Turning a Dependency Treebank into a PSG-style Constituent Treebank

Eckhard Bick
University of Southern Denmark
Institute of Language and Communication
E-mail: eckhard.bick@mail.dk

Abstract

In this paper, we present and evaluate a new method to convert Constraint Grammar (CG) parses of running text into Constituent Treebanks. The conversion is two-step - first a grammar-based method is used to bridge the gap between raw CG annotation and full dependency structure, then phrase structure bracketing and non-terminal nodes are introduced by clustering sister dependents, effectively building one syntactic treebank on top of another. The method is compared with another approach (Bick 2003-2), where constituent structures are arrived at by employing a function-tag based Phrase Structure Grammar (PSG). Results are evaluated on a small reference corpus for both raw and revised CG input, with bracketing F-Scores of 87.5% for raw text and 97.1% for revised CG input, and a raw text edge label accuracy of 95.9% for forms and 86% for functions, or 99.7% and 99.4%, respectively, for revised CG. By applying the tools to the CG-only part of the Danish Arboretum treebank we were able to increase the size of the treebank by 86%, from 197.400 to 367.500 words.

1.Introduction

Though syntactic treebanks are a valuable resource for both linguistic research and machine learning based HLT applications, their usefulness for the research community as a whole is potentially limited by the degree to which they subscribe to a specific linguistic theory, and as Nivre (2003) points out, supported format conversions should therefore be a guiding design principle. This is especially true of treebanks with an automatic parse as a first stage, where technology may influence descriptive issues, and where information richness may be traded against accuracy. A case in point is the depth of a treebank, where the potentially better accuracy of shallow methods has to be balanced against the need of added processing stages and hybrid systems. In descriptive terms, in can be difficult to reconcile the strengths and weaknesses of different syntactic theories. For example, coordination is notoriously easier to handle in a constituent bracketing approach than in a dependency grammar description, while the opposite appears to be true for discontinuity (non-projectivity), clefting and raising. For such descriptive information not to be lost (or, on the contrary, to newly be created) in a conversion from one format to the other, more than simple filtering and string manipulation is necessary. Rather, grammatical rules (learned or linguist-written) have to be incorporated into the converter program, either algorithmically or in the form of an external grammar file.

In the Danish Arboretum-treebank project (Bick 2003-1), a robust Constraint Grammar (CG) parser was chosen for basic automatic annotation (http://visl.sdu.dk/constrant_grammar.html), in order to support a dual perspective (of constituent bracketing and dependency theory) with ”neutral” base tags covering morphology and syntactic function. Since CG annotation is token based and uses an incremental rule system, it is fairly straightforward to add secondary tags to provide for new information that might be needed for a new format conversion or corpus annotation. For instance, semantically inspired theta-roles can be added in the same formalism. However, since both dependency and constituent stucture is implicit and underspecified in classical CG[1], a CG-annotated corpus constituts a very shallow treebank at best, and added tools, or manual revision, are necessary to create a full-depth treebank.

2.Format conversion as an integrated step in treebank building

So far, a 3-step method has been applied: (1) CG-output is manually corrected, (2) used as input to a specialised PSG (Bick 2003-2), (3) corrected once again at the constituent tree level, (4) converted into TIGER format and fed into a converter program to create TIGER dependency trees. But though this double revision process does reduce the error rate, ensures that descriptive problems from both formats get due attention and even exploits the PSG-stage as a kind of consistency checker between revisions, it does not fully exploit the robustness of the CG stage. Thus, in spite of considerable time and effort spent on the PSG rules, they still produce (partial) parse failures for over 20% of newspaper genre sentences (on average 20-25 words per sentence) even for corrected CG input, and up to 50% on raw text.

An alternative method, *cg2dep* (Bick 2005), bypassing the regular PSG stage (2 and 3), creates dependency trees directly from the CG-format, using a procedural system with a compiled grammar of sequential, linguist-written attachment rules. In the example rule, subjects attach to present or past tense (i.e. finite) verbs to the right (R). Tag

[1] In one approach, Tapanainen and Järvinen (1997) describe an integrated parsing formalism (Finite Dependency Grammar, FDG) implementing full dependency structure between words or Tesnière-style multi-word nuclei, but most CG parsers, including the ones used by the VISL project, have been optimised for what could be called ”robust shallowness”, i.e. a maximally safe disambiguation of part of speech and syntactic function, rather than deep/complete structure.

conditions can be added to both potential heads and daughters, as well as BARRIER, NOT and TRANS conditions to formulate distance and scope restrictions. Both the dependent, target and condition fields may be filled with sets rather than individual tags, words or lexemes.

@SUBJ> -> (PR, IMPF) IF (R)

Cg2dep achieves an attachment accuracy of 93% on raw text, and 98.7% after manual correction of the CG-stage. For both raw text and corrected CG-input, the percentage of complete and correct structures was higher for the direct cg2dep method than for dependency trees arrived at with an intermediate PSG (up from 58.4% to 64.8% and 75.1% to 90.4%, respectively).

3.Head-based vs. dependent-based constituent building

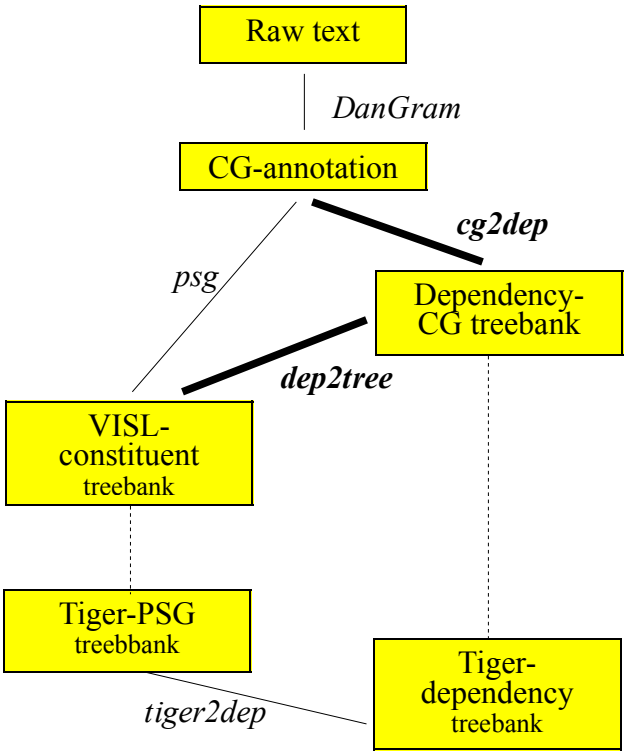

Illustration 1: Conversion paths and grammars
full arrow = grammar/context based system
hyphenated arrow = simple converter

The next stage, *dep2tree,* is a procedural program that creates bracketing from compiled head-daughter-dependencies. For instance, a noun phrase bracket is assembled by identifying either a safe np-head (e.g. noun) or a safe np-dependent (e.g. article), and by extracting all related pre- and postnominal dependency links (modifiers, relative clauses, complements), while raising the head's function to the newly created non-terminal (edge label). 11 different non-terminal form labels were created, mainly from head types, but rules could override this for functional reasons (i.e. adjectives as head of a subject np).

fcl	finite clause	np	noun phrase
icl	non-finite clause	vp	verb chain
acl	averbal (elliptic) clause	pp	prepositional phrase
par	paratagma	adjp	adjective phrase
x	underspecified (e.g. predicate)	advp	adverb phrase
		cp	conjunction phrase

Table 1: form labels

For illustration purposes samples of the two treebank formats[2] are provided below, first the dependency output (a) of *cg2dep,* then the constituent tree output (b) of *dep2tree.*To increase readability, some tags have been removed, such as lexeme/baseform as well as some inflexional and many secondary semantic tags.

(a) CG-dependency treebank notation:

Bagefter *(Afterwards)* ADV @ADVL> #1->11
blev *(was)* <aux> V IMPF AKT @FS-STA #2->0
han *(he)* PERS UTR 3S NOM @<SUBJ #3->2
af *(by)* PRP @<ADVL #4->11
både *(both)* ADV @FOC> #5->7
Peter=Duetoft <cjt-head> <hum> PROP @P< #6->4
og *(and)* <co-prparg> KC @CO #7->6
SF's <party> PROP @>N #8->9
ordfører *(spokesman)* <cjt> N UTR S IDF @P< #9->6
Steen=Gade <hum> <np-close> PROP @N< #10->9
kritiseret *(criticized)* <mv> V PCP2 STA @ICL-AUX< #11->2
for *(for)* PRP @<ADVL #12->11
$, #13->0
at *(to)* INFM @INFM #14->16
have *(have)* <aux> V INF AKT @ICL-P< #15->12
givet *(given)* <mv> V PCP2 AKT @ICL-AUX< #16->15
udvalget *(the committee)* N NEU S DEF @<DAT #17->16
materialet *(the material)* N NEU S DEF @<ACC #18->16
alt=for *(far too)* <aquant> ADV @>A #19->20
sent *(late)* ADV @<ADVL #20->16
$. #21->0

(with: ADVL=adverbial, ACC=accusative object, DAT=dative object, FS=finite clause, STA=statement, SUBJ=subject, FOC=focus marker, P<=argument of preposition, CO=coordinator, >N=prenominal, N<=postnominal, >A=adverbial pre-adject, AUX<=arugment of auxiliary, INFM=infinitive marker)

(b) Constituent tree format:

STA:fcl
fA:adv(<atemp>) Bagefter *(Afterwards)*
P:vp-
=Vaux:v-fin(IMPF AKT) blev *(was)*
S:pron-pers(UTR 3S NOM) han *(he)*
fA:pp

[2] No example of the original CG format is given, since apart from the added #x->y attachment tags, it is identical to the dependency format.

=H:prp af *(of)*
=DP:par
==FOC:adv både *(both)*
==CJT:prop(<hum> NOM) Peter_Duetoft
==CO:conj-c(<co-prparg>) og *(and)*
==CJT:np
===DN:prop(<party> GEN) SF's
===H:n("ordfører" UTR S IDF) ordfører *(spokesman)*
===DN:prop(<hum> NOM) Steen_Gade
-P:vp
=Vm:v-pcp2(STA) kritiseret *(criticized)*
fA:pp
=H:prp for *(for)*
=,
=DP:icl
==P:vp
===INFM:infm at *(to)*
===Vaux:v-inf(AKT) have *(have)*
===Vm:v-pcp2(AKT) givet *(given)*
==Oi:n(NEU S DEF NOM) udvalget *(the committee)*
==Od:n(NEU S DEF NOM) materialet *(the material)*
==fA:advp
===DA:adv(<aquant>) alt_for *(far too)*
===H:adv sent *(late)*
.

(with: STA=statement, S=subject, Od=direct object, Oi=dative object, P=predicator, Vm=main verb, Vaux=auxiliary, fA=free adverbial, H=head, DP=argument of preposition, DN=prenominal, DA=adverbial adject, CJT=conjunct, CO=coordinator, INFM=infinitive marker, FOC=focus marker)

To satisfy the target format (VISL, cp. http://visl.sdu.dk), dep2tree had to assign a number of non-trivial bracket types, among them co-ordination and small vp's (verb chains) as separate node-levels (cp. b), as well as discontinuous brackets due to gapped/fronted constituents, and elliptic brackets with underspecified edge labels (e.g. object or subject sharing in co-ordinated clauses). In part, this advanced bracketing was only possible, because conversion-supporting tags had been introduced at the CG-level. For instance, co-ordinators were marked for what they co-ordinate (e.g. <co-subj> for subject-coordination) and special vp-tags allowed to make a distinction between a coordination of the verb chain only (a1), a coordination of predicates (a3) or a coordination of finite clauses (a4) - all three of which involve otherwise identical dependency links from one finite verb onto another.

a1) *Denne opfattelse var og er meget udbredt* (this opinion was and is very common)
a2) *nye idéer kan gro frem og skabe efterspurgte produkter* (new ideas can grow and create desired products)
a3) *Adriana arbejder som nøgenmodel for kunstnere og syr skjorter* (Adriana works as an act model for artists and sews shirts)
a4) *... at FN's generalsekretær støtter idéen og at et barn fra hvert af medlemslandene vil ...* (... that the UN general secretary supports the idea and that a child from each of the member countries will ...)

In some cases coordination even forced us to consider undefined or novel constituent form types , as in (a2) where one auxiliary governs two coordinated non-finite arguments. While descriptively unproblematic in CG and dependency grammar, this construction breaks up the "little vp" constituent and creates a separate (coordinated) "argument of auxiliary" constituent (Oaux) as well as an isolated auxiliary predicator (Paux) not provided for in the original VISL system.

4. Evaluation

To evaluate the performance of the *dep2tree* compiler as a second stage to the *cg2dep* grammar compiler, as well as the coverage of the system's grammar, a small random text sample was extracted from Korpus90[3], consisting of 1497 words (1723 tokens, 122 sentences) of news text. A gold-standard corrected annotation was built for both the CG and constituent tree levels, yielding 729 non-terminal chunks. In the constituent format, crossing branches (non-projective dependencies) were expressed as discontinuous constituents, and their individual parts counted as separate non-terminals, resulting in a double penalty for crossing branch errors.

In a complete run on raw text, where almost complete disambiguation was forced, the combination of DanGram and the *cg2dep* and *dep2tree* stages achieved an F-score of 87.54 for matching chunks (table 2). Of these, 4.13% had a wrong form assigned in their edge label, while 13.99% had wrong functions (table 3). In a corresponding run on corrected CG-input, cg2dep + dep2tree achieved an F-score of 97.1% (table 2), with 0.28% wrong forms and 0.57% wrong functions (table 3). The original Arboretum-method, a phrase structure grammar with CG function tags as ”terminals”, had a considerably lower chunking F-score (89.4%, table 2), and correspondingly lower edge label accuracy (table 3) .

correct matches for chunks ...	***recall***	***precision***	***F-score***
in complete run (DanGram-CG + cg2dep + dep2tree)	86.3 %	88.8 %	87.5 %
with revised CG-input (cg2dep + dep2tree)	96.0 %	98.2 %	97.1 %
with revised CG-input (PSG-grammar)	88.3 %	90.6 %	89.4 %

Table 2: Bracketing accuracy.

edge label accuracy for matching chunks	edge label forms	edge label functions
complete run	95.9 %	86.0 %
revised CG-input + cg2dep + dep2tree	99.7 %	99.4 %
revised CG-input + PSG	98.1 %	92.2 %

Table 3: Edge label accuracy

[3] Korpus90 is part of a Danish corpus compiled by DSL for lexicographical work (www.dsl.dk), and constitutes one half of the Korpus90/2000 project (www.korpus2000.dk and corp.hum.sdu.dk).

Given the fact that the bracketing error rate of the combined constituent tree builder was considerably higher than that of the dependency stage in isolation (chapter 2), 12.5% versus 7% for raw text, or 2.9% versus 1.3% for corrected CG, further improvement and debugging of the *dep2tree* formalism should be attempted in the future.

5.The treebank

The Danish Arboretum treebank has currently 2 sections in different stages of completeness: One (a) where both the CG annotation and CG-derived constituent trees have undergone manual revision, and - due to funding constraints - another one (b) where only a basic revised CG version exists, without full syntactic trees. Using an automatic intermediate dependency stage, the method described in this paper was subsequently applied to section (b), enlarging the constituent treebank by about 85% (46% of the total, table 4). The creation of constituent tree structures for (b) also allowed export of the treebankdata to the PENN treebank and standard TIGER treebank formats, as well as ensuring compatibility with the unix treebank search tool t-grep2.

	sentences (words pr. sent.)	tokens (punctu-ation)	words
(a) revised CG + revised constituent treebank	12.003 (16.4)	227.444 (30.013)	197.431
(b) revised CG + dep2tree- generated constituent treebank	10.244 (16.5)	194.570 (25.460)	169.110
both parts	22.247 (16.5)	422.014 (45.473)	366.541

Table 4: Treebank section sizes

Given the higher accuracy of the dependency based method, we foresee a correspondingly more time-economical revision stage (for the new part of the treebank) than with the traditional method based on PSG rules. However, it is not clear, nor even likely, that the reduction in errors should be uniformly distributed across different grammatical categories and structures. Therefore, a qualitative error reduction analysis should be undertaken in order to focus future revision work more effectively. For instance, error patterns could be flagged for human revision by identifying overlaps and differences between the treebank results of the two alternative methods.

6.Conclusion

Departing from raw or hand-corrected Constraint Grammar input, we presented a new method of creating constituent tree structures from an automatic dependency annotation. The method achieved better results than a classical phrase structure approach, in terms of both bracketing and edge label accuracy. Edge label forms (np, vp, pp etc.) had a higher accuracy than edge label functions (subject, adverbial etc.) in all runs, possibly reflecting the close relation between lower level (PoS) tags and form labels.

The method was robust enough to be of practical value for treebank work, allowing the conversion of the CG-only part of the Danish Arboretum treebank into constituent tree format. On the other hand, bracketing error rates remain considerably higher than the underlying dependency error rates, indicating a need for further refinement of the formalism, possibly by adding further structural tags already at the CG-level (e.g. ellipsis markers, clause boundary markers or more complex coordination markers).

7.Bibliographical references

Bick, Eckhard (2001). ”En Constraint Grammar Parser for Dansk”. In: Widell, Peter & Kunøe, Mette (ed.): *8. Møde om Udforskningen af Dansk Sprog.* Århus: Århus Universitet 2001.

Bick, Eckhard (2003-1), “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, Eckhard (2003-2), “A CG & PSG Hybrid Approach to Automatic Corpus Annotation”. In: Kiril Simow & Petya Osenova (eds.), *Proceedings of SProLaC2003* (at Corpus Linguistics 2003, Lancaster), pp. 1-12

Bick, Eckhard (2005), "Turning Constraint Grammar Data into Running Dependency Treebanks". In: Montserrat Civit, Sandra Kübler & Ma. Antónia Martí (eds.), *Proceedings of the 4th Workshop on Treebanks and Linbuistic Theories, Barcelona, December 9-10, 2005),* pp.19-27. Universitat de Barcelona

Nivre, Joakim (2003), “Theory-Supporting Treebanks”. In: Joakim Nivre & Erhard Hinrich (eds.), *Proceedings of TLT 2003 (2nd Workshop on Treebanks and Linguistic Theory, Växjö, November 14-15, 2003)*, pp.117-128. Växjö University Press

Tapanainen, Pasi and Timo Järvinen. (1997). ”A non-projective dependency parser”. In: *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71, Washington, D.C., April. Association for Computational Linguistics.