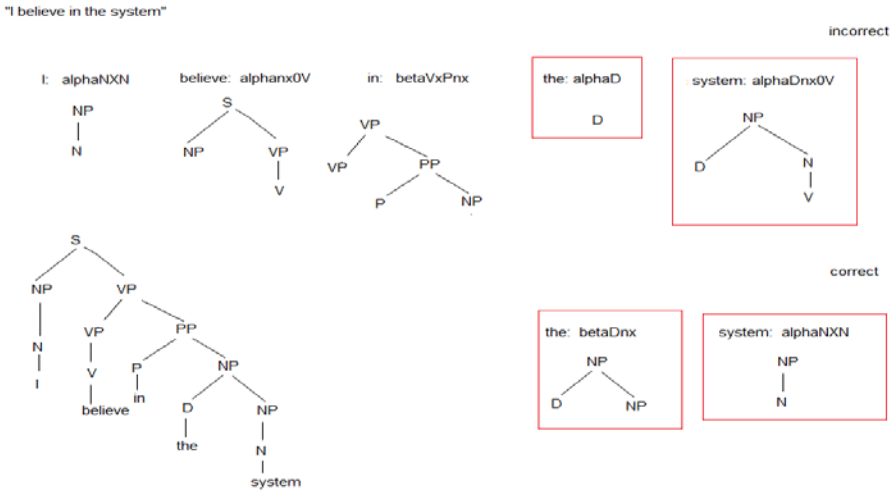


Figure 1: the output of the HMM for the sentence "I believe in the system" and its corrected version

3 Feature Selection

Depending on the type of errors occurred in the LTAG tree bank, several features can be used to detect them. For instance, the erroneous sequences that cannot lead to full parse trees can be analyzed by using the contextual information of the words (e.g., POS tag of the word and its neighbors, the morphological information of the word, the word itself and its neighbors, etc). The dependency information of the words can also be helpful for finding the errors in the sequences that might result in parse trees but incorrect parse trees.

Because of the huge size of the set of language words, among all contextual information of the words, three features were selected including the *Part Of Speech tag*, the *XTAG elementary tree*, and *supertag* of the words. Here the *supertag* is selected from the set of elementary trees of another LTAG used by the MICA parser. The XTAG elementary tree also is the elementary tree initially assigned to the word by using the aforementioned LTAG treebank builder. The main reason for using these features is their ability in encapsulating the complexity of the syntac-

tic environment of the words in the structural objects to be used by the discriminant based classifiers like neural networks.
The dependency information of the words can also be encoded into the feature vector by using an extra input item representing the information of governor/dependent relationship of the words. This information can be extracted from the dependency tree of the sentence generated by the MICA parse. Just like what was done for the contextual information of the words, here also instead of using the governor *word*, its MICA supertag is used.
To summarize, the feature vector used by the classifier would contain contain the following elements:

- The XTAG elementary tree of the word
- The POS tag of the word
- The MICA supertag of the word (dependent)
- The MICA supertag of the governor

Using this set of features, the only extra tool for development of the set of feature vectors is the MICA parser. The POS tag also is extractable from the MICA supertag sequences generated by the MICA parser.

4 Classification

As mentioned before, the LTAG treebank creation method introduces in Basirat and Faili is based on the hidden markov model which does not provide any clear solution for using extra information of the word such as syntactic environment information. The suggested classification approach proposed in this paper, however enables us to easily use a lot of essential information of the word such as POS tag and its dependency information.

The task of correcting annotations can be done in two steps: i) Detecting the XTAG elementary trees that are incorrectly assigned to the words. ii) finding the correct labels for them.

Detecting the errors can be considered as a binary classification problem in which each word is classified *correct* or *incorrect* with respect to the XTAG elementary tree. Despite the detection, in the correction phase the number of the classes is equal to the number of the XTAG elementary trees appeared used in the Treebank.

Although the number of XTAG elementary trees is more than 1000, just 115 trees out of them were used in our corpus (before and after correction).

The rest of this section, would elaborate the implementation of each of these classification algorithms.

4.1 Adaboost

In boosting algorithms training data are classified by some weak classifiers iteratively. In each iteration, Boosting reweights the training data, such that the weights of correctly classified instances are decreased and the others are increased.

The week classifier used in our algorithm is Random Forest. Although, Random Forest is not as weak as a naïve bayse[1], we coupled them with Adaboost in order to utilize their power to conquer the imbalanced problem we face here. The combination of Adaboost and Random Forest has been used in the traffic flow (Leshem, and Ritov, 2007) and cancer survivability (Thongkam et al, 2008) and improve the performance of them.

4.2 Multilayer Perceptron (MLP)

A three-layered feed-forward neural network (one hidden layer containing 30 neurons) was trained, using back propagation algorithm. The back propagation training algorithm with generalized delta learning rule is an iterative gradient algorithm designed to minimize the mean square error between the actual output of a multilayered feed-forward neural network and a desired output.

4.3 C4.5

One of the famous algorithms which divides and conquers a problem for constructing a decision tree is C4.5. The model describes the condition of independent attributes that leads to each class prediction. The approach selects and places an attribute at the root node to generate one branch for each possible value of the attribute. The criterion for attribute selection involves obtaining a maximum information gain using the information theorem (Quinlan, 1993). And then, the branches can split the instances into numerous partitions, including one for every attribute value. Finally, each partition recursively repeats the splitting process until all instances at a node are in the same class. A pruning strategy is applied to reduce size of the decision tree.

[1] We tried NB as the weak learners and the resulting performance was not satisfactory.

In the next section we elaborate the numerical results obtained from correcting the proposed LTAG Treebank.

5 Evaluation

The classification method has been run on a subset of sentences of Wall Street Journal (WSJ) annotated with the elementary tree of the XTAG grammar. To this end, among the all sentences shorter than 40 words, 1393 sentences were randomly selected to be annotated with the XTAG elementary tree. The annotation process has been done according to the Treebank creation method introduced in Basirat and Faili (2013). Then, output of the annotation process has been manually corrected in order to be used as the gold standard in the evaluation phase.

Table 2 gives some statistical information of these sentences.

	train	**test**
Total number of sent	1,293	100
Total number of words	12,630	1,042
Avg length of sent	9.7	10.42
Avg number of errors per sent	1.57	1.46
Total number of correct annotated words	10,600	896
Total number of miss-annotated words	2,030	146

Table 2: selected sentences annotate with XTAG elementary trees

We employed some standard metrics in error detection and correction in order to evaluate the output of the classifiers. The measures are as follows:

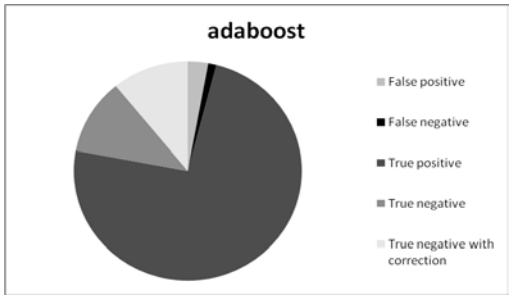
- *False positive* (FP): refers to real errors that were not identified by the classifier.
- *False negative* (FN): refers to correct annotated word that the classifier detected as real errors.
- *True positive* (TP): refers to correct annotated words that are also considered as correct in the gold data.
- *True negative* (TN): refers to correct annotated words that the classification method changed regardless of the correction.
- *True negative with correction* (TNC): are real errors that the classification method was able to replace with the correct XTAG elementary trees.

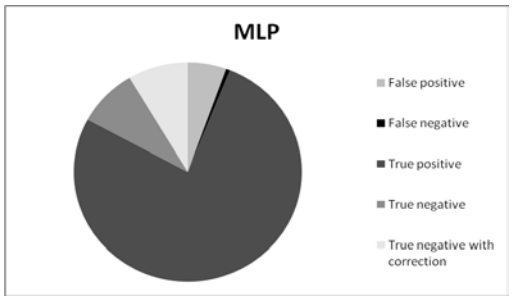
By comparing the result of each classifier to the gold data, all the mentioned measures are calculated. Table 3 contains the evaluation results of each classifier.

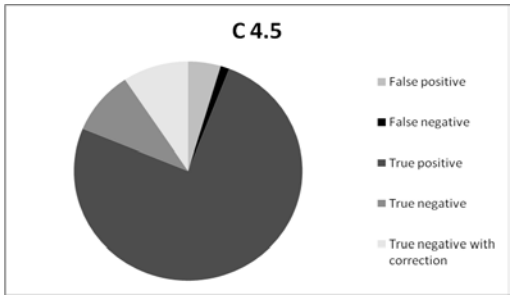
	Adaboost	**MLP**	**C4.5**
False positive	30	57	48
False negative	12	6	13
True positive	768	801	785
True negative	116	89	98
True negative with correction	113	89	97

Table 3: output statistical information of the classifiers

Fig. 2, 3, 4 demonstrate performance of each selected classifiers.







As we expected, the selected classifiers are strong enough to detect and correct a large proportion of errors correctly.

By using these metrics, we define four evaluation measures.

- Precision: The proportion of the correctly detected errors. That is, how many errors that the classifier detects were actually correct

$$precision = \frac{TP}{TP + FP}$$

- Detection Recall: The fraction of real errors detected by the classifier. That is, how many errors that have been detected by the classifier is actually error.

$$Detection\,\mathrm{Re}call = \frac{TN}{FP + TN}$$

- Correction Recall: The fraction of real errors corrected by the classifier. That is, how many errors that have been corrected by the classifier is actually error.

$$Correction\,\mathrm{Re}call = \frac{TNC}{FP + TN}$$

- Accuracy (A): the total number of correctly detected word divided by the total number of the word

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision, detection recall, correction recall and accuracy of each classifier is calculated and shown in Figure 5.

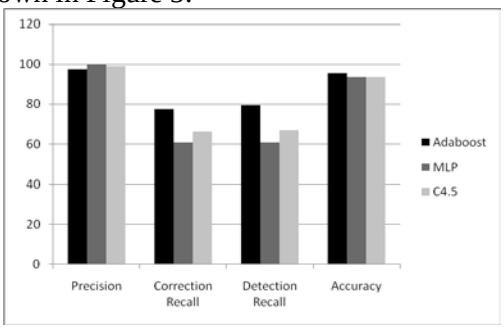


Figure 5: Precision, recall and accuracy of each classifier

According to Figure 5 although the MLP has a highest precision (100%), its recall is slightly lower than the C4.5 classifier. The accuracy of the C4.5 and MLP classifiers are almost equal. The Adaboost classifier appeared to outperform the others. It has the best detection and correction recall but its precision is only slightly lower than the best case (97.41%).

The rest of this section deals with the evaluation of the developed Treebank.

5.1 Evaluation of LTAG treebank

Precision, recall and F-measure are three primary evaluation criteria to measure the quality of a parse tree.

Table 4 represents quality of the LTAG-Treebank before applying the error correction. It shows the values of precision, recall and F-measure of the parse trees generated from the elementary tree sequences with respect to the gold parse trees available in the Penn-Treebank.

	Before correction
Precision	54.78
Recall	88.80
F-measure	67.76

Table 4: precision, recall and F-measure of LTAG treebank before correction

Table 5 shows same criteria for the parse trees after applying error detection and correction methods.

	precision	recall	F-measure
Adaboost	63.99	94.72	76.38
MLP	63.67	94.33	76.03
C4.5	63.40	94.84	76.00

Table 5: precision, recall and F-measure of LTAG treebank after correction

As can be seen, the value of F-measure of the parse trees after applying the error detection and correction could significantly be improved.

6 Conclusion

In this paper, we proposed an error detection and correction method for improving the quality of automatically created LTAG Treebank introduced in Basirat and Faili (2013). The problem was formulated as a sequence classification problem and tackled by using three classifiers Adaboost, Multi Layer perceptron (MLP) and C4.5.

Because of the imbalanced situation of the problem in which the ratio of correctly annotated words was much higher than the miss-annotated words, the Adaboost classifier with random forest as a week learner could provide better results in comparison with the other classifiers. By applying the classifiers on the LTAG treebank, in the best case, the value of F-measure of the treebank could be increased by 8 % compared with the initial treebank.

References

Ali Basirat and Hesham Faili. 2013. *Bridge the gap between statistical and hand-crafted grammars*, Computer Speech and Language, volume 27, Pages 1085-1104

Gideon Kotze, Vincent Vandeghinste, Scott Martens, Jorg Tiedemann. 2012. Large aligned treebanks for syntax-based machine translation. Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12). European Language Resources Association (ELRA).

Guy Leshem, Yaacov Ritov, 2007, *Traffic flow prediction using adaboost algorithm with random forests as a weak learner*, in: Proceedings of World Academy of Science, Engineering and Technology, vol. 19, Bangkok, Thailand, 193–198

Jaree Thongkam, Guandong Xu, and Yanchun Zhang. 2008. *AdaBoost algorithm with random forest for predicting breast cacer survivability*, IJCNN, 3062-3069. IEEE(2008)

John Ross Quinlan, 1993. *C4.5 programs for machine learning*. San Mateo, CA: Morgan Kaufmann

Naiwen Xue , Fei Xia, Fu-Dong Chiou, Marta Palmer. 2005. *The penn chinese treebank: Phrase structure annotation of a large corpus*. Natural Language Engineering , 11(2), 207-238.

Nathalie Japkowicz and Shaju Stephen.2002. *The class imbalanced problem: A systematic study.* Intelligent Data Analysis. Volume 6. 429-449.

Markus Dickinson, and W.Detmar Meurers, 2003 , *Detecting Errors in Part-of-Speech Annotation.* In The 10th Conference of European Chapter of the Associa-tion for Computational Linguistics(EACL-03).

Mitchell P. Marcus, Mary Marcinkiewicz, Beatrice Santorini.1993. *Building a large annotated corpus of english: The penn treebank.* Computational Linguistics - Special issue on using large corpora 19(2), 313 – 330.

Rahul Agarwal, Bharat Ambati, and Dipti Sharma. 2012, *A Hybrid Approach to Error Detection in a Treebank and its Impact on Manual Validation Time*, volume 7. Linguistic Issues in Language Technology.

Seyed Abolghasem Mirroshandel, Nasr Alexis, Joseph Le Roux. 2012. *Semi-supervised dependency parsing using lexical affinities*. ACL '12 Proceedings of 50th Annual Meeting of the Association for Computational Linguistics. Volum 1, 777-785

Srinivas Bangalore, Anoop Sarkar, Christine Doran, Beth Ann Hockey, 1998, *Grammar & parser evaluation in the xtag project*. In: Proceedings of the Workshop on Evaluation of Parsing Systems, Granada ,Spain. Language Resources and Evaluation Conference.

Srinivas Bangalore, Patrick Haffner, and Ga¨el Emami. 2005. *Factoring global inference by enriching local representations.* Technical report, AT&T Labs – Reserach.

Srinivas Bangalor, Pierre Boullier, Alexis Nasr, Owen Rambow, Benoit Sagot. 2009. *Mica: A probabilistic dependency parser based on tree insertion grammar.* NAACL-Short '09 Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, vol. Short Papers, 185–188.

Yoav Freund and Robert E. Schapire, 1996, *Experiments with a New Boosting Algorithm*, Proc. of 13th International Conference on Machine Learning, pp. 148—156.