Is this the end? Two-step tokenization of sentence boundaries

Linda Wiechetek
UiT Norgga árktalaš universitehta
Divvun
linda.wiechetek@uit.no

Sjur Nørstebø Moshagen
UiT Norgga árktalaš universitehta
Divvun
sjur.n.moshagen@uit.no

Thomas Omma
UiT Norgga árktalaš universitehta
Divvun
thomas.omma@uit.no

Abstract

A period does not only mark the end of a sentence; it can also be part of an abbreviation and numerical expressions. When analysing corpus text linguistically we need to know where a sentence begins and where it ends. In traditional corpus analysis, typically a sentence is identified before linguistic analysis is performed. In this work we propose an approach where we do basic linguistic analysis before we decide what a sentence is. As the interpretation of a period after abbreviations and numerical expressions is ambiguous, we leave the ambiguity in the initial tokenization. In a second step we remove the ambiguity based on the linguistic context of the period. We compare the previous approach to the new one and show how the new two-step approach to tokenization improves the identification of sentence boundaries.

Abstract

Piste ei ole vain merkki lauseen päättämisestä, se voi myös olla osa lyhennettä tai numeroa. Kun analysoidaan korpustekstejä kielitieteellisesti, täytyy tietää, missä on lauseen alku ja loppu. Perinteisesti lauseen loppu on löydetty ennen kielitieteellistä analyysia. Tässä artikkelissa ehdotamme menettelyä, jonka mukaan kielitieteellinen perusanalyysi on tehty ennen lauseiden löytämistä. Koska lyhyen ja numeerisen ilmaisun jälkeisen pisteen tulkinta on epäselvä, jätämme epäselvyyden selvittämisen tekemättä prosessoimisen alkuvaiheessa. Toisessa vaiheessa poistamme epäselvyyden lauseen kielellisen kontekstin perusteella. Vertailemme aikaisemmat lähestymistavat uuteen jä näytämme, miten uusi kaksivaiheinen lähestymistapa jäsentämiseen parantaa lauserajojen tunnistamisen.

Abstract

Čuokkis ii dušše mearkkat cealkkaloahpa; muhtumin dat lea oanádusa oassi ja lohkosátnedajaldaga oassi. Go mii analyseret korpusteavstta lingvisttalaččat, de dárbbahit diehtit gokko cealkka álgá ja gokko dat loahppá. Árbevirolaš korpusanalysas láve cealkka mearriduvvot ovdal lingvisttalaš vuođđoanalysa. Dán barggus mii árvalit lahkonanvuogi mas mii dahkat lingvisttalaš analysa ovdal go mearridit mii cealkka lea. Danin go čuoggá manná dulkot guovtti ládje oanádusa ja lohkosátnedajaldaga maŋis, de mii diktit ambiguitehta orrut álgotokeniseremis. Nuppi lávkkis mii fas dulkot daid nu ahte ambiguitehta jávká - čuoggá lingvisttalaš birrasa vuođul. Mii buohtastahttit ovdalaš lahkonanvuogi dainna ođđa vugiin ja čájehit got ođđa guovttelávkkat vuohki tokeniseret dahká álkibun mearridit cealkkarájiid.

1 Introduction

North Sámi is a Uralic language with a complex morphological structure spoken in Norway, Sweden and Finland by approximately 25 700 speakers (Simons and Fennig, 2018). In corpus analysis the first challenge is the correct identification of the basic syntactic frame, the sentence. While a combination of period and whitespace is typically seen as a reliable indicator for the sentence, there are many contexts in which this is not the case. In this paper we challenge this rather rigid and local analysis of potential sentence boundaries and suggest a flexible approach initially assuming ambiguity and later disambiguating this ambiguity by means of morpho-syntactic context conditions.

In the method we present we are using a morphological analyser as a center piece of tokenization of free text using *hfst-pmatch* and *hfst-tokenise*, and we specifically look at sentence boundary detection and disambiguation, using the morphological analysis of various abbreviated expressions to help identify sentence boundaries. Combining a transducer with a tokenization tool lets us delay the resolution of ambiguous tokenization, including sentence boundaries. We then use a Constraint Grammar (CG - see below) module that looks at the context of the ambiguous sentence boundaries to disambiguate and decide whether a period is also an end of sentence mark, or just part of an abbreviated expression or numerical expressions like dates and ordinals.

Due to the typology of North Sámi combined with the scarcity of corpus resources, using a lexicon-based finite state transducer (Beesley and Karttunen, 2003) for morphological analysis it is the only viable option. Both the typology and lack of corpus material are shared with most other Uralic languages, so what is done for North Sámi should be quite relevant for the other languages in the family, as well as for other languages with similar typology. For the same reason we do not consider statistical approaches fruitful, there just is not enough material for most of these languages. A comparison with deep machine learning methods would be an interesting task for a future project.

2 Background

In this section we will present the general framework and infrastructure for our approach and the motivation for replacing our previous approach to tokenization and sentence identification with a newer one.

2.1 Framework

The central tools used in corpus analysis in the *Giella*-framework are *finite state transducers* (FST's) and *Constraint Grammars* (CG). CG is a rule-based formalism for writing disambiguation and syntactic annotation grammars (Karlsson, 1990; Karlsson et al., 1995). The *vislcg3* implementation[1] we use also allows for dependency annotation. CG relies on a bottom-up analysis of running text. Possible but unlikely analyses are discarded step by step with the help of morpho-syntactic context.

Preprocess was built in the early days of the *Giella* infrastructure. It encodes some amounts of linguist knowledge, but isolated from the rest of the linguistic facts of each language. The fact that it is written in Perl makes it hard to include in shipping products such as grammar checkers. The major issue, though, is the fact that it makes linguistic decisions before there has been any linguistic analysis at all, which this article is all about.

All components are compiled and built using the *Giella* infrastructure (Moshagen et al., 2013). This infrastructure is useful when coordinating resource development using common tools and a common architecture. It also ensures a consistent build process across languages, and makes it possible to propagate new tools and technologies to all languages within the infrastructure. That is, the progress described in this paper is immediately available for all languages in the *Giella* infrastructure, barring the necessary linguistic work.

The North Sámi Constraint Grammar analysers take morphological ambiguous input, which is the output from analysers compiled as FST's. The source of these analysers is written in the Xerox *twolc* and *lexc* (Beesley and Karttunen, 2003) formalisms, compiled and run with the free and open source package HFST (Lindén et al., 2011).

We also rely on a recent addition to HFST, *hfst-pmatch* (inspired by Xerox pmatch (Karttunen, 2011)) with the runtime tool *hfst-tokenise* (Hardwick et al., 2015). Below we describe how this lets us analyse and tokenise in one step, using FST's to identify regular words as well as multiword expressions with full morphology.

The choice of purely rule-based technologies is not accidental. The complexity of the languages we work with, and the general sparsity of data, makes purely data-driven methods inadequate. Additionally, rule-based work leads to linguistic insights that feed back into our general understanding of the grammar of the languages.

2.2 Motivation

In the OLD (but still used) corpus analysis pipeline in the *Giella*-framework, sentence boundary identification is performed by the perl script *preprocess* before any other linguistic analysis of the sentence is made, cf. Figure 1.

In sentence boundary identification, the following expressions that are typically followed by a period need to be disambiguated: **abbreviations**, and **numerical expressions** (for example ordinals and dates). The full stop can either be part of the expression itself, with no sentence boundary implied, or it can also entail the end of the preceding sentence.

Preprocess is based on the following assumptions: It distinguishes between 'transitive' and 'intransitive' abbreviations. 'Transitive' and 'intransitive' are used in the following way in this context: Transitive abbreviations are for example *ee.= earret eará* 'amongst others', *vrd.=veardit* 'compare'. Intransitive abbreviations are for example *bearj.= bearjadat* 'Friday', *eaŋg.= eŋgelasgiella* 'English', *jna.= ja nu ain* 'and so on', *milj.= miljovdna* 'million', *ru.= ruvdnu* 'crowns'. In addition, abbreviations typically followed by numbers are listed separately, fore example, *nr.= nummar* 'number', *kap.= kapihttal* 'chapter', *tlf.= telefovdna* 'telephone'.

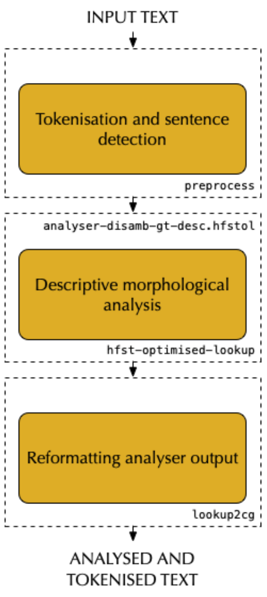
[1]`http://visl.sdu.dk/constraint_grammar.html` (accessed 2018-10-08), also Bick and Didriksen (2015)

While the period after transitive abbreviations like *vrd.=veardit* 'compare' in ex. (1) is never interpreted as a sentence boundary if followed by a word (capitalized or not) or a numerical digit, it is always interpreted as a sentence boundary after intransitive abbreviations like *jna.= ja nu ain* 'and so on' in ex. (2). In the case of abbreviations typically followed by numbers, the period is interpreted as a sentence boundary if followed by a capitalised word, but not if followed by a word with lower case letters or numerical digits like *13* after *kap.* 'chapter' in ex. (3).

(1) **vrd.** máinnašumiin kapihttalis 14
cmp. mention.COM chapter.LOC 14
'compare with the mentioning in chapter 14'

(2) ...mas lea diehusge
...where is of.course
smávva oasit nugo rap, rocka **jna.**
small parts like rap, rock and.so.on.
Jietnateknihkkáriid mii maid šaddat
Sound.technician.ACC.PL we also become
hui dávjá bivdit boahtit lulde.
very often ask come from.south
'...where are of course small parts like rap, rock and so on. We also often have to ask sound technicians to come from the south'

(3) gč. **kap.** 13.
see chapter 13
'See chapter 13'



INPUT TEXT
Tokenisation and sentence detection
preprocess
analyser-disamb-gt-desc.hfstol
Descriptive morphological analysis
hfst-optimised-lookup
Reformatting analyser output
lookup2cg
ANALYSED AND TOKENISED TEXT

Figure 1: OLD tokenization architecture

While these are reasonable criteria to distinguish between sentence boundaries and periods that are parts of the actual expression, there are a number of cases that cannot be resolved. *Preprocess* has absolutely no access to any linguistic criteria. That means that it cannot distinguish between nouns and proper nouns. If the capitalized word after a period is a proper noun, which is captialized also in the middle of a sentence, *preprocess* cannot take that into consideration. This is the case in the example below, where the intransitive currency abbreviation *ru.* 'crown(s)' is followed by a proper noun. In ex. (4), the period after the intransitive abbreviation *ru.* 'crown(s)' followed by the proper noun *Sámeálbmotfondii* 'Sámi people's fond (Ill.)' is interpreted as a sentence boundary by *preprocess*, cf. Figure 2, while it is interpreted as part of the expression with *ru.* before *fondii* 'fund (Ill.)'.

(4) ...lea várrejuvvon 16 000 000 **ru.** Sámeálbmotfondii 2009:s.
...is reserved 16 000 000 crowns Sámeálbmot.fond.ILL 2009.LOC
'...it is reserved 16 000 000 crowns to the Sámeálbmot-fond 2009.'

(5) ...lea várrejuvvon 16 000 000 **ru.** fondii 2009:s.
...is reserved 16 000 000 crowns fond.ILL 2009.LOC
'...it is reserved 16 000 000 crowns to the fond 2009.'

Also ordinals and date expressions like *02.03* in ex. (6) at the end of sentences can

```
Áššis
25/08
lea
várrejuvvonr
16 000 000
ru.
.
Sámeálbmotfondii
2009:s
.
```

Figure 2: Preprocess analysis of ex. (5)

```
Sámi parlamentáralaš ráđđi
čoahkkanii
Kárášjogas
02.03
.
```

Figure 3: Preprocess analysis of ex. (6)

cause problems when analyzed by *preprocess*.

(6) Sámi parlamentáralaš ráđđi čoahkkanii Kárášjogas **02.03**.
Sámi parliament council met Kárášjohka.LOC 02.03.
'The Sámi parliament council met in Kárášjohka on the 02.03.'

In the analysis of *preprocess* in Figure 3, the date expression is not recognized as such as the period is categorically removed from after **02.03** although it is part of the expression. The same is the case if the last expression in the sentence is an ordinal.

3 Our two-step approach

Below we describe our new approach to tokenization, which includes a two-step identification of sentence boundaries. We will then evaluate our new method and compare its results to the results of our previous approach. The new approach consists of two steps:

1. ambiguous tokenization with *hfst-tokenise* and
2. disambiguation of ambiguous tokenization with *mwe-dis.cg3*.

It has originally been introduced as a part of the North Sámi grammar checker to resolve compound error detection, cf. (Wiechetek, 2012, 2017).

The North Sámi corpus analysis consists of different modules that can be used separately or in combination, cf. Figure 4. The text is initially tokenised and morphologically analysed by the descriptive morphological analyser and tokeniser *tokeniser-gramcheck-gt-desc.pmhfst*. Any ambiguous tokenization is left as is, to be resolved later on. The following module, *analyser-gt-whitespace.hfst*, detects and tags certain whitespace delimiters, so that it can tag, for example the first word of paragraphs and other whitespace delimited boundaries. This can then be used by the sentence boundary detection rules later on, which enables detecting, for example, headers based on their surrounding whitespace. The valency annotation grammar *valency.cg3* adds valency tags to potential governors, i.e. (predominantly) verbs, nouns, adverbs and adjectives.

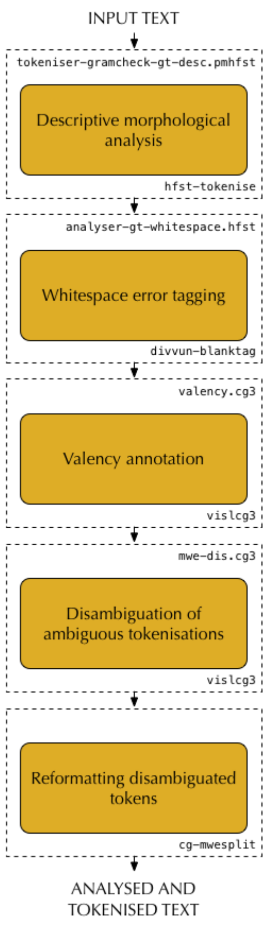
The subsequent module is the heart of the disambiguation of ambiguous tokenization. The Constraint Grammar file *mwe-dis.cg3* decides, among other things, whether a period is a sentence boundary or not, based on the morphological and other linguistic analysis of the surrounding tokens. Finally, the command line tool *cg-mwesplit* (part of the *vislcg3* package) reformats the disambiguated cohorts into their final form for further consumption by other *vislcg3* grammars.

3.1 Hfst-tokenize

A novel feature of our approach is the support for different kinds of ambiguous tokenizations in the analyser, and how we disambiguate ambiguous tokens using Constraint Grammar rules.

We do tokenization as part of the morphological analysis using the *hfst-tokenise* tool, which does a left-to-right longest match analysis of the input, where matches are those given by a *pmatch* analyser.

```
Define morphology @bin"analyser.hfst" ;
Define punctword  morphology &
                  [ Punct:[?*] ] ;
Define blank      Whitespace |
                  Punct ;
Define morphoword morphology
                  LC([blank | #])
                  RC([blank | #]) ;
regex [ morphoword | punctword ];
```


INPUT TEXT
tokeniser-gramcheck-gt-desc.pmhfst
Descriptive morphological analysis
hfst-tokenise
analyser-gt-whitespace.hfst
Whitespace error tagging
divvun-blanktag
valency.cg3
Valency annotation
vislcg3
mwe-dis.cg3
Disambiguation of ambiguous tokenisations
vislcg3
Reformatting disambiguated tokens
cg-mwesplit
ANALYSED AND TOKENISED TEXT
Figure 4: NEW tokenization architecture

The above *pmatch* definitions say that a word from the lexicon (*analyser.hfst*) has to be surrounded by a ”blank”, where a blank is either whitespace or punctuation. The LC/RC are the left and right context conditions. We also extract (intersect) the subset of the lexicon where the form is punctuation, and allow that to appear without any context conditions.

We insert *re-tokenization* hints in the lexicon at places were we assume there is a possible tokenization border, and our changes to *hfst-tokenise* let the analyser backtrack and look for other tokenizations of the same input string. That is, for a given longest match tokenization, we can force it to *redo* the tokenization so we get other multi-token readings with shorter segments alongside the longest match. This solves the issue of combinatorial explosion.

As a simple example, the ordinal analysis of *17.* has a backtracking mark between the number and the period. If the lexicon contains the symbol-pairs/arcs

```
1:1 7:7 ϵ:@PMATCH_BACKTRACK@
ϵ:@PMATCH_INPUT_MARK@ .:A ϵ:Ord
```

then, since the form-side of this analysis is 17., the input *17.* will match, but since there was a backtrack-symbol, we trigger a retokenization. The input-mark symbol says where the form should be split.[2] Thus we also get analyses of *17* and . as two separate tokens.

```
"<17.>"
        "17" A Ord Attr
        "." CLB "<.>"
                "17" Num "<17>"
```

To represent tokenization ambiguity in the Constraint Grammar format, we use *vislcg3 subreadings*,[3] where deeper (more indented) readings are those that appeared first in the stream, and any reading with a word-form-tag (`"<.>"` above) should (if chosen by disambiguation) be turned into a cohort of its own. Now we may run a regular Constraint Grammar rule to pick the correct reading based on context, for example `SELECT (".") IF (1 some-context-condition) …;` which would give us

```
"<17.>"
        "." CLB "<.>"
                "17" Num "<17>"
```

Then a purely mechanical reformatter named *cg-mwesplit* turns this into separate tokens, keeping the matching parts together:

```
"<17>"
        "17" Num
"<.>"
        "." CLB
```

3.2 Tokenization disambiguation

As mentioned above, disambiguation of ambiguous tokenization is done after the morphological analysis. Consequently, this step has access to undisambiguated morphological (but not full syntactical) information. In addition, lexical semantic tags and valency tags are provided. The rules that resolve sentence boundary ambiguity are based on transitivity tags of abbreviations, lexical semantic tags, and morphological tags. Some of them are specific to one particular abbreviation.

Version r173258 of the tokenization disambiguation grammar *mwe-dis.cg3* has 22 rules that handle sentence boundary disambiguation. The rule below removes a sentence boundary reading (”.” CLB) if it is part of a numerical expression and there is noun of the lexical semantic category *currency* (Sem/Curr) to the right of it.

```
REMOVE:num-before-curr ("." CLB) IF (0 Num)(1*> (>>>)
BARRIER (>>>) LINK 1 Sem/Curr OR ("kr"));
```

In the case of *mill.* ‘million’, the period can be both the end of the sentence (cf. ex. (7)) or not (cf. ex. (8)), depending on the semantic tag of the following token. If a noun of the type currency follows (for example *ruvnno* ‘crown (Gen.)’ in ex. (8)) the period should be part of the abbreviation expression.

(7)

Eará	buvttaduvvon	dietnasat:	4,6 **mill**.
other	manufactured	profits:	4.6 millions

‘Other manufactured profits: 4.6 millions’

[2]This also means we cannot reshuffle the input/output side of the FST. In practice, we use a flag diacritic in the lexicon, which will keep its place during minimisation, and after the regular lexicon is compiled, we turn the flag into the ϵ:`@PMATCH_INPUT_MARK@` symbol-pair.

[3]`https://visl.sdu.dk/cg3/chunked/subreadings.html` (accessed 2018-10-10)

(8) 1,0 **mill.** ruvnno nissondoaimmaide
1.0 mill. crowns women.activity.ILL.PL
'1.0 mill. crowns to women's activities'

In ex. (9), the affiliation **Bb.=Bargiidbellodat** 'labor party' is abbreviated and followed by a capitalized proper noun. However it is not a sentence boundary as it is followed by a female proper noun.

(9) **Bb.** Vibeke Larsen ii jurddaš nie.
Labor.party Vibeke Larsen not think so
'Vibeke Larsen from the labor party doesn't think that way'

The first rule below removes a sentence boundary reading ("." CLB) if it is part of an abbreviation expression and there is a noun of the lexical semantic category *currency* (Sem/Curr) to the right of it.

The second rule removes a sentence boundary reading ("." CLB) if it is part of an abbreviation expression of the lexical semantic category organization (*Sem/Org*) and it is followed by a human name (*Sem/Sur OR Sem/Fem OR Sem/Mal*).

```
REMOVE ("." CLB) IF (0 Num)
(1*> (>>>) BARRIER (>>>) LINK 1 Sem/Curr OR ("kr"));

REMOVE ("." CLB) IF (0 Sem/Org + ABBR)
(1*> Sem/Sur OR Sem/Fem OR Sem/Mal);
```

In ex. (10), the date is removed because it is preceded by an item of the class category (*kap=kapihttal* 'chapter').

(10) Dás čujuhuvvo Sámedikki jahkedieđáhussii kap **2.11.**
here referred Sámi.parliament report.ILL chapter 2.11.
'Here, they refer to The Sámi parliament's report chapter 2.11.'

```
REMOVE (Sem/Date) IF (-1 KLASS)(0 CLB LINK 0/1 (Num Arab));

LIST KLASS = "art" "ášši" "bálkáceahkki" č"uokkis"
"đdie.nr" "nr" "s" "siidu" "§" "§§" "paragráfa" "S.nr"
"st.đdie. nr" "od.prp.nr" "Ot.prp. nr" "oassi" "kap"
"kapihttal" "kapihtal";
```

4 Evaluation

In this section we are going to evaluate our new approach to tokenization. A common method for splitting sentences in a complete pipeline (used for example by *LanguageTool*) is to tokenise first, then do sentence splitting, followed by other stages of linguistic analysis.

The quantitative evaluation is split in two: the first part only looks at expressions ending in a full stop that are truly ambiguous with respect to sentence boundaries — the full stop can both be and not be a sentence boundary depending on the context. The evaluation looks at the performance of the pipeline for this specific subset of the corpus. The second evaluation looks at all instances of expressions ending in full stops, both the unambiguous and the ambiguous ones, and compare the performance of two different approaches as measured to a gold standard version of the tokenized text. Again, we only look at sentence boundaries, but in this evaluation we look at the overall performance. The two pipelines evaluated are the old pipeline and the new pipeline described elsewhere in this article.

Within the qualitative evaluation where we analyze the cases of unsuccessful sentence boundary identification and analyze the reasons for the shortcomings.

4.1 Quantitative evaluation of the NEW approach

In this part of the evaluation, we particularly focus on the performance of the tokenizer in contexts that are ambiguous, and we evaluate the identification of sentence boundaries rather than successful tokenization in general. In the quantitative evaluation of ambiguous sentence boundary (SB) detection we calculated both precision (correct fraction of all decisions), recall (correct fraction of all targeted constructions) and accuracy (all correct decisions as fraction of all decisions). As an evaluation corpus we chose a newspaper corpus containing the 2014 texts of the daily North Sámi newspaper *Ávvir*[4].The corpus used contains 556 644 space separated strings, as reported by the Unix tool `wc`. The exact number of tokens will vary depending on tokenization methods, as described below.

In Table 1, we evaluate the identification of sentence boundaries in ambiguous expressions, i.e. NOT all sentence boundaries. As true positives we count ambiguous sentence boundaries that have been correctly identified. As false positives we count ambiguous expressions that do not involve a sentence boundary that have falsely been identified as sentence boundaries. As true negatives we count the cases of ambiguous expressions that are not sentence boundaries and have not been identified as such. As false negatives count sentence boundaries that have not been identified as such. Precision is 48% and recall is 62% and can still be improved. However, due to the fact that abbreviations and time expressions hardly ever occur at the end of a sentence, the number of true positives is naturally low. The accuracy of the NEW pipeline is 97%, i.e. 97% of all ambiguous periods are correctly analyzed as either parts of the abbreviation or as a sentence boundary.

Measures	
True positives	64
False positives	69
True negatives	3 425
False negatives	39
Precision	48.1%
Recall	62.1%
Accuracy	97.0%

Table 1: Quantitative evaluation of ambiguous sentence boundary tokenization after morphological analysis

4.2 Comparison with the old pipeline

We also evaluated the NEW pipeline against the OLD one with respect to the identification of all sentence boundaries (not only the morphologically ambiguous ones). Using the manually annotated section of the corpus as the base, we constructed a smaller, gold standard (GS) corpus to compare against. The GS corpus contains 221 620 tokens as measured by the Unix tool *wc*, and 14 811 sentence delimiting full stops out of a total of 16 057 tokens ending in or consisting of full stops. This also implies that 1 246 tokens ending in full stops do *not* constitute a sentence boundary.

[4]`https://avvir.no/` (accessed 2018-10-08)

The result of comparing both the OLD and the NEW pipelines against the gold standard is summarised in Table 2. These numbers have been checked by manual search for problematic pairs, and by diffing the OLD and NEW directly against each other, to see whether we get similar results for true positives.

Measures	**OLD**	**NEW**
True positives	14 727	14 798
False positives	592	17
True negatives	669	1 228
False negatives	69	4
Precision	96.14%	99.89%
Recall	99.53%	99.97%
Accuracy	95.88%	99.81%

Table 2: Quantitative evaluation and comparison of the OLD and the NEW sentence boundary detection pipelines.

As documented by the recall results, both tokenization approaches are almost equally successful regarding the true positives. But when looking at the other numbers, the difference is quite striking. While precision and accuracy hover around 96% for the OLD pipeline, all measures for the NEW pipeline are around 99.9%. Most of the remaining errors can be removed by lexicon fixes and improved Constraint Grammar rules. Some of the problematic issues are discussed below.

4.3 Qualitative evaluation

While some errors in the sentence boundary identification are due to problems in the lexicon, others require more precise rules in the disambiguation grammar *mwe-dis.cg3*.

(11)

Lean	veallán	dás	ja	geahččan	**su**.
have.PRS.1SG	lie.PRFPRC	here	and	watch.PRFPRC	her;him

‘I have lain here and watched her;him.’

In ex. (11), the period after the pronoun *su* ‘her, his’ is not correctly identified as a sentence boundary, it is a false positive. Instead *su=sulli* ‘approximately’ is identified as an abbreviation with the period being part of the expression. The pronoun reading does not even appear in the analysis.

```
"<su.>"
"su" Adv ABBR Gram/NumNoAbbr <W:0.0000000000>
; "." CLB <W:0.0000000000> "<.>"
; "su" Adv ABBR Gram/NumNoAbbr <W:0.0000000000> "<su>"
```

This is due to an issue in the lexicon that is easily solvable, but had not been detected at the time of doing the analysis used as basis for the evaluation. The core of the issue is that some pronouns, such as *su* above, can also be abbreviated adverbs with obligatory full stops. What was missing in the lexicon is a signal for such abbreviated adverbs that will trigger retokenization. Such triggers are easy to add, and needs to be added only once for the whole category.

In ex. (12), the time expression *20.00* (Sem/Time-clock) is ambiguous with a date expression (Sem/Date). This needs to be corrected in the morphological analyzer as *20.00* is not a valid date. The time expression is erroneously removed and the sentence boundary is not identified.

(12)

gaskal	13.00	ja	**20.00**.
between	13.00	and	20.00

'between 1 and 8 pm.'

```
"<13.00>"
        "13.00" Num Arab Sg Gen <W:0.0000000000>
        "13.00" Num Sem/Time-clock Sg Gen <W:0.0000000000>
:
"<ja>"
        "ja" CC <W:0.0000000000>
:
"<20.00.>"
        "20.00" Num Sem/Date Sg Gen <W:0.0000000000>
;       "." CLB <W:0> "<.>"
;               "20.00" Num Sem/Time-clock Sg Nom <W:0> "<20.00>"
    REMOVE:2426:longest-match
;       "." CLB <W:0> "<.>"
```

In other cases, the error in the analysis is due to shortcomings in the disambiguation file *mwe-dis.cg3*. In ex. (13), the name is is followed by two initials before the surname. The period after the first initial is erroneously interpreted as a sentence boundary. There is a rule that removes the sentence boundary reading if there is one initial after a first and before the surname.

(13) Hilde **C.J.** Mietinen álgá maŋŋel geasi
Hilde C.J. Mietinen begins after summer.GEN
'Hilde C.J. Mietinen begins after the summer'

5 Conclusion

We have questioned the traditional approach to identifying sentence boundaries as the first step of text processing, before any linguistic analysis is done, and usually completely independent of any linguistic parsing at all. This introduces errors that are almost impossible to recover from in the remaining analysis steps.

We have demonstrated that by means of a basic linguistic analysis prior to tokenization sentence boundary detection can be substantially improved. We proposed a two-step tokenization that leaves initial ambiguity in sentence boundary detection until it can be disambiguated by means of linguistic context.

Our experiment showed that our system outperforms a state-of-the-art traditional sentence tokenizer. 97% of all ambiguous periods are correctly analyzed in the new tokenization approach. Disambiguation of all sentence boundaries give good results both in terms of precision, recall and accuracy, i.e. all are above 99.8%, and recall approaching 99.99%. Our method of ambiguous tokenization and ambiguity resolution by means of grammatical context allows us to improve tokenization significantly compared to the previous one-step-approach.

It would be an interesting topic for future work to compare our results with deep machine learning approaches, and whether deep learning can approach our results given the sparsity of data for the languages we work on.

The main new insight gained is that linguistic context is relevant and necessary when identifying sentence boundaries, and ambiguous tokenization should not be handled solely by a tokenizer without linguistic knowledge. Tokenization is not only important in corpus analysis but also in other tasks like grammar checking, machine translation and all other text processing of free text.

Acknowledgments

We especially would like to thank Kevin Unhammer for his work on the gramar checker framework that is the foundation of our work discussed above, and our colleagues in *Divvun* and *Giellatekno* for their daily contributions to our language tools and the infrastructure.

References

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Studies in Computational Linguistics. CSLI Publications, Stanford.

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

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. http://aclweb.org/anthology/W/W15/W15-1842.pdf.

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

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

Lauri Karttunen. 2011. Beyond morphology: Pattern matching with FST. In *SFCM*. Springer, volume 100 of *Communications in Computer and Information Science*, pages 1–13.

Krister Lindén, Miikka Silfverberg, Erik Axelson, Sam Hardwick, and Tommi Pirinen. 2011. Hfst—framework for compiling and applying morphologies. In Cerstin Mahlow and Michael Pietrowski, editors, *Systems and Frameworks for Computational Morphology*, Springer-Verlag, Berlin, Heidelberg, volume Vol. 100 of *Communications in Computer and Information Science*, pages 67–85.

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

Gary F. Simons and Charles D. Fennig, editors. 2018. *Ethnologue: Languages of the World*. SIL International, Dallas, Texas, twenty-first edition. http://www.ethnologue.com (Accessed 2018-10-09).

Linda Wiechetek. 2012. Constraint Grammar based correction of grammatical errors for North Sámi. In G. De Pauw, G-M de Schryver, M.L. Forcada, K. Sarasola, F.M. Tyers, and P.W. Wagacha, editors, *Proceedings of the Workshop on Language Technology* *for Normalisation of Less-Resourced Languages (SALTMIL 8/AFLAT 2012)*. European Language Resources Association (ELRA), Istanbul, Turkey, pages 35–40.

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.