Are Phrase Structured Grammars Useful in Statistical Parsing?

Aboy WONG and Dekai WU
HKUST
Human Language Technology Center
Department of Computer Science
University of Science and Technology
Clear Water Bay, Hong Kong
{aboy,dekai}@cs.ust.hk

June 1, 1999

Abstract

In this paper, we argued: 1) To parse accurately, a grammar is not necessary. 2) It is possible to parse deterministically by not conforming to an explicit grammar. We support the above claims by presenting our parser, which is lightweight, grammar-less, deterministic and have the highest accuracy among tag based parsers. The speed of our parser is more than 500 words per CPU second and only 6M of memory is needed for loading the parsing model. In our architecture, the grammatical information is captured by the parsing model. Our parsing model differs from others in that, extra information about how to group constituents are provided. Thus an explicit grammar is not needed in our algorithm.

Submission Type: Regular paper

Keywords: deterministic parsing, machine learning, phrase structured grammar, shift-reduce parser, statistical parsing

Word Count: 2972

Are Phrase Structured Grammars Useful in Statistical Parsing?

Abstract

In this paper, we argued: 1) To parse accurately, a grammar is not necessary. 2) It is possible to parse deterministically by not conforming to an explicit grammar. We support the above claims by presenting our parser, which is lightweight, grammar-less, deterministic and have the highest accuracy among tag based parsers. The speed of our parser is more than 500 words per CPU second and only 6M of memory is needed for loading the parsing model. In our architecture, the grammatical information is captured by the parsing model. Our parsing model differs from others in that, extra information about how to group constituents are provided. Thus an explicit grammar is not needed in our algorithm.

1 Introduction

In the history of parsing, most parsers have been based on phrase structured grammars.

In parsing, a grammar defines the grammaticality of the corresponding language. It tells what can be grouped to form constituents and thus, is a mean to construct the parse structure. The role of a grammar is so important that, conforming to the grammar is almost taken as a norm in parser designs. It is true that, with a set of rules defining what can be grouped to form constituents, one can avoid grouping in an arbitrary manner and make efficient $O(n^3)$ parsing algorithm possible. However, we also believe that conforming to a grammar is one of the reasons inhibiting the development of fast deterministic parsing algorithm.

The intriguing possibility of very efficient, human-like parsing by deterministic models was hypothesized by Marcus (Marcus, 1980). Yet to date, almost all high performance natural language parsers today still employ the same nondeterministic search paradigms (dynamic programming, memorization, backtracking) as non-statistical parsers. Although, improvement is parsing speed has been achieved by the employment of pruning strategies like thresholding and beam searching, the algorithm is still non-deterministic in nature.

In recently years, a number of statistical parsers that do not employ any explicit grammar are proposed. They are: SPATTER (Magerman, 1994), Collins' parsers (Collins, 1996; Collins, 1997) and CONTEXT (Hermjakob, 1997). All of them are state-of-the-art parsers. This motivate us to re-evaluate the importance of a grammar in statistical parsing.

Below is the issues that we want to discuss in this paper:

1. **Is grammar a must?** The notion of grammar has a long history. Apart from the computer field, there is a lot of grammar studies in the linguistic domain. We do not intend to deny the existence of grammaticality in natural languages, but want to argue about the way to capture/express the grammaticality of a languages in a parsing task. In a grammar based parser, the grammaticality is captured partly by the grammar. However, due to the ambiguous nature of natural language grammar, conforming to the grammar doesn't guarantee a correct parse. A parsing model is usually needed to rank among all possible parses and select the "best" one as the final output. So the grammaticality is also partly captured by the parsing model. In most grammar based parsers, the parsing model does not provide any means to construct the parse, it only act as an evaluator to rank among different parse structures. This is why a grammar is still needed to provide the "rules" for grouping constituents. However, if the parsing model can provide the necessary information for constructing the parse structures, a grammar may not be needed any more.

2. **What do we lose without a grammar?** One of the drawback of not conforming to an explicit grammar is that the parse output may not be grammatical in respect to a grammar. However, if we take another point of view, "not being grammatical in respect to a grammar" is not a serious problem. Our argument is that, a grammar is itself an approximation of the corresponding natural language. To construct a grammar which exactly defines/describes a natural language is very difficult, if not impossible. A grammar either constructed by hand or induced automatically is often over-generating and incomplete. It is difficult to construct and proof a grammar being complete (cover all pos-

sible sentences), due to the existence of numerous exceptions/irregularities in natural languages. So, if conforming to a grammar doesn't guarantee a grammatical parse in respect to the corresponding language, "not being grammatical in respect to a grammar" should not be a big deal.

3. **Do we benefit by not employing any explicit grammar?** To construct a parse structure which conforms to a grammar, extensive matchings between production rules and intermediate substructures/input is almost a must. To avoid re-doing the matchings between the grammar and previously constructed substructures, a common approach is to store the incomplete and completed constituents in some data structures. In order to keep track of the intermediate substructures efficiently, most existing parsing algorithms employs dynamic programming approach. Other approaches like compiling a grammar into LR-type action table as in Tomita's Generalized LR Parser (Tomita, 1991), still results in a nondeterministic algorithm. This is again due to the ambiguous nature of natural language grammar. If we don't need to conform to an explicit grammar during parsing, we will have more freedom in the development of parsing algorithm. We do not neglect the possibility of a deterministic, grammar based parsing algorithm, but we believe employing a phrase structured grammar is one of the reasons that inhibits the use of fast deterministic parsing algorithm.

2 A Deterministic Parser

As we have mentioned above, an explicit grammar is not a must. We listed a few grammar-less parsers as examples, they are: SPATTER (Magerman, 1994), Collins' parsers (Collins, 1996; Collins, 1997) and CONTEXT (Hermjakob, 1997). However, we are not claiming only about the redundancy of a grammar, we also claim for the possibility of improvement in parsing accuracy and speed by discarding an explicit grammar.

Although the above parsers has higher accuracy than those based on explicit grammars. However, since they use extra features, like lexical identities and semantic information, the gain in accuracy may solely due to the improvement of the parsing model and is nothing related to whether an explicit grammar is used. Thus, they do not seems to be good examples to support our claims.

We support our claims by presenting a grammar-less, deterministic parser which out-performs other grammar based parsers in terms of speed and accuracy. The features used by our parser includes constituent labels and part-of-speech(POS) of input words only. These are similar to those being used by other grammar based parsers.

2.1 Base Parsing Model

The basis of our parser is a shift-reduce parser (Aho et al., 1986) consisting of a stack, an input stream and a decision control mechanism. The core part of our work is to learn the decision control mechanism, for which we employ a novel Shift/Reduce decision algorithm and a novel Constituent Labeling decision algorithm.

2.1.1 Child-Type Tagging

In ordinary shift-reduce parsers, a Reduce action refers to a specific production, and it groups constituents in a way that matches the right-hand-side of the production. However, we have no explicit productions in our system. Instead, a Reduce action chooses a group of constituents as determined by a tagging scheme inspired to some extent by (Magerman, 1994; Ramshaw and Marcus, 1995). Whenever a new constituent is pushed onto the stack, either by a Shift action or a Reduce action, a decision tree is used to tag it with one of four Child-Type values:

- UNARY, which means the constituent is the only child of its parent.
- LEFT, which means the constituent is the left-most child of its parent.
- RIGHT, which means the constituent is the right-most child of its parent.
- MID, which means the constituent is a middle child of its parent.

Assigning the constituent a Child-Type tag directly determines whether to Shift or Reduce. LEFT or MID signifies an incomplete constituent and therefore represents a Shift action. Conversely, RIGHT or UNARY represents a Reduce action where a single constituent is grouped in the case of a UNARY tag, or otherwise all constituents up to the top LEFT tag in the stack.

2.1.2 Constituent Labeling

Apart from making decisions on the constituent boundaries, we need to to assign a label to every constituent. In ordinary shift-reduce parsers, the label of the new constituent formed by a Reduce action is given by the left-hand-side of the production. Again, since we have no explicit productions, an alternative method is required. The solution is to introduce a second decision tree to predict the constituent label.

2.2 Improved Parsing Model

We improved the base parsing model by introducing a Base NP model and expanding the POS tag for prepositions.

2.2.1 Base NP Model

Base NP is a noun phrase that doesn't include any recursive NPs. Lots of works focusing on Base NP bracketing has been reported (Ramshaw and Marcus, 1995; Collins, 1996; Cardie and Pierce, 1998). It is commonly believed that finding Base NP before parsing can improve parsing accuracy.

In our NP model, the tag set used is similar to our Child-Type Tag system. The tags includes:

- UNARY, means the word itself is a Base NP
- LEFT, means the word is the left-most child of a Base NP
- RIGHT, means the word is the right-most child of a Base NP
- MID, means the word is a middle child of a Base NP
- OUT, means the word is outside of a Base NP

A separate decision tree is learned for this task.

2.2.2 Preposition Tag Expansion

Prepositional Phrase attachment is another area which receives lot of attentions (Brill and Resnik, 1994; Collins and Brooks, 1995; Ratnaparkhi, 1998; Yeh and Vilain, 1998). Lexical or semantic features are used to tackle the problem. However, in our model, we restrict ourselves to simple syntactic features in order to keep our model slim. So we anticipate a poor performance on prepositional phrase attachments.

To remedy that, we expands the current POS tag set as follows. Preposition that appears more than 100 times in the training data are extracted. There are totally 49 of them. We create a new POS tag for each of them, so at the end, we have 49 POS tags representing the 49 most frequent prepositions and a POS tag representing the rest of prepositions.

3 Experiments and Results

We use parsed sentences from sections 2–21 (about 40000 sentences) of the Penn Wall Street Journal corpus, release 2 (Marcus and Marcinkiewics, 1993), as the training data and section 23 (about 2400 sentences) as the testing data. The PARSEVAL measures (et al., 1991) are used as the evaluation criteria:

$$\textbf{LP} = \frac{\#correct\ constituents\ proposed}{\#constituents\ proposed}$$

$$\textbf{LR} = \frac{\#correct\ constituents\ proposed}{\#constituents\ in\ treebank\ parse}$$

CBs = #constituents which violate constituent boundaries with a constituent in the Treebank parse.

where LP is Labeled Precision, LR is Labeled Recall and CB is Crossing Bracket.

A constituent is correct if and only if it spans the same set of words (ignoring punctuation, i.e. all tokens tagged as commas, colons or quotes) and has the same label as a constituent in the Treebank parse.

m, n	#nodes in Decision Tree	error rate
3, 3	148k	6.274%
4, 3	160k	6.107%
4, 4	162k	6.092%
5, 3	170k	6.011%
5, 4	169k	6.005%

Table 1: Results of Child-Type tagging

3.1 Base Model:Child-Type Tagging Results

Below is the features that we used in training the Child-Type tagging module.

$s_1, ..., s_m, t_1, ..., t_m, c_1, c_2, i_1, ..., i_n$, where:

- s_1 represent the label of the constituent on the top of stack,
- s_k represent the label of k-th constituent counted from the top of the stack,
- t_k is the corresponding Child-Type tag of s_k,
- c_1 and c_2 is the first and last child label of s_1,
- i_k is the POS tag of the $k-th$ word in the input

We have run a couple of experiments by varying m and n. The training examples are extracted from section 2–21, and the testing examples are extracted from section 22. Table 1 shows the error rates of the Child-Type tagging results.

m, n	#nodes in Decision Tree	error rate
3, 2	24k	2.017%
4, 3	26k	2.027%

Table 2: Results of Constituent Labeling

Model	B	N	P
Lab. Precision	0.7926	0.7978	0.8146
Lab. Recall	0.8039	0.8108	0.8233
Precision	0.8284	0.8298	0.8455
Recall	0.8402	0.8433	0.8545
CB/Sent.	1.9030	1.87981	1.61671
Sent. w/0CB	0.4700	0.4812	0.5104
Sent. w/≤2CBs	0.7151	0.7220	0.7522
speed(words/s)	733.488	623.986	541.877

Table 3: Parsing results

3.2 Base Model:Constituent Labeling Results

In the Constituent Labeling module, we use the followings as features,
$s_2, ..., s_m, t_2, ...t_m, c_1, c_2, c_3, c_4, i_1, ...i_n$, where:

- s_k, t_k and i_k has the same meaning as in the Child-Type Tag model,
- c_1, c_2, c_3 and c_4 is the first, second, second last and last child of the top constituent in stack

Table 2 shows the labeling results with different values of m and n. Training data comes form section 2–21 and testing data comes from section 22 of the Penn Treebank.

3.3 Parsing Results

We use the best model from each of the above 2 modules to build our parser. Table 3 shows the parsing result of the Base Model(B), the Base Model with Base NP Model(N), and the Base Model with both Base NP Model and Preposition Tag Expansion(P).

The precision and recall differs from the Labeled precision and recall in that, a constituent is correct if the span is correct, the constituent label is not considered.

From the table, we observed a tiny improvement in accuracy by incorporating the Base NP model. However the parsing speed drops at the same time. By expanding the POS tag set for prepositions, the increase in accuracy is more significant. Around 2% increase in Labeled Precision and Labeled Recall is observed, but the speed drops further.

Model:	PC	PL	O	PL	O
Sent. Len.	2–12	2–12	2–12	2–40	2-40
Ave. Sent. Len.	8.7	8.3	8.4	21.6	21.7
Lab. Precision(%)		87.1	88.9	81.9	81.5
Lab. Recall(%)		85.2	91.8	79.5	82.3
Precision(%)	88.6	89.8	91.4	83.0	84.6
Recall(%)	91.7	90.7	94.3	80.7	85.5
Ave. CBs		0.27	0.09	1.99	1.59
Sent. w/0CBs(%)		84.5	92.5	41.5	51.8

PC=PCFG, PL=PLCG, 0=Our Model
Unavailable data are left as blank.

Table 4: A Comparison between Tag based parsers

4 Comparison to Other Parsers

We compare our result with 2 parsers. One is based on probabilistic context free grammar(PCFG) (Charniak, 1996) and the other is based on probabilistic left corner grammar(PLCG) (Manning and Carpenter, 1997). Our work differs from the above works, in that, our parsing algorithm is deterministic. There is no redundant sub-structures generated by our parser. Moreover, there is no explicit grammar in our system.

To the best of our knowledge, the accuracy of our parser is the highest among all tag based parsers evaluated on the Penn Treebank corpus. A comparison of the results is shown in table 4.

5 Conclusion

In this paper, we argued: 1) To parse accurately, a grammar is not necessary. 2) It is possible to parse deterministically by not conforming to an explicit grammar. We demonstrate this by presenting our parser, which is lightweight, grammar-less, deterministic and have the highest accuracy among tag based parsers. The speed of our parser is more than 500 words per CPU second and only 6M of memory is needed for loading the decision trees. In our architecture, the grammatical information is captured by the parsing model. Our parsing model differs from other parsing models in that, extra information about how to group constituents are provided. Thus an explicit grammar is not needed in our algorithm.

References

Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. 1986. *Compilers. Principles, Techniques, and Tools.* Addison-Wesley, Reading, Massachusetts.

Eric Brill and Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of The 15th International Conference on Computational Linguistics*, pages 1198–1204.

Claire Cardie and David Pierce. 1998. Error-driven pruning of treebank grammars for base noun phrase identification. In *Proceedings of COLING-ACL'98*, pages 218–224.

Eugene Charniak. 1996. Tree-bank grammars. In *Technical Report CS-96-02, Dept of Computer Science, Brown University.*

Michael Collins and James Brooks. 1995. Prepositional phrase attachment through a backed-off model. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 27–38.

Michael John Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191.

Michael John Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of 35th Annual Meeting of the Association for Computational Linguistics*, pages 17–23.

Black et al. 1991. A procedure for quantitatively comparing the syntatic coverage of english grammars. In *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop.*

Ulf Hermjakob. 1997. Learning parse and translation decisions from examples with rich context. In *Doctoral dissertation.*

David M. Magerman. 1994. Natural language parsing as statistical pattern recognition. In *Doctoral dissertation.*

Christopher D. Manning and Bob Carpenter. 1997. Probabilistic parsing using left corner language models. In *Proceedings of the Fifth International Workshop on Parsing Technologies, MIT, Boston MA, 1997.*

B. Santorini Marcus and M. Marcinkiewics. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

Mitchell P. Marcus. 1980. *A Theory of Syntactic Recognition for Natural Language.* MIT Press, Cambridge, Massachusetts.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *ACL Third Workshop on Very Large Corpora*, pages 82–94.

Adwait Ratnaparkhi. 1998. Statistical models for unsupervised prepositional phrase attachment. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING-ACL'98)*, pages 1079–1085.

Masaru Tomita. 1991. *Generalized LR Parsing.* Kluwer, Dordrecht.

Alexander S. Yeh and Marc B. Vilain. 1998. Some properties of preposition and subordinate conjunction attachments. In *Proceedings of COLING-ACL '98: 26th Annual Meeting of the Association for Computational Linguistics and 17th International conference on Computational Linguistics*, pages 1436–1442.