The Danish Dependency Treebank and the DTAG Treebank Tool

Matthias Trautner Kromann
Center for Computational Modelling of Language
Department of Computational Linguistics
Copenhagen Business School
http://www.id.cbs.dk/∼mtk

1 Introduction

The Danish Dependency Treebank (DDT) is a dependency treebank with 5,540 sentences totalling 100,200 tokens. DDT consists of 536 randomly selected texts from the morphosyntactically tagged Danish PAROLE corpus [5].[1] The principles behind DDT are described in an 110 page annotation manual draft [7]. A research assistant has created the annotations manually and verified them once (at a rate of 1,000 words/day), and approximately 25% of the texts have been double-checked by a linguist. The treebank is freely available for research and can be obtained from the author.

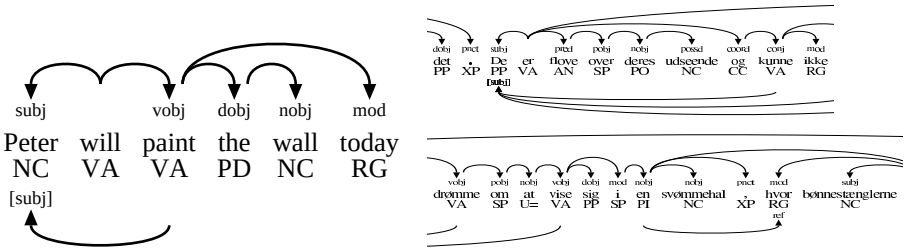
2 The DTAG treebank tool

The treebank has been created with the DTAG treebank tool [6]. DTAG can handle any syntax graph consisting of nodes connected by directed labeled edges. The nodes may correspond to words, phrases, or fillers (phonetically empty words), and may have any number of associated attributes for encoding word class, lemma, etc. The labeled edges may form any graph structure, including trees, discontinuous graphs, cyclic graphs, and graphs where nodes may have any number of in-coming or out-going edges (thus, DTAG allows a wider range of graphs than most other treebank tools, including the tools in the TIGER project [9]).

DTAG displays syntax graphs continuously by generating a PostScript file which is viewed by a PostScript viewer. Nodes are drawn in linear order, with

[1] PAROLE-DK is a balanced corpus of written text consisting of 1,553 text samples, each containing 150–250 tokens, which cover a wide range of different text mediums, genres, and topics.

different attributes shown on separate lines. Directed edges are drawn above or below the nodes, with the edge label shown at the arrow head (see graph below left). DTAG can print graphs corresponding to entire texts by splitting the graphs into several lines and pages (see excerpt below right). DTAG can also be used to print multi-speaker dialogue and discourse annotations. The precise formatting (colors, dashes, placement of edges and attributes, etc.) can be configured by the user.



DTAG is command-line based. For example, to specify a directed edge from node 34 to node 45 with label "subj", the user must type the command "34 subj 45" (DTAG allows any string without white-space to be used as a label). DTAG also has commands for visualizing the differences between two graphs, and for performing search-and-replace within all texts in the treebank. A search is specified as a constraint on a set of node variables. Simple constraints specify conditions on a node's associated attributes, or the relative node order or connecting edges of two nodes, and can be combined into complex constraints by the logical operators 'and', 'or', 'not', 'exist', and 'all' (the use of 'exist' and 'all' means that DTAG's query language is slightly more powerful than TIGERSearch [9]). For example, the following DTAG command is used to find all instances in the treebank where a verb $V has a "der" expletive $E, but no direct object $D:

```
find -corpus ($E expl $V) & ($E[lemma] =~ /^der$/)
    & !  exist($D, $D dobj $V)
```

The entire set of DTAG commands is described in [6]. The file format used by DTAG is described in [7]. We are currently working on extending DTAG with a parser and a grammar learner, so that DTAG can generate a probabilistic dependency lexicon from a treebank and use it for annotating text automatically.

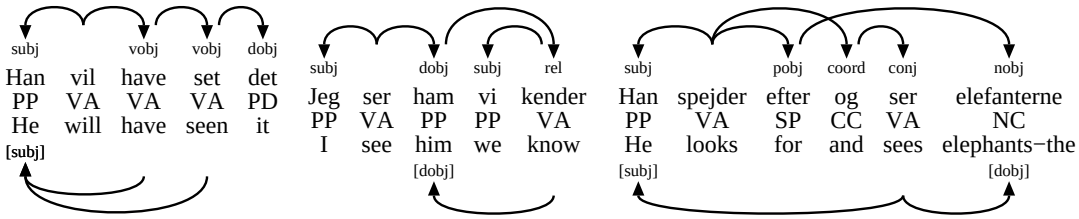
3 Linguistic analyses and comparison with other treebanks

The DDT analyses are based on Discontinuous Grammar [7] [8], a computational dependency-based syntax formalism closely related to Word Grammar [4]. The

analyses encode primary dependencies (complements and adjuncts), as well as secondary filler dependencies (where a word fills more than one dependency) and syntactically determined coreference (in relative clauses, resumptive pronouns, etc.). Secondary dependencies are necessary for recovering the underlying functor-argument structure from the annotations. The most important edge labels for primary dependencies in DDT are shown below (cf. [7]).

Complement edges		**Adjunct edges**	
aobj	adjectival object	**appa**	parenthetical apposition
avobj	adverbial object	**appr**	restrictive apposition
conj	conjunct of coordinator	**coord**	coordination
dobj	direct object	**list**	unanalyzed sequence
expl	expletive subject	**mod**	modifier
iobj	indirect object	**modo**	dobj-oriented modifier
lobj	locative-directional object	**modp**	parenthetical modifier
nobj	nominal object	**modr**	restrictive modifier
numa	additive numeral	**mods**	subject-oriented modifier
numm	multiplicative numeral	**name**	additional proper name
part	verbal particle	**namef**	additional first name
pobj	prepositional object	**namel**	additional last name
possd	possessed in genitives	**pnct**	punctuation modifier
pred	subject or object predicate	**rel**	relative clause
qobj	quotation object	**title**	title of person
subj	subject	**xpl**	explication (colon)
vobj	verbal object		

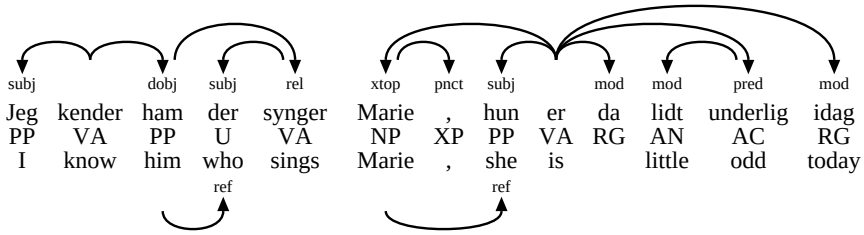
In DDT, multiple dependencies are encoded by selecting one dependency as primary, and the other dependencies as secondary. Secondary dependencies are encoded with filler edges, which can be recognized by the square brackets surrounding the edge label. Penn Treebank II [10] also encodes these filler dependencies, but most other treebanks leave them unspecified, including the Prague Dependency Treebank [2], the Tiger/Negra treebank [3] (except in coordinations), and the Danish Arboretum [1]. Some filler constructions in DDT are shown below:



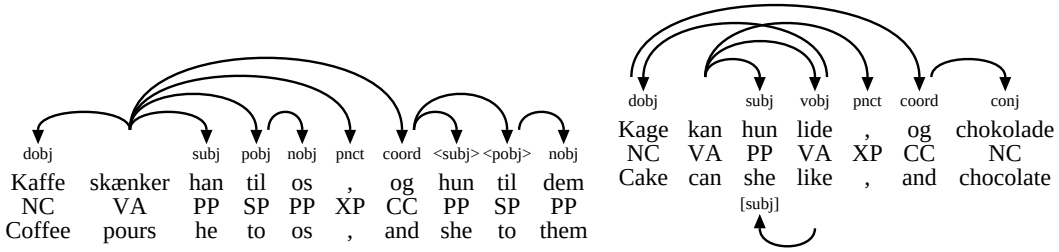
In the left example, the finite verb "will" receives the primary subject dependency, whereas the two infinite verbs receive a secondary filler subject dependency. In the middle example, the relativized NP "him" functions as primary direct object of the main verb "see" and as secondary direct object of the relative verb "know". In the

right example, the two conjuncts share the initial NP (as subject) and the final NP (as nominal object and direct object, respectively).

Syntactically determined coreference is encoded by a "ref" edge from the antecedent to the anaphora. In the example below left, the relative pronoun is coreferent with the relativized NP. In the example below right, the resumptive pronoun is coreferent with the external topic. Coreference is also encoded in PT-II [10], but not in PDT [2], Tiger [3], or Arboretum [1].



Gapping edges represent dependencies between a word and an elided head, and are marked by angular brackets around the edge label. In the gapping coordination below left, the dependents of the elided head in second conjunct have been encoded as gapping dependents of the coordinator. The same analysis is found in PT-II [10] and PDT [2], but apparently not in Tiger [3], or Arboretum [1].[2]



4 Conclusion

DDT is a purely dependency-based treebank with relatively deep annotations which include secondary dependencies, like PT-II [10], but unlike PDT [2], Tiger [3], and Arboretum [1]. Our DTAG annotation tool works for graphs with cyclicity, discontinuity, and multiple dependencies. DTAG's treebank search engine includes the powerful 'exist' and 'all' operations, unlike TIGERSearch [9].

[2] PDT [2] and Tiger [3] analyze the coordinator as the head of the coordination. However, since the coordinator and the second conjunct form a unit in discontinuous coordinations, our analysis of the first conjunct as the head seems preferable. Unlike DDT, Word Grammar [4] does not use a dependency-based analysis.

References

[1] E. Bick. A CG & PSG Hybrid Approach to Automatic Corpus Annotation. In K. Simow and P. Osenova, editors, *Proceedings of SProLaC2003 (at Corpus Linguistics 2003, Lancaster)*, pages 1–12, 2003.

[2] A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká. The Prague Dependency Treebank: Three-Level Annotation Scenario. In A. Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers, 2001.

[3] S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. The TIGER Treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories, Sozopol*, 2002.

[4] R. Hudson. An encyclopedia of English grammar and Word Grammar. http://www.phon.ucl.ac.uk/home/dick/enc-gen.htm, 2003.

[5] B. Keson and O. Norling-Christensen. PAROLE-DK. Det Danske Sprog- og Litteraturselskab. http://korpus.dsl.dk/e-resurser/parole-korpus.php, 1998.

[6] M. Kromann. DTAG treebank tool. http://www.id.cbs.dk/∼mtk/dtag, 2003.

[7] M. Kromann, L. Mikkelsen, and S. Lynge. Danish Dependency Treebank. Annotation manual. Dept. of Comp. Ling., Copenhagen Business School. http://www.id.cbs.dk/∼mtk/treebank, 2003.

[8] M. T. Kromann. Optimality parsing and local cost functions in Discontinuous Grammar. In *Proceedings of the Joint Conference on Formal Grammar and Mathematics of Language (FGMOL-01), Helsinki, August 10-12, 2001. Electronic Notes in Theoretical Computer Science 53*, 2001. http://www.helsinki.fi /esslli/courses/readers/fgmol.

[9] E. König, W. Lezius, and H. Voormann. TIGERSearch 2.1 User's Manual. IMS, Univ. of Stuttgart. http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch, 2003.

[10] M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. The Penn treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*, 1994.