Time as a Measure of Parsing Efficiency

Robert C. Moore
Microsoft Research
One Microsoft Way
Redmond, Washington 98052, USA
bobmoore@microsoft.com

Abstract

Charniak and his colleagues have proposed implementation-independent metrics as a way of comparing the efficiency of parsing algorithms implemented on different platforms, in different languages, and with different degrees of "incidental optimization". We argue that there are easily immaginable circumstances in which their proposed metrics would mask significant differences in efficiency; we point out that their data do not, in fact, support the usability of such metrics for comparing the efficiency of different algorithms; and we analyze data for a similar metric to try to quantify the degree of variation one might expect between such metrics and actual parse time. Finally, we propose a methodology for making cross-platform comparisons through the use of reference parser implementations.

1 Introduction

The title "Time as a Measure of Parsing Efficiency" may seem highly tautologous, since in computer science "efficiency", unless otherwise qualified, is usually taken to mean speed of execution. However, Charniak and his colleagues (Caraballo and Charniak, 1998; Charniak, Goldwater, and Johnson, 1998; Blaheta and Charniak, 1999) have argued for another metric—edges popped off the agenda of a chart parser—as being platform independent. Now Roark and Charniak (2000) propose a related measure, "events considered", that is applicable to a wider range of approaches.

The present paper attempts to make the case for going back to time as the primary measure of parsing efficiency. First, we explore some general issues concerning the type of metrics proposed by Charniak and his colleagues. Then we demonstrate using empirical data that implementation-independent measures similar to that proposed by Charniak can vary in how well they correlate to execution time by as much or more than the improvements reported by Charniak et al. in some of their experiments. Finally, we propose a methodology for comparing parse times on different platforms, through the use of reference parser implementations.

2 Implementation-Independent Parser Metrics

As mentioned above, Charniak and his colleagues (Caraballo and Charniak, 1998; Charniak, Goldwater, and Johnson, 1998; Blaheta and Charniak, 1999; Roark and Charniak, 2000) have argued for implementation-independent measures of parsing efficiency, including edges popped off the agenda of a chart parser and, currently, "events considered", which is defined by Roark and Charniak (2000) as "the number of 'events', however they are defined for a particular parser, for which a probability must be calculated, in order to find the parse." It is argued in these papers that such metrics enable parsing efficiency to be compared at the algorithmic level without being concerned with the degree of low-level optimization that has been performed.

Roark and Charniak have applied the events-considered metric both to a best-first parser and to a word-synchronous beam-search-based parser. These parsers differ in many respects, but have a number of attributes in common that bear on the current discussion:

- They both are probabilistic, and seek to find the parse with the highest probability according to some model.
- They are both heuristically pruned. That is, they do not explore all analyses that

have non-zero probability, nor are they even guaranteed to explore all analyses that might turn out to have the highest probability.

- They both prune partial analyses on the basis of a figure of merit that can be viewed as based on an heuristic estimate of the expected probability that the full model would assign to extensions of a given partial analysis.

Within this framework, let us consider some of the ways that the number of events considered could fail to correlate with parse time in an essential way; that is, not based on what Roark and Charniak refer to as "incidental optimizations". First, the full probability models might take vastly different amounts of time to compute. For example, maximum-entropy, or exponential, models (Berger, Della Pietra, and Dell Pietra, 1996) are believed by some to hold the promise of significantly higher accuracy in a variety of tasks than the more conventional, simpler models that Charniak and colleagues have used. Unfortunately, maximum entropy models are much more expensive to compute than these simpler models, if they are normalized to produce true probability distributions.[1] Thus if we compared a parser that used a normalized exponential model to compute the probability of exactly the same events that one of Roark's and Charniak's parsers considers, we would expect to find it much slower than theirs. This might be a trade-off worth making, if it delivered better parsing accuracy, but it would be ludicrous to claim it was equally efficient, simply because it considered no more events than Roark's and Charniak's parsers.

Another way that the events-considered metric might fail to correlate with parse time concerns the figure of merit used to prune he search. The efficacy of pruning is perhaps the most important determinant of parsing efficiency in the types of parsers considered by Roark and Charniak. The better the figure of merit is as a predictor of the probability assigned by the full model to the best extension of a partial analysis, the smaller the number of partial analyses that need to be extended to find the best full analysis. It is easy to imagine, however, that some "better" figure of merit might take more time to compute than it repays in reducing the search space. Suppose a there is a sophisticated figure of merit that allows reducing the number of events considered by half over a cruder figure of merit, but it takes so long to compute that it increases the amount of time per event considered by a factor of ten. The events-considered metric will rate the sophisticated figure of merit as twice as good, even though in reality a parser that used it would be five times slower. This is a classic problem in heuristic search and has been discussed extensively in the literature on computer chess and automatic theorem proving.

3 Empirical Evaluation

The arguments in the preceding section are in a sense speculative, since they discuss possibilities that might occur, but we have not demonstrated that they do occur in practice. For that, empirical evidence is required. At first blush, Roark and Charniak might seem to have produced empirical evidence to the contrary, since they present graphs of impressively linear relationships between events considered and time. These linear relationships, however, are demonstrated only with respect to one parser at a time, where the only thing that is varied is the degree of pruning. Thus, although Roark and Charniak argue for events-considered metric as a way of comparing very different parsing algorithms, they demonstrate only that it correlates well with efficiency for essentially identical algorithms, where only a single parameter is varied.

We have recently carried out a series of experiments (Moore, 2000) comparing the efficiency of several variants of left-corner parsing with a number of other well-known context-free parsing algorithms. In contrast to the studies of Charniak et al., these experiments looked at non-stochastic algorithms, and the comparisons are made on the basis of exhaustive, rather than heuristically-pruned, parsing. Thus we cannot directly apply either Roark and Charniak's events-considered metric, or the earlier edges-popped-of-the-agenda metric, to our results. However, the total number of edges in the chart, which we do have data on, is a very similar sort of implementation-independent metric, and is often used informally to judge parser ef-

[1]Note that Charniak (2000) has also explored *unnormalized* exponential models, which are fast to compute.

ficiency. (Arguably, this would be the same as the edges-popped-of-the-agenda metric, when parser is allowed to run to exhaustion.)

To evaluate the suitability of using total number of edges in the chart as an efficiency metric, we will present a selection of results from our CFG parsing experiments, including edge statistics not previously published. Results for five parsing algorithms on three different grammars are included. The parsing algorithms consist of two variants of left-corner parsing (LC_1+BUPM and LC_2+BUPM), an Earley/Graham-Harrison-Ruzzo parser (E/GHR), a Cocke-Kasami-Younger parser (CKY), and a generalized LR parser without look-ahead (GRL(0)). (The identifiers for these parsers are the ones used in our earlier report.) It should be mentioned that all these parsers represent the best of several implementations of the general approach, and all parsers are implemented using similar techniques and data structures wherever possible. Furthermore, all algorithms are implemented in the same language (Perl5) on the same platform (Windows 2000, 550 MHz Pentium III). Thus we believe that the performance differences are genuinely representative of inherent differences in the algorithms, and not just irrelevant implementation details.

The grammars used are independently motivated by analyses of natural-language corpora or actual applications of natural language processing. The CT grammar was compiled into a CFG from a task-specific unification grammar written for CommandTalk (Moore et al., 1997), a spoken-language interface to a military simulation system. The ATIS grammar was extracted from an internally generated treebank of the DARPA ATIS3 training sentences. The PT grammar is Charniak's PCFG grammar extracted from the Penn Treebank, with the probabilities omitted. The most significant variation among the grammars is the degree of ambiguity of the test sets associated with each grammar. The CT test set has 5.4 parses/sentence with the CT grammar, the ATIS test set has 940 parses/sentence with the ATIS grammar, and the PT test set has 7.2×10^{27} parses/sentence with the PT grammar.

Table 1 shows the results of applying these five parsers to the three grammars and their associated test sets. The first column gives the average number of chart edges per sentence, including both complete and incomplete edges (where incomplete edges are generated). For the GLR(0) parser, this is equivalent to the number of edges in the graph-structured stack used by most implementations of GLR parsing. The second column gives the average number of seconds per sentence to parse exhaustively. This includes only time to populate the chart, and does not include time to extract parses. The final column compares the second column to the first, to derive an average number of milliseconds per chart edge. The more constant this number is across the different parsers and grammars, the better total edges in the chart will be as a measure of parser efficiency.

If we look at the last column in detail, we see that total number of chart edges generated does have some crude validity as a measure of parsing efficiency; since the majority of the test cases fall around 0.1 milliseconds per edge. However, the variation is fairly large. The two left-corner parsers make a particularly interesting comparison, because they differ only in a single detail, and produce exactly the same edges. In these parsers incomplete edges are subjected to two tests before being added to the chart. The mother of an incomplete edge has to be a possible left corner of the next daughter required by some previous incomplete edge at the appropriate position in the input; furthermore, the next daughter of the incomplete edge being tested has to have the next token in the input as a possible left corner. These tests are independent, so they can be performed in either order. In LC_1+BUPM the check on the mother is performed first, and in LC_2+BUPM the check on the next daughter is performed first. These results show that performing the check on the mother first is 14% to 68% slower than performing the check on the next daughter first. This is a substantial difference that cannot be detected looking only at the edges added to the chart.

There are several places in the data where the numbers of chart edges strongly, but incorrectly, predict which of two parsers should be faster on a given grammar. For example, the LC_1+BUPM parser generates only about half as many edges as the E/GHR parser with the ATIS grammar, but is nevertheless 35% slower.

Grammar	Parser	Edges/sent	Sec/sent	msec/edge
CT	LC_1+BUPM	165.3	0.0219	0.132
	LC_2+BUPM	165.3	0.0191	0.116
	E/GHR	283.0	0.0448	0.158
	CKY	1598.2	0.1540	0.096
	GLR(0)	159.0	0.0214	0.135
ATIS	LC_1+BUPM	673.4	0.119	0.177
	LC_2+BUPM	673.4	0.071	0.105
	E/GHR	1276.6	0.088	0.069
	CKY	537.3	0.078	0.145
	GLR(0)	1282.5	0.143	0.112
PT	LC_1+BUPM	6675.4	1.14	0.171
	LC_2+BUPM	6675.4	0.90	0.135
	E/GHR	11143.9	0.92	0.083
	CKY	5785.6	1.70	0.294
	GLR(0)	51285.9	28.86	0.563

Table 1: Chart edge and time statistics for CFG parsing algorithms

It is also notable that the CKY and GLR(0) parsers are not even self-consistent in terms of time per edge across grammars. They take substantially more time per edge on the PT grammar than on the other two.

One way to quantify the uncertainty about the relationship between edges produced and parse time is to use statistical analysis to estimate how many fewer edges one parser has to produce to be reasonably certain that it is actually faster than another. We have performed a crude version of such an analysis on the data in Table 1, under the assumption that they represent a random sample of parsers and grammars, suggesting the following: To be 99% certain that one parser is faster than another, it must produce about 80% fewer edges; to be 95% certain, it must produce about 70% fewer edges; and to be 90% certain, it must produce about 60% fewer edges. This analysis undoubtedly suffers from the paucity of the data, and performing a wider range of experiments might well give us tighter bounds, but it at least represents a first cut at quantifying the variation in time-per-edge across different parsing algorithms and grammars. (See the Appendix for information on how the analysis was performed.)

If we look at Charniak's latest paper reporting improved parser efficiency (Blaheta and Charniak, 1999), we find a reported reduction in edges popped off the agenda of about 60%. We do not doubt that this result is in fact a substantial improvement over the previous method it was compared to, but in the context of the variation in time per edge among the parsers presented here, we would be hard pressed to claim this result represents a statistically significant improvement in parsing speed, without seeing parsing times not provided by Blaheta and Charniak.

4 Towards a Cross-Platform Methodology

The substantial variation in time per edge among the parsers and grammars discussed here strongly suggests that, in making efficiency comparisons, actual parse times should be measured whenever that would be meaningful. For a particular research team reporting incremental progress of a continuing research program, this hardly seems too much to ask.

For cross-research-group comparisons the matter is more problematical. How is one to compare a parser written in Lisp running on an 400 MHz UltraSPARC processor to one written in C++ running on a 750 MHz Pentium III? The answer proposed here is to create a set of reference parser implementations, in a variety of languages, coupled with reference grammars and test sets. Obviously, for different classes of grammar—unification grammars, CFGs, and PCFGs—meaningful comparisons remain difficult; but for a given class, this approach should make cross platform comparisons straightforward. For example, for CFGs, a particular variant of CKY could be chosen, and implemented as efficiently as possible in C, Lisp, Prolog, Perl(!), and any other language considered relevant. The source code for implementations would need to be provided, so that the claim to be the best possible implementation of the algorithm in the language could be examined, and improvements made, without changing the basic algorithm. Then, whenever anyone claims to have an improved algorithm for CFG parsing, written in any of the supported languages and any platform that supports the language, they could meaningfully measure how much faster their algorithm is than CKY on the standard grammars and test sets, by timing their parser against the reference CKY parser for their platform. Similarly, reference parser implementations could be created for PCFGs and unification grammars to facilitate cross-research-group studies of parser efficiency for those formalisms.

5 Conclusions

Charniak and his colleagues have proposed implementation-independent metrics as a way of comparing the efficiency of parsing algorithms implemented on different platforms, in different languages, and with different degrees of "incidental optimization". We have tried to argue that there are easily immaginable circumstances in which their proposed metrics would mask significant differences in efficiency; we have pointed out that Roark and Charniak's data do not, in fact, support the usability of their metric for comparing the efficiency of different algorithms; and we have analyzed data for a similar metric (chart size) to try to quantify the degree of variation one might expect between such metrics and actual parse time.

Finally, we have proposed a methodology for making cross-platform comparisons through the use of reference parser implementations. This methodology does not, unfortunately, address Roark and Charniak's desire for a metric that is insensitive different degrees of incidental optimization. Indeed, in our own work comparing different parsing algorithms we have tried to be scrupulous in comparing implementations that are as similar as possible in that regard. In effect, Charniak et al. seem to be seek metrics which obviate the need for controled experiments, but in the end, there is no substitute for them.

References

Berger, A., S. Della Pietra, and V. Della Pietra (1996) "A Maximum Entropy Approach to Natural Language Processing," *Computational Linguistics*, Vol. 22, No. 1, pp. 39–71.

Blaheta, D., and E. Charniak (1999) "Automatic Compensation for Parser Figure-of-Merit Flaws," in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, College Park, Maryland, pp. 513–518.

Caraballo, S., and E. Charniak (1998) "New Figures of Merit for Best-First Probabilistic Chart Parsing," *Computational Linguistics*, Vol. 24, No. 2, pp. 275–298.

Charniak, E., S. Goldwater, and M. Johnson (1998) "Edge-Based Best-First Chart Parsing," in *Proceedings of the Sixth Workshop on Very Large Corpora*, Montreal, Canada, pp. 127–123.

Charniak, E. (2000) "A Maximum-Entropy-Inspired Parser," in *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, Seattle, Washington, pp. 132–139.

Moore, R., et al. (1997) "CommandTalk: A Spoken-Language Interface for Battlefield Simulations," in *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Association for Computational Linguistics, Washington, DC, pp. 1–7.

Moore, R. (2000) "Improved Left-Corner Chart Parsing for Large Context-Free Grammars," in *Proceedings of the Sixth International Workshop on Parsing Technologies*, ACL/SIGPARSE, Trento, Italy, pp. 171–182.

Roark, B., and E. Charniak (2000) "Measuring Efficiency in High-Accuracy, Broad-Coverage Statistical Parsing," in *Proceedings of the COLING 2000 Workshop on Efficiency in Large-Scale Parsing Systems*, Luxembourg.

Confidence Level	Method 1	Method 2
0.99	0.19	0.17
0.95	0.30	0.29
0.90	0.39	0.38

Table 2: Edge ratios for different significance levels

Appendix

The statistical analysis mentioned in Section 3 was performed by comparing the milliseconds/edge values for all pairs of parsing algorithms, for each grammar and test set. Specifically, we computed the t statistic for logarithms of ratios of milliseconds/edge values for sets of pairs of parsers applied to the same grammar and test set. We used logarithms of ratios rather than the ratios themselves, on the grounds that, *a priori* we would expect values of n and $1/..n$ to be about equally likely for this ratio.

We computed the t statistic in two different ways. Method 1 was simply to compute the t statistic for the set of logarithms of ratios of the milliseconds/edge values for all pairs of parsing algorithms, for each grammar and test set. This is not statistically valid, however, because the values we are computing the t statistic for are not independent, due to re-use of milliseconds/edge values in different pairwise comparisons. Essentially, we have manufactured 30 data points out of an original set of only 15 data points.

In Method 2, we maintained independence by randomly selecting pairs of parsers so that each parser was only counted once, applied to each grammar and test set. Since this only looks at a subset of the possible pairwise comparisons, we repeated this 1000 times, so that each possible pair would be selected approximately the same number of times, and averaged the results. We then used these t statistics to compute the ratios of edges required to be certain that the parser producing fewer edges would actually take less time than the other, at the 0.99, 0.95, and 0.90 confidence levels, using a single-tailed test (See Table 2). The two methods did not differ greatly in their results, except that Method 2 produced slightly greater uncertainty. This may have been simply due to the smaller number of data points used in each iteration in Method 2.