

A modular cascaded approach to complete parsing

Samar Husain, Phani Gadde, Bharat Ambati, Dipti Misra Sharma and Rajeev Sangal.

Language Technologies Research Centre, IIT-Hyderabad, India

{phani_gadde, ambati}@students.iit.ac.in, {samar, dipti, sangal}@mail.iit.ac.in

Abstract

In this paper, we propose a modular cascaded approach to data driven dependency parsing. Each module or layer leading to the complete parse produces a linguistically valid partial parse. We do this by introducing an artificial root node in the dependency structure of a sentence and by catering to distinct dependency label sets that reflect the function of the set internal labels vis-à-vis a distinct and identifiable linguistic unit, at different layers. The linguistic unit in our approach is a clause. Output (partial parse) from each layer can be accessed independently. We applied this approach to Hindi, a morphologically rich free word order language using MST Parser. We did all our experiments on a part of Hyderabad Dependency Treebank. The final results show an increase of 1.35% in unlabeled attachment and 1.36% in labeled attachment accuracies over state-of-the-art data driven Hindi parser.

1. Introduction

Partial analysis of a sentence, where small linguistic units are identified and their significance fleshed out, has proven to be a successful strategy for numerous NLP tasks. It helps to take a step forward towards relevant complete analysis. The task of POS tagging and chunking [1], [2], [10] can, in that sense, be viewed as a step towards complete parsing. Indeed, the combined task of POS tagging, morph analysis and chunking is commonly called shallow parsing. Shallow parsing is robust and generally gives high speed performance and helps in making the task of complete parsing easier. Moreover, for numerous applications like, speech recognition, dialog systems, information extraction, machine translation etc. partial analysis comes in handy [3], [11], [12]. Parsing free word order languages (such a Czech, Hindi, Turkish, etc.) has, in spite of the recent availability of annotated data, proven to be a challenging task. Parsing experiments [22] in these languages have not reached the performance obtained for English. Various reasons have been stated for the low performance, among them, (a) Small corpus size, (b) complex linguistic

phenomenon, and long distance dependencies, and (c) non-projective structures, are the most frequently stated [21], [22], [7]. The approach proposed in this paper tries to minimize the adverse effect of (b) and (c). For almost all such free word order languages dependency analysis and thereby dependency parsing has been preferred over phrase structure. It is well established that dependency framework is better suited for such languages [6], [16], [20], [22]. Dependency parsing is basically identifying the relations between different words in a sentence. The final analysis is a tree with sentential words forming the nodes of this tree. Due to the availability of annotated corpora in recent years, data driven dependency parsing has achieved considerable success [21].

In this paper, we propose a modular cascaded approach to data driven dependency parsing. Each module or layer leading to the complete parse produces a linguistically valid partial parse. We do this by introducing an artificial root node in the dependency structure of a sentence and by catering to distinct dependency label sets that reflect the function of the set internal labels vis-à-vis a distinct and identifiable linguistic unit, at different layers. The linguistic unit in our approach is a clause¹. In effect, we separate out intra-clausal dependencies from inter-clausal and handle them separately at different layers. Output (partial parse) from each layer can be accessed independently. We applied this approach to Hindi², a morphologically rich free word order (MoR-FWO) language using MST Parser [19]. We did all our experiments on a part of Hyderabad Dependency Treebank (HyDT) [4]. The final results show an increase of 1.35% in unlabeled attachment and 1.36% in labeled attachment accuracies over state-of-the-art data driven Hindi parser [7]. A similar approach has

¹A clause is a group of word such that the group contains a single finite verb chunk.

²Hindi is a verb final language with free word order and a rich case marking system. It is one of the official languages of India, and is spoken by ~800 million people.

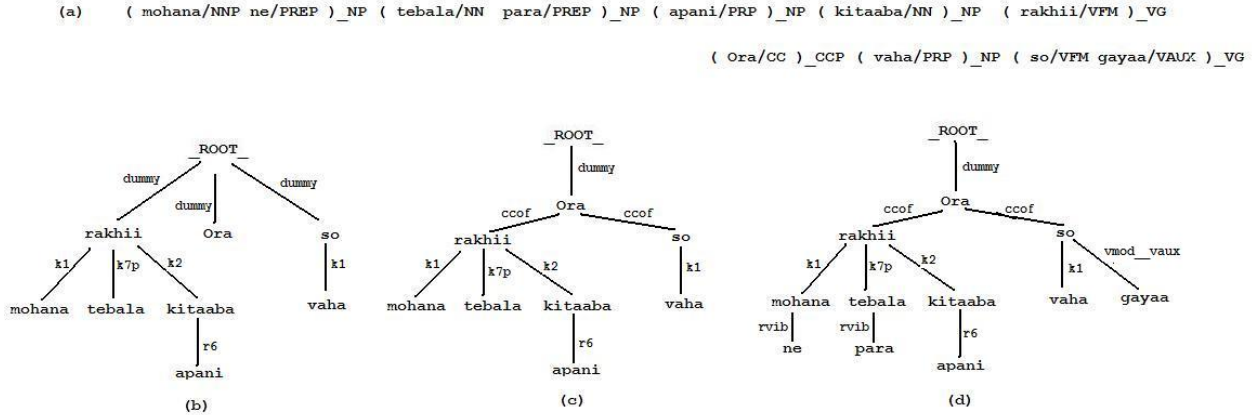


Figure 1: Analysis of example 1 by various layers. (a) POSCH, (b) IRCL, (c) INCL, and (d) IRCH.

been tried to parse Hindi using a grammar driven approach [8]. Constraint based grammar driven approaches have also been successfully tried for Hindi [5], [6], [8], [9].

The paper is arranged as follows. In Section 2 we give a detailed description of the proposed approach. Section 3 describes the experimental setup. In Section 4 we evaluate the approach and compare the results with the state-of-art data driven integrated³ parser. Section 5 gives some general remarks and observations on the experiments and their relevance. We conclude the paper in Section 6.

2. Approach

2.1 Background

Data driven parsing is usually a single stage process wherein a sentence is parsed at one go. Many attempts have, however, tried to divide the overall task into sub-task. One trend has been to first identify dependencies and then add edge labels over them [13]. The other trend has been towards performing smaller linguistically relevant tasks as a precursor to complete parsing [1], [2], [3], [6], [23]. The task of POS tagging and chunking was introduced accordingly. We know that there are tradeoffs in both these trends when compared to ‘integrated’ parsing. In this paper we have tried to address such issues and we argue that many problems faced during integrated parsing can be overcome.

In our approach we divide the task of parsing into the following sub-tasks (layers):

- (a) POS tagging, chunking (POSCH),
- (b) Intra-clausal parsing (IRCL),
- (c) Inter-clausal parsing (INCL),

(d) Intra-chunk dependencies (IRCH) identification

(a) POSCH is treated as pre-processing to the task of parsing. A bag represents a set of adjacent words which are in dependency relations with each other, and are connected to the rest of the words by a single incoming dependency arc from or to its root node. Thus a bag is an unexpanded dependency tree connected to the rest only by means of its root. A noun phrase or a noun group chunk is a bag in which there are no verbs, and vice versa for verb chunks. The relations among the words in a chunk are not marked and hence allow us to ignore local details while building the sentence level dependency tree. In general, all the nominal inflections, nominal modifications (adjective modifying a noun, etc.) are treated as part of a noun chunk, similarly, verbal inflections, auxiliaries are treated as part of the verb chunk [10].

(b) IRCL identifies only specific dependencies and marks the appropriate labels. This layer, in particular, handles the argument structure of the verb, noun-noun genitive relation, infinitive-verb relation, infinitive-noun relation, adjective-noun, adverb-verb relations, nominal coordination, etc.

(c) INCL then tries to handle more complex relations such as conjuncts, relative clause, etc. Basically, all the relations that function as relating two clauses are handled here. Hence, subordinating conjuncts, coordinating conjuncts and relative clauses are handled at this layer.

(d) IRCH dependencies are finally identified as a post-processing step to (b) and (c). Once this is done, the chunks can be removed and we can get the complete dependency tree. We will not discuss IRCH in this paper.

³Integrated here means a single pass. The parser identifies all the dependencies and the edge labels at on go.

In the partial parses (dependency trees) of (b) and (c), each node is a chunk and the edge represents the relations between the connected nodes labeled with suitable relations⁴. After removing the chunks in (d) each node is a lexical item of the sentence. Figure 1, shows the four steps for example 1.

Eg. 1: *mohana ne tebala para apani kitaaba*
 'Mohan' 'ERG' 'table' 'on' 'his' 'book'
rakhii Ora vaha so gayaa
 'kept' 'and' 'he' 'sleep' 'PRFT'
 'Mohan placed his book on the table and he slept'

From (a) to (d) in Figure 1, outputs of each of the previously discussed layers have been shown. Note that one can use any of these outputs independently.

We usually measure the accuracy of parsing in terms of unlabeled attachment accuracy (UA), labeled attachment accuracy (LA) and labeled accuracy (L) which follows from the above two tasks. High LA in Hindi is a harder task compared to UA due to reasons specified in previous parsing experiments [7]. Mainly, (a) Small corpus size, (b) complex linguistic phenomenon, and long distance dependencies, and (c) Non-projective structures, have been pointed as impediment to high performance. The fact that the UA of Hindi, a MoR-FWO language, is high close to 90% and LA is still low near 70% demonstrates (b) and (c) above. The proposed approach tries to minimize the damage done by the above reasons.

2.2 Architecture

The parser tries to analyze the given input sentence, which has already been tagged and chunked, in two stages; it first tries to extract intra-clausal dependency relations. These relations generally correspond to the argument structure of the verb, noun-noun genitive relation, infinitive-verb relation, infinitive-noun relation, adjective-noun, adverb-verb relations, etc. In the second stage it then tries to handle more complex relations such as conjuncts, relative clause, etc. What this essentially means is a two-stage resolution of dependencies, where the parser selectively resolves the dependencies of various lexical heads at their appropriate stage, for example verbs in the 1st stage and conjuncts and inter-verb relations in the 2nd stage. The key ideas of the proposed layered architecture are: (a)

There are two layers (stages), (b) The 1st stage handles intra-clausal relations, and the 2nd stage handles inter-clausal relations, (c) The output of each layer is a linguistically valid partial parse that becomes, if necessary, the input to the next layer, and (d) The output of the final layer is the desired full parse.

Note that by following the above approach we are able to get 4-fold advantage, (1) Each layer in effect does linguistically valid partial parsing, (2) by dividing the labels into different functional sets (intra-clausal and inter-clausal) we localize the dependencies that need to be identified, (3) by attacking the problem in a modular way, i.e. handling only clauses at 1st stage, we tremendously reduce non-projective structures, and (4) such a cascaded layered approach reduces the overall complexity. It helps in machine learning because different sets of features (though overlapping) may play a greater role in getting high accuracy. We elaborate on these points in Section 5.

The 1st stage output for example 1 is shown in figure 1 (b). In figure 1 (b), the parsed clause subtrees '*mohana ne tebala para apani kitaaba rakhii*' and '*vaha so gayaa*' are attached to `_ROOT_`. The coordinating conjunct '*Ora*' is also seen attached to the `_ROOT_`. By introducing `_ROOT_` we are able to attach all unprocessed node to it. The dependency tree thus obtained in the 1st stage is partial, but linguistically sound⁵. `_ROOT_` ensures that the parse we get after each stage is connected. Later in the 2nd stage the relationship between various clauses are identified. The 2nd stage parse for the above sentences is also shown in figure 1 (c). Note that by doing so; we have easily divided the entire dependency tags into two sets, one that is relevant for intra-clausal dependencies, and two, which show inter-clausal dependencies. We have also, therefore, localized the dependencies that the parser needs to identify at each stage.

3. Experimental Setup

3.1 Parser

Since we compare our results with [7], we use MST⁶ parser [19]. MST has an implementation of Chu-Lui-Edmonds MST algorithm [14], [15]. It uses online large margin learning as the learning algorithm [18]. Unless otherwise specified, we have trained the annotated data with the same parameter setting and feature set as described by [7].

⁴All the relations marked by the parser are syntactico-semantic labels. For a detailed analysis see [6]. Many relations shown in the diagrams of this paper are described in [4]. For the complete tagset description, see <http://ltrc.iit.ac.in/MachineTrans/publications/technicalReports/tr032/treebank.pdf>

⁵Interestingly, it is easy to see that, if required, 1st stage parser can independently be used only as a clause identifier.

⁶MST version 0.4b

We use F3⁷ for FEATS, conjoined feature set and k=5.

3.2 Dataset

All the experiments have been performed on a subset of the HyDT [4] for Hindi. The annotated labels on the arcs of a dependency tree are syntactico-semantic. The subset contains around 1504 sentences with an average length of 12.18 words. The total training, development and testing sentences were 1200, 100 and 204 respectively.

3.2.1 Stage1 data

Since 1st stage aims to parse only clauses, the original treebank needs to be modified to prepare the training data. We introduce a special dummy node named `_ROOT_` which becomes the head of the sentence. All the clauses are connected to this dummy node with a *dummy* relation. In effect we remove all the inter-clausal relations. The steps to do this conversion are:

- Add a dummy `_ROOT_` node to the gold standard tree.
- Find all sub-trees that have *ccof*⁸ or *nmod_relc*⁹ relation with its parent.
- Attach those sub-trees and the respective parents to the new node `_ROOT_`, with a *dummy* relation

At the end of the conversion, the parses in the treebank are converted to ‘1st stage parses’, i.e., parse trees which one would get at the end of 1st stage of parser. Note that a ‘*ccof*’ relation can represent either coordinating conjunct or a subordinating. Other types of coordination like nominal, infinitival, adjectival, etc. are all handled in the 1st stage itself. This is understandable since coordinations are intra-clausal. The settings while training MST for stage 1 have already been discussed in Section 3.1. We’d like to point that there are very few non-projective structures in the 1st stage parsing. Indeed, [17] have shown that intra-clausal non-projective structures in HyDT are less. Section 5 fleshes out this point.

3.2.2 Stage2 data

Stage 2 handles all the relations left out in 1st stage. As mentioned earlier, this amounts to handling only the inter-clausal relations. The partial parse obtained from 1st stage becomes the input to the 2nd stage. Hence, it handles only the *dummy* relations obtained in

the 1st stage parse. The training data for 2nd stage is obtained by converting the gold parse to suit 2nd stage needs. The conversion is also required so that we don’t reparse 1st stage clausal sub-trees.

Since we know that after 1st stage all the parsed clauses are attached to `_ROOT_` (Figure 2), we replace those sub-trees with their respective heads. We learn the relations between them in the 2nd stage. This is clearly depicted in Figure 3. The head of each sub-tree is a place holder for the whole clause we parsed in 1st stage. Though we are only taking the head of each sub-tree, we provide the features that are characteristic of the sub-tree, thereby reflecting

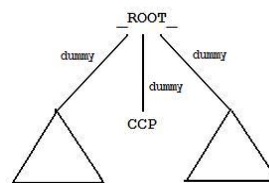


Figure 2: Stage1 training input

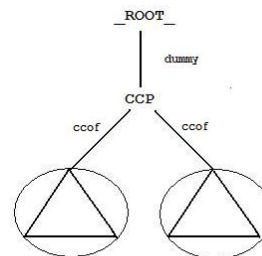


Figure 3: Stage2 training input

the whole sub-tree using that single node. This helps us to make the parser learn only the inter-clausal relations.

Note here that we make an assumption that the inter-clausal relations exist only between the heads (in most cases, a finite verb) of various clauses or in case of multiple conjuncts between conjuncts themselves. There are exceptions to this. We’ll have more to say about it in Section 6. Also note that all the relations obtained via Stage 1 are not modified in the 2nd Stage.

Taking the head of a 1st stage parsed sub-tree as a single node that is representative of this entire sub-tree requires using extra features while training for 2nd stage. To do this, one must judiciously isolate properties of the root of this sub-tree as well as its internal nodes properties. The features can be broadly summarized as follows: (a) *Structural features of the sub-tree*, such as width, depth, branching, total no. of nodes, etc., and (b) *The characteristics of the nodes*

⁷F3 in [8] uses TAM (tense, aspect and modality) class for verbs, or postposition for nouns in the FEATS column of the CoNLL format.

⁸Arc label signifying conjunct relation. Only conjuncts connecting finite verbs are valid finds for the conversion process.

⁹Arc label signifying relative clause

(including the root) such as morph features, POS/Chunk tags, arc label, domination (parent, grandparent, great-grandparent) feature, sibling features, valency, etc.

For the present set of experiments we decided to use the same features¹⁰ as that of integrated parsing. This has been done so as to make the comparison between the performances of 2-stage parsing and integrated parsing unbiased.

4. Evaluation

The two-stage parser was evaluated on 192 test sentences. For testing both the stages, the gold trees were used. The results in Table 1 clearly show that the complete parse from the layered approach gives better results than integrated parsing. We see that there is a rise of 1.35% in UA (unlabeled attachment) and 1.36% in LA (labeled attachment) over state-of-the-art data driven integrated parser. The final parse for the layered approach is obtained by combining the partial parses from the 2 stages. It is worth stressing here that the rise over integrated parsing is not significant because we have used the same features as that of integrated parsing. This was done to make the comparison unbiased. As mentioned earlier there are many important layer specific features that still needs to be tried and that can improve the performance further.

Table 1: Accuracies of both stage together

Details	Accuracy	
Full (Stage1 + Stage 2)	<i>LA</i>	72.66
	<i>UA</i>	90.80
	<i>L</i>	74.45
Integrated	<i>LA</i>	71.30
	<i>UA</i>	89.45
	<i>L</i>	73.49

5. Discussion and general observations

As seen in the results, the overall accuracy of Hindi parsing increases by 1.35% for UA and 1.36% for LA. Note here that the increase in the performance is not very large; this is because the features used in the 2nd stage are still same as that of integrated parsing. This was done to make the comparison unbiased. Nevertheless, by using the same features we get some increase in the overall performance. Further analysis

shed some light on the reasons for the increase in the overall performance.

The first clear reason was the reduction of non-projective structures. There are only few left in intra-clausal parsing. This is a direct consequence of breaking up the task into stages. We know that 1st stage only caters to localized dependencies and the search space of the parser is a clause, this holds true for non-projective sentences as well. [7] reports non-projectivity as the major reason for low performance in Hindi parsing. This indeed is true; our approach shows that there are ways in which one can minimize the problems posed by non-projectivity.

The second important reason was the reduction in the error of the tags. This also is a clear consequence of handling intra-clausal and inter-clausal tags separately. In particular we found that the labels handled by 2nd stage have a precision and recall of 94.36% and 92.63% respectively for LA, compared to 89.34% and 89.84% for the same labels during integrated parsing. We know that all the attachments identified during 2nd stage are long distance and it is difficult to identify such long distance dependencies accurately during integrated parsing. However, by reducing the sub-tree as a single node in the 2nd stage, the 2nd stage focuses on the learning of such long distance relationships.

Though there is a significant increase in the accuracy of parsing, the overall accuracy is still not very high. We believe this is due to very small corpus available for Hindi. Another reason is lack of specific learning cues. For example to identify the subject or object in a clause, unlike English, position doesn't always provide clinching cue for subject/object identification. Hindi on the other hand does have a rich inventory of postpositions/case-markers that can help identify many verbal arguments including subject and object. But this also is not always useful, as a single clause might have multiple nouns that are unmarked (i.e. have \emptyset postposition). Subject/object agreement, however, does provide a vital cue. However, [7] has shown that automatic learning of agreement in Hindi is very difficult. We also found it to be true. [7] has shown that in many such cases minimal semantics can help the parser significantly. We plan to explore this direction in the future.

Our approach show how by partial parsing, the adverse effects of (a) non-projective structures, (b) complex linguistic phenomenon, and (c) long distance dependencies, can be minimized considerably. The results thus obtained are consistently better than that of integrated parsing.

¹⁰This will include POS/Chunk tags, morph features, suffix info., etc.

6. Conclusion and future directions

In this paper we described a modular cascaded approach to complete Hindi dependency parsing. Each such layer produces an independent partial parse that becomes the input to the next layer. All the partial parses are linguistically valid parses. We show how by separately catering to only intra-clausal and inter-clausal label the dependencies can be made local and subsequently the search space reduced for the parser. We show how by introducing a dummy root node in each stage we can easily get partial parses.

The evaluation showed that the approach performs better than integrated parsing. There was 1.35% increase in UA and 1.36% in LA over the state-of-the-art data driven Hindi parsing. The analysis shows that the main reason for this is reduction of non-projective structures and increase in the correct identification of dependency tags due to removal of long distance dependencies. This we explained was a direct consequence of modular parsing.

There are still a few relations that need to be incorporated into the scheme. Present 2nd stage parsing assumes that all the inter-clausal relation exists between clausal heads. We know that this is not the case for relative clause construction where the relative clause sub-tree modifies a tree internal node of the matrix clause. Also, the performance of individual stages can be further increased by using layer-specific features, the present set of features have been chosen so as to make the comparison with integrated parsing unbiased. We saw how for the 2nd stage many potential features have not been explored to make the comparison unbiased. We plan to address these issues in the future work.

7. References

- [1] S. Abney. Part-of-Speech Tagging and Partial Parsing. In K. Church, S. Young, and G. Bloothoof, editors, *Corpus-Based Methods in Language and Speech*. Kluwer Academic Publishers, 1996.
- [2] S. Abney. Partial Parsing via Finite-State Cascades. *Natural Language Engineering*, 2(4):337–344, 1997.
- [3] G. Attardi and F. Dell’Orletta. Chunking and Dependency Parsing. *LREC Workshop on Partial Parsing: Between Chunking and Deep Parsing*. Marrakech, Morocco, 2008.
- [4] R. Begum, S. Husain, A. Dhawaj, D. Sharma, L. Bai, and R. Sangal. Dependency annotation scheme for Indian languages. In *Proceedings of IJCNLP*. 2008.
- [5] A. Bharati and R. Sangal. Parsing Free Word Order Languages in the Paninian Framework. *Proc. of ACL*. 1993.
- [6] A. Bharati, V. Chaitanya and R. Sangal. *Natural Language Processing: A Paninian Perspective*, Prentice-Hall of India, New Delhi. 1995.
- [7] A. Bharati, S. Husain, B. Ambati, S. Jain, D. Sharma, and R. Sangal. Two semantic features make all the difference in parsing accuracy. In *Proc. of 6th ICON, Pune, India*. 2008.
- [8] A. Bharati, S. Husain, D. Sharma, and R. Sangal. A two-stage constraint based dependency parser for free word order languages. In *Proc. of COLIPS IALP. Thailand*. 2008.
- [9] A. Bharati, R. Sangal, and T. P. Reddy. A Constraint Based Parser Using Integer Programming, In *Proc. of ICON*, 2002.
- [10] A. Bharati, D. M. Sharma, L. Bai and R. Sangal. AnnCorra: Annotating Corpora Guidelines for POS and Chunk Annotation for Indian Languages. *LTRC-TR31*. 2006.
- [11] A. Buczyński and A. Wawer. Shallow parsing in sentiment analysis of product reviews. *LREC Workshop on Partial Parsing: Between Chunking and Deep Parsing*. Marrakech, Morocco. 2008.
- [12] X. Carreras and L. M’arquez. Introduction to the CoNLL-2005 Shared Task. In *Proceedings of CoNLL*. 2005.
- [13] W. Chen, Y. Zhang and H. Isahara. A Two-Stage Parser for Multilingual Dependency Parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*. 2007.
- [14] Y.J. Chu and T.H. Liu. On the shortest arborecence of a directed graph. *Science Sinica*, 14:1396–1400. 1965.
- [15] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240. 1967.
- [16] R. Hudson. *Word Grammar*, Basil Blackwell, 108 Cowley Rd, Oxford, OX4 1JF, England. 1984.
- [17] P. Mannem and H. Chaudhry. Insights into Non-projectivity in Hindi. In *ACL-IJCNLP Student paper workshop*. 2009.
- [18] R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *Proceedings of ACL*. pp. 91–98. 2005.
- [19] R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. Non-projective dependency parsing using spanning tree algorithms. *Proc. of HLT/EMNLP*, pp. 523–530. 2005.
- [20] I. A. Mel’čuk. *Dependency Syntax: Theory and Practice*, State University, Press of New York. 1988.
- [21] J. Nivre, Dependency Grammar and Dependency Parsing. *MSI report 05133*. Växjö University: School of Mathematics and Systems Engineering. 2005.
- [22] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov and E. Marsi. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2), 95–135. 2007.
- [22] S. M. Shieber. Evidence against the context-freeness of natural language. In *Linguistics and Philosophy*, p. 8, 334–343. 1985.
- [23] P.Li Shiuan and C. Ting Hian Ann. A Divide-and-Conquer Strategy for Parsing. In *Proc. of IWPT*. 1996.