From Partial toward Full Parsing

Heshaam Faili
Department of Electrical and Computer Engineering
University of Tehran
Tehran, Iran
hfaili@ut.ac.ir

Abstract

Full-Parsing systems able to analyze sentences robustly and completely at an appropriate accuracy can be useful in many computer applications like information retrieval and machine translation systems. Increasing the domain of locality by using tree-adjoining-grammars (TAG) caused some researchers to use it as a modeling formalism in their language application. But parsing with a rich grammar like TAG faces two main obstacles: low parsing speed and a lot of ambiguous syntactical parses. In order to decrease the parse time and these ambiguities, we use an idea of combining statistical chunker based on TAG formalism, with a heuristically rule-based search method to achieve the full parses. The partial parses induced from statistical chunker are basically resulted from a system named supertagger, and are followed by two different phases: error detection and error correction, which in each phase, different completion heuristics apply on the partial parses. The experiments on Penn Treebank show that by using a trained probability model considerable improvement in full-parsing rate is achieved.

Keywords

Full Parsing, Partial Parsing, Tree Adjoining Grammar, SuperTagging

1. Introduction

In many applications like information retrieval and Rule-based machine translation systems, accurate deep parse structure of a sentence is required; hence a lot of research is being done on introducing methods to produce deep hierarchical syntactical structure of a given natural language sentence [6]. Over the last decade, there has been a great increase in the performance of parsers. Current parsers achieve to a score of about 90% when measuring just the accuracy of choosing these dependencies [4, 5 and 7]. The choice of formalism does not change the parsers' accuracy significantly, because in all approaches word-word dependencies are used as the only underlying information. But because of the inherent ambiguity in the natural languages, achieving to a full parses of a sentence is a big challenges.

Tree-adjoining-grammars (TAG) have some specific features, which are interested by researchers to be used as modeling formalisms in their language application. The parsing methods based on this formalism involve different problems such as a lot of ambiguities and low parsing speed. One of the main parsing algorithms based on TAG formalism is presented by Van Noord [10] which runs in $O(n^6)$ time complexity. This complexity in a real-size grammar (like XTAG [9]) is not acceptable, especially for a more complicated system like information retrieval and machine translation systems. Also, because of the ambiguities in the resulted parses, the output of this algorithm must be disambiguated by another approach.

To overcome the mentioned problems, we use an alternative approach which is based on statistical partial parsers. One of the partial parser systems which alleviate the TAG formalism problems in time complexity and ambiguity is named supertagging, proposed by (Bangalore and Joshi [2]). The idea behind supertagging is to extend the notion of "tag" from a part of speech to a tag that represents rich syntactic information. Each supertag can be thought as an element in TAG formalism.

They also introduced "lightweight" parsing which follows the supertagging. If words in a string can be tagged with this rich syntactic information, then Bangalore and Joshi claim, the remaining step of determining the actual syntactic structure is trivial [2]. They propose a "lightweight dependency parser" (LDA) which is a heuristically-driven, very simple program that creates a dependency structure from the supertags of the words. While the supertagging only requires a notion of syntactically relevant features, the stage of determining a syntactic structure requires a grammar that uses these syntactically relevant features. Given the correct supertags, LDA performs with an unlabeled accuracy of about 95%.

Although supertagging is a worthwhile notion pursuing the full-parsing, but approaching to a full-parse by the proposed lightweight parser has a major obstacle. Bangalore announced the accuracy of supertagging to be about 92% based on the experiments done on Penn Treebank [1]. This accuracy is not satisfiable to generate a complete deep structure of the sentence by using lightweight dependency analyzer. In a sentence with 15-words length, LDA parser determines the correct full-parse of the sentence with the probability about $95\% * (0.92)^{15} = 27.5\%$. For longer sentences, lower accuracy has been achieved. Nasr and Rambow try to improve the accuracy by changing the heuristic dependency linker with a non-lexical chart parser [8]. Like the original supertagger, their

International Conference RANLP 2009 - Borovets, Bulgaria, pages 71–75

method still has no access to lexical information and only information about the supertags is combined with a chart parser. They cut the error rate of the heuristic LDA by more than half.

In this paper, we present a full-parsing method by combining different heuristics with lightweight shallow parser. Our approach is still in the spirit of Bangalore's work in the sense that lexical information is only used during supertagging. The idea of this paper is based on finding the erroneous supertags which are most probable to be wrongly assigned, and then replacing them with proper candidates.

2. Full Parsing

Although full parsing based on fully correct supertags is very time-efficient [8], but acquiring the fully correct supertagging itself is the main obstacle. The probability of assigning correct supertag set $S=\{s_1,s_2,\ldots,s_n\}$ to all words of a sentence $W=\{w_1,w_2,\ldots,w_n\}$ is equal to product of the probability of correct assigning a single supertag s_i to i-th word w_i (i.e. $p(s_i \mid w_i)$). Based on the experiments done by Bangalore, the probability $p(s_i \mid w_i)$ is equal to 92.2%. So full parsing probability by linking all supertags resulted from supertagging process for a sentence with 15 words length is equal almost be 29.5% and with 25 words near to 13.1%. To overcome this problem, n-best supertagging that assign n-best supertags to each word was proposed by [1]. Based on this approach, by setting n = 3, supertagging correctness increased to 97.1% and accordingly the rate of fully-parsing for whole words in a sentence improved efficiently. (e.g. 74.5% for sentences with 15 words length and 64.3% for sentences with 25 words length). But using n-best supertags followed by lightweight analyzer is equal to find a combination of these supertags which satisfies all available syntactical constraints on TAG. For 3-best supertagger in 15 words length sentence, there are 3^{15} = 14,348,907 combinations which should be checked in order to choose the correct combination. In [8] a dynamic programming method to resolve this complexity is used.

This problem can be seen as a search problem in the state space of all supertags assigned to the words of the sentence. The initial state is a combination of those tags which are assigned by supertagger and the goal states are those which LDA succeed to make a fully dependency linkage between the supertags and hence in those states full-syntactic structure of the sentence is generated. Hill-climbing approach is chosen for search method and the accuracy of LDA is calculated as a heuristic performance measure of problem.

3. Search in the Supertag State Space

Same as other local search problems, the search can be divided into two distinct phases: error detection and correction. In fact, instead of associating n-best supertag to every word of the sentence, the most probable erroneous supertags resulted from n-best supertagging are detected and substituted with proper alternatives which are proposed by an error correction algorithm.

In each non-goal state (i.e. partial parse), error nodes are supertags that are wrongly assigned and therefore they are the cause of preventing LDA to produce exactly one dependency diagram as the correct full parse tree of the sentence. The result of LDA is a dependency diagram which links all supertags based on its syntactical behavior [1]. Four our experiments, we gathered 341 sentences, which are failed to be parsed by LDA, and analyzed the failing reasons. In the case of failing LDA to generate the full connected structure, one of the three cases may happen. These cases are shown in Table 1. As it's shown in the table, different heuristics for detecting the faulty nodes are demonstrated too. These heuristics show the supertags which are most probable to be wrong and should to be replaced with proper candidates.

Table 1. The cases in which supertagger fails to generate the full syntactic structure

Case 1	The LDA output diagram is not fully connected graph and it contains multiple partial graphs. In this case, substitution slots of some supertags are not filled by other tags. From the total 341 faulty test sentences, this case appears in 172 sentences, that is about 50% of all corpus fails to be parsed because of this problem.
Proposed faulty nodes in case 1	The partial trees' root is mostly an erroneous node, which its supertag should be substituted to better one (i.e. should to replace with another supertag which contains more substitution slots in order to make a link with other partial trees). Changing this node with proper one could correct 150 sentences from the total 172 faulty sentences of this case.
Case 2	Supertags of some words do not participate in the dependency diagram and so some child nodes are not included in its parent diagram. Either footnode or substitution slots are required to make a link between the orphan child and parent node. This case appears in more than 30% of test sentences.
Proposed faulty nodes in case 2	The root node of trees that some of their children are missed has a large potential to be wrongly assigned supertag. These missed children can be seen as slots that are not filled. In our experiments, the total faulty sentences of this case have been corrected by changing this node.

Case 3	In the 15% of mentioned faulty test sentences, the LDA output diagram has cycles in its dependencies and therefore is not a valid dependency structure diagram.
Proposed faulty nodes in case 3	In the case of existence any loop in the diagram, the verb nodes are usually ambiguous and have a large potential to be erroneous. By using this heuristic, the full parse structure of 70% of all unparsed sentences of case 3 is correctly acquired.

In each of the mentioned cases, the noisy nodes are detected and then replaced with some other supertags which will be proposed by other heuristics. So, the whole search for finding the full-parse can be summarized as the follows:

```
1- Use supertagger to achieve partial parse
2- Detect the full linkage by using LDA
3- In the case of using full linkage, stop
4- In the case of failure the full parses,
   check if one of the three mentioned cases
   happened
5- In the case of happening one of the
   mentioned cases, replace the faulty node
   proposed by error detection heuristic
   with a better candidate
6- Go to step 2
```

4. Error Correction Heuristics

After detecting the erroneous nodes, a list of proper candidates required to be substituted with the erroneous supertags. Three heuristics are presented here to propose the candidates to be replaced with the erroneous nodes, where each of which improves the full parsing rate and speed. These heuristics are as follows:

4.1 N-Best Heuristic

In this heuristic, the outputs of n-best supertagger are used as successor candidates. The n-best supertagger is a modified version of simple supertagger which proposes n supertags for each word of the sentence. Suppose that *m* is the number of faulty nodes which are detected by the previously mentioned heuristics and *n* is the number of n-best candidates which are predicted by supertagger, so finding the best combination in this space involves $O(m^n)$ cases. Breath first search (BFS) strategy is used to find the best match in this state space. That is for each node; all its successor nodes are generated first and then are evaluated by LDA as an evaluation function. The search terminated when the full parse structure of the input sentence is acquired.

4.2 XTAG-Based Heuristic

In this heuristic, a human-crafted grammar based on tree-adjoining formalism, named XTAG, is used. XTAG is an on-going project to develop a wide-coverage grammar for English using TAG formalism [9]. XTAG uses Lexicalized TAG, where each lexical item is associated to many elementary trees which can satisfy its structural constraints. In this heuristic, these associations between each lexical item and elementary trees are used as candidates to be replaced with the detected faulty nodes.

When an error node is detected, other TAGs, which are associated to those nodes' lexical in the XTAG grammar bank, are chosen as a substitution list. XTAG grammar contains 1226 elementary trees which are categorized into 26 different family trees, and each lexical item especially verb, associated to more than 10 elementary trees. Thus, the candidate list to be substituted with erroneous nodes in this method is much larger than previous one. Therefore, both the time and performance are much higher than n-best correction heuristic.

4.3 Trained Probability Model Heuristic

Although n-best is faster than XTAG heuristic, but the performance of full-parsing is much lower. In the first method the candidate list for correcting the error nodes is so shorter than the later one, and thus it needs less time to search among the combinations. Here a method using a trained probability model is proposed. In fact, for any supertags s_i, s_j, the probability of changing a faulty supertag s_j to supertag s_i (i.e. $P(s_i \mid s_j)$) which concludes a full-parse tree is calculated.

These probabilities are estimated by using maximum likelihood estimation method with counting the number of successful changes of faulty supertag (s_j) to correct supertag (s_i). By using from an annotated corpus of 40,000 sentences and their syntactic parses, these changes are computed in an iterative fashion. At each iteration, the sentences are tagged by the supertagger and the correctness of LDA algorithms is checked by the previously mentioned error detection heuristics and the erroneous nodes are detected. The faulty nodes then substituted with other supertags proposed by a combination of XTAG-based and 10-best heuristics. Each time an error supertag s_j is replaced with supertag s_i, the resulting parse structure is evaluated by PARSEVAL metrics [3]. If the result is a satisfiable full deep structure, the frequency of successful changing s_i to s_j increases one unit. The whole process of calculating the probability model $P(s_i \mid s_j)$ is shown in figure 1.

The training algorithm is terminated when the changes of the probabilities after running the experiment on the whole 40,000 sentences become ignorable. That is the total number of changes in whole probabilities becomes less that a predefined threshold. In our method, we set this threshold to be less than 0.05% of all entry values. At the end of process, all frequencies of changes in any faulty node should be normalized by using equation (1) in order to get

the probability $P(s_j \mid s_i)$. Having these probabilities, an ordered list of candidate nodes for any error supertag si is achieved, which can be used in the error correction method:

$$P(s_i \mid s_j) = \text{count}(s_i, s_j) / \Sigma_k \text{count}(s_k, s_j) \quad (1)$$

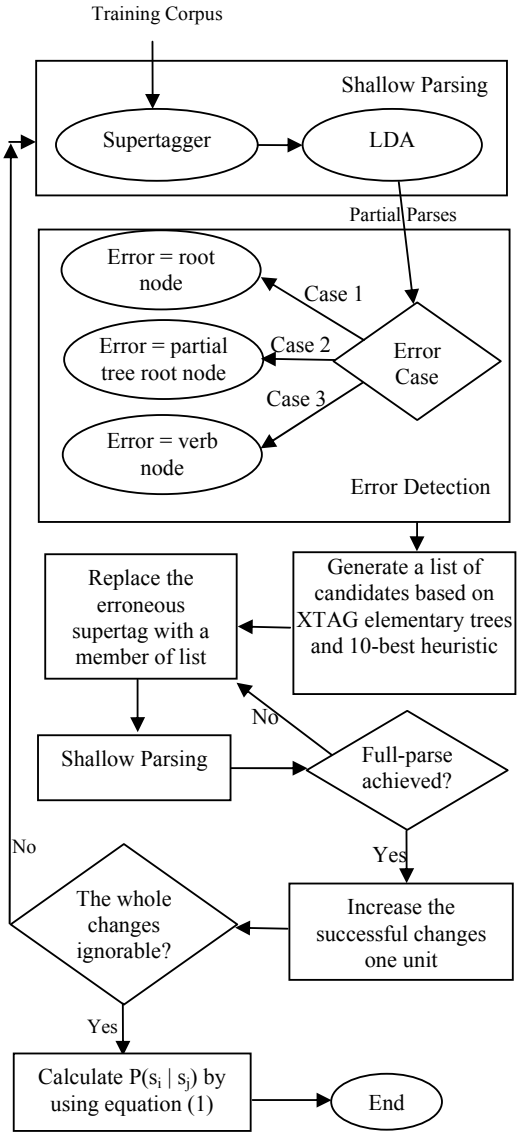

Figure 1. The whole process of calculating the probability model $P(s_i \mid s_j)$

5. Evaluation

In order to evaluate the proposed methods, 3000 sentences with their syntactic structure from Penn Treebank are selected as test corpus. These sentences are completely different from those that are used in the process of calculating the changing probabilities.

We divided the test corpus into three different categories based on the sentence length: the sentences shorter than 16 words, sentences with length between 16 and 25 words and sentences longer than 25 words[1].

The experiments include the evaluation of mentioned heuristics such as 1-best, 10-best, 25-best, XTAG based and trained probability model heuristics. In each experiment, the percentage of full-parsed sentences and parsing time are computed. Also, in order to evaluate the resulting full-parse quality, PARSEVAL metrics, introduced by [3], are calculated. We measure PARSEVAL metric only for those sentences which have been fully-parsed successfully.

Figure 2, 3 and 4 show the results of these experiments on each of the mentioned category. The evaluations show that considerable improvements both in time and percentage of full-parsed sentences are achieved by using the trained probability model heuristic. This method increases the full-parse rate from the native supertagger (1-best heuristic) by a factor of 3 in the first category, by a factor of 11 in the second category and by the factor of 21 in the third category. That is, the effects of the trained probability model in long sentences are more than short sentences.

Comparing the mentioned figures, shows that by increasing the sentence length, the percentage of full-parsing rate and parsing speed decreases dramatically. Also, in the trained probability model heuristic, the parsing speed increases about twice than XTAG-based heuristic, while the full-parsing rate also increases about 20%.

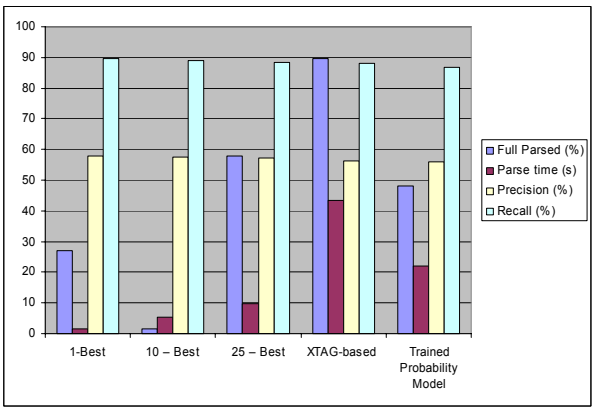

Figure 2: Experimental results on sentences shorter than 16 words

[1] Maximum length of selected sentences is bounded on 45 words.

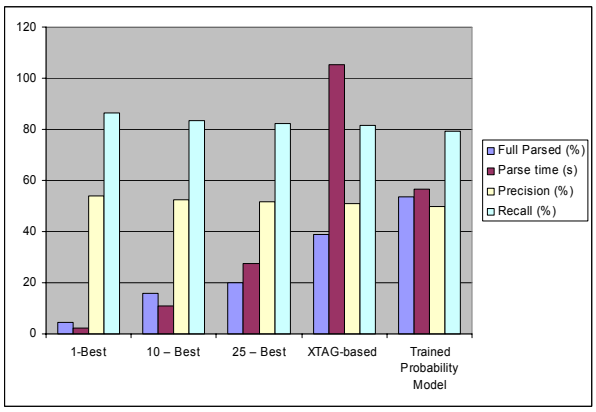

Figure 3: Experimental results on sentences between 16 and 25 words

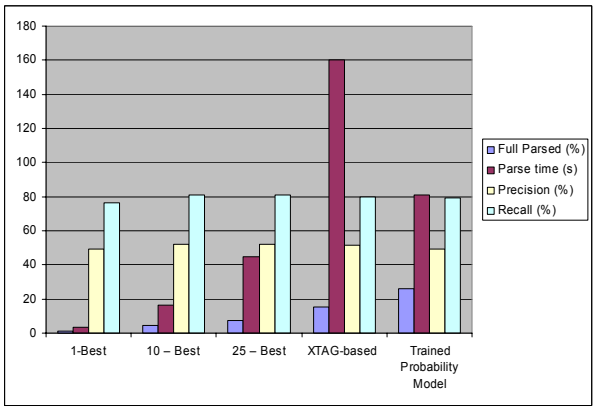

Figure 4: Experimental results on sentences longer than 25 words

6. Conclusion

Parsing is the one of the most important phases in many natural language applications, like information retrieval and rule-based machine translation systems, where it needs full-syntactic analysis for the input sentence. Although using more enriched grammar model, like TAG, is preferred because of its power in the descriptive model, but this kind of formalism lacks both in parsing speed and accuracy.

To overcome these problems, we've taken the benefits of speed and accuracy of a shallow parsing algorithm named supertagger. We introduced several heuristics which get the partial parses as the input and generate the full-parse structure of the sentence.

Several experiments on different data set selected from Penn Treebank show that by using error detection heuristics with a trained probability model to propose correcting candidates, the full-parsing rate as well as parsing speed have been improved significantly.

7. References

[1] Bangalore S., Complexity of Lexical Descriptions and its Relevance to Partial Parsing, PhD thesis, Department of Computer and Information Sciences, University of Pennsylvania, 1997.

[2] Bangalore S. and Joshi A., Supertagging: An approach to almost parsing, Computational Linguistics, 25(2), pp. 237–266, 1999.

[3] Black, E., Abnery, S., Flickinger, D. and et al., A procedure for quantitatively comparing the syntactic coverage of English grammars, DARPA Speech and Natural Language Workshop, pp. 306-311, 1991.

[4] Clark S., Hockenmaier J., and Steedman M., Building deep dependency structures with a wide coverage CCG parser. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Philadelphia, Pennsylvania, pp. 327–334, July 2002.

[5] Collins, M., Three generative, lexicalized models for statistical parsing, In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, Madrid, Spain, July 1997.

[6] Faili, H. and Ghassem-Sani, G., An Application of Lexicalized Grammars in English-Persian Translation, Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), Universidad Politecnica de Valencia, Spain, pp. 596-600, 2004.

[7] Hockenmaier J. and Steedman M., Generative models for statistical parsing with combinatory categorial grammar, In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Philadelphia, Pennsylvania, pp. 335–342, July 2002.

[8] Nasr A., and Rambow O., supertagging and full parsing, In Proceedings of the Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7), Vancouver, BC, Canada, 2004.

[9] XTAG research group, A Lexicalized Tree Adjoining Grammar for English, Technical Report IRCS 98-18, Institute for Research in Cognitive Science, University of Pennsylvania, pp. 5-10, 1998.

[10] Van Noord G., Head-corner parsing for TAG, In Computational Intelligence, 10(4), pp. 525–534, 1994.