Annotation schemes in North Sámi dependency parsing

Mariya Sheyanova
Fundamental and Applied Linguistics
Higher School of Economics
Moscow, Russia
masha.shejanova@gmail.com

Francis M. Tyers
Giela ja kultuvrra instituhtta
UiT Norgga árktalaš universitehta
N-9018 Romsa, Norway
francis.tyers@uit.no

Abstract

In this paper we describe a comparison of two annotation schemes for dependency parsing of North Sámi, a Finno-Ugric language spoken in the north of Scandinavia and Finland. The two annotation schemes are the Giellatekno (GT) scheme which has been used in research and applications for the Sámi languages and Universal Dependencies (UD) which is a cross-lingual scheme aiming to unify annotation stations across languages. We show that we are able to deterministically convert from the Giellatekno scheme to the Universal Dependencies scheme without a loss of parsing performance. While we do not claim that either scheme is *a priori* a more adequate model of North Sámi syntax, we do argue that the choice of annotation scheme is dependent on the intended application.

1 Introduction

Dependency parsing is an important step in many applications of natural language processing, such as information extraction, machine translation, interactive language learning and corpus search interfaces. There are a number of approaches to dependency parsing. These include rule-based approaches such as [1] for German and [2] for Portuguese, and statistical approaches, such as transition-based parsing, exemplified by MaltParser [3]. In rule-based approaches the knowledge source is a set of rules (such as constraints), while in statistical approaches the knowledge source is a collection of parsed sentences called *treebank*.

Both rule-based and statistical approaches have in common that the parses they output conform to a given annotation scheme, that is a set of rules which define what linguistic structure should be applied to given constructions. Annotation schemes can vary substantially according to how they encode an analysis of linguistic structure. For example, one scheme may decide that the auxiliary verb is the head of an auxiliary—main verb construction because of subject agreement, while another may decide that the main verb is the head because of case government of arguments.

The choice of representation can depend heavily on application. For certain applications, such as grammar checking, a more morphosyntactic scheme may be appropriate, while for others, such as machine translation, a more syntacto-semantic scheme may be more appropriate.

In this paper, we describe the conversion of the annotation schemes output by the Giellatekno parser to a corresponding UD-compliant scheme. We also provide comparative statistics on the composition of the corpus used in the experiments before and after the conversion. Afterwards, we report a comparison of parsing results using the two annotation schemes. The paper is structured as follows: Section 2 describes the corpus and source annotation scheme used in this paper; section 3 describes the conversion procedure and the target annotation scheme; section 4 describes an experiment in comparing annotation schemes and finally section 5 presents some concluding remarks.

2 Corpus

The corpus is a collection of sentences in North Sámi from a variety of genres (literature, news, religion, grammar examples) which have been manually disambiguated and annotated for shallow syntactic function. There are a total of 3,682 sentences comprising 35,061 tokens. In order to produce a treebank, this disambiguated corpus was processed with a rule-based parser, as described below.

2.1 Rule-based parser

Giellatekno's rule-based dependency parser [4] is based on the VISL Constraint Grammar formalism [5]. The parser consists of 394 rules, of which 242 are head-assignment rules in the form "set the head of token t to the head matched by the pattern p", and 152 are label-assignment rules in the form "set the label of token t to l in context c". The following two examples illustrate these rule types.

```
SETPARENT @SUBJ TO (*1 VFIN)
```

Set the head of a token with the function subject to the finite verb to the right

```
SUBSTITUTE (@FMV) (@FS-<SUBJ) TARGET V (-1 SPRED) (0 Qst)
```

A finite-main verb is a clausal subject if there is a previous predicate
and the current verb has a question marker

The parser is designed to be run as the last stage of a pipeline that consists of finite-state morphological analysis and constraint-grammar-based morphological disambiguation and shallow-function labelling. The same parser is used for other Sámi languages, such as Lule Sámi and South Sámi and has also been applied to parsing South Sámi, Faroese and Greenlandic. The parser has an F-score of 0.99 for North Sámi [4], but in practice this may be lower (see section 3.2 for details).

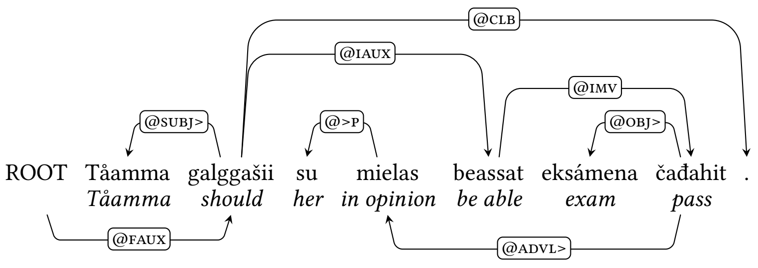
2.2 Annotation scheme

The dependency structure produced by the parser is a compromise between Sámi grammatical tradition and the conventions used in the VISL project [2]. The core distinctions in the scheme for verbs are: finite versus non-finite verbs, main verbs versus auxiliary verbs, and main clauses versus subordinate clauses. For nominal modifiers, the core distinction is the part-of-speech category of the head.

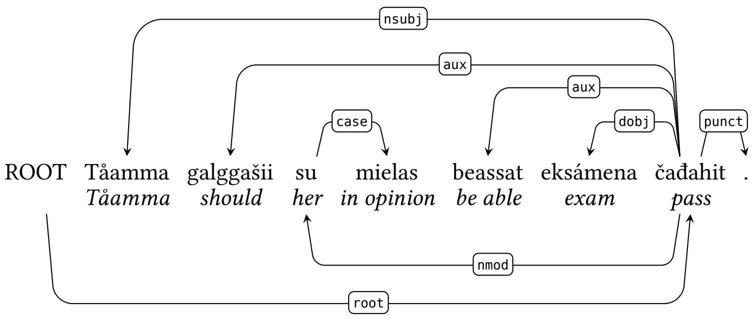
For verb complexes (i.e. a main verb along with any auxiliary or modal verbs), the annotation scheme follows the chained analysis, where the finite auxiliary is the head subsequent verbs are chained according to government. This analysis also applies to the negative verb, which is head when it is finite.

In co-ordination, the first conjunct is the head and subsequent conjuncts attach to the first conjunct. Conjunctions attach to their immediately preceding conjunct, and the dependency relation for each conjunct is the same as the head. There is a distinction between local conjunction (for example between modifiers) and global conjunction (between finite-verb clauses).

North Sámi has both prepositions and postpositions, in both cases the adposition is the head of adpositional phrase, and the head of the noun-phrase complement is



(a) Giellatekno



(b) Universal dependencies

Figure 1: Two structures for the sentence *Tåamma galggašii su mielas beassat eksámena čađahit.* 'Tåamma should in her opinion be able to pass the exam.' which illustrate different principles behind the two annotation schemes in head assignment and labels. Note that the Giellatekno annotation contains a non-projective dependency between *čađahit* 'to pass' and *mielas* 'in the opinion of'.

the dependent. For numeral phrases, if the numeral takes the case of its function, e.g. object in accusative and the nominal is in the genitive (2a), then the numeral is head, while if the numeral agrees with the nominal or is in the attributive form then the nominal is head (2b).

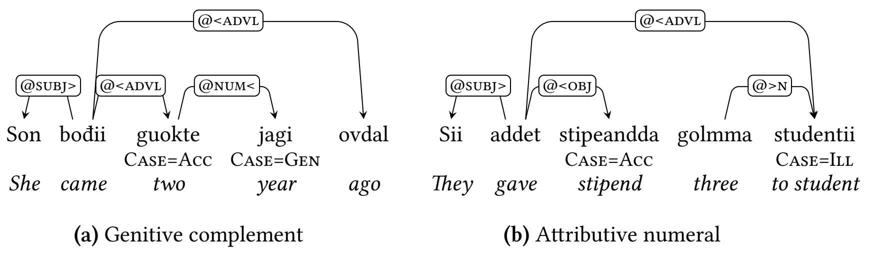


Figure 2: Annotation of numeral phrases in the Giellatekno scheme

Throughout the article we use the convention of prefixing an @ symbol to dependency labels using the Giellatekno scheme, labels without the @ are Universal dependency labels.

3 Conversion

3.1 Universal dependencies

Universal Dependencies [6] is an international collaborative project to make cross-linguistically consistent treebanks available for a wide variety of languages. In the UD annotation scheme, to improve cross-linguistic compatibility, dependency relations are primarily between content words, with function words attaching as leaf nodes. The motivation for this is that content words are more stable between languages, while languages can vary with how e.g. cases are used as opposed to adpositions, and analytic versus synthetic tense constructions. Thus, in auxiliary–main verb constructions, the main verb is the head and the auxiliary is attached as a dependent, if there is more than one auxiliary they are attached as siblings as opposed to a nested structure. In adpositional phrases, the complement of the adposition is the head and the adposition itself is a dependent attached with the `case` relation.

In terms of label categories, in contrast to the Giellatekno scheme where labels for non-verbal modifiers indicate the part of speech of the head of the relation, in the UD scheme, labels indicate the part-of-speech of the modifier. For example a noun phrase or adpositional phrase receives the label `nmod` ‘nominal modifier’, while an

Rule type	Count
$a \to b$	12
$a \to b$ with context	47
$a \to b$ with tree context	11
$a \to b$ with tree transformation	16
Total:	86

Table 1: Statistics on rule-types. $a \to b$ denotes that the rules change GT label a to UD label b. There was one additional rule which performed a tree transformation with no relabelling operation to reattach punctuation.

adjective modifying a nominal has the relation `amod` 'adjectival modifier'. Like in the Giellatekno scheme, a distinction is made between core arguments (subject, object, complement clause) and obliques (such as adverbials).

3.2 Preprocessing

An important step in the conversion process is the removal of sentences for which the rule-based parser produces a malformed tree. We count as malformed those trees which either: (a) have more than one node attaching to the technical root, (b) have cycles, (c) have no node attaching to the technical root, or (d) have orphan nodes, that is tokens which do not have a head. Out of a total of 3,682 sentences in the original corpus, 128 fall into class (b) or (d), 42 fall into class (a) and 8 fall into class (c). This leaves 3,504 sentences, or 30,955 tokens to be converted.

3.3 Rules

In order to convert from the Giellatekno annotation scheme to Universal dependencies, several types of transformations need to be carried out. The simplest are label replacements that do not require any context, for example replacing the Giellatekno label @FS-SUBJ with the UD label `csubj`. In this case we do not even need to look at the part-of-speech of the node as it is encoded in the label.

The second type are label replacements that require looking at other features of the node as context, for example to replace the Giellatekno label @>N we need to determine if the current token is a noun in genitive (in which case we output `nmod:poss`), a noun in a locative case (in which case we output `nmod`), an adjective (in which case we output `amod`), a demonstrative pronoun (in which case we output `det`).

The third type are label replacements that require looking at tree context but do not do any tree transformations, for example to convert the Giellatekno label @FMV, if the head is the technical root, then it is converted to `root`, if the head is a speech verb then we convert it to `ccomp`, if the head is another @FMV and there is no explicit coordinator then we convert it to `parataxis`.

The fourth type are rules that include both tree transformations (switching head-/dependent or moving nodes in the tree) and label substitutions. For example for adpositional phrases, the adposition is moved to be the dependent of the head nominal in the NP and label is changed to `case`. The label which attached the adposition to its head is retained and now is the relations between the head of the PP and the head noun.

All of the rules are implemented in XSLT [7], which is a declarative language for transforming XML trees. The transformations are run as a pipeline, with the output of one rule feeding as the input of the next. In general, rules are ordered by complexity with more complex transformation rules being run first. Table 1 gives a summary of the rule types and frequency of rule types. The transformation rules are available online.[1] In addition to the linguistic rules we also wrote a rule for reattaching punctuation in order to try and reduce the amount of non-projectivity.

3.4 Postprocessing

The rules described in section 3.3 may produce invalid trees, either as a result of an error in the output of the rule-based parser, or an error in the transformation rules.[2] We have tried to minimise the number of errors in the rules, but sometimes it might not be clear what the best course of action is. With regard to errors in the original parser output, one possibility would be to fix them manually before conversion or fix the rules, but this was not done in order to preserve reproducibility. Parser errors were however reported upstream to the authors of [4]. To remove invalid trees from the final output we ran the same validation scripts that were used to detect invalid trees in the input. After applying postprocessing we were left with 3,304 sentences, and 29,354 tokens. A summary of the statistics for the converted sentences can be found in Table 2

[1]`https://github.com/ftyers/UD_North_Saami`

[2]While the XSLT-based parser will not produce formally-invalid XML, because of the data format, which specifies linear order with an order attribute, it may produce trees where an order attribute points to a node that has been deleted.

	Giellatekno	**Universal dependencies**
Tokens	29,354	29,354
Sentences	3,304	3,304
Projective	2,596	2,927
Non-projective	708	377
Avg. distance from head	2.59	2.75
Relations	44	32
Relations (incl. direction)	58	32
Avg. labels/lemma	1.83	1.57

Table 2: Comparative statistics for the Giellatekno and UD annotation schemes. Relations (including direction) indicates the number of labels if head-direction indicators are taken into account.

4 Experiment

In order to test the utility of the two annotations schemes in a real-world setting, we trained and evaluated a number of models using the popular UDpipe toolkit [8]. UDpipe is a toolkit for data-driven tokenisation, part-of-speech tagging and dependency parsing; it learns a statistical model for each of the tasks from treebank data and applies this model to process unseen sentences.

We perform 10-fold cross-validation by randomising the order of sentences in the corpus and splitting them into 10 equally-sized parts. In each iteration we held out one part for testing and used the rest for training. We trained UDpipe for 10 epochs, the default setting. We calculated the labelled-attachment score (LAS) and unlabelled-attachment score (UAS) for each of the models. In addition to the LAS and UAS we also calculated the raw label accuracy. The same splits were used for both annotation schemes. Table 3 presents the results for each of the schemes.

Scheme	**UAS**	**LAS**	**Labels**
Giellatekno	[91.2, 93.4]	[88.8, 91.1]	[92.58, 94.83]
Universal dependencies	[78.2, 83.4]	[76.0, 80.9]	[78.00, 81.81]

Table 3: Results for parsing from the two annotation schemes, intervals are the high/low scores from 10-fold cross validation.

Despite the smaller label set, the UD annotation scheme performs worse than the GT scheme.

5 Concluding remarks

We have presented a comparison of two annotation schemes for dependency parsing of North Sámi. The labels and annotation guidelines in the Giellatekno scheme are closely coupled with Sámi morphosyntax, while the Universal dependencies scheme aims to be more cross-linguistically consistent. Preliminary results show that we are able to achieve better parsing performance using the Giellatekno annotation scheme, this is in contrast to other work such as [9] who find better results using Universal dependencies.

In terms of future work, we aim to convert the part-of-speech tags and features to UD-standard ones, and work with authors of the other Uralic treebanks to improve cross-linguistic compatibility. With the release of version 2.0 of the UD guidelines we also plan to convert update the rules to produce trees in line with the new guidelines. It is also our intention to investigate why the parsing performance is substantially worse for the UD-based representation, and to manually validate a proportion of the data to use as a gold standard in future experiments.

Acknowledgements

We are grateful to Trond Trosterud for help with the Giellatekno annotation scheme and to the reviewers for their helpful comments.

References

[1] I. Schröder, W. Menzel, K. Foth, and M. Schulz. “Modeling dependency grammar with restricted constraints”. In: *Traitement Automatique des Langues* 41.1 (2000), pp. 113–144.

[2] E. Bick. “Turning Constraint Grammar Data into Running Dependency Treebanks”. In: *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theory, TLT4*. Ed. by M. Civit and S. Kübler. 2005.

[3] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryiğit, S. Kübler, S. Marinov, and E. Marsi. “MaltParser: A language-independent system for data-driven dependency parsing”. In: *Natural Language Engineering* 13.2 (2007), pp. 95–135.

[4] L. Antonsen, T. Trosterud, and L. Wiechetek. “Reusing Grammatical Resources for New Languages”. In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'10)*. Malta: European Language Resources Association (ELRA), 2010.

[5] E. Bick and T. Didriksen. “CG-3 – Beyond Classical Constraint Grammar”. In: *Proceedings of the 20th Nordic Conference of Computational Linguistics, NODALIDA 2015, May 11-13, 2015, Vilnius, Lithuania.* 109. Linköping University Electronic Press, Linköpings universitet, 2015, pp. 31–39.

[6] J. Nivre, M.-C. de Marneffe, F. Ginter, Y. Goldberg, J. Hajič, C. D. Manning, R. McDonald, S. Petrov, S. Pyysalo, N. Silveira, R. Tsarfaty, and D. Zeman. “Universal Dependencies v1: A Multilingual Treebank Collection”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16).* 2016.

[7] J. Clark. *XSL Transformations (XSLT) Version 1.0.* World Wide Web Consortium, Recommendation REC-xslt-19991116. Nov. 1999.

[8] M. Straka, J. Hajič, and J. Straková. “UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16).* Portorož, Slovenia: European Language Resources Association (ELRA), May 2016. ISBN: 978-2-9517408-9-1.

[9] U. Sulubacak, M. Gökırmak, F. M. Tyers, Ç. Çöltekin, J. Nivre, and G. Eryiğit. “Universal dependencies for Turkish”. In: *Proceedings of COLING 2016.* 2016.