Building a Syntactically Annotated Corpus:

The Prague Dependency Treebank

Jan Hajič

ÚFAL MFF UK
Charles University
Malostranské nám. 25
CZ-11800 Prague 1
Czech Republic
hajic@ufal.mff.cuni.cz

1 Introduction

In present-day computational linguistics (CL), availability of annotated data (spoken utterances, written texts) is becoming a more and more important factor in any new developments. Apart from speech recognition, where statistical methods are almost exclusively *the* solution and where the data is a *conditio sine qua non*, textual data are being used for so-called training phase of various statistical methods solving many other problems in the field of CL. While there are many methods which use texts in their plain (or raw) form (for so-called unsupervised training), more accurate results may be obtained if annotated corpora are available.

With the increasing complexity of such tasks, the data annotation itself is a complex task. While tagged corpora (pioneered by Henry Kučera in the 60's) are now available for English and other languages, syntactically annotated corpora are rare. Inspired by the most widely used syntactically annotated corpus of English, the Penn Treebank (now becoming ready to be distributed in its Third edition by the Linguistic Data Consortium), we decided to develop[1] a similarly sized corpus of Czech with rich annotation scheme.

We have been in a very good position because as one of the leading personalities of the research team working on this project we have Jarmila Panevová, whose life-long intensive research in Czech Syntax both from the tectogrammatical and syntactico-semantic aspects combining a subtle empirical analysis with a rigorous explicit description (see e.g. Panevová 1971, 1973, 1974, 1975a, 1975b, 1978, 1979, 1991, 1994, and especially Panevová 1980) has been guiding our steps all the time.

The textual data used for the task contain general newspaper articles (40%; including but not limited to politics, sports, culture, hobby, etc.), economic news and analyses (20%), popular science magazine (20%), and information technology texts (20%), all selected from the Czech National Treebank.

2 The Prague Dependency Treebank Structure

The Prague Dependency Treebank (PDT) has a three-level structure. Full morphological tagging is done on the lowest level. The middle level, which is the core level of the current project, deals with syntactic annotation using dependency syntax; it is called the analytical level. The highest level of annotation is the

[1] The project was started by support from the grant GAČR No. 405/96/0198 (Formal specification of language structures), and the annotation effort has been made possible by the grant GAČR No. 405/96/K214 and by the project of the Ministry of Education of the Czech Republic No. VS96151. The development of some software tools used in this project has been supported by the grant GAČR No. 405/95/0190 and by the individual author's grant OSF RSS/HESP 1996/195.

tectogrammatical level, or the level of linguistic meaning. We annotate the same text on all three levels, but the amount of annotated material decreases with the complexity of the levels.

2.1 The Morphological Level

On the morphological level, a tag and a lemma is assigned to each word form as found in the input text. The annotation contains no (syntactic) structure; no attempt is even made to put together analytical verb forms, for example.

2.1.1 The tag system

Czech is an inflectionally rich language. The full tag set contains currently 3030 tags (including morphological variants, which are being distinguished). In order to save space, we present here their list in an abbreviated form using so-called tag variables, which represent categories which may take all possible values from a given set. These variables are:

Category	Variable	Value	Description
number	n	S	singular
		P	plural
		D	dual (only for feminine, instrumental case)
		X	both, or special combinations
case	c	1	nominative
		2	genitive
		3	dative
		4	accusative
		5	vocative
		6	locative (always prepositional)
		7	instrumental
		X	any
gender	g	M	masc. animate
		I	masc. inanimate
		F	feminine
		N	neuter
		X	any of the above
		Y	M or I (masculine)
		H	F or N(not masculine)
		Q	F or N (not masculine), but in special combinations only
		T	I or F (masc. inanimate or feminine)
		Z	M, I or N (not feminine)
		W	I or N (not masculine animate, not feminine)
degrees of comparison	d	1	base form (positive)
		2	comparative
		3	superlative
person	p,f	1	1st
		2	2nd
		3	3rd (p only)
		X	any (p only)
negation	a	A	affirmative
		N	negated form

The list of all possible tags follows. Bracketed part in the first column is optional.

Tag	Description
VFa	verb: infinitive
VMnpa	verb: imperative (only: VMS[23]a, VMP[123]a possible, 3rd pers.arch.,-2)
VPnpa	verb: indicative, present tense (*pracujeme*)
VUnpa	verb: indicative, future tense (where applicable; e.g. *být, pojedu*) but not perfective present
VPEnpa	verb: indicative, present tense, with -t' (contraction) (archaic, ex. *pracujut'*)
VUEnpa	verb: indicative, future tense, with -t' (contraction) (archaic, ex. *půjdut'*)
VRgna	verb: past participle, active mood (*dělal*): here only: VRYSa, VRQXa (for 'VRFSa' or 'VRNPa'), VRNSa, VRMPa, VRTPa
VREgna	verb: past participle, active mood + -t' (*bylt'*): here only: VREYSa, VREQXa (for 'VREFSa' or 'VRENPa'), VRENSa, VREMPa, VRETPa
VRAgnpa	verb: past tense, with contracted -s (jsi, ex. *byls*), only: VRAYS2a, VRAFS2a, VRANS2a
VSgn[c]a	verb: past participle, passive mood (*udělán*): here only: VSYSa, VSQXa (for 'VSFSa' or 'VSNPa' or 'VSYS4a'), VSNSa, VSMPa, VSTPa, VSFS4a (*udělánu*)
VSAgnpa	verb: past participle, passive mood, with contracted -s (jsi, ex. *dělánas*), only: VSAFS2a, VSANS2a
VGnga	verb: transgressive present (only: VGSYa, VGSHa, VGPXa)
VVnga	verb: transgressive past (only: VVSYa, VVSHa, VVPXa)
VCnp	verb: conditional (*být* only); in 3rd pers only VCX3
Ngnca	nouns (but only patterns kt1n, stn and irregular nouns can have neg. N, otherwise A)
Agncda	adjectives
AVGgnca	adjectives, verbal from present transgressive (*dělající*)
AVVgnca	adjectives, verbal from past transgressive (*dodělavší*)
A1gn	adjectives (*svůj-2/nesvůj-2/tentam*) used as adj. (*být nesvůj*)
ASMgnc	adjectives possessive (to masc.) (*-ův*)
ASFgnc	adjectives possessive (to fem.) (*-in*)
ACgn[c]a	adjectives, short forms (used also as if verbal pass. part.)ex. *rád, zdráv*; only: ACYSa, ACQXa (for 'ACFSa' or 'ACNPa' or 'ACYS4a'), ACNSa, ACMPa, ACTPa, ACFS4a
PPfnc	pronoun personal I/you (two lemmas: *já/ty*)
PPfCnc	pronoun personal I/you (two lemmas: *já/ty*) short form (S[234] only)
PP3gnc	pronoun personal he/she/it (one lemma: *on*) standalone (S[2347] only)
PP3Cgnc	pronoun personal he/she/it (one lemma: *on*) short form (S[234] only)
PP3Rgnc	pronoun personal he/she/it (one lemma: *on*) after prep. (*něho, ...*)
PPD	pronoun personal (he) with prep (*naň, proň, doň, ...*)
PPA2S1	tys
PDgnc	pronoun dem. this/that (lemma: *ten, tento, tamten, ...*)
PRnc	pronoun reflexive *se*, c from only [23467], n always X
PRCnc	pronoun reflexive *se*, short form (only [34])), n always X (*si, se*)
PRACncp	pronoun reflexive *se*, short form (only [34])), n always S + contracted -s (*jsi*, 2nd pers. of *být*): *ses, sis*; p == 2
PSfngnc	pronoun possessive my/your (lemma *můj/tvůj*) first number: 'inner' number (*můj/náš* (lemma *můj*), *tvůj/váš* (*tvůj*)) no inner gender! short forms by tag suffix -1 etc. plural gender always X; sg gender mainly by combinations
PS3gngnc	pronoun possessive 3rd pers (his/her/its/their), lemma: *jeho* first gn: inner gender/number; second gn: object gender/number; only some combinations used: PS3ZSXXX *jeho* (lemma: *jeho*)

Tag	Description
	PS3FSgnc *její* in sg. (lemma: *jeho*), but only:gnc=XPc FSX ZS[123567] NS4 IS4 MS4 PS3XPXXX *jejich* (lemma: *jeho*)
PSEgngnc	pronoun rel. (from poss. his/her/its/their), lemma: *jehož*; first gn: inner gender/number; second gn: object gender/number; only some combinations used: PSEZSXXX *jehož* (lemma: *jehož*) PSEFSgnc *jejíž* in sg. (lemma: *jehož*), but only:gnc=XPc FSX ZS[123567] NS4 IS4 MS4 PSEXPXXX *jejichž* (lemma: *jehož*)
PRSgnc	pronoun reflexive possessive *svůj* NB: short forms by -1 etc.
PLgnc	pronoun self (*sám*), all (*všechen*) NB: short forms by -1 etc.
PAEgnc	pronoun rel. or adjectival (other) (*jenž*)
PAERgnc	pronoun rel. *jehož* after prep. (*něhož, němuž, ...*) after prep. (see PP3R...)
PEc	pronoun rel. genderless/numberless (*což-1*)
PQFgnc	pronoun quest. or rel. adj. (other) (which, what—*jaký, který, co*)
PQKgc	pronoun quest. or rel. (*kdo, kdož*), gender M only!
PQCc	pronoun quest. or rel. (*co*), genderless
PQD	pronoun quest. or rel. with prep. (*nač, oč*)
PQAKcp	pronoun quest. rel. + *jsi* (*s*) (*kdos,cos*), p == 2 always
PIFgnc	pronoun indef. (*nějaký*)
PIKgc	pronoun indef. (*kdo*), gender M only!
PICc	pronoun indef. (*co*), genderless
PNFgnc	pronoun neg. (*nijaký, ...žádný*)
PNKc	pronoun neg. genderless/numberless indef. (*nikdo*)
PNCc	pronoun neg. genderless/numberless indef. (*nic, pranic*)
CRgnc	numerals: ordinal
CDgnc	numerals: generic (*-erý*), synt. adjective, also *dvojí, trojí*
CD1gnc	numerals: generic, have gender: *jedny, nejedny*
CD2nc	numerals: generic, (but: plural only, CD2Pc) (in endings!); counted noun must agree in c and plural number
CDJnc	numerals: generic, short forms (~ *město*) (but: singular only, CDJSc); counted noun in genitive plural! (in endings)
CFgnc	numerals: fractions (but: only feminine gender, CFFnc)
CBnc	numerals, basic: 5,6,7,8,9,10-20,30,40,50,60,70,80,90 n = S only also *sto-1, tisíc-1* (num.; *sto-2, tisíc-2, milión* noun)
CGgnc	numerals, basic: 1,2 (have gender); 1 - n = S only, 2 - n = P only,3,4 - n = P only and gender always X exc. FD also *sto-1, tisíc-1* (num. CGXnc; case not 1,4,5: *sto-2, tisíc-2, milión* noun) *půl-1, čtvrt-1* indecl. num. (CGXSX), (*půl-2, čtvrt-2* n.) case: usually only 2367
CX	numerals: roman
CQFgnc	numerals: que or rel w/gender/num (*kolikátý*)
CQc	numerals: quest. (*kolik*)
CIFgnc	numerals: indef. with gender/num (*nejeden*, S only (P see CD1), *několikátý, tolikátý*)
CIc	numerals: indef. (*tolik*, also *moc* and other "adv.") [1234567X]
CM	numeral: multiplicative (*-krát*), synt. adverb, number-*krát*
CIM	numeral: multiplicative (*-krát*), synt. adverb, indef. (*tolikrát*)
CQM	numeral: multiplicative (*-krát*), synt. adverb, quest. (*kolikrát*)
DB	adverbs, base form only (no neg. no grad.)
DGda	adverbs, with neg/grad (at least possible neg./grad.) DGda are the most common – derived from adj. etc.
Rc	prepositions

Tag	Description
RF	prepositions as first part of 'compound prep.' (*vzhledem*)
RVc	prepositions with vocalization (*-e*; also *-u*, distinguished by -2)
JE	conjunctions (coordinating, same level coordination)
JS	conjunctions (subordinating)
JC	conjunctions (coord.; numeric operators *krát, plus, minus, děleno*)
JVnp	conjunction "to/when": condit. form ("*aby/kdyby*"); 3rd pers: JVX3
T	particles (*ano, ne, ...*)
I	interjections (*hurá, hajdy*)
HYPH	hyphenated (first part of hyphenated compound, usually from adj.)
ABBRN	noun abbreviation
ABBRA	adjective abbreviation
ABBRV	verbal abbreviation
ABBRD	adverbial abbreviation
ABBRC	numerical abbreviation
ABBRX	unspecified abbr.
ZNUM	number using digits (<f num> tag and similar in csts)
ZIP	punctuation (<d> tag in csts)
ZSB	sentence boundary
NOMORPH	no result from morphology
NOMORPH1	no result from morphology, one letter only in word form

Each tag may be followed by a numerical variant and/or register indication, which is separated from the tag by a hyphen (-). The list of possible variant and register indications is presented in the following table.

Indication	Description
-1	other variant: less frequent
-2	other variant: much less frequent - archaic or bookish
-3	very archaic / possibly also colloquial (*-..t'*)
-4	archaic/bookish, standard at the time it was used (transgr. 'pres.' perf.)
-5	colloquial, but tolerated in public writing/speaking
-6	colloquial (should not be used in public writing; mostly spoken)
-7	colloquial (same as -6, but less preferred by speakers)
-9	special use (after certain prepositions, etc.)

For example, NIS6A-1 is the variant form of locative case of a masculine inanimate noun in singular, not negated.

2.1.2 Morphological Analysis

Morphological analysis is a process the input of which is a word form as found in the text, and the output of which is a set of possible lemmas which represent such form in the dictionary, and each lemma is accompanied by a set of possible tags (as defined in the previous section). For example, for the word form *ženu* the morphological analysis returns the following results:

lemma	tag(s)
žena (woman)	NFS4A
hnát (to rush)	VPS1A

This example exhibits an ambiguity at the lemma level, but no ambiguity within the lemmas. On the other hand, the word form *učení* displays both types of ambiguity:

lemma	tag(s)
učení (theory)	NNS1A,NNS2A,NNS3A,NNS4A,NNS5A,NNS6A,NNP1A,NNP2A,NNP4A,NNP5A
učený (educated)	AMP11A,AMP51A

There could be as many as five different lemmas for a given word form and as many as 27 different tags for one lemma.

Morphological analysis currently covers about 720,000 Czech lemmas (including derivations), and is based on about 210,000 stems. It can recognize about 20 million word forms.

Morphological analysis is the first step towards the first level tagging in the Prague Dependency Treebank. It can proceed fully automatically and very quickly (about 5000 word forms per second on average on a P6/200-based machine). We have developed special software tool (called `sgd` on a Unix platform, and `DA` under Windows) which allows for an easy manual disambiguation of the morphological output. It also helps the annotators to edit the output of the morphology, thus allowing for identification of possible problems and unknown words in the morphology itself. The disambiguation is being done by students and proceeds relatively quickly. The texts are processed twice and the results are automatically compared and manually corrected to get a single, as-much-as-possible error-free tagged text. The level of differences between any two annotators working on a single text is on average 5%, and there are virtually no opinion-type disagreements - the differences are typos, misunderstandings, etc.

2.2 The Analytical Level

The analytical (syntactic) level of annotation is central to the current stage of the project (Bémová et al. 1997). We have chosen the dependency structure to represent the syntactic relations within the sentence. The basic principles can be thus formulated as follows:

- The structure of the sentence is an oriented, acyclic graph with one entry node; the nodes of the tree are annotated by complex symbols (attribute-value pairs);
- The number of nodes of the graph is equal to the number of words in the sentence plus one for the extra root node;
- The annotation result is only
    - 1. the structure of the tree,
    - 2. the analytical functions of every node;

  an analytical function determines the relation between the dependent node and it governing node (which is the node one level up the tree). All the other node attributes (see the table below) are used as a guidance for the annotators, or they are used as an input or intermediate data for various automatic tools which participate in the annotation process, but are not considered to be the result of analytical annotation. In particular, the tags and lemmas are not disambiguated here,

but are merged into the resulting data structure from the manual annotation on the first (morphological) level.

The first 12 node attributes are summarized in the following table (there are 8 more "technical" attributes used for macro programming as intermediate data holders etc.):

Attribute name	Brief description
`lemma`	lemma (see sect. 2.1, The Morphological Level)
`tag`	morphological categories, or tag (see sect. 2.1, The Morphological Level)
`form`	word form, after minor changes in some cases (contractions, fixed idioms)
`afun`	the analytical function, or the type of dependency relation (towards the governing node)
`lemid`	lemma identification (reserved only at present)
`mstag`	morphological/syntactic tag (reserved only at present)
`origf`	original word form as found in the text
`origap`	formatting (preceding the original word form)
`gap1,gap2,gap3`	formatting info preceding `form`, parts 1,2,3
`ord`	sequence no. of the word form in a sentence (attribute `form`)

The annotation rules are described in the manual (Bémová et al. 1997), which is currently available as a technical report and in electronic form as an HTML document (in Czech; on the Internet at http://fairway.ms.mff.cuni.cz/fes2a/tb/doc/tbman.html). These rules follow, where possible, the traditional grammar books, but are both extended (where no guidance has been found in such books) and modified (where the current grammars are inconsistent with themselves). They are intentionally as independent of any formal theory as possible (even though the decision to use the traditional - at least in Prague - dependency representations is certainly not quite theory independent - but in fact, this decision made our lives easier because of several phenomena inherently occurring in Czech, which would otherwise result in the well-known "crossing brackets" problem).

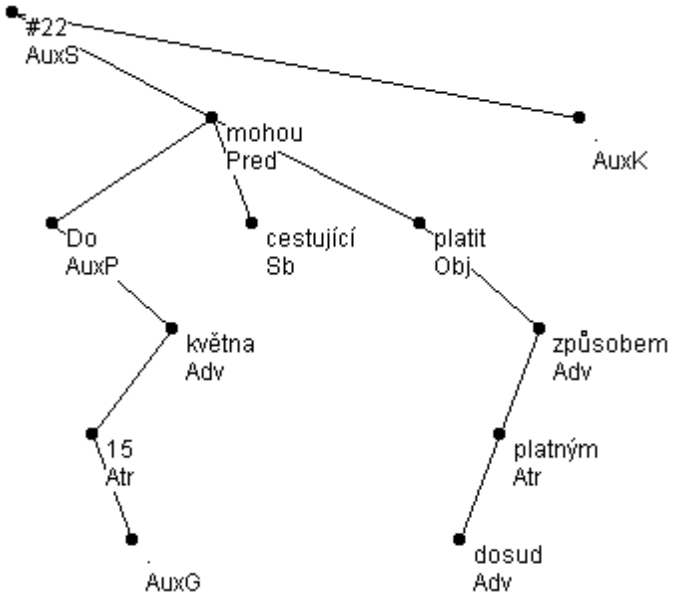
In the following table, all possible values of the analytical function attribute (`afun`) are described briefly. The existence of a suffixed version (`_Co` for coordination, `_Ap` for apposition, `_Pa` for parenthetical expressions) is marked by an `x`.

`afun`	`_Co`	`_Ap`	`_Pa`	Description
`Pred`	x	x	x	Predicate if it depends on the tree root (#)
`Sb`	x	x	x	Subject
`Obj`	x	x	x	Object
`Adv`	x	x	x	Adverbial (without a detailed type distinction)
`Atv`	x	x	x	Complement; technically depends on its non-verbal governor
`AtvV`	x	x	x	Complement, if only one governor is present (the verb)
`Atr`	x	x	x	Attribute
`Pnom`	x	x	x	Nominal predicate's nominal part, depends on the copula "to be"
`AuxV`	x	x	x	Auxiliary Verb "to be" (*být*)
`Coord`	x	x	x	Coordination node
`Apos`	x	x	x	Apposition node
`AuxT`	x	x	x	Reflexive particle *se*, lexically bound to its verb
`AuxR`	x	x	x	Reflexive particle *se*, which is neither Obj nor AuxT (passive)
`AuxP`	x	x	x	Preposition, or a part of compound preposition
`AuxC`	x	x	x	Conjunction (subordinate)
`AuxO`	x	x	x	(Superfluously) referring particle or emotional particle
`AuxZ`	x	x	x	Rhematizer or other node acting to stress another constituent
`AuxX`				Comma (but not the main coordinating comma)
`AuxG`				Other graphical symbols not classified as `AuxK`
`AuxY`	x	x	x	Other words, such as particles without a specific (syntactic)

afun	_Co	_Ap	_Pa	Description
				function, parts of lexical idioms, etc.
AuxS				The (artificially created) root of the tree (#)
AuxK				Punctuation at the end of sentence or direct speech or citation clause
ExD	x	x	x	Ellipsis handling (Ex-Dependency): function for nodes which "pseudo-depend" on a node on which they would not if there were no ellipsis.
AtrAtr, AtrAdv, AdvAtr, AtrObj, ObjAtr	x	x	x	A node (analytical function: an attribute) which could depend also on its governor's governor (and have the appropriate other function). There must be no semantic or situational difference between the two cases (or more, in case of several attributes depending on each other). The order represents annotator's preference, but is largely unimportant.

As an example of an analytical-level annotation of a sentence we present here the representation of the sentence (literal translation given after slashes; translation in parenthesis):

Do/Till 15./15th května/May mohou/can cestující/passengers platit/pay dosud/hitherto platným/valid způsobem/way. (Until May 15, the passengers can pay in the way currently used.)



The original word forms as well as the attribute values of the analytical function are displayed. This example shows

- the extra root node of the tree, with a number of the sentence within a file;
- the handling of analytical verb form (modal verb *mohou*+ infinitive *platit*)
- the fact that the verb is the governing node of the whole sentence (or of every clause in compound sentences), as opposed to the complex subject - complex predicate distinction made even in the otherwise dependency-oriented traditional grammars of Czech, such as (Šmilauer 1969)
- attachment of a manner-type adverbial to a analytical verb form
- date handling
- prepositional phrase structure (preposition on top)

and, of course, all the analytical functions assigned to these nodes

2.3 The Tectogrammatical Level

The tectogrammatical level annotation is based on the framework of Functional Generative Description (FGD) as developed in Prague by Peter Sgall and his collaborators since the beginning of the +1960's (for a most detailed and integrated formulation, see Sgall, Hajičová and Panevová 1986). The basic principles of annotation are different than on the analytical level. Instead of requiring every word becoming a node, we require that every autosemantic word becomes a node, plus all nodes deleted on the surface - and thus on the analytical level - will be added.

3 The Manual Annotation of the PDT at the Analytical Level

3.1 Organization

Not surprisingly, the effort to organize the structural annotation appears to be a complicated task. There is little experience with such a task: we have learned from the LDC's experience with Penn Treebank, but there was no other description available of similar projects. The annotation itself begun in November 1996 by constituting a working group of 8 people, 5 of them hired just for the annotation thanks to an external grant and other support. The remaining three are faculty members: the author of this article, who is primarily a computer scientist, and who is responsible for the data preparation, software tools, organization and general administration of the project, Jarmila Panevová, who is responsible for the annotation rules, and Alla Bémová, who, while annotating the treebank, also works on the annotation rules. All the newly hired people (Eva Buráňová, a linguist, now retired, former faculty member and member of the Institute, three students of linguistics at the Faculty of Arts: Jiří Kárník, Petr Pajas and Jan Štěpánek, and Zdeňka Urešová, also a linguist by education, with experience in lexicography) were not only computer-literate, but the students were also studying computer science with various intensity. Therefore their background allowed us virtually to skip any introduction to computational linguistics and we could start immediately with the annotation process itself.

The process of annotation has been (and still is) viewed as a cyclical process where the rules for annotation are being constructed based on the evidence found in the data. Thus we have explained the basic principles of annotation to the annotators, and asked them to use existing grammar books, most notably (Šmilauer 1969), an old, but still the best Czech grammar description. It also uses dependency framework, although there are some (easily identifiable and replaceable) deviations. We were aware of the fact that there are many gaps in such a traditional grammar from the point of view of an explicit annotation based on the basic principles stated above: mainly, the request to have each input word represented by a node in the tree (a request quite natural from the computational point of view) is largely not reflected in any human-oriented grammar description. Nevertheless, before starting to write authoritative guidelines based on such a grammar, we believed that a definite version can be constructed on the fly, with annotation corrections made later should the rules change.

After solving the initial hardware and software installation problems the annotation begun in December 1996. The key software tool used was the GRAPH program, developed by Michal Křen initially as his undergraduate thesis in 1995/96, and greatly enhanced afterwards (see also below under the section 3.2 Tools). This tool allows for graphical viewing and editing of the dependency representation of annotated sentences.

During the first 6 months (until May 1997) we have annotated roughly 4,000 sentences of newspaper texts. We held regular meetings every 2-3 weeks and solved problems which the annotators identified. We have generated an (rather unordered) set of rules for cases found in the data, which had not been treated previously in the grammar books or which had been treated inconsistently. After those six months it became apparent that there is a strong need to organize all these rules (plus those which could be used

directly from the existing grammar) into an "Annotator's Guidelines". The development of the Guidelines is considered to be the second phase of the project.

We have distributed those 4,000 sentences among all the annotators and divided the work in a different way. Four subtasks were defined:

- Complete rewrite of the existing grammar (Šmilauer 1969) from the annotation point of view based on examples taken from the grammar.
- Revision and merging of the above examples and the rules created during the first 6 months.
- Existing data (those 4,000 sentences) correction based on the Annotator's Guidelines.
- Technical editing of the guidelines, to be available both in an HTML format and as a printed manual, with rich indexing/hyperlink scheme.

Technical editing appeared to be quite a task - given that we wanted to have both a printed and an HTML version of the manual. Microsoft Word version 8 was supposed to be suitable for this task, and in principle it is (despite its well-known problems), but the guidelines creation process (done by Jiří Kárník) is very demanding.

The first task, undertaken by Zdeňka Urešová, which could be dubbed as "Šmilauer's syntax in examples", generated over 600 sheets of paper with every subsection from the Novočeská skladba represented by at least one example. All of these has been revised by Jarmila Panevová, and partially by others, to confirm or replace any questionable decisions taken in unclear cases. Meanwhile, the structure of the core chapters of the guidelines have been defined by the author of this article together with the initial indexing scheme. The indexing scheme has four levels: words, morphological characteristics, syntactically-based phenomena based on the traditional notions, and the resulting analytical functions. The indexing, or hyperlinking, in the manual is a very important feature, as it is impossible to structure the guidelines from several points of view simultaneously. The basic structure follows that of the traditional grammar book, with sections added where appropriate. This is based on the assumption that a typical annotator will be a linguist or somebody familiar with the traditional grammar. However, the indexes serve as entry points for other points of view, such as the "computational" one, starting with morphological and part-of-speech information, or the "experienced annotator" one, who remembers wording of the examples, but not quite how to annotate them. Also, the indexes should be also useful for the user of the treebank, who simply wants to know what an analytical function (such as Adv or Coord) has been used for. Later, we will develop also the terminological index.

All the annotators have helped to formulate the final wording in the Guidelines, and each of them is responsible for a certain section of the Guidelines (for example, for subject, or rhematizers and multiword units, etc.). Given their effort in this respect, and also their contribution to the formulation of the annotation rules during the first phase of the project, they all become not only the annotators, but also the authors of the Guidelines (Bémová et al. 1997).

Starting November 1997, even though the Guidelines are not quite complete and not all problems have been resolved, we started the third phase: annotation of new data continues. We expect to have 120,000 sentences annotated by the end of June of 1998. In the final phase of the project we will run consistency tests, which are now being prepared by people outside the group. We aim at half million words (200,000 sentences) to form the Prague Dependency Treebank at the end of the project. There are also other non-trivial tasks connected to the project: for example, the tagged data (level 1) have to be merged with the structurally annotated data, changes in morphology have to be incorporated, the resulting format must be converted to SGML to be included in the Czech National Corpus, etc. These tasks are being worked on by our collaborators from our Institute, mainly by those who are ultimately interested in using the PDT for their own research.

3.2 Tools

Manual annotation does not mean that people are typing complicated formal representations by hand into a computer. Even the first annotation attempts in the times when graphical editing was resource-

demanding and therefore not feasible were guided by software tools. These tools allowed the annotators to assign only a formally correct entry, avoiding expensive checking-and-correction process afterwards.

Based on the availability of computing power today, we decided that for the annotation of the PDT we use as advanced tools as possible. After one year of experience of annotating both morphologically and syntactically, it seems that this was the right choice.

3.2.1 Morphological disambiguation: `sgd` and `DA`

We use a special purpose tool for morphological annotation, which allows for an easy disambiguation of lemmas and tags as output by the morphological analyzer. The tool has first been implemented under the Linux operating system under the name `sgd` by Karel Skoupý (and is capable of running also on Solaris and other operating systems of the Unix type). It has been reimplemented also for the Windows 95 platform by Jiří Hana (under the `DA` name), to allow for annotators who do not have the possibility to install Linux on their home machines. The user interface is identical. The `sgd` tool is text-terminal based so it can be relatively easily (character coding problems aside) used from any `vt100`-capable terminal, as well as a from an `xterm` or similar programs.

The tools work full screen on texts in a SGML format (as defined by the Czech National Corpus' standard data type definition, namely, the `csts.dtd`) preprocessed by a morphological processor (see sect. 2.1.2 Morphological Analysis above). The annotators are presented with a list of ambiguous words as found in the input text (expandable to full text list, with ambiguous words marked by an asterisk). The full text context is also displayed in a separate window, with the active word marked by reverse video. The largest part of the screen is devoted to the disambiguation process itself. The annotator first chooses the correct lemma, and then, if needed (which is usually the case, as more than 45% words (tokens) are morphologically ambiguous in Czech), the correct tag. S/he has the possibility also to edit both the lemma and the tag, in case the morphological processor did not know the word altogether or made an error. The text is saved only with the lemmas and tags chosen by the annotators. We must admit that saving only the disambiguated lemmas and tags was a decision which now makes problems when we want to upgrade the annotated data using new lexicons and tags (as the morphology evolves, of course).

There are other tools related to morphological annotation, but these are mostly standard Unix tools (diff, flex, awk, perl etc.). These help to resolve differences between two annotators on the same text and to do other conversions of the material.

3.2.2 The analytical level annotation tool: GRAPH

The analytical level, even though we are interested in the structure and one attribute (analytical function) "only", is a major challenge because of its inherently non-linear nature. We have used a program called rather uninspiratively GRAPH. This program works under Microsoft Windows (3.1 and 95) and has been developed as a graduate thesis based on initial specification developed long before the annotation project actually began. It has changed a lot since then - there were about 40 versions of it with bug fixes, minor and major updates. The program allows for drag-and-drop style editing of trees with annotated nodes. It is not just for dependency-based formal representations, even though it has special features (such as visual node ordering) which were inspired by such formalisms. Several files can be opened concurrently, (sub)trees may be copied among them using multiple-buffer clipboard, files may be searched for node annotations. The display of trees (attributes to be displayed, colors, fonts, line thickness, etc.) is fully configurable to suit the task at hand as well as the annotator's preferences, which might depend on the hardware or other differences. The program can be completely mouseless driven, too.

One of the major features recently introduced into the GRAPH program is the possibility to use macros - or in other words, the program became programmable. The programming language (which is interpreted at the moment) is similar to C but contains only those constructs necessary for the annotation tasks. The

functions can be invoked interactively (by a keypress) or from the command line when starting the GRAPH program. These macros have been used so far for two different purposes:

- as shortcuts, long asked for by the annotators, to avoid opening 2 or 3 windows when selecting the appropriate analytical function for a node in the tree;
- for a preliminary assignment of analytical functions to nodes when the tree structure is built, but before the manual node annotation.

The programming facility is not intended to be used by the annotators, but they are able to use the macros prepared by programmers. We expect that we will use these macros also for tree checking and transformations, will it be necessary e.g. after changes made to the annotation rules. The programming language allows for almost all the editing operations made normally by the annotators, including tree restructuring. Thus in principle they could be used also for initial tree structure assignment, but it will have to be seen if such a "macro" is feasible to implement (in fact, that would be an attempt at parsing!).

The shortcuts allow to assign an analytical function to an active node by a simple keypress, or a Ctrl and/or Shift plus a key in case of functions "suffixed" by `_Co`, `_Ap` or `_Pa`. These macros also store the value previously assigned to this node, and another macro function, when invoked, can thus revert to the previous value, should the annotator decide that s/he made a mistake. There are also macros for node swap, for assignment of the `Atr` function to all nodes in a subtree (a frequent case near the leaves of the tree), and for special coordination and apposition handling.

The initial analytical function assignment is performed by a 800+ lines long function which tries to assign the most plausible analytical function to every node of a tree. The assignment is based on relatively simple hand-crafted rules. They are far from perfect, and sometimes intentionally disregard some complicated contexts, but as the first feedback from the annotators shows, they are correct in almost 80% cases. These rules will be complemented by a set of rules working with lists of verbs requiring certain type of object constructions, which should enhance the assignment of he `Obj` analytical function and avoid the need to search such lists in the Guidelines (similarly, the function can deal with multiword prepositions and conjunctions, rhematizers etc.).

The initial assignment function can also be used (under a different name) on a file as a whole, which means that the annotators don't have to run the macro on every tree. The batch feature of the GRAPH program also allows to run the same macro on many files using a single command, which might help should the work be divided between "structure" annotation and "analytical function" annotation (which is not the case at the moment, but it is an open question whether we will do it in the future or not).

The macros could be used also for complex searches in the whole treebank, but currently this is possible only at the program level. We plan to introduce also an interactive user interface to the macro programming language, which would allow also the user to specify a complex search query.

4 Treebank Usage: Parsing Unrestricted Text

The treebank can obviously be used for further linguistic research, as it contains a lot of material annotated in a way directly usable by original linguistic research, quickly searchable using different criteria. However, in the present contribution we will discuss a more "computational" usage of the treebank, namely, as a basis for creating a statistically-based parser of unrestricted written text.

4.1 The Problem of Parsing

There are many attempts to parse sentences of natural language at various levels (Brill 1993a, Brill 1993b, Collins 1996, Collins 1997, Ribarov 1996). We aim here at syntactico-semantic parsing of unrestricted text. It is a well-known fact that hand-crafted rules work well for restricted domains and vocabularies, whereas they generally fail for unrestricted text parsing. So far the (partial and imperfect, but still the best

available) answer to this problem has been statistical parsing based on training on manually annotated data.

Having such a resource available for Czech (the Prague Dependency Treebank as described in the previous sections), we decided to pursue this paradigm further, trying to combine the best of both worlds. While statistical models are good at discovering relations which no linguist would and even could consider without examining similar amount of data (which is typically impossible), the computational linguistics perspective is indispensable in guiding the selection process.

4.2 The Model

We have developed a statistical model which has been successfully used for tagging (full morphological disambiguation), where it improved accuracy by 10 percentage points, from 80% (Hladká 1994, Hajič and Hladká 1997a) to 90% (Hajič and Hladká 1997b). The statistical model is based on the exponential probabilistic model of the form

$$p(y|x) = e^{\sum_{i=1..n} \lambda_i f_i(y,x)} / Z_\lambda(x)$$

where $f_i(y,x)$ is a feature selector function, which returns 1 or 0 depending on the value of y and the context x, λ_i is its weight, and $Z_\lambda(x)$ is a normalization factor making the distribution a probabilistic distribution which sums to 1.

This crucial property of this model, used successfully for many applications in tagging as well as in machine translation, is the set of *n* features (typically in the order of hundreds). These features are selected automatically, based on objective criteria, from a much larger "pool" of available features. The selection of features may be guided by two different principles: a "minimal cross-entropy" principle, which compares the probability distribution constructed to the training data (using the cross-entropy measure, or simply the probability of training data), or "minimal error rate" (again, on training data). We have chosen the second principle, as it more directly attacks the problem at hand.

The selection of features, however, depends also on the values of λ_i. The basic method for feature weight computation is the Maximum Entropy method. Unfortunately this method involves several numerical iterative algorithms which makes it rather slow. We believe, based on our experience with similar models (and with smoothing, which displays a similar "weighting" issue, in general) that the exact weight computation is not so important to the resulting model performance, and thus that the values of λ_i may be roughly - and quickly - approximated instead. This would allow to select features from larger pools, enabling more sophisticated features to be selected.

4.3 The Parsing Scheme

It is simple to define what is to be predicted in tagging: for each input word form, it is the tag of that word. In parsing, however, there is no straightforward way to define the output random variable. It seems that the most natural way is to predict (within the dependency framework) the governor of each word. Each node is a k-tuple of attribute-value pairs, such as lemma, tag, form, analytical function, etc. The context of a dependency relation is then the linear list of nodes (with some values possibly unspecified) of the whole sentence, plus the set of dependency relations generated so far. Although in principle the dependency relations within a sentence can be generated in any order (just making sure that the result is a tree), we prefer them being generated in a top-down manner. It makes our life easier in the sense of getting rid of deferred feature evaluation and similar computationally expensive phenomena.

The pool of features to select from during the training phase of our exponential model contains features, which appeared at least once (or, in general, *m*-times) in the training data and which correspond to the following templates:

Dependent node	Skip Which Governor?	Immediate Governor	Parent of Immediate Governor	Left Brother

all values	do not skip	all values	any	any
all values	do not skip	POS or lemma only	POS or lemma only	any
all values	certain POS values	all values	any	any
all values	do not skip	POS or lemma only	any	all values

An example of such a feature could be the following: "The current node is a noun, its governor is a preposition, and the governor of its governor is a verb". Such feature, if selected, should be presumably given substantial weight, as it is the general scheme for many adverbials and objects (as opposed to the feature "The current node is a verb, its governor is a preposition and the governor of its governor is a verb").

The will be a certain number of features representing the non-projectivity of the tree. A feature f_p will return 1 if the number of edges of the tree causing non-projectivity is equal to p. These features will, presumably get negative weights to discount for too "wildly" non-projective trees.

If this set of templates is not sufficient enough, we will add more sophisticated features to the pool one by one, implemented as code segments. We assume indeed that these features will be selected by an objective selection process, but if not, we will not force their selection in any way.

Moreover, there will be a feature the task of which is to make sure that the result is a tree, not a general graph. Such feature will be formulated as the inverse relation (it will return the value 1 if the graph is *not* a tree) and it will be given the weight minus infinity (or, simply, a very large negative number). This feature must always be present.

The computation of the feature weights will be approximated very roughly from the inverse of the frequency of the feature, which should prefer more specific features.

The selection process will then proceed as follows: (a) it will start with the tree-preserving feature only, (b) it will set the feature-adding threshold to some large but reasonable number, (c) for each possible feature from the pool of features, it will approximate its weight, and compute the error reduction count on the training data should such feature be added permanently, (d) if the reduction is higher than the threshold, it will add the feature permanently to the set, and it will take another feature and continue with the step (c), (e) if all features have been tested (and possibly added to the resulting set of features), it will lower the initial threshold by half, and if it is not lower than a preset minimal threshold, it will start over again with the first feature from the pool with the step (c).

After the training phase is finished and the set of features with weights associated to them is completed, the parsing itself will proceed as follows: (a) it starts at the artificial single node making it the initial parse tree, (b) it puts all edges (dependencies) possible to expand the tree on the stack, together with their probability, (c) it gradually expands the stack in a breadth-first (parallel) manner (up to a certain breadth, called the "beam width"), and (d) after all edges have been added, it selects the most probable parse tree by simply taking the maximum of the product of all the edge probabilities.

5 Conclusions

Everybody would certainly agree that to build a treebank is a difficult task. Our belief is, however, that all the hard work will pay off - in that not only us who are building it, but all the computational linguists interested in morphology and syntax of natural languages in general and of Czech or other inflectional and free word order languages in particular will benefit from its existence. The building of the treebank has been very fruitful even now, halfway through the whole treebank annotation: we have been effectively forced to describe the syntactic behavior of Czech more explicitly and more widely (in the sense of overall coverage, including also so called "peripheral" phenomena) than ever.

6 References

Bémová et al. (1997): Anotace na analytické rovině - příručka pro anotátory [Annotation on the Analytical Level - Annotator's Guidelines], Technical Report #4 (draft), LJD ÚFAL MFF UK, Prague, Czech Republic (in Czech).

Brill, E. (1993a): Automatic Grammar Induction and Parsing Free Text: A Transformation-Based Approach. In: Proceedings of the 3rd International Workshop on Parsing Technologies, Tilburg, The Netherlands.

Brill, E. (1993b): Transformation-Based Error-Driven Parsing. In: Proceedings of the 12th National Conference on Artificial Intelligence.

Collins, M. (1996): A New Statistical Parser Based on Bigram Lexical Dependencies, In: Proceedings of the 34th Annual Meeting of the ACL'96, Santa Cruz, CA, USA, June 24-27, pp. 184-191.

Collins, M. (1997): Statistical Parser Based on Bigram Lexical Dependencies, In: Proceedings of the 35th Annual Meeting of the ACL/EACL'97, Madrid, Spain.

Hajič, J., and Hladká, B. (1997a): Probabilistic and Rule-based Tagger of an Inflective Language - A Comparison, In: Proceedings of the 5th Conference on Applied Natural Language Processing, ACL, Washington, DC, USA, pp. 111-118

Hajič, J., and Hladká, B. (1997b): Error-driven Tagging for a Rich, Structured Tagset, Based on an Exponential Model, Technical Report #3, LJD ÚFAL MFF UK, Prague, Czech Republic.

Hajič, J., and Hladká, B. (in print): Morfologické značkování češtiny [Morphological tagging of Czech], In: Slovo a Slovesnost, Vol. 58, No. 4, ÚJČ AV ČR, Prague.

Hajič, J., and Ribarov, K. (1997): Rule-Based Dependencies, In: Proceedings of the Workshop on Empirical Learning of Natural Language Processing Tasks, MLNet, Prague, Czech Republic, April 23-25, pp. 125-136

Hladká, B. (1994): Programové vybavení pro zpracování velkých českých textových korpusů [Software for Large Czech Corpora Annotation], MSc thesis, MFF UK, Prague, Czech Republic.

Panevová, J. (1971), Opisanie obstojatel'stva vremeni v generativnoj sisteme s neskol'kimi urovnjami (na materiale češskogo jazyka). Prague Bulletin of Mathematical Linguistics 15, 19-40.

Panevová, J. (1973), Věty s všeobecným konatelem [Sentences with a general Actor]. In: Studia Slavica Pragensia, Prague: Charles University, 133-144. Translated in: Contributions to Functional Syntax, Semantics, and Language Comprehension (ed. by P. Sgall), Prague: Academia and Amsterdam: John Benjamins, 1973, 203-221.

Panevová, J. (1974), On Verbal Frames in Functional Generative Description. Part I, Prague Bulletin of Mathematical Linguistics 22, 3-40, Part II, Prague Bulletin of Mathematical Linguistics 23, 1975, 17-52.

Panevová, J. (1975a), Rozvití předmětová a příslovečná, doplňující a určující [Object and adverbial modifications, complementing and specifying]. Naše řeč 58, 61-66.

Panevová, J. (1975b), Tzv. vedlejší věty místní a jejich významová stavba [The so-called local embedded clauses and their semantic structure]. Slovo a slovesnost 37, 284-290.

Panevová, J. (1978), Inner Participants and Free Adverbials. In: Prague Studies in Mathematical Linguistics 6, 227-254.

Panevová, J. (1979), From Tectogramatics to Morphemics. In the series: Explizite Beschreibung der Sprache und automatische Textbearbeitung IV, Prague: MFF UK.

Panevová, J. (1980), Formy a funkce ve stavbě české věty [Forms and functions in the structure of Czech sentence]. Prague: Academia.

Panevová, J. (1991), On a Classification of Adverbials. In: Proceedings of the 14th International Congress of Linguists I, (ed. by W. Bahner, J. Schild, D. Viehweger), Berlin: Akademie-Verlag, 804-806.

Panevová, J. (1994), Valency Frames and the Meaning of the Sentence. In: The Prague School of Structural and Functional Linguistics (ed. by Ph. L. Luelsdorff), Linguistic and Literary Studies in Eastern Europe 41, Amsterdam-Philadelphia: John Benjamins, 223-243.

Ribarov, K. (1996): Automatická tvorba gramatiky přirozeného jazyka [The Automatic Creation of a Grammar of a Natural Language], MSc thesis, MFF UK Prague.

Sgall, P. et al. (1986): The Meaning of the Sentence and Its Semantic and Pragmatic Aspects, Reidel Publishing Company, Dordrecht, Netherlands, Academia, Prague, Czech Republic.

Šmilauer, V. (1947), Novočeská skladba [Syntax of Contemporary Czech], 1st ed., Prague.

Šmilauer, V. (1969), Novočeská skladba [Syntax of Contemporary Czech], 3rd ed., SPN, Prague, 574 pp.