ML-Optimization of Ported Constraint Grammars

Eckhard Bick
University of Southern Denmark
Campusvej 55, DK-5230 Odense M
Email: eckhard.bick@mail.dk

Abstract

In this paper, we describe how a Constraint Grammar with linguist-written rules can be optimized and ported to another language using a Machine Learning technique. The effects of rule movements, sorting, grammar-sectioning and systematic rule modifications are discussed and quantitatively evaluated. Statistical information is used to provide a baseline and to enhance the core of manual rules. The best-performing parameter combinations achieved part-of-speech F-scores of over 92 for a grammar ported from English to Danish, a considerable advance over both the statistical baseline (85.7), and the raw ported grammar (86.1). When the same technique was applied to an existing native Danish CG, error reduction was 10% (F=96.94).

Keywords: Constraint Grammar, Machine Learning, Tagging

1. Introduction

Mature Constraint Grammar (CG) parsers can achieve very high accuracy (Karlsson et al. 1995), but contain thousands of manually crafted rules and depend on large amounts of expert labor. Input to a Constraint Grammar is traditionally provided by a lexical-morphological analyzer[1] that outputs tokens with one or more possible readings (tag lines) attached. The task of a CG rule is then to contextually remove wrong readings, select correct ones, and to add and disambiguate tags that can only be inferred in a contextual way, such as syntactic function, dependency and semantic roles.

A typical CG rule contains an a target (word form, lemma, tag or feature set), a tag/reading-manipulating action (e.g. REMOVE, SELECT, MAP, ADD, SUBSTITUTE, SETRELATION), and a list of one or more context conditions to be fulfilled in order for the action to be triggered. Context conditions may refer to words or features at an arbitrary distance in the sentence (or beyond), they may contain variables, regular expressions or numerical (e.g. frequency) conditions, and they can be negated, linked or conditioned as safe (referring only to unambiguous readings). Finally, unbounded conditions (i.e. without a distance position) may be "BARRIERed" by blocking features. The following (Spanish, French, German or Danish) rule, for instance, removes target readings that contain a finite verb tag (VFIN[2]) in favour of a noun reading (N) - IF an article (ART) or determiner (DET) is found to the left (*-1) with nothing but attributes (ALL - ATTR) inbetween (BARRIER), and if there is gender-number agreement ($$GN variable) between noun and article/determiner. As a safety measure, a second, negative (NOT) condition is present disallowing a noun phrase element (NPHR) at position 1 (i.e. immediately to the right of the target).

REMOVE VFIN IF (0 N + $$GN) (*-1C[3] ART OR DET BARRIER ALL - ATTR LINK 0 $$GN) (NOT 1 NPHR);

Since Constraint grammars are not data-driven in the statistical sense of the word, domain adaptation, for instance for speech (Bick 2012) or historical texts (Bick 2005), is traditionally achieved by extending an existing general grammar and/or its lexicon. However, due to its innate complexity, the general underlying grammar *as a whole* has properties that do not easily lend themselves to manual modification. Changes and extensions will usually be made at the level of individual rules, not rule interactions or rule regrouping, the effect of which is very difficult for a human grammarian to predict.

In particular, incremental contextual ambiguity reduction may activate dormant rules waiting for unambiguous context conditions to apply. Feed-back from corpus runs will pinpoint rules that make errors, and even allow to trace the effect on other rules applied later on the same sentence, but such debugging is cumbersome and will not provide information on missed-out positive, rather than negative, rule interaction. Therefore, optimization of rule-interaction (i.e. rule management at the grammar-level) is a major, and intrinsic, difficulty linked to the Constraint Grammar approach.

1 In the CG3 variant of the Constraint Grammar formalism, the introduction of variables and regular expressions makes it possible to perform morphological analysis within the grammar itself, but most systems only use this feature as a reserve method to handle out-of-vocabulary tokens.

2 VFIN, NPHR and ATTR are not actual tags, but examples of tag sets defined by the grammarian. Thus, VFIN comprises tense tags such as PR (present) or PAST, ATTR contains adjectives and participles, and NPHR amounts to all parts of speech allowed in a noun phrase chain (N, ADJ, ART, DET etc.).

3 The C means "unambiguous". With a C, the context would match also words that still have other readings on top of ART or DET.

In (Bick 2013), we have shown that machine learning techniques can be applied to monolingual grammar tuning, even with reduced grammars. Building on this research, we intend to show that ML is effective not only for optimizing grammars, but also for porting them from one language to another, a task that can be regarded as an extreme variant of domain adaptation. For the experiments presented here, we have ported the part-of-speech/ morphological section of an English grammar (EngGram) into Danish, using a CG-annotated section of the Danish treebank (Arboretum, Bick 2003) for training and evaluation.

2. Related work

To date, little work on CG rule tuning has been published. A notable exception is the µ-TBL system proposed in (Lager 1999), a transformation-based learner working with 4 different rule operators, and supporting not only traditional Brill-taggers but also Constraint Grammars. The system could be seeded with simple CG rule templates with conditions on numbered context positions, but for complexity reasons it did not support more advanced CG rules with unbounded, sentence-wide contexts, barrier conditions or linked contexts, all of which are common in hand-written Constraint Grammars. Therefore, while capable of building automatic grammars from rule templates and modeling them on a gold corpus, the system was not applicable to existing, linguist-designed CGs.

That automatic rule tuning can capture systematic differences between data sets, was shown by Rögnvaldsson (2002), who compared English and Icelandic µ-TBL grammars seeded with the same templates, finding that the system prioritized right context and longer-distance context templates more for English than Icelandic. For hand-written grammars, rather than template expression, a similar tuning effect can be expected by prioritizing/deprioritizing certain rule or context types by moving them to higher or lower rule sections, respectively, or by inactivating certain rules entirely.

Lindberg & Eineborg (1998) conducted a performance evaluation with a CG-learning Progol system on Swedish data from the Stockholm-Umeå corpus. With 7000 induced REMOVE rules, their system achieved a recall of 98%. An F-Score was not given, but since residual ambiguity was 1.13 readings per word (i.e. a precision of 98/113=86.7%), it can be estimated at 92%. Also, the lexicon was built from the corpus, so performance can be expected to be lower on lexically independent data.

Though all three of the above reports show that machine learning can be applied to CG-style grammars, none of them addresses the tuning of human-written, complete grammars rather than lists of rule templates[4]. In this paper we argue that this is possible, too, and that it can lead to better results than both automatic and human grammars seen in isolation. In particular we demonstrate that ML CG-tuning is useful to create seeding grammars for new languages from an existing template grammar in another language. Though such a ported seeding grammar cannot be expected to compete with mature taggers directly, it does provide a robust basis for human grammar creation, and will help to reduce overall development time for high-quality rule-based taggers for under-ressourced languages[5].

3. Grammar optimization and adaptation techniques

For our experiments, we divided the Danish gold corpus (70.800 tokens) randomly into 10 sections, using 90% for training and 10% for evaluation. Because of CPU runtime constraints, most parameter variations were tested with only one split, but for the best parameter combinations, 10-fold cross-validation was carried out with all 10 possible splits.

For each individual training run, the CG-compiler was run in trace mode, allowing the optimizer program to count how often each individual rule was used, and what its error percentage was. Based on these counts, the optimizer modified rule order or rule strictness (i.e. the degree of context ambiguity allowed for the rule in question).

3.1. Rule movement

The following rule movements were implemented:

- (a) promote good rules: moving a rule up one section, i.e. allowing it to be applied earlier, before other, more heuristic rules
- (b) demote dubious rules: moving a rule down one section, i.e. having it apply later, after other, safer rules
- (c) remove bad rules from the grammar, by moving them to a special "kill" section.

Bad rules were defined as rules that made more wrong than correct choice, while rules were regarded as good or dubious, if their error percentage was below or above certain empirically established error percentages (12.5 and

4 One author, Padró (1996), using CG-reminiscent constraints made up of close PoS contexts, envisioned a combination of automatically learned and linguistically learned rules for his relaxation labelling algorithm, but did not report any actual work on human-built grammars.

5 Danish is a small language, but not generally regarded as under-ressourced. It was chosen as target language only as a model, and because a gold corpus did not have to be constructed from scratch. In a production setting, both the creation of a gold corpus and manual completion of the ported grammar would be part of the work flow.

25, respectively[6]). Training runs were iterated to allow a rule to migrate up or down through the grammar's different heuristicity sections.

Of course it is also possible to reorder rules simultaneously rather than individually, by sorting rules according to a quality metric. However, experiments showed that it is difficult to achieve positive effects by sorting, at least when using actual rule error frequency as the only parameter, probably because this method does not really allow for the modeling of rule interaction - a rule that performs well may do so only because previous rules prevented it from making errors, which is why it needs to be tested individually and moved up incrementally rather than by sorting. Thus, sorting worked relatively best when human rule ordering was taken into account, by sorting sections individually and by weighting rule quality with a section factor. Only when each rule was run and evaluated in isolation, did sorting work well, albeit at a prohibitive time cost (20 hours, without iteration).

3.2. Rule strictness

Due to the complexity of CG rules, and the vast amount of possible context conditions, it is much more difficult to systematically vary, let alone create, rules than to move them. However, rules can be made more or less cautious without actually changing their context conditions, by adding or removing the so-called C-option for the individual context conditions. For instance, "*-1 VFIN BARRIER NON-ADJ/DET" is a context condition that looks left (*-1) until it finds a finite verb (VFIN), but can be blocked (BARRIER) if a non-prenominal (defined as something that is not an adjective or determiner) is found in between.

This condition can be made more cautions by using *-1C or less cautious by using CBARRIER. The former restricts the VFIN context to mean only unambiguous finite verbs, while the latter relaxes the blocking condition to words that unambiguously are something other than ADJ or DET. The optimizer uses C-relaxation for promoted (low-error) rules, and "C-stricting" for demoted (high-error) rules, either in situ or by cloning the rule with the changed C-condition. Obviously, relaxed clone-rules carry an added error risk, and are therefore added at the end of the grammar, to be moved up in later iterations if they perform well. A third relaxation technique was to clone a rule by moving its wordform condition, i.e. letting it apply in general rather than only for a certain problematic or high-frequency token.

3.3. Translation

An obvious problem for a ported grammar is that all token and lemma contexts, as well as word set definitions, are in the wrong language. We therefore wrote a machine-translation script that identified and translated English words occurring in rules or definitions. In addition, an effort was made to translate tags for valency potential, since these may contain references to prepositions and adverbs. As can be seen in section 4, this improved recall and decreased precision, with a combined F-score effect that was negative for the baseline run, but further improved optimized grammars. Apart from MT errors, the likely reason for this is that English and Danish are not "isomorphic" enough for this method to work automatically - there is simply no guarantee that ambiguity will reside in the same words, or that verbs bind the same prepositions. However, while we are only concerned with automatic tuning here, limited human effort would suffice to chose the correct analogues and improve performance.

3.4. Rule templates

Though the word class inventory of Danish is similar to that of English, there is no guarantee that an English grammar will contain rules addressing all disambiguation combinations that are relevant for Danish. Similarly, even simple bigram contexts may be absent if they are relevant only for Danish, and not for English. For instance, articles are unambiguous in English, but not in Danish, where they exhibit number and gender, and overlap with pronouns and numerals. We therefore added 1024 rule templates with ±1(C) contexts, for all possible PoS combinations, to the ported grammar to allow the ML system to fill in any coverage gaps.

4. Results

To establish a baseline, we ran a mini-grammar with only one rule (SELECT <fr=MAX>), choosing the most likely reading for each token, based on corpus frequency. While the raw English grammar[7] performed only marginally better than this baseline when run on Danish input (with a lower recall and a higher precision), it improved considerably when subjected to ML-optimization (error reduction amounting to 40% relative improvement). The table below shows results for various tuning options on top of rule killing (K), demoting (D), C-relaxation[8] (r) and C-stricting (s), with a 9:1 training-testing split for the gold corpus[9].

6 Raising the threshold improved recall, lowering it improved precision, but F-scores were lower in both cases.

7 The Danish input contained frequency information, and the English template grammar used the SELECT <fr=MAX> rule at the end, to remove any ambiguity remaining after the ordinary, context-based rules.

8 Separating C and BARRIER relaxation was also tried, with no marked difference. Adding relaxed rules in situ rather than at the end, led to F-scores below the baseline.

9 Results are test corpus performance at that iteration where training corpus performance stabilized or peaked. Slightly higher test F-scores may occur at later or earlier iterations, but using them to decide on the final grammar would make the test corpus part of the training setup.

	iteration	Recall (%)	Precision (%)	F-score
fMAX baseline	0	88.67	82.94	85.71
raw ported grammar	0	87.40	84.83	86.10
ported grammar with MT translations	0	90.65	81.37	85.76
DKrst, -fMAX	7	94.45	84.00	88.92
PDKrs, +fMAX	7	91.76	88.99	90.35
PDKrst, -fMAX	6	91.77	89.11	90.42
PDKrst, +fMAX	10	92.45	89.75	91.08
PDKrsw, +fMAX	22	93.57	88.83	91.14
PDKrstw, +fMAX	14	92.20	90.29	91.23
PDKrstw, pos-ified fMIN	30	92.69	90.32	91.49

Table 1: Cross-language grammar porting: PDK optimization plus variants

The data indicate that adding rule promoting (P), translation (t), frequency fail-safe (fMAX or fMIN) and wordform relaxation (w) each individually contributed to overall F-score improvement, though recall in isolation can be further maximized by adding only using translation to a simple DKrs combination. The top result was achieved with a more cautious frequency fail safe, where the lowest frequency reading was removed for each 15 PoS classes separately, rather than selecting the highest frequency reading, allowing the safest fMIN rules to migrate up through the grammar. Tenfold cross-evaluation confirmed this result, with an F-score increase of 5 percentage points over the baseline, corresponding to a 37% error reduction.

	R	dR	P	dP	F	dF
PDKrstw, pos-ified fMIN	92.65	4.94	90.46	5.16	91.54	5.05
PDKrstw, pos-ified fMAX	92.73	4.86	90.09	4.88	91.39	4.87

Table 2: 10-fold cross-validation

Finally, two additional techniques were tested - rule templates and individual rule evaluation. Both beat the ordinary optimization runs, but while the template run profited from PDK movements in an ordinary way, the individually sorted rules behaved more like a monolingual human grammar in that they did not tolerate promoting[10], while still allowing slight increases from rule-killing and demoting.

	iteration	Recall (%)	Precision (%)	F-score
individual rule sorting, +fMAX	1	92.92	90.55	91.76
individual rule sorting, + unused rules	1	92.88	90.74	91.80
individual rule sorting, +fMIN	1	93.04	90.69	91.85
PDKrstw, pos-ified fMIN, templates	5	93.11	91.35	92.22
individual rule sorting, +PDKrstw (i.e. no promoting), +fMAX,	18	93.68	91.14	92.39

Table 3: Rule templates and individual rule sorting

Of course, F-scores of around 92 do not approach state-of-the-art for PoS tagging, which for many languages can be as high as 96-97. However, it should be born in mind that our method has been developed in the framework of a rule-based, not a statistical parsing paradigm, and that the tuned grammar provides a reasonable and time-saving point of departure for the manual correction and addition of target-language rules. Furthermore, since the ML technique is language-independent, it could then be repeated in a second round of monolingual optimization. In order to estimate this potential, the same tuning technique was applied to a full-size, native Danish CG. In this experiment, a 10% error reduction was achieved even with an input grammar that performed at the 96% accuracy level.

	iteration	Recall (%)	Precision (%)	F-score
Danish grammar before optimization	0	97.65	95.44	96.54
PDKrsfw	1	97.61	95.77	96.68
PDKrsfw + fMAX	1	97.61	95.87	96.73
PDKrsfw + pos-ified fMIN	1	97.58	95.83	96.70
PDKrsfw + templates	1	97.58	95.83	96.70
DKrsfw (no promoting)	11	97.91	95.99	96.94
DKrsfw (no promoting)	14	97.91	95.99	96.94

10 Possibly because a globally established rule order cannot be improved by changes that are only measured locally

	iteration	Recall (%)	Precision (%)	F-score
+ fMAX				

Table 4: Monolingual results for a mature grammar (Danish)

As could be expected in a mature grammar, templates and additional statistical fail-safes had almost no effect compared to rule manipulation on its own (PDKrsfw). Interestingly, the DK-run was the only one with a sustained iteration gain, indicating that promoting low-error rules does more harm than good in a full grammar, possibly undoing beneficial effects from demoting.

5. Perspectives

Obviously, the grammar tuning achieved with the methods presented here does not represent an upper ceiling for performance increases. First, with more processing power, rule movements could be evaluated against the training corpus individually and in all possible permutations, rather than in-batch, eliminating the risk of negative rule-interaction from other simultaneously moved rules[11]. Second, multi-iteration runs showed an oscillating performance curve finally settling into a narrow band *below* the first maximum (usually achieved already in iteration 1 or 2, and never after 3). This raises the question of local/relative maxima, and should be further examined by making changes in smaller steps. Finally, while large scale rule reordering is difficult to perform for a human, the opposite is true of rule killing and rule changes such as adding or removing C-conditions. Rather than kill a rule outright or change *all* C-conditions in a given rule, a linguist might prefer to change or add individual context conditions to make the rule perform better, observing the effect on relevant sentences rather than indirectly through global test corpus performance measures. Future research should therefore explore possible trade-off gains resulting from the interaction between machine-learned and human-revised grammar changes.

6. References

Bick, E. (2003). Arboretum, a Hybrid Treebank for Danish, in: Joakim Nivre & Erhard Hinrich (eds.), *Proceedings of TLT 2003* (2nd Workshop on Treebanks and Linguistic Theory, Växjö, November 14-15, 2003), pp.9-20. Växjö University Press

Bick, E. & Módolo, M. (2005). Letters and Editorials: A grammatically annotated corpus of 19th century Brazilian Portuguese. In: Claus Pusch & Johannes Kabatek & Wolfgang Raible (eds.) *Romance Corpus Linguistics II: Corpora and Historical Linguistics* (Proceedings of the 2nd Freiburg Workshop on Romance Corpus Linguistics, Sept. 2003). pp. 271-280. Tübingen: Gunther Narr Verlag.

Bick, E., Mello, H., Panunzi, A. and Raso, T. (2012). The Annotation of the C-ORAL-Brasil through the Implementation of the Palavras Parser (2012). In: Calzolari, Nicoletta et al. (eds.), *Proceedings LREC2012* (Istanbul, May 23-25). pp. 3382-3386. ISBN 978-2-9517408-7-7

Karlsson, F., Voutilainen, A., Heikkilä, J. and Anttila, A. (1995). *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text.* Natural Language Processing, No 4. Mouton de Gruyter, Berlin and New York

Lager, T. (1999). The µ-TBL System: Logic Programming Tools for Transformation-Based Learning. In: *Proceedings of CoNLL'99*, Bergen.

Lindberg, N., Eineborg. M. (1998). Learning Constraint Grammar-style Disambiguation Rules using Inductive Logic Programming. *Proceedings of COLING-ACL 1998*: pp. 775-779

Padró, L. (1996). POS Tagging Using Relaxation Labelling. In: *Proceedings of the 16th International Conference on Computational Linguistics, COLING* (Copenhagen, Denmark). pp. 877--882.

Rögnvaldsson, E. (2002). The Icelandic µ-TBL Experiment: µ-TBL Rules for Icelandic Compared to English Rules. Retrieved 2013-05-12 from *[http://hi.academia.edu/EirikurRognvaldsson/Papers]*