Many shades of grammar checking – Launching a Constraint Grammar tool for North Sámi

Linda Wiechetek linda.wiechetek@uit.no
Sjur Moshagen sjur.n.moshagen@uit.no
Børre Gaup borre.gaup@uit.no
Thomas Omma thomas.omma@uit.no

Divvun UiT Norgga árktalaš universitehta

1 Introduction

This paper discusses the characteristics and evaluation of the very first North Sámi spell- and grammar checker. At its launch it supports *MS Word* and *GoogleDocs*[1], cf. Figure 1. We describe its component parts, the technology used, such as *Constraint Grammar* (Karlsson, 1990; Karlsson et al., 1995; Bick and Didriksen, 2015) and *Hfst-pmatch* (Hardwick et al., 2015), and its evaluation with a new evaluation tool specifically designed for that purpose.

Only the modules of the full-scale grammar checker described in Wiechetek (2017) that have been tested and improved sufficiently for the public to use are released in this launch. More advanced syntactic errors will be included at a later stage. The grammar checker modules are an enhancement to an existing North Sámi spellchecker (Gaup et al., 2006), following a philosophy of *release early, release often*, and using continuous integration (*ci*) and continuous delivery (*cd*) to deliver updates with new error correction types, as new parts of the grammar checker are sufficiently tested. Releasing at an early stage of development gives the user community early access to improved and much needed tools and allows the developers to improve the tools based on the community's feedback, essentially preferring early and frequent releases over feature-based releases to allow for a close feedback-based relation between developers and users.[2]

We started moving towards a "release early, release often" strategy in 2018 when working on the *Divvun installer*. The *Divvun installer* is a tool to simplify the installation and updates of the tools developed by the *Divvun* group. The users only have to install the *Divvun installer* app, and then select the languages they are interested in – everything is then installed and configured automatically. It also checks for updates to the installed tools with regular intervals, and allows us to provide updates to our users automatically. That makes it the center-piece of our release early, release often strategy.

Version	Date
Divvun 1.0	(2007-12-12)
Divvun 1.0.1	(2007-12-21)
Divvun 1.1	(2008-12-17)
Divvun 2.0	(2010-12-08)
Divvun 2.1	(2011-03-21)
Divvun 2.2	(2011-11-23)
Divvun 2.3	(2013-02-08)
Divvun 2.3	(2013-02-08)
Divvun 3.0	(2013-06-13)
Divvun 4.0	(2015-12-17)
Divvun 4.0.1	(2016-03-17)
Divvun 4.1	(2016-12-15)
Divvun 4.2	(2018-12-20)
GramDivvun 1.0 beta	**(2019-09-27)**

Table 1: *Divvun* release history (2007-2019)

This is a change of strategy based on previous experience with the feature-based release strategy, which presupposed extensive manual testing to avoid regressions. This leads to long release cycles and up to two years between the releases as can be seen in table 1. It also leads to less word coverage for the users while the system had already access to a larger lexicon.

Over the years, we have introduced automatic testing methods to maintain quality and avoid regressions. The automatic tests are integrated into our CI/CD system and cover errors we have experienced earlier, and as we find new, uncovered

[1] https://gsuite.google.com/marketplace/app/divvun_grammar_checker/611280167256

[2] https://medium.com/@warren2lynch/scrum-philosophy-release-early-release-often-a5b864fd62a8

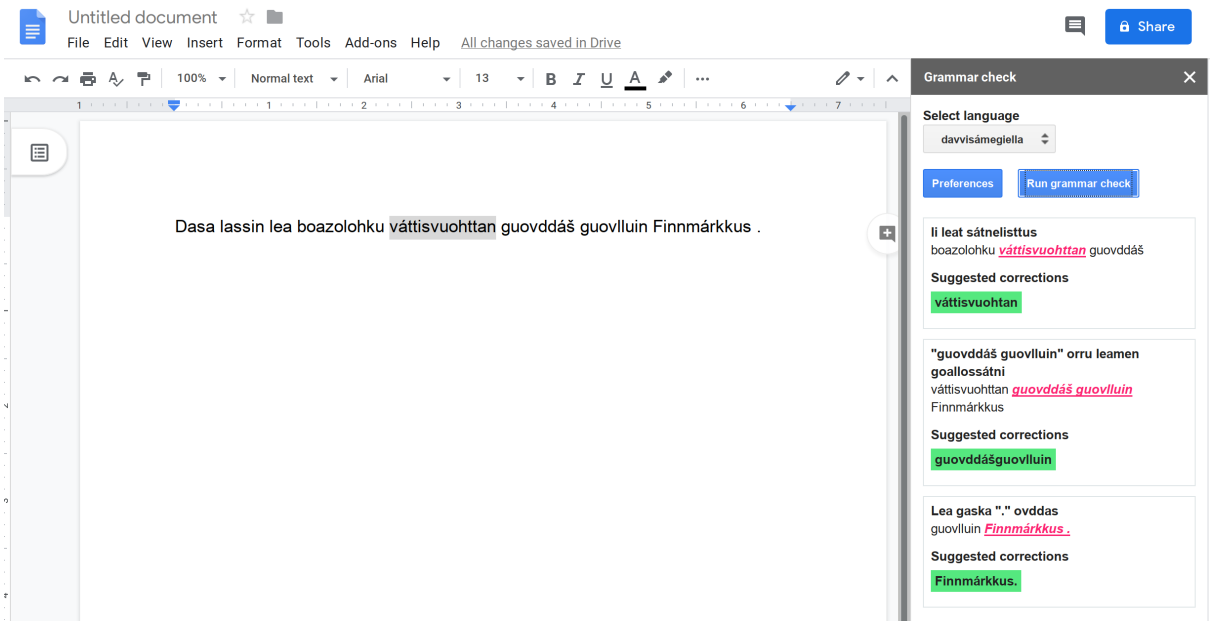

Figure 1: *GramDivvun 1.0 beta* in GoogleDocs

areas, we add tests for these as well. Automatic CI/CD means that if new commits do not break our tests, a new version of our software is built and released automatically. If tests fail, the nightly releases will not be built before the tests pass again. This assures us that the nightly releases will not contain regressions compared to earlier releases, and yet incorporates new words and features.

North Sámi is a Uralic language spoken in Norway, Sweden and Finland by approximately 25 700 speakers (Simons and Fennig, 2018). These countries have other majority languages, making all Sámi speakers bilingual. Bilingual users frequently face bigger challenges regarding literacy in the lesser used language than in the majority language due to reduced access to language arenas (Outakoski, 2013; Lindgren et al., 2016). Therefore, tools that support literacy, like spelling and grammar checkers, are more important in a minority language community than in a majority language community.

The released grammar checker is a light grammar checker in the sense that it detects and corrects errors that do not require rearranging the whole sentence, but typically just one or several adjacent word forms based on a grammatical analysis of the sentence. Additionally, a number of formatting errors are covered. The main new features are implemented by means of several Constraint Grammar-based modules. These include correction of formatting and punctuation errors, filtering of speller suggestions, much improved tokenisation and sentence boundary detection, as well as advanced compound error analysis and correction.

This paper shows how a finite-state based spellchecker can be upgraded to a much more powerful spelling and grammar checking tool by adding several Constraint Grammar modules.

2 Framework

2.1 Language tools

An open-source spelling checker for North Sámi has been freely distributed since 2007[3], the beginnings of which have been described by Gaup et al. (2006). The tool discussed in this paper includes the open-source spelling checker referenced above, but further developed and using the hfst-based spelling mechanism described in Pirinen and Lindén (2014). The spelling checker is enhanced with five Constraint Grammar modules, cf. Figure 2. It should be noted that the spelling checker is exactly the same as the regular North Sámi spelling checker used by the language community, but with the added functionality that all suggestions are returned to the pipeline with their full morphological analysis. The analyses are then used by subsequent Constraint Grammar rules during disambiguation and suggestion filtering.

[3] http://divvun.no/korrektur/korrektur.html

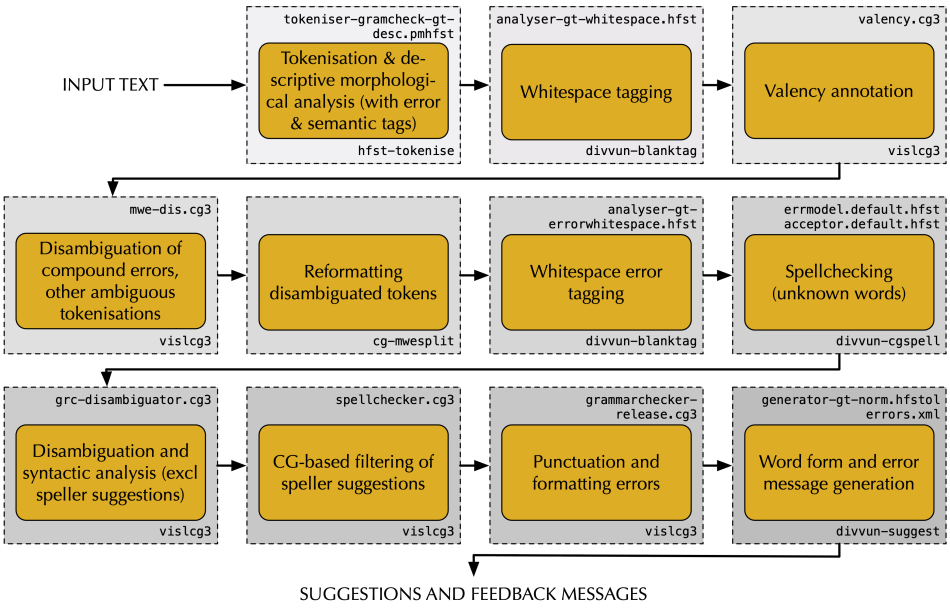

Figure 2: System architecture of *GramDivvun 1.0 beta*, the North Sámi grammar checker

All components are compiled and built using the *GiellaLT* infrastructure (Moshagen et al., 2013). Five constraint grammar modules, i.e. a valency grammar (*valency.cg3*), a tokenizer (*mwe-dis.cg3*), a morpho-syntactic disambiguator (*grc-disambiguator.cg3*), a disambiguation module for spellchecker suggestions (*spellchecker.cg3*) and a module for more advanced grammar checking (*grammarchecker-release.cg3*) are included in the spelling and grammar checker.

The current order of the modules has shown to be the most optimal one for our use and has been established during the work with the grammar checker. It follows the principle of growing complexity, and information necessary to subsequent modules is made available to them. Valencies for example are used in the disambiguation of compounds, which is why the module precedes the multi-word disambiguation module. The first whitespace tagging module precedes sentence boundary disambiguation because it is used to insert hints about the text structure. It marks, for example, first and last words in the paragraph, which is relevant when deciding if a period marks the end of sentence or is part of an abbreviation or a numeric expression. The second whitespace analyser is applied after the multiword disambiguation, but could also be applied later. Its purpose is to tag potentially wrong use of whitespace, and must be added before the final grammar checking module, but any position between the multiword disambiguation and grammar checking is going to work. Spellchecking is performed before disambiguation so that more sentence context is available to the syntactic analyser and disambiguator.

It should be noted that we do not use a guesser for out-of-vocabulary words. A major part of new words are formed using productive morphology of the language, like compounding and derivation, both of which are encoded in the morphological analyser. Also, the lexicon is constantly being updated, so that proper nouns and other potential out-of-vocabulary words will quickly be covered. As updates are made available frequently, this should not be a major issue for users, but some issues are discussed towards the end of this article.

There are both language specific and language independent parts in the *GiellaLT* infrastructure. The *GiellaLT* infrastructure is developed at *UiT The Arctic University of Norway* to support the development of language technology for the Sámi languages. It covers language independent components,[4] as well as language-specific code with a supporting build system.[5] Although initially developed for the Sámi languages, the infrastructure has been built with language independence in mind, and presently there are about a hundred languages in various stages of development in our infrastructure. The linguistic code is gen-

[4] cf. `https://github.com/divvun`

[5] cf. `https://gtsvn.uit.no/langtech/trunk/`

erally language-specific. However, the annotation of syntactic functions and dependencies is generalized for several related languages (South Sámi, Lule Sámi, North Sámi, and Pite Sámi) (Antonsen et al., 2010). The technical code on the other hand is language-independent. The infrastructure is not only valid for North Sámi, but can directly be used by any language in the *GiellaLT* system, e.g. other Sámi languages as well as any other language. Presently there is an early version of a working Faroese grammar checker in addition to the North Sámi one, and initial work has started for a number of the other Sámi languages.

The system does error detection at four different stages of the pipeline. Non-word typos are marked by means of the spellchecker. Secondly, a Constraint Grammar module marks whitespace errors in punctuation contexts based on input from the second whitespace analyser. Compound errors are identified by means of a Constraint Grammar-based tokenisation disambiguation file. And a fourth Constraint Grammar module for advanced grammatical errors and punctuation marks quotation errors.

As a tool intended to be used in production by regular users, it targets all types of errors, from technical typesetting errors such as wrong quotation marks and faulty use of spaces, via spelling errors to advanced grammatical error detection and correction. In the evaluation all of these are counted as grammar checker errors, as we want to evaluate the overall performance of the tool. The only errors not included in the evaluation are those that we do not target at all (which are quite a few in this first beta release).

The sentence in ex. (1) includes a typo (*Norgag* should be *Norgga* 'Norway' (Gen.)), a space error (before '.') and a compound error (*iskkadan bargguin* should be *iskkadanbargguin*) 'survey' (Loc. Pl.). In addition, there is a congruence error, i.e. *dáid* (Gen. Pl.) 'these' should be *dáin* (Loc. Pl.), i.e. it should agree in case and number with *iskkadan bargguin*. The launched grammar checker can detect the first three errors, but not the last one, since syntactic error rules are not included in this initial launch of the grammar checker.

(1) Oktiibuot 13 **Norgag** doaktára leat
altogether 13 Norway.GEN doctor.GEN have
leamaš mielde dáid
been with these.GEN
iskkadan bargguin_.
testing work.LOC.PL
'Altogether 13 Norwegian doctors have participated in these surveys.'

The whitespace analyser detects an erroneous space before '.' The suggested correction is *"<iskkadan bargguin.>"*. The tokenisation disambiguation module detects the compound error and suggests the combination of lemma and tags *iskkadanbargu+N+Sg+Com* resulting in the form *iskkadanbargguin*. The tokeniser is used to disambiguate between syntactically related n-grams and misspelled compounds, where the misspelling is an erroneous space at the word boundaries.

This module is clearly checking more than spelling conventions, i.e. grammar, as writing, for example, two consecutive nouns as one or two words has syntactic implications. Such noun-noun combinations do not necessarily need to be compounds even if the first element is in nominative case. They can also be syntactically related as in the agent construction in ex. (2) where *addin* 'giving' is a (nominalized) non-finite verb that modifies the second noun, *vuođđu* 'basis'.

This grammar checker finds compound errors by distinguishing between syntactic readings as the previous one and compound readings as *addinvejolašvuođa* 'giving possibility (Gen.)'.

(2) luossabivdu lea lunddolaš
salmon.fishing is natural
Golf-rávnnji **addin**
Golf-stream.GEN give.ACTIO.NOM
vejolašvuođa vuođđu
possibility.GEN basis
'salmon fishing is a natural resource for a possibility given by the Golf-stream'

2.2 Evaluation tools

Previously, we had evaluated our tools manually, but now we wanted to be able to measure the improvement of our tools more consistently.

2.2.1 Annotated corpus

An evaluation tool presupposes an annotated corpus. The corpus we use for evaluation contains 226 336 words and is manually annotated by a North Sámi speaker/linguist according to the following principles.[6]

[6]The raw corpus texts for the openly accessible corpus can be found at `https://gtsvn.uit.no/freecorpus/orig/sme`. The corpus used for the evaluation in this article can also be found at `https://gtsvn.uit.no/`

The tokens involved in the error are enclosed in parenthesis followed by a sign for their general error type (orthographic, real word, morpho-syntactic, syntactic, lexical, formatting, foreign words), which is followed by another parenthesis containing error subclassification and the expected correction. The subclassification may contain annotation that is customized for the main six different error types. Sometimes part of speech, and morpho-syntactic criteria are specified or the error is further explained, e.g. regarding congruence. The description is followed by a correction of the error after the pipe sign. In ex. (3), the tokens involved in the error are nouns, the syntactic error is a compound error and the correction is *riikkačoahkkinmáhpas*.

(3) Áššebáhpáriid gávnnat (riikkačoahkkin máhpas)¥(noun,cmp|riikkačoahkkinmáhpas)

Orthographic errors (marked by $) include non-words only. They are traditional misspellings confined to single (error) strings and the traditional speller should detect them. Real word errors (marked by ¢) cannot be detected by a traditional speller. Morpho-syntactic errors (marked by £) are case, agreement, tense, mode errors. They require an analysis of (parts of) the sentence or surrounding words to be detected. Syntactic errors (marked by ¥) require a partial or full analysis of (parts of) the sentence or surrounding words; word order, compound errors, missing words, redundant words and so on. Lexical errors (marked by €) include wrong derivations. Foreign words (marked by ∞) include single words in other languages like, for example, Norwegian *og* and *august*. Formatting errors (marked by ‰) include spacing errors in combination with punctuation.

(4) an.lávdegoddi
?.committee

When doing error mark-up, it can be challenging to find the appropriate corrections in cases like ex. (4), where a copy-paste error results in the first part of the word missing. Without the original text, it can be close to impossible to reconstruct the intended form from the remaining text.

freecorpus/orig/nodalida2019

2.2.2 Evaluation tool

The evaluation tool is written in python. It reads the corpus files, runs the grammar checker on the original version of each paragraph, compares the output with the gold standard markup in the corpus, and collects the results. The output is written to a text file, with a report for each paragraph and also the overall evaluation measures.

The paragraph report covers mark-up errors that do not correspond to grammar checker errors (usually false negatives) and grammar checker errors that do not correspond to mark-up errors (false positives or missing mark-up). When mark-up and grammar checker errors align, the report covers whether or not the mark-up errors correction is among the grammar checker suggestions, and reports any missing suggestions by the grammar checker.

The overall report provides precision, recall and F-score for all errors and also breaks these numbers down for the respective error classes.

As the grammar checker does not presently cover neither lexical, morpho-syntactic nor real word errors, these error types are filtered away as they are read in by the evaluation tool, such that the expected correct text as given by the corpus mark-up is the text used as input to the testing. As we expand the coverage of the grammar checker, new error types will be used in testing it as well.

3 Evaluation

3.1 Quantitative evaluation

We calculate both precision and recall.

$$\textbf{Precision} = \frac{\text{number of items correctly retrieved}}{\text{number of items actually retrieved}}$$

$$\textbf{Recall} = \frac{\text{number of items correctly retrieved}}{\text{number of items that should have been retrieved}}$$

A previous (manual) evaluation of compound error detection, which is the linguistically most advanced error type in *GramDivvun*, has resulted in a precision of 76.6% and a recall of 78.6% (Wiechetek et al., 2019). The development of the grammar checker evaluation tool presented in this paper has been informed by the previous evaluation, and many errors in the infrastructure and in the Constraint Grammar modules have been corrected. Therefore, higher precision and recall are expected in this evaluation.

As opposed to the evaluation done in Wiechetek et al. (2019), the evaluation of *GramDivvun 1.0 beta* is automatic. The evaluation is based on version *r183544* of the grammar checker[7]. An automatic evaluation is meaningful as the error corrections intended in this launch typically include a limited amount of adjacent word forms, which is relatively straightforward. Reference-less approaches as proposed in Napoles et al. (2016) are not an option as they require pre-existing and independently developed tools that are sensitive to grammatical errors, a luxury not available to most minority languages.

A part of the North Sámi *SIKOR* corpus (SIKOR2016) containing manual error markup for orthographical and grammatical errors is used as the evaluation corpus. It consists of 226 336 words. The genres represented in the evaluation corpus are news, blogs, and teaching materials.[8]

In the evaluation, we only consider spelling errors (only non-words), compound errors, space errors and punctuation errors (only quotation marks for now), i.e. only the error types we actually try to correct. The performance of the grammar checker is measured by means of precision and recall. Good precision has typically priority over good recall as users tend to react more critically to flagging correct input as errors as opposed to not flagging error input.

The results are presented in Table 2. Both precision and recall are above 85%, i.e. above the previous results for compound error detection. As a reference, when Bick (2015) evaluates his full-fledged grammar checker *DanProof*, for correcting both, spelling and compounding errors precision is 90.8% and recall is 86.8%. While our recall is close to Bick's results, precision can definitely be improved. Orthographic error detection (spell checking) performs best (87.3%), a good improvement compared to earlier results.[9] Formatting error detection performs slightly worse (82.2%). The results for syntactic (compound) error detection are lowest of all error types (73.9%), and slightly lower than the results in Wiechetek et al. (2019). This may be partly due to a smaller amount of compound errors in this corpus (344) compared to the one in Wiechetek et al. (2019) (458). The qualitative evaluation below will shed light on other possible reasons for *GramDivvun* shortcomings.

Measure	Overall	Orth	Format	Syn
Precision	85.4%	86.1%	85.0%	75.0%
Recall	85.8%	88.5%	79.6%	72.9%
F-Score	85.6%	87.3%	82.2%	73.9%
TP	1.319	991	277	51
FP	226	160	49	17
FN	219	129	71	19

Table 2: Precision, recall and F-Score of GramDivvun 1.0 beta (TP = true positives, FP = false positives, FN = false negatives)

3.2 Qualitative evaluation

In this section we analyse and discuss reasons for shortcomings in precision and recall in different modules of the grammar checker. Common reasons for false positives and false negatives are lexicon- and fst-related issues. These are, for example, missing items in the lexicon, cited fragments in other languages that do not receive an analysis and errors the treatment of clitics. Furthermore, there are named entity related issues (427 cases), misspelled compounds (i.e. written apart) that are not listed as such in the lexicon, and misspelled compounds that are not listed as such (split compounds with an error in the first part). In addition to lexicon-related issues, there are shortcomings in the disambiguation rules and in the error detection modules of the grammar checker, where Constraint Grammar rules do not recognize syntactically related words, and analyses them as compounds. Other reasons are related to the whitespace analyser, where space errors are erroneously detected because we do not recognize first/last words in a sentence.

[7] To obtain this version run: `svn co -r183544 https://gtsvn.uit.no/langtech/trunk langtech; cd langtech/giella-core; ./autogen.sh; ./configure; cd ../giella-shared; ./autogen.sh; ./configure; cd ../langs/sme; ./autogen.sh; ./configure -with-hfst -without-xfst -enable-grammarchecker -enable-tokenisers -enable-alignment -enable-reversed-intersect; make -j`

[8] `https://giellalt.uit.no/proof/nordplus/StavekontrolltestingOgNorplusprosjektet.html` (Retrieved 2019-06-19)

[9] `https://gtsvn.uit.no/biggies/trunk/techdoc/proof/spelling/testing2/sme/to/goldstandard/20180207-1355-corpus-gs-results.html`

3.2.1 False positives

To identify shortcomings in precision false positives need to be analysed.

One issue are shortcomings in the compound disambiguation rules, where in the case of two correct alternative spellings (hyphenated vs. non-hyphenated version and one word vs. two-word version), a correct one is replaced by the other correct one. In ex. (5), *Riddu Riđđu* is written without a hyphen (which is correct). However, the grammar checker corrects it to *Riddu-Riđđu*. This issue can be resolved by explicitly tagging the multiword spellings in the lexicon and making an exception in the Constraint Grammar rules that annotate an error.

(5) Guollefestivála geassemánus ja
fish.festival june.LOC and
Riddu Riđđu suoidnemánus.
Riddu Riđđu July.LOC
'The fishing festival in June and Riddu Riđđu in July'

In the case of first and last names, hyphenated and non-hyphenated versions of the same name are very common. The spelling of human names exhibits a great variation, and names are hard to standardize.

In ex. (6), *Inger Anna* is falsely corrected to *Inger-Anna* by the compound error detection module. Because of the great productivity of names, Constraint Grammar rules should contain special exceptions for human names.

(6) Sámegillii: **Inger Anna** Gaup Gustad.
Sámi.ILL: Inger Anna Gaup Gustad
'In Sámi: Inger Anna Gaup Gustad.'

In ex. (7), *rahppo* receives an error tag by *GramDivvun* even though it is correct. The form *rahppo* receives a regular derived analysis (i.e. the passive analysis of *rahpat* 'open') and an underived lexical error analysis (i.e. the indicative analysis of *rahppot* 'be opened') by the morphological analyser. The disambiguator should ideally remove the lexical error analysis in favour of the correct one. The solution is an adaption of the disambiguator so that possible error-tag analyses that compete with correct readings are removed.

(7) Go Norga **rahppo** "Európái"
when Norway open.up Europe.ILL
'When Norway opens up to Europe'

There are also syntactic issues as in ex. (8). The grammatical tokeniser does not recognize syntactically related words like *prošeakta* 'project' and *virggálaččat* 'professionally', but analyses them as a compound *prošeaktavirggálaččat* '?project professionally' based on the fact that it is a denominal derivation of *prošeaktavirgi* 'project position'. Constraint Grammar rules should be changed as to recognizing the syntactic relation between the words. An alternative solution is blocking the compound reading of two-word expression for derivations in the lexicon.

(8) Dát lea čielga hálddahuslaš dássi, gos
this is clear administrative level, where
prošeakta **virggálaččat** registrerejuvvo
project professionally register.PASS.3SG
ja álggahuvvo.
and start.PASS.3SG
'This is a clear administrative level, where a project is registered and started professionally.'

3.2.2 False negatives

In order to find the reasons for shortcomings in recall we have to look at false negatives. In ex. (9), the proper noun compound *Sámiid Hilat* is not hyphenated as it should be according to the norm. However, *Sámiid* is in genitive case, and in this case the compound disambiguation module removes the error reading, as genitive nouns can be prenominal modifiers. A possible solution is to list and error-mark the non-hyphenated *Sámiid Hilat* in the lexicon.

(9) Artihkkal aviissas **Sámiid Hilat**,
Article newspaper.LOC Sámi.GEN Hilat,
nr. 2 - 1978
nr. 2 - 1978
'The article in the newspaper Sámiid-Hilat, nr. 2 - 1978'

Lexicon issues include the treatment of clitics. We treat clitics and stand-alone particles in the same way. That means that certain words get erroneously analysed as combinations with stand-alone particles even if these are never to be found in combination with other words. In ex. (10) *dálkadat* has a spelling error, and should be *dálkkádat*. However, since it receives an analysis based on *dálkká* and the particle *dat*, the spelling error is not found.

(10) Suohkanis lea buorre **dálkadat**
county.LOC is good climate
'The county has a good climate.'

Another problem is the exact span of the marked-up area. Space errors, for example, cannot be marked on the (potentially missing) space itself. Instead, the surrounding words are marked. Since the Constraint Grammar rule for tagging these errors are not properly constrained, error tags are erroneously added to further words. In ex. (11), the double space error between *šat* and *ságastallat* is not found.

(11) ja dalle dáid birra eat
and then these.GEN.PL about not.3PL
dárbbaš šat__ságastallat
need anymore talk
'and then we don't need to talk about these things anymore'

This is an instance of error overlapping, i.e., two errors have overlapping contexts, and only the last error is flagged and corrected. This is another result of mixing user interface considerations with error detection and correction code. We encode the linear scope of the visual error marking in the Constraint Grammar rules, including the relevant surrounding words. This should better be left to a component closer to the user interface, and the Constraint Grammar rules should only tag the error strings themselves. Then two consecutive errors would not get conflicting mark-up, and the user interface component can resolve adjacent errors in the most appropriate way.

3.3 Discussion of the automatic evaluation

There is a certain amount of discrepancy between the identified errors reported by *GramDivvun*, and the manually marked-up errors in the corpus.

The manual markup of ex. (12) classifies *skuvla(* 'school(' as missing space before the parenthesis and *Oahpit* 'students' as a typo. *GramDivvun* does also find these two errors, but does not distinguish between them in the string positions for the errors: it uses the same area for both of them. That means that the identified typo includes the opening parenthesis as well in the identified erroneous part of the input, even though technically only *Oahpit* should be a typo.

However, as mentioned above, the manual mark-up marks *skuvla(* 'school(' and *Oahpit* 'students' as separate errors with different string spans. The evaluation tool reformats and adjusts the spans of the *GramDivvun* output to match the manual markup. It is therefore possible to do meaningful and usable comparisons, which is necessary for an automatic evaluation.

(12) skuvla(Oahpit
school(students

The major insight from the automatic evaluation is that our present grammar checker design is not optimal. We currently encode user interface information together with error detection and correction information at an early stage, adding information of the span of the error. This complicates the Constraint Grammar rules for error markup and correction. Rules extend the error span to neighbouring words, which themselves can have errors, creating a spaghetti of interdependences.

Untangling this mess has been a major effort during the evaluation. A cleaner design that avoids these problems is one of the objectives for the final release of *GramDivvun 1.0*. This means marking the span of an error at a later state, closer to the user interface, so it does not interfere with the rules for the actual error identification and correction.

For now, the automatic evaluation tool identifies the proper span of the errors. However, some errors may remain, both, in the evaluation and in the reported results.

4 Conclusion

In this paper we have presented *GramDivvun 1.0 beta*, the first released combined spelling and grammar checker tool for North Sámi that, in addition to spelling errors, detects and corrects punctuation errors, compound errors and white space errors. Additionally, this work also describes the complete infrastructure for a full-scale grammar checker and facilitates the implementation of any kind of grammatical error correction as soon as these are considered to be working well enough to be released. The infrastructure is available for any language within the *GiellaLT* framework.

We have exploited many shades of Constraint Grammar at all stages of the grammar checking process. Constraint Grammar is used in most error detection and correction. This includes context-aware filtering of spelling suggestions. The overall F-Score is 85.6%. Based on the qualitative evaluation and systematic error-debugging many (frequent) error types could be resolved after evaluating *GramDivvun 1.0 beta*, and better results are expected for future evaluations. Overall precision (85.4%) is at the moment not better than overall

recall (85.8%).

In addition, we have developed an automatic evaluation method, which will facilitate quality maintenance and allow us to release updated versions of the grammar checker more frequently. We have learned that manual corpus mark-up can get substantial support from the automatic testing, as the grammar checker often finds more errors than the ones visible to the human eye. We have also learned that the evaluation tool needs constant surveillance to ensure concordance with the desired features to be evaluated. Finally, diverging principles of error mark-up and grammar checker error detection pose challenges to the automatic evaluation.

Future plans include adding a name guesser, improving lexicon coverage, adding Constraint Grammar rules for variations in the lexicon (avoiding false positives) and refining the syntactic rules in the compound error detection module. Additionally, we plan to remove the user-interface elements from the present Constraint Grammar rules. As the infrastructure is ready, a wide range of real word errors and syntactic error detection rules will be included in future releases and turn *GramDivvun* into a full-scale North Sámi grammar checker.

Acknowledgments

We want to thank Kevin Brubeck Unhammer for his profound work with the grammar checker infrastructure, for very valuable discussions regarding the design of the grammar checker, for helping out with certain types of advanced Constraint Grammar rules, and for building our first prototype in LibreOffice.

References

Lene Antonsen, Linda Wiechetek, and Trond Trosterud. 2010. Reusing grammatical resources for new languages. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, pages 2782–2789, Stroudsburg. The Association for Computational Linguistics.

Eckhard Bick. 2015. DanProof: Pedagogical spell and grammar checking for Danish. In *Proceedings of the 10th International Conference Recent Advances in Natural Language Processing (RANLP 2015)*, pages 55–62, Hissar, Bulgaria. INCOMA Ltd.

Eckhard Bick and Tino Didriksen. 2015. CG-3 – beyond classical Constraint Grammar. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NoDaLiDa 2015)*, pages 31–39. Linköping University Electronic Press, Linköpings universitet.

Børre Gaup, Sjur Moshagen, Thomas Omma, Maaren Palismaa, Tomi Pieski, and Trond Trosterud. 2006. From Xerox to Aspell: A first prototype of a north sámi speller based on twol technology. In *Finite-State Methods and Natural Language Processing*, pages 306–307, Berlin, Heidelberg. Springer Berlin Heidelberg.

Sam Hardwick, Miikka Silfverberg, and Krister Lindén. 2015. Extracting semantic frames using hfst-pmatch. In *Proceedings of the 20th Nordic Conference of Computational Linguistics, (NoDaLiDa 2015)*, pages 305–308.

Fred Karlsson. 1990. Constraint Grammar as a Framework for Parsing Running Text. In *Proceedings of the 13th Conference on Computational Linguistics (COLING 1990)*, volume 3, pages 168–173, Helsinki, Finland. Association for Computational Linguistics.

Fred Karlsson, Atro Voutilainen, Juha Heikkilä, and Arto Anttila. 1995. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter, Berlin.

Eva Lindgren, Kirk P H Sullivan, Hanna Outakoski, and Asbjørg Westum. 2016. Researching literacy development in the globalised North: studying trilingual children's english writing in Finnish, Norwegian and Swedish Sápmi. In David R. Cole and Christine Woodrow, editors, *Super Dimensions in Globalisation and Education*, Cultural Studies and Transdiciplinarity in Education, pages 55–68. Springer, Singapore.

Sjur N. Moshagen, Tommi A. Pirinen, and Trond Trosterud. 2013. Building an open-source development infrastructure for language technology projects. In *NODALIDA*.

Courtney Napoles, Keisuke Sakaguchi, and Joel R. Tetreault. 2016. There's no comparison: Reference-less evaluation metrics in grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2109–2115.

Hanna Outakoski. 2013. Davvisámegielat čálamáhtu konteaksta [The context of North Sámi literacy]. *Sámi dieđalaš áigečála*, 1/2015:29–59.

Tommi A. Pirinen and Krister Lindén. 2014. State-of-the-art in weighted finite-state spell-checking. In *Proceedings of the 15th International Conference on Computational Linguistics and Intelligent Text Processing - Volume 8404*, CICLing 2014, pages 519–532, Berlin, Heidelberg. Springer-Verlag.

SIKOR2016. 2016-12-08. SIKOR UiT The Arctic University of Norway and the Norwegian Saami Parliament's Saami text collection. **URL:** *http://gtweb.uit.no/korp (Accessed 2016-12-08)*.

Gary F. Simons and Charles D. Fennig, editors. 2018. *Ethnologue: Languages of the World*, twenty-first edition. SIL International, Dallas, Texas.

Linda Wiechetek. 2017. *When grammar can't be trusted – Valency and semantic categories in North Sámi syntactic analysis and error detection*. PhD thesis, UiT The Arctic University of Norway.

Linda Wiechetek, Kevin Brubeck Unhammer, and Sjur Nørstebø Moshagen. 2019. Seeing more than whitespace – Tokenisation and disambiguation in a North Sámi grammar checker. In *Proceedings of the third Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 46–55.