Deriving enhanced Universal Dependencies from a hybrid dependency-constituency treebank

Lauma Pretkalniņa, Laura Rituma, and Baiba Saulīte

Institute of Mathematics and Computer Science, University of Latvia
Raiņa 29, Riga, Latvia, LV-1459
lauma@ailab.lv, laura@ailab.lv, baiba@ailab.lv

Abstract. The treebanks provided by the Universal Dependencies (UD) initiative are a state-of-the-art resource for cross-lingual and monolingual syntax-based linguistic studies, as well as for multilingual dependency parsing. Creating a UD treebank for a language helps further the UD initiative by providing an important dataset for research and natural language processing in that language. In this paper, we describe how we created a UD treebank for Latvian, and how we obtained both the basic and enhanced UD representations from the data in Latvian Treebank which is annotated according to a hybrid dependency-constituency grammar model. The hybrid model was inspired by Lucien Tesnière's dependency grammar theory and its notion of a syntactic nucleus. While the basic UD representation is already a *de facto* standard in NLP, the enhanced UD representation is just emerging, and the treebank described here is among the first to provide both representations.

Keywords: Latvian Treebank, Universal Dependencies, enhanced dependencies

1 Introduction

In this paper, we describe the development and annotation model of Latvian Treebank (LVTB), as well as data transformations used to obtain the UD representation from it. Since Latvian is an Indo-European language with rich morphology, relatively free word order, but also uses a lot of analythical forms, it was decided to use a hybrid dependency-constituency model (see Section 2.2) in the original Latvian Treebank pilot project back in 2010 (see Section 2.1).

Universal Dependencies[1] (UD) is an open community effort to create cross-linguistically consistent treebank annotation within a dependency-based lexicalist framework for many languages [3]. Since 2016 we have been participating by providing a UD compatible treebank derived from LVTB (Latvian UD Treebank or LVUDTB). UD provides guidelines for two dependency annotation levels—base dependencies (mandatory) where annotations are surface-level syntax trees, and enhanced dependencies where annotations are graphs with additional information for semantic interpretation. In order to generate the LVUDTB for each

[1] http://universaldependencies.org/

of the UD versions, a transformation (see Section 3) is applied to the current state of LVTB. Together with Polish LFG and Finnish TDT, PUD treebanks LVUDTB is among the first to provide enhanced in addition to basic dependencies.

2 Latvian Treebank

2.1 Development

Development of the first syntactically annotated corpus for Latvian (Latvian Treebank, LVTB) started with a pilot project in 2010 [6]. During the pilot a small treebank was created with texts from JRC-Acquis, Sofie's World, as well as some Latvian original texts [7]. In 2017 LVTB consisted of around 5 thousand sentences, one third of which were from Latvian fiction and another third from news texts. We are currently making a major expansion to LVTB, with a goal of balancing the corpus (aiming for 60% news, 20% fiction, 10% legal, 5% spoken, 5% other) and reaching about 10 thousand sentences by the end of 2019 [2].

Latvian Treebank serves as the basis for LVUDTB, which is a part of the UD initiative since UD version 1.3. Since UD v2.1. in addition to containing basic dependencies, LVUDTB also features enhanced dependencies as well.

2.2 Annotation model

The annotation model used in Latvian Treebank is SemTi-Kamols [1, 4]. It is a hybrid dependency-constituency model where the dependency model is extended with constituency mechanisms to handle multi-word forms and expressions, i.e., syntactic units describing analytical word forms and relations other than subordination [1]. These mechanisms are based on Tesnière's idea of a syntactic nucleus which is a functional syntactic unit consisting of content-words or syntactically inseparable units that are treated as a whole [4]. From the dependency perspective, phrases are treated as regular words, i.e., a phrase can act as a head for depending words and/or as a dependent of another head word [6]. A phrase constituent can also act as a dependency head.

A sample LVTB tree is given in Figure 1 on the left. Dependency relations (brown links in Fig. 1, left) match with grammatic relations in Latvian syntax theory [5]. Dependency roles are used for traditional functions: predicates, subjects, objects, attributes, and adverbs. They are also used for free sentence modifiers: situants, determinants, and semi-predicative components. A free modifier is a part of a sentence related to the whole predicative unit instead of a phrase or single word, and it is based on a secondary predicative relation or determinative relation. A situant describes the situation of the whole sentence. A determinant (dative-marked adjunct) names an experiencer or owner (it is important to note that the `det` role in LVTB is is not the same as the `det` role in UD). A semi-predicative component can take on a lot of different representations in the sentence: resultative and depictive secondary predicates, a nominal standard in

Table 1. Dependency types in Latvian Treebank

Role	Description	Corresponding UD roles
subj	subject	nsubj, nsubj:pas, ccomp, obl
attr	attribute	nmod, amod, nummod, det, advmod
obj	object	obj, iobj
adv	adverbial modifier	obl, nummod, advmod, discourse
sit	situant	obl, nummod, advmod, discourse
det	determinant	obl
spc	semi-predicative component	ccomp, xcomp, appos, nmod, obl, acl, advcl
subjCl	subject clause	csubj, csubj:pas, acl
predCl	predicative clause	ccomp, acl
attrCl	attribute clause	acl
appCl	apposition clause	acl
placeCl	subordinate clause of place	advcl
timeCl	subordinate clause of time	advcl
manCl	subordinate clause of manner	advcl
degCl	subordinate clause of degree	advcl
causCl	causal clause	advcl
purpCl	subordinate clause of purpose	advcl
condCl	conditional clause	advcl
cnsecCl	consecutive clause	advcl
compCl	comparative clause	advcl
cncesCl	concessive clause	advcl
motivCl	motivation and causal clause	advcl
quasiCl	quasi-clause	advcl
ins	insertion, parenthesis	parataxis, discourse
dirSp	direct speech	parataxis
no	discourse markers	vocative, discourse, conj

comparative constructions, etc. Other dependency roles are used for the different types of subordinate clauses and parenthetical constructions—insertions, direct speech, etc. Some roles can be represented by both a single word and a phrase-style construction, while others can be represented only by a phrase. Overview on dependency roles used in LVTB is given in the first two columns of Table 1.

There are three kinds of phrase-style constructions in the LVTB grammar model: x-words, coordination and punctuation mark constructions (PMC). X-words (nodes connected with green links, Fig. 1, left) are used for analytical forms, compound predicates, prepositional phrases etc. Coordination constructions (nodes connected with blue links, Fig. 1, left) are used for coordinated parts of sentences, and coordinated clauses. PMCs (nodes connected with purple links, Fig. 1, left) are used to annotate different types of constructions which cause punctuation in the sentence. In this case the phrase-style construction consists of punctuation marks, the core word of the construction, and clause introducing conjunction, if there is one. All three kinds of phrases have their own types. In case of x-words, these types may have even more fine-grained subtypes specified in the phrase tag. As each phrase type has certain structural limitations, it determines the possible constituents in the phrase structure. X-word types and their constituents are described in the first two columns of Table 2, coordination is described in Table 3, and PMC in Table 4.

Structural limitations can be different for each x-word type or subtype. This is important for data transformation to UD (see 3) because it affects which element of the x-word will be the root in the UD subtree. For example, each `xPred`

Table 2. X-words in Latvian Treebank

Phrase → constituent	**Description**	**Corresponding UD roles**
`xPred`	compound predicate	
→ `mod`	semantic modifier	*phrase head*
→ `aux`	auxiliary verbs or copula	`aux`, `aux:pass`, `cop`, `xcomp`, *phrase head*
→ `basElem`	main verb or nominal	`xcomp`, *phrase head*
`xNum`	multiword numeral	
→ `basElem`	any numeral	`nummod`, *phrase head*
`xApp`	apposition	
→ `basElem`	any nominal	`nmod`, *phrase head*
`xPrep`	prepositional construction	
→ `prep`	preposition	`case`
→ `basElem`	main word	*phrase head*
`xSimile`	comparative construction	
→ `conj`	comparative conjunction	`fixed`, `mark`, `case`, `discourse`
→ `basElem`	main word	*phrase head*
`xParticle`	particle construction	
→ `no`	particle	`discourse`
→ `basElem`	main word	*phrase head*
`namedEnt`	unstructured named entity	
→ `basElem`	any word	`flat:name`, *phrase head*
`subrAnal`	subordinative wordgroup analogue	
→ `basElem`	any word	`compound`, `nmod`, `nummod`, `amod`, `det`, `flat`, *phrase head*
`coordAnal`	coordinative wordgroup analogue	
→ `basElem`	any word	`compound`, *phrase head*
`phrasElem`	phraseological unit with no clear syntactic structure	
→ `basElem`	any word	`flat`, *phrase head*
`unstruct`	multi-token expression with no Latvian grammar, e.g., formulae, foreign phrases	
→ `basElem`	any token	`flat`, `flat:foreign`, *phrase head*

(compound predicate) must contain exactly one `basElem` and either exactly one `mod` in case of semantic modification or some `auxVerb`s in case of analytical forms and nominal or adverbial predicates. It is allowed to have multiple `auxVerb`s, if each of them have one of the lemmas *būt*, *kļūt*, *tikt*, *tapt*, or their corresponding negatives. Otherwise, only one `auxVerb` per `xPred` is allowed. Such restrictions result from a different approach to the distinction between modal and main verbs in Latvian syntax theory and UD grammar. These restrictions further simplify transformation to UD, distinguishing the auxiliaries from the main verbs according to the UD approach, as each of the described structure cases need to be transformed differently. Another x-word type where subtypes and structural limitation impact transformation rules, is `subrAnal` (analogue of subordinate-wordgroup) (see Table 5).

The annotation model also has a method for ellipsis handling. If the omitted element has a dependent, the omitted part of the sentence is represented by an accordingly annotated empty node in the tree. This new node is annotated either with an exact wordform or with a morphological pattern showing the features that can be inferred from context in the current sentence. No information from context outside the current sentence is added, and empty nodes without dependents are added only for elided auxiliary verbs.

Table 3. Coordination constructions in Latvian Treebank

Phrase → constituent	**Description**	**Corresponding UD roles**
`crdParts`	coordinated parts of sentence	
→ `crdPart`	coordinated part	`conj`, *phrase head*
→ `conj`	conjunction	`cc`
→ `punct`	punctuation mark	`punct`
`crdClauses`	coordinated clauses	
→ `crdPart`	coordinated clause	`conj`, `parataxis`, *phrase head*
→ `conj`	conjunction	`cc`
→ `punct`	punctuation mark	`punct`

Table 4. Punctuation mark constructions in Latvian Treebank

Phrase → constituent	**Description**	**Corresponding UD roles**
any PMC		
→ `punct`	punctuation mark	`punct`
any clausal PMC		
→ `conj`	conjunction	`mark`, `cc`
→ `no`	address, particle, or discourse marker	`vocative`, `discourse`
`sent`	sentence (predicative)	
→ `pred`	main predicate...	`root`, *phrase head*
→ `basElem`	... or main clause coordination	`root`, *phrase head*
`utter`	utterance (non-predicative)	
→ `basElem`	any non-depending word	`root`, `parataxis`, *phrase head*
`mainCl`	main clause (not subordinated; can be coordinated)	
`subrcl`	subordinated clause	
`dirSp`	direct speech clause	
→ `pred`	main predicate...	*phrase head*
→ `basElem`	... or clause coordination	*phrase head*
`insPmc`	insertion PMC	
→ `pred`	main predicate...	*phrase head*
→ `basElem`	... or other word	*phrase head*
`interj`	interjection PMC	
→ `basElem`	any interjection	`flat`, *phrase head*
`spcPmc`	secondary predication PMC	
`address`	vocative PMC	
`particle`	particle PMC	
`quot`	quotation marks not related to direct speech	
→ `basElem`	main word	*phrase head*

3 Universal Dependencies

Latvian Universal Dependency treebank is built from LVTB data with the help of an automatic transformation procedure[2], based on heuristics and an analytic comparison of the two representations. The transformation result for the sample sentence is given in Figure 1 on the right. Despite being developed without UD in mind, LVTB contains most of the necessary information, encoded either in labels or in the tree structure. Among some distinctions LVTB lacks is a distinction between complements taking (or not) their own subjects—UD `xcomp` vs. `ccomp`. Another problem is that LVTB does not distinguish determiners neither as part-of-speech (`DET` in UD) nor syntactic role (`det`), instead analyzing them as pronouns. This problem is partially mitigated by analyzing the tree structure, and in future we are planning to also consider the pronominal agreement.

[2] https://github.com/LUMII-AILab/CorporaTools/tree/master/LVTB2UD

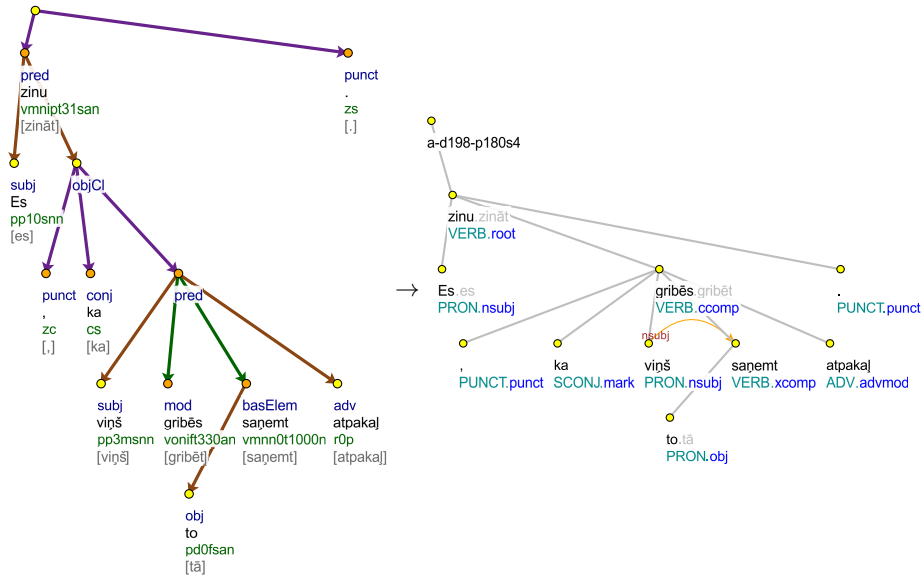

Fig. 1. Sample sentence: *Es*$_{\text{I}}$ *zinu*$_{\text{know.1PRS.SG}}$, *ka*$_{\text{that}}$ *viņš*$_{\text{he}}$ *gribēs*$_{\text{want.3FUT}}$ *to*$_{\text{it}}$ *saņemt*$_{\text{receive.INF}}$ *atpakaļ*$_{\text{back}}$. 'I know he'll want to get it back.'. Tree annotated as in Latvian Treebank on the left, and its UD analogue on the right.

The transformation was built for obtaining basic dependencies and only later, after the release of the UD v2.0 specification, adjusted to create enhanced dependencies. Thus to get an enhanced dependency graph we take annotations for a sentence from LVTB, derive the basic dependency graph from those annotations, and then apply some additional changes. However this approach leads to much more complicated code and more inaccuracies in the final tree, which is why in the future we plan on doing it the other way around, i.e., first constructing the enhanced graph and then reducing it to the basic graph. That would be a better approach because despite surface differences (an enhanced UD graph is not a tree, while LVTB representation is), the enhanced UD representation is closer to the LVTB representation than the basic one, e.g., several types of the enhanced UD edges can be obtained from LVTB distinctions for whether something is a dependent of a phrase as a whole or its part.

Transformation steps for a single tree from the hybrid model to UD:

1. Determine necessary tokens, add `XPOSTAG`s and lemmas from LVTB. Add information about text spacing and spelling errors corrected in the `MISC` field. Sometimes a word from LVTB must be transformed to multiple tokens, e.g., unnecessary split words (like *ne var* 'no can' instead of *nevar* 'can't') are represented as single `M`-level units in LVTB, but as two tokens in UD. If so, appropriate dependency and enhanced dependency links between these tokens are also added in this step.

2. From lemmas and `XPOSTAG` determine preliminary `UPOSTAG` and `FEATS` for each token.
3. Add null nodes for elided predicates (needed for enhanced dependencies) based on how ellipses are annotated in LVTB.
4. Build enhanced dependency graph "backbone" with null nodes, but without other enhanced dependency features. Constructions in LVTB that use dependency relations are directly transformed to a correct UD analogue just by changing the dependency relation labels. LVTB phrase style constructions are each transformed to a connected dependency subtree: every LVTB phrase-style construction is transformed to a single connected subtree and any dependent of such a phrase is transformed to the subtree root dependent.
5. Build basic dependency tree by working out `orphan` relations to avoid null node inclusion in the tree. Other relations are copied from enhanced dependency graph backbone.
6. Finish enhanced dependency graph by adding additional edges for controlled/raised subjects and conjunct propagation.
7. For all tokens update `UPOS` and `FEATS` taking into account the local UD structure. Most notable change being that certain classes of pronouns tagged as `PRON`, but labeled as `det`, are retagged as as `DET`.

Table 5. Phrase-style construction structural transformation

Phrase	**Root choice**	**Structure**
`xPred`	`mod`, if there is one; `basElem`, if all `auxVerb` lemmas are *būt*, *kļūt*, *tikt*, *tapt*; only `auxVerb` otherwise	other parts are root dependents
`xNum`	last `basElem`	other parts are root dependents
`xApp`	first `basElem`	other part is root dependent
`xPrep`	`basElem`	`prep` is root dependent
`xSimile`	`basElem`	`conj` is root dependent
`xParticle`	`basElem`	`no` is root dependent
`namedEnt`	first `basElem`	other parts are root dependents
pronominal `subrAnal`	first `basElem`	other parts are root dependents
adjectival `subrAnal`	last adjective `basElem`	other parts are root dependents
numeral `subrAnal`	first pronomen `basElem`	other parts are root dependents
set phrase `subrAnal`	`basElem`, who is not `xPrep`	`basElem` who is `xPrep`
comparison `subrAnal`	`basElem`, who is not `xSimile`	`basElem` who is `xSimile`
particle `subrAnal`	first `basElem`	other parts are root dependents
`coordAnal`	first `basElem`	other parts are root dependents
`phrasElem`	first `basElem`	other parts are root dependents
`unstruct`	first `basElem`	other parts are root dependents
`crdParts`	first `crdPart`	other `crdPart` are root dependents, other nodes are dependents of the next closest `crdPart`
`crdClauses`	first `crdPart`	the first clause of each semicolon separated part becomes a direct dependent of the root; parts between semicolons are processed same way as `crdParts`
any PMC	`pred`, if there is one; first/only `basElem` otherwise	other parts are root dependents

Steps 4 and 5 are done together in a single bottom-up tree traversal. An overview which LVTB roles correspond to which UD roles is given on Table 1. An

overview of which LVTB phrase part roles correspond to which UD dependency roles is given in Tables 2, 3 and 4. In these tables *phrase head* denotes cases where a particular constituent becomes the root of the phrase representing subtree, and thus, its label is assigned according to the dependency label of the phrase in the LVTB tree. Table 5 describes how to build a dependency structure for each phrase-style construction. If for a single LVTB role there are multiple possible UD roles, for both dependency head and dependent the transformation considers tag and lemma or phrasal structure.

Currently the transformation procedure gives some, but not all enhanced dependency types. The resulting treebank completely lacks any links related to coreference in relative clause constructions and some types of links for controlled/raised subjects. Enhanced dependency roles have subtypes indicating case/preposition information for nominal phrases, but no subtypes indicating conjunctions for subordinate clauses.

We did preliminary result evaluation by manually reviewing 60 sentences (approx. 800 tokens). We found 19 inaccuracies in basic dependencies: 1 due to the lack of distinctions in the LVTB data, 6 due to errors in the original data, and the rest must be mitigated by adjusting the transformation. Analyzing enhanced dependencies, we found 3 errors due to incorrect original data, and some problems that can be solved by adjusting the transformation: 8 incorrect enhanced dependency labels (wrong case or pronoun assigned) and 15 missing enhanced links related to conjunct propagation or subject control. There were no instances of enhanced dependency errors caused by lack of distinctions in LVTB data, however it is very likely that such errors do exist, and we didn't spot one because of the small review sample size. Thus, we conclude that while the transformation still needs some fine-tuning for the next UD release and further reevaluation, overall it gives good results, and situations where LVTB data is not enough to obtain a correct UD tree seem to be rare.

4 Conclusion

Developing a treebank annotated according to the two complementary grammar models has proven to be advantageous. On the one hand, the manually created hybrid dependency-constituency annotations help to maintain language-specific properties and accommodate the Latvian linguistic tradition. The involved linguists—annotators and researchers—appreciate this a lot. On the other hand, the automatically derived UD representation of the treebank allows for multilingual and cross-lingual comparison and practical NLP use cases. The hybrid model is informative enough to allow the data transformation not only to the basic UD representation, but to the enhanced UD representation as well. The transformation itself, however, is rather complicated because of many differences between the two models. Some theoretical differences are big, even up to whether some language phenomena are considered to be either morphological, syntactic, or semantic. But despite the differences, actual treebank sentences, where LVTB annotations are not informative enough to get a correct UD graph,

are rare. To keep up with the development of UD guidelines and LVTB data the transformation would greatly benefit from having even small but repeated result evaluations.

5 Acknowledgement

This work has received financial support from the European Regional Development Fund under the grant agreements No. 1.1.1.1/16/A/219 and No. 1.1.1.2/VIAA/1/16/188.

We want to thank Ingus Jānis Pretkalniņš constructive criticism of the manuscript and anonymous reviewers for insightful comments.

References

1. Barzdins, G., Gruzitis, N., Nespore, G., and Saulite, B.: Dependency-based hybrid model of syntactic analysis for the languages with a rather free word order. Proceedings of the 16th NODALIDA, 13–20, (2007)
2. Gruzitis, N., Pretkalnina, L., Saulite, B., Rituma, L., Nespore-Berzkalne, G., Znotins, A., and Paikens, P.: Creation of a Balanced State-of-the-Art Multiayer Corpus for NLU. Proceedings of the 11th LREC, Miyazaki, Japan (2018)
3. Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D.: Universal Dependencies v1: A Multilingual Treebank Collection. Proceedings of the 10th LREC, 1659–1666 (2016)
4. Nespore, G., Saulite, B., Barzdins, G., and Gruzitis, N.: Comparison of the SemTi-Kamols and Tesniere's dependency grammars. Proceedings of 4th HLT—The Baltic Perspective, Frontiers in Artificial Intelligence and Applications, Vol. 219, IOS Press, 233–240 (2010)
5. Lokmane, I. Sintakse. In Latviešu valodas gramatika. Rīga: LU Latviešu valodas institūts, 692–766 (2013)
6. Pretkalnina, L., Nespore, G., Levane-Petrova, K., and Saulite, B.: A Prague Markup Language profile for the SemTi-Kamols grammar model. Proceedings of the 18th NODALIDA, Riga, Latvia, 303–306 (2011)
7. Pretkalnina, L., Rituma, L., and Saulite, B.: Universal Dependency treebank for Latvian: A pilot. Proceedings of 7th HLT—The Baltic Perspective, Frontiers in Artificial Intelligence and Applications, Vol. 289, IOS Press, 136–143 (2016)