Archives of Control Sciences
Volume 15(LI), 2005
No. 3, *pages 251–261*

Across-Genres and Empirical Evaluation of State-of-the-Art Treebank-style Parsers

VASILE RUS and CHRISTIAN F. HEMPELMANN

This paper evaluates a series of freely available, state-of-the-art parsers on a standard benchmark as well as with respect to a set of data important to measure text cohesion in the context of learning technology. We outline advantages and disadvantages of existing parsing technologies and make recommendations. The performance reported uses traditional measures as well as novel dimensions for parsing evaluation to develop a gold standard for narrative and expository texts. To our knowledge this is the first attempt to evaluate parsers across genres.

Key words:

1. Introduction

The task of syntactic parsing is valuable to many language understanding applications including anaphora resolution, machine translation, question answering, and others. Syntactic parsing in its most general definition may be viewed as discovering the underlying syntactic structure of a sentence. The specificities include the types of elements and relations that are retrieved by the parsing process and the way in which they are represented. For example, Treebank-style parsers retrieve a bracketed form that encodes a hierarchical organization (tree) of smaller elements (called phrases), while Grammatical-Relations(GR)-style parsers explicitly output relations together with elements involved in the relation (**subj(John,walk)**).

The present parser evaluation was carried out as part of the Coh-Metrix project (CM; [10]) at the Institute for Intelligent Systems of University of Memphis. CM is a text-processing tool that provides new methods of automatically assessing text cohesion, readability, and difficulty. In its present v.1.1 few cohesion measures are based on parses, but its next incarnation v.2 will depend more heavily on hierarchical syntactic information.

We focus on Treebank-style parsers and adopt a constituent-based approach for evaluation. Since the parses are all derived in one way or another from the same data and

Tha Authors are with Institute for Intelligent Systems, Department of Computer Science, Department of Psychology The University of Memphis, Memphis, TN 38120, USA, e-mails: {vrus, chmplmnn}@memphis.edu

Received 21.10.2005.

generate similar, bracketed output. The major goal of this section is to consistently evaluate the freely available state-of-the-art parsers on a standard data set and across genre on data from learning technologies corpora. We report parsers' competiveness along an array of dimensions including performance, robustness, tagging facility, stability, and length of input they can handle.

This work will be extended with a study of misparses and mistags in their relation to measures planned for CM 2.0. These measures – causal, structural, coreferential, temporal, and spatial, as well as intentional and logical/additive cohesion, and in particular pronoun anaphora resolution – require complex algorithms operating on the cleanest possible sentence parse. A faulty parse will lead to a cascading error effect. Thus, the most apt parser needs to be chosen to provide parses of the desired quality.

1.1. Parser Types

While the purpose of this work is not to propose a taxonomy of all available parsers, we consider it necessary to offer a brief overview of the various parser characteristics. Parsers can be classified according to their general approach (hand-built-grammar-based versus statistical), the way rules in parses are handled (selective vs generative), the parsing algorithm they use (LR, chart parser, etc.), type of grammar (unification-based grammars, context-free grammars, lexicalized context-free grammars, etc.), the representation of the output (bracketed, list of relations, etc.), and the type of output itself (phrases vs grammatical relations). Of particular interest to our work are Treebank-style parsers, parsers producing an output conforming to the Penn Treebank (PTB) syntactic annotation guidelines. The project defined a tag set and bracketed form to represent syntactic trees that became a standard for parsers developed/trained on PTB. It also produced a treebank, a collection of hand-annotated text with syntactic information, of considerable size. Since building treebanks is extremely expensive major attempts to solve the parsing problem use PTB in a way or another, hence the name Treebank-style parsers.

Given the large number of dimensions according to which parsers can be distinguished, an evaluation framework that would provide both parser-specific (to understand the strength of different technologies) and parser-independent (to be able to compare different parsers) performance figures would be desirable. Accordingly, there are many studies in the literature about parsing techniques and different proposals circulate on how to evaluate the performance of parsers.

1.2. General Parser Evaluation Methods

Carroll et al. [3] broadly divide evaluation methods into non-corpus and corpus-based methods with the latter subdivided into unannotated and annotated corpus-based methods. The non-corpus method simply lists linguistic constructions covered by the parser's grammar. It is well-suited for hand-built grammars because during the construction phase the covered cases can be recorded. It has problems with capturing complexities occuring from the interaction of covered cases.

The most widely used corpus-based evaluation methods are: (1) the constituent-based (phrase structure) method and (2) the dependency/Grammatical-Relations(GR)-based method. The former has its roots in the Grammar Evaluation Interest Group (GEIG) scheme [11] developed to compare parsers with different underlying grammatical formalisms. It promoted the use of phrase-structure bracketed information and defined Precision, Recall and Crossing Brackets measures [1]. The GEIG measures were extended later to constituent information (bracketing info plus label) and have since become the standard for reporting automated syntactic parsing performance. Among the advantages of constituent-based evaluation are generality (less parser specific) and fine grain size of the measures. On the other hand, the measures of this method are weaker than the exact sentence measure [2], and it is not clear if they properly measure how well a parser identifies the true structure of a sentence. Lin [14] shows that many phrase boundary mismatches stem from differences between parsers/grammars and corpus annotation schemes. The reason is that usually treebanks (collections of manually annotated text with syntactic information) are constructed with respect to informal guidelines. Annotators often interpret them differently leading to a large number of different structural configurations.

The dependency-based and GR-based methods are closely related. The dependency-based method relies on a dependency-graph with arcs encoding certain dependencies. The GR-based method on the other hand relies on sets of parser-independent dependency relationships defined as a triplet (modifier, modifee, label) where the label is the type of relationship. Henceforth, the two are considered a single evaluation method. It can be applied indirectly to phrase structure analyses from parsers and treebanks by mapping phrasal structures into triplets. The mapping can be done at the cost of losing certain grammatical information.

In this paper we evaluate Treebank-style parsers which output phrase structures. An example of a sentence together with its parse tree and bracketed representation of the tree in Treebank-style format (shown below the sentence) is provided in Figure 1. On the right-hand side the constituents coresponding to the parse tree are listed. A constituent is a triplet of the form (Label, start index, end index) where *start* and *end index* are the position of the first and last word, respectively, in the sentence that are covered by the subtree rooted at node *Label*. The labels immediate above each word are *parts-of-speech* (the leaves of the tree). The upper nodes are labeled with *non-terminal* symbols corresponding to the major phrases: NP - Noun Phrase, VP - Verb Phrase, PP - Prepositional Phrase, and S - sentence.

[1]*Precision* is the ratio of correct constituents in the parse tree to the total number of constituents in the parse tree. *Recall* is the ratio of correct constituents in the parse tree to the total number of constituents in the gold standard parse tree (the correct or Treebank tree). *Crossing Brackets* is the percentage of sentences with no pairs of brackets crossing the gold standard bracketing (i.e. ((a b) c) has a crossing bracket measure of one if compared to (a (b c))) .

[2]*The exact sentence measure* is the percentage of sentences that are correctly parsed in their entirety. If one constituent is missing or is wrong, the sentence is not considered correct for the exact sentence measure, even though some constituents may be correct.

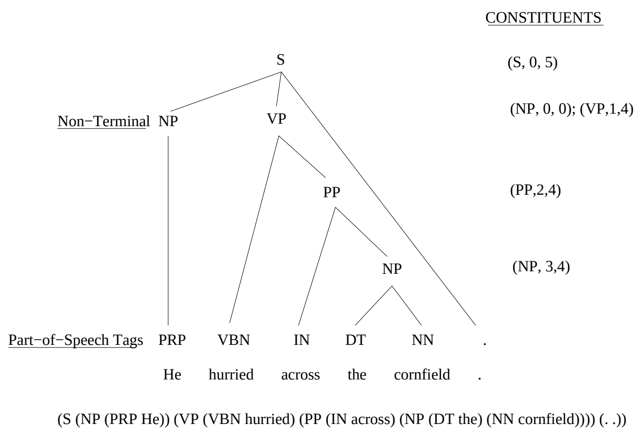
CONSTITUENTS
S
(S, 0, 5)
(NP, 0, 0); (VP,1,4)
Non−Terminal NP
VP
PP
(PP,2,4)
NP
(NP, 3,4)
Part−of−Speech Tags PRP VBN IN DT NN .
He hurried across the cornfield .
(S (NP (PRP He)) (VP (VBN hurried) (PP (IN across) (NP (DT the) (NN cornfield)))) (. .))

Figure 1. Example of sentence from Orlando text together with its parse tree, the bracketed representation of the parse tree and set of constituents.

There are two major approaches to evaluate Treebank-style parsers using the constituent-based method. First, there is the expert-only approach in which an expert looks at the output of a parser, counts errrors and reports different measures. The second approach is the more automated method, called the gold standard. It replaces the counting part of the former method with a software system that automatically compares the output of the parser with the gold standard, highly accurate, manually parsed data by human experts. The latter approach is more useful for scaling up evaluations to large collections of data while the expert-only approach is more flexible, allowing for evaluation of parsers from new perspectives.

We use the gold standard approach for evaluating the parsers. The evaluation is done from two different points of view. First, we offer a uniform evaluation for the parsers on section 23 of the Wall Street Journal section of PTB, the community norm to report parser performance. The goal of this first evaluation is to offer a good estimation of the parsers when evaluated on identical environments (same configuration parameters for the evaluator software). Besides the regular labelled precision (LP) and labelled recall (LR) measures, we also observe the following features which are extremely important for using the parsers in large scale text processing and to embed them as components in larger systems.

- **Self-tagging.** Whether or not the parser does tagging itself. It is advantageous to take in raw text since it eliminates the need for extra modules. Some versions of the parser we looked at in this paper do not have self-tagging facilities. In newer versions of the parsers the self-tagging feature is optional, the user having the opportunity to choose a better, external part-of-speech tagger.

- **Handles long sentences.** The ability of the parser to handle sentences longer than 40 words. The longer a sentence, the more computationally intensive the parsing process is. Some parsers simply do not handle sentences longer than 40 words.

- **Robustness.** Relates to the property of a parser to handle any type of input sentence and return a reasonable output for it and not an empty line or some other useless output. This is very important because larger systems that use parsing as a component would have cascading problems caused by the useless parser output. Useless output is worse than *incorrect but useable* output since it is very difficult to automatically detect. Moreover, incorrect parser output leads to incorrect overall output while useless output leads to unpredictable behavior.

- **Stability.** Whether or not the parser can process large collections of text without crashing or freezing.

- **Resource-needy.** If the parser requires large quantities of memory or it takes a long period of time to parse average sentences making it unusable in interactive systems. For instance, Collins' parser [7] takes a long time to initialize which makes it unusable in many applications.

Second, we evaluate the parsers on narrative and expository texts to study their performance on the two genres. The second evaluation exercise will provide input for learning technology projects. We use EVALB (developed by Satoshi Sekine and Michael Collins) to evaluate the bracketing performance of the output of a parser against a gold standard. The software evaluator reports numerous measures and we only report here the most important: labelled precision (LR), labelled recall (LR).

2. Evaluated Parsers

2.1. Apple Pie (AP)

Apple Pie [16] extracts a grammar from PTB v.2 in which S and NP are the only true non-terminals (the others are included into the right-hand side of S and NP rules). The rules extracted from the PTB have S or NP on the left-hand side and a flat structure on the right-hand side, for instance, $\mathbf{S} \rightarrow \mathbf{NP\ VBX\ JJ}$. Each such rule has the most common structure in the PTB associated with it, and if the parser uses the rule it will generate its corresponding structure. The parser is a chart parser and factors grammar rules with common prefixes to reduce the number of active nodes. Although the underlying model of the parser is simple, due to the large variety of linguistic constructs in the PTB it can't handle sentences longer than 40 words.

2.2. Charniak's Parser (CP)

Charniak in [4] presents a parser based on probabilities gathered from Penn Wall Street Journal PTB. It extracts the grammar and probabilities and, with a standard context-free chart-parsing mechanism, generates a set of possible parses for each sentence retaining the one with the highest probability (probabilities are not computed for all possible parses). The probabilities of an entire tree are computed bottom-up. In [5], he proposes a generative model based on a Markov-grammar. It uses a standard bottom-up, best-first probabilistic parser to first generate possible parses before they are ranked by the probabilistic model.

2.3. Collins's (Bikel's) Parser (CBP)

Collins's statistical parser [6] [7] is based on the probabilities between head-words in parse trees. It explicitly represents the parse probabilities in terms of basic syntactic relationships of these lexical heads. Collins defines a mapping from parse trees to sets of dependencies, on which he defines his statistical model. A set of rules defines a head-child for each node in the tree. The lexical head of the head-child of each node becomes the lexical head of the parent node. Associated with each node is a set of dependencies derived in the following way. For each non-head child, a dependency is added to the set where the dependency is identified by a triple consisting of the non-head-child non-terminal, the parent non-terminal, and the head-child non-terminal. The parser is a CYK-style dynamic programming chart parser.

2.4. Stanford Parser (SP)

The Stanford Parser that we use is the lexicalized version of the parser developed by Dan Klein and Chris Manning [13]. Klein and Manning also developed an unlexicalized version of the parser which rivals state-of-the-art lexicalized parsers. It uses a context free grammar with state splits. The parsing algorithm is simpler, the grammar smaller and fewer parameters are needed for the estimation. It uses a CYK chart parser which exahustively generates all possible parses for a sentence before it selects the highest probability tree. We did not use the unlexicalized version since we had trouble getting to it.

3. Experiments and Results

3.1. Text Corpus

We performed experiments on two data sets. First, we chose the norm for large scale parser evaluation which is section 23 of Wall Street Journal part of PTB. Section 23 has 2416 sentences. Since parsers have different parameters that can be tuned leading to (slightly) different results we decided to first report performance values on the standard data set and then use same parameter settings on the second data set for reliable comparison. The second experiment is on a set of texts from the Touchstone Applied Science Associates, Inc., (TASA) corpus grouped according to genre: narrative (3 texts) and expository (4 texts). The gold standard for the second data set was built manually by the authors starting from Charniak's output on those texts.

The four texts used initially were two expository and two narrative texts of reasonable length for detailed evaluation:

1. The Effects of Heat (SRA Real Science Grade 2 Elementary Science): expository; 52

sentences, 392 words: 7.53 words per sentence,
2. The Needs of Plants (McGraw-Hill Science): expository; 46 sentences, 458 words: 9.96 words/sentence, respectively
3. Orlando (Addison Wesley Phonics Take-Home Reader Grade 2): narrative; 65 sentences, 446 words: 6.86 words/sentence,
4. Moving (McGraw-Hill Reading - TerraNova Test Preparation and Practice - Teachers Edition Grade 3): narrative, 33 sentences, 433 words: 13.12 words/sentence. An additional set of three texts was chosen from the TASA corpus with an average sentence length of 13.06 (overall TASA average) or higher.
5. Barron17: expository; DRP[3]=75.14 (college grade); 13 sentences, 288 words: 22.15 words/sentence
6. Betty03: narrative; DRP=56.92 (5th grade); 14 sentences, 255 words: 18.21 words/sentence
7. Olga91: expository; DRP=74.22 (college grade); 12 sentences, 311 words: 25.92 words/sentence

3.2. General Parser Evaluation Results

The parameters file we used for *evalb* (http://nlp.cs.nyu.edu/evalb/), the software evaluator, was the standard one that comes with the package. Some parsers are not robust, meaning for some input they do not output anything, leading to empty lines that are not handled by the evaluator. For those parses we had to “align” the gold standard files with the output from parsers so that empty lines are eliminated from the output file together with their peers in the corresponding gold standard files [4] [5] .

Table 1. Performance of Parsers on Section 23 of Wall Street Journal part of PTB.

Parser	Performance(LR/LP/Tagging - %)			
	# Sentences	PTB	Expository	Narrative
Apple Pie	2250	43.71/44.29/90.26	41.63/42.70	42.84/43.84
Charniak’s	2416	84.35/88.28/92.58	91.91/93.94	93.74/96.18
Collins/Bikel’s	2416	84.97/87.30/93.24	82.08/85.35	67.75/85.19
Stanford	2086	84.41/87.00/95.05	75.38/85.12	62.65/87.56

In Table 1 we report the performance values on Section 23 of WSJ and in Table 3 the values for the second data set. The reported metrics are Labelled Precision (LP) and Labelled Recall (LR). Let us denote by *a* the number of correct phrases in the output from a parser for a sentence, by *b* the number of incorrect phrases in the output and by *c* the

[3]DRP=Degree of Reading Power.

[4]Applie Pie’s performance is reported for sentences $<$ 40 words in length.

[5]Stanford parser is not robust enough: only 2094 out of 2416 sentences in section 23 of WSJ were properly processed by the parser and thus the reader should note that the reported performance here for Stanford parser is only on those 2094 sentences. The output for the other sentences was null.

Table 2. Evaluation of parsers with respect to the criteria listed at the top of each column.

Parser	Self-tagging	Performance	Long-sentences	Robustness
Apple Pie	Yes	No	No	No
Charniak's	Yes	Yes	Yes	Yes
Collins/Bikel's	Yes	Yes	Yes	Yes
Stanford	Yes	Yes	No	No

Table 3. Performance of Parsers on the narrative and expository text.

File	Performance(LR/LP - %)			
	Apple Pie	Charniak	Stanford	Bikel/Collins
Barron17	41.83/41.51	94.68/96.14	72.24/86.76	86.69/84.44
Betty03	45.12/43.11	97.21/96.76	83.72/80.72	71.63/77.39
Heat	46.86/46.86	94.69/97.76	88.41/91.27	90.58/91.24
Moving	36.59/40.15	92.27/97.36	51.14/88.58	75.23/84.22
Olga91	33.45/33.92	85.37/88.13	57.84/70.94	65.61/75.20
Orlando	47.47/47.26	93.62/94.22	63.75/91.25	58.25/91.37
Plants	41.91/45.18	91.91/92.90	76.38/86.30	82.13/86.16

number of phrases in the gold standard for the same sentence. LP is defined as *a/(a+b)* and LR is defined as *a/c*. A summary of the other dimensions of the evaluation is offered in Table 2. The stable dimension is not reported because we did not find any bullet-proof parser so far. But we must recognize that some parsers are significantly more stable than others, namely CP and CBP. In terms of resources needed the parser are comparable, except for AP which uses less memory and processing time. The *Performance* column indicates if the performance (LP and LR) is high, i.e. above 85%. In summary, CP offers best performance overall. One should note that its LP/LR are higher even though its tagging performance is not the best. The LP/LR of AP is significantly lower, partly due to its outputting partial trees for longer sentences.

4. Discussion and Future Work

The reader should note in Table 1 that Charniak's tagging accuracy is worst among the three top parsers, but still delivers best overall parsing results, meaning its only-parsing performance is slightly better than the numbers in the table indicate. The numbers actually represent the tagging and parsing accuracy of the tested parsing systems. Nevertheless, this is what we would most likely want to know since one would prefer to pass input in raw text as opposed to tagged text. If one wants to have a finer grained performance comparison of only the parsing aspects, one should pass in perfect tags extracted from PTB and then measure the performance.

Table 4. Average Performance of Parsers.

Parser	Average (LR/LP - %)	Standard Deviation (%)	Average on Exp+Nar (LR/LP - %)	Standard Deviation on Exp+Nar (%)
AP	42.72/43.61	1.04/0.81	42.23/43.27	0.85/0.80
CP	90.00/92.80	4.97/4.07	92.82/95.06	1.29/1.58
CBP	78.26/85.94	9.22/1.17	74.91/85.27	10.16/0.11
SP	74.14/86.56	10.93/1.27	69.01/86.34	9.00/1.72

Table 4 shows average measures for each of the parsers on all texts (PTB, expository and narrative texts) in the second column and for expository and narrative in the fourth column. The third and fifth columns contain standard deviations for the previous columns, respectively. Here, Charniak's parser has the best average performance.

We analyzed the narrative and expository texts. The most common problems involved the well-documented adjunct attachment site issue, in particular for prepositional phrases ([1], [2], [8], [15]) as well as adjectival phrases. Similar misattachment issues for adjuncts are encountered with adverbial phrases, but they were rare in our corpus and postprocessing is assumed to provide less improvement at higher costs.

Another common problem are deverbal nouns and denominal verbs, as well as *-ing* forms of varying syntactic functions that have identical surface forms because of the impoverished morphology of English.

Problems with NP misidentification are particularly detrimental in view of the important role of NPs in CM 2.0 measures. This pertains in particular to the mistagging/-parsing of complex NPs and the coordination of NPs. Parses with major problems are expected to produce useless results for algorithms operating with them. Wrong coordination is another notorious problem of parsers (cf. [9], [12]). In our corpus we found 33 instances of miscoordination, of which 23 involved NPs. We plan to quantify the incidence of those common errors that are important for cohesion studies and report the findings.

5. Conclusion

The paper presented an evaluation for Treebank style, freely available parsers. We offered a uniform evaluation for four parsers: Apple Pie, Charniak's, Collins/Bikel's and the Stanford parser. Apple Pie was chosen mainly due to its robustness and speed.

A novelty of this work is the evaluation of the parsers along new dimensions such as stability and robustness and across genre, in particular narrative and expository. For the latter part we developed a gold standard for narrative and expository from TASA corpus. Both from the general and across-genres evaluations, Charniak's parser emerged as the most succesful candidate.

References

[1] S. ABNEY, R.E. SCHAPIRE and Y. SINGER: Boosting applied to tagging and pp attachment. *In Proc. Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, (1999), 38-45.

[2] E. BRILL and P. RESNIK: A rule-based approach to prepositional phrase attachment disambiguation. *In Proc. 15th Int. Conf. on Computational Linguistics*, (1994).

[3] J. CARROLL, E. BRISCOE and A. SANFILIPPO: Evaluation of Natural Language Processing Systems: Final Report, chapter Parser evaluation: current practice, EC DG-XIII LRE EAGLES Document EAG-II-EWG-PR.1., (1999), 140-150.

[4] E. CHARNIAK: Statistical parsing with a context-free grammar and word statistics. *Proc. Fourteenth National Conf. on Artificial Intelligence*, AAAI Press/MIT Press, Menlo Park, (1997).

[5] E. Charniak. A maximum-entropy-inspired parser. *In Proc. North-American Chapter of Association for Computational Linguistics*, Seattle, Washington, (2000).

[6] M. COLLINS: A new statistical parser based on bigram lexical dependencies. *In Proc. 34th Annual Meeting of the ACL*, Santa Cruz, (1996).

[7] M. COLLINS: Three generative, lexicalised models for statistical parsing. *In Proc. 35th Annual Meeting of the Association for Computational Linguistic*, Madrid, Spain, (1997).

[8] M. COLLINS and J. BROOKS: Prepositional phrase attachment through a backed-off model. *In Proc. Third Workshop on Very Large Corpora*, Cambridge, (1995).

[9] C. CREMERS: *On Parsing Coordination Categorially*. PhD thesis, Leiden University, 1993.

[10] A. C. GRAESSER, D.S. MCNAMARA, M. M. LOUWERSE and Z. CAI: Coh-metrix: Analysis of text on cohesion and language. *Behavior Research Methods, Instruments, and Computers*, **36**(2), (2004), 193-202.

[11] C. MACLEOD, R. GRISHMAN and J. STERLING: Evaluating parsing strategies using standardized parse files. *In Proc. 3rd Conf. on Applied Natural Language Processing*, (1992), 156-161.

[12] M. GROOTVELD: Parsing Coordination Generatively. PhD thesis, Leiden University, 1994.

[13] D. KLEIN and C. MANNING: Accurate unlexicalized parsing. *In Proc. 41st Annual Meeting of the Association for Computational Linguistic*, Sapporo, Japan, (2003).

[14] D. LIN: A dependency-based method for evaluating broad-coverage parsers. *In Proc. Int. Joint Conf. on Artificial Intelligence*, (1995), 1420-1427.

[15] A. RATNAPARKHI, J. RENYAR and S. ROUKOS: A maximum entropy model for prepositional phrase attachment. *In Proc. ARPA Workshop on Human Language Technology*, (1994).

[16] S. SEKINE and R. GRISHMAN: A corpus-based probabilistic grammar with only two non-terminals. *In Proc. Int. Workshop on Parsing Technologies*, (1995), 216-223.