Limitations of Current Grammar Induction Algorithms

Bart Cramer
School of Behavioral and Cognitive Neurosciences
University of Groningen
Groningen, the Netherlands
bart.cramer@gmail.com

Abstract

I review a number of grammar induction algorithms (ABL, Emile, Adios), and test them on the Eindhoven corpus, resulting in disappointing results, compared to the usually tested corpora (ATIS, OVIS). Also, I show that using neither POS-tags induced from Biemann's unsupervised POS-tagging algorithm nor hand-corrected POS-tags as input improves this situation. Last, I argue for the development of entirely incremental grammar induction algorithms instead of the approaches of the systems discussed before.

1 Introduction

Grammar induction is a task within the field of natural language processing that attempts to construct a grammar of a given language solely on the basis of positive examples of this language. If a successful method is found, this will have both practical applications and considerable theoretical implications.

Concerning the practical side, this will make the engineering of NLP systems easier, especially for less widely studied languages. One can conceive successful GI algorithms as an inspiration for statistical machine translation systems.

Theoretically, grammar induction is important as well. One of the main assertions in the nativist's position is the Poverty of the Stimulus argument, which means that the child does not perceive enough positive examples of language throughout his early youth to have learned the grammar from his parents, without the help of innate knowledge (or: Universal Grammar), that severely constrains the number of hypotheses (i.e. grammars) that he can learn. Proved more strictly for formal grammars, Gold's (1967) work showed that one cannot learn any type of superfinite grammar (e.g. regular languages, context-free languages), if one only perceives (an unlimited amount of) positive examples. After, say, n examples, there is always more than 1 grammar that would be able to explain the seen examples, thus these grammar might give different judgments on an $n + 1^{th}$ example, of which it is impossible to say in advance which judgment is the correct one.

But, given this is true, isn't the grammar induction pursuit deemed to fail? Not really. First, there are hints that children do receive negative information, and that they use it for grammar acquisition. Also, the strictness required by Gold is not needed, and an approximation in the framework of PAC (Probably Approximately Correct) or VC (Vapnis and Chervonenkis) could then suffice. This, and other arguments favouring the use of machine learning techniques in linguistic theory testing, are very well reviewed in Lappin and Shieber (2007).

Several attempts have been made to create such systems. The authors of these systems reported promising results on the ATIS and OVIS treebanks. I tried to replicate these findings on the more complicated Eindhoven treebank, which turned out to yield disappointing results, even inferior to very simple baselines. As an attempt to ameliorate this, and as an attempt to confirm Klein and Manning's (2002) and Bod's (2006) thesis that good enough unsupervised POS-taggers exist to justify using POS-tags instead of words in evaluating GI systems, I pre-

Proceedings of the ACL 2007 Student Research Workshop, pages 43–48,
Prague, June 2007.

sented the algorithms with both POS-tags that were induced from Biemann's unsupervised POS-tagging algorithm and hand-corrected POS-tags. This did not lead to improvement.

2 Current Grammar Induction Models

2.1 Algorithms

Grammar induction models can be split up into two types: tag-based and word-based grammar induction. The key feature that distinguishes between these two is the type of input. Tag-based systems receive part-of-speech tags as their input (i.e. the words are already labelled), and only induce rules using the given tags. This kind of work is done by, for instance, Klein and Manning (2005). On the other hand, word-based models accept plain text as its input, and have to extract both the categories and the syntactic rules from given input.

Recently, several word-based grammar induction algorithms have been developed: Alignment-Based Learning (van Zaanen, 2002), Adios (Solan et al., 2005), Emile (Adriaans, 1992; Adriaans and Vervoort, 2002) and GraSp[1] (Henrichsen, 2002). Although the means of computation and underlying aims differ, they all rely to a certain extent on Harris' principle (1951): if two word groups constitute the same category, then they can be interchanged in any sentence, without damaging the grammaticality of that sentence. Hence, these GI system depend on the inverse: if two word groups appear to occur in the same contexts, they probably possess the same syntactic characteristics.

The most prominent example of this principle is Alignment-Based Learning, or ABL, (van Zaanen, 2002). This algorithm consists of two stages. First, all sentences are aligned such that it finds a shared and a distinct part of all pairs of sentences, suggesting that the distinct parts have the same type. For example, consider the pair 'I saw the man' and 'I saw John'. Here, 'John' and 'the man' are correctly identified as examples of the same type (NP's in this case). The second step, that takes the same corpus as input, tries to identify the constituents in that sentence. Because the generated constituents found in the previous step might overlap, the correct ones have to be selected. Simple heuristics are used to achieve this, for example to take the constituent that was generated first (ABL-first) or to take the constituent with the highest score on some probabilistic function (ABL-leaf). For details, I refer to van Zaanen (2000). Because ABL compares all sentences in the corpus with all other sentences, the algorithm is quadratic in the number of sentences, but has low memory demands. Interestingly, ABL does not come up with an explicit grammar, but generates just a bracketed version of the corpus instead.

Adios (Solan et al., 2005) uses Harris' principle as well, although it attempts to create a grammar (either context-free or context-sensitive) more explicitly. The algorithm represents language as a directed pseudograph[2], with *equivalence classes* (initially single words) as nodes. Input sentences can be regarded as 'snakes' over the nodes in the graph. If enough support is found, words are merged into equivalence classes, or frequently occurring edges are put in a *path* (a *rule* in usual grammatical terms). This generalisation process is done iteratively, until convergence is reached.

Emile (Adriaans, 1992; Adriaans and Vervoort, 2002) is the system that to a greater extent tries to pinpoint its reasons to accept a linguistic hypothesis. Each rule is divided into *expressions* and *types*, where types should be the interchangeable part of two sentences. Instead of explicitly comparing each sentence with all other sentences, it incrementally builds up a table of type/expression pairs, and on the basis of this table rules are extracted. An example is given in table 1. This incrementality has two major

	John (.)	Pat (.)	Jim (.)
walks	x	x	
talks		x	x
smiles	x	x	

Table 1: An example of some context/expression pairs to show the workings of EMILE. Note that, under standard settings, a rule covering this entire table will be inferred, causing a phrase like 'John talks' to be accepted, although there was no such input sentence.

[1] As there was no current working version of this system, I did not include it in this project.

[2] This is a graph that allows for loops and multiple edges.

consequences: it makes the system vastly more efficient in terms of time, at the cost of rising memory demands, and it models time linearly, in contrast to ABL and Adios.

2.2 Evaluation

Different methods of evaluation are used in GI. One of them is visual inspection (Henrichsen, 2002). This is not a reproducible and independent evaluation measure, and it does certainly not suffice as an assessment of the quality of the results. However, Roberts and Atwell (2003) argue that this evaluation should still be included in GI discussions.

A second evaluation method is shown by Solan et al. (2005), in which Adios had to carry out a test that is available on the Internet: English as a Second Language (ESL). This test shows three sentences, of which the examinee has to say which sentence is the grammatical one. Adios answers around 60% correct on these questions, which is considered as intermediate for a person who has had 6 years of English lessons. Although this sounds impressive, no examples of test sentences are given, and the website is not available anymore, so we are not able to assess this result.

A third option is to have sentences generated by the induced grammar judged on their naturalness, and compare this average with the average of the sentences of the original corpus. Solan et al. (2005) showed that the judgments of Adios generated sentences were comparable to the sentences in their corpus. However, the algorithm might just generates overly simple utterances, and will receive relatively high scores that it doesn't deserve.

The last option for evaluation is to compare the parses with hand-annotated treebanks. This gives the most quantifiable and detailed view on the performance of a GI system. An interesting comparative study between Emile and ABL using this evaluation method is available in van Zaanen and Adriaans (2001) where F-scores of 41.4% (Emile) and 61.7% (ABL) are reported on the OVIS (Openbaar Vervoer Informatie Systeem[3]; Dutch) corpus, and 25.4% and 39.2% on the ATIS (Air Traffic Information System; English) corpus.

[3]This acronym means Public Transport Information System.

3 Experiment 1

3.1 Motivation

A major choice in evaluating GI systems is to decide which corpus to train the algorithm on. The creators of ABL and Emile chose to test on the ATIS and OVIS corpus, which is, I believe, an unfortunate choice. These corpora contain sentences that are spoken to a computer, and represent a very limited subset of language. Deep recursion, one of the aspects that is hard to catch in grammar induction, does not occur often. The average sentence lengths are 7.5 (ATIS) and 4.4 (OVIS). If we want to know whether a system is truly capable of bootstrapping knowledge about language, there is only one way to test it: by using natural language that is unlimited in its expressive power. Therefore, I will test ABL, Adios and Emile on the Eindhoven corpus, that contains 7K sentences, with an average length of approximately 20 tokens. This is, as far as I know, the first attempt to train and test word-based GI algorithms on such a complicated corpus.

3.2 Method

The Eindhoven corpus has been automatically annotated by Alpino (Bouma et al., 2000; van der Beek et al., 2002), a wide-coverage hand-written parser for Dutch, with around 90% dependency triple accuracy. Afterwards, this treebank has been manually corrected. The treebank does not literally contain trees, but graphs: some nodes can be copied, so that linguistic structure can be analyzed in more detail. However, by removing all double nodes it is still possible to retrieve a list of bracket-tuples from these graphs. The graphs are also non-concatenative, meaning that a constituent can span word groups that are not contiguous. Therefore, if a sentence contains a constituent $w_i...w_jw_k...w_l$, with $k - j > 1$, three bracket-tuples are generated: (i, j), (k, l) and (i, l).

Evaluation of the algorithm is done according to PARSEVAL, except for a few changes that are also proposed by Klein and Manning (2002). The set of bracket-pairs that is found in the Alpino treebank are called *facts*, and those from a grammar induction algorithm *predictions*. The intersection of the facts and predictions are called *hits*. From these we can compute the unlabeled precision, recall and F-score. The subtleties adopted from Klein and Manning are the following: constituents of length 0 or 1, constituents that span the whole sentence and constituents just excluding punctuation are not taken into account, as these are obvious predictions.

Three baselines were created: an algorithm that always branches left[4], idem for right-branching and an algorithm that performs binary branching on random points in the sentence. Note that left-branching and right-branching yield the maximum number of predictions.

3.3 Results

From the results in table 2, it can be seen that ABL scores best: it is the only one that is able to slightly outperform the random baseline. This is surprising, because it is the least complicated system of the three. Adios and Emile performed poorly. It appears that, with larger sentences, the search space become too sparse to actually induce any meaningful structure. This is expressed in the low number of predictions per sentence that Adios (1.5) and Emile (0.7) make. Adjusting support parameters, to make the algorithm accept more hypotheses, did not have the intended effect. Still, notice that Emile has a relatively high precision.

In sum, none of the systems is convincingly able to outperform the very simple baselines. Neither did visual inspection give the impression that meaningful information was derived. Therefore, it can be concluded that current word-based GI algorithms are not equipped to derive syntactic structure from corpora as complicated as the Eindhoven corpus.

4 Experiment 2

4.1 Motivation

The second experiment deals with the difference between tag-based and word-based systems. Intuitively, the latter task seems to be more challenging. Still, Klein and Manning (2002) and Bod (2006) stick to tag-based models. Their argumentation is twofold.

First, Bod assumes that unsupervised POS-tagging can be done successfully, without explicitly showing results that can confirm this. Klein and Manning did tag their text using a simple unsupervised POS-tagging algorithm, and this moderately harmed their performance: their Context-Constituent Model's F-score on Wall Street Journal text fell from 71.1% to 63.2%.

Second, Klein and Manning created context vectors for a number of non-terminals (NP, VP, PP), and extracted the two principal components from these vectors. They did the same with contexts of constituents and distituents. The distribution of these vectors suggest that the non-terminals were easier to distinguish from each other than the constituents from the distituents, suggesting that POS-tagging is easier than finding syntactic rules. However, this result would be more convincing if this is true for POS-tags as well.

4.2 Method

In order to test the argument above, and as an attempt to improve the results from the previous experiment, POS-tags were induced using Biemann's unsupervised POS-tagger (Biemann, 2006). Because that algorithm needs at least 50M words to work reliably, it was trained on the concatenation of the Eindhoven corpus and the CLEF corpus (70M words, also newspaper text). The tags of the Eindhoven corpus are then used as input for the GI algorithms, both under same settings as experiment 1. The evaluation was done the same way as in experiment 1.

The same method was carried out using hand-corrected tags. Large and equal improvements will imply the justification for tag-based grammar induction. If the models only improve on the hand-corrected tags, this will suggest the opposite.

4.3 Results

The results can be found in table 3. Generally, more predictions were made with respect to experiment 1, due to the denser search space. Only a convergence to the baseline was achieved, especially by Adios and Emile, that were very low in predictions in the first experiment. Again, none of the tested systems was able to clearly outperform the baselines.

Because using neither induced nor hand-corrected made the systems work more reliably, there seems to be no strong evidence in favor or against Bod's and Klein and Manning's conjecture. Therefore, there is no sound justification for tag-based grammar induction yet.

[4]For example: [[[I saw] the] large] house.

Method	Hits/Predictions	Precision	Recall	F-score
Left	5.8K / 119K	4.9%	9.2%	6.4%
Right	4.4K / 119K	3.6%	6.9%	4.8%
Random	11K / 93K	11.7%	17.3%	14.0%
ABL-leaf	4.0K / 24K	16.9%	6.4%	9.3%
ABL-first	13K / 113K	11.6%	20.8%	14.9%
Adios	319 / 11K	2.8%	0.5%	0.9%
Emile	912 / 5.2K	17.3%	1.5%	2.7%

Table 2: This table shows the results of experiment 1. Left, Right and Random are baseline scores. The two variants of ABL differ in the selection phase. 62.9K facts were found in the Alpino treebank.

	Induced tags				Hand-corrected tags			
Method	Hits/Pred.'s	Precision	Recall	F-score	Hits/Pred.'s	Precision	Recall	F-score
ABL-leaf	5K / 30K	16.8%	8.1%	10.9%	7.0K / 34K	21.0%	11.2%	14.6%
ABL-first	11K / 125K	9.2%	18.2%	12.2%	12.6K / 123K	10.3%	20.0%	13.6%
Adios	2.7K / 24K	11.2%	4.3%	6.3%	2.2K / 20K	11.0%	3.5%	5.3%
Emile	1.8K / 16K	11.2%	2.9%	4.6%	1.7K / 19K	8.9%	2.7%	4.1%

Table 3: This table shows the results of experiment 2. The baseline scores are identical to the ones in experiment 1.

5 Discussion

The results from experiment 1 and 2 clearly show that ABL, Adios and Emile have severe shortcomings, and that they cannot derive meaningful structure from language as complicated as the Eindhoven corpus. An important reason for this is that a corpus with only 7K sentences is not able to sufficiently cover the search space. This can be seen from the very low number of predictions made by Adios and Emile: there was not enough support to accept hypotheses.

But how should we proceed? Any algorithm based solely on Harris' principle can be either incremental (Emile) or non-incremental (ABL, Adios). The previous experiments show that very large corpora are needed to mitigate the very sparse search space, leading me to conclude that non-incremental systems are not suitable for the problem of grammar induction. Also, incremental systems have the advantage of an intuitive notion of time: it is always clear which working hypothesis of a grammar is maintained.

Emile retains a Boolean table with all combinations of types and expressions it has encountered up until a given moment. This means that very infrequent words demand a disproportionally large part of the memory. Therefore, all found words and rules should be divided into three groups: pivotal, normal and infrequent. Initially, all encountered words are infrequent. Transitions to the normal and pivotal stage occur when an estimator of the relative frequency is high enough, for example by taking the lower bound of the confidence interval (Mikheev, 1997). Ultimately, the number of words in the normal and pivotal stage will converge to a constant. For example, if the relative frequency of a word should be larger than 0.01 to become pivotal, there can only be 100 of these words. Because one can define upper limits for pivotal and normal words, the size of the bookkeeping table is limited as well. Also, when the system starts inducing syntactic categories of words, very infrequent words should not be parsed as a separate category initially, but as a member of another open-class category. This connects to the cross-linguistic tendency that infrequent words generally have simple complementation patterns.

One very important question remains: what intuitions should this imaginary system use to induce rules? First, all sentences should be sorted by length. Then, for each sentence, the following steps are taken:

- Update the bookkeeping tables.
- Parse the sentence as deeply as possible.
- If the sentence cannot be parsed completely, induce all possible rules that would make the parse complete. Add all these rules to the bookkeeping tables.

The last step deserves some extra attention. If the algorithm encounters the sentence 'he is such a (.)', we can safely infer that the unknown word at (.) is a noun. Inducing complementation patterns should be possible as well. Imagine that the algorithm understands NP's and transitive verbs. Then consider the following: 'John gave Tim a book'. It will parse 'John gave Tim' as a sentence, and 'a book' as a noun phrase. Because these two should be connected, a number of hypotheses are generated, for example: 'a book' is a complement of 'Tim'; 'a book' is a complement of 'John gave Tim'; 'a book' is a second complement of 'gave'. Naturally, only the last hypothesis is correct. All three inductions are included, but only the last is likely to be reproduced in later sentences in the corpus, because sentences of the form '(.) gave (.) (.)' are more likely than 'John gave Tim (.)' and 'Tim (.)'.

6 Acknowledgments

I would like to thank Jennifer Spenader, Gertjan van Noord and the anonymous reviewers for providing me their invaluable comments.

References

Pieter W. Adriaans and Mark R. Vervoort. 2002. The EMILE 4.1 grammar induction toolbox. In *Proceedings of the 6th International Colloquium on Grammar Induction (ICGI)*, pages 293–295, Amsterdam, the Netherlands.

Pieter W. Adriaans. 1992. *Language learning from a categorial perspective*. Ph.D. thesis, University of Amsterdam, NL.

Chris Biemann. 2006. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of ACL/COLING-2006 Students Research Workshop*, pages 7–12, Sydney, Australia.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proceedings of ACL/COLING-2006*, pages 865–872, Sydney, Australia.

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2000. Alpino: wide-coverage computational analysis of Dutch. In *Proceedings of Computational Linguistics in the Netherlands (CLIN)*, pages 45–59, Tilburg, the Netherlands.

E. Mark Gold. 1967. Language identification in the limit. *Information and Control*, 16:447–474.

Zellig S. Harris. 1951. *Methods in Structural Linguistics*. University of Chicago Press, Chicago.

Peter J. Henrichsen. 2002. GraSp: Grammar learning from unlabelled speech corpora. In *Proceedings of CoNLL-2002*, pages 22–28, Pennsylvania, PA, USA.

Dan Klein and Christopher D. Manning. 2002. A generative Constituent-Context Model for improved grammar induction. In *Proceedings of ACL-2001*, pages 128–135, Toulouse, France.

Dan Klein and Christopher D. Manning. 2005. Natural language grammar induction with a generative constituent-context model. *Pattern Recognition*, 9(38):1407–1419.

Shalom Lappin and Stuart M. Shieber. 2007. Machine learning theory and practice as a source of insight into universal grammar. *Computational Linguistics*, 43:1–34.

Andrei Mikheev. 1997. Automatic rule induction for unknown-word guessing. *Computational Linguistics*, 23(3):405–423.

Andrew Roberts and Eric Atwell. 2003. The use of corpora for automatic evaluation of grammar inference systems. In *Proceedings of the Corpus Linguistics 2003 conference*, pages 657–661, Lancaster, United Kingdom.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. 2005. Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences*, 102(33):11629–11634.

Leonoor van der Beek, Gosse Bouma, Robert Malouf, and Gertjan van Noord. 2002. The Alpino dependency treebank. In *Proceedings of Computational Linguistics in the Netherlands (CLIN) 2001*, pages 8–22, Enschede, the Netherlands.

Menno van Zaanen and Pieter W. Adriaans. 2001. Alignment-Based Learning versus EMILE: A comparison. In *Proceedings of the 13th Dutch-Belgian Artificial Intelligence Conference (BNAIC)*, pages 315–322, Amsterdam, the Netherlands.

Menno van Zaanen. 2002. Implementing Alignment-Based Learning. In *Proceedings of the 6th International Colloquium on Grammatical Inference (ICGI)*, pages 312–314, Amsterdam, the Netherlands.